Software Requirements Specification

for

Huddle : Social media App

Version 1.0

Prepared by

Ambaliya Jills	**23ceuoz002**	**23ceuoz002@ddu.ac.in**
Kevin Araniya	**23ceuoz003**	**23ceuoz003@ddu.ac.in**

Lab instructor : HRK

Course : Computer Engineering

Lab Section : A3

CONTENTS **2**

Revisions

Version	Primary Author(s)	Description of Version
0.0.1	Ambaliya Jills	
0.0.2	Kevin Araniya	Primary version

1 Introduction

1.1 Document Purpose

This document outlines the software requirements for Huddle, a mobile application designed to provide users with a private and secure platform to connect with friends, share photos and videos, and interact through a trusted social network.

The document details the complete system functionality from the perspective of both the end-user and the system, covering features such as user authentication, post creation (photos and videos), stories, direct messaging, and a unique QR code system for adding friends. It also describes the platform's non-functional aspects such as usability, performance, and security, ensuring that the application delivers an efficient and user-friendly social media experience.

1.2 Product Scope

Huddle allows users to create a private account, share photo and video posts with their network of friends, and view a chronological feed of content. The application enables users to add friends via a unique QR code, engage in one-on-one chats, and share temporary stories. Designed for accessibility on any modern mobile device, Huddle offers a private, secure, responsive, and user-friendly interface with efficient API-based content delivery.

1.3 Intended Audience and Document Overview

This document is intended for project developers, instructors, clients, and quality assurance teams involved in the development and evaluation of the Huddle application. It provides a structured description of the product, including its functionality, system constraints, user interactions, and performance requirements. Readers are advised to begin with the product overview to understand the core purpose of the app, followed by a detailed breakdown of specific software and design requirements.

1.4 Definitions, Acronyms and Abbreviations

 Flutter: A UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase.

 Dart: The programming language used for developing Flutter applications, optimized for building fast apps on any platform.

 Widget: The core building block of a Flutter application's user interface; every element on the screen is a widget.

 API: Application Programming Interface – allows communication between the Huddle app and its backend services.

 UI: User Interface – the visual layout and design elements of the app.

 JSON: JavaScript Object Notation – a lightweight format for data exchange used in APIs.

 QR Code: Quick Response code used within the app for quickly sharing user profiles and adding friends.

1.5 Document Conventions

- Font: Arial 12pt
- Section titles in bold
- Code and interface elements in monospace

1.6 References and Acknowledgments

- This document references the following resources:
  IEEE SRS Template

2 Overall Description

2.1 Product Overview

- The system is a cross-platform mobile application, **Huddle**, that provides a private and secure social media environment for users to connect with their trusted friends. The architecture includes a frontend built with Flutter and Dart, and a backend system that manages user data, authentication, and media storage via a secure API. Users can register an account, add friends, share photo and video posts, and view a feed of their friends' content. Key features that emphasize security and ease of use include a unique QR code system for adding friends and an encrypted local database to protect user information.

2.2 Product Functionality

 User Registration: Allows new users to create a secure account.

 Add Friend: Enables users to build their private network by adding trusted friends.

 Create Post: Allows users to share content (photos/videos) with their network.

 View Feed: Provides users with a feed to view posts from their friends.

 Generate QR Code: Creates a unique, personal QR code for each user's profile.

 Scan QR Code: Allows users to scan another user's QR code to quickly add them as a friend.

 Admin Oversight: Provides an administrator with access to view device logs and reset application data if necessary.

 Data Encryption: Ensures all user data stored on the device is encrypted for maximum security.

2.2 Design and Implementation Constraints

 The frontend must be built using the **Flutter framework** and **Dart** programming language to ensure cross-platform compatibility for both Android and iOS.

 A secure **RESTful API** must be used for all communication between the mobile application and the backend server.

 The application must implement a local **encrypted database** (e.g., using packages like sqflite with encryption or Hive with an encryption box) to store sensitive user data securely on the device.

 The system will not include public user discovery or content recommendation algorithms, focusing strictly on a private network model.

2.3 Assumptions and Dependencies

 Users have an active and stable internet connection to use online features such as registering, creating posts, and adding friends.

 The user's device must have a functional camera to use the "Create Post" and "Scan QR Code" features.

 The backend API services are assumed to be available, responsive, and secure.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

 Registration Screen: A clean interface for new users to create an account.

 Home Feed: The main screen displaying a scrollable feed of posts from the user's friends.

 Post Creation Interface: A screen that allows users to select media from their device, add a caption, and publish a post.

 Profile Screen: Displays the user's profile information and provides access to their unique, scannable QR code.

 QR Code Scanner: An interface that utilizes the device's camera to scan and process other users' QR codes.

 AppLock Screen: An overlay screen that prompts for biometric authentication (fingerprint or face) when the app is launched.

 Admin Dashboard: A secure, separate interface for the administrator to access system-level functions like viewing logs and resetting data.

3.1.2 Hardware Interfaces

- **Device Camera:** The application requires access to the device's front and rear cameras for creating posts and scanning QR codes.

3.1.3 Software Interfaces

 Backend API: The Huddle application will communicate with a custom backend server via a secure RESTful API. All data exchanges, such as user authentication, post uploads, and friend requests, will be handled through this API.

 Encrypted Database Engine: The application will use a local database engine with encryption capabilities (e.g., Hive with an encryption key) to store all local data securely.

3.2 Functional Requirements

R.1: User Registration

- Description: The system shall allow a new user to create a secure account.
- **R.1.1:** The user shall be able to register by providing a unique username, email, and a password.
- **Input:** Username, email, password.
- **Output:** A new user account is created in the system, and the user is logged in.

R.2: Social Connectivity

- **Description:** The system shall allow users to connect with friends.
- **R.2.1:** The user shall be able to add another user as a friend.
- **Input:** A request to add a specific user.
- **Output:** The two users are connected in the system.

R.3: Content Sharing

- **Description:** The system shall allow users to create and view posts.

- **R.3.1:** The user shall be able to create a post by uploading a photo or video from their device.
- **Input:** Media file (photo/video), optional caption.
- **Output:** The post is published and visible on their friends' feeds.

- **R.3.2:** The user shall be able to view a chronological feed of posts from their friends.
- **Input:** User opens the home screen.
- **Output:** A scrollable list of posts is displayed.

R.4: QR Code Functionality

 Description: The system shall use QR codes for easy friend connections.

 R.4.1: The system shall generate a unique, non-expiring QR code for each user.
 Input: User navigates to their profile.
 Output: A unique QR code image is displayed.

 R.4.2: The system shall be able to scan and decrypt another user's QR code to identify their profile.
 Input: A live camera feed viewing another user's QR code.
 Output: The scanned user's profile is identified and can be added as a friend.

R.4: Application Security

 Description: The system shall provide features to secure the application and its data.

 R.5.1: All data stored locally on the device shall be held within an encrypted database.
 Input: Data to be written to the local database.
 Output: Data is encrypted before being written to disk.

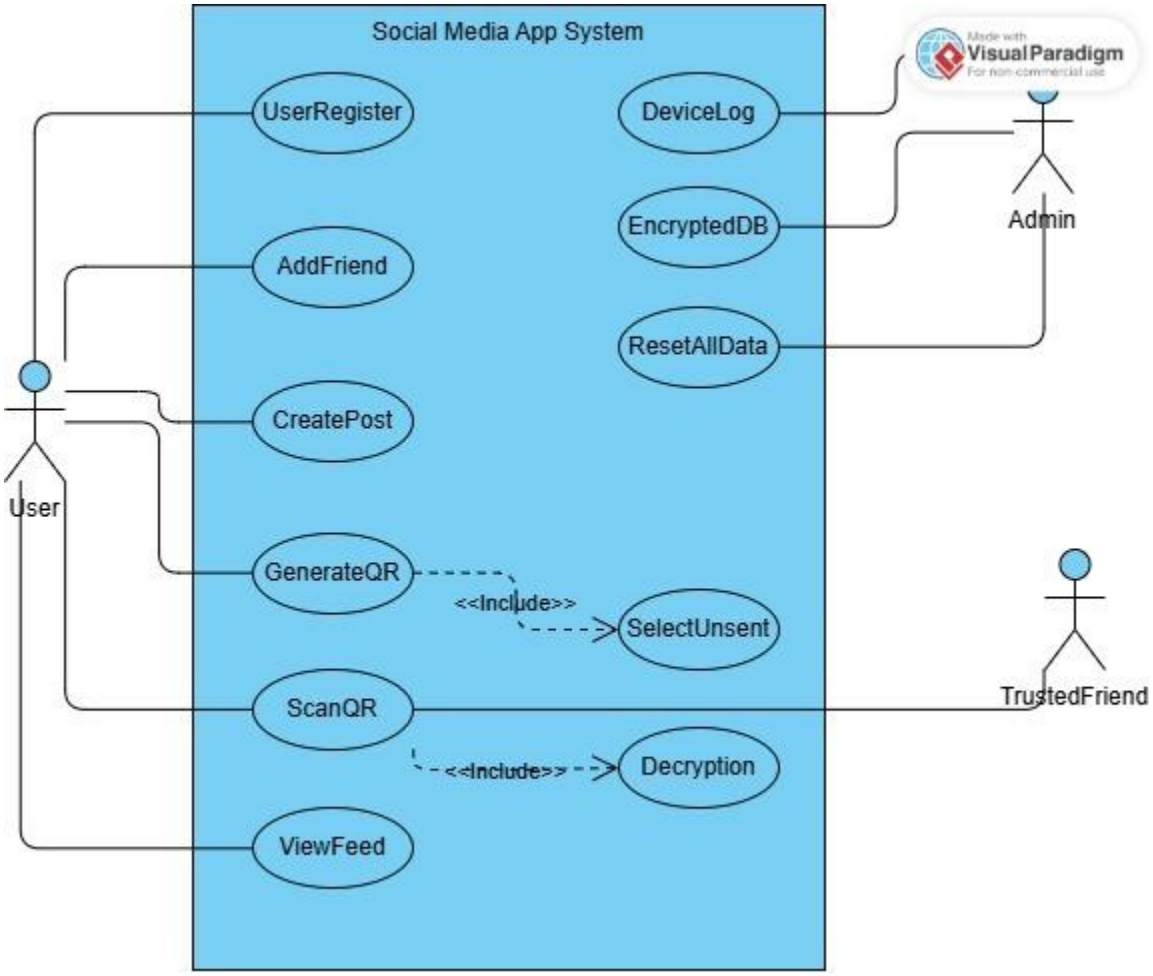
R.6: Administrator Functions

 Description: The system shall provide special capabilities for an administrator.

 R.6.1: The administrator shall be able to view device logs for troubleshooting.
 Input: Admin selects the "Device Logs" option.
 Output: A view of relevant, unsent device logs is displayed.

 R.6.2: The administrator shall have the ability to reset all application data.
 Input: Admin confirms the "Reset All Data" command.
 Output: All user data is wiped from the system's database.

3.3 Use Case Model



Use Case 1: Register User

- **Summary:** A new user creates an account in the Huddle system.
- **Actor:** User.
- **Preconditions:** The user has installed the application on their device.
- **Description:** The user opens the application and selects the option to register. They provide the required information (e.g., username, password). The system validates the information and creates a new account.
- **Exceptions:** The provided username is already taken; the input data is invalid.
- **Postconditions:** The user has an account and is logged into the system.

Use Case 2: Add Friend

- **Summary:** A user adds another user to their private network.
- **Actor:** User.
- **Preconditions:** The user is logged into the system.
- **Description:** The user finds another user's profile (e.g., by scanning a QR code) and initiates a request to add them as a friend. The system processes this connection.
- **Exceptions:** A network error prevents the connection from being made.
- **Postconditions:** The two users are now connected as friends within the system.

Use Case 3: Create Post

- **Summary:** A user shares content with their friends.
- **Actor:** User.
- **Preconditions:** The user is logged in.
- **Description:** The user selects the option to create a new post, attaches media, and publishes it. The system makes this post available in the feeds of the user's friends.
- **Exceptions:** The media upload fails due to a network error.
- **Postconditions:** The new post is visible to the user's network.

Use Case 4: Scan QR Code

- **Summary:** A user scans another user's QR code to view their profile.
- **Actor:** User.
- **Preconditions:** The user is logged in, and another user is displaying their generated QR code.
- **Description:** The user activates the QR scanner within the app. The system uses the device camera to read the code. This process includes a **Decryption** step to securely interpret the user data embedded in the QR code. The app then navigates to the scanned user's profile.
- **Exceptions:** The QR code is invalid or cannot be decrypted.
- **Postconditions:** The user is viewing the profile whose QR code was scanned.

Use Case 6: Admin Views Device Logs

- **Summary:** An administrator reviews system logs for maintenance or troubleshooting.
- **Actor:** Admin.
- **Preconditions:** The Admin is authenticated and has access to the admin panel.
- **Description:** The admin initiates the "SelectUnsent" function, which queries and displays the relevant **DeviceLogs** from the system.
- **Exceptions:** The Admin does not have the necessary permissions.
- **Postconditions:** The device logs are displayed to the Admin.

Use Case 7: Admin Resets All Data

- **Summary:** An administrator erases all data from the system.
- **Actor:** Admin.
- **Preconditions:** The Admin is authenticated and has access to the admin panel.
- **Description:** The Admin triggers the "ResetAllData" function and confirms the action. The system proceeds to wipe all user data from the database.
- **Exceptions:** The action fails due to a database error.
- **Postconditions:** The application's database is cleared of all user-generated content and accounts.

4 Other Non-functional Requirements

4.1 Performance Requirements

 The user's feed shall load completely within 4 seconds on a standard 4G or Wi-Fi connection.

 The QR code scanner shall recognize and decode a valid QR code within 1.5 seconds of it being in focus.

 User interactions such as creating a post or adding a friend shall provide visual feedback to the user almost instantaneously (< 500ms).

4.2 Safety and Security Requirements

 All user data stored locally on the device, including user information and cached content, must be stored in a database that is encrypted using a strong, industry-standard algorithm.

 All communication between the Huddle mobile app and the backend server must be conducted over HTTPS to protect data in transit from interception.

 Data embedded within user QR codes must be encrypted to prevent unauthorized access to user identifiers if the code is intercepted. The scanning process must include a secure decryption step.

 The Admin role must be protected by a separate, robust authentication mechanism to prevent unauthorized access to sensitive functions like viewing logs or resetting data.

4.3 Software Quality Attributes

 Usability: The application shall provide a clean, intuitive, and consistent user interface. Navigation shall be straightforward, and all interactive elements must be clearly identifiable and easy to use. Error messages must be clear and guide the user on how to proceed.

 Maintainability: The Flutter codebase shall be modular, well-structured, and thoroughly commented. Business logic, UI components, and service integrations shall be separated to support easy maintenance, debugging, and future feature enhancements.

 Portability: As a Flutter application, the project shall be buildable and deployable on both Android and iOS platforms from a single codebase. It must be compatible with a wide range of modern devices and screen sizes, ensuring a consistent experience across different environments.

Appendix A – Data Dictionary

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

Appendix B - Group Log

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