Technical Specifications

Recode – A Smart Coding Challenge Generator for Students

NWU

2025-08-29

Version 0.0.4

|  |  |  |  |
| --- | --- | --- | --- |
| REVISION HISTORY | | | |
| DATE | VERSION | DESCRIPTION | AUTHOR |
| 8/18/2025 | 0.0.1 |  |  |
| 8/24/2025 | 0.0.2 |  |  |
| 8/28/2025 | 0.0.3 | Modified and added some details. | Brandon van Vuuren |
| 8/29/2025 | 0.0.4 | Final Revisions and Formatting. | Brandon van Vuuren |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Introduction 5](#_Toc207375148)

[1.1 Purpose 5](#_Toc207375149)

[1.2 Intended Audience 5](#_Toc207375150)

[1.3 Stakeholder Focus 6](#_Toc207375151)

[ Project Manager 6](#_Toc207375152)

[ Developer 6](#_Toc207375153)

[ Tester / QA Engineer 6](#_Toc207375154)

[ Operator / End-User Perspective 6](#_Toc207375155)

[2. Project Scope 6](#_Toc207375156)

[In Scope: 7](#_Toc207375157)

[Out of Scope (MVP): 7](#_Toc207375158)

[Benefits of ReCode: 7](#_Toc207375159)

[Objectives: 8](#_Toc207375160)

[Document Conventions 8](#_Toc207375161)

[References 8](#_Toc207375162)

[3. Description 9](#_Toc207375163)

[Product Perspective: 9](#_Toc207375164)

[Features 9](#_Toc207375165)

[ Weekly Challenge Delivery 9](#_Toc207375166)

[ Lecture Slide Upload & Text Extraction 9](#_Toc207375167)

[ Automated Topic Detection (NLP) 9](#_Toc207375168)

[ Challenge Mapping & Retrieval 9](#_Toc207375169)

[ Code Submission & Automated Grading 9](#_Toc207375170)

[ Gamification – Badges, Points, and Titles 9](#_Toc207375171)

[ Progress Tracking & Profiles 10](#_Toc207375172)

[ User Account Management & Authentication 10](#_Toc207375173)

[4. User Overview 10](#_Toc207375174)

[Users: 10](#_Toc207375175)

[Administrators: 10](#_Toc207375176)

[Operating Environment 10](#_Toc207375177)

[5. Design and Implementation Constraints 11](#_Toc207375178)

[Documentation 11](#_Toc207375179)

[Content 11](#_Toc207375180)

[Assumptions / Dependencies 11](#_Toc207375181)

[6. System Features 11](#_Toc207375182)

[SYSTEM FEATURE 1: WEEKLY CHALLENGE DELIVERY 11](#_Toc207375183)

[SYSTEM FEATURE 2: LECTURER CONTENT INTEGRATION 12](#_Toc207375184)

[SYSTEM FEATURE 3: AUTOMATED TOPIC TAGGING 12](#_Toc207375185)

[SYSTEM FEATURE 4: CHALLENGE DATABASE & RETRIEVAL 13](#_Toc207375186)

[SYSTEM FEATURE 5: CODE SUBMISSION AND JUDGE0 EVALUATION 13](#_Toc207375187)

[SYSTEM FEATURE 6: USER AUTHENTICATION & ROLES 14](#_Toc207375188)

[SYSTEM FEATURE 7: GAMIFICATION MECHANICS 14](#_Toc207375189)

[SYSTEM FEATURE 8: STUDENT PROGRESS DASHBOARD 15](#_Toc207375190)

[SYSTEM FEATURE 9: LECTURE BASIC TOOL 15](#_Toc207375191)

[SYSTEM FEATURES 10: LECTURER’S ANALYTICAL VIEW 16](#_Toc207375192)

[SYSTEM FEATURE 11: LECTURE CUSTOMIZATION 16](#_Toc207375193)

[8. External Interfaces Requirements 17](#_Toc207375194)

[User Interface 17](#_Toc207375195)

[Hardware Interface 18](#_Toc207375196)

[Software Interface 19](#_Toc207375197)

[Communication Interface 20](#_Toc207375198)

[9. Additional Non-Functional Requirements 20](#_Toc207375199)

[Performance 20](#_Toc207375200)

[Safety 21](#_Toc207375201)

[Security 21](#_Toc207375202)

[Software Quality 22](#_Toc207375203)

# 1. Introduction

## 1.1 Purpose

ReCode is a **web-based platform** designed to simplify and enhance how first-year university students learn programming. The system leverages **gamification, automated challenge generation, and real-time feedback** to transform lecture content into interactive coding exercises.

Key aspects of ReCode include:

* **Weekly coding challenges** aligned with lecture material.
* **Reward-based system** (badges, ELO, ranks) to boost student motivation.
* **Automated grading** and feedback via Judge0.
* **Semester-based cycles** that reset every 6 months, allowing repeatability and improved curriculum alignment.

The purpose of this specification is to **document the functional and technical requirements of ReCode**, ensuring that stakeholders (academic staff, developers, and students) have a shared understanding of the project’s scope, design, and expected outcomes.

## 1.2 Intended Audience

This document is intended for stakeholders involved in the design, development, and evaluation of the ReCode platform:

* **Project Managers / Clients** – To verify alignment with institutional goals and student success strategies.
* **Developers** – To understand technical specifications, architecture, and stack decisions.
* **Testers / QA Engineers** – To design test cases against defined functional and non-functional requirements.
* **Lecturers (Administrators)** – To manage content uploads, student analytics, and system oversight.
* **Students (End-Users)** – As the primary beneficiaries, they engage with weekly coding challenges and tracking progress.

## 1.3 Stakeholder Focus

### Project Manager

* + *Scope & Objectives*: Understand the aims of the project and how they align with institutional teaching strategies.
  + *Strategic Value*: Evaluate benefits such as student engagement, performance analytics, and curriculum improvement.

### Developer

* + *Technical Specifications*: Detailed technology stack (FastAPI, Supabase, Redis, Judge0, React).
  + *Software Interfaces*: System architecture, API contracts, and user interface design principles.
  + *Operating Environment*: Requirements for hosting, deployment, and development environments.

### Tester / QA Engineer

* + *External Interface Requirements*: Expected behaviours, usability standards, accessibility compliance.
  + *Performance Requirements*: System responsiveness, grading latency, and concurrent user support.
  + *Software Quality*: Maintainability, error handling, testing strategies, and regression protocols.

### Operator / End-User Perspective

* + *Student Interfaces*: Interaction with coding challenges, progress dashboards, and badges.
  + *Lecturer Interfaces*: Lecture slide uploads, topic tagging, challenge mapping, analytics dashboards.
  + *Features Overview*: Comprehensive understanding of system functionality across both roles.

# 2. Project Scope

The **scope of ReCode** defines its objectives, advantages, and alignment with stakeholder needs. The platform is designed to transform how first-year university students learn programming by combining gamification, automation, and real-time feedback in a scalable web application.

## In Scope:

* Development of a web-based platform accessible via modern browsers.
* Lecture slide upload and parsing (python-pptx, with OCR extension planned).
* Automated NLP topic extraction (spaCy + Hugging Face), with lecturer override.
* Challenge mapping and retrieval from a curated database aligned to lecture content.
* Automated grading of coding submissions using Judge0.
* Gamified progression system (badges, ELO, titles, seasonal resets).
* Dashboards for students and lecturers to track progress, analytics, and course outcomes.
* Authentication and role-based access via Supabase (students, lecturers, admins).
* Deployment on cloud services (Supabase, Railway/Render, GitHub Actions CI/CD).

## Out of Scope (MVP):

* Mobile application (may be considered later).
* OCR-based parsing of non-pptx files (future work).
* Third-party LMS integrations (e.g., Moodle, eFundi).
* Large-scale AI tutors or chatbot assistants.

## Benefits of ReCode:

* Makes learning interactive, fun, and engaging.
* Provides real-time feedback on coding challenges.
* Works on a reward-based gamification model that improves student motivation.
* Generates analytics for lecturers at the end of each semester to identify strengths/weaknesses across the cohort.
* Enables consistent progression with badges (bronze → diamond) and ELO-based titles.
* Encourages practical, applied learning instead of rote memorisation.

## Objectives:

* Increase engagement by blending **academic content with gamified coding challenges**.
* Empower students with **progress tracking, achievements, and personalised learning feedback**.
* Provide lecturers with **insights into student performance and content effectiveness**.
* Ensure the system is **scalable, maintainable, and secure** to support long-term academic adoption.

## Document Conventions

This document uses the following conventions for clarity and consistency:

* **Headings/Subheadings**: Bold with hierarchical numbering (e.g., 1.0, 1.1).
* **Font**: Arial 11pt; Consolas for code snippets.
* **Figures/Tables**: Sequentially numbered (e.g., *Figure 1: System Architecture*).
* **Lists**: Numbered for steps; bulleted for related items.
* **Terminology**: Technical terms (e.g., NLP, Judge0, ELO) are defined in the Glossary.
* **Versioning**: Managed in the Revision History.

## References

This specification draws on:

* *ReCode Feasibility Study*
* *ReCode Business Case*
* *ReCode Functional Specifications*
* *CMPG323 Guidelines & Rubric*
* *Supabase Documentation*
* *FastAPI Documentation*
* *Judge0 API Documentation*
* *spaCy & Hugging Face Model Docs*
* *South African POPIA Act*

# 3. Description

## Product Perspective:

To improve the learning curve of our fellow first-year students, we are using a web-based system called ReCode. This is a robust, highly effective, well-maintained platform that is aimed at making the lives of students easier when it comes to learning how to code. Our goal is to increase student engagement by providing a practical, interactive approach to module content, making learning enjoyable rather than a tedious requirement.

Our system will include core operations that include providing coding challenges, real-time feedback where the students’ code will be assessed using Judge0, data collection of the students’ performance and rewards (badges and points).

## Features

* Weekly Challenge Delivery: The system will release a set of coding challenges to students on a weekly schedule. Each week’s challenges are categorised by difficulty. This structured weekly content ensures a steady progression through the semester and keeps students engaged.
* Lecture Slide Upload & Text Extraction: Lecturers can upload lecture content to the platform. The backend will extract text from the slides using a Python library (python-pptx, for .pptx files).
* Automated Topic Detection (NLP): The system analyses the extracted lecture text to determine the key programming topic or concept for that week.
* Challenge Mapping & Retrieval: Once a topic is identified (either by NLP or by lecturer confirmation), the system fetches a set of coding exercises corresponding to that topic.
* Code Submission & Automated Grading: Students can submit their solutions (code) for the weekly challenges through the front-end interface from which it will be sent to the backend and assessed using Judge0.
* Gamification – Badges, Points, and Titles: To motivate students, the platform includes a gamified reward system. Successful completion of challenges awards badge achievements (Bronze/Silver/Gold, corresponding to the challenge difficulty) and contributes to a point score.
* Progress Tracking & Profiles: The platform keeps a record of each student’s progress throughout the course. This includes which challenges have been completed, scores obtained on each, badges earned, current ELO score, and current title/rank.
* User Account Management & Authentication: The system provides secure user account creation and login functionality.

# 4. User Overview

## Users:

The users will mainly be the students; they will be utilising the system when participating in the coding challenges. They will be interacting with the system on the user interface, where they will be doing the challenges. They will also be able to view their performance through the marks they receive since the system automatically marks the students' work. Based on the students’ performance, they receive different badges, which is an indication of what skill level the student is at, and they are able to view this and see where they can improve and sharpen their skills. This also incentivises and encourages the student to improve and receive the badges.

## Administrators:

The administrators will be the lecturers; they will be handling the lecture side of the system, where they are able to upload or select their module at the beginning of each semester. They will be uploading the slides at the beginning or at the end of each lecture that will be covering the work that was done in class on that particular day, for the students to have access after class. The lectures are able to view the performance of the students, based on metrics such as the marks they receive, the duration of how long it takes a student to complete said challenge and the badges that students receive based on their performance. The lecturer is also able to extract all the data collected throughout the semester to then improve the module content and the way of teaching, based on how the students perform.

In addition, system administrators will oversee platform maintenance, security, and general role management. However, the assignment of academic roles (e.g., module-level lecturer or teaching assistant permissions) will remain the responsibility of departmental heads for each module.

## Operating Environment

ReCode will be a web-based platform; therefore, it will be run on any web browser for any student to access at any given time. If time constraints allow, then ReCode can also be implemented as a desktop software, which will at that point be accessible not only via a web browser but also on the students’ personal devices to allow for easier and quicker access with no need for internet access, but this will only be implemented if there is enough time. Otherwise, the system will remain a web-based platform.

# 5. Design and Implementation Constraints

The system must only be a web-based application because of time constraints; it needs to be implemented and deployed directly to a web server, whereby the system will be hosted and run on to allow for easy and simple use for the student’s convenience.

## Documentation

In order for us to have maintainability, usability, and knowledge transfer, the system will be supported by documentation that covers how the system is to be used and the configuration behind it. The documentation is organised for both technical and non-technical users.

## Content

The documentation will include:

* **User Guide** - Provides students with guidance on all the key components and features of the system regarding where they need to write their code, how to submit, etc.
* **Administrator Guide** - Outlines how to load module content, manage users, and extract student data and progress.

## Assumptions / Dependencies

Important software components like databases for user management, programming code-based Python libraries for student code assessment and feedback generation. It is expected that users will receive documentation and guidance to use the system effectively.

# 6. System Features

## SYSTEM FEATURE 1: WEEKLY CHALLENGE DELIVERY

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | Students receive a weekly set of 10 coding challenges whose topic corresponds to that week’s lecture slides. These challenges are divided by difficulty: 5 Bronze, 3 Silver, and 2 Gold |
| **STIMULUS RESPONSE SEQUENCES** | The lecturer uploads the teaching material (slides) to the system, and then a set of 10 weekly coding challenges is received by students based on the materials’ topic. |
| **FUNCTIONAL REQUIREMENTS** | * Generate an accurate set of coding challenges that matches the lecturer’s slides topic * Analyse the lecturer’s slides accurately * Release a set of coding challenges weekly |

## SYSTEM FEATURE 2: LECTURER CONTENT INTEGRATION

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | For the system to identify each week’s topic, lecturers upload slides to the system weekly |
| **STIMULUS RESPONSE SEQUENCES** | Lecturers upload slides to the system using the lecture user interface, then a topic for the coding challenge for that week is selected. |
| **FUNCTIONAL REQUIREMENTS** | * Identify keywords from the slides and extract the text * Carefully analyses the text to generate an accurate challenge |

## SYSTEM FEATURE 3: AUTOMATED TOPIC TAGGING

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | Use the extracted text from the lecturer's slides to identify key terms to tag the content with a topic |
| **STIMULUS RESPONSE SEQUENCES** | Get the keywords extracted from the lecture slides and generate a suitable topic. If the automated system cannot confidently pick a topic or chooses incorrectly, the lecturer can manually select the correct topic from a predefined list |
| **FUNCTIONAL REQUIREMENTS** | * Determine topics using key terms from lecture slides |

## SYSTEM FEATURE 4: CHALLENGE DATABASE & RETRIEVAL

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | A curated database that consists of coding challenges each with its suitable topic/topics |
| **STIMULUS RESPONSE SEQUENCES** | Receives identified topics from Automated Topic Tagging and manually by the lecturer and retrieves the corresponding coding challenges from the database. |
| **FUNCTIONAL REQUIREMENTS** | * Ensure that the topic identified exists in the database * Retrieve the correct coding challenge from the database based on the topic |

## SYSTEM FEATURE 5: CODE SUBMISSION AND JUDGE0 EVALUATION

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | Students enter their coding submissions into the platform and get marked automatically |
| **STIMULUS RESPONSE SEQUENCES** | A student submits their code solution to a coding challenge. The student will then get an immediate response indicating success or failure, and possibly the output or error message if the solution is wrong. |
| **FUNCTIONAL REQUIREMENTS** | * Run the code and check it against the problem’s test cases * Give students immediate feedback on their code submission * Automatically grade the students’ code * Can handle errors from students’ code submission |

## SYSTEM FEATURE 6: USER AUTHENTICATION & ROLES

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | To use the ReCode system, users need to sign up and log in using an email and a password |
| **STIMULUS RESPONSE SEQUENCES** | Sign up and log in to the system using an email and password; this grants access to the system. Users get a user interface and privileges that correlate with their user roles. |
| **FUNCTIONAL REQUIREMENTS** | * Ensure that the email entered exists and is correct * Provide users with the correct user interface and privileges |

## SYSTEM FEATURE 7: GAMIFICATION MECHANICS

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | A student receives badges, a score and a rank based on their performance on the coding challenges. |
| **STIMULUS RESPONSE SEQUENCES** | A student completes a coding challenge. The student will get a badge based on the difficulty of the challenge completed. Their score is added to the one their existing one, then a rank is given accordingly. |
| **FUNCTIONAL REQUIREMENTS** | * Award a student a badge based on the difficulty level of the challenge completed * Aggregate scores * Provide a student with a suitable rank based on their score |

## SYSTEM FEATURE 8: STUDENT PROGRESS DASHBOARD

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | Ensures that students get a simple yet aggregated view of their data |
| **STIMULUS RESPONSE SEQUENCES** | Get student performance data and display it on for the student to see. |
| **FUNCTIONAL REQUIREMENTS** | * Retrieve the correct performance information * Retrieving the information at a faster rate |

## SYSTEM FEATURE 9: LECTURE BASIC TOOL

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | The lecturer gets additional privileges, such as being able to change user roles. |
| **STIMULUS RESPONSE SEQUENCES** | Get input from the lecturer and carry out the specified actions |
| **FUNCTIONAL REQUIREMENTS** | * The lecturer can create new module instances * User roles can be changed for the lecturers’ students * Can view information on the different students * Can upload slides weekly |

## SYSTEM FEATURES 10: LECTURER’S ANALYTICAL VIEW

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | To assess how students are doing in the coding challenges, lecturers are provided with an overview of student participation |
| **STIMULUS RESPONSE SEQUENCES** | Input on what information to display, then display the requested data and a performance metric. |
| **FUNCTIONAL REQUIREMENTS** | * Only accessible to lecturers * Can aggregate and analyse the students’ performance data * Generate a graph based on the participation of students |

## SYSTEM FEATURE 11: LECTURE CUSTOMIZATION

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | To ensure that students receive a coding challenge with the correct topic, lectures can decide on which topic to choose. |
| **STIMULUS RESPONSE SEQUENCES** | The lecturer uploads a slide on the platform, a generated topic based on the text on the slide is displayed. The lecturer can accept the topic or change it by selecting a topic from the available ones. The coding challenge is then made available to students |
| **FUNCTIONAL REQUIREMENTS** | * Display the generated topic to the lecturer * Allow a lecturer to change the generated topic using the provided list of other topics * Can generate a coding challenge |

# 8. External Interfaces Requirements

## User Interface

The web-based interface offered by the ReCode platform is meant to ensure efficient and seamless interaction between the lecturers and students. The platform is accessible to end users through mobile, tablet, and desktop platforms. While lecturers can upload lecture content, track student progress and verify automated topic tagging, students can initiate programming challenges, view progress and engage with gamification elements through the interface.

**For Students:**

* Dashboard displaying **progress, badges, ranks, and accumulated points/ELO** throughout the semester.
* Weekly challenge interface with **Bronze, Silver, and Gold challenges** directly linked to lecture content.
* Higher-tier badges (**Platinum, Emerald, and Diamond**) representing advanced challenges that function like **semester tests and final exams**, contributing more heavily to overall scoring.
* **Running GPA-style score** that compares **practical coding performance (from challenges/badges)** to **theoretical performance (from traditional tests/exams)**, allowing both students and lecturers to track practical vs theoretical balance.
* Embedded code editor with **syntax highlighting, inline feedback, error messages**, and support for multiple programming languages.

**For Lecturers:**

* **Upload interface** for weekly lecture slides (PPTX/PDF).
* **Automated topic detection** workflow powered by NLP, with manual override for accuracy and control.
* **Analytics dashboard** providing reports on student performance, badge distribution, and progression trends, including comparisons between theory (exam/test scores) and practice (challenge outcomes).
* Tools for **semester-level evaluation**, enabling lecturers to identify weak areas across cohorts and refine teaching strategies.

**Accessibility and Standards:**

* Full compliance with **web accessibility standards** (keyboard navigation, ARIA labels, colour contrast).
* **Consistent UI elements** to reduce cognitive load and improve usability for both students and lecturers.
* Mobile- and desktop-friendly design to ensure universal access.

## Hardware Interface

ReCode is a platform hosted in the cloud that does not require specific hardware.

* End Users:
  + Students and lecturers both require personal computers or mobile devices that can run modern web browsers (Chrome, Firefox, Edge) with JavaScript enabled.
* Backend Infrastructure:
  + Containerised cloud services that run on platforms such as Fly.io or Railway.
  + Components include:
    - Supabase (Postgres + Auth) for authentication and storage.
    - Redis for task queues (e.g., Judge0 submissions, notifications, etc.).
    - Docker for containerization.
  + Docker, PostgreSQL, and Redis are necessary for development workstations.
  + GitHub Actions can be utilised by CI/CD pipelines to automate deployment and testing.

## Software Interface

The ReCode MVP integrates several software components to ensure dependable and synchronized operations:

* Core Application:
  + Backend built using FastAPI (Python), with microservices communicating via RESTful APIs (JSON).
  + Frontend functionality is implemented as a single-page app (React/Vue) that interacts with backend APIs.
* Third-Party Integrations:
  + Supabase – file storage, database and authentication.
  + python-pptx – parsing lecture slides.
  + spaCy NLP – automated topic tagging.
  + Judge0 API – auto-grading and sandboxed code execution.
  + N8n – workflow and notification automation.
* Authentication & Roles:
  + JWT (JSON Web Tokens) for safe and secure API sessions.
  + Role-based access: Students (view and submit challenges), Lecturers (upload, confirm topics and manage platform).
* Constraints:
  + Judge0 free-tier limits submissions to 50 per day.
  + A self-hosted Judge0 instance (via Docker) could remove this limitation; however, this is currently out of scope for the MVP but may be considered in future iterations.

## Communication Interface

Secure communication protocols are necessary for ReCode to offer interactive features and ensure data integrity:

* Protocols:
  + HTTP/HTTPS for communication between browser and API.
  + SSL/TLS encryption for all data transfers.
* Key Interfaces:
  + RESTful API calls between backend services (FastAPI, Supabase, Redis).
  + Webhooks for third-party events and analytics.
  + WebSockets for real-time updates (submission outcomes, coding progress).
  + Email reminders and authentication notifications.
* Security:
  + Encrypted storage and transmission of sensitive data.
  + All APIs are subject to strict role-based access control.
* Synchronization of data:
  + User progress, badges, and scores are synchronized in real time between frontend dashboards and backend services.

# 9. Additional Non-Functional Requirements

## Performance

**Key Performance Requirements:**

* Real-Time Responsiveness: The platform should be able to generate and release coding challenges within 5 seconds of a student request. Grading feedback should also return in under 3 seconds so that students don’t experience long delays while using the system.
* Scalability: For Recode, the system should support at least 200 students online at the same time. In the long run, it should be able to scale up to 5,000 users without slowing down feedback or grading speed.
* Reliability and Stability: The platform should run smoothly during academic sessions without crashing. If a failure does occur, it must recover quickly, keep student progress safe, and record the error for later review.
* Throughput: The grading system should handle around 50 submissions per hour during peak times without failing. Challenge generation and grading should run at the same time so one doesn’t delay the other.
* Data Persistence Efficiency: Student data, challenge results, and grading logs must be saved without interrupting the user experience. Saving should happen in the background, so the interface stays responsive.
* Precision and Alignment: Challenges created from lecture slides should match topics accurately. The topic extraction system (NLP) should reach at least 90% accuracy during MVP testing.

## Safety

The system should make sure that students’ work and lecturer content are protected from accidental loss or incorrect setup.

**Key Safety Requirements:**

* Data Protection Safety: Student submissions and grading feedback must always be saved safely. Automatic backups should be used so data can be restored if something goes wrong.
* Limit Configuration: Lecturers and admins can set up courses, challenges, and grading options. The system should warn if a wrong setup could affect normal student use.
* Compliance with Standards: The platform must follow academic and data protection standards such as POPIA.
* Emergency Stop: Lecturers or admins should be able to pause challenge generation or grading right away if there is a major problem, without having to shut down the entire system.

## Security

Security is important for protecting both users and data. The system should prevent unauthorised access, stop misuse, and keep data secure.

**Key Security Requirements:**

* Authentication and Authorisation: Only registered users should be able to access the system. Role-based access must be used, where students, lecturers, and admins each have specific permissions.
* Communication Security: All data sent between users, the backend, and third-party services should use HTTPS (TLS 1.2 or higher) to keep it safe from interception.
* Access Control: Student code must always run in a secure sandbox (Judge0) so malicious code can’t harm the system. Only approved actions such as grading and challenge generation should be allowed.
* Audit Logging: The system should keep a record of all logins, submissions, grading actions, and major changes, with timestamps and user IDs.
* Data Protection: Important data such as login details, lecture material, and grading results must be stored securely using encryption. Automated backups should also be in place.

## Software Quality

The overall quality of the system will affect how well it can be used and maintained in the future.

**Key Software Quality Requirements:**

* Uptime and Dependability: The platform should remain available during academic sessions and should not require frequent restarts.
* Error Handling: Errors such as failed logins, API issues, or submission problems must be handled properly, with clear error messages shown to the user and logs created for debugging.
* Maintainability: The codebase should be modular and follow consistent coding standards. Using containers like Docker will make deployment and scaling easier.
* Documentation: User manuals should be provided for both students and lecturers. Technical documentation for developers and administrators should also be available.
* Testing: The system should be tested regularly, including unit, integration, and system tests. Regression testing should also be done when new features are added to make sure old ones still work.