CS305 Final Project

Friendify App
High-Level Design Document

1) Introduction

1.1) Why this High-Level Design Document

The HLD document provides a means of communicating the overall design of the software system to stakeholders, such as developers, testers, and customers. It provides a clear and concise description of the system architecture and design, including its components, interfaces, and interactions. This helps ensure that the design is well-organized and easy to understand, which can improve the quality of the final product.

1.2) Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

1.3) Overview

The HLD will:

- present all of the design aspects and define them in detail
- describe the user interface being implemented
- describe the hardware and software interfaces
- describe the performance requirements
- include design features and the architecture of the project
- list and describe the non-functional attributes like:
    - security
    - reliability
    - maintainability
    - portability

- reusability
- application compatibility
- resource utilization
- Serviceability

2) General Description

2.1. Product Perspective

Friendify can be viewed as a software system that provides a platform for users to interact with each other in a safe and secure way. The app is designed to meet the needs of users who are looking for a convenient and easy-to-use way to connect with new people online.
The app's key features, such as login, email verification, and a verified profile tag, are designed to enhance the security and privacy of the app. By requiring users to verify their email address and providing a verified profile tag, the app aims to reduce the risk of fraud and impersonation, while also providing users with more confidence in the people they are interacting with.
The app's rating system is designed to provide users with more control over their experience on the platform. By allowing users to rate other users, the app aims to facilitate more meaningful and enjoyable conversations between users, while also reducing the likelihood of encounters with undesirable people.

2.2. Tools used

Frontend Development	Flutter
Backend Development	express.js
Database	MongoDB

2.3. Assumptions

The goal is to make this idea a reality using Software Engineering practices. In doing so, many documents are created, and it is assumed that design flaws will be found early on. It is also assumed that all aspects of this project have the ability to work together in the way the designer is expecting. Another assumption is that the current intended documentation will suffice to make this project count towards the Software Engineering

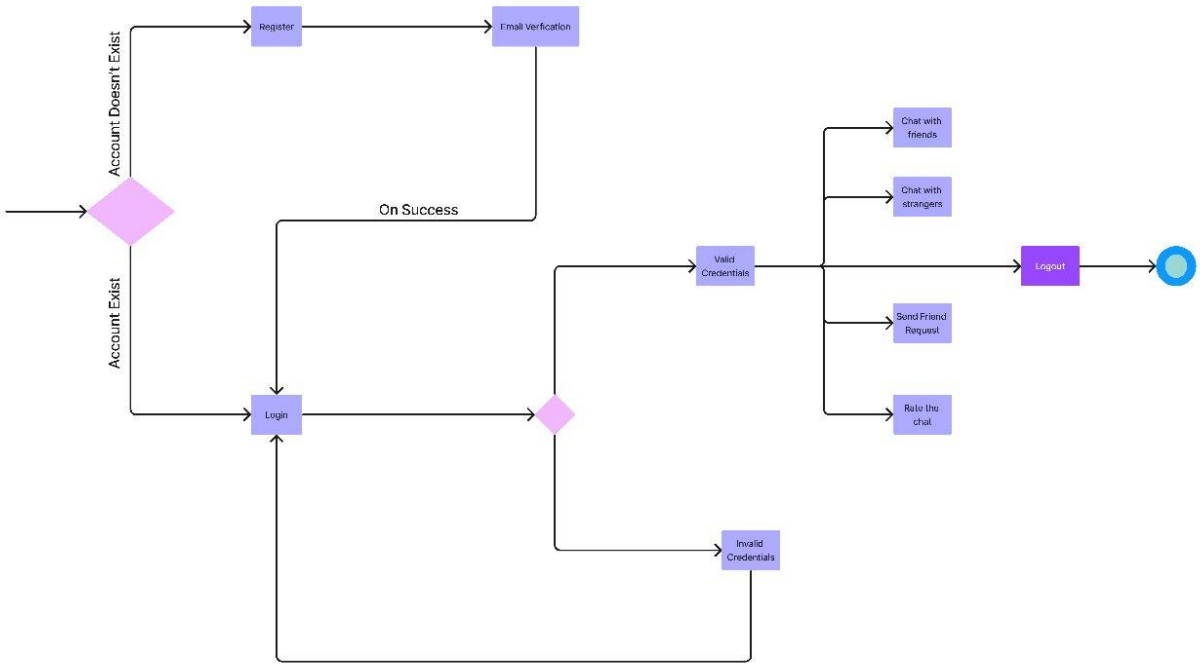
Subtract. There is also an assumption that none of the work or hardware will be stolen or sabotaged. Other assumptions relating to the use of the application include:

1. The users have a low to moderate grasp of English to navigate the application.
2. The users have a reliable internet connection to send and receive messages through the server.

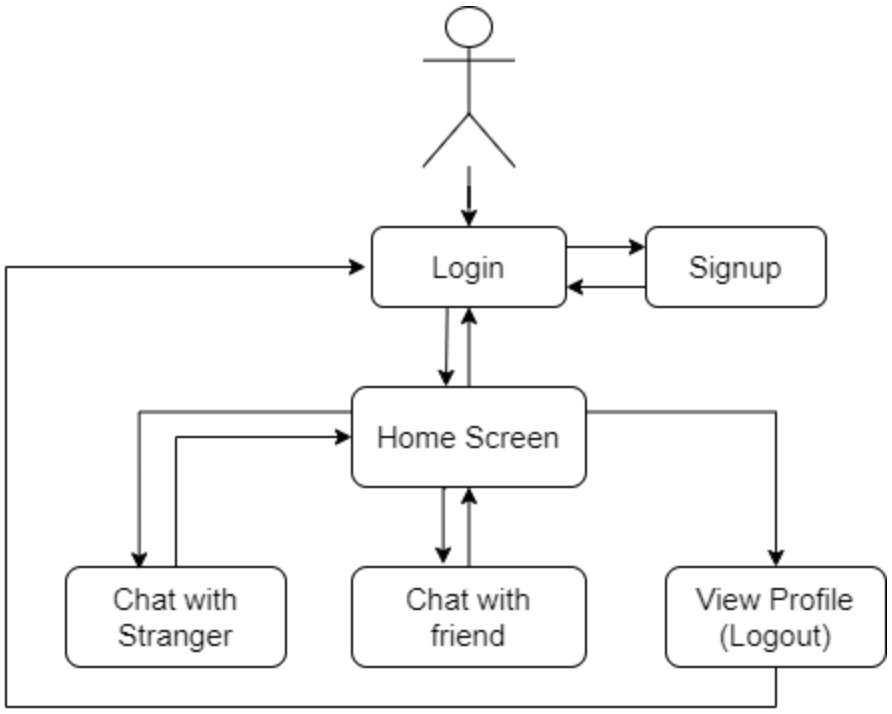
3) Design Details

3.1) Main Design Features

Design features include the architecture, backend where the whole processing of data takes place, frontend where the user interacts with the application, the database where the data(credentials of users) is stored, process relation. These designs are well explained with the help of diagrams shown below.



3.2) Application Architecture



3.3) Technology Architecture

Friendify is a cross-platform application.

3.3.1. Presentation Layer: This layer is responsible for the user interface and user experience of the app. It includes components such as the login screen, chat interface, and user profile screen. The presentation layer interacts with the application logic layer to get and display the data in the UI.

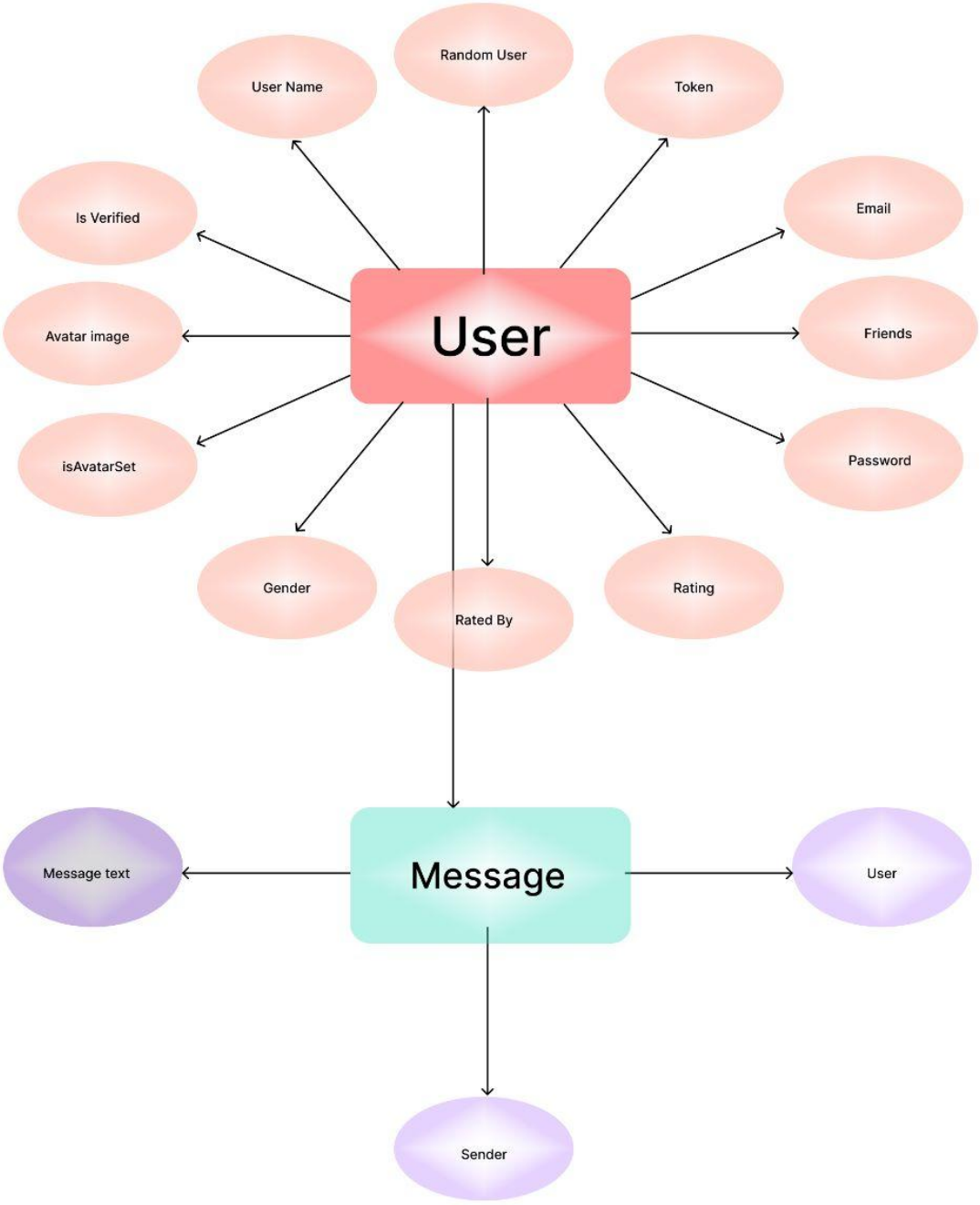
3.3.2. Application Logic Layer: This layer is responsible for the app's business logic and functionality. It includes components such as user authentication, chat management, friend management. This layer is responsible for implementing the app's functionality, validating user input, and enforcing business rules.

3.3.3. Data Management Layer: This layer is responsible for managing and storing the app's data. It includes components such as the user database, chat database, friend database. This layer interacts with the application logic layer to store and retrieve data as needed.

3.4) Standards

Database – Document Oriented NoSQL Database. This database is a non-relational database.
Inputs – Entered through a text field and stored in the database.
Security – A username and password are required for accessing the system.
Quality – by keeping the interface simple and direct, quality should be kept at a maximum.

3.5) Database design



3.6) User Interface

The user interface design of friendify is intuitive and user-friendly, allowing users to easily navigate the app's features and connect with others. The design is also consistent and visually appealing, creating a positive user experience.

3.7. Reports

The admin can log in to the backend anytime to get access to the log files, in order to find any malfunctioning of the Application.

3.8. Error Handling

Should errors be encountered, an explanation will be displayed as to what went wrong. An error will be defined as anything that falls outside the normal and intended usage. Log Files generated by the application will be used in order to mitigate the errors caused by the application.

3.9. Security

The application should be secure in terms of authorization and authentication. Authentication is about validating the user credentials like username and password to verify your identity. Authorization determines your ability to access the system and up to what extent the user can access the resources after the successful login.

3.10. Reliability

The app's server must be reliable and have high uptime to ensure that users can connect to the app and use its features without interruption. A reliable server will also prevent data loss, ensuring that user data is not lost due to server failures. Data security, error handling, testing, and user feedback can ensure that our app is reliable and provides a positive user experience.

3.11. Maintainability

The product will be built using modular components that are as independent as possible to make it easy for debugging , performance improvement and adaptation to changed environment .User interface and backend are the two

main components of our application. Both components will be completely independent from each other for future modifications in UI in terms of adding new customizations or styling or making better models in our database.

3.12. Portability

The app is developed using cross-platform development tool, Flutter, which enables developers to create a single codebase that can run on both android and iOS . Users will be required to install the app on their mobile. Approved users will get access to the app.

3.13. Reusability

To ensure the reusability of our random chat app, we made the app with modularity in mind, using standard interfaces, leveraging libraries and frameworks, and following standard coding practices. This increases the chances that other developers will be able to reuse our app's components in their own applications, thus increasing the overall value and impact of your app.

3.14. Application compatibility

The Friendify app will be compatible with both Android and iOS operating systems. The app will require a minimum operating system version of Android 6.0 (Marshmallow). Additionally, the app will require a stable internet connection for optimal functionality.