# Software Requirements Specification

## for

**Mobile application - Social Networking of MU students and alumni**

**MU Connect**

## Prepared by

**Group Name: Prime Developers**


| | | |
|---|---|---|
| Niyathi Vasasali | SE23LCSE005 | se23lcse005@mahindrauniversity.edu.in |
| Varshini Babers | SE22UCSE320 | se22ucse320@mahindrauniversity.edu.in |
| Sadhika Deva | SE22UCSE230 | Se22ucse230@mahindrauniversity.edu.in |
| Divya Moduga | SE22UCSE084 | Se23ucse084@mahindrauniversity.edu.in |
| Sathvick Hitesh | SE22UCSE111 | Se22ucse111@mahindrauniversity.edu.in |


| | |
|---|---|
| **Instructor:** | ***Dr. Vijay Rao Duddu*** |
| **Course:** | **Software Engineering** |
| **Lab Section:** | ***CS3 and CS4*** |
| **Teaching Assistant:** | ***Professor Nartakannai K*** |
| **Date:** | **02/03/2025** |

# Contents



# Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
|---|---|---|---|
| 0.0 | Sadhika Deva, Niyathi Vasasali, Divya Moduga | Basic website layout with description of the login portal and some specific pages (profiles, connect, post, home). | 10/03/25 |

# 1 Introduction

This document outlines the software requirements for the **Mahindra University Social Networking App (MU Connect)**. It serves as a blueprint for the development of a mobile application that facilitates professional networking, mentorship, and communication among MU students and alumni.

In this section, readers will find an overview of the project's objectives, purpose, and key functionalities. The document ensures that developers, testers, and stakeholders have a shared understanding of the application's requirements.

## 1.1 Document Purpose

The purpose of this document is to define the Software Requirements Specification (SRS) for the MU Connect App, a mobile application designed for social networking among Mahindra University (MU) students and alumni. This document outlines the detailed functional and non-functional requirements, ensuring clarity in development, implementation, and deployment. It serves as a reference for developers, testers, and stakeholders to align on the system’s objectives and expectations.
This SRS focuses on the core system rather than individual subsystems, covering both backend and frontend components of the mobile application.

## 1.2 Product Scope

The MU Connect App is designed to serve as an exclusive social networking platform for Mahindra University (MU) students and alumni. It facilitates connections, networking, and the sharing of academic and professional opportunities within the MU community. Key features of the app include job postings, event notifications, mentorship matching, and discussion forums, all tailored to enhance engagement and collaboration among users.

The application aims to create a professional and social networking platform for Mahindra University (MU) students and alumni. The system will provide features such as:

- User profiles with academic and professional details
- Direct messaging and group chats
- Job postings and mentorship opportunities
- University event updates
- Discussion forums for academic and career guidance

The platform will be developed for both Android and iOS, ensuring accessibility for all users.

## 1.3 Intended Audience and Document Overview

This document is intended for:

- **Developers:** To implement the system based on defined requirements.
- **Project Managers:** To monitor project progress.
- **Testers:** To validate functional and non-functional requirements.
- **University Stakeholders:** To ensure alignment with academic needs.

## 1.4 Definitions, Acronyms and Abbreviations

| Acronym | Definition |
|---|---|
| **MU** | Mahindra University |
| **SRS** | Software Requirements Specification |
| **API** | Application Programming Interface |
| **AI** | Artificial Intelligence |
| **UI/UX** | User Interface / User Experience |

## 1.5 Document Conventions

- This document follows the IEEE SRS template.
- Standard font: Times New Roman, Size: 12pt.
- Section titles in bold.

## 1.6 References and Acknowledgments

- Mahindra University website and student database policies
- Standard mobile application development guidelines
- Security and privacy compliance documents

# 2 Overall Description

## 2.1 Product Overview

The **MU Connect App** is designed with a **modular architecture** to ensure **scalability, maintainability, and flexibility**. It follows the **Model-View-Controller (MVC) architecture**, which separates concerns between the **user interface, business logic, and data management**.

The mobile application will function as a dedicated platform for MU students and alumni to connect, communicate, and collaborate. It will integrate authentication mechanisms, real-time messaging, and discussion forums.

***Key System Components:***

- **User Authentication Module:** Manages registration, login, and verification using a **college email-based authentication system** for secure access.
- **Profile Management Module:** Allows users to create, edit, and view profiles, including **academic background, career details, and interests**.
- **Messaging System:** Enables **real-time communication** through direct messages and group chats.
- **Event Management System:** Facilitates event creation, participation, and notifications for **college-related events**.
- **Search and Networking Module:** Implements **advanced search filters** to help users find students and alumni based on **location, interests, and profession**.
- **Database Management:** Uses a **relational database** to securely store user information, messages, and events.

## ~~2.2~~ Product Functionality

The core functionalities of the system include:

- **User Registration & Authentication:** Secure sign-up and login.
- **Profile Management:** Users can create and update their profiles.
- **Messaging System:** Direct and group messaging.
- **Job Board & Mentorship:** Posting and applying for job opportunities.
- **Event Management:** Displaying university events and allowing RSVPs.
- **Discussion Forums:** Themed conversations around academics and careers.

## 2.3 Design and Implementation Constraints

The development of the **MU Connect App** is subject to the following constraints, which limit the choices available to developers in terms of hardware, software, design, and implementation methodologies.

***1. Hardware Limitations***

- The application must be optimized for **Android and iOS devices** with varying hardware capabilities, including different screen sizes and processing power.
- **Memory constraints** must be considered, ensuring the app runs efficiently without excessive resource consumption.

***2. Software and Technology Constraints***

- The system must be developed using the **COMET (Collaborative Object Modeling and Enterprise Transformation) method** for software design. This method ensures a structured and modular approach to development.
- **UML (Unified Modeling Language) must be used** for system design and documentation, including class diagrams, sequence diagrams, and use case models.
- The application backend must follow a **RESTful API architecture** to facilitate seamless communication between the frontend and backend.
- A **relational database (e.g., PostgreSQL or MySQL)** must be used for structured data storage, ensuring efficient query handling and scalability.

***3. Security and Authentication Constraints***

- User authentication must be implemented using a **university email-based login system** with OAuth 2.0 or a similar secure authentication protocol.
- The system must comply with **data privacy regulations** to protect student and alumni information.

***4. Integration Constraints***

- The app must integrate with **third-party APIs** for functionalities such as email notifications and push alerts.
- Compatibility with the **university's authentication system** is mandatory to ensure secure user verification.

By adhering to these constraints, the **MU Connect App** will maintain a high standard of **performance, security, and scalability** while aligning with best practices in software design and implementation.

## 2.4 Assumptions and Dependencies

- Access to development tools such as Android Studio and Xcode.
- Internet connectivity for cloud-based features.
- Active participation from students and alumni.

*< assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. list of some major assumptions that might significantly affect our design.>*

# 3 Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

The **MU Connect App** features an intuitive and user-friendly interface designed for seamless navigation. The UI follows **modern UI/UX principles**, ensuring accessibility and efficiency for both students and alumni.

***Key UI Components:***

1. **Login & Authentication Screen**
    - Users log in using their **Mahindra University email**.
    - Secure authentication through **OAuth 2.0 or email verification**.
2. **Home Dashboard**
    - Displays **latest posts, job listings, and event notifications**.
    - Provides **quick access to networking and messaging features**.
3. **Profile Page**
    - Users can create and customize their profiles, including **academic background, career details, and interests**.
4. **Messaging Interface**
    - Supports **real-time direct messaging** and **group chats**.
    - Uses a **clean chat interface** with read receipts and notifications.

5. **Search & Networking**
    - Allows users to **search for students, alumni, and faculty** based on criteria like department, profession, and location.
6. **Event Management Screen**
    - Enables users to **create, browse, and join university-related events**.
7. **Settings & Notifications**
    - Users can **customize notifications, privacy settings, and app preferences**.

***User Interaction:***

- The interface is designed for **touch interaction**, featuring **intuitive gestures (swipes, taps, and long-press actions)**.
- **Bottom navigation bar** for quick access to key sections.
- **Dark mode & accessibility settings** for an inclusive experience.

*A graphical representation of the UI can be included here, such as wireframes or mockups of the home screen, profile page, and messaging interface.*

### 3.1.2 Hardware Interfaces

***Supported Device Types:***

1. **Smartphones (Android & iOS)**
    - The app will run on **Android (version X and above) and iOS (version X and above)**.
    - Optimized for **touch-based interaction** with responsive design.
2. **Tablets**
    - The app will support larger screen sizes with an **adaptive layout**.
    - Enhanced UI for **better multitasking and navigation**.
3. **Push Notification System**
    - Integrates with **mobile device notification services** (Firebase Cloud Messaging for Android, APNs for iOS).
4. **Camera & Media Access**

- Users may upload profile pictures, event images, or shared media through the **device camera and gallery**.

5. **Internet Connectivity**

   - The app requires an **active Wi-Fi or mobile data connection** for real-time features like messaging and event updates.

The system will ensure efficient **hardware-software interaction** to provide a smooth user experience across different devices.

### 3.1.3 Software Interfaces

The **MU Connect App** interacts with several software components to provide a seamless social networking experience for students and alumni. These interfaces facilitate communication between the mobile application and external services.

***1. Backend Server Interface***

- The mobile application communicates with the backend using **RESTful APIs**.
- The backend handles **user authentication, data storage, messaging, and event management**.
- All requests and responses follow a **JSON format** for structured data exchange.

***2. Authentication System***

- The app integrates with **Mahindra University's authentication system** to ensure that only verified students and alumni can log in.
- Uses **OAuth 2.0 or similar protocols** for secure login.

***3. Database Interface***

- The system interacts with a **relational database (PostgreSQL/MySQL)** for storing **user profiles, messages, events, and notifications**.
- The backend queries and updates the database as needed via **SQL queries and ORM (Object-Relational Mapping) tools**.

***4. Push Notification Services***

- The app integrates with **Firebase Cloud Messaging (FCM) for Android** and **Apple Push Notification Service (APNs) for iOS**.
- Notifications are triggered for **new messages, event reminders, and networking requests**.

***5. Third-Party APIs***

- The app may integrate with **external APIs** for additional features like:
    - **Email services (SendGrid, AWS SES, or similar)** for verification and updates.

These software interfaces ensure that the **MU Connect App** functions efficiently, maintaining secure, scalable, and real-time communication between its components.

## 3.2 Functional Requirements

- **F1:** The system shall allow users to create and manage profiles.
- **F2:** The system shall enable real-time messaging.
- **F3:** The system shall allow users to post and apply for job opportunities.
- **F4:** The system shall display upcoming university events.
- **F5:** The system shall allow discussions through forums.

## 3.3 Use Case Model

This section provides an overview of the primary use cases for the **MU Connect App**. Each use case is realized through interactions between modules, represented by **sequence diagrams and activity diagrams**. The realization is broken down into:

- **High-Level Interactions**: How modules communicate.
- **Low-Level Interactions**: How individual classes collaborate to implement each use case.

### 3.3.1 Use Case #1: User Registration & Authentication (U1)

**Author:** [Your Name]
**Purpose:**

- Allows a new user to register or log in using their Mahindra University email.
- Ensures secure access control to the application.

**Requirements Traceability:**

- R1: The system must support email-based authentication.
- R2: Users must have a university-verified email to register.

**Priority:** High

**Preconditions:**

- The user must have a valid Mahindra University email.

**Postconditions:**

- If successful, the user gains access to their profile and app features.
- If authentication fails, an error message is displayed.

**Actors:**

- **Primary:** User
- **Secondary:** Authentication Service, Database

**Extends:** None

**Flow of Events:**

1. **Basic Flow:**
    - User enters credentials and submits registration/login request.
    - AuthenticationService validates credentials.
    - If valid, user profile is created/retrieved from UserRepository.
    - System responds with success or failure.
2. **Alternative Flow:**
    - If the email is already registered, prompt the user to log in.
    - If the email is invalid, request re-entry.
3. **Exceptions:**
    - Incorrect password entered multiple times → Lock account for security.
    - Database connection failure → Show an error message.

**Includes:** None

**Notes/Issues:**

- Consider integrating **OAuth 2.0** for enhanced security.
- Multi-factor authentication (MFA) may be added in future updates.

### 3.3.2 Use Case #2: Posting Content on Feed (U2)

**Author:** [Your Name]
**Purpose:**

- Enables users to create and share posts on the app's feed.

**Requirements Traceability:**

- R3: Users must be able to submit text and media posts.
- R4: Posts must be visible to followers and network connections.

**Priority:** High

**Preconditions:**

- User must be logged in.

**Postconditions:**

- The post is successfully saved and displayed in the feed.

**Actors:**

- **Primary:** User
- **Secondary:** PostService, Database

**Extends:** None

**Flow of Events:**

1. **Basic Flow:**
    - User enters text/media and submits a new post.
    - PostService processes and saves the post via PostRepository.
    - Post is displayed in the feed.
2. **Alternative Flow:**
    - If the post contains inappropriate content, reject submission.
3. **Exceptions:**
    - Database failure → Post submission fails with an error message.
    - Internet connectivity issues → Post submission is retried or saved offline.

**Includes:** None

**Notes/Issues:**

- Implement **content moderation** to detect offensive content.

- Future updates may support **post editing** after submission.

### 3.3.3 Use Case #3: Sending & Receiving Messages (U3)

**Author:** [Your Name]

***Purpose:***

- Allows users to send and receive private messages in real-time.
- Ensures messages are stored securely and delivered efficiently.

***Requirements Traceability:***

- R5: Users must be able to send direct messages to other users.
- R6: Messages must be stored in a database and retrievable.
- R7: Users should receive real-time notifications for new messages.

***Priority: High***

***Preconditions:***

- Both sender and recipient must be registered users.
- The sender must be logged into the application.

***Postconditions:***

- The recipient receives the message instantly.
- The message is stored in the database.
- The recipient gets a notification.

***Actors:***

- **Primary:** User (Sender & Receiver)
- **Secondary:** MessageService, MessageRepository, NotificationService, Database

***Extends: None***

***Flow of Events:***

1. **Basic Flow:**

- User selects a recipient and types a message.
- MessageService stores the message via MessageRepository.
- NotificationService alerts the recipient in real-time.
- The recipient opens the chat to view the message.

2. **Alternative Flow:**
   - If the recipient is offline, they receive the message upon logging in.
   - Users can delete messages from their chat history (but data remains stored in the database).
3. **Exceptions:**
   - Network failure → MessageService retries sending the message.
   - Database connection issue → The message is temporarily stored locally.

***Includes: None***

***Notes/Issues:***

- Consider implementing **end-to-end encryption** for privacy.
- Future updates may support **voice and video messages**.

# 4 Other Non-functional Requirements

## 4.1 Performance Requirements

**P1. Authentication Speed:**

- User login (OAuth 2.0 with JWT) must complete within **2 seconds** under normal network conditions.
- Multi-Factor Authentication (MFA) verification must not exceed **5 seconds**.

**P2. Message Delivery Time:**

- Private messages should be delivered in real-time with a latency of **less than 500ms**.
- Read receipts should update within **1 second** after the recipient views the message.

**P3. Feed & Post Loading Speed:**

- The home feed should load **within 2 seconds** when a user opens the app.
- New posts should be displayed within **1 second** of submission.

**P4. Encryption & Security Processing:**

- End-to-End Encryption (E2EE) for messages must not add more than **200ms** overhead.
- SSL/TLS encryption should ensure secure connections with **negligible impact** on performance.

**P5. Role-Based Access & Authorization:**

- Access control checks (RBAC) must be processed in **under 300ms** for any action.
- Token validation and user authorization should not take longer than **500ms**.

**P6. System Scalability & Load Handling:**

- The system should support **10,000+ concurrent users** without degraded performance.
- Must handle **500+ login requests per second** under peak conditions.
- The database should support **1 million+ stored messages** while maintaining fast retrieval times.

**P7. Session Timeout & Auto Logout:**

- Inactive user sessions must be terminated after **15 minutes** of inactivity.

- Forced session revocation (e.g., due to suspicious activity) must be completed in **under 2 seconds**.

**P8. Admin Actions & Moderation:**

- Admin actions (e.g., banning a user, removing a post) should take effect within **3 seconds**.
- Reported content must be flagged and restricted in **under 1 second**.

## 4.2 Safety and Security Requirements

1. **User Identity & Authentication**
    - The system must enforce **OAuth 2.0 & JWT authentication** for secure user sessions.
    - Multi-Factor Authentication (MFA) must be required for sensitive actions (e.g., password reset, admin access).
    - Only verified **Mahindra University email accounts** can be used for registration.
2. **Data Protection & Privacy**
    - All messages and private user data must be **end-to-end encrypted (E2EE)**.
    - SSL/TLS encryption must be enforced for all network communications.
    - Passwords must be **hashed using bcrypt** before storage.
    - Users must have granular **privacy settings** to control post visibility and profile access.
3. **Access Control & Role-Based Security**
    - **Role-Based Access Control (RBAC)** must restrict data access based on user roles:
        - **Students:** Post on feed, send messages, connect, join events.
        - **Alumni:** Same as students + create alumni-only events, mentor students.
        - **Admins:** Moderate content, manage users, configure system settings.
    - Unauthorized access attempts must trigger **automatic account lockout** after 5 failed login attempts.
4. **Mobile App & API Security**
    - All API calls between the mobile app and backend must use **HTTPS with TLS 1.2+**.

5. **Safety & Moderation Features**
    - Users can **report inappropriate content or harassment**, triggering admin review.
    - An **AI-based moderation system** will flag offensive content for review.
    - Admins can **delete flagged posts and suspend accounts** violating community guidelines.
6. **Compliance with Security Standards**
    - The system must comply with relevant privacy laws, including:
        - **General Data Protection Regulation (GDPR)** for data privacy.
    - Security logs must be maintained for **audit trails of login attempts, role changes, and content moderation actions**.

## 4.3 Software Quality Attributes

**Reliability**

- The system must maintain **99.9% uptime** to ensure continuous availability for users.
- A **load-balanced architecture** will be implemented to handle traffic spikes.
- Automated **error logging and crash recovery** will be in place to detect and resolve failures.
- **Database replication** will be used to prevent data loss in case of system crashes.

**Usability**

- The user interface will follow **Material Design** principles for intuitive navigation.
- The mobile app must have an **average response time of ≤ 1 second** for user actions.
- **Dark mode** and **adaptive UI scaling** will be available for accessibility.
- A tutorial/onboarding process will guide new users on core functionalities.

**Maintainability**

- The system will follow **modular design principles**, enabling easy updates and feature expansions.
- Code must be documented using **Doxygen**, ensuring long-term maintainability.
- A **CI/CD pipeline** will be used for automated testing and deployment.

- Dependencies will be version-controlled to prevent conflicts.

**Security & Data Integrity**

- **OAuth 2.0 authentication** will be used to prevent unauthorized access.
- **Role-Based Access Control (RBAC)** ensures different user privileges.
- **Data encryption (SSL/TLS + E2EE)** will be enforced for messages and sensitive data.
- **Regular security audits** will be conducted to identify vulnerabilities.

**Scalability**

- The backend will be built using **a microservices architecture** to scale specific components independently.
- **Horizontal scaling** will allow multiple instances of services to be deployed dynamically.
- **Caching mechanisms (Redis, CDN)** will improve response times for frequently accessed content.

**Interoperability**

- The app will expose **RESTful APIs** to allow integration with third-party services (e.g., LinkedIn for alumni connections).
- A **standardized data exchange format (JSON)** will ensure smooth communication between components.

# 5 Other Requirements

We require the database of the students and alumni of Mahindra University. (But only if we need a lot of people on our app – our TA told us that having a functional app with around five people id also enough.)