21

**Tech Mavericks**

**Student Project Allocation Management and Online Testing**

Software Requirements Specification

**Version <1.2>**

Submitted in Partial Fulfillment for the Award of Degree of Bachelor of Technology in Information Technology from Rajasthan Technical University, Kota

## MENTOR: SUBMITTED BY:

**Mrs. Sanju Choudhary Vanshika Tahalwani (21ESKIT112)** (Dept. of Information Technology) **Tanmay Joshi (21ESKIT108)**

**COORDINATOR:**

## Mrs. Richa Rawal

(Dept. of Information Technology)

# DEPARTMENT OF INFORMATION TECHNOLOGY

**SWAMI KESHWANAND INSTITUTE OF TECHNOLOGY, MANAGEMENT & GRAMOTHAN**

**Ramnagaria (Jagatpura), Jaipur – 302017 SESSION 2024-25**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 20/11/2024 | 1.1 | Contain Basic Layout and requirement Specification for this Project | Vanshika Tahalwani,  Tanmay Joshi |
| 01/12/2024 | 1.2 | Added Module for Database, Authentication | Vanshika Tahalwani,  Tanmay Joshi |

**Table of Contents**

1. [Introduction 4](#_TOC_250023)
   1. [Purpose 4](#_TOC_250022)
   2. [Scope 4](#_TOC_250021)
   3. [Definitions, Acronyms and Abbreviations 4](#_TOC_250020)
   4. [References 4](#_TOC_250019)
   5. [Technologies to be used 4](#_TOC_250018)
   6. Overview 5
2. [Literature survey 6](#_TOC_250016)
   1. Objectives 6
   2. [Research Paper 7](#_TOC_250015)
3. [Specific Requirements 7](#_TOC_250014)
   1. Functional Requirements 7
   2. [Non- Functional Requirements 9](#_TOC_250013)
   3. [Hardware Requirements 9](#_TOC_250012)
   4. [Software Requirements 9](#_TOC_250011)
   5. [Agile Methodology 10](#_TOC_250010)
   6. [Business Process Model 11](#_TOC_250009)
   7. [Supplementary Requirements 12](#_TOC_250008)
4. [Overall Description 14](#_TOC_250007)
   1. [Use-Case Model Survey 15](#_TOC_250006)
   2. [Behaviors Diagrams 16](#_TOC_250005)
   3. Structural Diagrams 24
   4. Database Diagram 28
   5. Assumptions and Dependencies 28
5. [System Architecture](#_TOC_250004) 30
   1. [Client-Server Architecture 30](#_TOC_250003)
   2. Communications Interfaces 30
6. [Conclusion & Future scope 33](#_TOC_250001)
7. [Concerns / Queries / Doubts if any: 33](#_TOC_250000)

# 1.Introduction

# The Software Requirements Specification (SRS) for the *Student Project Allocation Management and Online Testing* outlines the functional and non-functional requirements for developing and deploying a platform that enhances communication and project management for students, supervisors, and coordinators. It covers the purpose, scope, key roles such as administrators, mentors, and mentees, along with relevant terminology, references, and technologies used. The SRS serves as a guide for developers and stakeholders, ensuring the system meets academic needs and facilitates efficient project management.

## Purpose

## The primary objective of this SRS is to establish a clear and unified understanding among all stakeholders, including developers, designers, project coordinators, supervisors, and students, regarding the functionalities, constraints, and objectives of the *Student Project Management and Online Testing*. By outlining and documenting these requirements, the SRS aims to guide the development team in creating a user-friendly, efficient platform that streamlines project management, improves communication, ensures timely submissions, and enhances the overall quality of student research projects.

## Scope

## The *Student Project Allocation Management and Online Testing* is envisioned as a comprehensive platform that offers a wide range of functionalities to support various stages of student project management. These include project topic selection, approval workflows, project progress tracking, communication between students and supervisors, report submission, and storage. This SRS details the specific requirements for each module within the system, outlining their functionalities, interactions, and constraints to ensure seamless, transparent, and efficient project management for all users involved.

## Definitions, Acronyms and Abbreviations

* + - **SRS**: Software Requirements Specification
    - **API**: Application Programming Interface
    - **UI**: User Interface
    - **DBMS**: Database Management System
    - **IoT**: Internet of Things

## References

## This SRS references academic standards, project management best practices, and any external documents or guidelines used in defining the requirements for the *Student Project Allocation Management and Online Testing*. These references ensure that the platform aligns with established methodologies and addresses the specific needs of academic institutions in managing final year projects efficiently.

## Technologies to be used

The "Student Project Management" platform utilizes cutting-edge technologies and frameworks to deliver its functionalities, including:

* + - **Frontend**: React for cross-platform mobile application development.
    - **Authentication**: Firebase Authentication for Mobile SMS Verification
    - **Backend**: Node.js for the backend server application.
    - **Database**: MongoDB for data storage and retrieval.

## 

## 1.6 Overview

This SRS document is structured systematically into distinct sections, each detailing specific facets of the "Student Project Management" platform. It includes an executive summary, system overview, detailed descriptions of functional and non-functional requirements, user interfaces, system constraints, assumptions, dependencies, and technical specifications. Additionally, it incorporates diagrams, use cases, and mockups to provide a comprehensive understanding of the system's architecture and functionality.

# Literature Survey

The literature survey conducted for the  *Student Project Allocation Management and Online Testing* SRS involve a thorough review of existing research, best practices in academic project management, and technological advancements related to web-based communication platforms and automated project tracking systems. This review helped identify key challenges in managing student projects and informed the design of the system to improve communication, efficiency, and transparency in the project lifecycle.

## Objective

 **Academic Project Management Systems**:  
The survey involved an in-depth analysis of existing academic project management systems, both traditional and web-based, evaluating their features, user interfaces, and the overall impact on project coordination and submission processes. Notable systems reviewed include platforms used by universities for managing final year projects, which highlight key functionalities such as topic selection, progress tracking, and report submission.

 **Web-based Communication Platforms**:  
An analysis of web-based communication platforms was conducted to understand their features, user experiences, and efficiency in facilitating seamless interactions between students, supervisors, and coordinators. Key aspects examined include message delivery, document sharing, notifications, and real-time collaboration, which are essential for managing academic projects.

 **Technological Trends**:  
A review of current technological trends in web development, automation, and cloud-based solutions was performed to identify emerging technologies that could enhance the Web-based Student Project Management System. The integration of these trends, such as automated workflows, cloud storage, and real-time notifications, can significantly improve project management efficiency and transparency.

 **Research Papers and Publications**:  
Relevant research papers, publications, and articles related to academic project management, student-teacher collaboration, and online educational platforms were surveyed. These resources provided insights into the methodologies, best practices, and challenges shaping the development of effective project management systems in academic settings.

 **Standards and Guidelines**:  
An exploration of academic guidelines, data management policies, and privacy standards related to online education platforms was conducted. This review ensures that the system complies with institutional regulations and adheres to best practices for protecting user data and ensuring ethical project management.

 **Findings and Insights**:  
The literature survey revealed valuable insights into the need for improved communication and transparency in academic project management. It identified opportunities for leveraging web technologies to automate routine tasks, streamline workflows, and facilitate better collaboration between students and supervisors, ultimately enhancing the quality and timeliness of student project submissions.

## Research Paper

|  |  |
| --- | --- |
| **Final Year Students' Projects Allocation and Management System**  *Written by: Lamido Yahaya, Abubakar Adamu*  One notable paper proposed a cloud-based system for project allocation and supervision at tertiary institutions. This system focused on facilitating communication between students and supervisors while improving project topic verification, allocation, and the digital submission of final reports. It emphasized resolving issues like duplication of project topics and offered a centralized digital space for project coordination, which mirrors the objectives of your system.  *(https://www.researchgate.net/publication/371314931\_Final\_Year\_Students%27Projects\_Allocation\_and\_Management\_System)* | ***Prototype Development of Final Year Project Management System to Monitor Student's Performance***  *Written by: Raznida Isha*  Another paper discussed an online final-year project management system that integrates dashboards for monitoring project progress, allowing supervisors and coordinators to streamline interactions and oversee project timelines efficiently. This dashboard-based approach simplifies managing academic projects, ensuring transparency and real-time updates throughout the project lifecycle, which aligns well with your goals​  (https://www.researchgate.net/publication/378372609\_Prototype\_Development\_of\_Final\_Year\_Project\_Management\_System\_to\_Monitor\_Student's\_Performance) |

**Conclusion**

The literature survey offered valuable insights and a comprehensive understanding of existing academic project management systems, communication platforms, and technological trends in web-based applications. These findings serve as a foundational reference, guiding the development and feature enhancement of the *Web-based Student Project Management System*. By aligning with best practices and leveraging technological advancements, the system is designed to address communication gaps, streamline project workflows, and ensure efficiency in academic project coordination and submission processes.

# Specific Requirements

## Functional Requirement

## The functional requirements outline the specific features and functionalities that the *Web-based Student Project Management System* must provide to address the needs of students, supervisors, and coordinators:

## Login Module:

* + **Description**: Enables role-based login for students, supervisors, and coordinators through email authentication. This module provides different levels of access based on the user role.
  + **Key Features**: Secure login, role-based access control, forgotten password recovery.

1. **Project Topic Selection Modules:**
   * **Description**: Allows students to browse and select project topics from a predefined list or propose new topics for approval by supervisors.
   * **Key Features**: Project topic browsing, proposal submission, topic approval workflow.

## Student-Supervisor Communication Module::

* + **Description**: Facilitates continuous communication between students and their supervisors via an integrated messaging system.
  + **Key Features**: Direct messaging, project-related discussion threads, notification of new messages.

## Project Progress Tracking Module:

* + **Description**: Enables supervisors to monitor the progress of student projects through milestones and deliverables.
  + **Key Features**: Progress timeline, milestone completion tracking, supervisor feedback integration.

## Submission and Approval Module:

* + **Description**: Automates the submission process for student reports, allowing supervisors to approve or request revisions.
  + **Key Features**: Report submission, document version control, approval or rejection notifications.

## Supervisor Assignment Module:

* + **Description**: Assigns supervisors to students based on their project topics and availability.
  + **Key Features**: Supervisor availability management, automated assignment, manual reassignment by coordinator.

## Dashboard Module:

* + **Description**: Provides a personalized dashboard for students, supervisors, and coordinators with relevant information such as project status, deadlines, and notifications.
  + **Key Features**: Centralized project overview, deadline reminders, role-specific dashboard content.

## Project Repository Module:

* + **Description**: Acts as a centralized repository for storing submitted project reports, making them available for future reference.
  + **Key Features**: Document storage, search functionality, access control.

## Duplication Detection Module:

* + **Description**: Automatically checks for duplication in submitted project reports to prevent the recycling of previous projects.
  + **Key Features**: Similarity checking, automated rejection of duplicate topics, and reports.

## Notifications Module:

* + **Description**: Sends notifications to users regarding project deadlines, approvals, and

messages.

* + **Key Features**: Alerts for deadline, message notification.

## 

## 3.2 Non-Functional Requirements

Non-functional requirements define the qualities that the system must possess, ensuring reliability, performance, and usability.

1. **Performance**:
   * The system must handle up to 500 concurrent users without performance degradation.
   * Response time for project progress updates and report submissions shall not exceed 2 seconds.
2. **Security**:
   * User data, including passwords, shall be encrypted and securely stored.
   * Role-based access control will be implemented to protect sensitive project information.
3. **Usability**:
   * The user interface will be intuitive and follow best practices for easy navigation, ensuring users can complete tasks without needing extensive training.
   * The system shall be accessible in multiple languages to accommodate diverse users.
4. **Reliability**:
   * The system must ensure 99.9% uptime to support uninterrupted project management.
   * Regular backups and disaster recovery plans must be in place to prevent data loss.
5. **Scalability**:
   * The system should be able to scale to accommodate an increasing number of users and projects as needed without affecting performance.

## Hardware Requirements

## The hardware requirements outline the necessary infrastructure for deploying and running the *Student Allocation Project Management and Online Testing*:

## Server Infrastructure:

## Description: The system backend must be hosted on a reliable server that ensures smooth

## performance and uptime.

## Specifications:

## Minimum of dual-core processors with 8 GB RAM for the backend server.

## Adequate storage for project files, reports, and system databases (minimum of 500 GB).

## Client Devices:

## Description: The web-based system must be accessible on commonly used devices.

## Specifications:

## Compatible with modern web browsers (Google Chrome, Mozilla Firefox, Microsoft Edge).

## Optimized for desktop and mobile devices with screen resolutions of 1280x720 and above.

## Software Requirements

Software requirements outline the necessary software components and dependencies for developing and maintaining the *Web-based Student Project Management System*:

1. **Backend Technologies**:
   * **Description**: Technologies used for building and managing the server-side functionalities.
   * **Specifications**:
     + **Node.js Boot Framework** for backend services, ensuring scalability and

integration of various modules.

* + - **MongoDB Database** for storing project data, student records, and reports.

1. **Frontend Technologies**:
   * **Description**: Technologies used for building the user interface and user experience.
   * **Specifications**:
     + **React.js** for creating dynamic, responsive user interfaces that cater to students,

supervisors, and coordinators.

* + - **Bootstrap** for a consistent, mobile-friendly design.

1. **Third-Party Integrations**:
   * **Description**: External services and APIs integrated into the platform to enhance functionalities.
   * **Specifications**:
     + Integration with email and notification services for real-time communication.
     + **Turnitin API** or similar service for detecting duplication in project submissions.

## Agile Methodology

The *Web-based Student Project Management System* adopts the Agile development methodology, a proven approach for managing dynamic project requirements and ensuring continuous improvements. Agile emphasizes collaboration, flexibility, and iterative development cycles to adapt to evolving student and supervisor needs throughout the project lifecycle.

Agile methodology is implemented in the following steps:

1. **Project Initiation**:
   * **Define** project goals, objectives, and scope.
   * **Assemble** the project team, including developers, designers, project coordinators, and supervisors.
2. **Product Backlog Creation**:
   * **Identify** and list the desired features and functionalities.
   * **Prioritize** these features based on importance and impact on the student project process.
3. **Sprint Planning**:
   * **Break down** prioritized features into manageable tasks for implementation in short sprints.
   * **Allocate resources** and estimate the effort required for each task.
4. **Sprint Execution**:
   * **Developers work** on tasks assigned for the sprint.
   * **Daily stand-up meetings** are held to track progress and adjust plans if needed.
5. **Continuous Integration and Testing**:
   * **Developers continuously integrate** their code into a shared repository.
   * **Testing** (automated and manual) ensures features meet quality standards.
6. **Sprint Review**:
   * **Showcase the completed work** to stakeholders at the end of each sprint.
   * **Collect feedback** for adjustments in the next sprint.
7. **Sprint Retrospective**:
   * **Reflect on successes** and challenges.
   * **Identify areas for improvement** and implement changes to improve team performance and product quality.
8. **Incremental Deployment**:
   * **Deploy tested features incrementally** into the live environment.
   * **Students, supervisors, and coordinators** can use and provide feedback on new functionalities.
9. **Continuous Feedback and Adaptation**:
   * **Regular feedback** from users is gathered and used to refine features and adjust project scope.
10. **Iterative Development**:

* **Repeat the cycle**, incorporating new features and improvements in each sprint.

## Business Process Model

1.**Login Module**:

* **Input**: Student’s or supervisor’s credentials (username and password).
* **Process**: Authenticate credentials.
* **Output**: Successful login or error message.

2. **Project Submission Module**:

* **Input**: Students upload project files (documentation, code, reports).
* **Process**: Store project files in the database.
* **Output**: Confirmation of successful project submission.

3.**Supervisor Feedback Module**:

* **Input**: Supervisor’s evaluation and feedback on student submissions.
* **Process**: Record feedback and evaluation scores.
* **Output**: Display feedback to students.

4.**Progress Tracking Module**:

* **Input**: Project status updates from students.
* **Process**: Track and display project progress.
* **Output**: Visual progress bar showing project milestones.

5. **Plagiarism Detection Module**:

* **Input**: Submitted project documents.
* **Process**: Analyze documents using a plagiarism detection tool.
* **Output**: Plagiarism report.

6. **Notification Module**:

* **Input**: System-generated notifications.
* **Process**: Filter notifications based on deadlines, updates, and supervisor feedback.
* **Output**: Display relevant notifications to users.

7. **Dashboard Module**:

* **Input**: User interaction with the dashboard.
* **Process**: Provide access to project details, submissions, and feedback.
* **Output**: Personalized dashboard with relevant project information.

This model outlines key processes in the *Student Project Management System*, detailing inputs, processes, and outputs to streamline project submission and evaluation.

## Supplementary Requirements

1.**Scalability**:

* The system architecture should be designed to **scale horizontally** to accommodate an increasing number

of students and supervisors.

2. **Documentation**:

* Comprehensive user manuals and developer documentation should be provided for **ease of use** and system maintenance.

3.**Security Measures**:

* **Data Encryption**: Ensure sensitive student and project data are encrypted to maintain confidentiality.
* **Role-based Access Control**: Different roles (e.g., student, supervisor, admin) should have specific access privileges.

1. **Compatibility and Integration**:

* **API Integration**: Provide well-documented APIs to allow for **seamless integration** with external systems

(e.g., plagiarism detection tools, third-party communication tools).

5.**Performance Optimization**:

* **Load Balancing**: Implement load balancing techniques to ensure the system performs optimally, especially during periods of high traffic like project submission deadline

# Overall Description

## Use-Case Model

**Actors: Mentees , Mentor , Project coordinator**

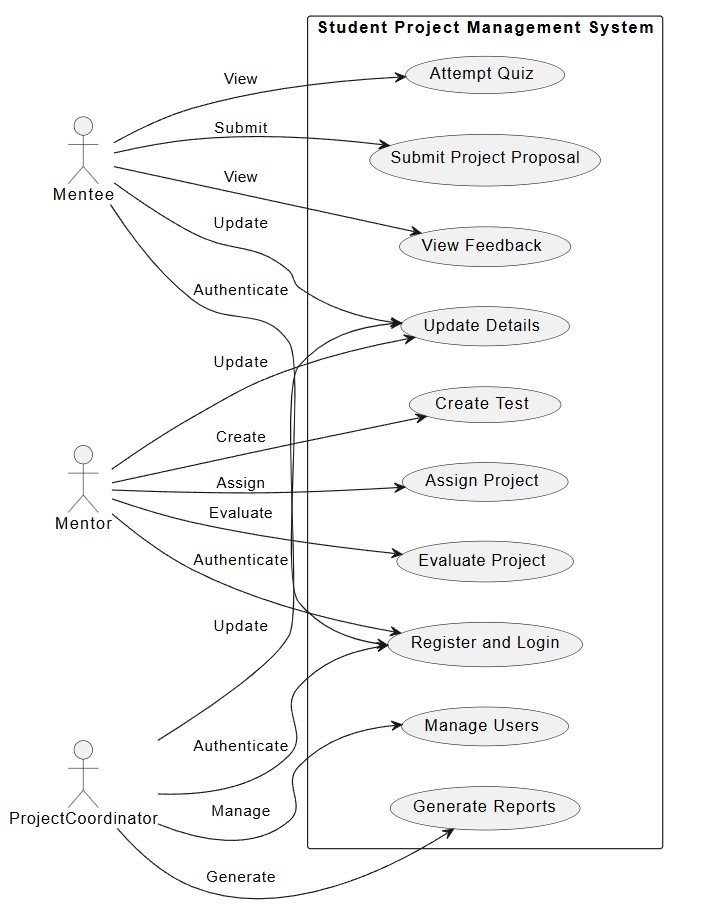
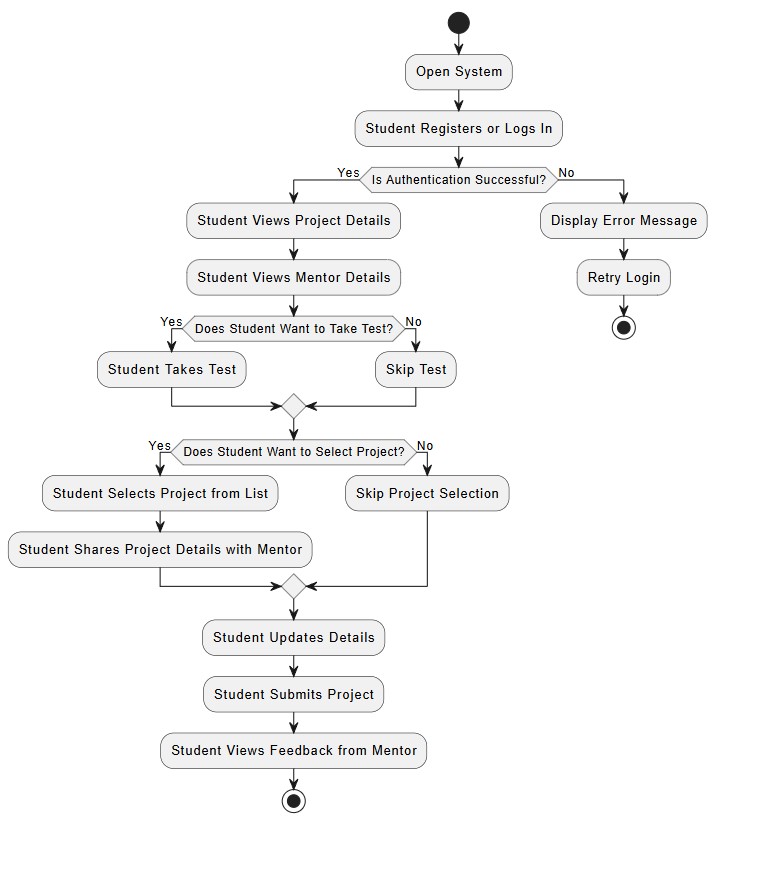
****

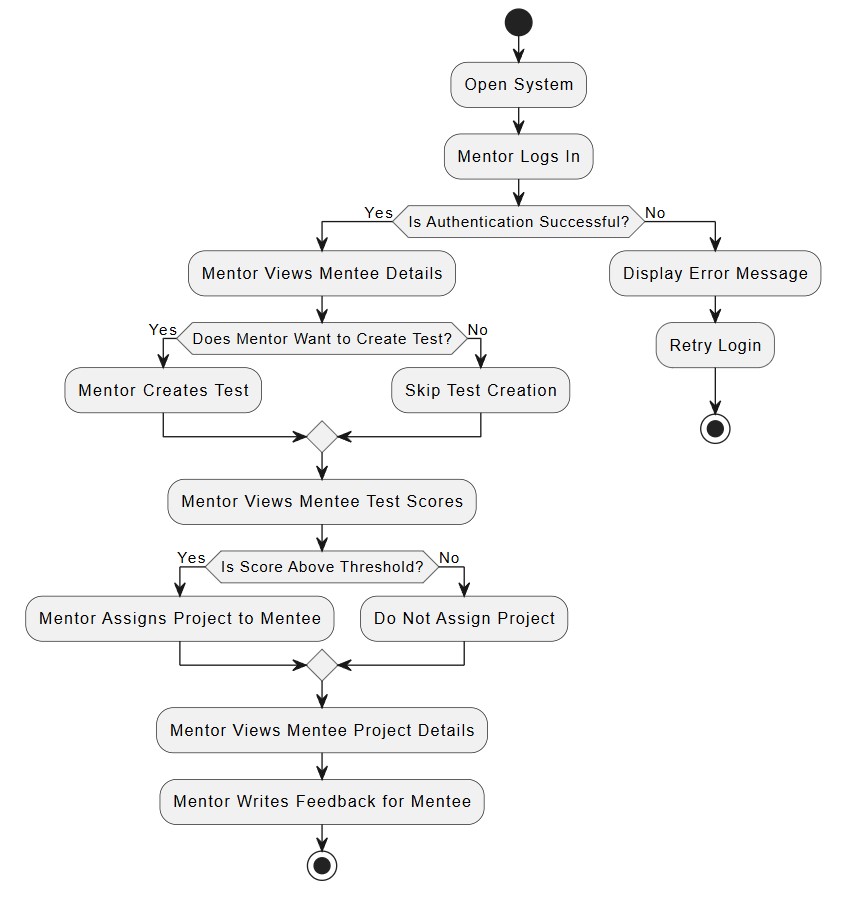
Figure 1: Use Case diagram of SPM

## Behaviors Diagrams

Activity Diagram for student:



* + - Activity Diagram for mentor



## 

* + - Activity Diagram

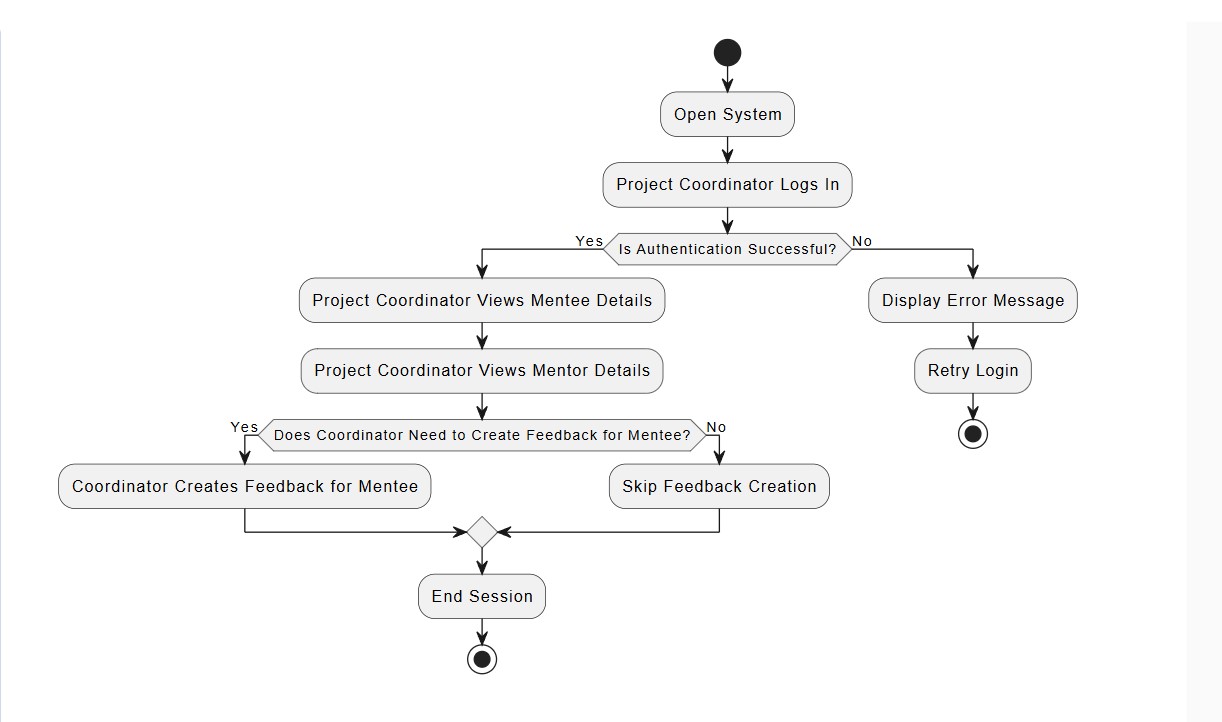
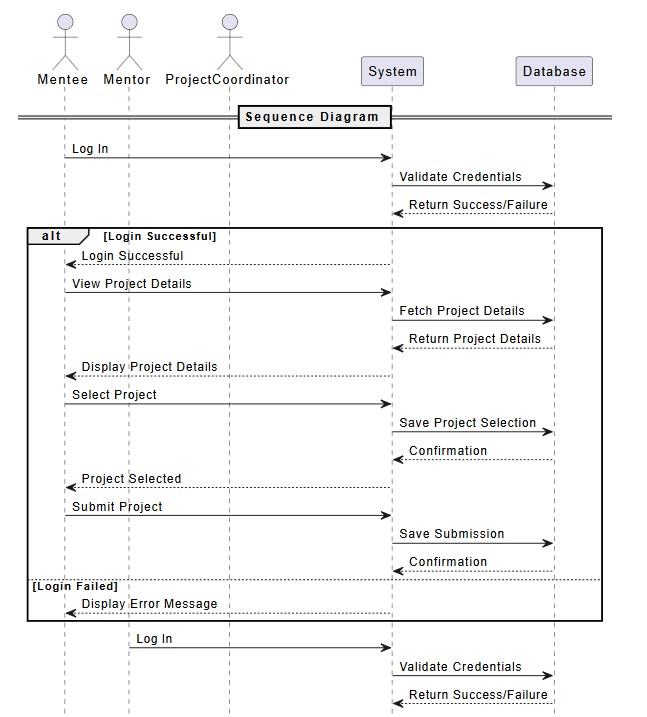


Figure 2: Activity Diagram of Web Baesed SPM

* + - Sequence Diagram



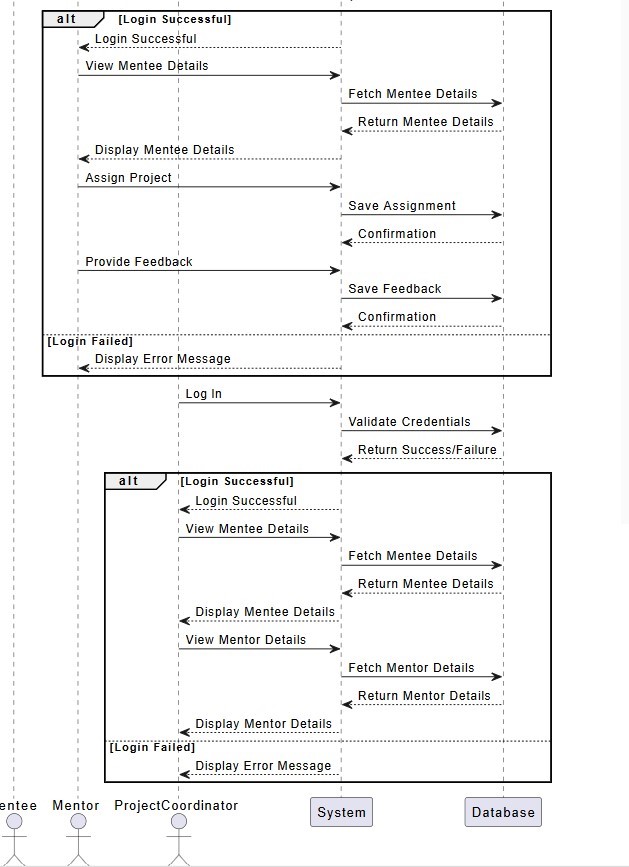
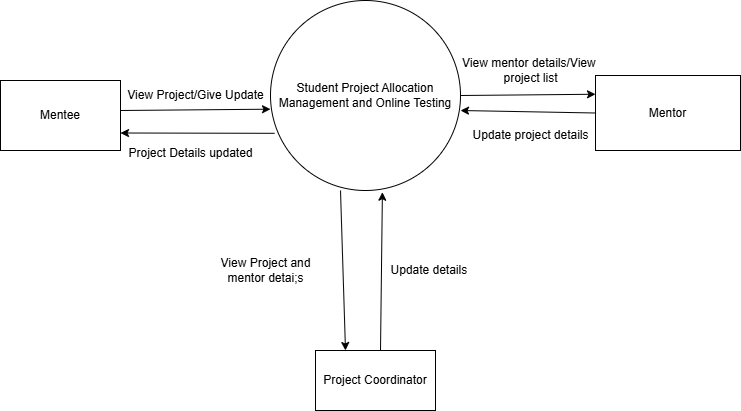


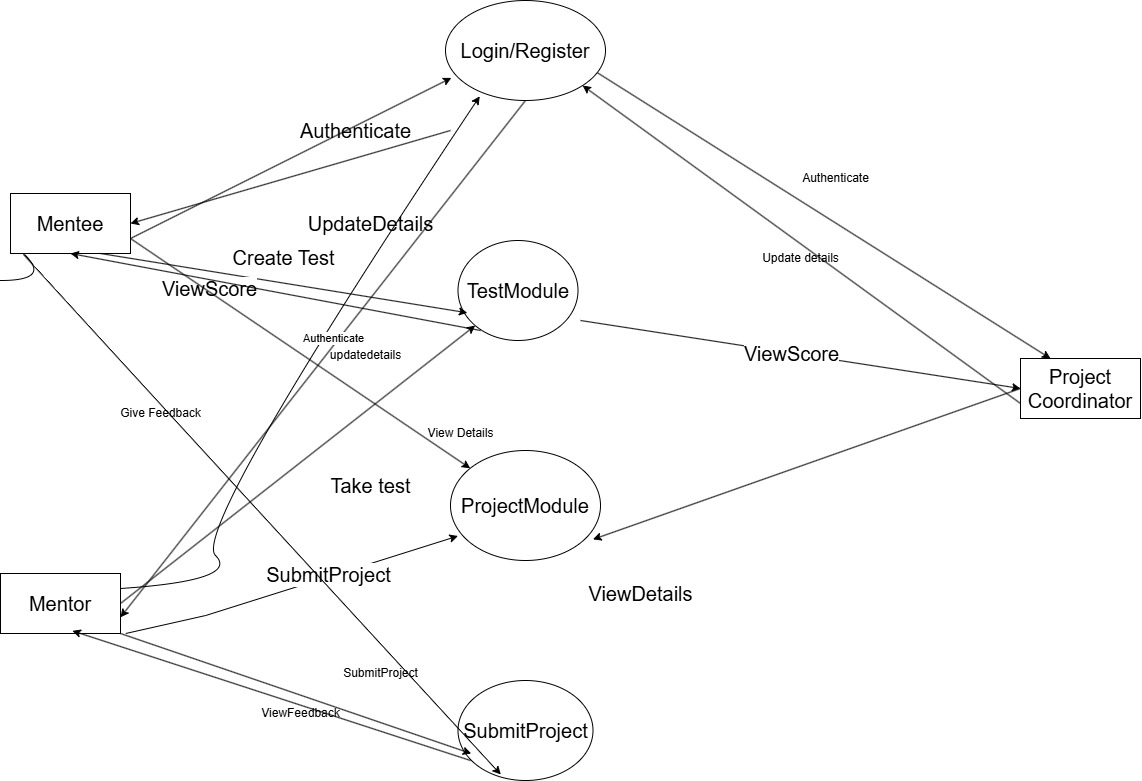
Figure 3:Sequence Diagram for all modules

* + - Data Flow Diagram – Level 0



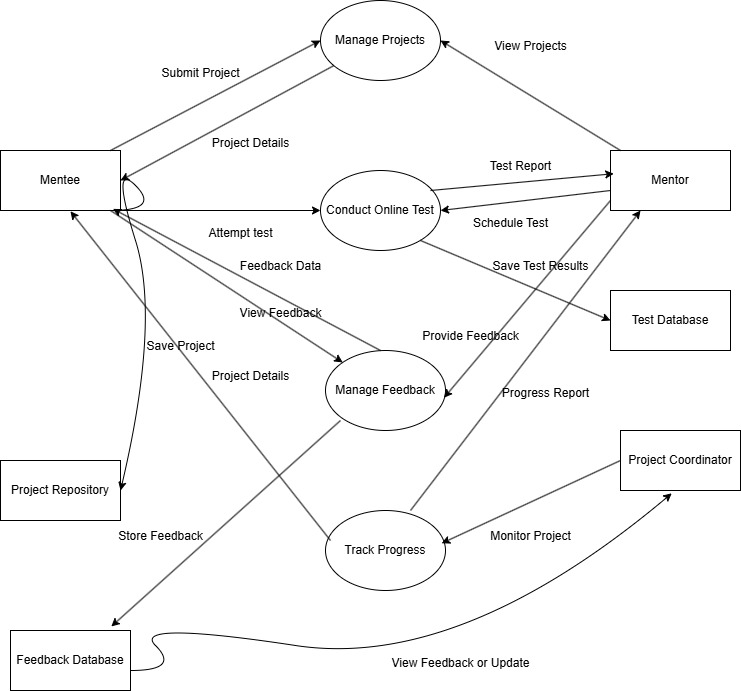
*Figure 4: DFD Level 0*

Data Flow Diagram – Level 1



*Figure 5:DFD Level 1*

* + - Data Flow Diagram – Level 2



*Figure 6: DFD Level 2*

## Structural Diagrams

* + - Deployment Diagram

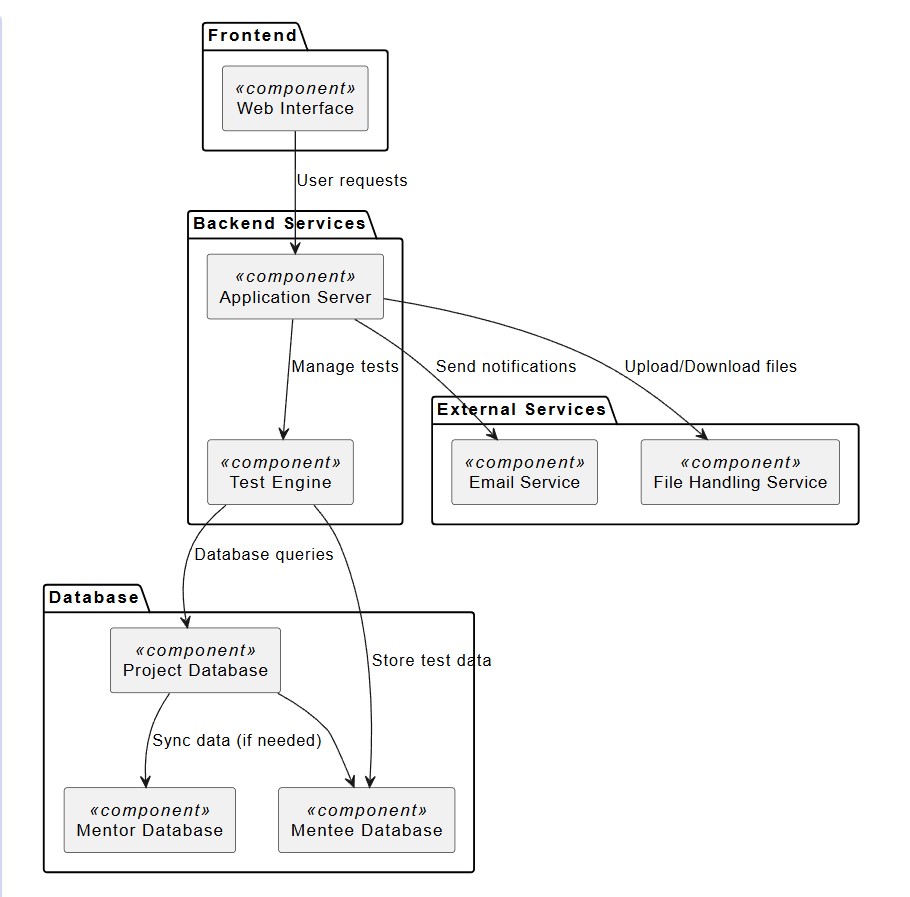


Figure 7: Deployment Diagram

Component Diagram

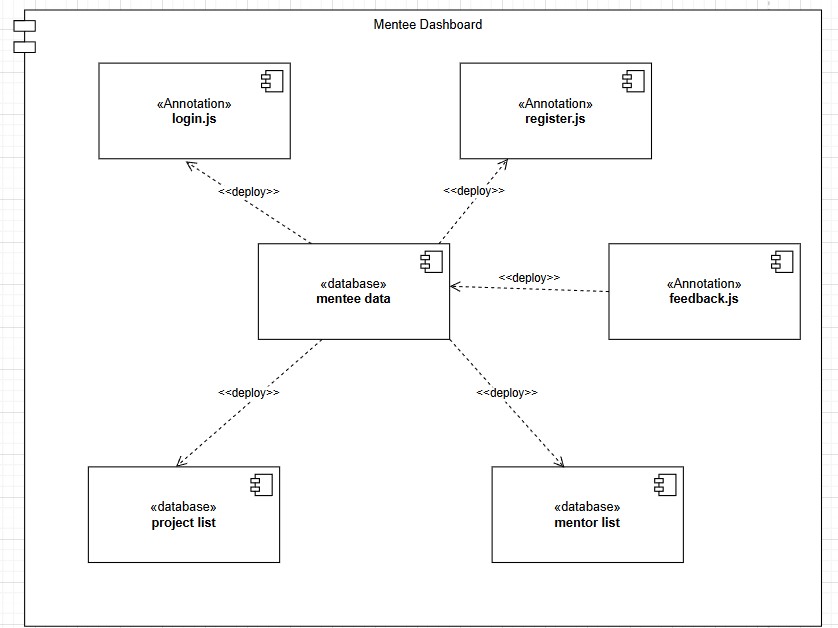


Figure 8: Component Diagram for mentee

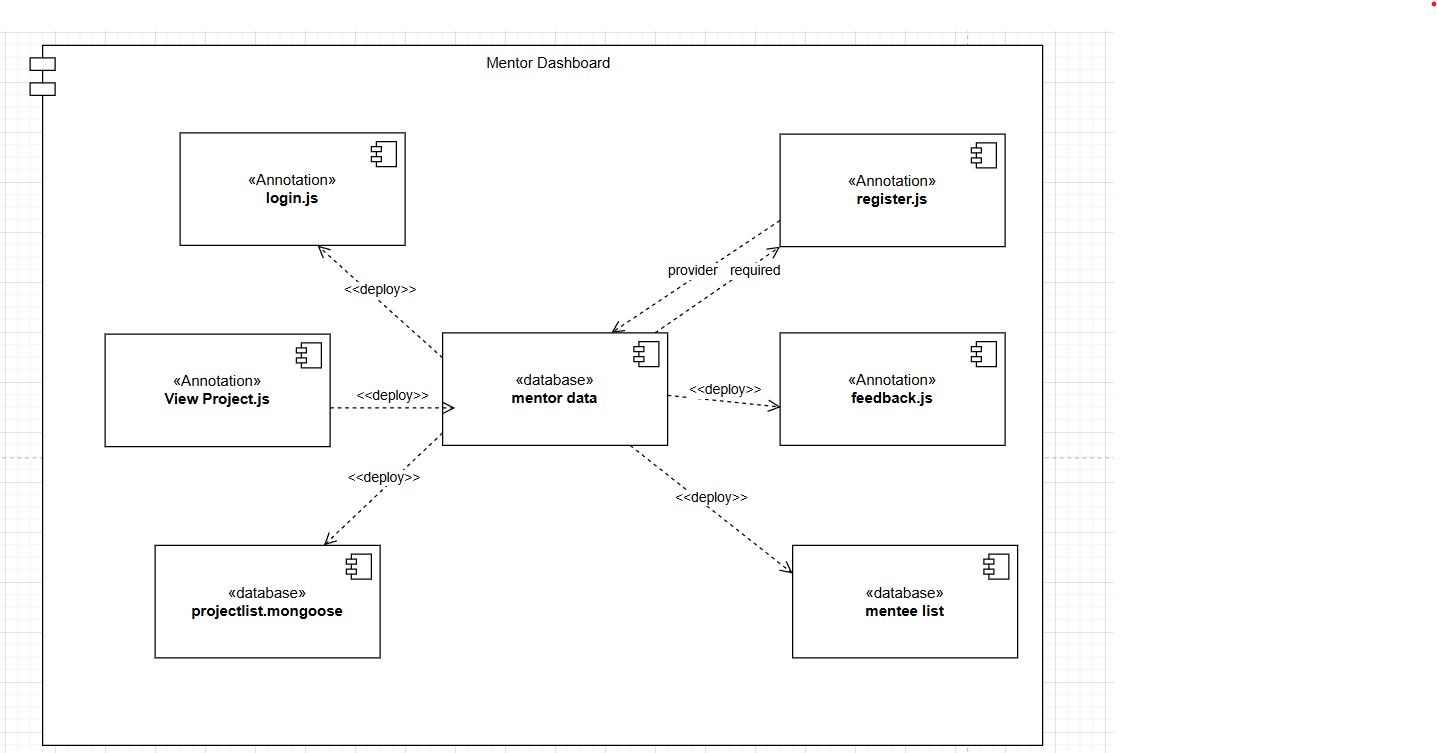


Figure 9: Component Diagram for mentor

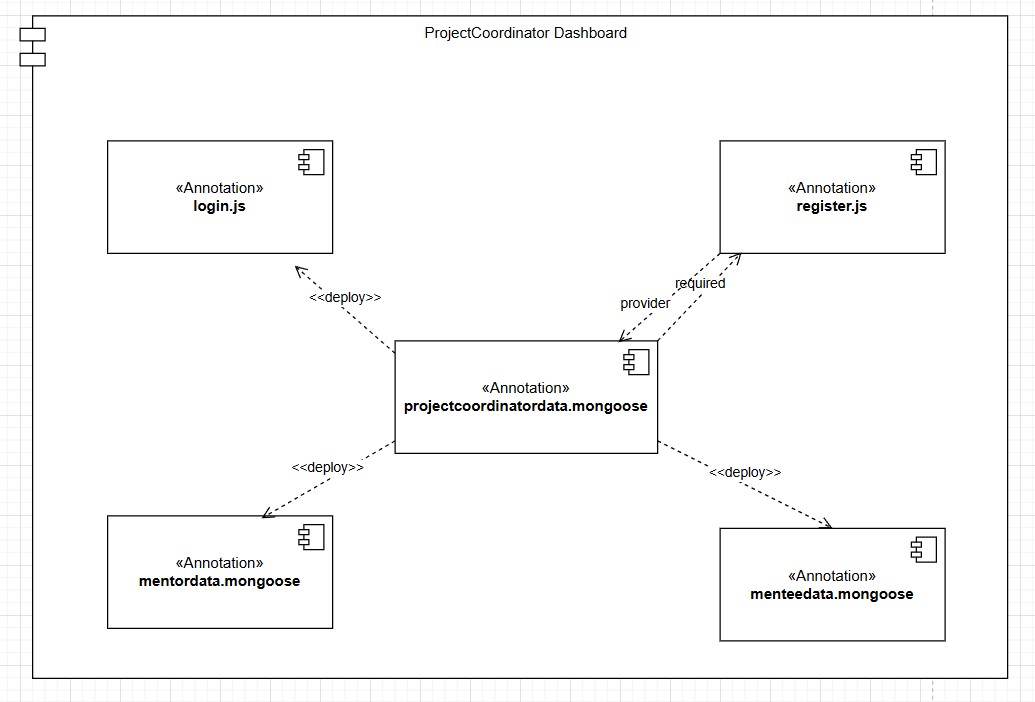
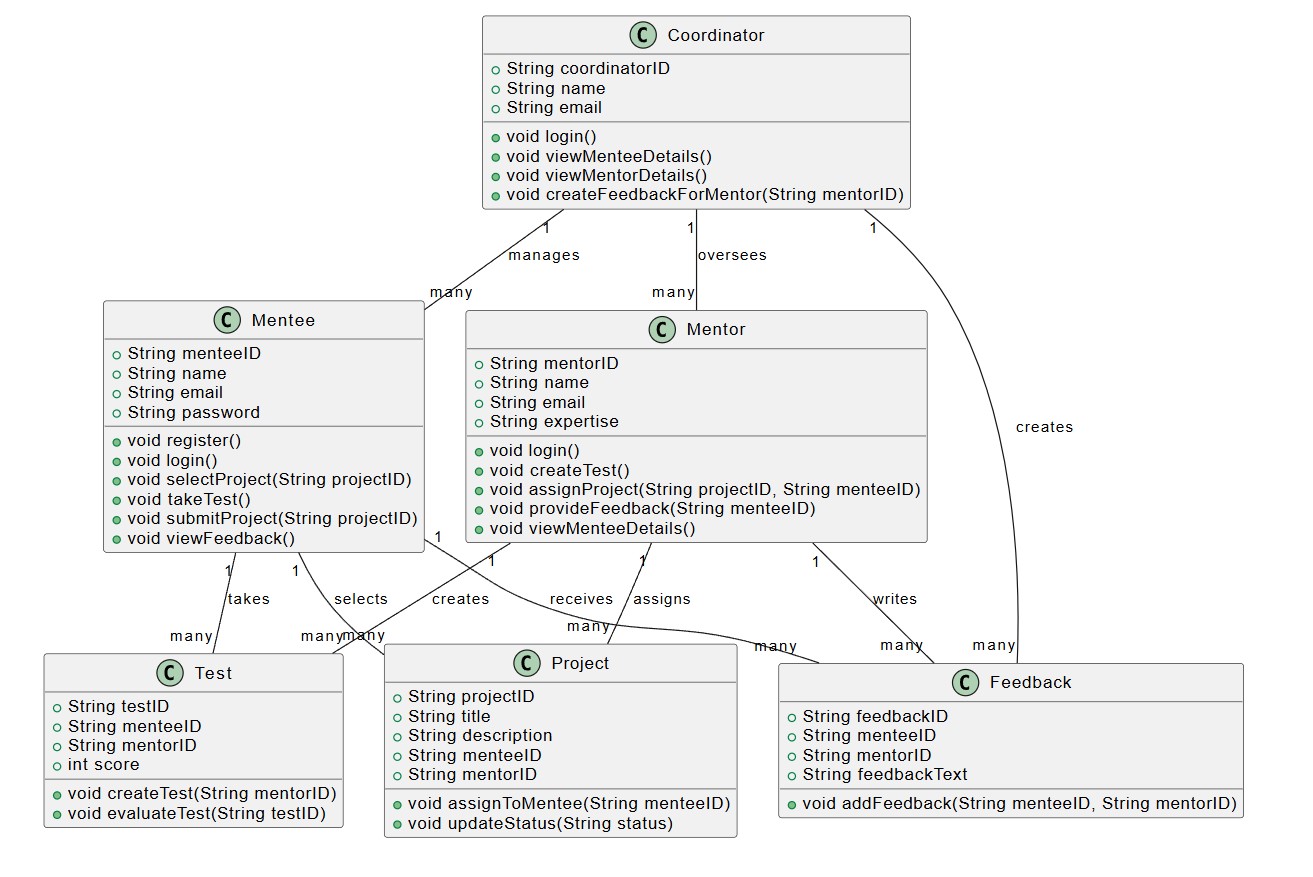


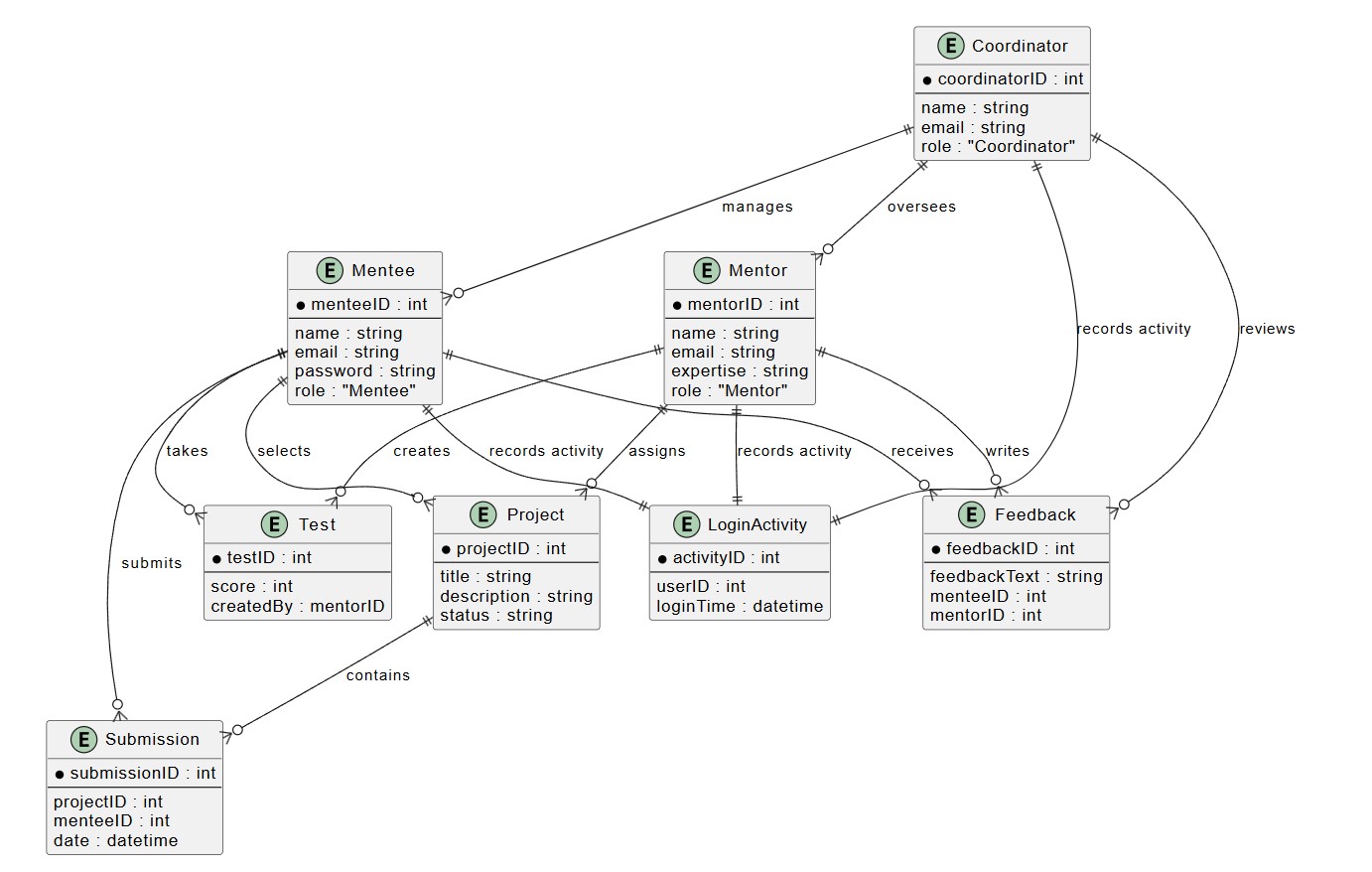
Figure 10: Component Diagram for project coordinator

* + - Class Diagram



*Figure 11: Class Diagram*

## Database Diagram

**

*Figure 12: Entity Relationship Diagram*

## Assumptions and Dependencies

1. **Technical Feasibility Assumptions**:

* **Hardware Infrastructure**: The assumption is that the necessary hardware, including servers, computers, and network components, will be available to support the system. These should meet the requirements for smooth functioning of the backend, frontend, and database systems.
* **Software Dependencies**: It is assumed that the required software, including technologies like Node.js, React, and MongoDB, will be available with compatible versions to run the system efficiently.
* **Third-Party Service Integration**: The project assumes the successful integration of third-party services like email APIs for notifications, cloud storage for file management, and secure authentication methods for user login.

2. **Subsystems or Component Availability**:

* **Database System**: Assumes continuous access to the chosen database (MongoDB ) for storing user data, project submissions, and feedback, without disruptions.
* **User Authentication**: Assumes that the authentication system (OAuth, JWT, etc.) will be available and functional to provide secure login and user management for both mentees (students) and mentors (faculty).
* **File Management**: Assumes the availability of cloud storage services or an integrated file management system to handle project document uploads and downloads effectively.

3. **Project-Related Assumptions**:

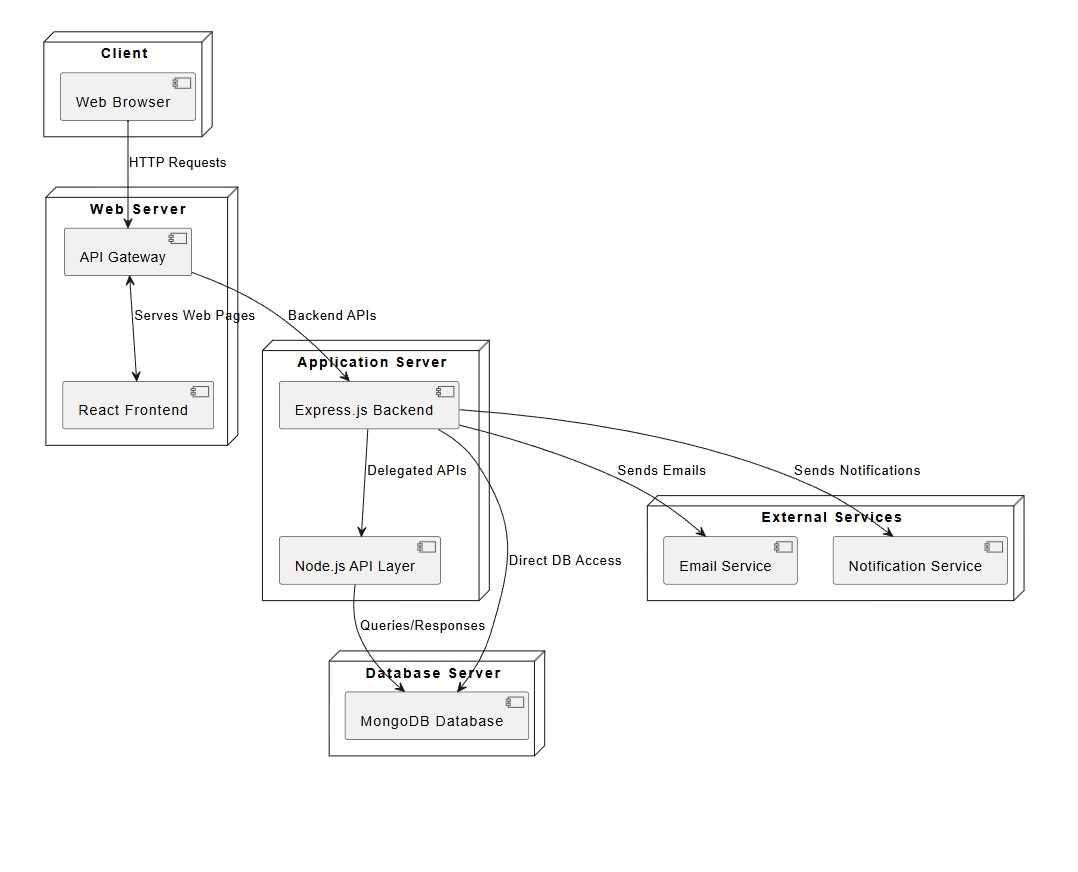
* **User Adoption**: Assumes active participation and adoption of the platform by mentees (students) and mentors (faculty) for project tracking, submission, and communication. The system relies on users being comfortable with using the platform for project management tasks.
* **Data Integrity**: Assumes that all data entered by users, including project submissions, timelines, and feedback, is accurate and up-to-date to ensure smooth project management.

4. **Dependencies on External Factors**:

* **Internet Connectivity**: Assumes that mentees and mentors will have reliable internet access to use the platform for real-time submissions, feedback, and communication. Any internet disruption could affect the platform's usability.
* **Compliance with Institutional Policies**: Assumes that the system adheres to institutional guidelines, data privacy regulations (such as GDPR for student information), and academic integrity standards.

# 5.System Architecture

## Client-Server Architecture

****

*Figure 13: Client Server Architecture*

## Communication Interface

Communication interfaces outline the interaction mechanisms between various components and subsystems within the **Student Project Management** platform. The following communication interfaces ensure smooth

data exchange and functionality:

1. **API Endpoints**:
   * **RESTful APIs**: The platform utilizes RESTful APIs to enable communication between the client-side application (React for the web interface) and the server-side backend (Node.js ). These APIs allow data exchange for key functionalities such as project submissions, mentee (student) authentication, project status tracking, and mentor (faculty) feedback.
   * **Endpoints for Data Exchange**: API endpoints are defined to handle specific tasks, such as:
     + **Mentee Authentication**: Validates user credentials for secure access to the platform.
     + **Project Management**: Handles the submission and retrieval of project documents and

statuses.

* + - **Mentor Feedback**: Facilitates the exchange of mentor reviews, comments, and

project evaluations.

1. **External Service Integration**:
   * **Email Notifications API**: Integrates with an email service provider (like SendGrid or SMTP) to send automated notifications regarding project deadlines, feedback, and announcements to mentees and mentors.
   * **Cloud Storage Service**: Communication with cloud storage platforms (such as AWS S3 or Google Drive) is used for storing and managing project files, allowing users to upload and download documents securely.
2. **Database Interaction**:
   * **Database Connectivity**: The backend (Node.js) connects to the MongoDB database through specific database connectors. These interfaces allow seamless interaction between the backend logic and the data storage layer.
   * **Query Interfaces**: SQL or NoSQL query interfaces are utilized for inserting, updating, retrieving, and managing student project data, including personal information, project submissions, and feedback records.
3. **WebSocket for Real-Time Communication**:
   * **Mentor-Mentee Chat System**: WebSocket or similar technology facilitates real-time communication between mentees and mentors, enabling instant messaging for discussing project updates or clarifications.

## Networking Interfaces:

* + - **Internet Connectivity:** Ensures connectivity between the client-side applications, server- side components, and external services via the internet.
    - **Secure Communication Protocols:** Integration of secure communication protocols (HTTPS, SSL/TLS) for encrypted data transmission between the clients and the server.

These communication interfaces define the pathways and methods through which different system components and external entities interact, ensuring effective data exchange and system functionality within the platform.

# 7.Conclusion & Future scope

## Conclusion

## Summary of Achievements:

## Accomplishments: The Student Project Management System successfully implemented key features, such as project submission, mentor feedback, and real-time communication, enabling smooth interaction between mentees and mentors.

## Key Objectives: The system met its primary objectives by providing a centralized platform for tracking project progress, enhancing collaboration, and streamlining administrative processes, such as grading and evaluation.

## Challenges Overcome: The project faced challenges, including database optimization and ensuring real-time communication, which were mitigated through efficient resource management, and adopting Web Sockets for communication.

## Impact:

## Benefits to Mentees: The platform enhances the learning experience by offering mentees a user-friendly interface to submit projects, track progress, and receive timely feedback from mentors.

## Contribution to Education: The system facilitates better academic project management, empowering both mentees and mentors by improving communication and reducing administrative workload.

## Future Scope

* **New Features**: Future features may include automated plagiarism detection, advanced analytics for project evaluations, and a peer-review system.
* **Technological Upgrades**: Opportunities include upgrading the frontend to support Progressive Web Applications (PWAs) for an improved user experience, and implementing AI to analyze mentee performance over time.
* **Scalability Considerations**: The platform can be expanded to accommodate more mentees and mentors, with enhanced support for larger institutions or multi-campus operations.

**Research and Development**:

* **AI Integration**: The platform can explore AI-driven project recommendations, predictive analytics for mentee performance, and automated grading tools.
* **Mobile App Enhancement**: Future improvements may include optimizing the mobile version for better responsiveness, offline access, and push notifications.

**Community Engagement and Partnerships**:

* **Collaborations**: Partnering with educational institutions or tech companies could bring in additional functionalities, like integrated learning modules or external project resources.
* **Community Growth**: Strategies to build a collaborative platform for students and mentors, such as integrating discussion forums and live Q&A sessions.

**Conclusion of Future Scope**:

* **Vision and Direction**: The vision for the **Student Project Management System** is to evolve into a comprehensive academic tool that not only manages projects but also supports learning through data-driven insights, fostering mentor-mentee collaboration.

# Concerns / Queries / Doubts if any:

# 0

# Project-related Queries:

# How can we implement a secure, scalable project submission system? Ensuring secure data transfer and storage for mentee project submissions is essential. The system will need a robust authentication system and encryption for sensitive data.

# What methods can we employ for real-time mentor-mentee communication? We need to evaluate the use of Web Sockets or other real-time communication tools to facilitate seamless interaction between mentors and mentees.

# How can we ensure smooth integration of different document formats? The system should support various document formats for project submission (PDF, Word, etc.), raising the question of compatibility and conversion handling.

# Technological Queries:

# Are there challenges in implementing role-based access using Node.js? Securing endpoints with role-based access control is crucial for ensuring that only authorized users can access certain features, such as project grading or submission evaluation.

# How can we ensure smooth performance under high user load? With multiple users (mentees, mentors, administrators) accessing the system, we must consider optimizing database queries and system architecture to handle high load efficiently.

# What are the best practices for UI development using React for the web interface? React's component-based structure can enhance modularity and reusability. However, ensuring responsiveness and smooth interaction across different devices poses challenges in UI optimization.

# Any suggestions for streamlining Agile methodology for iterative development? Using Agile methodology requires efficient sprint planning and continuous integration tools like Jenkins or GitLab. Iterative testing and feedback loops need to be established for optimal progress tracking.