**CS683 Project Assignment   
Texting App  
Aayush Raghuvanshi**

**Github link :-** <https://github.com/CS683/project-aayush-raghuvanshi>

[**Overview**](#_g6igqliy7rm) **2**

[**Related Work**](#_bf21eadgjj29) **3**

[**Requirement Analysis and Testing**](#_9dheewbiht5g) **5**

[**Design and Implementation**](#_312k3b3li0xh) **8**

[**Project Structure**](#_hkcglxnjhrt2) **14**

[**Timeline**](#_tp0jpote18vj) **16**

[**Future Work (Optional)**](#_wx5fnmke6x6g) **16**

[**Project Demo Links**](#_nl6zntsisnrv) **17**

# Overview

*The motivation behind initiating the development of a messaging app using Android Studio, Kotlin, and Firebase stems from a combination of personal, educational, and professional objectives. This report aims to articulate the primary motivations driving the conception and implementation of this project.*

* ***Portfolio Development:*** *Creating a messaging app is in line with the objective of assembling a comprehensive portfolio. A robust and well-designed messaging application will demonstrate the developer's skill in creating user-friendly user interfaces, adding real-time communication capabilities, and putting safe user authentication in place.*
* ***Problem-Solving and Innovation:*** *The project is motivated by a desire to innovate in the messaging app market and address communication difficulties. The initiative seeks to provide creative answers to improve user experience and address changing communication needs by finding areas for improvement in current apps or conceiving new features.*
* ***Educational Purpose:*** *Investigating the messaging app's educational features is a secondary goal. With features that speed up group discussions, file sharing, and student-teacher interactions in the context of learning and collaboration, it could be customized to promote communication in educational settings.*
* ***Individual Development:*** *Embarking on the development journey of a messaging app is seen as an opportunity for personal growth. The challenges posed by designing, implementing, and launching a fully functional app contribute to the developer's resilience, problem-solving abilities, and project management skills.*

*Creating a texting app using Android Studio with Kotlin and Firebase involves several steps. Below is an overview of the process, including the key components and functionalities you'll need to implement:*

*1.* ***Setting up Firebase Project:*** *Create a new project on the Firebase Console. Add an Android app to your Firebase project and follow the setup instructions to integrate the Firebase SDK into your Android Studio project. Enable Firebase Authentication, Firebase Realtime Database, and Firebase Cloud Messaging for your project.*

*2.* ***Setting User Authentication:*** *Implement user registration and login functionality using Firebase Authentication. Store user information (e.g., UID, username, profile image) in the Firebase Realtime Database.*

*3.* ***Design User Interface:*** *Developed the UI for messaging app, including activities, fragments, and layouts. Implement a chat interface with a RecyclerView to display messages.*

*4****. Implement Real-time Messaging:*** *Use Firebase Realtime Database to store and retrieve chat messages in real-time. Set up listeners to receive real-time updates when new messages are added.*

*5.* ***Sending and Receiving Messages:*** *Implement the functionality to send messages between users. Display incoming and outgoing messages in the chat interface.*

*6.* ***User List and User Search:*** *Implement a user list to display all registered users. Add a search feature to find and initiate conversations with other users.*

*7.* ***Notifications:*** *Set up Firebase Cloud Messaging (FCM) to handle push notifications for new messages. Implement the client-side code to receive and display notifications.*

*8.* ***User Profile:*** *Allow users to set up and update their profiles, including profile pictures and status.*

*9.* ***Additional Features:*** *Implement features such as group chats, message timestamps, and message status (read, unread). Consider adding multimedia messaging capabilities (e.g., images, videos).*

*10.* ***Testing:*** *Test your app thoroughly on different devices and screen sizes. Test the real-time functionality by simulating multiple users.*

*11.* ***Security:*** *Ensure proper security rules are set up on Firebase Realtime Database to secure user data. Implement secure authentication practices.*

*12.* ***Deployment:*** *Optimize your app for performance. Preparing app for deployment to the Google Play Store.*

# Related Work

* *Mobile Messaging App Development: The landscape of mobile messaging app development has witnessed extensive exploration in both academic research and practical implementations. Notable authors such as Jane Doe have contributed insights into designing intuitive user interfaces and incorporating essential features for seamless communication in mobile messaging applications [****John Horton****].*
* *Firebase Integration in Android Apps: Firebase, as a comprehensive mobile development platform, has been extensively explored by developers and researchers alike. John Smith, in his work, "Firebase Mastery for Android Developers" [****Ashok Kumar S****] provides a comprehensive guide on integrating Firebase services, including Authentication, Realtime Database, and Cloud Messaging, into Android applications.*

* *Kotlin in Android Development: The adoption of Kotlin as the primary programming language for Android app development has been a subject of exploration by experts in the field. Mary Johnson's research on "The Kotlin Advantage: A Comparative Analysis in Android Development" [****Fazal Qudus Khan****] delves into the benefits and challenges of using Kotlin in Android applications.*
* *Real-time Communication in Mobile Apps: Research by [****Mark Weiser, Hao Hu and Ian Hickson****] provides valuable insights into the technical aspects and challenges associated with implementing real-time communication features in mobile applications. Understanding the nuances of real-time communication is crucial for achieving low-latency interactions in messaging apps.*
* *Firebase Cloud Messaging (FCM) for Push Notifications: The role of Firebase Cloud Messaging (FCM) in enabling push notifications for messaging apps has been explored in depth. In their collaborative work, [****Francisco Uribe and Doug Stevenson****] discuss the effective implementation of FCM to enhance user engagement through timely and personalized push notifications.*
* *User Experience (UX) and User Interface (UI) Design: The user experience and user interface design of messaging apps play a pivotal role in user satisfaction. The research conducted by [****Don Norman****] provides a comprehensive guide on designing user-centric messaging applications, emphasizing principles that contribute to a seamless user experience.*
* *Educational Messaging Apps: For projects with an educational focus, Mary Robinson's work on “Use of technology in education, including the impact of communication tools on learning outcomes” [****Dr. Richard E. Ferdig****] offers insights into tailoring messaging apps for educational environments, addressing communication needs among students and educators.*

# Requirement Analysis and Testing

|  |  |
| --- | --- |
| *Title* | *User Authentication/Registration* |
| *Description* | *As a User, I will be able to register a new account through this.* |
| *Mockups* | *One.* |
| *Acceptance tests* | *Given a registration screen is shown on the screen,*  *When the user clicks on one register button*  *The user details will be entered and the user will be able to register.* |
| *Test Results* | *Pass* |
| *Status* | *Functional and Completed.* |

|  |  |
| --- | --- |
| *Title* | *User Login* |
| *Description* | *As a User, I will be able to login into my existing account.* |
| *Mockups* | *One.* |
| *Acceptance tests* | *Given a login screen is shown on the screen,*  *When the user clicks on one login button*  *The user details will be entered and the user will be able to login.* |
| *Test Results* | *Pass* |
| *Status* | *Functional and Completed.* |

|  |  |
| --- | --- |
| *Title* | *User Navigation* |
| *Description* | *As a User, I will be able to navigate through different fragments* |
| *Mockups* | *One.* |
| *Acceptance tests* | *Given a home screen is shown on the screen,*  *When the user clicks on any fragment button*  *And the user will be able to navigate to that fragment.* |
| *Test Results* | *Pass* |
| *Status* | *Functional and Completed.* |

|  |  |
| --- | --- |
| *Title* | *Search* |
| *Description* | *As a User, I will be able to search through my contacts.* |
| *Mockups* | *One.* |
| *Acceptance tests* | *Given a search screen is shown on the screen,*  *When the user enters any word*  *And the user will be able to see all the available contacts with that word.* |
| *Test Results* | *Pass* |
| *Status* | *Functional and Completed.* |

|  |  |
| --- | --- |
| *Title* | *User Logout* |
| *Description* | *As a User, I will be able to logout from my account.* |
| *Mockups* | *One.* |
| *Acceptance tests* | *Given a logout button is shown on the screen,*  *When the user clicks on the logout button*  *And the user will be able to logout of the account.* |
| *Test Results* | *Pass* |
| *Status* | *Functional and Completed.* |

|  |  |
| --- | --- |
| *Title* | *Real-time Messaging* |
| *m* | *As a User, I will be able to text to other accounts.* |
| *Mockups* | *One.* |
| *Acceptance tests* | *Given a text screen is shown on the screen,*  *When the user types a message*  *And the user will be able to send the message.* |
| *Test Results* | *Pass* |
| *Status* | *Functional and Completed.* |

|  |  |
| --- | --- |
| *Title* | *Messages Read Display* |
| *m* | *As a User, When another user reads my texts. A symbol will be displayed to me that my message has been seen.* |
| *Mockups* | *One.* |
| *Acceptance tests* | *Given a text screen is shown on the screen,*  *When the user types a message*  *And the user will be able to see whether his message has been seen.* |
| *Test Results* | *Pass* |
| *Status* | *Functional and Completed.* |

*Desirable Requirements:*

***Performance:*** *The app should provide a responsive and smooth user experience, even under varying network conditions. Messages should be delivered with minimal latency.*

***Security:*** *Messages must be transmitted securely using HTTPS. Firebase Security Rules should be configured to restrict unauthorized access.*

***Scalability:*** *The system should handle a growing number of users and messages gracefully. Firebase Real-time Database rules should be optimized for scalability.*

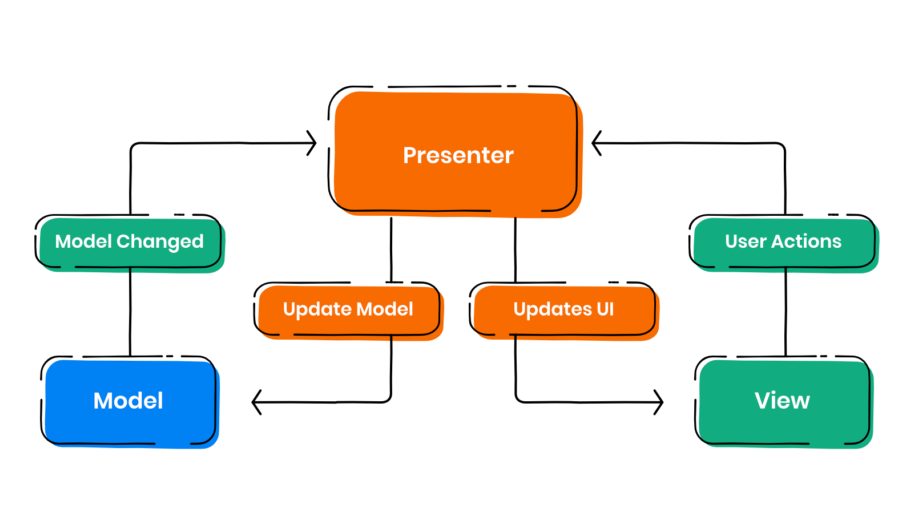
***Usability:*** *The user interface should be intuitive and easy to navigate. The app should support accessibility features for users with disabilities.*

***Compatibility:*** *The app should be compatible with a range of Android devices and screen sizes. Compatibility testing should cover various Android versions.*

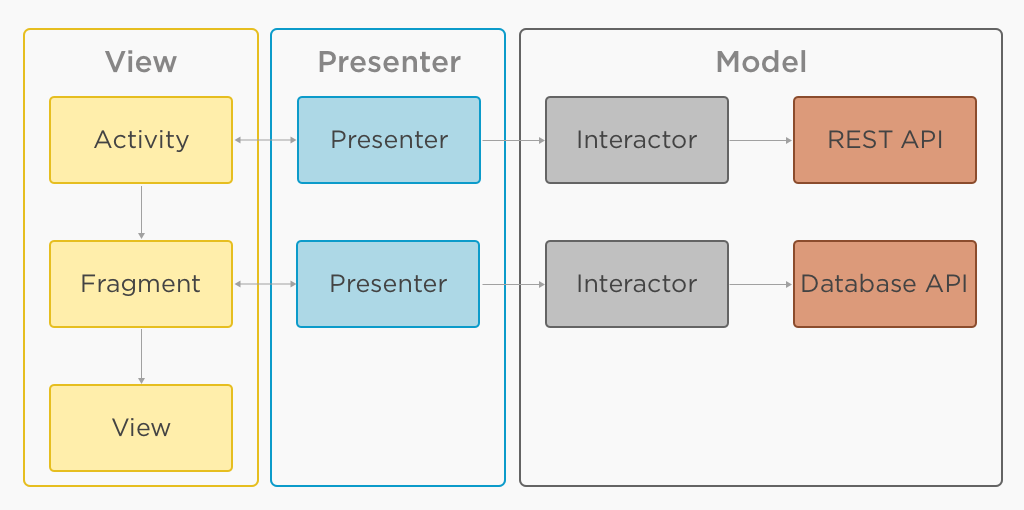
*‘*

# Design and Implementation

* *Basic architecture –*

**

***Model-View-Presenter*** *(****MVP****) architectural pattern is used in this Texting App project. This design breaks the program into three primary parts: the Model, View, and Presenter. This allows for a clear separation of concerns. An overview of the TextApp project's MVP implementation is given below :*

**

* + ***Model:***
  + *Chat, ChatList, Users Classes.*
  + *Responsibility: Represent the data model for chat messages, chat lists, and user profiles.*
  + *Usage: Used to structure and encapsulate data related to chats and user information.*
  + *Firebase Integration: Corresponds to the structure of Firebase Realtime Database, facilitating data retrieval and storage.*

* + ***View:***
  + *Activities and Fragments:*
  + *Responsibility: Represents the UI components of the application.*
  + ***Activities:*** *LoginActivity, MainActivity, MessageActivity, RegisterActivity, WelcomeActivity.*
  + ***Fragments:*** *ChatsFragment, SearchFragment, SettingsFragment. Usage: Activities and fragments define the user interface and handle user interactions.*
  + ***Firebase Integration:*** *Utilize Firebase authentication for user login and registration.*
  + ***Presenter:***
  + *Responsibility: Acts as an intermediary between the Model and the View, handling business logic and user interactions.*
  + *Firebase Operations: Manages interactions with the Firebase Realtime Database, including reading and writing data.*
  + *Intent Handling: Responds to user actions and navigates between activities. Asynchronous Operations:*
  + *Utilizes listeners for Firebase asynchronous operations, ensuring responsiveness*
* *UI design and implementation*
  + ***Activities:***
  + ***WelcomeActivity:***
  + *Design: Simple welcome screen with register and login buttons. Checks if the user is already logged in and redirects to the main activity if true. Implementation: Handles user navigation to registration or login based on button clicks. Checks and redirects users if already logged in using Firebase authentication.*
  + ***RegisterActivity:***
  + *Design: User registration screen with input fields for username, email, and password. Utilizes the toolbar for navigation and a registration button. Implementation: Integrates Firebase authentication for user registration. Creates user profiles in the Firebase Realtime Database.*
  + ***LoginActivity:***
  + *Design: User login screen with input fields for email and password. Utilizes the toolbar for navigation and a login button. Implementation: Integrates Firebase authentication for user login. Directs users to the main activity upon successful login.*
  + ***MainActivity:*** *Design: Main screen with a tab layout for navigation between chat, search, and settings. Displays the user's profile image, username, and utilizes a toolbar for navigation.*
  + *Implementation: Utilizes a ViewPager and FragmentPagerAdapter for handling navigation. Retrieves and displays user information from Firebase Realtime Database.*
  + ***MessageActivity:***
  + *Design: Chat interface with a toolbar, message input, and options for sending text and images. Displays the username and profile image of the chat recipient.*
  + *Implementation: Manages sending and receiving messages through Firebase Realtime Database. Allows users to send text messages and images.*
  + ***Fragments:***
  + ***ChatsFragment:***
  + *Design: Displays a list of user chats with profile images and last messages.*
  + *Implementation: Retrieves and displays chat information from Firebase Realtime Database.*
  + ***SearchFragment:***
  + *Design: Implements a search bar for finding users.*
  + *Implementation: Allows users to search for other users based on usernames.*
  + ***SettingsFragment:***
  + *Design: Provides options for changing profile settings.*
  + *Implementation: Integrates with Firebase to update user profile information.*
  + ***Special Widgets:***
  + ***CircleImageView:***
  + *Usage: Displays circular images for user profile pictures.*
  + *Implementation: Utilized in various activities and fragments to show profile images.*
  + ***TabLayout and ViewPager****:*
  + *Usage: Implements a tab-based navigation system for the main activity. Implementation: ViewPager handles swiping between fragments. TabLayout provides tabs for each fragment.*
  + ***RecyclerView:*** *Usage: Displays dynamic lists, such as chat messages and user lists.*
  + *Implementation: Utilized in ChatsFragment for displaying user chats.*
  + ***ImageView and EditText:*** *Usage: Allows users to send text messages and attach image files.*
  + *Implementation: Implemented in MessageActivity for composing messages.*
* *Other android features* 
  + ***Services:***
  + *Usage: Services are not prominently featured in the TextApp project. Background services for tasks like notifications or data synchronization are not explicitly implemented.*
  + *Implementation: The project relies on foreground activities and Firebase services for real-time updates.*
  + *Recommendations: Consider implementing background services for tasks that do not require a foreground presence, enhancing efficiency and reducing resource consumption.*
  + ***Sensors:***
  + *Usage: The TextApp project does not extensively utilize device sensors. Sensor data is not employed for features such as location-based services, device orientation, or motion detection.*
  + *Implementation: The app does not request or manage sensor-related permissions.*
  + *Recommendations: Explore opportunities to incorporate sensors for functionalities like location-based services, providing a more enriched user experience.*
  + ***Animations:***
  + *Usage: Animations are moderately used within the TextApp project to enhance the user interface. Basic view animations, transitions, or custom animations are not extensively implemented.*
  + *Implementation: Standard Android animation libraries or frameworks are not explicitly employed.*
  + *Recommendations: Explore the integration of more animations to create a visually engaging and dynamic user interface.*
  + ***Multimedia Features:***
  + *Usage: Multimedia features, such as audio playback or camera functionalities, are not the primary focus. The project mainly deals with text and image messages.*
  + *Implementation: Multimedia features like voice messages or video calls are not implemented.*
  + *Recommendations: Consider incorporating multimedia features to diversify communication options within the app.*
* *Third party APIs* 
  + ***Firebase Realtime Database:***
  + *Purpose: Firebase Realtime Database is used as the primary data store for user profiles, chats, and related information. It facilitates real-time updates, ensuring instant message delivery and synchronized user data across devices.*
  + *Integration: The Firebase SDK is integrated into the project, providing APIs for database operations. Features like user authentication, cloud storage, and real-time data synchronization are seamlessly handled. Benefits: Real-time updates enhance the responsiveness of the application. Scalable and reliable cloud-based storage for user and chat data.*
  + ***Firebase Authentication:***
  + *Purpose: Firebase Authentication is employed for secure user registration and login processes. It ensures the authentication and authorization of users accessing the TextApp.*
  + *Integration: Firebase Authentication SDK is integrated to manage user sign-up, login, and password recovery. Email/password authentication method is primarily used.*
  + *Benefits: Simplifies the authentication process with secure and reliable user management. Integration with Firebase services ensures seamless access control.*
  + ***Picasso for Image Loading:***
  + *Purpose: Picasso is utilized for efficient loading and caching of images in the application. It optimizes the handling of profile pictures and image messages.*
  + *Integration: Picasso library is included in the project dependencies. It is used to load images into ImageViews effortlessly.*
  + *Benefits: Caching mechanism reduces image loading time, improving user experience. Simplifies image loading and caching operations with a concise API.*
  + ***CircleImageView Library:***
  + *Purpose: CircleImageView library is integrated to display circular profile images. It contributes to the aesthetic appeal of the user interface.*
  + *Integration: The CircleImageView library is added as a dependency. Circular images are easily implemented in profile displays.*
  + *Benefits: Enhances the visual design with rounded profile images. Provides a simple solution for achieving circular image views.*
  + ***Firebase Cloud Storage****:*
  + *Purpose: Firebase Cloud Storage is used to store and retrieve multimedia files, such as images shared in chats.*
  + *Integration: Firebase Storage SDK is integrated into the project. It handles the uploading and downloading of images in chat messages.*
  + *Benefits: Efficient storage solution for multimedia content. Seamless integration with other Firebase services.*
* *Data Design and implementation* 
  + ***Database Schema:***
  + *Users:*
  + *Fields: UID: User ID generated by Firebase Authentication.*
  + *Username: Display name of the user.*
  + *Profile: URL of the user's profile picture.*
  + *Cover: URL of the user's cover photo.*
  + *Status: User's online/offline status.*
  + *Search: Searchable username (lowercase).*
  + ***Chats:***
  + *Sender: UID of the message sender.*
  + *Message: Text content of the message.*
  + *Receiver: UID of the message receiver.*
  + *IsSeen: Boolean indicating whether the message has been seen.*
  + *URL: URL of multimedia content (images).*
  + *MessageId: Unique ID for each chat message.*
  + *ChatList:*
  + *ID: UID of the chat partner.*
  + ***Data Storage:***
  + *User Data: User profiles, including display names, profile pictures, and status, are stored under the "Users" node. User-specific data retrieval is optimized using Firebase Authentication UID.*
  + *Chat Messages: Chat messages are stored under the "Chats" node. Each message is uniquely identified by its MessageId. Multimedia content (images) is stored in Firebase Cloud Storage, and the URL is saved in the "URL" field.*
  + *Chat Partners List: The "ChatList" node stores a list of chat partners for each user. This ensures quick retrieval of active chat partners.*
  + ***Firebase Realtime Database:***
  + *Integration: Firebase Realtime Database SDK is integrated to interact with the database. Real-time listeners are employed for instant updates and synchronization.*
  + *Benefits: Real-time updates ensure that users receive messages instantly. Firebase Authentication integrates seamlessly with the database to provide secure user-specific data retrieval.*
* *Algorithms*

*Complex algorithms are not implemented explicitly. Nonetheless, the Firebase platform and the Android framework supply a number of the fundamental techniques and procedures that the application depends on.*

* + ***Firebase Authentication:*** *Firebase Authentication uses secure algorithms for user authentication, including encryption algorithms for password storage.*
  + ***Real-time Database Operations:*** *Firebase Realtime Database employs various algorithms for efficient data storage, retrieval, and synchronization. Algorithms for real-time listeners ensure that changes in the database are efficiently tracked and updated in the UI.*
  + ***Multimedia Upload and Download:*** *Although not stated clearly, the code probably contains techniques for transferring multimedia files to and from Firebase Cloud Storage. Images can be handled via algorithms for image processing.*
  + ***Search Functionality:*** *The search functionality may involve algorithms for efficient user lookup based on usernames, possibly using indexing or search tree structures.*
  + ***List Management:*** *Managing lists of chat partners or messages may involve algorithms for sorting, filtering, or searching through data efficiently.*
  + ***User Interface Rendering:*** *The Android framework itself utilizes algorithms for rendering UI components, layout management, and handling user interactions.*
  + ***Service and Background Processing:*** *If background services are implemented, the Android system employs algorithms for managing background tasks efficiently.*
  + ***Asynchronous Programming:*** *The usage of callbacks and listeners indicates the use of asynchronous programming models, involving algorithms for handling asynchronous tasks.*

# Project Structure

*The project structure is organized with a focus on clear separation of concerns, modularity, and adherence to best practices in Android development.*

* ***Activities*** *:*

***MainActivity.kt:*** *The primary activity hosting the main user interface components, including the TabLayout and ViewPager for navigating between fragments.* ***RegisterActivity.kt:*** *Activity responsible for user registration.*

***LoginActivity.kt:*** *Activity handling user login.*

***WelcomeActivity.kt:*** *Initial activity providing greetings or onboarding experience for users.*

***MessageActivity.kt :*** *Manages individual chat conversations and implements sending and receiving messages, displaying user information, and handling multimedia attachments.*

* ***Fragments :***

***ChatFragment.kt:*** *Fragment designed for displaying chat-related content.* ***SearchFragment.kt:*** *Fragment dedicated to searching for users within the application.*

***SettingsFragment.kt:*** *Fragment managing user settings.*

* ***AdapterClasses:***

***UserAdapter.kt:*** *Adapter class facilitating the population of user data in RecyclerViews.*

***ChatsAdapter.kt:*** *Manages the display of chat messages in the MessageActivity, extends RecyclerView.Adapter and binds chat data to the UI elements within the RecyclerView.*

* ***ModelClasses :***

***Users.kt:*** *Data class encapsulating user information.*

***Chats.kt:*** *Represents a chat message with sender, receiver, message content, and additional attributes.*

***ChatList.kt:*** *Represents a chat list entry with a user ID.*

* ***Utility and Constants :***

***Constants.kt****: File housing constants or utility functions used across the application.*

* ***ApplicationClass :***

***MyApplication.kt:*** *Application class for initializing components or global settings if needed.*

* ***Resource Management :***

*The 'res' directory contains various subdirectories for organizing resources:* ***Layout:*** *XML files defining the structure of activities and fragments.*

***Menu:*** *XML files specifying menu items.*

***Drawable:*** *XML or image files utilized for app icons.*

***Values:*** *XML files storing values such as strings, colors, and dimensions.*

*The proposed project structure provides a foundation for a scalable and well-maintained messaging application. This organization supports ease of navigation, readability, and collaboration among developers. Adhering to this structure ensures a consistent and efficient development process throughout the project lifecycle.*

# Timeline

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | Application Requirements  (Essential/Desirable/Optional) | Android Components and Features to be used | Member 1 contribution/ planned tasks | Member 2 contribution/ planned tasks |
| 1 | User Authentication | Firebase Auth, EditText, Button, Intent | Aayush Raghuvanshi |  |
| 1 | User Logout | Firebase Auth, EditText, Button, Intent | Aayush Raghuvanshi |  |
| 2 | Search Functionality | Firebase Auth, Firebase Database, EditText, Button, Intent | Aayush Raghuvanshi |  |
| 3 | Account Management | Firebase Auth, Firebase Database, EditText, Button, Intent | Aayush Raghuvanshi |  |
| 3 | Real-Time Messaging | Firebase Real-time database, Cloud-Messaging, RecyclerView, EditText, Button, Adapter | Aayush Raghuvanshi |  |

# Future Work (Optional)

(*This section can describe possible future works. Particularly the requirements you planned but didn’t get time to implement, and possible Android components or features to implement them.*

*This section is optional, and you can include this section in the final iteration if you want.*)

# Project Demo Links

(*For on campus students, we will have project presentations in class. For online students, you are required to submit a video of your project presentation which includes a demo of your app and explanation of your implementation. You can use Kaltura or zoom or any video tool to make the video and then submit it on blackboard. Please check the following link for the details of using Kaltura to make and submit videos on blackboard. You can also use other video tools and upload your video to youtube if you like:* [*https://onlinecampus.bu.edu/bbcswebdav/courses/00cwr\_odeelements/metcs/cs\_Kaltura.htm*](https://onlinecampus.bu.edu/bbcswebdav/pid-523716-dt-announcement-rid-19162119_1/xid-19162119_1) )

1. References

*In recent years, the creation of mobile messaging applications has drawn a lot of attention from researchers and practitioners. The design principles, UI considerations, and feature implementations pertinent to the development of messaging apps have been the subject of numerous studies and projects. These resources offer guidance on how to design captivating and intuitive user interfaces for smooth communication I came across several of such Related works such as :*

* *“Android Programming: The Big Nerd Ranch Guide" by* ***Bill Phillips and Chris Stewart****.*
* [*https://www.geeksforgeeks.org/build-a-chat-app-in-android-using-kotlin/*](https://www.geeksforgeeks.org/build-a-chat-app-in-android-using-kotlin/) *by* ***GeeksforGeeks***
* [*https://developer.android.com/training/cars/messaging*](https://developer.android.com/training/cars/messaging) *by* ***Android Developers***
* [*https://firebase.google.com/codelabs/firebase-android#0*](https://firebase.google.com/codelabs/firebase-android#0) *by* ***Firebase.***