POLITEHNICA UNIVERSITY OF BUCHAREST

SOFTWARE ENGINEERING

Moodle++

Software Design Document

Team:

Stefanescu Bogdan-Laurentiu, gr.1241A
Stan-Soponaru Ioan-Alexandru, gr. 1241A
Cristea George-Razvan, gr. 1241A

Coordinator:

Prof. Nicolae GOGA

Date created: 25.11.2024

Saturday, January 18, 2025

Delivery Report

(will be delivered along with the project)

Name	Group	Project implementation [%, reason]	Signature
Stefanescu Bogdan-Laurentiu	1241A	33%	
Stan-Soponaru Ioan-Alexandru	1241A	33%	
Cristea George-Razvan	1241A	33%	

Delivery date: 12.12.2024

Table of Contents

System Design

According to the IEEE STD-1016-1998, *IEEE Recommended Practice for Software Design Descriptions*.

Document Change History

Depending on the chosen type of development, retention of all changes made to this document may be useful. For example, for a project using the *Waterfall w/ milestones* methodology, all the changes made between two project evaluation moments (*milestones*) will be retained. A chronological sorting of the list of changes is indicated.

Version	Date	The author/authors of the change	Details of changes
1.00	25.11.2024	Cristea Razvan, Stefanescu Bogdan, Stan-Soponaru Alexandru	Initial version/started working on the project
1.01	1.12.2024	Cristea Razvan, Stefanescu Bogdan, Stan-Soponaru Alexandru	Complete the **introduction** chapter
1.02	2.12.2024	Cristea Razvan, Stefanescu Bogdan, Stan-Soponaru Alexandru	Complete the **references** chapter and start working on the **decomposition description** chapter.
1.03	4.12.2024	Cristea Razvan, Stefanescu Bogdan, Stan-Soponaru Alexandru	Started working on the **dependency description** and **interface description** chapters.
1.04	7.12.2024	Cristea Razvan, Stefanescu Bogdan, Stan-Soponaru Alexandru	Continue working on **dependency** and **interface descriptions**. Begin working on the **detailed design** chapter.
1.05	9.12.2024	Cristea Razvan, Stefanescu Bogdan, Stan-Soponaru Alexandru	Complete **dependency** and **interface descriptions** chapters, as well as **detailed design** chapter. Begin working on the diagrams.
1.06	11.12.2024	Cristea Razvan, Stefanescu Bogdan, Stan-Soponaru Alexandru	Make final edits and evaluate the document.

1. Introduction

1.1. Purpose

State the purpose of this document: the presentation of the design of a system that solves the requirements of the proposed project.

The goal of this paper is to give the design specifications for the **Moodle++** platform, which is an upgraded learning management system (LMS) designed to satisfy the demands of modern education. This platform seeks to provide an intuitive, accessible, and scalable environment in which professors can quickly disseminate instructional materials and assignments while students can easily access, submit, and track their activities. **Moodle++** provides a dependable, secure, and user-friendly experience for all stakeholders by leveraging solid software design and adherence to worldwide norms.

1.2. Target Public

Describe the people that this document is addressed to, depending on the development model chosen for the project. For example, for a project using the *Waterfall w/ milestones* methodology, the target public may consist of *programmers, designers* and *project managers*. On the other hand, the absence of the *client* from the target public can be specified.

This document is intended for the following audiences:

- **Developers** oversee developing the platform, which includes both backend and frontend development.
- **System designers** are responsible for converting these requirements into a unified and scalable architecture.
- **Project managers** oversee the development process and ensure that milestones are met.
- **Quality Assurance Engineers** verify the system against the specifications to guarantee functionality and compliance.
- **End-users** (teachers and students) are excluded from the target audience because this publication is focused on technical specifications rather than user assistance.

1.3. Definitions, Acronyms and Abbreviations

The list of the terms used in this document. This document can be read by persons without a technical background, like the personnel from the marketing department.

- **GDPR**: General Data Protection Regulation
- **LMS**: Learning Management System
- **MFA**: Multi-Factor Authentication
- **SDD** (Software Design Document): A document outlining the software architecture, modules, processes, and interfaces of the system being developed.
- **IEEE**: Institute of Electrical and Electronics Engineers, a professional organization that sets standards for software design and other engineering disciplines.
- **HTTP** (Hypertext Transfer Protocol): The protocol used for transmitting hypermedia documents, such as websites, over the internet.
- **UI** (User Interface): The web-based interface that allows users to interact with the dispenser system through a browser.

- **Module**: A logical component of the system that performs a specific function.
- **Process**: A set of activities or operations executed to achieve a specific task
- **Authentication**: The process of verifying the identity of a user or system, typically involving credentials like a username and password or other secure tokens.
- **URL**: Uniform Resource Locator - The address used to access a resource on the internet, such as a web page or API endpoint.
- **JavaScript**: A programming language primarily used to create dynamic and interactive elements on websites.
- **TypeScript**: A superset of JavaScript that adds static typing, enabling developers to catch errors during development rather than runtime.
- **Next.js**: A React-based web development framework for building server-side rendered and static web applications. It enhances performance and development speed.
- **React**: Facebook developed and maintains a JavaScript library for designing user interfaces. It is component-based, allowing developers to design reusable UI elements, and is widely used for developing single-page applications (SPAs). React employs a virtual DOM to optimise rendering and updates, resulting in great performance and efficient UI changes.
- **NextAuth.js**: An open-source authentication library for Next.js that provides pre-built authentication strategies, including OAuth, JWT, and credentials-based authentication.
- **Tailwind CSS**: A utility-first CSS framework that provides predefined classes to style web applications quickly and consistently.
- **MySQL**: An open-source relational database management system used for storing and managing structured data.
- **S3 Bucket**: A cloud-based storage solution provided by Amazon Simple Storage Service (S3) for storing and retrieving data objects at scale.
- **AWS SDK**: Amazon Web Services Software Development Kit - A collection of tools and libraries enabling integration with AWS services like S3, Lambda, and EC2.
- **Prisma ORM**: An Object-Relational Mapping (ORM) tool that simplifies database operations by providing a type-safe API for interacting with databases.
- **Metadata**: Data that provides information about other data, such as file attributes, database schema details, or API descriptors.
- **JWT**: JSON Web Token - A compact, URL-safe method for securely transmitting information between parties as a JSON object, commonly used for authentication.

1.4. Document Structure

Describe the structure of the document, with one phrase for each chapter.

1**. Introduction**: Provides an outline of the project's objective and intended audience.

2. **References**: A list of standards, laws, and other documents that give background or recommendations for the project.

3. **Decomposition Description**: A complete dissection of the system's modules, processes, and data items.

4. **Dependency Description**: Identifying and explaining the dependencies between modules, processes, and data items.

5. **Interface Description**: A complete summary of the system's module and process interfaces.

6. **Detailed Design**: Comprehensive descriptions of modules and data elements, such as class diagrams and method details.

7. **Appendices**: Additional information such as diagrams, document evolution records, and conclusions.

2. References

List of references used in this paper, possible legislation that governs the application domain of the project/document.

[1] **IEEE STD**-1016-1998, *IEEE Recommended Practice for Software Design Descriptions*

[2] **Data Protection Laws**: The platform must abide by applicable data protection laws, such as the General Data Protection Regulation (GDPR) in the EU if it gathers any user data (such as grades or personal information). These regulations control the gathering, storing, and processing of user data.

Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)

[3] **European Accessibility Act**: This law guarantees that all users, including those with impairments, can use websites and online platforms throughout the European Union. It covers organizations in the public sector as well as other online services, such as learning environments.

Directive (EU) 2019/882 of the European Parliament and of the Council of 17 April 2019 on the accessibility requirements for products and services (Text with EEA relevance)

[4] **Authentication**: The process of validating a user's or system's identity, generally using credentials such as a login and password or other secure tokens. Authentication guarantees that only authorised users can access certain resources or activities.

RFC 4949 - Internet Security Glossary, Version 2, Internet Engineering Task Force (IETF), 2007.

[5] **URL (Uniform Resource Locator)**: The address used to locate internet resources such as web pages, APIs, and media files. URLs are crucial for browsers and programs to obtain specific data from web servers.

RFC 1738 - Uniform Resource Locators (URL), Tim Berners-Lee, Larry Masinter, and Mark McCahill, Internet Engineering Task Force (IETF), 1994.

[6] **JavaScript**: A programming language commonly used to create interactive and dynamic web content. JavaScript is compatible with both client-side and server-side environments, making it useful for web development.

ECMAScript® 2023 Language Specification (12th Edition), ECMA-262, ECMA International, 2023.

[7] **TypeScript**: A superset of JavaScript that includes static typing to improve code quality and reduce runtime mistakes during software development.

TypeScript Official Documentation, Microsoft Corporation.

[8] **Next.js**: A React-based framework for creating server-rendered, statically generated web applications. It offers optimal performance and scalability.

Next.js Documentation, Vercel.

[9] **NextAuth.js**: An authentication library for Next.js applications that supports several authentication methods, including OAuth, JWT, and custom credentials.

NextAuth.js Documentation.

[10] **Tailwind CSS**: A utility-first CSS framework for building custom user interfaces quickly with pre-designed classes and consistent styling.

Tailwind CSS Documentation, Tailwind Labs.

[11] **MySQL**: An open-source relational database management system that handles structured data and supports SQL queries.

MySQL Documentation, Oracle Corporation.

[12] **S3 Bucket**: AWS provides a scalable cloud-based storage solution for storing and retrieving massive volumes of data items.

Amazon S3 Documentation, Amazon Web Services (AWS).

[13] **AWS SDK**: A collection of software development tools that integrate with Amazon Web Services (AWS) for cloud-based operations.

AWS SDK Documentation, Amazon Web Services (AWS).

[14] **Prisma ORM**: An object-relational mapping (ORM) tool that simplifies database interactions through type-safe APIs and schema-driven development.

Prisma Documentation.

[15] **JWT (JSON Web Token)**: A concise, secure, and URL-safe way for sending information between parties as a JSON object, which is widely used for authentication.

RFC 7519 - JSON Web Token (JWT), Internet Engineering Task Force (IETF), 2015.

[16] **React**: A JavaScript library for creating user interfaces, including a component-based architecture and a fast virtual DOM for dynamic rendering.

React Documentation, Meta Platforms, Inc.

3. Decomposition Description

This article corresponds to chapter 6.2.1., Decomposition description, from [1].

3.1. Modules Description

Modules description according to 5.3.1 -5.3.10, from [1]. The modules can be identified on the use case diagram(s) from the Software Requirements Specification.

3.1.1. Description of Module 1

Name	User Interface Module
Type	Code Module
Purpose	Provides an intuitive and responsive interface for users to interact with the system.
Way of operating	The module renders views for students, teachers, and admins, enabling operations like logging in, managing courses, submitting assignments, and tracking attendance. It dynamically adjusts to user roles and device types (e.g., mobile, desktop).
Subordination	Integrates with the Account Management Module for authentication and role-based content delivery. Subordinates to other modules for displaying data like assignments, grades, and attendance.
Dependencies	Backend Modules for data retrieval and actions.
Resources	JavaScript/TypeScript, Next.js, Tailwind CSS

3.1.2. Description of Module 2

Name	File Manager Module
Type	Code Module
Purpose	Handles file storage, retrieval, and management in S3-compatible Server for user assignments, resources, and other uploaded content.
Way of operating	- Receives files from users (students/teachers) through the UI. - Validates file type, size, and permissions. - Generates pre-signed URLs for secure file uploads and downloads.
Subordination	Assignment Module, Course Module, User Module
Dependencies	- S3 Compatible Server for cloud storage. - MySQL Database for storing file metadata. - AWS SDK for file operations (upload, delete, generate URLs).

Resources	- S3 bucket configured with proper policies. - AWS SDK for JavaScript, - Next.js Backend - Database for metadata storage (MySQL). - User authentication service.

3.1.3. Description of Module 3

Name	Assignment and Grading Module
Type	Code Module
Purpose	To manage assignments, submissions, and grading functionality for students and teachers.
Way of operating	Teachers create assignments and specify due dates. Students submit tasks which are graded by teachers.
Subordination	The course module provides context for assignments.
Dependencies	Requires Assignment, Grade, and Submission tables in the database
Resources	Prisma ORM

3.1.4. Description of Module 4

Name	Course Module
Type	Code Module
Purpose	Manages subjects, courses, and student attendance within those subjects.
Way of operating	Admin creates subjects Teacher creates sessions (courses) Teacher manages student attendance
Subordination	Student and Teacher Management modules provide participants.
Dependencies	Requires Subject, Course, and Attendance tables in the database.
Resources	Prisma ORM

3.1.5. Description of Module 5

Name	Account Management Module
Type	Code Module

Purpose	Handles user accounts, roles, authentication, and session tracking.
Way of operating	Users login using external accounts and are assigned roles (Student, Teacher, Admin).
Subordination	Required for all other modules to manage user roles and authentication.
Dependencies	NextAuth.js
Resources	NextAuth.js, Prisma ORM, JWT for session management.

3.1.6. Description of Module 6

Name	Student and Teacher Management Module
Type	Code Module
Purpose	Manages teacher assignments to subjects and student enrolments in groups and subjects.
Way of operating	Teachers are assigned to subjects and courses.
Subordination	It depends on Subject and Course modules for creating associations.
Dependencies	Database tables for Student, Teacher, Group, and Enrolment.
Resources	Prisma ORM

3.2. Description of Concurrent Processes

Description of concurrent processes according to 5.3.1 – 5.3.10, from [1]. The processes can be identified on the sequence diagram(s) from the Software Requirements Specification.

3.2.1. Description of Process 1

Name	User Authentication and Authorization
Type	Process
Purpose	Verify user credentials and assign roles.
Way of operating	Authenticate user using external platforms and assign roles from the internal database.
Subordination	
Dependencies	Integration with OAuth for secure login User data module for storing the roles of the user
Resources	MySQL, NextAuth.js, OAuth

3.2.2. Description of Process 2

Name	Assignment Submission
Type	Process
Purpose	Handles student submissions of assignments and notifies teachers.
Way of operating	Students upload files via the UI. The system validates file size and type, stores the file in S3, and records the submission in the database.
Subordination	User Authentication and Authorisation
Dependencies	File Management Module for storing files, database for submission records.
Resources	Prisma ORM, S3 bucket, File APIs for upload and retrieval.

3.2.3. Description of Process 3

Name	Course Enrolment
Type	Process
Purpose	Manages the enrolment of students into courses and subjects.
Way of operating	Students or admins select courses for enrolment. The system updates the database and generates relevant links for subjects and assignments.
Subordination	User Authentication and Authorisation
Dependencies	User Management Module for student data, Subject Module for available courses.
Resources	Prisma ORM, database for storing enrolment records.

3.2.4. Description of Process 4

Name	Attendance Recording
Type	Process
Purpose	Tracks and records student attendance for subject sessions.
Way of operating	Students scan QR codes or manually check-in. Attendance records are stored in the database and linked to subjects.
Subordination	User Authentication and Authorisation, Course Enrolment
Dependencies	User Management Module for student data, Subject Module for available courses.
Resources	Database for storing attendance records, QR code generator/reader.

3.2.5. Description of Process 5

Name	Grading
Type	Process
Purpose	Assign grades to students based on their performance in assignments and exams.
Way of operating	Teachers enter grades via the UI or upload CSV files. Grades are stored in the database and linked to assignments and students.
Subordination	User Authentication and Authorisation, Course Enrolment
Dependencies	Assignment Module for assignment data, database for storing grades.
Resources	Prisma ORM, database for storing and retrieving grades

3.2.6. Description of Process 6

Name	File Upload
Type	Process
Purpose	Handles file uploads from users and stores them securely in S3.
Way of operating	Files are uploaded through the UI. Metadata is validated and stored in the database, while content is sent to the S3 bucket.
Subordination	User Authentication and Authorisation, Course Enrolment
Dependencies	S3 bucket for storage, File Management Module for metadata handling.
Resources	AWS SDK, S3 bucket, Prisma ORM for database operations.

3.3. Description of Data Modules

Description of data modules according to 5.3.1 – 5.3.10, from [1]. The data modules can be identified on the use case diagram(s) from the Software Requirements Specification.

3.3.1. Description of Data Module 1

Name	Assignment Data
Type	Data module
Purpose	Manage assignment-related data, including details, due dates, and types (homework, quiz, project).
Way of operating	Data is created by teachers, accessed by students, and linked to submissions and grades.
Subordination	User Data
Dependencies	Dependencies include Subject and Grade modules for contextual data.
Resources	Relational database (MySQL/PostgreSQL), Prisma ORM for queries.

3.3.2. Description of Data Module 2

Name	User Data
Type	Data module
Purpose	Stores and manages user information such as profiles, roles, and permissions.
Way of operating	User data is retrieved and updated through APIs and securely stored in the database.
Subordination	
Dependencies	Dependences include authentication services and session tracking modules.
Resources	Relational database (MySQL/PostgreSQL), Prisma ORM for queries.

3.3.3. Description of Data Module 3

Name	Grade Data
Type	Data module
Purpose	Stores and manages grades assigned to students for various assignments and activities.
Way of operating	Teachers input grades via the UI, which are stored in the database and retrieved for reports or analysis.
Subordination	User Data, Course Data
Dependencies	Dependencies include Assignment and User modules.
Resources	Relational database (MySQL/PostgreSQL), Prisma ORM for queries.

3.3.4. Description of Data Module 4

Name	Attendance Data
Type	Data module
Purpose	Tracks and stores student attendance for subject sessions.
Way of operating	Attendance is recorded through QR code scanning or manual entry and stored in the database.
Subordination	User Data, Course Data
Dependencies	Dependences include Subject and User modules for contextual information.
Resources	Relational database (MySQL/PostgreSQL), Prisma ORM for queries.

3.3.5. Description of Data Module 5

Name	File Metadata
Type	Data module
Purpose	Manages metadata for user-uploaded files, including file names, types, and links to S3 storage.
Way of operating	Metadata is stored in the database and linked to user accounts and assignments.
Subordination	User Data, Course Data
Dependencies	S3 bucket for file storage, User and Assignment modules for context.
Resources	Relational database (MySQL/PostgreSQL), S3 storage for file content.

3.3.6. Description of Data Module 6

Name	Course Data
Type	Data module
Purpose	Stores data about subjects, courses, and their associations with teachers and students.
Way of operating	Data is created and modified by teachers or admins. Linked with student enrolments and attendance.
Subordination	User Data
Dependencies	Dependencies include User, Enrolment, and Attendance modules.
Resources	Relational database (MySQL/PostgreSQL), Prisma ORM for queries

3.3.7. Description of Data Module 7

Name	Enrolment Data
Type	Data module
Purpose	Tracks student enrolments in subjects and groups.
Way of operating	Data is updated during enrolment or course assignment and linked to the subject and user records.
Subordination	User Data, Course Data
Dependencies	Dependencies include Subject, Group, and User modules.
Resources	Relational database (MySQL/PostgreSQL), Prisma ORM for queries.

4. Dependency Description

This chapter corresponds to chapter 6.2.2, *Dependency description*, from [1].

4.1. Dependencies among modules

The dependencies identified in chapter 3.1 are detailed. Detailed diagrams are recommended.

4.1.1. User Interface Module

- **Depends On**:
  - Account Management Module: For user authentication and role-based content delivery.
  - Backend Modules: For retrieving and displaying data like assignments, grades, and attendance.

4.1.2. File Manager Module

- **Depends On**:
  - Assignment Module: For linking uploaded files to specific assignments.
  - Course Module: For associating course-related files.
  - User Module: For managing ownership of uploaded files.

4.1.3. Assignment and Grading Module

- **Depends On**:
  - Course Module: Provides context for assignments and grading.
  - File Manager Module: For attaching files to assignments.
  - User Module: For linking grades to students and assignments.

4.1.4. Course Module

- **Depends On**:
  - Student and Teacher Management Module: For associating students and teachers with courses.
  - Subject and Attendance Data Modules: For managing course content and attendance tracking.

4.1.5. Account Management Module

- **Depends On**:
  - User Data Module: For storing and managing user credentials and roles.
  - Session Management: For tracking active user sessions.

4.1.6. Student and Teacher Management Module

- **Depends On**:
  - Subject and Course Modules: For assigning teachers and enrolling students.
  - User Module: For managing student and teacher roles.

4.2. Dependences among processes

The dependencies identified in chapter 3.2 are detailed. Detailed diagrams are recommended.

4.2.1. User Authentication and Authorization Process

- **Depends on**:
    - **Account Management Module** for verifying user credentials and assigning roles.
    - **User Data Module** for storing and retrieving user roles and authentication data.

4.2.2. Assignment Submission Process

- **Depends on**:
    - **User Authentication and Authorization Process** for ensuring that users are authenticated before submitting assignments.
    - **File Management Module** for storing files and generating secure file URLs.
    - **Assignment Data Module** for linking submitted assignments to specific courses and subjects.

4.2.3. Course Enrollment Process

- **Depends on**:
    - **User Authentication and Authorization Process** to ensure that only authenticated students can enroll in courses.
    - **Course Data Module** for storing and managing enrollment records.
    - **Subject Data Module** for ensuring that available courses are linked to the correct subjects.

4.2.4. Attendance Recording Process

- **Depends on**:
    - **User Authentication and Authorization Process** for validating the student.
    - **Course Enrollment Process** to ensure that the student is enrolled in the subject/course they are marking attendance for.
    - **Attendance Data Module** for recording and storing attendance details.

4.2.5. Grading Process

- **Depends on**:
    - **User Authentication and Authorization Process** to authenticate the teacher assigning grades.
    - **Course Enrollment Process** for associating grades with student enrollments.
    - **Assignment Data Module** for retrieving assignment-related information and calculating grades.

4.2.6 File Upload Process

- **Depends on**:

  - **User Authentication and Authorization Process** for ensuring the user is authenticated before uploading files.
  - **Course Enrollment Process** for associating uploaded files with specific courses and subjects.
  - **File Metadata Module** for storing file-related metadata in the database.
  - **File Management Module** for storing files in S3.

4.3. Dependencies among data modules

The dependencies identified in chapter 3.3 are detailed. Detailed diagrams are recommended.

4.3.1. Assignment Data Module

- **Depends on**:
  - **User Data Module** for linking assignments to the teachers who created them.
  - **Course Data Module** for associating assignments with specific courses.
  - **Grade Data Module** for storing grades related to assignments.

4.3.2. User Data Module

- **Depends on**:
  - **Account Management Module** for linking user credentials and roles.
  - **Session Management Module** for handling user sessions and tracking user activity.

4.3.3. Grade Data Module

- **Depends on**:
  - **Assignment Data Module** for storing grades assigned to assignments.
  - **User Data Module** for linking grades to specific students.

4.3.4 Attendance Data Module

- **Depends on**:
  - **User Data Module** for storing student attendance data.
  - **Course Data Module** for linking attendance to specific courses and subjects.

4.3.5. File Metadata Module

- **Depends on**:
  - **User Data Module** for linking files to specific users.
  - **Assignment Data Module** for associating files with assignments and grading.
  - **Course Data Module** for linking files to specific courses.

4.3.6. Course Data Module

- **Depends on**:
  - **Subject Data Module** for creating and managing courses under specific subjects.
  - **User Data Module** for associating students and teachers with courses.
  - **Enrolment Data Module** for managing student enrollments in courses.

4.3.7. Enrolment Data Module

- **Depends on**:

- **Course Data Module** for linking students to courses they are enrolled in.
- **Subject Data Module** for associating enrolled students with subjects.
- **User Data Module** for managing student records and enrollments.

5. Interface Description

This chapter corresponds to chapter 6.2.3, *Interface description,* from [1].

5.1. Module Interfaces

The module interfaces described in chapter 3.1 are detailed. Detailed diagrams are recommended.

5.1.1. Module 1 Interface

Name	User Interface Module
Type	Code Module
Purpose	Provides an intuitive and responsive interface for users to interact with the system.
Way of operating	The module renders views for students, teachers, and admins, enabling operations like logging in, managing courses, submitting assignments, and tracking attendance. It dynamically adjusts to user roles and device types (e.g., mobile, desktop).
Interface 1	Description: This interface handles the objects that will be displayed for the user.

5.1.2. Module 2 Interface

Name	File Management System
Type	Code Module
Purpose	Handles file storage, retrieval, and management in S3-compatible Server for user assignments, resources, and other uploaded content.
Way of operating	- Receives files from users (students/teachers) through the UI. - Validates file type, size, and permissions. - Generates pre-signed URLs for secure file uploads and downloads.
Interface 1	Description: This interface handles file uploads. getUploadKey() Input: File Metadata Output: S3UploadKey

5.1.3. Module 3 Interface

Name	Assignment and Grading Module
Type	Code Module
Purpose	To manage assignments, submissions, and grading functionality for students and teachers.
Way of operating	Teachers create assignments and specify due dates. Students submit tasks which are graded by teachers.
Interface 1	Description: This interface is handled by Prisma.

5.1.4. Module 4 Interface

Name	Course module
Type	Code Module
Purpose	Allow teachers to upload materials, assign grades, and manage class tasks.
Way of operating	Admin creates subjects Teacher creates sessions (courses) Teacher manages student attendance
Interface 1	Description: This interface is handled by Prisma.

5.1.5. Module 5 Interface

Name	Account Management Module
Type	Code Module
Purpose	Handles user accounts, roles, authentication, and session tracking.
Way of operating	Users login using external accounts and are assigned roles (Student, Teacher, Admin).
Interface 1	Description: This interface gets the user id from the current session and gets the student that matches that id: getCurrentStudent() Input: None Output: studentObject

5.1.6. Module 6 Interface

Name	Student and Teacher Management Module
Type	Code Module
Purpose	Manages teacher assignments to subjects and student enrolments in groups and subjects.
Way of operating	Teachers are assigned to subjects and courses.

Interface 1	Description: This interface gets the role from the current user's session: getRole() Input: None Output: String

5.2. Processes Interfaces

The processes interfaces described in chapter 3.2 are detailed. Detailed diagrams are recommended.

5.2.1. Process 1 Interface

Name	User Authentication Process
Type	Process
Purpose	To verify user credentials and roles before granting access to platform functionalities.
Way of operating	Authenticate user using external platforms and assign roles from the internal database.
Interface 1	*handleClientRequest(client)* Input: username: String – User's login username. password: String – User's login password. Output: Authentication token (valid for session duration). Role information (e.g., student, teacher, admin). Description: This interface verifies the user and decides whether to grant them access or not.

5.2.2. Process 2 Interface

Name	Assignment Submission
Type	Process
Purpose	Handles student submissions of assignments and notifies teachers.
Way of operating	Students upload files via the UI. The system validates file size and type, stores the file in S3, and records the submission in the database.
Interface 1	*uploadAssignment(formdata, course, assignment)* Input: formdata: FormData: User's data to upload. Course: The specified assignment's course. Assignment: The assignment at which to upload the file. Output: Redirect: returns a redirect to a certain page depending whethe**r** the upload was successful or not. Description: This interface handles the upload of an assignment.

5.2.3. Process 3 Interface

Name	Course Enrollment
Type	Process
Purpose	Manages the enrolment of students into courses and subjects.
Way of operating	Students or admins select courses for enrolment. The system updates the database and generates relevant links for subjects and assignments.
Interface 1	*enroll(role, name)* Input: User role: String: Specified user role User Name: String: Specified User Name Output: Link: a link to enroll. Description: This interface handles a file upload.

5.2.4. Process 4 Interface

Name	Attendance Recording
Type	Process
Purpose	Tracks and records student attendance for subject sessions.
Way of operating	Students scan QR codes or manually check-in. Attendance records are stored in the database and linked to subjects.
Interface 1	*enroll(role, name, course)* Input: Role: String: Specified user role Name: String: Specified User Name Course: String: The specified course to enroll to. Output: Link: a link to enroll. Description: This interface handles the enrollment of a student or teacher to a course.

5.2.5. Process 5 Interface

Name	Grading
Type	Process
Purpose	Assigns grades to students based on their performance in assignments and exams.
Way of operating	Teachers enter grades via the UI or upload CSV files. Grades are stored in the database and linked to assignments and students.
Interface 1	*grade(name, course, assignment)* Input: Name: String: The name of the student to grade. Course: String: The course to grade the student at. Assignment: String: The assignment to grade.

	Output: Response: returns a response depending on if the grading was successful or not. Description: This interface handles the grading given by a teacher to a student.

5.2.6. Process 6 Interface

Name	File Upload
Type	Process
Purpose	Handles file uploads from users and stores them securely in S3.
Way of operating	Files are uploaded through the UI. Metadata is validated and stored in the database, while content is sent to the S3 bucket.
Interface 1	*upload(formdata)* Input: formdata: FormData: User's data to upload. Output: Redirect: returns a redirect to a certain page depending wether the upload was successful or not. Description: This interface handles a file upload.

6. Detailed Design

This chapter corresponds to chapter 6.2.3, *Interface description*, from [1].

6.1. Modules detailed design

The modules described in chapter 3.1 are detailed. Detailed diagrams are recommended.

6.1.1. Module 1

Name	Name of module 1 (see 3.1.1)
Type	(see 3.1.1)
Purpose	(see 3.1.1)
Way of operating	(see 3.1.1)
Classes	Identification of constitutive classes

6.1.1.1. Module 1, class 1

Name	Name class 1
Purpose	Description of the purpose of this class within the module
Members	Identification of the class' members
Methods	Identification of the constitutive methods (functions)

Class diagram (see class diagrams from the Software Requirements Document).

6.1.1.1.1. Module 1, class 1, method 1

Name	Name of method 1
Purpose	Description of the purpose of this method within the class
Prototype	Description of the method's prototype
Input	Description of the method's input
Output	Description of the method's output
Caller	Routines that call this method
Calls	Other routines called by this method
Algorithm	Description of the algorithm/procedure of operating

...

6.1.2. Module 2

...

6.1.1. Module 1

Name	User Interface Module (see 3.1.1)

Type	Code Module (see 3.1.1)
Purpose	Provides an intuitive and responsive interface for users to interact with the system. (see 3.1.1)
Way of operating	- The module renders views for students, teachers, and admins, enabling operations like logging in, managing courses, submitting assignments, and tracking attendance. It dynamically adjusts to user roles and device types (e.g., mobile, desktop). (see 3.1.1)
Classes	None

6.1.2. Module 2

Name	File Manager Module (see 3.1.1)
Type	Code Module (see 3.1.1)
Purpose	Handles file storage, retrieval, and management in S3-compatible Server for user assignments, resources, and other uploaded content. (see 3.1.1)
Way of operating	- Receives files from users (students/teachers) through the UI. - Validates file type, size, and permissions. - Generates pre-signed URLs for secure file uploads and downloads. (see 3.1.1)
Classes	preUpload, Upload

6.1.2.1. Module 2, class 1

Name	preUpload
Purpose	Handles the preupload process for a file in the file upload form
Members	• file: Specifies the file. • fileSizeLimit: Specifies the file size limit. • fileSummary: Stores the name, type, link and size data for the file. • data: Specifies the fiile upload key for fileSummary.
Methods	None.

6.1.2.2. Module 2, class 2

Name	Upload
Purpose	Handles the upload process for a file in the file upload form
Members	• file: Specifies the file. • fileSummary: Stores the name, type, link and size data for the file. • bytes: An arrayBuffer of the specified file. • buffer: A Buffer from bytes.

	• upload: A PUT request that stores the response in this variable.
Methods	None.

6.1.3. Module 3

Name	Assignment and Grading Module (see 3.1.1)
Type	Code Module (see 3.1.1)
Purpose	To manage assignments, submissions, and grading functionality for students and teachers. (see 3.1.1)
Way of operating	Teachers create assignments and specify due dates. Students submit tasks which are graded by teachers. (see 3.1.1)
Classes	None

6.1.4. Module 4

Name	Course Module (see 3.1.1)
Type	Code Module (see 3.1.1)
Purpose	Manages subjects, courses, and student attendance within those subjects. (see 3.1.1)
Way of operating	Admin creates subjects Teacher creates sessions (courses) Teacher manages student attendance (see 3.1.1)
Classes	None

6.1.5. Module 5

Name	Account Management Module (see 3.1.1)
Type	Code Module (see 3.1.1)
Purpose	Manages subjects, courses, and student attendance within those subjects. (see 3.1.1)
Way of operating	Admin creates subjects Teacher creates sessions (courses) Teacher manages student attendance (see 3.1.1)
Classes	None

6.1.6. Module 6

Name	Student and Teacher Management Module (see 3.1.1)
Type	Code Module (see 3.1.1)

Purpose	Manages teacher assignments to subjects and student enrolments in groups and subjects. (see 3.1.1)
Way of operating	Teachers are assigned to subjects and courses. Students are enrolled in subjects and groups. (see 3.1.1)
Classes	None

6.2. Data Modules Detailed Design

The data modules described in chapter 3.3. are detailed. Detailed diagrams are recommended.

6.2.1. Data module 1
6.2.2. Data module 2
...

6.2.1. Module 1

Name	Assignment Data
Purpose	Manages assignment-related data, including details, due dates, and types (homework, quiz, project).
Way of operating	Data is created by teachers, accessed by students, and linked to submissions and grades.

6.2.2. Module 2

Name	User Data
Purpose	Stores and manages user information such as profiles, roles, and permissions.
Way of operating	User data is retrieved and updated through APIs and securely stored in the database.

6.2.3. Module 3

Name	Grade Data
Purpose	Stores and manages grades assigned to students for various assignments and activities.
Way of operating	Teachers input grades via the UI, which are stored in the database and retrieved for reports or analysis.

6.2.4. Module 4

Name	Attendance Data
Purpose	Stores and manages grades assigned to students for various assignments and activities.
Way of operating	Teachers input grades via the UI, which are stored in the database and retrieved for reports or analysis.

6.2.5. Module 5

Name	File Metadata
Purpose	Manages metadata for user-uploaded files, including file names, types, and links to S3 storage.
Way of operating	Metadata is stored in the database and linked to user accounts and assignments.

6.2.6. Module 6

Name	Course Data
Purpose	Stores data about subjects, courses, and their associations with teachers and students.
Way of operating	Data is created and modified by teachers or admins. Linked with student enrolments and attendance.

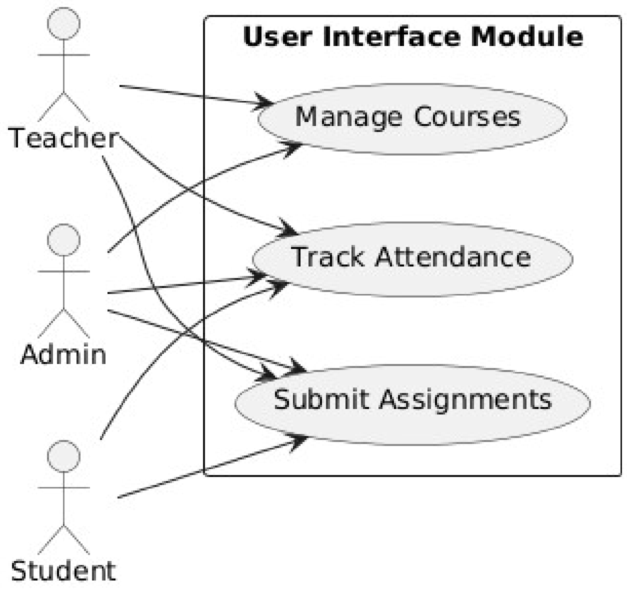
6.2.7. Module 7

Name	Enrolment Data
Purpose	Tracks student enrolments in subjects and groups.
Way of operating	Data is updated during enrolment or course assignment and linked to the subject and user records.

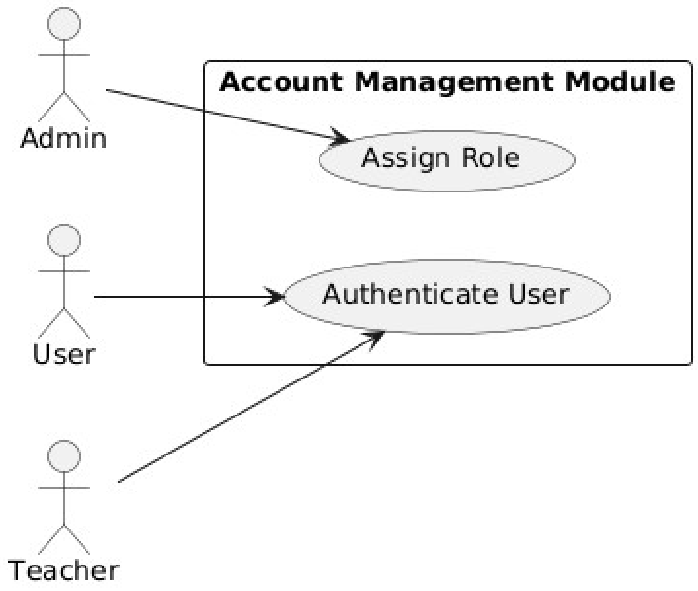
Appendices

A1. Use cases diagrams

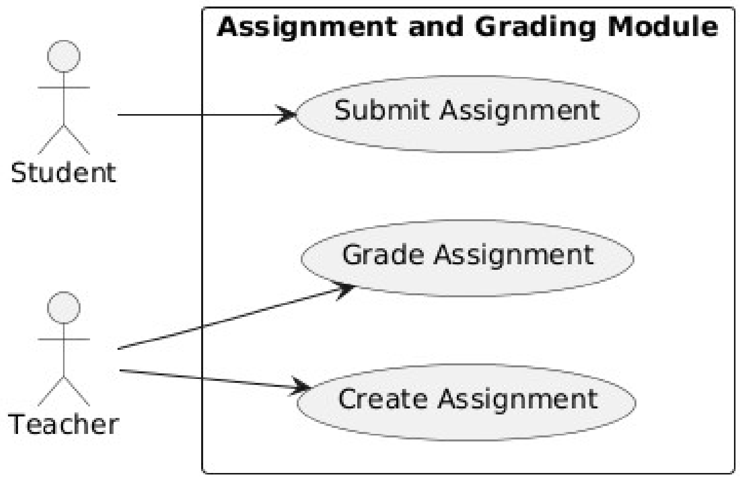
A1.1. User Interface Module



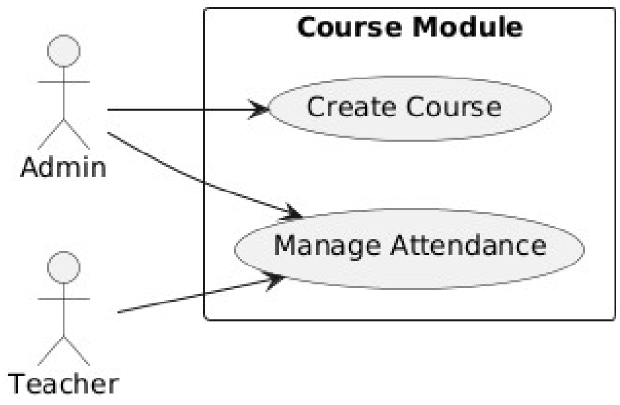
A.1.2. Account Management Module



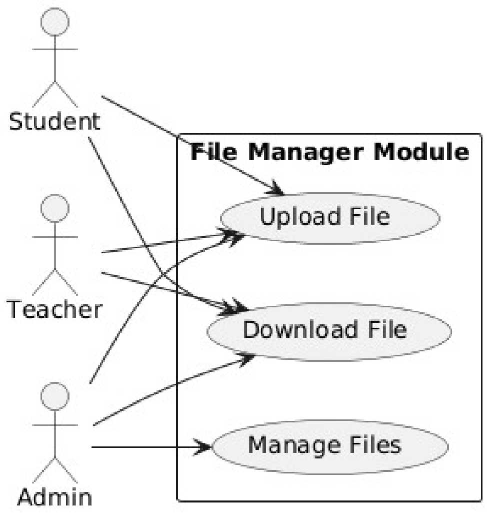
A1.3. Assignment and Grading Module



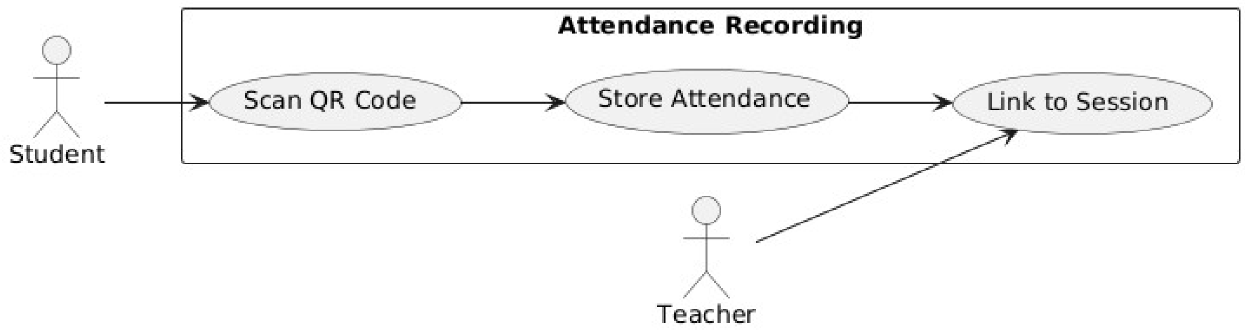
A1.4. Course Module



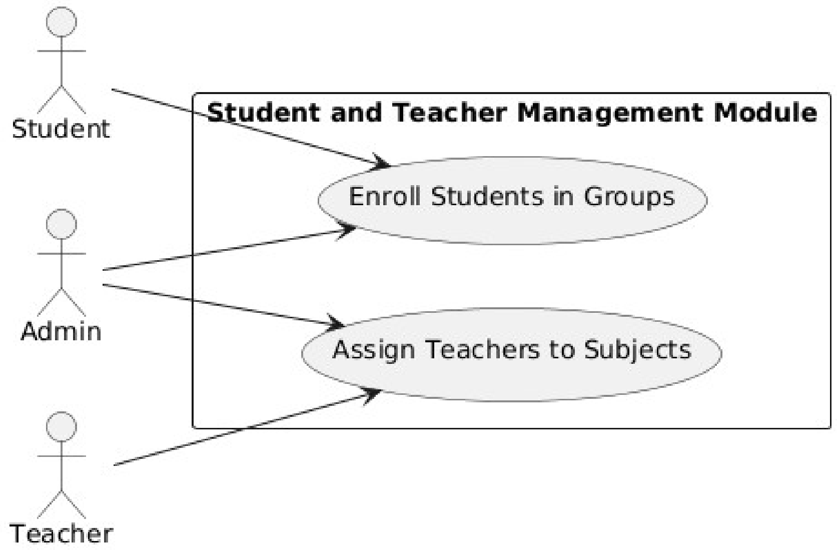
A.1.5. File Manager Module



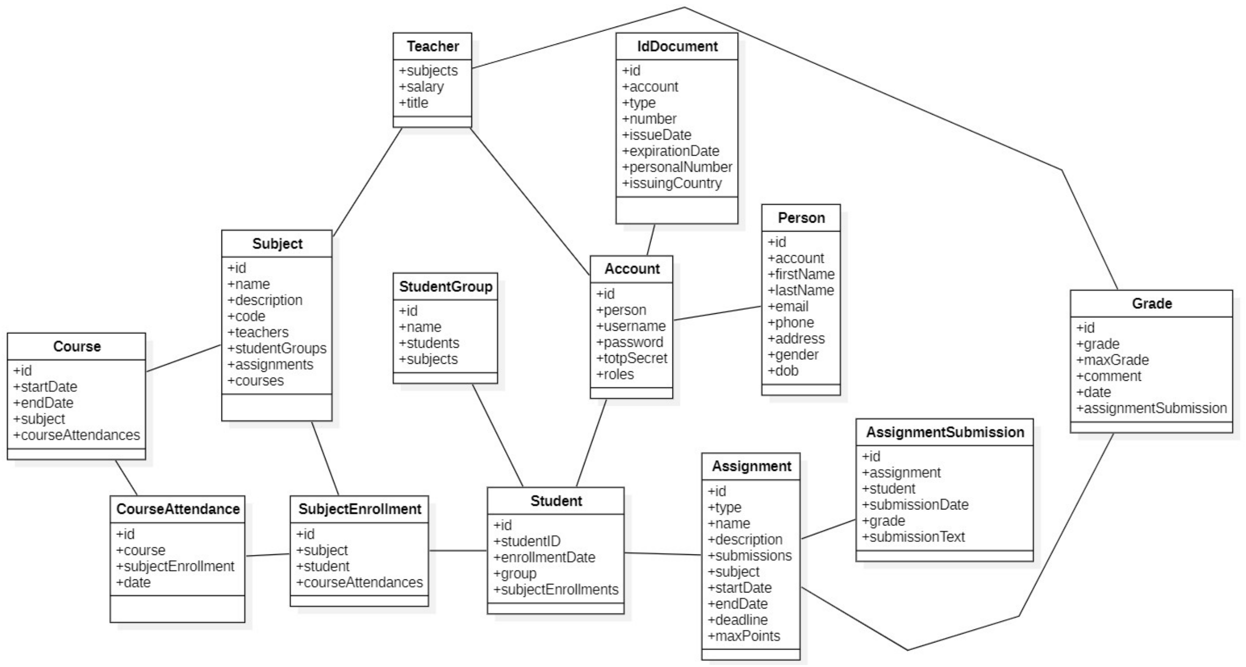
A.1.6. Attendance Recording



A1.7. Student and Teacher Management Module

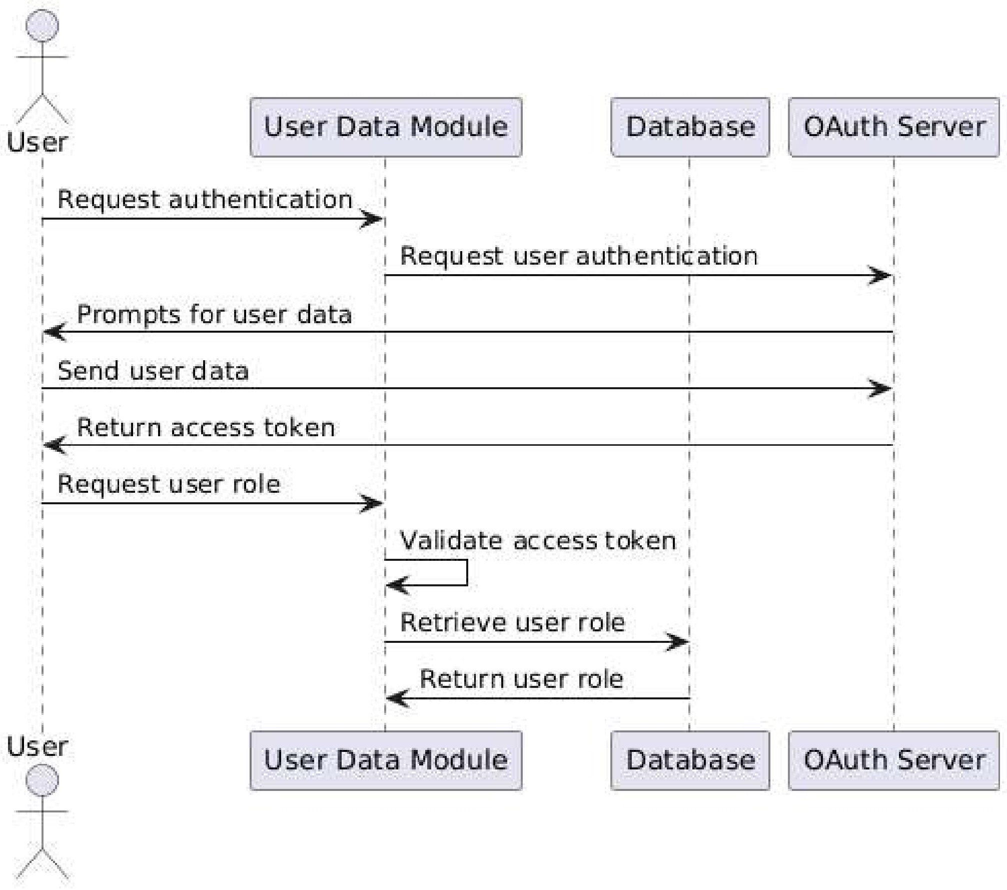

Student
Student and Teacher Management Module
Enroll Students in Groups
Admin
Assign Teachers to Subjects
Teacher

A2. Class diagrams

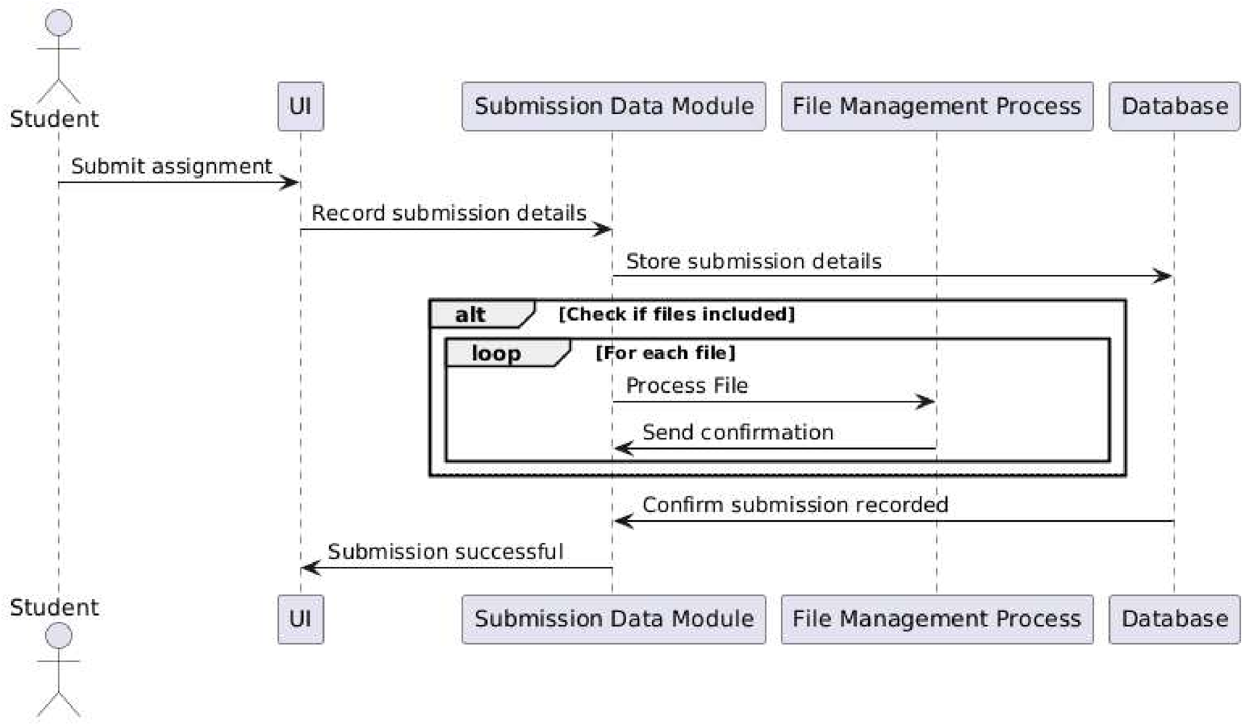


A3. Sequence diagrams

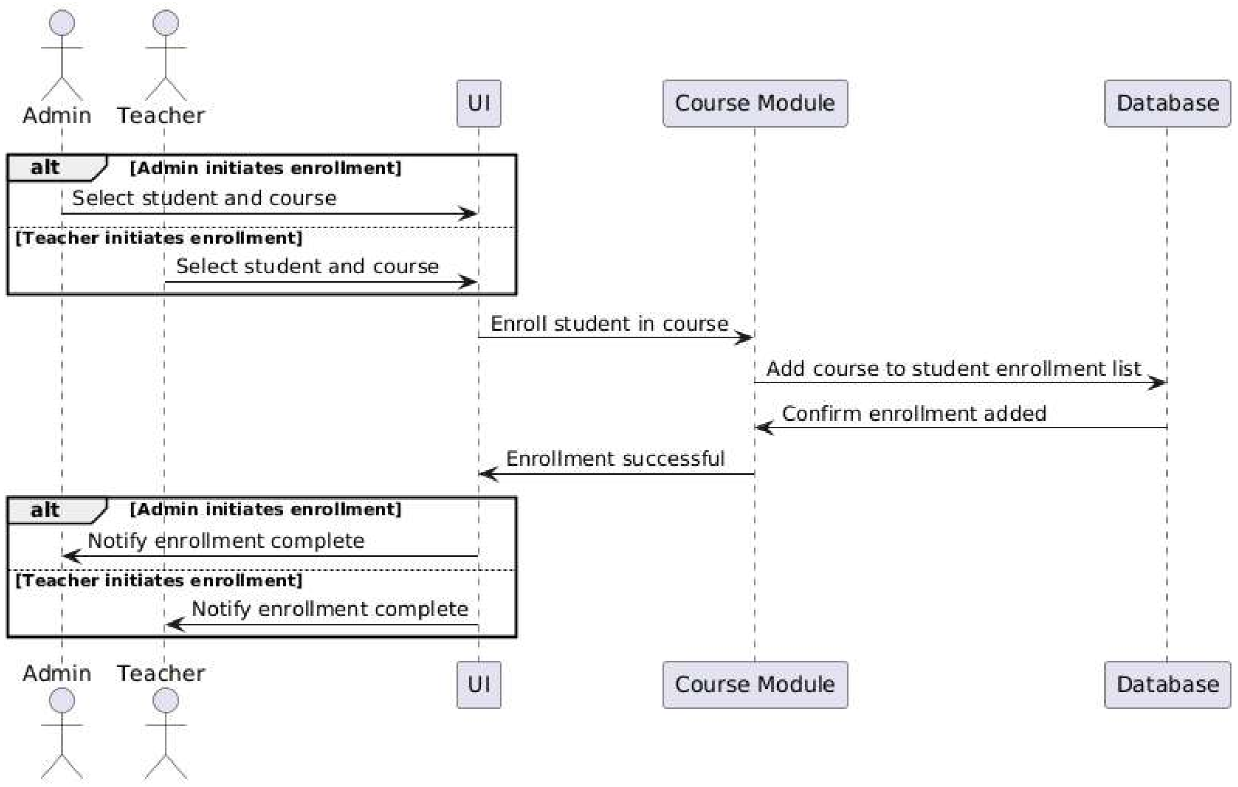
A3.1. User Authentication and Authorization



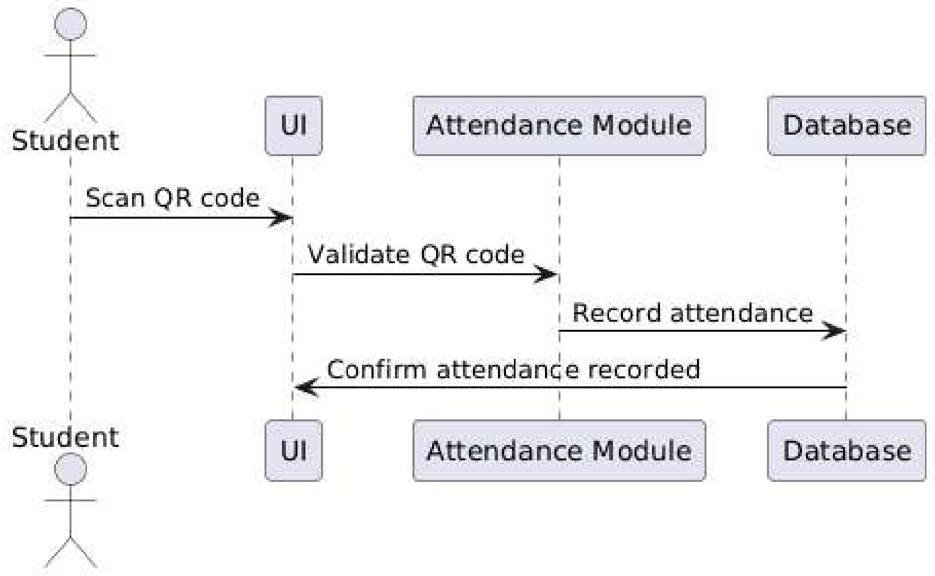
A3.2. Assignment Submission



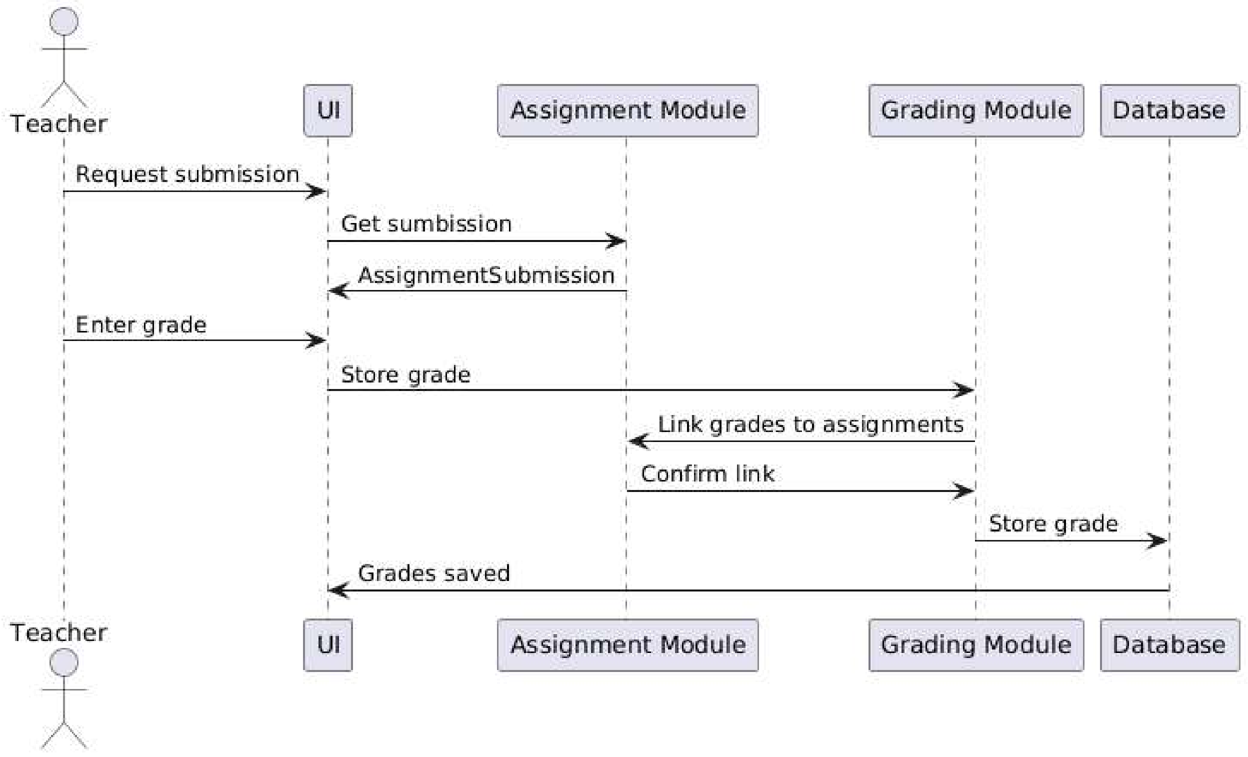
A3.3. Course Enrolment


Admin
Teacher
UI
Course Module
Database
alt [Admin initiates enrollment]
Select student and course
[Teacher initiates enrollment]
Select student and course
Enroll student in course
Add course to student enrollment list
Confirm enrollment added
Enrollment successful
alt [Admin initiates enrollment]
Notify enrollment complete
[Teacher initiates enrollment]
Notify enrollment complete

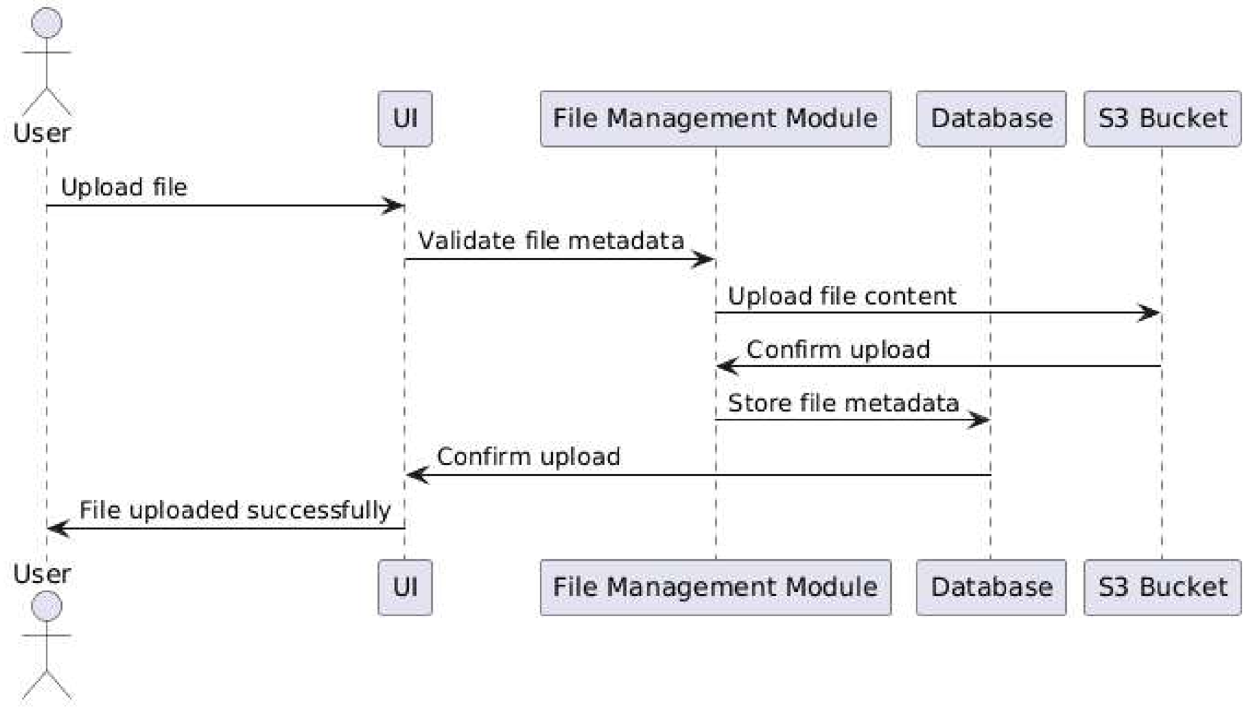
A3.4. Attendance Recording



A3.5. Grading



A3.6. File Upload



A4. Document evolution

This section displays the changes and updates made to the document throughout time, demonstrating its evolution and iterations.

Version	Date	The author/authors of the change	Details of changes
1.00	25.11.2024	Cristea Razvan, Stefanescu Bogdan, Stan-Soponaru Alexandru	Initial version/started working on the project
1.01	1.12.2024	Cristea Razvan, Stefanescu Bogdan, Stan-Soponaru Alexandru	Complete the **introduction** chapter
1.02	2.12.2024	Cristea Razvan, Stefanescu Bogdan, Stan-Soponaru Alexandru	Complete the **references** chapter and start working on the **decomposition description** chapter.
1.03	4.12.2024	Cristea Razvan, Stefanescu Bogdan, Stan-Soponaru Alexandru	Started working on the **dependency description** and **interface description** chapters.
1.04	7.12.2024	Cristea Razvan, Stefanescu Bogdan, Stan-Soponaru Alexandru	Continue working on **dependency** and **interface descriptions**.

			Begin working on the **detailed design** chapter.
1.05	9.12.2024	Cristea Razvan, Stefanescu Bogdan, Stan-Soponaru Alexandru	Complete **dependency** and **interface descriptions** chapters, as well as **detailed design** chapters. Begin working on the diagrams.
1.06	11.12.2024	Cristea Razvan, Stefanescu Bogdan, Stan-Soponaru Alexandru	Make final edits and evaluate the document.

A5. Conclusions regarding the activity

The **Moodle++** system, designed to satisfy modern educational needs, has a user-friendly interface, follows educational and regulatory norms for user administration, course management, and data synchronization, and meets rigorous system requirements for security, scalability, and performance.

Moodle++ was developed through an iterative approach to ensure clarity, consistency, and alignment with stakeholder expectations.