CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

A Better Programming Teaching Tool

An Interactive Teaching Platform with Course Management, Real-Time Insights and AI-Generated Hints

Bachelor's thesis in Computer science and engineering

Gustav Dalemo
Oscar Enes Rei
Victor Hedqvist
Filip Hultgren
Tim Persson
Alexander Werinos

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Bachelor's thesis 2025

A Better Programming Teaching Tool

An Interactive Teaching Platform with Course Management, Real-Time Insights and AI-Generated Hints

Gustav Dalemo
Oscar Enes Rei
Victor Hedqvist
Filip Hultgren
Tim Persson
Alexander Werinos

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

A Better Programming Teaching Tool
An Interactive Teaching Platform with Course Management, Real-Time Insights and AI-Generated Hints
Gustav Dalemo, Oscar Enes Rei, Victor Hedqvist, Filip Hultgren, Tim Persson, Alexander Werinos

Supervisor (Handledare): Alex Gerdes, CSE
Advisor: Alex Gerdes, Chalmers University
Examiners: Patrik Jansson and Arne Linde, Department of Computer Science and Engineering
Graded by teacher (Rättande lärare): Wolfgang Ahrendt, CSE

Bachelor's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

Bachelor's thesis template for LaTeX for CSE
An Interactive Teaching Platform with Course Management, Real-Time Insights, and AI-Generated Hints
Gustav Dalemo, Oscar Enes Rei, Victor Hedqvist, Filip Hultgren, Tim Persson, Alexander Werinos

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Learning to program presents significant challenges for beginners, not only due to technical hurdles, such as installing and configuring development environments, but also from the cognitive demands of mastering unfamiliar concepts and problem-solving strategies. These difficulties are often amplified in educational settings, where teachers frequently need to assist students with repetitive setup issues or answer similar conceptual questions. This thesis presents the design and implementation of a web-based application that aims to lower the entry barrier to programming by allowing students to write and execute code directly in the browser without any installation. The platform features real-time task submission, automated progress tracking, and AI-generated subgoals and hints that are tailored to each student's progress. Teachers can create and manage tasks, monitor student performance, and adapt content to meet students needs. Through the integration of a user-friendly interface, performance analytics, and controlled AI assistance, the platform aims to reduce the repetitive workload for teachers, foster a healthy and pedagogically sound relationship with AI among students, and ultimately provide a more efficient and effective programming education environment.

Sammanfattning

Att lära sig programmera innebär betydande utmaningar för nybörjare, inte bara på grund av tekniska hinder, såsom installation och konfiguration av utvecklingsmiljöer, utan också på grund av de kognitiva kraven att behärska obekanta koncept och problemlösningsstrategier. Dessa svårigheter förstärks ofta i utbildningssammanhang, där lärare ofta behöver hjälpa elever med återkommande installationsproblem eller besvara återkommande frågor.

Denna rapport presenterar designen och implementationen av en webbaserad applikation som syftar till att sänka tröskeln för att börja programmera genom att låta elever skriva och köra kod direkt i webbläsaren utan någon installation. Plattformen innehåller funktioner för uppgiftsinlämning i realtid, automatiserad framstegsöversikt samt AI-genererade delmål och ledtrådar anpassade efter varje elevs individuella utveckling.

Lärare kan skapa och hantera uppgifter, övervaka elevprestationer och anpassa innehållet efter elevernas behov. Genom integrationen av ett användarvänligt gränssnitt, prestationsanalys och kontrollerat AI-stöd, strävar plattformen efter att minska det repetitiva arbetet för lärare, främja en sund och pedagogiskt meningsfull relation till AI för elever, samt i slutändan erbjuda en mer effektiv och ändamålsenlig utbildningsmiljö för programmering.

Keywords: programming education, web-based learning, AI-generated hints, student progress tracking

Acknowledgements

We want to send a big thank you to our supervisor, Alex Gerdes, who gave us great support and guidance on our project. We also like to thank Daniel Berg and students from NTI Johanneberg who made user testing possible.

We acknowledge that AI-assisted tools, including Grammarly and LanguageTool, were used to improve the grammar and clarity of certain sections of this thesis. All content was authored, reviewed, and approved by the authors.

Contents

List of Figures

1

Introduction

As society becomes increasingly reliant on digital technologies, the demand for programmers continues to grow rapidly [1]. In response, educational institutions face the pressing challenge of equipping students with foundational programming knowledge. However, for many beginners, the technical complexity of setting up development environments and navigating unfamiliar tools can discourage engagement before learning has even begun [2]. These obstacles are aggravated in school environments, where class time is limited and students have varying levels of digital literacy.

Meanwhile, teachers face the dual challenge of delivering instructions for the entire class while simultaneously providing individual support tailored to each student's unique needs and ambitions. This can be overwhelming even for skilled teachers, particularly as class sizes grow [3] and many students repeatedly ask the same or similar questions. Consequently, teachers may find themselves devoting a significant amount of time to resolving technical and setup-related issues that are unrelated to the core educational content.

Furthermore, the rise of generative AI tools has introduced new complexity. Although these tools can offer helpful explanations and guidance, they are often misused by students to intentionally or unintentionally obtain complete solutions, thereby undermining the learning process. It also raises concerns about academic integrity, which has resulted in schools incorporating restrictive policies regarding the use of AI [4]. However, outright bans do little to address the underlying needs that lead students to seek AI support in the first place and are counterproductive. Given these challenges, there is a clear need for tools that better support both students and teachers, especially in the early stages of programming education.

1.1 Purpose

The purpose of this project is to develop a web-based educational platform that aims to simplify learning to program for students while reducing the workload on teachers. The platform is intended for use in introductory programming courses and is designed to:

- Lower the entry barrier for students by removing the technical setup with an accessible, installation-free environment.
- Reduce repetitive workload for teachers by allowing them to create, distribute, and manage programming tasks within a unified interface.
- Offer AI-generated, step-based hints that intelligently guide students in solving problems without revealing the full answers.
- Track student progress to allow teachers to provide quick and precise individual support.

By combining a user-friendly, browser-based coding environment with AI-generated hints and automated performance tracking, our system supports both effective learning and efficient teaching. The platform encourages students to focus on core programming skills without being overwhelmed by tooling or installation issues while also enabling teachers to focus their attention where it is most needed.

1.2 Identified Challenges to Address

In response to the needs identified in Section 1.1, this part outlines five key challenges that this project aims to address that commonly hinder effective programming education.

Technical Barriers for Students

Many introductory programming courses begin not with learning to code, but with overcoming logistical and technical obstacles. Students are often required to install development environments, configure packages, and navigate unfamiliar tools before writing a single line of code. These hurdles consume valuable classroom time, as they often require teacher intervention and risk discouraging students early in the learning process. This is particularly problematic in schools where a wide range of operating systems and devices are used, such as Chromebooks and tablets.

One way to mitigate these issues is to offer an installation-free browser-based environment. In such a system, all necessary tools could be pre-configured and accessed online, allowing students to start working immediately without setup. It would also ensure cross-platform compatibility and remove hardware-related limitations, ensuring that all students can participate fully in programming activities regardless of their device choice or technical expertise.

Reducing Teacher Workload

Programming teachers frequently face repetitive administrative and pedagogical tasks such as answering the same questions from multiple students, manually assessing student progress, and addressing setup and configuration issues. This reduces

the time available to focus on the educational content of the course and on students who require additional support or are ready for more advanced challenges.

The previously mentioned installation-free browser-based option is also a potential solution to reduce teacher workload. This would allow teachers to have full control of the environments in which their tasks will be run and would free up time that teachers would otherwise have to spend helping students with their setup-related issues. That, along with intelligent support tools such as teacher-defined and AI-generated hints, could help answer common student questions more efficiently. Furthermore, integrating automatic progress tracking of student interactions with tasks would allow teachers to not only provide more accurate individual support where needed, but also monitor the performance of a task and make group trends more visible. This could encourage them to make more agile decisions in their education methods. However, there is an ethical aspect that needs to be taken into consideration when monitoring individual students' progress and data. A detailed discussion on this topic can be found in Section 5.3.

Responsible integration of AI in the classroom

With AI being a relatively new tool, many schools prohibit its use [4], viewing it as a method of cheating. However, AI's capabilities make it incredibly tempting for students to use, possibly leading them to disregard these rules. By doing so, students not only risk disciplinary consequences, such as being suspended or other penalties [5], but they also compromise the quality of their learning. While AI can serve as a powerful assistant, it comes with risks, such as generating overly complex responses, hallucinating incorrect information, or causing students to rely too heavily on AI-generated solutions instead of developing problem-solving skills.

To address these challenges, an educational platform could embrace the use of AI in a controlled and structured manner. By controlling the interaction with AI, the platform could ensure that it does not provide direct solutions, but instead generates context-aware hints that encourage critical thinking and problem-solving. This would allow students to progress independently while reducing the risk of misuse or over-reliance on AI-generated content.

Usability and Interface Design

When starting out, professional integrated development environments (IDEs) can be overwhelming, with a steep learning curve due to their complex interfaces and broad feature sets. Beginners benefit from tools that are intuitive, simple, and that focus on core learning objectives.

To improve accessibility for beginners, a platform might prioritize a minimal and user-friendly design with only the essential features, such as a simplified code editor, syntax highlighting, and easy access to task instructions and hints directly within the browser. Reducing the cognitive load in this way could help students focus on the core learning objectives rather than navigating complex user interfaces.

Supporting a Range of Programming languages

Controlling the programming environment within a browser-based platform comes with both flexibility and responsibility. Unlike traditional tools, where users manage their runtime environments, our system must explicitly define and maintain the execution contexts for each supported language. This places the burden of compatibility on the platform itself.

A platform could start tackling this by supporting common beginner languages, preferably some that accommodate different programming paradigms, such as imperative and functional programming. Ultimately, a platform's architecture would be extensible, allowing for new languages to be added over time to ensure that it can evolve with the needs of the curriculum requirements.

1.3 Related Work

During our research we looked at applications that have solved similar problems to the ones we aim to solve. We also studied articles on how AI can enhance learning to understand what approaches are effective and how we could best implement AI in our own work.

Applications

For beginners trying to learn programming, there are a lot of alternatives like Scratch [6], Khan Academy [7], JETBRAINS Academy [8], CodeCombat [9], replit [10] and w3schools [11]. All of these websites are used by students of different ages and stages of their programming journey. They are well-known websites with both strengths and weaknesses.

Scratch is a block-based programming tool which is mostly used by beginners and children. It is used to give children a basic understanding of how programming works. Scratch is used in self-learning, in school environments, and in libraries [12]. While Scratch does not support text-based programming, it still serves as a good entry point for conceptual understanding of programming.

Khan Academy [7] is another similar programming website, it is a personalized learning platform not only for coding but also math, science, history, and other subjects. Khan Academy gives the teacher the ability to see an overview and the progress of their students. It also gives the student an overview of their progress across multiple courses.

JetBrains Academy [8] offers structured programming courses integrated with an IDE-based development environment. It requires the installation of an IDE and consists primarily of pre-made courses.

CodeCombat [9] and W3Schools [11] are well-known interactive platforms for learn-

ing programming. They include tutorials and exercises for learners with varying levels of experience. CodeCombat [9] also incorporates gamified elements to support fun learning. It also includes tools to monitor student progress and manage course content.

Replit [10] is a platform that allows users to write and execute code directly in the browser without installation. It also gives the user the ability to create applications with the help of AI. Replit [10] uses a chat-based AI system where the user can input prompts themselves. Teachers can create a classroom, share assignments, and see students' progress in real time.

Research on AI hints

In research conducted by Natalie Keisler et.al. [13], the authors explored how large language models (LLMs) can support programming education by providing various types of feedback. A key insight from their study is the importance of prompt formulation. To generate good responses from an LLM, prompts should not only include data such as the task description and the student's current solution, but also specify the type of feedback expected and the criteria the response should meet. These criteria might include avoiding direct code suggestions and instead focusing on explanatory text, limiting response length or not including code from the model solution.

To ensure students receive helpful and relevant guidance without revealing the full solution, tasks can be broken down into smaller, manageable subgoals. The subgoals consist of actions that can be taken to proceed with the task from where the student currently is. This approach is discussed in research by Anastasiia Birillo et al. [**one-step**], where the authors propose using subgoal decomposition to generate step-by-step programming hints. By identifying these subgoals, their system is able to provide focused suggestions that steer students in the right direction without revealing the entire solution to the task.

2

Background

This chapter outlines the technical methods and frameworks used in both the frontend and backend development of the platform. A basic understanding of these methods is essential to grasp the design and implementation decisions.

2.1 Authentication, Authorization and Security

Authentication, authorization, and security are essential in any web-based application. Authentication refers to verifying a user's identity, typically through credentials such as a username and password, but can also include biometric verification, security tokens, or multi-factor authentication methods. Authorization refers to what privileges the authenticated user has, what information the user can see, as well as which functions are available to the user. Security measures such as password hashing protect sensitive data by ensuring that passwords are never stored directly in plain text. Session management allows the system to track user activity securely after login, with sessions typically configured to expire after a certain period or upon logout.

2.2 Frontend Tools

This section lists the tools used in the frontend of the platform. Together, they allow for an interactive and user-friendly interface.

Next.js

The project uses Next.js, which is a framework built on top of the React library. React is a library developed by Meta for creating user interfaces [14]. Next.js can be used for full-stack development, but it is only used in the frontend for its advantages with routing [15]. It uses nested routing, which automatically creates routes based on the names of the project's folder structure. Each route is a directory that includes a page file and an optional layout file. The page file contains the page-specific content, and the layout file is used for persistent UI elements [16]. Next.js also allows dynamic routing. This makes it possible to route to pages even if the segment name might not be known in advance [17]. A segment refers to the portion of the URL that

corresponds to a specific folder or page name and determines the destination the user is routed to.

React

Since Next.js is a React framework, it uses functionality from React, like hooks and React components. Hooks are used so the developer can use different React features in their components. It can save user input, cache information for calculations, or connect and synchronize with external systems [18]. React components are reusable building blocks. These improve the quality of the code since they allow splitting up the source code for better flow and maintainability.

TypeScript

The frontend is written in TypeScript, a programming language built on JavaScript, that has a type sensitive syntax. JavaScript is a lightweight interpreted programming language, and it is the most well-known programming language for web development [19]. The benefit of using TypeScript over JavaScript is the type sensitive aspect minimizing runtime errors. When running TypeScript, it converts to JavaScript, meaning it can be run anywhere JavaScript can [20].

React-Bootstrap

React-Bootstrap is a frontend library which provides pre-built React components styled with Bootstrap [21]. Bootstrap is a frontend toolkit which gives the user prebuilt, customizable components and grid systems. It is used for both visualization and functionality, making frontend development easier. React-Bootstrap is a rebuilt version of the popular Bootstrap library [22]. Bootstrap is a frontend framework with ready-made responsive styling components. The difference between React-Bootstrap and Bootstrap is that React-Bootstrap uses true React Components, whereas Bootstrap is pure CSS components [21].

2.3 Backend Tools

The backend of the platform is developed using a collection of frameworks and technologies suited for building scalable, secure, and extensible web services. This section outlines the key technologies employed.

Spring Framework and Spring Boot

The Spring framework is an open-source Java platform that provides comprehensive infrastructure support for developing Java applications. Built on top of the Spring framework, Spring Boot streamlines the creation and deployment of Java applications. It offers embedded server support, auto-configuration capabilities, and a preconfigured "opinionated view". This opinionated view helps developers set up

fully working applications very quickly by intelligently providing pre-enabled products from the Spring ecosystem and third-party libraries into their application by adding certain dependencies to their classpath. Some of Spring Boot's key advantages are rapid application development (RAD) and its strong emphasis on modular design through dependency injection, supporting the Inversion of Control (IoC) design principle. These features make it particularly well-suited for the construction of RESTful web services and microservice architectures [23].

MongoDB and MongoDB Atlas

MongoDB is a widely used NoSQL database system designed around a document-oriented data model. Instead of storing information in relational tables, MongoDB stores data in flexible, JSON-like documents. This allows for variation of fields between documents, and the data structure can be changed over time [24]. The document model maps to the objects in the code, making them intuitive to work with. According to a survey, about 60% of all application changes required a corresponding database schema change, and about 90% report that each schema change took more than an hour to resolve [25]. Keeping application data models consistent with relational databases is difficult and inefficient, leading to production issues and increased coordination overhead. Therefore, to avoid these issues and to gain the benefits of a flexible and adaptable data model, we chose a document-based database for this project.

To simplify ddeployment of the database, this project uses MongoDB Atlas. MongoDB Atlas is a multi-cloud database service built by the same people behind MongoDB [26]. It is designed to simplify database management for developers and automate complex database tasks such as scaling, backups, and security. With its fully managed cloud infrastructure, developers can focus on building applications without the burden of manual database maintenance. Given the limited timeframe and the team's limited experience with deploying and managing databases, using a fully managed service was the easiest and lowest-risk solution. It also made it easier to collaborate across devices and ensure availability for all members of the development team.

Code Execution and Test Construction

Traditionally, when running code, you are running it directly on your computer. Since we want to streamline the process of starting out with learning programming, we instead move the code execution to another computer, a server. In this project, remote execution is achieved by the user submitting the code to our server, which then feeds this to an interpreter and returns the result. This way, we can quickly add support for a language as long as we can run the interpreter on our server. This approach is not without risk. We are permitting users to run unsanitized code directly on the server which is a large, if not the ultimate, security risk.

When structuring code to be run as tests, we rely on the Java framework ANTLR4.

ANTLR4 (Another Tool for Language Recognition) is a powerful parser generator used to read, process, execute, or translate structured text or binary files. It is widely used for building language interpreters, compilers, and domain-specific languages. ANTLR4 takes as input a grammar that describes the syntax of a language and automatically generates a parser in a target programming language such as Python, Java, Ruby or Haskell. The tool supports context-free grammars and has a vast library of community-made lexer and parser grammars. We use this to extract function definitions and variable declarations which, in turn are used to construct test programs.

AI and Prompt Engineering Concepts

AI has opened new doors for both students and teachers when used for educational purposes, allowing a more individually tailored educational experience. One of the most important parts is the usage of LLMs, for example ChatGPT, which is developed by OpenAI. LLMs are AI models capable of generating and understanding text similarly to how a human would. It has been trained on finding patterns in large amount of written data. For the most effective usage of these LLMs, prompt engineering is essential. Prompt engineering refers to the process of creating an instruction or a guide that is clear and easily understood by an AI model. It is an important part of using AI, and it guides the AI model in the right direction, controlling both the quality and relevance of the output.

3

Implementation & Theory

In this chapter, we will go through how we implemented our solution, and how those implementations aims to address the challenges identified in Section 1.2.

3.1 Scope and Limitations

Since the plan was not to fully release the platform, the focus has not been on the security aspects. While some basic security features are present, such as password hashing and the sessions being kept in the backend, it will not be secure enough for a production release.

This project is not designed to function like popular platforms such as w3schools [11] or Codecombat [9], which offer pre-made courses and tasks for students from the outset. Instead, it is intended for classroom use, where the teacher creates and adds the courses and tasks. Because of this, time will not be spent on creating pre-made courses or tasks. Except for debugging and testing reasons.

We do not intend to conduct a study on how well our program performs in a classroom setting compared to a more traditional way of learning programming. This is simply because we do not have either the time or the resources to conduct such a study. Instead, we will do a user evaluation, which gives us a good opportunity to receive feedback on how well our product performs and where it needs to be improved.

3.2 Architecture

The program follows a classic client-server architecture, with the frontend acting as the client and the backend as the server which communicates using a HTTP REST API. When the user performs an action, such as opening a new page or trying to log in, the client sends out an HTTP request. This could be a GET, POST, PUT, PATCH or DELETE which gets sent to the backend. The different requests make different actions, GET is used to request data, PUT creates or replaces data, POST sends data to the server, which in our case is used to save data. PATCH updates current data on the server, and DELETE removes current data on the server [27].

Figure 3.1 illustrates the detailed architecture of our program. It outlines the four main actions a client can initiate: /runcode with type CODE, /runCode with type TEST, /tasks and /getAiHints. Each action triggers a specific flow within the backend involving different handlers and modules. The figure shows the sequence of operations carried out for each request, from the initial client call through intermediate processing components such as test handlers, LLM clients and parsers. A detailed diagram of the software architecture can be found in Appendix C.1.

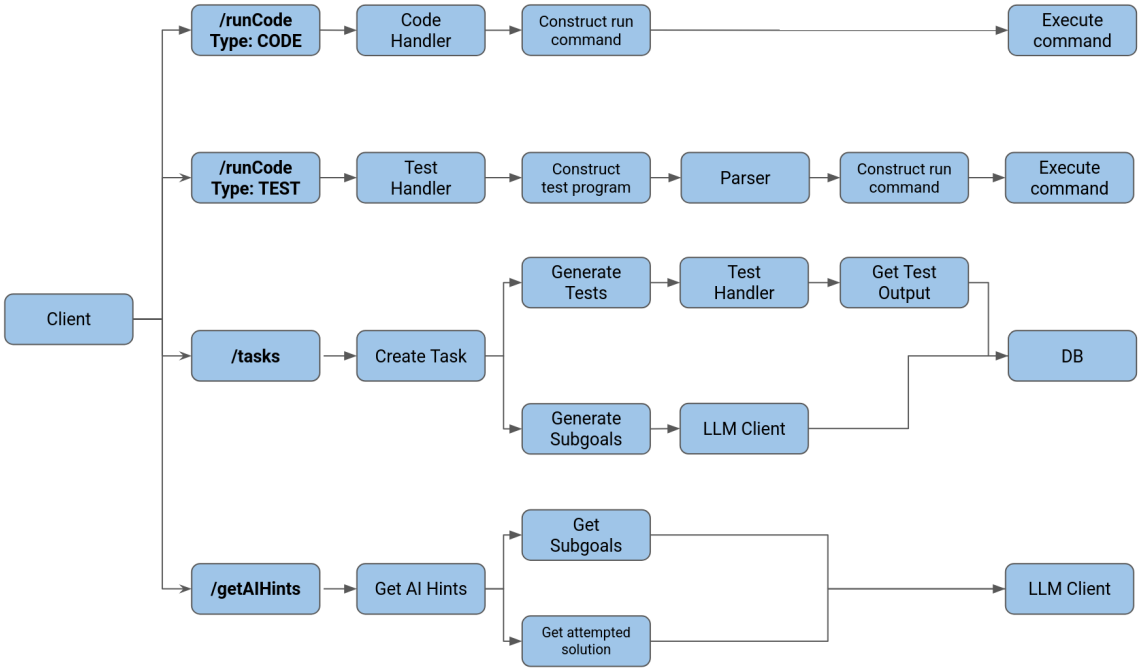

Figure 3.1: Integration between Frontend and Backend. It shows the request flow for the main actions.

3.3 Authentication and Authorization

The user needs an account to use the platform. All functionality, with the exception of registration and logging in, needs authorization to access information or perform an action. As shown in Figure 3.2, a user session is created in the backend when a user logs in. The session ID is then stored in the database and passed to the client's browser via browser cookies.

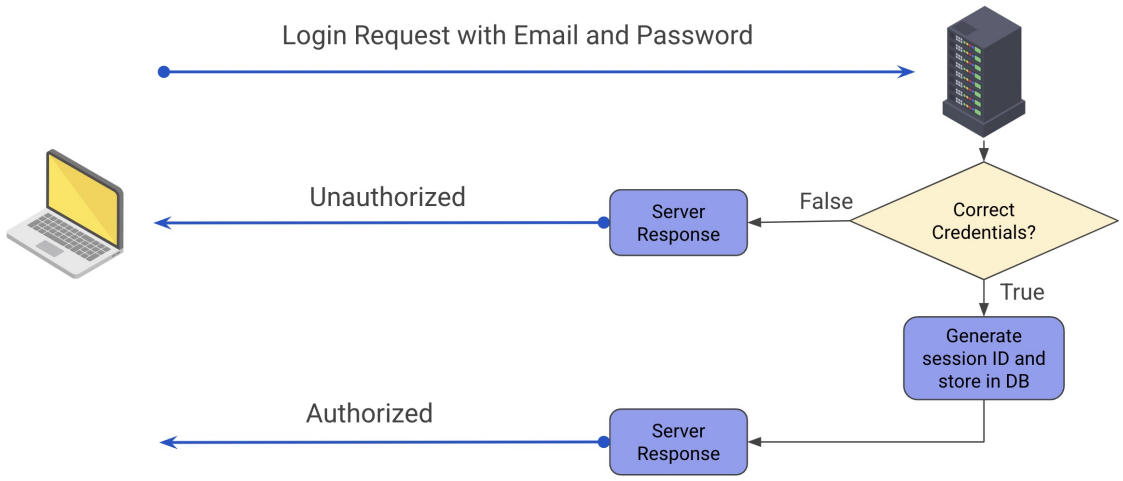

Figure 3.2: User login. The log in process of our program.

When logging out, the session ID stored in the browser gets passed in the header field of the logout request. The corresponding valid session stored in the database is then deleted (See Figure 3.3).

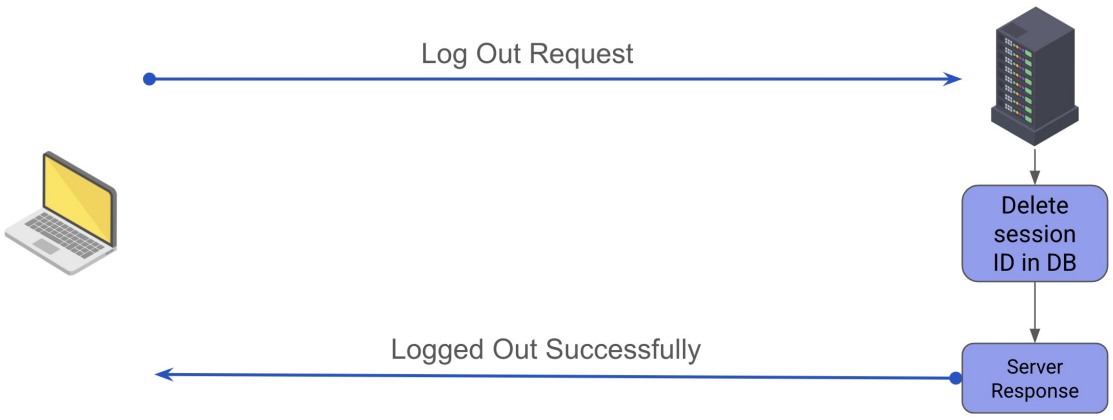

Figure 3.3: User logout. The log out process of our program.

When accessing a page or refreshing a page that requires authorization, a request is sent to the backend, which checks for a valid user session just like the logout request. Depending on whether the valid session exists or not, access to the page or the requested action triggered by the user is authorized accordingly (See Figure 3.4).

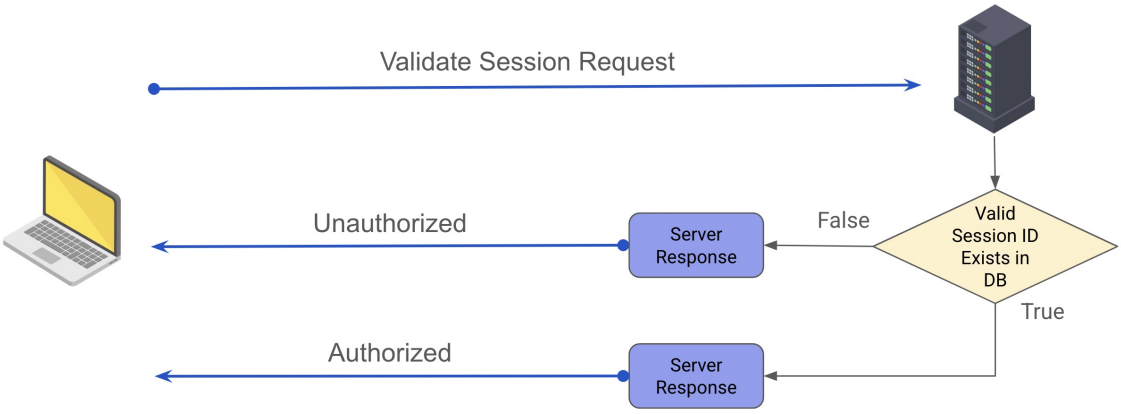

Figure 3.4: This visualizes what happens when a user navigates to a page that requires authorization.

Only one client can be logged in to a specific user at any given time, meaning there is only ever one valid session ID for a specific user. This is important because it allows the server to identify which user is sending a request and handle it accordingly. When a client is logging in to a user who is already logged in elsewhere, the newer authentication process takes priority, and the first user gets logged out. This is to prevent a user from being unable to log in to their account.

3.4 Task creation

When a client requests the creation of a task, the server follows a pipeline to validate the input, enrich the task with AI-generated subgoals later used for AI-hints, before finally saving the task to the database (see Figure 3.5). The process consists of the following steps:

1. Receive request: The server receives a request that contains a task to be created with attributes such as a description, a suggested solution defined by the teacher, the course to which it belongs, and other relevant task data.

2. Validate attributes: Before proceeding, the server validates the received attributes to ensure that the required fields are present and well-formed. If any attribute is missing or invalid, such as if the course to which the task belongs does not exist, the server returns an appropriate exception response to the client.

3. Extract task data: If validation is successful, the server extracts the task description and the suggested solution. These attributes are essential for the next step, where subgoals are generated.

4. Prompt AI for subgoals: The server formulates a prompt, as shown in Appendix B.1, for the AI component using the extracted information. The AI

is instructed to generate step-by-step intermediate objectives that can help guide the students toward the solution without directly providing it. The AI-generated hints are then partly based on these subgoals when providing hints for the students, which are discussed in more detail in Section 3.5.

5. Update and return task: Once subgoals are received from the AI, the task object is updated to include these subgoals. This updated version of the task is then stored in the database before being finally returned to the client.

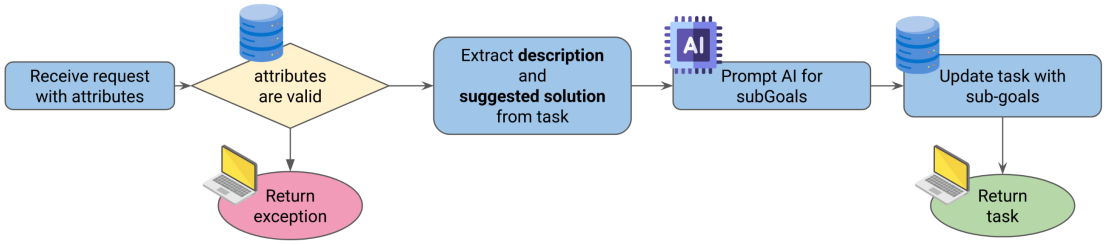

Figure 3.5: Server-side process for creating a task, including creation of AI-generated subgoals.

By automating subgoal generation at the time of task creation, the system directly supports teachers by reducing the need to manually author detailed guidance for each task, although they still have the option to include their own defined hints if they find that useful. This contributes to addressing one of the key challenges identified in Section 1.2, namely to reduce teacher workload.

3.5 AI-hints generation

With the aim of resolving the challenge of responsible integration of AI and reducing teacher workload (outlined in Section 1.2), the system uses AI-generated hints to help guide students when they feel stuck. To prevent the AI from providing the full solution to a task, but still be as useful and accurate as possible, the server follows a structured sequence of steps to generate these hints (see Figures 3.6 and 3.7):

1. Receive hint request: When a student requests a hint for a task, the proccess begins with updating the current state of the student solution. This includes saving the code written so far and incrementing the counter that tracks the number of AI hints used for that particular task (see Figure 3.7).

2. The server proceeds by fetching necessary context from the task. This includes task attributes such as description, the suggested solution defined by the teacher, and previously generated subgoals.

3. Hints enabled check: The task object also includes a boolean value that deter-

mines if AI-hints are enabled for that particular task. If AI-hints are disabled, the server returns an appropriate exception to the client.

4. Fetching commenced solution: To further improve the accuracy and context-based nature of the hint, the server proceeds by fetching the current state of the solution written by the student so far. This information is stored in the corresponding student task log.

5. Prompting AI: The gathered information is assembled and sent alongside a prompt that is designed to produce a context-aware hint that guides the student forward without revealing the full answer as shown in Appendix B.1. For generating AI hints we use the OpenAI API.

6. Return hint: Upon receiving the hint, the server processes it and delivers it to the client for immediate use in the interface.

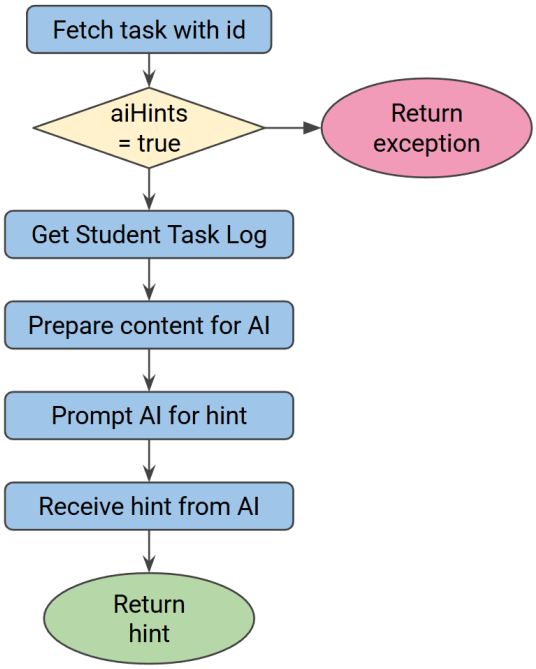

Figure 3.6: Server-side processing of an AI hint request.

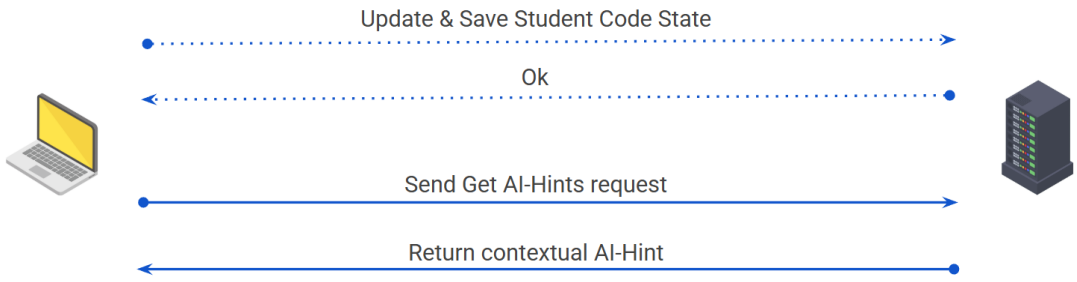

Figure 3.7: Client-server communication during an AI hint request.

By using subgoals as an intermediary structure, the system improves the pedagogical quality of the generated hints, and provides step-by-step support tailored to the student's current progress. The aim is to provide enough help to reduce the urge to use third-party AI tools, while still not revealing too much of the answer to encourage the student to solve the task independently. This also has the potential to free up time for teachers, allowing them to be where help is most needed.

3.6 Code Execution

To achieve the goal of lowering the barrier of entry for students and getting them started programming as fast as possible, we want to abstract the installation of a language and IDE. The obvious choice was to develop for the browser. A browser makes it possible to run on any device instantly while having just one development pipeline. Code is not executed in the browser but instead sent to a server which runs the code.

A request for running code consists of: [**user code**] + [**language**] + [**Type of request**] + [**Task ID**]. From this, we process the request differently depending on language and request type. For the type of CODE, we simply run the submitted code for the correct language and return the result (see Figure 3.8). For type TEST we manipulate the code and fetch test data from the task, more on this in Section 3.7.

A bash command is constructed from the recieved request where the user's code is included as an argument. Example when using python a Bash command looks like this: **python3 -e "print(“Hello World”)"**. How the bash command looks, of course, varies depending on the language. For example, Haskell uses **ghc -c /tmp/path/to/file.hs** to run interpreted code saved in a temporary file. The obvious issue with this approach is that of security. Anyone who has taken any interest in computer security knows that being able to execute code on a remote server is one of, if not the, most compromising vulnerability there is. To minimise this risk, we could run the user's code in a temporary user with minimal privileges, with no access to the file system. We also considered running containers with some sort of grouping. Several groupings were suggested, such as one container per student, course, teacher or language. The main concern for containers is resources, and that

each container would either have too much overhead or not enough resources to support a changed workload. We believe this is out of scope and not necessary to demonstrate the core ideas of the project.

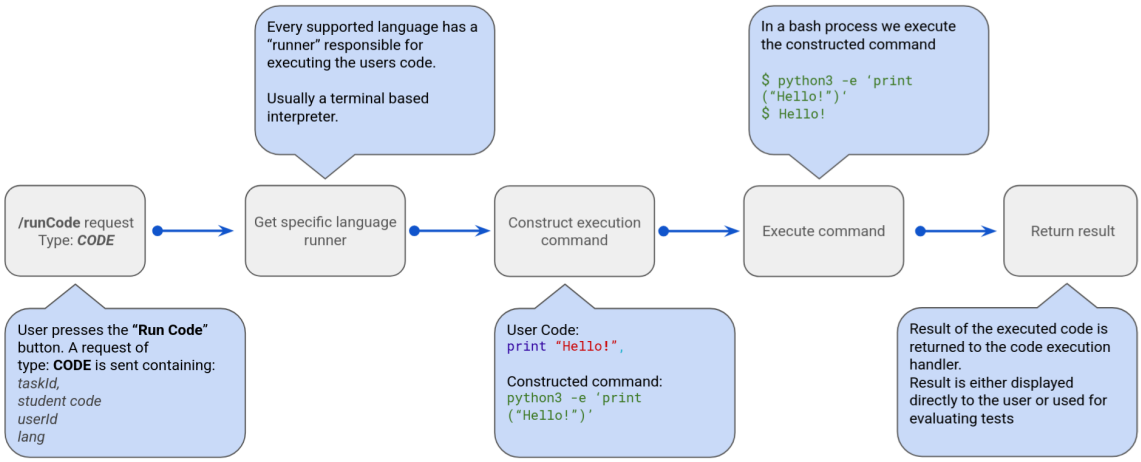

Figure 3.8: Order of operation when executing user-submitted code.

3.7 Test Construction

A big part of the project was the student and teacher overview. To get data in the overview, we need some way to run the generated tests on a student solution. The simplest way to ensure we can run a student's solution would be to force the use of a function signature. This way, we can just call the preset name of the function with the appropriate parameters. This might be a drawback since beginner courses teach the creation of users' own functions.

At the task creation stage we present the teacher with the option to "generate tests". At this stage the teacher can input test data that is then run against the suggested solution. The input and result of this are then stored together with the task and are used to test and verify a student's solution.

When executing tests, we parse the student's code and retrieve information about the solution and use this together with the test data, which was input at the creation stage of the task. In parsing, we retrieve the name and number of parameters in the function defined first. To execute the test, we construct a new temporary file consisting of [**variables, function definition, function call, printing result**]. The function call and printing of the result are done in a "main" function, which runs first on program execution. Below is an example of a Python program which reverses a list:

```
my_list = [1, 2, 3, 4, 5]

def reverse_list(input_list):
    reversed_list = []
```

```
    for i in range(len(input_list)-1, -1, -1):
        reversed_list.append(input_list[i])
    return reversed_list

if __name__ == "__main__":
    print(reverse__list(my_list))
```

This program is then run by the same logic that executes any other type of code, and the result is then compared to what is stored as correct in the task. The result is then returned to the user.

When constructing the function call, we can bump into the issue where a function does not return anything, only prints. When a function returns something, we want to print the result, when it does not, we only want to call the function. To achieve this, in Python, we check if the user's submitted code contains the string "return". If there is a return present, wrap the function call in a print statement, if not, just call the function.

3.8 User Evaluation

User evaluation was a very important part of the development process. We knew what problems our program aimed to solve, however, we did not know if our product had solved them. Therefore, we needed to perform user evaluation to find out if our solution had solved the problems that we aimed to solve. User evaluation also allowed us to receive feedback on our product which then helped us make small changes and bug fixes. These evaluations were done according to the steps in Appendix A.1 where the users were a group of high school students who were attending their first programming course. The students were asked to perform the tasks that are mentioned in Appendix A.1 and we encouraged them to use all the functionality that they could find. We did this because we wanted the students to explore our product themselves without us guiding them since that could affect the outcome, but we also wanted feedback on all of our functionality and not just parts of it. Parallel to the students performing our user evaluation, we had them fill in a survey which included all the most important information and questions that we wanted their feedback on. We decided to have this survey parallel to the user evaluation since it also acted as a guide for the students to follow in order to perform the tasks correctly. The result of the user evaluation is shown in Section 4.3.

4

Result

The result of the project is a platform that can be used in a classroom environment to improve both the learning experience for students and the teaching experience for teachers. Teachers have the ability to create courses containing tasks that students can solve with or without the help of hints that the teacher has written and AI-generated hints. Both students and teachers have access to performance statistics. Students can track their progress, while teachers can monitor student activity and adjust their teaching strategies accordingly.

4.1 Teachers Functionality

The teacher has a Dashboard, which is displayed in Figure 4.1. It acts as a homepage that contains basic information about the courses the teacher is a part of and the students of these courses. The dashboard also contains quick-actions to static features to ease navigation on the platform. It allows the teacher to see which courses they are a part of and links to pages to be able to edit and see statistics over these courses.

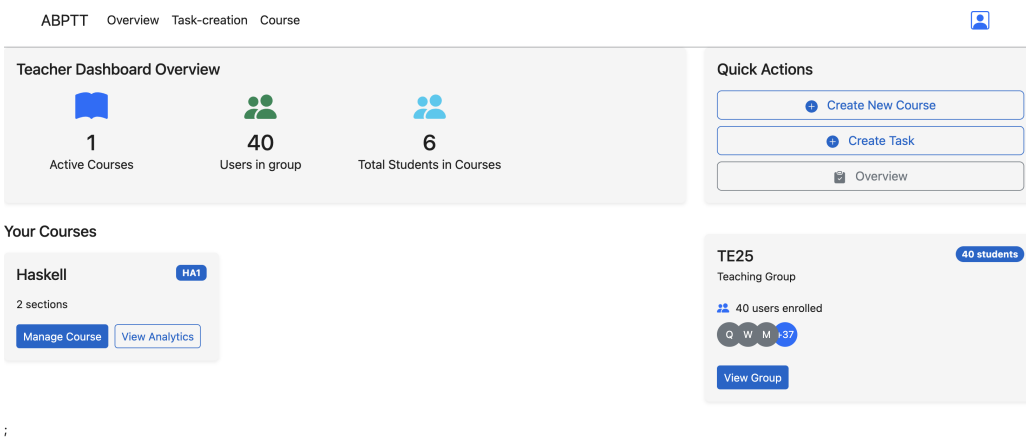

Figure 4.1: The teacher's dashboard. A visualization of what a teacher dashboard can look like. It includes statistics, courses, groups and quick actions.

The teacher has an overview page where they can get overall statistics on courses, sections and tasks and also see specific student's progress. These statistics include what tasks have been submitted and completed, but also how many submission attempts a student has made and how many AI hints were used to solve the task. See Figure 4.2 for the teacher overview for a specific task.

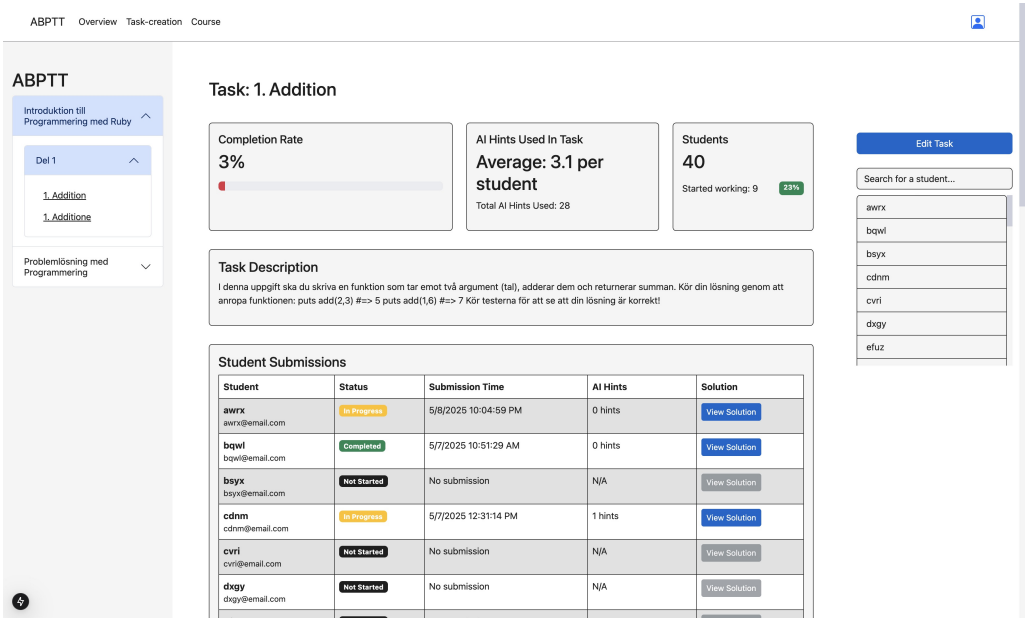

Figure 4.2: This page provides the teacher with an overview of a specific task, including student progress and relevant statistics.

The platform allows the teacher to create courses, sections, and tasks. When creating a course, the teacher must input a course name, course code and a description. At the same time, the teacher creates the course sections that help split the tasks of a course into different parts. For example, in a basic Python programming course, one section might include tasks focused on variables and data types, while another could cover control flow concepts such as if-statements and loops. Sections can also be used to organize tasks by week, helping to structure the course content over time. Before creating a course, the teacher sees a preview of how it will look, as seen in Figure 4.3. There is also a student enrollment feature which allows the teacher to add students to the course, and this can be seen in Figure 4.4. The teacher can search for specific students by name or email and sort them into different groups. There is also functionality for the teacher to edit their courses if something is wrong with them or if they just want to change some of their content.

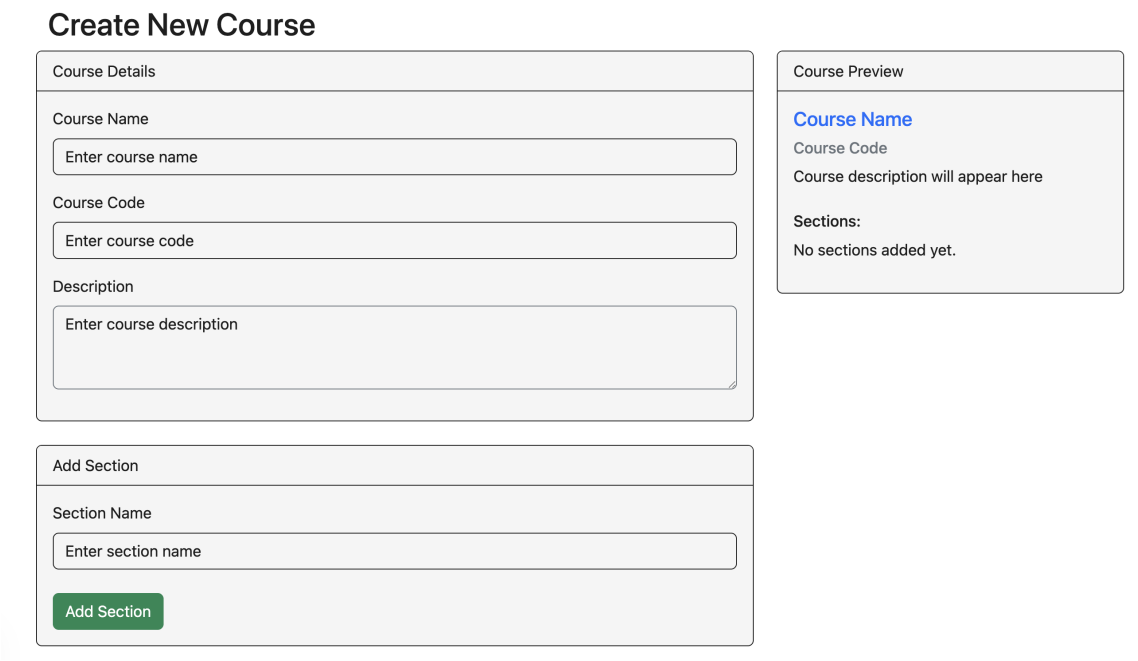

Figure 4.3: This is a part of the create course page. It includes forms to fill in when creating a course. It shows a visualization from the teacher's view of how it looks when creating a course.

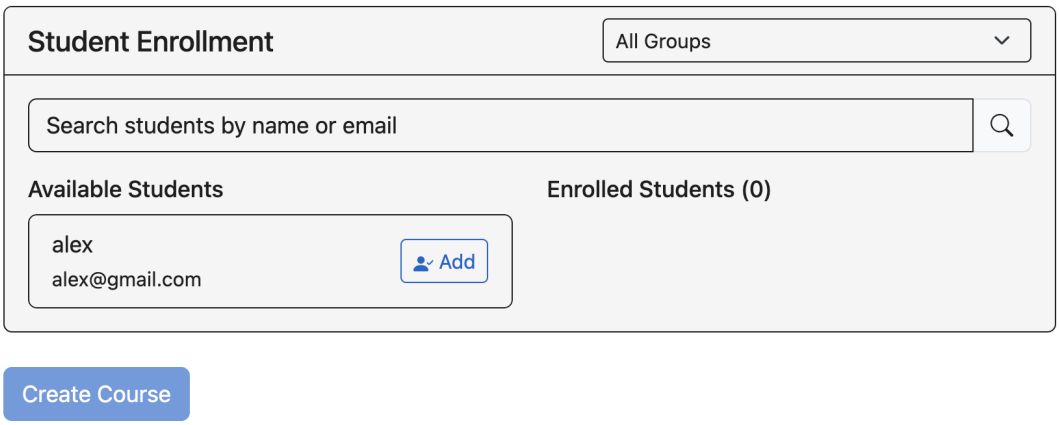

Figure 4.4: This is a part of the create course page. It includes a student enrollment form, which enables teachers to add students to their course and a submit button to create the course. It shows a visualization from the teacher's view of how it looks when creating a course.

The teacher also has a page for creating tasks as seen in Figure 4.5. When creating a task, the teacher can provide a task description specifying what the student is expected to do, a suggested solution used for evaluating student submissions and supporting the AI in generating appropriate hints, and a code template to help students get started on their task. When creating a task, the teacher has to specify a task name and what courses and sections it belongs to and which programming

languages the task is written in. Teachers have the option to create test cases for automatically verifying the correctness of student submissions, as well as to define tags for categorization. There is also an option to enable or disable AI-generated hints for students. In addition to AI-generated hints, teachers can manually write hints to guide students. These settings can be seen in Figure 4.6.

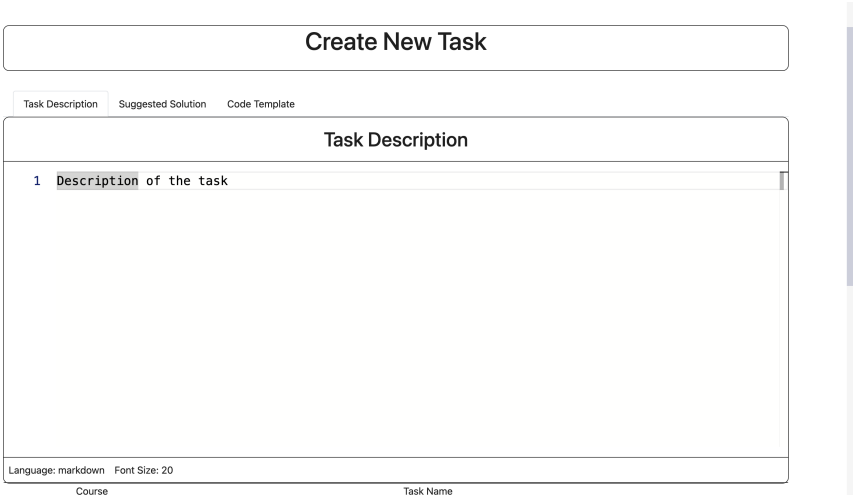

Figure 4.5: Part of the task creation. How the teacher inputs information when creating the task.

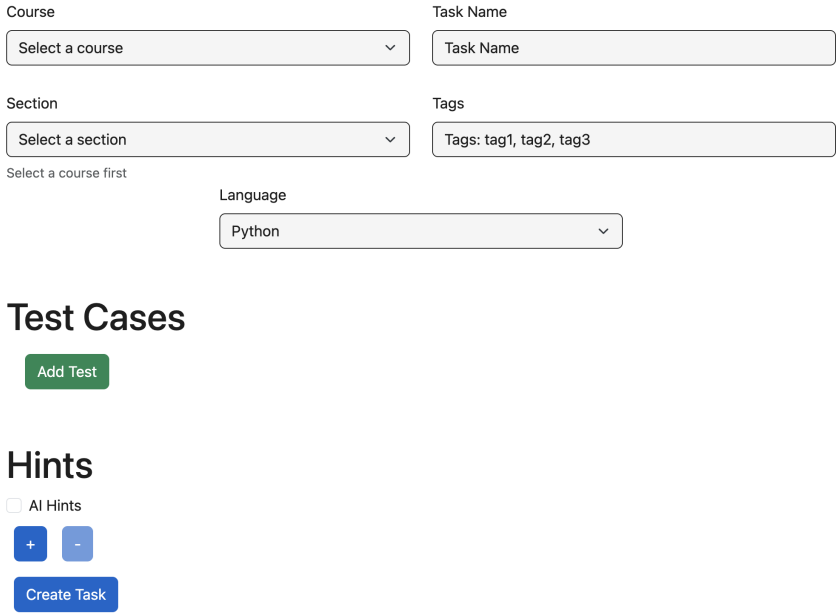

Figure 4.6: Part of the task creation. It shows what course and section the task should be assigned to and what the task name is. It also lets the teacher fill in more information like hints, programming language and tests.

4.2 Student Functionality

The student has a dashboard with similar functionality to the teacher, which is shown in Figure 4.7. It differs in which statistics are shown and which quick actions the student can use for easy navigation. Instead of getting an overview of its group, the student sees their recent tasks. This makes it easy for the student to navigate to tasks they have recently worked on and can continue where they left off.

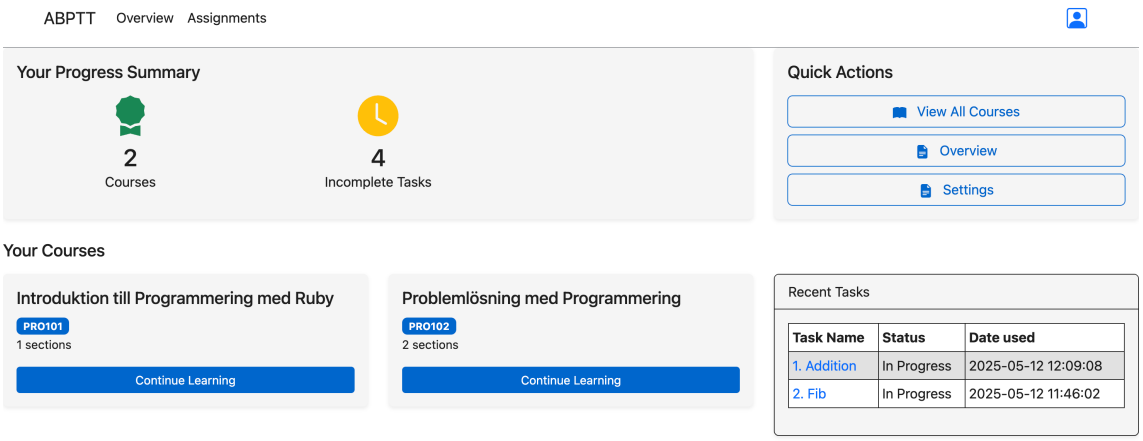

Figure 4.7: The student's dashboard. A visualization of what a student dashboard can look like. It includes statistics, courses, recent tasks and quick actions designed for students.

The students can see an overview page containing information and statistics on the progress. There is also an assignment page where a student can go through their courses and find tasks. On the assignment page, there is also a recent tasks panel to easily find tasks that have been worked on recently.

The assignments page is where the student will spend most of their time and work on different tasks. When going to the assignments page, the student will have to choose which task to work on. When a task has been chosen, the student sees the view that is shown in Figure 4.8. It consists of a code editor on the left side of the screen, containing a code template if one is given for the task. On the right side of the screen, the student can see the task description, the hints written by the teacher if there are any and the menu for AI hints if allowed by the teacher for that specific task.

If the teacher has allowed AI-hints, the student has the option to request an AI-generated hint. If the button is pressed, the student receives a hint tailored to their current progress within the task. These hints are presented as a step-by-step guide, offering information and guidance for the student without ever revealing the final solution.

When the student wants to run their code, they can press the "Run Code" button

to execute the code and see the output in the box below the editor. If they want to test their solution, they can press the "Run Test" button which runs the same tests generated at task creation. This will return either a "Test passed" or a "Test failed". If the solution passes the test, the student can then press the "Submit" button to submit their code.

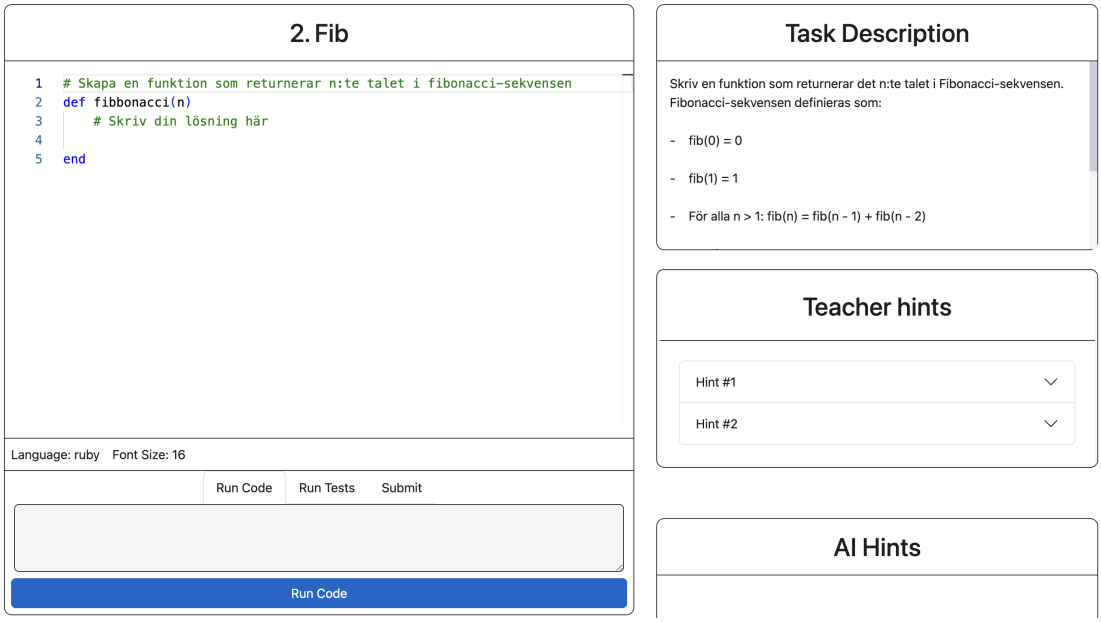

Figure 4.8: Student task-solving interface. It shows the interface used by students to solve coding tasks. It includes a code editor for writing code, a task description, and two separate hint sections—one with AI-generated suggestions and one with teacher-provided hints. Students can run, test, and submit their code within the same environment.

4.3 Result From User Evaluation

The results from the user tests were performed on 32 high school students. The reviews are mostly positive and show that our work has solved the problems that we intended to. The statistics show that the current programming course the students were attending was the first one for over 90% of them. This is relevant since this group of students is all in our target group.

UI and Task Solving

When the students were asked to navigate their way to the three tasks, we found that 87% of them found it easy to navigate our platform without any complications. The other 13% did not have negative feedback, but they did find it more challenging than the rest. This data shows that our platform is easy to navigate, where users do not have to spend too much time learning how to navigate. Instead, they can use our platform without complications, which is a big part of lowering the technical barriers of programming.

When we asked the question if the students ran the code and executed the tests we got that two students did not know how to run the code and the rest answered yes, around 87% of the students ran the tests, 10% did not run the tests and one student did not know that there were tests. These results show that most students found it easy to navigate, and that they used most of the functionality in our platform without complications.

AI Hints in Tasks

The users also used the AI-generated hints, where they were asked if the AI hints were helpful and relevant to the tasks.

Task 1 Addition

As illustrated in Figure 4.9, most students who used AI hints in this task found them helpful. Many used them to improve their code. Several students mentioned that the problem they were asked to solve was very simple and hints were not necessary.

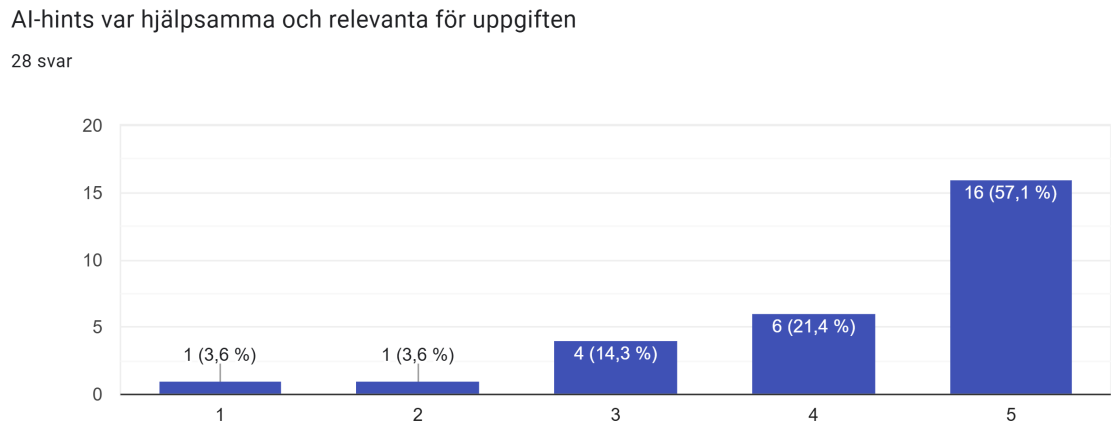

Figure 4.9: Diagram that shows how relevant the AI hints were in task 1.

Task 2 Fibonacci

As illustrated in Figure 4.10, many students found their hints in this task to be helpful. Some found the AI to give unnecessary hints such as changing naming. One student said that they were helpful only if you have previous knowledge. Some mentioned that the hints included methods that they were not taught in the course such as recursion.

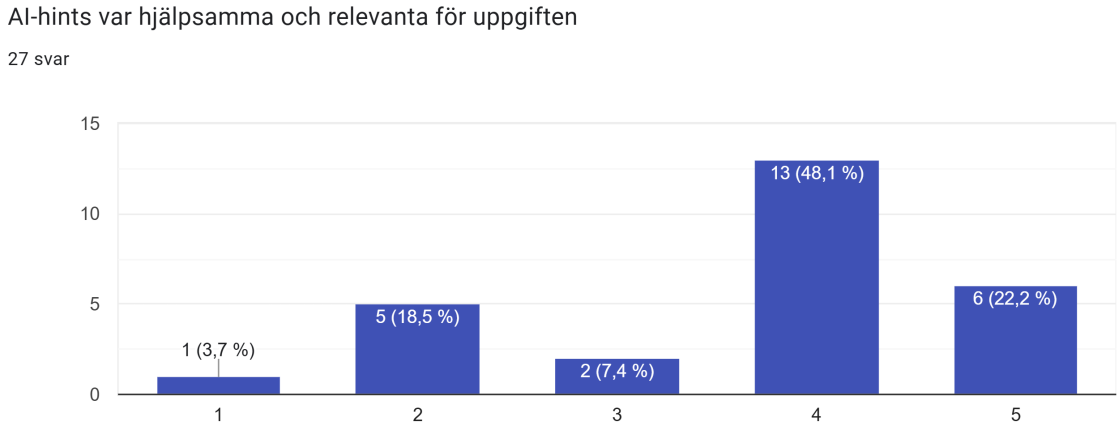

Figure 4.10: Diagram that shows how relevant the AI hints were in task 2.

Task 3 FizzBuzz Challenge

As illustrated in Figure 4.11, a large majority of students who used AI hints for this task found them helpful. One student mentioned that they liked that it gave them help that required them to think for themselves. Some students found them helpful in spotting typos and mistakes.

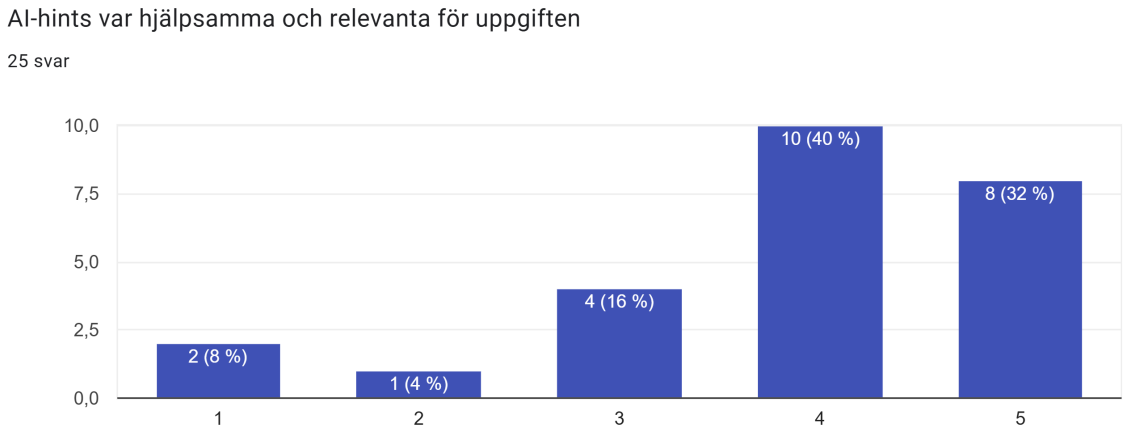

Figure 4.11: Diagram that shows how relevant the AI hints were in task 3.

The results show that our AI hints were, in most cases, useful and relevant. However, the cases where the students found the hints to include methods that they were not taught in the course happened because of how we have chosen to prompt the AI. We want the AI to give the students hints that lead them towards the suggested solution, in this case we had made the tasks ourselves and we did not know exactly what the students had learned or not. This led to us creating a task and a solution to the task that the students had not learned the methods necessary to solve it. The AI then hinted toward using these methods which confused the students. This shows that the teacher who creates the task must have a solution that uses the same methods as the users are expected to use. This also has a positive side to it since the AI hints are always based on what the teacher suggests. The students will then always receive hints that suggest the same methods that the teacher wants the students to learn and use.

Overview page

The students then gave feedback on our overview page. When navigating through this page, all but one student answered that they could find their personal overview page. They then answered if they could understand the information on this page on a scale from one to five where a one meant that they could not understand the information and a five meant that they could clearly understand. About 90% answered a 3 or higher with about 45% answering a 5. Some students also wrote that the overview page was easy to understand but it was quite a lot of information on one single page. This feedback shows that our overview page is highly relevant and useful but our choice of design and how we display this data is not optimized. One field where our overview page had a flaw was where the total amount of AI hints used was displayed. This field displayed the wrong information which was good feedback for us to be able to fix this problem.

5

Discussion

This chapter presents discussions regarding critical aspects of the project. It covers why we did things a certain way, future improvements, and challenges we encountered.

5.1 IDE vs Our Platform

This project was created with the intention of lowering the entry barrier for students who are learning to program for the first time. One of the key decisions made early in development was to delay the introduction of IDEs such as Visual Studio Code. Based on our own experience, setting up an IDE can be significant and overwhelming for beginners, as installation errors and configuration issues often lead to confusion and frustration.

The main goals of this project are to make programming more approachable for beginners and to reduce the repetitive workload placed on teachers. With our web-based approach, students can begin writing and running code immediately without the need to install additional software or understand how an IDE works. Learning both a programming language and a complex development tool simultaneously can be cognitively overwhelming. By allowing students to focus solely on learning the core programming concepts, without the added complexity of an IDE, they can build a stronger foundation and gain confidence early on.

A relevant pedagogical question arises: Are we doing students a disservice by bypassing the skill of setting up development environments? While environment configuration is an important part of a programmer's long-term skillset, it is arguably not essential at the earliest stage of programming. What is essential, however, is becoming familiar with core concepts such as variables, functions, debugging, and problem-solving, which our platform allows students to focus on fully. As discussed in the study by Martin Velez et al. [28], survey results showed that students, especially those in beginner courses, found the web-based IDE appealing precisely because it required no installation and provided a ready-to-use environment. Once students are comfortable with core programming concepts, the transition to a professional IDE might become smoother, as they no longer have to learn programming

fundamentals and a complex tool simultaneously. They are also better equipped to appreciate the value of advanced IDE features, which might have been overwhelming or confusing at an earlier stage.

To reduce complexity, our platform has an intentionally minimal feature set compared to full IDEs. IDEs often offer a wide range of advanced tools such as a debugger, code completion, and refactoring support. These are very useful for experienced programmers, but can be overwhelming for beginners. Our platform deliberately excludes many of these features to avoid this. However, this choice comes with trade-offs: students may not be exposed early to tools they will eventually have to learn as they are used in real-world development environments. Therefore, the aim with our project is not to replace IDEs, but rather serve as a transitional tool focused on the earliest stages of programming. Once students have developed a solid programming foundation, they can gradually be introduced to more advanced tools in subsequent courses.

5.2 Reflections on Code Execution

The ability to execute code is a corner stone in the functionality of the platform as well as test execution which has proven to be a rather challenging feature to get right.

Security

As mentioned earlier in the report, security is a big issue if we were to "launch" this platform to real users. To handle this, some sort of sandboxing or containment from the server's operating system would be necessary. Right now a user could possibly execute malicious code on the server gaining access to sensitive data or sabotaging the server.

A suggested first solution to this was that for every active user we would create a container that only handles the execution of code. This theoretically makes it much harder for the user to tamper with the server directly, but this itself introduces several issues. Running a container for every student would mean a great computing cost due to its overhead.

To further ensure security we would need to sanitize the users submitted code. For example block imports of python packages such as "os" [29] which interact with the operating system directly. This means a greater workload in implementing and maintaining language support as well as opening up for human error and potentially removing functionality.

An alternative solution to these issues would be to completely eliminate server-side code execution and rely on a framework such as WebAssembly (WASM). WASM was considered at the beginning of the project but after some research and tests we found WASM to have varying support for interpreting code and importing packages and

libraries. WASM is a low-level binary format that runs code at near-native speed in web browsers. It is designed to be a compilation target for languages like C, Java, and Rust, allowing performance-heavy applications, like games or video editors, to run efficiently on the web. To remain as language agnostic as possible, we decided against it. However, when writing this, all the languages we decided to implement (Haskell, Python, Ruby) have their own interpreter implemented in WASM. Since WASM is rather new and in very active development one could imagine that most popular languages, especially the ones used in education, if not already supported will be in the near future.

After reflecting on these issues one might raise the question if WASM is not the better choice after all. This might be the case and is left as an exercise for the reader.

Test Handling

As mentioned in earlier chapters, we construct a "test" program from the users' submitted code where we run the solution with known variables and compare them to a stored result. This works rather well except for the cases when the users do something unexpected.

When extracting the function signature, we assume that the function defined first is the one which is intended to solve the task. If this is not the case, the test most likely fails. The same goes if the users writes a function with more arguments or arguments of other types than the task creator intended. To resolve this, one could force the user to comply with a set function signature and then base their solution around this. Using an LLM to create tests depending on what the user's function signature looked like was also explored but quickly abandoned due to the unpredictability of results and fear that we would give students incorrect feedback.

In some languages, we decided to handle the case when a function did not return any value but instead just printed it. For example, take a task where the teacher asks the student to create a function that takes their name and prints to the terminal: **"Hello, <name>!"**. In that case, the teacher explicitly tells the student to print and not return. When later constructing tests, we naively call the function inside a print statement and match our saved test data to this output. If the function already prints something, we get two outputs and our tests fail. To resolve this, as mentioned in Section 3.7, we identify if a function contains a return statement or not. If a return statement is present, we assume that the user returns the intended data and we nest the function call inside a print statement. If not, we assume the data is printed and we just do the function call.

5.3 Ethical Aspects

When working towards school environments and students, we need to take into account the ethical aspects. Especially, how is user data stored and used?

Choice of AI Model and Protecting Data

AI, or LLMs, are an integral part of the project's "hint" aspect and pose somewhat of a challenge. Many of the openly available APIs that are used to interface with an LLM cannot ensure that the students' data is not used in ways we did not intend. To resolve this issue, we were planning on hosting our own LLM using open-source models such as LLama 3.3 or Qwen 2.5. Doing it this way, we would gain control over not just the data but also all the features of the LLM. However, we did not have time to implement this. Instead, we are using OpenAI's API, which is something that has to be changed if this project were to be released.

Monitoring and Storing Sensitive User Data

A key challenge with monitoring student progress is balancing effectiveness with students' privacy. Tracking individual student performance allows teachers to provide targeted support, but excessive monitoring may make students uncomfortable, discouraging engagement. On the other hand, providing only general class trends preserves student privacy but reduces the tool's ability to be accurate and to pinpoint individual needs.

If students are aware that every action, such as keystrokes, test runs, and hint usage, is being tracked, they may feel surveilled rather than supported. This perception can be damaging to learning, particularly if the purpose of the monitoring is not clearly communicated. Due to this, there is also the risk that students may choose to copy code into external AI tools to obtain hints or solutions outside of the monitored environment.

To mitigate these issues, transparency and boundaries must be established. Students should be informed about what aspects of their activity are recorded and what the purpose of the data collection is. In addition, it may be beneficial to allow certain types of activity, such as running test cases, to remain unmonitored. By offering a sense of privacy during exploration and experimentation, the platform can maintain student engagement while still providing teachers with valuable insights they need to provide support.

The goal is to foster trust between the students and the system. Monitoring should be framed not as surveillance, but as a mechanism for support and guidance.

Is AI hints actually supporting learning

AI is going to be a big part of programming in the future. Therefore, it is better to actually use it in a controlled way, which uses the AI more as a guide than a solution. There is no way to prevent students from using third-party AI tools, so we believe it is better to include it directly in our platform to hopefully replace tools that students tend to be tempted to use. Even though our implementation intentionally gives less help, students can hopefully see the benefits in only getting hints rather than the whole solution to encourage learning.

5.4 Implementation Challenges

This section discusses the mistakes made during the development of the project, including what the mistakes were, why they occurred, and possible improvements to avoid them.

Authentication

The authentication method initially implemented had some backend functionality, such as password hashing and salting, handled within the Next.js framework, while other backend functionality, like storing users and sessions, was managed by the Java backend. It quickly became clear that this approach was unsustainable, as it would likely lead to issues in the future when adding new features or further developing the authentication system. A clear structure with a proper separation between backend and frontend components is preferable. As a result, the authentication system was refactored, and the backend functionality previously handled by Next.js was moved entirely to the Java backend.

This was not the most optimal solution because it introduced a lot of manual state management in the frontend. This created a lot of work implementing necessary functionality like route protection, session management, and more, all of which are included in a library like Auth.js or something similar that supports OAuth 2.0. Because we manually handled these things, our solution had some flaws and bugs regarding some edge cases that could be exploited. This also meant that the platform had less functionality like caching and support for third-party authentication methods which could be very relevant for the intended users of the platform.

Design and Implementation

Early in the project's development, work was prioritized toward implementing functionality such as AI hint generation and code execution to demonstrate core functionality during the half-time presentation. As a result, the development of foundational elements such as authentication, session management, and a proper structure for handling data like users, courses, and sections was delayed. This approach led to extensive use of mocked data, which introduced several issues later on in the project. Since many of these initial features were designed under time constraints and tailored mainly for the demonstration, they were not built to integrate well into the design of the entire system. The core problem was that our code was not based around the user information for whom our planned API was written. The data structure that we used had the user object as a starting point with fetch requests to display correct, relevant data on a page based on the authenticated user. Seeing as user authentication was implemented after other functionality mentioned above, information had to be mocked to feed the page components data to operate on. It could be viewed as implementing the design we had created backward, leading to unnecessary issues which could have been avoided.

We also had issues with the design itself. The first rendition of our data structure did not account for handling a task state, meaning that when a student had begun solving a task, there was no actual way to save the state of the solution so that the student could continue at a later time. This was then hastily implemented later on, which left a lot to desire in functionality.

Updating the design in this way also introduced some confusion regarding how requests and responses were structured across our platform. A good practice to prevent a lot of issues relating to the data model changes would be to implement a validation schema for both the frontend and the backend with type checking at compile time. This would allow for easier understanding of how requests should be structured and minimize bugs and troubleshooting during the development process. We had to update our data structure model three times during the development process to achieve the specified functionality, leading to issues that would be preventable with the above approach.

5.5 Future Work

During the project, several features and improvements were identified that could significantly improve the functionality and usability of the platform. However, due to time constraints and other limitations, these improvements were not implemented. These features are explained and discussed for future development below.

Automatically Generated Solution Examples

A useful future improvement would be the ability for teachers to automatically generate solution examples for the tasks they create. Currently, teachers need to manually write and input these examples themselves. This works well enough for now, but automating this process could save valuable time and streamline task creation.

Hosting a LLM

As mentioned in Section 5.3, we did not have time to host our own LLM on a server, and this is something that should be done if this project is to be released.

Input and Terminal Support

One current limitation of the program is its inability to handle user input. This means that tasks requiring terminal input, such as reading values from the user during execution, cannot be created or run within the platform. This is a significant drawback, as many beginner-friendly programming exercises rely on user input to teach core concepts like variables, loops, and data validation. Without support for input, the range of tasks available to students is limited, potentially hindering their understanding of how real-world programs interact with users. To address this issue, a terminal could be added to the interface. This terminal could appear

alongside the standard output window, allowing it to be available for the student in tasks that need it. Introducing a terminal would significantly expand the types of tasks the platform can support, making it more effective for teaching foundational programming skills.

AI Hint Verification

Currently we naively assume that the AI hints are factually correct. We found no reliable way to verify that the hints provided are correct and useful. We attempted to mitigate this by giving the LLM as much context and direction as possible. No hints are created out of thin air but are based on subgoals which in turn are created from a suggested solution and problem description. As mentioned in Section 4.3 based on our survey most students found the hints helpful. However, the tasks used for testing were quite simple. Being unable to verify and predetermine exactly what an LLM produces is an inherent characteristic that is difficult to circumvent. Having several LLMs produce hints and later using another LLM to "choose the best one" or filtering out responses that only one LLM produces and the others disagree with might lead to more reliable answers.

Submission Modes: Practice vs Final Submission

As of the current implementation, students are allowed to submit their solutions an unlimited number of times. While this approach lowers pressure and encourages experimentation, it blurs the line between formative (practice) and summative (final) assessments. A planned improvement is to implement two distinct submission modes: *practice mode* and *submission mode*.

In *practice mode*, students would be able to test and refine their code freely, perhaps with access to hints and without their work being formally evaluated. In contrast, *submission mode* would represent a final or graded attempt, where additional restrictions may apply, like limited hint access, no code editing after submission, etc. Introducing this distinction could encourage students to take ownership of their final submissions and give teachers a clearer picture of student progress. It could also address the concerns discussed in Section 5.3 regarding student trust and transparency around monitoring during task solving. By introducing these distinct modes, the platform can clearly communicate when and how student activity is being tracked. *Practice mode* would allow for greater freedom and encourage experimentation, with minimal or no monitoring. In contrast, the *submission mode* would signal that the solution is being formally evaluated, and therefore prepares the student to be more deliberate with their solution and aware that more of the activity is being monitored.

Preventing Use of Restricted Language Features

An important consideration in task design is the ability to restrict access to certain features or functions of the language. For example, in a task that asks the student to reverse a list manually, the use of built-in functions such as `reverse()` would technically solve the task but undermine the educational value of the exercise. In its

current state, the platform does not enforce any constraints on the functions that can be used, leaving such instructions entirely to teachers. A task solved with built-in functions would pass through the system as being completed, so to catch this, a teacher would have to manually review the submitted attempt and potentially unmark it as "completed".

6

Conclusion

The purpose of this project was to create a user-friendly platform which would facilitate teaching of programming. Our platform aims to reduce the stress that a teacher might go through trying to reach out to every student with personalized assistance. The platform also aims to lower the technical barriers that make it difficult to learn about programming, while making sure that every student has access to all the assistance they need.

The results show that we have been able to implement most of the functionality that we aimed to do and we were able to fulfill the purpose through a few key features. The option to get AI-generated hints is one of our most impactful features, that reduces a lot of the stress that a teacher might go through since it enables the student to get personally tailored hints without having to call for the teacher's assistance. Another key feature in our platform is that it runs on the browser, which does not require any downloads. This heavily reduces the technical barriers and obstacles that students might encounter, reducing the stress of teachers troubleshooting and assisting students. The overview page is another necessary feature, especially for the teacher, since it gathers and displays the most important information and data into one page, which saves time and makes our platform easy to use.

In conclusion, we have created a platform that aims to solve the modern problems of programming for educational purposes. The platform supports both teachers and their students and allows for an easy teaching experience with a well-built foundation that allows for future improvements.

Bibliography

[1] Anthony Hughes. "The Software Developer Is Dead: Long Live The Software Developer". In: *Forbes* (2023). URL: https://www.forbes.com/councils/forbestechcouncil/2023/03/29/the-software-developer-is-dead-long-live-the-software-developer/.

[2] Sarita Singh. "Identifying Learning Challenges faced by Novice/Beginner Computer Programming Students: An Action Research Approach". In: *CEUR-WS* (2022).

[3] Mary Ellen Flannery. "Class Sizes: A Growing Issue Among Educators". In: *newToday* (2023).

[4] Ben Moore. "'I massively regret using AI to cheat at uni'". In: *BBC* (2024). URL: https://www.bbc.com/news/articles/cz04emrxp4xo.

[5] Tora Heidvall. "GU-student misstänks ha fuskat med AI – stängs av". In: *Sveriges Radio P4 Göteborg* (2024).

[6] MIT Media Lab. *Scratch*. 2024. URL: https://scratch.mit.edu/.

[7] Khan Academy. *Khan Academy*. 2025. URL: https://www.khanacademy.org/.

[8] JetBrains. *JetBrains Academy*. 2024. URL: https://www.jetbrains.com/academy/.

[9] CodeCombat Inc. *CodeCombat*. 2024. URL: https://codecombat.com/.

[10] *Replit*. URL: https://replit.com/~.

[11] W3Schools. *W3Schools*. 2024. URL: https://www.w3schools.com/.

[12] MIT Media Lab. *Scratch About*. URL: https://scratch.mit.edu/about.

[13] Dominic Lohr, Hieke Keuning, and Natalie Keisler. "You're (Not) My Type – Can LLMs Generate Feedback of Specific Types for Introductory Programming Tasks?" In: *arXiv preprint arXiv:2412.03516* (2024). URL: https://arxiv.org/pdf/2412.03516.

[14] Meta. *React*. URL: https://react.dev/.

[15] Vercel. *routing*. URL: https://nextjs.org/docs/app/building-your-application/routing.

[16] Vercel. *Nested routing*. URL: https://nextjs.org/docs/app/getting-started/layouts-and-pages.

[17] Vercel. *Dynamic routing*. URL: https://nextjs.org/docs/app/building-your-application/routing/dynamic-routes.

[18] Meta. *Hooks*. URL: https://react.dev/reference/react/hooks.

[19] Mozilla. *JavaScript*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript.
[20] Microsoft. *TypeScript*. URL: https://www.typescriptlang.org/.
[21] *React-Bootstrap*. 2025. URL: https://react-bootstrap.netlify.app/.
[22] Bootstrap team. *Bootstrap*. URL: https://getbootstrap.com/.
[23] geeksforgeeks. *SpringBootGeeks*. URL: https://www.geeksforgeeks.org/spring-boot/.
[24] mongodb. *mongodb*. URL: https://www.mongodb.com/company/what-is-mongodb.
[25] Joe McKendrick. "DevOps efforts stumble over database obstacles, survey suggests". In: *ZDNET* (2019). URL: https://www.zdnet.com/article/databases-slowing-down-application-release-cycles-survey-says/.
[26] mongodb. *mongodbAtlas*. URL: https://www.mongodb.com/docs/atlas/.
[27] Mozilla. *HTTP request method*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods.
[28] Martin Velez et al. "Student Adoption and Perceptions of a Web Integrated Development Environment An Experience Report". In: *ACM Technical Symposium on Computer Science Education (SIGCSE '20)* (2020). DOI: 10.1145/3328778.3366949.
[29] Matthijs Labots. *os-sys*. URL: https://pypi.org/project/os-sys/.

A

Appendix 1

A.1 A Better Programming Teaching Tool - Enkät

Tack för att du tar dig tid att fylla i denna enkät!

OBS! Alla dina svar är anonyma och vi samlar inte in några personliga uppgifter.

Vi önskar att du fullföljer några steg och svarar på tillhörande frågor så ärligt som möjligt. Vi tar gärna emot alla typer av feedback, både positiv och negativ. Det viktigaste för oss är att veta varför du tycker som du tycker. Så om du har positiv feedback, berätta gärna vad du tyckte om och varför. Om du är kritisk, kom gärna med konstruktiv kritik så vi kan förbättra produkten.

Smidigast blir det om du har uppe denna enkät i en flik och plattformen i en annan och växlar mellan flikarna allt eftersom du stegar dig igenom uppgifterna. Enkäten är uppdelad i 5 delar, som är uppdelade i först en uppgiftsbeskrivning, och sedan följer frågor angående den precis genomförde uppgiften.

* Anger obligatorisk fråga

Del 1 - *Om dig*

1. Vilket ämne på ditt program tycker du är mest intressant? *
   (välj det som stämmer bäst)

   - ○ Programmering
   - ○ Matematik
   - ○ Teknik
   - ○ Fysik
   - ○ Kemi
   - ○ Övrigt: ____

2. Hur mycket programmering har du gjort tidigare i skolan? *
   (välj det som stämmer bäst)

- ∘ Detta är min första programmeringskurs
- ∘ Jag har läst 1 kurs i programmering tidigare utöver denna
- ∘ Jag har läst 2 eller fler kurser i programmering utöver denna

3. Hur mycket programmerar du på fritiden utanför skolan? *
(välj det som stämmer bäst)

   - ∘ Jag programmerar ingenting på fritiden
   - ∘ Jag programmerar lite grann på fritiden
   - ∘ Jag programmerar mycket på fritiden

4. Hur skulle du själv uppskatta dina programmeringskunskaper? *

   (Väldigt nybörjare) ∘ 1 ∘ 2 ∘ 3 ∘ 4 ∘ 5 (Väldigt säker/avancerad)

Del 2 - *Logga in*

Öppna nu programmet i en ny flik/fönster om du inte redan gjort det.

1. Börja med att logga in som den användare du blivit tilldelad och fyll sedan i samma mail här: * ____

2. Det var enkelt att logga in *

   (Håller inte med) ∘ 1 ∘ 2 ∘ 3 ∘ 4 ∘ 5 (Håller helt med)

3. Efter du blev inloggad hamnade du på en "Startsida". Vad är ditt första intryck? * ____

Del 3 - *Navigation och att hitta uppgifter*

På plattformen finns två kurser som din användare är registrerad på.

- Den ena innehåller två uppgifter - en i varje sektion
- Den andra innehåller en uppgift i en sektion

Din uppgift är först att navigera dig genom hemsidan och hitta alla tre uppgifterna.

1. Kommentera gärna vad som var tydligt/otydligt med att hitta uppgifterna ____

2. Det var enkelt att hitta de tre uppgifterna i de olika kurserna *

   (Håller inte med) ∘ 1 ∘ 2 ∘ 3 ∘ 4 ∘ 5 (Håller helt med)

Del 4 - *Arbeta med uppgifterna*

När du hittat uppgifterna, är det dags att börja försöka lösa dem. Det viktiga är inte att du löser alla uppgifter utan att du utforskar hur plattformen fungerar.

Tänk särskilt på:

- Vilken hjälp du får längs vägen
- Hur det känns att testa din lösning
- Vilka knappar och funktioner som finns och var de är placerade
- Hur det känns att skicka in en lösning

Tips! Testa gärna allt som verkar kunna hjälpa dig. Det gör inget om du inte behöver all hjälp - vi vill veta hur funktionerna funkar i praktiken.

Del 4 - *1. Addition*

Navigera dig till och öppna upp **1. Addition** - Introduktion till Programmering med Ruby, *Del 1.* Kom ihåg att testa funktioner du stöter på längs vägen.

1. Det var lätt att förstå uppgiften *

   (Håller inte med) ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 (Håller helt med)

2. Jag använde AI-hints *

   ○ Ja
   ○ Nej

3. Var AI-hintsen hjälpsamma och relevanta för uppgiften? Har du några andra tankar om AI-hint funktionen? _____

4. AI-hints var hjälpsamma och relevanta för uppgiften

   (Håller inte med) ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 (Håller helt med)

5. Var det någon AI-hint som stack ut som extra bra eller dålig? Klistra gärna in den här _____

6. Uppstod något fel eller något annat du vill kommentera? _____

7. Jag har klickat "Submit" och lämnat in min lösning*

   ○ Ja

Del 4 - *2. Fib*

Navigera dig till och öppna upp **2. Fib** - Problemlösning med Programmering, *Del 1.* Kom ihåg att testa funktioner du stöter på längs vägen.

1. Det var lätt att förstå uppgiften *

   (Håller inte med) ∘ 1 ∘ 2 ∘ 3 ∘ 4 ∘ 5 (Håller helt med)

2. Jag använde AI-hints *

   ∘ Ja
   ∘ Nej

3. Var AI-hintsen hjälpsamma och relevanta för uppgiften? Har du några andra tankar om AI-hint funktionen? _____

4. AI-hints var hjälpsamma och relevanta för uppgiften

   (Håller inte med) ∘ 1 ∘ 2 ∘ 3 ∘ 4 ∘ 5 (Håller helt med)

5. Var det någon AI-hint som stack ut som extra bra eller dålig? Klistra gärna in den här _____

6. Uppstod något fel eller något annat du vill kommentera? _____

7. Jag har klickat "Submit" och lämnat in min lösning*

   ∘ Ja

Del 4 - *3. FizzBuzz*

Navigera dig till och öppna upp **3. FizzBuzz** - Problemlösning med Programmering, *Del 2.* Kom ihåg att testa funktioner du stöter på längs vägen.

1. Det var lätt att förstå uppgiften *

   (Håller inte med) ∘ 1 ∘ 2 ∘ 3 ∘ 4 ∘ 5 (Håller helt med)

2. Jag använde AI-hints *

   ∘ Ja
   ∘ Nej

3. Var AI-hintsen hjälpsamma och relevanta för uppgiften? Har du några andra tankar om AI-hint funktionen? _____

4. AI-hints var hjälpsamma och relevanta för uppgiften

   (Håller inte med) ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 (Håller helt med)

5. Var det någon AI-hint som stack ut som extra bra eller dålig? Klistra gärna in den här _____

6. Uppstod något fel eller något annat du vill kommentera? _____

7. Jag har klickat "Submit" och lämnat in min lösning*

   ○ Ja

Del 4 - *Allmänt om uppgiftslösning*

1. Körde/exekverade du din kod? *

   ○ Ja
   ○ Nej
   ○ Va? Kunde man köra koden?

2. Det var lätt att köra koden

   (Håller inte med) ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 (Håller helt med)

3. Fanns det något du gillade eller inte gillade med att köra koden? _____

4. Körde du tester? *

   ○ Ja
   ○ Nej
   ○ Va? Kunde man köra tester?

5. Var det tydligt ifall du klarade av uppgiften eller inte?

   (Håller inte med) ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 (Håller helt med)

6. Det var lätt att köra tester

   (Håller inte med) ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 (Håller helt med)

7. Fanns det något du gillade eller inte gillade med att köra tester? _____

Del 4 - *Allmänt om uppgiftslösning*

1. Har du några övriga kommentarer eller synpunkter angående uppgiftslösning? _____

Del 5 - *Översiktssidan*

Här följer frågor om översiktsidan

1. Kunde du hitta din personliga översikt? *
   - ○ Ja
   - ○ Nej
2. Jag förstod informationen som visades på översiktssidan *

   (Håller inte med) ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 (Håller helt med)
3. Det jag såg på översikten kändes meningsfullt och användbart *

   (Håller inte med) ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 (Håller helt med)
4. Fanns det något du gillade eller inte gillade med översiktssidan? _____

Del 5 - *Sammanfattande reflektion*

1. Vad var det bästa med plattformen? * _____
2. Vad saknade eller kändes otydligt? * _____
3. Skulle du vilja använda plattformen i dina studier? Varför / Varför inte? * _____

Tack för din medverkan!

B

Appendix 2

B.1 AI hint generating prompts

Del 1 - *Prompt for generating subgoals*

You are a teaching assistant for students learning programming for the first time. From a problem description and a suggested solution for a problem, return subgoals for solving the problem. For a specific subgoal that needs to be accomplished, further explain or breakdown of what this subgoal entails.

Return a list of subgoals that outline the steps required to implement the suggested solution.

Del 2 - *Prompt for generating AI hints*

You are a teaching assistant for students learning programming for the first time. Give only one single hint instead of a list of multiple hints. From a set of subgoals, a suggested solution and an attempted solution, create step-by-step hints for solving the problem based on the attempted solution and try to guide them toward the suggested solution. Be pedagogical as possible and do not reveal the final solution and also make sure to not reveal the suggested solution as the student does not have access to it. The student also does not have access to the subgoals as they are only used for you to make step by step hints. Make sure to use and refer to the students current code in their userCode. Give hints that are directly relevant to their attempted solution and write their own solution out when you are refering to it so that they understand easily. The tests are also included but do not reveal them to the student.

C

Appendix 3

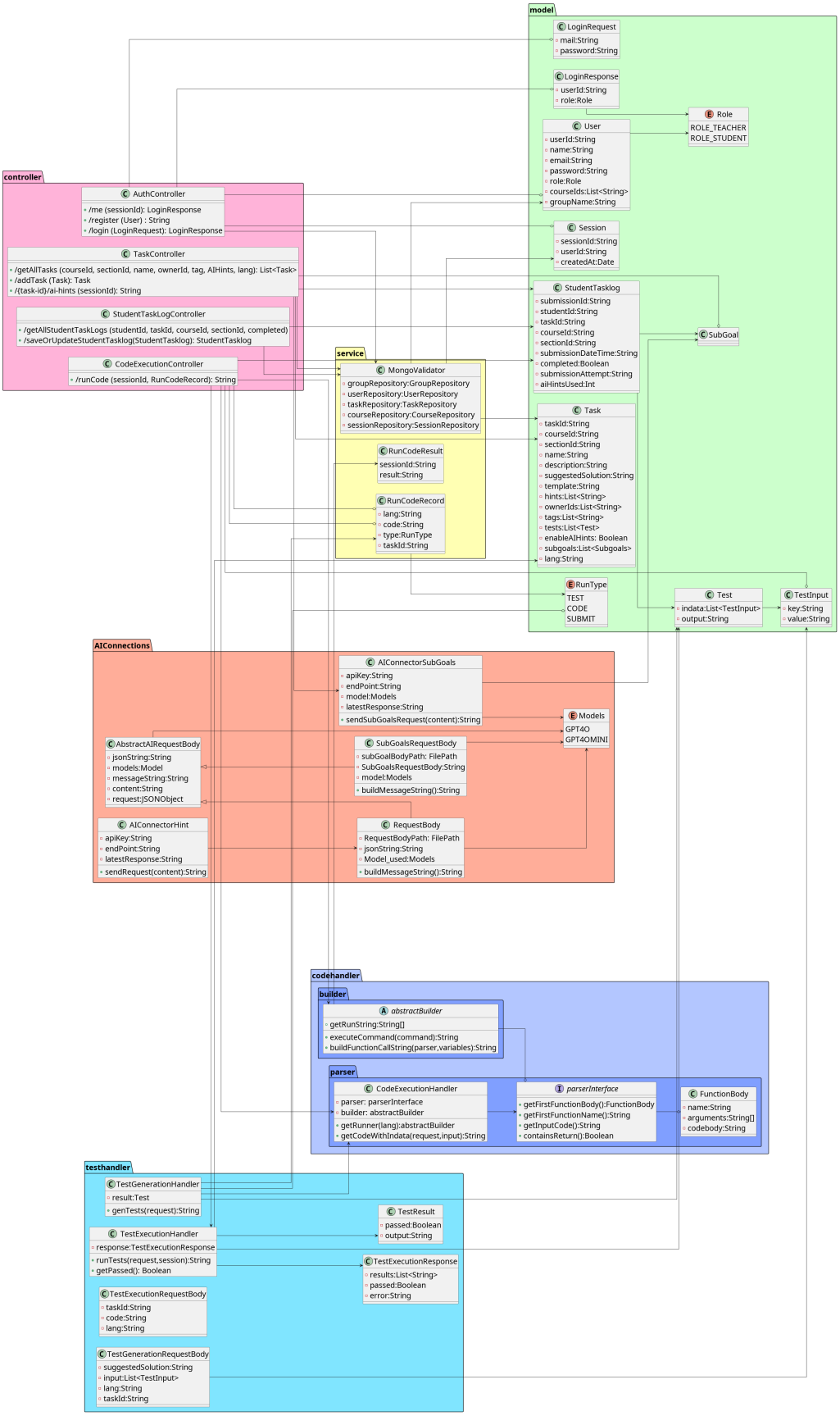

Figure C.1: Class diagram describing key functionality and dependencies in the backend.