Software Requirements Specification

for

Final Year Project Application

Version 1.0

Prepared by

Group Name: Project Hub

Veda Sri	**SE22UCSE044**	**se22ucse044@mahindrauniversity.edu.in**
Parthavi K	**SE22UCSE196**	**se22ucse196@mahindrauniversity.edu.in**
Ankita Sarkar	**SE22UCSE030**	**se22ucse030@mahindrauniversity.edu.in**
Vidhita Reddy	**SE22UCSE290**	**se22ucse290@mahindrauniversity.edu.in**
Srija Lukka	**SE22UCSE315**	**se22ucse315@mahindrauniversity.edu.in**

Instructor: Vijay Rao

Course: Software Engineering

Lab Section: CSE-1,3,4

Teaching Assistant: Swapna S

Date: 10/03/2025

Version	Primary Author(s)	Description of Version	Date Completed
Draft Type and Number	Full Name	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	00/00/00

1

1.1 Document Purpose

This Software Requirements Specification (SRS) document defines the functional and non-functional requirements for the Final Year Project Application System, a web-based platform designed to facilitate the project application process for final-year students at Mahindra University. The system enables students to browse available projects offered by faculty members, submit applications, and receive approvals or rejections. It also allows faculty members to manage project availability and student applications efficiently. Additionally, the platform includes a meeting scheduling feature to streamline communication between students and faculty.

This document serves as a comprehensive guide for developers, stakeholders, and testers by outlining the system's objectives, functionalities, constraints, and design considerations. It ensures that all requirements are well-defined, structured, and aligned with the project's goals, providing a reference throughout the development lifecycle.

1.2 Product Scope

The Final Year Project Application System is a web-based platform designed to streamline the process of project selection and application for final-year students at Mahindra University. The system provides real-time project availability, allowing students to browse and apply for projects based on their interests. Faculty members can manage project listings, review student

applications, and approve or reject applications through an intuitive interface. Additionally, the platform integrates a meeting scheduling feature to facilitate seamless communication between students and faculty.

The system enhances efficiency by reducing manual processes, ensuring transparency in project allocations, and improving collaboration between students and faculty. It eliminates confusion regarding project availability and application statuses, providing a structured workflow for both students and faculty. With secure authentication, role-based access, and a user-friendly interface, the platform aims to simplify and improve the project selection and management process, ultimately benefiting students, faculty, and university administrators.

1.3 Intended Audience and Document Overview

This document is intended for:

- **Developers** – To implement system features based on the specified requirements.
- **Faculty Members** – To manage projects, review student applications, and schedule meetings.
- **Students** – To browse available projects, apply, and track application status.
- **Testers** – To verify system functionality, performance, and security.
- **University Administrators** – To ensure integration with university policies and infrastructure.

The document is structured as follows:

1. **Introduction** – Overview, scope, and audience.
2. **Overall Description** – System functionalities, constraints, and dependencies.
3. **Specific Requirements** – Functional requirements, interfaces, and use cases.
4. **Non-functional Requirements** – Performance, security, and software quality attributes.
5. **Other Requirements & Appendices** – Additional technical details and data references.

Readers should start with the **Introduction** and proceed to relevant sections based on their role.

1.4 Definitions, Acronyms and Abbreviations

- **API** – Application Programming Interface (used for backend communication)
- **CRUD** – Create, Read, Update, Delete (basic database operations)
- **DBMS** – Database Management System
- **JWT** – JSON Web Token (used for authentication)
- **MU** – Mahindra University
- **SRS** – Software Requirements Specification
- **UI** – User Interface

1.5 Document Conventions

This document follows IEEE SRS standards using Arial 12pt font, single spacing and structured section numbering.

1.6 References and Acknowledgments

This document references the IEEE Standard for Software Requirements Specifications (IEEE 830-1998) as a guideline for structuring and defining software requirements. Additionally, it follows the University Guidelines for Final Year Projects Provided by Mahindra University to ensure compliance with academic standards.

We would like to acknowledge our instructor, teaching assistant, and Mahindra University faculty for their valuable guidance and support in shaping the requirements for this system. Their insights have helped in ensuring the platform meets the needs of both students and faculty effectively.

2

2.1 Product Overview

The **Final Year Project Application System** is a new, self-contained web-based platform designed to simplify the process of project allocation for final-year students and faculty at Mahindra University. Currently, project applications are managed manually, leading to inefficiencies such as miscommunication, delays, and lack of transparency. This system aims to replace such outdated methods by providing a structured, real-time platform where students can explore available projects, submit applications, and receive approvals or rejections. Faculty members can efficiently manage project listings, review student applications, and schedule meetings—all in one place.

The system consists of interconnected modules, including User Authentication, Project Management, Application Handling, Meeting Scheduling, and Notifications. It interacts with a relational database to store and manage data securely. The platform is designed to be user-friendly, scalable, and secure, ensuring smooth communication between students and faculty.

2.2 Product Functionality

The **Final Year Project Application System** provides the following major functionalities:

- **User Authentication** – Secure login and role-based access for students and faculty.
- **Project Management** – Faculty can create, edit, and delete project listings.
- **Project Application** – Students can browse available projects and submit applications.
- **Application Review** – Faculty can approve or reject student applications.
- **Meeting Scheduling** – Integrated calendar for faculty and students to schedule meetings.

- **Notification System** – Email and in-app notifications for approvals, rejections, and meeting updates.
- **Database Management** – Stores user details, projects, applications, and meeting schedules securely.

These features ensure an efficient and transparent project selection and management process.

2.3 Design and Implementation Constraints

The system will be developed using (fill this part) for the frontend, **Node.js with Express.js** for the backend and **MySQL/PostgreSQL** for the database. **JWT authentication** and **HTTPS** will ensure security. The **COMET method** will be used for structured design, and **UML modeling** for system visualization. The platform must be **responsive** and integrate with **email and calendar APIs** for notifications and scheduling.

2.4 Assumptions and Dependencies

- Users will have stable internet access to interact with the system.
- Faculty members will actively manage project listings by updating availability.
- Students will use valid university credentials for authentication.
- The system will rely on third-party email and calendar APIs for notifications and scheduling.
- The database server will be available and properly maintained to store project and user data securely.

Any changes to these assumptions may impact the system's functionality and performance.

3

3.1 External Interface Requirements

3.1.1 User Interfaces

The website will provide an intuitive interface where students can browse available projects, apply for projects, and track their application status. Faculty members will have a dashboard to manage projects, approve or reject student applications, and schedule meetings.

The admin will manage the users.

- **Students:**
  * View available projects
  * Apply for projects
  * Track application status
  * Receive meeting details
- **Faculty:**
  * Create and manage projects
  * Approve/reject student applications

* Schedule meetings with students

- **Admin:**
  * Manage users (students and faculty)
  * Oversee system operations

The user interface will include menus, buttons, and forms for interaction. The design will be responsive to work on different screen sizes.

3.1.2 Hardware Interfaces

The system will be a web-based application, accessible via:
- Desktop computers
- Laptops
- Tablets
- Smartphones

No additional hardware interfaces are required beyond standard computing devices.

3.1.3 Software Interfaces

- Frontend: React.js
- Backend: Node.js with Express.js
- Database: PostgreSQL (via Prisma ORM)
- Authentication: JWT-based authentication system
- External Services: Email notifications for application updates and meeting scheduling

3.2 Functional Requirements

3.2.1 F1: Student Registration and Authentication
- The system shall allow students to create an account using their university email.
- The system shall authenticate users via a secure login mechanism.

3.2.2 F2: Faculty Registration and Authentication
- Faculty members shall be able to register and log in securely.

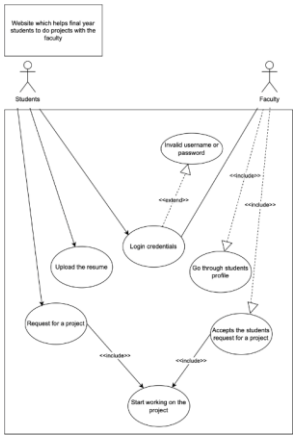
3.2.3 F3: Project Management
- Faculty members shall be able to create and update projects.
- The system shall display available projects to students.

3.2.4 F4: Student Applications
- Students shall be able to apply for projects.
- The system shall notify faculty when a student applies.

3.2.5 F5: Application Status Updates
- Faculty members shall approve or reject student applications.
- The system shall notify students of the decision.

3.2.6 F6: Meeting Scheduling
- Faculty shall schedule meetings with selected students.
- The system shall notify students about scheduled meetings.

3.3 Use Case Model



Use Case #1: Student Applies for a Project (U1)

Author: [Your Team Member Name]
Purpose: Allows students to browse projects and submit applications.
Requirements Traceability: Related to F3 and F4.
Priority: High
Preconditions:

- Student must be logged in.
- Student must upload a resume before applying for a project.

Postconditions:

- Application is submitted.
- Faculty is notified.

Actors:

- Student
- Faculty
- System

Flow of Events:

- Student logs in.
- Student uploads a resume (if not already uploaded).
- Student views available projects.
- Student selects a project and submits an application.
- System stores the application and notifies the faculty.

Use Case #2: Faculty Approves/Rejects Application (U2)

Author: [Your Team Member Name]
Purpose: Faculty evaluates student applications.
Requirements Traceability: Related to F5.
Priority: High

Preconditions:

- Faculty must be logged in.
- Faculty must review the student's profile before making a decision.

Postconditions:

- Student is notified of the application status.
- Actors:
- Faculty
- Student
- System

Flow of Events:

- Faculty logs in.
- Faculty reviews the student's profile (includes uploaded resume).
- Faculty views received applications.
- Faculty selects an application and approves/rejects it.
- System updates the application status and notifies the student.

Use Case #3: Meeting Scheduling (U3)

Author: [Your Team Member Name]
Purpose: Faculty schedules meetings with students for discussions.
Requirements Traceability: Related to F6.
Priority: Medium
Preconditions:

- Student's application must be approved.
- Faculty must be logged in.

Postconditions:

- Meeting details are shared with the student.

Actors:

- Faculty
- Student
- System

Flow of Events:

- Faculty logs in.
- Faculty selects an approved application.
- Faculty schedules a meeting.
- System sends meeting details to the student.

4.1 Performance Requirements

System Responsiveness: The application should respond to user actions, such as loading project listings or submitting applications, within 2 seconds under standard operating conditions.

Scalability: The system must support up to 10,000 concurrent users without performance degradation, ensuring smooth operation during peak usage times, such as project submission deadlines.

Data Processing: Updates to project availability statuses and application records should be processed and reflected in the system within 5 seconds to provide real-time information to users.

Calendar Functionality: The integrated calendar should handle scheduling and display of up to 500 simultaneous meetings without noticeable lag or performance issues.

4.2 Safety and Security Requirements

User Authentication: Implement robust authentication mechanisms to ensure that only authorized students and faculty can access the system. This includes secure login protocols and, where appropriate, multi-factor authentication.

Data Privacy: Comply with relevant data protection regulations to safeguard personal information. Ensure that user data is encrypted both in transit and at rest, and that access is restricted based on user roles.

Input Validation: All user inputs must be validated to prevent common security vulnerabilities such as SQL injection and cross-site scripting (XSS) attacks. This involves sanitizing inputs and employing parameterized queries.

Session Management: Secure session handling practices should be in place to prevent session hijacking and fixation attacks. This includes setting appropriate session timeouts and regenerating session IDs after login.

Audit Logging: Maintain comprehensive logs of user activities, especially actions related to project applications and approvals. These logs should be protected from unauthorized access and regularly reviewed for suspicious activities.

4.3 Software Quality Attributes

4.3.1 Reliability

Error Handling: The system should gracefully handle unexpected errors without crashing. Users should receive informative messages guiding them on corrective actions.

Data Integrity: Ensure that all data transactions, such as project applications and approvals, are processed accurately and consistently, even in cases of system failures.

4.3.2 Usability

User Interface (UI): Design an intuitive and accessible UI that adheres to usability standards, ensuring that users can navigate the system with ease.

Accessibility: Ensure the application meets accessibility standards (e.g., WCAG 2.1) to accommodate users with disabilities, providing features like screen reader support and keyboard navigation.

4.3.3 Maintainability

Code Modularity: Develop the system using modular code practices, allowing for easy updates and scalability. This includes separating concerns and adhering to coding standards.

Documentation: Provide comprehensive documentation for both users and developers, facilitating ease of use and future development or troubleshooting efforts.

4.3.4 Performance Efficiency

Resource Management: Optimize the application to efficiently use server and client resources, ensuring quick load times and responsiveness. This includes optimizing database queries and minimizing client-side rendering times.

Load Testing: Regularly conduct load testing to identify and address potential performance bottlenecks, ensuring the system can handle expected user loads.

4.3.5 Security

Compliance with OWASP Guidelines: Adhere to the OWASP Top Ten security risks to protect the application from common vulnerabilities. This includes regular security assessments and updates to address emerging threats.

Secure Development Practices: Incorporate security into the software development lifecycle, ensuring that security considerations are addressed at every stage of development.

<Data dictionary is used to track all the different variables, states and functional requirements that you described in your document. Make sure to include the complete list of all constants, state variables (and their possible states), inputs and outputs in a table. In the table, include the description of these items as well as all related operations and requirements.>

<Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist in determining the effort put forth to produce this document>