Software Requirements

Specification

**for**

**PROJECT NAME:Create a website that helps final year students apply for projects**

**offered by MU faculty**

# Version <1.0> Prepared byTeam no:29

**Group Name:**

|  |  |  |
| --- | --- | --- |
| **Jangidi Laxmi Santhoshi** | **SE22UARI059** | [**se22uari059@mahindrauniversity.edu.in**](mailto:se22uari059@mahindrauniversity.edu.in) |
| **Kuppireddy Sri Sphoorthy** | **SE22UARI079** | [**se22uari079@mahindrauniversity.edu.in**](mailto:se22uari079@mahindrauniversity.edu.in) |
| **Penmetsa Mounika** | **SE22UARI125** | [**se22uari125@mahindrauniversity.edu.in**](mailto:se22uari125@mahindrauniversity.edu.in) |
| **Purumandla Sahasra Reddy** | **SE22UARI135** | [**se22uari135@mahindrauniversity.edu.in**](mailto:se22uari135@mahindrauniversity.edu.in) |

|  |  |
| --- | --- |
| **Instructor:** | **Avinash Arun Chauhan** |
| **Course:** | **Software Engineering** |
| **Lab Section:** | ***Tuesday 8.25 am- 10.20 am*** |
| **Teaching Assistant:** | ***Swapna S*** |
| **Date:** | **10-03-2025** |

CONTENTS II

**Contents**

REVISIONS III

1. INTRODUCTION 1
   1. [Document Purpose 1](#_TOC_250015)
   2. [Product Scope 1](#_TOC_250014)
   3. [Intended Audience and Document Overview 1](#_TOC_250013)
   4. [Definitions, Acronyms and Abbreviations 2](#_TOC_250012)
   5. [Document Conventions 2](#_TOC_250011)
   6. [References and Acknowledgments 3](#_TOC_250010)
2. OVERALL DESCRIPTION 4
   1. [Product Overview 4](#_TOC_250009)
   2. [Product Functionality 5](#_TOC_250008)
   3. [Design and Implementation Constraints 5](#_TOC_250007)
   4. [Assumptions and Dependencies 5](#_TOC_250006)
3. SPECIFIC REQUIREMENTS 6
   1. [External Interface Requirements 6](#_TOC_250005)
   2. [Functional Requirements 7](#_TOC_250004)
   3. [Use Case Model 8](#_TOC_250003)
4. OTHER NON-FUNCTIONAL REQUIREMENTS 27
   1. [Performance Requirements 27](#_TOC_250002)
   2. [Safety and Security Requirements 28](#_TOC_250001)
   3. [Software Quality Attributes 29](#_TOC_250000)
5. OTHER REQUIREMENTS 29

APPENDIX A – DATA DICTIONARY 30

APPENDIX B - GROUP LOG 32

**Revisions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| 1.0 | Team 29 | Initial Draft. | 07/02/2025 |

**1 Introduction**

**MU Connect**(name of the website of our project) is a web-based system that assists final-year students at Mahindra University in searching and applying for faculty-supervised projects. It also provides faculty with tools to manage applications, select candidates, and schedule meetings.

The purpose of this document is to define the functional and non-functional requirements of the system. It serves as a guide for developers, testers, faculty, and university administration to ensure that all project goals and functionalities are met.

# Document Purpose

This Software Requirements Specification (SRS) document provides a detailed outline of the **MU Connect** system. It defines the system's objectives, user roles, major features, constraints, and dependencies. The document ensures that developers understand what needs to be built, testers know what needs to be verified, and university faculty and students understand how to use the system.

# Product Scope

**MU Connect** aims to simplify the project selection and application process for final-year students and faculty members. Key functionalities includes:

* ***User Authentication*** *– Secure login system for students and faculty.*
* ***Project Listings*** *– Faculty can post, update, and delete project offerings.*
* ***Application Submission*** *– Students can browse and apply for available projects.*
* ***Faculty Review & Approval*** *– Faculty can review applications, accept/reject students, and*

notify applicants of their decisions.

* ***Meeting Scheduling*** *– Integrated calendar for scheduling project discussions.*
* ***Chat Functionality*** *– Direct communication between students and faculty.*

The system improves efficiency, transparency, and organization by eliminating the need for manual tracking and communication via emails.

# Intended Audience and Document Overview

This document is intended for:

* ***Developers*** *– To understand system requirements and technical implementation.*
* ***Project Managers & Faculty*** *– To ensure system objectives align with university needs.*
* ***Students*** *– To learn about the application process and system functionalities.*
* ***Testers*** *– To validate that the system functions correctly.*
* ***University Administration*** *– To ensure compliance with academic regulations. The document is organized into the following sections:*
  1. ***Introduction*** *– Project overview and objectives.*
  2. ***Overall Description*** *– System perspective, functions, and constraints.*
  3. ***Specific Requirements*** *– Detailed functional and non-functional requirements.*
  4. ***External Interface Requirements*** *– System interaction with hardware/software components.*
  5. ***Other Requirements*** *– Additional system requirements, such as security and compliance*

# Definitions, Acronyms and Abbreviations

* ***MU Connect*** *– The project management system for Mahindra University.*
* ***SRS*** *– Software Requirements Specification.*
* ***UI*** *– User Interface.*
* ***API*** *– Application Programming Interface.*
* ***DB*** *– Database.*
* ***CRUD*** *– Create, Read, Update, Delete operations.*
* ***REST API*** *– A software architecture style for web services*

# Document Conventions

This document follows the IEEE formatting requirements to ensure consistency and readability:

* + 1. ***Formatting Conventions***
       - ***Font****: Arial, size 11 for body text.*
       - ***Headings****: Bold, numbered sections for easy navigation.*
       - ***Italics****: Used for important notes or references.*
       - ***Spacing****: Single-spaced text with 1-inch margins.*

## Naming Conventions

* + - * *All variables and database fields will follow the camelCase format.*
      * *API endpoints will follow RESTful principles.*
      * *Functions and classes in the codebase will use Pascal*Case.

# References and Acknowledgments

* *SOW Document (07-02-2025) – Detailed project scope*

*.*

* *ER Diagram for MU Connect – System entity relationships.*
* *IEEE Software Engineering Standards – Guidelines for writing SRS documents.*
* *Mahindra University Policies – Compliance with academic project guidelines.*

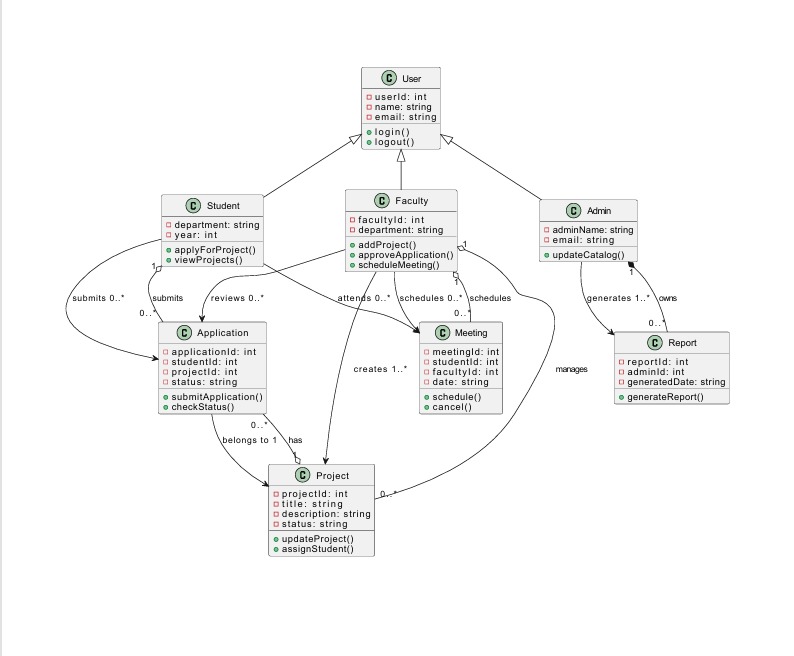
**2 Overall Description**

# Product Overview

**MU Connect** is a standalone, web-based system designed to facilitate project selection and management for final-year students and faculty members at Mahindra University. It replaces the traditional manual process of project allocation with a structured and automated approach. The system allows students to browse faculty-offered projects, submit applications, and receive real- time updates on application status. Faculty members can post projects, review applications, approve or reject candidates, and schedule meetings with selected students.

The system consists of the following major components:

* + - ***User Authentication Module*** *– Manages login, registration, and access control.*
    - ***Project Management Module*** *– Faculty can create and update projects.*
    - ***Application Processing Module*** *– Students can apply for projects, and faculty can review applications.*
    - ***Meeting Scheduler*** *– Facilitates appointment scheduling between faculty and students.*
    - ***Chat System*** *– Allows faculty and students to communicate directly.*



# Product Functionality

The main functions of MU Connect include:

* + - ***Secure authentication for users*** *(students and faculty).*
    - ***Project creation and management*** *by faculty.*
    - ***Project browsing and application*** *by students.*
    - ***Application review and approval*** *by faculty.*
    - ***Automated scheduling of meetings*** *for selected students.*
    - ***Chat system*** *for easy communication between students and faculty.*

# Design and Implementation Constraints

* + - ***Web-Based System:*** *MU Connect must be accessible through modern web browsers.*
    - ***Database Storage:*** *Project details, applications, and user credentials must be stored securely.*
      * ***Security Measures:*** *Authentication and authorization must be enforced using encryption and access control.*
      * ***Third-Party Integrations****: The system may integrate with email services, calendar applications, and notification APIs.*
      * ***Compliance with Standards****: The system must follow IEEE software engineering best*

practices.

* + - * ***Use of UML Diagrams****: The system design must include UML-based representations of workflows and interactions.*

# Assumptions and Dependencies

* + - *Users (students and faculty) will have internet access.*
    - *Faculty members will actively update and manage project listings.*
    - *The system will be hosted on a cloud server for scalability.*
    - *Users will follow Mahindra University guidelines while using the platform.*

**3 Specific Requirements**

# External Interface Requirements

## User Interfaces

The **MU Connect** web application provides an intuitive and responsive user interface for students and faculty, ensuring seamless interaction.

### User Interaction Flow:

* + - * + ***Navigation****: A top navigation bar allows users to switch between different sections such as "Projects," "Applications," and "Meetings."*
        + ***Project Listings****: A card-based display with filtering options to browse projects.*
        + ***Application Submission****: A form with pre-filled student details and a submit button for project applications.*
        + ***Faculty Dashboard****: A table displaying student applications with options to approve/reject.*
        + ***Meeting Scheduler****: A calendar widget where faculty can set up and modify meetings.*

### UI Elements:

* + - * + ***Buttons****: Apply, Approve, Reject, Schedule Meeting*
        + ***Forms****: Login, Application Submission, Project Creation*
        + ***Tables****: Student applications, faculty project listings*
        + ***Cards****: Display projects with details like description, faculty name, and availability*

## Hardware Interfaces

**MU Connect** is a web-based application, hardware interfaces are minimal. However, the system will support:

* ***Computers & Laptops****: Web-based access via browsers like Chrome, Firefox, and Edge.*
* ***University Servers****: Hosting and managing backend services for authentication and data storage.*
* ***Calendar API Integration****: Synchronization with Google Calendar or other scheduling tools.*

## Software Interfaces

The **MU Connect** platform interacts with multiple software components, ensuring smooth data exchange and user experience.

* ***Backend System****: Built with* ***Node.js & Express.js****, handling authentication, project management, and applications.*
* ***Database****:* ***MongoDB****, storing project details, student applications, and faculty approvals.*
* ***Authentication****: Secure login system using* ***university credentials*** *(OAuth or JWT).*
* ***Calendar API****: Integration with* ***Google Calendar API*** *for scheduling and meeting management.*

# Functional Requirements

Functional requirements define the expected behavior of the **MU Connect** system. These requirements ensure that students and faculty can efficiently interact with the platform for project applications and management.

### F1: User Authentication

* *The system shall provide secure login functionality for students and faculty using university credentials.*
* *The system shall ensure role-based access control, restricting unauthorized actions.*

### F2: Project Creation and Management

* *The system shall allow faculty members to create, update, and delete projects.*
* *Faculty shall be able to set project availability and define required skills.*

### F3: Project Browsing and Application

* *The system shall allow students to browse projects with filters based on faculty, category, and availability.*
* *The system shall allow students to apply for available projects with a single click.*

### F4: Application Review and Approval

* *The system shall allow faculty to review, approve, or reject student applications.*
* *The system shall provide a reason for rejection and notify the student.*

### F5: Meeting Scheduling

* *The system shall integrate with a calendar API to schedule meetings for approved students.*
* *The system shall allow faculty to set available time slots for meetings.*

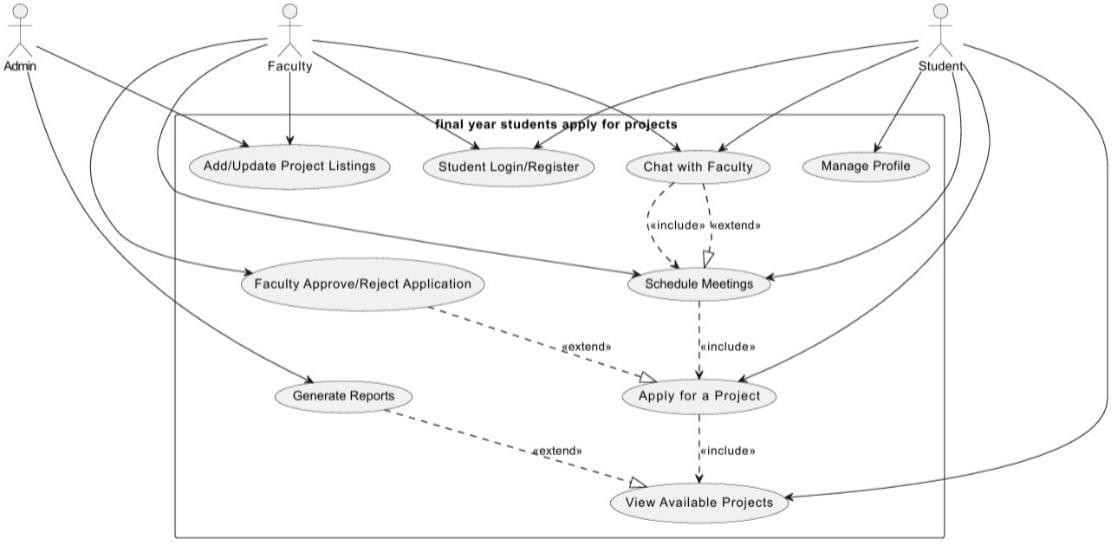
### F6: Notifications and Alerts

* *The system shall send real-time notifications for approvals, rejections, and meeting reminders.*

### F7: Chat System for Communication

* *The system shall include a chat feature for direct communication between students and faculty.*
* *The chat system shall support text messaging*.

# Use Case Model

****

* + 1. ***Use Case #1 : StudenLogin/Register (U1)***

***Author:*** *Jangidi Laxmi santhoshi*

### Purpose:

This use case allows students to either log in to an existing account or register as a new user. It ensures secure authentication, preventing unauthorized access.

### Requirements Traceability:

* ***R1:*** *Secure authentication for students*
* ***R2:*** *Unique student profile creation*
* ***R3:*** *Email verification before account activation*

### Priority: High

* *Essential as it is the* ***entry point*** *for students into the system.*

### Preconditions:

* *The student must have a* ***valid email ID and password*** *to log in.*
* *If registering, the student must provide* ***full name, email, and password****.*
* *The system database must be* ***active*** *to verify credentials.*

### Postconditions:

* ***Successful Login:*** *Student is redirected to the dashboard.*
* ***Successful Registration:*** *A new student profile is created, and a* ***verification email is sent****.*
* ***Failed Login:*** *The student is shown an* ***error message****.*

### Actors:

* ***Primary Actor:*** *Student*
* ***Supporting Actor:*** *System Database (Authentication Service)*

### Extends:

* ***Forgot Password (U6)*** *– If a student forgets their password, they can reset it.*

### Flow of Events:

1. ***Basic Flow***
   1. *The student opens the login page.*
   2. *They enter their* ***email*** *and* ***password****.*
   3. *The system checks the database for matching credentials.*
   4. *If correct, the student is redirected to the dashboard.*

### Alternative Flow

* 1. *The student selects* ***"Register"*** *instead of logging in.*
  2. *They enter* ***full name, email, and create a password****.*
  3. *The system verifies whether the* ***email is already registered****.*
  4. *If the email is* ***new****, the system:*
     + *Stores student details in the database.*
  5. *Once the student verifies via email, they can* ***log in successfully****.*

### Exceptions

* ***Incorrect Credentials:*** *The student enters an incorrect email/password. → "Invalid credentials. Please try again."*
* ***Duplicate Email on Registration:*** *The student tries to register with an already registered email. → "This email is already in use. Please log in."*
* ***Database Connection Failure:*** *The system fails to access student records. → "System error. Please try again later."*

### Includes:

* *None (email verification is handled internally).*

### Notes/Issues:

* ***Security:*** *Implement password hashing (****bcrypt, Argon2****)*

### Session Management: Use JWT or session cookies

* ***Future Feature:*** *Consider* ***Multi-Factor Authentication (MFA)***
  + 1. ***Use Case #2 : Faculty Login (U2) Author:*** *Jangidi Laxmi santhoshi*

### Purpose:

This use case allows faculty members to log in securely and access the system to manage projects, view student applications, and interact with students. It ensures only authorized faculty can access and modify project-related data.

### Requirements Traceability:

* ***R4:*** *Secure authentication for faculty members*
* ***R5:*** *Role-based access control to restrict faculty permissions*
* ***R6:*** *Session management to maintain login sessions securely*

### Priority: High

* *Essential because* ***faculty must log in before managing projects or approving applications****.*

### Preconditions:

* *The faculty must have a* ***registered account****.*
* *The system database must be* ***active*** *to verify login credentials.*
* *Faculty should have* ***valid login credentials (email and password)****.*

### Postconditions:

* ***Successful Login:*** *The faculty is redirected to their dashboard.*
* ***Failed Login:*** *The faculty is shown an error message.*

### Actors:

* ***Primary Actor:*** *Faculty*
* ***Supporting Actor:*** *System Database (Authentication Service)*

### Extends:

* ***Forgot Password (U6)*** *– If a faculty member forgets their password, they can reset it.*

### Flow of Events:

1. ***Basic Flow***
2. *The faculty member opens the login page.*
3. *They enter their* ***email and password****.*
4. *The system validates credentials against the faculty database.*
5. *If credentials match, the faculty is redirected to the dashboard.*

### Alternative Flow

1. *If the faculty session has expired, they must log in again.*
2. *The system redirects them to the login page with a message: "Your session has expired. Please log in again."*

### Exceptions (Error Handling)

* + ***Incorrect Credentials:*** *Faculty enters an incorrect email or password → "Invalid credentials. Please try again."*
  + ***Unverified Account:*** *The faculty account is* ***not yet verified*** *→ "Your account is pending verification. Contact admin."*
  + ***Database Connection Failure:*** *The system cannot fetch faculty records → "System error. Please try again later."*

***Includes:***

* + ***None (email verification is part of the same process).***

***Notes/Issues:***

* + ***Security:*** *Faculty passwords should be hashed (****bcrypt, Argon2****).*

### Session Management: Implement automatic logout after inactivity.

* + ***Role-Based Access:*** *Faculty should only see* ***authorized sections****.*

## Use Case #3: View Available Projects (U3)

***Author:*** *Kuppireddy Sri Sphoorthy*

### Purpose:

This use case allows students to view available projects listed by faculty. Students can browse project details, including descriptions, required skills, faculty names, and available slots before applying.

### Requirements Traceability:

* ***R7:*** *Students must be able to view all available projects.*
* ***R8:*** *The system should display project details, including faculty information and required skills.*
* ***R9:*** *Students should be able to filter and search projects based on criteria such as faculty name, project category, or availability.*

### Priority: High

* *This use case is essential because students* ***must browse projects before applying****.*

### Preconditions:

* *The student* ***must be logged in*** *to view projects.*
* *At least* ***one active project*** *should be available in the system.*

### Postconditions:

* *The student can view all* ***available projects*** *along with details.*
* *The student may proceed to* ***apply for a project****.*

### Actors:

* ***Primary Actor:*** *Student*
* ***Supporting Actor:*** *System Database (retrieves project listings)*

### Extends:

* ***Apply for a Project (U4)*** *– Viewing projects is a required step before applying.*

### Flow of Events:

1. ***Basic Flow***
2. *The student logs into the system.*
3. *They navigate to the* ***"Available Projects"*** *section.*
4. *The system retrieves and displays all* ***active projects****.*
5. *The student can click on a project to view its* ***detailed description****, including:*

* ***Project title***
* ***Faculty name***
* ***Project description***
* ***Required skills***

1. *The student can choose to* ***apply for a project*** *(Use Case U4).*

### Alternative Flow

* 1. *The student enters search criteria (e.g., faculty name, skill requirements).*
  2. *The system filters and displays* ***matching projects****.*

### Exceptions (Error Handling)

* ***No Projects Available:*** *If no projects are listed, the system displays → "No active projects are available at the moment."*
* ***Database Error:*** *If the project data cannot be retrieved, the system displays → "Error loading projects. Please try again later."*

### Includes:

* *None (search and filtering are internal functions).*

### Notes/Issues:

* ***User Experience:*** *Consider adding sorting and advanced filters.*
* ***Real-Time Updates:*** *Faculty should be able to update project details dynamically.*
* ***Future Feature:*** *Enable students to* ***bookmark projects*** *for later review.*

## Use Case #4: Apply for a Project (U4)

***Author:*** *Kuppireddy Sri Sphoorthy*

### Purpose:

This use case allows students to apply for a project they are interested in after viewing project details. It ensures that students can express their interest in faculty-listed projects, and faculty can review applications before selection.

### Requirements Traceability:

* ***R10:*** *Students must be able to submit an application for a project.*
* ***R11:*** *The system should validate that a student meets the application requirements (if applicable).*
* ***R12:*** *The faculty must be notified when a new application is submitted.*

### Priority: High

* *This use case is critical as it* ***connects students with faculty projects*** *and enables the selection process.*

### Preconditions:

* *The student* ***must be logged in****.*
* *The student* ***must have viewed at least one project (U3)****.*
* *The project must be* ***open for applications*** *and have available slots.*

### Postconditions:

* *The student's application is* ***successfully submitted*** *and stored in the system.*
* *The faculty receives a* ***notification*** *of the new application.*
* *The student can* ***view the application status*** *later.*

### Actors:

* ***Primary Actor:*** *Student (initiates the application process)*
* ***Supporting Actor:*** *Faculty (receives the application)*
* ***System Database:*** *Stores application details*

### Extends:

* ***View Available Projects (U3)*** *– A student must first view a project before applying.*

### Flow of Events:

1. ***Basic Flow***
   1. *The student navigates to the* ***"Available Projects"*** *section (U3).*
   2. *They select a project and click* ***"Apply Now."***
   3. *The system* ***displays an application form****, including:*
      * ***Student’s name*** *(auto-filled)*
      * ***Email*** *(auto-filled)*
      * ***Why are you interested in this project?*** *(text input)*

### Upload Resume (optional)

* 1. *The student fills out the form and clicks* ***"Submit Application."***
  2. *The system:*
     + *Stores the application details in the database.*
     + *Updates the* ***project’s available slots*** *(if applicable).*
     + *Notifies the* ***faculty*** *of the new application.*
  3. *The student is shown a confirmation message:* ***"Your application has been submitted successfully!"***

### Exceptions

* ***Project is Full:*** *If no slots are available, the system displays → "This project is no longer accepting applications."*
* ***Student Already Applied:*** *If the student tries to apply again, the system displays → "You have already applied for this project."*
* ***Database Error:*** *If the application cannot be saved, the system displays → "Application submission failed. Please try again later."*

### Includes:

* *None (as all steps are part of the application process itself).*

### Notes/Issues:

* *Should faculty have the option to request additional documents after application submission?*
* *Should students be allowed to apply for multiple projects at once?*
* *Consider adding an auto-reply feature from faculty upon application submission.*

## Use Case #5: Faculty Approve/Reject Application (U5)

***Author:*** *Penmetsa mounika*

### Purpose:

This use case enables faculty members to review student applications for their projects and either approve or reject them. It ensures that faculty have full control over the selection process, allowing them to accept qualified students while declining others.

### Requirements Traceability:

* ***R13:*** *Faculty must be able to view all student applications for their projects.*
* ***R14:*** *Faculty should be able to approve or reject applications.*
* ***R15:*** *The system must notify students of faculty decisions.*

### Priority: High

* *This use case is* ***essential*** *for project selection, ensuring that only approved students can participate in faculty projects.*

### Preconditions:

* *The faculty* ***must be logged in****.*

### The faculty must have at least one project with pending applications.

* *At least one student* ***must have applied for a project****.*

### Postconditions:

* *The faculty's* ***decision (approve/reject)*** *is recorded in the system.*
* *If approved, the student is added to the project team.*

### Actors:

* ***Primary Actor:*** *Faculty (reviews and makes decisions)*
* ***Supporting Actor:*** *Student (receives approval/rejection notification)*
* ***System Database:*** *Updates application status and stores decisions*

### Extends:

* ***Apply for a Project (U4)*** *– Faculty decisions can only occur if students have applied.*

### Flow of Events:

1. ***Basic Flow***
2. *The faculty logs into the system and navigates to* ***"My Projects."***
3. *They select a project and click* ***"View Applications."***
4. *The system displays a* ***list of student applications****, including:*
   * *Student Name*
   * *Application Date*
   * *Statement of Interest*
   * *Uploaded Resume (if applicable)*
5. *The faculty* ***reviews each application*** *and clicks either:*
   * ***"Approve"*** *→ The student is added to the project team.*
   * ***"Reject"*** *→ The student is removed from the applicant list.*
6. *The system:*

### Updates the application status.

* + ***Sends an email notification*** *to the student with the decision.*
  + ***Updates project availability*** *(reducing available slots if approved).*

### Alternative Flow (Request for More Information)

1. *The student is notified and prompted to provide additional details (e.g., more documents or clarification).*
2. *The application remains pending until the student responds.*

### Exceptions (Error Handling)

* + ***No Pending Applications:*** *If no students have applied, the system displays → "No applications available for this project."*
  + ***Faculty Decision Not Saved:*** *If the system fails to save faculty input, it displays → "Error saving decision. Please try again later."*
  + ***Database Connection Failure:*** *If the application list fails to load, it displays → "Unable to retrieve applications. Please refresh the page."*

### Includes:

* + *None (faculty decision-making is handled within this use case).*

### Notes/Issues:

* + *Should faculty be required to provide a reason for rejection?*
  + *Should approved students receive automatic project meeting invitations?*
  + *Consider allowing bulk approvals/rejections for efficiency.*

## Use Case #6: Chat with Faculty (U6)

***Author:*** *Penmetsa mounika*

### Purpose:

This use case enables students and faculty to communicate directly through a built-in chat system. It allows students to ask project-related questions and faculty to provide guidance in real- time.

### Requirements Traceability:

* ***R16:*** *The system must allow students and faculty to exchange messages.*
* ***R17:*** *The system must store chat history for future reference.*
* ***R18:*** *The system must send notifications for new messages.*

### Priority: Medium

* *This use case improves student-faculty interaction, making communication faster and more efficient.*

### Preconditions:

* *The student and faculty* ***must be logged in****.*
* *The student* ***must have applied for or been approved for a project*** *with the faculty.*

### Postconditions:

* *The student and faculty* ***successfully exchange messages****.*
* *The system* ***stores the chat history*** *for future reference.*

### Actors:

* ***Primary Actor:*** *Student (initiates or replies to messages)*
* ***Secondary Actor:*** *Faculty (responds to student queries)*
* ***System Database:*** *Stores and retrieves chat messages*

### Extends:

* ***Faculty Approve/Reject Application (U5)*** *– Students can only chat with faculty after applying or being approved.*

### Flow of Events:

1. ***Basic Flow***
2. *The student logs into the system and navigates to* ***"Messages."***
3. *They select a faculty member from the list.*
4. *The system opens a chat window.*
5. *The student types a message and clicks* ***"Send."***
6. *The system:*

### Stores the message in the database.

* + ***Sends a notification*** *to the faculty.*

1. *The faculty reads the message and responds.*
2. *The student receives a* ***notification*** *of the reply.*

### Alternative Flow (Faculty-Initiated Chat)

1. *The faculty starts a conversation with a student.*
2. *The student can respond in real-time or later.*

### Exceptions

* + ***User is Offline:*** *If the faculty or student is offline, the message is stored and delivered when they log in.*
  + ***Network Issue:*** *If the internet connection is lost, the system displays → "Message not sent. Please check your internet connection."*
  + ***Database Failure:*** *If chat history cannot be retrieved, the system displays → "Unable to load messages. Please try again later."*

### Includes:

* + *None (chat is a standalone feature).*

### Notes/Issues:

* + *Should faculty be able to disable messaging outside working hours?*
  + *Should the chat support file sharing for project-related documents?*
  + *Consider adding a video call feature in the future.*

## Use Case #7: Schedule Meetings (U7)

***Author:*** *Purumandla Sahasra Reddy*

### Purpose:

This use case allows students and faculty to schedule project-related meetings within the system. It ensures effective communication and collaboration between students and faculty regarding project progress and discussions.

### Requirements Traceability:

* ***R19:*** *The system must allow students and faculty to schedule meetings.*
* ***R20:*** *The system must provide available time slots for scheduling.*
* ***R21:*** *The system must send notifications to both parties upon meeting confirmation.*

### Priority: Medium

* *This use case enhances project collaboration and ensures structured discussions between students and faculty.*

### Preconditions:

* *The student* ***must be logged in****.*
* *The student* ***must have an approved project with the faculty****.*
* *The faculty must have* ***available time slots*** *for scheduling meetings.*

### Postconditions:

* *A meeting is* ***successfully scheduled*** *in the system.*
* *Both the faculty and student* ***receive notifications*** *about the scheduled meeting.*
* *The meeting details are* ***saved in the system*** *for future reference.*

### Actors:

* ***Primary Actor:*** *Student (requests a meeting)*
* ***Secondary Actor:*** *Faculty (confirms or modifies meeting request)*
* ***System Database:*** *Stores meeting schedules and sends notifications*

### Extends:

* ***Chat with Faculty (U6)*** *– Meetings can be scheduled through chat discussions.*

### Flow of Events:

1. ***Basic Flow***
2. *The student logs into the system and navigates to* ***"Schedule Meeting."***
3. *The system displays* ***available faculty members*** *for meeting requests.*
4. *The student selects a faculty member and* ***chooses an available time slot****.*

### Alternative Flow (Faculty-Initiated Meeting)

1. *The faculty* ***requests a meeting*** *with a student.*
2. *The student* ***receives a notification*** *and confirms the time.*

### Exceptions

* + ***No Available Time Slots:*** *If no time slots are available, the system displays → "No available slots. Please check again later."*
  + ***Faculty Declines Request:*** *If the faculty rejects the meeting, the system notifies the student with a message: "Meeting request declined. Please select another time."*
  + ***Database Connection Issue:*** *If the system fails to save the meeting, it displays → "Unable to schedule meeting. Please try again later."*

### Includes:

* + *None (meeting scheduling is a standalone feature).*

### Notes/Issues:

* + *Should the system integrate with Google Calendar or Outlook for external scheduling?*
  + *Should students be able to schedule recurring meetings for ongoing projects?*
  + *Consider adding an auto-reminder feature before the meeting starts.*

## Use Case #8: Generate Reports (U8)

***Author:*** *Purumandla Sahasra Reddy*

### Purpose:

This use case allows faculty and administrators to generate reports related to project applications, approvals, and system activity. The reports help in tracking student participation, faculty engagement, and project progress.

### Requirements Traceability:

* ***R22:*** *The system must allow faculty and admins to generate reports.*
* ***R23:*** *Reports should include project details, student applications, and approvals.*
* ***R24:*** *Reports must be available for download in multiple formats (PDF, Excel).*

### Priority: Medium

* *This use case is important for* ***monitoring system usage, faculty activity, and student applications****.*

### Preconditions:

* *The faculty or admin* ***must be logged in****.*
* *The system must have* ***data related to applications, approvals, and projects****.*

### Postconditions:

* *A report is successfully* ***generated and available for download****.*

### The report provides insights into project applications and approvals.

***Actors:***

* ***Primary Actor:*** *Faculty/Admin (requests and generates reports)*
* ***System Database:*** *Retrieves data and formats the report*

### Extends:

* ***Faculty Approve/Reject Application (U5)*** *– Report includes approval/rejection data.*
* ***Apply for a Project (U4)*** *– Report includes student applications.*

### Flow of Events:

1. ***Basic Flow***
2. *The faculty or admin logs into the system and navigates to* ***"Reports."***
3. *The system provides options for different* ***report types*** *(e.g., project applications, approvals, student participation).*
4. *The user* ***selects the report type*** *and specifies a date range.*
5. *The system retrieves* ***relevant data from the database****.*
6. *The report is* ***generated and displayed on the screen****.*
7. *The user has the option to* ***download the report in PDF or Excel format****.*

### Alternative Flow (Filtered Reports)

1. *The user applies* ***filters*** *(e.g., specific faculty, project, or student).*
2. *The system* ***generates a customized report*** *based on the selected criteria.*

### Exceptions

* + ***No Data Available:*** *If no data matches the selected criteria, the system displays → "No records found for the selected time period."*
  + ***Report Generation Failure:*** *If an error occurs, the system displays → "Unable to generate report. Please try again later."*
  + ***Download Issue:*** *If the file download fails, the system displays → "Error downloading report. Please check your connection."*

### Includes:

* + *None (report generation is a standalone function).*

### Notes/Issues:

* + *Should reports be auto-generated at scheduled intervals (weekly/monthly)?*
  + *Should reports be emailed to faculty/admins instead of manual downloads?*
  + *Consider adding graphical representations (charts, graphs) for better insights.*

**4 Other Non-functional Requirements**

# Performance Requirements

The MU Connect system must meet specific performance requirements to ensure smooth operation for students, faculty, and administrators. The following standards will be maintained:

### Frontend Performance:

* + - * ***P1:*** *The system shall load the* ***dashboard within 2 seconds*** *after a successful login.*
      * ***P2:*** *The project list shall be displayed* ***within 3 seconds****, even when* ***1,000+ projects***

exist.

* + - * ***P3:*** *Chat messages shall be delivered with* ***less than 500ms latency*** *for real-time interaction.*
      * ***P4:*** *The user interface shall be optimized for* ***smooth performance on mobile and desktop browsers****.*

### Backend Performance:

* + - * ***P5:*** *Faculty actions (approve/reject applications) shall be* ***processed within 2 seconds****.*
      * ***P6:*** *Report generation for a* ***6-month period*** *should take* ***no longer than 5 seconds****.*
      * ***P7:*** *The backend must handle* ***500 concurrent users*** *without performance degradation.*
      * ***P8:*** *API responses shall be optimized to return within* ***1 second for standard queries****.*

### Database Performance:

* + - * ***P9:*** *The database should efficiently handle* ***100,000+ student and faculty records****.*
      * ***P10:*** *Indexing shall be implemented to improve* ***query performance*** *for project search and application retrieval.*
      * ***P11:*** *Database transactions shall be* ***ACID-compliant*** *to ensure data consistency.*

# Safety and Security Requirements

The MU Connect system must ensure data protection while preventing unauthorized access. The following security measures shall be implemented:

### Frontend Security:

* + - * ***S1:*** *User authentication must use* ***OAuth 2.0 / JWT tokens*** *to ensure session security.*
      * ***S2:*** *The frontend shall prevent* ***Cross-Site Scripting (XSS)*** *and* ***Cross-Site Request Forgery (CSRF)*** *attacks.*
      * ***S3:*** *Password inputs and sensitive fields shall be* ***masked*** *to prevent exposure.*

### Backend Security:

* + - * ***S4:*** *The system shall enforce* ***role-based access control (RBAC)*** *to restrict access based on user type (Student, Faculty, Admin).*
      * ***S5:*** *All passwords must be* ***hashed using bcrypt (min 12 salt rounds)*** *before storage.*
      * ***S6:*** *The backend must* ***log and monitor suspicious login attempts*** *and block brute- force attacks.*

### Database Security:

* + - * ***S7:*** *Data must be encrypted* ***at rest and in transit*** *using* ***AES-256 and TLS 1.3****.*
      * ***S8:*** *Student details (email, phone numbers) shall be* ***protected and not visible to unauthorized users****.*
      * ***S9:*** *The system must comply with* ***GDPR and FERPA*** *to ensure legal data protection compliance.*
      * ***S10: Daily database backups*** *must be maintained, and* ***disaster recovery measures***

must be in place.

# Software Quality Attributes

### Reliability

* + - * *The system shall maintain* ***99.9% uptime*** *with automatic* ***failover mechanisms****.*
      * *The backend should* ***recover from crashes automatically*** *and restart services if necessary.*
      * *A* ***daily database backup*** *must be scheduled to prevent data loss.*

### Adaptability

* + - * *The system shall support multiple authentication providers (Google SSO, university logins).*
      * *The UI must adjust dynamically to mobile and desktop views.*
      * *The backend shall be built using a microservices architecture to allow modular feature expansion.*
    1. **Maintainability**
       - *The frontend and backend code shall be well-documented to ensure easy future modifications.*
       - *API documentation shall be available for seamless integration with third-party services.*
       - *Version control using GitHub/GitLab shall be enforced for tracking changes.*
       - *Automated unit testing shall be implemented for critical system features.*

### Usability

* + - * *The system shall follow Material UI / Bootstrap design principles for an intuitive user experience.*
      * *A help section with FAQs shall be included to assist first-time users.*
      * *The chat system shall have typing indicators and message read receipts.*

**5 Other Requirements**

This section defines additional essential requirements that are not covered in previous sections but are relevant to the MU Connect system.

### Database Requirements

* + - *The system shall use a MONGODB for data storage.*
    - *Database queries shall execute in under 2 seconds under normal load.*
    - *Automated daily backups shall be scheduled to prevent data loss.*
    - *The system shall enforce ACID compliance for transaction reliability.*

### Security & Legal Compliance

* + - *The system must comply with FERPA (protecting student records) and GDPR (data privacy).*
    - *Personally identifiable information (PII) such as emails and academic records must be encrypted and accessible only to authorized users.*
    - *Users must accept the Terms of Service and Privacy Policy before accessing the platform.*
  1. ***System Scalability & Reusability***
     + *The software shall be modular to allow easy integration of new features*
     + *The backend shall use a RESTful API to support future mobile app development.*
     + *The system shall be containerized using Docker for scalability and efficient deployment.*

**Appendix A – Data Dictionary**

The tables here defines key variables, constants, inputs, and outputs used in the **MU Connect**

system.

* 1. ***Key Data Elements***

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Type | Description | Possible Values |
| User\_id | Integer (Auto) | Unique ID for each user | 1001, 1002, … |
| user\_role | Enum | Defines user type | Student, Faculty, Admin |
| email | String | User email for login | [user@example.com](mailto:user@example.com) |
| password\_hash | String (Hashed) | Securely stored password | Hashed Value |
| project\_id | Integer (Auto) | Unique ID for each  project | 5001, 5002, … |
| application\_id | Integer (Auto) | Unique ID for student applications | 8001, 8002, … |
| application\_status | Enum | Status of student’s | Pending, Accepted, |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | project application | Rejected |
| meeting\_id | Integer (Auto) | Unique ID for scheduled meetings | 7001, 7002, … |
| chat\_id | Integer (Auto) | Unique ID for each  chat message | 9001, 9002, … |
| created\_at | Timestamp | Time when a record was created | YYYY-MM-DD HH:MM:SS |

* 1. ***Important Constants***

|  |  |  |
| --- | --- | --- |
| Constant Name | Value | Description |
| MAX\_LOGIN\_ATTEMPTS | 5 | Maximum failed login attempts before lockout |
| SESSION\_TIMEOUT | 15 minutes | Auto logout after inactivity |
| MAX\_FILE\_SIZE | 5MB | Maximum upload size for resumes |
| CHAT\_HISTORY\_LIMIT | 90 days | Duration for storing chat history |

* 1. ***Inputs & Outputs***

|  |  |  |
| --- | --- | --- |
| Input Variable | Source | Purpose |
| email | User Input | Used for login & authentication |
| password | User Input | Secure user authentication |
| project\_id | System | Identifies a specific project |
| application\_status | System | Tracks student application progress |
| meeting\_time | User Input | Schedule a meeting |

|  |  |  |
| --- | --- | --- |
| Output Variable | Destination | Purpose |
| login\_success | UI Display | Confirms successful login |
| dashboard\_data | UI Display | Shows student/faculty dashboard |
| application\_status | UI Display | Displays student application updates |
| chat\_message | UI Display | Displays chat messages |

* 1. *System States*

|  |  |  |
| --- | --- | --- |
| State Variable | Possible States | Description |
| user\_status | Active, Inactive | Defines if a user can access the system |
| application\_status | Pending, Accepted, Rejected | Tracks student applications |
| project\_status | Open, Closed | Indicates project availability |
| meeting\_status | Scheduled, Cancelled | Status of student-faculty meetings |
| chat\_status | Unread, Read | Message read status |

**Appendix B - Group Log**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Meetings* | *Date* | *Duration* | *Topics discussed* | | | |
| *1* | *Jan 25 ,2025* | *1 hr* | *Project finalization* | | | |
| *2* | *Feb 2 ,2025* | *2 hrs* | *Discussed what are core features for the system(frontend,backend,DB,..’)* | | | |
| *3* | *Feb 5,2025* | *1hr* | *Finalizing SOW* | | | |
| *4* | *Feb 18,2025* | *1hr* | *Made ER system* | *diagram* | *for* | *the* |
| *5* | *Mar 6, 2025* | *1hr* | *Use case project* | *and UML* | *for* | *the* |
| *6* | *Mar 10,2025* | *3hrs* | *For SRS finalization* | | | |