NORTH SOUTH UNIVERSITY

JUNIOR DESIGN

CSE299

SECTION 16

RIFAT AHMED HASSAN (RIH)

SEMESTER: SPRING 2024

SUBMITTED BY-

Group 5

Name	**ID**
Abdullah Al Wasif	2131465642
Al-Fawsesh Habib	2131230642
Shadman Khan	2122061642
Naym Hasan Khan	1621640042

Submission Date: 9 December , 2024

Declaration

This is to certify that this Project is our original work. No part of this work has been submitted elsewhere partially or fully for the award of any other degree or diploma. Any material reproduced in this project has been properly acknowledged.

Students' names & Signatures

1. Abdullah Al Wasif .....
2. Al-Fawsesh Habib .....
3. Shadman Khan .....
4. Naym Hasan Khan .....

APPROVAL

We, Wasif, Habib, Shadman and Naym, members of CSE 299 (Junior Project Design) from the Electrical and Computer Engineering department of North South University; have worked on the project titled “Chitter Chatter” under the supervision of Mr. Rifat Ahmed Hassan (RIH) sir as a partial fulfillment of the requirement for the degree of Bachelors of Science in Engineering and has been accepted as satisfactory.

Supervisor’s Signature

.....

Lecturer
Department of Electrical Engineering & Computer Science
North South University
Dhaka, Bangladesh.

Acknowledgments

By mercy of the Almighty, we have completed our junior design project entitled “Chitter Chatter”. Foremost, we would like to express our sincere gratitude to our advisor Rifat Ahmed Hassan (RIH) for his continuous support in our project progress throughout the whole CSE 299 course, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped us in all the time of research, writing and completing of this project. Our sincere thanks also go to North South University, Dhaka, Bangladesh for providing an opportunity in our curriculum which enabled us to have an industrial level experience as part of our academics. Last but not the least, we would like to thank our family as their inspiration and guidance kept us focused and motivated.

Abstract

This paper describes the motivation, design and implementation details behind a real time messaging app we have built named Chitter Chatter. Chitter Chatter is a real time messaging app by which user can easily communicate with others.

We always had a keen interest about how messaging applications like whatsapp, messenger work, how people can efficiently communicate with each others even from different parts of the world. Also, the existing messaging apps has a vast set of cutting edge features but how many of us even user all of them in our daily usage. These vast set of features made them cutting edge but at the same time made the apps laggy and heavy for older generations phone. So we thought of why not build an app that will not only offer the necessary features of a messaging app but also will be light. That's how the idea of Chitter Chatter came along.

We have built an app that not only looks beautiful and easy to use but also is lightweight. We used Flutter to built the frontend of the app and also to give functionality and responsiveness to the app. We have used Firebase as our backend and it just made our app building process easy and efficient by it's rich API options. By these two not only we have made our app visually pleasant to look at but also gave the necessary functionalities.

Some of the key features are easy email verification, password reset, sending message, photos and emojis, sharing status, adding new users, robust searching and many more.

The goal of this app is to provide us with the knowledge of learning how real time messaging systems work and to offer a lightweight and user-friendly version of some of the existing messaging apps.

List of Figures & Tables

Contents

1 Overview

1.1 Introduction

Real-time messaging apps revolutionized digital communication by enabling instant message transmission across devices and networks. These applications use sophisticated technologies to deliver messages in milliseconds after they are sent, creating near-instantaneous communication experiences. Users can exchange text, multimedia, and engage in group conversations with immediate delivery and read receipts. Unlike traditional messaging methods, real-time messaging apps provide live, synchronized communication that bridges geographical distances, allowing people to connect and interact as if they were in the same physical space.

1.2 Project Defination

Chitter-Chatter is a real-time messaging program that allows users to communicate instantly via text, photo and emoji. The platform, which is available across both web and mobile platforms, allows for seamless chatting and includes features such as sending message, photos, sharing status. Users can easily engage with friends, family, and colleagues using a simple, intuitive interface built for quick and easy digital interaction.

1.3 Purpose of our project/ motivation

The motivation of this app is to provide users with a simple, user-friendly and lightweight app that allows users to seamlessly communicate with others. Although the majority of other messaging apps have a large feature set, some users who are new to messaging apps and lack technological expertise may find these programs to be bulky and even overwhelming. Our app provides a pleasing experience yet is lightweight for the little bit older generations of phones.

1.4 Project Goal

The goal of Chitter Chatter is to create a lightweight, user-friendly real-time messaging app that simplifies communication for users of all technical levels. Designed to be visually appealing and responsive, the app offers essential features like efficient login and registration, text, photo, and emoji messaging, status sharing, and robust search capabilities. By focusing on simplicity and efficiency, Chitter Chatter aims to deliver a seamless messaging experience while remaining accessible to users with older devices, bridging the gap between necessary functionality and everyday usability.

1.5 Summary

Chitter Chatter is a real-time messaging app designed to simplify communication with essential features like text, photo, and emoji messaging, status sharing, and user-friendly search. Inspired by the complexities of existing messaging apps, it addresses the need for a lightweight alternative that performs well on older devices without overwhelming users. It ensures efficiency, responsiveness, and visual appeal. The app's goal is to provide a seamless, accessible messaging experience for users of all technical levels, bridging the gap between simplicity and functionality.

2 Existing Systems and solution adopted

2.1 Introduction

Some of the popular real-time messaging applications like WhatsApp, Facebook Messenger, and Telegram, really are on of their kind. While these platforms are feature-rich and offer cutting-edge technologies, they often face challenges such as excessive resource consumption, complexity, and lack of optimization for older devices. This section analyzes the impact of these issues on user experience and explains the solutions implemented in Chitter Chatter to address these shortcomings.

2.2 Existing Solutions

Apps like WhatsApp, Facebook Messenger, and Telegram may be the best but they come with some issues:

- **High Resource Usage:** Many apps are resource-intensive, leading to slower performance, especially on older devices.
- **Complex Features:** Overloaded with advanced features, they can overwhelm users who prefer simplicity.
- **Large App Size:** Excessive storage requirements make these apps less accessible for devices with limited space.
- **Limited Accessibility:** Heavy reliance on modern hardware excludes users with older or low-spec devices.

2.3 Proposed Solutions

To overcome the limitations of existing messaging apps, a set of targeted solutions is proposed:

- **Lightweight Design:** Optimize the app to ensure smooth performance even on older devices by reducing resource consumption.
- **Essential Features Only:** Focus on core functionalities like messaging, photo sharing, and status updates, avoiding unnecessary complexities.
- **RResponsive UI:** Focus on creating a user-friendly, visually appealing, and responsive interface.
- **Compact App Size:** Optimize assets and code to ensure the app requires minimal storage space without sacrificing functionality.

2.4 Adopted Solutions and reasons

To address the challenges identified in existing messaging apps, Chitter Chatter adopts the following solutions, ensuring improved performance, simplicity, and user experience.

- **Optimized Design:** The app is efficient for older devices, ensuring smooth performance and reduced resource usage.
- **Core Features Focus:** Chitter Chatter offers only essential features like messaging, photo sharing, and status updates, keeping the app simple and lightweight.
- **Firebase Backend:** Leveraging Firebase's rich APIs, Chitter Chatter ensures real-time synchronization and efficient data management. This not only enhances speed and reliability but also minimizes data usage, making it a cost-effective solution.
- **Flutter UI:** Built using Flutter, the app delivers a visually appealing and responsive interface. The cross-platform nature of Flutter ensures consistent performance across mobile and web platforms, while its flexibility allows for a clean and intuitive user experience.

2.5 Summary

The section highlighted limitations of popular messaging apps, such as high resource usage, complex features, and large app size. To address these challenges, Chitter Chatter adopts targeted solutions, including compatibility for older devices, a focus on essential features, efficient real-time communication powered by Firebase, and a responsive interface built with Flutter. These measures ensure a lightweight, user-friendly app that offers seamless performance, and accessibility across diverse devices.

3 Technical Description

3.1 Design Overview

Chitter-Chatter integrates a robust UI framework with reliable backend services and secure data management. Below is an overview of its design components:

- **User Interface (UI):** Built using Flutter, the app provides a cross-platform, responsive interface for both mobile and web users. The UI emphasizes simplicity, with intuitive navigation and visually appealing layouts.
- **Backend:** Powered by Firebase, the backend ensures real-time data synchronization and efficient handling of requests. Firebase Cloud Functions are used to manage server-side logic, enabling automation and dynamic functionality.
- **Database:** Firebase's Cloud Firestore, a NoSQL database, is used for real-time data storage and retrieval. The database stores user profiles, chat messages, pictures, and statuses.
- **Authentication:** Firebase Authentication handles all user authentication tasks, including sign-in, and registration. Email verification, password reset, and secure storage of credentials are implemented using Firebase APIs.
- **APIs:** Firebase APIs are used extensively for operations like authentication and data synchronization. Additional APIs are integrated for functionalities such as sending email verifications and handling forgotten passwords.

3.2 Client-Server Architecture

Chitter Chatter employs a robust client-server architecture to enable real-time communication and efficient data management. The client-side, developed using Flutter, provides a responsive and user-friendly interface that seamlessly interacts with the server. The server-side functionality is powered by Firebase, which handles backend operations and ensures reliable performance. Firebase serves as the central hub for managing the app's database, authentication, and real-time messaging. It stores user information, messages, and other data in a cloud-hosted NoSQL database, ensuring quick and scalable access. The authentication system, including sign-in, registration, email verification, and password reset, is implemented using Firebase Authentication APIs. This architecture allows the client-side application to send requests to Firebase's backend services, which process and return data in real-time.

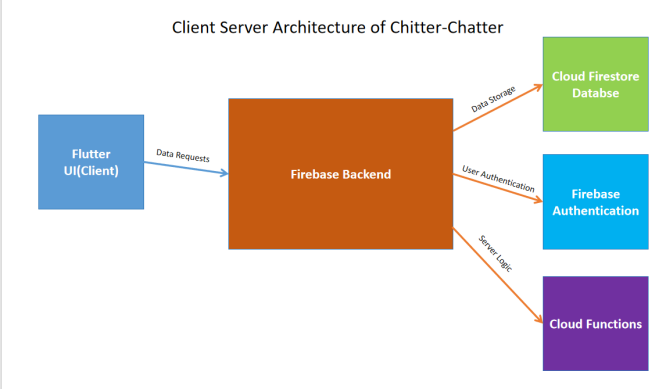

Fig 3.2.1: Client-Server Architecture

3.3 Database Design

Chitter Chatter uses Firebase Cloud Firestore as its database, providing a scalable, cloud-hosted NoSQL solution for storing and managing app data.

- Users Collection:
  - Each document is identified by a unique UserId.
  - Contains user profile information.
  - Has two sub-collections: chatList and Stories.
- Chats Collection:
  - Each document is identified by a unique ChatId.
  - Contains message-related information including participants, sender, receiver, and timestamp.
- Sub-Collections of Users:
  - ChatList: Contains chat-related metadata for each user.
  - Stories: Stores individual story documents with date and status.
- The schema uses standard data types:
  - string for text-based fields and also profile photo field as it is Base64 encoded.
  - boolean for true/false states.
  - timestamp for date and time information.
  - array¡string¿ for lists of items (like participants).

3.4 Database Schema

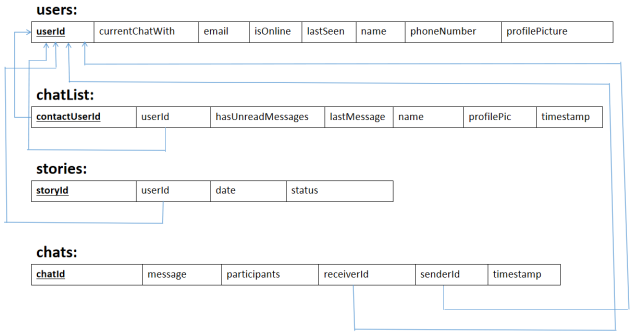

Fig 3.4.1: Database Schema

3.5 Technology Stack

3.5.1 UI Framework:

Flutter is a cross-platform UI framework used to build Chitter Chatter's responsive and visually appealing interface. It enables seamless development for both mobile and web platforms with a single code-base. Known for its fast rendering and customizable widgets, Flutter ensures a consistent and intuitive user experience across all devices.

3.5.2 Programming Language:

Dart is the programming language used to develop Chitter Chatter's frontend with Flutter. Designed for client-side development, Dart offers features like a rich standard library, strong typing, and asynchronous programming, enabling smooth and efficient application performance.

3.5.3 Backend:

Firebase is the backend platform for Chitter Chatter, enabling real-time communication, secure authentication, and seamless app functionality. Its APIs handle critical processes like user sign-in, registration, email verification, and password recovery. Firebase ensures efficient backend operations and enhances app performance through its powerful tools and integrations.

3.5.4 Database:

Firestore, a NoSQL database by Firebase, is used in Chitter Chatter to manage and store app data efficiently. It supports real-time data synchronization, ensuring instant updates for messages, user profiles, and statuses across devices. Firestore's flexible structure enables dynamic data handling, while robust indexing ensures quick and efficient queries. Its security rules safeguard data access and maintain user privacy, making it ideal for scalable applications like Chitter Chatter.

3.6 User Interface (UI) Design

3.6.1 Design Principles:

Chitter Chatter's UI design is rooted in simplicity, accessibility, and aesthetic appeal. By adhering to minimalist design principles, the app avoids overwhelming users with unnecessary elements, creating an interface that is both functional and visually clean. Consistent use of colors, typography, and spacing establishes a cohesive visual identity. Throughout the project we tried different styles and emphasized on clarity and intuitive navigation, ensuring that users can access features easily, regardless of their familiarity with messaging apps.

3.6.2 Responsive Layout:

Built with Flutter, Chitter Chatter's layout is fully responsive, adapting seamlessly to various devices, including smartphones, tablets, and web browsers. This ensures that users experience a consistent look and feel, regardless of screen size or orientation. Feature like platform-specific optimizations ensure that the app is consistent on responsiveness and aesthetics across devices, delivering a smooth experience for both mobile and web users.

3.6.3 User Experience (UX) Considerations:

The app prioritizes user comfort and engagement by focusing on intuitive interactions and thoughtful design elements. Icons and buttons are clearly labeled and logically placed, making navigation straightforward. Features like real-time feedback for actions, smooth transitions between screens enhance usability. Additionally, the design takes into account accessibility, ensuring the app is user-friendly for individuals with varying levels of technological expertise. This focus on UX creates a satisfying and efficient experience for all users.

3.7 Description of the system

3.7.1 Registering to the app

User will be able to register to the app with essential information like email, phone number and password. So that, user can register to the app as a user and use all the features of the app.

Confirmation

Success:

- User inputs the email address.
- User inputs the phone number with appropriate country code.
- User enters a strong password that must contain capital letters, small letters, numbers, special characters and a minimum length of 6.
- User re-enters the same password.
- User clicks on the register button to move forward in the next screen.

Failure:

- If user enters an email that was already registered to the app, user can not use the same email and will get an error message.
- If user enters a phone number that was already registered to the app, user can not use the same phone number and will get an error message.
- If user enters a password that does not follow the convention, user will get error message.
- If user re-enters a password that does not match with the previous password that was entered user will get error message.

3.7.2 Picking profile photo and name

User will be able to select profile picture and name. So that, user can make their profile intuitive and distinguishable.

Confirmation

Success:

- After user have provided valid information in the registration screen, user is redirected to this screen.
- User clicks on the little plus icon on the profile photo selecting area.
- User selects a photo from the gallery.
- User types in the name of his choice.
- User clicks on the tick icon to move forward in the next screen.

Failure:

- If user do not pick any image and try to move on to the next screen, user will get proper error message.
- If user do not type in any name and try to move on to the next screen, user will get proper error message.

3.7.3 Verifying email

Users will be able to verify their email address. So that, no one else can arbitrarily use their email address to create any account.

Confirmation

Success:

- After user is done selecting their profile picture and name user can come to this screen.
- User clicks on the verify button on the screen.
- A verification email will be sent to the provided email address.
- That email will contain a link and when user clicks on the link the user becomes verified.
- As soon as the user is verified the verify button will turn to continue button.
- User can click on the resend email button to get the verification email again.

Failure:

- If for some reason the user do not get verified even after clicking on the link the verify button will not turn into continue button and user can not move to the next screen.
- Then user have to click on resend email button to get the email again.

3.7.4 Logging into the app

User will be able to log into their account. So that, user can use the app's features.

Confirmation

Success:

- User provides his email and password.
- User clicks on the login button.
- User logs into his account and moves to the homepage.

Failure:

- If user provides wrong email or password user will get appropriate error message.

3.7.5 Resetting password

Users will be able to reset their password. So that, they do not have to worry about forgetting their password.

Confirmation

Success:

- User clicks on the forgot password button.
- User is then redirected to the forgot password screen.
- User provides an email address and clicks on the send email button.
- A password reset email is sent to the provided email address.
- The email will contain a link and user clicks on the link.
- User then redirected to another screen where he will have to type in his new password and click on save button to reset his password.

Failure:

- If user do not get any reset email user have to reload the screen and click on the send email button again.

3.7.6 Getting help for different features

User will be able to get some walk-through for some core features of the app. So that, user can always look at the guidelines for using a features.

Confirmation

Success:

- User clicks on the help icon.
- User User then redirected to the help and support page where users get various guides for features.
- User clicks in any of them.
- User can see step by step guideline to use that features.

Failure:

- If due to connectivity issue the icon is not visible user can reload the app.

3.7.7 Searching through the recent chat list

Users can search through the recent chat list to find any desired contact. So that, they do not have to scroll through the list.

Confirmation

Success:

- User clicks on the search icon.
- User then moved to the search page.
- User search users by name.
- User finds the desired contact or person.
- User clicks on that person to move into the chat screen.

Failure:

- If the person is not found the app will give the user proper feedback.

3.7.8 Logging out form the account

Users will be able to log out from their account. So that, they can keep their account safe while login elsewhere and join from another account.

Confirmation

Success:

- User clicks on the logout icon.
- User then logged out from his account and moves to the login screen.

Failure:

- If due to connectivity issue the icon is not visible user can reload the app.

3.7.9 Changing profile information

Users will be able to edit their profile information. So that, they can make necessary changes whenever they find it necessary.

Confirmation

Success:

- User clicks on the user's profile picture or name on the homepage.
- User then redirected to the profile screen.

- User can edit their name, phone number or profile photo.
- User clicks on the save changes button.
- User's profile info changed.

Failure:

- If user keep any of the field blank user will get appropriate error message and can not change the profile information.

3.7.10 Displaying recent chat list

Users will be able to see the recent contacts who they are connected with. So that, they can quickly find users who they have made a conversation with.

Confirmation

Success:

- In the home page under the chats tab user finds the recent chat list.
- This are the users who are connected to the current user.
- User clicks on any of the users.
- User redirected to the chat screen of that contact.

Failure:

- If due to connectivity issue the list is not shown, user will see loading indicator or proper feedback.

3.7.11 Deleting user from the chat list

Users will be able to delete any of the contact in the chat list. So that, they can remove unwanted contacts.

Confirmation

Success:

- User clicks on the bin icon on the right side of each contacts.
- User gets a pop-up confirmation screen.
- User clicks on delete and that contact is deleted from the chat list.

- User clicks on cancel, the app will do nothing.

Failure:

- If the contact is not deleted on first try user can reload the app to check if the contact is deleted or not.
- If the contact is not deleted then user have to repeat the same process again.

3.7.12 Finding saved contacts

Users will be able to see contacts which are saved on their mobile phones. So that, they can add a new user by saving his contact number on their mobile.

Confirmation

Success:

- User clicks on the green button on the bottom right corner of the screen.
- User then redirected to the contacts page where all the contacts which are saved in the users mobile phone will be shown.
- User clicks on any of the contact.
- User goes to that contact's chat screen.
- If that contact was not already present in the recent chat list, the user will be added to the recent chat list.

Failure:

- If user clicks on a contact that is not registered in the app the user will see appropriate error message like "The contact is not registered in the app".

3.7.13 Searching saved contacts

Users will be able to search contacts which are saved on their mobile phone. So that, they can find a specific contact quickly by just writing the name of the user.

Confirmation

Success:

- User clicks on the search icon in the top right corner in the contacts page.
- User then redirected to the search page.

- User types in the name of the user.
- User finds the user.
- User can click on that user to move to that chat screen if that user is registered in that app.

Failure:

- If no user found of that name user will get feedback like ”No user found.”
- After finding the contact when user clicks on the contact and if that is not registered in the app the user will see appropriate error message like ”The contact is not registered in the app”.

3.7.14 Viewing active status

Users will be able to see if the other user is online or offline also when the user was last seen. So that, they can know when to actively participate in a conversation or leave a message for future reference.

Confirmation

Success:

- User can see other user’s active status just below the user’s name.
- When user is in the chat screen the status indicator will turn green meaning the user is online.
- If any user leaves the chat screen the indicator will turn Grey meaning offline.
- User can also see when the user was last active.

Failure:

- If the indicators are not updated due to connectivity issue user have to restart the app.

3.7.15 Sending message

Users will be able to send message to other users. So that, they can actively communicate with others.

Confirmation

Success:

- User types in message in the input text field in the chat screen.
- User clicks on the send button on the right side of the text input field.
- The message is sent.

Failure:

- If due to connectivity issue the message is not send user can check his connection or restart the app, as message will not be sent.

3.7.16 Sending emoji

Users will be able to send emojis to other users. So that, they can make their chatting experience fun.

Confirmation

Success:

- User clicks on the emoji icon in the left most side in the input text field.
- User gets various kinds of emojis.
- User clicks on the emojis and then click on the send button to send the emojis.

Failure:

- If there is connectivity issue the emojis will not load. User have to fix the connectivity issue first.

3.7.17 Sending pictures

User will be able to send pictures to other users. So that, they can express themselves better and send pictures to others.

Confirmation

Success:

- User clicks on the attachment icon just to the right of the emoji icon in the input text field.

- User can select pictures from the gallery.
- User clicks on the picture and the picture is sent.

Failure:

- If there is connectivity issue the pictures in gallery will not load. User have to fix the connectivity issue first.

3.7.18 Indication of unread message

Users will be able to know if they have unread messages. So that, they can respond to messages that they have not read yet.

Confirmation

Success:

- User gets a new message.
- The chat gets a blue indicator on the profile picture and the chat is highlighted.
- User clicks on the chat.
- User reads the message.
- The indicator disappears.

Failure:

- If the user can not see the indicator instantly user may have connectivity issue. User can reload the app.

3.7.19 Searching messages

Users will be able to search old messages. So that, they can quickly locate specific details shared in past conversations.

Confirmation

Success:

- User clicks on the search icon on the rightmost corner of the chat screen
- User then redirected to the search screen.
- User types in the message.

- User finds the searched message.

Failure:

- If the searched message does not exists, user will see appropriate messages like ”No messages found”.

3.7.20 Sharing notes

Users will be able to share short texts on the app. So that, they can share thoughts, announcements, or simple messages without starting a full conversation.

Confirmation

Success:

- User clicks on the status tab in the homepage.
- User redirected to the status screen where user can see short notes shared by other users as well as by him.
- User clicks on the plus(+) button on the left most corner of the status screen.
- User clicks on the plus icon.
- A text field will appear and user can type in a short note, maximum length of 250 characters.
- User clicks on the share button to share the note among all other users.

Failure:

- If the user can see the uploaded note instantly user might have some connectivity issue and have to send the status again.

3.8 Authentication And Security

3.8.1 User Registration:

- Users provide a unique email, phone number, and password during registration.
- The app checks the uniqueness of the email through firebase authentication and phone number through firestore query, triggering an error if duplicates are found.
- Passwords must meet specific criteria: uppercase, lowercase, numbers, special characters, and a minimum length of 6 characters.
- After registration, an email verification process is triggered, requiring users to verify their email before logging in.

3.8.2 Email Verification:

- After registration, Firebase sends a verification email with a confirmation link.
- The user must click the link to verify their email, ensuring account validity.
- Users cannot go to login screen until the email is verified, adding an extra layer of security.

3.8.3 Login Process:

- Users must log in with their email and password.
- Firebase Authentication checks the credentials and allows access if they match.
- If the user provides incorrect credentials, an error message is displayed, notifying them of the invalid email or password.

3.8.4 Password Storage and Security:

- Passwords are stored securely in Firebase Authentication using hashed encryption, ensuring that they are never stored in plain text.
- Firebase uses advanced encryption algorithms like SCRYPT to hash passwords, making it computationally infeasible to retrieve the original password.

3.8.5 User Data Storage:

- Firebase Authentication handles storing email, and password.
- Sensitive data is stored securely, and only hashed passwords are kept in Firebase's database.
- Additional user information, such as phone number, profile details and preferences, are stored in Firebase Firestore, with strict security rules to control access.

3.8.6 Security Measures:

- SSL/TLS encryption ensures secure data transmission during authentication processes.
- Firebase Security Rules protect user data in Firestore, limiting access based on authentication status and user permissions.

4 Project Progress

Milestone	Tasks	Date
Week-1	Learning basics of flutter, firebase, Landing Screen	19/09/2024
Week-2	Learning basics of flutter, firebase, design of login, Signup Screen, user info screen.	26/09/2024
Week-3	Email verification, forgot password, Homepage	03/10/2024
Week-4	Chat screen layout, online and offline status, Recent Chat list	10/10/2024
Week-5	Chat screen input field design, Homepage Search, Help screen, logout	17/10/2024
Week-6	User profile screen, recent chat list, deleting from recent chat list	24/10/2024
Week-7	Contacts page, contacts search, adding new user, Sending message and emoji	31/10/2024
Week-8	Sending emoji and photo, dynamic recent chat list, optimizing screens for web version	07/11/2024
Week-9	Changing online and offline status dynamically, unread message indicator, optimizing screens for web version	14/11/2024
Week-10	Status screen and posting status, complete help screen	21/11/2024
Week-11	Testing and finding bugs, fixing some major bugs	28/11/2024
Week-11	Final Presentation	28/11/2024
Week-12	Final Demo	05/12/2024

TABLE 4.1: WEEKLY PROGRESS

5 Result and Discussions

5.1 App

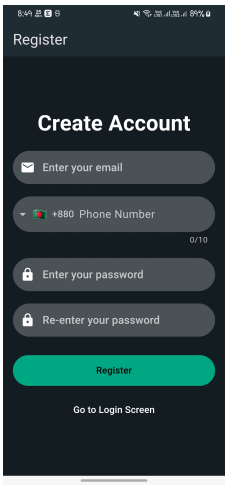

Fig 5.1.1: Create Account Screen

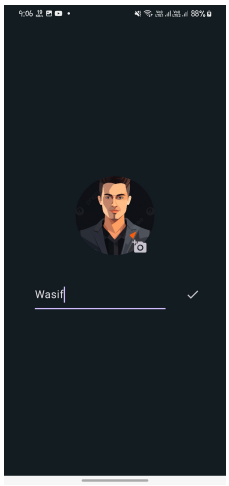

Fig 5.1.2: Name Selection Screen

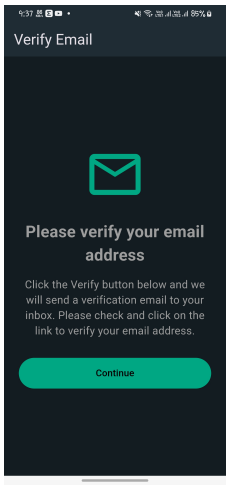

Fig 5.1.3: E.V. Screen

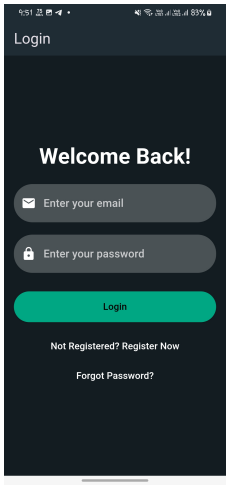

Fig 5.1.4: Login Screen

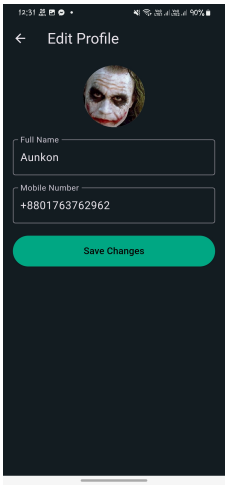

Fig 5.1.7: Profile Screen

Fig 5.1.10: Messaging Screen

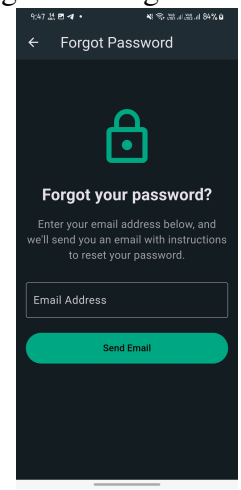

Fig 5.1.5: Password Reset Screen

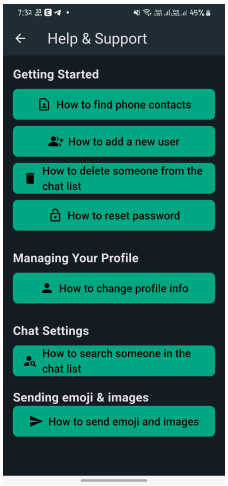

Fig 5.1.8: Help Screen

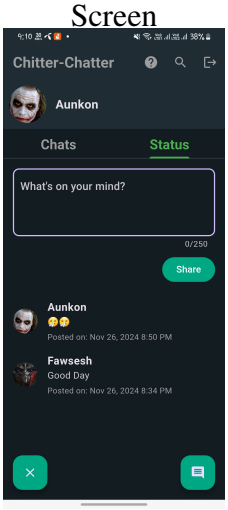

Fig 5.1.11: Status Screen

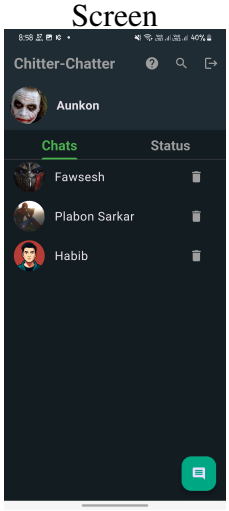

Fig 5.1.6: HomePage

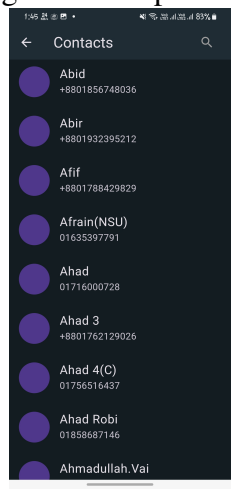

Fig 5.1.9: Contact Screen

5.2 Web

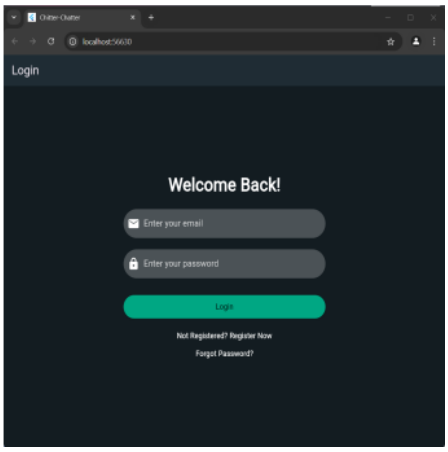

Fig 5.2.1: Login Screen

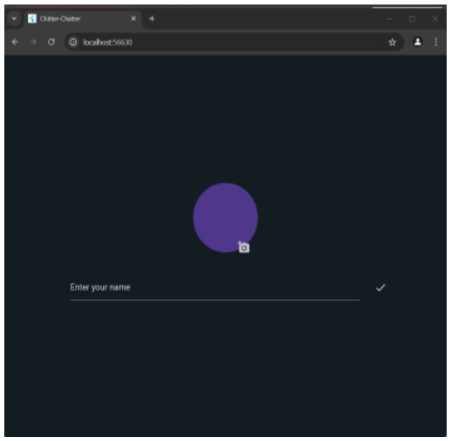

Fig 5.2.2: Info set screen

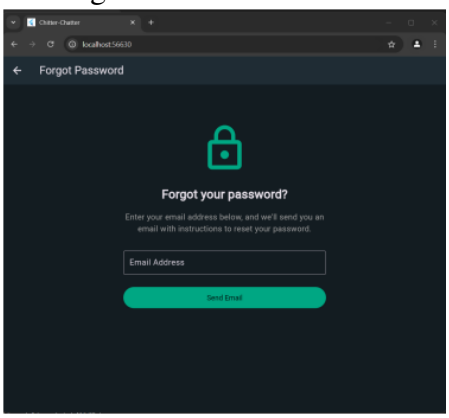

Fig 5.2.3: Password Reset Screen

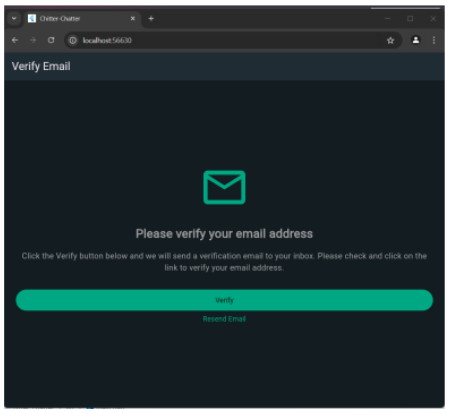

Fig 5.2.4: E.V Screen

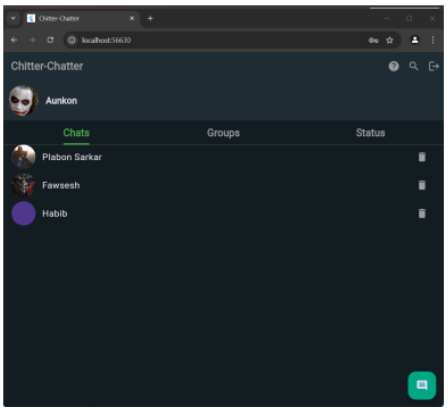

Fig 5.2.5: HomePage

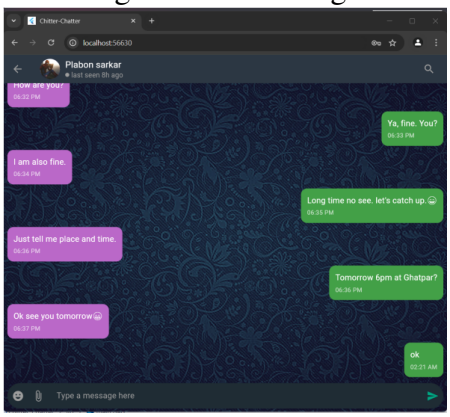

Fig 5.2.6: Chat Screen

5.3 Future Consideration

Chitter Chatter aims to evolve into a comprehensive real-time communication platform with several advanced features planned for future updates. Group messaging will enable users to interact and collaborate in dedicated chat rooms, fostering a sense of community. End-to-end encryption will be implemented to ensure that all conversations remain private and secure, protecting user data from unauthorized access. File sharing functionality will allow users to exchange documents, videos, and other media seamlessly. Additionally, audio and video calling features will provide a more interactive and engaging communication experience, making Chitter Chatter a versatile and all-in-one messaging solution.

6 Conclusion

Chitter Chatter represents a streamlined and efficient approach to real-time messaging, combining essential features with a lightweight design. By leveraging Flutter for a responsive UI and Firebase for backend functionality, the app delivers a user-friendly experience that is accessible across devices. Core features such as email verification, password security, and intuitive navigation ensure reliability and ease of use, while robust authentication and data security measures prioritize user privacy. The app addresses common issues in existing platforms, offering a fast and seamless alternative suited to diverse user needs. With plans to introduce advanced features like group messaging, end-to-end encryption, file sharing, and audio/video calling, Chitter Chatter is positioned to grow into a versatile communication tool. This project underscores the integration of cutting-edge technologies to create a scalable, secure, and user-focused messaging app, meeting both current demands and future expectations.

7 GitHub Link

Here is the link of our GitHub project repository.
https://github.com/whowasif/ChitterChatter-299-project.git

8 References

[1] Flutter and dart Tutorial, "Flutter & Dart Tutorial for Beginners," YouTube, Apr. 15, 2020. Available: https://www.youtube.com/watch?v=VPvVD8t02U8.

[2] Flutter Documentation, "Learn Flutter: Get Started with Fundamentals," Flutter Official Documentation. Available: https://docs.flutter.dev/get-started/fundamentals.

[3] Firebase Fundamentals, "Firebase Full Course – Learn Firebase for Web, iOS, Android," YouTube, Aug. 10, 2021. Available:https://www.youtube.com/watch?v=fgdpvwEWJ9M&t=4s.

[4] Dart Basics, "Dart Cheat Sheet," Dart Official Documentation. Available: https://dart.dev/resources/dart-cheatsheet.