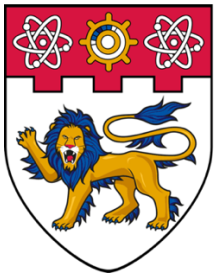

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SCSE21-0804
Full-stack Web Development for Auto-Assessment Platform

Final Report

Author
Lee Jun Wei (U1922896C)

Project Supervisor
Dr Loke Yuan Ren

Examiner
A/P Anwitaman Datta

Submitted in Partial Fulfilment of the Requirements of the Degree of Bachelor of Engineering (Computer Science) of the Nanyang Technological University

School of Computer Science and Engineering

2022

Abstract

Assessments in educational institutions are essential for educators to evaluate the efficacy of instruction and a means for students to measure their individual progress. With the ongoing Covid-19 pandemic prevalent across the globe, many educational institutions have adopted online-based platforms for managing and administering assessments to their students. The School of Computer Science and Engineering of Nanyang Technological University is no exception. Presently, the SC1007 Data Structures and Algorithms course administers various assessments to students through the HackerEarth online platform. Although the platform is full-featured, it incurs a high cost to utilise, and it is preferable to have an in-house platform where complete control over all aspects of the system is possible.

This project is a continuation of two past year students' progress, where an automated assessment platform (AASP) that can perform automated grading of programming questions was developed. However, various core issues were identified in the existing platform, such as its maintainability and security vulnerabilities.

A new AASP has been designed and developed in this project to address the existing issues. The new platform promotes maintainability and security by considering aspects such as selecting a more commonly taught primary programming language, a web framework that focuses on forwards-compatibility and reducing the complexity of the technology stack by integrating an open-source code execution engine called Judge0.

Acknowledgement

I would like to express my sincerest gratitude to my project supervisor Dr Loke Yuan Ren, for his patience, insightful comments and constructive feedback that have helped me tremendously throughout the project.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Assessments in educational institutions serve as an important feedback channel for educators to gauge the effectiveness of the lessons and provide learners with a measure of their progress. Traditionally, assessments are administered to learners physically in a paper-based format. However, with the prevalence of technology, online-based platforms have become more popular in recent years.

The adoption of these online-based platforms introduces many benefits to educational institutions. These include instantaneous grading, ease of distribution and objectivity of marking [1]. On top of that, with the prevailing COVID-19 pandemic around the globe, many educational institutions were forced to migrate to online-based platforms to replace traditional in-person activities [2].

At Nanyang Technological University (NTU), assignments and tests for the *SC1007 Data Structures and Algorithms* course are managed and disseminated through HackerEarth, an external commercial platform for online technical skill assessments. Although the platform is feature-packed, a high cost is incurred for the utilisation of the platform, and an in-house solution that offers complete control over all aspects of the system is desired.

1.2 Prior Work

Soh Yan Quan, Kenneth was the first student who designed and developed the first version of the **A**uto-**As**sessment **P**latform (AASP) [3]. The platform supports creating and distributing quizzes consisting of both programming and structured questions. Subsequently, Yap Guan Sheng continued development, implemented additional features and integrated an improved user interface [4].

1.3 The Problem

While many features have been developed by the project's predecessors Soh and Yap, the platform is not ready to be deployed for actual use due to issues such as critical bugs, insecure codes and unpatched security vulnerabilities. In addition, due to the technology stack selected, it is challenging to maintain and continue development by the successors of the project. These are described in greater detail in **Chapter 2**.

1.4 Objectives

The project's objective is to design and develop a new in-house Auto-Assessment Platform (AASP) that is fast, reliable and intuitive and can deliver the best user experience to the end-users. Furthermore, the platform will be designed with Maintainability in mind to support and facilitate the project's continued development by future successors. This encompasses selecting suitable technologies and frameworks and simplifying the technology stack to reduce complexity wherever possible. Although a new platform will be designed and implemented from the ground up, works from the project's predecessors will be studied, and positive traits will be integrated wherever possible.

1.5 Scope

The scope of the project includes:

- To understand the needs of the end-users of an online assessment platform.
- To understand the design, technology stack and implementation of the existing platform developed by the predecessors.
- To analyse the existing platform to identify desirable characteristics and areas that could be improved.
- To plan, design and implement a new Auto-Assessment Platform that addresses the current issues.

1.6 Organisation of Report

Chapter 1 provides an overview of the project. This includes the background, prior works, problems and the project's objectives.

Chapter 2 reviews the platform developed by the project's predecessors and other related products in the market to identify desirable characteristics and issues.

Chapter 3 discusses the design methodology and planning stages of the AASP.

Chapter 4 talks about the implementation of the system, including the architecture, key features of the implemented AASP and deployment information.

Chapter 5 describes the suggestions for possible future works.

Chapter 6 provides a conclusion to the project.

1.7 Project Plan

The Gantt chart (Figures 1 and 2) outlines the breakdown of the various project phases.

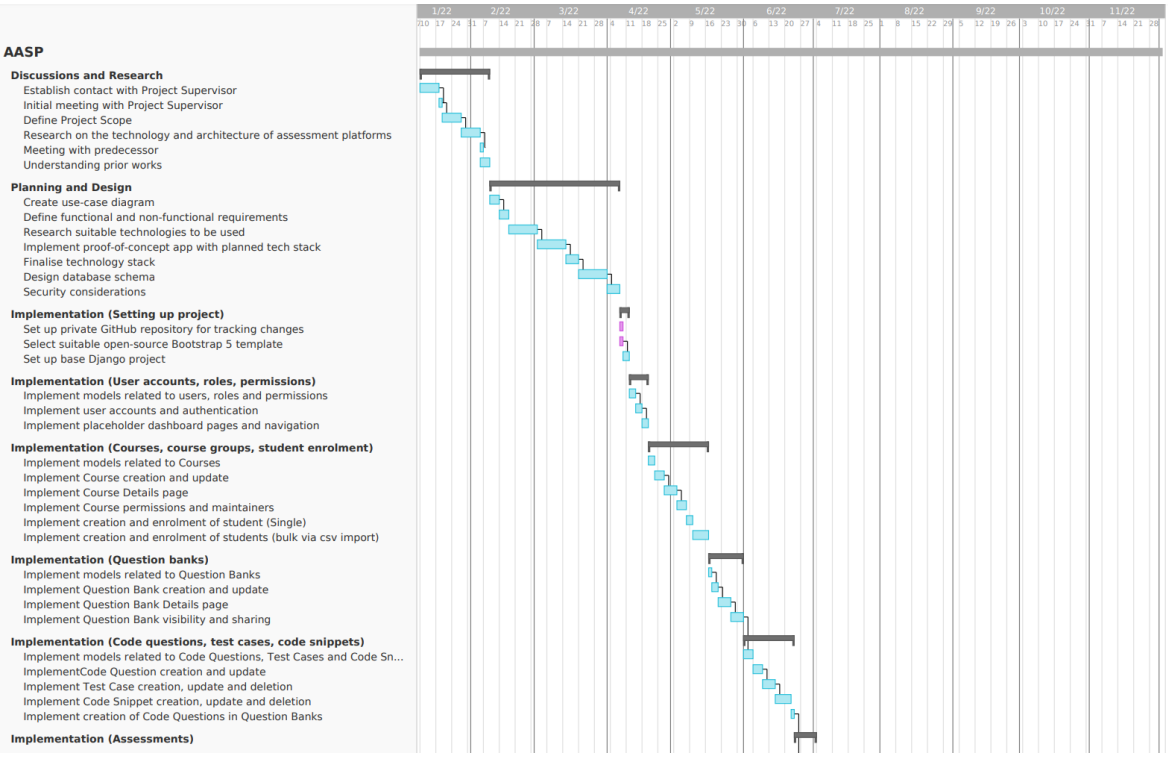

Figure 1 - Gantt Chart

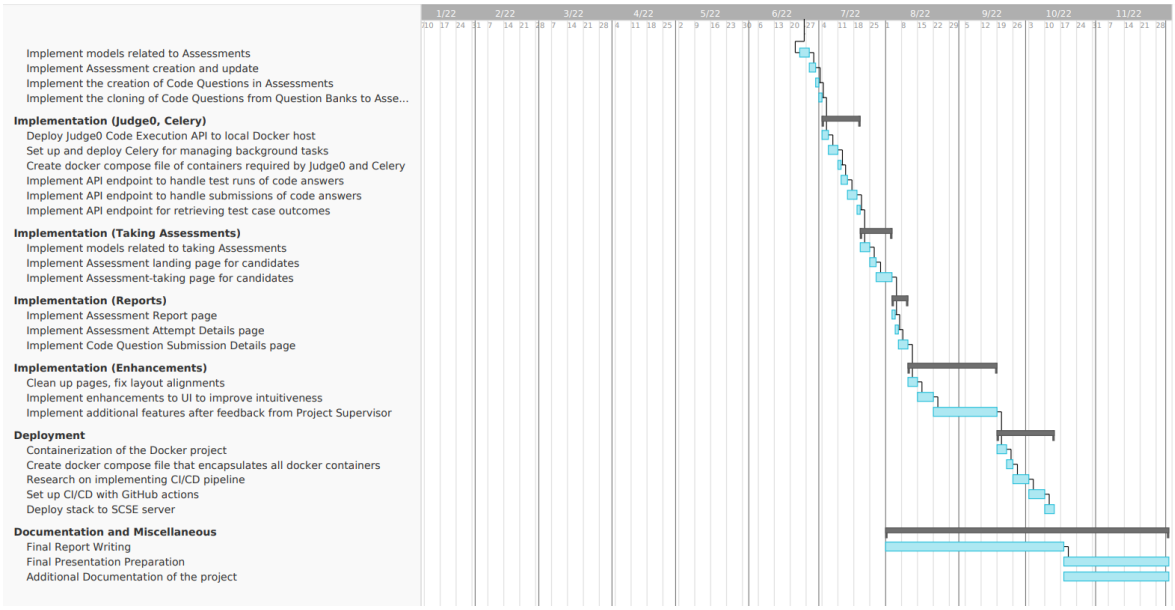

Figure 2 Gantt Chart (continued)

2 Literature Review

2.1 Prior work

In this section, we review the AASP that Soh and Yap previously developed to highlight its desirable characteristics and issues.

2.1.1 Desirable Characteristics

2.1.1.1 Code Editor

The utilisation of a code editor over a normal HTML text area in the user interface provides the user with an improved user experience when using the platform. For example, the integration of a code editor can provide helpful features such as code folding, syntax checking, syntax highlighting and automatic indentations.

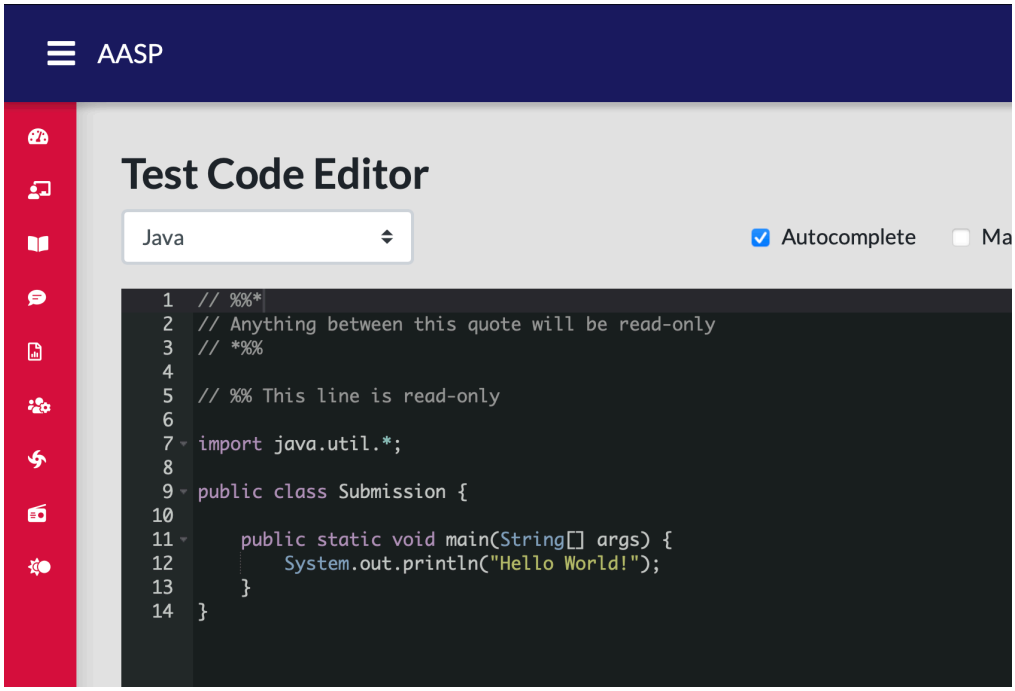

Figure 3 Page containing the ACE Code Editor

In the current AASP, the ACE Code Editor is integrated across pages that accept the input of code (Figure 3). The ACE Code Editor is an embedded code editor

written in JavaScript. It is performant and comparable to native editors like Sublime and Vim [5].

2.1.1.2 Bulk Creation of User Accounts

Online assessment platforms will potentially be used by a massive number of students who are enrolled in one or more courses. It will take an abysmally long period of time if these student accounts need to be created manually by an administrator, one at a time. Thus, it is vital that the platform provides a means of creating these student accounts in bulk, such as by importing student details contained in a file.

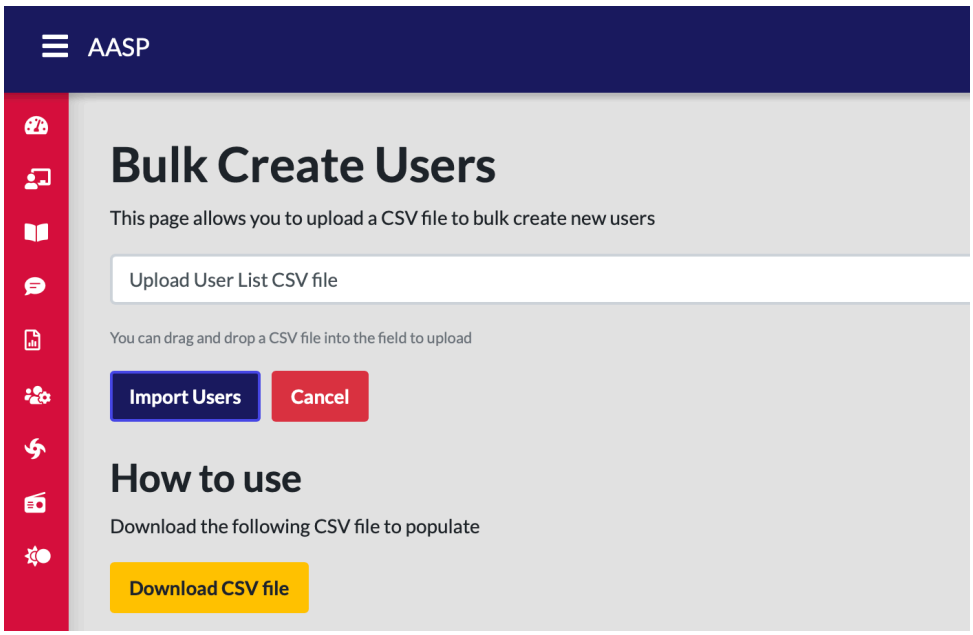

Figure 4 Bulk Create Users Page

The current AASP provides this feature by allowing administrators to import users from a CSV file with a predefined format (Figure 4). For users who are less savvy and to reduce the likelihood of human error, the platform also provides a template file for users to populate directly.

2.1.1.3 Containerisation

The current technology stack of the AASP utilises containerisation technologies to enable easy deployment and possible horizontal scaling of various parts of the platform in the future. The containerisation technology used is the Docker Engine.

2.1.1.4 Continuous Integration and Continuous Delivery (CI/CD)

The predecessors of the project implemented a Continuous Integration and Continuous Delivery (CI/CD) pipeline between the GitLab repository and the SCSE server. The pipeline is triggered when a commit is pushed to a configured repository branch.

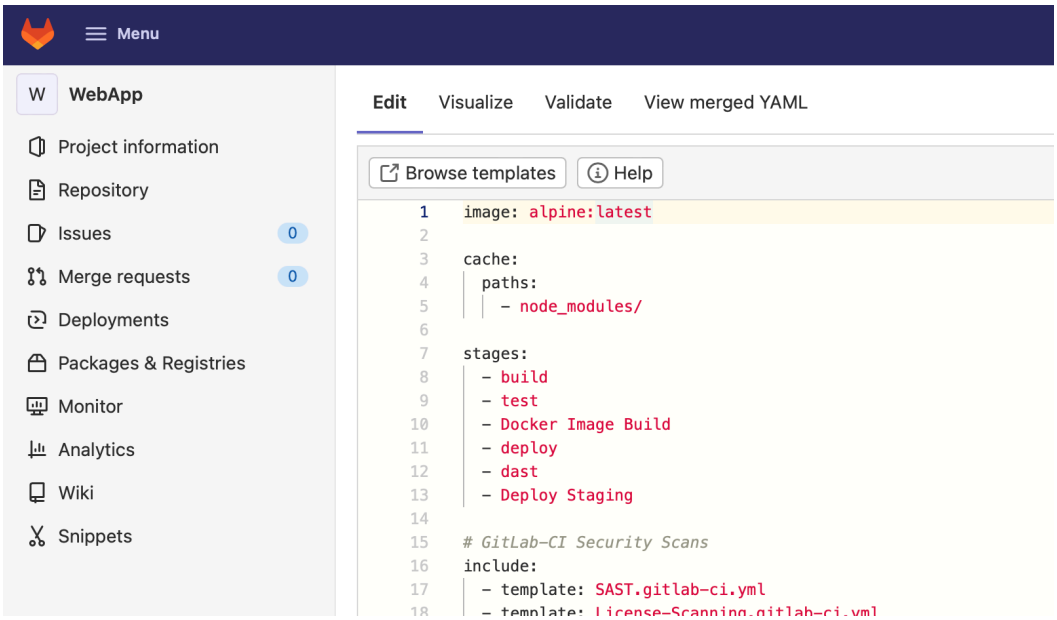

Figure 5 AASP CI/CD configuration (in GitLab)

The configuration contains operations such as running tests, building Docker images and running commands on the SCSE server to pull the updated images (Figure 5). This streamlines the process of updating the platform without manually redeploying the containers after each update.

2.1.2 Issues

2.1.2.1 Maintainability

The current AASP was built on frameworks designed to run on the Node.js engine, which runs JavaScript code. This may present a challenge to future project successors as the JavaScript language is typically not taught in educational institutions as part of a Computer Science (CS) curriculum.

In addition, the Pug Templating Engine selected to generate the frontend has a unique syntax that is very different from regular HTML. This can be observed from example code snippets from Pug's documentation below.

A snippet of a pug template:

```
doctype html
html(lang="en")
  head
    title= pageTitle
    script(type='text/javascript').
      if (foo) bar(1 + 5);
  body
    h1 Pug - node template engine
    #container.col
      if youAreUsingPug
        p You are amazing
      else
        p Get on it!
      p.
        Pug is a terse and simple templating language with a
        strong focus on performance and powerful features.
```

HTML generated from the Pug snippet above:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Pug</title>
    <script type="text/javascript">
      if (foo) bar(1 + 5);
    </script>
  </head>
  <body>
    <h1>Pug - node template engine</h1>
    <div id="container" class="col">
      <p>You are amazing</p>
      <p>
        Pug is a terse and simple templating language with a strong focus on
        performance and powerful features.
      </p>
    </div>
  </body>
</html>
```

Furthermore, the containers responsible for secure code execution were forked off inactive and now depreciated projects [6][7]. Utilising these projects may be dangerous as there may be bugs or security vulnerabilities that remain undiscovered due to their lack of utilisation by other developers. Also, these code execution services are written in the Go programming language, which is also not commonly taught in CS programmes.

These details may seem insignificant at first, but upon further cogitation, it can be recognised that the Maintainability of the project would be impacted. Moreover, by utilising technologies based on lesser-known programming languages and frameworks or those that require learning a special syntax, successors of the project are confronted with a steep learning curve.

A possible approach to ease the learning curve would be to utilise programming languages and frameworks that are more commonly taught, such as Python. Furthermore, it would be beneficial to integrate existing open-source projects where large communities of active users and maintainers keep the projects updated.

2.1.2.2 Security

While good coding practices, such as the use of prepared statements, were observed, other aspects, such as input sanitisation, were not adequately enforced. During an initial exploration of the platform, a cross-site scripting vulnerability, also known as XSS, was discovered.

Cross-site scripting is a type of injection attack where malicious scripts are injected into otherwise harmless and legitimate websites [8]. It occurs when an attacker uses a vulnerable web application to send malicious code, usually browser-side scripts, to other unsuspecting web application users. The vulnerability makes it possible for attackers to perform malicious actions such as cookie theft, redirecting the user to

another website (e.g., to conduct phishing attacks), or running arbitrary JavaScript code on the victim's browser.

The type of XSS vulnerability discovered on the platform is stored XSS, where the malicious payload is persistent and stored in the web application. Any user who navigates to the affected pages will be served with the malicious payload. Figures 6 and 7 demonstrate the XSS vulnerability by displaying a benign JavaScript alert in the browser. The XSS payload can be provided as the name of a question, and it will be injected when a user visits the *Question Bank* page containing the question.

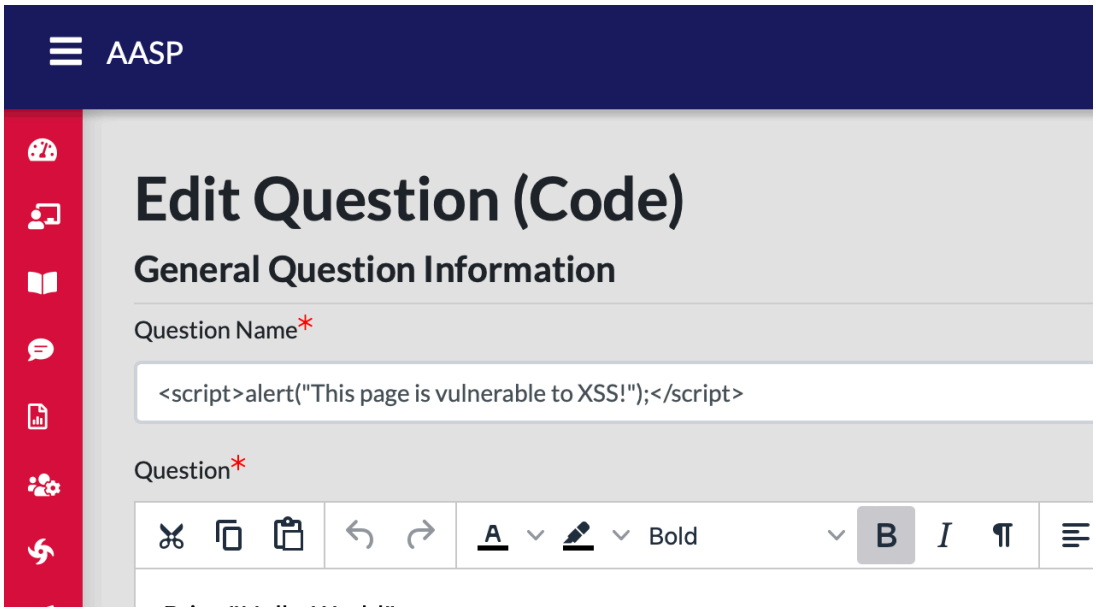

Figure 6 XSS payload Example

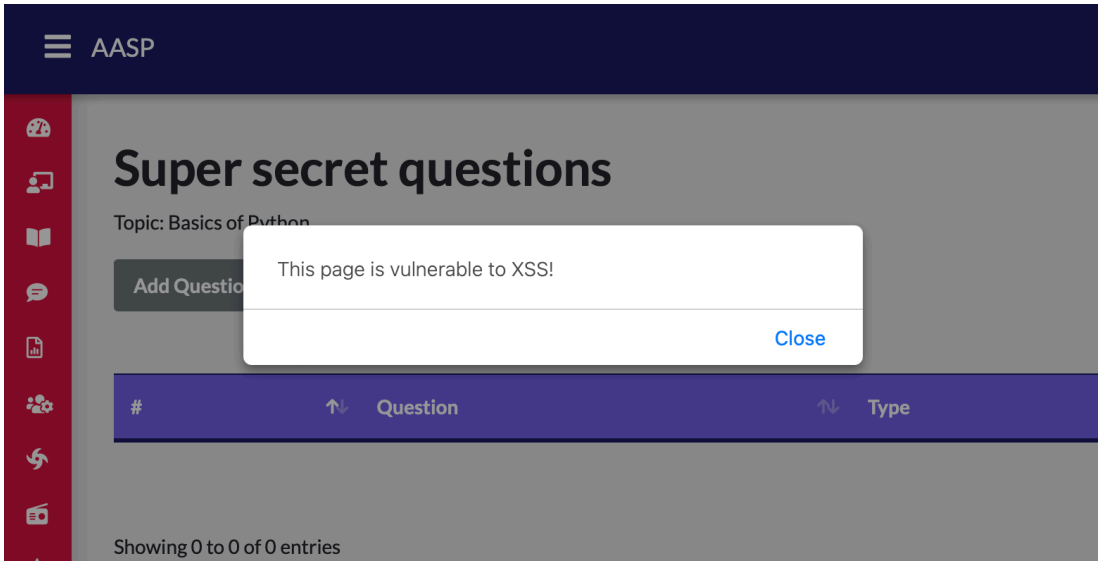

Figure 7 Stored-XSS payload executed

2.1.2.3 Reliability

The current version of the AASP lacks reliability due to the presence of bugs across multiple areas of the platform.

During normal usage of the platform, some pages would not load correctly and would require multiple page refreshes for the elements to be loaded. In some cases, when presented with unexpected inputs, the bugs could crash the entire web application.

For example, a bug happens when an HTML image tag with an invalid *src* attribute is provided as a question name. Upon visiting the *Question Bank* page containing the question, the web application will completely crash and exit. This results in the platform being not accessible for some time until the container restarts itself. Malicious actors could thus easily leverage this attack vector to perform a denial-of-service (DDoS) attack on the platform, denying access to legitimate end-users.

2.2 Code Evaluation

For the grading of assessment results to be performed automatically, the system must have the means to analyse the submitted code and determine its correctness automatically. There are two main approaches to evaluating code answers in automated assessment platforms; dynamic and static [9].

The dynamic analysis approach involves first compiling the code and subsequently executing the program against predefined test cases set by the educator. These test cases consist of two main parts; input and expected output. The input is the data that will be passed to the program during execution, while the expected output refers to the output that the program must generate for the test case to be passed. In addition, each test case may contain additional constraints, such as execution time or memory limits. This approach allows for determining both the correctness of the output and the efficiency of the code.

On the other hand, the static analysis approach employs techniques to gather information from the code without having to execute it [10]. Some examples of these techniques include verifying the syntactic and semantic correctness, measuring the software metrics such as the number of lines and detecting code redundancies. This approach allows for measuring the quality of a program but often cannot certify the functionality of a code.

Both approaches undoubtedly have their strengths and weaknesses. Still, as a programming solution's correctness and efficiency are critical criteria in code evaluation, it can only be guaranteed by the dynamic approach. Popular online assessment platforms such as HackerEarth and HackerRank utilise the dynamic approach, where correctness is determined solely by the program’s outputs and time-space complexity [11].

2.3 Code Execution Systems

A code execution system is the backbone of all assessment platforms. While automatic assessment platforms are concerned with interacting with the various end-users to create and administer assessments, the code execution system is solely concerned with compiling, executing and returning the execution results. The critical requirements of the system include; sandboxing and security, robustness, dynamic assessment of code and scalability [12].

A. Sandboxing and security. As the system is expected to run arbitrary code from users that could potentially be malicious, it must be compiled and executed in a sandboxed environment. This prevents malicious code from being able to manipulate resources that are outside the bounds of the sandbox it is contained in. In addition, other measures should also be put in place. For example, long-running code, such as ones with infinite loops, must be terminated once a predefined limit has been reached. This ensures that no single submission is allowed to hog resources indefinitely, preventing new submissions from being processed.

B. Robustness. The system must be able to recover from internal errors, and malicious attempts from an adversary.

C. Dynamic assessment of code. The system should be able to dynamically assess the submitted codes using metrics such as functional correctness, execution time and memory consumption (time and space complexity).

D. Scalability. The system must be scalable to support a potentially large number of concurrent users. This is especially important for platforms where many users are expected to access the platform within a short timeframe.

2.4 Related Platforms

In this section, we explore other related products available on the market and highlight the desirable characteristics that these platforms possess. In particular, we will look at three popular products: HackerRank, HackerEarth and LeetCode. HackerEarth and HackerRank are online platforms where consumers can practice solving programming challenges, and businesses can use to source, screen and hire engineers [13][14]. LeetCode similarly offers programming challenges for practice but also allows businesses to sponsor contests to identify top talent, amongst other services [15].

2.4.1 User Interface

2.4.1.1 Horizontally split panes

When displaying code-based questions to the candidates, two sections are generally the most important. The first section is the question; it contains a brief description, sample inputs and their corresponding outputs. The second section is the code editor, which the candidate will use to construct their solution to the problem and view its outcome.

After inspecting and experiencing the user interfaces of several platforms, the layout that provided the best user experience and intuitiveness was a layout with horizontally split panes for the two sections. The sizes of the left and right panes are resizable to suit every user's preference. Figures 8 and 9 show the user interfaces of HackerRank and LeetCode, respectively.

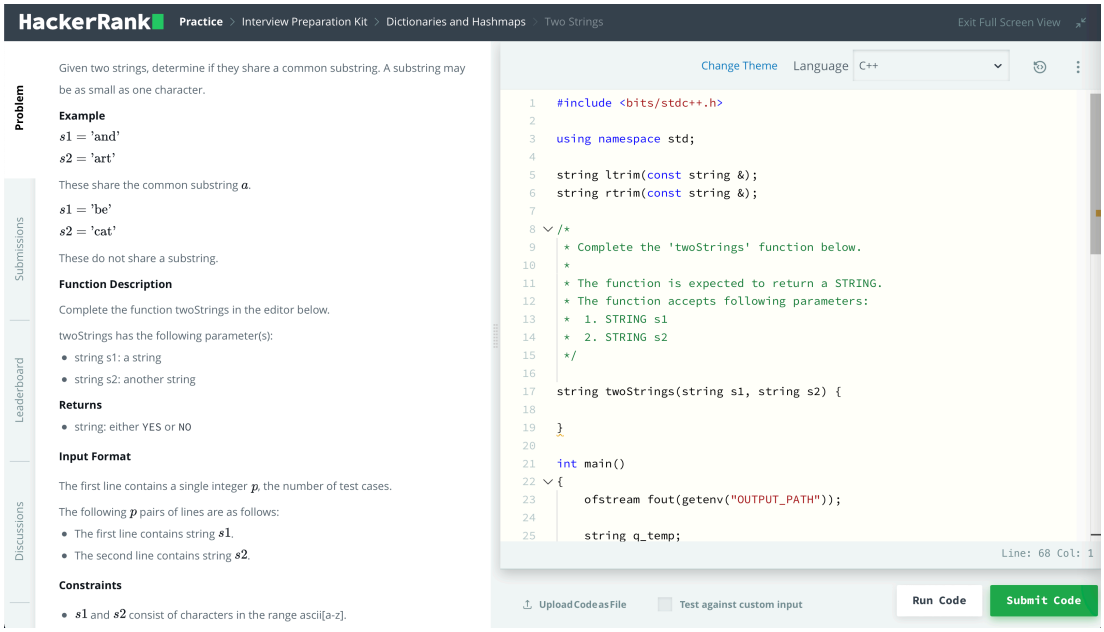

Figure 8 Horizontally split layout of HackerRank

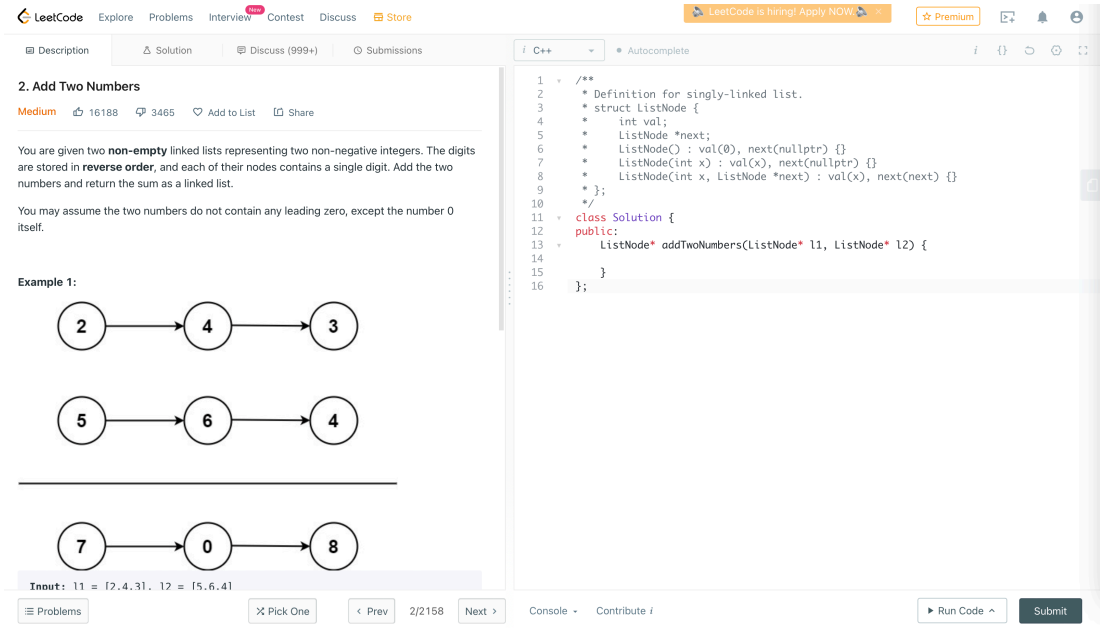

Figure 9 Horizontally split layout of LeetCode

The horizontally split layout is superior to the traditional layout of having the question at the top, followed by the code editor at the bottom. The traditional layout leads to a poor user experience as candidates will be forced to scroll to the top of the page to view the question and then scroll down again to resume working on their solution.

2.4.1.2 Text formatting

The flexibility of text formatting is also an essential feature. When displaying questions, regular text formatting such as font size, font weight, italics and underlining are insufficient.

For a question to be clearly presented, it should contain headings for different sections and clear formatting of different types of components. For example, code snippets should be presented in a monospaced font, and mathematical expressions should be rendered correctly and not displayed in a plain-text format (Figure 10).

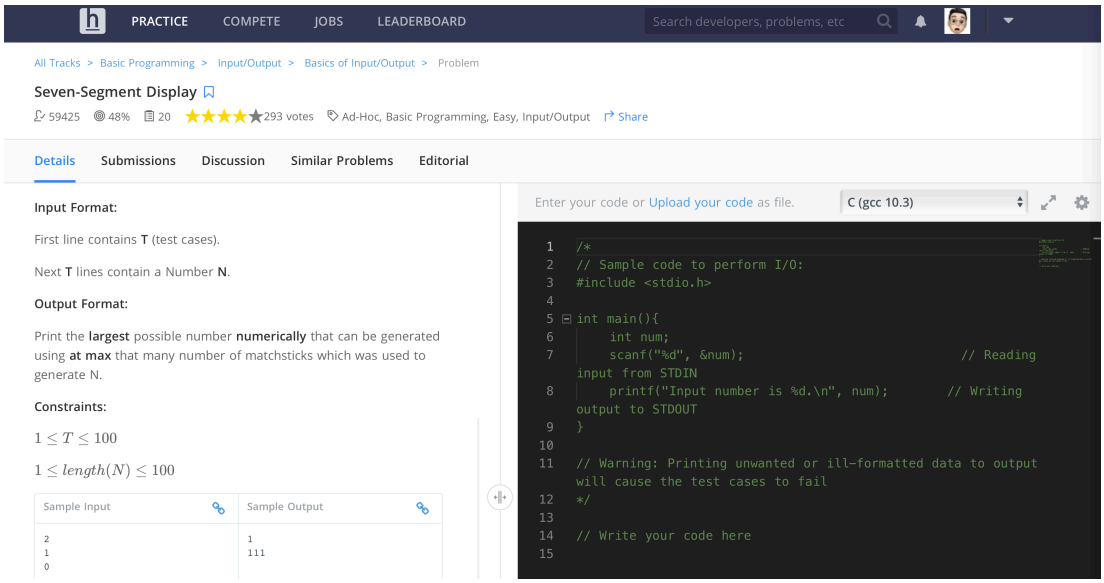

Figure 10 Various formatting components in the question details of HackerEarth

Support for Markdown with a math extension or plugin could be added to achieve this flexibility for text formatting. The Markdown format utilises a simple syntax and can be easily transformed into a wide variety of formats, including but not limited to HTML [16]. Alternatively, support for LaTeX Markup could be added as well.

2.4.2 Useful features

2.4.2.1 Run Example Test Case before Submission

These platforms offer a feature where users can quickly test their code by compiling and running it against only the sample test case. This allows users to quickly determine if their code contains syntactic or semantic errors and rectify them. After the user is satisfied with the results of the sample test case, the user can submit the answer as a complete submission, where it will be compiled and executed against the complete list of test cases.

This helps to reduce the time required for users to verify their codes while simultaneously reducing the computational load and avoiding wasted processing time on the backend server.

2.4.2.2 Submission History

The platforms store all submissions from a user, regardless of the outcomes of the test cases, and presents the user with the submission history (Figure 11). The ability to view past submissions is helpful for candidates to understand what may have gone wrong and subsequently iterate towards a correct solution.

Figure 11 A table containing the submission history for a question (HackerRank)

These platforms usually also allow users to view the code submitted in previous attempts (Figure 12).

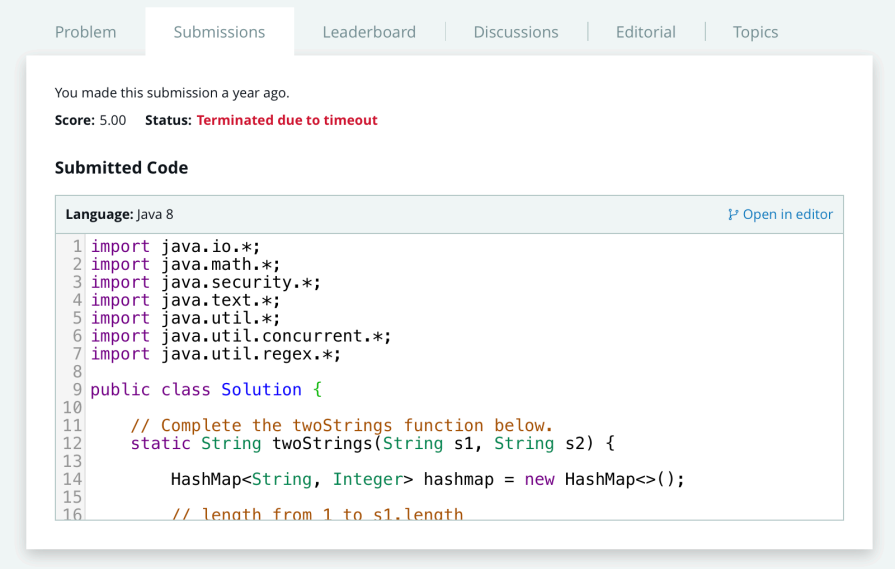

Figure 12 Viewing submitted source code (HackerRank)

3 Design Methodology

3.1 Use Case Diagram

To begin designing a system, we must first model the key use cases of the various actors involved in the AASP. Figure 13 represents the use case diagram of the AASP.

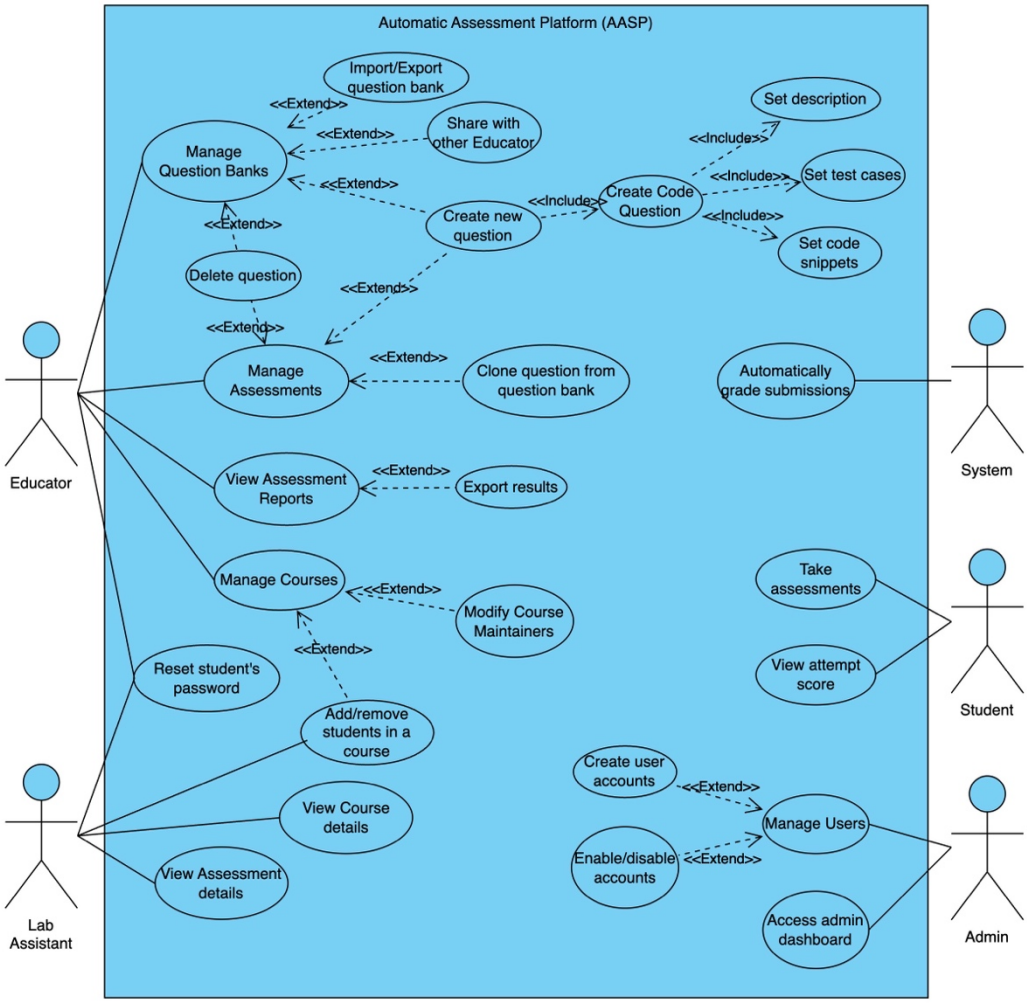

Figure 13 Use Case Diagram

3.2 Functional Requirements

3.2.1 General

1. The system must implement user authentication.
    1.1. Users must be able to log in.
    1.2. Users must be able to log out.
    1.3. The system must require users to log in.
    1.4. Each user account can only be logged into from one device at a time.
2. The system must redirect users to the correct dashboard during login.
    2.1. Educators must be redirected to the Educator dashboard.
    2.2. Lab Assistants must be redirected to the Lab Assistant dashboard
    2.3. Students must be redirected to the Student dashboard
3. The system must be able to automatically grade Assessment attempts.
4. The system must enforce strict authorisation procedures to ensure only authorised users are granted access to resources.

3.2.2 Educators

1. Educators must be able to manage Code Questions.
    1.1. Educators must be able to create, update and delete Code Questions.
    1.2. Educators must be able to manage Test Cases in Code Questions.
        1.2.1. Educators must be able to create Test Cases in a Code Question.
        1.2.2. Educators must be able to update Test Cases in a Code Question.
        1.2.3. Educators must be able to delete Test Cases from a Code Question.
    1.3. Educators must be able to manage Code Snippets in a Code Question.
        1.3.1. Educators must be able to create Code Snippets in a Code Question.
        1.3.2. Educators must be able to update Code Snippets in a Code Question.
        1.3.3. Educators must be able to delete Code Snippets from a Code Question.
2. Educators must be able to manage Question Banks.
    2.1. Educators must be able to create, update and delete Question Banks.

2.2. Educators must be able to manage questions in a Question Bank that belongs to them.

2.2.1. Educators must be able to create questions.

2.2.2. Educators must be able to update questions.

2.2.3. Educators must be able to delete questions.

2.3. Educators must be able to manage read-only permissions of the Question Bank that belongs to them.

2.3.1. Educators must be able to grant read-only permissions to other Educators.

2.3.2. The owner must be able to revoke read-only permissions.

2.4. Educators must be able to export Question Banks and its Questions as a JSON-formatted file.

2.5. Educators must be able to import Question Banks from a JSON-formatted file.

3. Educators must be able to manage Courses.

3.1. Educators must be able to create, update and delete Courses.

3.2. Educators must be able to designate other Educators or Lab Assistants to the course as Maintainers.

4. Educators must be able to manage students of a Course where they are the owner or maintainer.

4.1. Educators must be able to enrol a single student.

4.2. Educators must be able to enrol students in bulk by uploading a CSV-formatted file.

4.3. Educators must be able to remove students from the Course.

4.4. Educators must be able to view a list of students enrolled in the course.

4.4.1. Educators must be able to filter the students list.

4.4.2. Educators must be able to export the students list as a CSV-formatted file.

5. Educators must be able to manage Assessments of a Course where they are the owner or maintainer.
    5.1. Educators must be able to create, update and delete Assessments.
    5.2. Educators must be able to manage questions in an Assessment.
        5.2.1. Educators must be able to clone questions from Question Banks.
        5.2.2. Educators must be able to create new questions.
        5.2.3. Educators must be able to update questions.
        5.2.4. Educators must be able to delete questions.
    5.3. Educators must be able to preview Assessments before they are published.
    5.4. Educators must be able to publish Assessments.
6. Educators must be able to view Assessment Reports of a Course where they are the owner or maintainer.
    6.1. Educators must be able to view completed submissions.
        6.1.1. Educators must be able to view the submissions for each question of the Assessment.
        6.1.2. Educator must be able to view the outcome of each test case of a submission.
    6.2. Educators must be able to view ongoing attempts.
    6.3. Educators must be able to export all attempts as a CSV-formatted file.

3.2.3 Lab Assistants

1. Lab Assistants must be able to view details of a Course where they are designated as a maintainer.
2. Lab Assistants must be able to manage students of a Course where they are designated as a maintainer.
    2.1. Lab Assistants must be able to enrol a single student.
    2.2. Lab Assistants must be able to enrol students in bulk by uploading a CSV-formatted file.
    2.3. Lab Assistants must be able to remove students from the Course.

2.4. Lab Assistants must be able to view a list of students enrolled in the course.

2.4.1. Lab Assistants must be able to filter the students list.

2.4.2. Lab Assistants must be able to export the students list as a CSV-formatted file.

3. Lab Assistants must be able to view details of Assessments of a Course where they are designated as a maintainer.

3.2.4 Students

1. Students must be able to view the Assessments available to them.
   1.1. Students must be able to view the details of an Assessment.
   1.2. Students must be able to view the status of an Assessment.
   1.3. Students must be able to view scores of a completed Assessment attempt.
2. Students must be able to take an Assessment in the system.
   2.1. Students must be able to start an attempt if it is within the stipulated time period.
   2.2. Students must be able to view the remaining time for the attempt.
   2.3. Students must be able to test their code on the sample test case.
   2.4. Students must be able to submit their code to be run against the test cases for grading.
   2.5. Students must be able to navigate between different questions of an Assessment.
   2.6. Students must be able to submit an Assessment attempt.

3.3 Non-functional Requirements

1. The System should be able to support at least 600 users concurrently.
2. The System should be able to log in a user within 5 seconds.
3. The System should be deployable on any server running a Linux operating system.
4. The System should be supported by any modern web browser.

3.4 Activity Diagrams

This section presents the activity diagrams of key activities that the various user groups are concerned with.

3.4.1 Educators

To create a new question bank, Educators can do so by visiting the Question Banks page from the dashboard (Figure 14).

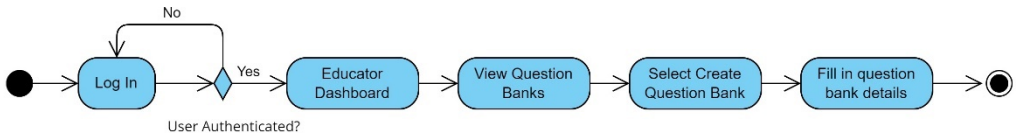

Figure 14 Activity Diagram for Create Question Bank

To create a question in an existing question bank, Educators can do so by visiting the target question bank's page (Figure 15).

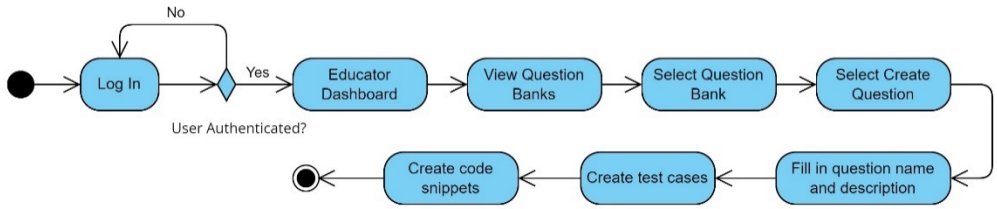

Figure 15 Activity Diagram for Create Question in Question Bank

To create a new course, Educators can do so by visiting the Courses page from the dashboard (Figure 16).

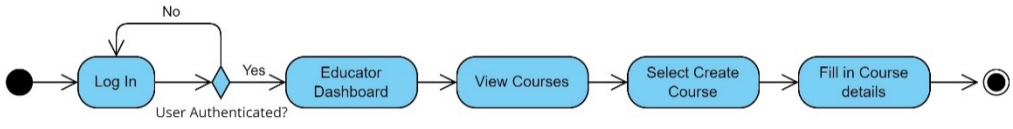

Figure 16 Activity Diagram for Create Course

There are two options for enrolling students into courses. The Educator/Lab Assistant can either enrol a single student or in bulk by uploading a CSV file (Figure 17).

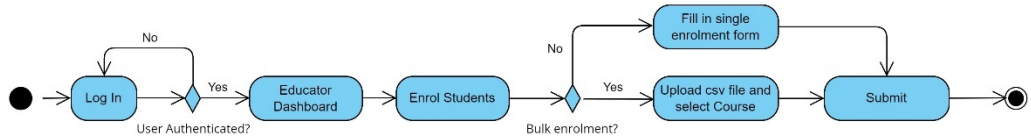

Figure 17 Activity Diagram for Enrol Students to Course

To create an assessment, Educators can browse to the target course page and select the create assessment button (Figure 18).

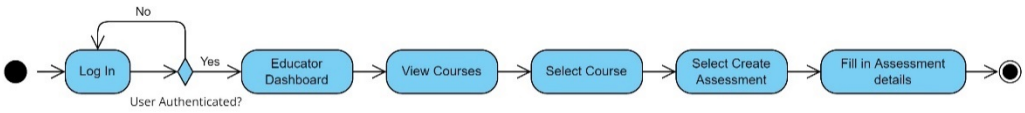

Figure 18 Activity Diagram for Create Assessment

3.4.1.1 Add Questions to Assessment

The Educator have two options for adding questions to an Assessment. Questions can either be created from scratch or cloned from an existing Question Bank that the account has access to (Figure 19).

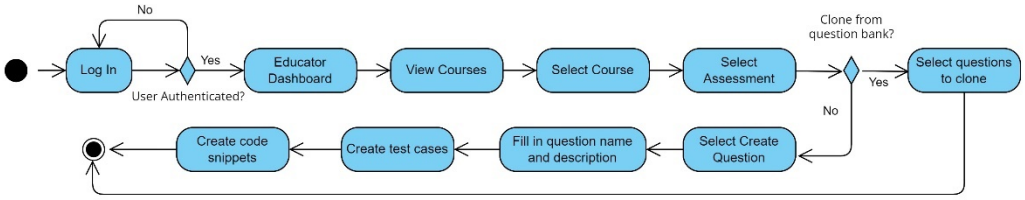

Figure 19 Activity Diagram for Add Questions to Assessment

3.4.2 Lab Assistants

Lab Assistants would routinely need to view the details of an assessment. These details include the availability period, instructions, and more importantly the access PIN that candidates would need in order to start a new attempt. The Lab Assistant may view the details of an assessment by selecting it directly from the dashboard (Figure 20).

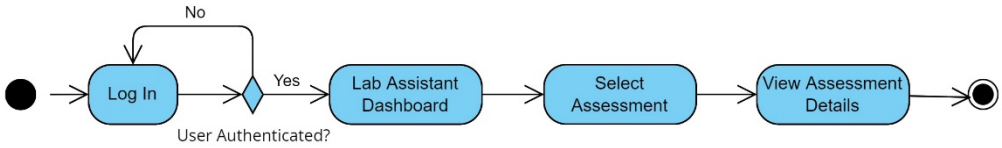

Figure 20 Activity Diagram for View Assessment Details

The Lab Assistant can reset the password of a student that is currently enrolled in a course where the account is a Maintainer. This activity also applies to Educators, although this would primarily be the Lab Assistant's task. This can be done by browsing to the course page where the target student is enrolled (Figure 21).

Figure 21 Activity Diagram for Reset Student Password

3.4.3 Students

To attempt an assessment, the student can view a list of available assessments from the dashboard and select the desired assessment to attempt. This will bring the student to a landing page where assessment details, such as the instructions, are shown. If the assessment is not within the availability period, the student will not be able to start an attempt. If there is an ongoing attempt, the student can resume it.

If the assessment is protected by a PIN, the candidate must supply a correct PIN to begin a new attempt (Figure 22).

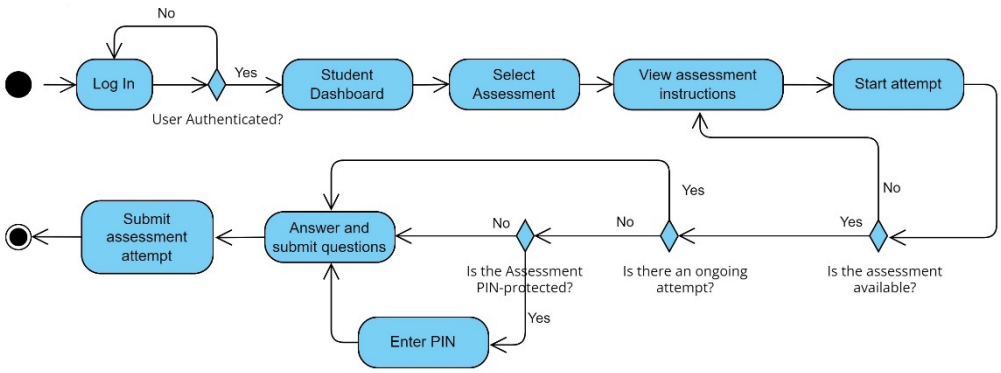

Figure 22 Activity Diagram for Taking Assessments

3.4.4 All Roles

All logged-in users will have the ability to change their account passwords. This requires the users to know the current password and must be entered in the password change form for verification. The user can access the password change page from the dashboard (Figure 23).

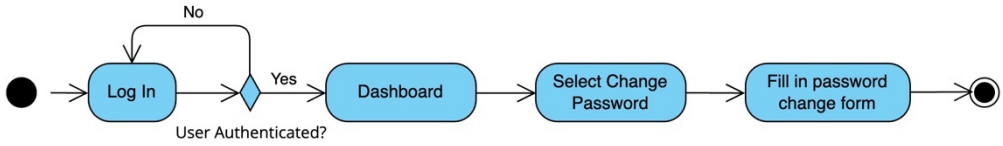

Figure 23 Activity Diagram for Change Password

3.5 User Interface

This section will showcase the initial wireframe designs of the various key pages of the AASP. The interfaces are intentionally kept simple, and unnecessary elements and complications are avoided. Consistency of the layout and design is also enforced throughout all pages of the platform.

3.5.1 Dashboards

Depending on the user's role (educator, lab assistant or student), the user will be redirected to a different dashboard upon login. This feature is transparent to the user and ensures that the dashboard is not simply generic. Instead, each user role will be presented with only the information and features that are the most relevant to them. Figures 24 and 25 are the wireframe designs for the Educator and Student dashboards, respectively.

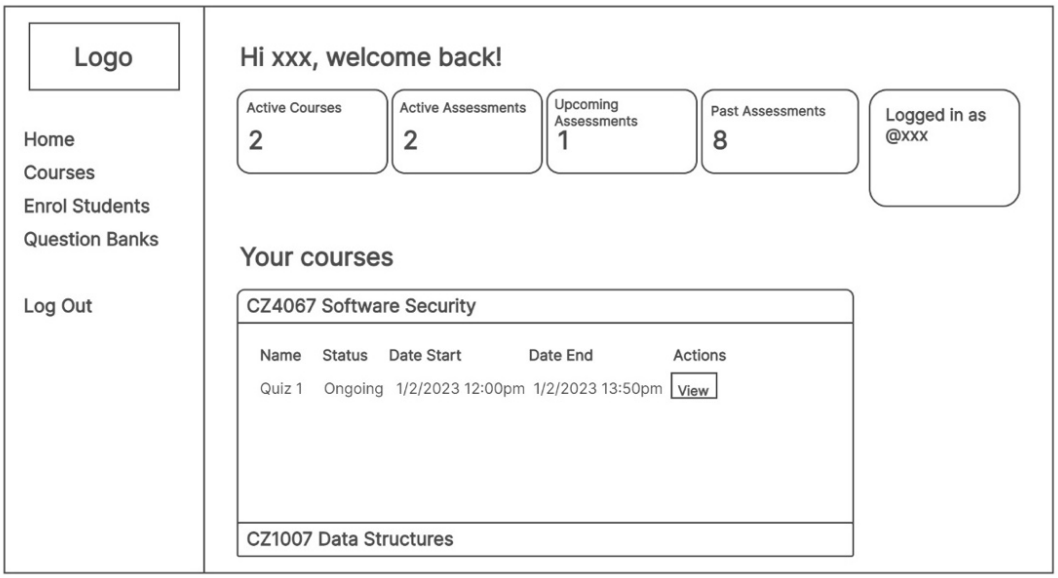

Figure 24 Dashboard for Educators (Wireframe)

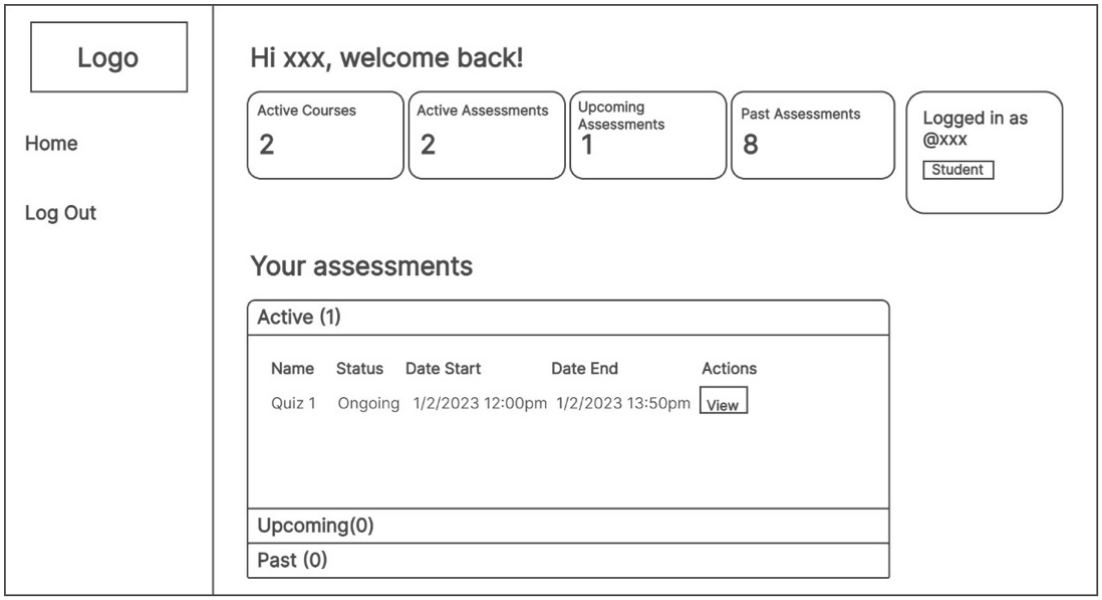

Figure 25 Dashboard for Students (Wireframe)

3.5.2 Assessment Attempt Page

The assessment attempt page will follow the excellent design of other related platforms, specifically the horizontally split panes (Figure 26). The horizontally split layout with resizable panes for the question and code areas allows students to refer to the question description without having to scroll up and down repeatedly.

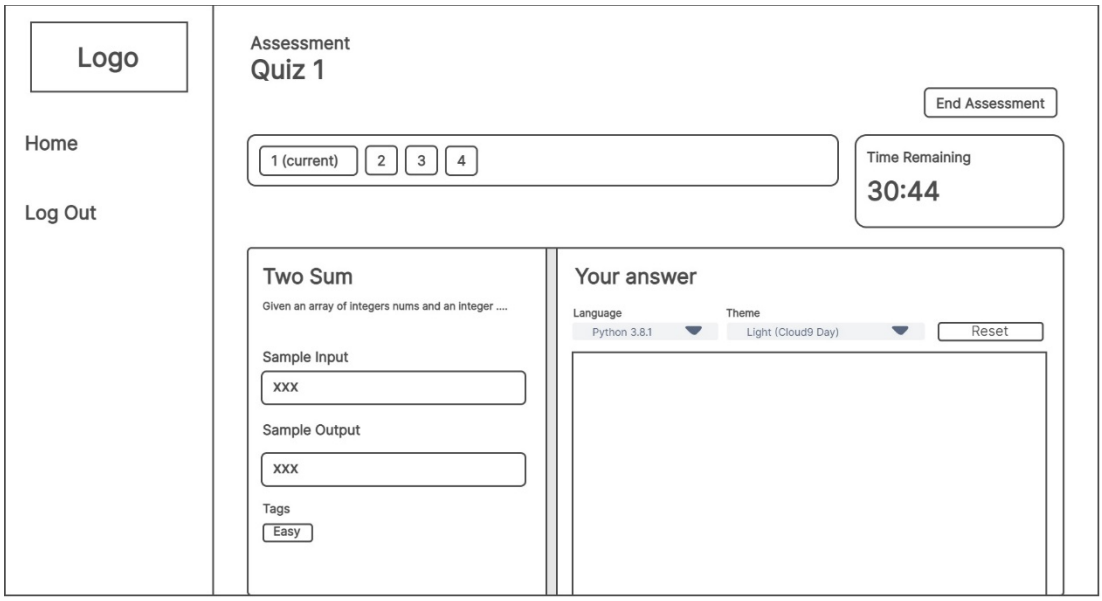

Figure 26 Assessment attempt page (Wireframe)

The question number buttons at the top of the page will also be colour-coded to reflect the question's state. For example, the button will be green if the corresponding question has been attempted and grey if it has not been attempted.

3.6 Database

The following is the Entity Relationship Diagram (ERD) of the database schema (Figure 27).

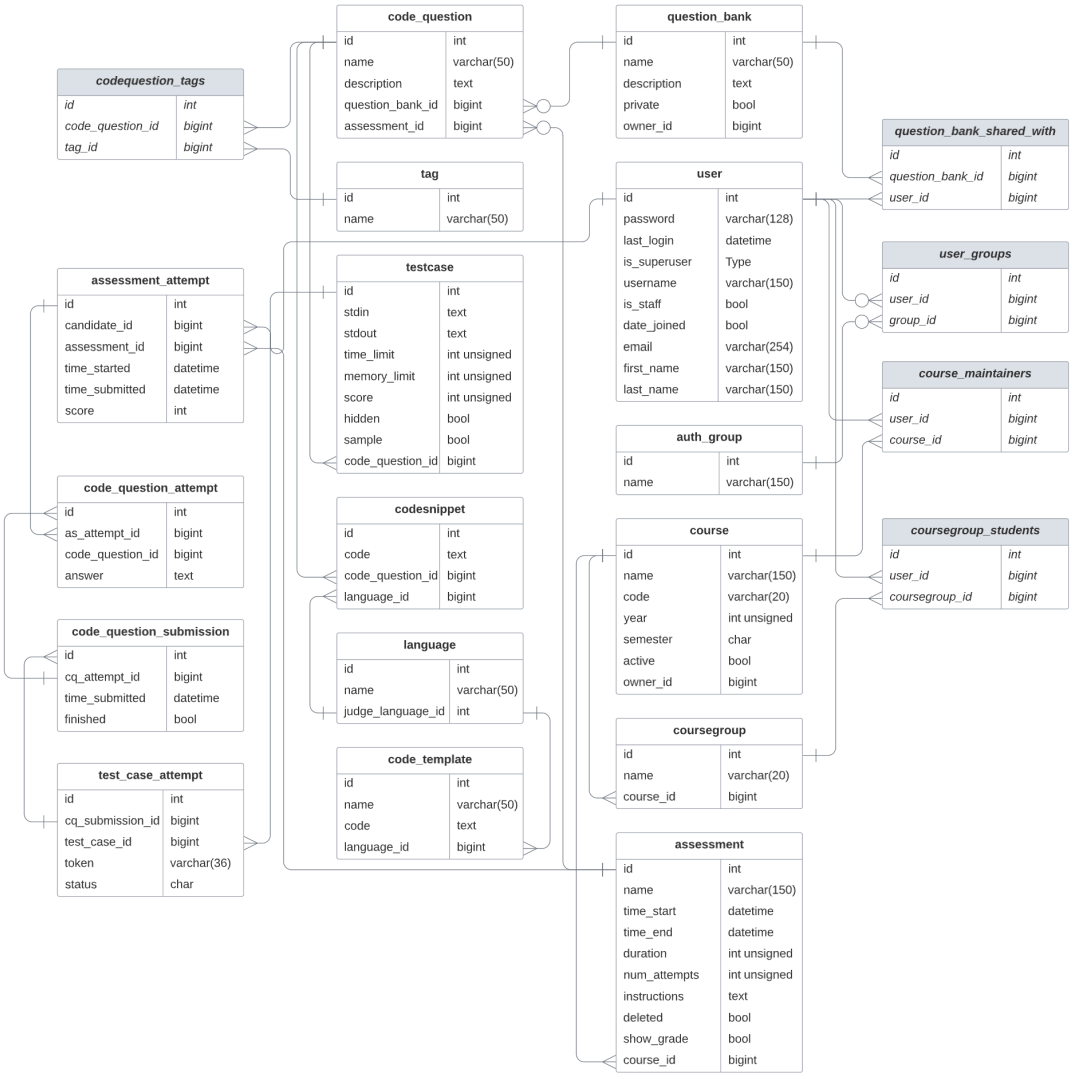

Figure 27 Entity Relationship Diagram

3.7 Access Control

Access control is critical for a multi-user system such as the AASP. Therefore, it is important for us to enforce strict authentication and authorisation procedures that will only permit access to resources that a particular user is authorised to access.

The access control model of the AASP consists of three levels:

1. User authentication
2. Group-level permissions
3. User-level permissions

Firstly, any user of the AASP must be authenticated via the login page before any other page may be accessed. If an attempt is made to visit another page, the user will be redirected back to the login page for user authentication.

Secondly, different pages and features of the AASP will only be available after verifying the user's group membership. For example, only users from the Educator group are authorised to access pages related to Question Banks. Any violation will result in a redirection to an appropriate page with a corresponding message.

Finally, access control at the user-level is also implemented before certain resources and actions can be accessed. For example, only the owner of a question bank is allowed to modify the question bank. Similarly, any violation will result in a redirection to an appropriate page with a corresponding message.

3.8 Security Considerations

3.8.1 Network Exposure

The AASP stack consists of several services such as a web server, web application, databases and message queues. To reduce the attack surface of the system, these services should only be exposed on the network if it is strictly required for end-users to realise the system's functionalities.

As the end-users of the AASP will only interact with the system via the web browser, only the web server and its related ports (i.e., 80 and 443) should be exposed to the host network and allowed through the firewall. When using Docker as the container engine, other services housed in separate containers should be configured to communicate with each other via a separate Docker network that is isolated from the host network.

3.8.2 Transport Layer Security

The Transport Layer Security protocol (TLS) encrypts data that is sent over a network or the internet. To prevent malicious actors from being able to intercept and view confidential information such as passwords, it is important for TLS to be configured on the webserver. This ensures that communications between end-users and the server are encrypted and thus not easily susceptible to eavesdropping or man-in-the-middle attacks.

3.8.3 Web Application Security

If other aspects of a system have been secured, the web application itself is the only attack vector remaining for malicious actors to attempt to exploit to compromise a system. Thus, it is critical for the web application to be secure, robust and free from vulnerabilities.

Taking into consideration that the past, present and future successors of this project are undergraduates, and likely are not highly proficient in both full-stack development and cyber security, a good approach would be to select a mature, "batteries-included" and web framework that would make it less likely for developers to unknowingly introduce security vulnerabilities to the application.

For example, Django ships with many security features that protect the application from common attacks, such as cross-site scripting (XSS), cross-site request forgery (CSRF), and SQL injection (SQLi). These security features are enabled by default, and the developers must set explicit configurations to disable any of the features. The related sections from the Django documentation are also highlighted to caution developers of the possible consequences of the action.

3.9 Maintainability Considerations

As the project will likely be handed over multiple times to future successors, it is paramount that maintainability is considered and maximised wherever possible. To achieve this, there are several core aspects to consider.

3.9.1 Primary Programming Language

The selection of the primary programming language is crucial. Ideally, it should be commonly taught and widely known to reduce the steepness of the learning curve while also not compromising excessively on performance.

3.9.2 Framework Selection

The selection of frameworks is also critical. Different frameworks are designed with varying sets of guiding principles that result in unique properties that make each framework stand out from its counterparts. For example, some frameworks prioritise flexibility, while other frameworks are opinionated. Although flexibility may sound universally desirable, it is usually more suited for experienced developers who

possess great experience and know exactly how to leverage the flexibility that a framework provides.

In the context of this project, it would be more apt to select an opinionated framework that reduces the number of decisions the developer would have to make in terms of the design or structure of the project. This minimises decision fatigue on various aspects of a project, such as defining a project structure, user authentication method, permissions model, and more.

3.9.3 Simplicity

The system architecture should be designed to be as simple as possible while not compromising on the desired requirements. One means of reducing complexity is to integrate existing open-source projects when it is suitable. This results in having to write less code, and subsequently less code needs to be maintained over time.

Another strategy we can employ to reduce the complexity of a project is containerisation. For example, a containerisation technology like Docker enables provides the ability for the project to be deployed on any operating system that supports the Docker Engine. The developer will also not be required to perform the tedious and often time-consuming task of manually installing and managing the dependencies of services that make up the project.

3.10 Reusability Considerations

The various components of this project should be designed to be reusable. For example, suppose a future project successor decides to swap out the frontend framework. In that case, the backend framework should be reusable and capable of being integrated with the new frontend. Backend frameworks like Flask and Django provide various options for server-side template rendering, but the developer also

has the option of exposing the business logic as a REST API to support the use of client-side frontend frameworks such as React and Vue.

On the other hand, if data from the platform needs to be exported, it should be made available in a universally accepted data format such as comma-separated values (CSV) or JavaScript Object Notation (JSON). This maximises the ability to reuse the exported data in other applications.

Finally, if the developer wishes to start from the ground up, the project should have clear, concise and robust documentation that the developer can study for reference. Documentation consists of components such as the project report, comments in the code and commit descriptions in the version control system. This allows the developer to understand the thought processes during the design and implementation of the project and possibly gain inspiration or avoid certain pitfalls that may arise.

4 Implementation

4.1 System Architecture

4.1.1 Overview

This section provides an overview of the system architecture of the AASP. The successive sections will then describe each component in more detail.

4.1.1.1 Diagram

Figure 28 is a diagram of the system architecture at a glance.

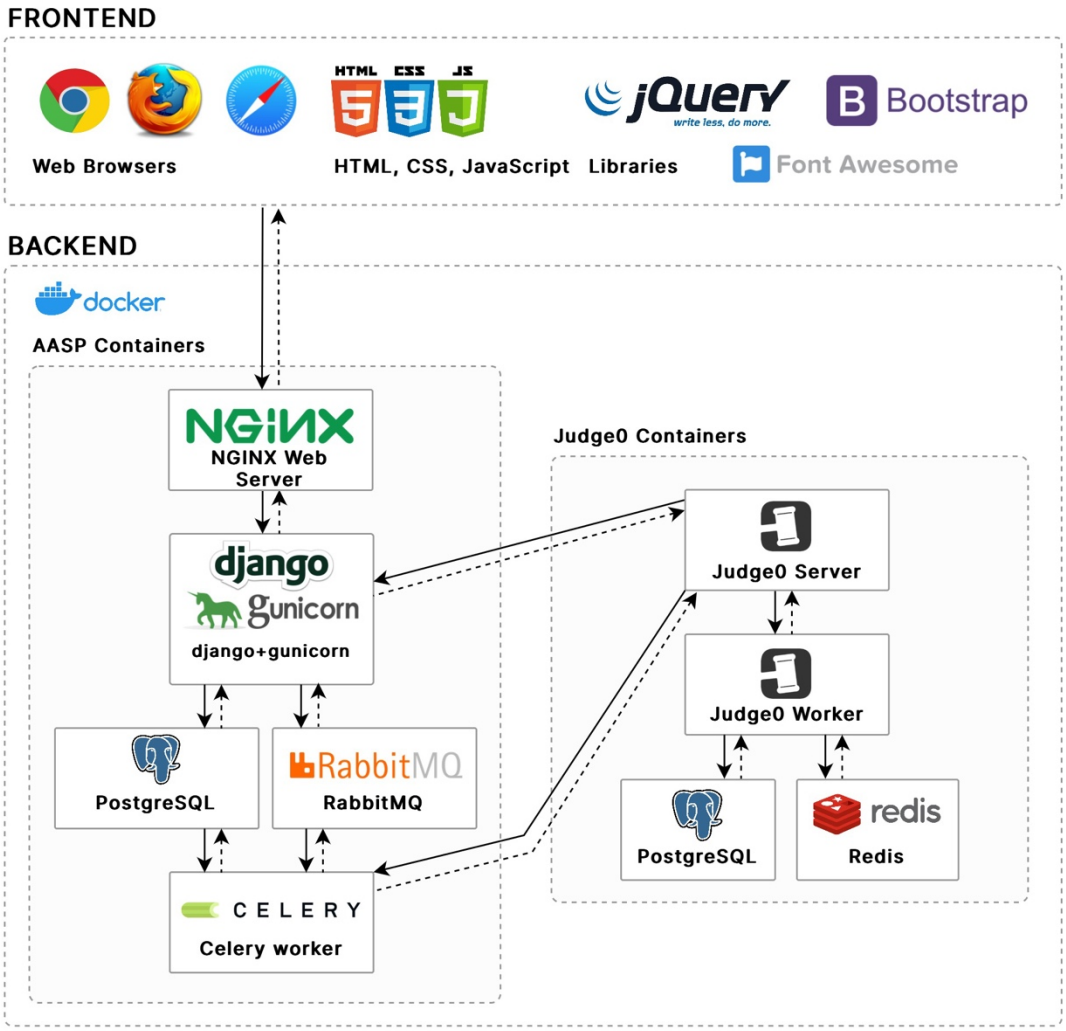

Figure 28 Architecture Diagram

4.1.1.2 Services

The AASP has been containerised into multiple containers, each responsible for providing a single service.

- **Nginx**: This container hosts the web server that serves the Django web application to end-users. The web server also serves as a reverse proxy and provides Transport Layer Security (TLS) for communications between the web application and end-users to remain secure.
- **Django, Gunicorn**: The Django web application is contained here. Gunicorn is a Python Web Server Interface Gateway (WSGI) that allows the webserver to interface and communicate with the Django web application.
- **PostgreSQL**: This container contains the database engine that is used by the AASP to persist data.
- **RabbitMQ**: This container runs RabbitMQ, which provides the message queue service for Celery. Celery allows us to queue and run long-running tasks asynchronously.
- **Celery**: This container contains the celery worker that will obtain tasks from the queues and execute the tasks.

The Judge0 open-source project also provides a containerised deployment for the system. The following is a breakdown of the containers used by Judge0.

- **Judge0 Server**: The server that servers the API endpoints that the AASP uses to interface with Judge0 for code execution services.
- **Judge0 Worker**: The Judge0 worker that receives, compiles and executes code submissions.
- **PostgreSQL**: This container contains the database engine that is used by Judge0 to persist submission data.
- **Redis**: This container runs the Redis service that provides a task queue for Judge0.

4.1.2 Containerisation with Docker

Virtualisation is a process where software is used to create an abstraction layer over computing hardware that allows hardware elements of a single computer to be shared among multiple virtual computers. The most common form of virtualisation is the use of virtual machines, where each virtual machine encapsulates a complete guest operating system. Containerisation, on the other hand, is a form of virtualisation that operates at the operating system level.

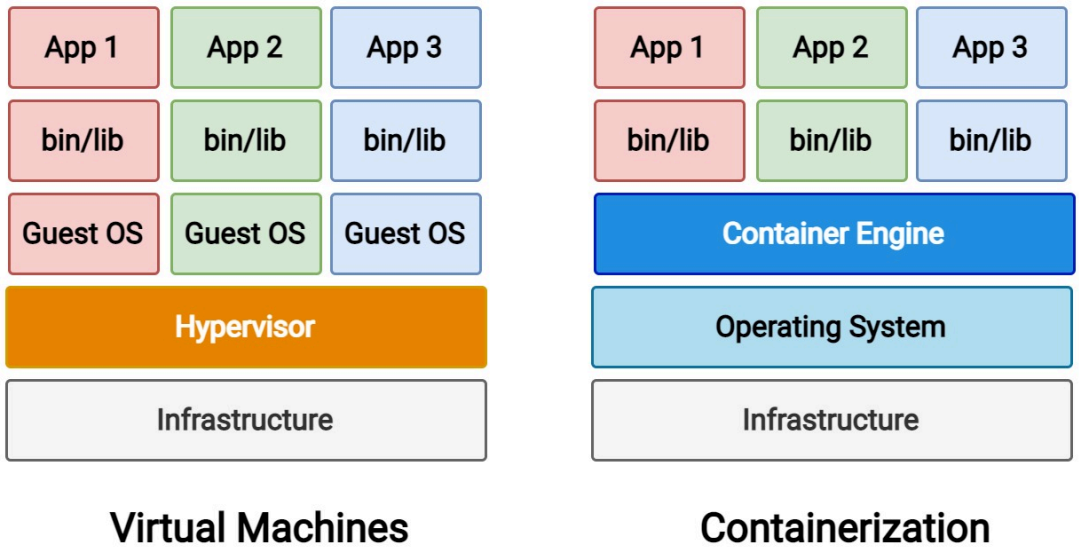

Figure 29 Virtual Machines vs Containerisation

Containers sit on top of a physical server and its host operating system and share the host operating system's kernel, binaries, and libraries. This makes containers extremely lightweight and operate with much lesser overheads than virtual machines. The differences are depicted in Figure 29.

The Docker Engine has been selected for containerising the application for this project. Docker is an open-source platform for building, deploying and managing containerised applications. The Docker Engine is the software responsible for hosting the containers.

By containerising the application into containers, we unlock the possibility of horizontally scaling the various components of the application if it is deemed required in the future. In addition, it also allows us to quickly and easily develop, build and deploy the application across any operating system that supports the Docker Engine.

4.1.3 Frontend Framework

4.1.3.1 Bootstrap 5

Bootstrap is a free and open-source frontend development framework. It provides a powerful toolkit consisting of a collection of HTML, CSS and JavaScript tools.

4.1.3.2 Mazer Dashboard Template

Mazer is a beautiful free and open-source Bootstrap 5 template. Using a frontend template allows us to focus more on the development of features for the platform instead of additional effort on tasks such as selecting complementary colours, fonts, and other design elements of the user interface.

4.1.4 Backend Framework

The Django Web Framework has been selected to be the backend framework for the platform. Founded in 2003, Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design [17]. Compared with other frameworks, it can be considered a “batteries-included” framework that provides all the critical components of a web application. These include components such as user authentication, session management, a powerful object-relational-mapping (ORM) engine, and an intuitive templating engine, amongst many other things.

Utilising Django over other frameworks will drastically improve the Maintainability of the project. Since the primary language is Python, commonly taught in

universities as an introductory language, it would require less effort to understand the code base. In addition, apart from being extremely well-documented, Django is committed to API stability and forwards-compatibility [18]. This means that code written on a version of Django will continue to work with future releases. In rare cases, only minor changes may be required when upgrading the version of a Django project.

More importantly, Django comes built-in with robust security features that help developers create more secure applications. For example, by utilising Django Forms and the ORM for interfacing with the database, proper security precautions are taken to prevent attacks such as SQL injections. On the other hand, manually constructing database connections and building SQL statements is much more prone to human error and insecure coding practices of inexperienced developers, potentially leading to security vulnerabilities.

4.1.5 Database

A database is needed to persist data for the web application. The database will store information such as user accounts, assessments, candidate attempts and assessment results.

The PostgreSQL database management system was chosen as Django officially supports it, and it is the most recommended database engine in the Django community. Thus, utilising this PostgreSQL over other relational database engines such as SQLite or MySQL will ensure that more Django features are supported and reduce the likelihood of any limitations in the future.

4.1.6 Code Execution Engine

In order to support programming questions on the platform, it is crucial to have a secure and efficient means of compiling, executing and grading codes. To avoid

reinventing the wheel or adding additional complexity to the project, a free and open-source project called Judge0 has been chosen and integrated into the AASP.

Judge0 is an Online Code Execution Engine (OCES) that provides features relating to the submission of codes and the retrieval of execution results. The Judge0 Community Edition can be self-hosted and deployed as part of the AASP technology stack. In addition, its RESTful API endpoints and clear documentation, defined in accordance with the OpenAPI specification, make integration into the backend framework effortless.

4.1.7 Asynchronous Task Queue

For the web platform to be responsive to users, long-running procedures should not hold up the response to a request. Instead, these long-running procedures or tasks should be off-loaded to a worker process and executed asynchronously.

Celery is a task queue implementation for Python applications. In the context of the AASP, it allows tasks to be added to the task queue by the Django web application and subsequently be retrieved by a worker process to execute. This allows the Django web application to rapidly serve and respond to new requests while these long-running tasks are executed in the background.

In addition, Celery also enables the platform to support scheduled tasks that runs at fixed intervals. These tasks may include sending reminder emails, cleaning the database, etc.

4.2 Key Features

The AASP has been developed in accordance with the design and system architecture detailed in the earlier sections. The following sections will showcase the key features of the AASP.

4.2.1 User Roles

Three user roles have been defined in the AASP: Educator, Lab Assistant and Student. Each role is only authorised to perform a subset of actions on the platform. Educators are allowed to manage question banks, courses and assessments. Lab Assistants are concerned with tasks such as enrolling students, resetting the password of student accounts and facilitating assessments if they are done in person. Finally, student accounts can be enrolled into courses and will have access to assessments belonging to their courses.

4.2.2 Dynamic Dashboards

Depending on the user's role, a different dashboard layout will be displayed to the user that will consist of the most relevant elements. The Educator and Lab Assistant dashboards organise the information from the Course point of view (Figure 30). On the other hand, the Student dashboard displays information based on the status of the assessments (Figure 31).

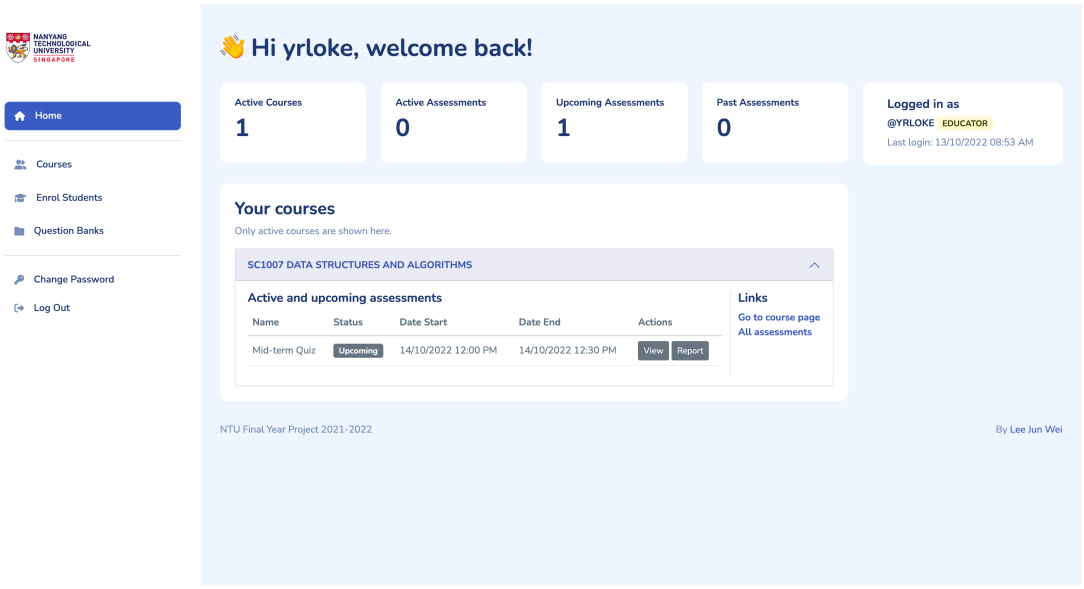

Figure 30 Educators' dashboard

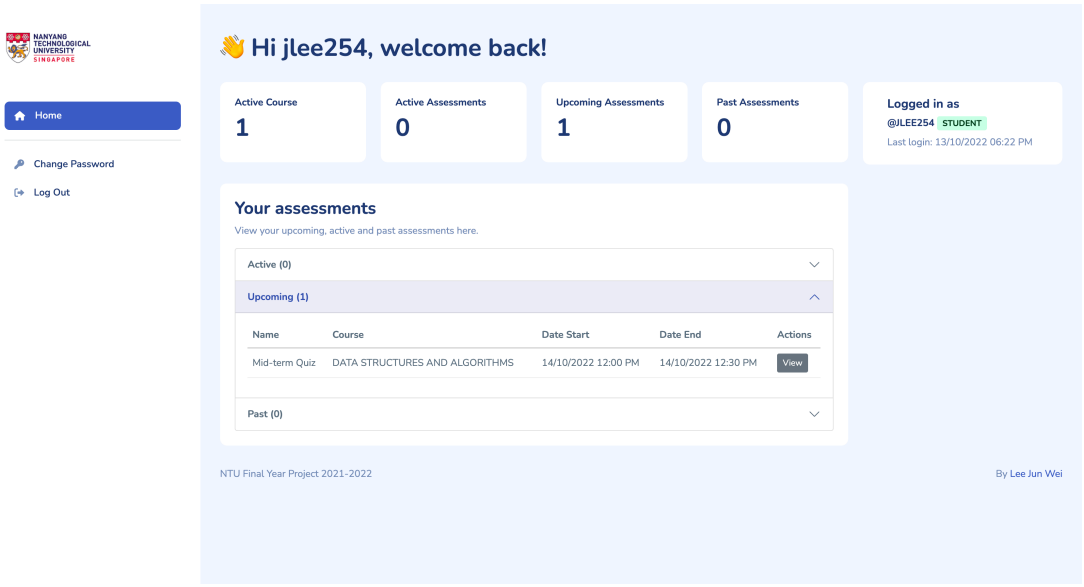

Figure 31 Students' dashboard

4.2.3 Question Banks

Question banks are a logical collection of questions that educators can create and manage. Visibility of question banks is private by default but can be made public by the owner. In addition, owners can also decide to share read-only access with other educators, allowing them to view and clone questions into their Assessments. Question banks can also be imported from and exported to JSON files to facilitate reusability and ease of sharing.

The question banks page is divided into three sections, as seen in Figure 32, depending on the ownership and visibility status of the question banks.

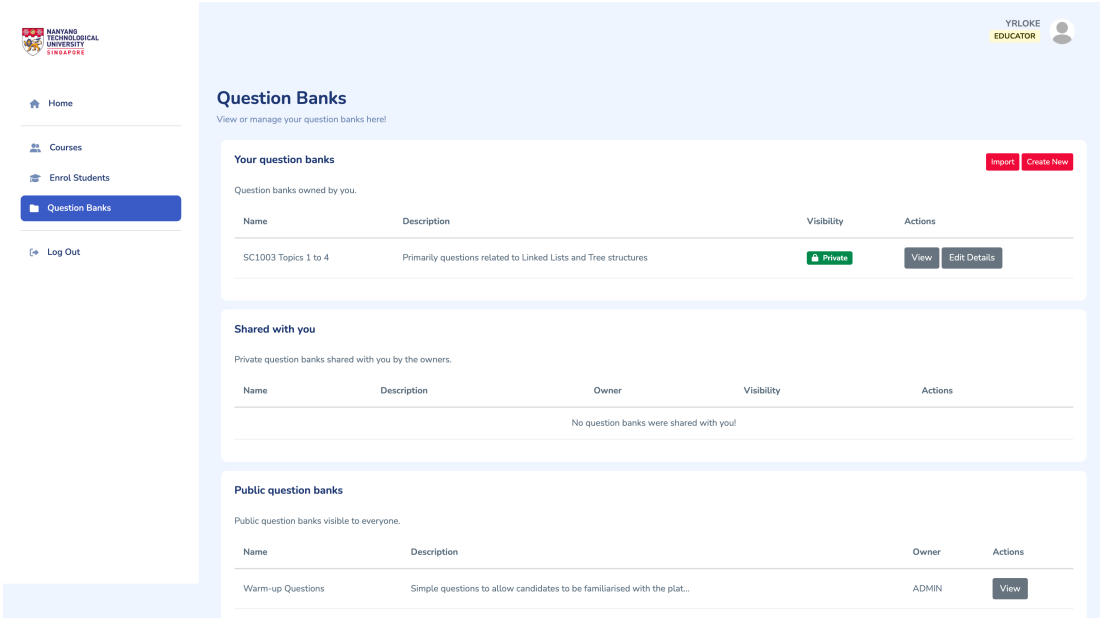

Figure 32 Question Banks page

Figure 33 shows the details page of a question bank, where basic information about the question bank is displayed along with the list of questions.

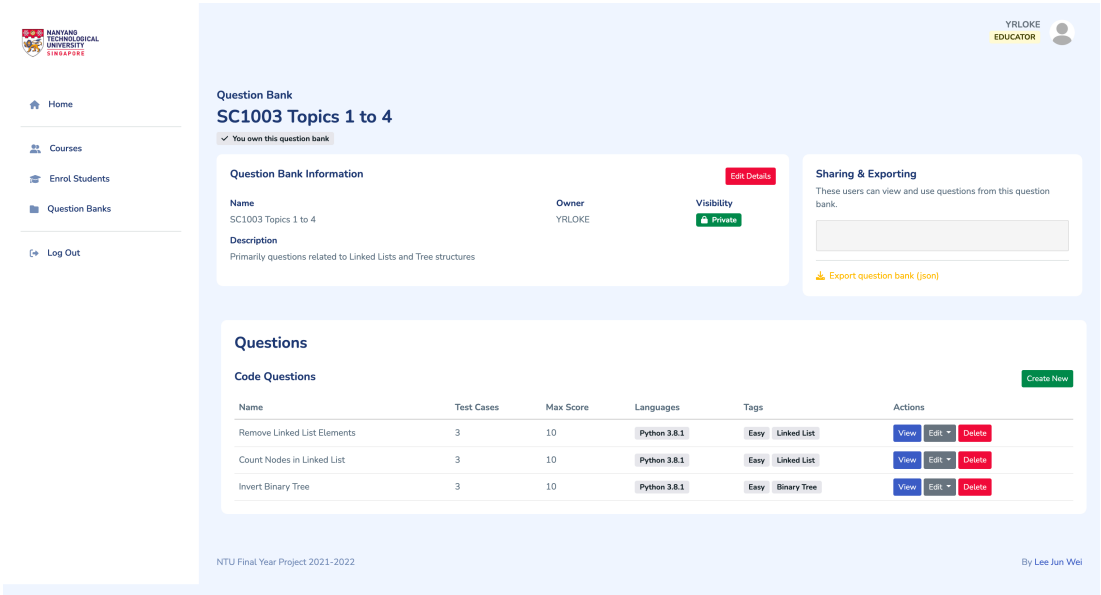

Figure 33 Question Bank details page

4.2.4 Courses

A course is a logical structure that contains a collection of students, course groups and assessments.

If management access to the course needs to be delegated to other authorised users such as other educators or lab assistants, the course owner can do so by adding these users as Maintainers to the course via the Maintainers section in the page (Figure 34).

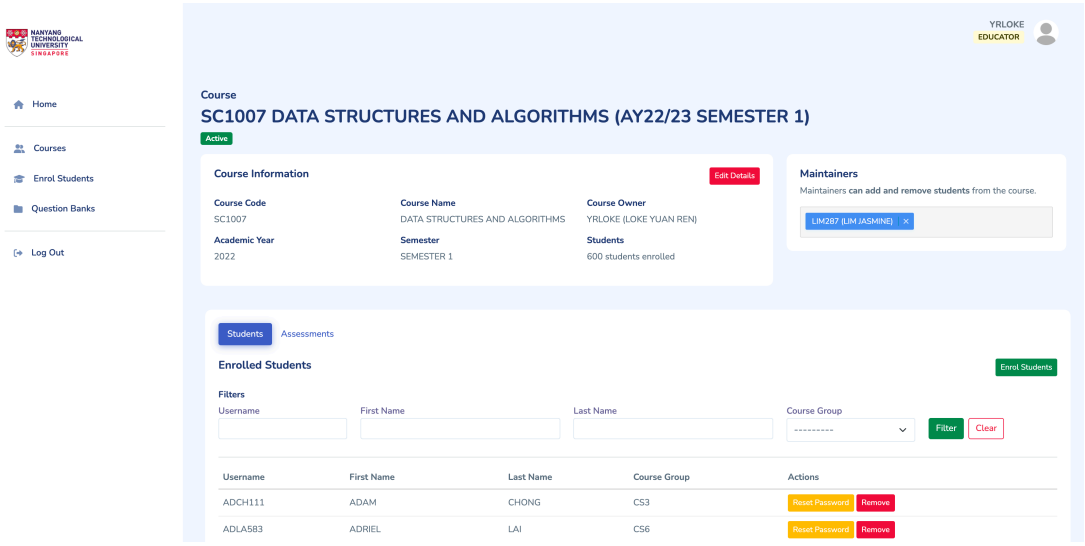

Figure 34 Course details page

4.2.5 Assessments

Assessments can be created for a course and published to students who are enrolled in the course. By default, an Assessment is marked as “Unpublished” and will only be published if explicitly set by the Educator (Figure 35).

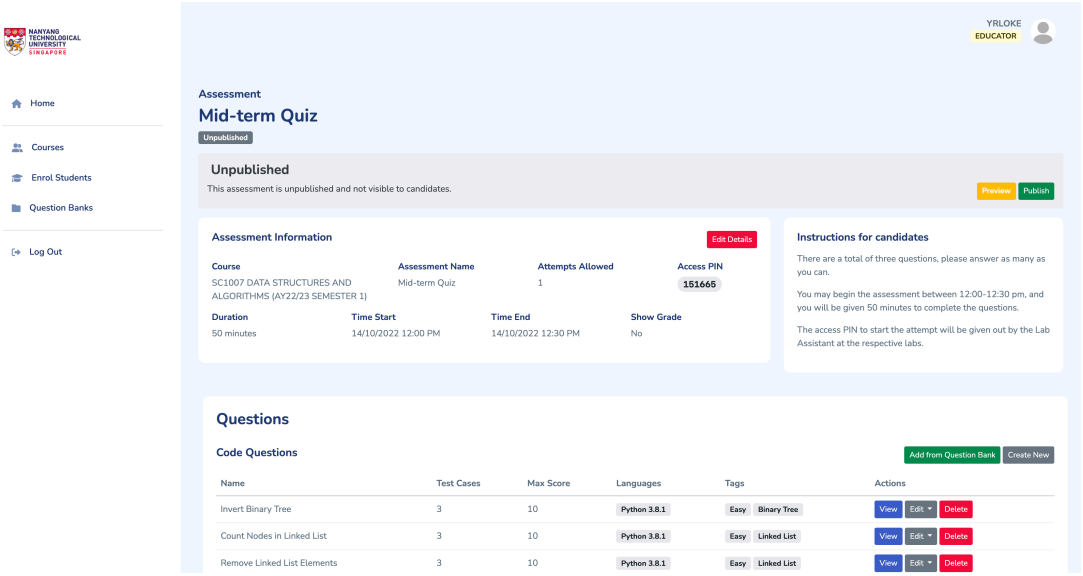

Figure 35 Assessment details page

Assessments can be populated with questions by cloning them from a question bank (Figure 36) or created from scratch.

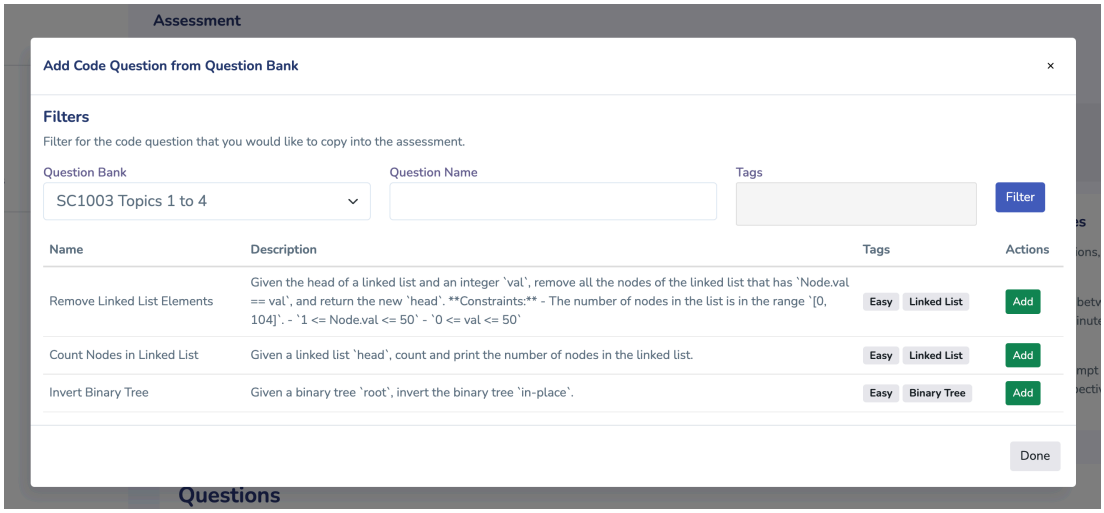

Figure 36 Cloning questions from question banks

Educators can also preview assessments before publishing them. This allows Educators to attempt the assessment from the a candidate's perspective and ensure that the questions are formatted correctly, and test cases are error-free.

4.2.6 Assessment Reports

Educators can view the report of an assessment that provides an overview of completed and ongoing attempts of the assessment (Figure 37). The breakdown of scores of each attempt can also be conveniently exported as a CSV file for easy tabulation.

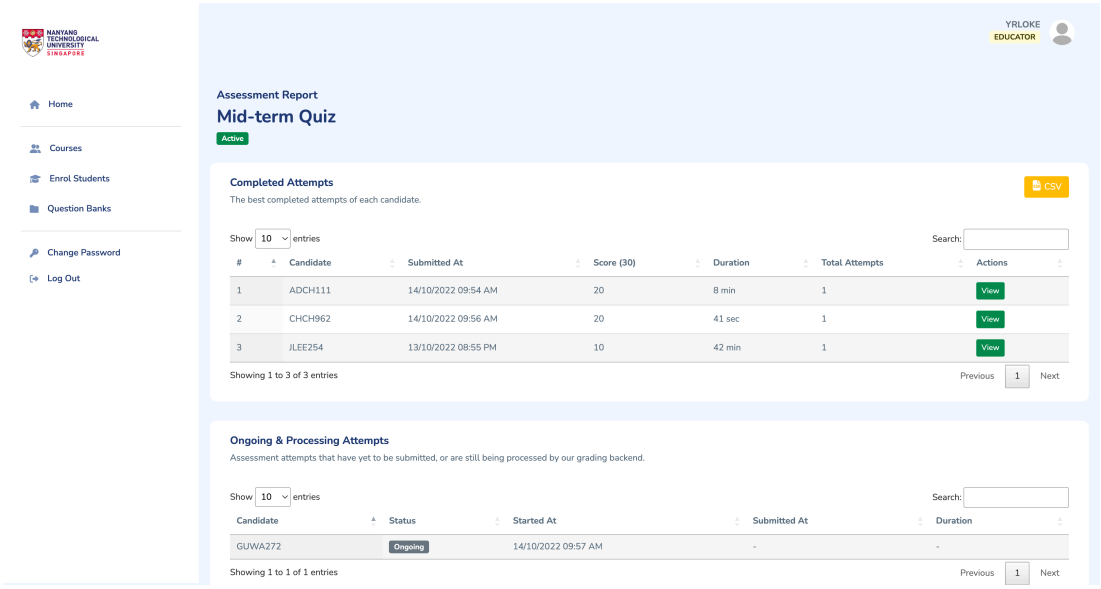

Figure 37 Assessment report page

For each assessment attempt, the educator can view the details of the submitted codes for each question and the corresponding result of each test case (Figure 38).

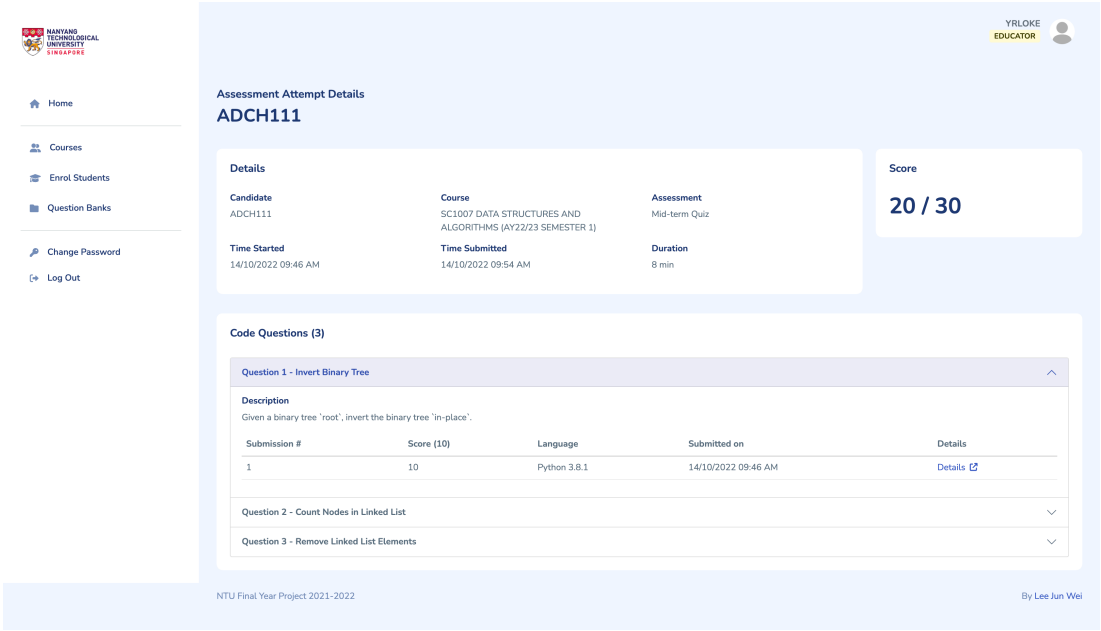

Figure 38 Assessment Attempt Details page

4.2.7 Taking Assessments

4.2.7.1 Landing Page

Students can view the information of an assessment on the assessment landing page after the educator has published an assessment. The landing page (Figure 39) displays an array of information such as the duration, availability period, instruction, number of allowed attempts and the candidate's attempt history (if any).

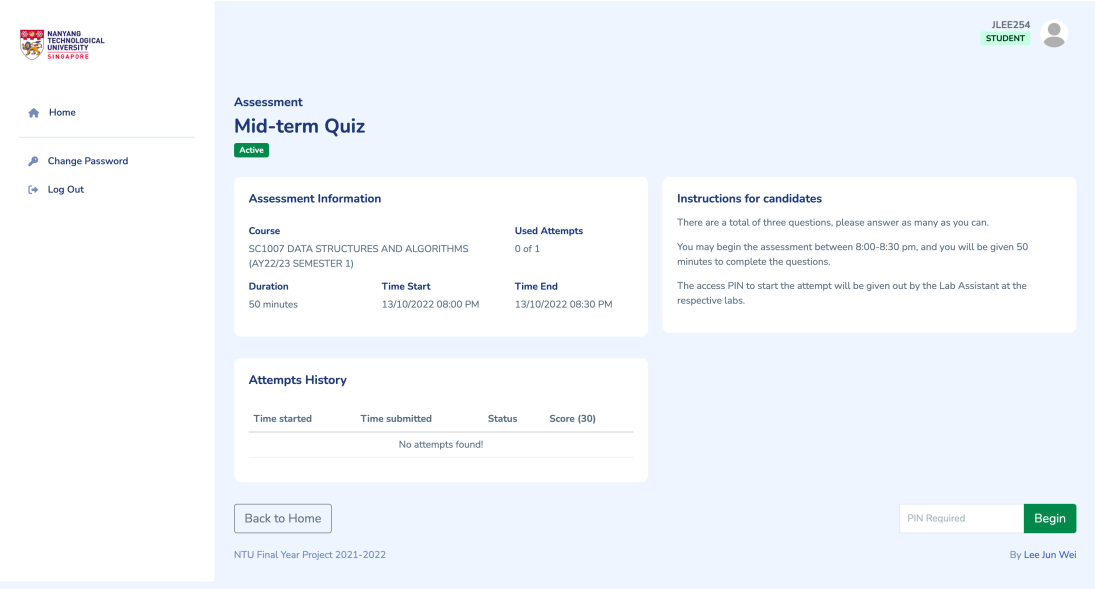

Figure 39 Assessment landing page

If the assessment period has not yet begun or has already lapsed, or if the maximum number of attempts has been reached, an appropriate message will be displayed, and the student will not be allowed to begin an attempt (Figure 40).

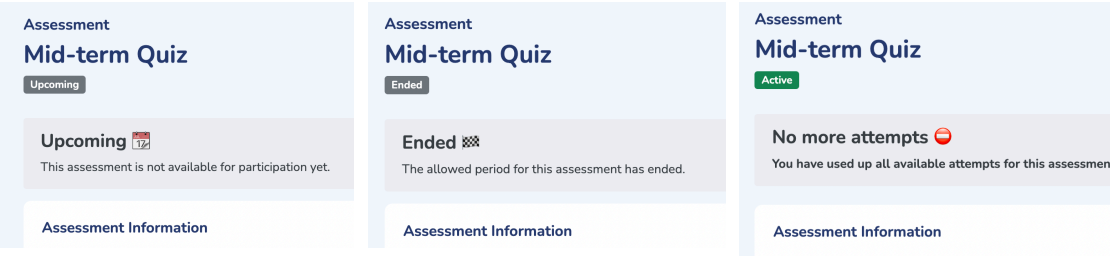

Figure 40 Assessment messages

4.2.7.2 Assessment-taking Page

The assessment-taking page (Figure 41) has a minimal layout that only displays the essential elements for answering the questions. It incorporates the horizontally split layout with resizable panes, mentioned in earlier chapters, to maximise user experience.

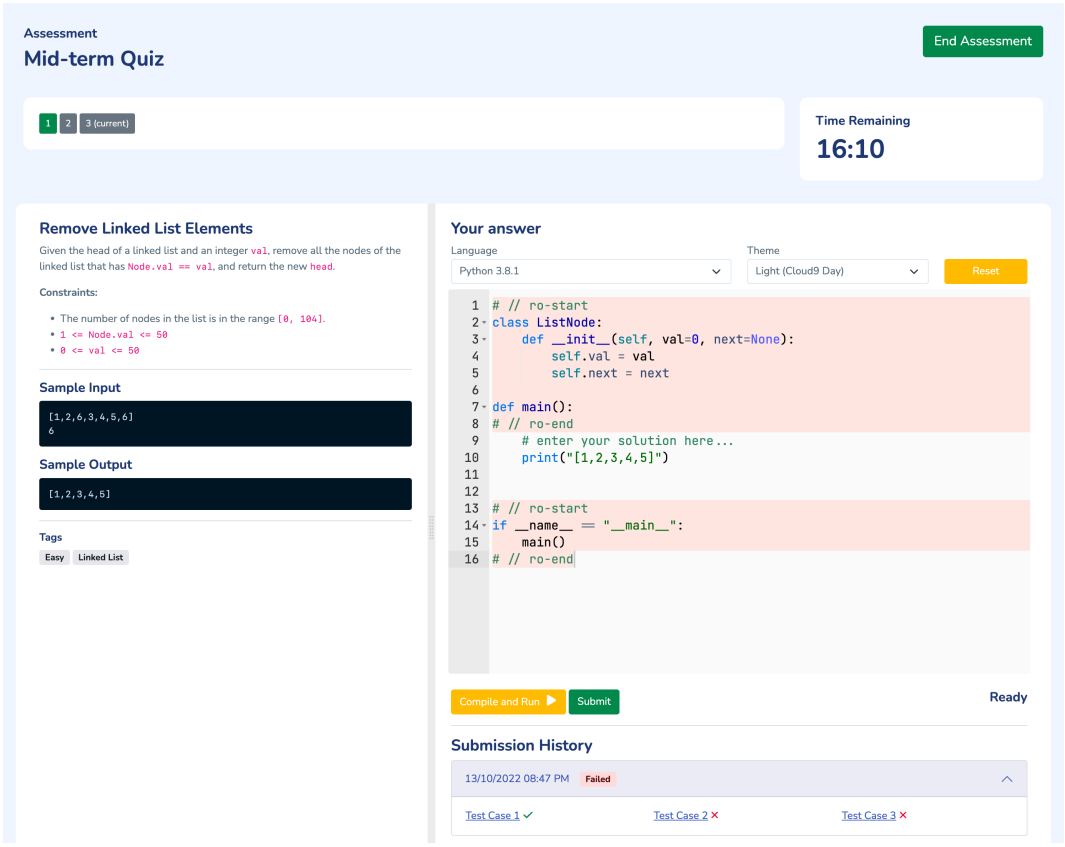

Figure 41 Assessment-taking page

4.3 Deployment and Setup

4.3.1 System Specifications

The School of Computer Science and Engineering (SCSE) has kindly provisioned a virtual desktop instance dedicated for the deployment of this project. Table 1 contains the system specifications of the allocated instance.

CPU	x2 vCPU, Intel(R) Xeon(R) Gold 6148 @ 2.40GHz
Memory	5.77 GB (8 GB Swap)
Disk	100 GB
Operating System	Ubuntu 20.04.5 LTS (Focal Fossa)
Docker Engine	Version 20.10.18, build b40c2f6

Table 1 System Specifications

4.3.2 Deployment Process

The system has been containerised with Docker, with its corresponding service definitions defined in a docker-compose file. This streamlines the deployment process, essentially reducing it to a three-step process: clone, configure, and deploy.

The following steps assume Docker Engine with Docker Compose has already been installed in the target system.

1. Clone the repository from GitHub and check out the main branch.

```
# via http
git clone https://github.com/leejunweisg/aasp
# or via ssh
git clone git@github.com:leejunweisg/aasp.git
```

2. Make a copy of the template configuration file, update the secret values and save.

```
# make a copy of the template file
cp ./config/.env_prod ./config/.env
# update secrets
vi ./config/.env
```

3. Start the containers with docker-compose.

```
sudo docker-compose -f docker-compose.yml up -d
```

Once all containers have been pulled, built and started, the AASP web platform will be assessable via the web browser at the virtual desktop's IP address (e.g., https://172.21.148.184/).

5 Future Works

5.1.1 More Question Types

The platform currently only supports programming (code) questions. To make assessments more flexible and all-rounded, additional question types such as Multiple-Choice Questions (MCQ) or short-answer questions can be added in the future.

5.1.2 Email Notifications

Email integration is a useful feature that should be considered as well. With the ability to send emails from the backend, the system will be able to disseminate automatic notifications and reminders and allow users to reset their account passwords without needing to be first logged in.

5.1.3 Proctoring

Proctoring features could be added to the AASP to enforce a strict assessment environment. For example, webcam snapshots can be taken to identify a candidate, keystrokes can be recorded to provide a better picture of the solution formulation approach, and specific actions such as tab switches could be detected and prevented.

5.1.4 Custom Judges

AASP currently utilises the Judge0 OCES for compiling, executing and judging the output of the test cases. Judge0 determines the correctness by a simple comparison of an expected output and actual output. The approach is simple and works universally for any programming language. This is suitable for calculation-based questions that involve computing simple values when given a specific input.

However, this approach does not allow for more robust checking of correctness, such as comparing two potentially complex data structures, such as linked lists. In order

to check for the correctness of a custom data structure, it has to be printed in its entirety in the standard output. Thus, the ability to utilise custom judges for programming questions will provide a more convenient means for the comparison of expected and actual outputs.

6 Conclusion

The project consists of several stages. The project started off with performing research on various related platforms available in the market. This activity provided insights into what automated assessment platforms are responsible for, their common features and good design practices. Next, research was conducted on existing methods for code evaluation covering dynamic and static analysis approaches. Once a fair baseline of understanding has been established, an analysis of the existing platform developed by the project's predecessors was conducted. This involved studying the reports and code base, followed by a practical exploration of the platform. During this process, several core issues were identified and provided the motivations to define this project's objective.

The objective of this project is to design and develop a new in-house Automatic Assessment Platform (AASP) that is fast, reliable and intuitive for end-users and, at the same time, addresses the issues observed from the existing platform. Since maintainability and security were identified as core issues of the previous system, they are the core focus of this project.

The new AASP was then carefully designed and planned. This included decisions such as selecting Python as the primary programming language to minimise the learning curve and the selection of Django as the backend framework for its forwards-compatibility and its collection of security features that are enabled by default.

Finally, the AASP was implemented in accordance with the design defined in **Chapter 3.** The platform boasts a clean user interface to provide end users with a positive user experience. The complexity of the technology stack has also been drastically reduced by utilising an open-source code execution engine, Judge0. The

project was then containerised with Docker, which allows deployment to be completed using a simple three-step process described in **Chapter 4**.

Although the system is fully functional, enhancements are always welcomed if a successor is keen. Some suggestions for future works that could be considered are listed in **Chapter 5**.

Overall, this project was challenging albeit fruitful, as one developer must take on the full range of tasks: developing the frontend and backend and preparing the project for deployment. When met with an obstacle, there is no one to discuss approaches and share the development load with. This called for personal discipline to remain consistent in the development and provided me with numerous opportunities for independent learning.

7 References

[1] J. Cook and V. Jenkins (2010), “Getting started with e-assessment”, University of Bath.

[2] Müller, A. M., Goh, C., Lim, L. Z., & Gao, X. (2021). COVID-19 Emergency eLearning and Beyond: Experiences and Perspectives of University Educators. Education Sciences, 11(1), 19. MDPI AG. Retrieved from http://dx.doi.org/10.3390/educsci11010019

[3] Soh Y. Q. (2021), “Automated Assessment Platform (AASP)”, Nanyang Technological University.

[4] Yap G. S. (2021), “Development of Auto-Assessment System”, Nanyang Technological University.

[5] Ace - The High Performance Code Editor for the Web. Retrieved September 12, 2022, from https://ace.c9.io/

[6] GitHub. prasmussen/glot-containers: Docker containers for running code. Retrieved July 10, 2022, from https://github.com/prasmussen/glot-containers

[7] GitHub. prasmussen/glot-code-runner: Code runner. Retrieved July 10, 2022, from https://github.com/prasmussen/glot-code-runner

[8] OWASP Foundation. Cross Site Scripting (XSS) | OWASP Foundation. Retrieved May 16, 2022, from https://owasp.org/www-community/attacks/xss/

[9] K. A. Rahman and M. J. Nordin (2007), “A Review on the Static Analysis Approach in the Automated Programming Assessment Systems”, National Conference on Programming 07.

[10] Ala-Mutka, K. M. (2005). A survey of Automated Assessment Approaches for programming assignments. Computer Science Education, 15(2), 83–102. https://doi.org/10.1080/08993400500150747

[11] V C, S., Srinivasa Prasad, S., Dheemanth, G. R., & Kumar, N. S. (2019). Assessment of quality of program based on static analysis. 2019 IEEE Tenth

International Conference on Technology for Education (T4E). https://doi.org/10.1109/t4e.2019.00072

[12] H. Z. Došilović and I. Mekterović (2020), "Robust and Scalable Online Code Execution System," 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), pp. 1627-1632, doi: 10.23919/MIPRO48935.2020.9245310.

[13] Kosner, A. W. (2014). Hackerrank solves tech hiring crisis by finding programmers where they live. Forbes. Retrieved September 10, 2022, from https://www.forbes.com/sites/anthonykosner/2014/06/12/hackerrank-solves-tech-hiring-crisis-by-finding-programmers-where-they-live/?sh=16b76ad8b644

[14] Venkatesha, B. (2015). Coolest start-ups 2015: HackerEarth helps find top-quality coders for tech companies. Business Today. Retrieved September 10, 2022, from https://www.businesstoday.in/magazine/cover-story/story/business-today-coolest-start-ups-2015-hackerearth-success-48878-2015-06-07

[15] LeetCode. LeetCode - The World's Leading Online Programming Learning Platform. Retrieved September 10, 2022, from https://leetcode.com/

[16] Ovadia, S. (2014). Markdown for librarians and academics. Behavioral & Social Sciences Librarian, 33(2), 120–124. https://doi.org/10.1080/01639269.2014.904696

[17] Django Software Foundation. The web framework for perfectionists with deadlines. Django. Retrieved August 24, 2022, from https://www.djangoproject.com

[18] Django Software Foundation. API stability | Django documentation | Django. Django. Retrieved August 24, 2022, from https://docs.djangoproject.com/en/4.1/misc/api-stability/