Functional Requirements

FR1: User Authentication and Roles

- **FR1.1**: System shall support user authentication for Admin, Lecturer, Tutor, and Student roles.
- **FR1.2**: System shall restrict access to features based on user roles.

FR2: Module Management

- **FR2.1**: Admin shall be able to create new modules.
- **FR2.2**: Admin shall be able to edit module details.
- **FR2.3**: Admin shall be able to delete modules.

FR3: Assignment Management

- **FR3.1**: Admin/Lecturer shall be able to create assignments for a module.
- **FR3.2**: Admin/Lecturer shall be able to edit assignment details.
- **FR3.3**: Admin/Lecturer shall be able to delete assignments.

FR4: Marking Script Management

- **FR4.1**: Admin/Lecturer shall be able to create marking scripts using:
  - **FR4.1.1**: GATLAM
  - **FR4.1.2**: Random Number Generator
  - **FR4.1.3**: Coverage-based algorithm
  - **FR4.1.4**: Manually
- **FR4.2**: Admin/Lecturer shall be able to delete marking scripts.
- **FR4.3**: Admin/Lecturer shall be able to edit marking scripts.
- **FR4.4**: Admin shall be able to upload custom interpreter
- **FR4.5**: Interpreter shall translate marking script into executable code depending on the marking script used
- **FR4.6**: Admin/Lecturer shall be able to set marking to manual mode

FR5: Mark Allocator

- **FR5.1**: A Mark Allocator shall be generated from the memo output
- **FR5.2**: Admin/Lecturer can edit the mark allocator
- **FR5.3**: Mark Allocator shall determine the weight of correct code for marking

FR6: Code Submission

- **FR6.1**: Students shall be able to upload their code files.

- **FR6.2**: Students shall receive marks and feedback for their submission

FR7: Reporting and Statistics

- **FR7.1**: System shall provide live statistics per assignment.
- **FR7.2**: Statistics shall be available as downloadable reports.
- **FR7.3**: Statistics shall be displayed in graph form.

FR8: Code Viewer and Runner

- **FR8.1**: System shall allow viewing code without downloading.
- **FR8.2**: System shall allow running code without downloading.
- **FR8.3**: System shall show output and stack trace of execution.

FR9: Execution Environment

- **FR9.1**: Student submissions shall be run in containerized environments.
- **FR9.2**: Student output shall be matched to marker output outside of the container.

FR10: Plagiarism Detection

- **FR10.1**: System shall support plagiarism detection per assignment.
- **FR10.2**: System shall compare ASTs before invoking MOSS.
- **FR10.3**: Plagiarism shall optionally be displayed in graph form

FR11: AI Assistance

- **FR11.1**: System shall provide AI-generated summaries of exceptions.
- **FR11.2**: System shall provide AI-generated summaries of incorrect outputs.

FR12: Gamification and Progression

- **FR12.1**: System shall support achievements and other gamified elements.
- **FR12.2**: System shall support unlocking tasks by completing previous tasks.

FR13: Grading System

- **FR13.1**: System shall calculate grades per assignment.
- **FR13.2**: System shall allow different grade weights per task.
- **FR13.3**: System shall display grades to students.
- **FR13.4**: System shall support time and space complexity analysis.

FR14: Submission Rules

- **FR14.1**: Admin shall be able to configure:
    - **FR14.1.1**: Submission deadlines (date and time)

    - **FR14.1.2**: Late submission policy
    - **FR14.1.3**: Submission count limit (including infinite)

FR15: Security

- **FR15.1**: System shall restrict student access to memo content.
- **FR15.2**: System shall isolate containers to prevent memo leakage.

FR16: Support System

- **FR16.1**: System shall have a ticketing system (Feature Flag enabled).

Quality Requirements

QR1: Performance

- **QR1.1**: The system must have an average code submission time of less than 10 seconds
- **QR1.2**: The system must be able to process and store up to 250 code submissions over a 12-hour period without performance degradation
- **QR1.3**: Graceful timeouts, if a job takes too long it is ended

QR2: Scalability

- **QR2.1**: The system architecture must support horizontal scaling of compute resources (e.g., worker nodes or containers) to handle increased load.
- **QR2.2**: Stateless API layer

QR3: Availability

- **QR3.1**: The system must maintain at least a 99.5% uptime during the semester
- **QR3.2**: The system must include real-time health monitoring for core components

QR4: Usability

- **QR4.1**: The system must provide a user interface on mobile devices
- **QR4.2**: The system must provide an accessible user interface on differently scaled viewports
- **QR4.2**: The system must provide users with active responses for loading, errors or successful execution

QR5: Security

- **QR5.1**: Student code execution must happen in sandboxed containers
- **QR5.2**: Communication must be secured with TLS

- **QR5.3**: Passwords and student information must be protected via hashing
- **QR5.4**: System sections must only be accessible to the appropriate roles

QR Justification

- **QR1.1**: Due to the amount of student submissions and the limited containers submission turnaround must be fast
- **QR1.2**: Based on class sizes and estimated peak daily usage, this amount must at least be able to be handled
- **QR1.3**: Prevents runaway/infinite loops in student code from monopolizing resources
- **QR2.1**: Deadlines produce load spikes, meaning that workers need to be scalable when demand is high and return to normal when it is low
- **QR2.2**: Statelessness makes it easy to add/remove API servers later
- **QR3.1**: Students need predictable access during the semester, and to reduce load on tutors and lecturer in the event of an extension
- **QR4.1**: Many students may access their profile using a tablet, it also makes it much easier for a tutor during a session to look at a student's mark if it is available on mobile
- **QR4.2**: Students use varied devices, responsive design avoids broken layouts
- **QR4.3**: Immediate, clear feedback improves user experience for both students and lecturers, and prevents accidental duplicate submissions
- **QR5.1**: Student code must not compromise the host system or other students' submissions
- **QR5.2**: Prevents eavesdropping, especially across a public network such as on campus
- **QR5.3**: Protects students passwords and privacy in compliance with POPIA
- **QR5.4**: Ensures lecturers, tutors, and students only see the functions appropriate to their roles, preventing unauthorized access and tampering with data