Sapphire Frontend

A Web-Based Course Grading Management System

Matthias Link, BSc

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's Degree Programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Ao.Univ.-Prof. Dr. Keith Andrews
Institute of Interactive Systems and Data Science (ISDS)

Graz, 27 Oct 2021

Sapphire Frontend

Ein webbasiertes Verwaltungs- und Bewertungssystem für Lehrveranstaltungen

Matthias Link, BSc

Masterarbeit

für den akademischen Grad

Diplom-Ingenieur

Masterstudium: Informatik

an der

Technischen Universität Graz

Begutachter

Ao.Univ.-Prof. Dr. Keith Andrews
Institute of Interactive Systems and Data Science (ISDS)

Graz, 27 Oct 2021

Diese Arbeit ist in englischer Sprache verfasst.

Abstract

Grading university courses with hundreds of students can be a difficult and time-consuming task. An important aspect of grading a mass course is the need to distribute the grading process among several staff members. Splitting the workload into smaller chunks allows lecturers to delegate parts of the grading process to tutors and ensures timely feedback for students. It is important to provide a fair and comprehensible grading process for everyone involved. Handling online submissions also needs to be considered. It is necessary to reliably determine which student or student group submitted which files at which time.

Current solutions range from simple spreadsheet-based solutions to complete course management systems. Even though there are many options available, there is a lack of an integrated yet streamlined grading system for university courses.

This Master's thesis describes the frontend of the Sapphire project, an integrated web-based grading management system. Students are provided with a powerful file management system on a per-exercise basis. A list of ratings is used as the basis for the grading process, allowing lecturers to easily manage the important grading criteria for each exercise and enabling tutors to indicate failed criteria at the press of a mouse button. Sapphire supports lecturers and tutors to effectively manage the grading of large courses and provides students with online submission of exercises and detailed and timely feedback of grading.

Kurzfassung

Die Bewertung von Studierendenleistungen im Rahmen von Massenkursen an Universitäten erweist sich als schwierige und zeitaufwändige Aufgabe. Ein wichtiger Punkt dabei ist die Verteilung des Benotungsprozesses auf mehrere Kursmitarbeiter. Die Verteilung der erforderlichen Arbeitsleistung, ermöglicht es dem Kursleiter Teilaufgaben an Studienassistenten weiterzugeben und zeitnahe Rückmeldung an Studenten zu gewährleisten. Fairness und Verständlichkeit sind für alle Beteiligten wichtige Aspekte des Bewertungsprozesses. Darüber hinaus ist ein standardisierter und benutzerfreundlicher Abgabeprozess für die Studierenden ein wesentliches Kriterium für eine professionelle Abwicklung. Dazu gehört auch feststellen zu können, welche Studentengruppe oder Student eine Abgabe getätigt hat.

Bestehende Lösungen für diese Problematik reichen von einfachen tabellenkalkulationsbasierten Lösungen bis hin zu kompletten Kursverwaltungssystemen. Der Markt bietet viele verschiedene Angebote. Bestehende, vollständig integrierte und optimierte Benotungssysteme für Universitäten bieten jedoch keine zufriedenstellenden Lösungen.

Diese Masterarbeit beschreibt die Kernaspekte der Benutzeroberfläche des Sapphire Projekts, ein integriertes Studentenverwaltungs- und Kursbewertungssystem. Das Hauptaugenmerk bei der Implementierung liegt dabei auf der Benutzerfreundlichkeit. Studenten wird es pro Aufgabe ermöglicht, Dateien und Ordner über einen leistungsstarken Filebrowser zu verwalten. Eine Liste von Bewertungen bildet die Basis für den Benotungsprozess. Diese ermöglicht es Kursleitern die Lernziele einer Aufgabe auf einfache Art und Weise zu verwalten und erlaubt es Tutoren, nicht erfüllte Kriterien per Mouse-Click zu markieren. Die benutzerfreundliche Implementierung von IT-gestützten Abgabe- und Bewertungsprozessen unterstützt sowohl Lehrende als auch Studierende bei der effektiven Leistungsermittlung im Rahmen von Massenkursen an Universitäten.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The document uploaded to TUGRAZonline is identical to the present thesis.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Dokument ist mit der vorliegenden Arbeit identisch.

Date/Datum

Signature/Unterschrift

Contents

List of Figures

List of Tables

List of Listings

Acronyms

AJAX Asynchronous JavaScript And XML. 38, 46, 49, 64

API Application Programming Interface. 64

CD Compact Disc. 21, 60

CSS Cascading Style Sheet. 1, 11–14, 18, 20, 24–26, 50, 51, 69, 105, 106, 131, 143, 144

CSV Comma Separated Value. 22, 34, 35, 54, 125–128

DB Database. 63, 64

DNS Domain Name Service. 11

DOM Document Object Model. 68, 69

DSL Domain Specific Language. 8, 9

DVD Digital Versatile Disc. 21, 60

ERB Embedded Ruby. 7, 9

FS File System. 24, 44, 50, 65

HCI Human-Computer Interaction. 19, 20, 22, 25, 31, 51, 54, 56, 74

HTML Hyper Text Markup Language. 1, 7, 9, 11–13, 18, 20, 25, 26, 50, 51, 64, 68, 74, 75, 104–107, 109, 131, 143, 144, 148, 149

HTTP Hyper Text Transfer Protocol. 7–9, 11, 45, 59, 60, 68

INM Internet and New Media. 19, 20, 25, 27, 51

JS JavaScript. 1, 12, 14–18, 24, 25, 63, 64, 68, 69, 105, 143

JSON JavaScript Object Notation. 59

KV Store key-value store. 24, 60

MC Multiple Choice. 56

MIME Multipurpose Internet Mail Extensions. 11, 12, 43, 50

MVC Model-View-Controller. 7

PDF Portable Document Format. 50

RESTful Representational State Transfer. 59

RVM Ruby Version Manager. 4, 5

SASS Syntactically Awesome CSS. xviii, 13–15, 18, *Glossary:* Syntactically Awesome CSS (SASS)

SCSS Sassy CSS; Syntax extension to the SASS language. 13, 14, 18, *Glossary:* Syntactically Awesome CSS (SASS)

SQL Structured Query Language. 8, 9, 68

STI Single Table Inheritance. 64, *Glossary:* Single Table Inheritance (STI)

SVG Scalable Vector Graphics. 55

TCP/IP Transmission Control Protocol/Internet Protocol. 11, *Glossary:* Transmission Control Protocol/Internet Protocol (TCP/IP)

TS TypeScript. 16, 17

UI User Interface. 18, 20, 22, 24, 30, 49, 54, 71–75, 87, 101, 130, 139

URL Uniform Resource Locator. 8, 11, 30, 46, 68, 69, 83, 97, 105, 129, 130, 143, 152

UTF-8 8-Bit UCS Transformation Format. 43

XML Extensible Markup Language. 11

XSS Cross-Site Scripting. 105, 143

Glossary

Mixin Snippet of code which bundles common CSS constructions in one method call. xix

Nesting Defining parts of code inside another piece of code, in order to enhance the parent's functionality. xix, 13, 14

Partial Rails' term for a small snippet of code, written in any markup language, which can be placed anywhere in an application's view layer. 60, 64

Polyfill Provide new functionality to older browsers which do not support the latest standard, without impairing performance on modern ones. 12

Serialisation The process of converting a complex data structure into a string, in order to allow this data structure to be stored in a database or be transmitted over the network. 59, 64

simple_form A form generator used in Sapphire. It extends Rails' default form generator by providing a simpler, less verbose interface. 60

Single Table Inheritance (STI) Storing different classes of objects, with similar attributes, in a single database table. 64

Superset A set of functions or specifications which enhance a base set of functions or specifications. 13

Syntactically Awesome CSS (SASS) CSS preprocessor which enhances CSS with variables, nesting, imports, loops, mixins, and operators [Catlin et al. 2015]. 13

Transmission Control Protocol/Internet Protocol (TCP/IP) Underlying web technology used for communication between a client and a server [ISI 1981a][ISI 1981b]. 11

Acknowledgements

I would like to express my appreciation for my colleagues at the university for providing my with a constant stream of ideas. Their advice helped me with key decisions during the development process. Special thanks goes out to my advisor Keith Andrews for providing me with essential hints during the years of developing and refining Sapphire. Further I would like to thank him for the sheer endless amount of time spent on proofreading this thesis.

Furthermore, I would like to thank Thomas Kriechbaumer for his constant work on the Sapphire project and the many hours spent including building the backend of Sapphire, fixing bugs, writing unit tests, reviewing code, keeping the servers running, and always lending a friendly ear whenever problems arose.

Helmut Leitner, for maintaining the server infrastructure for the Sapphire project. Stefan Pranger, who currently maintains the production version of Sapphire, for his contributions to the Sapphire project. I would like to acknowledge the contributions of the tutors and students of HCI and INM, who provided me with constant feedback on new features and helped me improve the frontend of Sapphire.

Last but not least I want to thank my friends, family, and all the other people who paid close attention to this project and who provided me with advice whenever I needed it.

Matthias Link
Graz, Austria, 27 Oct 2021

Credits

In particular, I would like to thank the following persons for their contributions:

- Chapters 2, 3, and 5 were written jointly by the author and Thomas Kriechbaumer.
- Keith Andrews provided a simple LaTeXthesis skeleton [Andrews 2019e], from which I derived my own LaTeXtemplate.

Chapter 1

Introduction

This thesis describes the implementation of the frontend of Sapphire, a web-based course management and grading system. Chapter 2 introduces the basic concepts of the Ruby programming language. The web framework Ruby on Rails, in short Rails, is described in Chapter 3. The frontend of Sapphire is based on the modern Hyper Text Markup Language (HTML), Cascading Style Sheet (CSS), and JavaScript (JS) technologies, described in Chapter 4.

The basic concepts of the online grading system Sapphire are introduced in Chapter 5. The following chapters describe technical aspects in more detail. Chapter 6 presents the family of interfaces used for managing courses, terms, students, and staff members. The submission-related interfaces provided to students are described in Chapter 7. Chapter 8 describes the process of implementing a fast and reliable grading interface for tutors. The interfaces for configuring the asynchronous export features of Sapphire are described in Chapter 9.

Selected details of unusual aspects of Sapphire are presented in Chapter 10. Finally, an outlook of improvements to existing features as well as future features is given in Chapter 11. This thesis concludes with remarks on the implementation process in Chapter 12. Appendices A, B, and C provide user guides for students, tutors, and lecturers respectively.

Chapters 2, 3, and 5 were written jointly with Thomas Kriechbaumer [Kriechbaumer 2014], who was responsible for implementing the Sapphire backend. The remainder of this thesis is solely the work of the author.

Chapter 2

The Ruby Language

During the planning phase of every software project, one of the first questions is the determination of the programming language to be used for each system under consideration. In the early days of software applications, a common choice was a compiled language like C, C++, or Java. During the last decade, scripting-based languages gained more and more momentum within the community. In the case of web development, scripting languages, like Perl and PHP have been used since the beginning.

Due to recent enhancements and runtime speed improvements, high-level scripting languages like Python and Ruby are now used in web development for server-side application logic. This chapter describes the scripting language Ruby, its reusable software libraries, and a typical Ruby development environment with the most commonly used tools and helper applications.

2.1 Language Design

Ruby is a strictly object-oriented programming language with strong dynamic, reflective, and almost no functional paradigms. The language and the standard library are not restricted to simple scripting tasks. Instead deep integration into the operating system is possible, allowing the developer to use Ruby in a general purpose fashion. A distinction must be made between the software programming language described here [Thomas et al. 2013], and the hardware design and notion specification developed at the Oxford University also called Ruby [Jones 2014].

Yukihiro "Matz" Matsumoto created the initial Ruby language in the early 1990's in Japan. Due to the easy-to-understand syntax, garbage collection, and its strong dynamic approach with duck-typing, the Ruby language gained popularity very fast around the year 2000, also opening up to the English-speaking community. The Ruby interpreter runs on every modern platform. The language core application is written in plain C, and is therefore easily portable to any architecture. Tanaka et al. [2011] used this approach to create a Ruby interpreter running on an embedded system.

The feature set of the language allows a simple approach to meta programming, with the creation of metaclasses and mixins while also honouring the well-known inheritance of classes as described in Sánchez Cuadrado and García Molina [2007].

The syntax of Ruby provides a type-less interface to all variables and objects using the duck-typing principle [Ruby Community 2019]. Using a dynamic-typing approach leads to the loss of explicit information and metadata. The need for static typing might occur and can only be addressed in a more complex and abstract combination of various techniques. Type-safe usage might for example be advantagous during interaction with a database. Therefore, type checking systems can be implemented [An et al. 2009] to ensure correct behaviour as expected by the developer.

A special feature are so-called blocks. A block describes a set of instructions and in general consists of a few lines of code. This allows the developer to easily use anonymous code execution in a different

context. Variables can be passed into the block-like arguments for a function. This feature is comparable to lambda functions [Günther and Fischer 2010], which can also be realised with a derivative of the block syntax.

2.2 Gems

Most programming languages have the capability to provide pieces of shared code to the programmer on demand. In most environments and communities this functionality is provided as packages contained, distributed, and installed with a package manager. Python has EasyInstall and pip projects to achieve the mentioned tasks. Node.js makes use of the npm utility. There are many different package managers, each well-tuned to the specific needs of a programming environment and language.

In the Ruby world, such small, reusable, independent and easily installable software code libraries are called gems. The default gem manager, as well as the corresponding website, is called RubyGems. It is pre-installed in every default Ruby environment and makes use of the flexibility and dynamic nature of the language itself. The website offers to download each gem in the latest version, while also providing a fallback solution for older versions, if still needed by a project.

The most convenient way of managing the dependencies of a given project is by making use of a so-called `Gemfile`, which holds a list of required gems, and optionally their version information. The Bundler gem can read this list and install all missing gems or update all gems automatically. This ensures a very easy workflow for Rails applications during deployment. By invoking the bundler command, all gems are installed in the correct version, and native extensions are even compiled during the installation process.

2.3 Environment

During a development workflow, it is often important to obtain the latest version of specific gems or even an upcoming or almost deprecated version of Ruby itself. To ease this switching process, the community came up with the concept of Ruby environment management tools. These little helper tools take care of downloading, installing, and switching between different versions of Ruby, changing a gemset (a collection of gems of a predefined version), or simply maintaining the current stable release of each library and the Ruby interpreter.

Most Unix-based operating systems, like Linux in all its different flavours, Mac OS X, and FreeBSD, ship with a Ruby version which is not state-of-the-art. An environment manager allows developers to obtain the latest Ruby version, as well as different implementations of the Ruby language itself such as the standard MRI Ruby, Rubinius, or JRuby. Developers can also work on different projects with different requirements in terms of Ruby version and gems. The environment manager takes care of switching the Ruby version, exchanging the installed gems, and setting up matching links and binaries for the developer to use.

2.3.1 Ruby Version Manager

The Ruby Version Manager (RVM) is designed to manage multiple installations of different Ruby interpreters and the related gemset and toolchain. RVM setup can be done on a per user basis, as well as a system-wide configuration. This helper tool allows the user to define a default Ruby version to be used every time a new shell session is launched. It is possible to switch between different gemsets.

RVM is designed as a collection of shell scripts loaded into an active shell session. The Ruby versions are installed into a predefined location which is added to the environment variable `PATH` for the shell to access the binaries. The installation can be tricky in a shared environment, due to different permissions in system-wide directories.

2.3.2 rbenv

The rbenv project provides a more application-centric abstraction layer for managing different Ruby versions. It is designed around the idea of using a specific environment for a specific application and therefore provides multiple configuration options and ways of persisting these for the user. Many developers choose rbenv over RVM because of the better integration in a collaborative environment for multiple users to share the same Ruby version and environment settings.

The key concept of rbenv are so-called shims, which are basically catch-all clauses for the PATH environment variable. A shim must be prepended to the already existing directories in the PATH. This allows the manager to take control over all Ruby related executables and insert them into the call hierarchy.

In addition to shims, the gemset workflow in rbenv makes use of the already existing Bundler application, allowing the user to specify the gems and optionally a specific version number for each gem. The need for loading multiple files into the developer's shell sessions is superfluous, because everything is already managed with the correctly placed shim.

Chapter 3

Ruby on Rails

Ruby on Rails, or in short Rails, was originally part of 37signal's basecamp application, which was developed by David Heinemeier Hanson. Its core was extracted and released as open source software in 2004 [Ruby et al. 2013]. This chapter describes the underlying concepts of Rails, mainly the Model-View-Controller (MVC) pattern, the web stack, and railties.

3.1 Model-View-Controller Pattern

The Model-View-Controller (MVC) pattern [Fowler 2002, pp. 330] is one of the core concepts of Rails, which affects the whole structure of Rails applications. There are three main parts: Models, Views, and Controllers.

- *Models*: Models contain the data of the web application, which typically are stored in a database, such as MySQL or PostgreSQL. Models themselves can be interconnected by typical relational database connectors such as 1:N mappings. A model contains business logic as well as methods needed to modify its related data.
- *Views*: Views are required to present the data stored in the models appropriately to the user. Views in Rails are written in Embedded Ruby (ERB), which basically is HTML with the addition of ERB tags, which look like this `<%...%>`. The content of each ERB tag is evaluated by the Ruby language and the result is inserted into the HTML in place of the tag.
- *Controllers*: Controllers prepare the models required for a certain action and pass the data on to the view layer, where it is rendered by the rendering logic, typically a templating engine. The controller is the first part of the pipeline and connects the model with the views.

3.2 Rails Web Stack

This section describes the web stack when using Rails. When a request is sent to a Rails application, a web server handles the request and calls the web application. The following section describes how a request is handled within the framework.

3.2.1 Rack

Rack defines a simple interface for Ruby-based web applications to communicate with the web server. It is basically an array, where the first entry is the Hyper Text Transfer Protocol (HTTP) status code, followed by a hash of HTTP headers and lastly by the HTML body. This interface is designed to be lightweight while providing an easy way to provide middleware, which will be described in the next section.

3.2.2 Middleware

A request to a web application is not processed directly by the application itself, but first passes through a stack of preprocessors, called middleware. As the name suggests, it is situated between the server and the application. Rails' middleware stack is rather large, consisting by default of at least 21 individual parts, called frames. It serves several purposes, such as logging the HTTP request, parsing the request, preparing sessions, and caching database queries.

3.2.3 Routing

After the request has been propagated through the middleware stack, it has to be serviced by the application. Since a Rails application usually consists of more than one controller, the request has to be associated with the corresponding one. For every Rails application, a special file written in plain Ruby exists, where all mappings between requests and controllers are stored. These routes also define the Uniform Resource Locator (URL) schema and paths under which specific resources and pages are accessible. The router is able to take parameters, such as HTTP headers, the requested path, and subdomains into account and instantiate and execute the appropriate controller as requested.

3.2.4 Serving The Request

After the controller has been instantiated the action determined by the request is called on the controller. This method then either constructs a response using models and views, or just redirects the user to another location (such as a login page). When the response has been created, it passes back through the middleware stack to the web server. The web server then sends the data back to the user.

3.3 Railties

Rails itself is split into parts, which are called railties. A railtie provides initialisers and hooks in order to extend the framework's functionality. Every component of Rails is a railtie, which makes them easily extendable and exchangeable. The following section describes the core railties, which were heavily used throughout the Sapphire project.

3.3.1 ActiveRecord

The standard way of achieving data persistence and communication with a database is ActiveRecord. This railtie implements the ActiveRecord pattern, which was originally introduced by Martin Fowler in 2002 [Fowler 2002, pages 160–164]. It provides an easy-to-understand Domain Specific Language (DSL) by not only mapping corresponding columns to method names, but also the model's associations. As a result, a developer does not have to write Structured Query Language (SQL) queries by hand. Instead, ActiveRecord compiles them to suitable queries for the current database software backend.

ActiveRecord is able to handle so-called scopes. These are small parts of the DSL which are defined directly on the model. They are often used to provide mechanisms for filtering, ordering, or grouping records. Combining scopes is also possible, so reusing scopes is highly encouraged and leads to very concise code.

Usually, the development process takes place on the developer's computer, while the production environment stays untouched. This is necessary to prevent any data loss on the servers. Eventually, as development of the web application progresses, the developer will make changes to the schema of the database. Migrations are ActiveRecord's solution to this problem. These small files, which provide an incremental history of changes to the database schema, are stored in the project and are used either to setup an empty database, or migrate an existing database with all its tables and content to the revised schema.

As a database is migrated, ActiveRecord first checks which migrations have already been executed, and which are still pending. Complex changes to the schema and associated migration of data, can be tested before running them in the production environment.

ActiveRecord supports three relational SQL databases: MySQL, PostgreSQL, and SQLite. Due to migrations and the simple DSL, switching between databases is easy, since no SQL query has to be touched and the corresponding SQL syntax is chosen automatically. Similar to other railties, ActiveRecord is completely independent of Rails and its features can be used outside of Rails without any drawbacks.

3.3.2 ActionPack

ActionPack combines ActionController and ActionView, two railties which are tightly linked together. ActionController provides basic controller behaviour, while ActionView is responsible for the view layer of a Rails application.

ActionController provides a basic controller, from which all application controllers are derived. It provides fundamentals such as request hooks, basic HTTP authentication, and redirect handling. On every request, a separate ActionController is instantiated, according to the routing mechanism discussed in Section 3.2.3. An ActionController passes all of its instance variables to the view layer, where the data is rendered accordingly. A controller calls methods on various model objects and then presents the results to the user by rendering a view template.

An ActionView is usually an ERB file. It is used to generate the appropriate HTML, which is then served to the user. It uses the instance variables previously defined by the controller to insert data at specific places in the HTML structure. ActionView also provides helpers for frequently used functionality, such as date formatting, translation, and number formatting. The developer can create additional helper methods to simplify the ERB markup and extract all logic into Ruby code, leaving only simple render calls inside the ERB tags.

Chapter 4

Frontend Web Technologies

In essence, the web is a very large hyperlinked and modular application, which can be accessed with the aid of a web browser. A web developer can add additional functionality and information by setting up a web server and pointing a domain via the Domain Name Service (DNS) to this server, which then responds to HTTP queries from the web browser clients by sending HTML pages.

Current versions of many underlying technologies, such as Transmission Control Protocol/Internet Protocol (TCP/IP) [ISI 1981a][ISI 1981b] and DNS [Mockapetris 1987] have been standardised a long time ago and have not been changed since. In contrast, frontend technologies have been changing rapidly. According to W3C [2017], HTML is currently at version 5 and according to W3C [2015], the latest version 3 of CSS is still work in process. While those standards were only released to the public recently, many browsers already support at least parts of them. This section presents the latest developments of frontend technologies and highlights the most important features used in the Sapphire project.

4.1 HTML 5

HTML is a markup language based on Extensible Markup Language (XML). It is used by websites to define elements and content in a web browser. The latest version 5 introduces several new tags, attributes, and semantics, which not only improve the crawlability by search engines, but also support web developers to provide more functionality with less code and external dependencies. Furthermore, this new standard aims to improve performance on mobile devices, hand in hand with several improvements to CSS which are presented in Section 4.2.

Firstly, the HTML doctype was changed to `<!DOCTYPE html>`. Every web page containing this string in the first line is considered to be a HTML 5 document and is interpreted accordingly by modern browsers. Older versions will not recognise the new doctype and will present the page either in a compatibility mode or treat the page as if it was written in a previous version of HTML.

HTML 5 improves the language by adding structural elements such as `<section>`, `<header>`, `<nav>`, `<footer>`, `<article>`, `<aside>`, and `<figure>`. Compared to using a plain `<div>` tag, these elements improve the readability of the source code for humans as well as for search engine crawlers by adding context to otherwise unstructured data.

Furthermore, HTML 5 provides several media container elements such as `<video>`, `<audio>`, and `<embed>`. The `<video>` and `<audio>` elements provide native interfaces for several different common media types, like MP3, Ogg, H.256 and WebM, although browsers are not required to support all of them. Instead, the HTML 5 standard defines a `<source>` element consisting of two attributes: `src` and `type`, as shown in Listing 4.1. The `src` attribute specifies the URL from which the given resource can be downloaded. The `type` specifies the media type of the given resource in Multipurpose Internet Mail

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
  Your browser does not support the audio element.
</audio>
```

Listing 4.1: Using the <source> element as part an <audio> element. [W3Schools 2019]

```
<img src="kitten.jpg"
     srcset="kitten.jpg 1024w, kitten-medium.jpg 640w, kitten-small.jpg 320
         w" />

<img src="kitten.jpg"
     srcset="kitten-hd.jpg 2x, kitten-medium.jpg 1.5x" />
```

Listing 4.2: Using the srcset attribute of an img element. [RICG 2014]

Extensions (MIME) format, which helps the browser to decide which version should be downloaded. In case a browser does not support any of the specified media types, the browser simply renders the text given inside in the media container. This can then be used, for example, to instruct users to upgrade their browsers to a newer version.

HTML 5 now provides two solutions for responsive images, the <picture> element and the srcset attribute. Both of them allow the web developer to describe which image should be loaded for different display sizes. Listing 4.2 shows an example of how the srcset attribute is used with HTML 5. Given a specific size used for display it basically instructs the browser which version of an image to chose. The <picture> element extends this functionality by making use of additional <source> elements. These work similarly to the <source> elements inside media containers, but instead of specifying different MIME types the developer adds CSS media queries via the media attribute, as shown in Listing 4.3. Jehl [2014] created a JS-based Polyfill version, which backports this functionality to HTML 4 and therefore allows developers to provide improved website performance on older websites as well.

While HTML 5 adds many new features, it also drops support for several older elements. These regressions include the removal of frames, which are still extensively used, especially by older websites like TUG [2019]. Several short-hand elements previously used to apply basic styling to text, such as <u>, <s>, and <tt> have also been removed, on the premise that styling should now be done via CSS instead of HTML.

4.2 CSS 3

CSS, alongside HTML, is another very important part of web development. This standard defines a simple, yet powerful, styling language for the web. It allows developers to define multiple rules consisting of selectors and an associated set of properties with their respective values. Selectors identify elements in the HTML structure and the specified attributes are then applied to those matches. Listing 4.4 shows a set of CSS rules, which define how the headline tags <h1> and <h2> should be displayed and define how <small> elements contained within <h1> and <h2> elements should be displayed.

The first version of the CSS standard was released in 1996 by Lie and Bos [1996]. Since then, it has been improved continuously. Currently, the W3C is working on the third version of the CSS standard. Even though it has not been officially released in its final form, modern browsers already support large

```
<picture>
  <source media="(min-width: 40em)"
    srcset="kitten.jpg 1x, kitten-hd.jpg 2x">
  <source
    srcset="kitten-small.jpg 1x, kitten-small-hd.jpg 2x">
  <img src="kitten.jpg" alt="Cute kitten">
</picture>
```

Listing 4.3: Using the <picture> element. [RICG 2014]

```
h1, h2 {
  font-weight: bold;
  color: #000;
}

h1 small, h2 small {
  font-size: 0.6em;
  color: #666666;
}
```

Listing 4.4: A simple set of CSS rules.

portions of it. The new standard continues the constant effort of improving the separation of the page's textual content from the way it is actually displayed. Doing so improves the readability of the page's source code, helps search engines to identify information more accurately, and improves maintainability for developers.

CSS 3 supports many new features when compared to the previous version. First of all, several new types of selectors have been added. A prominent example is the `elementA + elementB` combinator. It matches elements of type `elementB` which immediately follow elements of type `elementA`. Many new pseudo-classes have been added as well. The most commonly used ones are `:first-child`, `:last-child`, `:nth-child(n)` and `:not(selector)`.

Furthermore, CSS 3 allows developers to create column-separated text layouts without the need for additional HTML elements, to add custom border images, and to use CSS transitions and animations in 2D as well as 3D. Developers can now add text shadows, background gradients, web fonts, and much more.

4.3 SASS and SCSS

Syntactically Awesome CSS (SASS) is a CSS preprocessor which extends the standard with additional syntactical elements, such as nesting, variables, functions, and mixins. SASS was originally defined as a separate language, which considered whitespace to be significant similar to Python. Real life applications require a syntax which is closer to CSS. Therefore, another syntax specification was added to the SASS compiler starting with version 3.X. This is called Sassy CSS (SCSS) and is a superset of CSS. In contrast to SASS, SCSS relies on block formatting and uses braces to delimit them instead of significant whitespace. Due to the closeness of the syntax of SCSS to CSS, unmodified CSS files are valid SCSS files as well. The friction caused when migrating from plain CSS to SASS was mitigated. Integrating SASS into an existing project simply requires changing the extension of the CSS files from `.css` to `.scss`.

```
$headline-color: #000;

h1, h2 {
  font-weight: bold;
  color: $headline-color;

  small {
    font-size: 0.6em;
    color: lighten($headline-color, 40);
  }
}
```

Listing 4.5: A simple set of SCSS rules, using nesting, variables, and functions.

Like in many other programming languages, variables can be defined once and then used several times throughout the code. Variables in SASS are denoted with the dollar sign. In the CSS context, variables are especially useful for colour definitions. When a base colour for borders is used over and over again, in case this colour needs to be changed, a developer only needs to alter one line of code instead of having to go through every CSS file.

SASS offers additional ways of manipulating values such as colours and sizes by using functions. Some functions are predefined, such as `lighten`, `darken`, `saturate`, and `opacify`, but the developer is not limited to those and can define further custom function as well. By using functions instead of fixed colour values, a developer can define relative colours which only depend on few base colour values. Instead of defining absolute values, a developer then uses function calls to manipulate the base colour. Listing 4.5 provides a short example of how this procedure can be implemented, resulting in the same output as shown in Listing 4.4. This example also shows the usage of rule nesting, which enhances the readability and maintainability of SCSS code and encourages developers to write SCSS in a modular way.

Functions still do not provide the flexibility needed to define a complete parametric layout. This is related to the fact that often several different CSS rules are used in combination in order to achieve a specific goal. As a result, SASS provides the concept of mixins. These are a set of either static or parameterised CSS rules, which are often used in conjunction. Listing 4.6 shows the definition of a mixin. This mixin can be used to shorten a block of text by appending an ellipsis, in case it would not completely fit the given size on the page. This mixin is then included into an existing SCSS rule, using the `@include` directive.

From version 3.1 [Dev 2015], Rails fully integrated SASS as part of the asset pipeline, which resulted in widespread usage of SASS. Today, every new Rails application includes SASS as its default CSS preprocessor. Additionally, many gems use Rails' SASS integration to provide extensible CSS frameworks, such as Bootstrap [McDonald et al. 2015] and Foundation [ZURB 2015], to the Rails community.

4.4 CoffeeScript

CoffeeScript is to JS what SASS is to CSS. It defines a programming language, which is then compiled and served as JS to a web browser. It generally aims to simplify JS by adding additional and improved language constructs. One can think of CoffeeScript as a macro-language such as LaTeXwhich is then expanded to JS. While it provides the same sets of features, it abstracts the difficult parts of JS and replaces them with improved interfaces. The closeness to JS is highlighted by CoffeeScript's documentation:

> "The golden rule of CoffeeScript is: 'It's just JavaScript'." Ashkenas [2015]

```
@mixin shortened-text($max-width) {
  overflow: hidden;
  text-overflow: ellipsis;
  max-width: $max-width;
}

.summary {
  @include shortened-text(70%);
}
```

Listing 4.6: Definition and usage of a SASS mixin called shortened-text.

While there are many small improvements over JS, only the main differences will be highlighted here. CoffeeScript relies on implicit variable definitions, allowing developers to simply assign variables right where they are needed. The compiler then creates the appropriate `var` statements at the beginning of the appropriate variable scope. This concept also provides benefits to the developer, since the same variable name might have been used in a surrounding scope. Altering such a variable's value might unintendedly influence other parts of the application. Furthermore, CoffeeScript creates a JS closure for every compiled file, in order to counteract side-effects with other scripts.

Another important difference to JS is that CoffeeScript relies on significant whitespace, similar to Python. As a result, many explicit language constructs in JS can be introduced implicitly. These include surrounding brackets for code blocks in functions, object definitions, and control blocks and surrounding parentheses for function calls and control statements.

Originally, JS was intended to be a scripting language with rudimentary support for classes. CoffeeScript tackles this problem by adding class inheritance along with the `super` method, which is used to reference the parent class. Consequently, developers are able to create rich class hierarchies, which are often necessary for single-page web applications.

CoffeeScript introduces the `@` symbol, the meaning of which depends on the context. In contrast to JS, CoffeeScript provides two different ways of defining a function by either using the `->` or the `=>` operator. The `->` operator works in a similar way to an original JS function definition and defines the `@` symbol as the equivalent of `this` in JS. In contrast, the `=>` operator proxies the `@` symbol to the `@` symbol of the surrounding scope. This is especially useful when functions are used as a callback, where it is important to stay in the same context. Listing 4.7 shows an example of how this concept might be used. It also highlights the usage of default values in function definitions and the assignment of instance variables as part of a function's signature in the `constructor` method.

Another important fact about function definitions in CoffeeScript is that every function implicitly returns the last result of an evaluation. This is very similar to Ruby's behaviour and requires developers to take special care of return values. Like in JS, there is no concept of private or public parts of a class' interface. Instead, the most common practice is to use an underscore in front of a function definition in order to "mark" it as private for fellow developers. Nevertheless, these methods are still callable from parts of the application other than the class itself.

Similar to SASS, Rails integrates CoffeeScript as part of the asset pipeline, starting from version 3.1 [Dev 2015]. Hence, CoffeeScript's popularity increased rapidly and is now used by default in every new Rails application.

```
class Greeter
  constructor: (@person = "World") ->
  greet: ->
    console.log "Hello, #{@person}!"

class Sheldon extends Greeter
  greet: ->
    knock = =>
      console.log "*knock, *knock, #{@person}!"
    setInterval knock, 500

greeter = new Greeter()
greeter.greet()
// Hello, World!

sheldon = new Sheldon("Penny")
sheldon.greet()
// *knock, *knock, Penny! *knock, *knock, Penny! ...
```

Listing 4.7: A simple Greeter class written in CoffeeScript, showing the usage of classes, inheritance, function definitions, and the @ symbol.

```
function subtract(a: number, b: number): number {
  return a - b;
}
```

Listing 4.8: Defining a `subtract` function in TS.

4.5 TypeScript

TypeScript (TS) is a language developed by Microsoft [2019], which heavily influenced the development of ECMAScript Version 6 [ECMA 2015; Wikipedia 2019]. TypeScript (TS) is a superset of JS and provides a type system layer on top of JS. TS was developed at Microsoft to ameliorate the shortcomings of JS when developing large-scale applications. Similar to CoffeeScript, TS is transpiled to JS when used in web applications.

Inline type annotations are the recommended way of declaring the types of variables and return values of functions when developing new applications with TS. With inline type annotations, developers are able to provide type annotations directly in the source code, as shown in Listing 4.8. The TS compiler detects and checks the types during the compilation process. Alternatively, TS provides declaration files for defining the types and return values of functions and objects similar to header files in C or C++, as shown in Listing 4.9. The main use for declaration files is to define interfaces for source code not written in TS, such as third-party libraries.

TS provides three primitive data types: string, number, and boolean. The special any data type is used to indicate variables which contain values of more than one data type. The void data type represents an empty value, which, for example, is used for functions without a return value. An instance of a class are either annotated with the generic Object data type, or with the class name corresponding to the class of the object.

Since TS is a superset of JS, it supports the definition of classes and supports inheritance, shown in Listing 4.10. This example also shows that it is not mandatory to specify explicit type annotations for

```
declare namespace calculations {
  subtract(a: number, b: number): number;
}
```

Listing 4.9: Using a declaration file to define the type interface of existing functions or objects.

```
class Greeter {
  protected person: string;

  constructor(person = "World") {
    this.person = person;
  }

  greet(): void {
    console.log(`Hello, ${this.person}!`);
  }
}

class Sheldon extends Greeter {
  constructor(person?: string) { super(person) }

  greet() {
    let knock = (): void => {
      console.log(`* knock, *knock, ${this.person}!`);
    };

    setInterval(knock, 500);
  }
}

let greeter = new Greeter();
greeter.greet();
// Hello, World!

let sheldon = new Sheldon("Penny");
sheldon.greet();
// *knock, *knock, Penny! *knock, *knock, Penny! ...
```

Listing 4.10: A simple Greeter class written in TypeScript, illustrating the use of type annotations, type inference, and class inheritance.

return values and variables. If types are clear from the context, TS automatically assumes the correct data type, for example, when overriding functions of a super class. It is also possible for developers to omit type annotations in certain sections of the source code, which results in TS falling back to the original JS behaviour and not performing type checks on these sections.

Sapphire does not currently use TS in its source code, since TS was not as widespread, when Sapphire was initially developed. Furthermore, Coffeescript is considered a first-class citizen of Rails since version 3.1 [Dev 2015] and is tightly integrated with the Rails asset pipeline.

4.6 Foundation

Foundation [ZURB 2019] is a very sophisticated frontend framework developed by ZURB and consisting of both CSS and JS. As ZURB puts it:

> "Foundation is made by ZURB, a product design company in Campbell, California. We've put more than 15 years of experience building web products, services and websites into this framework." ZURB [2013a]

There are many advantages of using an existing frontend framework, compared to starting a CSS and JS framework from scratch. One of the most important is that it has been used in many projects before, where many difficulties have already been overcome. Foundation is very well maintained and has support for many recent technologies defined in CSS 3, including responsive design patterns, based on media queries.

Since plain CSS files do not provide any form of parameterisability, Foundation uses SASS to define CSS rules. A developer is then able to customise many different variables ranging from breakpoint widths and element sizes, through colour codes, to font families. Additionally, Foundation provides a mature grid system, which not only ensures ease of use when developing a web page, but is also heavily tested across a large variety of browsers [ZURB 2013b].

The Foundation framework consists not only of basic CSS rules, but also provides a variety of so-called components. These are combinations of HTML snippets and CSS rules, enhanced with JS, which provide common web application User Interface (UI) elements, such as tabs, navigation bars, and overlay elements. All of these components are fully responsive, which makes them suited not only for prototyping, but also for production use out of the box.

4.7 Bootstrap

Bootstrap [Bootstrap 2019b] is a HTML, CSS, and JS frontend framework based on SCSS and jQuery [Bootstrap 2019c]. The first version of Bootstrap was initially developed at Twitter and released to the public in 2011 [Bootstrap 2019a]. Bootstrap provides a responsive CSS grid system, optimised for both mobile and desktop devices. A set of utility classes is also provided by Bootstrap, which allows developers to configure sizing, visibility, spacing, colours, borders, and shadows of elements in a responsive manner.

Furthermore, a large set of preconfigured components is provided by Bootstrap, including navigation bars, forms, modals, spinners, and pagination. Components are based on HTML snippets and CSS classes. Some components are enhanced with JS to provide additional functionality. Developers are able to customise Bootstrap to a high degree using SCSS variables, and there is a wide variety of themes created by the Bootstap community.

Sapphire currently does not use Bootstrap. At the start of development of Sapphire, a decision was made by the developers to base the frontend on Foundation, since at the time Foundation provided cleaner layouts and was more advanced than Bootstrap.

Chapter 5

Sapphire

At a university, there are many courses where students have to submit exercises to the lecturer for assessment and receive a grade based on the performance in these assessments. The task of evaluating, reviewing, and grading a mass course with hundreds of students can be very time-consuming. The sheer number of students requires a simple, yet powerful and flexible, system to manage the various tasks and exercise-specific details of a given course. This chapter describes the architecture of the Sapphire web-based course grading management system.

5.1 A New Approach: Sapphire

In Sapphire, students use a web-based submission system to hand in exercises. After the exercises have been assessed, each student is able to review their points and deductions, providing speedy and precise feedback. The lecturer is ultimately responsible for each student's grades. Often, teaching assistants (tutors) assist the lecturer in the grading of exercises. Sapphire uses a points deduction framework, in which a tutor generally only has to assess whether a particular criterion applies or not (binary decision) and the lecturer is responsible for setting the amount of deduction (number of points) corresponding to each criterion. Tutors have an easy-to-use interface where each submission by each student or student group in a specified tutorial group is accessible and can be quickly assessed. Automated tasks and checks can also be configured, in order to reduce the workload for each tutor.

At Graz University of Technology three Bachelor's degree programmes related to computer science are offered: Computer Science, Software Development and Business Management, and Information and Computer Engineering. The Computer Science degree program offers courses on a wide range of theoretical topics from the fields of computer science and mathematics, many of which are supplemented by practical exercises. The Software Development and Business Management degree program is similar to the Computer Science degree program, however some in-depth theoretical courses are substituted with courses on Economics. The Information and Computer Engineering degree program mainly focuses on topics related to electronics and signal processing. Additionally, the Graz University of Technology offers a fourth non-technical degree program called "Lehramt Unterrichtsfach Informatik", which focuses topics related to teaching computer science lessons to children. Even though each of the four degree programs specialises in a different field, all students are required to attend several core courses in the beginning, such as Internet and New Media (INM) [Andrews 2014c] and Human-Computer Interaction (HCI) [Andrews 2019d]. Students at Graz University of Technology use the TUGRAZonline system [TUG 2019], an online campus management tool, to register for courses at the beginning of each semester.

Sapphire was originally developed with the requirements of the INM course in mind, which has since been discontinued. Around 450 students attended the compulsory INM course in the first semester. INM was aimed at new students of Graz University of Technology to familiarise them with basic internet

TA report	points	Test	Group 01	Group 02	Group 03	Group 04	Group 05	Group 06	Group 07	Group 08	Group 09	Group 10	Group 11	Group 12	Group 13	Group 14	Group 15
Handed In	152		x	x	x	x	x	x	x	x	x	x	x				
Title page	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
missing	0!																
something missing	-2																
executive summary	4	4	4	4	4	2	4	4	4	4	4	4	4	4	4	4	4
missing	0!																
too short/ long (<200 or >500 words)	-2																
bad summary	-2					x											
test procedure – methodology	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
missing	0!																
faulty or missing test procedure	-2																
test procedure – user profiles + test users	5	5	5	5	5	5	5	5	5	5	5	5	2	5	5	5	5
missing	0!																
faulty or missing user profile	-2																
missing fields in overview table	-3												x				
test procedure - tasks	4	4	4	4	4	4	4	4	4	0	4	4	4	4	4	4	4
missing	0!																
missing tasks	-4																
used wrong tasks	-4									x							
test procedure - enviroment	6	6	6	6	6	6	6	4	4	6	4	4	6	6	6	6	6
missing	0!																
not IE or Firefox or Chrome (without good reason	-2																
browser not stated	-2																
version not stated	-2										x						
faulty or missing photos of room	-2							x	x			x					
test procedure - training	4	4	4	4	4	3	4	3	4	0	3	3	3	4	4	4	4
missing	0!									x							

Figure 5.1: An Excel spreadsheet was previously used for grading HCI.

technologies and the IT infrastructure provided by the university. Students of INM were required to complete five exercises and a written examination at the end of the term. The first two exercises required students to create and reply to newsgroup postings, which are used as the basis for public communication for courses at the Graz University of Technology. During the third exercise, emails had to be sent to the respective tutor and some web research needed to be conducted. The results of the web research had to be published as a mini web site during the fourth exercise. Finally, for the fifth exercise, students were given an unstyled HTML template and were required to create three distinct CSS files for it.

The initial requirements for the Sapphire course management system were extended to support the HCI course. Attending HCI is compulsory for students in the second semester. Around 330 students attend HCI per term. Participants of HCI evaluate the UI of a given web site for their practical exercises. In order to complete the HCI course, students are required to conduct both a heuristic evaluation and a thinking aloud test. For both types of interface evaluation, students hand in a plan before conducting the test and a final report after the test has been completed. Additionally, students present their findings to their tutor.

5.2 Previous Workflow with Spreadsheets

In the previous grading workflow, each tutor used a spreadsheet consisting of several worksheets: one for each exercise, one overview, and an additional sheet for adding students, exported from TUGRAZonline [TUG 2019]. The exercise worksheets were formatted as follows: The first two columns contained the ratings and corresponding point deductions, grouped into rating groups, as shown in Figure 5.1. Each rating group provided a specific number of points. Each rating indicated a criterion which a student commonly did wrong, resulting in a deduction from the points total for this rating group. The points of a rating group could not be less than zero. The first row of an exercise worksheet consisted of the students' names, sorted by surname then forename.

At the beginning of each course, an experienced tutor was responsible for copying a spreadsheet from a previous instance of the course and clearing out all evaluations and students. This spreadsheet template was distributed among the other tutors, who were responsible for maintaining the spreadsheet for their own tutorial group throughout the course.

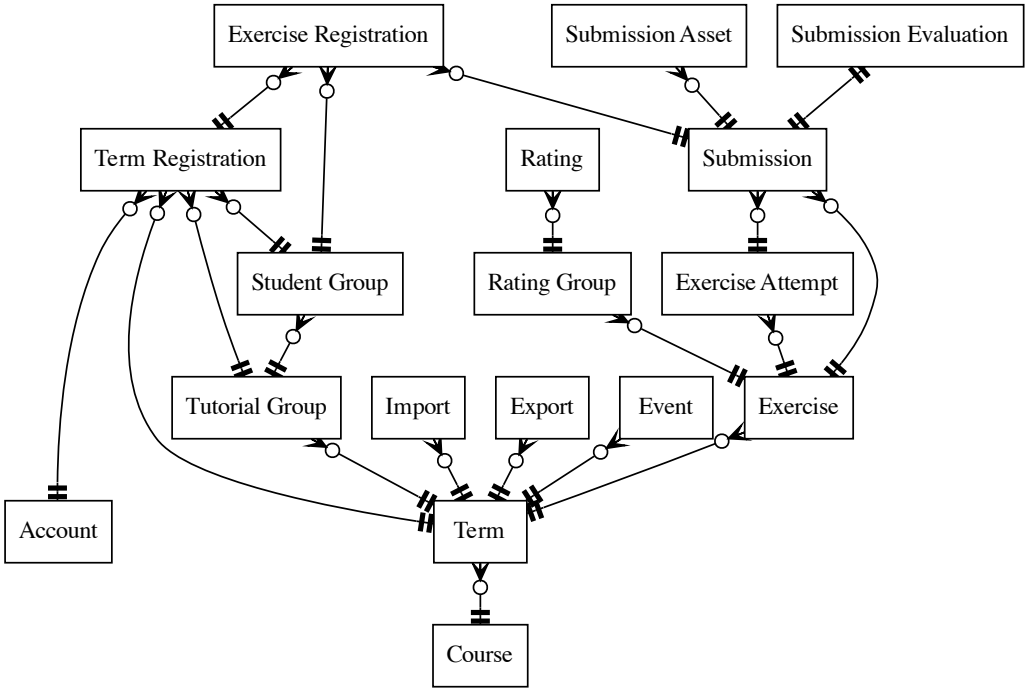

Figure 5.2: Sapphire's underlying data model.

During the course, several exercises had to be done by the students, which were submitted through a number of different mechanisms, for example by posting to a newsgroup, uploading files to a server, or simply sending an email to their tutor. The tutor had to gather and evaluate all submissions. Doing so involved multiple specific tools and grabbers, as well as automated checkers. When a tutor found an error in a student's submission, this was noted with an "`x`" in the appropriate cell of the spreadsheet.

At the end of each course, every tutor combined all submissions, renamed them to a specific naming scheme, burned them onto a Compact Disc (CD) or Digital Versatile Disc (DVD) and handed them in to the lecturer, along with the final spreadsheet containing the points and grades of all students.

5.3 Sapphire Data Model

The Sapphire data model is shown in Figure 5.2. Terms serve as the basis for each new semester of a course. Every term is comprised of a mandatory name and course to which it belongs, an optional description, and several associated records. The most important record types are:

- *Tutorial Group*: Sapphire is designed to handle large university courses of hundreds of students. The number of students and submissions is usually too large to be handled by a single person. Therefore, each student is assigned to one of several tutorial groups. Each tutorial group is managed by one or more tutors, spreading the workload amongst them.

- *Student Group*: Sapphire supports both individual and group exercises. At the beginning of a term, a student chooses a group of people to work with during the course. A student group typically comprises three or four students. Sapphire internally reflects this relationship with student groups.

- *Exercise*: An exercise defines a unit of work for students. During a term, several exercises have to be completed in order for a student to receive a positive grade. Sapphire supports both group exercises and individual exercises. Every exercise has an associated number of achievable points as well as a deadline by which a submission has to be handed in. Sapphire is capable of restricting upload sizes or even the whole upload process, if desired by the lecturer.

- *Submission*: A submission is an instance of work handed in by a student or student group for a specific exercise. Students are able to create and attach files to submissions. Tutors grade submissions based on the rating system. At the end of the term, the points of a student's submissions are summed up and the total number of points is used to calculate each student's grade.
- *Submission Asset*: A submission comprises several submission assets, which represent the files of a submission. Submission assets provide basic interactions with the file system, such as reading and writing files, calculating file sizes, and detecting content types.
- *Submission Evaluation*: Submission evaluation records are closely related to submissions and always exist in conjunction with them. Submissions are responsible for keeping track of files and enforcing access control. Submission evaluations are solely responsible for managing the grading related parts of submissions, such as keeping track of points, maintaining the data entered by tutors, and tracking the evaluation status.
- *Exercise Attempt*: For some exercises, students are allowed to submit more than one submission. For example, students are allowed to retake the final examination of HCI in case they did not receive the necessary amount of points to pass the course during the first attempt [Andrews 2019b]. Exercise Attempts are used to configure Sapphire for more than one submission per exercise and student. Exercise Attempts records are optional and are only present in the database if an exercise allows multiple attempts.
- *Rating Group*: At the heart of the grading system of Sapphire are so-called rating groups. A rating group is related to a specific part of an exercise. Each rating group is assigned a specific number of starting points, which, when summed up, make up the total achievable points of an exercise. Rating groups are configured such that the resulting points are restricted within a range of points. Usually, the point range is configured for a minimum of 0 points and a maximum equivalent to the starting points of a rating group.
- *Rating*: A rating group contains one or more ratings. Ratings are used as the basis for deductions and bonus points. By default, rating groups credit the student's submission with the starting points of the rating group. Ratings are used to alter the number of points credited to a student.
- *Term Registration*: While many records are term-specific, user accounts are not. Users are able to log into Sapphire once and have access to all terms in which they participated. The purpose of term registrations is to link user accounts to terms. Additionally, term registrations are responsible for tracking the role during the term, tutorial group, student group, and points of a user during a term. Sapphire distinguishes between three roles: *lecturer*, *tutor*, and *student*. Lecturers are responsible for administrative tasks during a term, such as managing exercises and student groups. The main task of tutors is to evaluate submissions. Both the lecturers and tutors comprise the staff members and are able to access grading-related UIs. Students are responsible for submitting exercises and are able to view preliminary results. The grading-related UIs are not accessible to students.
- *Exercise Registration*: Submissions have to be associated with the students who submitted them. In Sapphire, exercise registrations are used to establish this association. For group exercises, it is possible for one submission to have multiple exercise registrations, one for each student of a student group. Another important aspect of exercise registrations is keeping track of individual subtractions in case one student contributed less than others to a submissison.
- *Import*: At the beginning of each term, a list of students is exported in Comma Separated Value (CSV) format from the TUGRAZonline system and imported into Sapphire. Since the structure of the CSV might be subject to change, Sapphire includes a dedicated preprocessing stage. Lecturers are able to specify which columns of a TUGRAZonline export to use and to which columns of the Sapphire database these columns map.

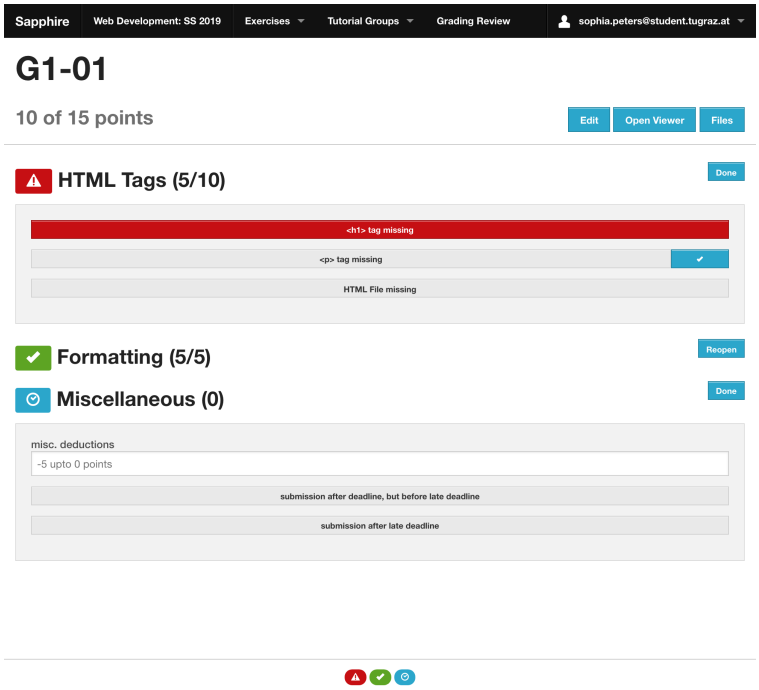

Figure 5.3: The Single Submission Evaluations interface is used by tutors during grading.

- *Export*: Sapphire strives to optimise many workflows related to term management. However, there are tasks which cannot be completely automated. Sapphire therefore provides a set of exports to allow further processing and archival of term-related data with an external application.
- *Event*: Sapphire is a multi-user application and many users are capable of making changes to the database. Keeping track of these changes manually is naturally difficult. Therefore, Sapphire automatically keeps track of important changes to the database and displays them in a concise list. Additionally, Sapphire enforces access restrictions on a per-event basis. For example, students are not able to see events created by changes to the ratings of an exercise.

5.4 Key Advantages of Sapphire

The goal of Sapphire is to improve the grading workflow as a whole. The first improvement was to take away the initial setup time, by introducing a simple mechanism to duplicate all exercises and their respective rating groups and ratings, which strips away hours of work cleaning up the spreadsheet and bug fixing.

Furthermore, "integrated evaluations" were introduced, where ratings are displayed alongside submissions on a single page per exercise and student. By removing the overhead of finding the correct submissions and files and stepping by away from the classic tabular approach, a simpler grading interface is provided for tutors, as shown in Figure 5.3. The risk of a tutor grading the wrong student, by accidentally opening a different submission, is eliminated. A tabular overview of ratings and points for each student is still provided as a separate overview page. Grading on a variety of end-user devices is now supported, since Sapphire is able to display all submissions responsively, without the need for additional applications other than a modern web browser.

Another benefit of using a web-based application is its flexibility. Releasing intermediate results on a large scale was impossible while using Excel spreadsheets. Sapphire enables the lecturer to publish

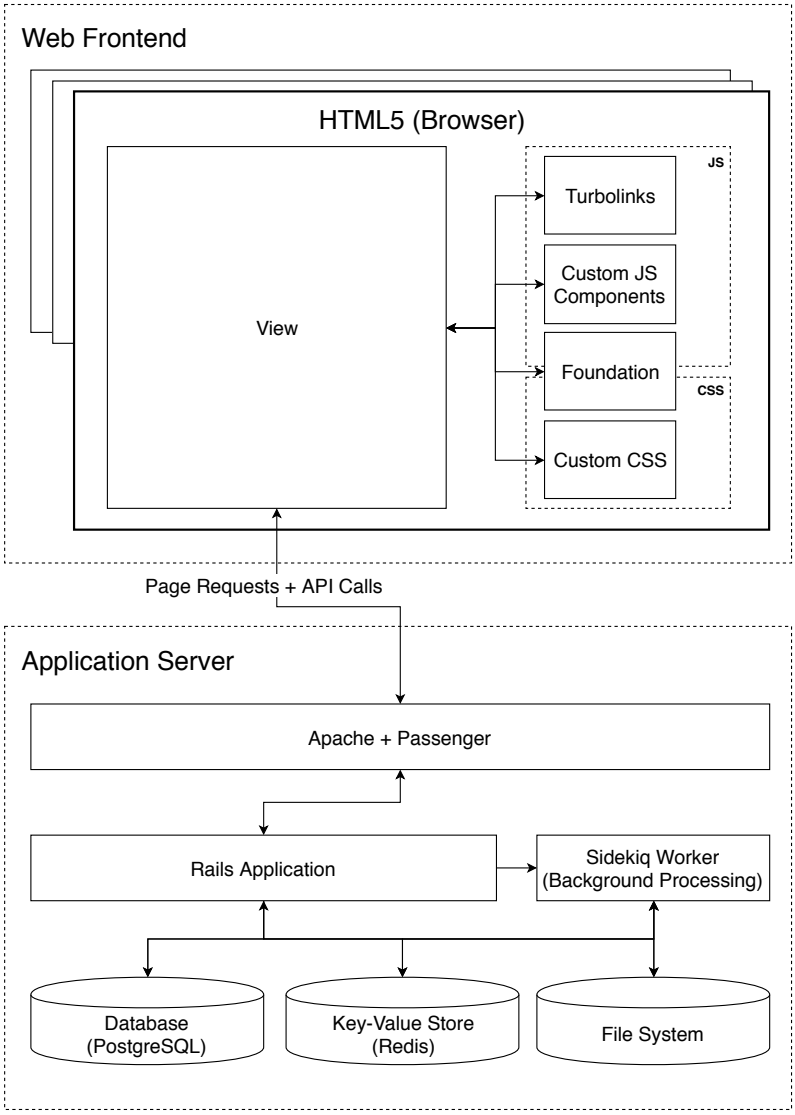

Figure 5.4: The architecture of the Sapphire web application.

results on a per exercise basis, while maintaining students' privacy, since it is only possible for a student to view their own results.

5.5 Sapphire Application Architecture

Sapphire implements a typical web application architecture, shown in Figure 5.4. The frontend of Sapphire is rendered by web browsers. Page layouts are based on the Foundation [ZURB 2019] frontend framework and rely on Turbolinks [Turbolinks 2019] to provide a speedy UI. Custom JS and CSS components are used to to enhance the UI with optimised functionality and styling.

The Rails application is hosted on an Apache [ASF 2017] web server, extended with the Passenger [Phusion 2019] module to provide Rails hosting capabilities. Sapphire utilises PostgreSQL [PostgreSQL 2019] as its main database engine and a Redis [Sanfilippo 2019] key-value store (KV Store) for handing background jobs queues. The File System (FS) of the server is used to store files uploaded to Sapphire. In addition to the web server, a Sidekiq [Sidekiq 2019] worker process is used to process background jobs outside of the request-response cycle.

5.6 Sapphire Subsystems

Sapphire consists of several submodules, each serving a particular purpose. This section describes two important user-facing parts of Sapphire: the submission viewers, used by tutors to view submitted files directly in Sapphire, and the points overview, which provides early feedback to students taking the course.

5.6.1 Submission Viewers for Tutors

Submission viewers are one of the key features of Sapphire. They provide easy access to submitted files and allow tutors to quickly open submissions, without having to worry about where these files are located in the file system.

Submissions displayed by a submission viewer are not rendered within the usual Sapphire layout. Instead, submissions are shown within a specialised environment, optimised for displaying the submission accurately and without falsification. A viewer provides additional information about the current submission within a semi-transparent overlay located at the top right corner of the page. The overlay becomes opaque when hovering over it and provides the name and matriculation number of the student whose submission is currently displayed. Additionally, with the combination of a select box and JS, an easy way of switching between the different files of a submission is provided.

5.6.1.1 HTML Submission Viewer

The HTML Submission Viewer is shown in Figure 5.5. It was originally responsible for displaying websites submitted for Exercise 4 of INM [Andrews 2014a]. Currently the HTML Submission Viewer is responsible for displaying reports submitted to the HCI course [Andrews 2019c]. The HTML Submission Viewer displays HTML files and automatically loads additional resources which are part the submission, such as CSS files, images, and videos. Since the relative path to these resources is different in Sapphire than it is on the students' web space, all relatively linked link, img, video and anchor elements have to be rewritten in order to link them correctly to Sapphire's internal serving mechanisms. Therefore, before displaying the submission to the tutor, the HTML files created by the students are parsed and all links to relatively linked assets, such as images and link tags, are replaced. Furthermore, the destinations of relatively linked anchor tags are altered to match those of the viewer. After all necessary links have been switched to match Sapphire's convention, the head and the body sections of the HTML file are extracted and placed separately into a specially crafted HTML template, to simplify later injection of further styles and HTML tags, such as the overlay. For this particular viewer, the overlay allows the tutor to easily see which further HTML pages were submitted by the student and to switch between them if desired. The original submission can be downloaded by the tutor as a ZIP file [Lindner 1993], if inspection of the original files is deemed necessary.

5.6.1.2 CSS Submission Viewer

The CSS Submission Viewer is shown in Figure 5.6. It was used for Exercise 5 of INM [Andrews 2014b] to display a website with a predefined set of stylesheets the tutor can choose between. For this particular exercise, students submitted three distinct CSS files for a predefined HTML file. The CSS Submission Viewer is responsible for applying one of the submitted stylesheets to the predefined HTML file.

The CSS Submission Viewer takes the given HTML file and renders the stylesheets inline into special containers, which by default do not affect the styling of the HTML template at all. This both reduces the number of server requests and enables stylesheets to be switched interactively via JavaScript. Additionally, some simple JS is injected, which loads the first stylesheet as soon as the browser finishes loading the page. The overlay on the top right allows switching the stylesheets to accommodate the tutor's needs. The JS replaces the previous stylesheet with a new one by replacing the contents of a special style element. Hence, stylesheets do not interfere with one another, while additionally providing the possibility of quickly switching between stylesheets, without having to reload the page.

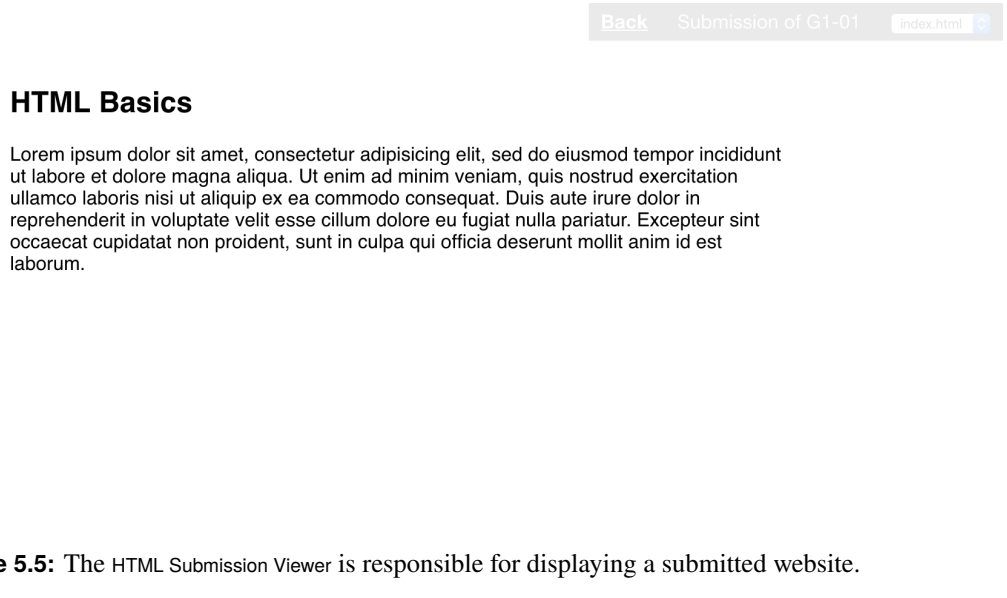

Figure 5.5: The HTML Submission Viewer is responsible for displaying a submitted website.

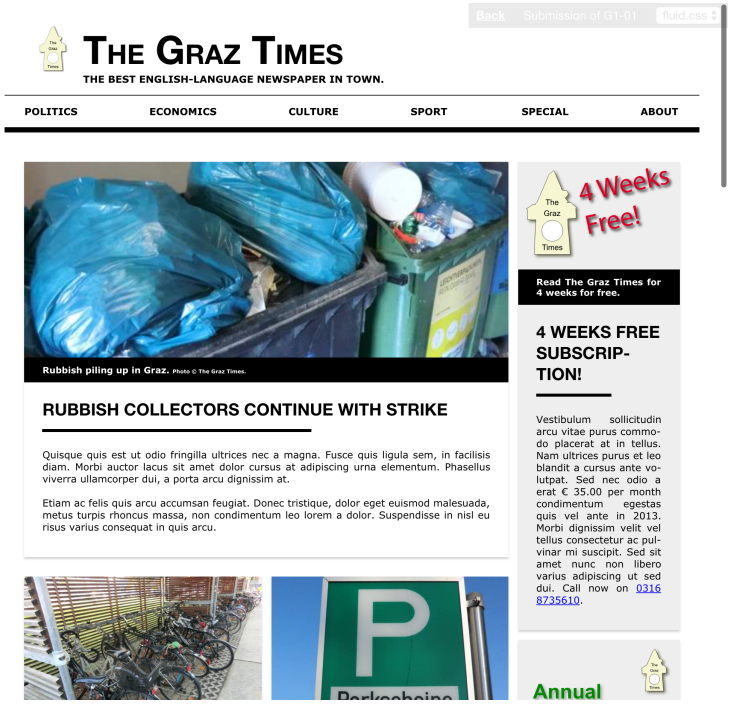

Figure 5.6: The CSS Submission Viewer is responsible for displaying submitted CSS files using a predefined HTML template.

5.6.2 Points Overview for Students

Evaluating submissions from over 400 students for a mass course like INM takes a great deal of time and effort, even if the workload is spread over multiple tutors, resulting in around 60-80 students per tutor (for the last few years). In the previous workflow with spreadsheets, exercises were given out faster than the tutors were able to finish evaluating the previous exercise submissions. Therefore, students did not receive timely feedback on their already submitted work and had to complete the following exercises in good faith. This creates a certain problem of students potentially making similar mistakes for several exercises in a row. Giving early feedback on the performance of individual students, based on actual submission evaluations, would increase the quality for upcoming exercises and reduce errors.

For each exercise, a tutor needs a certain amount of time to evaluate all the submissions. Then, at least one meeting is needed with the lecturer and all other tutors to discuss any edge cases, in order to provide consistent grading over all tutorial groups. These meetings are typically held on a weekly basis during the term and are mandatory for all tutors.

Using Sapphire, students can obtain incremental feedback on their performance and review their mistakes in the Results section of Sapphire. The tutor can evaluate submissions and the lecturer publishes preliminary results directly to the students. This speeds up the evaluation period for each exercise and helps prevent students repeating mistakes already made in one of the previous exercises.

All students have access to their personal results in the Sapphire system. The entered data is hidden from students as long as there are open issues to discuss with the lecturer and other tutors. Once the lecturer signs off, the results for the particular exercise are marked as preliminary and are only finalised after the grading review process. Email notifications are sent to students, allowing them to review their results immediately. The personal student-based points overview is sorted by exercise and shows the corresponding submission with all submitted files or other data (newsgroup posts, websites). A list of mistakes is provided to the student to indicate which requirements of the exercise specification were not met. This prevents the student from making the same mistake multiple times. If a student wants additional information about certain ratings or mistakes, the tutor's contact email address is provided to enable the students to quickly send an email.

The points overview for each student is available as soon as the lecturer publishes the first preliminary results for an exercise and remains available for some time after the finalisation of grades. Each student receives login information to access Sapphire and its subsystems. The personal points overview page is considered confidential to a single student and staff members and therefore login information is not shared or published to other students. The global points overview, containing points totals and grades is available to the lecturer and tutors. The grading scale, which indicates the points necessary to obtain a particular grade, is provided to students.

The publishing schedule for points and results is typically decided at the weekly meetings of the lecturer and tutors. This determines a fixed time slot for each tutor to finish evaluating the current exercise and publish the preliminary results. Since each tutor works on their own, the data can be published on a per-tutorial group basis.

Chapter 6

Term Management

Terms are at the core of Sapphire and serve as the basis for providing the interactions between the students and the staff administrating a university course. A term consists of tutorial groups, students and student groups, staff members, exercises, and ratings. Sapphire implements a streamlined yet easy to use family of administrative interfaces to reduce the time spent by lecturers performing administrative tasks.

6.1 Creating a New Term for a Brand New Course

Creating new terms in Sapphire is a simple task. By navigating to the course overview page, a lecturer is able to click on the plus button at the end of a course's term list. The New Term Modal dialogue window for creating a new term is opened, as shown in Figure 6.1. After entering a title and an optional description, a new term is created by clicking the Save button.

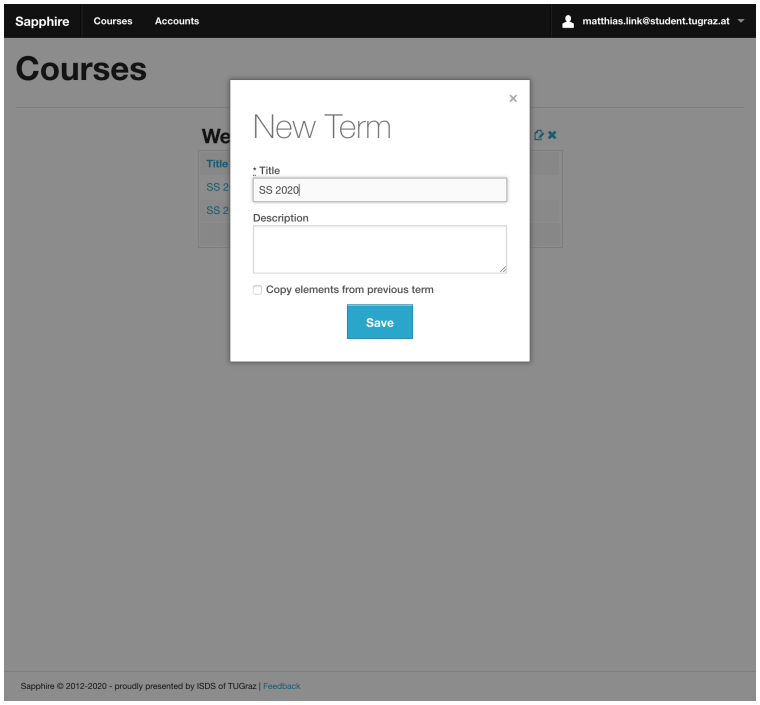

Figure 6.1: The New Term Modal used for creating new terms.

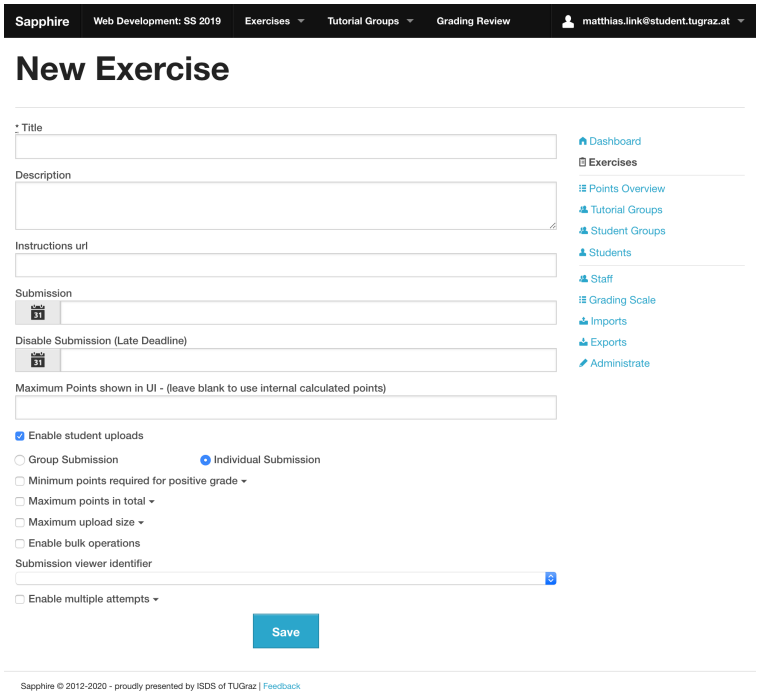

Figure 6.2: The Exercise Form to manage the key attributes of an exercise.

6.2 Managing Exercises

Exercises are managed through the Exercise Form, shown in Figure 6.2. It allows the lecturer to specify the title, description, and a URL with detailed instructions for an exercise. Sapphire supports both a soft and a hard deadline. Students are required to upload submissions before the soft deadline, to avoid incurring a late submission penalty. Once the hard deadline has passed, uploads are prohibited.

Disk space is one of the main concerns when accepting uploads from hundreds of students. Sapphire allows lecturers to specify the maximum submission size in bytes. Specifying an upper limit on submission size allows the staff to estimate the size requirements for a term and increase the size of hard disk storage accordingly.

Sapphire supports both individual and group submissions. While individual submissions belong to only one student, group submissions belong to multiple students of a student group. Student groups are sometimes subject to changes during a term. Sapphire assumes the students of a student group at the time of a submission are responsible for that submission.

Sapphire calculates the maximum points shown in the UI of an exercise, based on the sum of starting points of rating groups. While this procedure is sufficient most of the time, the need sometimes arises to customise the maximum points. It is possible for lecturers to override the maximum points shown in the UI, which impose a soft limit on achievable points, although students are still able to receive more points than shown in the UI by receiving bonus points for a submission. Lecturers are also able to specify a hard limit on the maximum achievable points per exercise. It is necessary to configure a hard limit on the maximum points of an exercise if ratings are configured to add to the starting points of one or more rating groups instead of deducting from the starting points.

Furthermore, it is possible for a lecturer to define a minimum number of points threshold on a per-exercise basis. Students who either do not attempt this exercise or acquire less points than the specified threshold, automatically receive a negative grade, regardless of their total points. For example, by utilising the minimum points threshold, lecturers are able to require students to correctly answer a minimum number of questions in an examination.

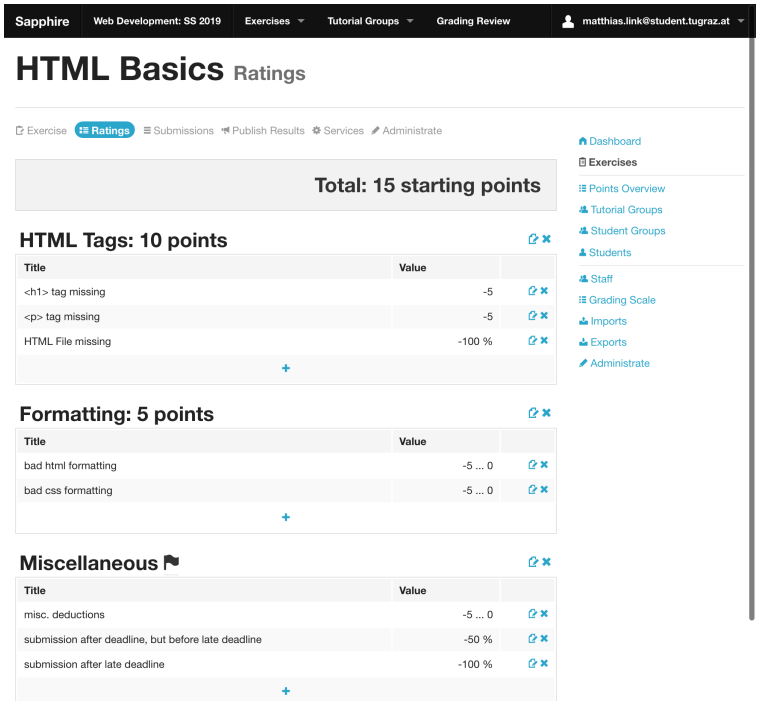

Figure 6.3: Rating groups and ratings in the Ratings Editor. Individual ratings are collected into rating groups.

Uploads are the most common way for students to submit files for an exercise. However, uploads are not the only way of creating submissions in Sapphire. Disabling student uploads and enabling bulk operations allows tutors to create submissions on the behalf of the students. Bulk grading operations, as described in Section 8.1.6, are designed for entering the results of examinations, such as the HCI Multiple Choice Test [Andrews 2019b]. Tutors sometimes need to view a submission in a specific environment, as described in Section 5.6.1. By specifying a submission viewer, tutors will see an Open Viewer button during the grading process.

Some exercises, such as the previously mentioned HCI MC Test, allow students to attempt the exercise multiple times during a term. In order to use this feature, a lecturer has to enable the Multiple Attempts feature and customise the list of exercise attempts. Sapphire only considers the points acquired in the most recent attempt when calculating the final grade.

6.3 Managing Rating Groups and Ratings

The Ratings Editor is a simple yet powerful editor, providing access to the rating system of Sapphire. Ratings and rating groups are closely related. Both can be edited through a simple list-based interface, as shown in Figure 6.3.

6.3.1 Rating Groups

A rating group can be added to the list for a specific exercise by clicking on the Add Rating Group button. The New Rating Group Modal is opened allowing the user to configure a new rating group, as shown in Figure 6.4. Rating groups require a title and the number of starting points. Optionally, a description can be specified. New rating groups are appended to the end of the list. Lecturers are able to reorder rating groups by dragging and dropping them into the desired position.

Every rating group has an associated points range, which by default ranges from zero to the number of starting points. Ratings modify the number of points received by a student for a specific rating group.

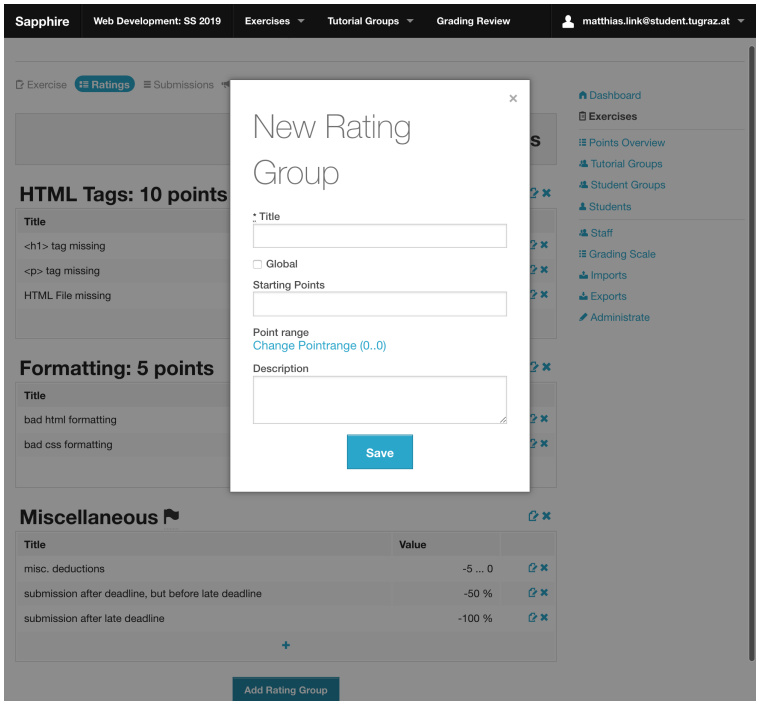

Figure 6.4: The New Rating Group Modal of the ratings editor.

During grading, the tutor activates a particular rating if its criterion applies to that submission. It is possible that the sum of all potential deductions is greater than the number of starting points. As a result, it would be possible for students to receive negative points for a rating group, affecting the points gained from other rating groups. Exposing the points range to the lecturers allows for more fine-grained control over how overflows are handled.

The global option modifies the influence of percentage deductions. Usually, deductions are restricted in scope to the rating group. Global percentage deductions are used to deduct a percentage from the final points for a submission. Global percentage deductions are applied after all the local rating group points have been calculated.

6.3.2 Ratings

Individual ratings are used to alter the points initially credited to a rating group. New ratings are added to a rating group by clicking the Plus icon at the bottom of the ratings table of the rating group. A new modal is opened, containing the New Rating Modal shown in Figure 6.5. Ratings require a title and a rating type.

Sapphire supports a variety of different rating types, as shown in Table 6.1. Fixed ratings are simple boolean ratings displayed as buttons during the grading process. Variable ratings allow for customisation by the tutors and are displayed as an input field. Ratings have an individual set of customisable options, each with their own restrictions. Deductions must be negative, while bonus points must be positive. Item-based ratings provide a multiplication factor, by which the number entered during the grading process is multiplied.

The option Display this rating during bulk operation instructs Sapphire, as the name suggests, to add this rating to the bulk operations interface, described in Section 8.1.6. Specifying an Automated Checker Identifier allows Sapphire to perform simple grading tasks automatically via the command line. Additionally, a lecturer can specify a description, which is shown during the grading process as a tooltip.

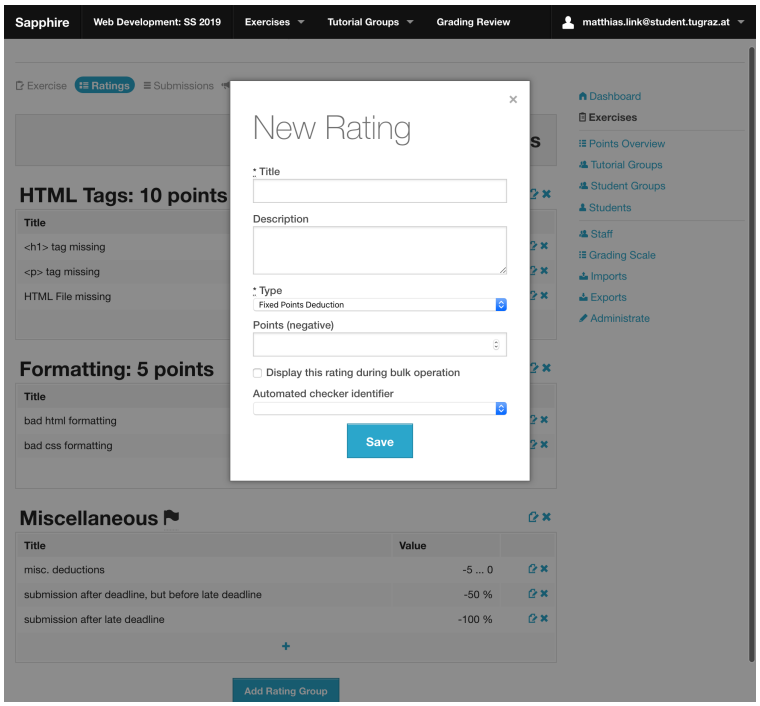

Figure 6.5: The New Rating Modal of the ratings editor.

Name	Options	Display	Use Case
Fixed Points Deduction	Points (Negative)	Button	Simple deductions, e.g. missing the group name in a report.
Fixed Percentage Deduction	Percentage (0..100)	Button	Deductions on top of other deductions, e.g. -50% for submitting after the deadline.
Variable Points Deduction	Minimum Points, Maximum Points	Field	Deductions based on a tutor's judgement, e.g. quality of a section.
Variable Percentage Deduction	Minimum Percent, Maximum Percent	Field	Deductions based on a tutor's judgement, e.g. overall quality of a report.
Per Item Points Deduction	# items minimum, # items maximum, Multiplication Factor	Field	# missing items in a collection of expected length, e.g. expected 8 sceenshots, only 5 submitted.
Per Item Points	# items minimum, # items maximum, Multiplication Factor	Field	# of expected items, e.g. correct answers in an examination.
Fixed Bonus Points	Points (Positive)	Button	Simple bonus points, e.g. hand-made graphs.
Variable Bonus Points	Minimum Points, Maximum Points	Field	Bonus points based on tutor's judgement, e.g. bonus depending on submission quality.
Plagiarism	*none*	Button	Fixed -100% global deduction for plagiarised submissions.

Table 6.1: The types of ratings supported by Sapphire. The Options column shows which additional options are available to lecturers during configuration. The Display column shows which input is shown to staff members during grading.

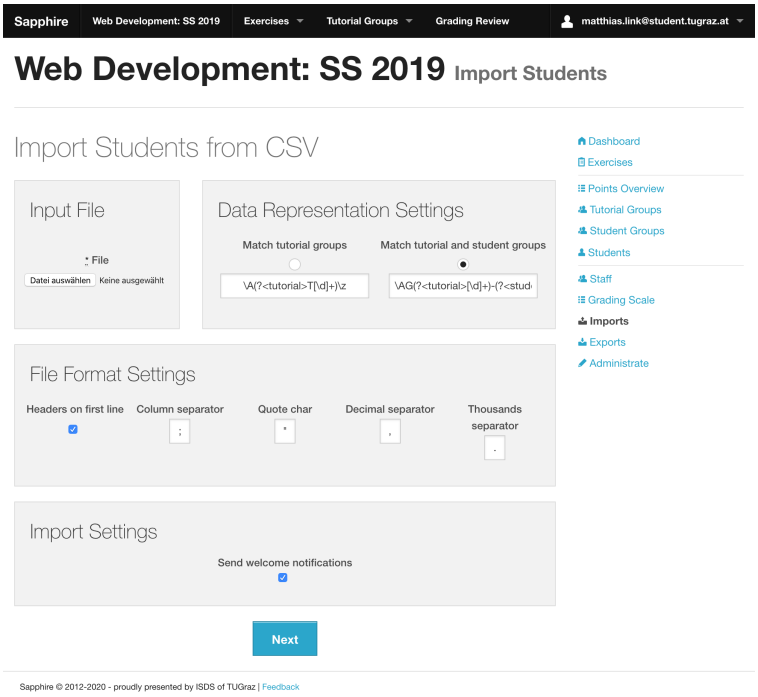

Figure 6.6: The Student Import Form is used to import students at the beginning of a term.

Sapphire monitors changes to a rating. Whenever a change occurs, Sapphire automatically marks all associated evaluations and submissions as outdated. If tutors miss changes to a rating during a meeting, they are able to quickly identify all affected submissions in the submission list.

As mentioned previously, new ratings are added to the end of a rating group. A rating can be repositioned within a rating group by drag-and-drop. Sapphire also supports dragging and dropping ratings from one rating group into another.

6.4 Student Management

Students are enrolled in a course for a particular term. During a semester, students create submissions to exercises in order to receive a grade at the end of the term. Every student receives a dedicated Sapphire account allowing them to participate in one or more terms.

The vast majority of students are imported into Sapphire following an export from TUGRAZonline, as described in Section 6.4.1. Once the term is set up, students need to be welcomed to the course. Section 6.4.2 describes the emails sent to students at the beginning of the term, called Welcome Notification emails. During the course of a term, students might sign up late or leave the term unfinished. The process of managing individual students is presented in Section 6.4.3. Section 6.4.4 describes the interface for managing student groups.

6.4.1 Importing Students From TUGRAZonline

At Graz University of Technology, students use the TUGRAZonline intranet system to manage their studies and course enrollments [TUG 2019]. At the beginning of each semester, students select their desired courses and sign up directly in TUGRAZonline. During the sign-up process, students are able to choose their preferred student group within a particular tutorial group. Manually enrolling hundreds of students in a term would require significant time and effort. Luckily, TUGRAZonline provides the means to export a CSV of students enrolled in a course. Sapphire can import this CSV file to automatically enroll students in a Sapphire term.

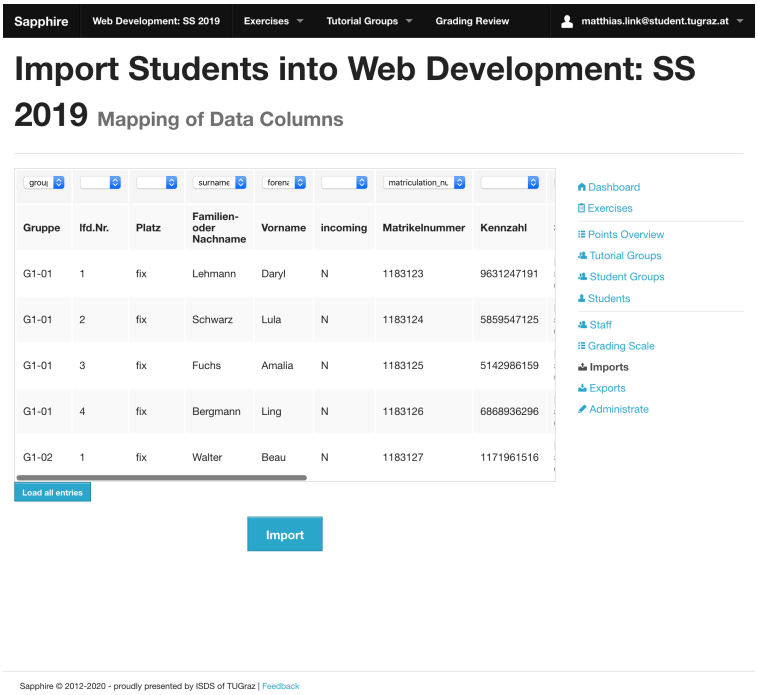

Figure 6.7: The Import Mapping Editor specifies which columns of the student CSV file to import.

Importing students into Sapphire is a multi-step process. First, a lecturer selects the import file in the Student Import Form, shown in Figure 6.6. The data representation settings are used to customise the way Sapphire handles the tutorial group string of the TUGRAZonline export and are based on regular expressions. Since student groups are optional, the import is customisable to either only match a student's tutorial group or to match a student's tutorial group along with a student group.

Even though widely used, the CSV format is not standardised. Sapphire is capable of dealing with many well-known variants, by allowing customisation of columns separators, quote characters, decimal separators, and thousands separators. The Headers on first line option instructs Sapphire to ignore the first line of the CSV, since it consists only of headers. The Send welcome notifications option configures Sapphire to send welcome notifications to all students once the import has finished, as described in Section 6.4.2. Sapphire automatically populates the form with default options, reducing the workload as well as the possibility for errors.

Once the basic import options are configured, the process is continued by clicking the Next button. The user is presented with to a new screen called the Import Mapping Editor shown in Figure 6.7. The editor displays a table of the first few rows of the CSV file. Above the first row, Sapphire renders a set of select boxes for the lecturer to choose the appropriate mapping of columns to Sapphire's internal variables. Sapphire pre-populates the select boxes based on the headers of the CSV file.

The import is started by clicking the Import button. Internally, Sapphire takes care of creating tutorial groups, student groups, and term registrations. New student accounts are created based on the email addresses specified in the import file, if they do not previously exist. Once this process is finished, Sapphire sends out welcome notifications, if the corresponding option has been selected.

6.4.2 Sending Welcome Notifications

Sapphire is designed to be used with dedicated user accounts. Since accounts are created during the import phase, there is the need to inform owners of newly created accounts about their log-in credentials,

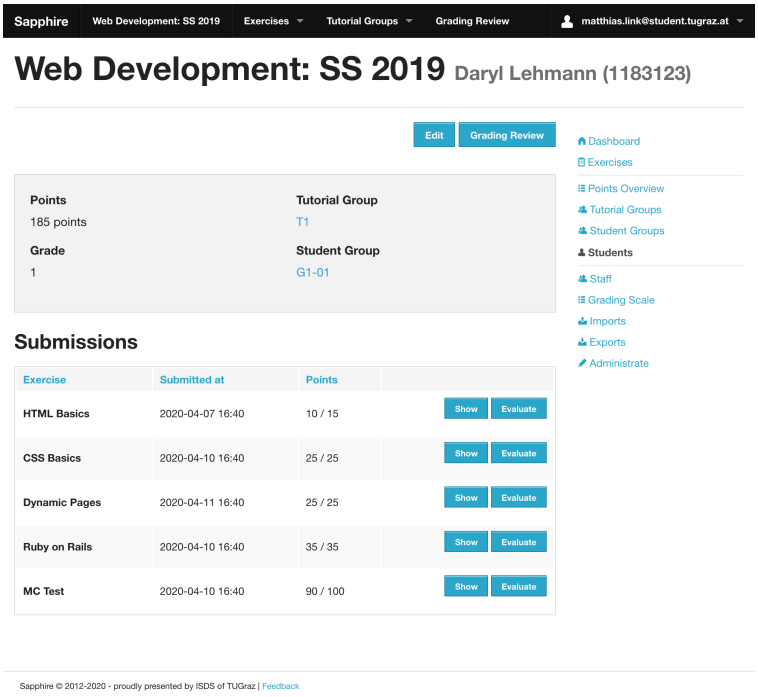

Figure 6.8: The Student Detail Panel provides an overview of key attributes and submissions of a student.

and previously existing students about the start of a new term. To this end, Sapphire sends Welcome Notification emails.

Sending passwords via email raises safety concerns. Instead, Sapphire instructs new students to use the Forgot Your Password feature to request an email with a token to change their password. Existing students are also instructed to use this feature, in case they have forgotten their password from a previous term.

6.4.3 Managing Students of a Term

Most students are automatically created by the student import and assigned to a term of a course at the beginning of the term. However, there are some cases, where students need to be added to a term, for example international students arriving late, or removed from a term, if a student decides to quit a course mid-term. The student management feature provides of a family of interfaces with this functionality.

An overview is provided for each student of a term, as shown in Figure 6.8. The Student Detail Panel provides a summary of key attributes of a student: the tutorial group, student group, total points, and grade. Additionally, a list of submissions is provided summarising the attended exercises as well as the achieved points.

New students are added through the Add New Student Editor, shown in Figure 6.9. First, a lecturer searches for either the student's name or matriculation number and selects the matching record from the dropdown. Next, the tutorial group must be selected, as well as an optional student group. Finally, the student is added to the term by clicking Save.

Editing a student is handled via the Student Editor, which is very similar to the Add New Student Editor in terms of functionality and is shown in Figure 6.10. It differs in that it does not provide a search field. Beneath the editing form is the so-called Danger Zone, a concept regularly used throughout Sapphire. It signals potentially dangerous actions to the user along with a short description of what damage the action might cause. By clicking Delete Student, the student is removed from the term.

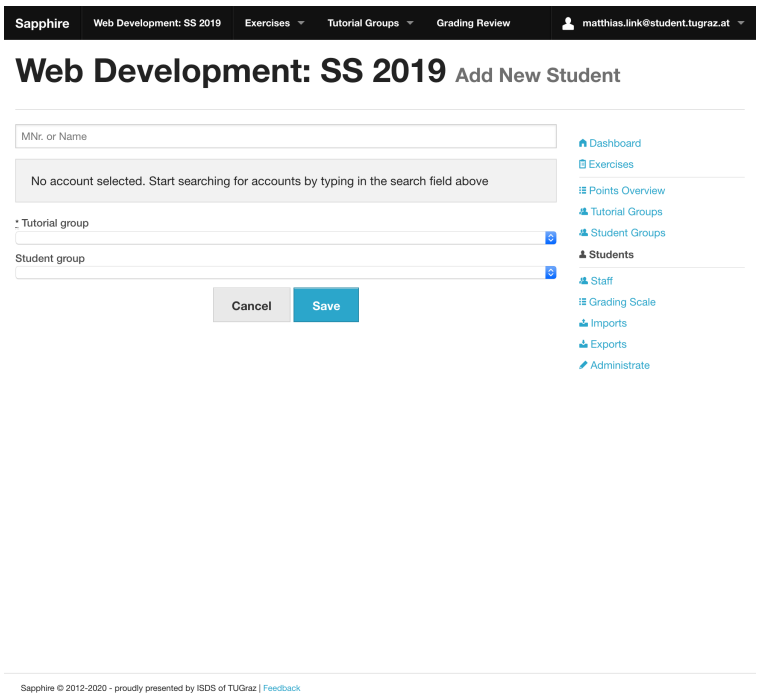

Figure 6.9: The Add New Student Editor is used to search for and add new students to a term.

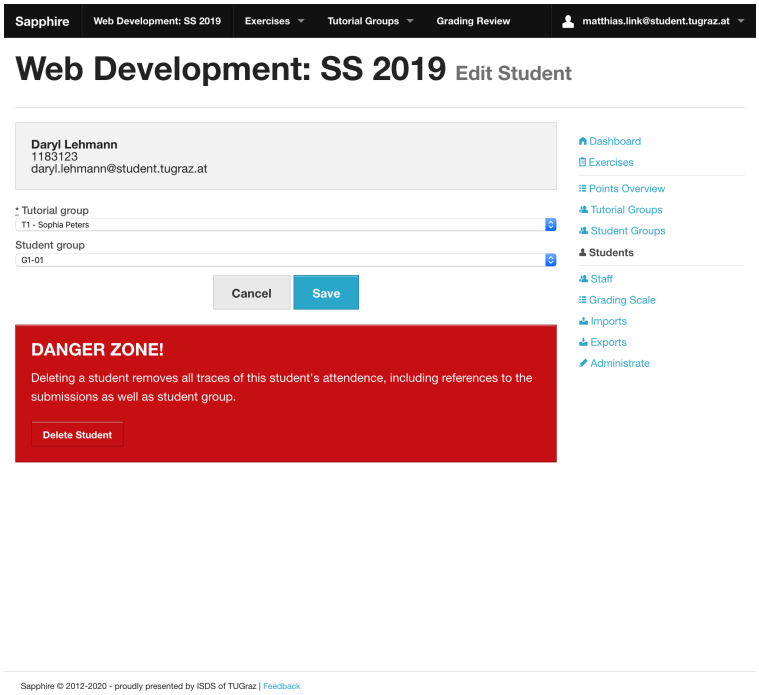

Figure 6.10: The Student Editor is used to alter key attributes of a student during a term. It can also be used to delete a student and all of their associated data from the term.

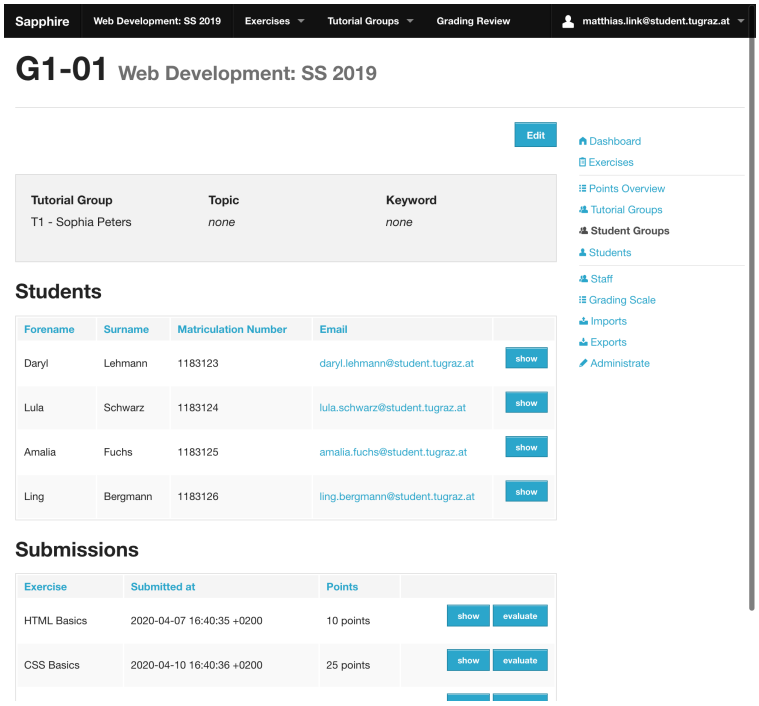

Figure 6.11: The Student Group Detail Panel provides an overview of key attributes of a student group including its students and submissions.

6.4.4 Student Group Management

Student groups are usually created in the process of a student import, similar to the students. However, it is sometimes necessary to change student group assignments during the course of a term. The Student Group Detail Panel provides an overview page of the key attributes of a student group, as shown in Figure 6.11. The Student Group Detail Panel is similar in terms of functionality to the Student Detail Panel, presented in Section 6.4.3. The Student Group Detail view consists of a panel showing the associated tutorial group, topic, and keyword. The description of the student group is shown as well, in case it is filled in. Additionally, a list of associated students and submissions is provided to tutors and lecturers of the term.

The Student Group Editor consists of two columns, as shown in Figure 6.12.. The first column is used to change the metadata of a student group. When editing an existing student group, a Danger Zone section is displayed, allowing the lecturer to remove the student group from the term. Student groups must have a title and be assigned to a tutorial group. Optionally, a lecturer can specify a topic, keyword, and a description for a student group. The second column is responsible for searching for students. A search is issued by typing into the search field. In the background, an Asynchronous JavaScript And XML (AJAX) call is sent to Sapphire and a list of corresponding hits is returned. Students are added to the student group by dragging and dropping students from the search results into the Students section. To remove students from the student group, a lecturer has to hover over a student in the list and click the red Cross button which then appears. Changes performed by the lecturer are not persisted to the database until the Save button is pressed.

6.5 Staff Management

Staff members are responsible for administrative tasks during a term. Sapphire supports both multiple lecturers during a term as well as multiple tutors per tutorial group. Staff members are usually added after the student import has finished, since the import is responsible for creating tutorial groups.

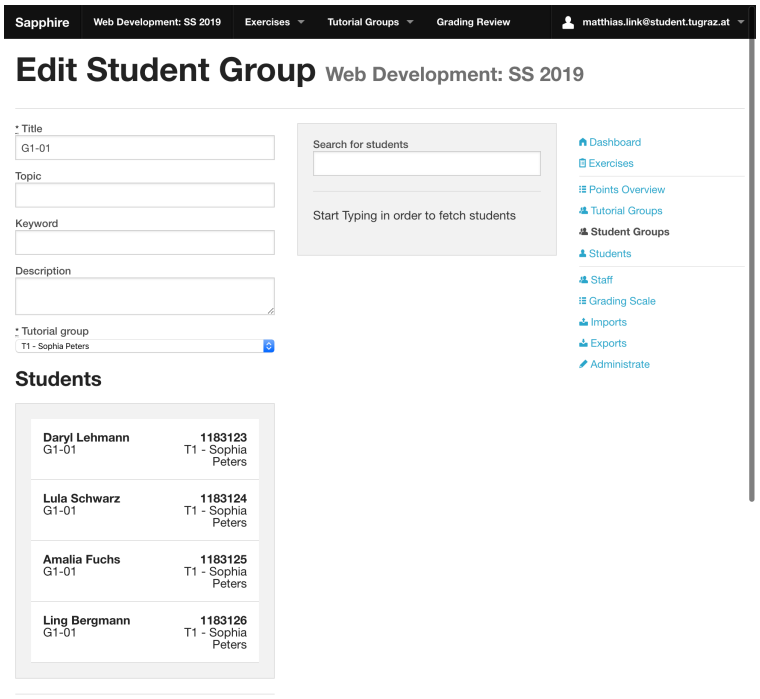

Figure 6.12: The Student Group Editor is used to create and alter student groups during a term. New students are added by searching in the right column and dragging students to the Students list in the left column. Students are removed by hovering over the corresponding entry in the Students list and clicking on the red Cross button.

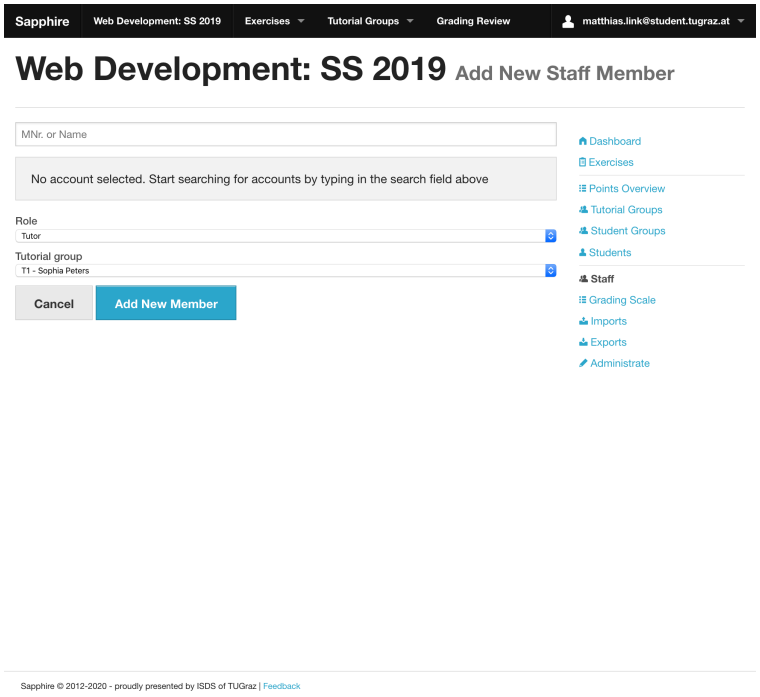

Figure 6.13: The Add New Staff Member Editor is used to add new staff members to a term.

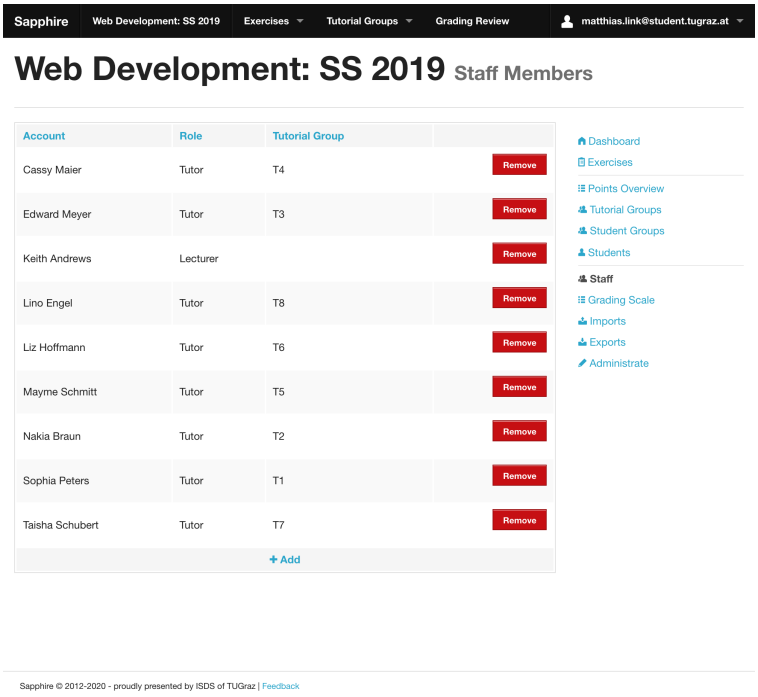

Figure 6.14: The Staff Members Table shows the staff members currently associated with a term.

To add new staff members, the Add New Staff Member Editor is used, shown in Figure 6.13. First, a lecturer chooses a user account by typing in the search field and selecting the desired account from the dropdown. Then, one of two roles is selected. Lecturers are responsible for managing the term as a whole. Tutors are responsible for grading students in their tutorial group. Finally, the staff member is added to the term by clicking the Add New Member button. Staff members can be removed by clicking the Remove button in the Staff Members Table shown in Figure 6.14.

6.6 Preparing for a New Term of an Existing Course

Consecutive terms are only loosely connected to a previous term through the course association. This design decision provides flexibility on a term-to-term basis. However, this flexibility comes at a cost, when a new term is created at the beginning of a new semester. Usually, the course structure of the new term is very similar to the previous one. Lecturers would have to invest much time and effort in order to prepare a new Sapphire term from scratch to mimic the previous term. To ameliorate this, Sapphire provides a dedicated term duplication service. Its main purpose is to reduce the setup time of a new term by basing it on a previous one.

While creating a new term, a lecturer is able to select an option called Copy elements from previous term. Upon checking this option, a new box is shown in the New Term Modal, as shown in Figure 6.15. First, a source term has to be chosen. Next, the lecturer has to decide which elements to copy to the new term. The available options include the lecturers, exercises and their associated ratings, and the grading scale. Finally, Sapphire asynchronously prepares the new term once the form is submitted.

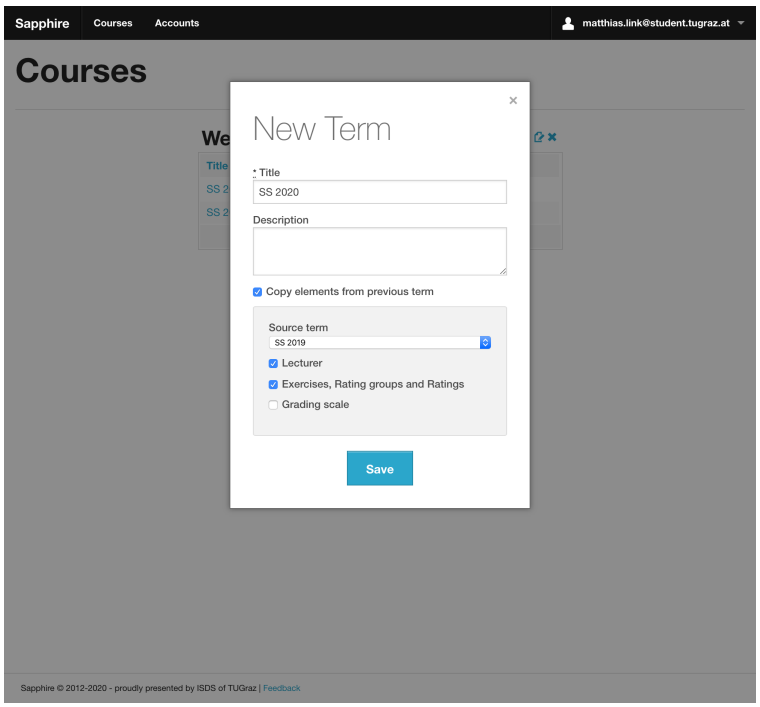

Figure 6.15: Checking the Copy elements from previous term checkbox a new box displays inside the New Term Modal. Lecturers are able to customise the duplication process, by selecting to copy the Lecturer, Exercises, Rating groups and Ratings, and Grading scale elements from the previous term.

Chapter 7

Submission Management

One of the main tasks of Sapphire is to keep track of submissions and attached files. In Sapphire, a submission comprises several different sub-concepts, each responsible for a specific task in the submission and grading process. Students are provided with a powerful submission editor, enabling them to upload individual files or a ZIP [Lindner 1993] archives. Sapphire stores files outside of the public directory of the web server. At runtime, a file system is emulated, providing common user interactions with the files of the submission without compromising the security and integrity of the web server.

7.1 Internal Submission Structure

In the context of Sapphire, a submission consists of a family of related classes, each responsible for a single task:

- *Submission*: Submission records are used for storing a reference to the exercise and, for group exercises, a reference to the student group which submitted the exercise. It is also responsible for checking the combined file size of all submitted files.
- *Submission Asset*: A submission usually contains several records of class SubmissionAsset. This class is in charge of storing and retrieving files submitted by students. Additionally, it keeps track of a file's MIME type, which is then used while serving files through the application itself. Furthermore, it provides the means to read the contents of a file and convert it to 8-Bit UCS Transformation Format (UTF-8) on the fly. This functionality is especially useful in the view layer, since Sapphire provides HTML views with UTF-8 encoding.
- *Submission Evaluation*: Once a submission is created, Sapphire additionally creates a SubmissionEvaluation record in the database. A Submission Evaluation is responsible for managing the resulting points of the submission. Additionally, it monitors when and by whom the latest change was made.
- *Exercise Registration*: Sapphire needs to keep track of whom a submission belongs to. Records of type ExerciseRegistration are responsible for this task. An exercise registration consists of three references: the exercise, the student, and the submission. Additionally, exercise registrations provide basic sanity checks and a place to put individual point reductions for each student. The use of the latter is restricted to group exercises. Sapphire does not require a submission to have exercise registrations. In some cases, submissions are not directly uploaded to Sapphire, but are grabbed from another service, such as newsgroup postings. There is no guarantee that Sapphire is able to associate submissions with a student in these situations. In order to prevent loss of data, Sapphire is capable of handling anonymous submissions, which are associated to a student at a later point in time.

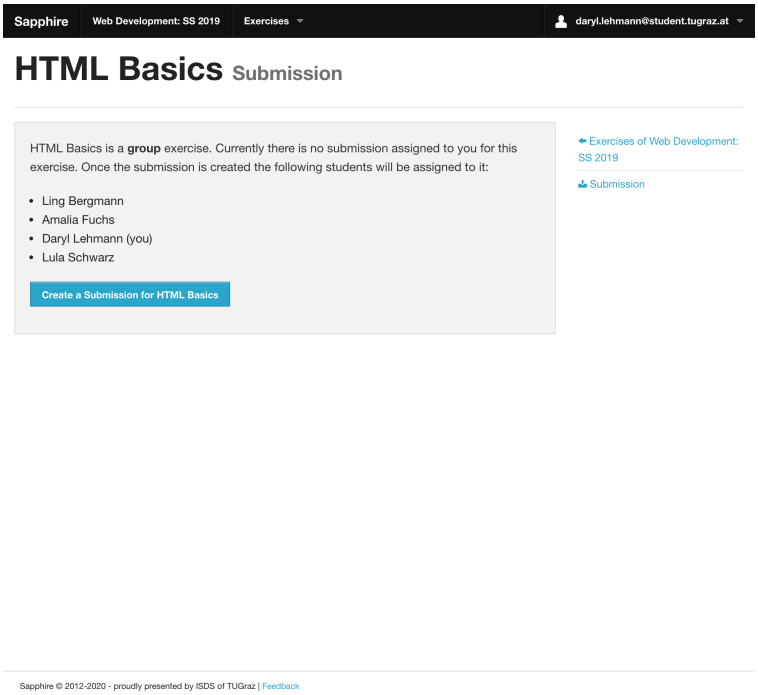

Figure 7.1: Creating a group submission as a student. Sapphire asks the student to confirm the members of the group.

7.2 Submission Editor

There are several ways of submitting files to Sapphire. The most straightforward is to upload files directly to Sapphire, using the Submission Editor. While uploading files is the most basic approach, Sapphire is capable of accepting submissions from other sources as well. In the context of Sapphire, the term Submission Editor refers to a family of interfaces, which allow students to upload and modify their submissions.

7.2.1 Creating New Submissions

The first time the student clicks on an exercise in Sapphire, Sapphire asks the student to confirm the current group constellation, as shown in Figure 7.1. The confirmation step allows the students to verify that the group members are up-to-date. Once the student presses the Create a Submission for Exercise button, Sapphire internally creates the related database records and the student is redirected to the Submission Tree.

7.2.2 Submission Tree

The Submission Tree is central to the Submission Editor. It consists of the current directory path, followed by a table of files and subdirectories, with their respective file sizes, as shown in Figure 7.2. Sapphire does not actually store the submitted files this way, but instead emulates the FS at runtime. In Sapphire, all SubmissionAsset records are stored in a single table with an associated file path. The associated files are stored in a separate directory not accessible to the web server. While rendering the Submission Tree, Sapphire constructs a tree of directories and files in memory and uses this virtual tree as the basis for the view, hence the name Submission Tree.

Emulating a FS implies additional workload for the server as well as decreased performance compared to plain file system operations. However, an emulated FS also provides key benefits. Accepting file uploads always imposes a security risk. It is crucial not to execute files uploaded to the server, since they might contain malicious code. Additionally, it is important to manage access to the uploaded files. A SubmissionAsset must only be accessible to the corresponding student and student group as well as staff

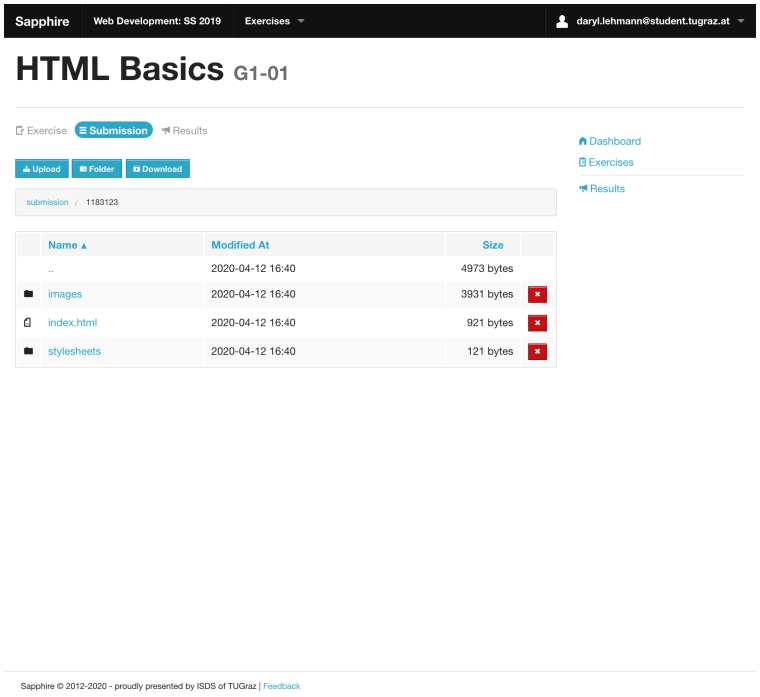

Figure 7.2: The Submission Tree as seen by a student. The Submission Tree emulates a file system similar to that of a desktop operating system on top of the internal storage mechanism of Sapphire.

members. While it is possible to configure a web server to provide this functionality, it is often difficult to implement and maintain. Managing access control in the web application is simpler and more reliable. Since the files are not stored in a publicly visible directory, the web server will not try to execute them in response to an external request.

The contents of the current directory are displayed in the Directory Table. The table consists of five columns: icon, name, modification date, size, and a Remove button. The headers of the Directory Table provide special links, allowing users to sort the table by name, modification date, or size.

The Submission Tree provides an intuitive system for navigating the directory tree. Parent directories are either accessible through the path indicator, or the `..` link inside the current directory. Subdirectories are accessible by clicking on a directory name in the current directory table. Sapphire always shows a root folder called *submission* in the path indicator, to allow users to quickly navigate to the root directory.

Users are able to remove files from the submission by clicking the red Remove button. Sapphire is both capable of removing single files as well as whole directories. In order to prevent data loss, Sapphire asks for confirmation before removing the files from the submission. Many web applications only hide records from the user without physically deleting the records. Sapphire does not follow this philosophy, but instead physically deletes the records as well as their associated files.

Sapphire provides a download button above the Submission Tree view. Via a click on the download button, users are able to download the contents of the current directory including subdirectories. Since HTTP does not support downloading multiple files in one request, Sapphire creates a zip archive on-the-fly and streams it to the user.

7.2.3 New Folders

Users are able to create new folders via a click on the New Folder button. The New Folder Modal is opened, as shown in Figure 7.3. Sapphire automatically checks if the folder already exists, once a user starts typing the name of the new folder. If no such folder exists, the Create button is enabled, otherwise disabled.

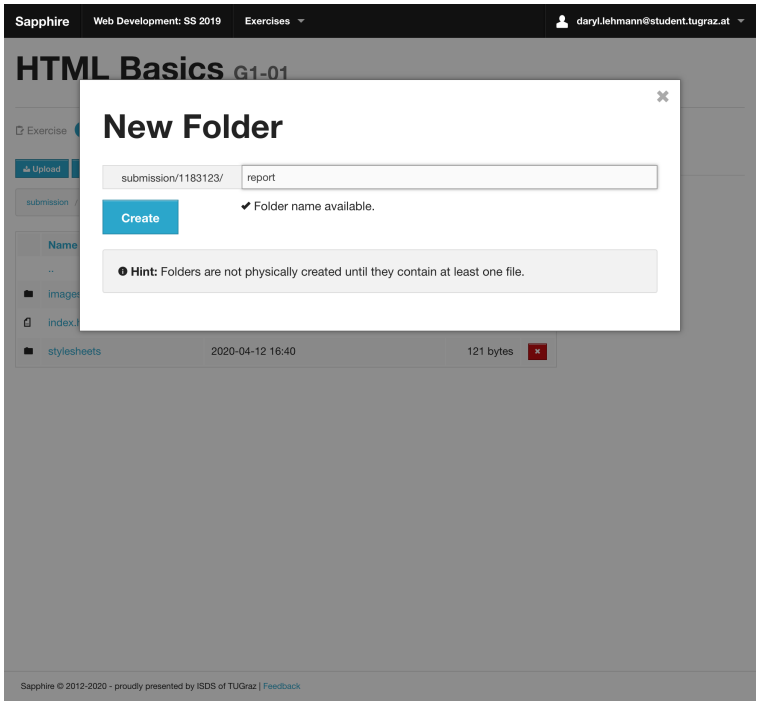

Figure 7.3: The New Folder Modal creates a new folder from within the Submission Tree view.

Sapphire does not physically create folders. Instead, folders are based on the paths of the SubmissionAsset records. Upon submitting the New Folder Form, Sapphire simply redirects the user to the corresponding URL, without modifying any records in the database. Since no SubmissionAsset records yet exist within that path, Sapphire shows the new empty folder to the user.

7.2.4 Uploading Files

Uploads are handled through a dedicated Upload New Files Modal, which is opened by a click on the Upload button in the Submission Tree, as shown in Figure 7.4. The form provides a large drop zone area. Users are able to drag and drop files from their desktop into the drop zone. Dragging and dropping single files is tedious for submissions containing many files. Therefore, Sapphire is capable of accepting multiple files at once and even directories, if the browser supports this feature. Alternatively, users are able to click on the drop zone and a traditional file selection dialogue is opened. By default, new files are added to the current directory of the submission. However, the path for new files is customisable by clicking the Edit button and entering a new path.

File uploads are handled via AJAX, allowing Sapphire to show the upload progress and, after completion, the upload status. Users might drop many files at once into the drop zone. The implementation of the upload script ensures that no more than 5 files are uploaded in parallel. The limit of 5 parallel uploads was chosen to ensure fast upload speeds, without demanding too many resources from the server.

Submissions usually consist of many files, situated in different sub-directories. Since, at the time of writing, only Google Chrome [Google 2019] supports dragging and dropping folders into web forms, Sapphire implements special handling of ZIP archives. Sapphire automatically schedules an extraction job in the background, once a ZIP archive is received.

Upon receiving an upload, Sapphire validates the size of the whole submission and rejects the file if it would exceed the limit set by the lecturer. ZIP archives are treated differently in the submission size check. Instead of checking the compressed size, Sapphire computes the extracted size of the ZIP file and uses the extracted size as the basis the submission size check.

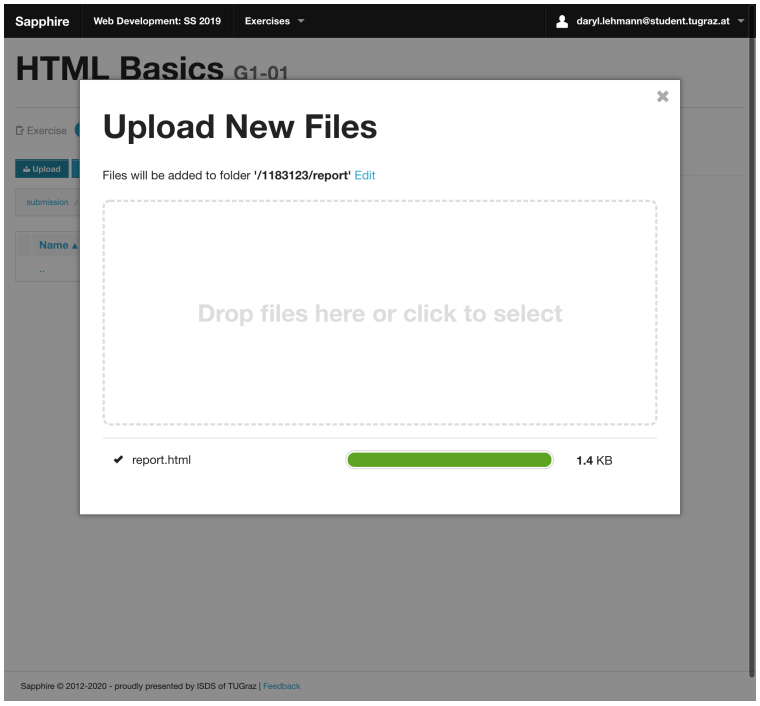

Figure 7.4: The Upload New Files Modal is used to upload files to a submission. This interface supports both dragging and dropping files from the desktop, as well as opening a traditional file selection dialogue. The upload status and progress of each file is shown in a list below the file drop zone.

Once the submission size check is passed, Sapphire stores the uploaded files in a dedicated directory and adds an item to the event feed indicating the upload of a new file. Creating a new entry in the event feed for each new upload would quickly clutter the event feed. Therefore, Sapphire concatenates upload events into a single event over the course of 30 minutes between uploads.

7.2.5 Moving or Renaming Files

Although moving or renaming files is a common task when using a file browser, Sapphire currently does not support moving or renaming files. Instead of renaming files, users are currently required to remove and reupload them. However, the developers intend to implement this feature in a future release, as described in Section 11.1.3.

Chapter 8

Grading

Much thought and design effort was invested to provide Sapphire with an integrated and optimised grading experience for both tutors and students alike. The grading interface evolved over time, from a tabular spreadsheet look-alike to a slimline responsive interface. Lecturers are able to interactively adjust the grading scale used to map points to grades. Students are notified once the preliminary results of individual exercises become available and can access their own detailed results.

8.1 Grading Submissions

The grading interfaces are one of the key features of Sapphire. The main grading UI has been redesigned three times since the early versions of Sapphire. The interface evolved from a spreadsheet look-alike to an integrated responsive UI. A secondary bulk grading interface is provided for entering the results of written examinations. The list of submissions is another important aspect of the grading interface providing vital information about the grading progress.

8.1.1 Grading Table

Once a submission is uploaded to Sapphire, a staff member is required to grade the submission. In the early stages of Sapphire, the developers implemented a fully functional Excel spreadsheet clone called the Grading Table, which provided a table of checkboxes for each rating and student (or student group) in a tutorial group. This is shown in Figure 8.1. Additionally, staff members were able to transpose the Grading Table if desired. These options were then persisted in the database and used by every subsequent page request of the grading table.

While this approach worked for a small number of students, it proved to be impractical for larger numbers of 70 or more students per tutorial group, since both the application server and the browser struggled with performance issues. On the server side, performance issues were fixed by exercising a combination of Rails' internal caching methods and some cleverly designed AJAX scripts. Improving issues on the browser side proved to be a harder task, mainly due to the sheer amount of data which needs to be presented. In the end, the developers settled on an underperforming grading page, which was optimised as far as possible.

The poor performance of the Grading Table view led the developers to completely rethink the whole grading process. When using the Grading Table, the process boiled down to the following steps. First, the evaluator has to download and open a student's (or student group's) submission. Then, the whole grading table, containing all evaluations for all students (or student groups) of a specific tutorial group, needs to be opened. Next, the column corresponding to a particular student or student group needs to be located. After these initial steps, the evaluator is able to grade a student's submission using the provided ratings, but has to make sure to stay in the correct column and not to tamper with other students' ratings.

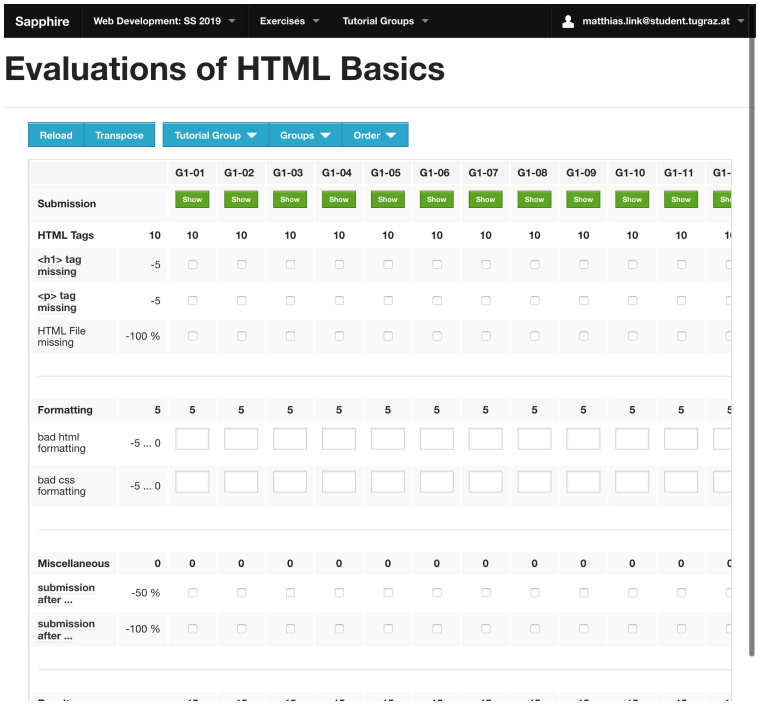

Figure 8.1: The Grading Table view, based directly on the previously used Excel spreadsheets. Each row represents a rating and each column represents a student group or student.

While this approach was used for many years, it proved to be error prone, since mixing up columns and submissions was very likely. Therefore, the developers took a radical approach not to use a complete table of ratings, but instead to show the submission alongside its associated files and ratings in a single page. This trail of thought resulted in the development of another grading front end, the Single Evaluation view.

8.1.2 Single Evaluation View

The Single Evaluation view, shown in Figure 8.2, aims to improve the grading process as a whole. It consists of two columns. The left-hand column contains inline rendered versions of submitted files (SubmissionAssets) and the right-hand column provides a complete list of ratings for the associated exercise. Both columns are individually scrollable, which enables a tutor to stay focused on specific parts of a submission, without loosing the context of the corresponding ratings.

Sapphire provides inline rendering for a wide variety of file types, including plain text files, images, and Portable Document Format (PDF) files. In addition, Sapphire provides syntax highlighting for many inline-rendered plain text file types, such as HTML, CSS, and emails, based on their MIME type. Sapphire provides hyperlinks to the raw versions for all inline rendered files. Doing so not only provides the convenience of simply clicking a link, but also prevents the tutor from making any mistakes when searching for the corresponding file in another interface.

This approach tackles many of the problems described with the original Grading Table. First of all, it greatly reduces the load on the application server as well as on the browser, as there is no need to lay out and render a large table. This improvement is especially noticeable when using mobile or tablet devices. Grading submissions on those devices was previously nearly impossible, due to a lack of access to the FS, as well as the lack of performance of the device itself. Providing inline files not only eliminates the possibility of mixing up submissions and students, but enables grading on devices which do not allow complete FS access, such as iPads and iPhones. Moreover, Sapphire adds links to the raw version of a file, in case a tutor still wants to grab the file and view it in a program of choice.

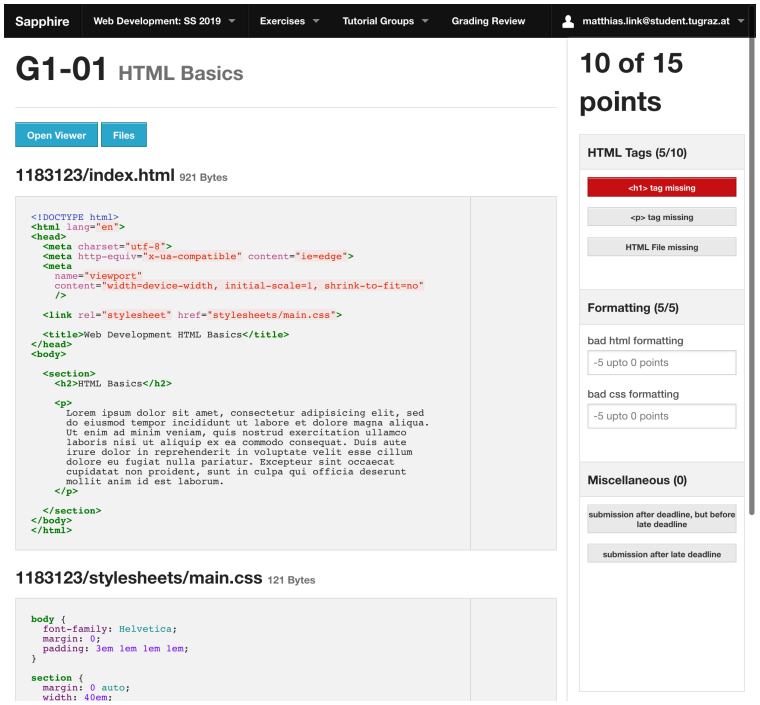

Figure 8.2: The first iteration of the Single Evaluation view. Submitted files are presented as a list and text-based files are syntax-highlighted. The right-hand panel contains the ratings used to grade the submitted files.

While syntax highlighting takes some time, Sapphire is still able to deliver quick responses to the client. This is done by caching the HTML chunk representing an inlined file, and reading from the disk instead of actually performing syntax highlighting over and over again.

Furthermore, Sapphire adds “Previous" and “Next” buttons to the page, which allow tutors to navigate through submissions made by their other students or student groups to the same exercise. At first glance, this might not look like a huge benefit to the grading process, but it prevents the tutor from having to go back to the list of all submissions and look for the next one to grade.

8.1.3 Submission Viewers

Some exercises, such as INM Exercise 4 [Andrews 2014a], require the tutor not only to look at the source file of a submission, but also to examine the live version as if it were hosted on a web server. Therefore, the concept of Submission Viewers was introduced. These are small pieces of software which enable Sapphire to display a submission in different ways and environments, such as emulating a basic web server or applying alternative submitted CSS files within a predefined skeleton. Section 5.6.1 provides further details on this matter.

8.1.4 Improving the Single Evaluation View

The Single Evaluation View proved to be a good interface for submissions containing only a few files. The inline versions of the assets were presented to the tutors in a plain-text manner without additional formatting other than syntax highlighting. This way of presenting files is optimal for grading the technical aspects of submitted files, such as the format of the source code or correct usage of HTML tags.

However, tutors reported that the interface was cumbersome to use for submissions containing entire reports. For report-based exercises like HCI Exercise 1 [Andrews 2019a], the focus of the grading process often resides on the text of the report, rather than the underlying technical details. Tutors were extensively

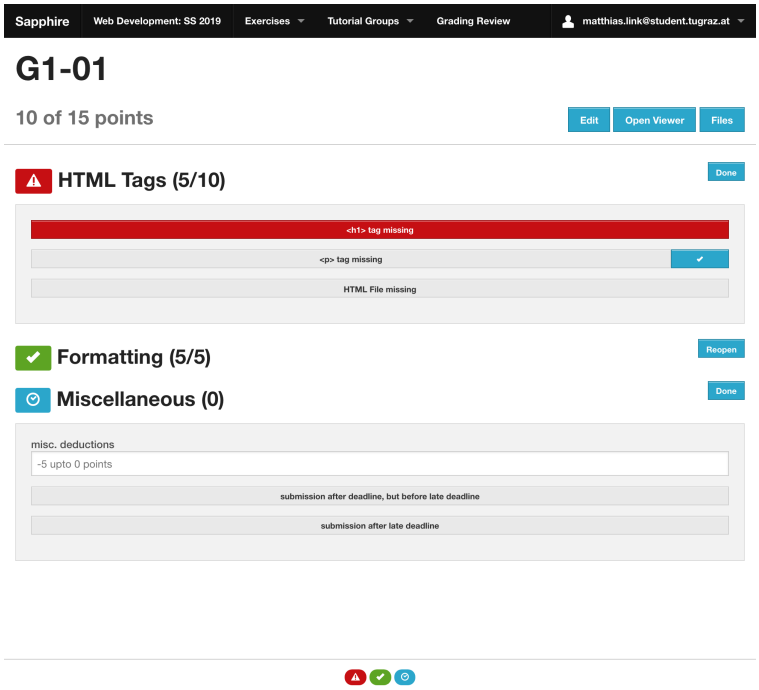

Figure 8.3: The latest iteration of the Single Evaluation view no longer displays a dedicated list of submitted files (assets). The first rating group shows a red warning sign, indicating that changes to the rating group were made after it was last evaluated. After reviewing the changes, tutors click the checkmark button and the warning status is resolved. The second rating group is marked as finished and is collapsed to take up less amount of screen space. The third rating group is not yet marked finished and is denoted with a clock sign. The status of all rating groups is displayed at the bottom of the page.

using the submission viewer feature of Sapphire, while mostly ignoring the plain-text versions of the submitted files.

This observation, along with the discontinuation of the INM course, lead to further streamlining of the grading interface, as shown in Figure 8.3. The Single Evaluation View was simplified by removing the list of inline submission assets. The resulting single column layout was further improved by increasing the font and button sizes to make use of the newly acquired screen space.

During the grading process, tutors follow different strategies. While some tutors prefer grading one submission after the other, other tutors prefer grading all submissions to an exercise in parallel on a per rating group basis. The former strategy is well supported by Sapphire, since it presents the whole set of rating groups in one list. Tutors using the latter approach reported that they were spending too much time scrolling to the right section of the rating list to continue grading the next submission.

The developers of Sapphire tackled this problem by allowing tutors to mark specific rating groups as completed. Completed rating groups are collapsed, taking up only a fraction of the vertical screen space compared to their expanded form. Even though this feature reduces the time tutors spend regaining context, it does not completely solve the problem, especially for submissions with many rating groups. Therefore, an additional progress indicator was introduced at the bottom of the page. The progress indicator provides a quick overview of the grading progress as well as a quick way for tutors to navigate to a desired rating group. The combination of these new features resulted in tutors reporting the problem to be resolved.

While redesigning the Single Evaluation view, another long-standing issue was tackled. Theoretically, all ratings are fixed by the lecturer before tutors start grading submissions. In practice, however, this only

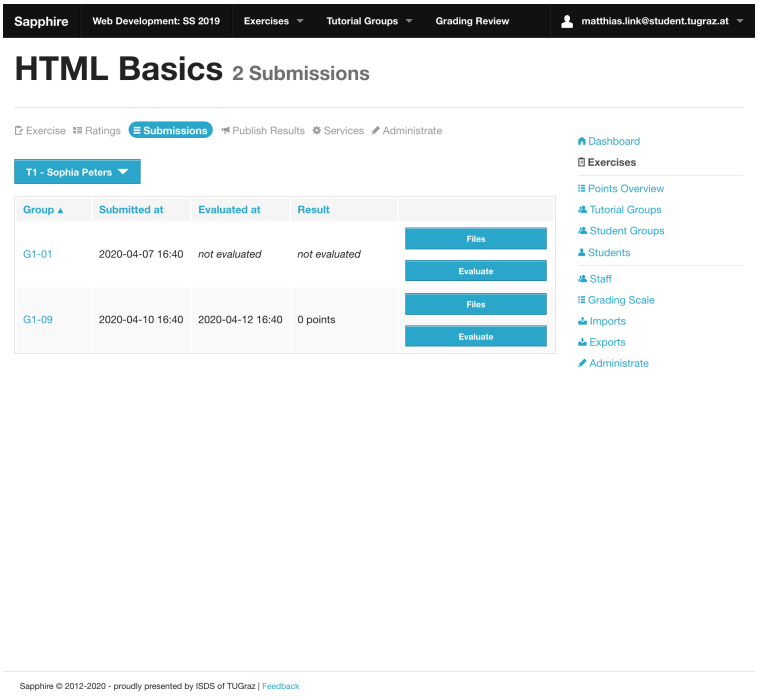

Figure 8.4: The Submissions Table view of an exercise, as seen by a staff member.

partially holds true. During the grading process, tutors might encounter previously unspecified problems with a submission. The lecturer then needs to either alter existing ratings or create new ratings in order to account for the newly discovered problem. By the time a lecturer incorporates such changes into the ratings of an exercise, several tutors might have already completed grading parts of the submissions. At this time, the need arises for tutors to revisit already graded submissions.

Previously, tutors had to study the event feed of a term to see which ratings had been altered and then manually revisit the corresponding submissions. With the redesign of the Single Evaluation view, the concept of outdated evaluations was introduced. Sapphire automatically marks the evaluations associated with a changed rating as "needing review". This mark is then visualised in the list of submissions, as well as in the Single Evaluation View. Marking submissions in this way allows tutors to quickly identify and review the grading of ratings, while requiring minimal amounts of manual checking.

8.1.5 Submissions List

In Sapphire, the Submissions Table is used to provide a list of submissions to an exercise, as shown in Figure 8.4. The table of submissions is the main part of the view. The submissions of the tutorial group associated with the staff member are shown by default. The dropdown selector above the table of submissions allows the submissions of a different tutorial group to be selected.

The Submissions Table lists the student or student group, submission date, evaluation date, and the result. A column containing the name and matriculation number of a student is added for individual exercises. Sapphire provides links to the Submission Tree view and the Single Evaluation view for each submission.

If multiple attempts are enabled, Sapphire includes a column for the exercise attempt. In case of multiple attempts, Sapphire uses the most recent submission as the basis for the points received for an exercise. Previous submissions are considered to be outdated and are greyed out in the submission list.

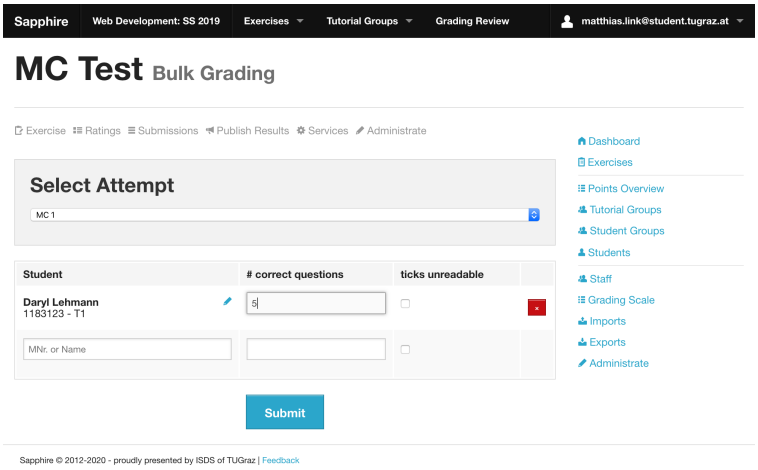

Figure 8.5: The Bulk Grading UI is used for grading multiple students at once.

8.1.6 Bulk Grading

Multiple choice tests, such as the HCI MC Test [Andrews 2019b], are taken by participating students in the form of a traditional written exam. For each participant, a submission needs to be created and the results need to be entered by tutors. This task proved to be time-consuming and error-prone with the UIs of Sapphire. Instead, an admin would create the submissions and import the corresponding results from the command line, based on CSV files created by the tutors.

The Bulk Grading UI was created to streamline the task of entering the results of multiple choice tests, as shown in Figure 8.5. The form is based on a table-like interface, similar to a spreadsheet application. Each row corresponds to a student. In the first column, a search field is provided. Subsequent columns are comprised of a subset of ratings, based on the Show during bulk grading option.

Tutors search for students by entering either the name or matriculation number and select the corresponding result. Once a student has been chosen, Sapphire automatically appends another empty line at the bottom of the table. Next, the number of correct answers is entered. This process is repeated until all results have been entered. The interface is fully usable with only the keyboard, to further increase the speed of entering results.

8.2 Adjusting the Grading Scale

Sapphire provides a sophisticated editor for adjusting the grading scale, which is used to map points to grades, as shown in Figure 8.6. The Grading Scale Editor consists of three parts. The Grade Distribution Table showing the grade distribution regardless of current changes to the grading scale per tutorial group, a configuration panel, and an interactive Grade Distribution Chart, for adjusting the grading scale.

The Grade Distribution Table displays an overview of grades per tutorial group. The table provides insight into which tutorial group performs best and how the grades are distributed amongst the different tutorial groups. The overall number and percentage of students receiving a specific grade is shown at the end of each table row. The second to last row of the table shows the sum of students receiving a grade in each

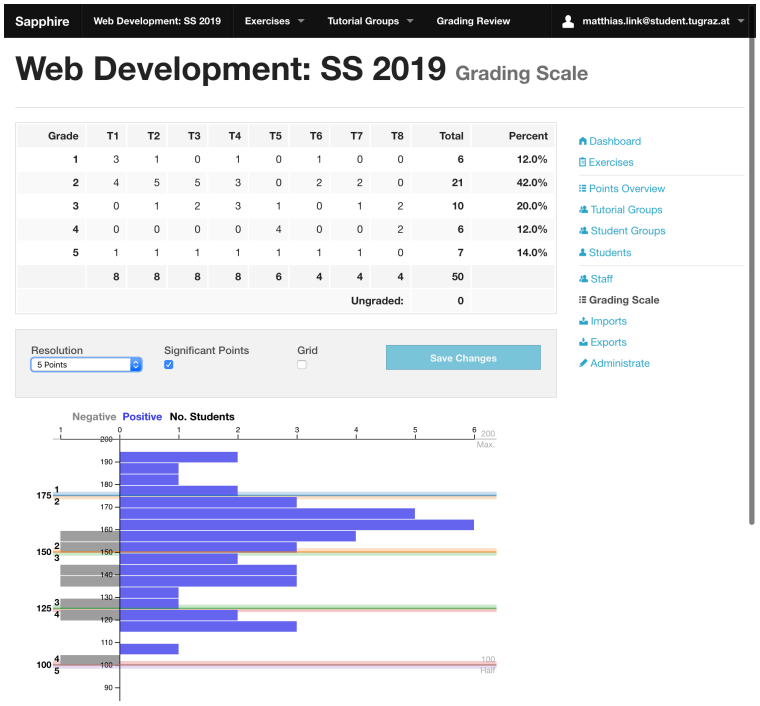

Figure 8.6: The Grading Scale Editor is used to adjust the mapping of points to grades. At the top, a table shows the distribution of grades per tutorial group. Underneath, a configuration panel is followed by an interactive distribution chart, which is used to adjust the grading scale.

tutorial group. The last row shows the number of students who did not participate in any exercise and therefore remain ungraded.

The basis for the Grade Distribution Chart is a vertically-aligned bar chart implemented with Scalable Vector Graphics (SVG). The points are plotted on the vertical axis and the number of students are plotted on the horizontal axis. The blue rightward bars show the number of students having achieved a specific number of points, who are eligible for a positive grade. The gray leftward bars show the number of students having achieved a specific number of points, who are not eligible for a positive grade, possibly because they failed an exam.

The range of grades is adjusted interactively by either dragging and dropping the boundary markers or by double-clicking on a boundary and entering the number of points to be used as a grade boundary. The points entered as a grade boundary are inclusive in the upward direction. Students having received equal or more points will receive the grade above the boundary and students having received fewer points will receive the grade below the boundary.

The configuration panel beneath the Grade Distribution Table controls the appearance of the Grade Distribution Chart. The thickness of bar is controlled via the resolution option. By default, the bars are plotted on a per-point basis, resulting in a very detailed chart with fine bars. Increasing the chart resolution results in the bars representing a larger range of points, which leads to a more general representation of the points distribution. Users are able to toggle the visibility of special lines indicating the number of half and maximum points via the Significant Points option.

The height of the chart is based on the maximum number of points. It is possible for the chart to have a height larger than the available screen height. Scrolling down the chart results in the x axis not being visible, making it difficult for the user to identify the number of students of a bar further down the chart. This problem is tackled by two different techniques: The Grid option displays thin vertical lines ranging from the top to the bottom of the chart at regular intervals. Additionally, a special cursor line is shown

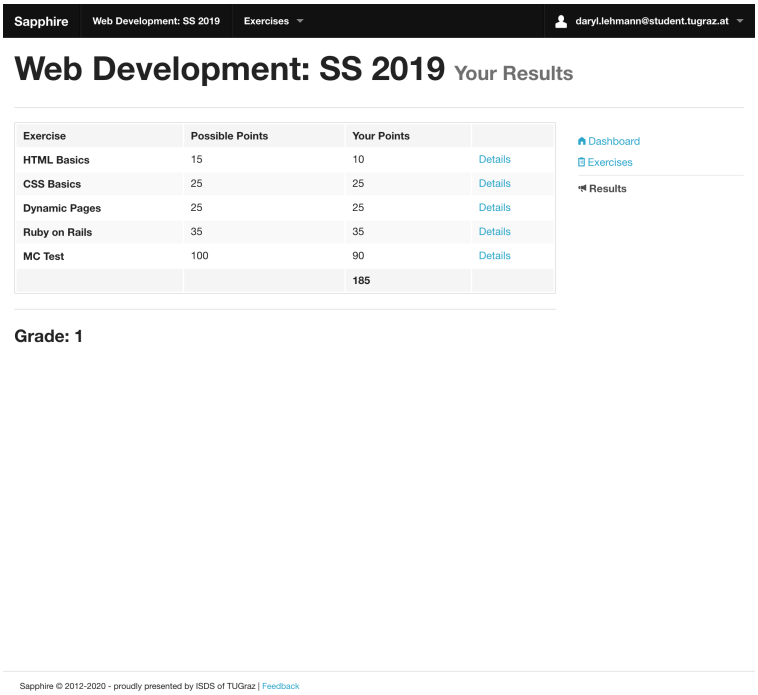

Figure 8.7: Overview of preliminary results as seen by a student.

at the mouse pointer position above the chart. The cursor line shows the number of points at the current location and the range of points and number of students represented by the bars below the cursor.

8.3 Publishing Preliminary Results

Once the grading for all submissions of an exercise has been finished, the lecturer of the term can publish those results, either by publishing preliminary results for each tutorial group individually, or for all tutorial groups at once. A student then receives an email inicating that the results of an exercise have been published, alongside a link to those results in Sapphire. There, the student is presented with an overview of all available results, which can be seen in Figure 8.7. The student is then able to review the detailed results on a per-exercise basis.

Such a review page can be structured in two different ways. The first consists of two columns and is a non-editable version of the Single Evaluation View described in Section 8.1.2. The first column contains inline rendered versions of submitted files and the second column contains a list of applicable ratings. The second way of presenting a submission is a single-column view of applicable ratings, which is used for submissions without any associated file, such as the Multiple Choice (MC) Test of HCI [Andrews 2019b].

Instead of showing a complete list of all ratings to the students, Sapphire presents them with a list of only these ratings which resulted in a points deduction or granting of bonus points. Doing so has two benefits. First, it is easy for a student to see to which parts of a submission had ratings applied, without having to scroll through a complete list of sometimes more than 100 ratings. Second, not all available ratings are disclosed to individual students at once. This should prevent students of subsequent terms optimising their submissions simply to satisfy all the ratings, while not necessarily understanding the main concepts of an exercise.

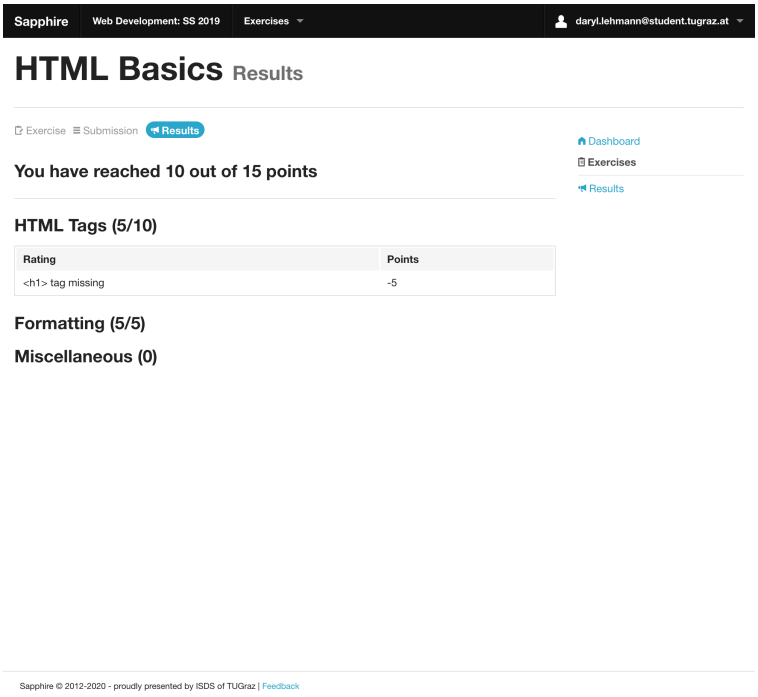

Figure 8.8: Preliminary results of an exercise as seen by a student.

8.4 Points Overview

Gaining an overview over the grades and distributions is important for both lecturers and tutors during the course of a term. Sapphire provides a dedicated Points Overview consisting of three parts, shown in Figure 8.9. A grade distribution table provides statistics of the number and percentages of students with specific grades. The exercise overview provides information about the minimum points and maximum points available for each exercise. Beneath these elements, a table provides detailed data on the performance of individual students for each tutorial group. Students are identified by matriculation number and the table shows their points for each exercise, total points, and overall grade.

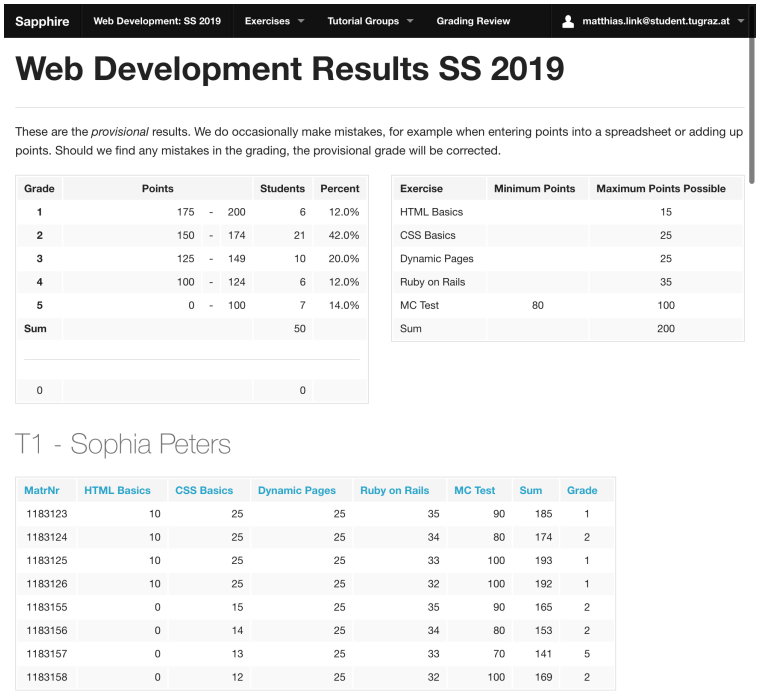

Figure 8.9: The Points Overview provides an aggregated points distribution table, an overview table of points per exercise, and a detailed table of points and grade per student for each tutorial group.

Chapter 9

Sapphire Exports

At the end of a term, lecturers are required to both enter the results into the TUGRAZonline system and archive the submissions along with associated grading information for inquiries at a later date. Sapphire assists lecturers with these tasks by providing powerful export capabilities. Since preparing an export requires a certain amount of time, background jobs are used to provide this functionality. Once an export is finished, Sapphire automatically notifies the lecturer via email.

Every exporter can have a unique set of options, which need to be stored in the database as well. Since SQL requires developers to provide a schema beforehand, these options cannot simply be stored in plain table columns without leaving several blank columns. Every time the set of options of an exporter is changed, the database schema has to be adapted as well. Therefore, Sapphire internally stores these options internally in a hash which is serialised to one single column of type text using JavaScript Object Notation (JSON).

One of the most important parts of Sapphire is to deal with courses encompassing hundreds of students in a term. Exporting data in different formats may require extensive processing and generate very large file sizes. Therefore traditional exporting techniques, such as requesting an export and streaming the data through the same HTTP request cannot be applied. This request might timeout before the first bits of data can be transfered to the user. Additionally, exporters should be extensible and capable of providing the means of creating multiple files at once.

Even though exports might take many seconds or even minutes in order to be completed, interactivity should still be provided to the user. Nielsen [1993] shows that a maximum response time of under 100 milliseconds is needed for the user to remain unnoticed, one second to keep the user's trail of thoughts and a maximum of ten seconds to keep the user's attention. This is the reason why Sapphire implements data exports using background jobs and workers.

When a user requests an export, Sapphire internally creates a record in the database containing the information needed to create the export. In addition, a job is submitted to Sidekiq [Sidekiq 2019] which is used to perform background tasks outside a HTTP context. Sidekiq's background process then picks up the new job and starts processing the export. Once the export has finished processing, or if an error was raised during processing, Sapphire notifies all lecturers and admins of the related term via email about the updated status, and provides a download link where applicable.

In addition to the notifications sent via email, a user can look at the exporter's status via the web interface. Sapphire provides a simple but powerful Representational State Transfer (RESTful) interface, allowing the user to create new exports and to update and delete existing ones. Existing exports can be downloaded as well, in case the download link provided in the notification email has been lost.

Each exporter has a unique set of options that need to be customised by the user. Sapphire implements two different approaches for a developer to integrate a new exporter form. The first approach is to use

the generalised form generation routine, which simply renders form fields based on the options provided by the exporter. Although the general way of generating forms is very simple to use, it is very limited in terms of layouting the form. A developer might want to provide hints, additional information, and documentation alongside the form fields, which is why Sapphire provides another way of creating more advanced exporter forms. Just before the form fields are rendered, Sapphire tries to lookup a partial, which follows the same naming scheme as the exporter. When Sapphire is able to detect such a partial, it renders this one instead of the generic form fields. A form field partial can be formatted like any other form in Sapphire using simple_form's form generator. It is recommended that developers use the generic form generator when an exporter still is in development. As soon as the exporter is fully implemented, the form should be adapted to fit Sapphire's interface and match its look and feel.

Exports usually consist of more than one file in a well-defined directory structure. In HTTP, only one file can be downloaded per request, since a HTTP response message can only carry one message body [Fielding and Reschke 2014]. To work around this, Sapphire produces a single ZIP file containing the entire folder structure created by an exporter. A user then can conveniently download a single archive file, which can later be unzipped to restore the original directory structure.

In order to perform work in the background, a so-called daemon process needs to be started. A daemon is a program, which is started once and continues running until somebody tells it to stop. The Sidekiq daemon reacts to changes to the job queue and starts executing new jobs as they are enqueued. In contrast to other records in the Sapphire database, Sidekiq jobs are stored in a separate Redis database, which is a KV Store. Redis is optimised to handle large numbers of clients in parallel and provides powerful interfaces for handling concurrent requests to the KV Store.

While other solutions exist such as Delayed Job [Collective Idea 2015], which would not require additional databases, the team has chosen to use Sidekiq instead. In contrast to other solutions, Sidekiq uses threads instead of separate processes for processing jobs. This results in much lower memory footprint and as a result enables scaling more easily, as described by Perham [2012].

Two exporter modules are currently implemented. The Submission Exporter exports the original submissions of an entire term in the form of a ZIP file. The Grading Exporter exports the grading for an entire term in form of several Excel spreadsheets, one per tutorial group.

9.1 Submission Exporter

The submission exporter exports all of the submissions of a specified term. As described in Section 5.2, tutors were previously required to burn all submissions of students in their tutorial group, together with their filled out spreadsheet onto a CD or DVD. For consistency the submissions were structured with a very specific naming and directory schema.

Sapphire requires a user, who is about to start a submission export, to specify the schema the files need to be placed in. Each individual file path has to be unique, which is why one cannot simply specify a static path directory path on where to place the files. Sapphire provides a powerful path substitution algorithm, which allows to specify the required file path in great detail, while still providing a great amount of flexibility. Since the naming schema is reused in consecutive terms, Sapphire pre-populates the required fields with default values.

For path generation, Sapphire currently distinguishes between solitary and group exercises, for each of which an individual format can be specified. This works by entering a path containing placeholders, indicated by a starting percent sign followed by an identifier in curly brackets. For example "solitary_exercises/%{matriculation_number}/%{exercise}". Table 9.1 shows the available placeholders for path generation. Additionally, Sapphire provides the possibility of specifying a "base path", which will be prepended to the ones mentioned above. The path substitution algorithm is applied to the base path as well.

Placeholder	Description
%{av_grade}	Inserts the average grade for the student group
%{course}	Inserts the course's title, e.g. hci
%{exercise}	Inserts the exercise title, e.g. ex1-he-plan
%{matriculation_number}	Inserts the student's matriculation number, e.g. 1234123
%{student_group}	Inserts the student group identifier, e.g. g1-01
%{term}	Inserts the term's title, e.g. ss2014
%{tutorial_group}	Inserts the tutorial group identifier and the name of its tutors, e.g. t1-matthias

Table 9.1: Placeholders supported by the Submission Exporter.

9.2 Grading Exporter

The Grading Exporter exports the grading for an entire term in the form of an Excel spreadsheet. The developers strived to being able to use Sapphire as a drop-in replacement for the original workflow, without implications on the final products, and allowing both to coexist and both being used during the same term. As a result, Sapphire creates spreadsheets as close to the original as possible. This not only includes generating separate sheets for each exercise, grading overviews, and student lists but also keeping the original format and colour-coding.

The calculation of points has to be consistent throughout Sapphire. This is achieved by recalculating the points only when the evaluation changes, and caching the current results. The grading exporter only needs to access the cached values from the database, without actually calculating the resulting points itself.

The purpose of generating spreadsheets is to archive the resulting grading. For simplicity, Sapphire exports only the calculated values to spreadsheet cells, rather than any formulae needed to dynamically recalculate such values. Doing so has several benefits: First of all, the Grading Exporter is able to access cached points mentioned above, which is far quicker than recalculating them over and over again. Furthermore, the exporter does not need to understand, how points are actually calculated and how to translate these calculations into Excel formulae. This benefits the intoduction of new rating types as well, as only one part of Sapphire needs to be changed, the underlying grading algorithm. Therefore the exporter's complexity is largely reduced and remains easily maintainable.

The exporter is very flexible in itself, as it lets the user choose which worksheets to include in the export. One can choose between the summary sheet, the exercise sheets, the student overview and the group overview. While, by default, all sheets are included in the export, a user can deselect those not needed. Figure 9.1 shows a generated spreadsheet filled in with dummy data.

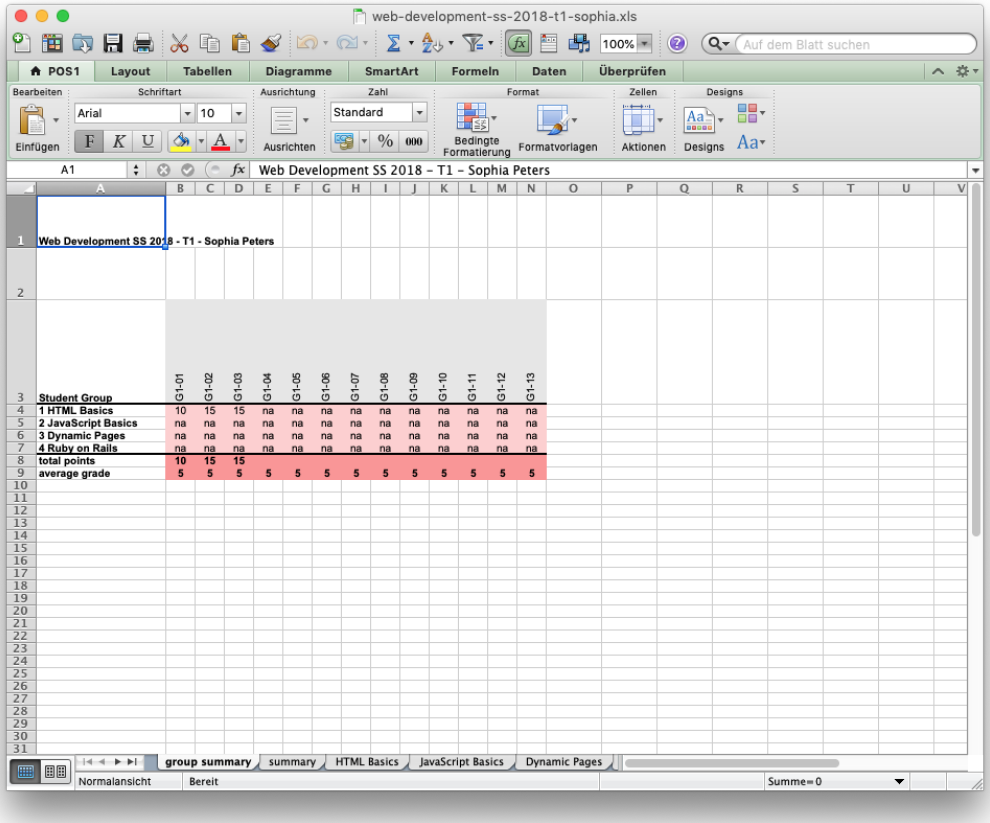

Figure 9.1: A spreadsheet exported via the Grading Exporter.

Chapter 10

Selected Details of the Implementation

Sapphire is a large project, consisting of numerous subsystems. While many subsystems are worth a closer look, this chapter focuses on three outstanding ones in terms of implementation and provided features. The Event System supports dynamic tracking of changes to individual records of a term. The Grading Review interface is used during the face-to-face grading review where students can ask questions about their grading. It provides fast access to a student's submissions and grades without disclosing the results of other students at the same time. The sortable tables interface uses JS to allow table rows to be sorted client-side.

10.1 Event System

The Event System is one of the latest additions to Sapphire. It provides a slick interface for all types of event occurring in the system, as shown in Figure 10.1. Events are persisted in the Database (DB), in order to provide a quick overview about the latest changes inside a term. Events are displayed in the

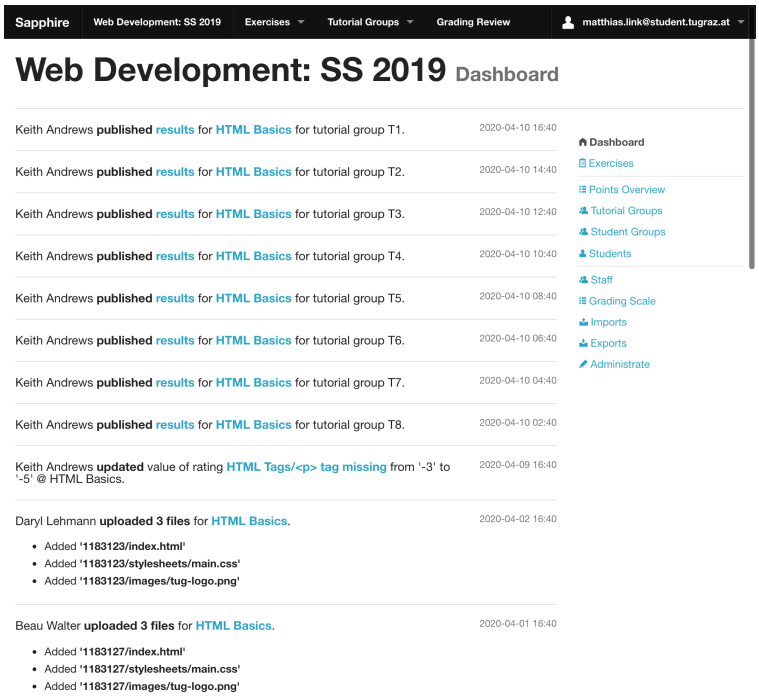

Figure 10.1: List of events displayed in a term's dashboard, as seen by a staff member.

term's dashboard. Some events are restricted, depending on the current user's access rights. While staff members are able to see all types of events, students are only allowed to see changes to their own submissions, as well as the changes to the publication status of grading.

10.1.1 Types of Events

Sapphire logs events including the creation, update and deletion of records, the publication and concealment of preliminary results, and changes to submissions, including initial uploads and the extraction of ZIP archives. Internally, Sapphire defines a dedicated event class for each type of action and subject record. Each event class tracks the time it was created and is associated with a specific term, the user who initiated the action, and the subject record.

10.1.2 Storage of Events

Since Sapphire supports many different types of events, storing those in the DB proves difficult. Each event has a unique set of additional attributes in order to keep track of individual changes. The most straightforward way would be to store each class of events in a separate DB table. This might sound reasonable at first, but proves to be disadvantageous when many different event types need to be sorted by their creation date, which is necessary to display a chronological list of events in the dashboard.

A solution was found with the help of Rails' Single Table Inheritance (STI). To avoid the need for additional DB columns to store each individual event's attributes, the attributes are serialised into a string-based format. The result of this process is then stored in a plain text column. While this approach allows developers to add new events quickly, it makes it difficult to filter records by attributes, as there are no separate columns that would support this behaviour. However, ordering and filtering by those attributes is currently not needed, as they are only needed to store additional context information in order to allow the event to be displayed properly.

10.1.3 Rendering Event Views

Since each class of events has a unique set of attributes, Sapphire needs to render specific HTML snippets for each type of event. This is implemented with the aid of partials. Each event class has a corresponding Partial defining how the event should be rendered. Each partial is cached by Rails, in order to improve the rendering performance of the event list and improve response times.

While it is usually sufficient to perform caching on a per event basis, there are use cases where an even finer approach needs to be taken. For example, when a student uploads files for a submission, the links in the corresponding event HTML snippet point to the submission page, in order to allow all group members to verify and update the submission in a quick way. In contrast, staff members will generally need to evaluate the submission, so the event links directly to the single evaluation page, which is much more appropriate for a staff member. Sapphire is capable of dynamically constructing cache keys for events based on additional dependencies, such as the user's role in a term. This approach allows Sapphire to maintain fast performance by caching the resulting HTML snippets, while maintaining the flexibility of rendering different versions depending on additional factors.

When visiting a term's dashboard, a user is presented with a list of events. Due to the sheer number of events, especially for staff members, it is not possible to directly render the whole list of events at once. Hence, a combination of AJAX and a corresponding Application Programming Interface (API) is used to provide a responsive interface. When a user visits the dashboard, an empty list is returned to the user, along with some JS. This script then issues another request to the server to fetch the event list. In addition, it listens for scroll events inside the browser. Once the user approaches the end of the page, and the API indicates there are more events available for the user, the JS automatically requests additional parts of the list, without any need for further user interaction. This method of interaction is also known as "infinite scrolling" [Ahuvia 2013].

10.2 Grading Review Interface

After provisional points and grades are published, many students have questions or queries. A grading review is a face-to-face meeting between a student and their tutor or lecturer to clarify any issues with or questions about the grading of their work and isn usually scheduled at the end of each term.

The grading review interface provides an easy-to-use interface for staff members to use during grading reviews. It enables a staff member to search for a particular student, provide a summary of the student's results, and drill down into detailed results per exercise.

10.2.1 Reasons to Create a Dedicated View

In the days of the Excel spreadsheet, doing grading reviews was somewhat tedious for staff members, since student's submissions were stored somewhere in the tutor's FS, and their results were contained in the Excel spreadsheet.

The spreadsheet contained all individual ratings along with their respective point reductions. As a result, students were able to gain deep insight into the internal grading process, which could then be disclosed to other students. This knowledge could then be exploited in subsequent terms.

Furthermore, students were able to see the results of other students whose columns in the spreadsheet were close to that of the student attending the grading review. Therefore, information considered private to individual students could be unintentionally disclosed to other students.

Last but not least, the problems mentioned in Section 8.1.1 were also present during the grading review. Columns in the large spreadsheet could easily be mixed up, and the grading for a different student be presented by mistake.

10.2.2 Workflow During a Grading Review

The Grading Review interface was created to mitigate the problems mentioned above. At the start of the grading review, students are asked to provide their student id, and their matriculation number or surname is entered into the search box, as shown in Figure 10.2. Sapphire returns a list of students, consisting of the student's full name, matriculation number, and tutorial group, as shown in Figure 10.3. This information is used to correctly identify the student, without disclosing any information about the grading of other students. The staff member then selects the appropriate record by pressing the Show button and is instantly able to provide feedback on the student's grading.

In the *Grading Review Detail* screen, Sapphire presents the staff member with a short overview of the exercises a student attended, as shown in Figure 10.4. This view is used by the tutor to gain a quick impression on the student's grading. Since time is often limited during a grading review, this view can be used as a basis for negotiation about which exercise submissions should be looked at in detail.

In addition to the summary, Sapphire renders a separate two-column view for each exercise. Similar to the Single Evaluation view presented in Section 8.1.2, this view contains inline versions of the submission's files as well as a list of ratings which were applied, as can be seen in Figure 10.5. The decision to list only applied ratings is a tradeoff between how much information is disclosed and how much remains private. While other methods of informing students about their results were considered, like requiring tutors to write a short summary about a student's mistakes, this method proved to be the most practical in terms of minimising staff workload.

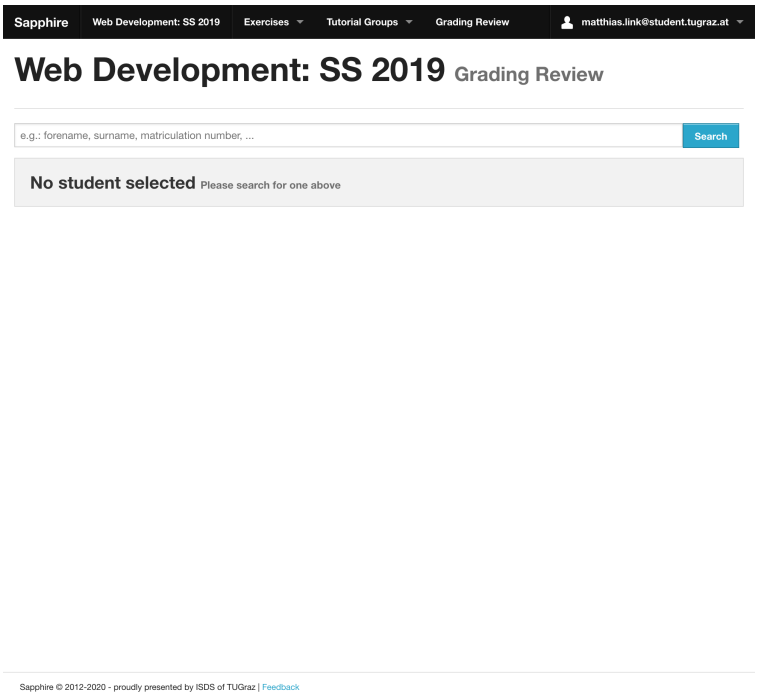

Figure 10.2: The Search Form provided by the Grading Review interface.

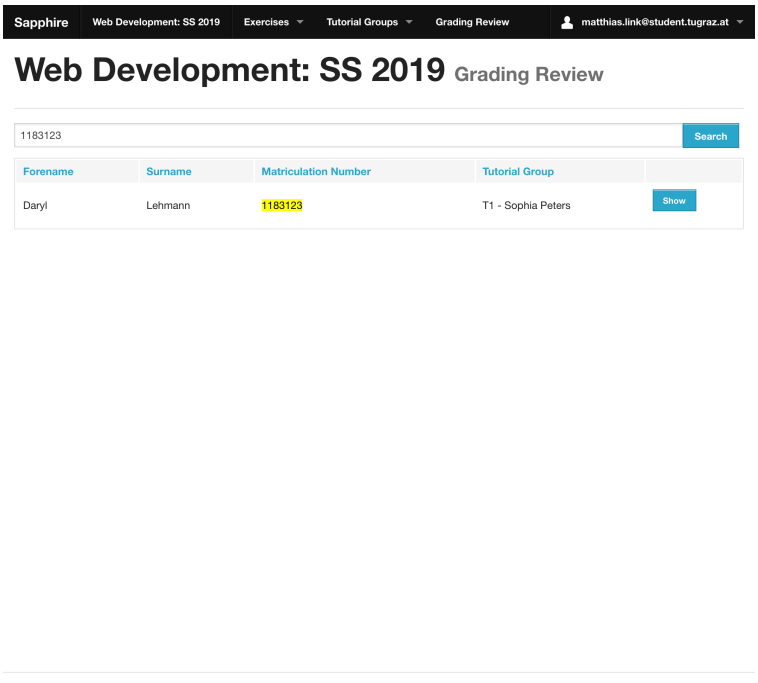

Figure 10.3: The Grading Review search results show matching students.

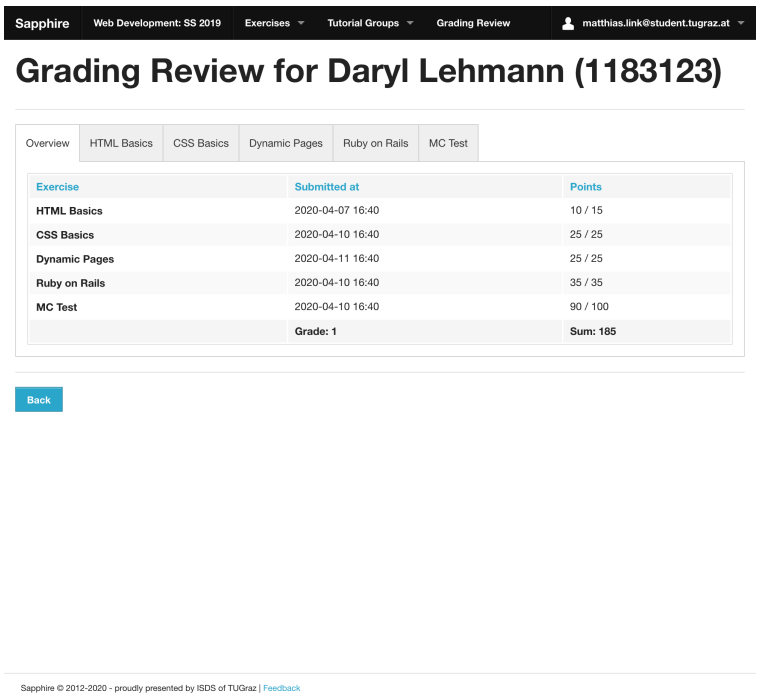

Figure 10.4: The Grading Review Overview tab showing an overview of a student's results for all exercises.

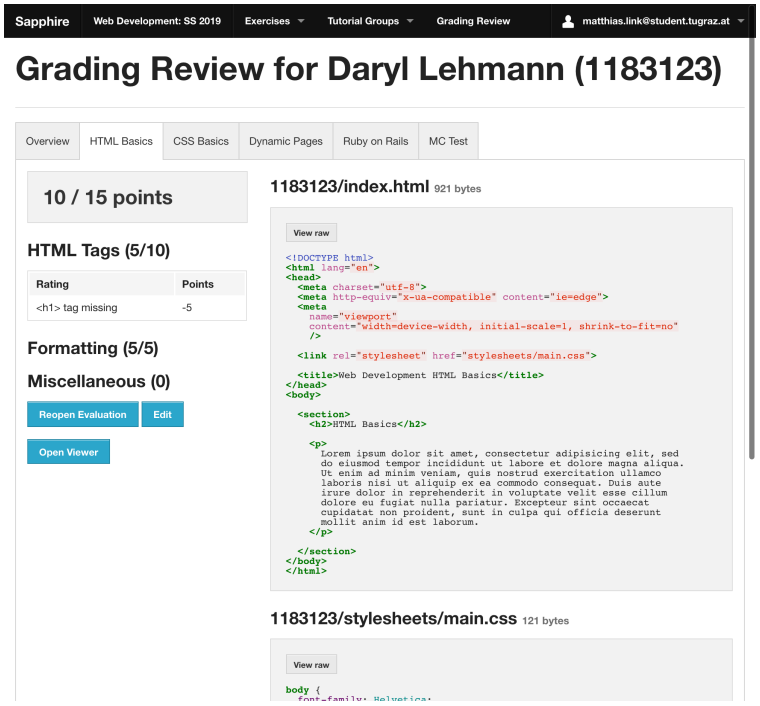

Figure 10.5: The Grading Review Submission tab showing a student's result for a particular exercise submission.

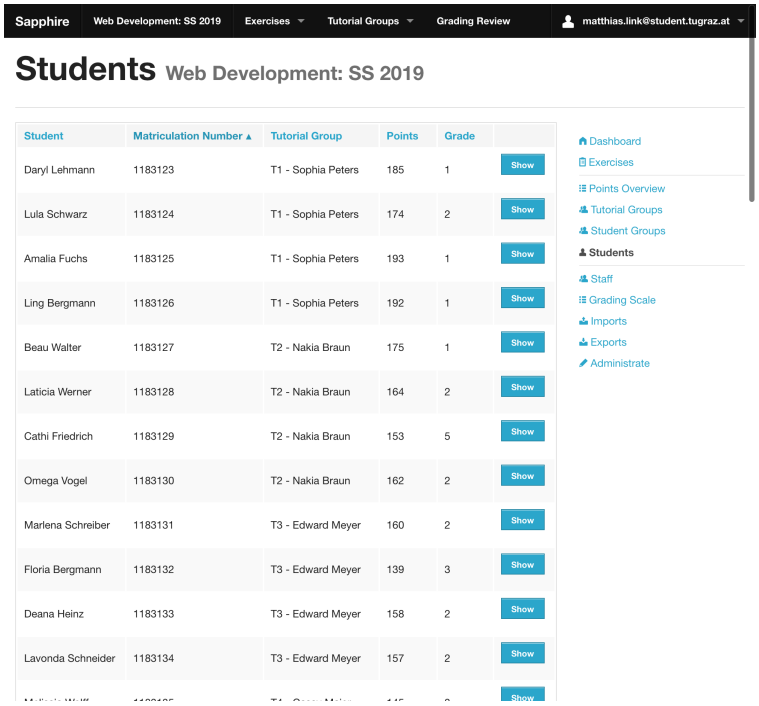

Figure 10.6: Reordering the Students table by Matriculation Number using client-side JS.

10.3 Sortable Tables

In traditional implementations of sortable tables, a server-side approach is used. When the table headers are generated, the application inserts specially crafted links into the web page. The links are based upon the current request parameters with additional parameters for changing the sort column and direction. A user is able to reorder the table according to a chosen column by clicking on the link in the table header. A request is sent to the server, which leads to the generation of a new page based on the updated parameters.

While this approach proved feasible for many years, it also suffers from numerous drawbacks. A very obvious one is the need for additional requests to the server, which might lead to poor performance. Large record sets and record sets comprising multiple database tables are expensive in terms of time needed to generate the table. On the server-side, an SQL query needs to be constructed and sent to the database. After the database finishes processing the query, the web application needs to read back the results and render an updated version of the table. During the rendering process, the links in the table headers also need to be updated. In addition, security concerns need to be taken into account. Since HTTP requests might contain malicious data, all parameters processed by the web application need to be validated and sanitised before being sent to the database. The resulting HTML code is not easily cacheable, since the parameters and columns change. Reusability is limited as well, since the parameters of sort links are dependent on the current request parameters.

Instead of sending requests to the server, Sapphire implements table reordering on the client side in JS, as shown in Figure 10.6. Before a page is displayed to the user, the JS searches for tables with a special class in the browser's Document Object Model (DOM). Whenever a matching table is found, link elements are injected into the headers of the table. In contrast to traditional links, the header link elements do not navigate the browser to another URL. Once a click is detected, JS catches the corresponding event and reorders the table based on the clicked column, without the need to send an additional request to the server.

By default, the JS sorts the table contents based on string comparisons. In some use-cases, this behaviour might result in unwanted results. Integer-based columns, for example, would be sorted incorrectly,

since values starting with “3” would be sorted after values starting with “10”. The developers solved this problem by specifying optional sort-value and sort-type attributes on the table cells. Additionally, a third attribute was introduced to always sort specific entries to the top of the table, regardless of the sort direction. These attributes support sophisticated reordering behaviours.

The JS-based solution elegantly solves many problems introduced by the traditional approach. Since no additional requests to the server are needed, the new table order is applied almost instantly. Additionally, there is no need to manage and validate the sort parameters in the URL and the server is capable of caching the table as soon as it is constructed the first time. Since the solution is based on CSS selectors, the Sortable Table feature is easily reusable on other pages without the need for additional code. Another benefit of performing the reordering process on the client side is the reduction of load on the server, increasing the overall performance of Sapphire.

However, the JS-based approach comes with some drawbacks of its own. Since click events are processed via JS, the browser does not update the URL bar at the top and it is not possible to share a link including the reordering steps applied by the user. Furthermore, the JS-based approach is only applicable for tables which fit on a single page. Once paging is introduced, the JS does not have access to the whole record set and is therefore not able to perform the correct ordering steps. Since the JS-based approach reorders elements in the DOM, additional care needs to be taken when used in combination with other JS functions attaching event listeners to the same DOM nodes.

As a result, the JS-based table sorting approach provides a quick and reusable solution for small to medium-sized tables. However, the JS-based implementation is not a one-fits-all solution. In Sapphire, almost all tables currently make use of the JS-based sorting behaviour. The simplicity of solely specifying a CSS class outweighs the drawbacks of this solution.

Chapter 11

Future Work

While significant effort has been invested to push the Sapphire project to its current status, there is still much room for improvement. This chapter describes upcoming features and improvements, which are either in the design phase or already in development.

11.1 Improvements to Existing Features

During the development process of Sapphire many design decisions were made. Some of these were based on rough predictions and estimates done without exactly knowing the impact on later developments. Consequently, some features in Sapphire are only partially complete and need additional work in order to optimally fit into the ecosystem.

11.1.1 Evaluation Process

Currently, submission evaluations are only able to represent two states: “submitted” and “evaluated”. The recent terms revealed that at least two more states are needed, “in evaluation”, and “confirmed”. Furthermore, the process of evaluating a submission should be integrated into Sapphire, by supporting this multi-stage process. Doing so should improve the way tutors are able to organise the evaluation process. As a consequence, it should be more clear to tutors, which evaluations are completely finished, which ones are work in progress, which ones are fresh, and which ones need to be revisited.

11.1.2 Searching for Ratings in the Single Evaluation View

The number of ratings tends to grow over the course of several consecutive terms. In recent terms, tutors expressed the need for searching for ratings during the grading process in the Single Evaluation view. Sapphire hides rating groups which have been marked as completed, so the search functionality of the browser is insufficient for this use-case and a dedicated interface needs to be implemented.

11.1.3 Drag-and-Drop in Submission Tree

The submission tree view currently does not support moving files from one folder to another folder via the UI. Students need to remove files from the submission, navigate to the destination folder, and upload the files once again. In order to improve this workflow, drag-and-drop-based interaction has been considered. Students enter a dedicated edit mode, which allows them to select the desired entries in the submission table. The selected files and directories are then moved to the destination folder via drag-and-drop.

11.1.4 Renaming Files in Submission Tree View

Similar to moving files, Sapphire currently does not support renaming files in the submission tree view. In order to rename files, students are required to delete the files from the submission and upload the files once again, with an updated file name. In future versions Sapphire should provide a UI for students to directly rename files through the web frontend.

11.1.5 Submissions Export per Tutorial Group

The Submissions Export feature allows lecturers to create a ZIP archive of all submissions of a term. Lecturers have requested the ability to create archives for a specific tutorial group. In a future version, Sapphire should implement a UI, which allows lecturers to specify which tutorial groups are included in a Submissions Export archive.

11.1.6 Remove UI Points

Sapphire automatically calculates the maximum points for each exercise by summing up the starting points of each rating group. This procedure is sufficient for deduction-based exercises. However, some rating groups are not based on deductions. For example, for examinations, the main rating group is configured to have zero starting points and points are awarded based on the number of correctly answered questions. Basing the maximum points on the sum of starting points results in the examination exercise having zero maximum points.

As a workaround, the concept of UI Points was introduced, which allows lecturers to override the maximum points shown in the UI. While this solution fixes the problem of wrongly shown points, there were several instances of wrongly entered UI Points in recent terms. In the future, the maximum points calculation needs to be improved, eliminating the need for UI Points.

11.1.7 Improve Automated Checkers

The automated checkers are currently embedded in the source code of Sapphire and are triggered via the command line. Lecturers are only able to specify an Automated Checker Identifier on a per-rating basis, without any additional configuration options. Since some requirements change over the course of consecutive terms, the automated checkers need to be adapted for new terms as well.

When changing the behaviour of an automated checker, potential inconsistencies are introduced, which might result in changing the results of already finished terms. In order to fix this problem, the automated checkers need to be refactored to provide dynamic configuration options, which are specified along with the Automated Checker Identifier in the ratings dialogue. The configuration options will then be duplicated during the term copy phase and will be configurable on a per-exercise basis for each term.

11.1.8 Publish Results per Exercise Attempt

It is currently possible for lecturers to publish results on an per-exercise basis. However, lecturers might allow students to attend an exercise more than once, for example retaking failed exams. There are currently two options for handling this case in Sapphire, each with different drawbacks: The first option is to publish the results once the results of first attempt are entered into Sapphire, which has the benefit of students receiving their results in a timely manner. However, since the results have already been published, Sapphire will not send notifications to students if the results of a consecutive term are entered into Sapphire. The second option is to delay the publication of results until the results of all attempts have been entered into Sapphire, which delays feedback for all students regardless of which exercise attempt the student has participated in.

In order to solve this problem, Sapphire should implement result publication on a per-attempt basis. Then, students would be able to receive timely feedback and receive additional notifications, once the results of a consecutive attempt are available. Additional care needs to be taken for students having attended only one of several attempts. Sapphire should only send result publication notifications to students having participated in this specific attempt.

11.1.9 Expose Accounts Management Feature

Sapphire administrators are currently able to create, update, and delete Sapphire accounts through a special Accounts UI. Lecturers are able to create accounts through the Student Import feature, as described in Section 6.4.1.

A limited version of the Accounts feature should be made available to lecturers, allowing lecturers to create new accounts, and update the name of student accounts participating in one of the lecturer's terms. However, lecturers should not be able to change an account's e-mail or password, since these attributes are used for authentication and might lead to security problems. The account removal feature needs additional consideration, since an account might be related to terms managed by another lecturer.

11.1.10 Renaming Certain Buttons and Concepts

Throughout the UI of Sapphire both terms "solitary exercise" and "individual exercise" are used to refer to an exercise attended by one student at a time. In a future version of Sapphire only the term "individual exercise" should be used throughout the UI to reduce the possibility of confusing the users of Sapphire.

Furthermore, the terms "administrate", "edit", and "update" are used interchangably to describe an editing action throughout the Sapphire UI. Due to similar reasons mentioned above, a future version of Sapphire should only use "edit" to describe an editing action.

Performing these changes might seem trivial at first. However, changing these terms in the UI may also lead to several additional changes throughout the test suite of Sapphire. As a result, changing these terms has been postponed to a later date in favor of developing new features for Sapphire.

11.2 New Features

Even though Sapphire has a rich set of features, some features are still to be implemented.

11.2.1 Commenting System

A commenting system is one of the features most requested by lecturers and tutors. The idea is to provide a simple interface for annotating ratings and submitted files. Additionally, there is the need for specific visibility levels, as a comment might either be addressed to a student or to another staff member.

The currently proposed solution aims for a "post-it" metaphor. The interface will provide a text field with one additional submit button. The visibility level will be represented through colours. Blue notes will only be visible to staff members, while green ones will be visible to every one. The developers will also try to add the capability for adding inline-comments, similar to GitHub [2015] for Merge Requests. Keeping the comments in the same context as the file itself should reduce friction when using the commenting system.

Additionally, Sapphire will provide the means to reply to a comment. During grading, tutors often reach a point where they are not sure whether the submission fulfills a specific rating or not. Therefore, they would like to post a question to the lecturer, who is responsible for making a decision. The preferred method of communicating this decision would be within Sapphire itself. While posting another comment

onto the same line would be a feasible solution, a problem of ordering and displaying those posts would arise. The simplest way to mitigate this is by using comment threads, which by default are displayed in chronological order.

11.2.2 Proposed Ratings

During the grading process, tutors sometimes find problems which are not covered by a rating at that point in time. The current workflow requires tutors to take notes, which are then discussed during staff meetings. The lecturer usually creates the missing ratings and the tutors need to reopen the evaluations of the concerned submissions.

In future, Sapphire could support proposing ratings directly from an integrated UI. The proposed ratings are then approved by the lecturer, who also specifies the corresponding deductions in the same process. The integrated solution will then automatically update the concerned submissions without requiring further interaction by the tutors. The integration of Proposed Ratings lessens the possibility for error, while at the same time streamlining the grading process.

11.2.3 Submission Sizes UI

The maintainers of Sapphire are currently monitoring the file sizes of each term via console scripts on the production server. Using scripts to monitor the file system size was originally only a temporary solution. However, this solution is still in use today. In future, Sapphire should implement a family of UIs for lecturers to monitor the file sizes directly from the web frontend per exercise and per term.

11.2.4 Submission Excerpts

Submissions often entail handing in one large HTML file containing a report, such as in Andrews [2019a]. An interesting proposal is to automatically extract configurable sub-sections of a report, so-called Submission Excerpts, with the possibility for further processing and filtering of the extracted parts, like converting HTML to plain text. Automated checks could then be chained to such excerpts, for example counting words, HTML validation, or plagiarism checking.

11.2.5 Plagiarism Checker

Another problem which arises when grading a mass course is that at some point students try to minimise their amount of work needed to pass the course. This includes working in groups for solitary submissions, using solutions from previous terms, and copy-and-pasting texts from the internet. While solutions exist for checking for plagiarism on the internet, checking inter-submission plagiarism is still somewhat tedious.

Currently, relevant sections of a submission are extracted after each exercise deadline by a system administrator via command-line scripts. The extracted sections are then manually uploaded to an external plagiarism checking service. Once the plagiarism checks are completed, tutors are required to review the results of the plagiarism check using an external service. If plagiarism is detected, tutors are required to activate the corresponding Plagiarism rating in the *Single Evaluation* interface. A simplified workflow would pass the extracted sections and any parameters to an external plagiarism checker automatically through an API, without the staff member needing to leave the Sapphire system. Plagiarism check results could then be inspected from within Sapphire too.

11.2.6 HTML Validator Integration

Another much anticipated feature is the integration of a HTML Validator. For several exercises, such as HCI's first exercise [Andrews 2019a], students are required to hand in their solutions as valid HTML files. Currently, tutors are required to download the HTML files and validate them with the help of existing

validators, such as the W3C Markup Validation Service [Oskoboiny et al. 2019]. Sapphire provides an interface for specifying automated checkers. However, the integration of existing online HTML validators is difficult, due to rate limitations and privacy concerns. Either a HTML validator could be implemented in plain Ruby, or a self-hosted HTML validation service could be set up and connected to Sapphire.

11.2.7 Configure Notification Emails

Sapphire sends out welcome notification emails, welcome back notifications emails, and result publications notification emails to students. The contents of notification emails is currently hard-coded and lecturers are not able to customise the text of emails sent to students. In a future version of Sapphire, lecturers should be able to customise email templates on a per-term basis through the UI. An important consideration during the implementation of this feature is the Copy elements from previous term feature, as described in Section 6.6. Since notification texts are unlikely to vary widely in consecutive terms, an additional option should be introduced to the New Term Modal, allowing lecturers to copy the notification texts as well.

Chapter 12

Concluding Remarks

This Master's thesis presented the frontend of the Sapphire project. The process of building a feature-rich yet simple online grading platform can take many years. Development of Sapphire started in 2012 and the first version was used in production in 2013. To this day, there are still issues to fix and new features to implement.

Many users, including lecturers, tutors, and students have commented on the benefits of using an integrated submission and grading system. One key benefit of Sapphire over other submission and grading systems is its usability. By providing simple interfaces, which work responsively on a variety of end user devices, the developers were able to create a sophisticated and intuitive grading and submission system. In the future, Sapphire will be further improved, and new developers have been introduced to the project. The author of this thesis is looking forward to these new developments and features.

Appendix A

Student Guide

Sapphire is an online course management and submission platform. This guide is intended for students of university courses which use the Sapphire system, and provides insight into the most important features and typical workflows.

A.1 Authentication

User accounts are used to manage access to the Sapphire course management system. Authentication is the means for signing into user accounts, requesting a password reset, editing the main attributes of an account, and changing the password.

A.1.1 Login

The first step in using Sapphire is to log into a Sapphire account, using the Login Form shown in Figure A.1. At the beginning of each term, the lecturer sends out an email, confirming registration to the course. Students are able then to sign in and take part in exercises of the current term.

First-time users of Sapphire must use the password reset feature to request a new password, as presented in Section A.1.2. The Password Reset Form is accessible through the Forgot your password? link beneath the Login Form. Previous users of Sapphire are able to reuse their credentials from a former term. Sapphire uses a role-based authentication system on a per-term basis. It is therefore possible to use a single Sapphire account for multiple terms.

A.1.2 Forgot Your Password

The Forgot Your Password allows users to reset their Sapphire account password. The process of resetting a user password is started by entering the email address of the account into the Password Reset Form and clicking the Reset button, as shown in Figure A.2. If the account is present in the database, Sapphire sends an email containing a password reset link. The user is allowed to choose a new password by visiting the included link, as shown in Figure A.3.

A.1.3 Changing Passwords

In Sapphire, passwords are changed on the user's profile page, which is accessible through the Edit Account link in the navigation bar. The user is presented with read-only meta-data as well as a form for changing the password, as shown in Figure A.4.

In order to change the password, the current password needs to be typed in first. Then, a new password is entered into the Password field, followed by the same password in the Password confirmation field. The

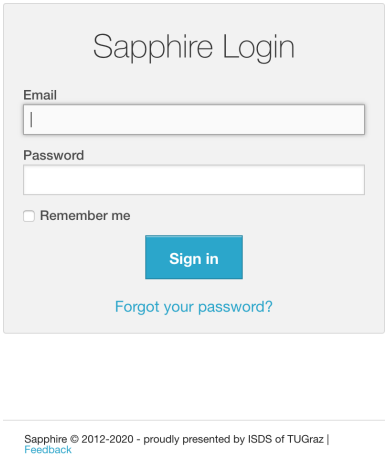

Figure A.1: The Sapphire Login Form.

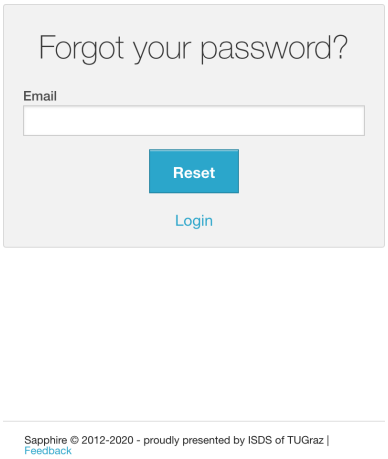

Figure A.2: The Forgot Your Password Form allows users to request a password reset link.

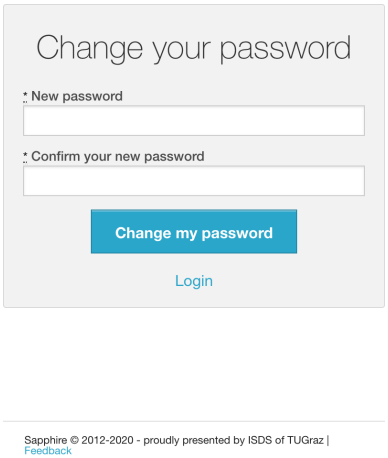

Figure A.3: The Change Your Password Form allows users to change their password after requesting a password reset link.

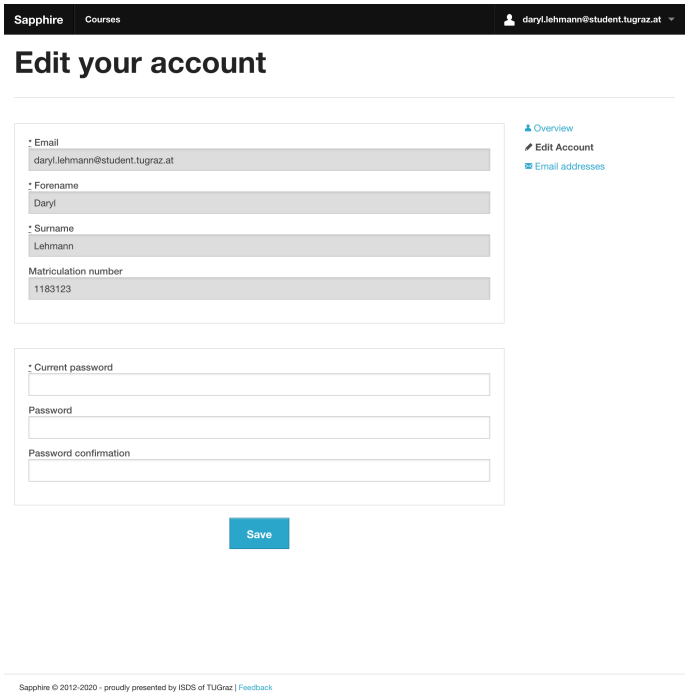

Figure A.4: Using the Edit Your Account Form to change the current password.

account password is updated once the user clicks on Save, as long as the current password is correct, and the password field matches the password confirmation field.

A.2 Courses Overview

Once the user signs into Sapphire, an overview of registered course terms is displayed in the Courses Overview, as shown in Figure A.5. Each row of the table consists of a link to the term dashboard, the number of exercises, and the number of tutorial groups.

A.3 Term Dashboard

The Term Dashboard serves as the starting point for each term of a particular course. It displays of a list of events, providing an overview of the latest changes in Sapphire, as shown in Figure A.6. For students, the following families of events are displayed:

- *Submission Events*: Sapphire keeps track of the time a student creates and updates a submission along with information about the changed files. Additionally, Sapphire tracks the time of extraction of uploaded ZIP files. Sapphire only displays events related to submissions owned by the student.
- *Result Publication Events*: During the course of the term, the lecturer publishes results, as the grading for each exercise is finished. Sapphire keeps track of these events and adds an event to the event feed.

A.4 Exercises

Exercises serve as the basis for submissions in Sapphire. Each exercise is worth a specific number of points, which sum up to the total points of the term. There are two types of exercise: individual exercises and group exercises. Each exercise is configured to have a deadline by which the submissions needs to

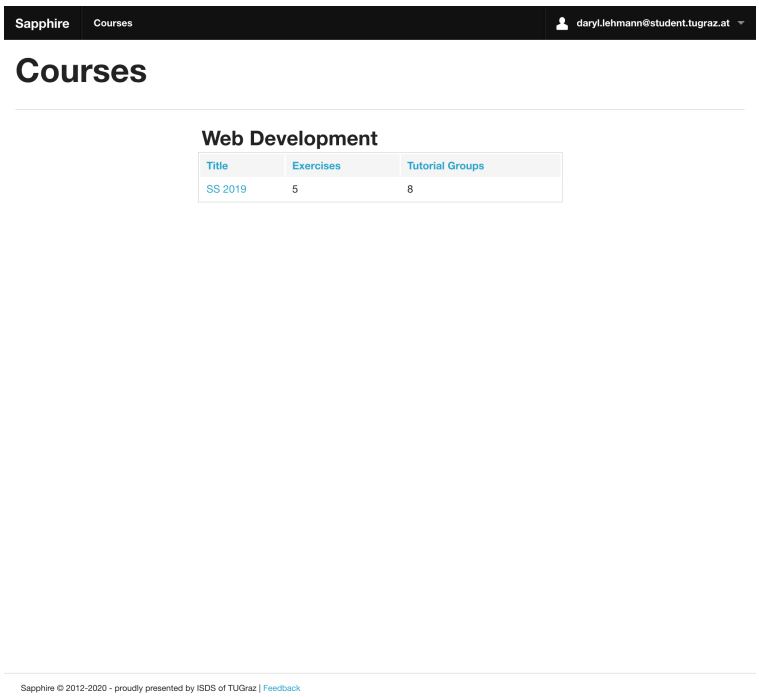

Figure A.5: The Courses Overview is displayed after logging into Sapphire, and contains a list of course terms the student is registered for.

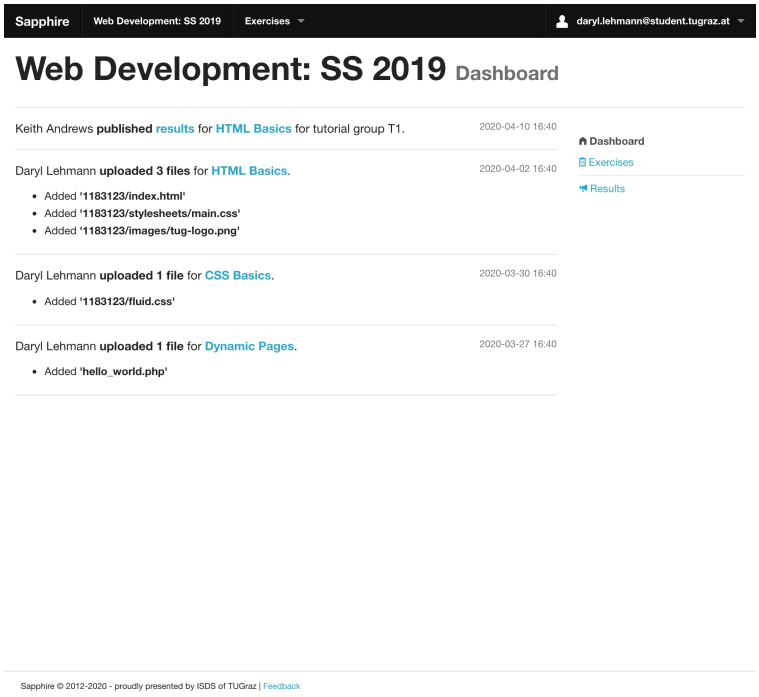

Figure A.6: The Term Dashboard shows the event feed.

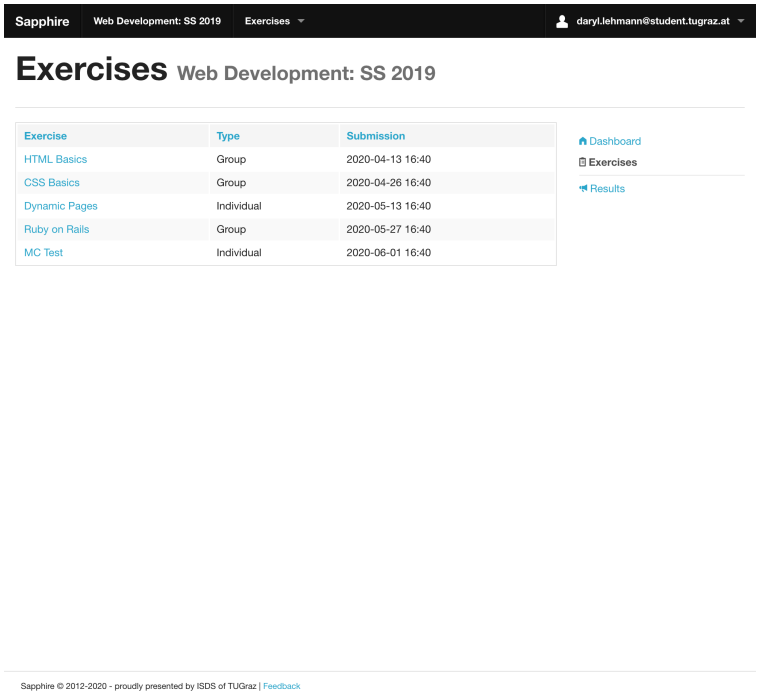

Figure A.7: The Exercises Table provides an overview of the exercises belonging to a term.

be finished. Optionally, a lecturer is able to configure a late deadline, after which changes to submissions are prohibited. It is therefore possible for students to change a submission after the regular deadline, but before the late deadline. Sapphire considers the time of the last change to be the time of submission. Some exercises require uploading files to the Sapphire system, while other exercises are managed by staff members (lecturers and tutors).

A.4.1 Exercises Table

The Exercises Table provides an overview of all the exercises belonging to a term, as shown in Figure A.7. The Exercises Table includes the name of the exercise, its type, and the submission deadline. The user is navigated to the corresponding detail page by clicking on the exercise title.

A.4.2 Exercise Detail

The Exercise Detail page provides a summary of the attributes of an exercise, as shown in Figure A.8. The information includes the exercise description, a URL to the instructions, the achievable points, the type of the exercise, maximum upload size, minimum points required, deadline, and late deadline. Empty attributes are hidden from the interface to improve readability.

A.5 Submissions

During the course of a term, students are required to take part in exercises. Submissions to exercises are graded by staff members. Once the grading of an exercise is finished, the results can be published for students to view, as described in Section A.6. There are two types of submissions in Sapphire: individual submissions and group submissions. Individual submissions require each student to upload files to Sapphire individually. Group submissions require at least one member of the student group to upload and manage the submission in Sapphire.

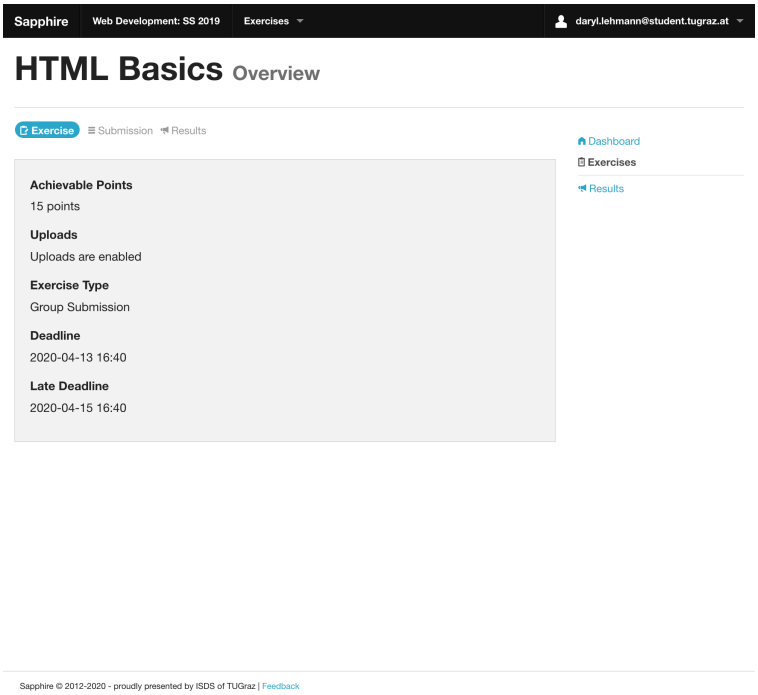

Figure A.8: The Exercise Detail page provides a summary of the attributes of an exercise.

A.5.1 Creating a Submission

In order to upload files to Sapphire, students are required to first create a submission, as shown in Figure A.9. Sapphire displays a short text describing the type of the exercise and includes information about which students will be assigned to the submission. Once the Create a Submission for Exercise button is clicked, Sapphire prepares a submission in the background and redirects the user to the Submission Tree interface.

A.5.2 Submission Tree

The Submission Tree interface provides users with the ability to navigate and manage the files and folders of a submission, as shown in Figure A.10. The Submission Tree interface consists of a table listing the files and subfolders inside the current folder, called the File Table. Sapphire shows the modification date and file size of each entry in the File Table. Users are able to navigate into a folder by simply clicking on its name. To navigate to a parent folder, users can either use the .. link in the first row of the File Table or use the breadcrumb navigation above the File Table. The raw content of a file is accessible by clicking on the corresponding file name. Sapphire allows users to delete files and folders by clicking on the red x button of the corresponding entry in the File Table.

A.5.3 Uploading Files

The Upload New Files Modal is used to upload files to Sapphire, as shown in Figure A.11. The Upload New Files Modal is opened by clicking the Upload button in the toolbar of the Submission Tree interface. Users are able to drag and drop files from the desktop into the Drop Zone area of the Upload New Files Modal, which is indicated by the text Drop files here or click to select. Alternatively, users are able to click on the Drop Zone to select files from a dialogue provided by the browser, as the text suggests. By default, Sapphire assumes that new files should be added to the current folder. Users are able to change the destination folder by clicking on the Edit link.

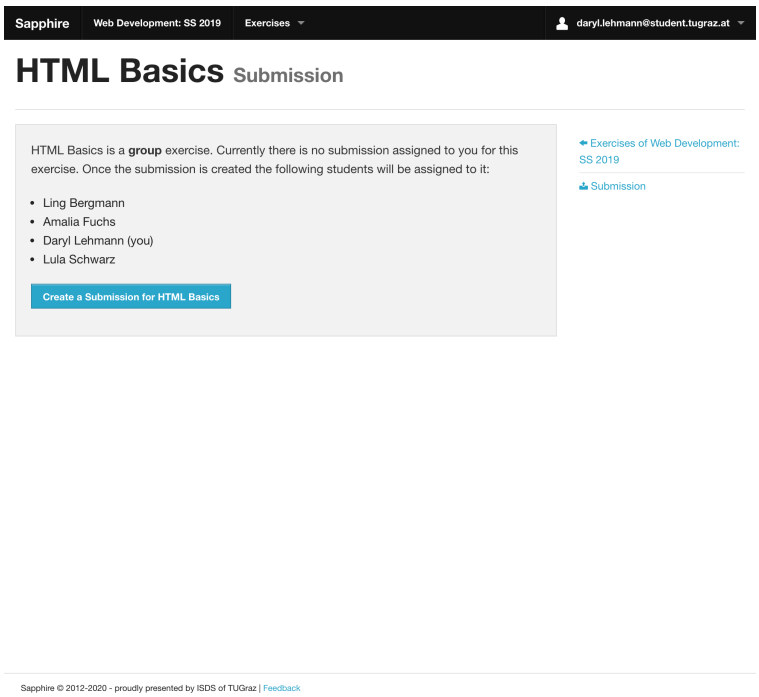

Figure A.9: Creating a group submission.

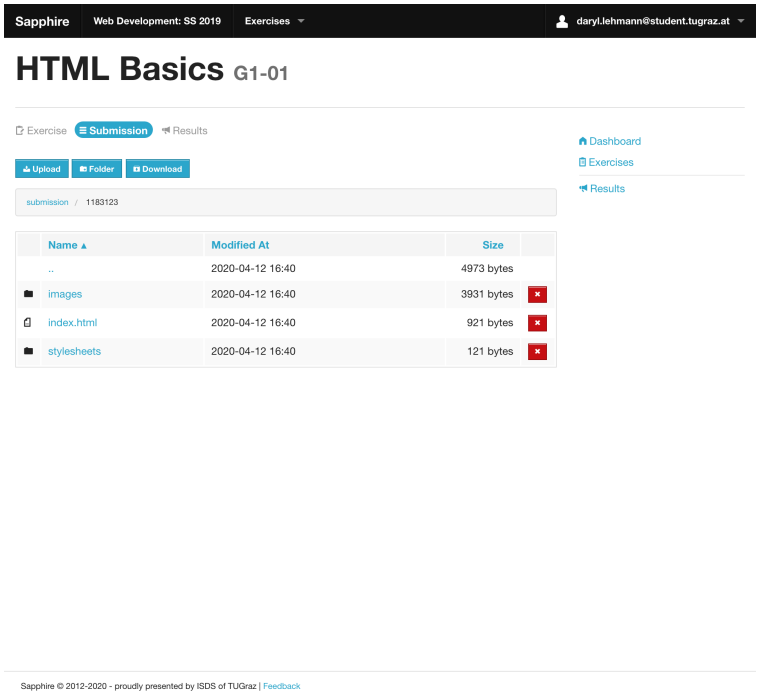

Figure A.10: The Submission Tree interface allows users to navigate and change files, similar to the file browser of an operating system.

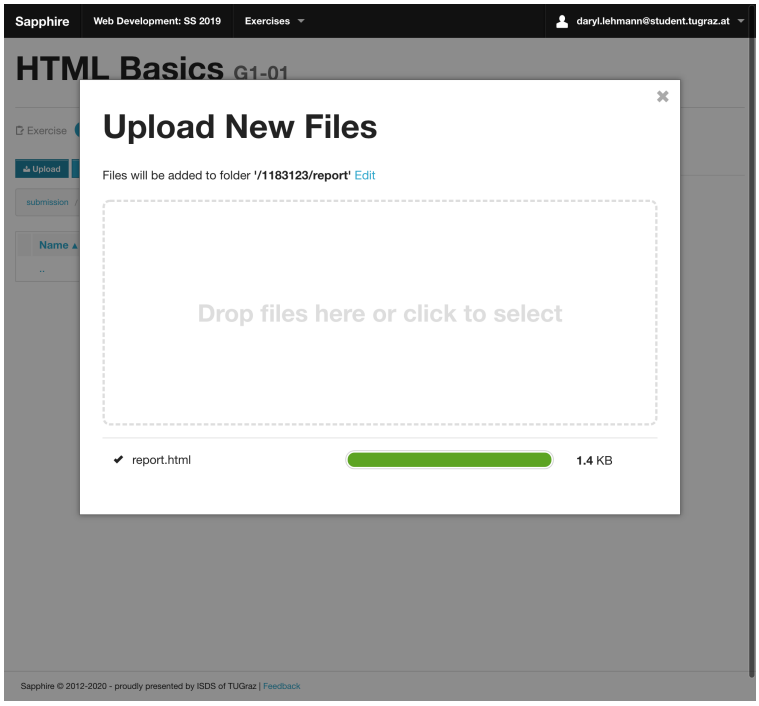

Figure A.11: The Upload New Files Modal is used to upload files to a submission.

Sapphire also supports uploading complex folder structures in the form of ZIP archives. Sapphire automatically extracts ZIP archives in a background process, provided the maximum upload size will not be exceeded. Some browsers, for example Google Chrome, support uploading folders directly, eliminating the need to create a ZIP archive.

A.5.4 Downloading Files

Sapphire allows users to download the contents of the current folder in the Submission Tree interface. The download process is started by clicking the Download button in the toolbar of the Submission Tree interface. Sapphire dynamically creates a ZIP archive, which is streamed to the browser as a file download.

A.5.5 Creating Folders

New folders are created via the New Folder Modal by clicking the Folder button in the toolbar of the Submission Tree interface, as shown in Figure A.12. Users are able to both enter a new folder name as well as a subfolder path, using slashes as path dividers. Once the Create button is clicked, the user is navigated to the corresponding folder path. Creating folders in Sapphire differs from traditional file browsers in that folders are not physically created until files are uploaded to the folder.

Sapphire automatically checks that the folder name is available and indicates the status below the input field. The result of the availability check does not prohibit users from submitting the form. Instead, the availability status is displayed to allow users to catch errors early during the submission process.

For security reasons, Sapphire simulates the file system shown to its users. While Sapphire's file system is similar to traditional ones in terms of functionality, it does not implement renaming or moving files from one folder to another. Instead, users are currently required to remove and reupload files and folders. However, it is likely this feature will be implemented in a future release of Sapphire.

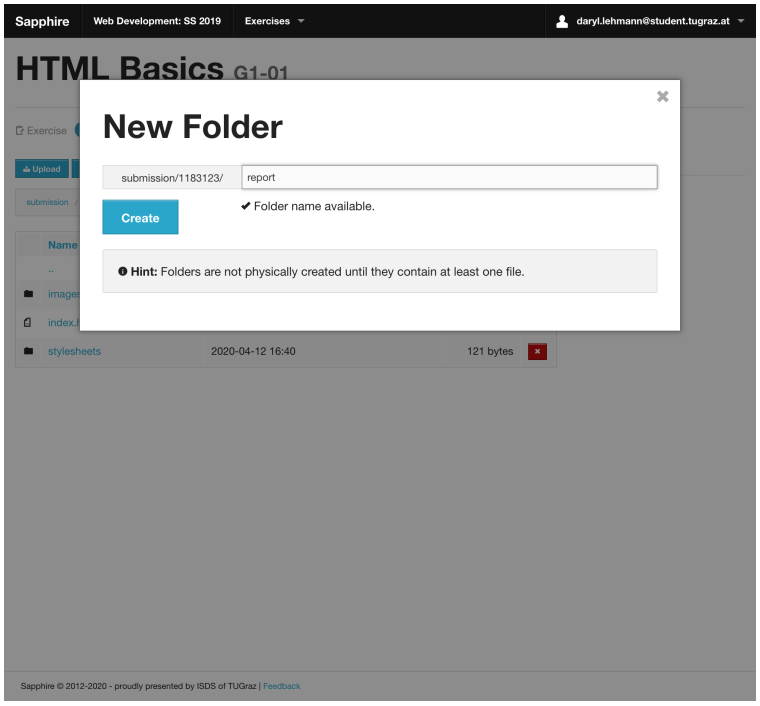

Figure A.12: The New Folder Modal is used to create a new folder.

A.5.6 Limitations

Despite its appearance, the file system of Sapphire is simulated during runtime. Therefore, there are several limitations regarding the functionality of the file system. One of the most noticeable is that Sapphire does not support moving or renaming files and folders though the UI. Instead, users are required to download and remove the concerned files from Sapphire, then renaming the files on the local file system, and reuploading them to Sapphire.

A.6 Results

Sapphire allows students to look at the results published during the course of a term as grading progresses. The Results Table shows of an overview of all the available results, as shown in Figure A.13. For each exercise, the possible points and received points are shown. The preliminary grade based on the points so far is shown underneath the table.

Additionally, Sapphire provides details on the grading of a particular exercise via the Details link. The Results Detail page, shown in Figure A.14, first shows the points reached and points possible at the top of the page. Underneath, detailed information about point deductions is shown. Sapphire shows the results of each rating group, as well as details for those ratings which have been applied.

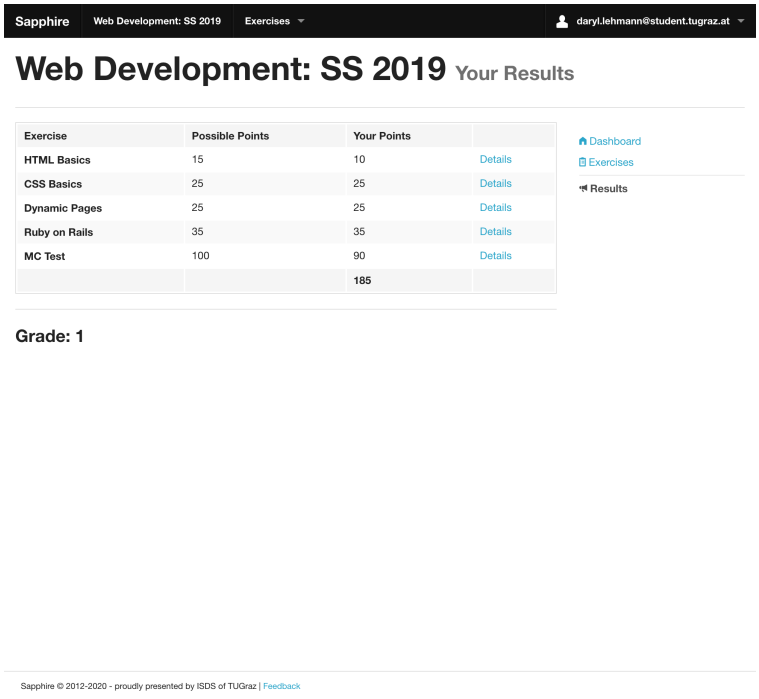

Figure A.13: The Results Table provides an overview of a student's results, as far as they are available.

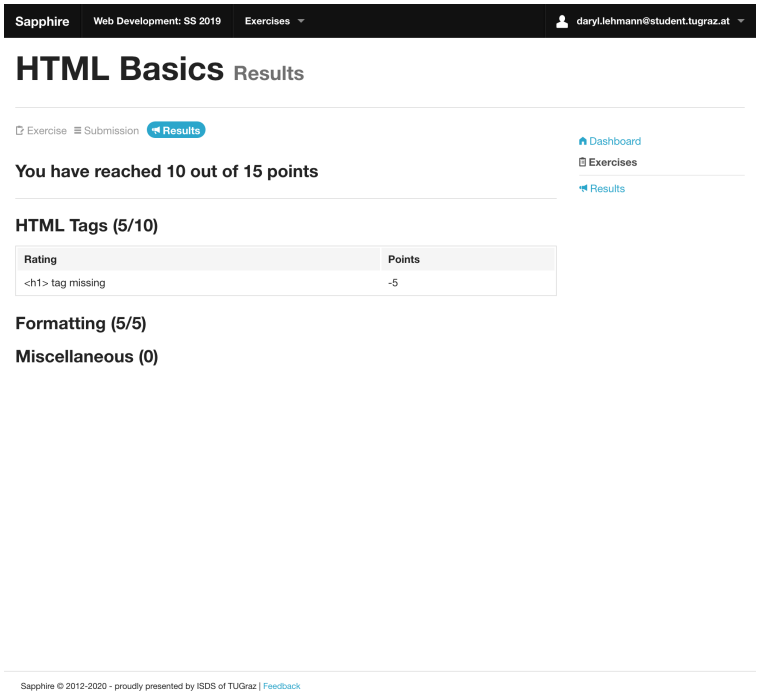

Figure A.14: The Results Detail page provides detailed information about the grading for a particular exercise, including the results of each rating group and details for those ratings which have been applied.

Appendix B

Tutor Guide

Sapphire is an online course management and submission platform. This guide is intended for tutors of university courses which use the Sapphire system, and provides insight into the most important features and typical workflows.

B.1 Authentication

User accounts are used to manage access to the Sapphire course management system. Authentication is the means for signing into user accounts, requesting a password reset, editing the main attributes of an account, and changing the password.

B.1.1 Login

The first step in using Sapphire is to log into a Sapphire account, using the Login Form shown in Figure B.1. Tutor accounts are assigned by lecturers to the corresponding terms. However, the tutor needs to create an account first.

First-time users of Sapphire must use the password reset feature to request a new password, as presented in Section B.1.2. The Password Reset Form is accessible through the Forgot your password? link beneath the Login Form. Previous users of Sapphire are able to reuse their credentials from a former term. Sapphire uses a role-based authentication system on a per-term basis. It is therefore possible to use a single Sapphire account for multiple terms.

B.1.2 Forgot Your Password

The Forgot Your Password allows users to reset their Sapphire account password. The process of resetting a user password is started by entering the email address of the account into the Password Reset Form and clicking the Reset button, as shown in Figure B.2. If the account is present in the database, Sapphire sends an email containing a password reset link. The user is allowed to choose a new password by visiting the included link, as shown in Figure B.3.

B.1.3 Changing Passwords

In Sapphire, passwords are changed on the user's profile page, which is accessible through the Edit Account link in the navigation bar. The user is presented with read-only meta-data as well as a form for changing the password, as shown in Figure B.4.

In order to change the password, the current password needs to be typed in first. Then, a new password is entered into the Password field, followed by the same password in the Password confirmation field. The

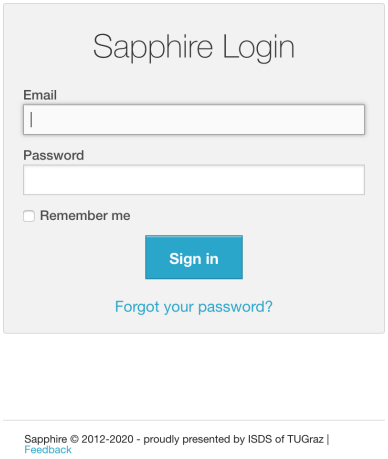

Figure B.1: The Sapphire Login Form.

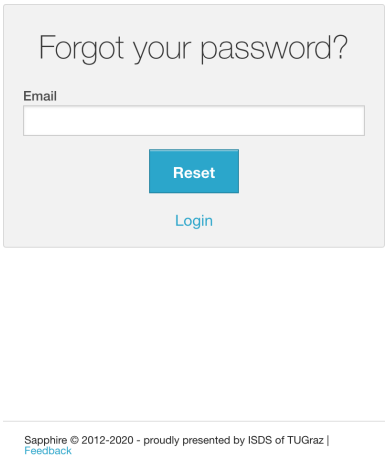

Figure B.2: The Forgot Your Password Form allows users to request a password reset link.

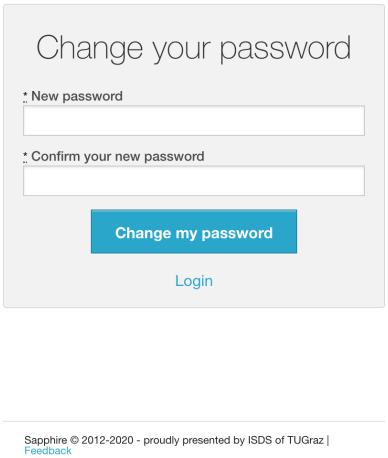

Figure B.3: The Change Your Password Form allows users to change their password after requesting a password reset link.

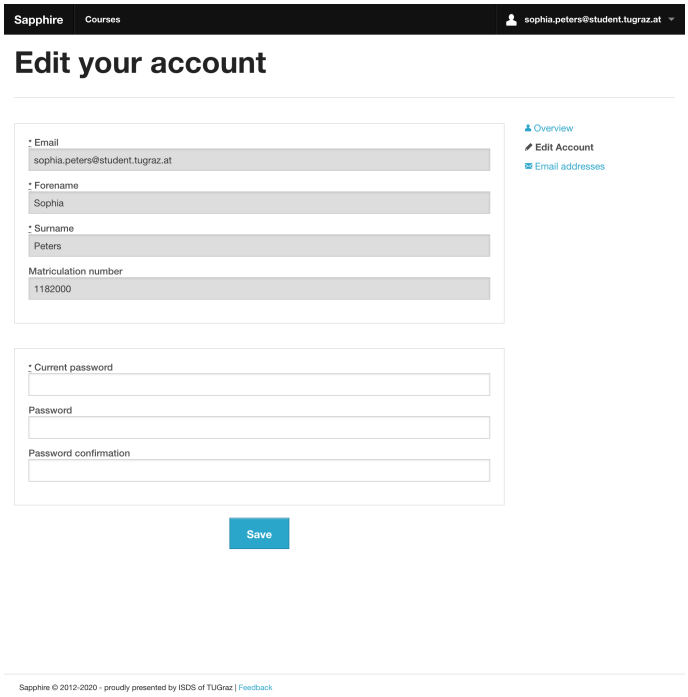

Figure B.4: Using the Edit Your Account Form to change the current password.

account password is updated once the user clicks on Save, as long as the current password is correct, and the password field matches the password confirmation field.

B.2 Courses Overview

Once the user signs into Sapphire, an overview of registered course terms is displayed in the Courses Overview, as shown in Figure B.5. Each row of the table consists of a link to the term dashboard, the number of exercises, the number of tutorial groups, and the number of students.

B.3 Term Dashboard

The Term Dashboard serves as the starting point for each term of a particular course. It displays of a list of events, providing an overview of the latest changes in Sapphire, as shown in Figure B.6. For tutors, the following families of events are displayed:

- *Submission Events*: Sapphire keeps track of the time a student creates and updates a submission along with information about the changed files. Additionally, Sapphire tracks the time of extraction of uploaded ZIP files.
- *Result Publication Events*: During the course of the term, the lecturer publishes results, as the grading for each exercise is finished. Sapphire keeps track of these events and adds an event to the event feed.
- *Rating Group Events*: The points and titles of rating groups are subject to change throughout the term. Lecturers are able to create, update, and delete rating groups of an exercise. Sapphire keeps track of these changes to provide a transparent history of changes to staff members (lecturers and tutors).

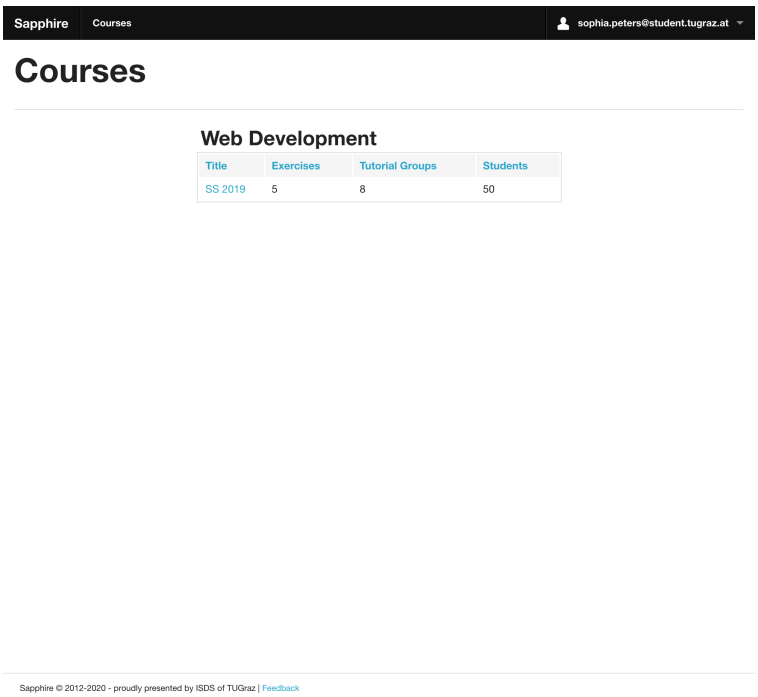

Figure B.5: The Courses Overview is displayed after logging into Sapphire, and contains a list of course terms the tutor is registered for.

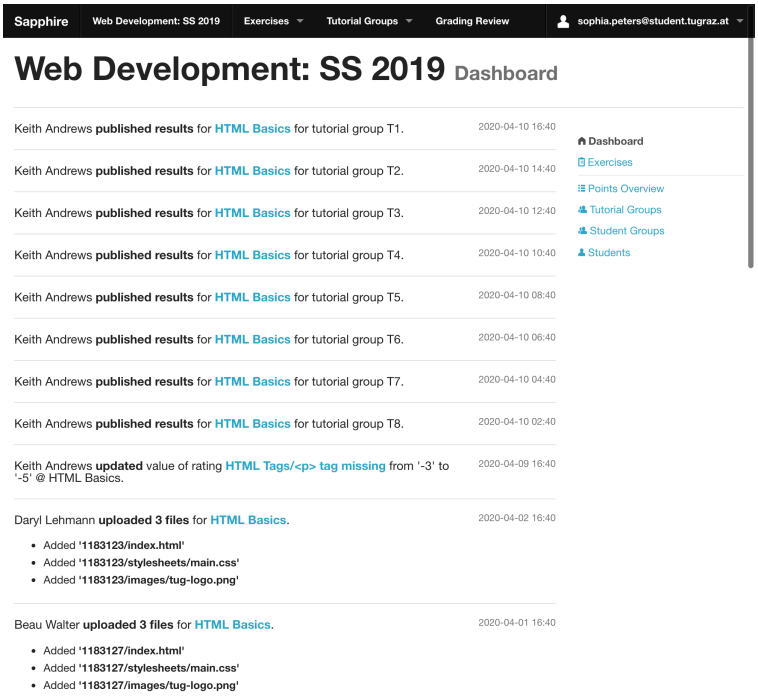

Figure B.6: The Term Dashboard shows the event feed.

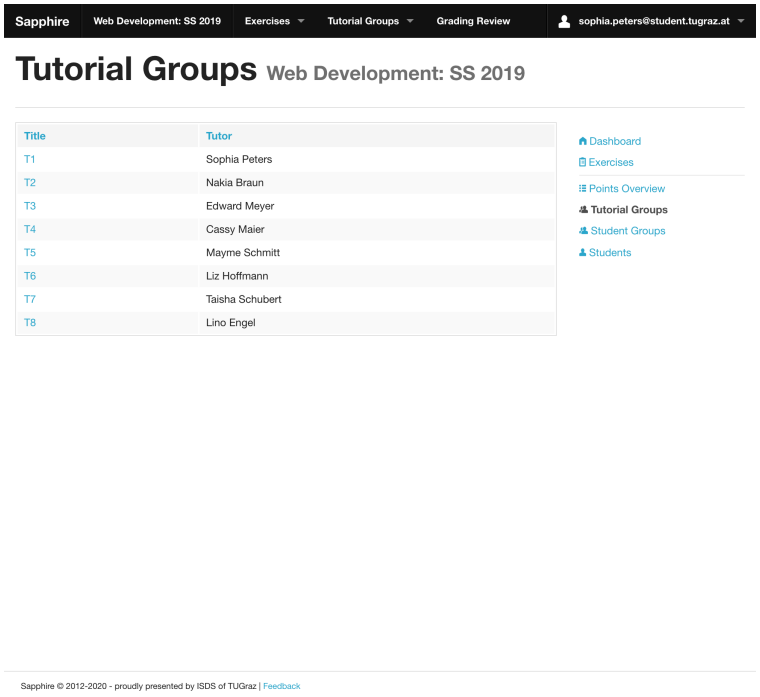

Figure B.7: The Tutorial Groups table provides an overview of the tutorial groups of a term.

- *Rating Events*: Sapphire enables lecturers to create, update, and delete the ratings of a rating group. Sapphire keeps track of these changes to the database, similar to rating groups. As with rating group events, tutors are kept informed about changes to the grading structure.

B.4 Tutorial Groups

Sapphire is designed to handle large university courses of hundreds of students. The number of students and submissions is usually too large to be handled by a single person. Therefore, each student is assigned to one of several tutorial groups, to split the workload of managing and grading students amongst several staff members.

B.4.1 Tutorial Groups Table

Sapphire provides an overview of the tutorial groups of a term in the Tutorial Groups table, as shown in Figure B.7. The table consists of the name of each tutorial group and the tutor or tutors responsible. Clicking on the tutorial group title navigates the user to the Tutorial Group Detail page.

B.4.2 Tutorial Group Detail

The Tutorial Group Detail page provides a table of students belonging to a particular tutorial group, as shown Figure B.8. The table displays the name, matriculation number, and email address for each student. Clicking on the Show button navigates the user to the Student Detail page, described in Section B.6.2

B.5 Student Groups

Sapphire supports submissions by groups of students via the concept of student groups. Student groups represent the current group constellation of students. Moving a student from one group to another during a term does not affect their association with any previous submissions as part of another group.

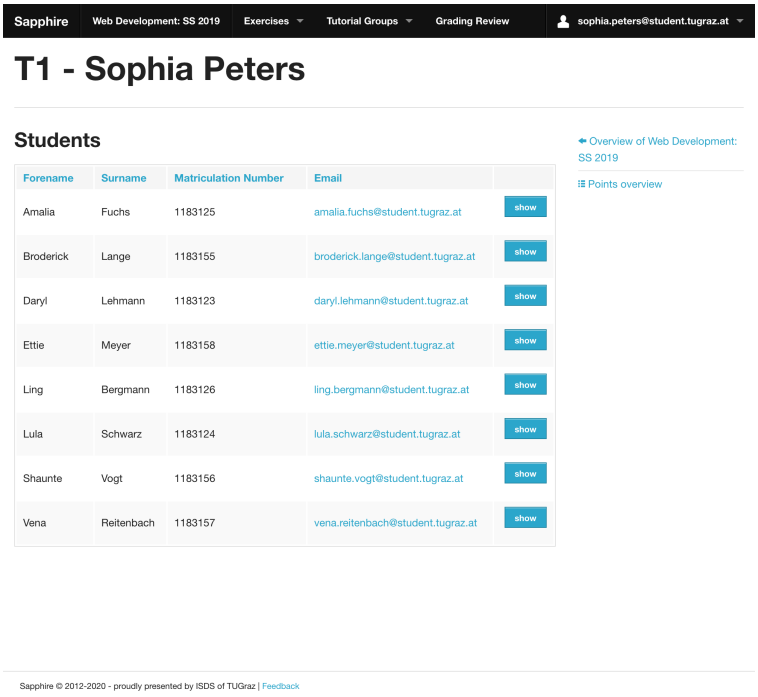

Figure B.8: The Tutorial Group Detail page provides a table of students of the tutorial group.

B.5.1 Student Groups Table

The Student Groups table provides an overview of the student groups currently available in a term, as shown in Figure B.9. The Student Groups table shows the title, topic, tutorial group, and number of students in each student group. For each student group in the table, the corresponding Show button navigates the user to the Student Group Detail page.

B.5.2 Student Group Detail

The Student Group Detail page provides an overview of a student group, as shown in Figure B.10. The panel on top of the page provides information about the associated tutorial group and optional topic and keyword attributes.

A Students table is provided beneath the information panel, listing the students currently belonging to the student group. The Students table shows the forename, surname, matriculation numbers, and email addresses of each student in the group. A Show button allows users to navigate to the student detail page, described in Section B.6.2.

The Student Group Detail page concludes with the Submissions table, containing a list of group submissions by the student group. The Submissions table includes information about the exercise, date of submission, and resulting points. Individual submissions by group members are not included. Both Show and Evaluate buttons are displayed next to each submission. The Show button navigates the user to the file browser of the submission, described in Section B.8.2. The Evaluate button navigates the user to the submission evaluation page, described in Section B.9.1.

B.6 Students

Managing hundreds of students is one of the core features of Sapphire. Students are responsible for creating submissions during the course of a term and Sapphire enables students to view results throughout the term as they are published. Lecturers are responsible for managing students and student groups.

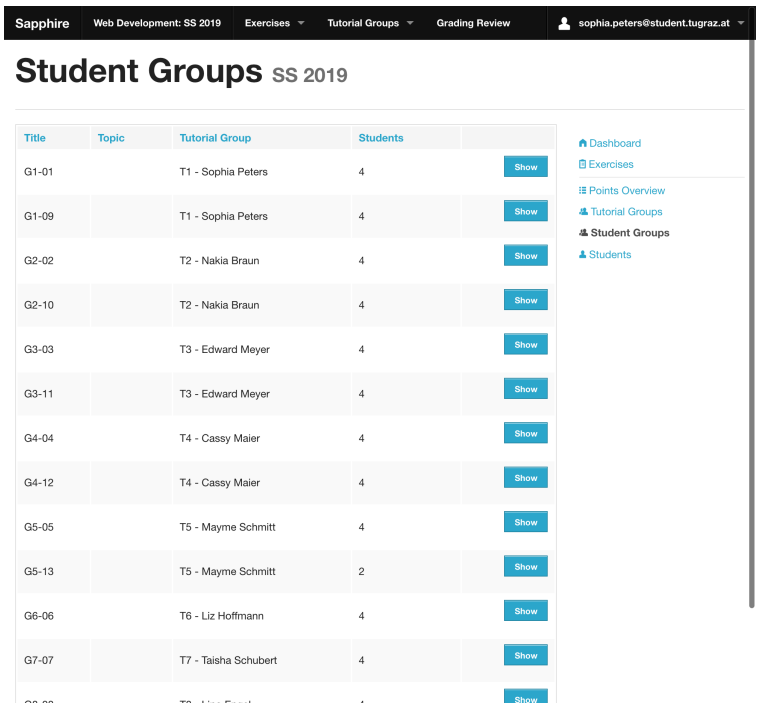

Figure B.9: The Student Groups table provides an overview of the student groups of a term.

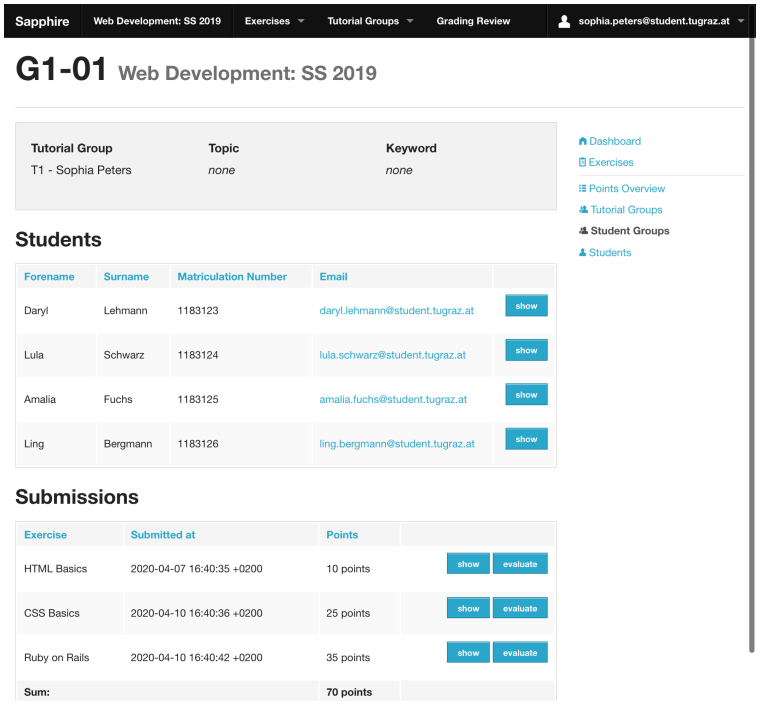

Figure B.10: The Student Group Detail page provides information about the attributes, students, and submissions of a student group.

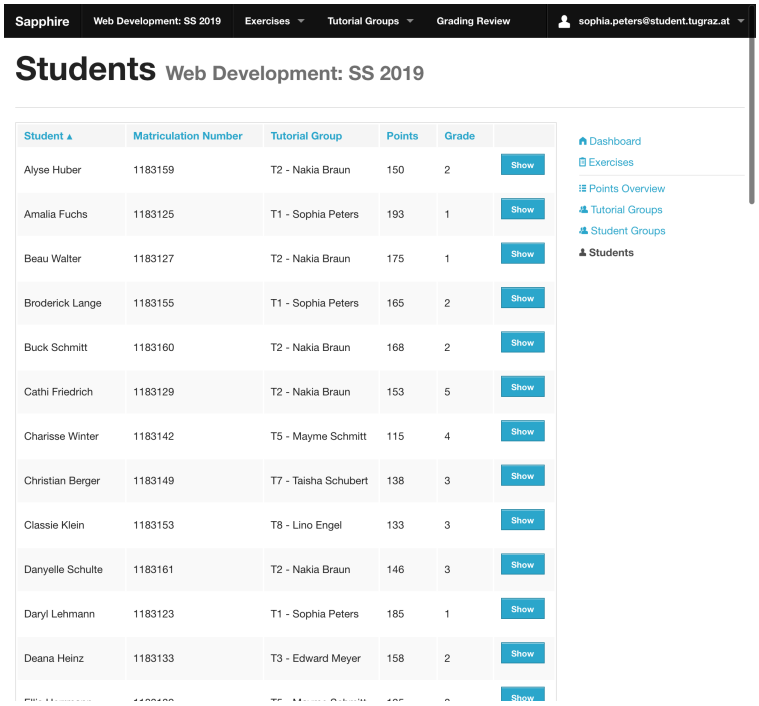

Figure B.11: The Students table displays a list of students participating in a term.

B.6.1 Students Table

The Students table displays a list of students currently participating in a term, as shown in Figure B.11. The table provides the name, matriculation number, tutorial group, received points, and preliminary grade for each student. Detailed information about each student can be obtained clicking on the Show button in the corresponding row.

B.6.2 Student Detail

The Student Detail page provides detailed information about a particular student of a term, as shown in Figure B.12. Sapphire presents a Grading Review button at the top of the page, which is used to navigate to the Grading Review page of the student, described in Section B.11. The information panel located below displays the current total points, preliminary grade, tutorial group, and student group of the student.

Beneath the information panel, a table of submissions by the student is shown. For each submission, the exercise, date of submission, and resulting points are displayed. Next to each submission, the Show button is used to navigate to the file browser of the submission, described in Section B.8.2. The Evaluate button navigates to the submission's evaluation page, described in Section B.9.1.

B.7 Exercises

Exercises serve as the basis for submissions in Sapphire. Each exercise is worth a specific number of points, which sum up to the total points of the term. There are two types of exercise: individual exercises and group exercises. Each exercise is configured to have a deadline by which the submissions needs to be finished. Optionally, a lecturer is able to configure a late deadline, after which changes to submissions are prohibited. It is therefore possible for students to change a submission after the regular deadline, but before the late deadline. Sapphire considers the time of the last change to be the time of submission. Some exercises require uploading files to the Sapphire system, while other exercises are managed by staff members (lecturers and tutors).

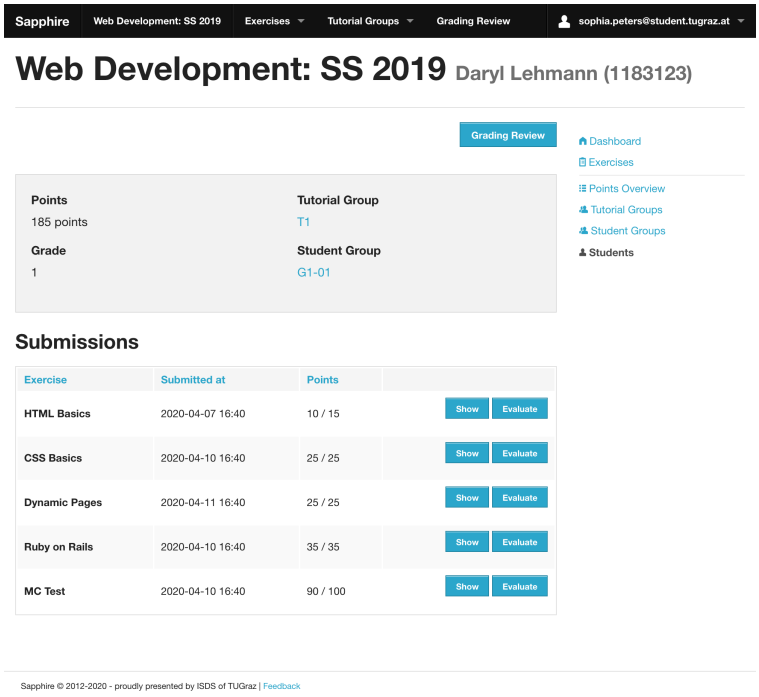

Figure B.12: The Student Detail page presents detailed information about a student of a term.

B.7.1 Exercises Table

The Exercises Table provides an overview of all the exercises belonging to a term, as shown in Figure B.13. The Exercises Table includes the name of the exercise, its type, and the submission deadline. The user is navigated to the corresponding detail page by clicking on the exercise title.

B.7.2 Exercise Detail

The Exercise Detail page provides a summary of the attributes of an exercise, as shown in Figure B.14. The information includes the exercise description, a URL to the instructions, the achievable points, the type of the exercise, maximum upload size, minimum points required, deadline, and late deadline. Empty attributes are hidden from the interface to improve readability.

B.8 Submissions

During the course of a term, students are required to take part in exercises. Usually, submissions are managed by students themselves. However, Sapphire also allows staff members to manage submissions, due to, for example, students being unable to upload to Sapphire directly. Additionally, staff members can create graded submissions using the Bulk Grading interface, described in Section B.9.2.

B.8.1 Submissions Table

The Submissions Table displays a list of current submissions for an exercise, as shown in Figure B.15. For each submission, Sapphire shows the student group, date of submission, date of evaluation, and the resulting points. For individual submissions, the name of the student is included. For exercises allowing multiple attempts, Sapphire also includes the attempt in the Submissions Table. If ratings or rating groups are changed by the lecturer after the submission was last evaluated, Sapphire shows a warning sign next to the points in the points column. The warning sign indicates to staff members that the evaluation may be out of date and that the submission's evaluation should be reviewed.

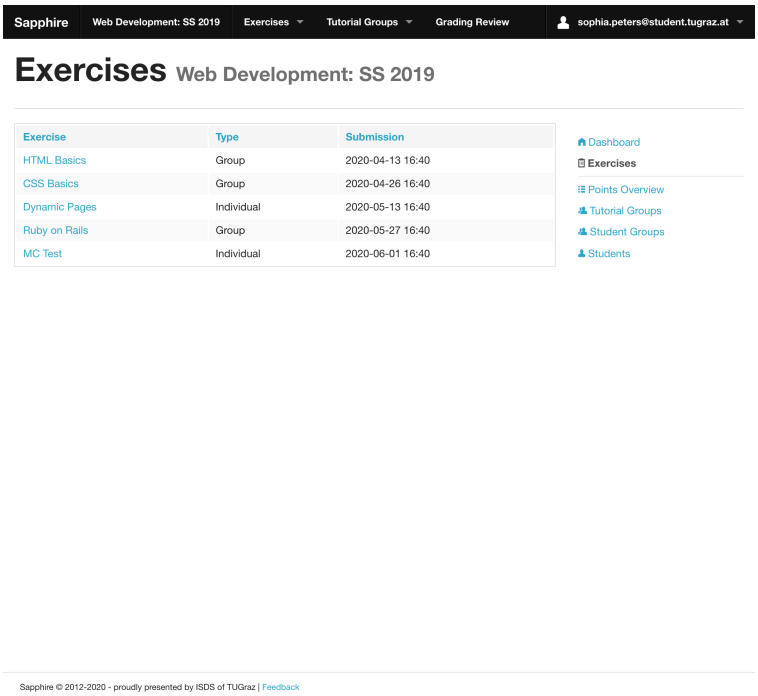

Figure B.13: The Exercises Table provides an overview of the exercises belonging to a term.

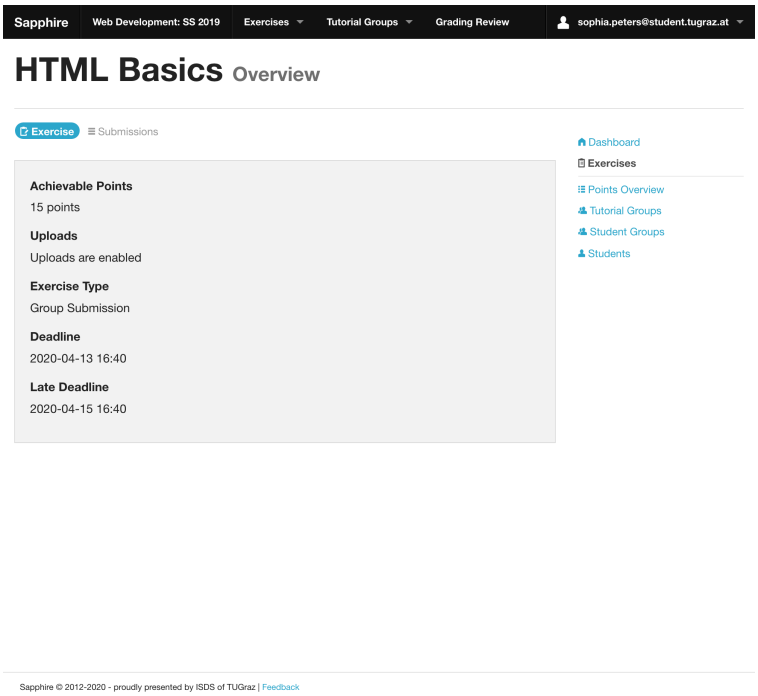

Figure B.14: The Exercise Detail page provides a summary of the attributes of an exercise.

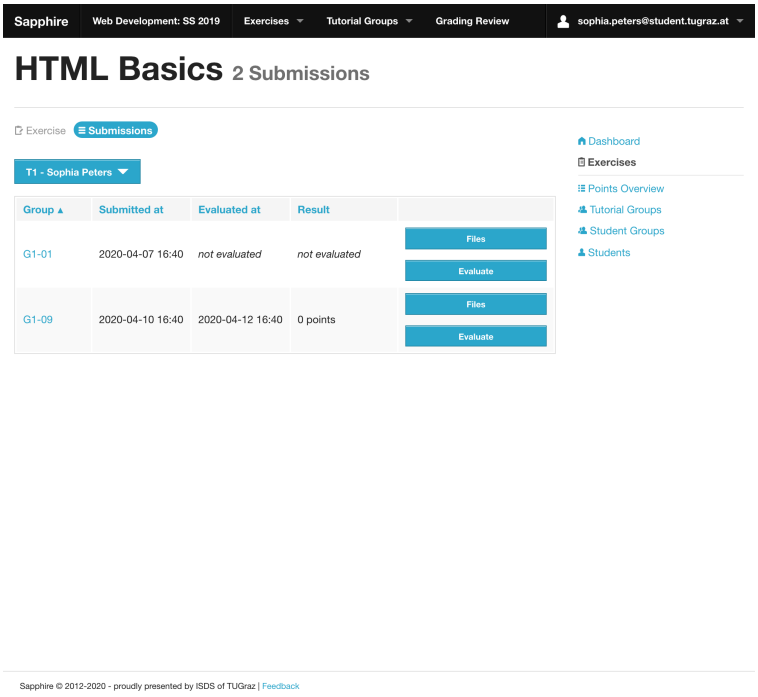

Figure B.15: The Submissions Table displays a list of the current submissions for an exercise.

B.8.2 Submission Tree

The Submission Tree interface provides users with the ability to navigate and manage the files and folders of a submission, as shown in Figure B.16. The Submission Tree interface consists of a table listing the files and subfolders inside the current folder, called the File Table. Sapphire shows the modification date and file size of each entry in the File Table. Users are able to navigate into a folder by simply clicking on its name. To navigate to a parent folder, users can either use the .. link in the first row of the File Table or use the breadcrumb navigation above the File Table. The raw content of a file is accessible by clicking on the corresponding file name. Sapphire allows users to delete files and folders by clicking on the red x button of the corresponding entry in the File Table.

B.8.3 Uploading Files

The Upload New Files Modal is used to upload files to Sapphire, as shown in Figure B.17. The Upload New Files Modal is opened by clicking the Upload button in the toolbar of the Submission Tree interface. Users are able to drag and drop files from the desktop into the Drop Zone area of the Upload New Files Modal, which is indicated by the text Drop files here or click to select. Alternatively, users are able to click on the Drop Zone to select files from a dialogue provided by the browser, as the text suggests. By default, Sapphire assumes that new files should be added to the current folder. Users are able to change the destination folder by clicking on the Edit link.

Sapphire also supports uploading complex folder structures in the form of ZIP archives. Sapphire automatically extracts ZIP archives in a background process, provided the maximum upload size will not be exceeded. Some browsers, for example Google Chrome, support uploading folders directly, eliminating the need to create a ZIP archive.

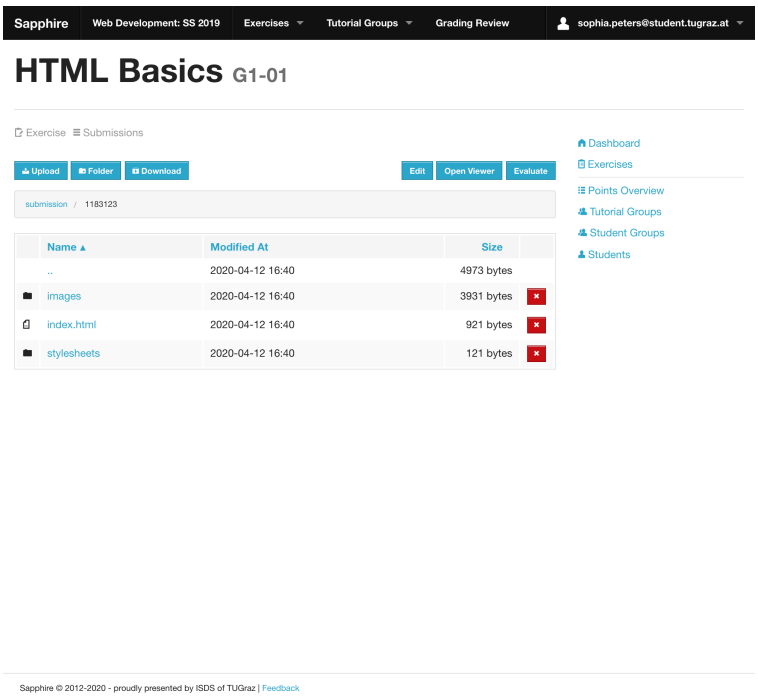

Figure B.16: The Submission Tree interface allows users to navigate and change files, similar to the file browser of an operating system.

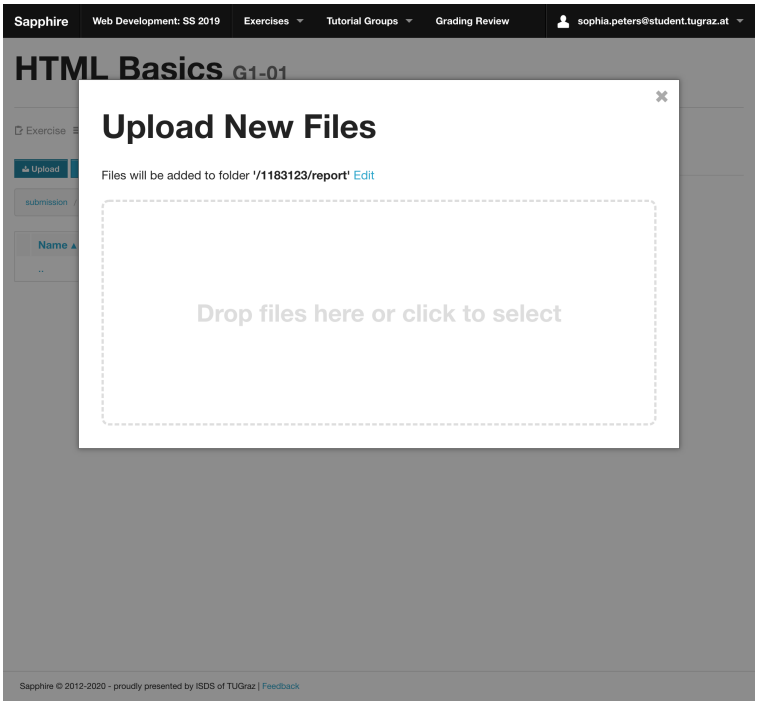

Figure B.17: The Upload New Files Modal is used to upload files to a submission.

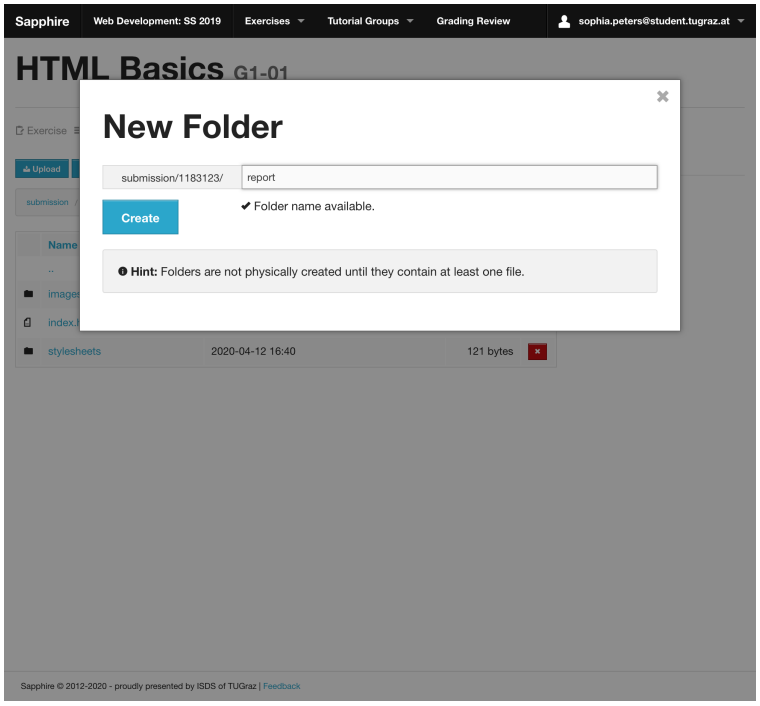

Figure B.18: The New Folder Modal is used to create a new folder.

B.8.4 Downloading Files

Sapphire allows users to download the contents of the current folder in the Submission Tree interface. The download process is started by clicking the Download button in the toolbar of the Submission Tree interface. Sapphire dynamically creates a ZIP archive, which is streamed to the browser as a file download.

B.8.5 Creating Folders

New folders are created via the New Folder Modal by clicking the Folder button in the toolbar of the Submission Tree interface, as shown in Figure B.18. Users are able to both enter a new folder name as well as a subfolder path, using slashes as path dividers. Once the Create button is clicked, the user is navigated to the corresponding folder path. Creating folders in Sapphire differs from traditional file browsers in that folders are not physically created until files are uploaded to the folder.

Sapphire automatically checks that the folder name is available and indicates the status below the input field. The result of the availability check does not prohibit users from submitting the form. Instead, the availability status is displayed to allow users to catch errors early during the submission process.

For security reasons, Sapphire simulates the file system shown to its users. While Sapphire's file system is similar to traditional ones in terms of functionality, it does not implement renaming or moving files from one folder to another. Instead, users are currently required to remove and reupload files and folders. However, it is likely this feature will be implemented in a future release of Sapphire.

B.8.6 Limitations

Despite its appearance, the file system of Sapphire is simulated during runtime. Therefore, there are several limitations regarding the functionality of the file system. One of the most noticeable is that Sapphire does not support moving or renaming files and folders though the UI. Instead, users are required to download and remove the concerned files from Sapphire, then renaming the files on the local file system, and reuploading them to Sapphire.

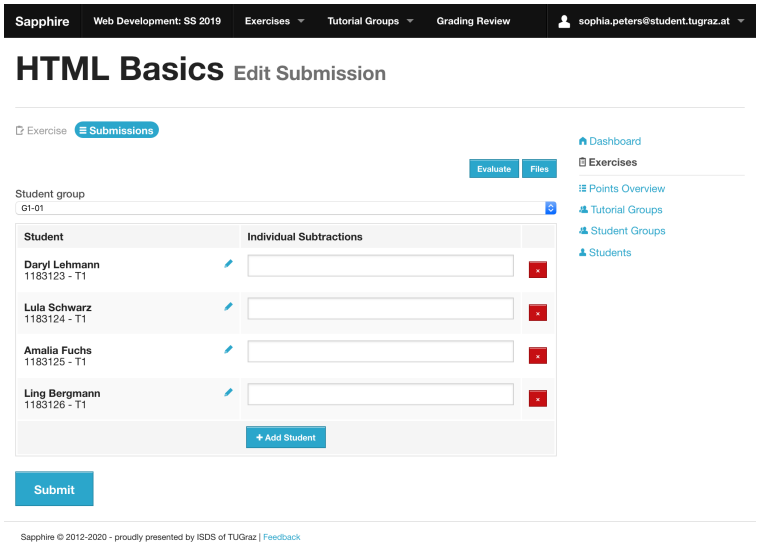

Figure B.19: The Edit Submission interface for submissions lets staff members change the associated student group, exercise attempt, and associated students, along with managing any individual point deductions.

B.8.7 Administrating Submissions

Sapphire enables staff members (tutors and lecturers) to modify submissions using the Edit Submission interface, shown in Figure B.19. Staff members are able to configure the students assigned to a submission. The students comprising a student group can only be changed by lecturers during the course of a term. However, if the students assigned to a student group are changed after the submission has been created, the students assigned to the submission need to be altered manually via this interface. In order to simplify this process, both lecturers and tutors are able to perform this action.

By clicking the Add Student button, Sapphire adds an entry to the table of students. Next, the staff member needs to search for the relevant student by entering either the name or matriculation number in the search field and selecting the corresponding entry from the list of search results displayed as a dropdown. Existing student associations can be changed similarly, by clicking the Pencil icon next to the name of the student, and proceeding with searching for another student. A student can be removed from a submission by clicking the x button in the corresponding row.

Staff members are able to set individual point deductions for students associated with a submission, for example if one student has not contributed as much to a submission as other group members. In case multiple attempts are enabled for an exercise, the attempt of the exercise can be changed as well. All changes to the submission need to be confirmed by clicking the Submit button. Sapphire performs the changes only after the form has been submitted.

B.9 Grading

The ability to grade and manage hundreds of students is one of the key features of Sapphire. A family of interfaces is responsible for providing this functionality. The Single Evaluation interface, described in Section B.9.1, provides the universal grading interface for staff members (tutors and lecturers). There is, however, a second interface called the Bulk Grading interface, which is described in Section B.9.2. While

the Single Evaluation interface is primarily used for report-based submissions, the Bulk Grading interface is used to enter results determined outside the scope of Sapphire, for example examination scores.

B.9.1 Single Evaluation

The Single Evaluation interface provides the primary grading interface, and is usually used for report-based submissions. The interface is based on three components: the Header, the List of Ratings, and the Footer, and is shown in Figure B.20.

The Header prominently displays the submitter of the submission. Sapphire either displays the student group or the student, depending on whether the exercise is a group or individual exercise, respectively. The current points for the submission and a set of three buttons are shown beneath the name of the submitter. The Edit button navigates to the administrative interface of the submission, described in Section B.8.7. The Open Viewer button is shown for exercises with a submission viewer enabled. The Open Viewer button navigates to the Submission Viewer for the submission, as described in Section B.10. The Files button navigates to the Submission Tree interface, described in Section B.8.2.

The List of Ratings constitutes the main section of the single evaluations interface. The ratings of an exercise are collected into rating groups. Each rating group is worth an initial number of points. Ratings are used to deduct from the initial points of a rating group. Sapphire displays either a button or an input field for each rating, depending on its type. The points are saved and are updated automatically whenever a rating button is clicked or an input field is filled in. Rating buttons are initially grey. Clicking on a rating button turns the button red and the underlying deductions or bonus points are applied.

It is common for the List of Ratings to be several screens long. In order to keep things organised, it is possible to collapse fully graded ratings groups. By clicking on the Done button next to the title of the rating group, Sapphire hides the ratings of the rating group and marks the rating group as completed.

The status of a rating group is indicated by a badge of a specific colour and with a specific icon in front of the rating group title. Blue badges with a clock icon represent either unstarted or unfinished rating groups, which is the default. Green badges with a checkmark icon denote finished rating groups. Red badges with a warning sign icon are used to mark rating groups where one or more ratings have been changed since the submission was last evaluated and need to be reviewed.

The ratings of rating groups needing review are always shown, regardless of whether the rating group is marked as done or not. A review is confirmed by clicking on checkmark button next to the changed rating. Once all ratings under review are confirmed, Sapphire reverts the status of the rating group to its previous state, either unfinished or done.

The Footer of the Single Evaluation interface provides a summary of the status of each rating group for the submission. Each rating group is represented by its corresponding badge. Clicking on a badge scrolls the browser to the corresponding rating group in the List of Ratings.

B.9.2 Bulk Grading

The Bulk Grading interface is the secondary grading interface of Sapphire and allows staff members to enter the results of multiple students at once, as shown in Figure B.21. Staff members are able to access the Bulk Grading interface via the Bulk Grading button in the Submissions Table, described in Section B.8.1. However, the Bulk Grading interface needs to be enabled by lecturers first for a particular exercise.

If the exercise supports multiple attempts, the attempt needs to be selected from the Select Attempt panel first. The Grading Table underneath provides the main functionality of the Bulk Grading interface. The first column of the table is responsible for searching for and selecting students or student groups, depending on the exercise type. A search is initiated by entering either the matriculation number or the name of the student, or the name of the student group. Sapphire starts searching for the student or student group as

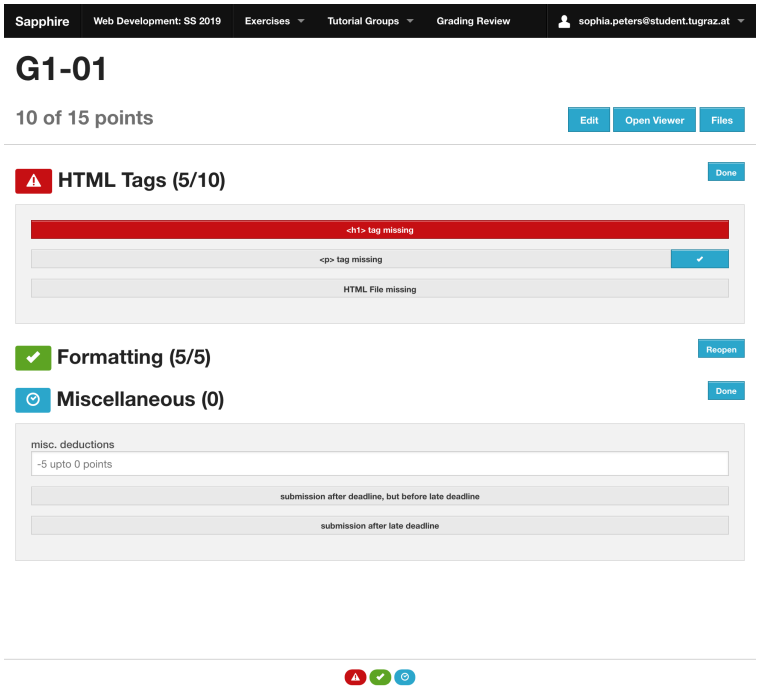

Figure B.20: The Single Evaluation interface allows staff members to grade submissions.

soon as the first digit or letter is entered. The staff member is able to select the student or student group from the dropdown showing the search results. Sapphire automatically appends another empty row to the Grading Table once a student or student group is selected. In order to change a previously selected student, the Pencil icon needs to be clicked, which reverts the corresponding table row to displaying the search field.

The remaining columns are used to display the ratings of the exercise. The name of each rating is displayed in the header row of the Grading Table. The inputs for each rating are displayed in the table cells of that column. Depending on the rating type, either an input field or checkbox is displayed as rating input, for variable and fixed ratings, respectively. Clicking on the red x button in the last column of the table removes of the corresponding table row.

The evaluations are saved to the database by clicking the Submit button. Sapphire automatically determines whether to update an existing evaluation or create a new submission for each row of the Grading Table.

B.10 Submission Viewers

Submission viewers are a vital part of the grading process. Sapphire provides the means for looking at the content of individual files from the Submission Tree interface. However, some submissions require a special environment to be viewed correctly. In particular, Sapphire provides a HTML Submission Viewer, described in Section B.10.1, and a CSS Submission Viewer, described in Section B.10.2.

B.10.1 HTML Submission Viewer

The HTML Submission Viewer provides a web-server like environment for viewing HTML files, as shown in Figure B.22. The toolbar in the top right of the interface allows users to quickly switch between the HTML files of a submission. Even though Sapphire is a web application, the HTML submitted by students is not directly displayed to the user. Instead, Sapphire parses the HTML file first and replaces relative

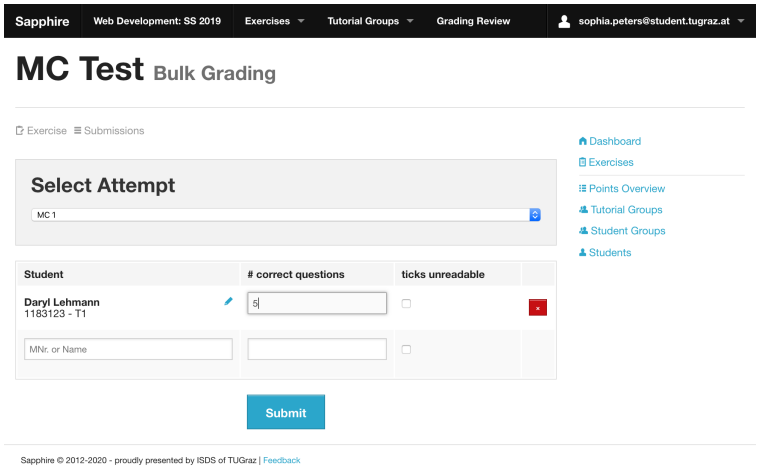

Figure B.21: The Bulk Grading interface is used to enter multiple results of externally graded exercises into Sapphire.

URLs with the internal URLs of the Sapphire system. Additionally, the resulting HTML is stripped of JS code to prevent Cross-Site Scripting (XSS) attacks. Thus, the HTML differs from the submitted file. Therefore, it is important not to rely solely on what is displayed in the HTML Submission Viewer for grading. Staff members are advised to download the files of the original submission, especially when the displayed HTML file looks strange.

B.10.2 CSS Submission Viewer

The CSS Submission Viewer provides the means to view alternative submitted CSS files in the context of a given HTML file. A predefined HTML template is used, to which the submitted stylesheets are applied, as shown in Figure B.23 for the Graz Times HTML template. Staff members are able to switch between the alternative CSS files of a submission using the toolbar in the top right corner of the CSS Submission Viewer.

B.11 Grading Review

The *grading review* is a special meeting at the end of term, where students are able to ask questions about the grading process and to query certain deductions. Staff members clarify misunderstandings and correct any errors.

B.11.1 Grading Review Dashboard

The Grading Review Dashboard serves as the starting point for a grading review with a specific student. Grading is a sensitive topic and it is vital not to disclose information about other students. Therefore, the Grading Review Dashboard only shows the Search Form by default, as shown in Figure B.24.

The specific student is selected using the Search Form. A staff member initiates a search by entering a student's name or matriculation number into the search field and clicking the Search button. Sapphire returns a table of matching students consisting of the name, matriculation number, and tutorial group, as

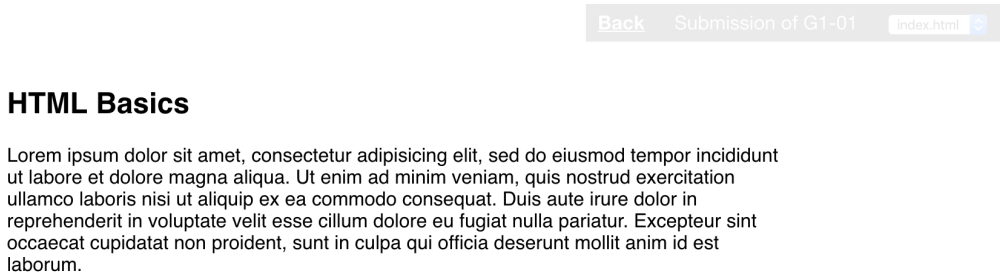

Figure B.22: The HTML Submission Viewer displays HTML files and included assets as if they were served from a web server.

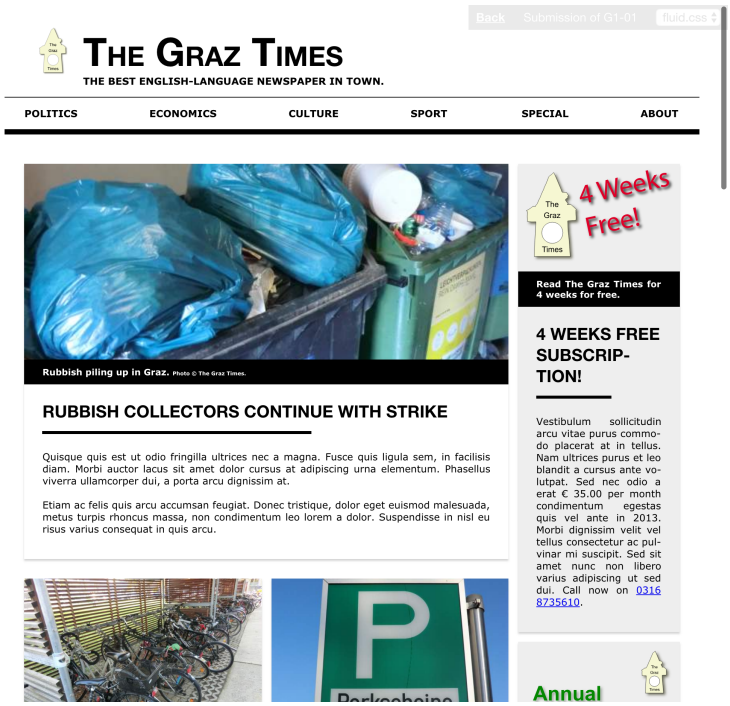

Figure B.23: The CSS Submission Viewer applies one of multiple submitted CSS files to a predefined template, such as the Graz Times template shown here.

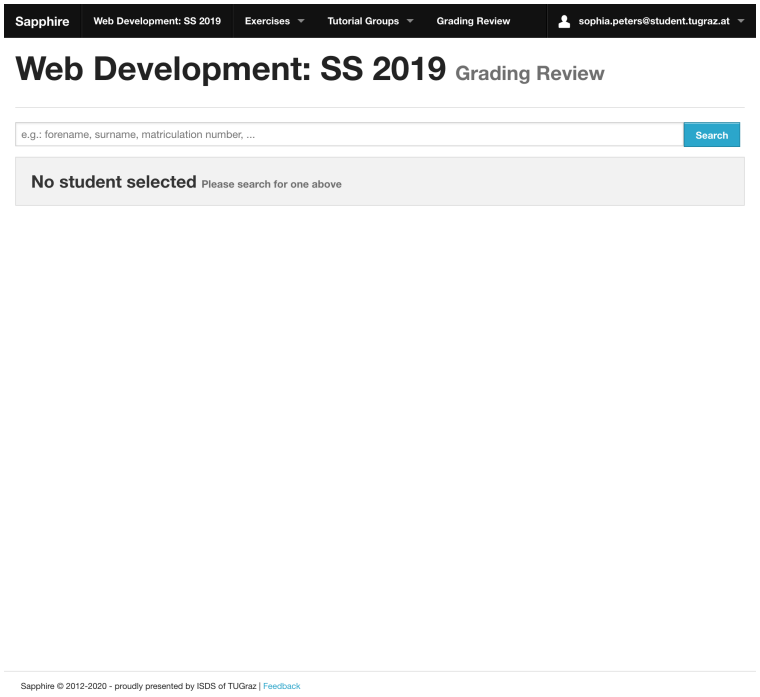

Figure B.24: The Grading Review Dashboard initially only shows a Search Form to avoid unnecessary disclosure of information.

shown in Figure B.25. A grading review is started by clicking the Show button in the relevant table row, which leads to the Grading Review Detail interface.

B.11.2 Grading Review Detail

The Grading Review Detail interface provides detailed information about the grading of a specific student. A panel of tabs is at the core of the interface, allowing users to quickly switch between the Overview Tab and an Evaluation Detail tab for each submission. When the Grading Review Detail interface is first accessed, Sapphire displays the Overview Tab. It shows an Overview Table of all of the submissions of a specific student, as shown in Figure B.26. For each submission, the Overview Table shows the associated exercise, submission date, and the points. In the footer of the Overview Table, Sapphire provides the preliminary grade and the total points achieved during the term.

By clicking on an Evaluation Detail tab, staff members are able to access a summary of the evaluation for each submission, as shown in Figure B.27. The list of rating groups along with applied ratings is shown on the left side of the interface. Sapphire provides three buttons beneath the Rating Groups List. The Reopen Evaluation button navigates the user to the Single Evaluation interface, described in Section B.9.1. The administrative interface of the submission is accessible through the Edit button, described in Section B.8.7. If a Submission Viewer is enabled for the exercise, Sapphire provides an Open Viewer button, which navigates the user to the corresponding Submission Viewer, as described in Section B.10. Inline versions of the submitted files are shown on the right side of the Grading Review Detail interface. Sapphire performs syntax highlighting for human readable files, such as HTML files. Additionally, a View Raw button is provided for each file, which allows users to view the raw content of the file.

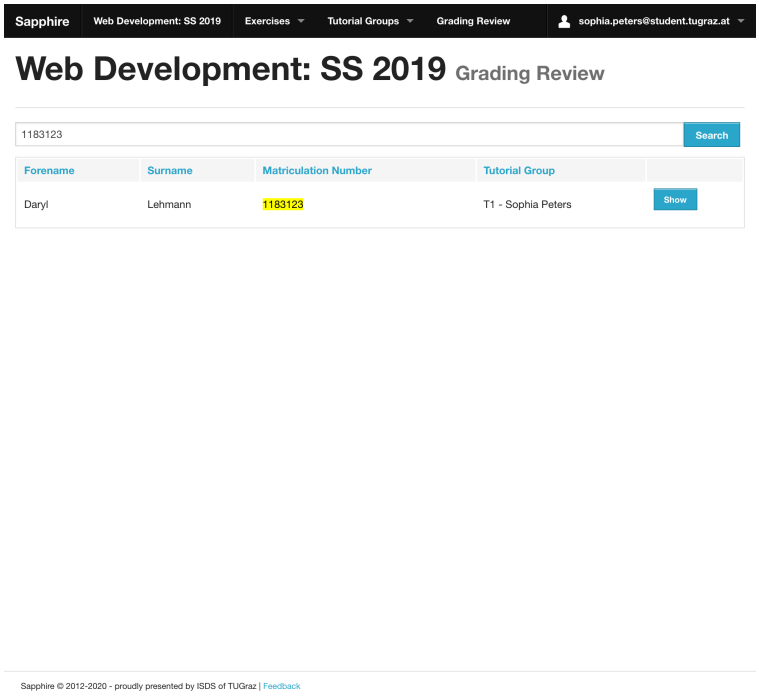

Figure B.25: Searching for a specific student in the Grading Review Dashboard.

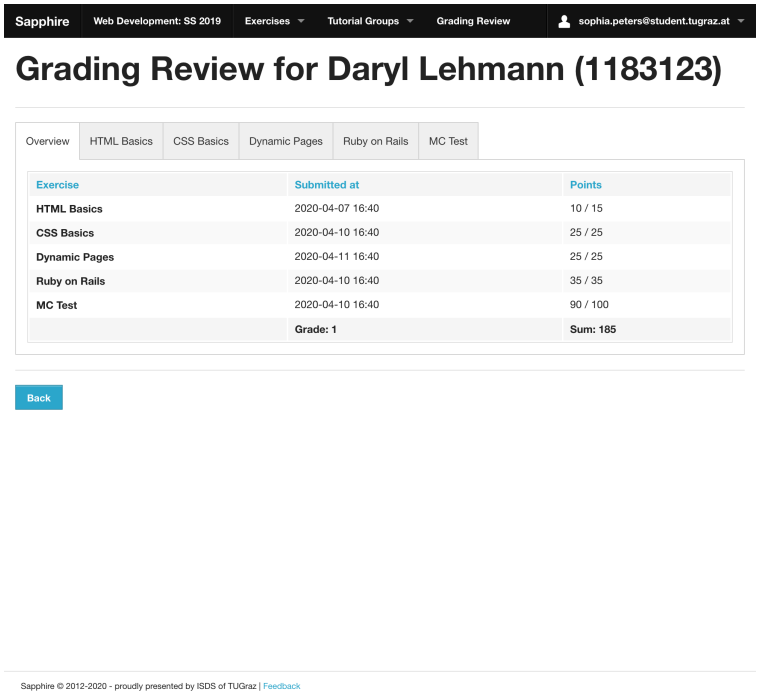

Figure B.26: The Overview Tab of the Grading Review Detail interface provides a summary of all submissions of a specific student. Each of the remaining tabs gives access to the evaluation of a particular submission.

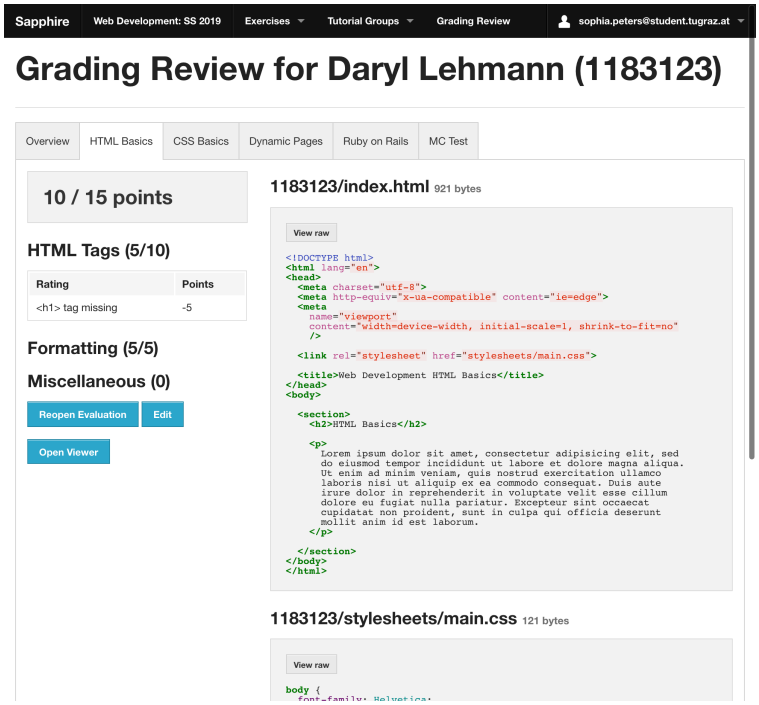

Figure B.27: The Evaluation Detail tab of the Grading Review Detail interface provides detailed information about the evaluation of a specific submission, here for the HTML Basics exercise.

B.12 Points Overview

Sapphire provides an overview of the overall results for a term, called the Points Overview, which is shown in Figure B.28. Staff members are able to access the Points Overview interface by clicking the Points Overview link in the sidebar of the term dashboard, described in Section B.3.

The Grade Distribution Table at the top left of the Points Overview interface provides a summary of the grade boundaries and the distribution of grades over all students in the term. For each grade, the points range, number of students, and the respective percentage is displayed. In addition, Sapphire lists a special *ungraded* grade, which represents the number of registered students who are ungraded, typically because they handed in no submission during the course of the term. The Exercises Table, located at the top right of the Points Overview interface, shows the name, minimum points required, and maximum points possible for each exercise. The total points possible (excluding bonus points) is shown in the last row of the Exercises Table.

Beneath the Grade Distribution Table and Exercises Table, Sapphire provides an overview of the results for each student, grouped by tutorial group. For each tutorial group of the term, a Points Table is shown. For each student of the tutorial group, a row is added to the Points Table, showing the matriculation number, points for each exercise, total points, and resulting grade. Some students might not have handed in a submission for every exercise. Missing exercises are denoted with *na*, which stands for *not available*.

In addition to the overall points overview for the entire term, Sapphire provides a dedicated Points Overview for each tutorial group of the term, which is accessible through the Points Overview link in the sidebar of the Tutorial Group Detail interface, described in Section B.4.2. The information provided is similar to the overall Points Overview. However, the grade distribution statistics are calculated based only on the students belonging to that tutorial group and Sapphire only provides a Points Table for that tutorial group.

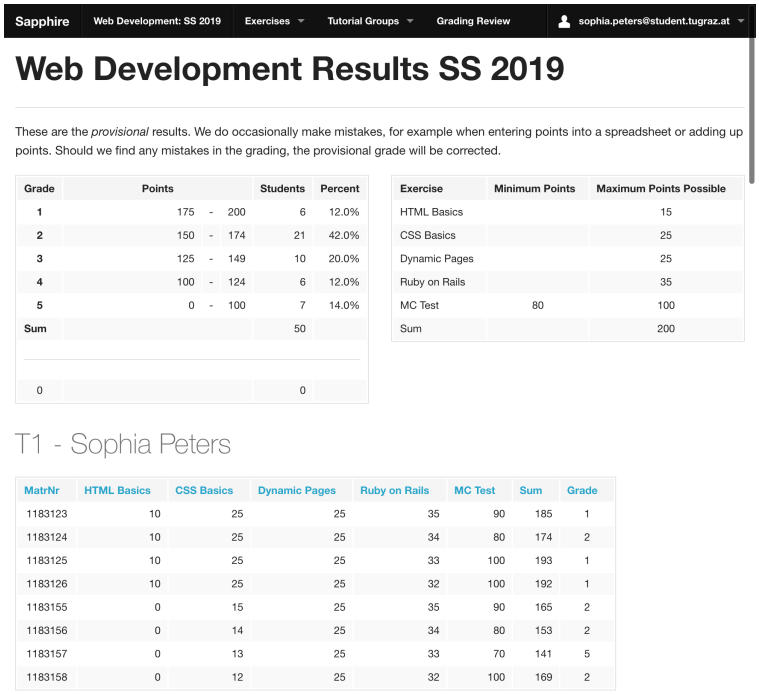

Figure B.28: The Points Overview interface provides an overview of the overall results for a term, including the points per exercise, total points, and grade for each student.

Appendix C

Lecturer Guide

Sapphire is an online course management and submission platform. This guide is intended for lecturers of university courses which use the Sapphire system, and provides insight into the most important features and typical workflows.

C.1 Authentication

User accounts are used to manage access to the Sapphire course management system. Authentication is the means for signing into user accounts, requesting a password reset, editing the main attributes of an account, and changing the password.

C.1.1 Login

The first step in using Sapphire is to log into a Sapphire account, using the Login Form shown in Figure C.1. Since lecturers are responsible for managing the access to Sapphire courses, an admin first needs to create a lecturer account.

First-time users of Sapphire must use the password reset feature to request a new password, as presented in Section C.1.2. The Password Reset Form is accessible through the Forgot your password? link beneath the Login Form. Previous users of Sapphire are able to reuse their credentials from a former term. Sapphire uses a role-based authentication system on a per-term basis. It is therefore possible to use a single Sapphire account for multiple terms.

C.1.2 Forgot Your Password

The Forgot Your Password allows users to reset their Sapphire account password. The process of resetting a user password is started by entering the email address of the account into the Password Reset Form and clicking the Reset button, as shown in Figure C.2. If the account is present in the database, Sapphire sends an email containing a password reset link. The user is allowed to choose a new password by visiting the included link, as shown in Figure C.3.

C.1.3 Changing Passwords

In Sapphire, passwords are changed on the user's profile page, which is accessible through the Edit Account link in the navigation bar. The user is presented with read-only meta-data as well as a form for changing the password, as shown in Figure C.4.

In order to change the password, the current password needs to be typed in first. Then, a new password is entered into the Password field, followed by the same password in the Password confirmation field. The

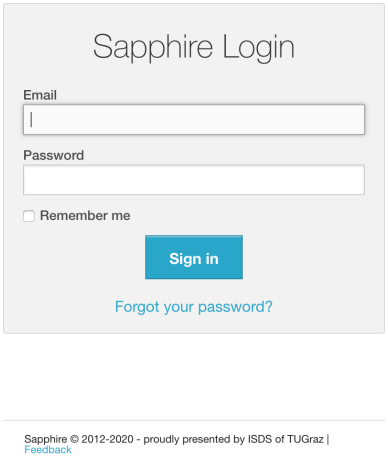

Figure C.1: The Sapphire Login Form.

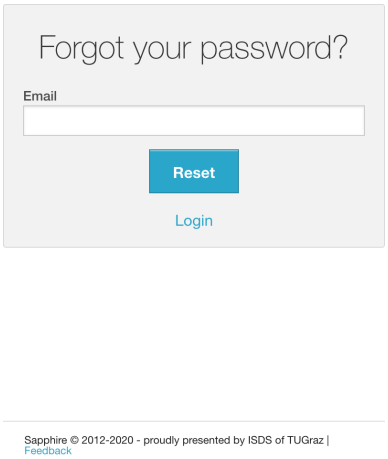

Figure C.2: The Forgot Your Password Form allows users to request a password reset link.

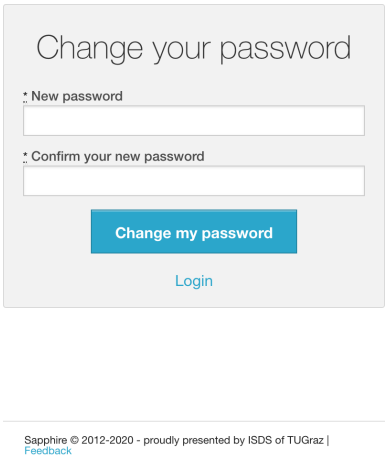

Figure C.3: The Change Your Password Form allows users to change their password after requesting a password reset link.

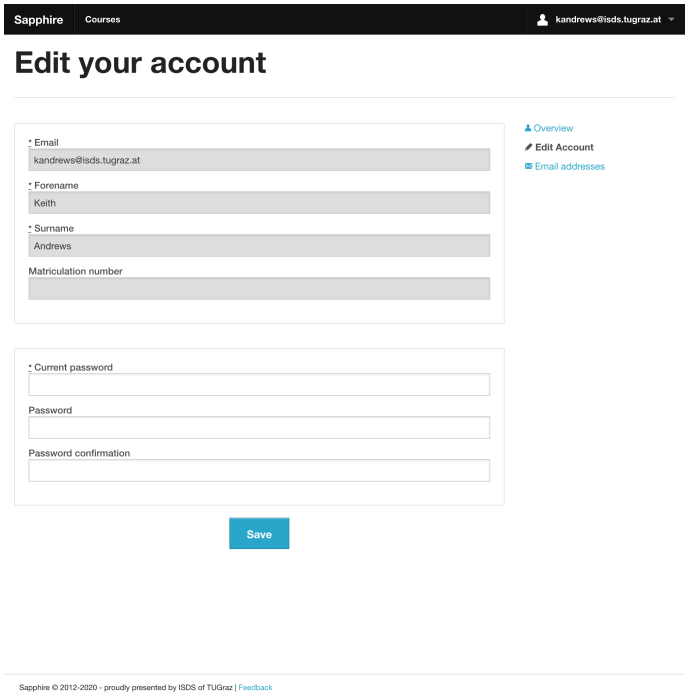

Figure C.4: Using the Edit Your Account Form to change the current password.

account password is updated once the user clicks on Save, as long as the current password is correct, and the password field matches the password confirmation field.

C.2 Courses Overview

Once the user signs into Sapphire, an overview of registered course terms is displayed in the Courses Overview, as shown in Figure C.5. Each row of the table consists of a link to the term dashboard, the number of exercises, the number of tutorial groups, and the number of students.

C.3 Term Dashboard

The Term Dashboard serves as the starting point for each term of a particular course. It displays of a list of events, providing an overview of the latest changes in Sapphire, as shown in Figure C.6. For lecturers, the following families of events are displayed:

- *Submission Events*: Sapphire keeps track of the time a student creates and updates a submission along with information about the changed files. Additionally, Sapphire tracks the time of extraction of uploaded ZIP files.
- *Result Publication Events*: During the course of the term, the lecturer publishes results, as the grading for each exercise is finished. Sapphire keeps track of these events and adds an event to the event feed.
- *Rating Group Events*: The points and titles of rating groups are subject to change throughout the term. Lecturers are able to create, update, and delete rating groups of an exercise. Sapphire keeps track of these changes to provide a transparent history of changes to staff members (lecturers and tutors).

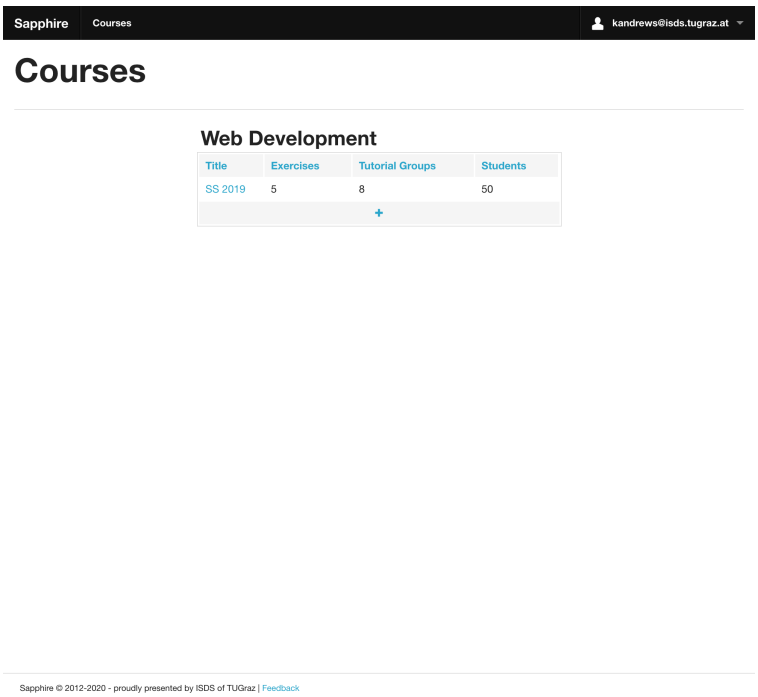

Figure C.5: The Courses Overview is displayed after logging into Sapphire, and contains a list of course terms the lecturer is registered for.

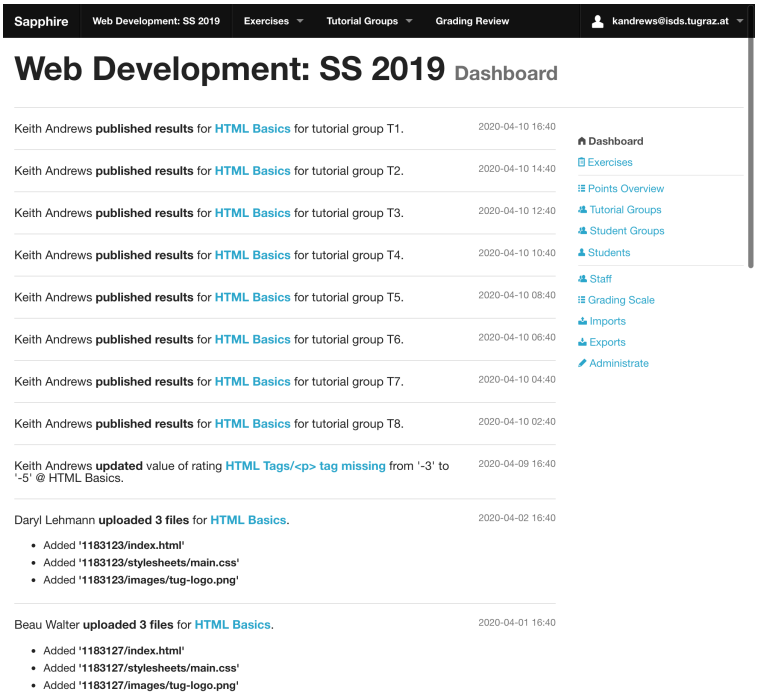

Figure C.6: The Term Dashboard shows the event feed.

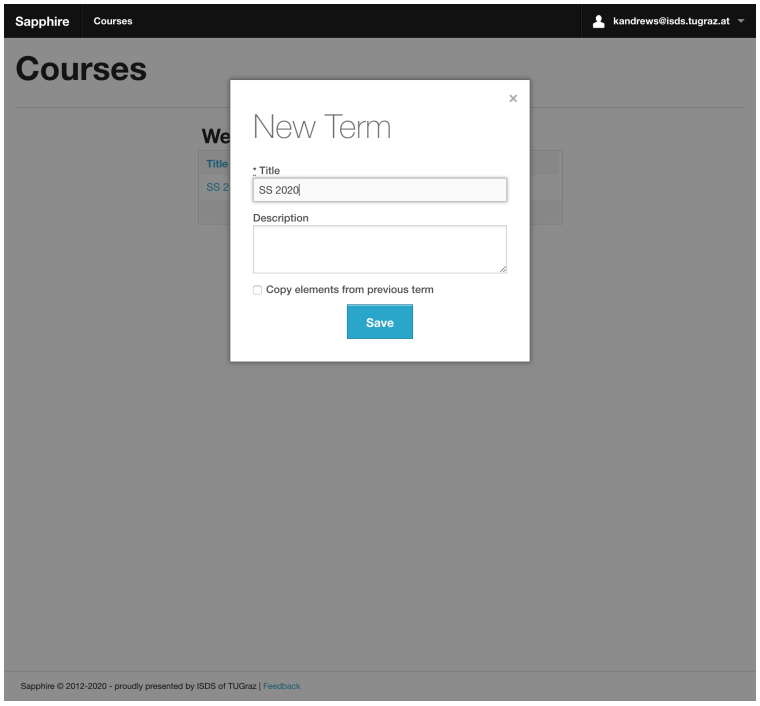

Figure C.7: Using the New Term Modal.

- *Rating Events*: Sapphire enables lecturers to create, update, and delete the ratings of a rating group. Sapphire keeps track of these changes to the database, similar to rating groups. As with rating group events, lecturers are kept informed about changes to the grading structure.

C.4 Terms

Lecturers are responsible for managing the terms of a course. A term is one instance of a course, held in a particular year, semester, or university term. This section describes the main aspects of the term feature in Sapphire.

C.4.1 Creating New Terms

To add new terms, a plus button is displayed at the bottom of the term table in the Courses Overview. Once clicked, the New Term Modal is displayed, shown in Figure C.7. It is mandatory to fill in the title of the new term. A previous term of the same course can be selected to serve as the basis for the new term by selecting the Copy elements from previous term option. A previously hidden panel is opened, containing additional options to copy the lecturer, exercises, and grading scale over to the new term, as shown in Figure C.8.

C.4.2 Editing Terms

Existing terms are updated via the Edit Term interface, shown in Figure C.9, which is accessed by clicking on the Administrate button in the sidebar. The form allows lecturers to change both the title and the description of a term. The changes are saved to the database by clicking the Save button.

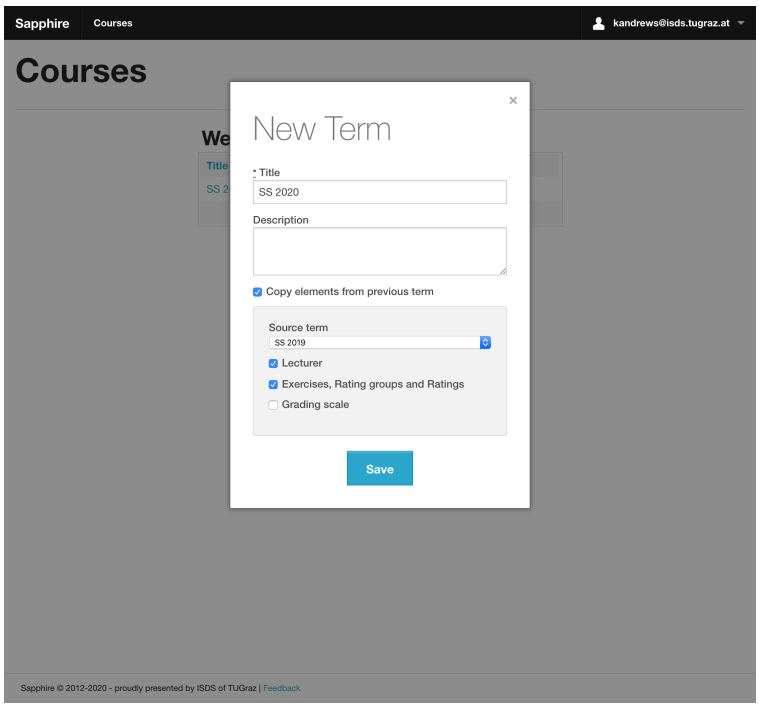

Figure C.8: Copying elements from a previous term using the New Term Modal.

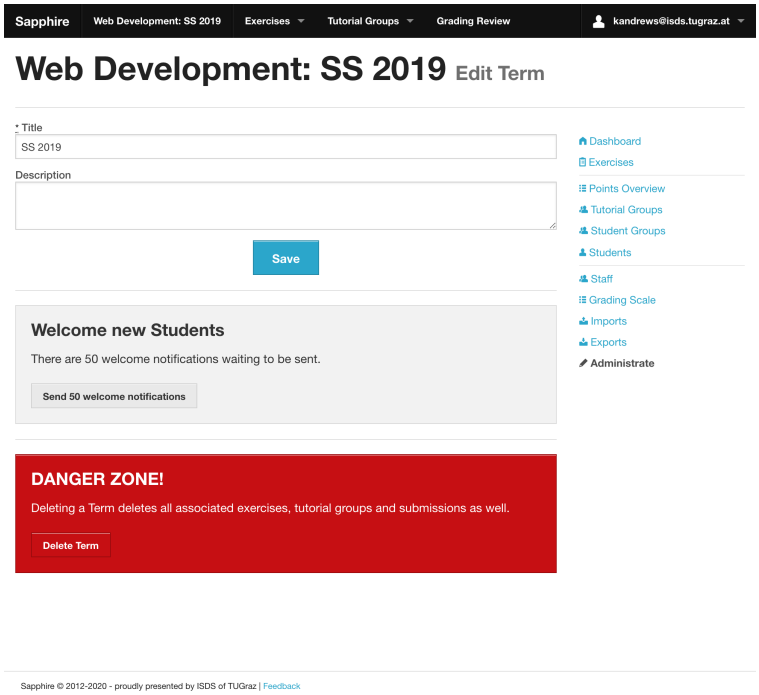

Figure C.9: The Edit Term interface is used to edit the title and description of a term, to send missing Welcome Notifications, and to delete the term along with its associated records.

C.4.3 Sending Welcome Notifications

There are two possible ways of sending Welcome Notifications by email to registered students. The first is to send Welcome Notifications as part of the import process, as described in Section C.8. The second is to send Welcome Notifications via the Edit Term interface of the term, as shown in Figure C.9. A click on the Send X welcome notifications button in the Welcome New Students section starts the process of sending pending Welcome Notifications. Sapphire keeps track of which Welcome Notifications have already been sent and proceeds with sending only those not yet sent.

C.4.4 Deleting Terms

It is possible to delete a term along with its associated records. The Delete Term button is located in the special Danger Zone section at the bottom of the Edit Term interface, as shown in Figure C.9. However, clicking this button needs to be carefully considered, since all data linked to the term is physically removed from the database and the hard disk. Restoring term data is only possible with the aid of a backup. The user is prompted for confirmation, before the term is actually deleted.

C.5 Tutorial Groups

Sapphire is designed to handle large university courses of hundreds of students. The number of students and submissions is usually too large to be handled by a single person. Therefore, each student is assigned to one of several tutorial groups, to split the workload of managing and grading students amongst several staff members.

C.5.1 Tutorial Groups Table

Sapphire provides an overview of the tutorial groups of a term in the Tutorial Groups table, as shown in Figure C.10. The table consists of the name of each tutorial group and the tutor or tutors responsible. Clicking on the tutorial group title navigates the user to the Tutorial Group Detail page.

C.5.2 Tutorial Group Detail

The Tutorial Group Detail page provides a table of students belonging to a particular tutorial group, as shown Figure C.11. The table displays the name, matriculation number, and email address for each student. Clicking on the Show button navigates the user to the Student Detail page, described in Section C.7.2

C.5.3 Creating Tutorial Groups

There are two ways to create tutorial groups in Sapphire. Firstly, tutorial groups are created during the import process, as described in Section C.8. Secondly, Sapphire allows lecturers to create new tutorial groups during the course of a term, by clicking the New Tutorial Group button in the tutorial groups table, shown in Figure C.12. Lecturers are required to specify the title of the tutorial group. It is also possible to specify an optional short description.

C.5.4 Editing Tutorial Groups

Sapphire allows lecturers to change both the title and the description of a tutorial group during the course of a term, as shown in Figure C.13. Lecturers are able to access the Edit Tutorial Group interface through the Administrate link in the sidebar of the Tutorial Group Detail page.

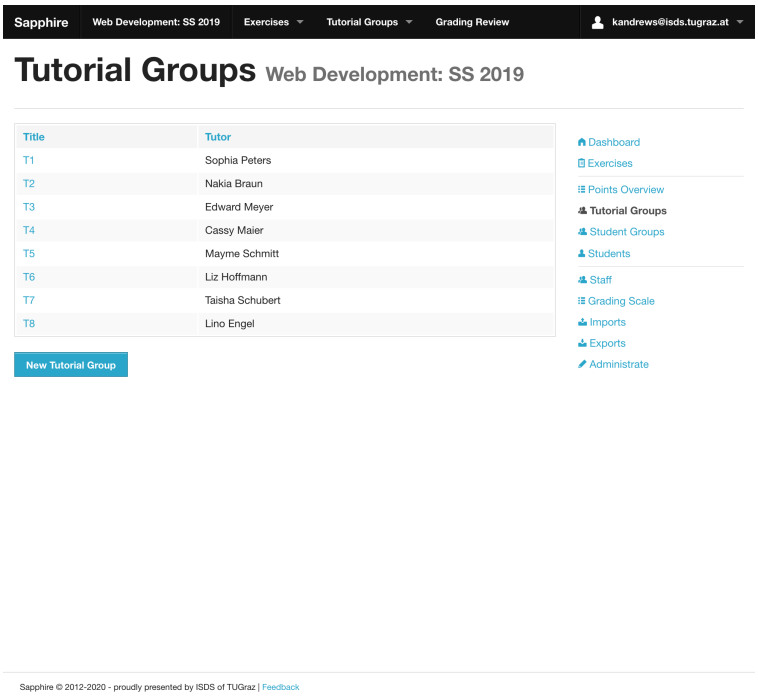

Figure C.10: The Tutorial Groups table provides an overview of the tutorial groups of a term.

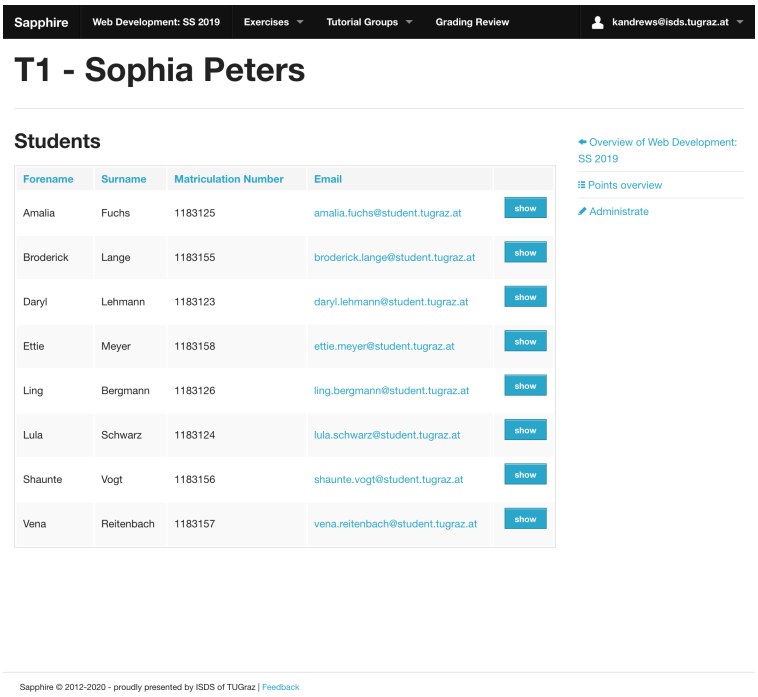

Figure C.11: The Tutorial Group Detail page provides a table of students of the tutorial group.

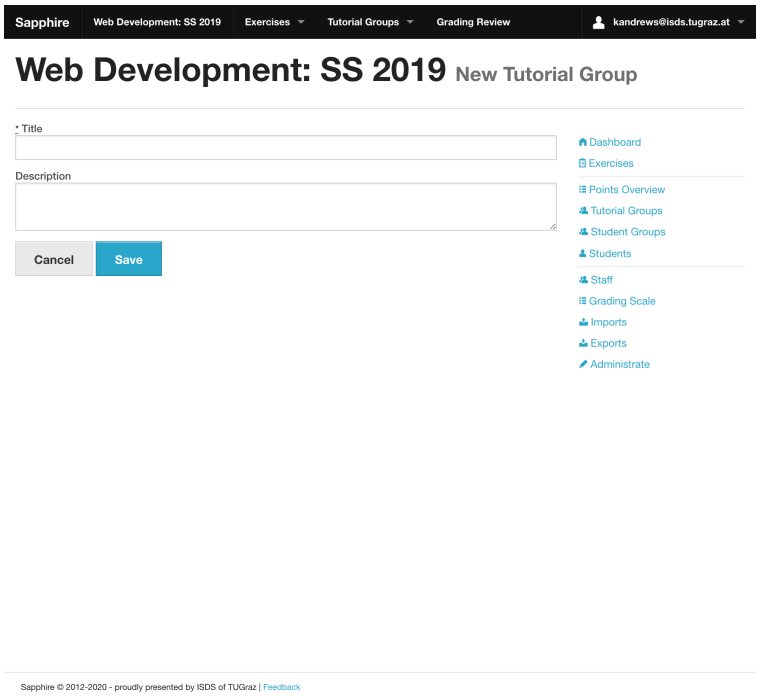

Figure C.12: Creating a new tutorial group during the course of the term.

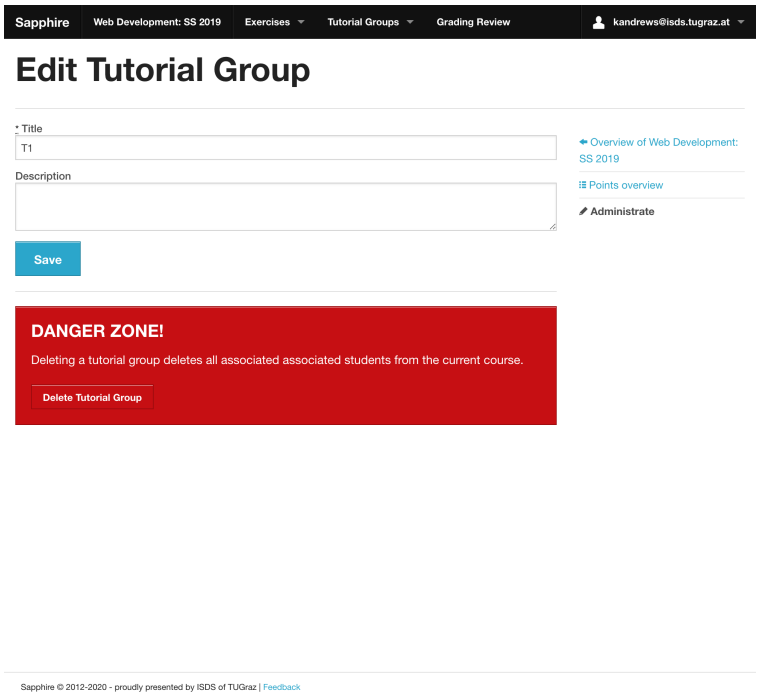

Figure C.13: The Edit Tutorial Group interface for editing a tutorial group.

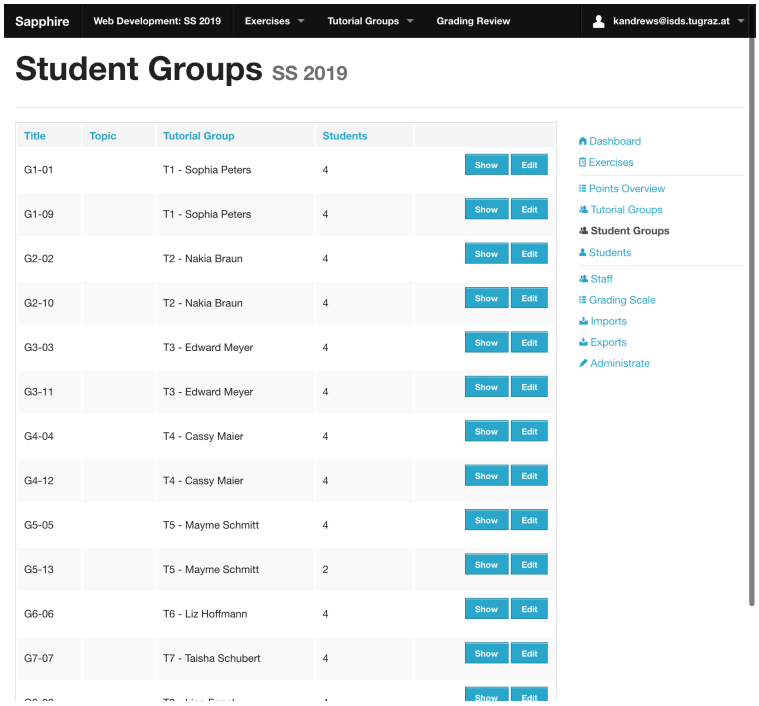

Figure C.14: The Student Groups table provides an overview of the student groups of a term.

C.5.5 Deleting Tutorial Groups

Lecturers are able to delete tutorial groups during the course of a term. The process is started by clicking the Delete Tutorial Group button in the Danger Zone section of the Edit Tutorial Group interface, as shown in Figure C.13. Deleting a tutorial group needs to be carefully considered, since students and tutors associated with the tutorial group will be removed from the term. However, the submissions of associated students will not be removed.

C.6 Student Groups

Sapphire supports submissions by groups of students via the concept of student groups. Student groups represent the current group constellation of students. Moving a student from one group to another during a term does not affect their association with any previous submissions as part of another group.

C.6.1 Student Groups Table

The Student Groups table provides an overview of the student groups currently available in a term, as shown in Figure C.14. The Student Groups table shows the title, topic, tutorial group, and number of students in each student group. For each student group in the table, the corresponding Show button navigates the user to the Student Group Detail page. Additionally, the table provides an Edit button, which navigates to the administrative interface of the student group, described in Section C.6.4.

C.6.2 Student Group Detail

The Student Group Detail page provides an overview of a student group, as shown in Figure C.15. The panel on top of the page provides information about the associated tutorial group and optional topic and keyword attributes.

A Students table is provided beneath the information panel, listing the students currently belonging to the student group. The Students table shows the forename, surname, matriculation numbers, and email

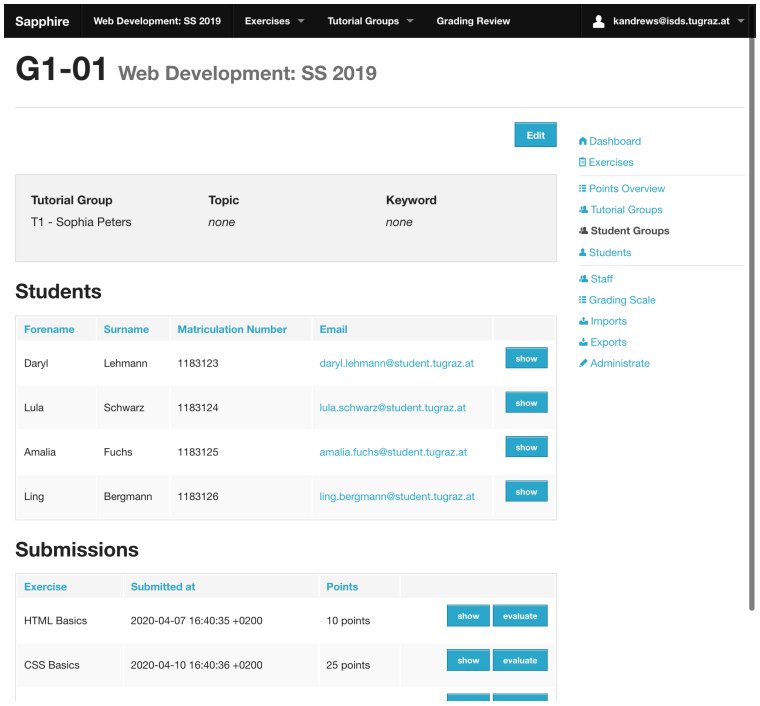

Figure C.15: The Student Group Detail page provides information about the attributes, students, and submissions of a student group.

addresses of each student in the group. A Show button allows users to navigate to the student detail page, described in Section C.7.2.

The Student Group Detail page concludes with the Submissions table, containing a list of group submissions by the student group. The Submissions table includes information about the exercise, date of submission, and resulting points. Individual submissions by group members are not included. Both Show and Evaluate buttons are displayed next to each submission. The Show button navigates the user to the file browser of the submission, described in Section C.11.2. The Evaluate button navigates the user to the submission evaluation page, described in Section C.12.1.

C.6.3 Creating Student Groups

Sapphire provides two mechanisms for creating student groups, similar to tutorial groups. Firstly, student groups are created during the student import process, as described in Section C.8. Secondly, Sapphire provides a New Student Group button in the student groups list.

In the New Student Group interface, a lecturer is able to configure a new student group, as shown in Figure C.16. Sapphire requires the lecturer to specify a title and a tutorial group, to which the new student group is assigned. Optionally, a topic and a keyword describing the student group can be specified. Both attributes are used during the submission export process, described in Section C.18.2. Additionally, lecturers are able to provide a short description of the student group in the corresponding field.

In order to add a student to the student group, lecturers first need to search for the relevant student in the search form, located to the right of the interface. Sapphire supports searching for both parts of the name as well as the matriculation number of the student. Once the relevant student is found, the student is added to the student group by dragging the corresponding record from the list of search results to the list of students in the left side of the interface. Sapphire creates the student group once the Save button is pressed.

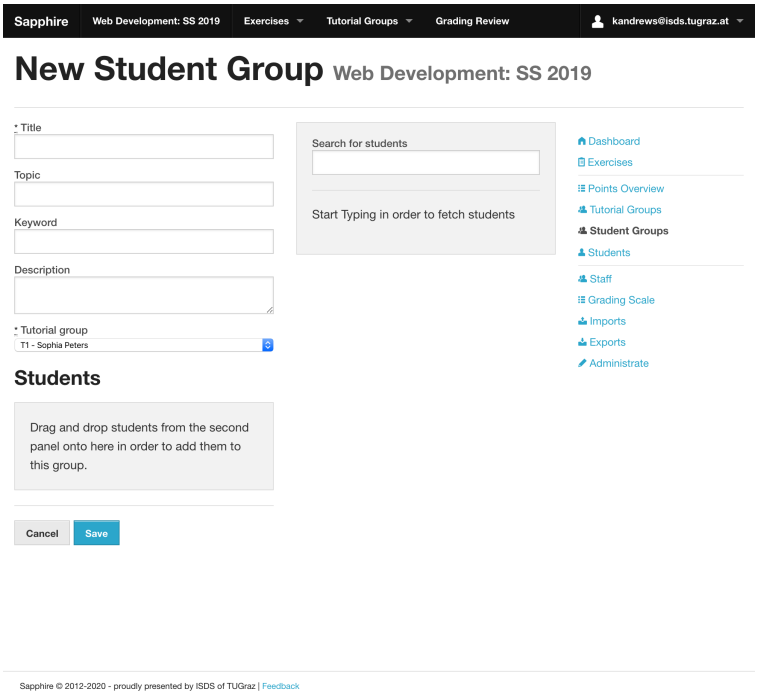

Figure C.16: Creating a new student group using the New Student Group interface.

Sapphire restricts students to membership of at most one student group per term. Students who are currently in a different student group are moved to the new group and removed from their previous group.

C.6.4 Editing Student Groups

Sapphire allows lecturers to make changes to student groups during the course of a term. Moving a student from one student group to another does not affect the previous submissions associated with the student. Lecturers are able to perform changes to the title, description, topic, keyword, and student assignments via the Edit Student Group interface of the student group, as shown in Figure C.17. The behaviour of the interface is very similar to that of the New Student Group interface, described in Section C.6.3 above.

C.6.5 Deleting Student Groups

Lecturers are able to delete student groups during the course of a term. The process is started by clicking the Delete Student Group button in the Danger Zone section of the Edit Student Group interface, described in Section C.6.4. Deleting student groups completely removes them from the database. During the removal process, Sapphire deletes references to associated submissions and current student group members will be left without a student group. However, neither the submissions nor the students will be removed from the database.

C.7 Students

Managing hundreds of students is one of the core features of Sapphire. Students are responsible for creating submissions during the course of a term and Sapphire enables students to view results throughout the term as they are published. Lecturers are responsible for managing students and student groups.

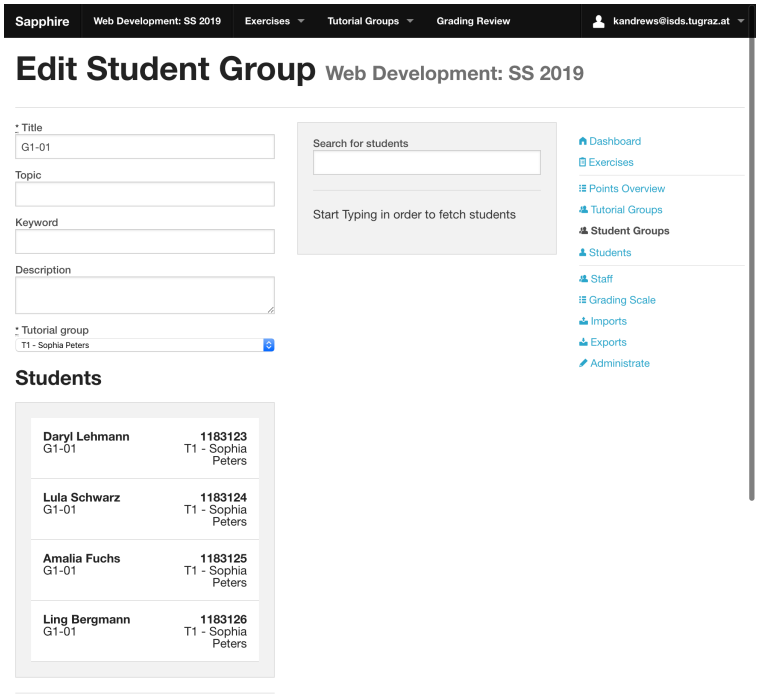

Figure C.17: The Edit Student Group interface of a student group.

C.7.1 Students Table

The Students table displays a list of students currently participating in a term, as shown in Figure C.18. The table provides the name, matriculation number, tutorial group, received points, and preliminary grade for each student. Detailed information about each student can be obtained clicking on the Show button in the corresponding row.

C.7.2 Student Detail

The Student Detail page provides detailed information about a particular student of a term, as shown in Figure C.19. Sapphire presents two buttons at the top of the page. The Edit button is used to navigate to the Administrate Student interface for the student, described in Section C.7.4. The Grading Review button is used to navigate to the Grading Review page of the student, described in Section C.16. The information panel located below displays the current total points, preliminary grade, tutorial group, and student group of the student.

Beneath the information panel, a table of submissions by the student is shown. For each submission, the exercise, date of submission, and resulting points are displayed. Next to each submission, the Show button is used to navigate to the file browser of the submission, described in Section C.11.2. The Evaluate button navigates to the submission's evaluation page, described in Section C.12.1.

C.7.3 Creating Students

Sapphire provides two ways to add new students to a term. The most common option is to import students via the import feature, described in Section C.8. The other option is to use the Add Student interface, shown in Figure C.20. Lecturers are required to search for the student account using the search form, which needs to be created first by an administrator. Lecturers will be able to create accounts themselves, once the Account Management feature is available to lecturers, as described in Section 11.1.9. Sapphire searches for students by name or matriculation number while typing in the search field. After selecting the appropriate student, the lecturer is required to specify the tutorial group to which the student belongs.

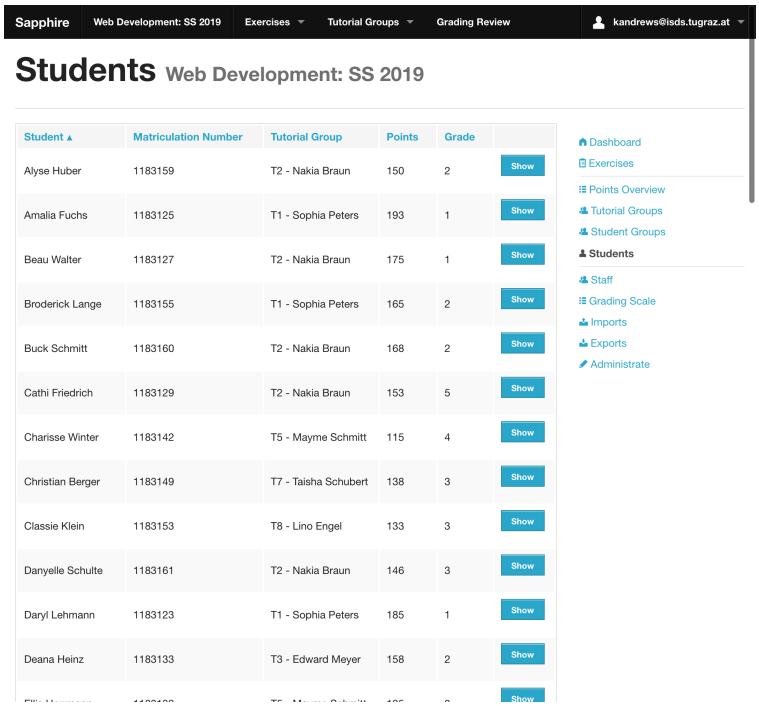

Figure C.18: The Students table displays a list of students participating in a term.

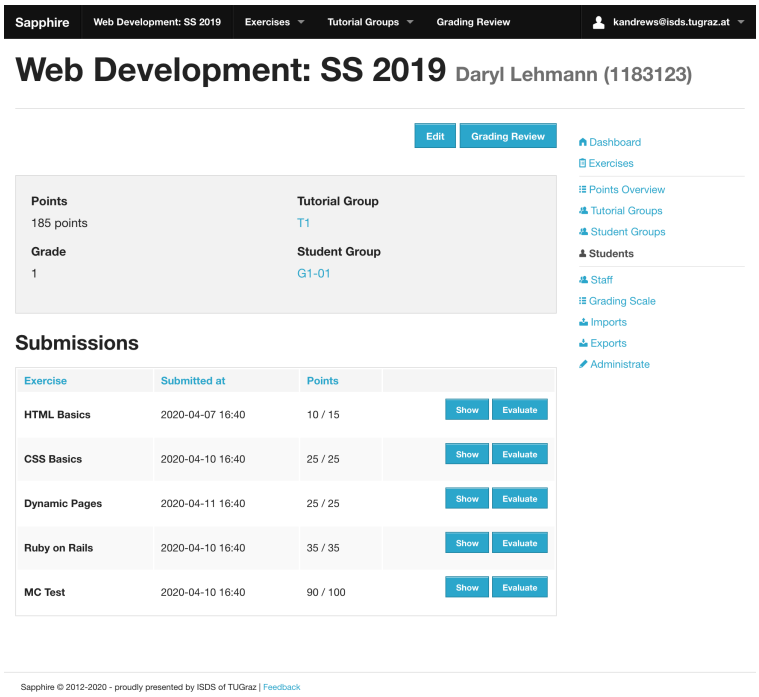

Figure C.19: The Student Detail page presents detailed information about a student of a term.

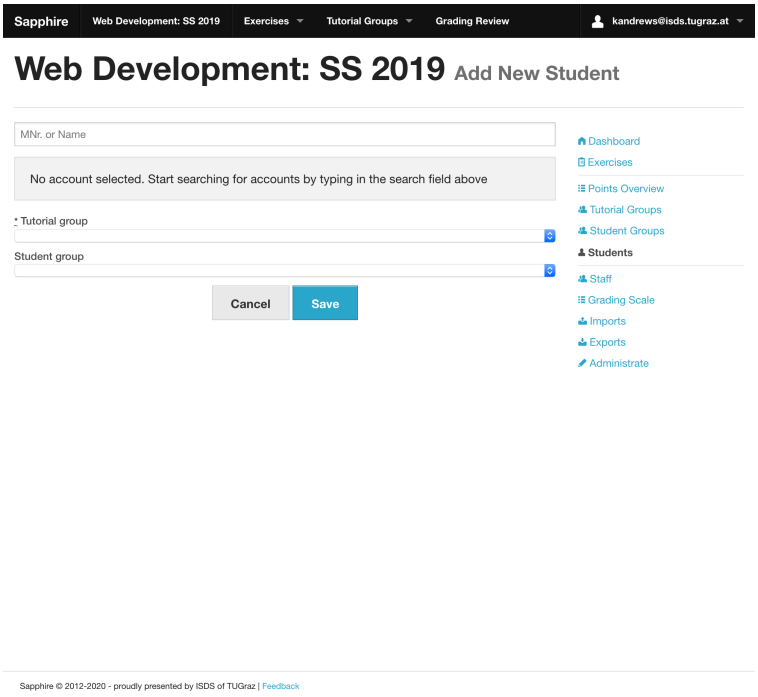

Figure C.20: The Add Student interface allows lecturers to manually add students to a term.

Additionally, an optional student group can be specified. Finally, the student is added to the term by clicking the Save button.

C.7.4 Editing Students

Sapphire enables lecturers to change the tutorial group and student group a student is assigned to via the Edit Student interface, shown in Figure C.21. While encouraged, it is not required for the student to belong to the same tutorial group to which the student group is assigned.

C.7.5 Deleting Students

The Delete Student button is located in the Danger Zone section of the Edit Student interface, as shown in Figure C.21. However, starting the deletion process needs to be considered carefully. Upon deletion, Sapphire completely removes the student record, along with its associations to submissions and the student group.

C.8 Importing Students

Sapphire encourages lecturers to import students from the TUGRAZonline campus management system at the beginning of each term. During this process, Sapphire creates tutorial groups, student groups, and student accounts based on a CSV upload file.

C.8.1 Existing Imports

The Import Students interface is accessible to lecturers via the Imports link in the sidebar of the term. The Import Students interface consists of two parts, shown in Figure C.22. The Existing Imports table is displayed on top of the page. It displays previously imported CSV files along with the date of creation, date of modification, whether Welcome Notification emails were sent, and the overall import status. The original

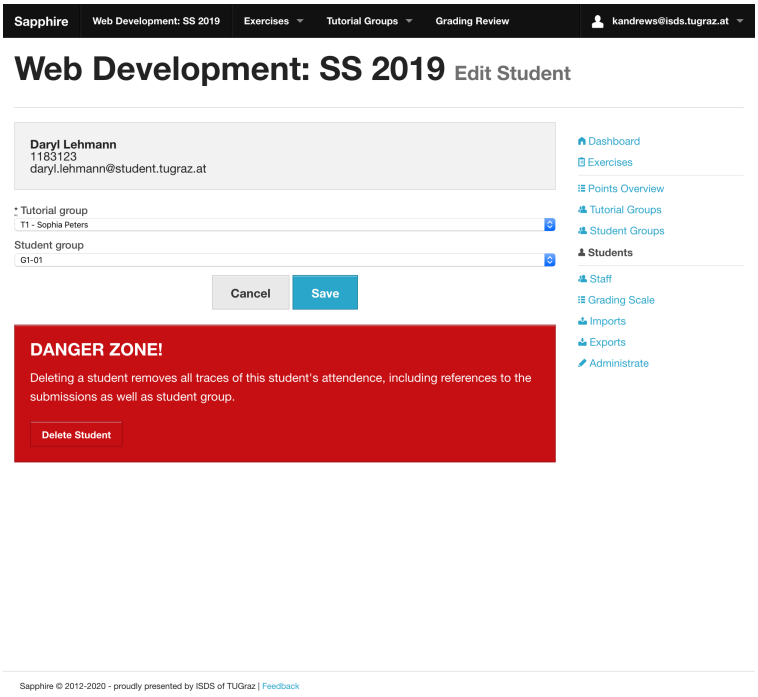

Figure C.21: The Edit Student interface for editing students.

CSV file is downloadable via a download button. Lecturers are able to review the configured column mapping, by clicking on the magnifier icon. By clicking the x button, lecturers are able to remove the CSV file from Sapphire to free up disk space on the server. While removing CSV files, Sapphire does not remove the associated student accounts from the database.

C.8.2 New Imports

The Import Students from CSV form is used to create new imports, and is displayed beneath the Existing Imports table. First, lecturers are required to download a CSV export of the students of the term from the TUGRAZonline system, which handles student registrations. A snippet of a sample CSV export from TUGRAZonline is shown in Listing C.1. The CSV file then needs to be selected in the File field of the Input File section.

Next, the lecturer must decide if Sapphire should create student groups in addition to tutorial groups, by choosing either the Match tutorial groups or Match tutorial and student groups option. Underneath each option, users are able to customise the regular expression used to parse the tutorial group and optional student group identifiers. When choosing the Match tutorial groups option, Sapphire, matches tutorial groups in the form of `TX` by default, where `X` is an integer number. When choosing the Match tutorial and student groups, Sapphire defines a default format to be `GX-Y`, where `X` is an integer denoting a tutorial group and `Y` is an interger denoting the student group number.

Furthermore, users are able to change the CSV parsing behaviour of Sapphire in the File Format Settings section. Sapphire supports the following options:

- *Headers on first line*: Sapphire parses the first line of the CSV as headers and does not try to translate the first line into database records. This option is true by default.
- *Column separator*: The character used to split the CSV line into columns, a semicolon by default.
- *Quote character*: The character used to quote CSV values, a double quote by default.

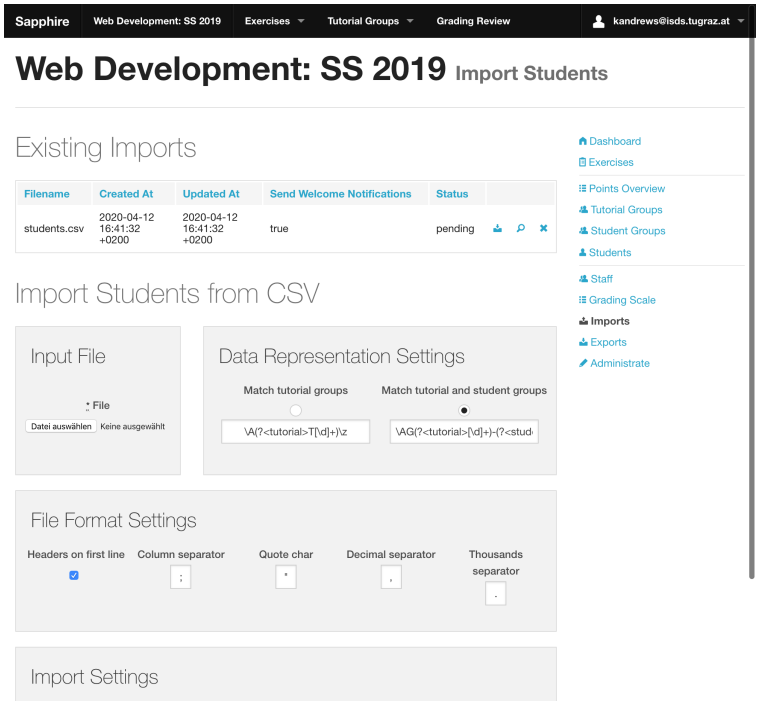

Figure C.22: The import section of Sapphire consists of the Existing Imports table and the Import Students from CSV Form.

- *Decimal separator*: The character used to denote decimals, a comma by default.
- *Thousands separator*: The character used to separate groups of thousands, a dot by default.

Once the initial settings have been configured, the configuration of the import process is continued by configuring the mapping from CSV columns to the internal data representation of Sapphire. Upon clicking the Next button, the lecturer is navigated to the Column Mapping Editor, shown in Figure C.23. Sapphire displays the uploaded CSV in tabular form using the parsing settings specified in the previous step. Sapphire makes an educated guess as to the column mappings. Lecturers are then required to review the automatically assigned columns, since the format of CSV exports from the TUGRAZonline system might have changed. Sapphire allows the specification of the following columns:

- *Group*: The Group column is used during the matching phase to create both tutorial groups and optional student groups.
- *Email*: The Email column is used to uniquely identify a student account.
- *Matriculation Number*: The Matriculation Number column specifies the matriculation number of a student. This column is used as a fallback for looking up existing student accounts, in case an email address is not matchable.
- *Forename*: The Forename column specifies the forename of the student.
- *Surname*: The Surname column specifies the surname of the student.
- *Comment*: The Comment column is optional. It is solely used during the import process to provide problem descriptions, in case Sapphire fails to parse a line of the CSV.

Finally, users are able to start the import process by clicking the Import button. An information page is shown by Sapphire during the course of the import. Sapphire provides statistics about progress and how many student accounts, student groups, and tutorial groups have been created.

```
1 Gruppe;lfd.Nr.;Platz;Familien- oder Nachname;Vorname;incoming;Matrikelnummer;
      Kennzahl;Studium;Semester im Studium;Anmeldedatum;E-Mail;Anmerkung
2 G1-01;1;fix;Lehmann;Daryl;N;1183123;9631247191;F 033 521 (UG2002)
      ;2;10.10.2018,11:02;daryl.lehmann@student.tugraz.at;""
3 G1-01;2;fix;Schwarz;Lula;N;1183124;5859547125;F 033 521 (UG2002);1;06.10.2018,02:11;
      lula.schwarz@student.tugraz.at;""
4 G1-01;3;fix;Fuchs;Amalia;N;1183125;5142986159;F 033 521 (UG2002);3;08.10.2018,01:42;
      amalia.fuchs@student.tugraz.at;""
5 G1-01;4;fix;Bergmann;Ling;N;1183126;6868936296;F 033 521 (UG2002)
      ;3;14.10.2018,09:54;ling.bergmann@student.tugraz.at;""
```

Listing C.1: A typical CSV export file created by the TUGRAZonline system.

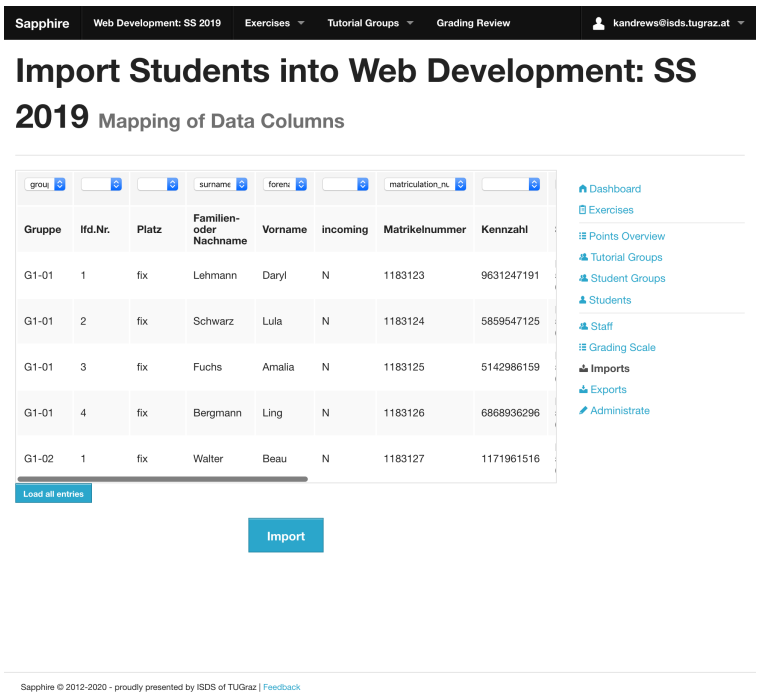

Figure C.23: The Column Mapping Editor is used to configure the mapping of columns in the CSV file to the internal representation of Sapphire.

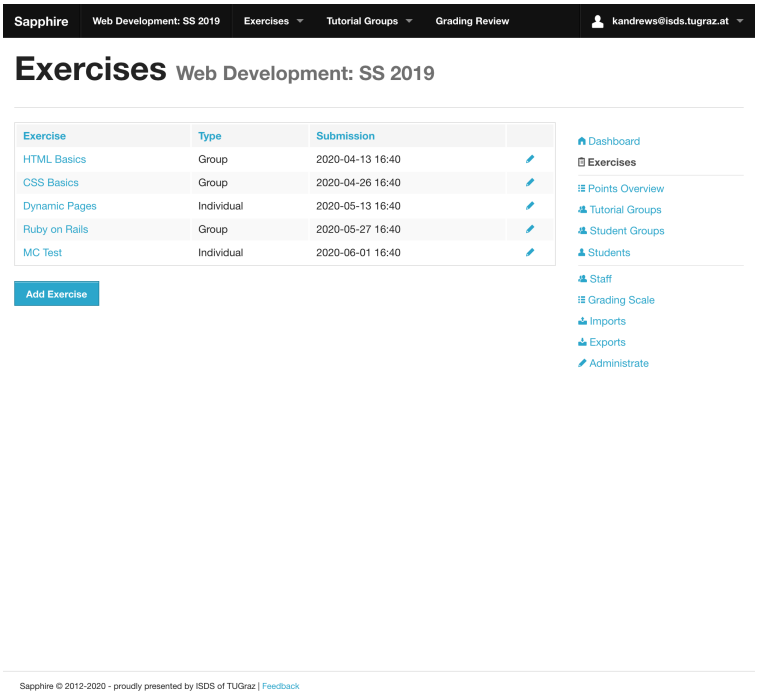

Figure C.24: The Exercises Table provides an overview of the exercises belonging to a term.

C.9 Exercises

Exercises serve as the basis for submissions in Sapphire. Each exercise is worth a specific number of points, which sum up to the total points of the term. There are two types of exercise: individual exercises and group exercises. Each exercise is configured to have a deadline by which the submissions needs to be finished. Optionally, a lecturer is able to configure a late deadline, after which changes to submissions are prohibited. It is therefore possible for students to change a submission after the regular deadline, but before the late deadline. Sapphire considers the time of the last change to be the time of submission. Some exercises require uploading files to the Sapphire system, while other exercises are managed by staff members (lecturers and tutors).

C.9.1 Exercises Table

The Exercises Table provides an overview of all the exercises belonging to a term, as shown in Figure C.24. The Exercises Table includes the name of the exercise, its type, and the submission deadline. The user is navigated to the corresponding detail page by clicking on the exercise title.

C.9.2 Exercise Detail

The Exercise Detail page provides a summary of the attributes of an exercise, as shown in Figure C.25. The information includes the exercise description, a URL to the instructions, the achievable points, the type of the exercise, maximum upload size, minimum points required, deadline, and late deadline. Empty attributes are hidden from the interface to improve readability.

C.9.3 Creating Exercises

Sapphire enables lecturers to create new exercises. Clicking the Add Exercise button opens the New Exercise Form, shown in Figure C.26. The form allows the following attributes to be specified for an exercise:

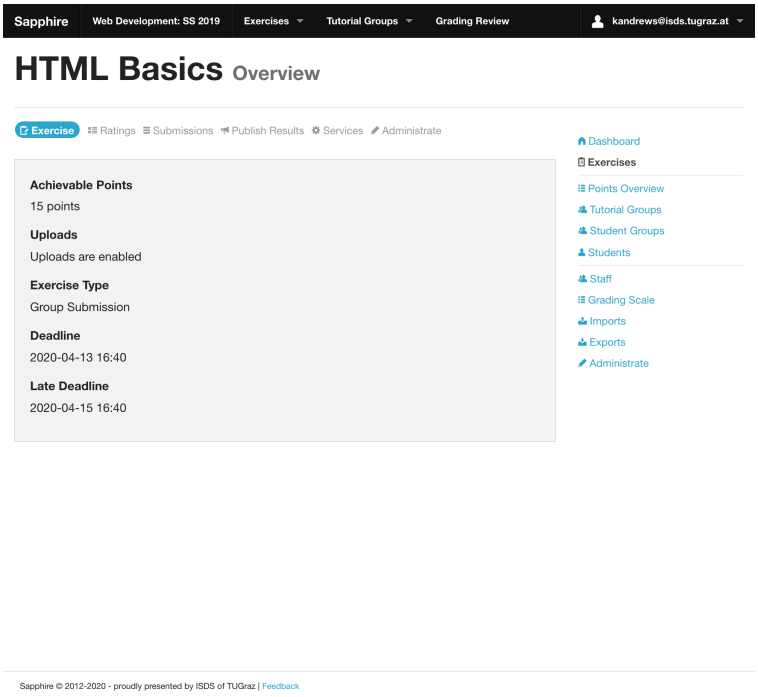

Figure C.25: The Exercise Detail page provides a summary of the attributes of an exercise.

- *Title*: The title of the exercise. The title is used in many places of the UI of Sapphire. It should be both short and descriptive.
- *Description*: The description field is used to provide concise instructions to the exercise. The intended use-case of this field is to give students and staff members context about the key learnings of an exercise, in case the name of the exercise is not self-explanatory.
- *Instructions URL*: The instructions URL allows lecturers to specify an external URL with detailed exercise instructions.
- *Submission Deadline*: Sapphire allows lecturers to specify the deadline by which an exercise must be submitted. The deadline is shown to students in both the Exercises Table and the Exercise Detail page. When specified, Sapphire prohibits students from making further changes to submissions after the deadline has passed.
- *Late Submission Deadline*: Some courses allow submissions after the submission deadline up to a certain point in time (but which then typically incur a points deduction). The late deadline is used to override the point in time uploads to submissions are prohibited and students are no longer able to modify their submissions.
- *Maximum Points shown in UI*: Sapphire implements a flexible grading system based on rating groups and ratings. Usually, each rating group is assigned a specific number of points. Ratings are used to deduct from the initial points assigned to the rating group. The sum of points provided by each rating group is used to calculate the maximum points for each exercise. However, some exercises need a different approach to grading. Multiple choice tests, for example, might be based on the number of correct questions, each worth a specific number of points. In this particular case, the sum of points assigned to the rating groups would be zero and Sapphire would wrongly indicate zero points as the maximum for this exercise. In order to account for this problem, lecturers are able to override the maximum points shown in the Sapphire UI. If left blank, Sapphire uses the internally calculated points instead. The developers of Sapphire are aware that this is a suboptimal solution

which needs to be changed in a future version of Sapphire.

- *Enable student uploads*: The Enable student uploads flag specifies whether student uploads for an exercise are currently allowed or not. For example, exercise submissions might be handed in to the lecturer in person, or uploads might be manually enabled two weeks before the submission deadline. In a future version of Sapphire, this flag could be replaced with a date-field to specify a point in time from which onwards uploads should be allowed to Sapphire.
- *Group Submission* or *Individual Submission*: Sapphire supports both group submissions and individual submissions. The radio button allows the lecturer to specify the type of submission for the exercise.
- *Minimum points required for positive grade*: Some exercises require students to achieve a minimum number of points in order for them to receive a positive grade at the end of the term. This field allows lecturers to specify a threshold of points on a per-exercise basis.
- *Maximum points in total*: While ratings are usually used for deductions, it is also possible to define bonus points ratings. In order to introduce an upper limit of points per exercise, lecturers are able to specify a maximum total number of points using this field.
- *Maximum upload size*: When enabling student uploads, it is good practice to specify an upper file size limit per submission. It allows estimation of the maximum storage space needed for a particular exercise and term and enables administrators to increase the storage space before problems arise.
- *Enable bulk operations*: The Bulk Operations interface is a dedicated interface for entering multiple externally graded submissions at once. It is particularly useful for entering the results of paper-based examinations.
- *Submission viewer identifier*: This select box allows lecturers to specify a submission viewer for an exercise. Two submission viewers are currently available. The HTML Submission Viewer is used for displaying HTML-based reports. The legacy CSS Submission Viewer is useful for grading alternative CSS files applied to a given HTML file.
- *Enable multiple attempts*: This option allows lecturers to define multiple attempts per exercise. Each attempt is comprised of a title and a date. Examinations, for example, might be attempted more then once during the course of a term. The Multiple Attempts feature is tightly integrated with the Bulk Operations feature.

C.9.4 Editing Exercises

The Edit Exercise interface for an exercise allows lecturers to edit the attributes of the exercise, as shown in C.27. The interface is accessible through both the pencil icon in the Exercises Table, described in Section C.9.1, and through the Administrate link in the sidebar of the Exercise Detail page. The Edit Exercise interface works in a similar way to the New Exercise Form described in Section C.9.3.

C.9.5 Deleting Exercises

The Delete Exercise button is located in the Danger Zone of the Edit Exercise interface of the exercise. Deleting an exercise physically removes all related records from the database as well as the file system. Therefore, deleting an exercise is potentially dangerous and needs to be considered carefully.

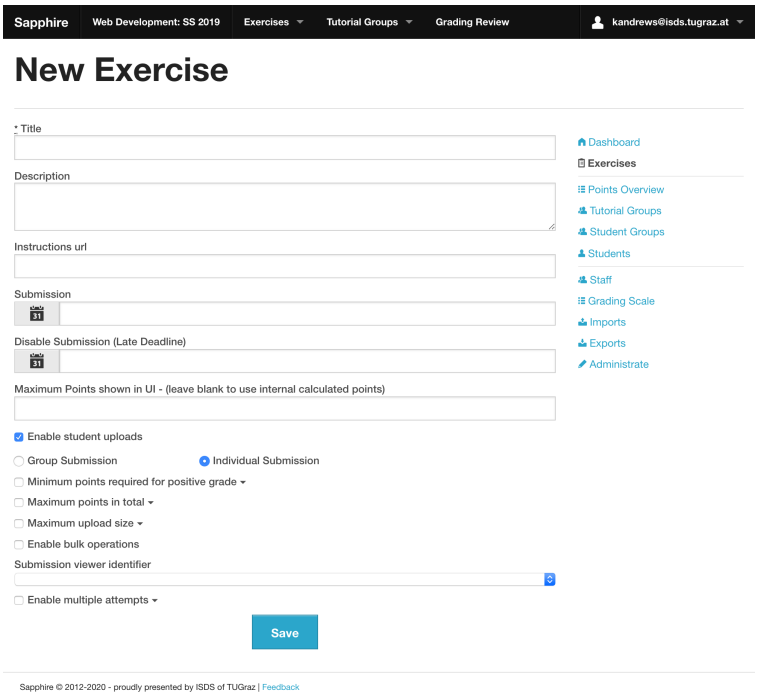

Figure C.26: Creating a new exercise with the New Exercise Form.

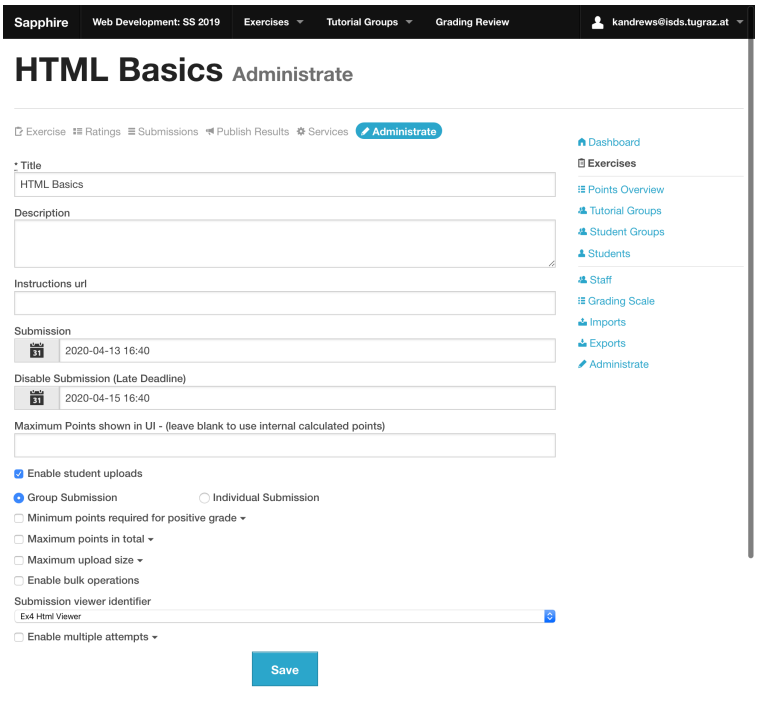

Figure C.27: The Edit Exercise interface for an exercise.

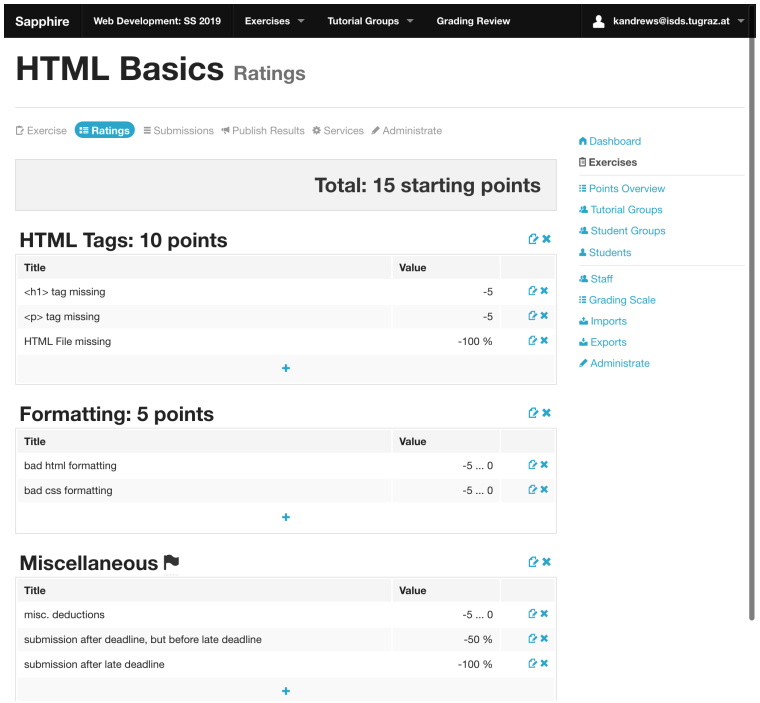

Figure C.28: The Ratings Editor interface displays the ordered list of rating groups for an exercise and their associated ratings.

C.10 Rating Groups and Ratings

Rating groups and ratings are the basis for grading submissions. Lecturers are able to configure the set of rating groups and ratings through the Ratings Editor interface, shown in Figure C.28. In Sapphire, each rating group is assigned a specific number of starting points. Ratings are used to modify the starting points provided by the rating group, with either associated deductions or bonus points.

At the top of the Ratings Editor interface, the total number of starting points of all the rating groups in the exercise is shown, allowing lecturers to quickly identify the number of points an exercise is worth. Underneath, the List of Rating Groups displays the ordered list of rating groups with their associated ratings.

C.10.1 Rating Groups

For each rating group, the List of Rating Groups displays the title, points, and a table of ratings. In addition, Edit and Delete buttons are shown next to the title of each rating group, as can be seen in Figure C.28. New rating groups can be added to an exercise by clicking the Add Rating Group button at the very bottom of the Ratings Editor. Sapphire displays the New Rating Group Modal, shown in Figure C.29. New rating groups are appended to the bottom of the List of Rating Groups. Rating groups can be reordered as desired using drag-and-drop.

Lecturers are able to specify the title, global status, starting points, point range, and description of the rating group. Global rating groups behave differently to non-global rating groups, in that percentage deductions and bonus points are applied to the exercise as a whole, rather than being limited to the specific rating group. Global rating groups are marked with a flag icon in the List of Rating Groups. By default, the points of a rating group range from zero to the starting points. By clicking the Change Pointrange link, a new panel is opened, allowing lecturers to change the default range. The Save button needs to be clicked, in order to persist any changes to the database.

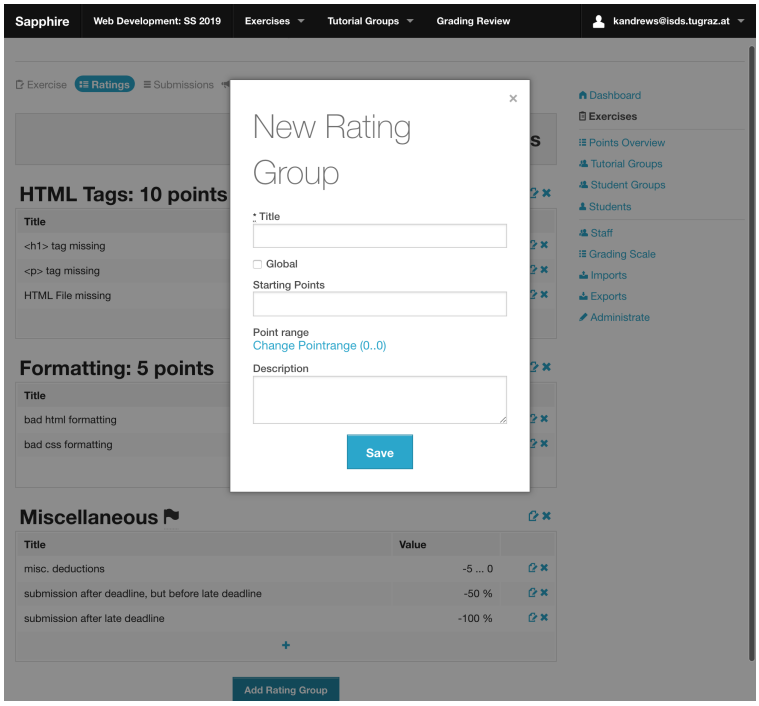

Figure C.29: The New Rating Group Modal is used to add new rating groups to an exercise.

Clicking the Edit button next to a rating group's title displays the Edit Rating Group Modal, shown in Figure C.30. Its functionality is very similar to that of the New Rating Group Modal. A rating group is deleted by clicking the Delete button next to its title. Any associated ratings are also deleted.

C.10.2 Ratings

The List of Rating Groups in the Ratings Editor displays a Ratings Table for each rating group, as can be seen in Figure C.28. The Ratings Table displays the title and value of each rating in the rating group, along with Edit and Delete buttons.

A lecturer can add a new rating by clicking the Plus button at the bottom of the Ratings Table, which opens the New Rating Modal, shown in Figure C.31. Fields are provided for the title and description of the rating. The type of rating is specified in the corresponding dropdown. Depending on the type of rating, additional options are shown below the rating type field. Sapphire currently supports nine different rating types, as shown in Table C.1. Checking the Display this rating during bulk operation option displays the rating as part of the Bulk Grading interface, described in Section C.12.2. Automated checking capabilities are enabled by specifying an Automated checker identifier.

Lecturers are able to change existing ratings using the Edit Rating Modal, by clicking on the Edit button of the corresponding rating. It is shown in Figure C.32. The functionality provided by the Edit Rating Modal is equivalent to that of the New Rating Modal. Sapphire allows lecturers to reorder ratings by drag-and-drop, similar to rating groups. Moving a rating into another rating group is done similarly by dragging the rating from one Rating Table and dropping it into another. A rating is deleted by clicking the Delete button next to its title.

C.11 Submissions

During the course of a term, students are required to take part in exercises. Usually, submissions are managed by students themselves. However, Sapphire also allows staff members to manage submissions,

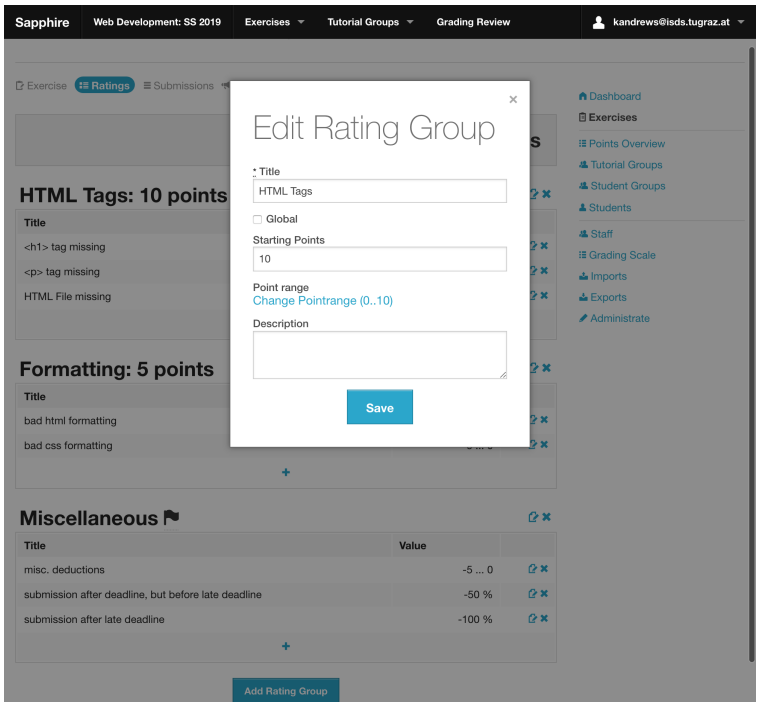

Figure C.30: The Edit Rating Group Modal allows lecturers to modify an existing rating group.

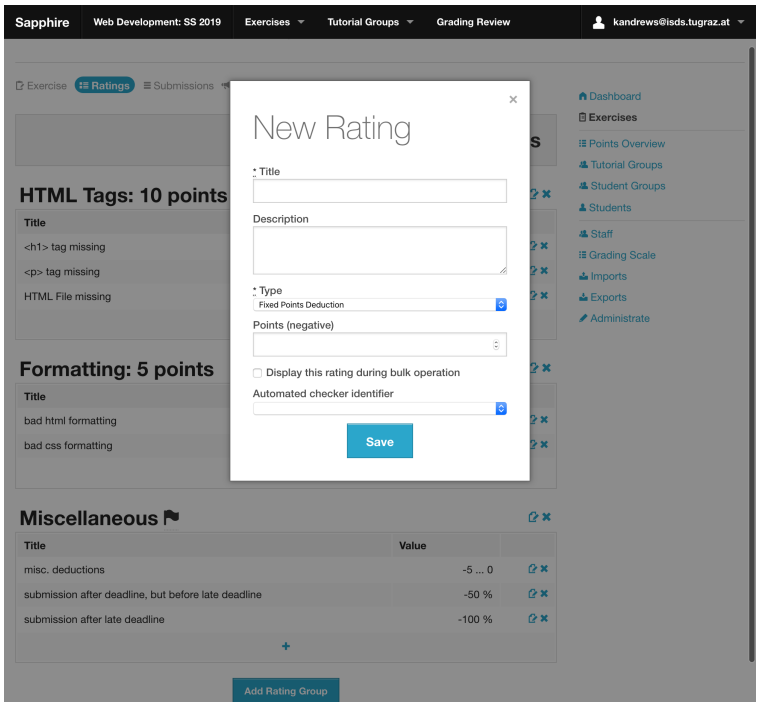

Figure C.31: The New Rating Modal allows lecturers to configure new ratings.

Name	Additional Options	Display	Use Case
Fixed Points Deduction	Points (Negative)	Button	Simple deductions, e.g. missing group name in a report.
Fixed Percentage Deduction	Percentage (0..100)	Button	Deductions on top of other deductions, e.g. -50% for submitting after the deadline.
Variable Points Deduction	Minimum Points, Maximum Points	Field	Deductions based on a tutor's judgement, e.g. quality of a section of a report.
Variable Percentage Deduction	Minimum Percent, Maximum Percent	Field	Deductions based on a tutor's judgement, e.g. overall quality of a report.
Per Item Points Deduction	# items minimum, # items maximum, Multiplication Factor	Field	# missing items in a collection of expected length, e.g. expected 8 screenshots, only 5 submitted.
Per Item Points	# items minimum, # items maximum, Multiplication Factor	Field	# of expected items, e.g. correct answers in an examination.
Fixed Bonus Points	Points (Positive)	Button	Simple fixed bonus points, e.g. student completed an optional task.
Variable Bonus Points	Minimum Points, Maximum Points	Field	Bonus points based on tutor's judgement, e.g. bonus depending on submission quality.
Plagiarism	*none*	Button	Fixed -100% global deduction for plagiarised submissions.

Table C.1: The nine types of rating supported by Sapphire. The Additional Options column shows which additional options are available to lecturers during configuration. The Display column shows which corresponding input element is shown to staff members during evaluation.

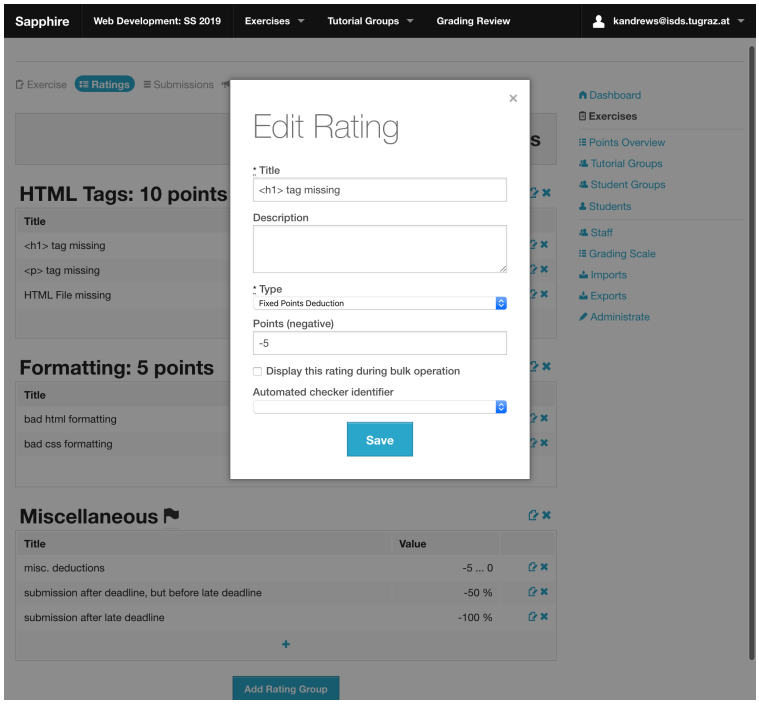

Figure C.32: The Edit Rating Modal allows lecturers to modify existing ratings.

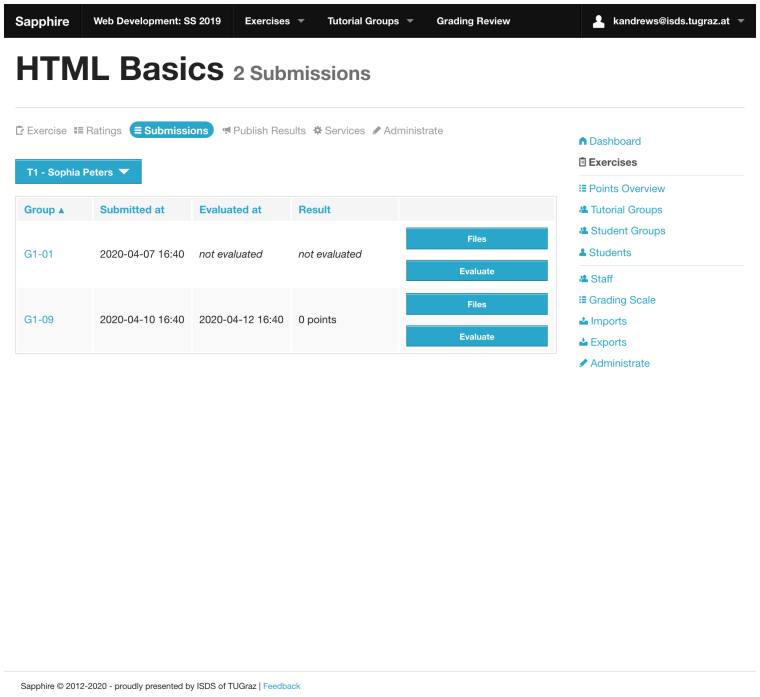

Figure C.33: The Submissions Table displays a list of the current submissions for an exercise.

due to, for example, students being unable to upload to Sapphire directly. Additionally, staff members can create graded submissions using the Bulk Grading interface, described in Section C.12.2.

C.11.1 Submissions Table

The Submissions Table displays a list of current submissions for an exercise, as shown in Figure C.33. For each submission, Sapphire shows the student group, date of submission, date of evaluation, and the resulting points. For individual submissions, the name of the student is included. For exercises allowing multiple attempts, Sapphire also includes the attempt in the Submissions Table. If ratings or rating groups are changed by the lecturer after the submission was last evaluated, Sapphire shows a warning sign next to the points in the points column. The warning sign indicates to staff members that the evaluation may be out of date and that the submission's evaluation should be reviewed.

C.11.2 Submission Tree

The Submission Tree interface provides users with the ability to navigate and manage the files and folders of a submission, as shown in Figure C.34. The Submission Tree interface consists of a table listing the files and subfolders inside the current folder, called the File Table. Sapphire shows the modification date and file size of each entry in the File Table. Users are able to navigate into a folder by simply clicking on its name. To navigate to a parent folder, users can either use the .. link in the first row of the File Table or use the breadcrumb navigation above the File Table. The raw content of a file is accessible by clicking on the corresponding file name. Sapphire allows users to delete files and folders by clicking on the red x button of the corresponding entry in the File Table.

C.11.3 Uploading Files

The Upload New Files Modal is used to upload files to Sapphire, as shown in Figure C.35. The Upload New Files Modal is opened by clicking the Upload button in the toolbar of the Submission Tree interface. Users are able to drag and drop files from the desktop into the Drop Zone area of the Upload New Files Modal, which is

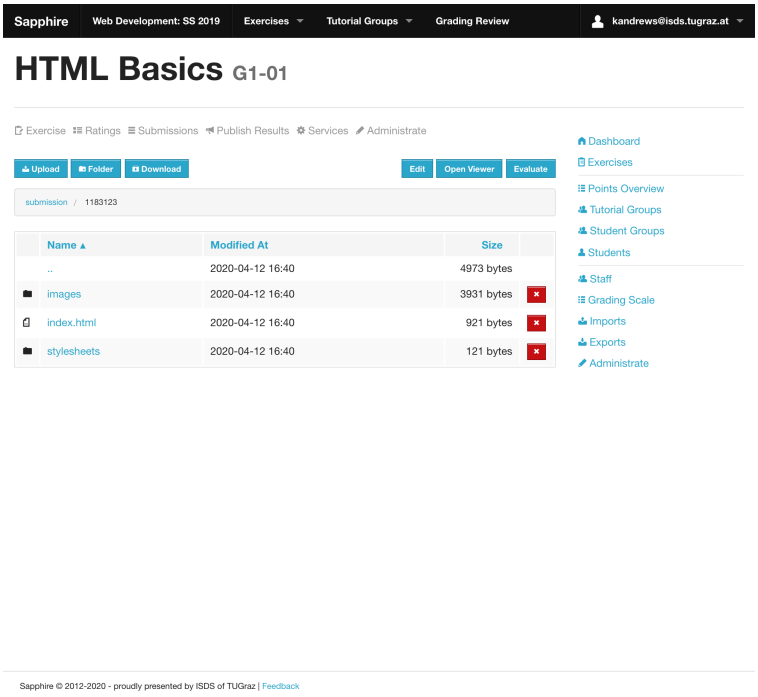

Figure C.34: The Submission Tree interface allows users to navigate and change files, similar to the file browser of an operating system.

indicated by the text Drop files here or click to select. Alternatively, users are able to click on the Drop Zone to select files from a dialogue provided by the browser, as the text suggests. By default, Sapphire assumes that new files should be added to the current folder. Users are able to change the destination folder by clicking on the Edit link.

Sapphire also supports uploading complex folder structures in the form of ZIP archives. Sapphire automatically extracts ZIP archives in a background process, provided the maximum upload size will not be exceeded. Some browsers, for example Google Chrome, support uploading folders directly, eliminating the need to create a ZIP archive.

C.11.4 Downloading Files

Sapphire allows users to download the contents of the current folder in the Submission Tree interface. The download process is started by clicking the Download button in the toolbar of the Submission Tree interface. Sapphire dynamically creates a ZIP archive, which is streamed to the browser as a file download.

C.11.5 Creating Folders

New folders are created via the New Folder Modal by clicking the Folder button in the toolbar of the Submission Tree interface, as shown in Figure C.36. Users are able to both enter a new folder name as well as a subfolder path, using slashes as path dividers. Once the Create button is clicked, the user is navigated to the corresponding folder path. Creating folders in Sapphire differs from traditional file browsers in that folders are not physically created until files are uploaded to the folder.

Sapphire automatically checks that the folder name is available and indicates the status below the input field. The result of the availability check does not prohibit users from submitting the form. Instead, the availability status is displayed to allow users to catch errors early during the submission process.

For security reasons, Sapphire simulates the file system shown to its users. While Sapphire's file system is similar to traditional ones in terms of functionality, it does not implement renaming or moving

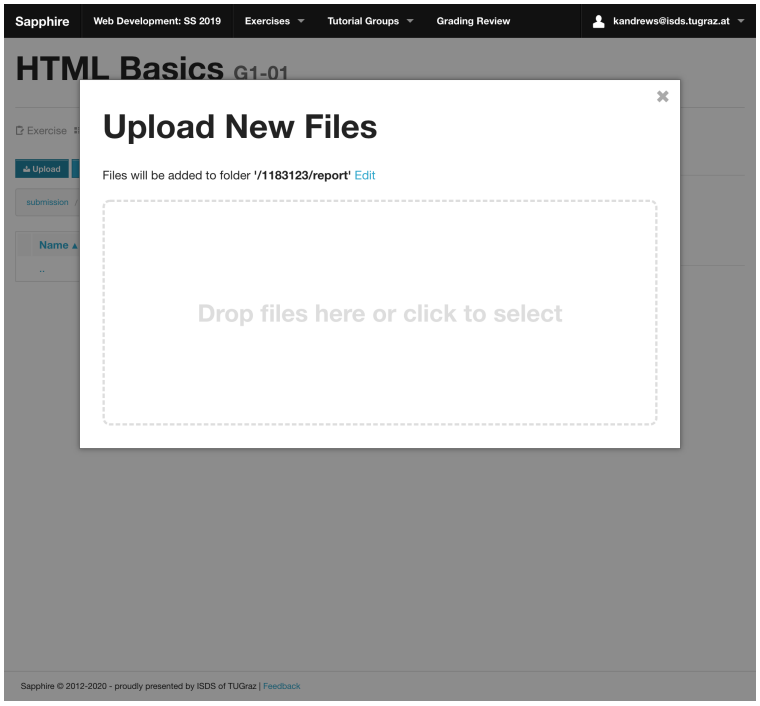

Figure C.35: The Upload New Files Modal is used to upload files to a submission.

files from one folder to another. Instead, users are currently required to remove and reupload files and folders. However, it is likely this feature will be implemented in a future release of Sapphire.

C.11.6 Limitations

Despite its appearance, the file system of Sapphire is simulated during runtime. Therefore, there are several limitations regarding the functionality of the file system. One of the most noticeable is that Sapphire does not support moving or renaming files and folders though the UI. Instead, users are required to download and remove the concerned files from Sapphire, then renaming the files on the local file system, and reuploading them to Sapphire.

C.11.7 Administrating Submissions

Sapphire enables staff members (tutors and lecturers) to modify submissions using the Edit Submission interface, shown in Figure C.37. Staff members are able to configure the students assigned to a submission. The students comprising a student group can only be changed by lecturers during the course of a term. However, if the students assigned to a student group are changed after the submission has been created, the students assigned to the submission need to be altered manually via this interface. In order to simplify this process, both lecturers and tutors are able to perform this action.

By clicking the Add Student button, Sapphire adds an entry to the table of students. Next, the staff member needs to search for the relevant student by entering either the name or matriculation number in the search field and selecting the corresponding entry from the list of search results displayed as a dropdown. Existing student associations can be changed similarly, by clicking the Pencil icon next to the name of the student, and proceeding with searching for another student. A student can be removed from a submission by clicking the x button in the corresponding row.

Staff members are able to set individual point deductions for students associated with a submission, for example if one student has not contributed as much to a submission as other group members. In case multiple attempts are enabled for an exercise, the attempt of the exercise can be changed as well. All

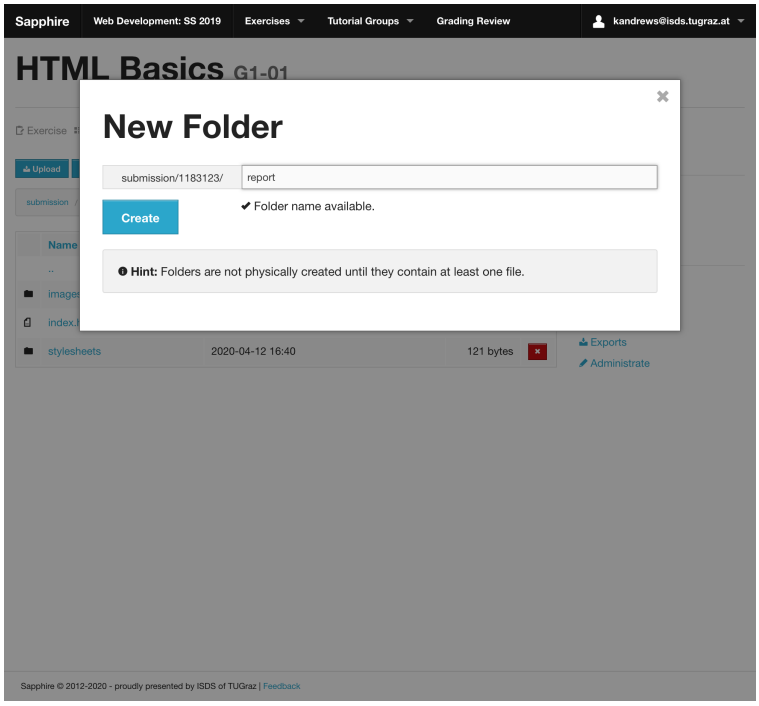

Figure C.36: The New Folder Modal is used to create a new folder.

changes to the submission need to be confirmed by clicking the Submit button. Sapphire performs the changes only after the form has been submitted.

C.12 Grading

The ability to grade and manage hundreds of students is one of the key features of Sapphire. A family of interfaces is responsible for providing this functionality. The Single Evaluation interface, described in Section C.12.1, provides the universal grading interface for staff members (tutors and lecturers). There is, however, a second interface called the Bulk Grading interface, which is described in Section C.12.2. While the Single Evaluation interface is primarily used for report-based submissions, the Bulk Grading interface is used to enter results determined outside the scope of Sapphire, for example examination scores.

C.12.1 Single Evaluation

The Single Evaluation interface provides the primary grading interface, and is usually used for report-based submissions. The interface is based on three components: the Header, the List of Ratings, and the Footer, and is shown in Figure C.38.

The Header prominently displays the submitter of the submission. Sapphire either displays the student group or the student, depending on whether the exercise is a group or individual exercise, respectively. The current points for the submission and a set of three buttons are shown beneath the name of the submitter. The Edit button navigates to the administrative interface of the submission, described in Section C.11.7. The Open Viewer button is shown for exercises with a submission viewer enabled. The Open Viewer button navigates to the Submission Viewer for the submission, as described in Section C.13. The Files button navigates to the Submission Tree interface, described in Section C.11.2.

The List of Ratings constitutes the main section of the single evaluations interface. The ratings of an exercise are collected into rating groups. Each rating group is worth an initial number of points. Ratings are used to deduct from the initial points of a rating group. Sapphire displays either a button or an input

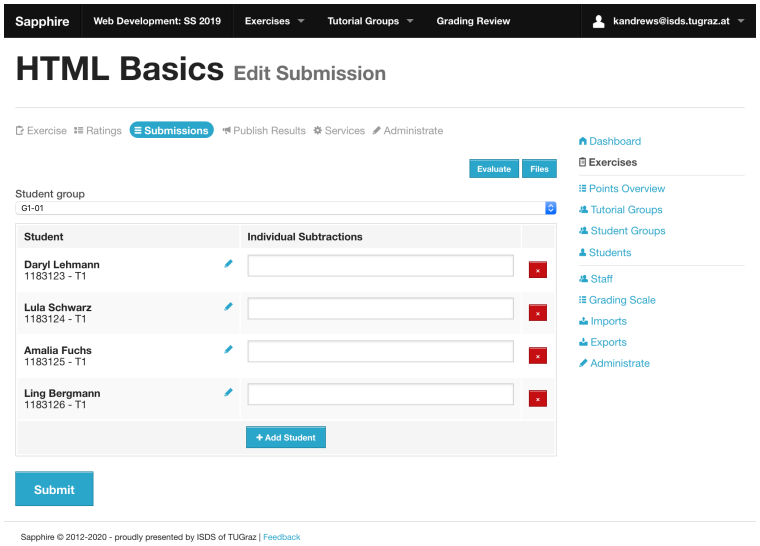

Figure C.37: The Edit Submission interface for submissions lets staff members change the associated student group, exercise attempt, and associated students, along with managing any individual point deductions.

field for each rating, depending on its type. The points are saved and are updated automatically whenever a rating button is clicked or an input field is filled in. Rating buttons are initially grey. Clicking on a rating button turns the button red and the underlying deductions or bonus points are applied.

It is common for the List of Ratings to be several screens long. In order to keep things organised, it is possible to collapse fully graded ratings groups. By clicking on the Done button next to the title of the rating group, Sapphire hides the ratings of the rating group and marks the rating group as completed.

The status of a rating group is indicated by a badge of a specific colour and with a specific icon in front of the rating group title. Blue badges with a clock icon represent either unstarted or unfinished rating groups, which is the default. Green badges with a checkmark icon denote finished rating groups. Red badges with a warning sign icon are used to mark rating groups where one or more ratings have been changed since the submission was last evaluated and need to be reviewed.

The ratings of rating groups needing review are always shown, regardless of whether the rating group is marked as done or not. A review is confirmed by clicking on checkmark button next to the changed rating. Once all ratings under review are confirmed, Sapphire reverts the status of the rating group to its previous state, either unfinished or done.

The Footer of the Single Evaluation interface provides a summary of the status of each rating group for the submission. Each rating group is represented by its corresponding badge. Clicking on a badge scrolls the browser to the corresponding rating group in the List of Ratings.

C.12.2 Bulk Grading

The Bulk Grading interface is the secondary grading interface of Sapphire and allows staff members to enter the results of multiple students at once, as shown in Figure C.39. Staff members are able to access the Bulk Grading interface via the Bulk Grading button in the Submissions Table, described in Section C.11.1. Lecturers

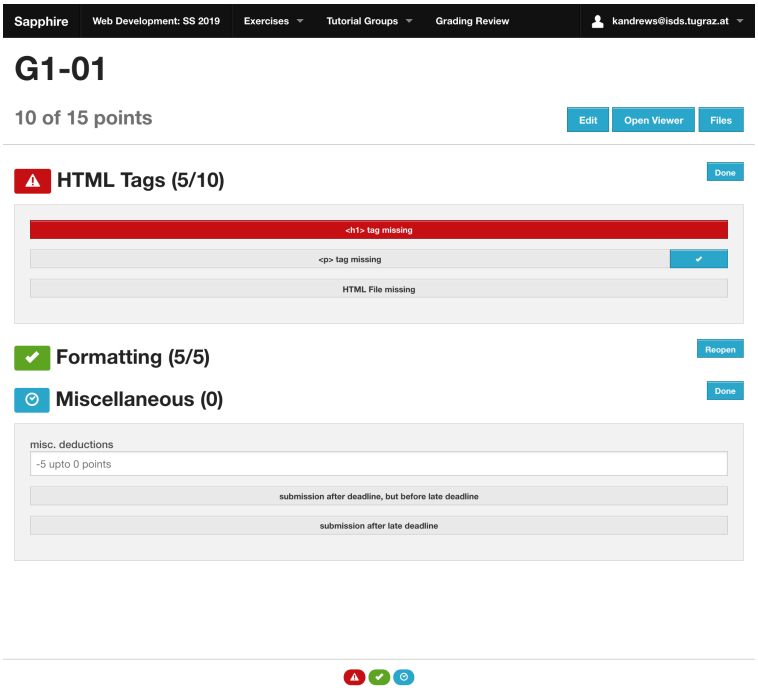

Figure C.38: The Single Evaluation interface allows staff members to grade submissions.

are responsible for enabling the Bulk Grading interface and configuring which ratings are to be shown, as described in Section C.9.4 and Section C.10, respectively.

If the exercise supports multiple attempts, the attempt needs to be selected from the Select Attempt panel first. The Grading Table underneath provides the main functionality of the Bulk Grading interface. The first column of the table is responsible for searching for and selecting students or student groups, depending on the exercise type. A search is initiated by entering either the matriculation number or the name of the student, or the name of the student group. Sapphire starts searching for the student or student group as soon as the first digit or letter is entered. The staff member is able to select the student or student group from the dropdown showing the search results. Sapphire automatically appends another empty row to the Grading Table once a student or student group is selected. In order to change a previously selected student, the Pencil icon needs to be clicked, which reverts the corresponding table row to displaying the search field.

The remaining columns are used to display the ratings of the exercise. The name of each rating is displayed in the header row of the Grading Table. The inputs for each rating are displayed in the table cells of that column. Depending on the rating type, either an input field or checkbox is displayed as rating input, for variable and fixed ratings, respectively. Clicking on the red x button in the last column of the table removes of the corresponding table row.

The evaluations are saved to the database by clicking the Submit button. Sapphire automatically determines whether to update an existing evaluation or create a new submission for each row of the Grading Table.

C.13 Submission Viewers

Submission viewers are a vital part of the grading process. Sapphire provides the means for looking at the content of individual files from the Submission Tree interface. However, some submissions require a special environment to be viewed correctly. In particular, Sapphire provides a HTML Submission Viewer, described in Section C.13.1, and a CSS Submission Viewer, described in Section C.13.2.

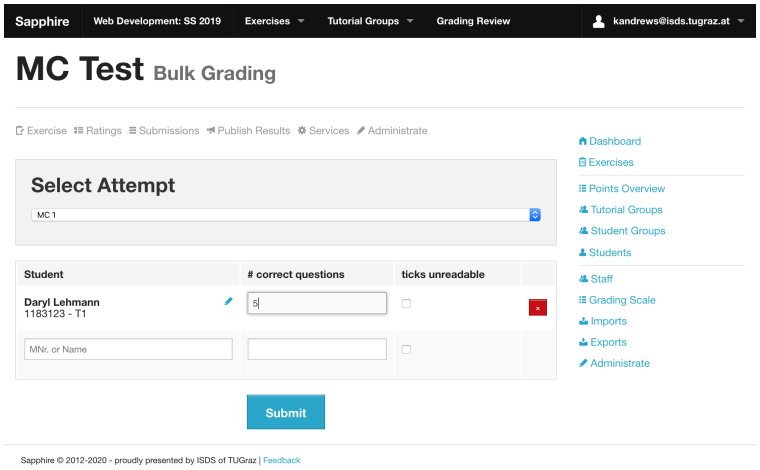

Figure C.39: The Bulk Grading interface is used to enter multiple results of externally graded exercises into Sapphire.

C.13.1 HTML Submission Viewer

The HTML Submission Viewer provides a web-server like environment for viewing HTML files, as shown in Figure C.40. The toolbar in the top right of the interface allows users to quickly switch between the HTML files of a submission. Even though Sapphire is a web application, the HTML submitted by students is not directly displayed to the user. Instead, Sapphire parses the HTML file first and replaces relative URLs with the internal URLs of the Sapphire system. Additionally, the resulting HTML is stripped of JS code to prevent XSS attacks. Thus, the HTML differs from the submitted file. Therefore, it is important not to rely solely on what is displayed in the HTML Submission Viewer for grading. Staff members are advised to download the files of the original submission, especially when the displayed HTML file looks strange.

C.13.2 CSS Submission Viewer

The CSS Submission Viewer provides the means to view alternative submitted CSS files in the context of a given HTML file. A predefined HTML template is used, to which the submitted stylesheets are applied, as shown in Figure C.41 for the Graz Times HTML template. Staff members are able to switch between the alternative CSS files of a submission using the toolbar in the top right corner of the CSS Submission Viewer.

C.14 Publishing Results

Sapphire allows lecturers to publish the results of an exercise as a whole or on a per-tutorial-group basis, using the Publish Results interface shown in Figure C.42. The interface is accessible through the Publish Results link in the submenu of an exercise.

The Publish Results interface consists of a table of tutorial groups. The first column of the table displays the name of the tutorial group along with the associated tutors. The second column shows the current publication status for each tutorial group. In the rightmost column, Sapphire either displays a Publish X or Conceal X button, depending on whether the results are currently concealed or published, respectively.

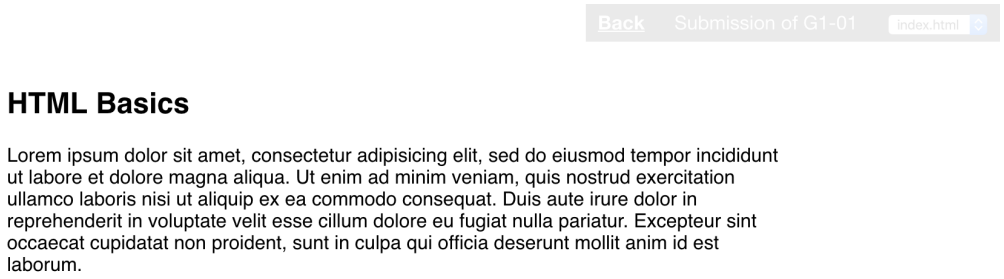

Figure C.40: The HTML Submission Viewer displays HTML files and included assets as if they were served from a web server.

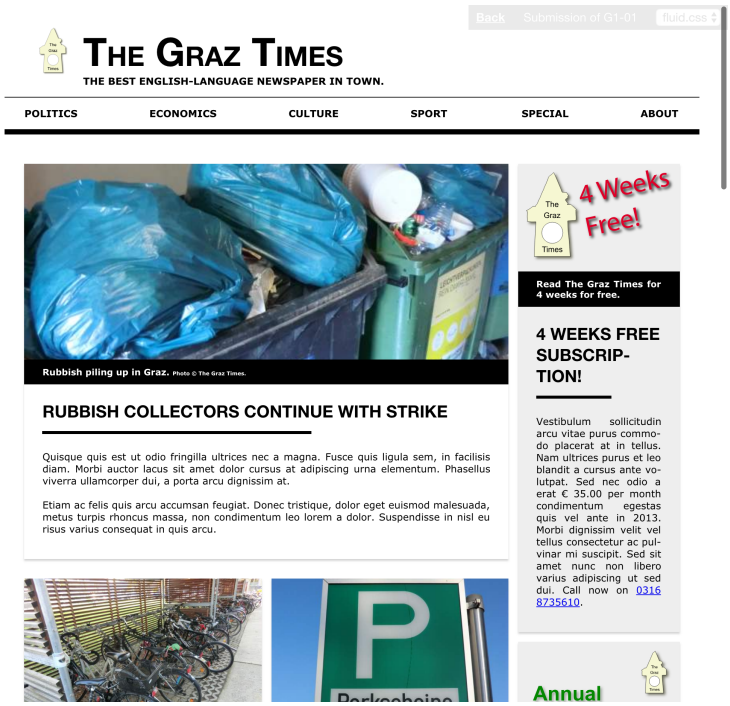

Figure C.41: The CSS Submission Viewer applies one of multiple submitted CSS files to a predefined template, such as the Graz Times template shown here.

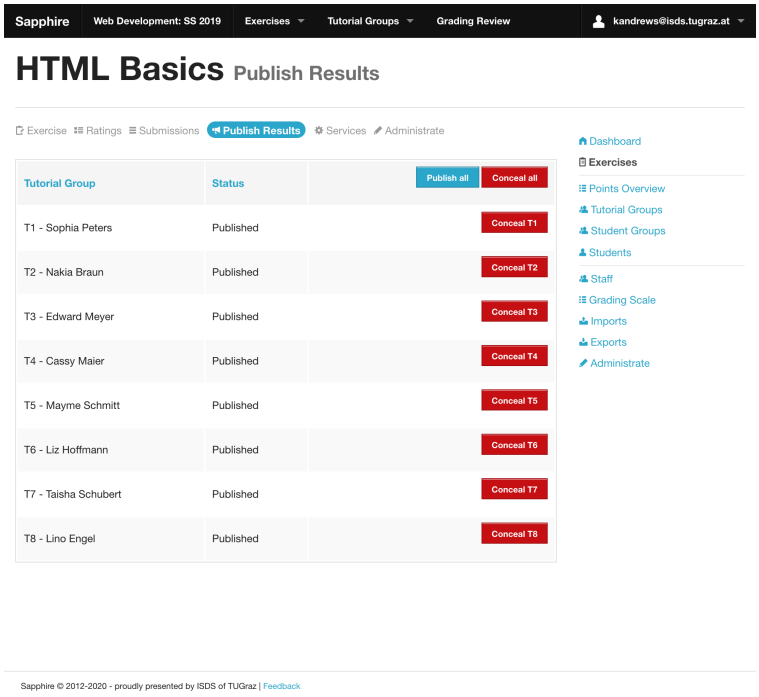

Figure C.42: The Publish Results interface allows lecturers to publish the results of an exercise as a whole or on a per-tutorial-group basis.

By clicking the button, lecturers are able to toggle between publishing and concealing results for each tutorial group individually.

In addition to changing the publication status of exercise results per tutorial group individually, lecturers are able to change the status of all tutorial groups at all once. The Publish All and Conceal All buttons publish and conceal the results of all tutorial groups, respectively, when clicked.

C.15 Grading Scale

The grading scale is the mapping from points to grades. Configuring the grading scale is an important task for the lecturer at the end of a term. The Grading Scale Editor interface is provided by Sapphire to allow lecturers to configure the grading scale of a term, and is shown in Figure C.43. The Grading Scale Editor interface is comprised of two parts. The Grade Distribution Table provides an overview of the grade distribution over the tutorial groups of the term. The Grade Distribution Chart shows the grade distribution over all students and also provides interactive functionality to set the grade boundaries.

The Grade Distribution Table is shown at the top of the page and displays the distribution of grades for each tutorial group. The total number of students receiving a specific grade, and the respective overall percentages are shown in the Total and Percent columns. The total number of students per tutorial group and the total number of students receiving a grade are shown at the bottom of the table. Additionally, Sapphire shows the number of ungraded students, who have not participated in any submission of the term.

The Grade Distribution Chart provides an interactive grading scale editor based on vertical bar charts. The points are plotted on the vertical axis and the number of students on the horizontal axis. The number of students is plotted in two directions. Blue bars on the righthand side represent students who are eligible for a positive grade, having reached a certain number of points. Gray bars on the lefthand side represent students receiving a negative grade, despite reaching a certain number of points, due to missing submissions or not having reached the minimum number of points for an exercise.

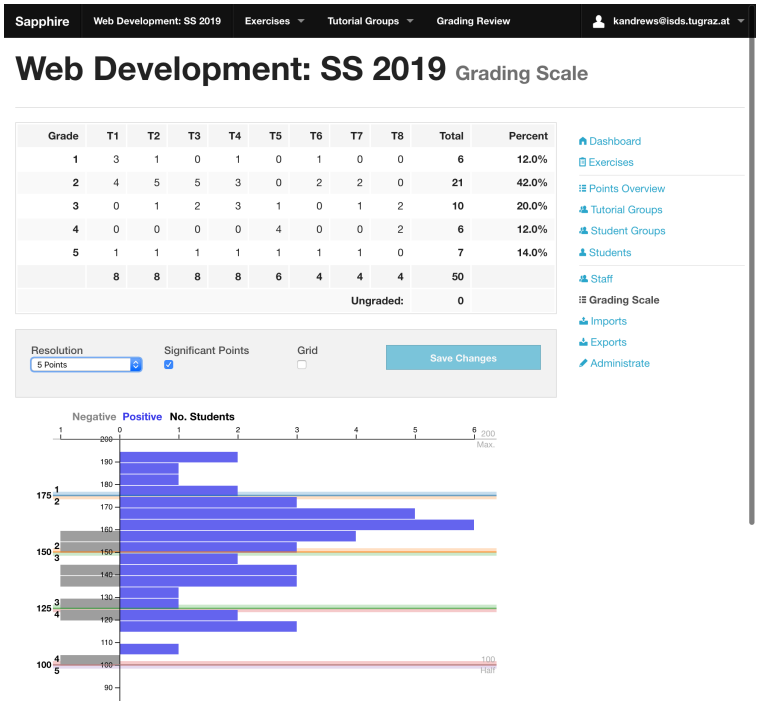

Figure C.43: The Grading Scale Editor displays an overview of current grades for the term and allows lecturers to configure grade boundaries directly in the Grade Distribution Chart at the bottom.

In order to change the grade boundaries, lecturers can either drag and drop the boundary markers between two grades or double-click on a boundary marker to reveal an input field. The number of points specified as a boundary are inclusive in the direction of higher grades. Therefore, students having an equal or higher number of points to the boundary receive the grade specified above the boundary. Students having less points than the boundary receive the lower grade. Lecturer are required to confirm changes to the grading scale in order to persist the changes to the database, by clicking on the Save Changes button.

Lecturers are able to change the appearance of the Grade Distribution Chart using the control panel above the chart. The Resolution option controls the thickness of the bars representing students. By default, bars are plotted in one point increments providing fine-grained insight into the distribution of points. Increasing the resolution results in the bars representing a range of multiple points, which, in turn, results in the bars providing a more general overview of the distribution of grades. Sapphire provides markers for the significant points numbers of half and total points of a term. The Significant Points option allows lecturers to control the display of these markers.

The height of the Grade Distribution Chart is based on the maximum number of points possible for a term. Therefore, it is possible for the Grade Distribution Chart to grow larger than the available screen size, resulting in the x axis not being visible when scrolling down. The Grid option allows lecturers to show thin lines at regular intervals ranging from the top to the bottom of the chart. Additionally, Sapphire provides a special Cursor Line at the position of the mouse cursor (pointer device), when hovering over the Grade Distribution Chart. When hovering over a bar, the Cursor Line displays the number of students corresponding to the bar and the points or point range (for thicker bars) of the bar.

C.16 Grading Review

The *grading review* is a special meeting at the end of term, where students are able to ask questions about the grading process and to query certain deductions. Staff members clarify misunderstandings and correct any errors.

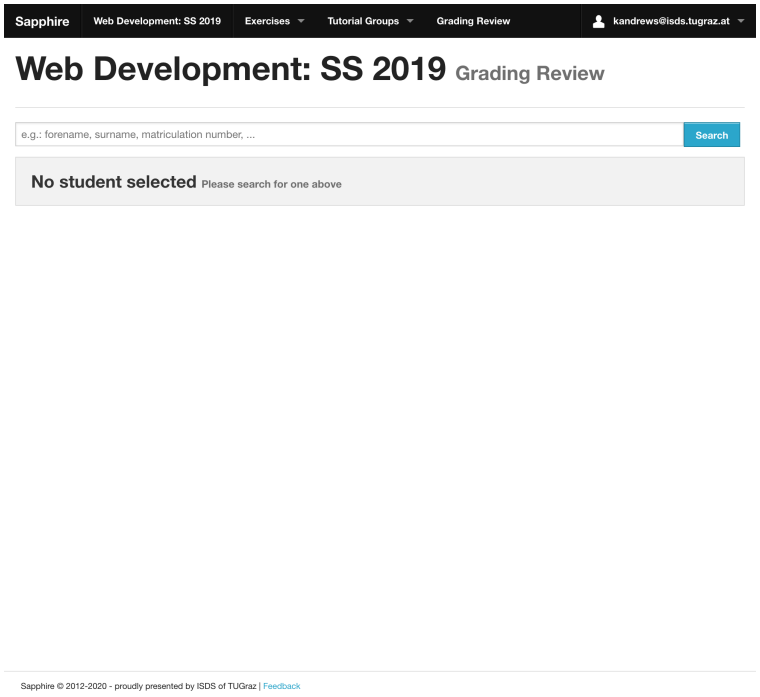

Figure C.44: The Grading Review Dashboard initially only shows a Search Form to avoid unnecessary disclosure of information.

C.16.1 Grading Review Dashboard

The Grading Review Dashboard serves as the starting point for a grading review with a specific student. Grading is a sensitive topic and it is vital not to disclose information about other students. Therefore, the Grading Review Dashboard only shows the Search Form by default, as shown in Figure C.44.

The specific student is selected using the Search Form. A staff member initiates a search by entering a student's name or matriculation number into the search field and clicking the Search button. Sapphire returns a table of matching students consisting of the name, matriculation number, and tutorial group, as shown in Figure C.45. A grading review is started by clicking the Show button in the relevant table row, which leads to the Grading Review Detail interface.

C.16.2 Grading Review Detail

The Grading Review Detail interface provides detailed information about the grading of a specific student. A panel of tabs is at the core of the interface, allowing users to quickly switch between the Overview Tab and an Evaluation Detail tab for each submission. When the Grading Review Detail interface is first accessed, Sapphire displays the Overview Tab. It shows an Overview Table of all of the submissions of a specific student, as shown in Figure C.46. For each submission, the Overview Table shows the associated exercise, submission date, and the points. In the footer of the Overview Table, Sapphire provides the preliminary grade and the total points achieved during the term.

By clicking on an Evaluation Detail tab, staff members are able to access a summary of the evaluation for each submission, as shown in Figure C.47. The list of rating groups along with applied ratings is shown on the left side of the interface. Sapphire provides three buttons beneath the Rating Groups List. The Reopen Evaluation button navigates the user to the Single Evaluation interface, described in Section C.12.1. The administrative interface of the submission is accessible through the Edit button, described in Section C.11.7. If a Submission Viewer is enabled for the exercise, Sapphire provides an Open Viewer button, which navigates the user to the corresponding Submission Viewer, as described in Section C.13. Inline versions of the

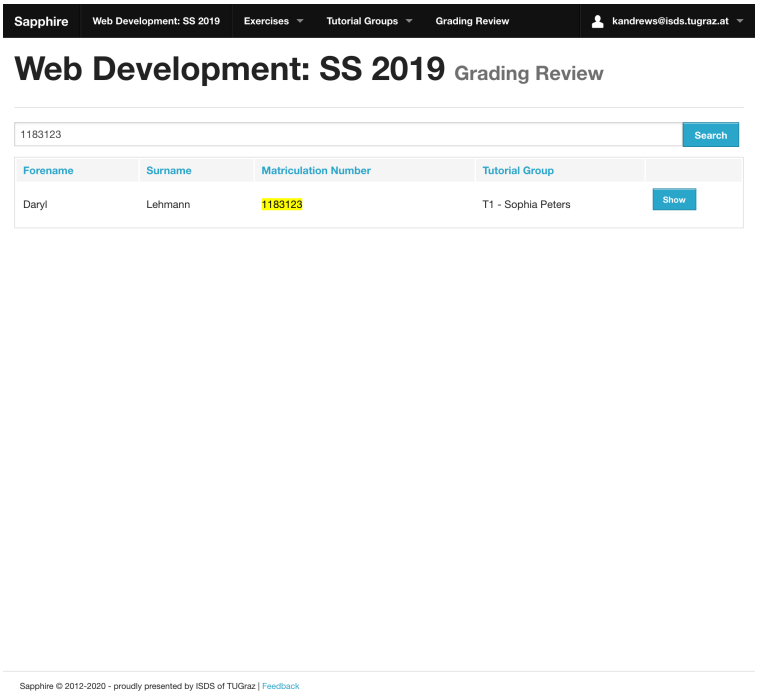

Figure C.45: Searching for a specific student in the Grading Review Dashboard.

submitted files are shown on the right side of the Grading Review Detail interface. Sapphire performs syntax highlighting for human readable files, such as HTML files. Additionally, a View Raw button is provided for each file, which allows users to view the raw content of the file.

C.17 Points Overview

Sapphire provides an overview of the overall results for a term, called the Points Overview, which is shown in Figure C.48. Staff members are able to access the Points Overview interface by clicking the Points Overview link in the sidebar of the term dashboard, described in Section C.3.

The Grade Distribution Table at the top left of the Points Overview interface provides a summary of the grade boundaries and the distribution of grades over all students in the term. For each grade, the points range, number of students, and the respective percentage is displayed. In addition, Sapphire lists a special *ungraded* grade, which represents the number of registered students who are ungraded, typically because they handed in no submission during the course of the term. The Exercises Table, located at the top right of the Points Overview interface, shows the name, minimum points required, and maximum points possible for each exercise. The total points possible (excluding bonus points) is shown in the last row of the Exercises Table.

Beneath the Grade Distribution Table and Exercises Table, Sapphire provides an overview of the results for each student, grouped by tutorial group. For each tutorial group of the term, a Points Table is shown. For each student of the tutorial group, a row is added to the Points Table, showing the matriculation number, points for each exercise, total points, and resulting grade. Some students might not have handed in a submission for every exercise. Missing exercises are denoted with *na*, which stands for *not available*.

In addition to the overall points overview for the entire term, Sapphire provides a dedicated Points Overview for each tutorial group of the term, which is accessible through the Points Overview link in the sidebar of the Tutorial Group Detail interface, described in Section C.5.2. The information provided is similar to the overall Points Overview. However, the grade distribution statistics are calculated based only on the students belonging to that tutorial group and Sapphire only provides a Points Table for that tutorial group.

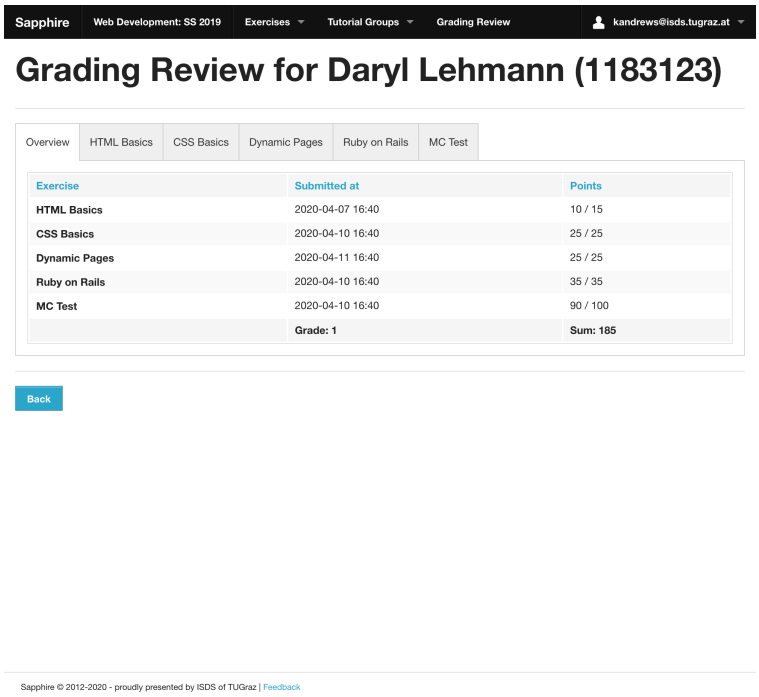

Figure C.46: The Overview Tab of the Grading Review Detail interface provides a summary of all submissions of a specific student. Each of the remaining tabs gives access to the evaluation of a particular submission.

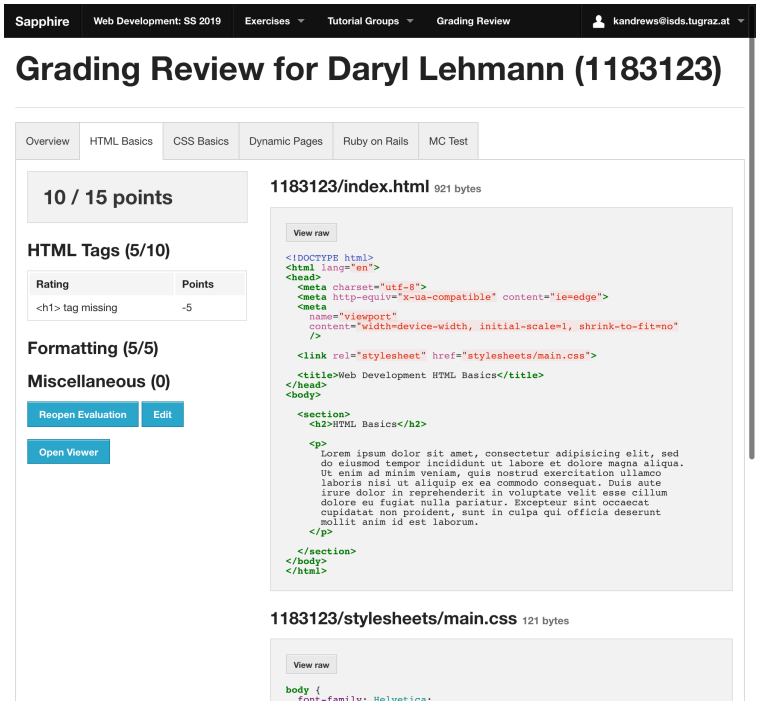

Figure C.47: The Evaluation Detail tab of the Grading Review Detail interface provides detailed information about the evaluation of a specific submission, here for the HTML Basics exercise.

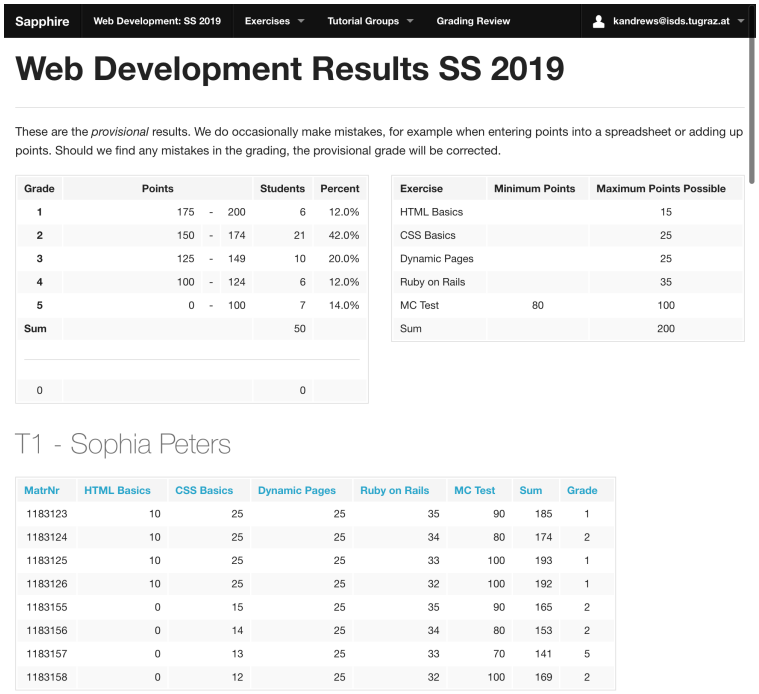

Figure C.48: The Points Overview interface provides an overview of the overall results for a term, including the points per exercise, total points, and grade for each student.

C.18 Exports

Exports play a vital role in archiving the results and submissions of a term. Lecturers are able to initiate new exports as well as view and download previous exports. Exports are generated using a background job. The lecturer is notified by email once an export has finished.

C.18.1 Exports Table

The Exports Table displays a list of previously generated exports, as shown in Figure C.49. The table indicates the type of export, export status, creation date, and last (status) change date. Additionally, Sapphire provides a Download and a Delete button, allowing lecturers to download or delete the corresponding export. The New Export button takes the lecturer to the New Export interface.

C.18.2 Creating Exports

The New Export interface is used for preparing new exports. The first step in creating a new export is to choose an export type, as shown in Figure C.50. Sapphire provides a Grading Export and a Submission Export, which are described in Section C.18.3 and Section C.18.4, respectively. Since creating an export can take many minutes or sometimes hours, exports are generated in a background job, allowing users to continue using Sapphire in the meantime. Once an export is finished, Sapphire notifies the lecturers of the term by email that the export has finished.

C.18.3 Grading Export

A Grading Export creates an archive of the grading and results of a particular term in the form of one Excel spreadsheet per tutorial group. Figure C.51 shows the New Grading Export panel. The lecturer starts the Grading Export by clicking on the Start button. The Cancel button takes the lecturer back to the Exports Table, described in Section C.18.1.

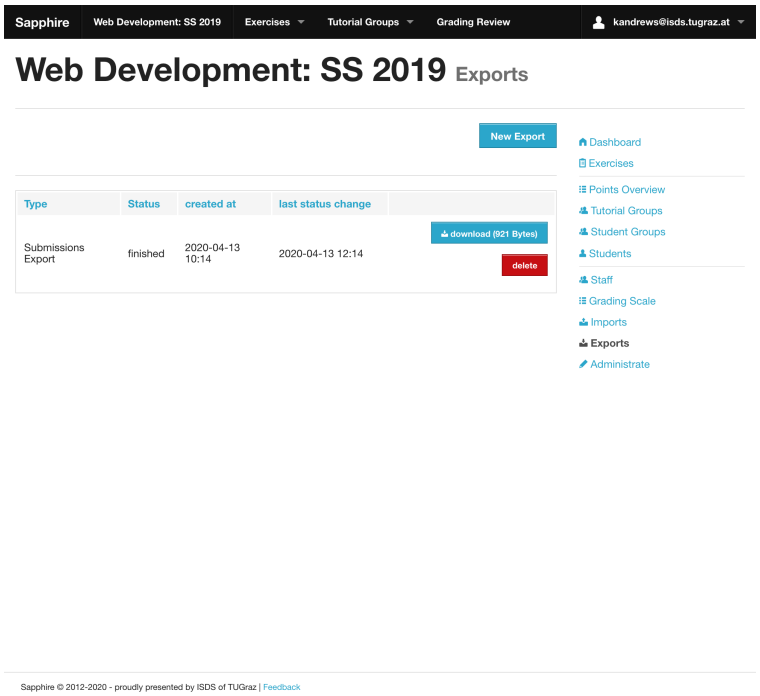

Figure C.49: The Exports Table provides a list of previously generated exports.

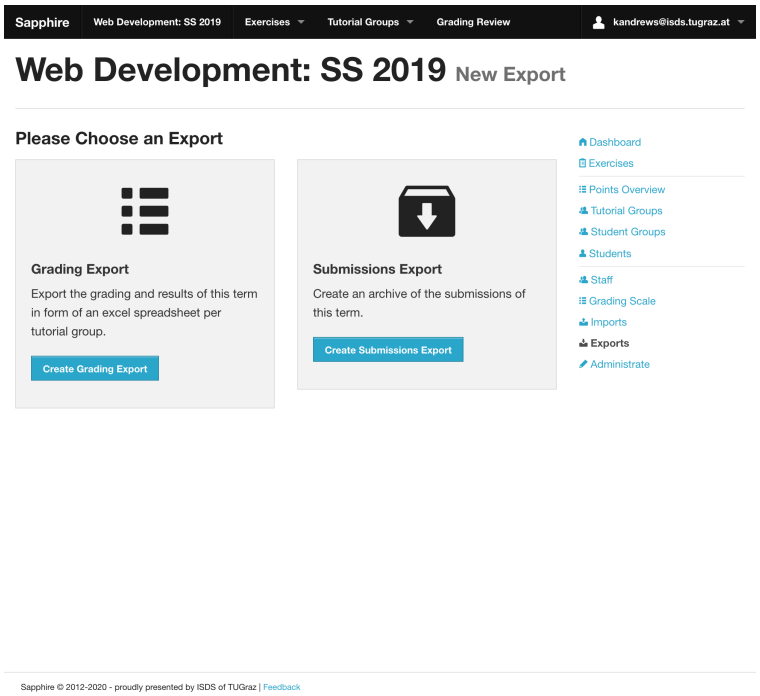

Figure C.50: Choosing the type of export in the New Export interface.

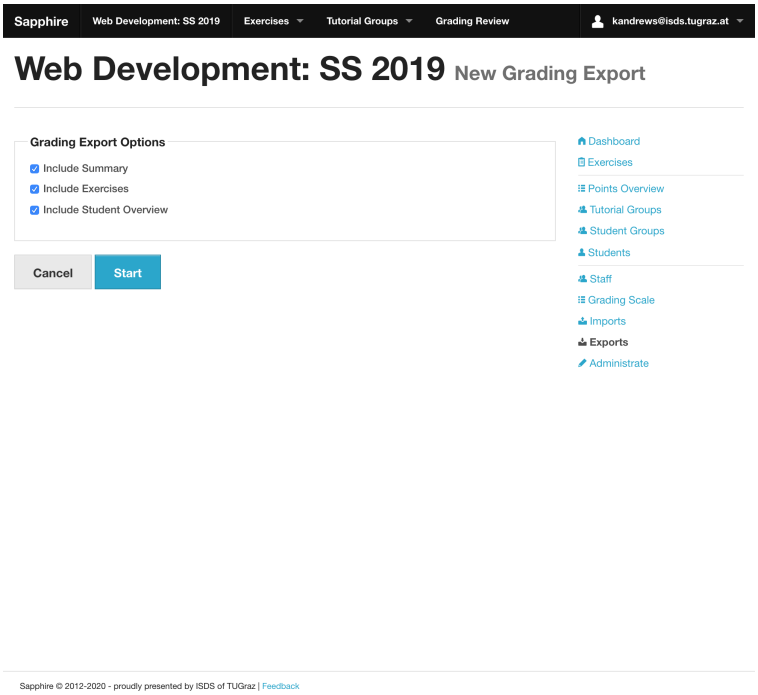

Figure C.51: The options for creating a new grading export in the New Grading Export panel.

Three configuration options are available for a Grading Export:

- *Include Summary*: The Include Summary option adds a Summary sheet to each Excel spreadsheet, which provides an overview of the students of the tutorial group, along with the student group, points per exercise, total points, and grade. The distribution of grades is included as well. If there are student groups, an additional Group Summary sheet is included in each exported spreadsheet, displaying the points each group achieved for each group exercise, the total points for the group exercises, and the average grade of students in the student group.
- *Include Exercises*: The Include Exercises option adds a sheet for each exercise to the spreadsheet. An exercise sheet consists of a list of rating groups and ratings and the corresponding evaluation results for each student or student group (depending on whether the exercise is an individual or group exercise).
- *Include Student Overview*: The Include Student Overview option includes a Students sheet, which provides details about each student in the tutorial group. The Students sheet displays the name, matriculation number, student group, email address, sbox username, and the URL of the webspace provided to the student by the university.

Spreadsheets exported by Sapphire include basic formatting, as shown in Figure C.52. It is important to note that the spreadsheets generated by the Grading Export contain only *static* values. No formulae (to calculate, say, aggregate values) are included. Instead, Sapphire inserts the (static) results of such calculations into the spreadsheets. Hence, it is not possible to change values inside an exported spreadsheet and have such changes propagate to the summary pages, for example to automatically update the grade of a student.

C.18.4 Submissions Export

A Submissions Export creates a ZIP archive of all the submissions of a term. Sapphire allows lecturers to customise the folder structure of the ZIP archive in the New Submissions Export panel, shown in Figure C.53.

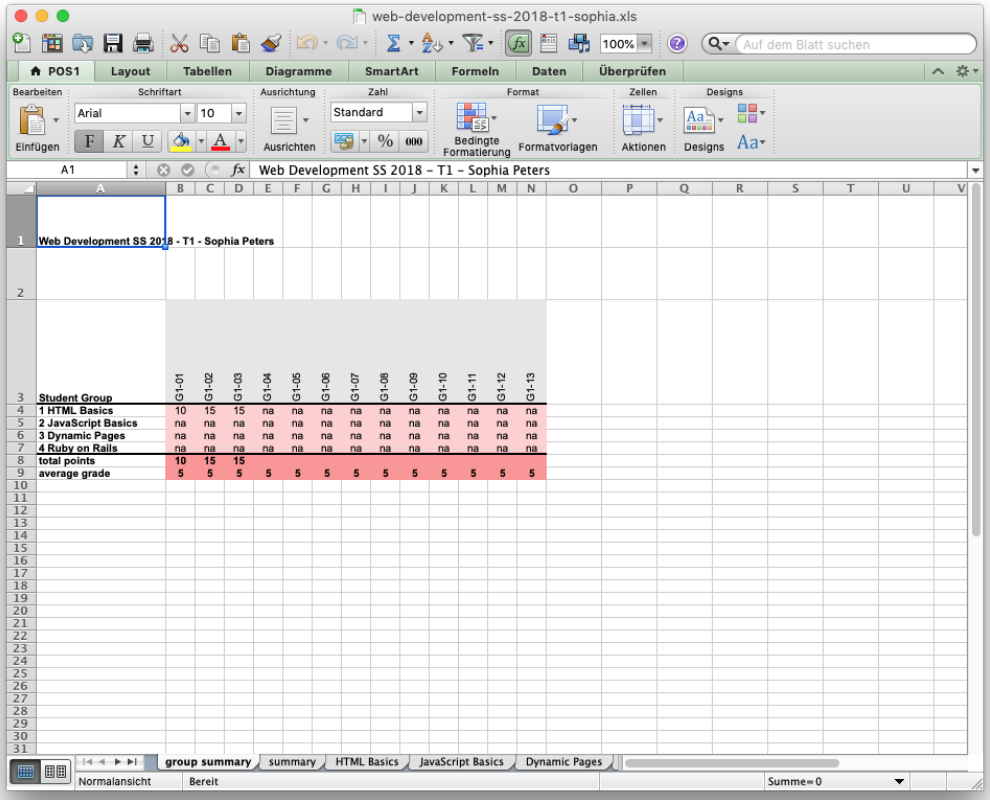

Figure C.52: The Group Summary sheet of an Excel spreadsheet created by the Grading Export.

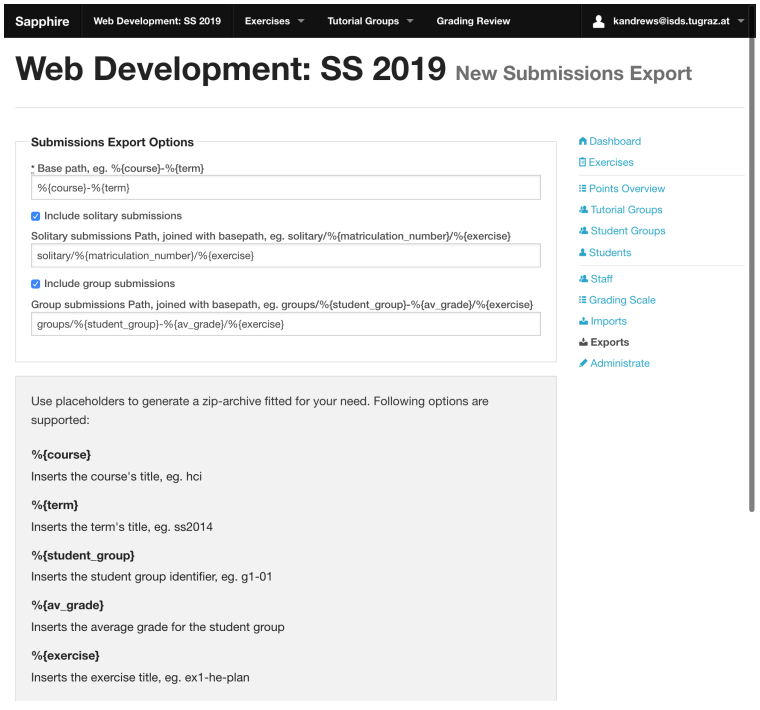

Figure C.53: The options for creating a new submissions export in the New Submissions Export panel.

Placeholder	Example	Description
%{av_grade}	`1`	Average grade of the student group, rounded to the nearest integer.
%{course}	`hci`	Name of the course.
%{exercise}	`html-basics`	Name of the exercise.
%{matriculation_number}	`1234123`	Matriculation number of the student.
%{student_group}	`g1-01`	Name of the student group.
%{term}	`ss2019`	Name of the term.
%{tutorial_group}	`t1-sophia-matthias`	Name of the tutorial group and its tutors.

Table C.2: The placeholders Sapphire uses for organising submissions in the exported ZIP archive.

The Base Path option allows lecturers to specify a folder path inside which the submissions are placed. The Solitary Submissions Path and Group Submissions Path options customise the paths of solitary and group submissions, respectively. Both the Solitary Submissions Path and Group Submissions Path are appended to the Base Path for the archiving process, along with paths of files and folders of the submission. Lecturers are able to specify whether solitary or group submissions should be added to the archive through the respective options Include solitary submissions and Include group submissions.

In order to avoid file name collisions, lecturers are required to configure the Base Path, Solitary Submissions Path, and Group Submissions Path to deliver unique paths for each submission. Sapphire provides several placeholders for this purpose. The available placeholders for path generation are shown in Table C.2. Placeholders are denoted with a percentage sign, followed by an opening brace, the name of the placeholder, and a closing brace. During the substitution phase, whitespace is replaced with a dash and the resulting string is transformed into lowercase letters.

Sapphire fills in common defaults for the Base Path, Solitary Submissions Path, and Group Submissions Path fields. The default value for Base Path is `%{course}-%{term}` and is used for the name of the ZIP file as well as the top-most directory inside the ZIP archive, for example `web-development-2019`. The value `solitary/%{matriculation_number}/%{exercise}` is the default value of the Solitary Submissions Path field. During the process of preparing the ZIP archive, the value of the Solitary Submissions Path field is appended to the value of the Base Path field and is used to provide the directory structure for solitary submissions. For example, files uploaded by a student with matriculation number 1234567 for the Dynamic Pages exercise will be placed into `web-development-2019/solitary/1234567/dynamic-pages`.

The Group Submissions Path field has a default value of `groups/%{student_group}-%{av_grade}/%{exercise}`. Similar to the Solitary Submissions Path value, the Group Submissions Path value is appended to the Base Path value. For example, files submitted for the HTML Basics exercise by the student group G1-01, having an average grade of 3, will be placed into `web-development-2019/groups/g1-01-3/html-basics`.

A text file named `export.txt` is inserted into every ZIP archive created by the Submissions Export to indicate the timestamp of export generation. An example of the structure of a Submissions Export ZIP archive using the default values is shown in Figure C.54.

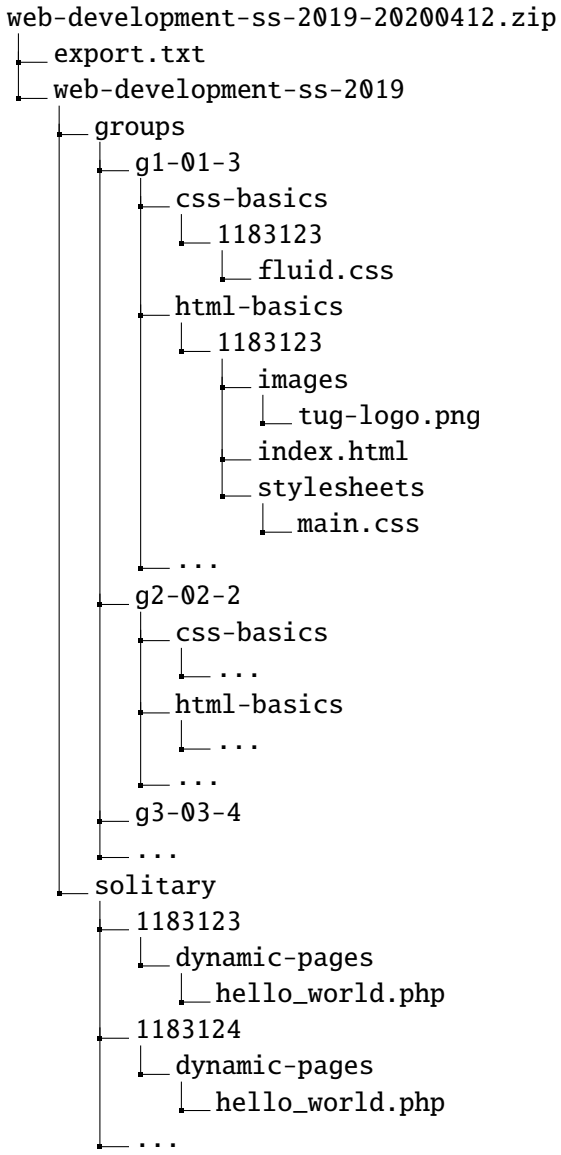

Figure C.54: An example of a folder structure created for a Submissions Export using the default options.

Bibliography

Ahuvia, Yogev [2013]. *Infinite Scrolling: Let's Get to the Bottom of This*. 03 May 2013. https://smashingmagazine.com/2013/05/infinite-scrolling-lets-get-to-the-bottom-of-this/ (cited on page 64).

An, Jong-hoon, Avik Chaudhuri, and Jeffrey S. Foster [2009]. *Static Typing for Ruby on Rails*. 24th IEEE/ACM International Conference on Automated Software Engineering, 2009 (ASE '09). 2009, pages 590–594. doi:10.1109/ASE.2009.80. https://cs.umd.edu/projects/PL/druby/papers/drails-ase09.pdf (cited on page 3).

Andrews, Keith [2014a]. *Exercise 4: Build a Web Site (Polyglot XHTML5)*. 2014. https://courses.isds.tugraz.at/inm2014/exercises/exer4.html (cited on pages 25, 51).

Andrews, Keith [2014b]. *Exercise 5: Three Style Sheets (CSS3)*. 2014. https://courses.isds.tugraz.at/inm2014/exercises/exer5.html (cited on page 25).

Andrews, Keith [2014c]. *Internet and New Media*. 2014. https://courses.isds.tugraz.at/inm2014/ (cited on page 19).

Andrews, Keith [2019a]. *HCI Exercise 1*. 2019. https://courses.isds.tugraz.at/hci/practicals/exer1.html (cited on pages 51, 74).

Andrews, Keith [2019b]. *HCI Multiple Choice Test*. 2019. https://courses.isds.tugraz.at/hci/practicals/mctest.html (cited on pages 22, 31, 54, 56).

Andrews, Keith [2019c]. *HCI Practical Exercises*. 2019. https://courses.isds.tugraz.at/hci/practicals/ (cited on page 25).

Andrews, Keith [2019d]. *Human-Computer Interaction*. 19 Apr 2019. https://courses.isds.tugraz.at/hci/ (cited on page 19).

Andrews, Keith [2019e]. *Writing a Thesis: Guidelines for Writing a Master's Thesis in Computer Science*. Graz University of Technology, Austria. 27 Mar 2019. https://ftp.isds.tugraz.at/pub/keith/thesis/ (cited on page xxiii).

ASF [2017]. *The Apache HTTP Server Project*. Apache Software Foundation. 20 Feb 2017. https://httpd.apache.org/ (cited on page 24).

Ashkenas, Jeremy [2015]. *CoffeeScript*. 28 Oct 2015. https://coffeescript.org/ (cited on page 14).

Bootstrap [2019a]. *About Bootstrap*. 07 Feb 2019. https://getbootstrap.com/docs/4.2/about/overview/ (cited on page 18).

Bootstrap [2019b]. *Bootstrap - The Most Popular HTML, CSS, and JS Library in the World*. 07 Feb 2019. https://getbootstrap.com/ (cited on page 18).

Bootstrap [2019c]. *Bootstrap Introduction*. 07 Feb 2019. https://getbootstrap.com/docs/4.2/getting-started/introduction/ (cited on page 18).

Catlin, Hampton Lintorn, Natalie Weizenbaum, and Chris Eppstein [2015]. *CSS with Superpowers*. 23 Sep 2015. https://sass-lang.com/ (cited on page xix).

Collective Idea [2015]. *Delayed::Job - Database Based Asynchronous Priority Queue System – Extracted from Shopify*. 16 Jan 2015. https://github.com/collectiveidea/delayed_job (cited on page 60).

Dev, Vijay [2015]. *Ruby on Rails 3.1 Release Notes*. 28 Oct 2015. https://guides.rubyonrails.org/3_1_release_notes.html (cited on pages 14–15, 17).

ECMA [2015]. *ECMAScript 2015 Language Specification*. 18 Jun 2015. https://ecma-international.org/ecma-262/6.0/ECMA-262.pdf (cited on page 16).

Fielding, Roy Thomas and Julian F. Reschke [2014]. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. RFC 7230*. Jun 2014. https://tools.ietf.org/html/rfc7230 (cited on page 60).

Fowler, Martin [2002]. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 02 Nov 2002. ISBN 0321127420 (cited on pages 7–8).

GitHub [2015]. *GitHub Features*. 06 Nov 2015. https://github.com/features (cited on page 73).

Google [2019]. *Google Chrome*. 25 Feb 2019. https://google.com/chrome (cited on page 46).

Günther, Sebastian and Marco Fischer [2010]. *Metaprogramming in Ruby: A Pattern Catalog*. Proc. 17th Conference on Pattern Languages of Programs (Reno, Nevada). PLOP '10. ACM, 16 Oct 2010, pages 1–35. ISBN 145030107X. doi:10.1145/2493288.2493289. https://hillside.net/plop/2010/papers/gunther.pdf (cited on page 4).

ISI [1981a]. *Internet Protocol. RFC 791*. Information Sciences Institute, University of Southern California. Sep 1981. https://tools.ietf.org/html/rfc791.html (cited on pages xix, 11).

ISI [1981b]. *Transmission Control Protocol. RFC 793*. Information Sciences Institute, University of Southern California. Sep 1981. https://tools.ietf.org/html/rfc793.html (cited on pages xix, 11).

Jehl, Scott [2014]. *Picturefill, A Responsive Image Polyfill*. 06 Dec 2014. https://scottjehl.github.io/picturefill/ (cited on page 12).

Jones, Geraint [2014]. *Ruby: A Notation and Design Discipline Intended for the Development of Regular Integrated Circuits and Similar Hardware and Software Architectures*. 28 Feb 2014. http://cs.ox.ac.uk/geraint.jones/ruby/ (cited on page 3).

Kriechbaumer, Thomas [2014]. *Sapphire Back-End*. 28 Feb 2014. https://ftp.isds.tugraz.at/pub/theses/kriechbaumer-2014-bsc.pdf (cited on page 1).

Lie, Håkon Wium and Bert Bos [1996]. *Cascading Style Sheets, level 1*. 17 Dec 1996. https://w3.org/TR/REC-CSS1-961217 (cited on page 12).

Lindner, Paul [1993]. *Registration of a New MIME Content-Type/Subtype*. 20 Jul 1993. https://iana.org/assignments/media-types/application/zip (cited on pages 25, 43).

McDonald, Thomas, Tristan Harward, Peter Gumeson, and Gleb Mazovetskiy [2015]. *Bootstrap for Sass*. 27 Oct 2015. https://github.com/twbs/bootstrap-sass (cited on page 14).

Microsoft [2019]. *TypeScript - JavaScript that Scales*. 07 Feb 2019. https://typescriptlang.org/ (cited on page 16).

Mockapetris, Paul V. [1987]. *Domain Names - Concepts and Facilities. RFC 1034*. Nov 1987. https://tools.ietf.org/html/rfc1034.html (cited on page 11).

Nielsen, Jakob [1993]. *Response Times: The 3 Important Limits*. 01 Jan 1993. https://nngroup.com/articles/response-times-3-important-limits/ (cited on page 59).

Oskoboiny, Gerald et al. [2019]. *Markup Validation Service*. 25 May 2019. https://validator.w3.org/ (cited on page 75).

Perham, Mike [2012]. *Asynchronous Processing for Fun and Profit*. Talk at Ruby Conference 2012. 02 Nov 2012. https://confreaks.tv/videos/rubyconf2012-asynchronous-processing-for-fun-and-profit (cited on page 60).

Phusion [2019]. *Passenger - Enterprise Grade Web App Server for Ruby, Node.js, Python*. 19 Apr 2019. https://phusionpassenger.com/ (cited on page 24).

PostgreSQL [2019]. *PostgreSQL: The World's Most Advanced Open Source Relational Database*. 19 Apr 2019. https://postgresql.org/ (cited on page 24).

RICG [2014]. *Responsive Images Community Group*. 06 Dec 2014. https://responsiveimages.org/ (cited on pages 12–13).

Ruby, Sam, David Thomas, and David Heinemeier Hansson [2013]. *Agile Web Development with Rails 4*. 4th Edition. Pragmatic Bookshelf, Oct 2013. ISBN 1937785564 (cited on page 7).

Ruby Community [2019]. *Ruby in Twenty Minutes - Page 4*. 06 Feb 2019. https://ruby-lang.org/en/documentation/quickstart/4 (cited on page 3).

Sánchez Cuadrado, Jesús and Jesús García Molina [2007]. *Building Domain-Specific Languages for Model-Driven Development*. IEEE Software 24.5 (2007), pages 48–55. ISSN 0740-7459. doi:10.1109/MS.2007.135. http://dslab.konkuk.ac.kr/Class/2010/10RE/Building%20Domain-Specific%20Languages%20for%20Model-Driven%20Development%20.pdf (cited on page 3).

Sanfilippo, Salvatore [2019]. *Redis*. 19 Apr 2019. https://redis.io/ (cited on page 24).

Sidekiq [2019]. *Sidekiq: Simple, efficient background processing for Ruby*. 2019. https://sidekiq.org/ (cited on pages 24, 59).

Tanaka, Kazuaki, Yukihiro Matsumoto, and Hiroshi Arimori [2011]. *Embedded System Development by Lightweight Ruby*. 2011 International Conference on Computational Science and Its Applications (ICCSA). 2011, pages 282–285. doi:10.1109/ICCSA.2011.62 (cited on page 3).

Thomas, David, Andy Hunt, and Chad Fowler [2013]. *Programming Ruby 1.9 & 2.0*. Pragmatic Bookshelf, 04 Jul 2013. ISBN 1937785491 (cited on page 3).

TUG [2019]. *TUGRAZonline*. 25 May 2019. https://online.tugraz.at/ (cited on pages 12, 19–20, 34).

Turbolinks [2019]. *turbolinks/turbolinks: Turbolinks Makes Navigating Your Web Application Faster*. 19 Apr 2019. https://github.com/turbolinks/turbolinks (cited on page 24).

W3C [2015]. *CSS Current Status*. 18 Oct 2015. https://w3.org/standards/techs/css (cited on page 11).

W3C [2017]. *HTML 5.2 W3C Recommendation*. 14 Dec 2017. https://w3.org/TR/html52/ (cited on page 11).

W3Schools [2019]. *HTML <source> Tag*. 25 Feb 2019. https://w3schools.com/TAGs/tag_source.asp (cited on page 12).

Wikipedia [2019]. *TypeScript*. 07 Feb 2019. https://wikipedia.org/wiki/TypeScript (cited on page 16).

ZURB [2013a]. *Foundation 4 Documentation*. 22 Nov 2013. https://web.archive.org/web/20151008142713/http://foundation.zurb.com/docs/v/4.3.2/ (cited on page 18).

ZURB [2013b]. *Foundation 4 Documentation - Support*. 22 Nov 2013. https://web.archive.org/web/20151109122957/http://foundation.zurb.com/docs/v/4.3.2/support.html (cited on page 18).

ZURB [2015]. *Foundation::Rails*. 27 Oct 2015. https://github.com/zurb/foundation-rails (cited on page 14).

ZURB [2019]. *Foundation - The Most Advanced Responsive Front-End Framework in the World*. 07 Feb 2019. https://foundation.zurb.com (cited on pages 18, 24).