Development of a Gamified Application for Programming Education

Semester Thesis

(Studienarbeit)

Semester:	Fall 2023
Project Team:	Lukas Messmer
	Mathias Fischler
Advisor:	Frieder Loch

OST

Eastern Switzerland
University of Applied Sciences

Abstract

Programming exercises are a crucial component in teaching aspiring computer scientists practical knowledge about software development. At OST, many courses offer a wide range of different programming exercises, ranging from algorithm development in Java to artificial intelligence training in Python.

In this semester thesis, we were tasked with bringing gamification into the realm of university exercises. The main goal was to develop a software system that incentivizes students to engage repeatedly and consistently with programming assignments.

Approach

For this thesis, we employed a user-centered design process to evaluate and conceptualize a gamified application for programming exercises. We thereby proceeded as follows:

- We defined a user base consisting of 2 user groups
- We conducted 11 user interviews with different people from this user base
- We derived 101 tangible user stories from these interviews
- We conceptualized 5 aspects of an application that addresses these user stories, using existing applications and scientific papers as a foundation for our ideas
- We prototyped different features to ensure their technical feasibility
- We defined 40 functional requirements for our minimal viable product, which will be developed as part of our Bachelor's thesis

Conclusion

Through this thesis, we have concluded that an application designed to motivate students at OST to engage more regularly with their exercises cannot be developed on the basis of gamification alone. As such, our thesis focuses on the conceptualization of a software system that aims to unify and simplify the process of creating, managing, conducting and evaluating exercises. Such an application, although broader than initially intended, could improve the value of exercises at OST.

Contents

1. Introduction

1.1. Context

Programming exercises are a crucial component in teaching aspiring computer scientists practical knowledge about software development. At OST - Eastern Switzerland University of Applied Sciences, many courses offer a wide range of different programming exercises, ranging from algorithm development in Java to artificial intelligence training in Python.

In this semester thesis, we were tasked with bringing gamification into the realm of university exercises. Gamification describes the process of applying game-like elements (e.g. scoring, competition, etc.) to other areas of activity in order to increase motivation and encourage engagement (Oxford University Press, 2023). The main goal is to develop a software system that incentivizes students to engage repeatedly and consistently with programming assignments.

1.2. Assignment

In order to create the aforementioned system, we were given these assignments for our semester thesis. The original version (written in German) can be found in the appendix under Section I.

1. Analyze existing applications used at the OST to formulate requirements and needs of future users of such a gamified system.
2. Research similar products and scientific papers to define objectives and boundaries of the application to develop.
3. Formulate requirements on the basis of which a comprehensive technology selection can be made, with a focus on free and open source solutions.
4. Use a user-centered approach to create a human-focused software product.
5. Create a functional prototype based on the findings of this thesis, including a detailed documentation and at least one user-centered test run.
6. Reflect on the developed product and formulate concrete improvement suggestions for the chosen approach and the created system.

Furthermore, we are required to select a suitable project management method, describe work packages and milestones, and track the time invested at a work package level. All of this must be documented in the final paper.

1.3. Formalities

This semester thesis will be rewarded with 8 ETCS upon successful completion. The expected workload is 240 hours distributed over 14 weeks (~17h/week). The semester thesis starts on **Monday, September 18, 2023** and ends on **Friday, December 22, 2023 at 17:00**. Meetings with the supervisor are held weekly unless otherwise agreed.

All results of this semester thesis (e.g. source code, documentation, etc.) must be made available to the project participants for further use. The documentation is expected to meet common quality standards for scientific papers. All references to external sources must be cited in the bibliography using the APA 7 style. All written texts must follow the Guideline for Gender-Sensitive Wording of the Swiss Confederation (Bundeskanzlei BK, 2023) where applicable.

2. Approach

2.1. Context

As described in Section 1.1, the goal of this semester thesis is to "develop a software system that incentivizes students to engage repeatedly and consistently with programming assignments". While the assignment imposes some requirements on the evaluation of this software system, it does not provide any tangible features that the resulting application should contain.

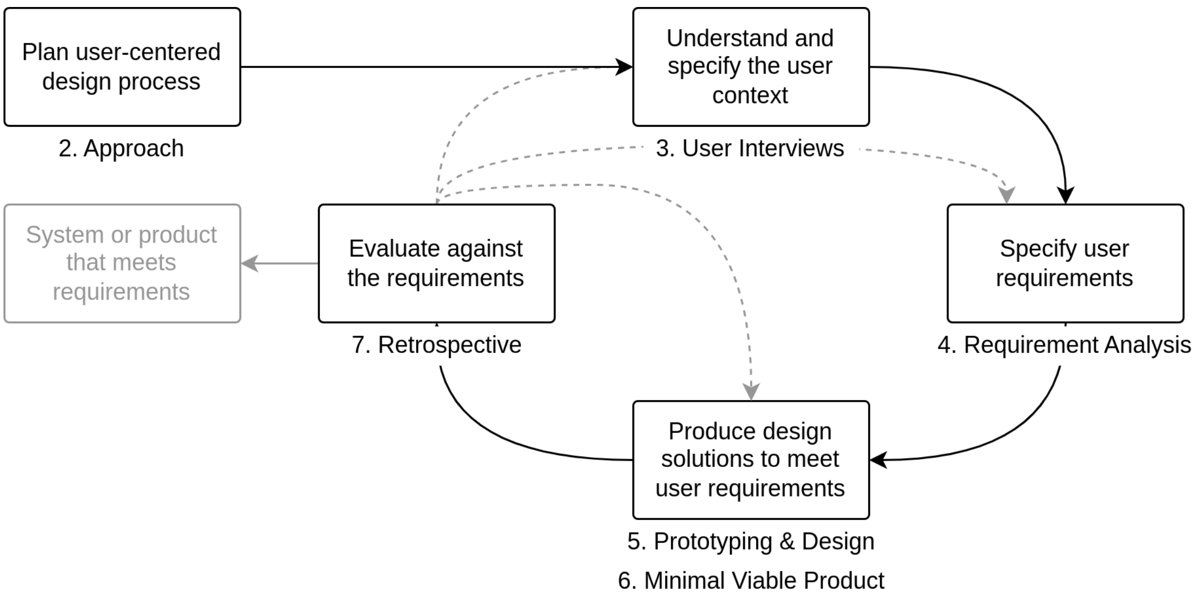
For this reason, we decided to broaden the scope of the semester thesis to include not only the question of how we can use gamification in programming exercises. Rather, we want to find out what a software system for solving exercises should look like in the first place. This means, that we want to find answers to these questions:

- How are the exercises currently being solved?
- Which features must an application for solving exercises include?
- How can we use gamification to improve engagement with exercises?

In doing so, we strive to develop a software system that does not only meet the requirements of this semester thesis, but will be actively used at the university over an extended period of time.

2.2. Process

In this semester thesis, we have decided to place emphasis on close collaboration with the intended user base of the application to be developed. This is in line both with the requirements of this semester thesis as well as the user-centered design process we employ (International Organization for Standardization, 2019). Our approach is illustrated in Figure 1, which shows how the chapters of this documentation correspond to the user-centered design process.



Plan user-centered design process
2. Approach
Understand and specify the user context
3. User Interviews
System or product that meets requirements
Evaluate against the requirements
7. Retrospective
Specify user requirements
4. Requirement Analysis
Produce design solutions to meet user requirements
5. Prototyping & Design
6. Minimal Viable Product

Figure 1: Each chapter corresponds to a step in the user-centered design process

2.2.1. Proceedings

More precisely, we have decided to structure our work process into 7 steps. These steps are worked on using an agile project management approach, which is described in the appendix under Section II. We will therefore proceed as follows:

1. Determine the user base for the application to be developed
2. Interview this user base to find out how exercises are currently being conducted and what problems are being encountered
3. Derive tangible requirements from the interviews conducted
4. Conceptualize an application that addresses these requirements using existing applications and scientific papers as a basis for ideas
5. Evaluate and prototype solutions for the most important requirements
6. Define a Minimal Viable Product (MVP[Glossary]) with a concrete technology selection
7. Look back at our approach and formulate specific improvements and suggestions for further work

2.3. Thesis Scope

In order to keep this semester thesis manageable within the given time frame, we have decided to focus exclusively on conceptualizing and prototyping the most complex parts of the planned application. This means that no functioning software system will be developed at the end of this thesis. Instead, we have agreed with our supervisor that the actual implementation of the prototyped application will be done as part of our Bachelor's thesis.

Furthermore, we will only perform one iteration of the user-centered design process we employ. While we would have liked to present our developed ideas to the user base in the form of workshops, we found that organizing these would have taken away too much time from the conceptualization process. For this reason, the planned software system will not be presented to the user base as part of this semester thesis, but rather as an extracurricular activity.

3. User Interviews

3.1. Context

As discussed in Section 2, our first step in developing a software system for programming exercises is to identify who the user base of such an application is and what requirements they impose. To do this, we will first define our target user base and then conduct a series of user interviews in order to understand their problems and needs. Our goal with these interviews is to investigate the following questions:

- How are exercises conducted at the moment?
- What tools are used to conduct exercises?
- Why are exercises conducted with these tools?
- What problems are encountered when conducting exercises?

By answering these questions, we can determine what kind of features an application for programming exercises must include and how it can be integrated into the existing methods and tools currently being used. This process of deriving tangible requirements from the conducted user interviews is discussed in Section 4.

3.2. Approach

3.2.1. Defining the User Base

Considering the assignment and scope of this thesis (defined in Section 1), we have concluded the following definition for our target user base:

> "Our user base is everyone at OST who is involved in either solving or making exercises for the Bachelor in Computer Science."

Based on this definition we were further able to separate the user base into two groups, namely those who "make" exercises (i.e. lecturers, assistants) and those who "solve" exercises (i.e. students). To keep these groups consistent and to remove any ambiguities that might arise if the context of the software system changes in the future[1], we have decided to use the following terms for the two groups:

- **Exercise Makers**: Users who create, manage and conduct exercises
- **Exercise Solvers**: Users who consume and solve exercises

3.2.2. Planning the User Interviews

After we identified the user base of our application, we were able to start planning the user interviews. Our goal was to find representatives of the two specified groups and ask them how exercises are conducted and what problems are being encountered. We mainly followed the guidelines for user interviews as taught in the lecture User Experience (Brügger et al., 2023) in combination with the methodologies learned in Rhetorical Communication (Verhein-Jarren & Murbach, 2021).

[1]For example, choosing "Lecturer" instead of "Exercise Maker" is inappropriate if the exercises are created by the lecturer's assistant. Likewise, choosing "Student" instead of "Exercise Solver" is inappropriate if the application is ever used outside of a scholastic context.

3.3. Results

3.3.1. General

In total, 7 interviews with exercise makers and 4 interviews with exercise solvers were conducted. The user interviews were primarily held in (Swiss) German and can be found in paraphrased format in the appendix under Section IV. In addition, some members of our user groups were asked informally about the topics identified in our interviews in order to recognise outliers or confirm certain statements. However, these informal exchanges are not documented in this paper.

3.3.2. Identified Tools & Methods

Through our user interviews, we discovered that a wide range of different tools and methods are used to conduct exercises at OST. We also learned that these tools are usually not standardized and that even the ones that are (namely Moodle [Glossary] and GitLab [Glossary]) are not consistently used by all exercise makers.

To get a better sense of how diverse these tools and methods are, we have compiled a non-exhaustive list showing what was mentioned in our user interviews:

Platforms:
- GitLab (M1-Q1)
- Microsoft Teams (M2-Q2)
- E-Mail (M2-Q7)
- Moodle (M3-Q2)

Applications:
- Learningapps.org (M1-Q6)
- CodeWars (M3-Q5)
- Coder (M4-Q2)
- Gitpod (M4-Q2)
- Custom (M6-Q5)

Exercise Methods:
- Programming (M1-Q2)
- Theory (M1-Q2)
- Pen & Paper (M4-Q2)
- Hardware (M5-Q4)

Markup Languages:
- Jekyll (M1-Q3)
- HTML (M1-Q3)
- Markdown (M2-Q2)
- AsciiDoc (M3-Q2)
- LaTeX (M4-Q2)

Tools:
- GitLab-Pages (M1-Q1)
- GitLab-Wikis (M1-Q5)
- CodeTour (M2-Q5)
- Docker (M4-Q2)
- Cisco Packet Tracer (M5-Q4)
- Lab Topology Builder (M5-Q6)
- Jupyter-Notebooks (M7-Q2)
- Nvidia-Labs (M7-Q3)
- HPC-Cluster (M7-Q3)

This variety in tools and methods poses two major problems for our application to be developed. Firstly, it increases complexity if we want to build our application in such a way that it can be easily integrated into the existing ways exercises are conducted. Secondly, if we were to avoid this complexity by developing a standalone application, we would worsen the problems exercise solvers face with the many tools already in use by exercise makers.[2] This is outlined in more detail in Section 3.3.3.

3.3.3. Identified Problems

In addition to determining which tools and methods are currently being used at OST, we also wanted to understand what problems are being encountered by both exercise makers and exercise solvers in regard to exercises. Our results are illustrated in Figure 2, which shows the problems we discovered in our user interviews in the form of qualitative speech bubbles.

[2] This possibility reminded us of an infamous xkcd: https://xkcd.com/927/

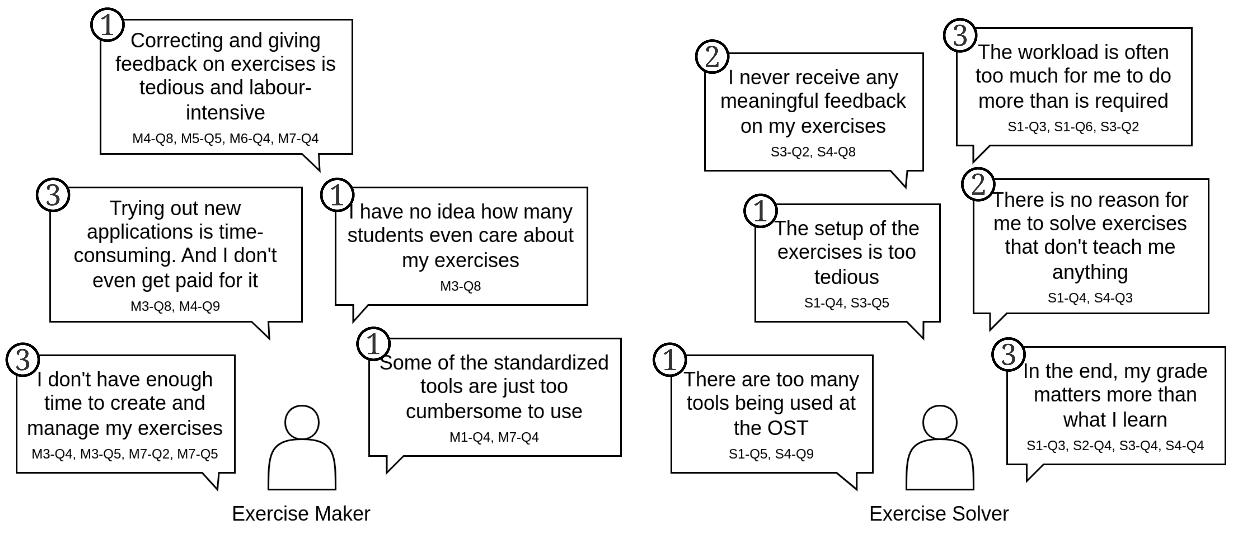

① Correcting and giving feedback on exercises is tedious and labour-intensive M4-Q8, M5-Q5, M6-Q4, M7-Q4
③ Trying out new applications is time-consuming. And I don't even get paid for it M3-Q8, M4-Q9
① I have no idea how many students even care about my exercises M3-Q8
③ I don't have enough time to create and manage my exercises M3-Q4, M3-Q5, M7-Q2, M7-Q5
① Some of the standardized tools are just too cumbersome to use M1-Q4, M7-Q4
Exercise Maker
② I never receive any meaningful feedback on my exercises S3-Q2, S4-Q8
③ The workload is often too much for me to do more than is required S1-Q3, S1-Q6, S3-Q2
① The setup of the exercises is too tedious S1-Q4, S3-Q5
② There is no reason for me to solve exercises that don't teach me anything S1-Q4, S4-Q3
① There are too many tools being used at the OST S1-Q5, S4-Q9
③ In the end, my grade matters more than what I learn S1-Q3, S2-Q4, S3-Q4, S4-Q4
Exercise Solver

Figure 2: Our user interviews revealed various problems in regard to exercises

We further recognized that the problems illustrated in Figure 2 can be categorized into three groups: Organizational problems ①, Quality problems ② and Systematic problems ③.

Organizational problems primarily stem from the various tools and methods being used at OST and how these tools are often too tedious or insufficient to use. As such, we believe that these problems may be solved by using standardized or specialized software systems. Quality problems, on the other hand, are mainly concerned with the perceived inadequate quality of exercises provided by exercise makers. While we think that these problems can only be solved by the exercise makers themselves, they may be aided by tools that help them gather information about the quality of their exercises. Finally, systematic problems arise from the way universities operate (e.g. how exams are administered, how lecturers are paid, etc.). Consequently, we believe that these problems cannot directly be solved by exercise makers, exercise solvers, or any software system, as they would require systematic changes.

3.3.4. Conclusion

Based on our user interviews, we have realized that "a software system that incentivizes students to engage repeatedly and consistently with programming assignments" cannot be created through gamification mechanisms alone. This is because it would not only ignore the underlying problems that hinder consistent engagement with exercises (i.e. tedious setup, excessive workload, etc.), but would actively worsen them by creating another tool that would require additional effort for both exercise makers and exercise solvers to use.

For this reason, we have decided to shift our attention from creating an application that uses gamification in programming assignments, to creating an application that aims to unify, simplify and improve both the organizational and quality aspects of exercises. As such a software system exhibits a much larger scope than originally intended, we have agreed with the supervisor of this semester thesis that the project will be continued as part of our Bachelor's thesis in the spring semester of 2024.

Consequently, the following chapters will focus on conceptualizing an application that addresses the problems we identified in our user interviews, rather than focusing solely on what we were tasked with in our assignment (see Section 1).

3.4. Statistical Significance

While the findings of our user interviews mirror the general topics of discussion we often perceive from being part of these user groups ourselves, we recognize that 11 interviews may not be sufficient to significantly cover all problems encountered with exercises. As a complete and formal evaluation of the problems faced by our defined user base is not feasible in the given time frame, and because we believe that our qualitative interviews have provided sufficient information on the questions defined in Section 3.1, we nevertheless allow ourselves to proceed with the conceptualization of the application to be developed. If time allows, further user interviews may be conducted to deepen our understanding of the problems encountered.

4. Requirement Analysis

4.1. Context

After conducting the user interviews in Section 3, we continued with formulating tangible requirements from our results. These requirements are written in the form of user stories, which can be found in the appendix under Section VI.

4.2. Approach

4.2.1. Defining the Requirements

In order to define concrete requirements from our user interviews, we went through all of our paraphrased documents (see appendix Section IV) and tried to formulate meaningful user stories from them.

Afterwards, we triaged and categorized these user stories for organizational purposes using multiple approaches. Firstly, we decided which user stories are out of scope (i.e. do not fit the application to be developed) or have a low priority (i.e. are too specific or unclear to be considered at the moment). Then, we ranked the user stories based on importance (i.e. how often and with which urgency the user story was mentioned in our user interviews) and by complexity (i.e. how much time and resources we predict the user story will require for implementation). Finally, we separated the user stories based on our two user groups (i.e. exercise makers and exercise solvers).

4.3. Results

4.3.1. General

In total, 101 user stories were formulated based on our conducted user interviews. We furthermore recognized that some of these user stories can be grouped into one of 5 categories, with each category representing what we consider an “epic” in our application. Figure 3 shows a broad overview of these epics, which are elaborated in more detail in Section 4.4 and Section 4.5.

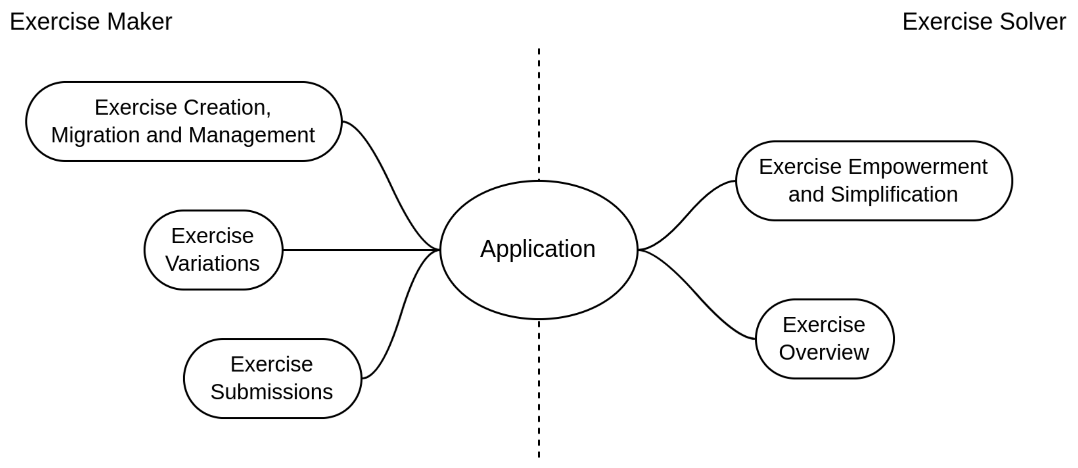


Figure 3: The user stories of our application can be roughly divided into 5 epics

In Section 5 we will discuss how we intend to address these epics. For this purpose, each epic will be given its own chapter, detailing the problems they entail and what solutions we propose in response. These proposed solutions will thereby be heavily influenced by the user stories outlined in Section 4.4 and Section 4.5.

4.4. Requirements of Exercise Makers

4.4.1. Exercise Creation, Migration and Management

The first epic we identified is concerned with the creation, migration and management of exercises by exercise makers. It contains multiple user stories expressing that exercise makers would like to continue using the environment and technologies they are comfortable with and that any migration to a new software system should be as simple as possible. For this reason, we have defined the core requirement of this epic (#5) as follows:

> "As an exercise maker, I want to reduce the overhead for creating and managing exercises."

The following table shows all user stories that extend this core requirement[3].

IID	Requirement	I.	C.
16	I want to easily migrate to a new solution.	5	5
3	I want to use GitLab (VCS) to manage my exercises.	5	4
14	I want to automatically synchronize access rights to my exercises (e.g. Moodle, Adunis)	5	4
7	I want to write exercises in markup languages that I am comfortable with.	4	3
↳ **8**	I want to write exercises in Markdown.	4	2
↳ **10**	I want to write exercises in HTML.	1	3
72	I want to manage my exercises without using GitLab.	2	4
↳ **73**	I want to manage my exercises directly inside the application.	2	2
6	I want to publish my exercises when I change them in GitLab.	2	3
15	I want to reuse exercises that have been created before (e.g. for a different course).	2	2
17	I want to automatically provision the components needed for my exercises (e.g. VMs, Hardware, etc.)	1	4
13	I want to link my exercises directly in the corresponding Moodle course.	2	1

I.: Importance, C.: Complexity, IID: Issue ID

[3]Please note that this table and all subsequent tables do not include any user stories that we consider to be "out of scope" or "low priority". These user stories can instead be found in the appendix under Section VI.

4.4.2. Exercise Variations

The next epic we identified is concerned with the wide range of exercise variations currently being used by exercise makers. It contains multiple user stories expressing that exercise makers would like to provide exercises in a variety of different formats. For this reason, we have defined the core requirement for this epic (#27) as follows:

> "As an exercise maker, I want to provide a wide variety of different exercises."

The following table shows all user stories that extend this core requirement.

IID	Requirement	I.	C.
29	I want to provide practical exercises (e.g. programming tasks)	5	5
28	I want to provide theory exercises (e.g. text, images)	5	3
33	I want to provide exercises that integrate with other tools (e.g. virtual labs).	2	5
34	I want to provide exercises that span over a longer period of time (e.g. mini-projects)	1	3
76	I want to combine theoretical exercises with practical exercises.	1	1

I.: Importance, C.: Complexity, IID: Issue ID

4.4.3. Exercise Submissions

The last epic we identified for exercise makers is concerned with exercise submissions. It contains multiple user stories expressing that exercise makers would like exercise solvers to submit their solutions for exercises and that these solutions should then be evaluated in various different ways. For this reason, we have defined the core requirement for this epic (#42) as follows:

> "As an exercise maker, I want exercise solvers to submit their solutions for an exercise."

The following table shows all user stories that extend this core requirement.

IID	Requirement	I.	C.
70	I want to evaluate the submissions of exercise solvers using custom scripts (e.g. SQL Parser).	4	5
↳ **71**	I want to allow exercise solvers to run the evaluation scripts before submission.	3	1
65	I want submissions to be done in different formats (e.g. file upload, plain text, etc.)	4	4
46	I want to provide feedback directly on the submitted solution.	4	3
↳ **48**	I want to provide feedback in form of text.	3	1

↳ **49**	I want to provide feedback in form of an in-person discussion.	2	1
44	I want to allow exercise solvers to submit solutions as a team.	3	3
↳ **45**	I want to allow exercise solvers to form teams by themselves.	3	2
43	I want to make submissions mandatory.	3	1
51	I want to view all submitted solutions of an exercise in one central location.	3	1
53	I want to allow exercise solvers to view the submissions of other exercise solvers.	1	2

I.: Importance, C.: Complexity, IID: Issue ID

4.4.4. Other

In addition to the requirements already discussed, we have formulated additional user stories that do not fit into any of the previous epics. The following table therefore shows all user stories of exercise makers that have not been mentioned before.

IID	Requirement	I.	C.
58	I want to provide automatic testing for an exercise.	5	5
↳ **59**	I want to prevent exercise solvers from optimizing their code specifically for the test cases.	2	3
35	I want to structure my exercises chronologically and thematically.	5	4
23	I want to incentivize exercise solvers to solve the provided exercises (e.g. with prices).	4	4
18	I want to provide exercises in collaboration with other exercise makers.	4	3
40	I want to provide my exercises to exercise solvers who have no knowledge of Git.	4	3
62	I want to allow exercise solvers to access help autonomously when problems arise.	3	3
22	I want to know how many exercise solvers are solving the provided exercises.	4	2
68	I want to allow exercise solvers to use the programming environment they are comfortable with (e.g. IDE)	2	4
60	I want to allow exercise solvers to test their code manually (e.g. using the main method)	2	2
61	I want to explain code to the exercise solvers using artificial intelligence.	1	4

69	I want to force exercise solvers to setup a local programming environment.	2	2
19	I want to provide exercises in different settings.	1	3
↳ **20**	I want to provide exercises for university courses (e.g. FH).	1	1
63	I want to allow exercise solvers to ask questions about exercises anonymously.	2	1

I.: Importance, C.: Complexity, IID: Issue ID

4.5. Requirements of Exercise Solvers

4.5.1. Exercise Empowerment and Simplification

The first epic we identified for exercise solvers is concerned with exercise empowerment and simplification. It contains multiple user stories expressing that exercise solvers would like to work on exercises as they see fit and that the barrier to entry (i.e. setup, installation) for exercises should be as low as possible. For this reason, we have defined the core requirement for this epic (#78) as follows:

> "As an exercise solver, I want to be in control of how I solve exercises."

The following table shows all user stories that extend this core requirement.

IID	Requirement	I.	C.
84	I want to solve exercises in an environment I am comfortable with (e.g. IDE).	5	5
↳ **94**	I want exercises to be easy to set up.	5	5
↳ **97**	I want exercises to be provided in a Git repository.	3	5
82	I want to be in control of where I solve exercises (e.g. campus, home).	4	4
79	I want to be in control of when I solve exercises.	5	3
↳ **80**	I want to save exercises in order to work on them at a later time.	5	2
83	I want to access exercises from different devices (e.g. mobile).	2	3
↳ **95**	I want to solve exercises in an online editor.	4	4
↳ **98**	I want to view the solutions of an exercise in a browser.	3	1
81	I want to be in control of which exercises I solve.	5	1
↳ **85**	I want to see the solutions of an exercise without having to solve it.	5	1

I.: Importance, C.: Complexity, IID: Issue ID

4.5.2. Exercise Overview

The last epic we identified is concerned with the overview of exercises. It contains multiple user stories expressing that exercise solvers would like to have a better overview over exercises in a given course, especially regarding which exercises are mandatory and which are optional. For this reason, we have defined the core requirement for this epic (#86) as follows:

> "As an exercise solver, I want to have an overview of my exercises."

The following table shows all user stories that extend this core requirement.

IID	Requirement	I.	C.
89	I want to know which exercises belong to which lecture (i.e. week).	5	2
↳ **90**	I want to know which exercises belong to the current lecture.	4	2
91	I want to know which exercises belong to which topic.	3	2
88	I want to know which exercises are mandatory.	4	1

I.: Importance, C.: Complexity, IID: Issue ID

4.5.3. Other

In addition to the requirements already discussed, we have formulated additional user stories that do not fit into any of the previous epics. The following table therefore shows all user stories of exercise solvers that have not been mentioned before.

IID	Requirement	I.	C.
93	I want to solve exercises that are similar to the exam.	5	5
100	I want to receive feedback for my exercise submissions.	5	5
↳ **107**	I want my solutions to be validated automatically.	4	5
↳ **101**	I want to view feedback together with my submission.	3	4
↳ **108**	I want to know why my solution is right or wrong.	5	1
↳ **102**	I want to receive feedback in the form of text.	4	1
↳ **103**	I want to receive feedback directly on the relevant lines of code.	1	4
104	I want to solve exercises in a reasonable amount of time.	3	4
105	I want to solve exercises multiple times (e.g. for repetition).	2	2
↳ **106**	I want to remind myself which exercises I would like to repeat.	1	1

I.: Importance, C.: Complexity, IID: Issue ID

5. Prototyping & Design

5.1. Exercise Creation, Migration and Management

5.1.1. Context

In this section, we will focus on requirement #5:

> "As an exercise maker, I want to reduce the overhead for creating and managing exercises."

5.1.2. Problem

In our user interviews, we discovered that one of the most significant problem exercise makers face in regards to their exercises is the time required to both create and manage them. This problem thereby arises not only from the many responsibilities exercise makers have in addition to their exercises (M4-Q1, M6-Q1), but also from the fact that many exercise makers are paid based on a predefined number of working hours (M3-Q8), meaning that they have to fit their responsibilities into a limited time frame if they do not wish to work for free.

While we do believe that a standardized application for managing exercises will inherently reduce the individual time required to create, setup and distribute exercises, we must also take into account that switching to a new software system will initially require additional work for exercise makers. If exercise makers do not deem this workload to be justified, our application will simply not be used and its entire purpose would be defeated. For this reason, we need to consider the different ways exercises are handled currently and must define strategies on how we can ensure an easy migration for exercise makers.

5.1.3. Proposed Solution

The solution we propose takes into account how exercises are handled currently and tries to integrate the different external applications already in use. Firstly, exercises are assigned to courses which resemble university courses or similar (#19, #20). For each course, we create an internal[4] Git repository to manage the files belonging to that course. Furthermore, some structural information about these files will be stored inside a database, allowing us to manage potential metadata about exercises without changing the original file structure[5].

Since many exercise makers already use GitLab, we allow our internal Git repository to be a clone of the existing GitLab repository (#3). Additionally, each course may reference an existing course in Adunis [Glossary]. If this reference is set, our application will automatically grant the correct access rights to exercise solvers based on the provided configuration (#14). This collaboration between our application and both GitLab and Adunis is illustrated in Figure 4.

[4]In this context, "internal" means that the repository is created at a location that the application has full control over.

[5]For example, such metadata may include the location of exercise files, the structure of a course or any additional configurations.

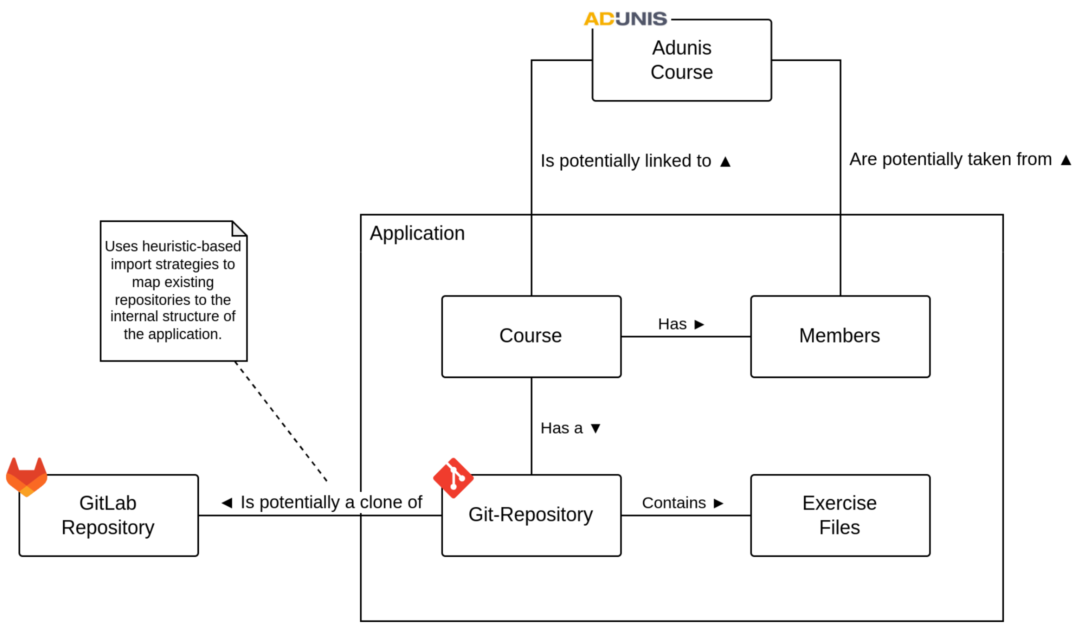

ADUNIS
Adunis Course
Is potentially linked to ▲
Are potentially taken from ▲
Application
Uses heuristic-based import strategies to map existing repositories to the internal structure of the application.
Course
Has ►
Members
Has a ▼
GitLab Repository
◄ Is potentially a clone of
Git-Repository
Contains ►
Exercise Files

Figure 4: Our application integrates with both GitLab and Adunis

5.1.3.1. Concept & Reasoning

We chose Git as our tool for managing exercises for two main reasons: Firstly, it allows us to easily integrate existing GitLab repositories into our application. If an exercise maker wants to create a new course using pre-existing data from a GitLab repository, we can create a clone of that repository and store it internally in our application, automatically retaining the original upstream repository as a Git remote.

This interplay between our application and existing GitLab repositories is illustrated in Figure 5. In this example, an exercise maker creates a new course by providing a GitLab repository. The repository is then imported into our internal data structure, which is described in more detail in Section 5.1.3.2.

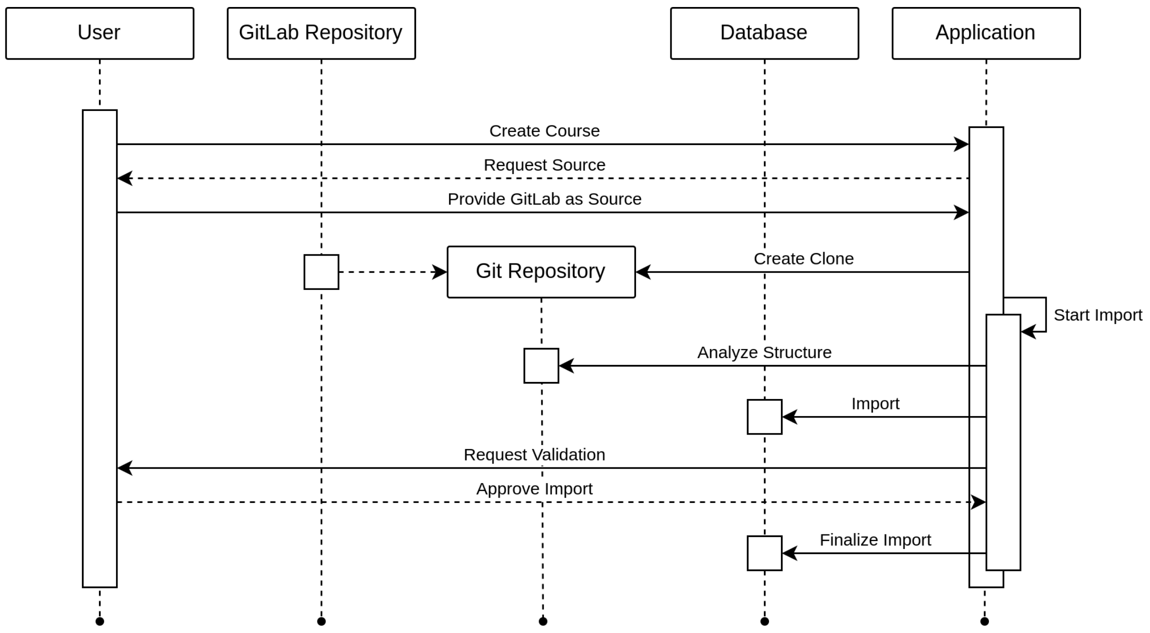


Figure 5: A course can be imported from an existing GitLab repository

In addition, using an internal Git repository also allows us to manage exercises of exercise makers who do not use GitLab (#72) or want to make changes to the exercises through a user interface (#73). In this case, all changes will create new commits in our internal repository. If an existing GitLab repository has been set as remote, we can furthermore trigger a pull request to the original repository to keep the exercises in sync. This situation is illustrated in Figure 6.

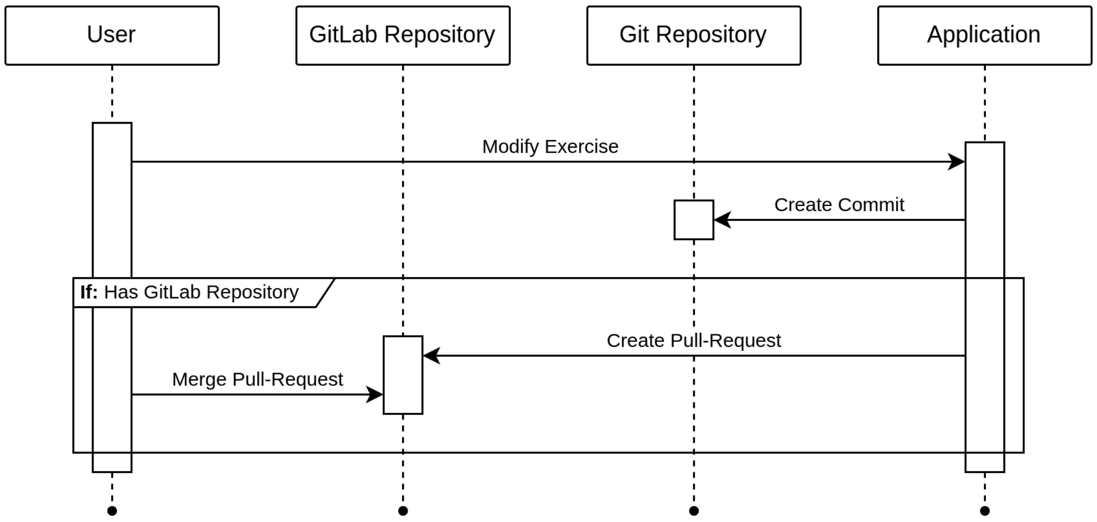


Figure 6: Modifications to exercises can trigger pull requests to the original repository

The second reason we chose Git is that it allows us to use the versioning features already present in the software. This means, for example, that changes to exercises may be done in a different branch that is only published to the exercise solvers when merged into the main branch. Furthermore, we can prevent an ecosystem lock-in by always allowing exercise creators to clone the internal repository, even if it was created entirely within our application. By opting for Git, these features are given to us without any additional development costs.

5.1.3.2. Import & Heuristics

As discussed in Section 5.1.3, we aim to reduce the workload required for migration by allowing existing courses to be imported into our application (#16). Since there is currently no standardized approach on how exercises are held or structured (M3-Q6), we must implement different import heuristics and strategies in order to match the existing exercises to our internal structure (see Section 5.2)[6].

There are several problems we must thereby solve. For example, one of the problems we face is recognizing the hierarchical and chronological structures of different courses. By analyzing different GitLab repositories at OST, we discovered that many of them structure their exercises using folders that represent weeks, exercises or similar. This means that recognizing such patterns in a file system would allow us to automatically import such hierarchical structures.

[6]For consistency, "migration" refers to the process of exercise makers switching to our application, while "import" refers to the process of storing an existing course in our internal structure. This is because "migration" usually implies that something is being moved or permanently changed, which only applies to the former but not the latter.

In our technical prototype, we have successfully evaluated a heuristic in which all folders containing a numeric part in their name are grouped together if their non-numeric part match. Furthermore, we noticed that the largest group with a matching prefix or postfix is often the primary structure of the course. This is illustrated in the following example:

```
└─ course          ── repository root
  └┬ .git          ── ignored as it contains no numeric parts
   ├ week 1        ┐
   ├ week 2        ┼─ selected as part of the prefix group: "week "
   ├ week 3        ┘
   ├ example-1     ┐
   └ example-2     ┴─ not part of the largest prefix group
```

The same heuristic can also be applied to detect exercises in a folder:

```
└─ course                ── repository root
  └┬ ...                 ── other folders emitted for brevity
   ├ week 2              ── a folder containing exercises
   │ └┬ exercise 1       ┐
   │  ├ exercise 2       ┼─ largest prefix group: "exercise "
   │  ├ exercise 3       ┘
   │  └ solutions.pdf    ── files and other folders are ignored
   └ ...
```

5.1.3.3. Patterns & Strategies

The above example is just one of many possible heuristics to simplify the import of an existing GitLab repository into our internal data structures, reducing the manual effort needed when migrating to our application. To allow such heuristics to coexist in a reusable, maintainable and testable way, we first considered the *Blackboard* pattern (Buschmann et al., 1996). This is an approach for solving non-deterministic problems at best-effort level by combining different techniques, so-called *Knowledge Sources*.

However, our initial approach of using the *Blackboard* pattern in our prototype, while working, proved to be quite complex at the *Knowledge Source* level due to the recursive nature of both the file system and our course design (see Section 5.2 ff.). For this reason, as part of our Bachelor's thesis, we will likely refactor the current prototype into a *Visitor* pattern (Gamma et al., 1994) that traverses the file structure while using the *Knowledge Sources* at a more granular level. This will avoid multiple full passes over the already analyzed structure, allowing an easier combination of multiple heuristics into partial results.

5.2. Exercise Variations

5.2.1. Context

In this section, we will focus on requirement #27:

"As an exercise maker, I want to provide a wide variety of different exercises."

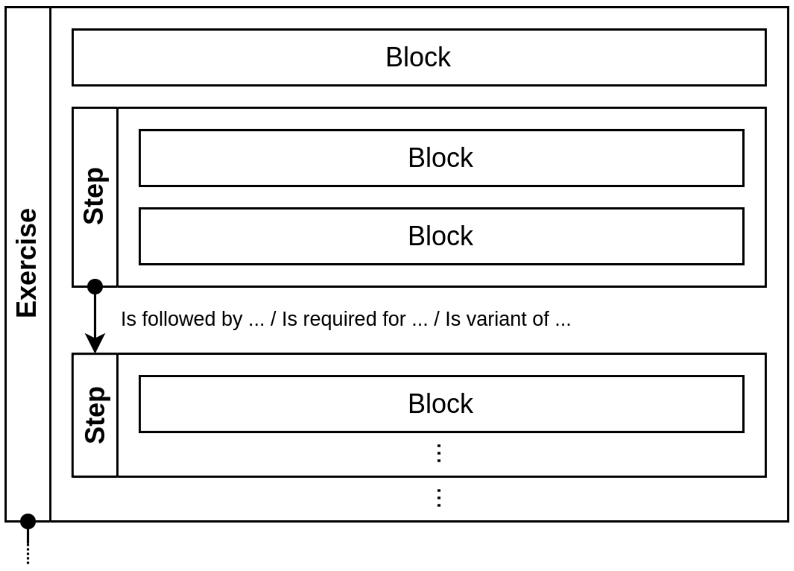
5.2.2. Problem

From the information we gathered in our user interviews, we found that a wide variety of different exercise formats exist within OST. These variations include, for example, exercises that require implementing certain algorithms (M1-Q2), using certain external tools (M5-Q6, M7-Q3) or creating diagrams on certain domain-specific problems (M4-Q2). By analyzing some exercises in more detail (see Section 5.2.3.1), we also know that these exercises are usually not standardized across different courses or even across different lectures within the same course.

While this seems to be a natural result of the loose collaboration of exercise makers (M3-Q6) and the fact that exercises usually evolve over an extended period of time (M7-Q5), it poses a problem if we want to design a data structure that can be universally used by different exercise makers in different settings. Furthermore, we must also consider that our user interviews may not cover all existing exercise formats, which means that our data structure must allow extensions whenever new variants of exercises are discovered.

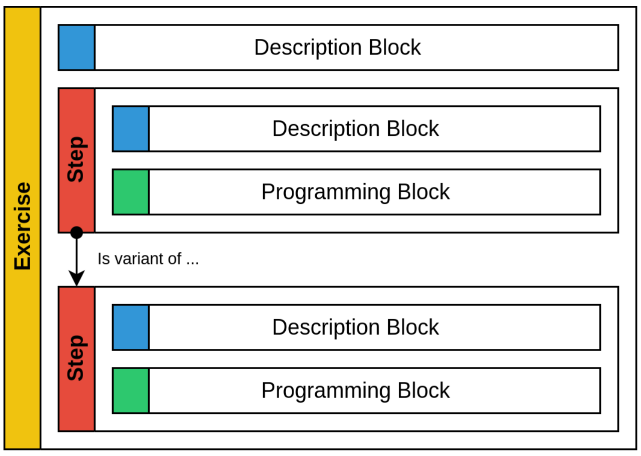
5.2.3. Proposed Solution

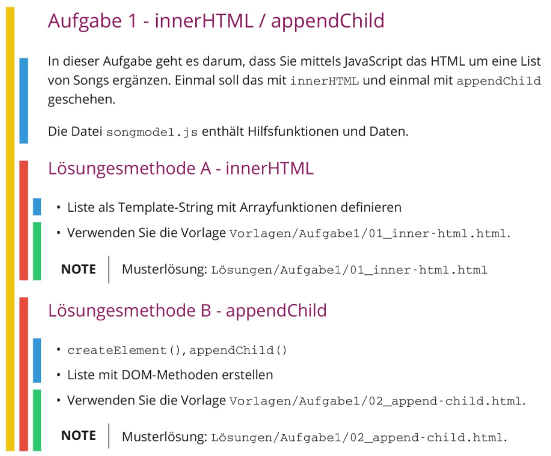
The solution we propose tries to solve the problems mentioned by using a modular data structure for exercises. This means that each exercise is divided into one or more "blocks", which furthermore can be grouped into so-called "steps". A block thereby represents some kind of content within an exercise. For example, a block might represent a text block, a programming block (#29), a multiple-choice block (#28), a block that accesses external resources (#33) or similar. Both steps and blocks are designed to be composed arbitrarily, allowing a wide variety of exercise formats to be included in our application. Finally, we also allow exercises and steps to be linked with different logical bindings, allowing us to model dependencies that are currently usually expressed using the exercise description.


Exercise
Block
Step
Block
Block
Is followed by ... / Is required for ... / Is variant of ...
Step
Block
⋮
⋮
Figure 7: Our exercises will use a modular data structure

5.2.3.1. Concept & Reasoning

The data structure we propose allows us to represent a wide variety of existing exercises in our application. This claim can be validated by translating existing exercises from different courses into this data structure, which is demonstrated in Figure 8 and furthermore in the appendix under Section VII.




Aufgabe 1 - innerHTML / appendChild

In dieser Aufgabe geht es darum, dass Sie mittels JavaScript das HTML um eine List von Songs ergänzen. Einmal soll das mit `innerHTML` und einmal mit `appendChild` geschehen.

Die Datei `songmodel.js` enthält Hilfsfunktionen und Daten.

Lösungesmethode A - innerHTML

- Liste als Template-String mit Arrayfunktionen definieren
- Verwenden Sie die Vorlage `Vorlagen/Aufgabe1/01_inner-html.html`.

NOTE Musterlösung: `Lösungen/Aufgabe1/01_inner-html.html`

Lösungesmethode B - appendChild

- `createElement(), appendChild()`
- Liste mit DOM-Methoden erstellen
- Verwenden Sie die Vorlage `Vorlagen/Aufgabe1/02_append-child.html`.

NOTE Musterlösung: `Lösungen/Aufgabe1/02_append-child.html`.

Figure 8: Our data structure is able to represent a variety of different exercises

Additionally, dividing exercises into blocks allows us to implement arbitrarily complex logic on a block level, ensuring a strict separation of concern. For example, a text block may contain logic to render text from different markup languages, while a programming block may handle rendering an online editor and sending submitted code to an external test runner. These ideas are explained in more detail in Section 5.3 when talking about exercise submissions.

Finally, our data structure also allows implementing additional blocks whenever new variations of exercises are discovered. While we do not want to make any concrete architectural decisions at the moment, we believe that this could even take the form of a plugin-based system in which exercise makers can program their own blocks as needed.

5.2.3.2. Course Structure

To structure exercises within a course, we introduce the concept of both “groupings” and “grouping kinds”. A grouping kind thereby represents some higher-level theme (e.g. week) containing a set of chronological groupings (e.g. week 1, week 2, etc.). Exercises can then be assigned to exactly one grouping per grouping kind, allowing them to be structured both thematically and chronologically within a course (#35). This is demonstrated in the following example, which shows a course consisting of four groupings separated into two grouping kinds (i.e. “week” and “topic”).

```
└─ course             ─── repository root
  └┬ exercise 1       ─┐              ─┬─ topic 1
   ├ exercise 2       ─┼─ week 1      ─┘
   ├ exercise 3       ─┘              ─┐
   ├ exercise 4       ─┐              ─┼─ topic 2
   └ exercise 5       ─┴─ week 2      ─┘
```

The structural composition of a course is therefore illustrated in Figure 9.

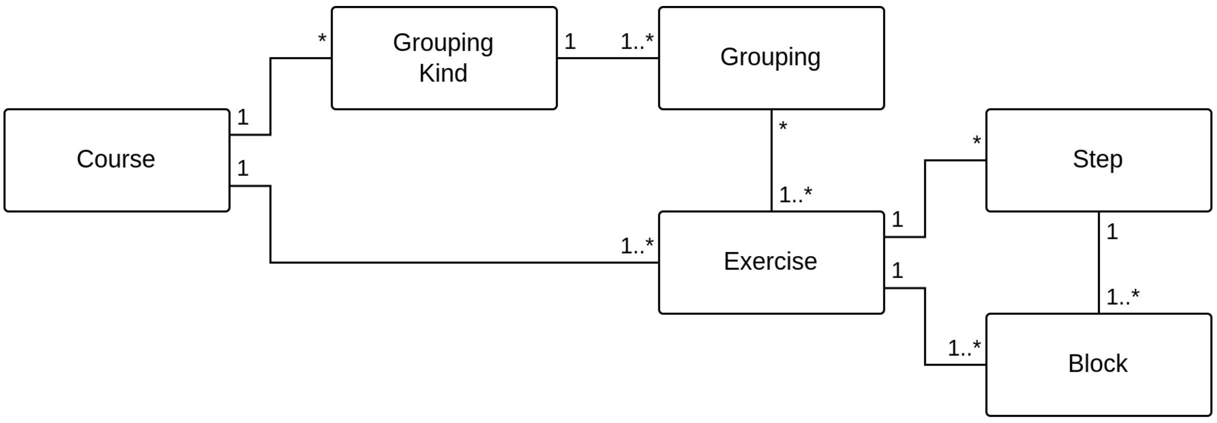


Figure 9: The structural composition of a course

5.3. Exercise Submissions

5.3.1. Context

In this section, we will focus on requirement #42:

> "As an exercise maker, I want exercise solvers to submit their solutions for an exercise."

5.3.2. Problem

The problem with this requirement is not necessarily how exercise solvers can submit their solutions in our application. As we will see, this can be solved by using the architecture presented in Section 5.1 (i.e. by using an internal Git repository).

The more critical problem with this requirement is how we can model the different demands that the exercise makers place on the submissions themselves. For example, we know from our user interviews that some mandatory programming exercises are considered fulfilled when a certain number of automated test cases have been passed (M6-Q4). In other cases, however, an exercise is considered fulfilled when manual feedback has been provided by an exercise maker (M2-Q7, M6-Q4) or even by another exercise solver (M4-Q8). While we can solve these issues on a block level, doing so might result in redundancies if certain functionalities can be applicable for multiple types of blocks. Furthermore, we must once again consider that there may be additional requirements for submissions that we have not yet discovered.

5.3.3. Proposed Solution

The solution we propose expands on the modular data structure introduced in Section 5.2. Firstly, blocks, steps and exercises can now be extended with attributes that provide additional functionalities to each element. With this, we can model situations where an element may be mandatory (#43), where an element may allow providing different forms of feedback (#46, #48, #49) or where an element may be evaluated using automatic tests (#58) or user-defined scripts (#70).

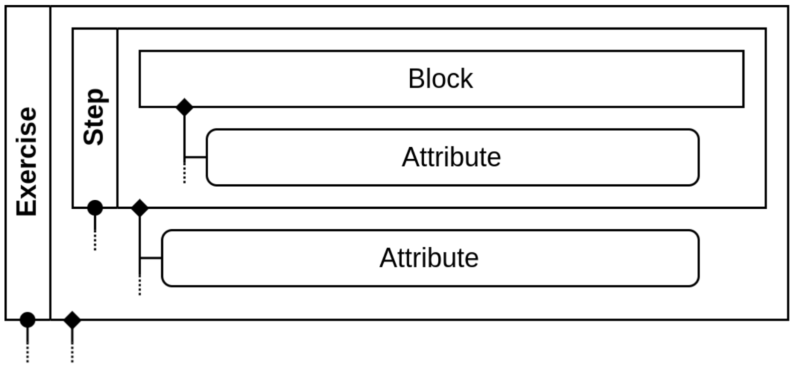


Figure 10: Attributes extend our modular data structure with additional functionalities

In addition to attributes, we also introduce the concept of input and output for all elements. This means that an element may receive input that affects its internal state and that an element may produce output that affects any external state. For example, a `ManualFeedback` attribute may produce an output called `isSolutionAccepted`, which can then be passed to the `isFulfilled` input of a `Mandatory` attribute. Finally, we allow these inputs and outputs to be combined using a series of logical (e.g. `And`, `Or`) and conditional nodes (e.g. `GreaterThan`, `Equal`). By doing this, we create a system that allows different exercise forms and variants to be modeled entirely within our application.

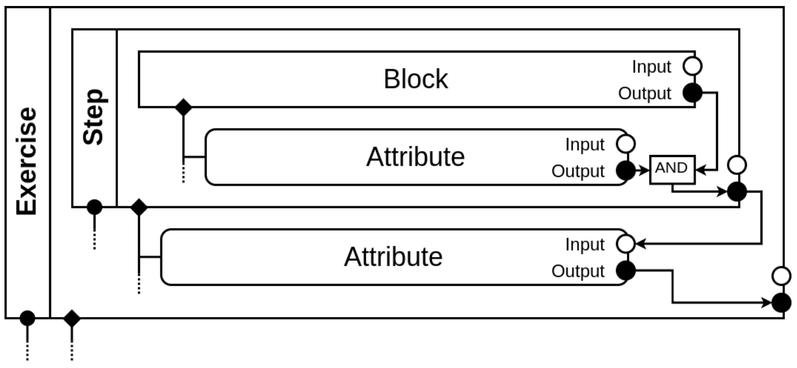


Figure 11: Inputs, outputs and nodes allow different exercise flows to be modeled

5.3.3.1. Submission System

Once an exercise maker has defined the necessary attributes on an exercise, we can determine which parts of it can be modified and submitted by exercise solvers[7]. With this information, we can create a clone of the course repository (see Section 5.1.3) for each exercise solver containing only the editable files. Whenever an exercise solver now changes these files, we can create a new commit with all the changes in this personal repository. Each commit is therefore considered a submission within our application. As explained in Section 5.4.3, the same principles can also be applied when changes are made locally.

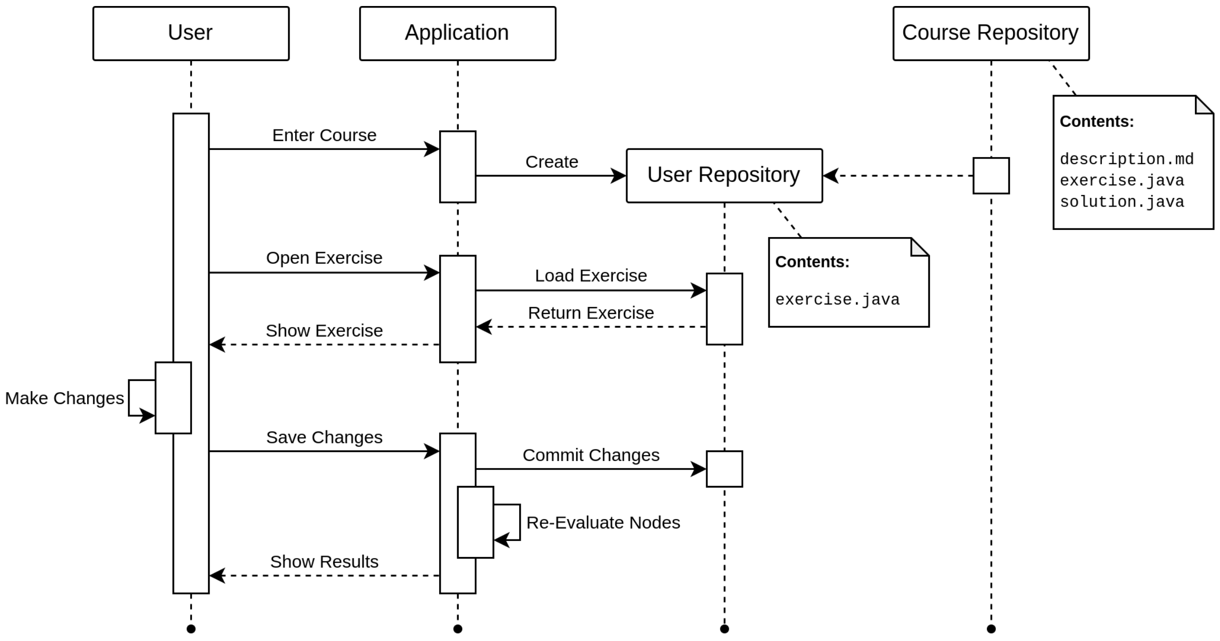


Figure 12: Each exercise solver can make changes to exercises within a personal repository

[7]For example, the code of a programming block may be modifiable, while the test cases are not.

As illustrated in Figure 12, all nodes (i.e. inputs and outputs) of an exercise are re-evaluated whenever a submission is made. This means, for example, that a submission may trigger automatic tests from which the resulting test cases determine whether a mandatory exercise has been fulfilled.

5.3.3.2. Submission View

In our application, we will provide a separate view for exercise makers in which they can see all submissions for a given exercise. Since all commits are considered submissions, an exercise maker will not only be able to see the final solution (i.e. the latest commit), but also how that solution was reached. Furthermore, since we are now in full control of the submitted data, we can perform various statistical analyses of exercises. For example, we can determine how many people have submitted a solution for an exercise (#22) or how many lines of code were written on average to solve a programming assignment[8].

5.3.3.3. Gamification

Until now, the gamification aspect of our application has been mostly put aside in order to conceptualize a system that provides added value even without game-like elements. However, now that we have understood how the creation and submission of exercises works internally, we can talk about how we can introduce gamification into our application.

Firstly, we want to talk about Development & Accomplishment as described by Yu-kai Chou in his Octalysis framework (Chou, 2015). The concept of input and output that we introduced in Section 5.3.3 can not only be used to configure the behavior of exercises, it also allows us to implement gamification based on the actions of exercise solvers. This means that we can reward Points or Badges when certain criteria are reached[9]. These ideas are expanded upon in Section 5.4 (ff.) when discussing the requirements of exercise solvers.

Functionally, this can be modeled by introducing gamification nodes alongside the logical and conditional nodes already described. While some gamification mechanisms may be implemented on an application-wide scope, allowing exercise makers to use gamification nodes in the same way as they use other nodes might lead to more specialized and creative experiences for exercise solvers. At the same time, we recognize that the misapplication of gamification can cause considerable harm to player engagement, which is discussed in depth in the works by Yu-kai Chou (Chou, 2015). For this reason, we reserve the right to disable these gamification nodes for exercise makers until we know that they understand how gamification should be applied[10].

[8]We understand that not all exercise solvers may want their data to be shared with exercise makers and will therefore ensure that this can be disabled in accordance with data protection laws.

[9]For example, an exercise solver may receive a “Java Guru” badge when all exercises in a Java course have been completed with at least 90% of test cases passing.

[10]For example, this could be done by offering “meta-courses” that unlock different types of gamification nodes.

5.4. Exercise Empowerment and Simplification

5.4.1. Context

In this section, we will focus on requirement #78:

“As an exercise solver, I want to be in control of how I solve exercises.”

5.4.2. Problem

In our user interviews, we discovered that there are two problems related to this epic that explain why exercise solvers do not consistently engage with exercises. Firstly, many exercise solvers are put off by the setup necessary to start working on an exercise in the first place (S1-Q4, S3-Q5). As we have learned, exercise solvers often have to install environments and tools that they will either never use outside of their lectures or that require simply too much effort to get working (M4-Q4). Secondly, many exercise solvers seem to be discouraged by the prerequisites that some exercise makers put on their exercises, like requiring that exercises need to be solved in a certain order or that certain exercises must be completed in order to be admitted to the exam (S4-Q8).

While we believe that the first problem can be solved by expanding on the various concepts already introduced in previous chapters, we think that the second problem is more deeply rooted in the inherent conflict between the interests of exercise makers and exercise solvers. For this reason, we must consider how our application should deal with this conflict between our two user groups and develop a solution that is appropriate for the academic context.

5.4.3. Proposed Solution

The solution we propose solves many of the functional concerns of this requirement by determining that the application should be developed as a web-based platform. By choosing a web-based approach rather than a native solution, we can ensure that exercise solvers can access their exercises where they want (#82), from which device they want (#83), and when they want (#79).

Since our application will use internal Git repositories to manage the files and submissions of each exercise solver, we can furthermore allow great flexibility in which tools may be used to work on the provided exercises. While we believe that all exercises should be entirely solvable within our application (#95), as this completely reduces the setup time needed to start working on an exercise (#94), giving exercise solvers the ability to clone the internal repository (#97) allows them to work in whatever environment they feel most comfortable with (#84). This idea is illustrated in more detail in Figure 13.

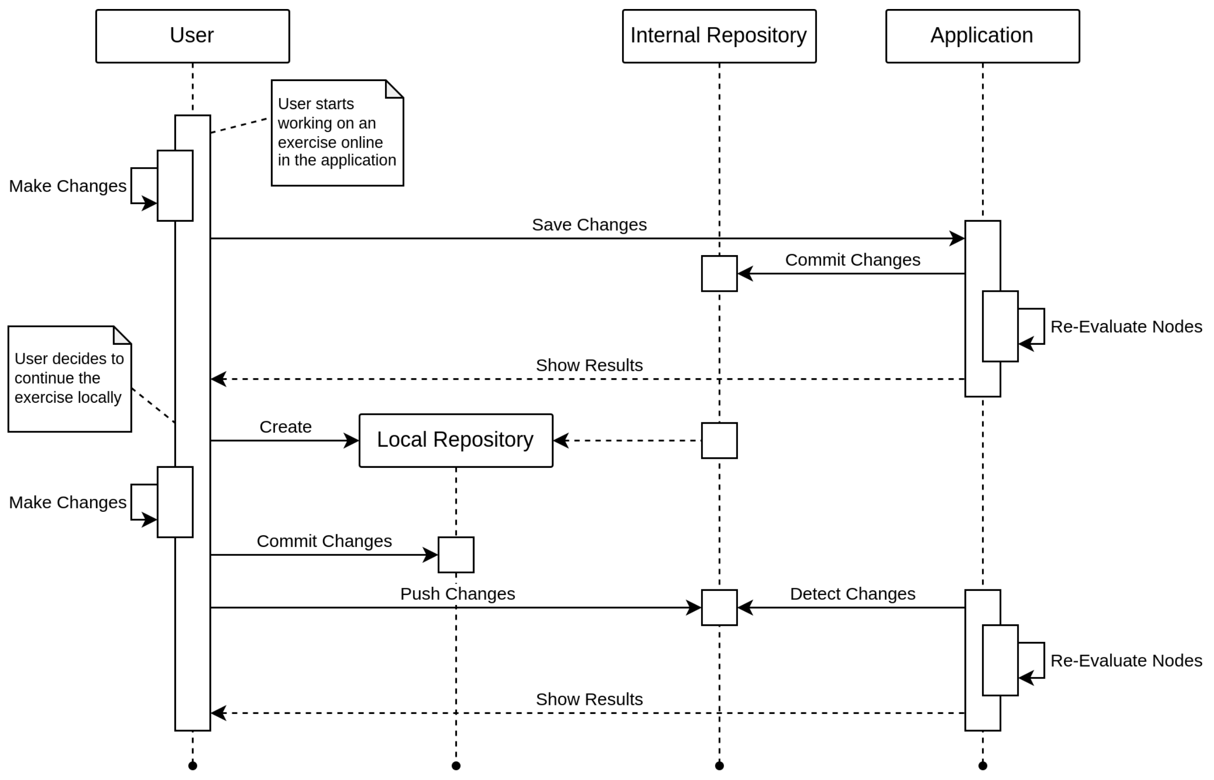


Figure 13: The exercise solvers can work both online and locally

Figure 13 also takes into account two other aspects: Firstly, the submission mechanism introduced in Section 5.3.3.1 is still applicable even if exercise solvers choose to work in a local repository, with the only difference being that submissions are not recognized until the changes have been pushed to the internal (i.e. remote) repository. Secondly, since all evaluation is done on the application side, an exercise solver is not required to install any programs or tools in order to work on an exercise (#96). This means, for example, that a Java exercise may be solved locally without ever having to install the Java Development Kit[11] or even a proper IDE [Glossary].

5.4.3.1. Conflicting Interests

With our proposed solution, we believe that we can reduce the effort required by exercise solvers to start working on their exercises, while at the same time allowing them to be more flexible in how they want to work. However, we have yet to address the problem that the way exercise solvers want to work does not necessarily correspond to the way exercise makers would like their exercises to be worked on[12]. To better understand this conflict, we have tried to formulate some scenarios that show examples of this problem:

- As an exercise solver, I want to solve exercises when I want.
- As an exercise maker, I want exercises to be solved for or after a given lecture.

- As an exercise solver, I want to solve exercises where I want.
- As an exercise maker, I want exercises to be solved in the given exercise lesson.

[11]This was jokingly described as a “dream come true” by one of the project members.

[12]This claim is derived from our assignment, because if exercise makers were satisfied with the way exercise solvers work, there would be no reason to write this thesis in the first place.

- As an exercise solver, I want to view solutions of an exercises when I want.
- As an exercise maker, I want solutions to only be viewed after an exercises has been attempted.

While we respect the didactic reasoning of exercise makers, we believe that in order to resolve this conflict we must prioritize the interests of exercise solvers. The reason for this decision is that we must not interfere with the core interests of exercise solvers, even if these interests are not as sustainable as those of exercise makers. Examples have shown that exercise solvers will go to great lengths to circumvent restrictions put in the way of achieving their core interests[13]. This means that our application should always adhere to these principles:

- Exercises must not be limited to a certain time frame (unless mandatory)
- Exercises must not be limited in the number of submissions
- Solutions must not be limited to solving the exercises first
- Gamification must not violate these principles

5.4.3.2. Gamification

Although we have already stated that we do not want to interfere with the core interests of exercise solvers, we believe that there may be ways to use gamification in order to promote a more sustainable way of working. This could be done by using several techniques that fall under the core drives of Scarcity & Impatience and Loss & Avoidance (Chou, 2015).

For example, we mentioned that exercise makers would like exercise solvers to attend their exercise lessons regularly. While we cannot force exercise solvers to attend these lessons (see principles in Section 5.4.3.1), we can use mechanisms such as Time-Limited Rewards to incentivize them to be there in person. These rewards could take the form of badges as introduced in Section 5.3.3.3, which can only be obtained by scanning a QR code given out in the exercise lesson itself. Alternatively, we could specify that all earnable points for an exercise are doubled if it is solved within the corresponding week. We could even go as far as creating Collection Sets for courses in which a set may only be completed if an exercise solver has attended a given number of exercise lessons.

Similar ideas could also be applied when considering that exercise makers would like exercise solvers to attempt an exercise before looking at its solutions. Here we could introduce Progression Limitations in which rewards for an exercise may be blocked or limited if its solution is looked at beforehand[14]. While such systems are not without flaws[15], it would also allow us to add more value to some of the other gamification mechanisms we have introduced. For example, points could now be

[13]Anecdotally, there are stories of exercise solvers sharing Excel spreadsheets that match the public IDs of test cases to potential fixes in the code, allowing them to solve exercises faster even though the specific test cases are hidden.

[14]For example, there may be a “Model Student” badge that requires a hundred exercises to be solved. If an exercise solver chooses to look at the solution without attempting the exercise first, we could block the exercise from contributing to that badge

[15]For example, consider exercise solvers who decide one year down the line that they would like to pursue certain badges, just to realize that they have been locked out from most of them.

transformed into 🕹️ Exchangeable Points that may be spent to acquire “free passes” which allow solutions to be viewed without loss of progression[16]. Since we consider gamification to be an optimal part of our application, this could all be done without violating the principles defined in Section 5.4.3.1.

[16]Duolingo [Glossary] uses a similar system called “Streak Freeze” to prevent streaks from ending when a user does not have time to complete an exercise (Duolingo, n.d.).

5.5. Exercise Overview

5.5.1. Context

In this section, we will focus on requirement #86:

“As an exercise solver, I want to have an overview of my exercises.”

5.5.2. Problem

A common concern exercise solvers voiced in our user interviews is that information about a course and its exercises is often communicated in a non-transparent way (S4-Q9). Because of the loose collaboration between exercise makers, information about courses is often scattered across different channels (e.g. Teams and Moodle), making it difficult for exercise solvers to find the information they need (M3-Q6). Furthermore, we discovered that even exercises within the same course are sometimes distributed inconsistently, mixing different platforms and tools that require additional work to discover and learn (M1-Q5, M7-Q3).

While some of these problems are indirectly solved through various concepts discussed in previous chapters, we must think about how we can effectively communicate information about both exercises and courses. This is not only limited to communication features, such as email notifications or similar, but also requires thinking about different aspects of the user interface.

5.5.3. Proposed Solution

The solution we propose tackles this problem from multiple angles. Firstly, we can solve some of the mentioned issues by building a user interface that focuses on providing concise information about the exercises of a course. Based on our user interviews and the architectural decisions made in previous chapters, we believe that such an interface must include the following:

- An overview of all exercises on a single page
- The relationships between these exercises (e.g. is required for)
- The structural grouping of an exercise (e.g. lecture, week)
- The thematic grouping of an exercise (e.g. topic)
- The progress or state of an exercise (e.g. exercises done)
- The relevancy of an exercise (e.g. is mandatory)
- The recommended exercises at any given time (e.g. of the current week)

To better illustrate these ideas, we created a Figma prototype (see Figure 14) which shows how this user interface could look like. In this example, we show the main page of a course consisting of two lectures (i.e. weeks) and seven exercises.

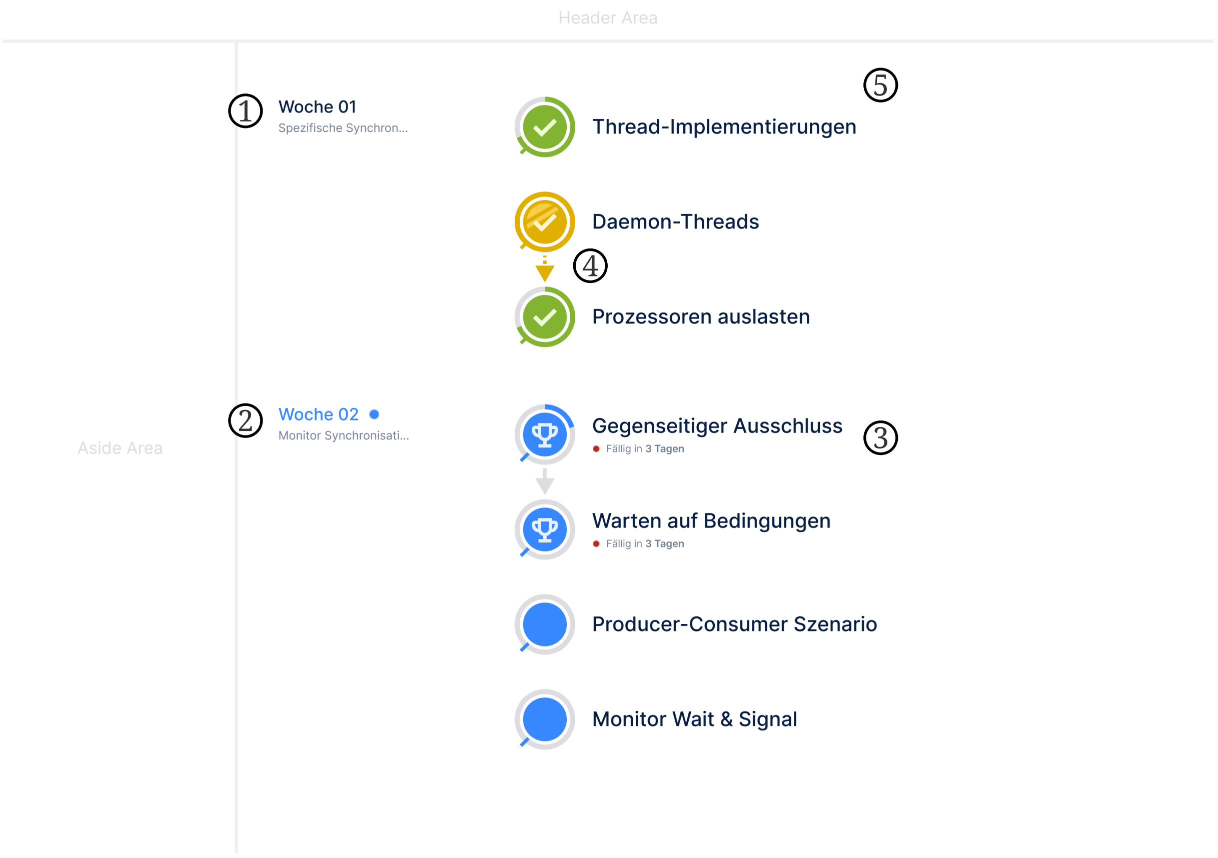


Figure 14: The exercise view shows all exercises of a course in one central location

5.5.3.1. Concept & Reasoning

The user interface presented in Figure 14 is designed to effectively communicate which exercises belong to which week and topic ① (#89, #91), which belong to the current week ② (#90), which are mandatory ③ (#88), and which exercises have relationships to other exercises ④. In addition, all exercises have been moved to the top level of a course in order to reduce nesting and improve navigation ⑤.

Our goal was to implement a concept known in didactics as a learning path (*de. Lernpfad*), which gives the exercise solvers a predefined structure on how they may solve their exercises (i.e. from top to bottom), while still allowing them to pick and choose which exercises they want to work on (Universität Zürich, n.d.). Furthermore, by introducing a Progress Bar for exercises, we can ensure that exercise solvers always know where they stand within this learning path.

5.5.3.2. Gamification

The implemented learning path in combination with the progress bar is further inspired by "research trees" found in various video games (Wikipedia, 2023a). With this, we aim to visually connect our user interface to the aesthetics of these systems.

In addition to the progress bar, we also introduce 4 different states for exercises:

- **Idle**: Exercises that have not yet been started are considered "idle", which is indicated by a blue color and an empty progress bar.

- **Started**: Exercises that have been started but not yet completed are considered “started”, which is indicated by a blue color and a partially filled progress bar.
- **Done**: Exercises that are completed to a certain percentage (e.g. 75%) are considered “done”, which is indicated by a green color and a partially filled progress bar.
- **Golden**: Exercises that are fully completed are considered “golden”, which is indicated by a gold color and a completely filled progress bar.

The idea behind the “done” and “golden” states are two-fold: Firstly, by considering an exercise done even if it is not fully completed, we are trying to reduce the motivation required to start working on that exercise by making the “win state” easier to achieve. Secondly, by adding a “golden” state, we aim to motivate exercise solvers to take the extra time to solve some of the exercises to completion. The psychological reasoning behind these game techniques is discussed in the works of Yu-kai Chou (Chou, 2015). Furthermore, these mechanisms also take into account the principles introduced in Section 5.4.3.1 by focusing on incentivizing the completion of an exercise rather than enforcing it.

5.5.3.3. Communication Mechanism

In addition to a concise user interface, we must also consider how different information can be communicated between exercise makers and exercise solvers. As this is a broad topic that touches all aspects of our application, we want to focus on two examples to illustrate how our application should handle communication mechanisms.

Firstly, we believe that exercise solvers should be able to ask questions about exercises directly on an exercise itself. This could take the form of a comment section for exercises where different questions can be discussed, upvoted and shared with others[17]. With this, we believe that exercise solvers will be able to find and solve questions about exercises more efficiently as they will no longer be discussed in Moodle forums or Teams chats where they first need to be discovered.

Secondly, we think that changes to existing exercises should be communicated with exercise solvers if they occur while a course is running. This could be done by tagging an exercise with a special symbol indicating that it has been changed and might require resolving. This is especially important for mandatory exercises where a change may result in an exercise solver not passing a course even though they did so previously. To ensure this information is received, we may even need to send out email notifications instructing exercise solvers to resubmit their work.

These two examples illustrate how communication should be handled within our application. Our goal thereby is to ensure that relevant information is relayed consistently and transparently in order to reduce the mental overhead for both exercise makers and exercise solvers.

[17]Codewars [Glossary] has a similar feature called “Kata Discourse” where users can discuss issues, suggestions or questions directly on an exercise (called a *Kata*) itself (Codewars, n.d.).

6. Minimal Viable Product

6.1. Context

Having conceptualized an application in Section 5 that addresses the different problems and requirements we discovered in our user interview, it is possible to define the Minimum Viable Product (MVP) that we aim to develop as part of our Bachelor's thesis. To do this, we will first define a concrete list of functional requirements our application should contain and will then determine which technologies best fit these requirements.

6.2. Feature Scope

6.2.1. Must-Have Criteria

The following table contains all functional requirements that we consider to be necessary in order for our application to be usable, while still being realizable within the given time frame of our Bachelor's thesis. An extensive discussion on how we plan to realize these requirements can be found in Section 5.

IID	Requirement
112	The application contains a course selection.
↳ **111**	The course selection shows all courses a user belongs to.
113	The application contains an exercise selection.
↳ **114**	The exercise selection shows all exercises of a course.
↳ **115**	Each exercise has a progress bar.
116	The application contains an exercise view.
↳ **117**	The exercise view shows the contents of an exercise.
↳ **118**	An exercise can contain an online editor for coding tasks.
↳ **119**	An exercise can contain a description.
↳ **120**	An exercise allows the submission of a solved coding task.
123	The application allows working locally using Git.
↳ **124**	A course can be cloned as a Git repository.
↳ **125**	An exercise can be solved on a separate branch.
126	The application allows CRUD on courses and exercises.
↳ **127**	A new/empty course can be created.
↳ **128**	A new/empty exercise can be created.
↳ **129**	An exercise can be edited.
↳ **130**	An exercise can be restructured in its grouping.
↳ **131**	An exercise can be restructured in its order.
133	The application allows importing courses and exercises.
↳ **134**	An existing course can be imported from GitLab.
↳ **135**	An existing exercise can be imported from an archive or folder.

↳ **136**	The title of a course is determined heuristically.
↳ **137**	The weekly structure of a course is determined heuristically.
↳ **138**	The description of an exercise is determined heuristically.
↳ **139**	The coding files of an exercise are determined heuristically.
↳ **140**	The order of an exercise is determined heuristically.
144	The application supports OAuth integration.
↳ **145**	A user can log in using OAuth.
↳ **146**	The role of a user is automatically determined using OAuth.
↳ **147**	A course can only be edited by its owner.

6.2.2. Should-Have Criteria

The following table contains additional requirements that we consider feasible to be implemented as part of our Bachelor’s thesis. However, if their technical complexity does not match our predictions, we may replace or remove any of these criteria in order to ensure that our must-have features are prioritised.

IID	Requirement
116	The application contains an exercise view.
↳ **121**	Submitted coding tasks are compiled automatically.
↳ **122**	Submitted coding tasks can be automatically validated using test cases.
126	The application allows CRUD on courses and exercises.
↳ **132**	The test cases of a coding task can be configured.
133	The application allows importing courses and exercises.
↳ **141**	The solution of an exercise is determined heuristically.
↳ **142**	The test cases of an exercise are determined heuristically.
↳ **143**	The Adunis reference of a course is determined heuristically.
144	The application supports OAuth integration.
↳ **148**	The application can gain permissions to read GitLab repositories.
↳ **149**	Additional allowed editors can be added to a course by its owner.
↳ **150**	The access rights for a course are automatically imported from Adunis.

6.3. Technologies

6.3.1. General

Knowing the scope of the MVP as well as having a rough understanding of the long-term development of our application, we are now able to establish a technology stack that fits the requirements of our software system. This means, generally speaking, that we need to select technologies for the following areas:

- Web Server
- Front-End
- File Storage with Versioning
- Database
- Deployment

6.3.2. Criteria

In order to choose between the many technologies available, we have defined several criteria that we consider significant for the development of a long-lasting application. These criteria are in order of importance:

- **Open Source Availability**: Is the technology open source?
- **Long-Term Support**: Is the technology widely used and likely to receive updates?
- **Familiarity**: Do we and the employees at OST (Department of Computer Science)[18] have any prior experience with the technology?
- **Scalability**: Does the technology scale with the potentially growing requirements?
- **Ease of Replacement**: Can the technology be easily replaced in the future?
- **Cost of Setup**: Is the technology easy to set up and run?

6.3.3. Chosen Technologies

6.3.3.1. Core Technologies

On the basis of our defined criteria, we have chosen the following technologies for the development of our application:

- Web Server: ASP.NET [Glossary] with C# [Glossary]
- Front-End: React [Glossary]
- File Storage with Versioning: Git [Glossary]
- Database: SQLite [Glossary]
- Deployment: OST Portainer [Glossary] Cluster

These technologies best meet the criteria we have specified. Notable exceptions are SQLite and Portainer, which we believe to be insufficiently scalable if the application is to grow over time[19]. However, their low cost of set up in combination with their ease of replacement (when using an ORM [Glossary] and Docker [Glossary] respectively) allows us to replace these technologies in the future. Alternative technologies include

[18]We included the Department of Computer Science in our criteria for choosing a technology because it is the most likely place where our application will be maintained after the completion of our Bachelor's thesis.

[19]Please note that in the case of Portainer, the technology itself is not the problem, but rather the corresponding OST cluster, which has been described as insufficient in some cases (M4-Q6).

PostgreSQL [Glossary] and MariaDB/MySQL [Glossary] for SQLite and Kubernetes [Glossary] for Portainer. There may also be other higher-performance clusters at OST that could be used for the deployment of our application.

6.3.3.2. Packages & Libraries

In addition to our core technologies, we have also determined a non-exhaustive set of packages and libraries that we intend to use for the development of our application:

- Testing Library: xUnit.net [Glossary]
- Component Library: Radix Primitives [Glossary] and Base UI [Glossary]
- Inversion of Control Library: Autofac [Glossary]
- Git Integration Library: LibGit2Sharp [Glossary] (based on libgit2 [Glossary])
- ORM Library: Entity Framework Core [Glossary]

6.3.4. Considered Alternatives

6.3.4.1. Blazor

The greatest advantage of Blazor [Glossary] in combination with C# would be the ability to share code between the backend and frontend. However, the lack of mature components and packages to use when developing Single Page Applications (SPA [Glossary]) was the main reason we chose React.

6.3.4.2. Svelte & Vue.js

Both Svelte [Glossary] and Vue.js [Glossary] are known to be more performant than React (Kodaps, 2022)[20], which would allow us to build a faster and more efficient web application. However, as both the team and the employees at OST only have limited knowledge of these two technologies, React was chosen instead.

6.3.4.3. Node.js

Since we chose React as our SPA library, using Node.js [Glossary] would once again give us the advantage of being able to share code between the frontend and backend. However, due to limited experience with this technology as well as concerns about the underlying language, we opted for ASP.NET instead.

[20]Other benchmarks show similar results.

7. Retrospective

7.1. Context

In the final part of our semester thesis, we want to look back on our work and reflect upon our approach and results. To do so, we will first summarize what we have done in this thesis, evaluate the positive and negative aspects of our work and finally outline what future work we believe could be done with regard to our application.

7.2. Review & Summary

Over the last 14 weeks of our semester thesis, we have done the following:

- We defined a user base consisting of 2 user groups.
- We conducted 11 user interviews with different people from this user base.
- We derived 101 tangible user stories from these interviews.
- We conceptualized 5 aspects of an application that addresses these user stories, using existing applications and scientific papers as a foundation for our ideas.
- We prototyped both the import mechanism and the user interface for exercises to ensure their technical feasibility.
- We defined 40 functional requirements for our MVP, which will be developed as part of our Bachelor's thesis.
- We determined that the technologies for the development of this application will be ASP.NET, React, Git, SQLite and Portainer.

The core realization we made in this semester thesis is that an application that focuses exclusively on gamified exercises would not be sufficient to motivate the students at OST to engage more repeatedly and consistently with their programming assignments. This is based on our conclusion that such an application would not address any of the underlying problems both students and lecturers have with the way exercises are conducted at the moment. Instead, we believe that increased engagement with programming assignments can be achieved by developing a software system that streamlines the process of exercise creation, management, execution and evaluation, which is what we have conceptualized as part of this thesis.

7.3. Positives & Negatives

7.3.1. Positives

Looking back at the assignment defined in Section 1.2, we believe that we have surpassed the requirements of this semester thesis. We claim this because we not only completed all required tasks, but also critically assessed the wider context in order to conceptualize an application that adds long-term value to the people at OST.

We further think that the approach we have chosen for this semester thesis has been successful. Through our user interviews, we discovered fundamental problems and concerns both students and lecturers have in regard to exercises, which allowed us to conceptualize our application based on real, qualitative data rather than on assumptions about the needs of our user base. This is visible in our paper by the many references we included to our user interviews and user stories.

On a personal note, we feel that we have addressed an issue that has been a cause for contention for at least our own time at OST. We always considered our exercises to be substandard, and being able to conceptualize an application that has the potential to fundamentally change the way they are conducted is something we consider a personal achievement in both our student and programmer careers.

7.3.2. Negatives

However, we also recognize the flaws in both our thesis and the way we have worked. Firstly, we believe that our research of scientific papers has been lacking the depth required to fundamentally understand the various topics. Due to the limited time frame of this thesis, some papers were only read on a high level basis without going into the details of the subjects discussed. While we believe this to be sufficient for the conducted conceptualization, it carries the risk of us misunderstanding and therefore misapplying certain concepts in our application.

Similarly, we also think that the conducted user research in regard to students (i.e. exercise solvers) is not exhaustive enough to cover all the different subgroups that may exists. For example, no user interviews were conducted with students who do most of their exercises consistently. Such a lack of information may result in important concerns and distinctions being lost during the development of our application. To mitigate this, additional workshops will be conducted with a larger portion of the user base, as mentioned in Section 2.3.

Finally, we also had to omit many ideas and concepts in order to keep this thesis manageable. While some of these ideas can be found in Section 7.4, we would have liked to analyze a few of them in more detail to ensure that we have a good understanding of the larger scope of our application.

7.4. Outlook

To conclude this semester thesis, we want to talk about possible future works that could be done to expand upon the conceptualization conducted in this paper. As this semester thesis will be continued as a Bachelor's thesis, we do not want to list activities that will be performed then (e.g. additional research, implementation, etc.). Rather, we want to focus on various aspects of the application that we brainstormed while working on this thesis but did not have enough time to elaborate on in more detail.

As such, Table 1 contains a list of various ideas and concepts that we consider feasible to be included in separate works. These ideas are rooted in the user stories defined in Section 4 and have in some cases already influenced certain architectural decisions discussed in this thesis.

Detailed Analytics	To help exercise makers make more informed decisions about their own courses, both statistical evaluations (e.g. time spent on exercises) and qualitative evaluations (e.g. causes of failed test cases) could be provided in our application. As we already store all submissions of exercises (see Section 5.3), we could provide existing data if such a feature were to be implemented as part of a different thesis.
AI Integration	To support exercise solvers in their exercises, an AI model trained for explaining certain concepts or detecting certain mistakes could be implemented in our application. This could allow exercise solvers to find solutions to their problems faster while at the same time reducing the workload of exercise makers with regard to answering questions. As of the time of writing, an AI model incorporating these ideas is planned to be developed as part of an upcoming Bachelor's thesis.
More "Block" Formats	We already talked about how the "block" concept discussed in Section 5.2 allows us to model different types of exercises. An interesting semester thesis could thereby be the creation of different kinds of blocks with different features, such as a multiple choice block[21] or similar.
Exam Mode	The ability to switch to a separate "exam mode", in which exercise solvers complete a set of exercises that follow the structure of the final exam, could significantly improve the learning process for exams. This could be achieved by allowing exercise makers to label certain questions as "exam questions" and then defining how their exam is usually structured (e.g. by configuring which topics with which difficulties come in which order). Our application could then generate a practice exam that can be solved in preparation for the real exam.
Repetition Features	Different features for the repetition of exercises may help exercise solvers to learn topics in a more sustainable way. This could take the form of "repetition exercises" that are automatically generated after a few weeks, based on exercises that a exercise solver has previously struggled with. We could also allow exercise solvers to mark difficult exercises themselves in order to find them easier at a later time.

Table 1: Further ideas that could be analyzed in future works

[21]For example: https://github.com/osandadeshan/markdown-quiz-generator

References

I List of Figures

II List of Tables

III Glossary

III.i Applications

Adunis	Adunis is an application that provides an overview of the individual study plan with the chosen modules, important dates and examinations as well as the current and planned performance status for students at OST. (OST - Eastern Switzerland University of Applied Sciences, n.d.)
Codewars	Codewars is an educational community for computer programming. On the platform, software developers train on programming challenges known as kata. (Wikipedia, 2023b)
Duolingo	Duolingo is an educational learning app that offers courses on music, math and over 40 languages. (Wikipedia, 2023c)
GitLab	GitLab is an open-core DevSecOps software package used to develop, secure, and operate software. (GitLab, n.d.)
Moodle	Moodle is a free and open-source learning management system. It is used for blended learning, distance education, flipped classroom and other online learning projects in schools, universities, workplaces and other sectors. (Wikipedia, 2023d)

III.ii Technologies

ASP.NET	ASP.NET is a server-side web-application framework developed by Microsoft designed for web development to produce dynamic web pages. (Wikipedia, 2023e)
Autofac	Autofac is an Inversion of Control container for .NET Core, ASP.NET Core, .NET 4.5.1+, Universal Windows apps, and more. (Autofac, n.d.)
Base UI	Base UI is a library of headless ("unstyled") React UI components and low-level hooks. (MUI, n.d.)
Blazor	Blazor is a free and open-source web framework by Microsoft that enables developers to create web apps using C# and HTML. (Wikipedia, 2023f)
C#	C# is a general-purpose high-level programming language supporting multiple paradigms. C# encompasses static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. (Wikipedia, 2023g)
Docker	Docker is a platform designed to help developers build, share, and run container applications. (Docker, n.d.)

Entity Framework Core	Entity Framework Core is an open-source object–relational mapping (ORM) framework for .NET. (Wikipedia, 2023h)
Git	Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. (Git, n.d.)
Kubernetes	Kubernetes (K8s) is an open-source system for automating the deployment, scaling and management of containerized applications. (Kubernetes, n.d.)
LibGit2Sharp	LibGit2Sharp brings libgit2, a native Git implementation, to the managed world of .NET. (LibGit2Sharp, 2023)
MariaDB/ MySQL	MariaDB is a popular open-source relational databases. It's made by the original developers of MySQL and guaranteed to stay open-source. (MariaDB Foundation, n.d.)
Node.js	Node.js is an open-source, cross-platform JavaScript runtime environment. (OpenJS Foundation, n.d.)
Portainer	Portainer is a container management software to deploy, troubleshoot, and secure applications across cloud, datacenter, and Industrial IoT use cases. (Portainer, n.d.)
PostgreSQL	PostgreSQL is an open-source object-relational database system that has earned a reputation for reliability, feature robustness, and performance. (PostgreSQL Global Development Group, n.d.)
Radix Primitives	Radix Primitives provides unstyled, accessible, open-source React primitives for web apps and design systems. (WorkOS, n.d.)
React	React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on components. (Wikipedia, 2023i)
SQLite	SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. (SQLite Consortium, 2023)
Svelte	Svelte is a free and open-source front-end component framework and language maintained by the Svelte core team members. (Wikipedia, 2023j)
Vue.js	Vue.js is an open-source model–view–viewmodel front end JavaScript library for building user interfaces and single-page applications. (Wikipedia, 2023k)

libgit2	libgit2 is a portable, pure C implementation of the Git core methods provided as a re-entrant linkable library with a solid API. (libgit2, n.d.)
xUnit.net	xUnit.net is a free, open-source, community-focused unit testing tool for the .NET Framework. (.NET Foundation, n.d.)

III.iii Abbreviations

IDE	Integrated Development Environment
MVP	Minimal Viable Product
ORM	Object Relational Mapper
SPA	Single Page Application

Bibliography

.NET Foundation. *About xUnit.net*. Retrieved December 20, 2023, from https://xunit.net/

Autofac. *Autofac*. Retrieved December 20, 2023, from https://autofac.org/

Brügger, A., Bill, R., & Peuker, S. (2023). *User Experience (M_UX)*. https://studien.rj.ost.ch/allModules/41106_M_UX.html

Bundeskanzlei BK. (2023). *Leitfaden zum geschlechtergerechten Formulieren*. https://www.bk.admin.ch/bk/de/home/dokumentation/sprachen/hilfsmittel-textredaktion/leitfaden-zum-geschlechtergerechten-formulieren.html#download_als_pdf__content_bk_de_home_dokumentation_sprachen_hilfsmittel-textredaktion_leitfaden-zum-geschlechtergerechten-formulieren_jcr_content_par_tabs

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-Oriented Software Architecture, a System of Patterns* (9781118725269).

Chou, Y.-k. (2015). *The Octalysis Framework for Gamification & Behavioral Design*. https://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/

Codewars. *Kata Discourse*. Retrieved December 20, 2023, from https://docs.codewars.com/concepts/kata/discourse/

Docker. *Make better, secure software from the start*. Retrieved December 20, 2023, from https://www.docker.com/

Duolingo. *What is a streak?*. Retrieved November 18, 2023, from https://support.duolingo.com/hc/en-us/articles/204980880-What-is-a-streak-

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software* (0–201–63361–2).

Git. *Git*. Retrieved December 20, 2023, from https://git-scm.com/

GitLab. *Software. Faster*. Retrieved December 20, 2023, from https://about.gitlab.com/

International Organization for Standardization. (2019). *Human-centred design for interactive systems (ISO Standard No. 9241-210:2019)*. https://www.iso.org/standard/77520.html

Ken Schwaber, & Jeff Sutherland. (2020). *The Scrum Guide*. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

Kent Beck, Mike Beedle, Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, & Dave Thomas. (2001). *Principles behind the Agile Manifesto*. https://agilemanifesto.org/principles.html

Kodaps. (2022, May 23). *Angular vs Vue.js vs React vs Svelte: The Statistics*. https://www.kodaps.dev/en/blog/angular-vs-vue-js-vs-react-vs-svelte-in-2022-by-the-numbers

Kubernetes. *Production-Grade Container Orchestration*. Retrieved December 20, 2023, from https://kubernetes.io/

libgit2. *libgit2*. Retrieved December 20, 2023, from https://libgit2.org/

LibGit2Sharp. (2023, August 4). *LibGit2Sharp*. https://github.com/libgit2/libgit2sharp

MariaDB Foundation. *MariaDB Server: The open source relational database*. Retrieved December 20, 2023, from https://mariadb.org/

MUI. *Base UI - Overview*. Retrieved December 20, 2023, from https://mui.com/base-ui/getting-started/

OpenJS Foundation. *Download Node.js*. Retrieved December 20, 2023, from https://nodejs.org/en

OST - Eastern Switzerland University of Applied Sciences. *Adunis*. Retrieved December 20, 2023, from https://www.ost.ch/de/die-ost/campus/campus-rapperswil-jona/infrastruktur-und-ict/adunis

Oxford University Press. (2023). *gamification*. https://doi.org/10.1093/OED/7320229446

Portainer. *Accelerate container adoption*. Retrieved December 20, 2023, from https://www.portainer.io/

PostgreSQL Global Development Group. *PostgreSQL: The World's Most Advanced Open Source Relational Database*. Retrieved December 20, 2023, from https://www.postgresql.org/

SQLite Consortium. (2023, December 5). *What Is SQLite?*. https://www.sqlite.org/index.html

Universität Zürich. *Lernen mit Lernpfaden*. Retrieved November 22, 2023, from https://teachingtools.uzh.ch/de/tools/lernen-mit-lernpfaden

Verhein-Jarren, A., & Murbach, R. (2021). *Rhetorische Kommunikation für IngenieurInnen (M_RheKI)*. https://studien.rj.ost.ch/allModules/31909_M_RheKI.html

Wikipedia. (2023a). *Technology tree*. https://en.wikipedia.org/wiki/Technology_tree

Wikipedia. (2023k, October 5). *Vue.js*. https://en.wikipedia.org/wiki/Vue.js

Wikipedia. (2023h, November 15). *Entity Framework*. https://en.wikipedia.org/wiki/Entity_Framework

Wikipedia. (2023e, November 19). *ASP.NET*. https://en.wikipedia.org/wiki/ASP.NET

Wikipedia. (2023f, November 19). *Blazor*. https://en.wikipedia.org/wiki/Blazor

Wikipedia. (2023b, November 20). *Codewars*. https://en.wikipedia.org/wiki/Codewars

Wikipedia. (2023g, December 15). *C Sharp (programming language)*. https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

Wikipedia. (2023i, December 18). *React (software)*. https://en.wikipedia.org/wiki/React_(software)

Wikipedia. (2023c, December 19). *Duolingo*. https://en.wikipedia.org/wiki/Duolingo

Wikipedia. (2023d, December 19). *Moodle*. https://en.wikipedia.org/wiki/Moodle

Wikipedia. (2023j, December 19). *Svelte*. https://en.wikipedia.org/wiki/Svelte

WorkOS. *Core building blocks for your design system*. Retrieved December 20, 2023, from https://www.radix-ui.com/primitives

Appendix

I Original Assignment

Entwicklung einer gamifizierten Anwendung für die Programmierausbildung

Beteiligte Personen

Diese Arbeit wird verfasst von
Mathias Fischler; mathias.fischler@ost.ch
Lukas Messmer; lukas.messmer@ost.ch

Betreuer dieser Arbeit
Prof. Dr.-Ing. Frieder Loch; frieder.loch@ost.ch

Problembeschreibung

Gamification beschreibt den Einsatz von Spielelementen in einem spielfremden Umfeld. So können bestimmte Anwendungen motivierender gestaltet werden. In diesem Projekt soll der Einsatz von Gamification für die Programmierausbildung erprobt werden. So sollen Studierende motiviert werden, sich wiederholt und nachhaltig mit Programmieraufgaben auseinandersetzen.

Formulierung eines konkreten Auftrags

Die Arbeit soll die folgenden Aspekte adressieren. Nach Absprache kann selbstverständlich im Verlauf der Arbeit vom beschriebenen Auftrag abgewichen werden.

Analyse der bestehenden Anwendung. Auf Basis einer bestehenden Anwendungen sollen Anforderungen und Bedarfe der zukünftigen Nutzerinnen und Nutzer analysiert werden.
Recherche und Abgrenzung. Es werden bestehende Anwendungen mit einer vergleichbaren Funktionalität betrachtet, um diese vom Ziel der Arbeit abzugrenzen. Relevante Quellen aus der wissenschaftlichen Literatur sind zu analysieren und bei der Lösungsfindung einzubeziehen.
Technologieauswahl. Es werden Anforderungen formuliert, auf deren Basis eine nachvollziehbare Technologieauswahl durchgeführt wird. Grundsätzlich sollen freie und quelloffene Technologien bevorzugt werden.
Mensch-zentrierte Analyse. Bei der Konzeption von allen User Interface-Komponenten wird ein mensch-zentrierter Ansatz verfolgt. Dies beinhaltet, zum Beispiel, die Entwicklung und Evaluation von Konzepten mit tatsächlichen Usern.
Prototypische Implementierung und Evaluation. Der gewählte Ansatz wird prototypisch implementiert. Die Softwarearchitektur und das Ergebnis werden ausführlich dokumentiert. Es soll eine Implementierung mit potentiellen Usern in einer realistischen Umgebung getestet werden.
Kritische Reflektion. Das gewählte Vorgehen wird kritisch reflektiert. Es werden konkrete Verbesserungsvorschläge am Vorgehen und am Ergebnis diskutiert.

Darüber hinaus ist eine geeignete Projektmanagementmethode auszuwählen und zu beschreiben. Die Beschreibung von Arbeitspaketen und einer angemessenen Anzahl von Meilensteinen ist obligatorisch. Die Planung ist zu dokumentieren und regelmässig im Projekt fortzuschreiben. Die Arbeitszeiten werden auf der Ebene der Arbeitspakete erfasst.

Umfang und Form der erwarteten Resultate

Die Ergebnisse der Arbeit (Quellcode der Software inkl. Dokumentation, sowie der schriftliche Projektbericht) werden den Projektbeteiligten zur weiteren Nutzung zur Verfügung gestellt.

Bei der Erstellung des Berichts werden die üblichen Qualitätsstandards für die Erstellung wissenschaftlicher Arbeiten angewendet. So müssen z.B. alle Quellen (z.B. für Definitionen) konkret und spezifisch nachgewiesen werden. Dies gilt auch für alle Abbildungen, sofern sie nicht selbst erstellt wurden. Quellennachweise erfolgen im Literaturverzeichnis im APA-Stil.

Bei der Erstellung des Berichts sollen die Empfehlungen des Bundes zur geschlechtergerechten Sprache einbezogen.

Anfangs- und Abgabetermin

Start der Bearbeitung: **Montag, 18. September 2023**
Abgabe: **Freitag, 22. Dezember 2023 (17:00 Uhr)**

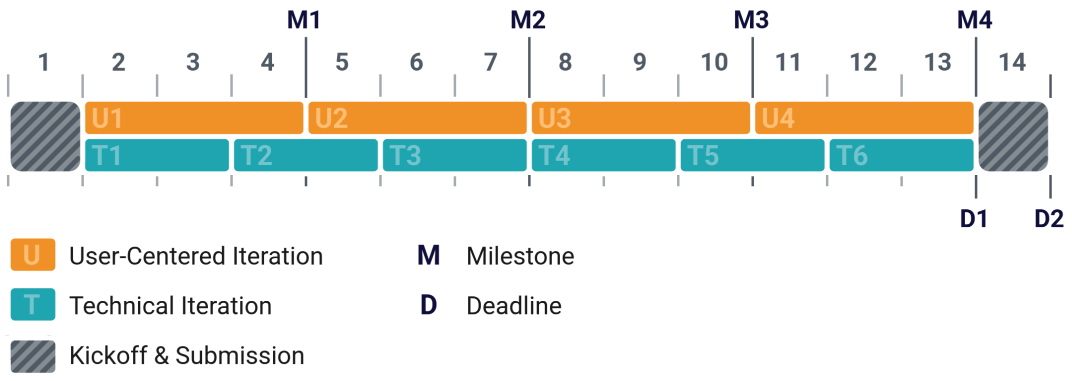
Zulässige Hilfsmittel und weitere Betreuung

Alle verwendeten Hilfsmittel werden in der Arbeit aufgeführt. Die Betreuung erfolgt durch die genannte Betreuungsperson. Es werden wöchentliche Beratungstermine vereinbart und von den Studierenden protokolliert.

II Project Plan

As the complete scope of the project was initially unknown, we used an incremental and iterative approach. Some purely technical work could be done starting day one, which made a decoupled/parallel project plan and structure reasonable.

As seen in the figure below, the project was sliced into four blocks of three weeks each for the user-centered iterations, and into six blocks of two weeks for the technical iterations. The thesis was flanked by the *Kickoff* (week 1) and the *Submission* (week 14). During those weeks, we were focused on planning and consolidating the project, respectively.



Overview of the iterations and milestones within the semester thesis

At the end of each user-centered iteration, a milestone was located. Each milestone had measurable requirements, which were defined before the start of the preceding iteration. These requirements were documented, discussed with the project advisor and verified for completeness at the end of the iteration.

II.i Process

As our team consisted only of two members, processes were kept simple to reduce overhead. The process was broadly inspired by iterative and agile processes/process frameworks, like *Scrum* (Ken Schwaber & Jeff Sutherland, 2020, The Scrum Guide), and the *Agile Manifesto* (Kent Beck et al., 2001, Twelve Principles of Agile Software). Additionally, we followed the principles of the *User-Centered Design Process* (International Organization for Standardization, 2019).

To ensure we were on track for our upcoming milestones, team internal meetings were held at least once a week (*weekly*[22]). For each milestone a review and retrospective was held. Additionally, weekly meetings with the project advisor were held to ensure the project is on track. To avoid formalities, no meeting guidelines were defined for both form and time.

[22]Called *weekly*, paying homage to Scrum's *daily*

III Questionnaire

III.i Example Questions

#	Questions
T1-Q1	*Was für Plattformen und Tools verwendest du für Übungen?*
T1-Q2	*Was für Plattformen und Tools verwendest du zum Lösen von Übungen?*
T1-Q3	*Welche Plattformen verwendest du am liebsten, und wieso?*
T1-Q4	*Welche Plattformen werden für das Einrechen und Feedback von Testaten verwendet?*
T1-Q5	*Welche bevorzugst du, und wieso?*
T1-Q6	*Gehst du in Übungen? Wie oft/welche Fächer/wieso?*
T1-Q7	*Wie regelmässig löst du Übungsaufgaben?*
T1-Q8	*In welcher Form kommen Übungen daher? Beispiele: Code, Code-Projekt, Fragen, Multiple-Choice, Fliesstext*
T1-Q9	*Verwendest du die Übungen in der Lernphase?*
T1-Q10	*Mit deinen Lösungen? Mit Musterlösungen?*
T1-Q11	*Bekommt man Zugriff auf Musterlösungen? Bzw. wann?*
T1-Q12	*Bekommst/Gibst du Feedback auf Übungen und Testate?*
T1-Q13	*Hättest du gerne mehr Feedback? Bzw. in welcher Form (Fliesstext, Gut/Schlecht, etc)?*
T1-Q14	*Gibt es etwas, was du gerne im Bezug auf Übungen, Tesate und Feedback hättest? (Plattform/Features oder Inhaltlich/Betreuung)*
T1-Q15	*Was stört dich sonst noch? Hast du noch andere Bemerkungen?*

IV User Interviews

IV.i Exercise Makers

IV.i.i Interview of the 21.09.2023

#	Questions
M1-Q1	*Welche Tools verwendet ihr für die Vorlesung?* Wir verwenden GitLab-Pages, welche automatisch über unser Versionierungssystem (GitLab) bei einer Änderung veröffentlicht werden.
M1-Q2	*Welchen Inhalt haben diese GitLab-Pages?* Die GitLab-Pages beinhalten Theorie, Praxisaufgaben und Vorlesungsvideos.
M1-Q3	*Wie erstellt ihr Inhalte für diese Pages?* Die Inhalte werden über ein Markdown-Format via Jekyll erstellt. Ich persönlich schreibe die Seiten aber direkt mit HTML, da ich so schneller bin. Jekyll scheint mir aber noch einige praktische Funktionen zu haben, wie z.B. die Erstellung von kleinen Repetitionsfragen.
M1-Q4	*Wieso verwendet ihr nicht andere Tools?* Im Gegensatz zu Moodle erlauben uns diese Pages die Inhalte besser zu gliedern und in Form eines strukturierten Ablaufs (ähnlich wie bei vielen anderen Online-Tutorials) zu Verfügung zu stellen. Andere Tools wie Jupiter-Notebooks wurden auch einmal in Betracht gezogen, sind aber in der Handhabung zu mühsam.
M1-Q5	*Was fehlt dir aktuell noch bei eurer Lösung?* Ich würde enorm gerne Codebeispiele direkt aus Quellen wie angular.io und ähnlichem in meine Pages einbinden. Man könnte das mit Iframes lösen, aber diese müsste man speziell programmieren, damit nur die korrekten Inhalte angezeigt werden. Ausserdem sind Iframes allgemein etwas limitiert. Die Seite angular.io hat so etwas ähnliches. Sie kennzeichnen solche Codeteile dann mit “excerpt”. Ausserdem würde ich mir gerne mehr Inhaltsbeiträge von den Studenten wünschen. In einem anderen Fach müssen die Studenten Zusammenfassungen und Prüfungsaufgaben für den Unterricht hochladen. Aktuell löse ich das über GitLab-Wikis, aber es wäre toll, wenn die Studenten dafür nicht in ein separates Tool wechseln müssten. Das bricht rein optisch schon den Zusammenhang zwischen Theorie und Praxis, welchen ich den Studenten gerne vermitteln würde. Ausserdem wäre es toll, wenn andere Studenten die Prüfungsaufgaben bewerten könnten, damit ich besser einschätzen kann, ob ich eine Aufgabe an der Endprüfung bringen soll oder nicht.

M1-Q6	*Gibt es noch weitere Tools, die du verwendest?* Für die Berufsschule verwende ich noch die Seite learningapps.org. Diese Seite bietet eine Vielzahl von unterschiedlichen Übungsaufgaben zum online lösen, ist aber leider schrecklich veraltet. Insbesondere auf den mobilen Geräten lässt sich die Seite nicht korrekt bedienen.

IV.i.ii Interview of the 28.09.2023

#	Questions
M2-Q1	*Welche Vorlesungen / Module betreust du aktuell?* Ich betreue aktuell die Module Objektorientierte Programmierung 1 (OOP1), Cloud Solutions (CldSol) und Web Engineering 3 (WE3).
M2-Q2	*Welche Tools verwendest du für OOP1?* In OOP1 arbeiten wir hauptsächlich über Microsoft Teams. Die Übungsaufgaben werden dabei mit GitLab verwaltet. Für die Aufgabenstellungen verwenden wir Markdown-Dokumente, die wir in ein PDF konvertieren und dann manuell auf Microsoft Teams hochladen. Die Codeteile der Übungen sind für die Studenten auf GitLab verfügbar, werden aber – da im ersten Semester noch nicht alle Studenten mit Git umgehen können – ebenfalls manuell als ZIP-Datei auf Microsoft Teams hochgeladen. In den Vorlesungen selbst verwende ich auch gerne Gitpod, um den Studenten irgendwelche Codebeispiele zu zeigen. Das ist besonders hilfreich, da es in St. Gallen einfacher ist, die eingebauten Dozierendenpulte anstelle meines Laptops zu verwenden und ich so nur PowerPoint und einen Browser installiert haben muss. Als Alternative zu Gitpod gibt es meines Wissens auch noch das Tool Coder.
M2-Q3	*Gibt es in OOP1 obligatorische Abgaben?* Nein, nicht mehr. Die Studenten können aber weiterhin gewisse Aufgaben abgeben.
M2-Q4	*Wie macht ihr diese Abgaben?* Die Abgaben laufen über die sogenannten Assignments in Microsoft Teams. Das gute daran ist, dass ich die abgegebenen Codeteile dann selber bearbeiten kann. So kann ich den Studenten direkt an der betroffenen Stelle im Code ein Feedback geben. Da sich das Ganze auch mit OneDrive synchronisieren lässt, muss ich nicht einmal mehr Microsoft Teams aufmachen, sondern kann direkt den Ordner mit allen Abgaben in meinen Editor öffnen und bearbeiten.
M2-Q5	*Gibt es aktuell noch andere Tools, die du gerne verwenden würdest?*

	Aktuell evaluieren wir in OOP1 das Visual Studio Code Plugin “CodeTour”. Mit diesem Plugin kann man Touren durch seinen Applikationscode definieren, welche andere Benutzer dann abspielen können. Das ist besonders für Onboarding von neuen Mitarbeitern interessant, könnte aber auch in den Vorlesungen verwendet werden, um die Studenten in komplexere Codebeispiele einzuführen. Das praktische an diesem Tool ist, dass die ganze Tour in einer JSON-Datei gespeichert wird. Dadurch werden die Touren signifikant wartbarer als aufgenommene Videos, welche man schon bei kleinen Änderungen neu aufnehmen müsste. Das CodeTour-Plugin hat jedoch noch einige Bugs und funktioniert darum noch nicht ausnahmslos.
M2-Q6	*Wie stehst du zu automatisierten Tests?* Automatisierte Tests sind ganz klar ein wichtiger Teil von Software Engineering. Es ist jedoch interessant zu sehen, dass einige Studenten recht kritisch gegenüber automatischen Tests sind. Diese Studenten wollen den Output ihrer Funktion lieber sehen und testen diese oft manuell z.B. über Konsolenausgaben in der Main-Methode. Das gibt ihnen scheinbar mehr Sicherheit als ein grünes Häkchen neben einem Testcase, der ihnen sagt, dass die Methode korrekt funktioniert.
M2-Q7	*Wie sieht es bezüglich den Aufgaben und Tools im Modul CldSol aus?* In CldSol ist das Ganze etwas einfacher. Die Studenten bekommen da eine Aufgabenstellung, welche sie in Gruppen erarbeiten müssen. Das Endresultat geben sie dann am Ende des Semester – im Normalfall per E-Mail – ab. Wenn ich die Arbeit korrigiert habe, sende ich – ebenfalls per E-Mail – mein Feedback zusammen mit einem ausgefüllten Bewertungsraster an die Studenten zurück. Zusätzlich dazu gibt es unter dem Semester noch einige weitere, nicht obligatorische Programmieraufaben.
M2-Q8	*Hast du noch andere Anmerkungen?* Ja. Aktuell verfolge ich natürlich auch die Entwicklungen im Bereich AI mit. Einerseits finde ich es problematisch, dass die Studenten nun viele Übungsaufgaben über AIs lösen können, ohne die Übungen dabei tatsächlich zu verstehen. Gleichzeitig sehe ich auch Potenzial darin, dass künstliche Intelligenzen die Codeabschnitte der Übungen den Studenten in textueller Form beschreiben könnten. Ich habe dazu bereits GitHub Copilot für IntelliJ ausprobiert und muss sagen, dass es in gewissen Sprachen erstaunlich gut funktioniert. In anderen Sprachen sind die Resultate aber eher schlecht.

IV.i.iii Interview of the 02.10.2023 (1/2)

#	Questions
M3-Q1	*Welche Vorlesungen / Module betreust du aktuell?* Ich betreue aktuell die Module Web Engineering 1, 2 und 3 (WE1 - 3) sowie einige Zertifikats- und Weiterbildungskurse (CAS).
M3-Q2	*Welche Tools verwendest du für WE1?* In WE1 verwenden wir nun seit etwa 2 Jahren GitLab-Pages. Die Übungsaufgaben werden also in einem GitLab-Repository verwaltet. Eine GitLab-Pipeline generiert dann aus diesem Repository die benötigten PDF-, ZIP- und HTML-Dokumente, auf welche die Studierenden dann über die GitLab-Page zugreifen können. Die PDF- und HTML-Dokumente werden dabei aus AsciiDoc-Vorlagen erzeugt. Da Moodle jedoch weiterhin als Hauptplattform für den Unterricht dient, verlinken wir die einzelnen GitLab-Pages der Übungen nach wie vor im entsprechenden Moodle-Kurs.
M3-Q3	*Welche Inhalte haben diese Übungen?* Die Übungen bestehen aus Theorie- und Programmieraufgaben.
M3-Q4	*Wie funktioniert diese Lösung für dich?* Die Umstellung auf die GitLab-Pages war zu Beginn etwas aufwändig. Seit der Umstellung funktioniert diese Lösung aber recht gut, insbesondere da der Wartungsaufwand bei Anpassungen dank der automatischen GitLab-Pipeline reduziert wurde. Problematisch ist aber, dass wir noch kein HTTPS für die GitLab-Pages einrichten konnten. Das führt neben den sicherheitstechnischen Problemen auch dazu, dass wir im Moodle nicht direkt auf diese GitLab-Pages verlinken können. Moodle, bzw. der Browser, blockiert Weiterleitungen von HTTPS auf HTTP standardmässig. Das bedeutet, dass die Studenten den Link zu den Übungen immer selbständig in einem neuen Fenster öffnen müssen. Ausserdem bin ich kein grosser Fan von den generierten PDF-Dokumenten, da die Seitenumbrüche in einigen Aufgaben einfach schlecht platziert sind.
M3-Q5	*Werden sonst noch andere Tools verwendet?* Aktuell biete ich den Studenten noch CodeWars für das freiwillige Selbststudium von JavaScript an. CodeWars ist eine Lernplattform für das Lösen von verschiedenen Codeaufgaben und beinhaltet auch Gamification in Form von Scoring-Systemen und Leaderboards. Ausserdem kann man nach dem Abschliessen einer Aufgabe auch die Lösungen von anderen Benutzern ansehen. Es gibt immer jemand, der eine Aufgabe in einer Codezeile lösen konnte.

	Eine Zeit lang gab es auch ein intern entwickeltes Puzzle-Tool für Übungsaufgaben im Webbereich. Besonders nennenswert waren da die CSS-Aufgaben, bei welchen die Lösungen der Studenten über einen Pixelvergleich mit der Musterlösung verglichen wurde. Also wurde nicht der Programmcode, sondern die Darstellung der Webseite überprüft. Die Erfassung von neuen Aufgaben war jedoch viel zu aufwändig, wodurch das Tool nicht sonderlich viel gebraucht wurde. Seit der Migration unserer Server ist es wahrscheinlich auch nicht mehr online. Das GitLab-Repository existiert jedoch noch.
M3-Q6	*Sind die Weiterbildungskurse ähnlich aufgebaut?* Nein, überhaupt nicht. Bei den Weiterbildungskursen wechselt das Thema inklusive Dozenten etwa alle 4 Wochen. Meines Wissens gab es auch schon Fälle, in welchen bei 25 Kurstagen 18 verschiedene Dozenten anwesend waren. Dabei sprechen sich die Dozenten auch nicht wirklich ab und jeder Dozent hat seinen eigenen Vorlesungs- und Arbeitsstil. Es ist jedoch so, dass Moodle mittlerweile beinahe Pflicht ist für die Weiterbildungskurse, obwohl es immer noch einige Dozenten gibt, die stattdessen Microsoft Teams verwenden.
M3-Q7	*Wie sieht es mit den Übungen in diesen Weiterbildungskursen aus?* In den meisten Fällen werden die Übungen in die Kursfolien eingebaut und von den Teilnehmer direkt im Kurs gelöst. Ich persönlich verwende dabei meistens ähnliche Übungen wie aus den “Web Engineering”-Vorlesungen, welche ich manuell nach pflege.
M3-Q8	*Hast du noch weitere Anmerkungen?* Ja. Unabhängig davon, was ihr schlussendlich entwickeln werdet, wäre es wichtig, dass für die Dozenten, Professoren und Übungsbetreuer ein möglichst kleiner Overhead entsteht. Das ist insbesondere wichtig, da die Migration und das Einarbeiten in neue Programme für die Dozenten und Professoren grundsätzlich nicht als Arbeitszeit gilt. Allgemein stelle ich mir auch oft die Frage, wie viele Studenten tatsächlich die Übungen machen. Von hundert angemeldeten Studenten sind am Ende sowieso nur etwa zwanzig physisch in den Übungen. Ausserdem scheinen Erfahrungen aus der App Quest der OST zu zeigen, dass Gamification nur schlecht wirkt, wenn es für die Teilnehmenden nichts zu gewinnen gibt. Weiter muss man natürlich beachten, dass das Ausführen von fremden Code immer ein Sicherheitsrisiko darstellt. Einer unserer Mitarbeiter musste schon einmal recht viel Zeit investieren, damit wir Code von den Studenten sicher in einem Docker-Container ausführen können.

IV.i.iv Interview of the 02.10.2023 (2/2)

#	Questions
M4-Q1	*Welche Vorlesungen / Module betreust du aktuell?* Ich betreue aktuell die Module Datenbanksysteme 1 (Dbs1), Objektorientierte Programmierung 1 und 2 (OOP1 - 2), Projekt- und Qualitätsmanagement (PmQm), Data Engineering (DatEng), sowie einige Arbeiten im SE Project (SEProj). Wie du siehst also eine Menge.
M4-Q2	*Kannst du mir eine grobe Übersicht über die Tools geben, die du in diesen Modulen verwendest?* In DatEng arbeiten wir mit Docker, was hauptsächlich daran liegt, dass so das Starten von den verschiedenen Programmen und Datenbanken für die Studenten einfacher geht. Aus einem ähnlichen Grund verwenden wir in Dbs1 auch Coder, wobei da aber weiterhin z.B. Diagramme mit Papier und Bleistift gezeichnet werden. In OOP1 und 2 verwenden die Studenten dagegen ihre eigene Programmierumgebung. Jedoch brauchen wir in diesen zwei Modulen auch teilweise Gitpod. Dazu ist zu sagen, dass die Übungsaufgaben in praktisch allen Modulen den Studenten als PDF bereitgestellt werden.
M4-Q3	*Werden die Unterlagen zu diesen Modulen auf GitLab verwaltet?* Grundsätzlich schon. Bei Dbs1 ist dies aber nicht der Fall. Dort werden die Unterlagen entweder direkt im Moodle oder auf Microsoft Teams verwaltet.
M4-Q4	*Kannst du mir mehr über Dbs1 und Coder erzählen?* Wir verwenden Coder in Dbs1, damit wir den Studenten einfach eine betriebsfähige Umgebung bereitstellen können. Das bedeutet, dass die Studenten die verschiedenen Abhängigkeiten – wie z.B. Visual Studio Code, PgAdmin, Projektdateien, etc. – nicht selber auf ihren Geräten installieren müssen. Es ist insbesondere für Quereinsteiger sehr frustrierend, wenn sie die gesamte erste Übungslektion nur mit der Installation von irgendwelchen Programmen beschäftigt sind. Ausserdem gibt es Personen, die nicht extra für ein Modul neue Tools installieren wollen. Der Nachteil dabei ist natürlich, dass die Installation inklusive der möglichen Fehlerquellen nicht gelernt wird. Weiter gibt uns Coder die Möglichkeiten, einen eigenen SQL-Parser in die Umgebung vorzuinstallieren. Dieser Parser evaluiert die obligatorischen Abgaben der Studenten auf verschiedene Kriterien, wie z.B. die Anzahl Tabellen, Abfragen und Clean-Code. Zuvor wurde dieses Tool nur intern zur Korrektur der Abgaben verwendet, jedoch gab es aus meiner Sicht keinen Grund, es den Studenten vorzuenthalten. Die

	selbständige Überprüfung ihrer SQL-Abgaben gibt den Studenten mehr Sicherheit und uns weniger Korrigierungsaufwand.
M4-Q5	*Wieso habt ihr euch für Coder entschieden?* Hauptsächlich aus dem Grund, dass Gitpod kein Self-Hosting mehr zulässt.
M4-Q6	*Wie funktioniert Coder für dich?* Wirklich verwenden werden wir Coder in Dbs1 erst in den nächsten Wochen. Aus Erfahrung hat sich aber gezeigt, dass Coder teilweise Probleme bei der parallelen Verwendung hat. Es scheint so, als ob dem Programm einfach zu wenig Leistung auf unseren Servern zugeteilt wird. Solche Probleme fallen leider erst bei der aktiven Verwendung des Tools auf.
M4-Q7	*Wie sieht es mit den Theorieaufgaben in Dbs1 aus?* Die Theorieaufgaben werden wie erwähnt einfach als PDF bereitgestellt. Meines Wissens gibt es auch gute Moodle-Plugins für solche Theorieaufgaben, diese habe ich aber bisher noch nicht ausprobiert. Allgemein bin ich sowieso nicht so ein Fan von reiner Theorie, sondern bin der Meinung, dass man die diese immer mit einem Praxisbeispiel verknüpfen soll.
M4-Q8	*Wie sieht es mit der Korrektur von Aufgaben in Dbs1 aus?* Die Problematik bei Dbs1 ist, dass sich die Aufgaben nicht immer automatisch überprüfen lassen. Im Gegensatz zu objekt-orientieren Sprachen gibt es in SQL nicht wirklich ein allgemeines Testframework für alle Anwendungen. Ausserdem gibt es Aufgaben, wie z.B. das Erstellen von UML-Diagrammen, welche sich einfach nicht allgemeingültig Lösen lassen. Dieses Problem gibt es auch in anderen Fächern wie OOP1/2 und da muss man einfach manuell Feedback geben. Leider ist manuelles Feedback nicht nur sehr Aufwändig, sondern meistens für uns Unterrichtende auch nicht wirklich interessant. Trotzdem muss man fokussiert bleiben und aufpassen, dass man nicht einfach alles durchwinkt. Ich denke, Dbs1 würde da sehr stark von Peer-Review-Aufgaben profitieren, welche es auch in anderen Fächern gibt. Interessanterweise ist es für die Studenten meistens recht einfach, selber eine Lösung zu erstellen, jedoch enorm schwer, die Fehler in den Lösungen der anderen zu finden. Darum bin ich der Meinung, dass die Studenten auch etwas aus diesen Peer-Reviews lernen könnten.
M4-Q9	*Hast du noch weitere Anmerkungen?*

	Ja. Ich denke das Moodle bereits enorm viele praktische Funktionen eingebaut hat, wie z.B. Peer-Reviews, Quizzes, Repetitionsfragen und so weiter. Das Problem ist dabei der Zeitaufwand, der es für die Einarbeitung in diese Funktionen braucht. Jedes neue Tool muss man zuerst kennenlernen und wie ihr sicher bereits von anderen Lehrkräften gehört habt, haben die meisten nicht wirklich die Zeit dazu. Wenn ihr also ein neues Tool entwickelt, solltet ihr darauf achten, dass die Einarbeitungszeit möglichst gering ist.

IV.i.v Interview of the 09.10.2023

#	Questions
M5-Q1	*Welche Module / Vorlesungen betreust du aktuell?* Ich betreue aktuell die Praktika in Computernetze 1 (CN1).
M5-Q2	*Wie sind die Praktika in CN1 aufgebaut?* Die Praktika bestehen grundsätzlich aus Pre-Studies und Labs. In den Pre-Studies bereiten sich die Studenten selbständig durch das Erarbeiten eines Dokuments auf die Labs vor. In den Labs wird dann der gelernte Stoff mit Praxisaufgaben vertieft.
M5-Q3	*Wie sehen die Pre-Studies aus?* Die Pre-Studies sind in LaTeX geschriebene PDF-Dokumente. Sie beinhalten neben erklärenden Texten auch Theoriefragen, welche die Studenten lösen müssen. Die Dokumente werden dabei auf GitLab verwaltet.
M5-Q4	*Wie funktionieren die Labs?* In den Labs haben die Studierenden die Möglichkeit, das Gelernte in der Praxis anzuwenden. Dazu erhalten sie einen mehrheitlich in sich abgeschlossenen Auftrag, welchen sie im Lab bearbeiten müssen. Dabei wird viel mit Hardware gearbeitet, wie z.B. mit Switches, Netzwerkkabeln und so weiter. Die Bearbeitung des Auftrags erfolgt meistens im Team. Als Ergänzung geben wir den Studierenden auch noch die Möglichkeit, einiges mit dem Cisco Packet Tracer auszuprobieren.
M5-Q5	*Wie werden die Aufträge kontrolliert?* Während den Labs gehen wir auf die Studierenden zu, um zu kontrollieren, ob die Pre-Studies wirklich gelöst wurden. Wir fragen sie normalerweise, ob sie noch offene Fragen zum bearbeiteten Thema haben. Wenn nichts kommt, testen wir ihr Wissen mit einigen Verständnisfragen. So können wir relativ gut herausfinden, ob sich jemand tatsächlich mit dem Pre-Study auseinandergesetzt hat.

	Bei den Aufträgen in den Labs sieht es ähnlich aus. Es gibt für uns nicht wirklich eine Möglichkeit, universell zu überprüfen, ob ein Lab korrekt gelöst wurde oder nicht. Die Zusammenarbeit zwischen Studierenden und Betreuer ist also enorm zentral. Automatische Tests, so wie ihr es aus der Softwareentwicklung kennt, sind bei uns in den Labs nur begrenzt einsetzbar. Man könnte jedoch in gewissen Fällen z.B. bestimmte Code-Snippets mittels Pattern-Matching auf die Korrektheit prüfen.
M5-Q6	*Sieht es im Modul Computernetze 2 (CN2) ähnlich aus?* Nicht ganz. Während wir in CN1 viel mit Hardware arbeiten, verwenden wir in CN2 mehrheitlich virtuelle Labs. Wir haben dazu ein Programm namens Lab Topology Builder (LTB), welches intern am Institut für Netzwerke und Sicherheit (INS) entwickelt wurde. Dieses Programm erlaubt es uns, virtuelle Netzwerke mit Router, Services und so weiter aufzusetzen. Anschliessend kann man das Netzwerk über eine SSH-Verbindung mit der Command Line ansprechen. Das Backend dazu wurde vor kurzem umgeschrieben und es stehen aktuell Ideen im Raum, dass virtuelle Netzwerk irgendwie in einem Web-GUI darzustellen. Das wäre sicher etwas, was sich in eurer Tool integrieren lassen würde.
M5-Q7	*Hast du noch weitere Anmerkungen?* Ja. Was uns enorm helfen würde, wäre ein Programm mit einem eingebauten Hardware-Kiosk. Für unsere Übungen und Labs müssen wir oftmals verschiedene Komponenten wie z.B. virtuelle Maschinen beantragen. Es wäre enorm praktisch, wenn beim Aufsetzen eines Kurses automatisch die benötigte Hard- und Software provisioniert werden würde. Einige Vorstosse in diese Richtung gibt es bereits bei uns am INS, aber ich könnte mir das auch als Teil von eurem Tool vorstellen.

IV.i.vi Interview of the 10.10.2023

#	Questions
M6-Q1	*Welche Module / Vorlesungen betreust du aktuell?* Ich betreue aktuell ausschliesslich Automatisierung mit Python (AutPy), da ich nebenbei noch viel anderes zu tun habe.
M6-Q2	*Wie ist das Modul aufgebaut?* AutPy funktioniert nach dem “Flipped Classroom”-Prinzip. Die Studierenden besuchen alle 2 Wochen ein Jupyter-Lab (Praktikum). In diesem Lab sollen sich die Studierenden selbständig die Theorie zu einem Thema aneignen und anschliessend verschiedene Übungen dazu lösen. Die Theorie stammt dabei mehrheitlich aus einem entsprechenden Fachbuch, welches ich ihnen in zusammengefasster Form zur Ver-

	fügung stelle. Zusätzlich dazu müssen die Studierenden in Zweiergruppen ein Abschlussprojekt erarbeiten.
M6-Q3	*Wie wird das Modul benotet?* Die Modulnote setzt sich zu 1/3 aus den selbständigen Übungen und zu 2/3 aus dem Abschlussprojekt zusammen. Anders gesagt werden vier Blöcke aus den Praktika und das Abschlussprojekt benotet. Eine Prüfung gibt es in diesem Modul nicht, dafür erhalten die Studierenden auch nur 2 ECTS.
M6-Q4	*Wie wird die Bewertung durchgeführt?* Die Praktika werden mittels vollautomatisierten Tests bewertet. Bei den Abschlussprojekten ist dies leider nicht möglich. Dort ist die Bewertung manuell und durch das enorm aufwändig. Bei den etwa hundert eingeschriebenen Studierenden bin ich da schnell mal einige Tage beschäftigt. Aus diesem Grund habe ich auch Zweiergruppen eingeführt, da ich so den Bewertungsaufwand um die Hälfte reduzieren kann. Ein bisschen Automation habe ich aber, da ich ausschliesslich eine Checkliste manuell ausfüllen muss und dann die Bewertung automatisch an die Studierenden per E-Mail versendet wird. Ansonsten habe ich leider nicht das Gefühl, dass sich die Abschlussprojekte sinnvoll automatisieren lassen.
M6-Q5	*Welche Tools verwendest du dazu?* Die Tools habe ich grösstenteils selber entwickelt. Dazu gibt es auch einen Talk von mir auf YouTube. Im Nachhinein habe ich aber festgestellt, dass es bereits einige Lösungen in diesem Bereich gibt. Nbgrader wäre z.B. ein Tool, mit welchem man Aufgaben und Bewertung direkt ins Jupyter Notebook integrieren könnte. Ausserdem denke ich, dass es im Bereich Gamification auch schon einiges gibt.
M6-Q6	*Gibt es aktuell noch Probleme bei der Modulgestaltung?* Ich finde es aktuell noch schwierig, die richtige Balance zwischen «Lernen» und «Eigenleistung» bei den Studierenden zu finden. Ein gutes Beispiel sind da die automatisierten Tests. Bis anhin habe ich die konkreten Testfälle jeweils für die Studierenden versteckt, damit diese nicht damit anfangen, ihre Funktionen spezifisch auf die Testfälle zu optimieren. Jedoch hat dies auch dazu geführt, das Studierende an gewissen Stellen stecken geblieben sind und nicht wussten, wieso ihre Tests fehlschlagen. Besonders frustrierend war das dann, wenn der Fehler eigentlich nur klein war. Aus diesem Grund sind die Testfälle in diesem Semester für die Studierenden sichtbar. Ich bin aktuell noch am experimentieren, wie viele Infos ich den Studierenden tatsächlich geben will. Immerhin soll ich im Modul im-

	mer noch die erbrachte Eigenleistung benoten. Andere Mechanismen, wie z.B. die Limitierung der Anzahl erlaubten Abgaben, sind leider auch problematisch. Es steht aber ganz klar fest, dass die manuelle Kontrolle, ob die Übungen auch tatsächlich selbständig gelöst werden, vom Umfang her einfach unmöglich ist. Automatische Plagiatserkennung wäre da ein interessanter Ansatz, um dieses Problem zu lösen.
M6-Q7	*Hast du noch weitere Anmerkungen?* Ja. Grundsätzlich ist das Engagement von den Studierenden in diesem Modul recht hoch, was wohl daran liegt, dass die Übungen 1/3 der Note ausmachen. Man merkt jedoch, dass sich einige nicht trauen, bei Fragen auf mich zuzukommen. Weiter stellt sich mir auch die Frage, wie sich eurer Tool von den bereits existierenden Lösungen unterscheiden kann.

IV.i.vii Interview of the 13.10.2023

#	Questions
M7-Q1	*Welche Module / Vorlesungen betreust du aktuell?* Ich betreue aktuell AI Foundations (AIFo) und Parallele Programmierung (ParProg).
M7-Q2	*Welche Tools verwendest du für AIFo?* In AIFo arbeiten wir mit Jupyter Notebook. Zu Beginn haben wir diese Notebooks auf JupyterHub und Moodle zu Verfügung gestellt. Mittlerweile laden wir diese aber nur noch auf Moodle hoch. Das Problem mit JupyterHub war, dass wir die Notebooks immer separat zu Moodle hochladen mussten, was zu Duplikaten und allgemeiner Redundanz führte. Ausserdem sind die zu trainierenden AI-Modelle einfach genug, dass die Studierenden diese auch auf ihren eigenen Geräten ausführen können.
M7-Q3	*Welche Tools verwendest du für ParProg?* In ParProg verwenden wir eine breite Palette an Tools und Sprachen. Zu Beginn arbeiten wir mit Java und .NET. Danach müssen die Studierenden mit CUDA und C programmieren. Dazu stellen wir ihnen auch entsprechende Online-Labs bereit, damit auch Studierende ohne Nvidia-Grafikkarten die Aufgaben lösen können. Anschliessen gibt es noch einige Aufgaben, in welchen die Studierenden auf unserem HPC-Cluster (High Performance Computing Cluster) ihren Programmcode ausführen müssen.
M7-Q4	*Werden diese Aufgaben bewertet?*

	In AIFo gibt es zu den Übungen grundsätzlich keine Bewertung. Jedoch müssen die Studierenden ein Mini-Projekt abschliessen. In ParProg müssen die Studierenden mehrere Testate erfüllen. Dazu ist zu sagen, dass das Anschauen und Bewerten der Abgaben enorm viel Zeit in Anspruch nimmt. Das liegt insbesondere daran, dass ich für jede einzelne Abgabe die Dokumente aus Moodle herunterladen, kompilieren, ausführen, etc. muss. Es wäre enorm praktisch, wenn die Studierenden und ich die Abgaben z.B. im selben Interface anschauen könnten.
M7-Q5	*Hast du sonst noch Probleme mit den aktuellen Übungen?* Ja. Viele der Übungen, die ich übernommen habe, sind nicht mehr auf dem aktuellsten Stand. Ausserdem sind einige davon entweder zu einfach oder zu schwer. Das führt dazu, dass die Studierenden die Übungen schlichtweg nicht machen. Es wäre sicher interessant, wenn sich die Schwierigkeit der Aufgaben adaptiv an die Fähigkeiten der Studierenden anpassen würde. Das könnte man sicher mit AI oder graphenbasiert lösen, würde aber deutlich mehr theoretischen Aufwand und eine grössere Menge an Übungen voraussetzen. Mit einer solchen adaptiven Skalierung würden die Aufgaben aber sicher spannender werden, als sie jetzt sind.
M7-Q6	*Hast du noch weitere Anmerkungen?* Ja. Meine Kinder verwenden aktuell ein Tool namens Anton. In diesem Tool können sie verschiedene Aufgaben (z.B. über Mathematik) lösen und erhalten so virtuelle Punkte. Diese virtuellen Punkte können sie dann verwenden, um irgendwelche Videospiele im Tool zu spielen. Es ist erstaunlich, wie gut diese Strategie bei meinen Kindern funktioniert. Vielleicht lassen sich Studierende auch mit solchen Strategien motivieren. Wahrscheinlich müsste man ihnen aber eher Snacks und Kaffee statt Videospiele anbieten.

IV.ii Exercise Solvers

IV.ii.i Interview of the 29.09.2023

#	Questions
S1-Q1	*In welche Übungen gehst du? Wie oft?* Ich gehe nur in Pflichtübungen. Ausser bei einem Dozenten, mit dem Rede ich gerne (aber löse keine Übungsaufgaben).
S1-Q2	*Löst du die Übungsaufgaben dafür alleine?* Nur wenn mir ein Thema nicht klar ist. Zum Beispiel manchmal bei Mathe-Modulen. Beziehungsweise wärend der Lernphase. Oder wenn es Pflicht ist, natürlich.
S1-Q3	*Wie lernst du (mit Übungen) in der Lernphase?* Am liebsten lerne ich mit alten Prüfungen. Es geht mir da aber um das Prüfungsformat. Oft gehe ich auch einfach alle Folien durch. Bei Unklarheiten gehe ich zur dazugehörigen Übung und sehe mir die Musterlösung der Übung an. Ansonsten löse ich Übungen quasi nie. Testate löse ich natürlich um das Modul zu bestehen.
S1-Q4	*Was hält dich davon ab, Übungen zu lösen? Was stört dich?* Oft lerne ich nichts davon (TZ-Student). Die Hürden sind aber auch mühsam. Oft muss ich jede Woche ein Zip herunterladen. Oder ein neues Git-Repository. Sofern alle Übungen in einem Repo sind (z.B. wie bei C++) ist alles gut. Dann muss ich auch das Setup lokal nur einmal machen. Die Hürden für die Übungsaufgaben sollten möglichst klein sein. Einfaches Setup ist wichtig. Ich brauche zum Lösen gerne meine eigenen Tools, browse aber gerne in Online-Editoren durch Lösungen.
S1-Q5	*Wie gehst du mit der Lernphase um? Was gefällt dir, was verwendest du nicht gerne?* Ich lerne nicht gerne mit Anki oder Quizlet. Kahoot gefällt mir. Ich störe mich, dass viele Infos versträut sind (Moodle, Gitlab, Teams, etc.).
S1-Q6	*Hast du noch etwas zum Thema Testate?* Ich störe mich, wenn die Arbeitslast zu gross ist für einen kleinen Zeitrahmen. C++ war gut - zwei-drei Wochen Zeit, aber Arbeitslast etwa eine Übungswoche. Feedback in den Dateien im Zip zu bekommen war ok, aber nicht optimal. Reviews auf Gitlab wären besser, haben aber das Problem dass nicht alle Git gut beherrschen am Anfang des Studiums. Rein schriftliches Feedback ist aber ok.

IV.ii.ii Interview of the 05.10.2023

#	Questions
S2-Q1	*Welche Programmier-Module hast du im letzten Semester besucht?* Das war Parallele Programmierung (ParProg) und Functional Programming (FP).
S2-Q2	*Hast du die Übungen für diese Module besucht?* Für Parallele Programmierung war ich aufgrund eines Zeitkonflikts nicht vor Ort. Bei FP war ich fast immer in den Übungen und bin auch meistens die gesamte Lektion anwesend geblieben.
S2-Q3	*Hast du bei ParProg die Übungen trotzdem gelöst?* Ausser den Testaten habe ich jeweils am Ende der Themenblöcke (Jeweils 3-5 Wochen) die Übungen des Blockes überflogen, bzw. deren Lösungen. Die Testate waren im Umfang etwa 1.5-Mal so gross wie die Übungsaufgaben.
S2-Q4	*Wie bist du beim Lernen vorgegangen?* Bei ParProg habe ich einfach die Probeprüfungen durchgeschaut und die Übungen die mit Prüfungsaufgaben thematisch übereinstimmten nochmals überflogen.
S2-Q5	*Und bei FP?* Dort habe ich bei allen Aufgaben, bei denen mir die Lösung beim Überfliegen noch nicht 100% zu offensichtlich waren, nochmals vollständig gelöst. Das hat gut funktioniert!
S2-Q6	*Mit was für Tools hast du jeweils gearbeitet?* Bei ParProg habe ich lokal gecoded und dann den Code auf den Cluster kopiert. Bei FP habe ich jeweils lokal mit VS Code gearbeitet. Dabei habe ich, je nach dem, die Tests selber geschrieben oder die zur Verfügung gestellten verwendet um die Resultate zu testen. Zudem mussten wir in den Übungen unsere Abgaben der Übungsleiterin erklären.
S2-Q7	*Sonst noch etwas? Zum Beispiel eine Beschwerde oder eine Idee?* Nein, da fällt mir gerade nichts ein.

IV.ii.iii Interview of the 05.10.2023

#	Questions
S3-Q1	*Welche Informatik-Module hast du im Frühjahrssemester 2023 besucht?* Software Engineering Practices 2 (SEP2), Distributed Systems (DSy), Web Engineering 2 (WE2) und Cloud Operations (CldOp)

S3-Q2	*Hast du alle Übungen besucht?* Nein, nur SEP2 und CldOp. Bei DSy musste ich nur eine Arbeit schreiben und bei WE2 habe ich nur die Testate gelöst. Die waren aber auch aufwendig und sehr lehrreich! Bei SEP2 habe ich aber auch nur die Pflicht-Übungen gelöst. Dort haben wir aber teilweise Pair-Programming und andere Katas gelöst. Leider war das Feedback nur Pass/No Pass.
S3-Q3	*Wie waren die Übungen bei CldOp?* Wir mussten vor allem auf dem Cluster manuell die Übungen lösen. Die Kontrolle war eine Musterlösung mit Beispiel-Output oder Verhalten. Die haben wir jeweils auf Teams gefunden.
S3-Q4	*Wie hast du bei WE2 gelernt?* War mir nicht wichtig genug. Ich habe einfach eine Zusammenfassung und Folien durchgeschaut. Hat gereicht.
S3-Q5	*Ok, gehen wir zurück in frühere Semester, hattest du sonst noch ein Modul mit Übungen bei denen du teilgenommen hast?* Ja, z.B. AI Foundations. Dort haben wir online in Jupyter Notebooks gearbeitet. Das war gut, da das Modul nicht um Python ging. So mussten wir nicht alles lokal Aufsetzen.
S3-Q6	*Wie bist du mit dem Tooling zufrieden? Hast du dazu noch Bemerkungen?* Bei WE2 wäre es gut, hätte ich immer git verwendet. Vielleicht wäre dort ein Zwang positiv. Sofern es Fachfremd ist, wäre ein Online-Environment für das Tooling eine wilkommene Entlastung. Je nach Modul.

IV.ii.iv Interview of the 06.10.2023

#	Questions
S4-Q1	*Welche Informatik-Module hast du im Frühjahrssemester 2023 besucht?* Distributed Systems (DSy) und Data Analytics.
S4-Q2	*Hast du dort die Übungen besucht?* Bei DSy gab es nur eine Arbeit, keine Übungen. Bei Data Analytics habe ich die Übungen nicht besucht, aber wöchtenlich angeschaut.
S4-Q3	*Was meinst du mit angeschaut?* Ich habe die Lösungen überflogen, da ich das meiste bereits verstanden hatte. Mitterlweile weiss ich, was es in solchen Modulen braucht, und lerne effizient auf die Prüfungen.
S4-Q4	*Und in der Lernphase?*

	Je nach Thema/Kapitel habe ich die Übungen doch noch gelöst, sofern sie Aufgaben in den Musterprüfungen entsprachen. Wenn es einfach war habe ich es aber einfach weggelassen.
S4-Q5	*Hast du viel gecoded?* Nein, da war kaum coding notwendig.
S4-Q6	*Ok, gehen wir in ein anderes Semester, hattest du andere Module, in denen du Übungen gelöst hast?* Ja, z.B. das C++ Modul (CPP). Dort war ich auch wöchentlich anwesend.
S4-Q7	*Wie und welche Übungen hast du gelöst? Erzähl ein wenig davon.* Die Übungen musste man grösstenteils eh machen, da sie für Testate relevant waren. Zudem wusste ich, dass ich sie eh machen musste, um die Prüfung zu bestehen, da ich noch nie mit CPP zu tun hatte. Gearbeitet habe ich lokal, wir hatten da ja ein git-Repository mit allen Übungen gruppiert nach Woche. Da wir Tests hatten, habe ich quasi Test-Driven gearbeitet. Wenn ich nicht wusste wie weiter, habe ich mal geschaut, welche Tests noch failen und habe dann dort weitergemacht. Das fand ich noch gut.
S4-Q8	*Wie sieht es mit Testaten aus? Erzähl mir da ein bisschen.* Ich finde sie eher Mühsam, unabhängig vom Modul, da sie einfach erzwungener Aufwand sind. Das Feedback ist dann aber häufig doch nur Ok/Nicht Ok, was ein wenig schade für den Aufwand ist. Ein textuelles Feedback wäre mir schon lieber. Bei Peer-Reviews sieht das anderst aus, dafür ist das Feedback extrem vom Mit-Studierenden abhängig. Das hat die Feedbacks oft nicht so wertvoll gemacht. Offenere Testate wären cool, ohne klar messbare Aufgabenstellung, und textuelles Feedback von Dozierenden.
S4-Q9	*Stört dich sonst noch etwas am Lernprozess?* Ja, viele Module haben ihre Informationen und Unterlagen an mehreren Orten verteilt. Besonders Teams ist eine Katastrophe, besonders wenns noch verschiedene Channels für Übungen und Vorlesungen gibt. Ein brauchbares Tool mit einem zentralen Ort für alles wäre schön.
S4-Q10	*Und Tooling? Wie stehst du zu Online-Editoren?* Zu Online-Editoren stehe ich skeptisch, manchmal sind sie einfach nicht online. Probleme hatte ich auch mit dem Cluster, der ist nur vor Ort verfügbar (oder via VPN, aber da hatte ich Probleme) und manchmal war er auch einfach down. Lokal arbeiten zu können ist mir wichtig.

V Application Analysis

V.i Remark

During our semester thesis, we realised that a formal evaluation of the tools used at OST would be too time-consuming compared to the benefits we expected. As such, only one formal evaluation was conducted, as presented below.

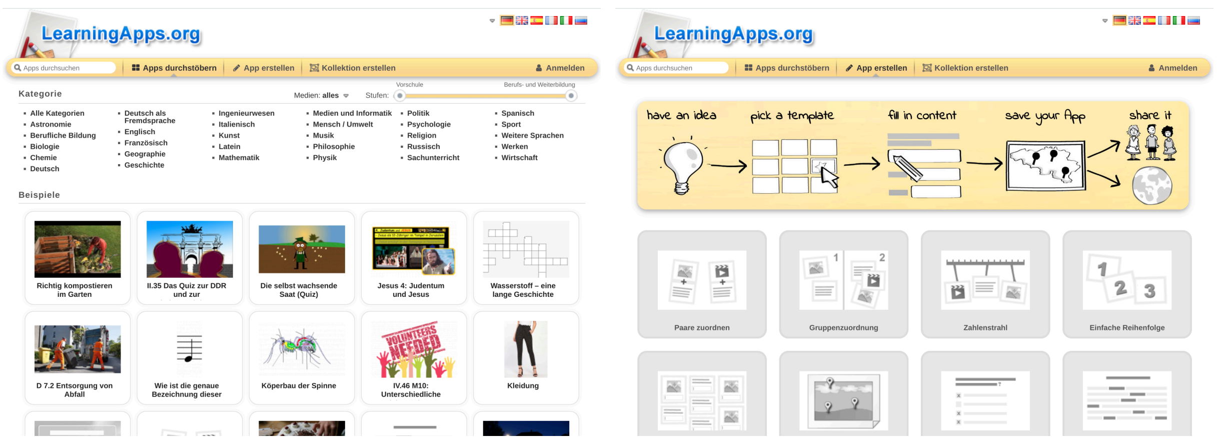
V.ii Analysis of LearningApps.org

V.ii.i Description

LearningApps.org is a free website that allows creating, solving and sharing online exercises. The exercises come in a total of 21 different formats, ranging from multiple-choice quizzes and pair matching to crossword puzzles and horse races. The website also includes a section containing exercises created by other users, and allows both rating and sharing them via link or iframe.

The application was used for the analysis from 29.09.2023 to 01.10.2023. It was available during the analysis at https://learningapps.org/.

V.ii.ii Images



Creating (*right*) and sharing (*left*) online exercises using LearningApps.org

V.ii.iii Positive Impressions

We were positively surprised by the way LearningApps.org handles the creation of exercises. The amount of pre-existing exercise formats is quite useful for finding inspiration on how a particular exercise could look like. Creating a new exercise is also pretty intuitive, as a user is only required to enter the most basic information for an exercise to work. For example, to create a new crossword puzzle, a user only has to enter the individual questions/words of the puzzle and the result word. The layout of the puzzle is then automatically generated by the tool.

In addition, being able to share created exercises with other users is also useful. LearningApps.org already hosts a large number of exercises, presumably created by various teachers from all over the world. The website also allows filtering exercises by categories. For example, if a user wants to create an exercise about ancient Egypt, they can first look in the History category and the Egypt subcategory to see if there already exists an exercise that fits their needs. The search for existing exercises is further aided by the fact that each exercise displays how many times it has been viewed and how other users have rated it using a 0 to 5 star rating.

V.ii.iv Negative Impressions

The first thing one notices when opening LearningApps.org is the outdated design of the website. The presentation simply no longer meets the standards expected from modern websites. This concerns not only the visual appearance of the page, but also its behavior on other devices, such as smartphones or tablets. The website does not have a layout optimized for mobile devices, which means that solving exercises on a smartphone is simply not possible, as the size of all UI elements are designed for keyboard and mouse.

Furthermore, while we like the idea of being able to share exercises with other users, the search process simply lacks most of the features necessary to make it useful. While one can filter by category, there is no way to sort the exercises in any meaningful way (e.g., by rating, number of views, etc.). It is also unclear what the default sorting of the website is, as it is not by alphabetical name, views, rating, or creation date (or at least what we think is the creation date, since it is not properly labeled). Also, there seems to be no way to limit the exercises to a specific language, which leads to the website being overloaded with exercises in foreign languages.

There are also some technical issues that we have noticed while using the website. Exercises cannot be opened in a new tab, which means there is no way to try out an exercise while keeping the current exercise overview open. Questions, for example in a crossword puzzle, can only be reordered using up/down buttons. There is no way for the user to drag and drop a question or to enter a position for a question to be moved to, which makes rearranging questions slower than simply copying and pasting the question text. Although these problems are comparatively minor, we still wanted to include them in this analysis.

VI Requirements

VI.i Exercise Makers

IID	Requirement	I.	C.
5	I want to reduce the overhead for creating and managing exercises.	5	5
↳ 16	I want to easily migrate to a new solution.	5	5
↳ 3	I want to use GitLab (VCS) to manage my exercises.	5	4
↳ 14	I want to automatically synchronize access rights to my exercises (e.g. Moodle, Adunis)	5	4
↳ 7	I want to write exercises in markup languages that I am comfortable with.	4	3
↳ 8	I want to write exercises in Markdown.	4	2
↳ 10	I want to write exercises in HTML.	1	3
↳ 9	I want to write exercises in Jekyll.	0	0
↳ 11	I want to write exercises in AsciiDoc.	0	0
↳ 12	I want to write exercises in LaTeX.	0	0
↳ 72	I want to manage my exercises without using GitLab.	2	4
↳ 73	I want to manage my exercises directly inside the application.	2	2
↳ 6	I want to publish my exercises when I change them in GitLab.	2	3
↳ 15	I want to reuse exercises that have been created before (e.g. for a different course).	2	2
↳ 17	I want to automatically provision the components needed for my exercises (e.g. VMs, Hardware, etc.)	1	4
↳ 13	I want to link my exercises directly in the corresponding Moodle course.	2	1
↳ 64	I want to write exercises in WYSIWYGs (e.g. Microsoft Word).	0	0
27	I want to provide a wide variety of different exercises.	5	5
↳ 29	I want to provide practical exercises (e.g. programming tasks)	5	5

↳		**28**	I want to provide theory exercises (e.g. text, images)	5	3
↳		**33**	I want to provide exercises that integrate with other tools (e.g. virtual labs).	2	5
↳		**34**	I want to provide exercises that span over a longer period of time (e.g. mini-projects)	1	3
↳		**76**	I want to combine theoretical exercises with practical exercises.	1	1
↳		**30**	I want to provide video exercises.	0	0
↳		**31**	I want to provide exercises from external sources (e.g. Codewars)	0	0
↳		**32**	I want to provide exercises that must be solved non-digitally (e.g. using hardware).	0	0
42			I want exercise solvers to submit their solutions for an exercise.	5	5
↳		**70**	I want to evaluate the submissions of exercise solvers using custom scripts (e.g. SQL Parser).	4	5
	↳	**71**	I want to allow exercise solvers to run the evaluation scripts before submission.	3	1
↳		**65**	I want submissions to be done in different formats (e.g. file upload, plain text, etc.)	4	4
↳		**46**	I want to provide feedback directly on the submitted solution.	4	3
	↳	**48**	I want to provide feedback in form of text.	3	1
	↳	**49**	I want to provide feedback in form of an in-person discussion.	2	1
	↳	**47**	I want to provide feedback in form of an evaluation matrix.	0	0
	↳	**50**	I want to send my feedback to the exercise solvers using e-mail.	0	0
↳		**44**	I want to allow exercise solvers to submit solutions as a team.	3	3
	↳	**45**	I want to allow exercise solvers to form teams by themselves.	3	2
↳		**43**	I want to make submissions mandatory.	3	1
↳		**51**	I want to view all submitted solutions of an exercise in one central location.	3	1

↳	53	I want to allow exercise solvers to view the submissions of other exercise solvers.	1	2
↳	52	I want to edit all submitted solutions locally in the editor I am comfortable with.	0	0
↳	54	I want to calculate a grade based on the submitted solutions of an exercise solver.	0	0
↳	55	I want to automatically detect plagiarism in the submitted solutions.	0	0
↳	77	I want exercise solvers to review the submissions of other exercise solvers (e.g. peer reviews).	0	0
58		I want to provide automatic testing for an exercise.	5	5
↳	59	I want to prevent exercise solvers from optimizing their code specifically for the test cases.	2	3
35		I want to structure my exercises chronologically and thematically.	5	4
23		I want to incentivize exercise solvers to solve the provided exercises (e.g. with prices).	4	4
18		I want to provide exercises in collaboration with other exercise makers.	4	3
40		I want to provide my exercises to exercise solvers who have no knowledge of Git.	4	3
62		I want to allow exercise solvers to access help autonomously when problems arise.	3	3
22		I want to know how many exercise solvers are solving the provided exercises.	4	2
68		I want to allow exercise solvers to use the programming environment they are comfortable with (e.g. IDE)	2	4
60		I want to allow exercise solvers to test their code manually (e.g. using the main method)	2	2
61		I want to explain code to the exercise solvers using artificial intelligence.	1	4
69		I want to force exercise solvers to setup a local programming environment.	2	2
19		I want to provide exercises in different settings.	1	3
↳	20	I want to provide exercises for university courses (e.g. FH).	1	1
↳	21	I want to provide exercises for courses of further education (e.g. CAS).	0	0

63	I want to allow exercise solvers to ask questions about exercises anonymously.	2	1
36	I want to include excerpts from different sources (e.g. code-snippets of official documentations)	0	0
37	I want to receive content contributions from exercise solvers.	0	0
↳ **38**	I want exercise solvers to be able to rate the content contributions of other exercise solvers.	0	0
41	I want to show code examples in my browser.	0	0
56	I want to provide discovery tours to explain the code base of an exercise.	0	0
↳ **57**	I want discovery tours to be expandable and maintainable (e.g. JSON under VC)	0	0
66	I want to simplify the setup of my exercises for the exercise solvers.	0	0
↳ **67**	I want to provide setup templates for my exercises (e.g. Docker).	0	0

I.: Importance, C.: Complexity, IID: Issue ID

VI.ii Exercise Solvers

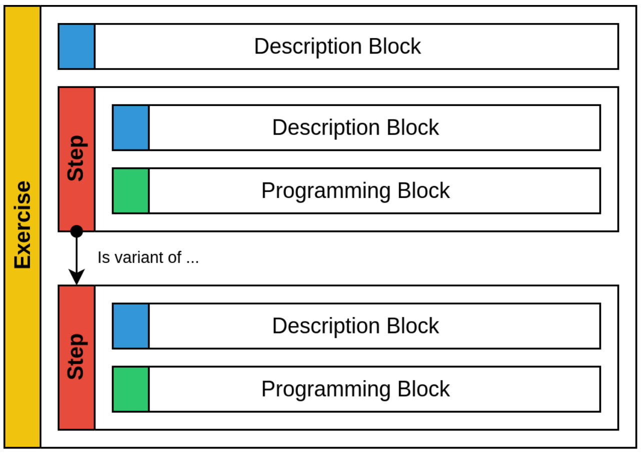
IID	Requirement	I.	C.
78	I want to be in control of how I solve exercises.	5	5
↳ **84**	I want to solve exercises in an environment I am comfortable with (e.g. IDE).	5	5
↳ **94**	I want exercises to be easy to set up.	5	5
↳ **97**	I want exercises to be provided in a Git repository.	3	5
↳ **96**	I want to solve exercises without having to install tools locally.	0	0
↳ **82**	I want to be in control of where I solve exercises (e.g. campus, home).	4	4
↳ **79**	I want to be in control of when I solve exercises.	5	3
↳ **80**	I want to save exercises in order to work on them at a later time.	5	2
↳ **83**	I want to access exercises from different devices (e.g. mobile).	2	3

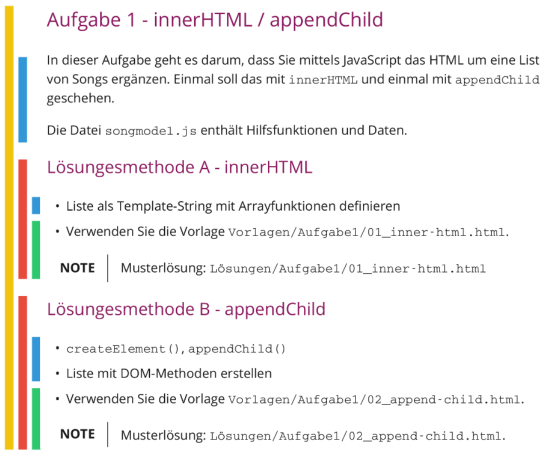
↳ ↳	95	I want to solve exercises in an online editor.	4	4
↳ ↳	98	I want to view the solutions of an exercise in a browser.	3	1
↳	81	I want to be in control of which exercises I solve.	5	1
↳ ↳	85	I want to see the solutions of an exercise without having to solve it.	5	1
	86	I want to have an overview of my exercises.	5	5
↳	89	I want to know which exercises belong to which lecture (i.e. week).	5	2
↳ ↳	90	I want to know which exercises belong to the current lecture.	4	2
↳	91	I want to know which exercises belong to which topic.	3	2
↳	88	I want to know which exercises are mandatory.	4	1
↳	87	I want to view all contents of a lecture in one central location.	0	0
↳	92	I want to know which exercises have mandatory attendance.	0	0
	93	I want to solve exercises that are similar to the exam.	5	5
	100	I want to receive feedback for my exercise submissions.	5	5
↳	107	I want my solutions to be validated automatically.	4	5
↳	101	I want to view feedback together with my submission.	3	4
↳	108	I want to know why my solution is right or wrong.	5	1
↳	102	I want to receive feedback in the form of text.	4	1
↳	103	I want to receive feedback directly on the relevant lines of code.	1	4
	104	I want to solve exercises in a reasonable amount of time.	3	4
	105	I want to solve exercises multiple times (e.g. for repetition).	2	2
↳	106	I want to remind myself which exercises I would like to repeat.	1	1

99	I want to solve exercises that use game-like elements (e.g. Kahoot).	0	0
109	I want exercise makers to enforce certain tools and standards.	0	0
110	I want to solve exercises with a more dynamic scope (e.g. mini-projects).	0	0

I.: Importance, C.: Complexity, IID: Issue ID

VII Examples of our Data Structure





Aufgabe 1 - innerHTML / appendChild

In dieser Aufgabe geht es darum, dass Sie mittels JavaScript das HTML um eine List von Songs ergänzen. Einmal soll das mit `innerHTML` und einmal mit `appendChild` geschehen.

Die Datei `songmodel.js` enthält Hilfsfunktionen und Daten.

Lösungesmethode A - innerHTML

- Liste als Template-String mit Arrayfunktionen definieren
- Verwenden Sie die Vorlage `Vorlagen/Aufgabe1/01_inner-html.html`.

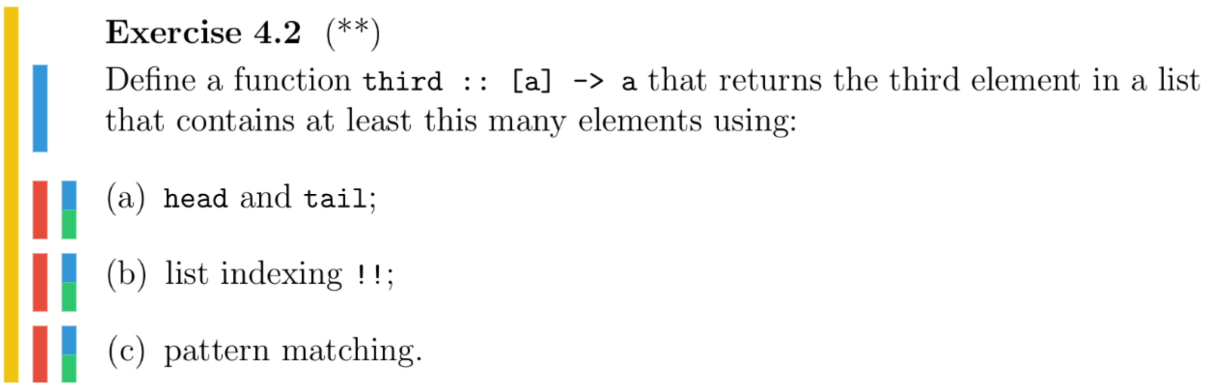
NOTE Musterlösung: `Lösungen/Aufgabe1/01_inner-html.html`

Lösungesmethode B - appendChild

- `createElement(), appendChild()`
- Liste mit DOM-Methoden erstellen
- Verwenden Sie die Vorlage `Vorlagen/Aufgabe1/02_append-child.html`.

NOTE Musterlösung: `Lösungen/Aufgabe1/02_append-child.html`.

Our data structure applied to an exercise of Web Engineering 1



Exercise 4.2 (**)

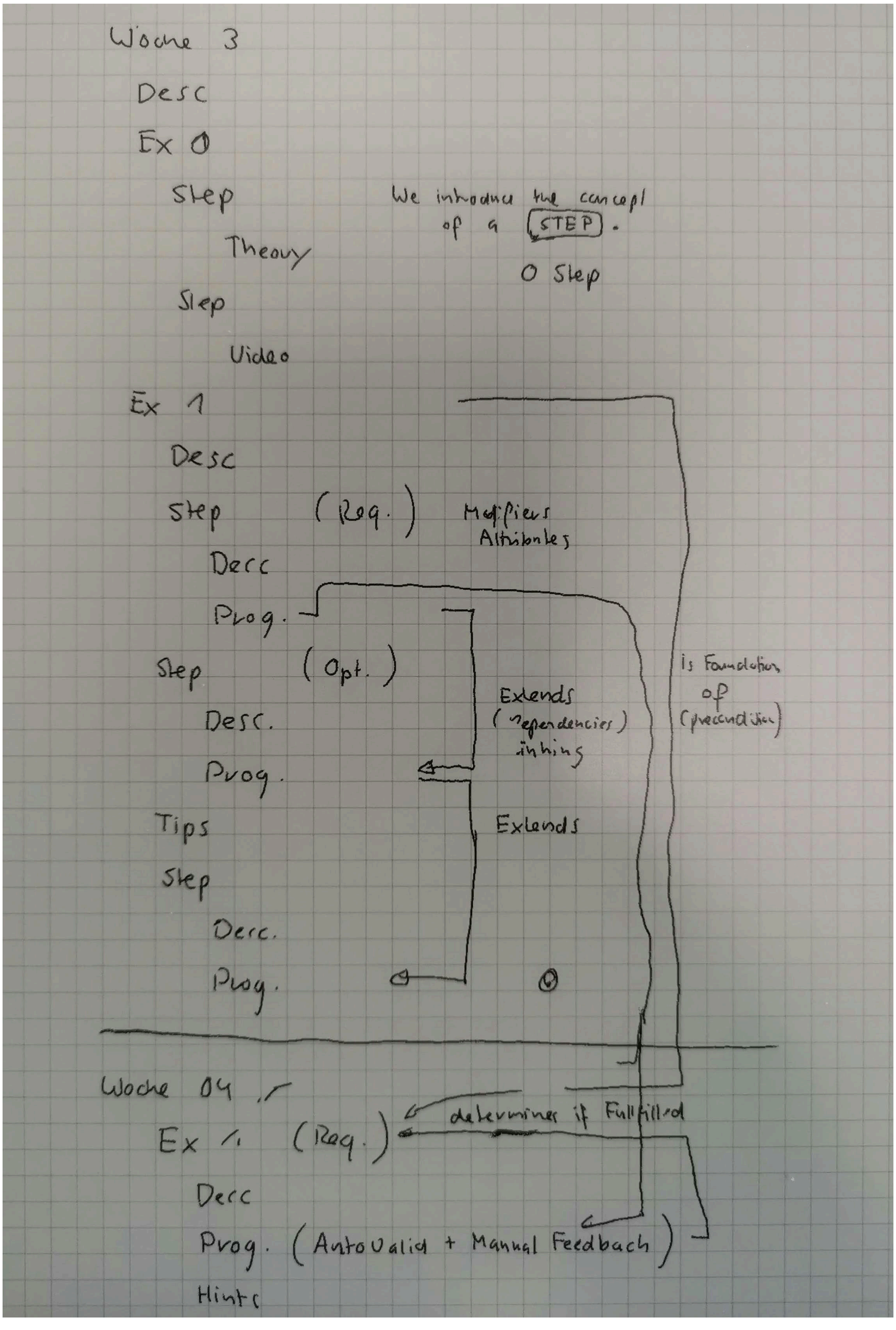
Define a function `third :: [a] -> a` that returns the third element in a list that contains at least this many elements using:

(a) `head` and `tail`;

(b) list indexing `!!`;

(c) pattern matching.

Our data structure applied to an exercise of Functional Programming



Our data structure applied to exercises of C++

VIII Acknowledgements

Mathias: My friend and ex-coworker Jan Hohenheim for being my Rubber-Duck when I struggled with getting Image Magick running on GitLab CI and for showing me tools to find relevant research papers.