# Software Requirements Specification

**for**

# MUVerse - Social Networking for Mahindra University Students

**Version: 1.0**

**Prepared by**

**Group Name:** ***SPACS***


| | |
|---|---|
| **D. Safdar Hussain** | **SE22UCSE085** |
| **P. Chaitra** | **SE22UCSE193** |
| **K. Pavithra** | **SE22UCSE136** |
| **K. Pranvith** | **SE22UCSE133** |
| **B. Shivani** | **SE22UCSE048** |
| **Alekhys Raavi** | **SE22UCSE022** |


**Instructor:** ***Vijay Rao Sir***

**Course:** ***Software Engineering***

**Lab Section:** ***IT- Block 2 (GF)***

**Teaching Assistant:** ***Surya Phani Teja Sir***

**Date:** ***April 7th***

## Table of Contents


**1 INTRODUCTION ..... 4**
1.1 Purpose ..... 4
1.2 Scope ..... 4
1.3 Definitions, Acronyms, and Abbreviations ..... 4
1.4 References ..... 4

**2 USE CASE VIEW ..... 4**
2.1 Use Case ..... 4

**3 DESIGN OVERVIEW ..... 4**
3.1 Design Goals and Constraints ..... 5
3.2 Design Assumptions ..... 5
3.3 Significant Design Packages ..... 5
3.4 Dependent External Interfaces ..... 5
3.5 Implemented Application External Interfaces ..... 5

**4 LOGICAL VIEW ..... 5**
4.1 Design Model ..... 6
4.2 Use Case Realization ..... 6

**5 DATA VIEW ..... 6**
5.1 Domain Model ..... 6
5.2 D ata Model (persistent data view) ..... 6
*5.2.1 Data Dictionary ..... 6*

**6 EXCEPTION HANDLING ..... 6**

**7 CONFIGURABLE PARAMETERS ..... 6**

**8 QUALITY OF SERVICE ..... 7**
8.1 Availability ..... 7
8.2 Security and Authorization ..... 7
8.3 Load and Performance Implications ..... 7
8.4 Monitoring and Control ..... 7

# 1 Introduction

MUVerse is a dedicated mobile social networking application tailored for Mahindra University students. The platform provides seamless communication, collaboration, and social engagement within the university community by enabling real-time messaging, media sharing, group discussions, and event coordination.

## 1.1 Purpose

This Software Design Specification (SDS) outlines the detailed architectural and design elements of MUVerse, a mobile social networking platform for Mahindra University students. It serves as a guide for developers, testers, and stakeholders by detailing system components, data flow, use case realization, and quality considerations. It ensures that implementation aligns with the Software Requirements Specification (SRS) and overall project objectives.

## 1.2 Scope

MUVerse is a university-exclusive social networking app offering secure, real-time communication among students. This document focuses on the system's design, including functional breakdowns, class structures, communication protocols, user interaction models, and backend architecture. It covers modules like:

- Chat & Messaging (Implemented)
- Media Upload (Implemented)
- Events & News Feed (Planned)
- User Authentication (via MU Student API + Firebase)

## 1.3 Definitions, Acronyms, and Abbreviations

| Term | Definition |
|---|---|
| MU | Mahindra University |
| SRS | Software Requirements Specification |
| SDS | Software Design Specification |
| UI/UX | User Interface / User Experience |
| API | Application Programming Interface |
| MERN | MongoDB, Express.js, React, Node.js |
| Firebase | Google platform for authentication & realtime DB |
| Socket.IO | Library for real-time communication |
| Cloudinary | Media storage and transformation service |
| TailwindCSS | Utility-first CSS framework |
| DaisyUI | Tailwind component library |
| Vite | Fast frontend build tool |

### 1.4 References

- Firebase Docs: https://firebase.google.com/docs
- React Native Docs: https:/ /reactnative.dev/docs
- TailwindCSS Docs: https://tailwindcss.com/docs
- DaisyUI Docs: https://daisyui.com/docs
- Socket.IO Docs: https://socket.io/docs
- Cloudinary Docs: https://cloudinary.com/documentation

## 2 Use Case View

### 2.1 Use Case Summary

The following core use cases form the backbone of MUVerse's functionality. They are derived from the SRS and current implementation status:

| Use Case ID | Name | Status | Priority |
|---|---|---|---|
| U1 | Send Instant Message | Implemented | High |
| U2 | Create Group Chat | Implemented | Medium |
| U3 | Upload and Share Media | Implemented | High |
| U4 | Post Event & RSVP | Pending | Medium |
| U5 | View University News Feed | Pending | Low |

**Use Case Details**

**U1: Send Instant Message**

Actors: Student (Sender), Student (Recipient)
Description: Enables students to send real-time text messages via WebSocket.
Flow:
1. Student selects a contact.
2. Types and sends a message.
3. Message is delivered in real time with toast notification.
4. Stored in Firebase.

**U2: Create Group Chat**
Actors: Student (Creator), Students (Members)
Description: Allows students to create groups with chat functionality.
Flow:
1. Student selects 'Create Group'.
2. Adds members, names group.
3. Members are notified and group appears in sidebar.

**U3: Upload and Share Media**
Actors: Student
Description: Uploads images using Cloudinary and sends in chat.
Flow:
1. User selects media in chat input.
2. Image is uploaded to Cloudinary.
3. Link is rendered inline in chat.

**U4: Post Event & RSVP**
Actors: Student
Description: Post university events and allow others to RSVP.
Status: Planned.

**U5: View University News Feed**
Actors: Student
Description: View a chronological feed of announcements.
Status: Planned.

# 3 Design Overview

## 3.1 Design Goals and Constraints

- Ensure real-time, secure, and scalable communication among Mahindra University students.
- Mobile-first experience with React Native.
- Use Tailwind CSS + DaisyUI for UI.
- Modular, component-based architecture.
- Constraints include open-source tools, Firebase usage, deployment on Render, and strict academic deadlines.

## 3.2 Design Assumptions

- Only verified MU students use the system.
- Firebase free tier is sufficient for expected scale.
- MU will provide student verification API.
- App will be used on mobile devices with the internet.
- No admin/mod panel required initially.

## 3.3 Significant Design Packages

- Auth: Handles Firebase + MU API login.
- Chat: WebSocket messaging + Firebase backup.
- Media Upload: Cloudinary image integration.
- UI Components: Tailwind + DaisyUI reusable components.
- Pages: LoginPage, HomePage, etc.
- State: Zustand stores.
- Backend API: Node + Express REST endpoints.

## 3.4 Dependent External Interfaces

| External Interface | Used By | Purpose |
|---|---|---|
| Firebase Auth | Frontend | Handles authentication |
| Firebase DB | Chat | Stores persistent messages |
| MU Student API | Backend | Verifies student credentials |
| Cloudinary | Media Upload | Stores uploaded images |
| Socket.IO | Chat | Real-time message transport |
| Render | Deployment | Hosts Frontend/Backend |

### 3.5 Implemented Application External Interfaces (and SOA web services)

| Interface | Module | Description |
|---|---|---|
| /api/auth/login | Backend | Handles login and user verification |
| /api/chat/send | Chat | Sends messages with WebSocket + backup |
| WebSocket/chat | Socket.io | Live communication |
| /upload/image | Media Upload | Uploads media to Cloudinary |

# 4 Logical View

## 4.1 Design Model

The system is modularly designed with each component responsible for specific functionality. The main modules include:
- Auth Module: Handles user login, session storage, and profile data.
- Chat Module: Manages chat containers, individual messages, and group chat logic.
- Media Module: Compresses, uploads, and previews images via Cloudinary.
- UI Layer: Reusable components including Navbar, Sidebar, Skeleton Loaders, etc.
- State Management: Zustand stores (useAuthStore, useChatStore, useThemeStore).
The frontend follows a component-based hierarchy using React Native. Pages like HomePage, LoginPage, and ProfilePage integrate components and business logic. The backend Node.js server uses Express routers and controllers to manage RESTful APIs

## 4.2 Use Case Realization

This section describes how core use cases are implemented across the system's modules:

- U1 (Send Instant Message): The frontend emits a `message` event through Socket.IO, which is handled by the backend to broadcast to recipients and store in Firebase.
- U2 (Create Group Chat): The user selects contacts and names the group. Backend stores the group metadata, and members receive a socket-based event update.
- U3 (Upload Media): User selects image, client compresses it, sends it to /upload/image, and displays Cloudinary-hosted URL inline.

# 5 Data View

## 5.1 Domain Model

The domain model for MUVerse includes key entities such as ‘User’, ‘Message’, ‘Group’, ‘Event’, and ‘Post’. Relationships include:
- A ‘User’ can send many ‘Messages’ and belong to multiple ‘Groups’.
- A ‘Group’ contains multiple ‘Users’ and many ‘Messages’.
- An ‘Event’ is posted by a ‘User’ and includes RSVP responses from other users.
- A ‘Post’ represents items in the university news feed, linked to the author ‘User’.

## 5.2 Data Model (persistent data view)

Data is stored in MongoDB and Firebase. MongoDB holds structured records (users, groups, events), while Firebase manages real-time chat messages.

### 5.2.1 Data Dictionary

| Field Name | Type | Description |
|---|---|---|
| userId | String | Unique MU student ID |
| name | String | User's full name |
| message | String | Chat message text (max 500 chars) |
| timestamp | DateTime | Message or post timestamp |
| groupId | String | ID of the group chat |
| imageUrl | String | Cloudinary URL for shared image |
| eventId | String | Event identifier in DB |
| rsvpList | Array | List of users who RSVP'd to an event |

# 6 Exception Handling

Invalid Login: Thrown when student credentials don’t match; handled with toast error and retry prompt.
Message Send Failure: If WebSocket fails or Firebase is down, message is cached and retry is offered.
Image Upload Error: Cloudinary issues trigger error toast; upload is cancelled.
Network Errors: Displayed as toast notifications; app retries fetches periodically.
All exceptions are logged in browser/dev console and backend server logs (Node.js).

# 7 Configurable Parameters

| Parametername | Usage | Dynamic? |
|---|---|---|
| FIREBASE_URL | Firebase backend config | No |
| MAX_GROUP_MEMBERS | Limit number of users in group (100) | Yes |
| MEDIA_MAX_SIZE_MB | Limit on upload file size (50MB) | Yes |
| THEME | Default UI theme (light/dark) | Yes |
| WS_RETRY_INTERVAL | WebSocket reconnect time (5s) | Yes |

# 8 Quality of Service

## 8.1 Availability

MUVerse is designed to provide high availability, ensuring minimal downtime even during peak usage. Firebase's real-time database and Render's uptime guarantee help maintain continuity.

- System shall maintain 99.9% uptime (i.e., <43 minutes/month of downtime).
- Failover strategy includes retry mechanisms and local cache fallbacks.

## 8.2 Security and Authorization

- All user data is encrypted using AES-256 both at rest and in transit.
- TLS 1.3 is used for all WebSocket and HTTP connections.
- Firebase Authentication is integrated with MU Student API for secure login.
- Session tokens are time-bound and stored securely.
- No sensitive data is exposed in URLs or local storage

## 8.3 Load and Performance Implications

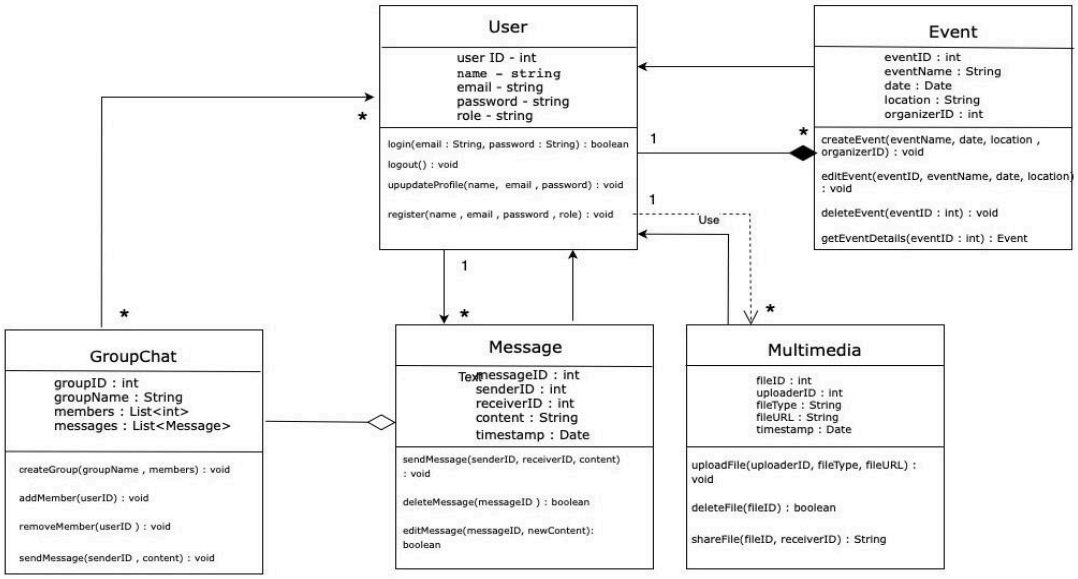
- System designed to handle 10,000 concurrent users with Firebase autoscaling.
- Messages are delivered in <2 seconds under normal network conditions.
- Media upload (≤50MB) completes in ≤10 seconds on 10 Mbps connections.
- All frontend pages load in ≤3 seconds using Vite for fast bundling

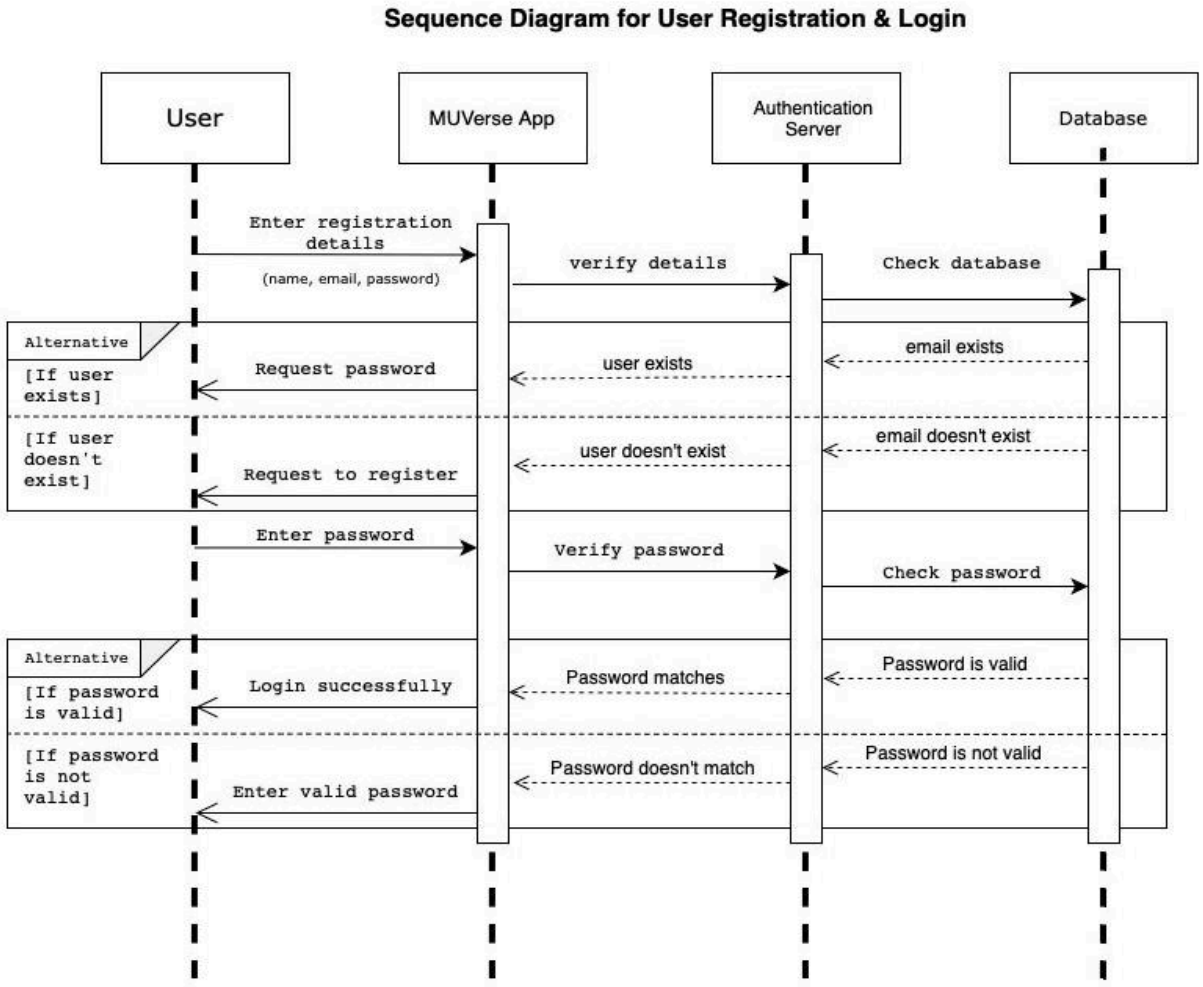
## 8.4 Monitoring and Control

- WebSocket server logs user connectivity and failures.
- Firebase Analytics monitors message delivery stats.
- Backend logs exceptions and endpoint failures with timestamps.
- Admin dashboard (planned) will allow for manual event/news moderation in future versions.

## Appendix A: UML Diagrams

This appendix contains the key UML diagrams for MUVerse's design.



## Activity Diagram for Send Instant Message