Software Requirements Specification

for

Learning Management System for the Mahapola Ports and Maritime Academy

Version 1.0 approved

Prepared by Team Laser

University Of Moratuwa

23/12/2024

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This document details the proposed Learning Management System for the Mahapola Ports and Maritime Academy. This will be the initial release of the System. This document outlines the system's functionalities, performance expectations, design constraints, and other relevant factors necessary for the development and deployment of the said Learning Management System.
This document will be a guide for the developers, the users of the Port Authority and the supervisors from the Faculty of Information Technology, University of Moratuwa.

1.2 Document Conventions

The abbreviations needed for the interpretation of the document are available in Appendix A.
The UML and ER Diagrams for the project have been created using draw.io (diagrams.net)
The UI/UX design for the system has been developed using Figma.

The priority for each use case will be considered the same unless specifically mentioned.

1.3 Intended Audience and Reading Suggestions

The document is intended for the following audiences.

- Developers
- Stakeholders from Mahapola Ports and Maritime Academy
- Faculty members from the Faculty of Information Technology, University of Moratuwa

It is recommended to read the document sequentially, starting with the introduction for context, then Appendix A for abbreviations.
Sections 4 and 5 contain the core functional and non-functional requirements, hence it is suggested to have special attention to those sections.

1.4 Product Scope

This LMS will provide a centralized platform for the MPMA to manage and deliver educational content, hold quizzes, track student progress, generate certificates, manage OJT, and facilitate communication between instructors and students. The initial scope focuses on a web-based application.

Key features included in this scope are:

- User management (administrators, lecturers, students).
- Overview Dashboards
- Course creation and management.
- Content uploading and delivery (documents, quizzes, etc.).

- Assessment creation and grading.
- Communication tools (announcements, chat).
- Reporting and analytics on student activity and course performance.

Future expansion, such as a mobile application, is outside the scope of this initial version but will be considered in the architectural design of the backend.

1.5 References

- The IEEE Standard SRS document template.
- Interactive Figma project for UI/UX designs:
  - https://www.figma.com/proto/3koUUP4cs8PWZx3Njcrf2J/Low-Fidelity-Diagram?node-id=0-1&t=WyHQ0IoD9BQ7kD81-1
  - https://www.figma.com/proto/3koUUP4cs8PWZx3Njcrf2J/Low-Fidelity-Diagram?node-id=180-667&t=WyHQ0IoD9BQ7kD81-1
  - https://www.figma.com/proto/3koUUP4cs8PWZx3Njcrf2J/Low-Fidelity-Diagram?node-id=5-2&t=WyHQ0IoD9BQ7kD81-1
- Draw.io project for diagrams:
  - Available on the Private Git Repository
  - https://github.com/RandithaK/mpma
- Material Toolkit for UI design

2. Overall Description

2.1 Product Perspective

This LMS is a new web application designed to replace existing manual processes used for learning management at the MPMA. It will be a central platform accessible through standard web browsers. While currently only accessible using a web browser, the backend architecture will be designed with potential future integration with other academy systems in mind such as the MPMA's existing attendance system. The backend will be built using Java Spring Boot, providing APIs that can potentially be connected to other applications, including a future mobile app.

2.2 Product Functions

The LMS will provide the following core functions:

- Initiation of the System with Super Administrator
- User Management: Creation, modification, and deletion of user accounts (administrators, lecturers, students), role assignment, and permission management.
- Course Management: Creation, modification, enrollment management, and lecturer and student assignment.
- Content Delivery: Uploading, organizing, and delivering several types of learning materials (documents, videos, presentations, links). Content access control based on user roles and course enrollment.
- Assessment and Grading: Creation of quizzes, assignments, and exams, automated grading for certain assessment types, manual grading capabilities, and gradebook management.

- Communication and Collaboration: Announcements, and potentially direct messaging between students and lecturers.
- Reporting and Analytics: Generation of reports on student progress, course completion rates, assessment scores, and user activity.
- Administrative dashboards for system overview.

2.3 User Classes and Characteristics

- **Administrator:**
  - Technical proficiency: Medium to High.
  - Responsible for system configuration, user management, course creation (potentially), and generating reports. Needs a comprehensive view of the system.
- **Lecturer:**
  - Technical proficiency: Medium.
  - Responsible for creating and managing course content, conducting assessments, grading student work, and communicating with students. Needs tools for content upload and interaction.
- **Student:**
  - Technical proficiency: Low to Medium.
  - Responsible for accessing course materials, submitting assignments, taking assessments, and communicating with lecturers. Needs a user-friendly interface for learning.

2.4 Operating Environment

- **Backend Server**: Linux Server
- **Backend Technology**: Java Spring Boot
- **Frontend Framework**: NextJS
- **Database**: PostgreSQL
- **Containerization**: Docker
- **Web Browsers**: Compatible with modern web browsers, both desktop and mobile (Chromium based, Safari and Firefox).
- **Hardware Platform:** VMs provided by the MPMA, all costs borne by MPMA

2.5 Design and Implementation Constraints

- **Technology Stack**: The system must be built using the specified technologies: Java Spring Boot (backend), NextJS (frontend), PostgreSQL (database), and Docker for deployment.
- **Database Choice**: PostgreSQL is the designated database system.
- **Operating System**: The backend server will run on a Linux environment.
- **UI/UX Design**: The user interface will have the functionalities drafted in the Figma design.
- **Diagramming**: System architecture and other relevant diagrams will be created using Draw.io considering long term support and the project being available for free.
- **Scalability**:
  - **Backend**: The backend architecture should be designed to accommodate potential future expansion, including the possibility of a mobile application. Implementing using a containerizing mechanism such as docker will make the system easily scalable.
  - **Frontend**: NextJS with React will be used as a future mobile app that can be implemented using React Native easily.

- **Security**: Appropriate security measures must be implemented to protect user data and prevent unauthorized access with protections against common attack vectors.
- Languages: English language will be supported

2.6 User Documentation

In-app tooltips will be provided as the system will inherently be self-explanatory.

2.7 Assumptions and Dependencies

- It is assumed that the necessary server infrastructure (Linux server) will be available for deployment as agreed in the initial meetings.
- The development environment (including necessary software and libraries) will be available for the development team.
- It is assumed that the MPMA will need a mobile application at some point and the system backend should be able to handle such an application with minimal change.

3. External Interface Requirements

3.1 User Interfaces

The user interface will be a web-based application accessed through standard web browsers mentioned earlier in section 2.4. The UI/UX design will be based on the mockups and prototypes created in Figma.

Key considerations include:

- **Intuitive Navigation**: Easy-to-understand menus and navigation patterns for all user roles.
- **Responsive Design**: The interface should be responsive and adaptable to different screen sizes and devices (desktops, laptops, tablets, and smart phones).
- **Accessibility**: Adherence to accessibility guidelines (WCAG2) to ensure usability for most users.
- **Consistent Design Language**: Maintaining a consistent look and feel throughout the application.

3.2 Hardware Interfaces

- Client-side: Standard desktop, laptop computers, tablets, smartphones, with a compatible web browser and stable internet connection.
- Server-side: The application will reside in Docker Containers on a compatible Linux server(s). Specific hardware specifications will depend on anticipated user load and data storage requirements.

3.3 Software Interfaces

- **Backend API**: Java Spring Boot will provide RESTful APIs for communication with the NextJS frontend. API documentation will be generated using tools like Swagger.
- **Database Interface**: PostgreSQL will be accessed through Java within the Spring Boot application.

- **External Integrations** (Future): Potential future integrations with other academy systems will be considered during API design.

3.4 Communications Interfaces

- **Protocol**: HTTP/HTTPS for web communication between the client and server.
- **Data Format**: JSON for data exchange between the frontend and backend APIs.
- **Security**: Secure communication over HTTPS will be enforced.
- **Notification Integration**: The system may need to push notifications (e.g., for assignment submissions, announcements).

4. System Features

4.1 User Management

4.1.1 Description and Priority:

- Allows administrators to manage user accounts, roles, and permissions. [High]

4.1.2 Stimulus/Response Sequences:

- Admin navigates to the user management page.
- Admin clicks "Add User," fills in user details including the account type, and clicks "Save." -> System creates a new user account.
- Admin selects a user and clicks "Edit," modifies user details, and clicks "Save." -> System updates user information.
- Admin selects a user and clicks "Delete" -> System prompts for confirmation and deletes the user account.

4.1.3 Functional Requirements:

- REQ-UM-1: The system shall allow administrators to create new user accounts with specified roles (Administrator, Lecturer, Student).
- REQ-UM-2: The system shall allow administrators to edit existing user account information (name, email, reset password, etc.).
- REQ-UM-3: The system shall allow administrators to delete user accounts.
- REQ-UM-4: The system shall enforce password complexity requirements.

4.2 Course Management

4.2.1 Description and Priority:

- Enables administrators to create and manage courses. [High]

4.2.2 Stimulus/Response Sequences:

- Administrator navigates to the course management page.
- Instructor clicks "Create Course," enters course details (name, description), and clicks "Save." -> System creates a new course.
- Admin or Lecturer (with Admin approval) can enroll/unenroll students in a course.

4.2.3 Functional Requirements:

- REQ-CM-1: The system shall allow administrators to create new courses.
- REQ-CM-2: The system shall allow administrators to edit course details (name, description, syllabus, start/date, duration).
- REQ-CM-3: The system shall allow administrators to allocate/deallocate lecturers to courses.
- REQ-CM-4: The system shall allow administrators and authorized lecturers to enroll and unenroll students in courses.

4.3 Material Delivery

4.3.1 Description and Priority:

- Provides a platform for lecturers to upload and manage learning materials for students. [High]

4.3.2 Stimulus/Response Sequences:

- Instructor navigates to a specific course.
- Instructor clicks "Add Content," selects a file (document, video, etc.), and uploads it. -> System stores the content.
- Student navigates to a course and views available content.

4.3.3 Functional Requirements:

- REQ-CD-1: The system shall allow lecturers to upload various file types (such as PDF).
- REQ-CD-2: The system shall allow lecturers to organize content into modules or sections within a course.
- REQ-CD-3: The system shall allow lecturers to set visibility for content.
- REQ-CD-4: The system shall provide students with a user-friendly interface to access and download course content.
- REQ-CD-5: The system shall provide mechanisms for embedding external content (such as links).

4.4 Assessment and Grading

4.4.1 Description and Priority:

- Enables lecturers to create and manage assessments (quizzes, and upload activities), and grade student submissions. [High]

4.4.2 Stimulus/Response Sequences:

- Instructor navigates to a course and clicks "Create Assessment."

- Instructor selects assessment type (quiz, and upload activities), defines questions, points, and due dates. -> System creates the assessment.
- Student accesses an assessment, answers questions, and submits.
- The lecturer reviews submissions and assigns grades.

4.4.3 Functional Requirements:

- REQ-AG-1: The system shall allow lecturers to create quizzes with various question types (multiple choice, true/false, short answer, etc.).
- REQ-AG-2: The system shall allow lecturers to create assignments with submission guidelines and deadlines.
- REQ-AG-3: The system shall allow for automated grading of objective quiz questions.
- REQ-AG-4: The system shall provide lecturer with tools to manually grade assignments and provide feedback. (online or download options)
- REQ-AG-5: The system shall maintain a gradebook for each course, displaying student scores and overall grades.

4.5 Communication and Collaboration

4.5.1 Description and Priority:

- Facilitates communication between lecturer and students within the LMS. [Medium]

4.5.2 Stimulus/Response Sequences:

- A lecturer creates an announcement for a course. -> Students enrolled in the course receive the announcement.
- A student posts a question to the lecturer and the lecturer can respond.

4.5.3 Functional Requirements:

- REQ-CC-1: The system shall allow lecturers to create and post announcements for specific courses.
- REQ-CC-2: The system may (future consideration) include a direct messaging feature between users.

4.6 Reporting and Analytics

4.6.1 Description and Priority:

- Provides administrators and instructors with insights into user activity and course performance. [Medium]

4.6.2 Stimulus/Response Sequences:

- Administrator navigates to the reporting dashboard.
- Administrator selects a report type (e.g., course completion rates) and specifies parameters. -> System generates the report.

4.6.3 Functional Requirements:

- REQ-RA-1: The system shall provide reports on student enrollment and course completion rates.
- REQ-RA-2: The system shall provide reports on student performance in assessments.
- REQ-RA-3: The system shall provide administrators with an overview dashboard of system usage and key metrics.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The system should respond to user actions within a reasonable time, without timeout.
- The system should be able to handle concurrent users without significant performance degradation.
- The system should be able to store and manage learning materials and user data.

Note: The number of concurrent users for which the system should be load tested has not been communicated yet.

5.2 Safety Requirements

- Data integrity must be maintained to prevent data loss or corruption.
- User authentication and authorization mechanisms must be robust to prevent unauthorized access.

5.3 Security Requirements

- User authentication will be required for access to the LMS, JWT will be implemented.
- User access levels and permissions will be strictly enforced to control access to specific features and data.
- Sensitive data (e.g., passwords) will be encrypted both in transit and at rest.
- The system will be protected against common web vulnerabilities.

5.4 Software Quality Attributes

- **Usability**: The system should be user-friendly and easy to navigate for all user roles.
- **Maintainability**: The codebase should be well-structured to facilitate future maintenance and updates.
- **Reliability**: The system should be stable and operate without frequent errors or failures.
- **Scalability**: The backend architecture should be scalable to accommodate future growth in users and data.
- **Testability**: The system should be designed to be easily tested at various levels (unit, integration, system).

5.5 Business Rules

- Only registered users can access the LMS, and access level is based on their role.

- Students cannot enroll in courses themselves and should be enrolled by the administrator.
- Lecturers can only manage courses they are assigned to.
- Administrators have full access to assign lectures and students and view analytics of the system, including the overall academic calendar.

6. Other Requirements

6.1 Database Requirements

- PostgreSQL database will be used to store all application data. Database schema will be designed to ensure data integrity and efficient querying. Regular database backups will be performed.
- The database will be shared with the proposed ERP system of the MPMA.

6.2 Future Considerations for Mobile Application

The backend API (developed with Java Spring Boot) should be designed with RESTful principles to facilitate communication with potential future mobile applications (e.g., using React Native).

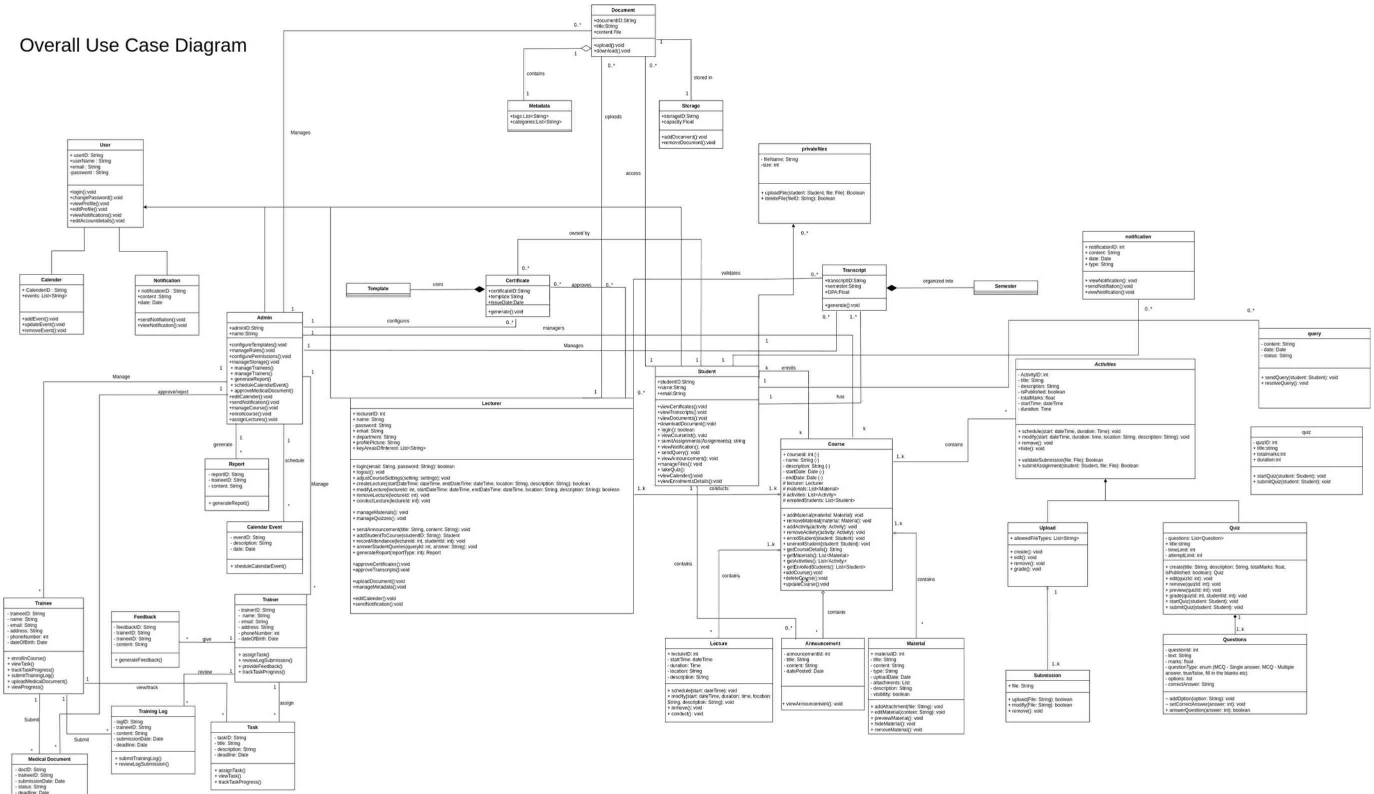
Appendix A: Glossary

Abbreviations needed for interpretation of the Project

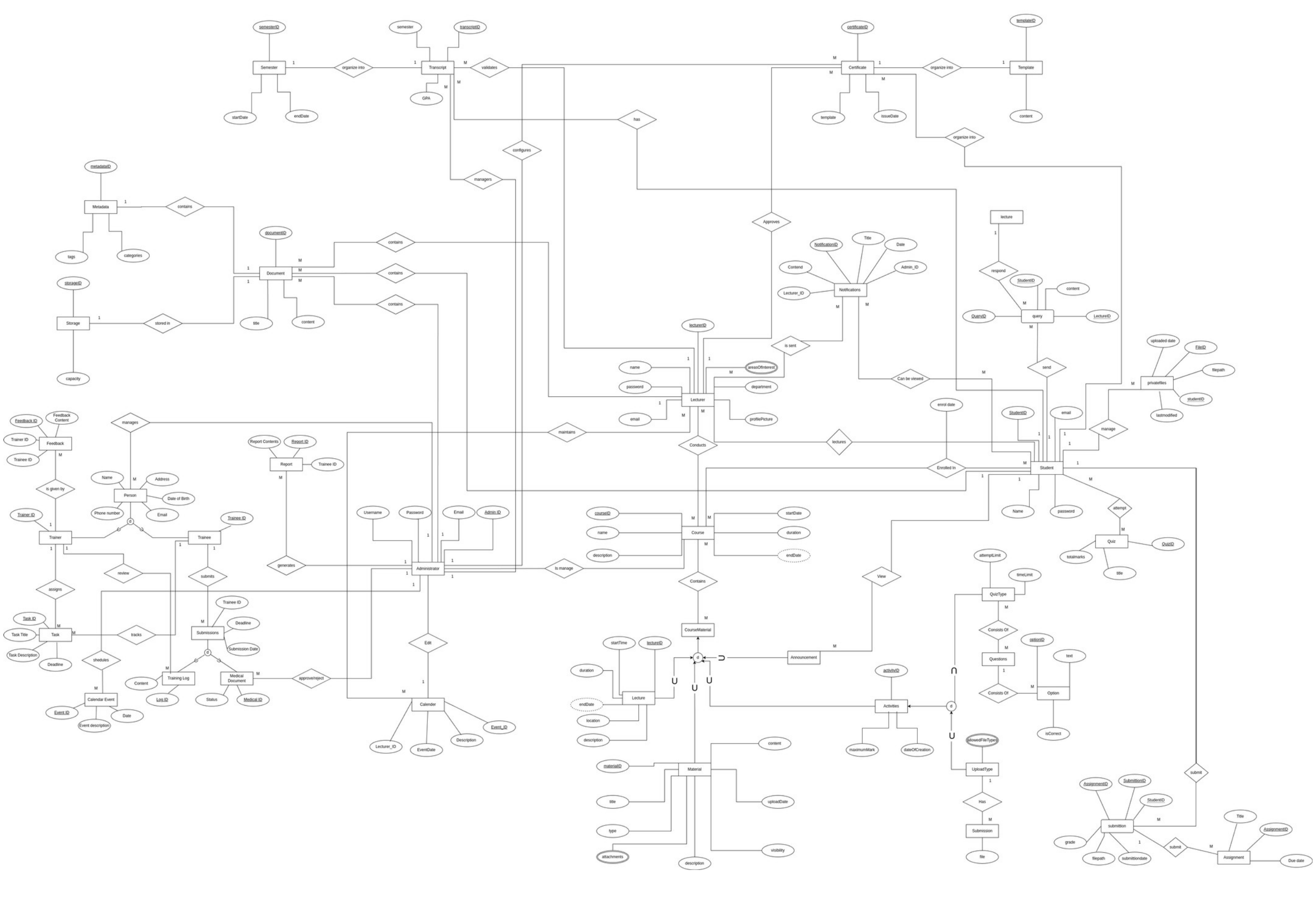
- MPMA – Mahapola Port & Maritime Academy
- SLPA – Sri Lanka Ports Authority
- OJT – On-the-Job Training
- VM – Virtual Machine
- UML – Unified Modeling Language
- EER – Enhanced Entity Relationship
- ERP – Enterprise Resource Planning
- LMS – Learning Management System
- API – Application Programming Interface
- UI – User Interface
- UX – User Experience
- JSON – JavaScript Object Notation
- HTTP – Hypertext Transfer Protocol
- HTTPS – Hypertext Transfer Protocol Secure

Appendix B: Analysis Models

Overall Use Case Diagram



Overall Enhanced Entity Relationship Diagram



Appendix C: To Be Determined List

- The system load was not mentioned in the initial discussion.
- Color preferences of the MPMA were not mentioned to us.