Automated grading and Plagiarism Checker

Group 12

April 23, 2022

Abstract

Due to the increasing number of students in the program TCS, teachers need a tool to help automate the process of grading assignments. This suggests the need for our system - ”Ammit”. Our project aims to deliver a global and scalable design for such a tool that would both test correctness of students’ code and check all submissions for plagiarism. The tool will also equalize the hardware students’ code is ran on, allowing code efficiency to be evaluated, along with correctness. This report elaborates on the design choices and the architectural design of the system. It goes in depth into the development process, from start to finish. In the future, we hope this system can be deployed by different courses within the TCS program.

Contents

Chapter 1

Introduction

Due to logistical problems arising from the various programming assignments in the curriculum of TCS and the increasing number of students, grading assignments has become challenging for teachers. Our project aims to address this issue by developing a system to automatically grade programming assignments and additionally integrate a plagiarism checker. We aim to develop a high-accuracy tool for auto-grading as well as a user interface that allows students to upload their code and view their results. There are two representative stakeholders that are most closely related to our project - teachers and students. The goal from teacher's point of view is to reduce the time and resources needed to grade assignments without creating a difference in the outcome dependant on whether this tool is used. From the student's perspective, the product should allow them to view their own grade and feedback through interface of the system.

In Chapter 2, the domain of the proposed system is identified. In Chapter 3, the method of requirement elicitation is illustrated and user stories are listed. In Chapter 4, the research regarding plagiarism checking and sandboxing is discussed to set the stage for a good design. In Chapter 5, the project proposal of the system is described with requirements and mock-up proposal. The reflection on initial design and planning are also explained. In Chapter 6, the overall global design of the application is discussed with design decisions, architectural design, stakeholder analysis and design of the plagiarism checker. In Chapter 7, the design that concerns with the interaction of end user is discussed, detailed use cases are given and design for front end is elaborated. In Chapter 8, the implementation of the system is explained. The specific implementations of the Angular frontend, Spring Boot backend, plagiarism checker and Canvas integration are illustrated. In Chapter 9, the approach of testing of the system with testing plans, strategy and schedule are described. The result of testing is also given at the end. In Chapter 10, future plans to address the shortcomings of the current system are discussed. Finally, in Chapter 11, an analysis of measures taken to improve security of the auto grading system is provided.

Chapter 2

Domain Analysis

This chapter is concerned with identifying the domain of the proposed system. By delving into the subtleties of the domain, we can ensure that the solutions we provide can more effectively be of service to the clients. It also enables faster development and allows us to make a more adaptable system, due to deepened understanding of the domain.

2.1 Introduction to the Domain

The domain is "Automated Assignment Grading". The motivation behind the domain analysis is to understand the intrinsic need for the system. This leads to a more a scalable design and a solution that is closer to what is desired. Due to the growing number of students in the study program each year, our clients want to automate the process of grading assignments, without impacting the integrity of the students' results. Automation is the use of technology to reduce human intervention in a given process. Grading is generally a simple process that requires the repetitive task of checking the given answer and comparing its similarity to the correct answer. In the context of programming assignments, that is done by giving the program input and inspecting the generated output. This makes grading an adequate process for automation, since removing human intervention is simple and effective.

2.2 General Knowledge of the Domain

Broadly, grading of this scale is rarely done by a single person. Multiple teachers or teaching assistants are involved in the task and it still takes a considerable amount of time. We are developing this system to automate the grading process and it should be able to be scaled to many different courses. Understanding the differences between these courses is important if we want the software to be useful to different teachers.

2.3 Clients and interested parties

The system is being developed for the clients directly and they stand to benefit from it by saving time on grading. The system will allow a single teacher would be able to easily and quickly grade all the students from a single course, eliminating the need for other people to be involved in the grading process and making it drawn out. The students benefit from having to wait less for their grades, and, should the system be successfully deployed, it would give students the benefit of multiple submissions of their code, allowing them to improve their code and know whether they will pass.

The Canvas Integration team and the IT department of the university (LISA) are parties interested in the development of the system, due to respectively their involvement in the meetings or the maintenance of the system. Please refer to section 6.3 for all stakeholders within this domain.

2.4 Development Environment

During meetings with our clients we established that they have no preference concerning what frameworks and programming languages we use for the software. Hence we chose to use frameworks that were listed under Tools & Frameworks in the Development Guidelines for Canvas Plugins, so that maintenance of the software is straight forward. The back-end is written in Java with the Spring Boot framework and front-end is written with the Angular framework. The database we're using is H2database, which is a development database used in Java. More detailed information on the frameworks we used and why we chose them can be found in chapter 8.

2.5 Tasks and procedures currently performed

Traditionally, grading programming assignments is a very labor intensive task. A teacher has the option to grade the assignments themselves, or to get help from teaching assistants that all grade some amount of assignments, but these traditional methods require a lot of resources for the purposes of grading.
In module 7, the project on graph isomorphism currently holds a delivery session in which students are given instances one by one and have a limited amount of time to solve each one. Their grade is based on how many instances they solve, some of which are necessary for a passing grade. This allows all the students to

be graded in a relatively small time frame, but comes with some limitations. Namely, it is hard to organize and it has the added downside of potentially inflating the grade of students that are running their code on superior hardware or vice versa.

2.6 Competing software

We have identified two existing systems that fit our client's requirements at least partially: Github Classroom and CodeGrade. The former of the two has a problem with privacy policy that doesn't align with our university's requirements, thus cannot be used by our clients. CodeGrade on the other hands shouldn't cause any privacy issues for the university, however it is very expensive, including a fee per assignment and per student, while still requiring a lot of adjustment in workflow on creation of assignments.
By researching the existing software we have concluded that our system can be indeed useful to the university. We have also expanded our knowledge of the domain and moved forward with our design more concretely, having the competing software as a frame of reference.

2.7 Similarities across existing UT Software

Since the system is integrated directly into Canvas, we have designed it such that the user experience is consistent with what can be expected from Canvas. Our users are plenty familiar with that software, ergo working with our software shouldn't be challenging for them.

Chapter 3

Requirements Specification

This section covers our requirements elicitation process and the resulting requirements are given as user stories and system requirements. The requirements are given according to the MoSCoW principle and further separated into functional and non-functional requirements

3.1 Requirements Elicitation

The requirements elicitation process started with the first meeting where the client explained the desired system. After the meeting we formalised the requirements into user stories. We presented these user stories during the next meeting and used the feedback to improve them and add any missed requirements.

3.2 User Stories

We provide all user requirements in the form of user stories from the perspective of teachers and students. The requirements are prioritised according to the MoSCoW principle.

3.2.1 Must

- As a teacher, I want to create programming assignments, so that student can submit their solutions to be graded automatically.
- As a teacher, I want to choose the programming language of the assignment.
- As a teacher, I want to provide test instances (input and output), so that student solutions can be checked for correctness and speed.
- As a teacher, I want to specify which test instances are required for a passing grade, so that assignments can be graded automatically.
- As a teacher, I want to view all submissions and results from all students, so that I can judge the overall progress.
- As a teacher, I want to see which submissions/students are suspected of plagiarism, so that fraud can be prevented without checking all submissions.
- As a student, I want to submit my solution for grading, so that I can get fast feedback and see if my solution needs to be improved.
- As a student, I want to resubmit my solution, so that I can improve my grade.
- As a student, I want to see the results of all past submissions with a breakdown per test instance.
- As a student, I want the submissions to be shared for everyone in my group, so that working together is easier.

3.2.2 Should

- As a teacher, I want to set a deadline and a starting time for the assignment.
- As a teacher, I want to edit the assignment after it is created.

3.2.3 Could

- As a teacher, I want to create Python Notebook assignments, so that students do not need to use an IDE.
- As a teacher, I want to provide feedback to students, so that they can improve their solution.
- As a student, I want to download my solutions.

3.2.4 Won't

- As a student, I want to see leaderboards and my standing, so that I am motivated to improve my solution.

3.3 Functional Requirements

User stories describe how end users will interact with the system, but for developing the system concrete functional requirements are needed. The above user stories have been used translated into concrete functionalities the system needs to offer. The requirements are still prioritised according to the MoSCoW principle.

3.3.1 Must

- Assignments can be created. An assignment has a name and a programming language.
- Assignments can be done in python or other programming languages. These assignments have any amount of test instances.
- The test instances have an input (file) and an output (string)
- Instances can be flagged as mandatory - required to pass for a passing grade.
- The system provides an overview of all submissions and overall progress statistics for the teacher.
- The system can execute a plagiarism check and display similarity scores to the teacher.
- Students can upload solutions.
- The system tests the uploaded solution and shows the results to the student.
- Students can see, per instance, if the test passed, timed-out or threw an error.
- Students can resubmit solutions to improve their grade. The best submission is counted.
- The system shows students an overview of all past submissions.
- The system treats a student group as a single user.

3.3.2 Should

- Assignments have a start date and a deadline set by the teacher. Solutions can only be submitted in this period.
- Assignments can be edited after creation. The dates can be moved and test instances added/changed/removed.

3.3.3 Could

- Assignments can be done in a python notebook. Testing the solution takes place in the notebook.
- Students can receive feedback on their solution. Either automatically based on their results or manually by the teacher.
- Submissions can be downloaded.

3.3.4 Won't

- The system maintains a leaderboard of best solutions and students can see their position in the ranking.

3.4 Non-functional Requirements

We have also identified several non-functional requirements. These are general criteria by which the system can be judged rather than specific functions it must perform.

- Every submission should eventually be executed and tested.
- Groups must not have access to submissions of other groups.
- There must be a limit on how long a submission is allowed to run.
- Students should see the results of their submission within 10 minutes if there are no other queued submissions.
- Shared code where only variable/function names have been changed must be detected as plagiarism.
- Student code run on the server should under no circumstance impede the normal running of the server.
- Plagiarism checking should take into account code that is copied from websites
- Plagiarism checking should take into account code that is autogenerated by technology like Copilot

Chapter 4

Research

In this chapter there will be findings and discussion on research into various topics that are encompassed within our application, namely advanced plagiarism checking and sandboxing considerations. Research sets the stage for a good design. The findings will be used to justify some of the hard design decisions.

4.1 Plagiarism checking

4.1.1 checking for inter-student plagiarism

Since plagiarism checking is integral to the functioning of the application, there needed to be done research into how and what to incorporate into the application. Considering the effort that needs to be taken to create a properly working system that does correct analyzing on this front, it was natural to opt for a ready made solution. A common solution to the problem of plagiarism checking, is that a teacher uses the free service offered by the American university Stanford, called MOSS. The teacher would then use a submission script to hand in a batch of student files. A drawback of using MOSS is that it is hosted on an external server, where the report is also stored. The link of the report is normally the output of the scripts accessing the MOSS server. This raises some interesting questions about the storage of student files on an external server, not hosted in the country. Looking into alternatives, there are quite some tools that specifically are able to run on our server, unlike MOSS. Looking at a comparative study of multiple of those tools [1], we can identify some interesting options. CCFinderX (CCFX) seems a good alternative to MOSS, given that it used in the right way. Unlike some other tools, it does need fine tuned parameters to function optimally. When used with the default settings, CCFX has rather low accuracy, as was found in [1]. The authors of [1] do also in their paper describe a setting which manages to get high precision and recall both, which seems promising.

Like MOSS it is also token based processing of the text Other often found methods include line based, and abstract syntax tree (AST) based text processing. Line based is generally a bit less effective, while AST based processing has rather bad (O(3)) time complexity [2]. The reason line based or direct matching is generally less effective, is that often the code plagiarism is not copied verbatim, rather there is usually some obfuscation of the source happening. This is rather easily achievable by hand or even free programs that do source code obfuscation for you such as proguard. Using an AST based checker bypasses that problem by instead looking whether the control flow of the program is equal to (part of) another program. The study [1] found that a token based tool such as CCFX can have decent accuracy in these cases as well, however.

4.1.2 checking for code reused from the web

In non programming related academic misconduct, the material that's plagiarized from is either other students, which we are able to match against in this case, or academic papers and articles. The latter is often stored in one of few academic databases. This does make it easier to query against, since the databases are made for that purpose, and contain relevant material to query against. Whereas in the case of code plagiarism it be from many different websites Since there are many great articles and answers to questions to be found online, it is a natural place to look for solutions to your own problems. Students as such often use of websites like StackOverflow, where answers are abundant. This presents an interesting challenge. Where token or AST based plagiarism detectors can find obfuscated code, this is harder when there is also a websearch to be done for this. Websites do not provide you with an AST of their program, nor is the content structured in such a way that it is always easy to programmatically parse for code within blocks of text. There has been some effort to do semantic websearches to get around this problem, by using techniques like fuzzy string matching. Article [3] gives an example of how to use a popular fuzzy string matching library called fuzzywuzzy, which uses Levenshtein distance to compare similarity between text, and [1] provides results on the accuracy of this method.

4.1.3 Checking for auto-generated code

Due to the increasing availability of good text generators, particularly with the advent of Transformers, which are state of the art at the moment of writing, it has become harder to verify whether text, or code, has been authentically written by a human [4]. A good example of this is the Transformer that goes by the name GTP-3 [5], or the more recent Copilot [6], which is specifically made for auto-generating computer code. This poses some interesting issues for educational programs. This means that it would be beneficial to find a method to detect AI-generated work. It may in the future be possible to do this by looking for specific signatures of AI in the text. Currently there is much work going into finding such signatures for different kinds of Neural Nets, like Generative adversarial networks (GAN)s. GANs are particularly troublesome because they are particularly good at generating so called "deepfake" picture and video material. Yet in the case of GANs there oftentimes still exist subtle differences between the work of the neural net and the authentic work, which makes it possible to be detected [7] [8]. There is at least a small amount of work been done on detecting whether text is faked by one of the GPT models [9] [10] [11], but not into code specifically, as far as we could find. As such it is hard to give any insight into the ease with which this will be able to be detected in the future.

4.2 Sandboxing

Since the code is student written, this means we should not trust the code to work properly at best, and be malicious (for research purposes of course) at worst. Ideally we want to run the student code on the server, however. This is important because all student submissions then get run on equal hardware, and as such the timed assignments will for instance have an equal base. Since we do not want to break the server our code is working on, and ideally also do not want to break our own application, there is need to separate the untrusted code and sandbox it within our app. There is a few ways to go about sandboxing code. Firstly, we could remove certain imports and inbuilt functions from the language of choice before execution. This is however quite hard to do exhaustively, since there may always be more functions that have some kind of way to segfault or other vulnerabilities. Then there's the option to go for a different version of the intended language, for python this means that there's a few different python implementations, and at least one of them, PyPy [12], is created to allow restrictions in the interpreting of code, most notably with a module called "sandbox". Pypy sandbox however does state on their website that the software is not meant for production at this moment. Then there is also the option to do sandboxing at the operating systems level, and spawn the process less privileged and/or deny access to the operating systems standard library, of which Docker containers [13], LXD containers [14] and jails like firejail [15] are good examples. All of the aforementioned options are based on linux cgroups [16] and namespaces. These technologies are often used to for instance create sandboxed environments for programming or hacking competitions.

In addition to sandboxing the running student code, it may also be wise to run the entire server within a virtual machine, to ensure that our application will not harm the other services running on the server. This could be achieved by running Kubernetes [17] on the server and hosting our server within a Docker container, for instance. It may be the case that this is necessary for the production version of the application, but not yet in scope for this design project. The project will be set up in such a way to allow for future exploration of this however. In this case it would still be wise to nest another level of virtualization within this container, to prevent the student code from interacting with the rest of the server.

Another interesting option would have been to offload the running of student code to yet another server, the JupyterHub [18] server. This would in effect sandbox students code too, since the architecture is such that every student get their own "server" running in a virtual machine, which means it could crash and only impact their own system. Servers can be easily restarted on that platform as well. Reason it was not immediately pursued any further was the clients reluctance to put this server under the load of an entire module's worth of students, considering he server has only limited capacity. So while this avenue sounds promising, it may be only worthwhile in the future after it has been scaled up to sufficient capacity.

In the future it is certainly possible that the sandboxer will be swapped out for an entirely different implementation. Since for maintainability and extension of the program we want to abstract running of the process, we introduce a surface that can be called which makes it possible to modularly swap out the concrete implementation of the sandboxer.

Chapter 5

System Proposal

In this chapter, the project proposal, mock-up proposal and requirement proposal that we delivered to our clients and how their feedback was incorporated into the final product are discussed. There is also a section with our reflection on the initial design and the initial planning we had on the project.

5.1 Initial Design Draft

In the early stages of the project, when we only knew the rough outlines of what we were supposed to do, we made a design draft, which included a high-level class diagram of the entire system, along with a list of basic requirements listed as either student or teacher requirements. Since the clients knew that the project has very few concrete requirements, they expected us to actively build on the design ourselves, and they were happy with our initial work.
One of the clients gave us feedback on the idea to execute the code on the server, which sparked a discussion concerning client-side versus server-side execution. This turned out to be a very important design decision upon which we elaborate further in section 6.1.1.

5.2 Project Proposal

Our project proposal contained the following sections: Project description, Project organization, Planned deliverables, Planning, Risk Analysis and Task Division. It is included in Appendix B. When we presented the project proposal to our clients, we received some useful feedback from our clients, which included correcting our organization to be "agile-like", rather than "scrum-like" and also changing the Risk Analysis section, which we had misinterpreted. However, we were glad to find out that our clients were on the same page as us on the overall structure of the project and could continue developing the design with confidence.

5.3 Requirements Proposal

At the start, the project was very abstract, thus we had a lot of freedom as to how we approach the requirements specification. We interviewed our clients and held brainstorming sessions to create a Requirements Proposal. These requirements served as guidelines for the development phase of the project, however due to the agile structure of our project new requirements emerged throughout the project, which were also discussed with the clients.

5.4 Mock-up Proposal

With the requirements in mind, we wanted to begin development. Beforehand, we wanted to discuss the general look and workflow of the system with the clients. We created low fidelity prototypes for the student view and the teacher view. When we presented them to our clients we got the confirmation we wanted from our clients to begin development of the project. It allowed us to confidently make practical progress on the project.

5.5 Reflection on initial design and system proposals

Making design drafts and discussing proposals with our clients were very conducive to the overall development of the design project. Incorporating our clients feedback allowed us to proceed with confidence, and gave our clients control over the development, to ensure that everything is going as intended. Despite starting out as an abstract concept, through the system proposals, we managed to make the project into a concrete list of requirements with a stable prototype.

Chapter 6

Global Design

In this chapter the overall design of the application will be discussed. Important discussions will be incorporated about the kind of crucial discussions that had to be made; an analysis of the relevant stakeholders will be had; then some components will be treated in a high level design overview. In depth design decisions about specific functionality will be deferred to a later chapter.

6.1 Design Decisions

In this section there will be outlined which specific choices have been made regarding implementation of the application.

6.1.1 Scale of the system

The system requested by our clients needs to support assignments where code can be executed not only on the server but also on the student's laptop. These requirements were too conflicting to be implemented within the time available and therefore we focused on server-side execution because this would work better for the module 7 project. For this reason the client-side execution requirement is categorized as "could". The system could be extended to support Python Notebook assignments - test instances would be tied to API endpoints where the notebook would send and answer and the server would respond with whether it was correct or not.

Language and frameworks choices

Firstly it is important to choose the correct language and tooling to write a successful application. There are a few factors regarding this which are important to note:

- what is fitting in the context
- what is maintainable
- what is the team familiar with

Contextually, during the initial domain analysis, it was found that the UT, or LISA in particular, mostly works with only a few specific frameworks. As such they are familiar with these, which means it will be easier to maintain the application, which makes it viable for long term deployment.

In this case the developers at LISA mostly make use of a combination of the Java web-framework called Spring Boot, and the front end JavaScript library Angular. As such these were a natural first choice, since the team was familiar with Java already, and to a certain extend also with JavaScript, albeit not so much with Angular in particular. However, considering this is the standard, the choice was made to set the application up for longevity.

When working with Spring Boot there is some flexibility regarding choice in databases. For the ease of use, the framework H2 was chosen, which is a common and well supported choice for development in Spring Boot.

6.2 Architectural design for the application

In this section there will be an overview of the architecture for the entire application. We will additionally go over a few higher level design choices that had to be made up front, before further designing the application.

A choice was made to go for a few specific frameworks and libraries as outlined earlier. This resulted in a high level outline of the system which is described in figure 6.1.

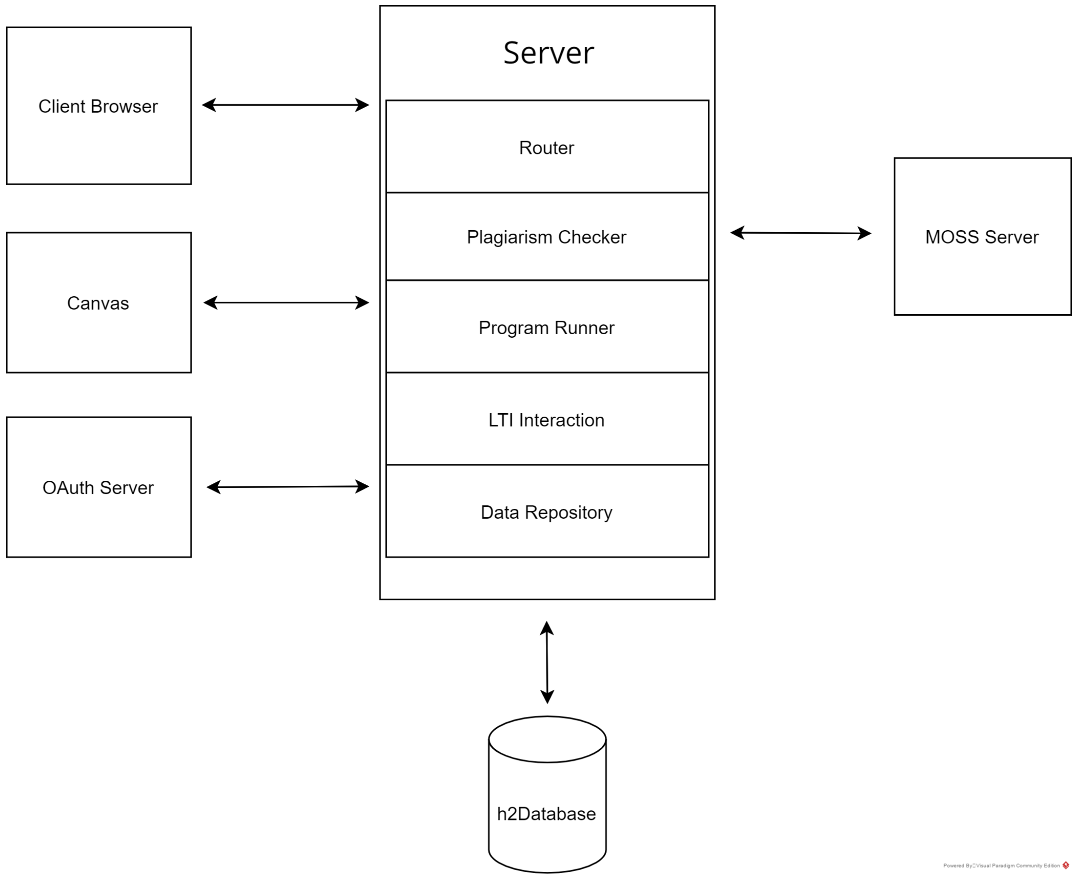


Figure 6.1: Simplified block diagram for the system

6.2.1 General principle of design in Spring Boot

Working with Spring Boot, there are a few general design principles that come with that choice. Spring Boot uses techniques such as "dependency injection" and "inversion of control" which make the design more scalable partly because they are for instance easier to test. For this application it was chosen to stick to these principles and create a design that in line with Spring-Boot applications.

6.2.2 Client- versus server-side execution

During the requirement discovery phase we found that there were two seemingly conflicting requirements. One of the clients wanted the systems to run student code on the client side, and the other wanted it to be run on the server-side. The reason for this difference was mainly that the code that should run on the client-side was resource intensive, since for their use case a machine learning model needed to be trained by students. The reason there was also the requirement to run code on the server side, was to aid fair grading, since for that specific use case, the run-time aspect of the code was crucial to grading. Since not all students have access to the same hardware, this would create unfair advantages for some, while disadvantaging others.

Client-side execution within the checker itself however seemingly presented some difficulties. Those mainly arose from the fact that it is harder to assess whether the grading part of the code is done appropriately, or it is tampered with to create artificially higher scoring outputs. The client however assured us that this part of the process should be handled by the teacher creating the assignment, and not necessarily by our design. Nevertheless, it is beneficial if the design takes into account possibilities to make this process easier.

6.2.3 LTI vs Canvas API vs standalone application

When considering the implementation of the systems backend, three possibilities were discovered and considered from a convenience, feasibility of implementation within the allotted time frame, and long term benefit perspective. The first possibility considered was creating a standalone application or independent web site, which would be available completely separately from the client's current learning management system (LMS) of Canvas. This type of application was considered to be the easiest to implement, as it would not require specific communication with the LMS. However, not being able to communicate with Canvas would limit the functionality of this application in a crucial manner by not allowing for pulling or posting grades and submissions, and not having access to student or instructor user information.

As communication with Canvas was considered a crucial requirement, this application was planned to be implemented as an external tool within the Canvas website. For these types of tools, there were two options for implementation - utilising the Canvas API or the more global IMS LTI standard. The Canvas API allows for quick integration with the current system and provides all functionality required, but opens a risk for the tool to not work if the client swaps to a different LMS in the future. On the other hand, LTI is designed and built for working with multiple LMS's, allowing this application to continue working with minor modifications even if Canvas is replaced in the future. The only downside found for this type of design was the limitation of specific functionality only provided by the Canvas API such as downloading of assignments directly from Canvas. However, this functionality can be integrated in the future for specific LMS's such as Canvas in this case.

As the design requested was for a global and long term plan, the choice was made to utilise LTI for the communication between the backend and Canvas. Since both students and teachers already use Canvas to hand in assignments and enter grades respectively, the choice was made to design and integrate this tool as a plugin that runs within the LMS itself. This allows for convenient access to the grading tool and allows this system to utilise the authentication system of Canvas itself to identify and authorise users and their access to Canvas.

6.2.4 High level design

The design that our clients agreed on for the application is shown in Figure 6.2. We have conformed with this design in our implementation of the prototype. However, we are missing the "Course" entity, due to time constraints in setting up the Canvas communication. However, further adding it to the current structure of the prototype should not be difficult, given the current structure of the project.
The design is simple, which allows it to be flexible, which was a requirement our clients insisted on, because they wanted the product to be used by potentially multiple different courses. Due to the simplicity however, some constraints as to what assignments can be graded are introduced. Since the grading is achieved by looking at the output and comparing it to a given expected output, the teachers need to adjust the assignment to have a linear and well-defined structure for output on given input. This means that some types of assignments cannot be graded using this tool, for example, any assignments that need to handle input and output.

6.3 Stakeholder analysis

- Teachers, the main stakeholder, will use the software repeatedly over time.
- TAs, may have to deal with questions and support for the software
- Students, will use the software to hand in their assignments, and view how they did.
- Previous students, when potentially it is found out they shared their code with current students.
- Programme management, potentially deciding to use the software, handling issues with the software.

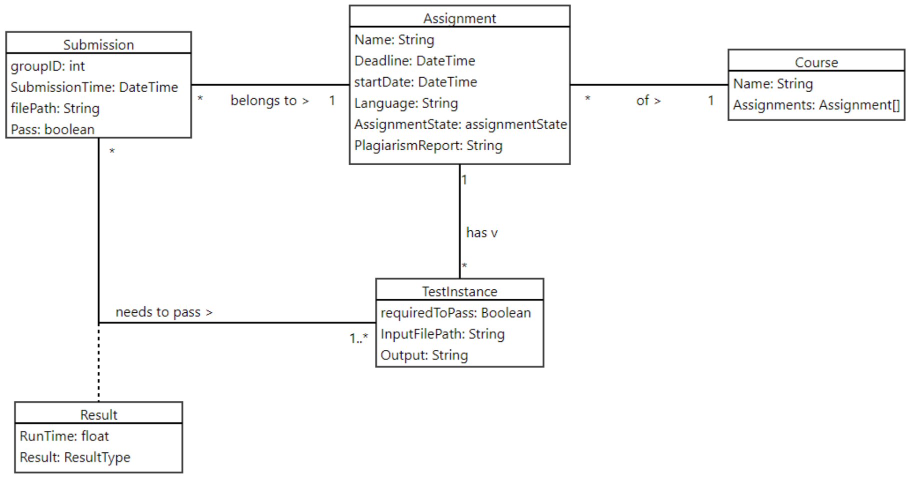


Figure 6.2: This class diagram offers a high-level, simplified view of the design

- Study advisors, handling issues with grading, student's feelings about automated grading and plagiarism detection. May have to be in contact with students and teachers at the same time to resolve issues concerning the application and it's function.
- Competition, may be influenced by design or try to see whether our application didn't infringe on their concept. They may see changes in the customerbase.
- Stanford (Moss server), the load we place on their servers may negatively impact them.
- UT LISA, they may be handling future maintenance and further development. Potentially will also see negatives if load on the servers will be high due to running of instances of submissions.
- BOZ, holding a small stake since this is concerning grading.
- Examination board, they may be influenced by the plagiarism checker since they would be involved in the process of fraud suspicions

By analyzing the possible stakeholder of the auto grading system, we built a stakeholder onion model that displays a relationship of stakeholders. Stakeholders are placed in the onion model according to their relativity and importance within the system. The stakeholders that are most closely related to the system and most affected by it are located close to the center of the onion model. Figure 6.3 indicates the stakeholder onion diagram of the auto grading system.

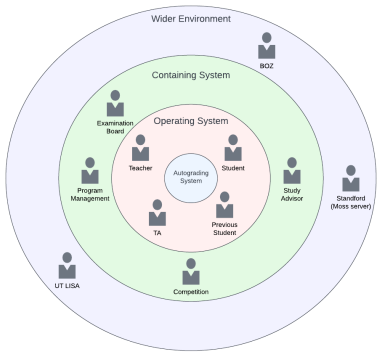


Figure 6.3: Stakeholder onion diagram

We built the onion model in 4 layers, 1) core system, 2) operating system, 3) containing system and 4) wider environment. The core system is the auto grading system in this case. The stakeholders that are in the second layer which is the operating system represent the stakeholders who interact directly with the system. The third layer is called 'containing system' and it contains stakeholders who may not have direct interaction with the system but get affected from it. The fourth layer, wider environment, is populated with stakeholders who are not involved in the main operations or maintenance of the system but still have an influence.

6.4 Design of the plagiarism checker

The plagiarism checker is one of the core functionalities the system must have. This means it also makes sense to divert a little extra effort to make sure that it's design has been set up in a scalable and maintainable manner. This section goes through how this design approach.

The plagiarism checker should be able to get files submitted to it, be prompted to run, and prompted to return a result. This is mainly implemented by having the abstraction of a plagiarism checking service, which connects with a router to accept calls from the user's side, and on the other side it spawns an instance of whatever plagiarism checker is chosen. This instantiated plagiarism checker is given the files for a selected assignment, which are then checked, and a report is finally returned that can retrieved.

The plagiarism checker has been designed in such a manner that it is easy to abstract some functionality that any plagiarism checker could have, and create specific instances to one depending on your need. A practical example of this would be, all the plagiarism checkers should provide some report when they are done running, and as such we can abstract this functionality.

Since many different courses could be running at any given moment within the application, we do choose to implement the client to the plagiarism checker in such a way that multiple can be spawned. This ensures that multiple requests could be handled simultaneously, which is not a problem for the Moss server for instance, so it makes sense to not limit ourselves either.

The only drawback of this approach may be that there may be multiple requests to check the same data, and the requests themselves are rather costly, due to the overhead of uploading many files, and then sending them over the internet in their entirety.

Multithreading may be problematic in terms of file safety however. If files get modified at the same time as they are being sent, then this will give an undefined result. This again could be handled by saving state about the assignment current status.

When the assignment has been closed and is not already sent to the checker, files may be modified, otherwise when already there has been a request for checking the files, then they must remain untouched until the operation has been completed. We can show the states of the assignment like: ¡insert nice diagram going through the states of an assignment¿.

Chapter 7

User Design

In this chapter, the part of the design that concerns interaction of the end user with the system will be explained. First, use cases are given which explain how and why users will interact with the system and then the resulting frontend design is presented.

7.1 Use case Diagram

There are two main users who are closely involved in the system, students and teachers. The figure 7.1 below demonstrates the use case diagram of the auto-grading system.

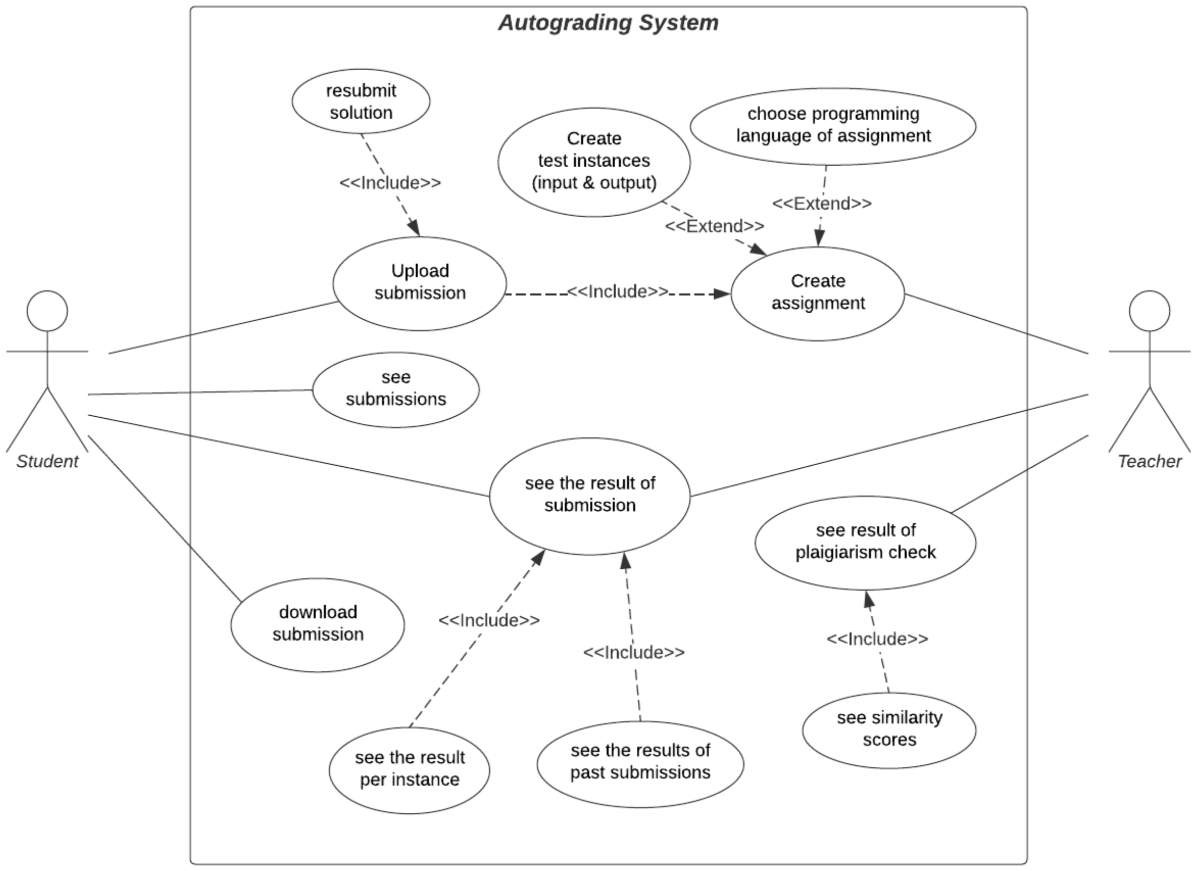


Figure 7.1: Use case Diagram for the system

The specific description of the use case diagram will be explained through expanded use case descriptions. We selected important use cases and described them in a expanded versions below. Table 7.1 to Table 7.5 are all about expanded use case descriptions.

Use case :	Upload submission
Actors :	Student
Goal :	Upload the improved solution
Flow of events :	
Actor action	System response
1 The student chooses an assignment 3 The student uploads a file	2 Displays past submissions, results 4 Displays success message 5 Reloads submissions, including new one

Table 7.1: Uploading a solution

Use case :	See the result of submission
Actors :	Student
Goal :	To view the result of solution
Flow of events :	
Actor action	System response
1 The student chooses an assignment 3 The student chooses a submission	2 Displays past submissions 4 Displays result with breakdown per test

Table 7.2: Checking submission results

Use case :	See the result of groups
Actors :	Teacher
Goal :	To gauge the overall progress
Flow of events :	
Actor action	System response
1 The teacher chooses an assignment 3 The teacher chooses a group	2 Displays the best result per group 4 Displays all submissions of group

Table 7.3: Check results as the teacher

Use case :	Create the assignment
Actors :	Teacher
Goal :	To create an assignment for students
Flow of events :	
Actor action	System response
1 Choose programming language 2 Set deadline 3 Create test instances	4 Displays the assignment for students

Table 7.4: Create an assignment

Use case :	See the result of plagiarism checker
Actors :	Teacher
Goal :	To see if plagiarism is detected
Flow of events :	
Actor action	System response
1 Start plagiarism checking process 3 Check the results of the checker	2 Send submissions to plagiarism checker 4 Displays the similar files

Table 7.5: Check for plagiarism

7.2 Frontend design

This section will cover the user interface of our plugin and how the different views implement the use cases we have identified.

7.2.1 Teacher views

Teachers have access to four different views in AMMIT. They can see an overview of all assignments, create new assignments, see submissions for a specific assignment and see the results of the plagiarism checker. All teacher views are shown in appendix A.

Overview of all assignments

This view is shown in A.1 and it is the entrypoint into our tool for teachers. It shows all assignments in the system with their start time and deadline. From here a teacher can select an assignment to go to the view for submissions or click on "Create new assignment" to go to the form for creating a new assignment. A color palette is used to make the view more interesting, while fitting into the Canvas theme.

Assignment creation form

Teachers can use this form, shown in A.4, to create new assignments. The teachers starts by filling in some basic information: name, start time and deadline of the assignment. They choose one of the supported languages. Files that are provided to students and are necessary for executing their code should be uploaded, for example libraries, files containing utility functions. Then teachers can add any amount of test instances to the assignment. A test instance has five parameters:

- **Input** - given as a single file of any type.
- **Output** - a single line of text.
- **Time limit** - how long the program is allowed to run the test before being timed-out.
- **Score** - how many points students get if they pass this instance.
- **Required for passing** - Students pass the assignment if all required instances are passed.

Single assignment view

At the top of the page (shown in A.2) are the controls for plagiarism checking - the teacher can start the checker and go to the result page once it is done. There is also a count for how many groups have passed the assignment which is the fastest way for a teacher to judge the overall progress. Below the teacher can see the best submission from each group and select a single group for a more detailed overview. The more detailed overview on the right side lets the teacher view the members, all past submissions and results of the group. The teacher can also download each submission. The table of best results lets a teacher easily see which test instances students find the hardest.

Plagiarism checker results

Once the plagiarism checking has finished, teachers can go to the result page which can be seen in A.3. This page shows suspicious submissions - pairs of submissions which have a high similarity score. The teacher can then investigate the suspicious submissions by downloading them from the previous page and comparing them manually.

7.2.2 Student views

Students have access to two different views in AMMIT. They can see an overview of all assignments that teachers have created, submit solutions and see the results of each submission. All student views are shown in appendix A.

Overview of all assignments

The overview of all assignments for students is shown in A.5, the only difference with the teacher view is the lack of the button for creating new assignments. Students can see all assignments in the system with their start time and deadline. Student can select the appropriate assignment to submit their solutions.

Solution Upload and history

Students can use this page, shown in A.7, to upload their solutions. On the right side of the page, there is a button that lets student browse their files. When the student chooses a solution to upload and clicks the upload button, the success message is displayed which can be seen in A.6. The submissions is also shown on the left but likely has not been graded yet.
After the system has finished auto-grading the submission, students can see the results of the solution on the left side. It displays the time the student uploaded the file, indicates if the solution is a ”pass” or a ”fail” and shows the total score of the solution.
By expanding a submission, the student sees a breakdown per test instance and they can also download their submission. For each test instance the student sees if their program produced a correct or incorrect output, if it took too long and was timed-out or if it threw an exception. All of the different options are color-coded for visual clarity. The runtime of each test is also given. If it contains incorrect answer, the system will display a ’wrong result’ with a red box. If the time set by the teacher is exceeded, it will be marked as ’time out’ with an orange box. If the result is correct and not timed out, it shows ’Pass’ with a green box. Student can also see how long each test instance takes to run. This breakdown is shown in A.8.

Chapter 8

System Design

In this chapter, technical design implementation will be laid out and explained. The chapter begins with the frontend implementation in Angular, followed by the backend, written in Java with Spring Boot. Then the plagiarism checker is covered and lastly integration with canvas using LTI is explained.

8.1 Angular frontend

The frontend of AMMIT has been developed in Angular. Angular applications are broken down into components - in this application each component implements a single view. There are also several services for communicating to the backend and classes/interfaces that represent the same objects on the backend.
Routing in the application is done with a RouterModule - each component is connected to a route, specified in app-routing.module.ts. Routes are protected with AuthGuard - it gets the user's role from the backend using the roles service and restricts access to pages based on it. It also redirects teachers and students to their respective assignment selection screen when the plugin is first loaded.

8.1.1 Components

Since the components are a one-to-one match with the views, there are four teacher components and two student components. The views from a user perspective are described in 7.2, this section will provide a more technical overview. First the student components are explained, then the teacher.

Assignment selection (assignment-selection)

When initialized, this page uses the assignment service to load all assignments. A card is created for each assignment using the ngFor Angular directive. Each assignment also gets a semi-random color - the color is based on the assignment ID and therefore wont change between sessions. The date service is used for formatting the start and end times. When a student clicks an assignment they are routed to the submission page and the assignment id is passed as a query parameter.

Submission page (st-project)

The page loads all submissions of the student's group when the page is opened and whenever a submission is uploaded. The submission fetching is handled by the submission service while the upload is handled by the st-upload service. The assignment id that was passed from the previous page is passed on to the backend to connect the submission to an assignment.

Teacher assignment selection (assignment-selection-teacher)

The only difference to the student assignment selection component 8.1.1 is the button that leads to the page for creating new assignments via a router link.

New assignment page (new-assignment)

This page contains a form for filling in assignment details and adding test instances. The form fields are bound to fields of the Assignment and Instance classes. The files need to be handled separately to be sent to the spring boot backend - each file is added as a form data field and the json-stringified assignment is added as another form data field. This is handled by the assignment service. Test instances are pushed and popped from a list as they are added and removed.

Best submission page (groups)

The page fetches the best submissions of each group using the submission service and a count of how many of them are passing is shown on the top. When a submission is selected it loads all submissions of that group and displays them the same as in the student page. Current members of the group are also fetched from the backend using the roles service and their names are shown. Plagiarism checking can be started which uses the moss service.

Plagiarism checker results (moss-results)

If plagiarism checking has finished, the results page loads the html received from the moss server via our backend and displays it using the innerHTML property. The html needs to be loaded in the backend because Canvas uses https while moss is only available over http and loading http from https is forbidden in some browsers.

8.2 Spring Boot backend

Our backend is built in Java using the Spring Boot framework and based on the Oxford Spring Security LTI 1.3 library. Spring makes it easy to develop web service applications. The building blocks of a Spring application are entities, repositories, services and controllers which we will explain in more detail in the next sections.

8.2.1 Entities

Entities in spring are objects we want to store in our system and in the database. The system has four entities: assignment, submission, test instance, result. The relationships between these entities are shown in figure 8.1. All of these entities are also tables in the database. This is implemented using the @Table annotation which automatically turns all of their fields into columns and creates additional tables for managing the relationships between the entities.

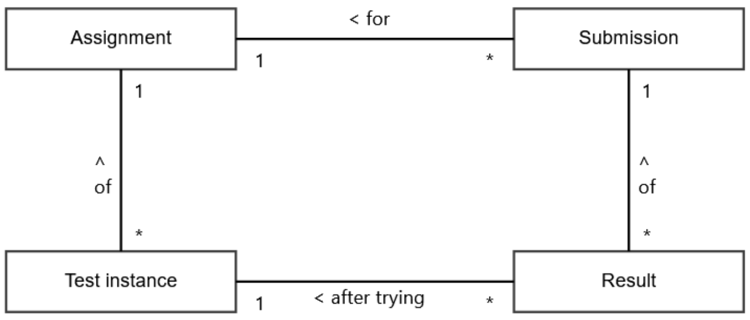


Figure 8.1: Class diagram for entities.

8.2.2 Repositories

Repositories are interfaces that provide methods for interfacing with the database. For example, making the assignment repository provides methods for getting all assignments from the database or deleting assignments etc. without needing to write any boilerplate code. There is a repository for each entity.

8.2.3 Services

Assignment, submission and result services offer another layer of abstraction from the database. They offer more specific methods for retrieving information, for example, getting all submissions of an assignment. They use the repositories explained above to interface with the database. The plagiarism checking service is different since it is not related to an entity or repository. It provides methods for interfacing with the plagiarism checker - starting the checking process and getting the results. The submission testing service is the core of the grading process. The grading process is explained in more detail in 8.2.5. Other classes can use the service to add submissions to the queue.

8.2.4 Controllers

Controllers provide the API endpoints used by the frontend to communicate to the server:

- **AssignmentController** - getting all assignments, creating/getting/deleting a single assignment.
- **MossController** - initiating plagiarism checking, getting results either as a URL or as an HTML ready to be embedded into the page.
- **SubmissionsController** - uploading and downloading submissions, getting all or only the best submissions of an assignment and all submissions of a group.
- **UserController** - getting the role of the current user and getting members of a group.

8.2.5 Grading

The SubmissionTestingService has a grading thread that is always running, taking submissions from the queue and testing them. Testing a submission starts with getting the language of the assignments. The command for executing the submission depends on the language. Then for each test instance defined in the assignment, a new thread is started where the program is executed with the test input as a command line argument. This thread is allowed to run for the time specified in the test instance, if it has not finished by then, it is terminated. After finishing or being terminated, the result is stored (correct, incorrect, timed-out, runtime exception). Once all tests are finished, the overall submission result is determined by checking if all required tests passed and then the results are saved in the database after which they can be seen by the group and teacher. To provide consistent results, all submissions are run consecutively to prevent longer execution times due to sharing the server with other submissions. All test instances are also run one-by-one for the same reason. For fairness, groups are only allowed to have one submission in the queue at a time - if there is already a submission in the queue when a new one is uploaded, the older one is dropped from the queue and will not be graded. The states a submission object goes through in the grading process are shown in figure 8.2.

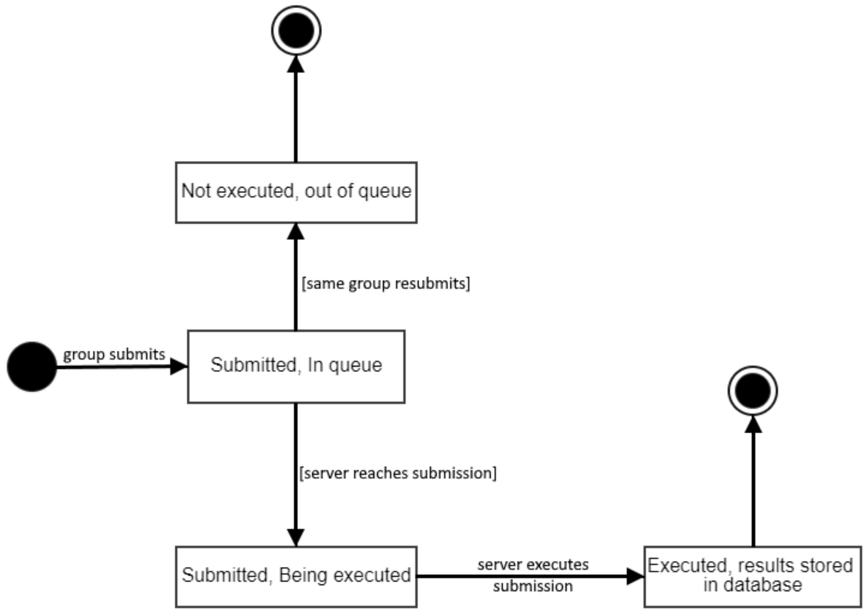

Not executed, out of queue
[same group resubmits]
group submits
Submitted, In queue
[server reaches submission]
Submitted, Being executed
server executes submission
Executed, results stored in database

Figure 8.2: State machine diagram for submissions.

8.3 Plagiarism checker system design

8.3.1 Introduction to the plagiarism checker system

As stated before, the aim when designing the plagiarism checking back-end was to provide enough abstraction to allow it that the actual plagiarism checking engine could be swapped out when for instance requirements change. For this first version of the plagiarism checking system, Moss may provide enough functionality to get started, but in the future a plagiarism checker may be subbed in or even ran in parallel to the current engine. To facilitate this we create the abstraction of a Plagiarism checker interface. This interface should naturally try to capture all of the desired functionality that one could want from a plagiarism checker. This means it should provide abstraction for functionality such as uploading files, uploading potential base files, specifying a language for the programming assignments, running the checker on the provided files, etc..

8.3.2 System setup

In 8.3 there is a diagram showing the overall class layout of the plagiarism checking system. Some smaller (convenience) methods were omitted from the classes. In the diagram the connections between the various classes and interfaces is visible. The Router will route incoming requests to the PlagiarismCheckingService. The PlagiarismCheckingService has access to the files and submissions via the other services; using the plagiarismChecker interface it accesses the methods available for plagiarism checking.

The currently accessible plagiarism checker implementation is the MossClient. This MosClient handles its end to perform the necessary tasks to check the submissions for plagiarism. Using Java sockets it establishes a connection with the Moss Server at Stanford. Files that are submitted are then uploaded over this connection, and checked for plagiarism on the server. The server will return a URL, which can be used to view the results of the plagiarism check.

The MossClient can also retrieve the html page at that URL mentioned before. The functionality could be further extended to parse the html, using a library called JSOUP, which is also used for retrieving the html, for instance.

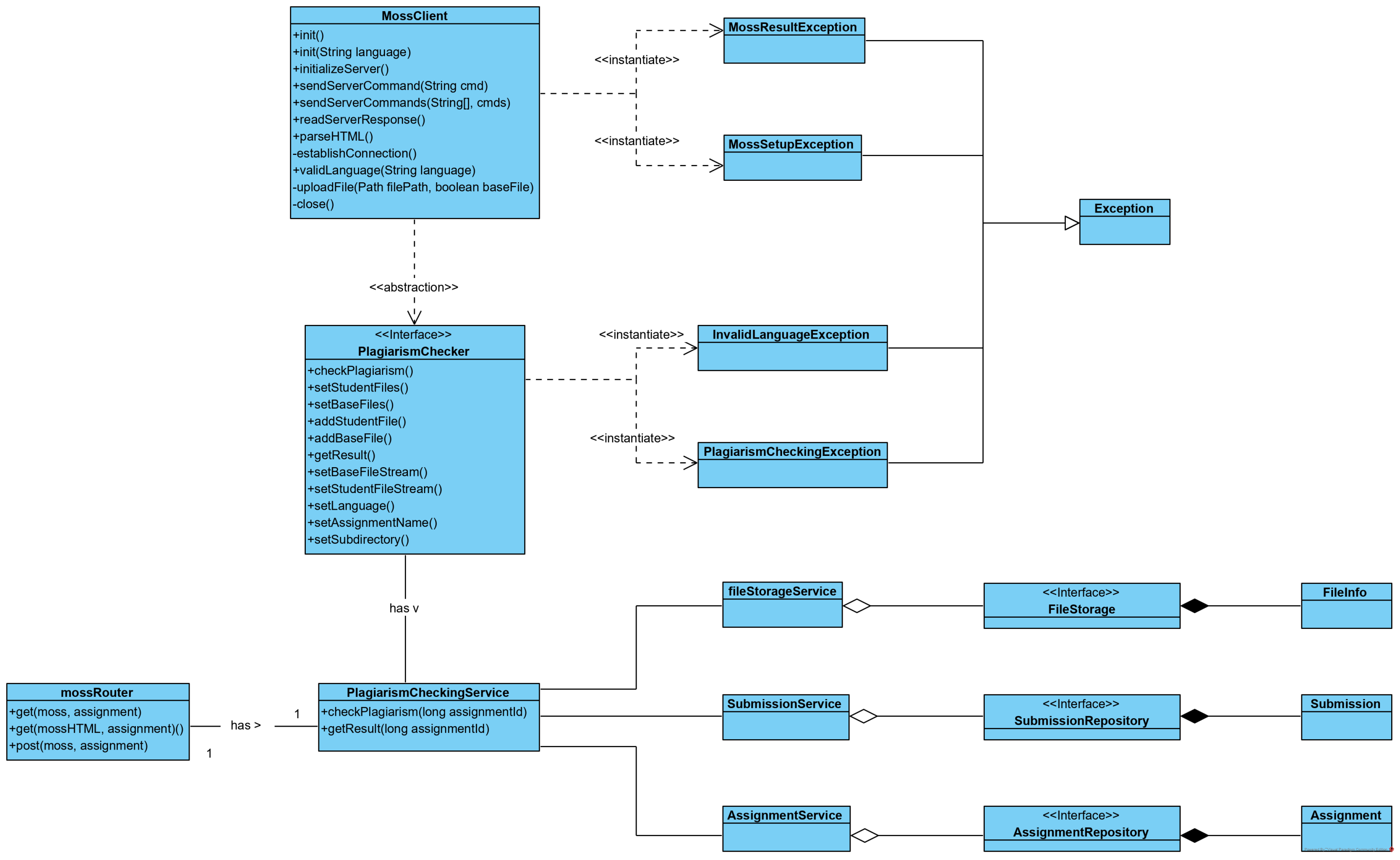


Figure 8.3: Class diagram for the current plagiarism checking system

8.3.3 Communication flow

The flow of communication between the client and server(s) can be seen in diagram 8.4. On the clientside there will be a request to check plagiarism for the assignment (actions 1-3 in the diagram). Then the system upon receiving the request will invoke the moss plagiarism checker, or whatever plagiarism checker is specified on the back end, through the plagiarismCheckingService. That plagiarism checking service will grant the plagiarism checker access to the files to process them.

In case there is another request before the next request is done to restart the plagiarism checker, the plagiarismCheckingService will check whether there is already a run going on. If this is the case, then a message will be returned to the client that this is the case, so the user can be displayed a message to wait until this run has finished, this interaction can be seen in actions 18 through 29.

When the client requests to see the results and the execution has finished, then the result will be passed back to the client. Before then when the execution finished the result was stored in the database on the server, so it can be retrieved easily.

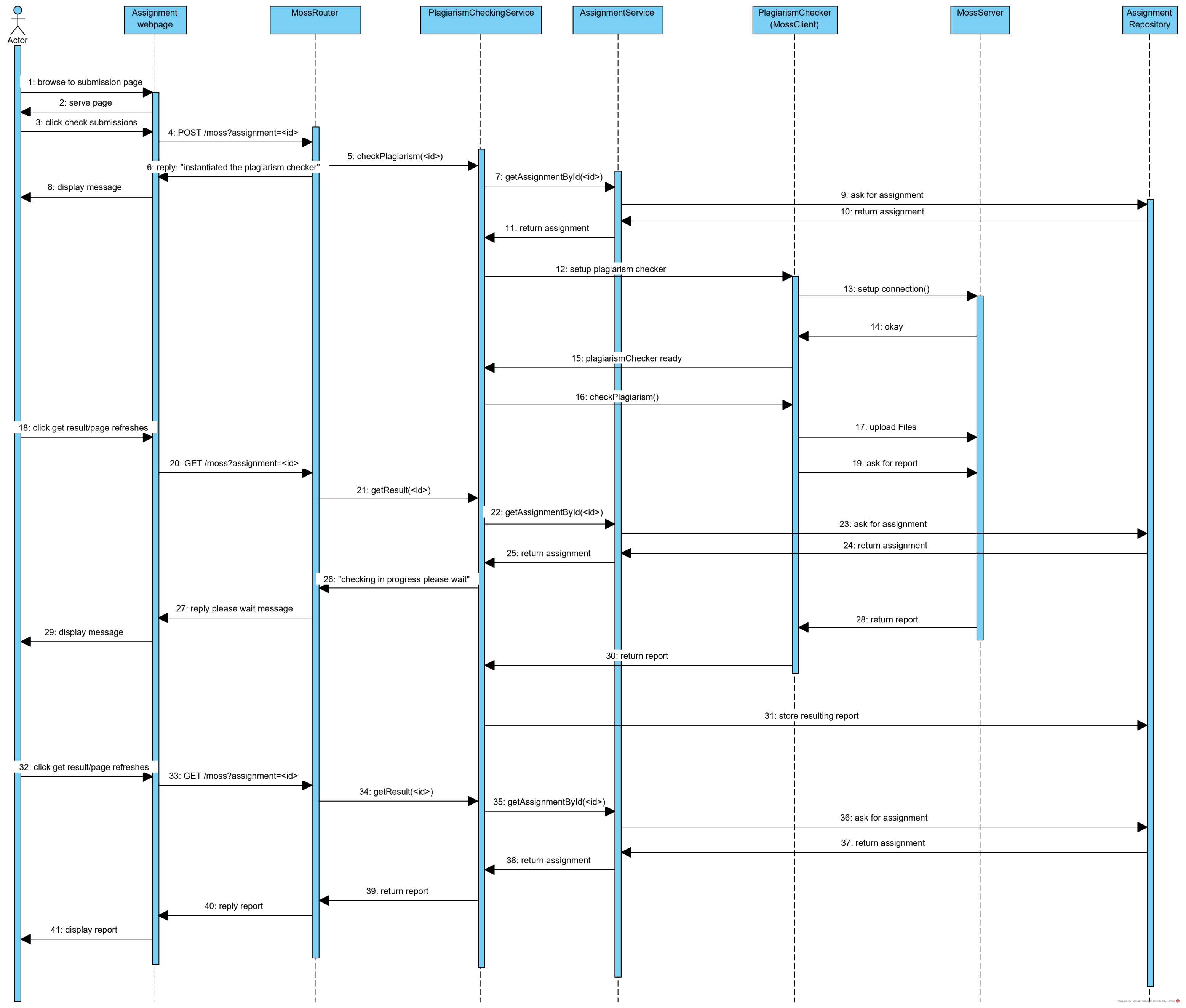


Figure 8.4: Sequence diagram for the plagiarism checking process

8.4 LTI 1.3 Backend

For the backend, the plan was to have a long-term and sustainable system built. For this purpose, the decision was made to utilise the IMS Learning Tools Interoperability framework [19], which provides for clear and rich integration between learning platforms and tools. Using the Oxford Spring Security LTI 1.3 library [20] provided a good starting point for development as it included integration of LTI 1.3 and Spring Boot. Although implementation of this was the most complex out of the three options - which also included a stand-alone application and directly utilising the Canvas API - it had the best long term benefits. Where this framework truly shines is in its capability to be integrated with multiple LMS's, meaning that a possible change in the LMS used by the Client would not make this design obsolete but allows it to still work after simple modifications to data request formats. In the following sections the authentication process, data request, and framework specific advantages will be expanded on.

8.4.1 LTI 1.3 Resource Requests

This application needed to communicate and work with Canvas to provide access to data regarding groups and the assignments of the course that the plugin is used in. To facilitate this, LTI provides a protocol flow for authorization and resource access [21]. This flow can be separated to three stages consisting of obtaining authorization, obtaining access information, and finally accessing resources. This process will be explained in the context of this application in the following subsections using Figure 8.5 from the IMS Security Framework documentation.

Obtaining Authorization

The steps A and B shown in Figure 8.5 is a setup step performed before the tool is deployed on the server. OAuth 2.0 is an authentication specification developed by the IETF OAuth Working Group. It is considered the industry-standard protocol for authentication, providing a specific authentication flow for web and desktop applications, as well as mobile phones and living room devices [22]. LTI 1.3 uses OpenID Connect, which is an identity layer that builds on top of the OAuth2 protocol [23]. Using this protocol enables third-party applications to make requests as a user, without the need to get their password again. To allow for this tool to request access tokens for each user of the plugin, the full OAuth2 token request workflow needed to be implemented [24]. This workflow uses a set client ID and client secret and a developer key set up within the Canvas course itself for setting up the authorization of the tool itself. The result of this setup is an authorization grant which can be used to authenticate users and allow it to request the authorization server, or Canvas, for access tokens when users launch the tool.

Obtaining Access Information

Following the retrieval of an authorization grant for the tool itself, access tokens could be requested by the plugin once users launch the tool. This token retrieval process, described by steps C and D in Figure 8.5, was an important part of the resource request flow as both students and teachers needed to have separate views and functionality. It allows the application, or a Canvas plugin in this case, to verify the identity of the user based on the authentication performed by an Authorization Server, or Canvas, and retrieve an access token for each user. The token retrieval process provides access to basic profile information about the user such as their name, the course they are coming from, and their role within the course by making use of claims in JSON Web Tokens (JWTs) [25]. Thus, the user does not need to log in separately, but rather this information comes from Canvas when entering the plugin. The aforementioned data provided within the JSON Web Token is used to display a separate view based on the context specific role of the user within the course the plugin is set up in. This role within the token is also reused to allow access to separate data within the local repositories of this plugin, such as grading information and assignment status. Using Canvas as the authentication server allowed for a secure, trustworthy, and seamless authentication process.

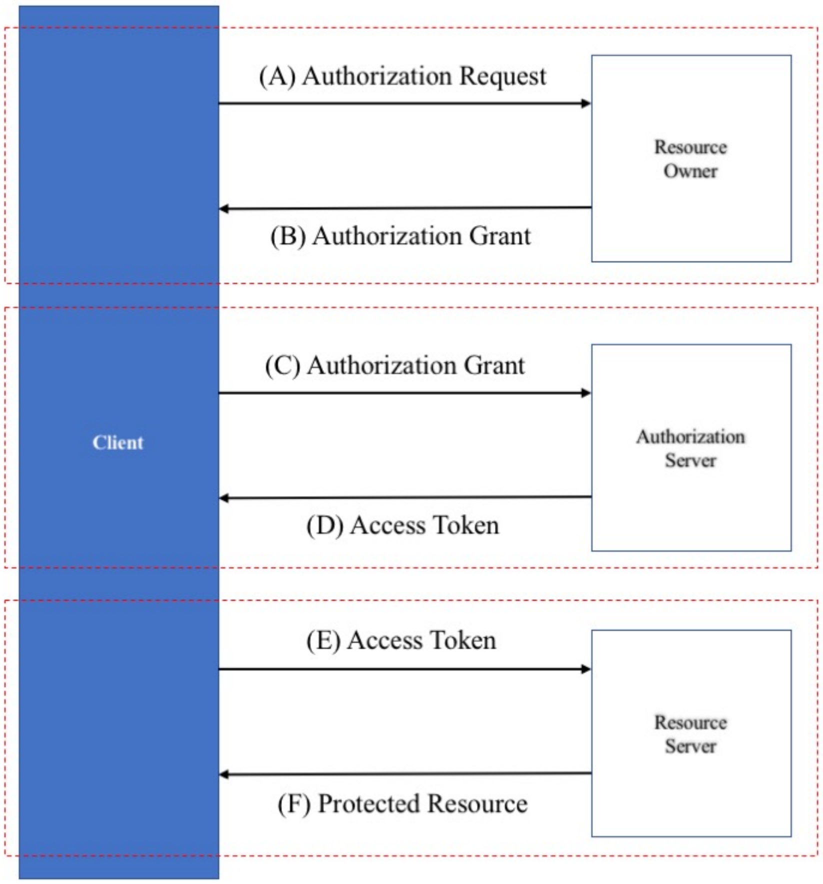


Figure 8.5: Web services abstract protocol flow [21].

Accessing Resources

For the purpose of this system, the system needs to request group and assignment information from Canvas. When making resource requests using the LTI protocol, shown as steps E and D in Figure 8.5, the verified JWT needs to be attached for the resource server to verify the authenticity of the user when a the request is being made. This token provides access to specific resources from Canvas based on the information attached to the token. For example, a students token can be used to request the group identifiers of the groups the student is a part of. As a further optimization, it was found that certain types of information can be communicated and passed to the plugin using a process called variable substitution [26]. Variable substitution allows for Canvas to expose certain data using custom fields when the tool is launched. This type of information can be context specific, meaning it is dependent on the course and status of the user itself when they launch the plugin from the course, allowing functionality such as immediately gaining access to a student's context based group identifier or even the identifiers and names of all groups within the course. This type of data access was found to be noticeably faster and more efficient than separately requesting this information from the canvas API through GET requests.

Chapter 9

Testing

In this chapter, various testing plans will be explained to evaluate whether the system contains proper functionalities and based on the testing strategies, the results will be discussed.

9.1 Test Plan

9.1.1 Analysis of the product

Our application is primarily a web app. It will have functionality to handle handing in of assignments by students after which the assignments will get checked on their quality and potential plagiarism. As such, it will be used by teachers and students alike. Teachers will upload their assignments for the students, and view results of the tools analysis. Students will interact mostly by handing in their assignments. Since there is a good chance that multiple students will hand in their assignment at the same moment, the server side must be able to handle that load. On the client side, it will be an angular application that gives the users a few forms to fill in. Examples will be forms for the name and due date of the assignment when you create it, or for students it will present the opportunity to hand in an assignment. The server side will mainly be trying to execute the students' code as well as run the plagiarism checker when the code is handed in by students. The server will most likely be a Unix server, but the hardware has not been settled on.

9.1.2 Strategy and Objectives

Test Types

Depending on the component we want to test, a different strategy would be best taken. There is multiple interesting types of tests for our project such as:

- Usability tests, for the front end
- System tests, to see how far the system is in line with requirements
- Integration test, all of the components need to work together in the end
- Smoke test, for whenever a new component is ready.

Method

For the front end we primarily want to do exploratory user testing. Normal functionality such as selecting the files to upload, creating assignments forms should be all work properly. Apart from that, the forms need to be tested for vulnerabilities such as code injections, this can be handled by unit tests. Most likely it will be out of scope to test all browsers available thoroughly for correct performance. Selenium can help automate the front end testing. The flagging of plagiarism must be rather robust, but since we'll be using a 3rd party checker, this is mostly out of scope for us. Since the application will be running students code on a server, it makes sense that the running of the code must be sandboxed in some way, or at least code should be checked for imports, so that there's no access to the filesystem. Furthermore integration within canvas is a major part of the application, so unit tests should be written to test for correct behavior. If possible within the timeframe, we would like to test for usability of our application with end users as well. Ideally all of the team members will test their own code when their implementing it. Weekly there can be integration tests to see whether the implemented components work together as of that moment.

Schedule

Testing can at best be done while implementing the functionality, so the test schedule will try to follow the implementation schedule. Weekly we will do an integration test with the existing features. Before testing the integration it is good to specify which components will be tested at that time, and as such we'll plan in 2 hours to go over the components, and how to test them. Smoke tests will be done whenever a new component is ready, to see if there's unforeseen issues between the components. In week 7 and week 9 we will do system tests, to make sure the system is in line with the requirements we have set for it and how ready it is for handing over to the client.

9.2 Requirements to be tested

Since there are two main stakeholders who are directly related to this system, the requirements that needed to be tested are categorized into teacher and student.

Teacher

- View an instructor page
- Create new assignment
- Set start date and deadline
- Choose programming language of assignment
- Upload files provided to students
- Create test instances with input and output
- Add new test instances
- Specify which test instances are required for a passing grade
- View overview of assignments that were created
- View members of the group
- View result of submissions(result of each test instance, total score) per group
- Start plagiarism checker
- View the result of plagiarism

Student

- View a learner page
- Select appropriate assignment to submit solutions
- Choose files to upload
- Upload appropriate files
- View result of past submissions
- View result per instance
- Share the submission within a group
- Resubmit the solutions
- Download solutions

9.3 Test Result

9.3.1 Usability Test

The automatic grading system can only run in a closed environment. The system restricted access to unauthorized applicants, so as a system administrator, we conducted the usability testing by ourselves with several criteria as shown below. To determine whether our web application is interactive and user-friendly, we construct the strategy for usability testing. Among the 10 general principles for interaction design from Nielsen and Molich(1990) [27], we selected four critical criteria that are suitable to evaluate the usability of our application. Here are the four main principles :

User control and freedom : System should have 'emergency exit' to let user leave from action when they confront unwanted situation. User should freely control him/herself by getting away from current interaction.

Recognition rather than recall : It is important for user not to remember information from interface. System should minimize users' memory load by making data visible and easily retrievable.

Help users recognize, diagnose, and recover from errors : System should describe the error in plain word and suggest a solution so that user can correctly realize the issue and take the next action.

Visibility of system status : It may be essential to provide current status of the system and let user keep track on what is going on by suggesting feedback within reasonable time.

These are the principles that we utilized to strengthen the usability of the system. Since the stakeholders for the system can be generally separated into two groups, teacher and student, the usability analysis of front end will be categorized into two group, 1) Teacher view 2) Student view. This analysis is a self-assessment to improve usability within the design project group.
1) Teacher view

- User can easily recognize current status of the system by viewing the overview of created assignment with start time and deadline of each assignment ('Overview of all assignments page')
- System allow user back out of the process and get away from current interaction with the system ('New assignment form page- Back button')
- User can receive immediate feedback from the system on the results of each group submission. Result of each instance(pass, timeout, incorrect outcome) and the total score is visible. ('Submission overview page')
- User can receive immediate feedback from the system by viewing the results of plagiarism checker with specific similarities and files. ('Plagiarism checker results page')

2) Student view

- System let 'upload' button activate only when the user selects a file with 'Choose File' button so that user can easily grasp how to interact with the system ('Submission Upload page')
- User can get receive immediate feedback from the system through the message('Uploaded the file successfully:' + filename) when they successfully upload a file ('Submission Upload page')
- User can view the information about uploaded files (Uploaded file, time, result, score) in real time, so user can recognize the data in the interface easily and minimize their memory load. Also, the each result is colored differently based on the outcome so that user can immediately recognize the result. ('Submission Upload page')

Chapter 10

Future Work

In this chapter we will shed some light on areas of future development we identified during implementation of the application. Of these areas, sandboxing is the most important as it is necessary for the system to be used by the university.

10.1 Assignment and Grading Services Integration

One of the components of LTI 1.3 is the Assignment and Grading services, which provides tools direct access and management of gradebook columns. This service contains three subservices, allowing management of line items, posting of scores, and getting grades from the platform's gradebook [28]. By using deep linking, the plugin would even be able to create assignments and post them on Canvas, removing the necessity to do it separately from Canvas itself [29]. The integration of these components was left for future work but would provide the final functionality necessary to completely manage assignments and grading from within the plugin itself.

10.2 Plagiarism checking improvements

There is a few regards in which the plagiarism checking strategy could in the future be improved. Currently the plagiarism checker does check for inter student plagiarism, by comparing between the submissions, but not against any other sources that could be used. Because of this, it would be wise to improve the strategy to perhaps by a multistage process. Submissions could be compared against each other to begin with, but then a second stage could further check other sources, like popular websites such as StackOverflow. Then even a third stage could be added to do analysis to detect whether it is likely a human wrote the code. By creating a pipeline such as this, multiple sources of plagiarism could be taken on. Processing of the files could be done in parallel, to speed up the process. A new report format or dashboard could be designed to show a comprehensive overview of this data to the teachers within Canvas.

10.3 Sandboxing of student code

Due to the complexity of implementing an appropriate sandbox, we did not manage to get it fully set up. The most promising direction to achieve tight sandboxing seems to be operating systems level sandboxing which has been described before in chapter 4. When the sandbox is operational there are many settings that can be tweaked to reach tighter security.

10.4 More programming language support

Currently the system only supports Python assignments but the programming language has been implemented as an interface which allows for more languages to be implemented easily. Ideally the language to be added would support being executed with a single command. This is the case for Python - tests are called as "python <student submission> <test instance input>". Any language that can be called in the same way should be trivial to add.

Chapter 11

Security

11.1 running student code on the server

Since student code may be broken, or in sporadic cases even malicious, it is import to consider how these cases are appropriately handled. Most sandbox tools available on linux rely on three core principles:

- restriction of what's visible (namespaces)
- restriction of what callable (seccomp-bpf)
- restriction of privileges (capabilities)

These allow the system to effectively wall off parts of the system while presenting itself to certain entities, like processes. under linux its possible to bind mount a filesystem where a user may be restricted to this filesystem. Processes under that user may only view that part of the filesystem and nothing else. There are a few other restrictions like restricting the user in seeing other users. These fall under the nomer of namespaces. Each namespace restricts access to a some element of the operating system.

Additionally one may "harden" the kernel. This means reducing the amount of syscalls a process may make to just the necessary ones. linux has over 350 available syscalls to it's kernel, but a normal process could for instance only use 50 of those, leaving the others without use, but still available for exploitation. Using seccomp-bpf means that you provide the kernel with a list of syscalls (policy) that are necesssary for you application, and then the kernel will make sure the process gets monitored using the policy, and any fouls will be cause to get the process for instance killed by the kernel (depending on the specific policy for that syscall; there are multiple options in what to do when a unwanted syscall is called).

Then finally there is the capabilities part. The capabilities refer to the parts of the permissions a user can have. For instance the root on Linux has all capabilities on the system. But a regular user could be restricted in using networking capabilities, since those may not be necessary for that user to have. Keep in mind that a Linux user does not have to be a person in this case. A process could just be run under a certain user on the system.

11.2 Teachers and students

Our application will be used by both teachers and students which means that we need to prevent students from accessing some resources. For example, students should not be able to see the test instances for any assignment because it would allow them to hard-code a solution that passes all instances. Students should also not be able to see the results of other groups or download the submissions of other groups. All of the actual security and role enforcement is done on the backend but roles are also used on the frontend to improve the user experience.

11.2.1 Backend

We can use the LTI API to get the role of the user that is currently logged into canvas and using our plugin. There are two roles that are relevant to our application: ”Instructor” and ”Learner”, these represent teachers and students respectively. We use these roles on the backend to check if the user is allowed to access the requested resource. For example, teachers can get all submissions from all groups for a given assignment, while a student can only see the submissions made by their group. Only teachers can create assignments.

11.2.2 Frontend

Since security should be ensured on the backend and not the frontend, we only use roles on the frontend to decide which views are shown to the user. This does not provide security but it avoids showing the user a view which they can not use. For example, students are not shown the view with all submissions from all groups because due to their role they would not see the submissions of other groups anyway.

Bibliography

[1] C. Ragkhitwetsagul, J. Krinke, and D. Clark, *A comparison of code similarity analysers.* 2018. DOI: https://doi-org.ezproxy2.utwente.nl/10.1007/s10664-017-9564-7.

[2] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier, *Clone detection using abstract syntax trees.* 1998.

[3] S. Li. "Natural language processing for fuzzy string matching with python," Medium. (Dec. 6, 2018), [Online]. Available: https://towardsdatascience.com/natural-language-processing-for-fuzzy-string-matching-with-python-6632b7824c49 (visited on 04/20/2022).

[4] M. Mindzak and S. E. Eaton. "Artificial intelligence is getting better at writing, and universities should worry about plagiarism," The Conversation. (), [Online]. Available: http://theconversation.com/artificial-intelligence-is-getting-better-at-writing-and-universities-should-worry-about-plagiarism-160481 (visited on 04/22/2022).

[5] L. Floridi and M. Chiriatti, "GPT-3: Its nature, scope, limits, and consequences," *Minds and Machines*, vol. 30, no. 4, pp. 681–694, Dec. 1, 2020, ISSN: 1572-8641. DOI: 10.1007/s11023-020-09548-1. [Online]. Available: https://doi.org/10.1007/s11023-020-09548-1 (visited on 04/22/2022).

[6] M. Chen, J. Tworek, H. Jun, *et al.*, "Evaluating large language models trained on code," *arXiv:2107.03374 [cs]*, Jul. 14, 2021. arXiv: 2107.03374. [Online]. Available: http://arxiv.org/abs/2107.03374 (visited on 04/22/2022).

[7] K. Johnson. "AI researchers use heartbeat detection to identify deepfake videos," VentureBeat. (Sep. 3, 2020), [Online]. Available: https://venturebeat.com/2020/09/03/ai-researchers-use-heartbeat-detection-to-identify-deepfake-videos/ (visited on 04/20/2022).

[8] N. Yu, L. Davis, and M. Fritz, "Attributing fake images to GANs: Learning and analyzing GAN fingerprints," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, ISSN: 2380-7504, Oct. 2019, pp. 7555–7565. DOI: 10.1109/ICCV.2019.00765.

[9] L. Fröhling and A. Zubiaga, "Feature-based detection of automated language models: Tackling GPT-2, GPT-3 and grover," *PeerJ Computer Science*, vol. 7, e443, Apr. 6, 2021, ISSN: 2376-5992. DOI: 10.7717/peerj-cs.443. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8049133/ (visited on 04/20/2022).

[10] S. Gehrmann, H. Strobelt, and A. Rush, "GLTR: Statistical detection and visualization of generated text," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Florence, Italy: Association for Computational Linguistics, 2019, pp. 111–116. DOI: 10.18653/v1/P19-3019. [Online]. Available: https://www.aclweb.org/anthology/P19-3019 (visited on 04/20/2022).

[11] F. Harrag, M. Debbah, K. Darwish, and A. Abdelali, "BERT transformer model for detecting arabic GPT2 auto- generated tweets," p. 8,

[12] PyPy Project. "PyPy's sandboxing features — PyPy documentation." (), [Online]. Available: https://doc.pypy.org/en/latest/sandbox.html (visited on 04/21/2022).

[13] Docker. "What is a container?" Docker. (), [Online]. Available: https://www.docker.com/resources/what-container/ (visited on 04/21/2022).

[14] Canonical Ltd. "Linux containers - LXD - introduction." (), [Online]. Available: https://linuxcontainers.org/lxd/introduction/ (visited on 04/20/2022).

[15] Firejail Project. "Firejail," Firejail. (), [Online]. Available: https://firejail.wordpress.com/ (visited on 04/20/2022).

[16] Red Hat Inc. "Chapter 1. introduction to control groups (cgroups) red hat enterprise linux 6," Red Hat Customer Portal. (), [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/resource_management_guide/ch01 (visited on 04/20/2022).

[17] Kubernetes. "Kubernetes." (), [Online]. Available: https://kubernetes.io/ (visited on 04/21/2022).

[18] Project Jupyter. "About jupyterhub." (), [Online]. Available: https://jupyter.org (visited on 04/21/2022).

[19] IMS Global Learning Consortium. "Learning tools interoperability — IMS global learning consortium." (), [Online]. Available: https://www.imsglobal.org/activity/learning-tools-interoperability (visited on 04/21/2022).

[20] University of Oxford. "GitHub - oxctl/spring-security-lti13: A LTI 1.3 implementation for spring security that builds on the OAuth2 support." (), [Online]. Available: https://github.com/oxctl/spring-security-lti13 (visited on 04/21/2022).

[21] IMS Global Learning Consortium. "IMS security framework 1.0 — IMS global learning consortium." (), [Online]. Available: https://www.imsglobal.org/spec/security/v1p0/ (visited on 04/21/2022).

[22] IETF OAuth Working Group. "OAuth 2.0 — OAuth." (), [Online]. Available: https://oauth.net/2/ (visited on 04/21/2022).

[23] OpenID Connect Working Group. "OpenID connect — OpenID." (), [Online]. Available: https://openid.net/connect/ (visited on 04/21/2022).

[24] Instructure Inc. "OAuth2 - canvas LMS REST API documentation." (), [Online]. Available: https://canvas.utwente.nl/doc/api/file.oauth.html (visited on 04/21/2022).

[25] M. Jones, J. Bradley, and N. Sakimura, "JSON web token (JWT)," Internet Engineering Task Force, Request for Comments RFC 7519, May 2015, Num Pages: 30. DOI: 10.17487/RFC7519. [Online]. Available: https://datatracker.ietf.org/doc/rfc7519 (visited on 04/21/2022).

[26] Instructure Inc. "Canvas LMS REST API documentation." (), [Online]. Available: https://canvas.instructure.com/doc/api/file.tools_variable_substitutions.html (visited on 04/22/2022).

[27] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," pp. 249–256, 1990. DOI: 10.1145/97243.97281. [Online]. Available: https://doi.org/10.1145/97243.97281.

[28] IMS Global Learning Consortium. "Learning tools interoperability assignment and grade services version 2.0 — IMS global learning consortium." (), [Online]. Available: https://www.imsglobal.org/spec/lti-ags/v2p0/ (visited on 04/21/2022).

[29] Instructure Inc. "Using deep linking to select resources - canvas LMS REST API documentation." (), [Online]. Available: https://canvas.instructure.com/doc/api/file.content_item.html (visited on 04/21/2022).

Appendix A

Screenshots of the plugin

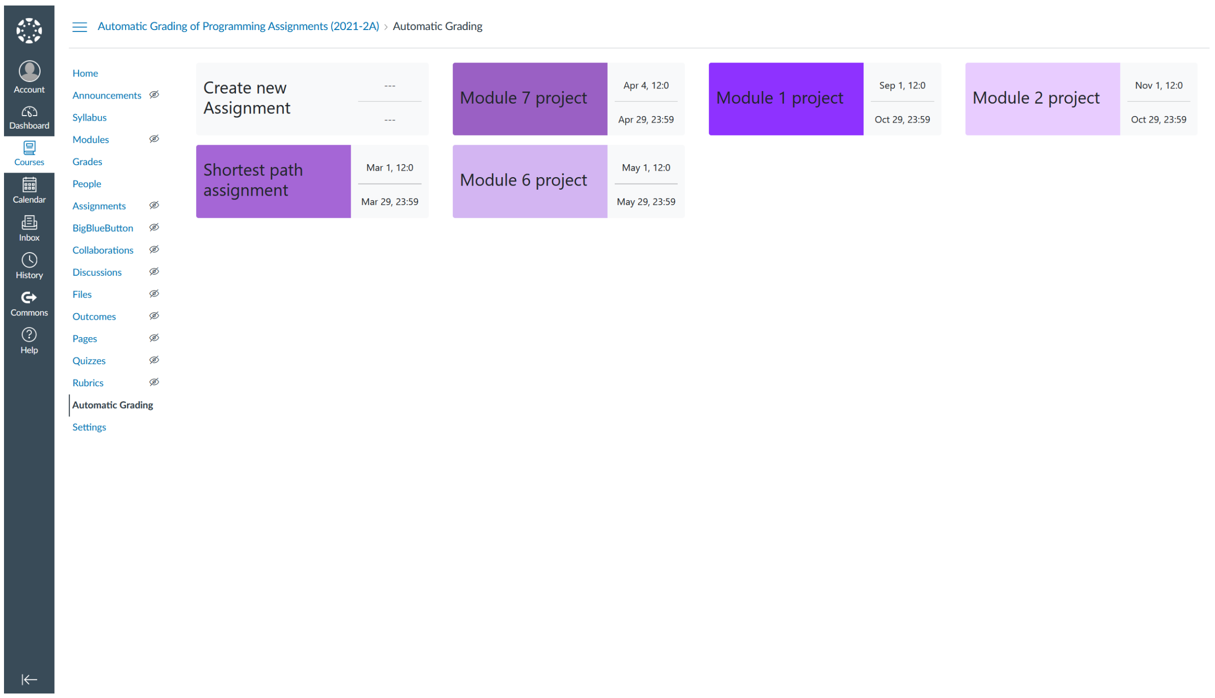
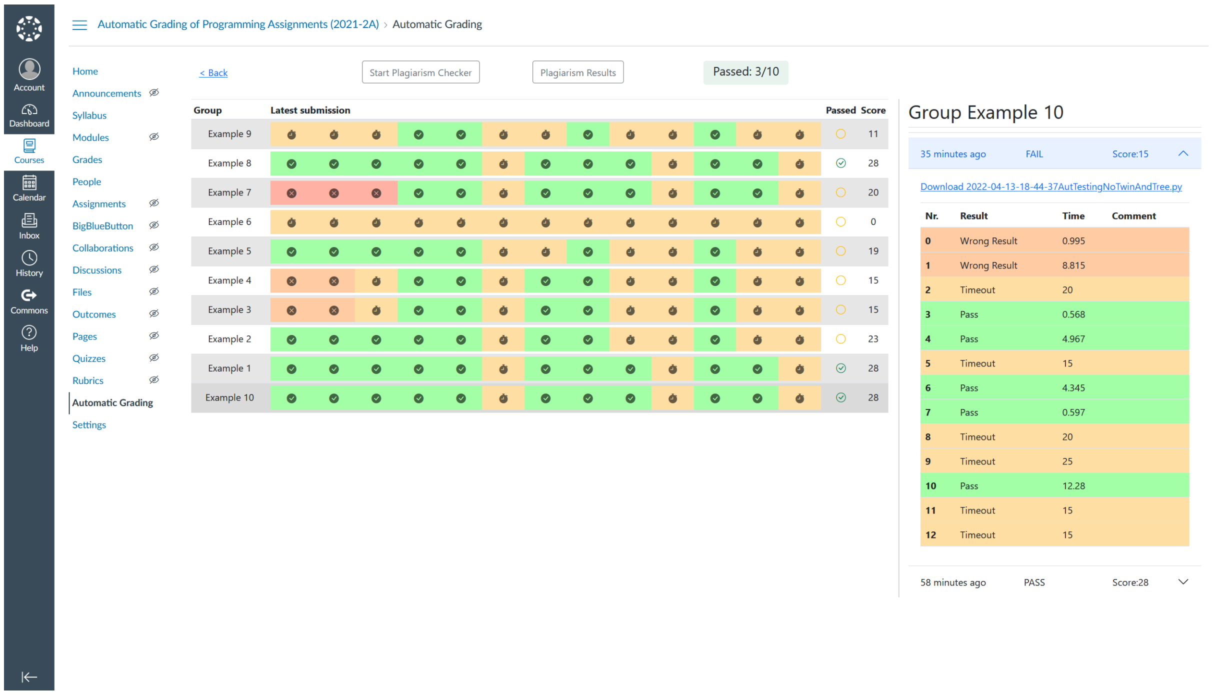


Figure A.1: Assignment selection for teachers



Automatic Grading of Programming Assignments (2021-2A) › Automatic Grading

< Back | Start Plagiarism Checker | Plagiarism Results | Passed: 3/10

Group	Latest submission													Passed	Score
Example 9	⏱	⏱	⏱	✓	✓	⏱	⏱	✓	⏱	⏱	✓	⏱	⏱	○	11
Example 8	✓	✓	✓	✓	✓	⏱	✓	✓	✓	⏱	✓	✓	⏱	✓	28
Example 7	✗	✗	✗	✓	✓	⏱	✓	✓	✓	⏱	✓	✓	⏱	○	20
Example 6	⏱	⏱	⏱	⏱	⏱	⏱	⏱	⏱	⏱	⏱	⏱	⏱	⏱	○	0
Example 5	✓	✓	✓	✓	✓	⏱	⏱	✓	⏱	⏱	✓	⏱	⏱	○	19
Example 4	✗	✗	⏱	✓	✓	⏱	✓	✓	⏱	⏱	✓	⏱	⏱	○	15
Example 3	✗	✗	⏱	✓	✓	⏱	✓	✓	⏱	⏱	✓	⏱	⏱	○	15
Example 2	✓	✓	✓	✓	✓	⏱	✓	✓	⏱	⏱	✓	⏱	⏱	○	23
Example 1	✓	✓	✓	✓	✓	⏱	✓	✓	✓	⏱	✓	✓	⏱	✓	28
Example 10	✓	✓	✓	✓	✓	⏱	✓	✓	✓	⏱	✓	✓	⏱	✓	28

Group Example 10

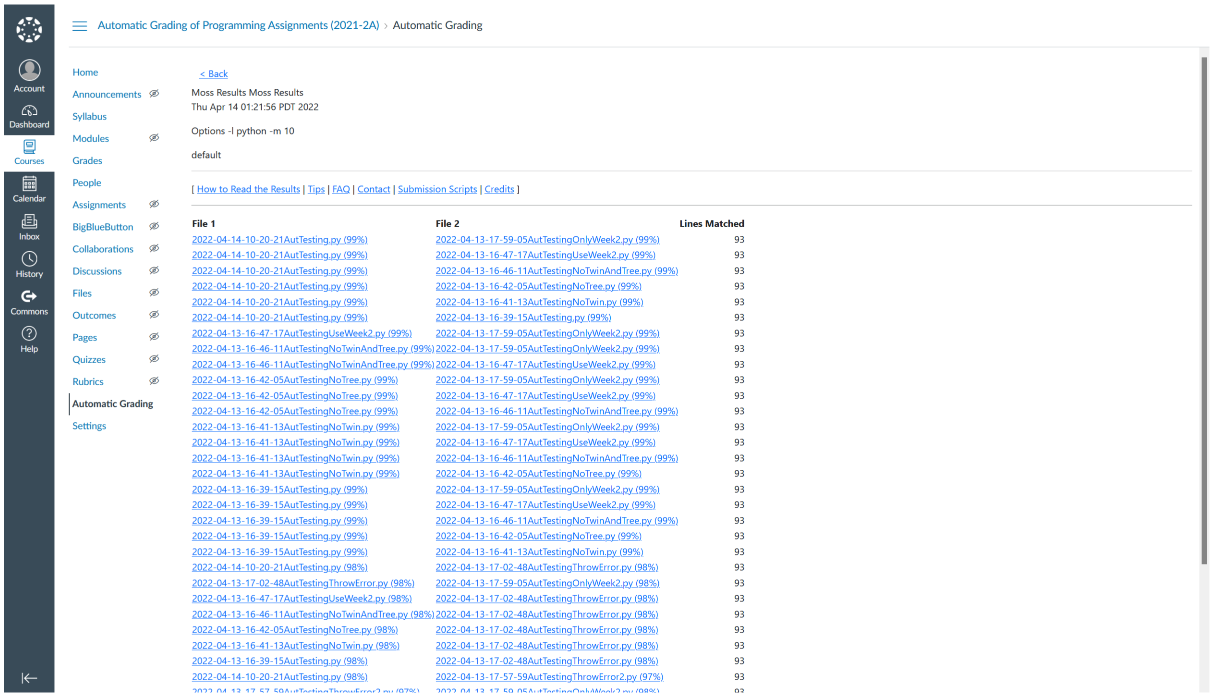
35 minutes ago FAIL Score:15

Download 2022-04-13-18-44-37AutTestingNoTwinAndTree.py

Nr.	Result	Time	Comment
0	Wrong Result	0.995	
1	Wrong Result	8.815	
2	Timeout	20	
3	Pass	0.568	
4	Pass	4.967	
5	Timeout	15	
6	Pass	4.345	
7	Pass	0.597	
8	Timeout	20	
9	Timeout	25	
10	Pass	12.28	
11	Timeout	15	
12	Timeout	15	

58 minutes ago PASS Score:28

Figure A.2: Submission view with a group selected



Automatic Grading of Programming Assignments (2021-2A) › Automatic Grading

< Back

Moss Results Moss Results
Thu Apr 14 01:21:56 PDT 2022

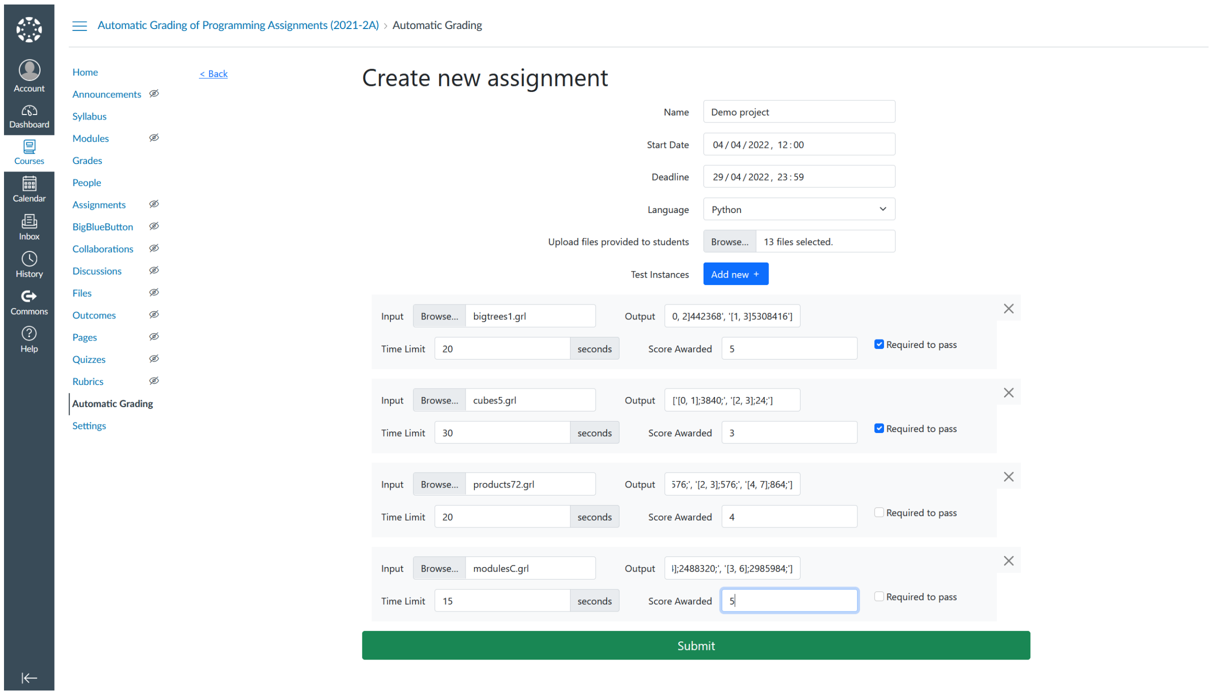
Options -l python -m 10

default

[How to Read the Results | Tips | FAQ | Contact | Submission Scripts | Credits]

File 1	File 2	Lines Matched
2022-04-14-10-20-21AutTesting.py (99%)	2022-04-13-17-59-05AutTestingOnlyWeek2.py (99%)	93
2022-04-14-10-20-21AutTesting.py (99%)	2022-04-13-16-47-17AutTestingUseWeek2.py (99%)	93
2022-04-14-10-20-21AutTesting.py (99%)	2022-04-13-16-46-11AutTestingNoTwinAndTree.py (99%)	93
2022-04-14-10-20-21AutTesting.py (99%)	2022-04-13-16-42-05AutTestingNoTree.py (99%)	93
2022-04-14-10-20-21AutTesting.py (99%)	2022-04-13-16-41-13AutTestingNoTwin.py (99%)	93
2022-04-14-10-20-21AutTesting.py (99%)	2022-04-13-16-39-15AutTesting.py (99%)	93
2022-04-13-16-47-17AutTestingUseWeek2.py (99%)	2022-04-13-17-59-05AutTestingOnlyWeek2.py (99%)	93
2022-04-13-16-46-11AutTestingNoTwinAndTree.py (99%)	2022-04-13-17-59-05AutTestingOnlyWeek2.py (99%)	93
2022-04-13-16-46-11AutTestingNoTwinAndTree.py (99%)	2022-04-13-16-47-17AutTestingUseWeek2.py (99%)	93
2022-04-13-16-42-05AutTestingNoTree.py (99%)	2022-04-13-17-59-05AutTestingOnlyWeek2.py (99%)	93
2022-04-13-16-42-05AutTestingNoTree.py (99%)	2022-04-13-16-47-17AutTestingUseWeek2.py (99%)	93
2022-04-13-16-42-05AutTestingNoTree.py (99%)	2022-04-13-16-46-11AutTestingNoTwinAndTree.py (99%)	93
2022-04-13-16-41-13AutTestingNoTwin.py (99%)	2022-04-13-17-59-05AutTestingOnlyWeek2.py (99%)	93
2022-04-13-16-41-13AutTestingNoTwin.py (99%)	2022-04-13-16-47-17AutTestingUseWeek2.py (99%)	93
2022-04-13-16-41-13AutTestingNoTwin.py (99%)	2022-04-13-16-46-11AutTestingNoTwinAndTree.py (99%)	93
2022-04-13-16-41-13AutTestingNoTwin.py (99%)	2022-04-13-16-42-05AutTestingNoTree.py (99%)	93
2022-04-13-16-39-15AutTesting.py (99%)	2022-04-13-17-59-05AutTestingOnlyWeek2.py (99%)	93
2022-04-13-16-39-15AutTesting.py (99%)	2022-04-13-16-47-17AutTestingUseWeek2.py (99%)	93
2022-04-13-16-39-15AutTesting.py (99%)	2022-04-13-16-46-11AutTestingNoTwinAndTree.py (99%)	93
2022-04-13-16-39-15AutTesting.py (99%)	2022-04-13-16-42-05AutTestingNoTree.py (99%)	93
2022-04-13-16-39-15AutTesting.py (99%)	2022-04-13-16-41-13AutTestingNoTwin.py (99%)	93
2022-04-14-10-20-21AutTesting.py (98%)	2022-04-13-17-02-48AutTestingThrowError.py (98%)	93
2022-04-13-17-02-48AutTestingThrowError.py (98%)	2022-04-13-17-59-05AutTestingOnlyWeek2.py (98%)	93
2022-04-13-16-47-17AutTestingUseWeek2.py (98%)	2022-04-13-17-02-48AutTestingThrowError.py (98%)	93
2022-04-13-16-46-11AutTestingNoTwinAndTree.py (98%)	2022-04-13-17-02-48AutTestingThrowError.py (98%)	93
2022-04-13-16-42-05AutTestingNoTree.py (98%)	2022-04-13-17-02-48AutTestingThrowError.py (98%)	93
2022-04-13-16-41-13AutTestingNoTwin.py (98%)	2022-04-13-17-02-48AutTestingThrowError.py (98%)	93
2022-04-13-16-39-15AutTesting.py (98%)	2022-04-13-17-02-48AutTestingThrowError.py (98%)	93
2022-04-14-10-20-21AutTesting.py (98%)	2022-04-13-17-57-59AutTestingThrowError2.py (97%)	93

Figure A.3: Plagiarism checker results



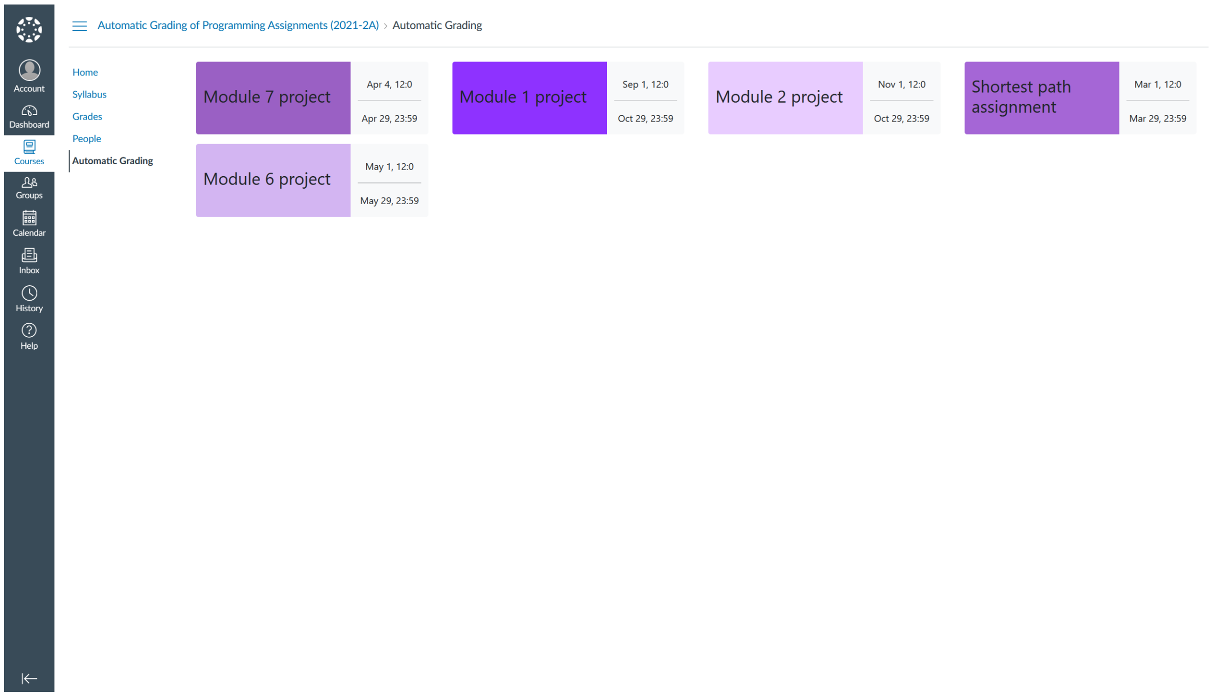
Create new assignment

Name	Demo project
Start Date	04 / 04 / 2022 , 12 : 00
Deadline	29 / 04 / 2022 , 23 : 59
Language	Python
Upload files provided to students	Browse... 13 files selected.
Test Instances	Add new +

Input	Browse... bigtrees1.grl	Output	0, 2]442368', '[1, 3]5308416']			
Time Limit	20 seconds	Score Awarded	5	✓ Required to pass		
Input	Browse... cubes5.grl	Output	['[0, 1];3840;', '[2, 3];24;']			
Time Limit	30 seconds	Score Awarded	3	✓ Required to pass		
Input	Browse... products72.grl	Output	576;', '[2, 3];576;', '[4, 7];864;']			
Time Limit	20 seconds	Score Awarded	4	Required to pass		
Input	Browse... modulesC.grl	Output	];2488320;', '[3, 6];2985984;']			
Time Limit	15 seconds	Score Awarded	5	Required to pass		

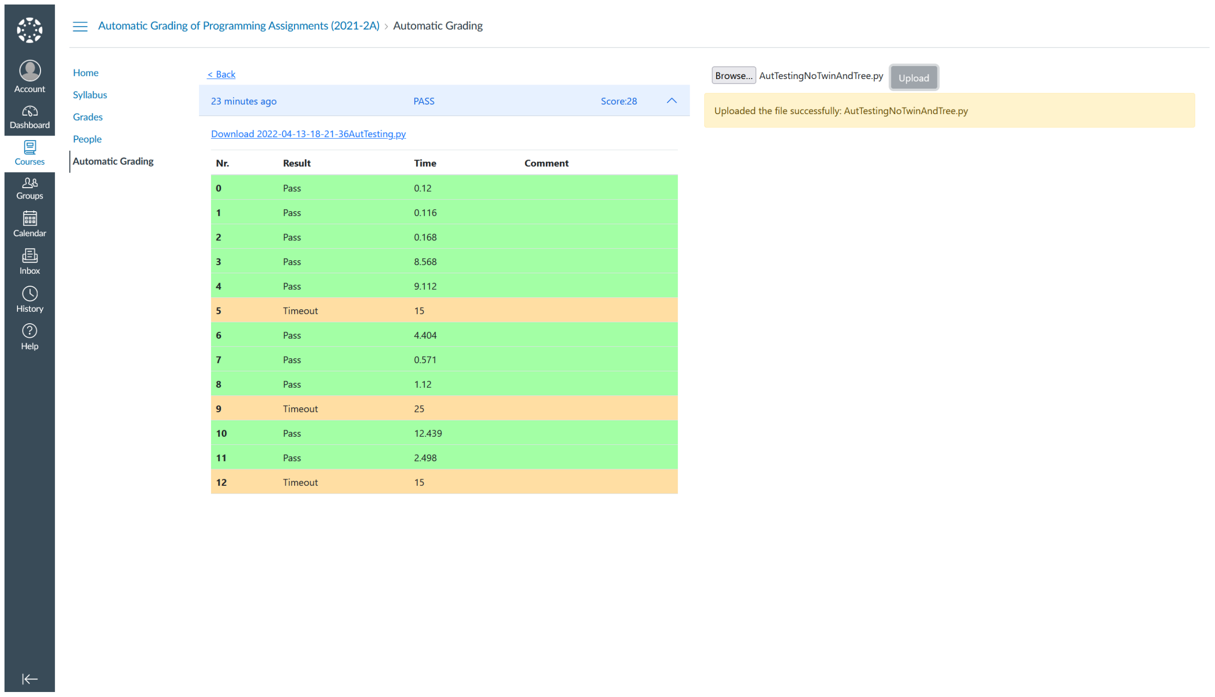
Submit

Figure A.4: New assignment form



Module 7 project	Apr 4, 12:0	Apr 29, 23:59
Module 1 project	Sep 1, 12:0	Oct 29, 23:59
Module 2 project	Nov 1, 12:0	Oct 29, 23:59
Shortest path assignment	Mar 1, 12:0	Mar 29, 23:59
Module 6 project	May 1, 12:0	May 29, 23:59

Figure A.5: Assignment selection for students



Automatic Grading of Programming Assignments (2021-2A) › Automatic Grading

< Back

23 minutes ago PASS Score:28

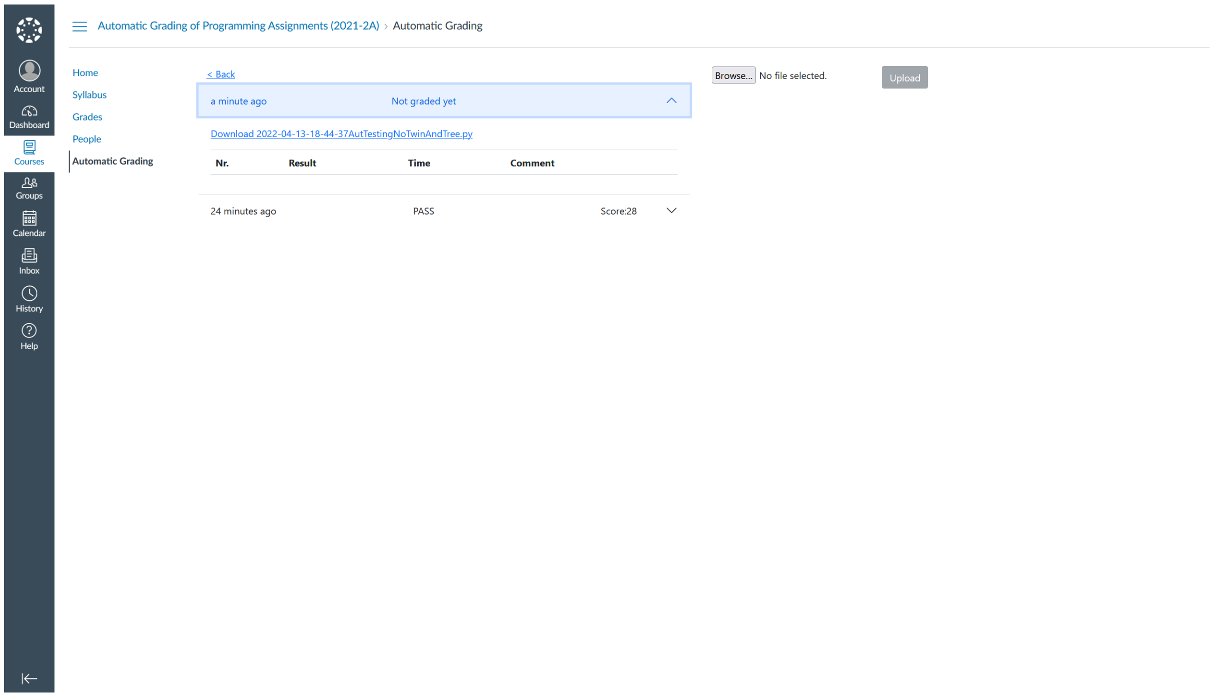
Download 2022-04-13-18-21-36AutTesting.py

Nr.	Result	Time	Comment
0	Pass	0.12	
1	Pass	0.116	
2	Pass	0.168	
3	Pass	8.568	
4	Pass	9.112	
5	Timeout	15	
6	Pass	4.404	
7	Pass	0.571	
8	Pass	1.12	
9	Timeout	25	
10	Pass	12.439	
11	Pass	2.498	
12	Timeout	15	

Browse... AutTestingNoTwinAndTree.py Upload

Uploaded the file successfully: AutTestingNoTwinAndTree.py

Figure A.6: Submission view, not graded yet



Automatic Grading of Programming Assignments (2021-2A) › Automatic Grading

< Back

a minute ago Not graded yet

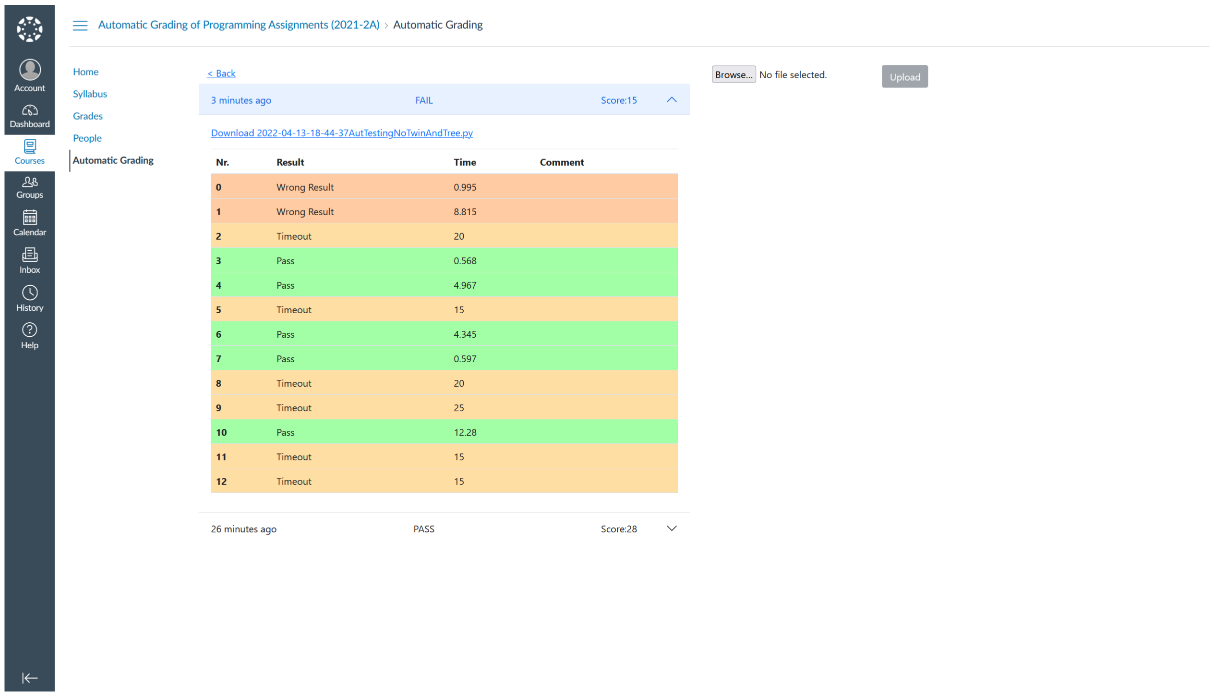
Download 2022-04-13-18-44-37AutTestingNoTwinAndTree.py

Nr.	Result	Time	Comment

24 minutes ago PASS Score:28

Browse... No file selected. Upload

Figure A.7: Submission view, not graded yet



3 minutes ago FAIL Score:15

Download 2022-04-13-18-44-37AutTestingNoTwinAndTree.py

Nr.	Result	Time	Comment
0	Wrong Result	0.995	
1	Wrong Result	8.815	
2	Timeout	20	
3	Pass	0.568	
4	Pass	4.967	
5	Timeout	15	
6	Pass	4.345	
7	Pass	0.597	
8	Timeout	20	
9	Timeout	25	
10	Pass	12.28	
11	Timeout	15	
12	Timeout	15	

26 minutes ago PASS Score:28

Figure A.8: Submission view, graded

Appendix B

Project Proposal

B.1 Project Description

Due to logistical problems arising from the various programming assignments in the curriculum of TCS and the increasing number of students, grading assignments has become challenging for teachers. Our project is to address this issue by developing a system to automatically grade programming assignments and additionally integrate a plagiarism checker.
We aim to develop a high-accuracy, rapid tool for auto-grading as well as a user interface that allows students to use the tool and teacher to view results. There are two representative stakeholders that are most closely related to our project - teachers and students. The goal from teacher's point of view is to reduce the time and resources needed to grade assignments without creating a difference in the outcome dependant on whether this tool is used. From the student's perspective, the product should allow them to view their own grade and feedback through interface of the system. For these reasons, much of the work will generally involve developing a flexible design for this system, which is universal and scalable.

B.2 Project Organization

We are going to be working on this project with a scrum-like structure. We want to work closely with the clients, holding weekly meetings, from which we will derive feedback about the project and whether it is headed in the right direction. We want to incorporate the feedback as soon as possible, so that the product does not deviate from the client's vision. The meetings will be lead by different members of the team, depending on who has the most motivation and reason to take charge.
As for within-group meetings, we will not incorporate daily meetings into our project organization. Instead we will opt out for a more flexible schedule, in which whenever a meeting is necessary we can schedule one ahead of time. During these meetings we will discuss progress and assign new tasks to different members.
We have not yet decided on what roles different people in the team will have and what tasks they will have, but we want to be flexible and let everyone work on the things they prefer.

B.3 Planned Deliverables

B.3.1 Backend

For this project, it is intended for the server to automatically evaluate students' assignments and provide feedback to help in grading. For this purpose, two types of users should be supported in the login system to differentiate the tasks of the teachers and students. Additionally, the assignments submitted by students must be checked for plagiarism using MOSS. The submissions are intended to be pulled using the canvas

API for this auto-grader. If time permits, a database should also be set up to store the results of assignments for administrative and statistics purposes.

B.3.2 Frontend

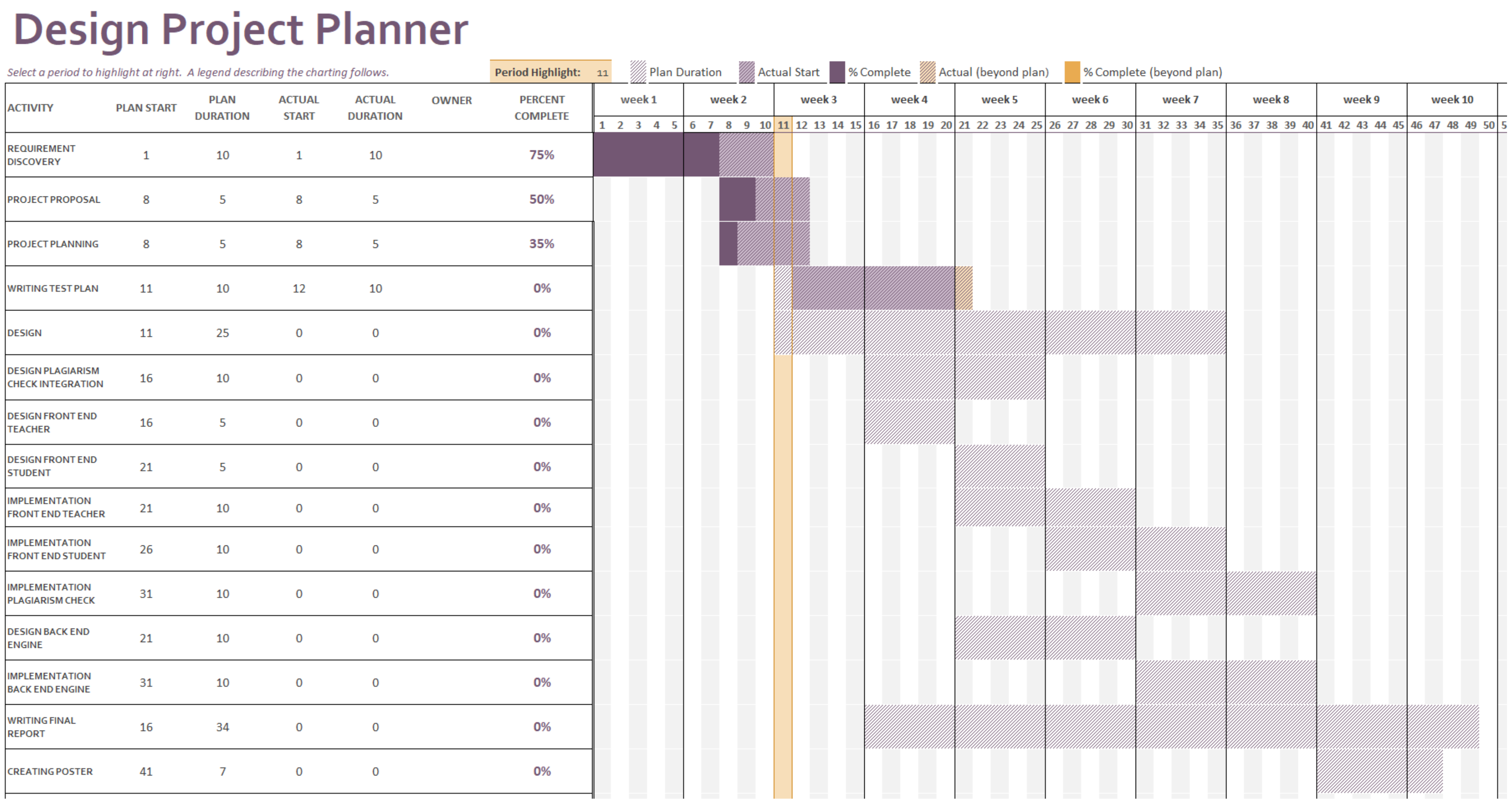
The tool should provide a user interface allowing the creation of assignments and defining which programs need to provide correct results for the student to pass the overarching assignment. Students should be able to check whether their own submission provided correct results and what the run-time was, while teachers naturally should be able to see the results of all students' submissions. If time permits, the assignments could provide more useful feedback for the student to see what went wrong, such as whether the program timed out, an exception was thrown, or the result was incorrect.

B.3.3 Documentation

As this project's focus is on designing a correct system, the majority of the planning includes time allocated for creating a global design which can incorporate all of the features required in the future - even if they are not implemented during the duration of this project. In the planning, time is also allocated for writing the report and creating a poster to present the work done.

B.4 Planning

To aid the planning of this project, we have made a Gantt chart to track our deliverables, which is shown in Figure B.1. In this chart we can see in which week we are, when a activity should start, when it actually started, how much time it needed, and as a result of all of the above we can keep track of what our current progress is. Activities are defined on a high level on purpose, so as to abstract away a little to keep oversight.



Design Project Planner

Select a period to highlight at right. A legend describing the charting follows.

Period Highlight: 11

▨ Plan Duration ▨ Actual Start ■ % Complete ▨ Actual (beyond plan) ■ % Complete (beyond plan)

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	OWNER	PERCENT COMPLETE
REQUIREMENT DISCOVERY	1	10	1	10		75%
PROJECT PROPOSAL	8	5	8	5		50%
PROJECT PLANNING	8	5	8	5		35%
WRITING TEST PLAN	11	10	12	10		0%
DESIGN	11	25	0	0		0%
DESIGN PLAGIARISM CHECK INTEGRATION	16	10	0	0		0%
DESIGN FRONT END TEACHER	16	5	0	0		0%
DESIGN FRONT END STUDENT	21	5	0	0		0%
IMPLEMENTATION FRONT END TEACHER	21	10	0	0		0%
IMPLEMENTATION FRONT END STUDENT	26	10	0	0		0%
IMPLEMENTATION PLAGIARISM CHECK	31	10	0	0		0%
DESIGN BACK END ENGINE	21	10	0	0		0%
IMPLEMENTATION BACK END ENGINE	31	10	0	0		0%
WRITING FINAL REPORT	16	34	0	0		0%
CREATING POSTER	41	7	0	0		0%

Figure B.1: Gantt chart for planning

B.5 Risk Analysis

We identified the possible risks regarding the use of our developed system and analyzed the probability and severity of them. These values are on a scale of low, moderate or high. We also look at what actions we can take to minimize the probability of them happening and how to make their consequences less severe.

- **Plagiarism false positive**
  - **Description**: The system could falsely flag a pair of students for plagiarism, which can have serious consequences for the students .
  - **Probability**: Low. A pair of students is very unlikely to be flagged for plagiarism if they did not share any code. However, is it likely that some of the pairs will get flagged due to the amount of submissions.
  - **Severity**: High. Accusations of plagiarism can have dire consequences for the students.
  - **Actions to minimize probability**: The threshold for what is and is not considered plagiarism can be adjusted with enough example programs including some plagiarized ones.
  - **Actions to minimize severity**: A teacher should inspect all suspicious pairs of submissions by hand to determine if the code appears plagiarized.
- **System failure under heavy load**
  - **Description**: During peak usage by the students the server could fail and the students would be unable to submit and test their implementations.
  - **Probability**: High. The team has no experience with building systems of this scale.
  - **Severity**: Moderate. The development process of the students would be delayed, however, the downtime before the server is restarted should be rather short.
  - **Actions to minimize probability**: Stress test the system before deployment and improve until it is reliable.
  - **Actions to minimize severity**: Make the system as easy and fast to restart as possible to shorten the downtime.
- **Malicious students**
  - **Description**: The system will be executing code written by the students. A malicious student could use this to access other students' submissions or take the system down altogether.
  - **Probability**: Moderate. There are always students who try to penetration test the systems provided by the university, for example the public server for module 2. The students are generally doing it for fun, however, without malicious intents.
  - **Severity**: High. Without any preventive measures in place, there are unlimited consequences for the system. All submissions could be leaked and all past results deleted.
  - **Actions to minimize probability**: There is no way to stop the students from trying to break the system.
  - **Actions to minimize severity**: We will investigate how other similar systems deal with this issue later in the development process.

B.6 Task Division

The project is in its early stages, therefore there is no task division yet, but we will divide tasks evenly when the time comes, according to our preferences for front-end or back-end development. We plan to divide the design tasks evenly with all team members doing an equal amount of work.