**Software Design Specifications**

**for**

**Mobile application for Social Networking of MU Students**

**Prepared by**

**Group Name: Team 1**

|  |  |  |
| --- | --- | --- |
| **Anunay Korupolu** | **se22uari019** |  |
| **Chatur Varma Inampudi** | **se22uari034** |  |
| **Krushik Teja Thati** | **se22uari078** |  |
| **Madala Venkata Bhargav** | **se22uari086** |  |

|  |  |
| --- | --- |
| **Instructor:** | **Prof. Avinash Arun Chauhan** |
| **Course:** | **Software Engineering** |
| **Lab Section:** | ***AI*** |
| **Teaching Assistant:** | ***Nartkannai K*** |
| **Date:** | **05/04/2025** |

**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**

**1.1** **Purpose**

This document serves to define and elaborate on the software design for the MU-Connect mobile application. The application aims to provide a seamless user experience for sharing blogs, messaging, managing followers, and viewing notifications. This specification will guide developers, designers, testers, and other stakeholders in building, maintaining, and scaling the application.

**1.2** **Scope**

This document covers both the frontend (Flutter-based UI) and backend (Supabase) components of the application. MU-Connect is a mobile-first social networking app for students. It includes modules for:

* Authentication
* Chat and group chat
* Blog post creation and interaction
* Notifications
* Profile management

The backend uses **Supabase** (authentication, real-time sockets, API, database).

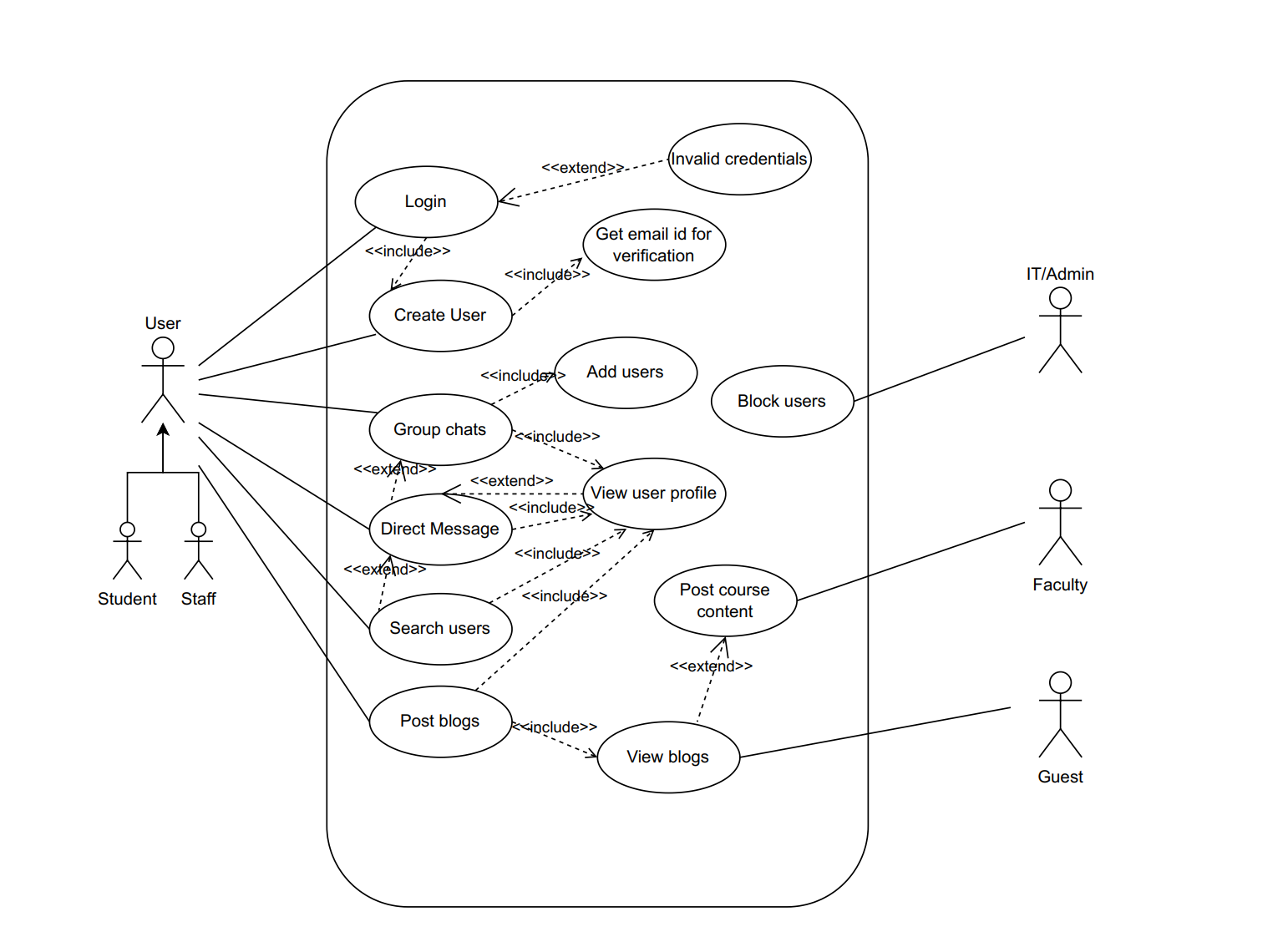
**1.3** **Definitions, Acronyms, and Abbreviations**

* **UI**: User Interface
* **API**: Application Programming Interface
* **SDS**: Software Design Specification
* **DB**: Database
* **Supabase**: A backend-as-a-service platform used for data storage and APIs
* **JWT**: JSON Web Token

**1.4** **References**

* Flutter Documentation: [https://flutter.dev](https://flutter.dev/)
* Supabase Documentation: <https://supabase.com/docs>
* PostgreSQL Documentation: <https://www.postgresql.org/docs/>
* TypeScript Documentation: <https://www.typescriptlang.org/docs/>

**2** **Use Case View**



**2.1** **Use Case**

The application supports the following use cases:

* Register/Login with Supabase authentication
* View and edit user profiles
* Follow or unfollow other users
* Access followers/following lists
* View, create, or delete blog posts
* View real-time notifications
* Send and receive messages using Supabase

**Actors**: Registered User, Guest User, Supabase DB

**Relationships**: Use cases are interconnected through authentication and state management layers. For example, profile editing requires a logged-in state, and messaging requires valid user tokens.

1. **Design Overview**

**3.1** **Design Goals and Constraints**

* **Modularity**: Ensure clean separation of concerns across app layers (UI, business logic, data)
* **Realtime Messaging**: Use Supabase WebSocket for minimal latency
* **Cross-platform Support**: Consistent experience on Android and iOS
* **Constraint**: Supabase must handle large scale concurrent messaging

**3.2** **Design Assumptions**

* Users access the app predominantly on mobile devices
* Users will have stable internet connections
* Backend components are scalable with Supabase

**3.3** **Significant Design Packages**

* **Auth Module**: Handles login, registration, JWT generation
* **Profile Module**: Manages user data, avatars, and bios
* **Blog Module**: CRUD operations for blog content
* **Messaging Module**: Supabase-driven, handles private and group messages
* **Notification Module**: Manages real-time alerts for follows, likes, messages

**3.4** **Dependent External Interfaces**

The table below lists the public interfaces this design requires from other modules or applications.

|  |  |  |
| --- | --- | --- |
| **External Interface** | **Module** | **Description** |
| Supabase Auth | Auth | User login and signup |
| Supabase Realtime DB | Messaging | Real-time communication |
| Supabase Database | Blog, Profile, Followers | User data, blogs, messages |
| Supabase Storage | Blog, Profile, Media | Media storage |

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

The table below lists the implementation of public interfaces this design makes available for other applications.

|  |  |  |
| --- | --- | --- |
| **Interface Name** | **Module** | **Functionality** |
| Authentication | Auth | Login, Sign up |
| Messaging Interface | Messaging | Send/Receive messages |
| Blog API | Blog | Post, fetch blogs |

**4** **Logical View**

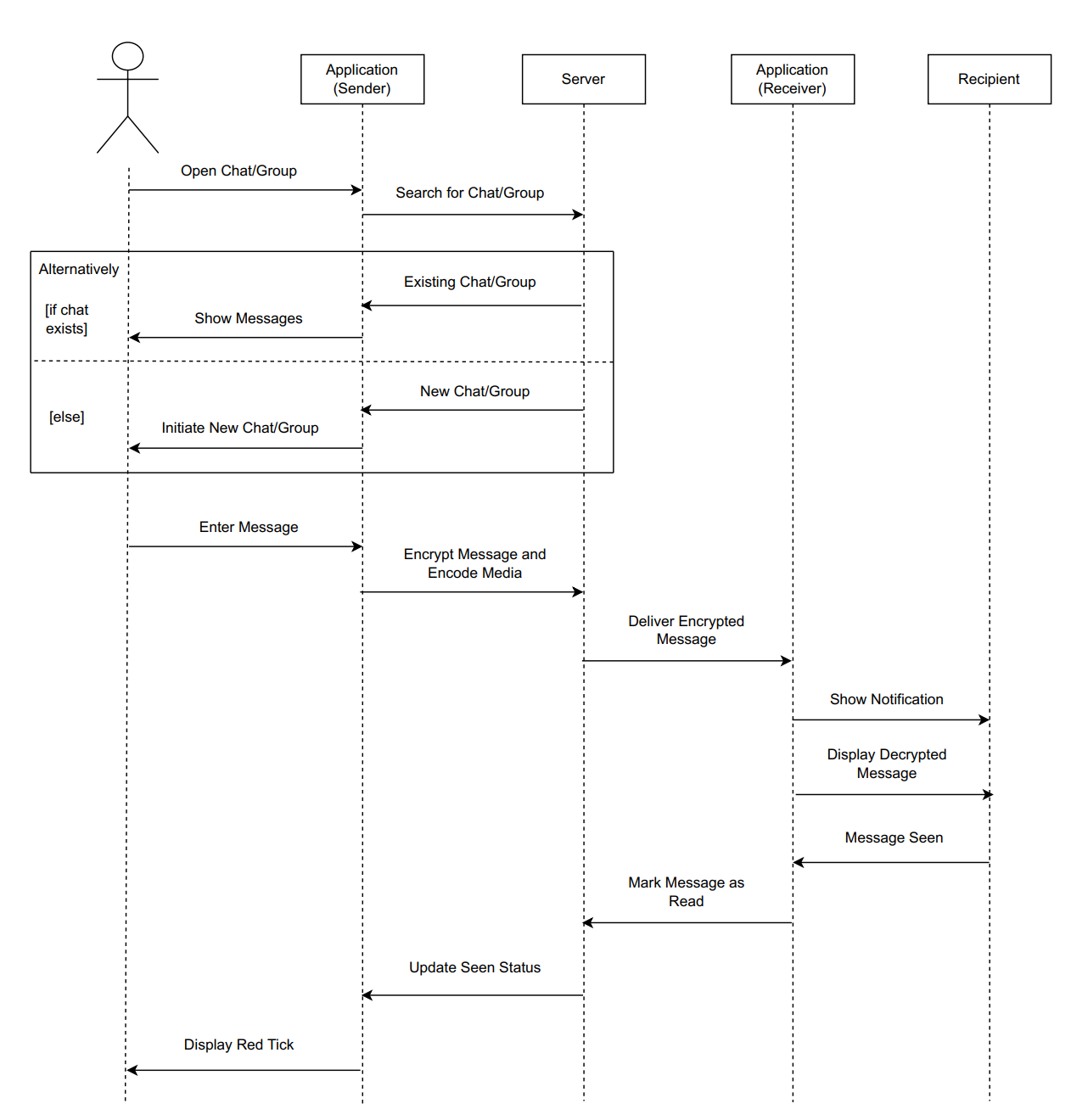
**4.1** **Design Model**

The system is built using a Client-Server Architecture, separating the frontend mobile interface from the backend services.

* Client (Mobile App):  
   Developed using Flutter, the mobile app handles user interactions, rendering UI components, and invoking API requests to interact with the backend.
* Server (Backend Services):  
   The backend is powered by Supabase. Supabase manages database operations and RESTful API endpoints, and also handles authentication and real-time features (e.g., messaging).
* Communication:  
   The client and server communicate via HTTP and WebSocket protocols using secured API endpoints and authentication tokens.

This architecture ensures scalability, separation of concerns, and seamless integration of third-party services.

**4.2** **Use Case Realization**



Use case diagrams and sequence diagrams illustrate:

* Clicking “Followers” leads to FollowersScreen
* Pressing “Notifications” leads to NotificationScreen
* Profile editing updates DB and UI in real-time

**5** **Data View**

**5.1** **Domain Model**

* User (uid, name, profilePic)
* Message (id, senderId, receiverId, content, timestamp)
* BlogPost (id, userId, title, content, tags)
* Notification (id, userId, content, isRead)

**5.2** **Data Model (persistent data view)**

**Supabase Database**

* users/{uid}
* blogs/{blogId}
* messages/{chatId}
* notifications/{notifId}

**5.2.1** **Data Dictionary**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| uid | String | Unique user ID |
| content | Tet | Blog or message content |
| timestamp | DateTime | Time of message/post |
| isRead | Boolean | Notification read status |

**6** **Exception Handling**

* Empty text fields → show UI error
* Network failure → retry with message
* Unauthorized access → redirect to login
* Supabase auth failures handled with alert messages
* Supabase API failures trigger retry logic and offline cache
* All exceptions logged to SupabaseCrashlytics

**7** **Configurable Parameters**

This table describes the simple configurable parameters (name / value pairs).

|  |  |  |
| --- | --- | --- |
| Parameter | Usage | Dynamic |
| MaxMessageLength | Maximum characters in a message | Yes |
| API\_BASE\_URL | Supabase endpoint | No |
| themeMode | Light/Dark mode toggle | Yes |
| itemsPerPage | Blog pagination | Yes |
| rertyAttempts | Retry attempts on failed API calls | Yes |

**8. Quality of Service**

**8.1** **Availability**

* Supabase provides 99.9% uptime for Auth and DB
* Supabase backups run daily to preserve data
* Offline caching where applicable

**8.2** **Security and Authorization**

* Auth handled via Supabase Authentication
* Each user session scoped with JWT
* Role-based access for admin actions
* Supabase RLS (Row-Level Security) used to enforce data privacy
* All API endpoints validated against user roles

**8.3** **Load and Performance Implications**

* Supabase: optimized indexes for fast queries
* Supabase handles >10K messages/day with WebSocket layer
* Lazy loading in lists
* Realtime updates using Supabase listeners

**8.4** **Monitoring and Control**

* Crashlytics for crash reports
* Supabase console for auth and analytics
* Supabase logs for API calls and DB stats