Ardent Computech Pvt. Ltd.
Drives you to the Industry
Module 132, SDF Building, Sector V, Salt Lake, Pin - 700091
www.ardentcollaborations.com

MECHAT - MESSAGING APP
ANDROID DEVELOPMENT USING JAVA

A Project Report for Industrial Training and Internship

submitted by

ANKIT RAJ,
MOHOR BANERJEE
SAYAN SHIL
AKASH KUMAR
PRASUN JHA ,
SAKET KUMAR
ABHIJEET GAURAV
SOHAM PAUL
SOURABH KUMAR

Under the guidance of
ANINDYA MUKHERJEE

In the partial fulfilment of the award of the degree of
B-TECH

in the
COMPUTER SCIENCE AND ENGINEERING Dept.
of
NARULA INSTITUE OF TECHNONLOGY

at
Ardent Computech Pvt. Ltd.

ARDENT GROUP OF COMPANIES
INTERNSHIP | DEVELOPMENT | PROJECT
An ISO 9001 : 2015 Certified Company

CERTIFICATE FROM SUPERVISOR

This is to certify that **ANKIT RAJ, AKASH KUMAR, SAYAN SHIL, MOHOR BANERJEE, PRASUN JHA, SAKET KUMAR, SOURABH KUMAR, ABHIJEET GOURAV, SOHAM PAUL** have completed the project titled "MECHAT - MESSAGING APP ANDROID DEVELOPMENT USING JAVA" under my supervision during the period from "04.02.2025" to "12.02.25" which is in partial fulfillment of requirements for the award of the **B-TECH** degree and submitted to the Department of "**COMPUTER SCIENCE AND ENGINEERING DEPARTMENT**" of **"NARULA INSTITUE OF TECHONLOGY"**.

_____

Signature of the Supervisor

Date:

Name of the Project Supervisor: ANINDYA MUKHERJEE

BONAFIDE CERTIFICATE

Certified that this project work was carried out under my supervision

"ANDROID DEVELOPMENT USING JAVA"is the bonafide work of

Name of the student: ANKIT RAJ

Signature:

Name of the student: SAYAN SHIL

Signature:

Name of the student: AKASH KUMAR

Signature:

Name of the student: MOHOR BANERJEE

Signature:

Name of the student: SAKET KUMAR

Signature:

Name of the student: SOURABH KUMAR

Signature:

Name of the student: PRASUN JHA

Signature:

Name of the student: ABHIJEET GOURAV

Signature:

Name of the student: SOHAM PAUL

Signature:

PROJECT MENTOR

SIGNATURE:

Name: Mr. ANINDYA MUKHERJEE

EXAMINERS

Ardent Original Seal

ACKNOWLEDGEMENT

The successful completion of any task is marked by recognizing the individuals whose unwavering support made it achievable. Their continuous guidance and motivation were instrumental in the success of our endeavors.

We would like to take this moment to extend our heartfelt appreciation to our project mentor, Mr. ANINDYA MUKHERJEE, for his invaluable suggestions, guidance, and encouragement throughout the development of this project.

Additionally, we express our sincere gratitude to all the faculty members of Ardent Computech Pvt. Ltd. for their unwavering support.

ABSTRACT OF THE PROJECT

This project focuses on the development of a secure and user-friendly messaging application designed for real-time communication. The app provides essential messaging features, including secure login and signup functionality, private one-on-one chats, user profile customization, and the ability to upload profile pictures via the integrated camera. The data, including user details and chat history, is securely stored in a **Firebase database**, ensuring real-time synchronization and data security.

The app offers a straightforward, intuitive interface, allowing users to easily navigate through its features and manage their settings, including notifications and privacy preferences. Additionally, the app supports a **logout** option for enhanced security. By utilizing **Firebase Authentication** and **Firestore**, the app ensures that user data is protected and updated instantly across devices.

This messaging app is built using **JAVA** to support both Android platforms, ensuring cross-platform compatibility. The project emphasizes secure data storage, efficient messaging, and user privacy, with future plans to include additional features such as group chats, voice/video calling, and a web version. The app aims to offer a reliable and personalized communication solution for users while maintaining high standards of security and privacy.

CONTENTS

INTRODUCTION

Introducing our innovative messaging app, crafted to provide a modern, secure, and user-friendly communication platform. This app is designed to cater to the needs of users looking for a seamless messaging experience while ensuring privacy and convenience at all times.

One of the standout features of the app is the Login and Signup pages, which provide a straightforward method for users to create accounts and access the platform. The Signup page allows new users to quickly register by entering their details, while the Login page ensures that existing users can easily access their accounts using their credentials. The process is secure, designed to protect user data and privacy, and guarantees that only authenticated users can access the messaging service.

Upon logging in, users can easily connect with others using the app's core feature: the chatting functionality. This allows users to engage in private conversations with a variety of people in real-time, making it ideal for both personal and professional communication. The intuitive interface ensures that users can send messages, share media, and communicate without any interruptions. With a focus on simplicity, the chatting feature is designed for ease of use, ensuring smooth and instant communication.

The app also includes a logout option, giving users complete control over their sessions. With just a click, users can log out of the app, ensuring their account is secure when not in use. This feature is especially useful for shared devices or when users need to ensure their account stays protected from unauthorized access.

For a more personalized experience, the app comes equipped with a camera option. This feature enables users to set a new profile picture, adding a personal touch to their profiles. Whether for social or professional purposes, the ability to update profile images ensures that users can express themselves fully and keep their profiles fresh.

Furthermore, users can access the Settings section, where they can customize various aspects of the app. Whether adjusting notification preferences or modifying account details, the settings allow for complete personalization and control over the user experience.

In terms of data management, all user information and interactions are securely stored in a Firebase database. Firebase offers a powerful backend solution that ensures real-time synchronization of data, meaning messages are delivered instantly across devices. Additionally, Firebase provides robust security, ensuring that user data, including profile details, chat history, and media files, are encrypted and stored safely.

With a focus on ease of use, security, and customization, this messaging app offers everything users need to stay connected, secure, and in control of their communication. Whether you're looking to chat with friends, colleagues, or new contacts, this app provides a fully integrated solution that guarantees a smooth, efficient, and personalized experience.

FEASIBILITY STUDY

A feasibility study for the messaging app involves analyzing various aspects to determine if the project is viable, both in terms of technical development and business potential. Below is an outline of the feasibility study focusing on different key areas:

1. Technical Feasibility

Technical feasibility evaluates if the technology required to build the app is available, reliable, and scalable. Here's how it breaks down:

- **Technology Stack: The app relies on front-end technologies such as React Native or Flutter for cross-platform development (iOS and Android), ensuring that it can reach a wider audience. The back-end system uses Firebase for database management and user authentication, which simplifies the development process and ensures real-time messaging synchronization.**
- **Data Storage: Firebase offers a reliable cloud-based database, which is scalable and secure. Firebase Authentication and Firestore will manage user data and chat history effectively, with real-time syncing for messages. Firebase's data security measures (such as encryption and access control) will ensure the app is secure and protects user data.**
- **Camera Integration: The app integrates camera functionality for users to set profile pictures. This is achievable using native device camera APIs, and cross-platform frameworks such as React Native already have libraries like react-native-camera for implementing such features.**
- **Scalability: Firebase offers scalability by handling a high number of concurrent users without compromising performance. As the user base grows, Firebase can scale dynamically, so long as costs are managed.**
- **User Interface: Creating an intuitive and user-friendly interface with easy-to-navigate features is entirely feasible, with plenty of UI libraries available for both Android and iOS to meet aesthetic and functional requirements.**

2. Operational Feasibility

Operational feasibility checks if the app can be successfully operated once developed:

- **User Experience: With features like user authentication, real-time chat, profile management, camera integration for profile pictures, and settings customization, the app offers a comprehensive user experience. The app's functionality is common in messaging apps, meaning users will find it easy to navigate.**

- **Maintenance: Firebase's backend solution reduces the burden on maintaining servers or databases, as Firebase takes care of scalability and reliability. Regular updates can be pushed through the app store for bug fixes or new features.**

- **Data Privacy and Compliance: Storing data securely in Firebase ensures compliance with data protection regulations like GDPR (General Data Protection Regulation) and CCPA (California Consumer Privacy Act). Implementing features such as account deletion and data export helps comply with these regulations.**

- **User Support: With in-app support or FAQs, users can get help for technical issues or feature-related queries. A support team can be set up for larger-scale operations.**

3. Economic Feasibility

Economic feasibility evaluates the costs and potential return on investment (ROI):

- **Development Costs: Building the app involves upfront costs for development, UI/UX design, and integration of Firebase. Using cross-platform tools (like React Native or Flutter) reduces development costs by creating one codebase for both iOS and Android. Firebase's free tier can handle a small user base, but additional costs will arise as the user base grows due to data storage and messaging volume.**

- **Monetization: Possible revenue streams include:**

  - **In-app Ads: Ads can be displayed to users, providing a steady revenue stream.**

  - **Premium Features: Offering features like custom themes, extra storage, or enhanced security for a premium subscription.**

  - **Partnerships/Branding: Collaboration with other companies or influencers to promote their brand within the app.**

- **Operational Costs: Firebase charges based on usage, so costs will scale with the number of active users. As the app grows, you may need to factor in expenses for server scaling, maintenance, and user support.**

- **ROI Potential: With increasing demand for messaging apps and a focus on secure, private communication, the app has a strong potential for attracting users. By offering a seamless experience with minimal barriers to entry, it could capture a sizable user base, leading to profitability through ads or premium subscriptions.**

4. Legal Feasibility

Legal feasibility ensures that the app complies with laws and regulations:

- **User Data Protection: The app must comply with privacy regulations like GDPR and CCPA. Firebase offers built-in features for security and user privacy, but it is essential to implement user consent mechanisms, data encryption, and clear privacy policies within the app.**
- **Copyright: The app must ensure that no third-party copyrighted content (such as images, videos, or other media) is used without permission. Any integrations with third-party services should have clear agreements and licensing to avoid infringement.**
- **Terms of Service and End-User License Agreements (EULA): A legal framework is required to protect both the users and developers. This includes outlining the terms of use, privacy policies, and user rights within the app.**

5. Market Feasibility

Market feasibility assesses if there is a demand for this app:

- **Target Market: The app targets a broad audience, ranging from teenagers and young adults to professionals who require secure, efficient communication. Messaging apps have proven demand globally, especially in regions with a high smartphone penetration rate.**
- **Competitive Analysis: The messaging app market is competitive, with major players like WhatsApp, Telegram, and Signal dominating the space. However, by offering unique features such as seamless integration with Firebase, camera options for profile images, and personalized settings, the app can differentiate itself by focusing on privacy, security, and real-time features.**
- **User Acquisition: To attract users, the app could leverage social media marketing, referral programs, or partnerships with influencers. Early-stage marketing efforts could include offering the app for free with basic features and then introducing premium features over time.**

6. Risk Feasibility

Risk feasibility involves analyzing potential risks:

- **Privacy and Security Risks: As messaging apps often handle sensitive data, there is an inherent risk of data breaches or security vulnerabilities. Implementing end-to-end encryption and ensuring compliance with data security standards can mitigate this risk.**
- **User Retention: With competition from established apps, user retention could be a challenge. To address this, continuous updates, feature enhancements, and a personalized user experience can keep users engaged.**
- **Scalability: If the app experiences rapid growth, it may face challenges in terms of database storage, server capacity, and costs. Ensuring that Firebase's scaling options are leveraged effectively will mitigate such risks.**

Conclusion

The feasibility study indicates that the messaging app is technically viable, economically sustainable, and legally compliant. By using reliable technologies like Firebase and focusing on a user-friendly experience, the app has the potential to succeed in a competitive market. However, careful consideration must be given to privacy, security, and scalability to ensure long-term success. With an effective monetization strategy and continuous updates, the app can achieve both user growth and profitability.

Scope of the Project

The **scope of the messaging app project** outlines the objectives, features, functionalities, and limitations of the app, defining what will be included and what will not. It serves as a roadmap for development and helps stakeholders understand the extent and boundaries of the project. Below is an in-depth look at the scope of the project:

1. Objectives of the Project

- **Develop a Secure Messaging Platform**: Create an intuitive and secure messaging app that allows users to communicate easily and privately with others in real-time.
- **User-Friendliness**: Ensure the app has a simple, easy-to-navigate user interface (UI) and experience (UX), minimizing barriers for new users.
- **Real-Time Communication**: Provide real-time messaging with instant delivery, as well as features like text, images, and profile customization.
- **Data Security and Privacy**: Prioritize user data security through encryption, secure logins, and strict data storage protocols, using Firebase as the backend to ensure privacy.

2. Features Included in the Project Scope

- **User Authentication (Login/Signup)**:
  - Users can register (sign-up) for the app by creating an account using an email, phone number, or other authentication methods.
  - Users can log in to access their accounts securely.
  - Password recovery and two-factor authentication can be implemented for enhanced security.
- **User Profiles**:
  - Users can create and update their profiles with personal information such as names and profile pictures.
  - Camera functionality allows users to upload a profile picture directly from the device's camera or gallery.
- **Chatting Functionality**:
  - Users can send and receive real-time messages with individuals.
  - The app will support text-based conversations, and users will be able to send images, emojis, and possibly voice messages.

    - Message history will be stored, so users can review previous conversations.
- **User Settings**:
    - Users can access a settings page where they can update their account details, change privacy settings, and manage notifications.
    - Options to manage security settings, such as password changes and data privacy preferences, will be available.
- **Logout Option**:
    - A secure logout feature will allow users to safely end their session and ensure the app is not left open on shared or public devices.
- **Real-Time Database**:
    - All user data, including messages and profile information, will be stored in a **Firebase database** for fast access and synchronization across devices.
    - Data will be stored securely, with measures like encryption to protect sensitive user information.

3. <u>Limitations and Exclusions</u>

- **Group Chat Functionality**: Group chats or multi-party conversations are not included in the initial version of the app. This feature may be considered in future updates.
- **Advanced Features**: Features such as video calling, voice calling, or screen sharing are not included in the first release. These can be considered for future versions.
- **Web Version**: The current scope is limited to mobile platforms (Android). A web and IOS version may be considered for future versions but will not be included in the initial release.
- **Advanced Customization**: Although users will be able to upload profile pictures and update basic settings, more advanced customization options (e.g., custom themes, backgrounds, etc.) will not be part of the initial scope.
- **In-App Payments/Monetization**: The current scope excludes any in-app purchases, subscriptions, or monetization features like ads. Future versions may include premium features or monetization.
- **AI Features**: While there are no AI-based features like smart replies or message filtering in the initial release, these could be part of a later version.

4. <u>Timeline and Phases of Development</u>

- **Phase 1 - Planning and Design**:

    - Finalizing the app's requirements, defining the user interface (UI) and user experience (UX) designs.
    - Creating wireframes and prototypes to visualize the layout and flow of the app.
- **Phase 2 - Development**:
    - Setting up Firebase for user authentication and data storage.
    - Implementing core features such as login, sign-up, chatting functionality, and profile management.
    - Integrating camera functionality for profile pictures.
- **Phase 3 - Testing**:
    - Performing alpha and beta testing to identify bugs, performance issues, or UI/UX improvements.
    - Testing app functionality on different devices and platforms (iOS and Android).
    - User testing to gather feedback and make necessary improvements.
- **Phase 4 - Launch**:
    - Launching the app on the Google Play Store and Apple App Store.
    - Marketing and promoting the app to attract initial users.
- **Phase 5 - Post-Launch Support and Updates**:
    - Providing maintenance and updates to fix bugs, improve performance, and add minor features.
    - Monitoring user feedback and responding to any issues or suggestions.

5. <u>Technology Stack</u>

- **Frontend Development**:
    - **XML Layout**

  Backend:
    - **Firebase** for authentication, real-time database management, and cloud storage.
    - **Java**
- **Database**:
    - **Cloudinary** for real-time data storage.
    - **Firebase Authentication** for managing user credentials and login functionality.
- **Other Technologies**:
    - **Camera API** for profile picture functionality.

6. Future Considerations

- **Group Messaging**: Adding the ability for users to create and participate in group chats.
- **Voice and Video Calls**: Implementing voice and video calling functionality for richer communication experiences.
- **Web Platform**: Expanding the app's availability to a web version.
- **Premium Features**: Introducing in-app purchases, subscription models, or ad-based revenue streams.
- **Advanced Security Features**: Adding end-to-end encryption, two-factor authentication, and other security measures.

Conclusion

The scope of this project defines a messaging app focused on simplicity, security, and real-time communication, utilizing Firebase for seamless back-end integration. The app will be designed to cater to mobile users, offering essential features like secure logins, chatting, profile customization, and privacy settings. While the initial version will have a limited set of features, there is potential for future enhancements, including group chat, voice/video calling, and a web version. The project is designed with scalability and user privacy in mind, setting the stage for a future-proof solution.

Hardware and Software Requirements

Hardware Requirements:

1. Development Machine:
    - Processor: Intel Core i3 or equivalent
    - RAM: 8 GB minimum (16 GB recommended)
    - Storage: 256 GB SSD minimum (512 GB recommended)
    - Graphics: Integrated or dedicated graphics with support for HD resolution
    - Internet Connection: Stable high-speed connection for downloading dependencies and updates
2. Testing Devices:
    - Smartphones/Tablets: Android devices with at least Android 7.0 or higher for testing app functionality

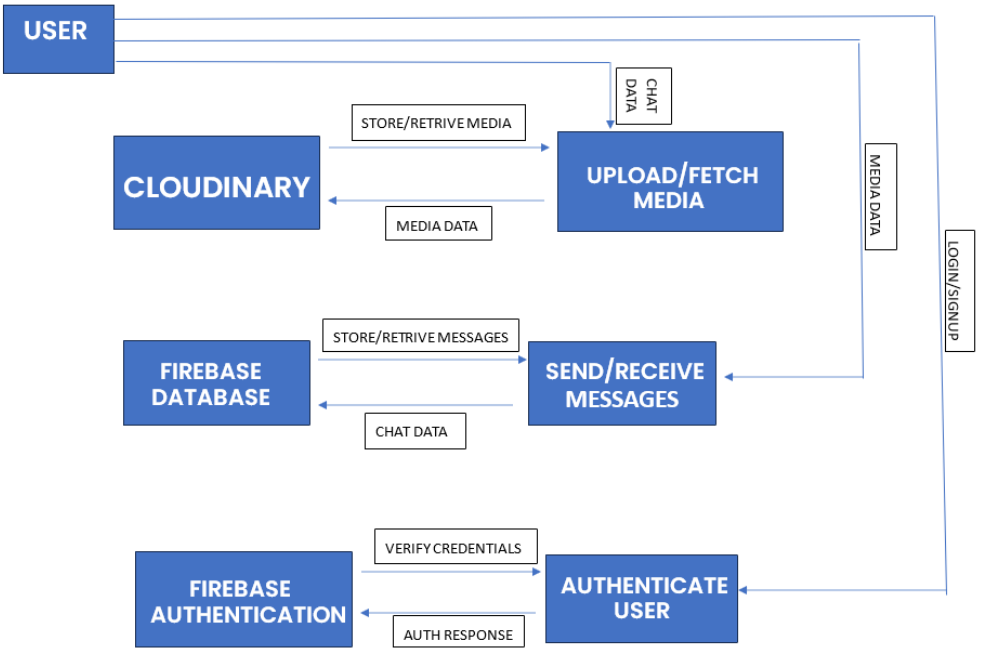
Software Requirements:

1. Development Tools:
    - IDE: Android Studio
    - Build Configuration Language: Java
2. Programming Languages:
    - Frontend: XML for UI design ○ Backend: Java for application logic
3. Firebase Services:
    - Firebase Authentication: For secure user login and registration
    - Cloud Firestore: For real-time database management
    - Cloudinary: For storing images and other files
4. External Dependencies:
    - Firebase BOM: For managing Firebase dependencies
    - Firebase Analytics: For app usage analytics
    - AppCompat: For backward-compatible UI elements
    - Material Components: For modern UI components
    - Activity: For activity lifecycle management
    - ConstraintLayout: For flexible and efficient layout design
    - Firebase Auth: For authentication
    - Firebase Firestore: For database access
    - Picasso: For Loading Images
    - Cloudinary: For file storage
    - Glide: For image loading and caching
    - Glide Compiler: For Glide annotation processing
    - JUnit: For unit testing
    - JUnit Extensions: For extended testing functionalities

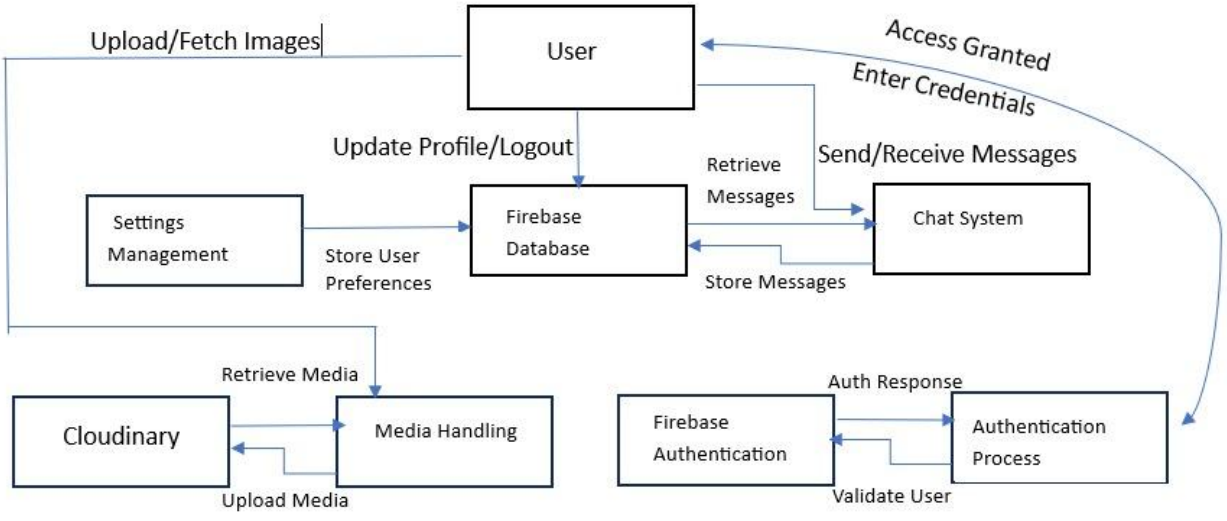
    - Espresso Core: For UI testing
5. Testing Frameworks:
    - JUnit: For unit testing
    - Espresso: For UI testing

These requirements ensure a robust development environment for building and testing the MeChat app, leveraging the latest tools and technologies for optimal performance and functionality.

Data Flow Diagram of the MeChat Application



Level 0 Data Flow Diagram



Level 1 Data Flow Diagram

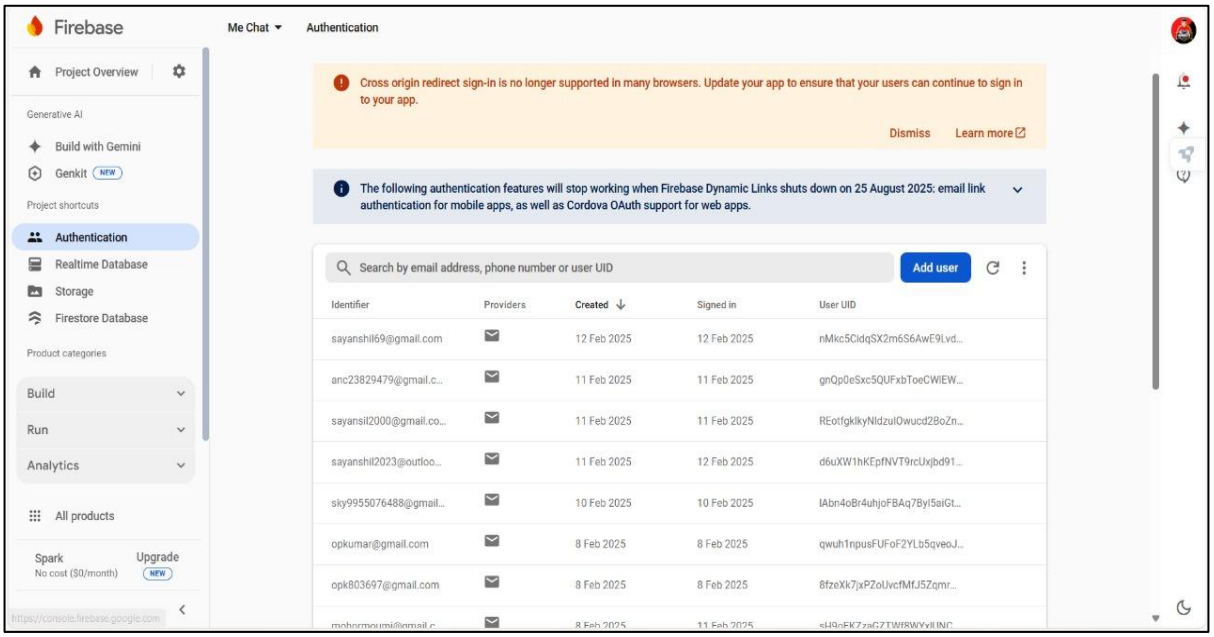
The Level 1 Data Flow Diagram (DFD) for the "MeChat" application outlines the primary processes, data stores, and the flow of information within the system. It provides a high-level overview of how user interactions, data management, and system integrations work together to deliver a seamless chat experience. Below is a detailed breakdown of the key components and processes:

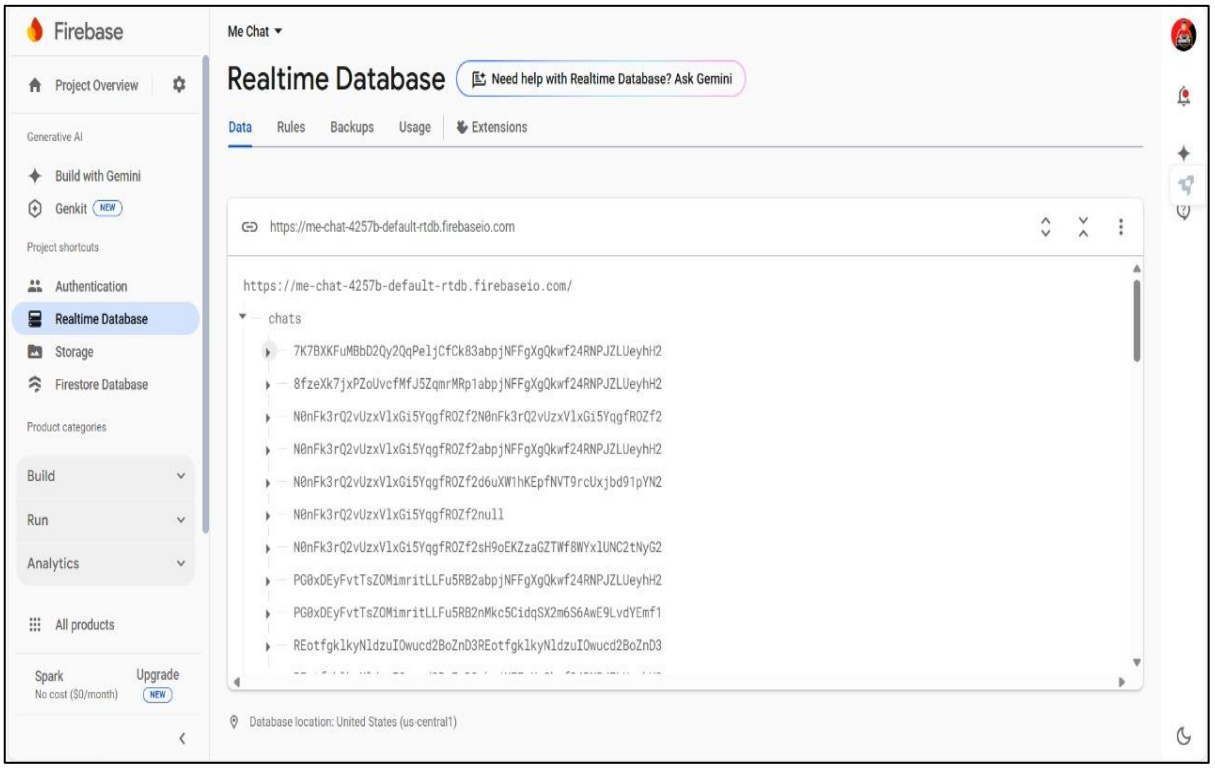
1. **User Interaction**:
    - Users interact with the application by performing actions such as:
        - Uploading or fetching images.
        - Entering login credentials for authentication.
        - Updating their profile information or logging out.
        - Sending and receiving messages within the chat system.
2. **Authentication Process**:
    - The system uses **Firebase Authentication** to validate user credentials.
    - Upon successful validation, an **authentication response** is generated, granting access to the application's features.
    - If credentials are invalid, access is denied, ensuring secure user verification.
3. **Message Handling**:
    - Users can send and receive messages through the chat system.
    - Messages are stored in a **Firebase database** for persistence and can be retrieved when needed.
    - The system ensures efficient management of chat history and real-time message delivery.
4. **Media Management**:
    - Media files (e.g., images) are uploaded by users and managed using **Cloudinary**, a cloud-based media handling service.
    - Cloudinary handles the storage, retrieval, and optimization of media files, ensuring smooth integration with the chat system.
5. **Data Storage and Retrieval**:
    - The **Firebase database** stores critical data such as:
        - User preferences (e.g., notification settings, themes).

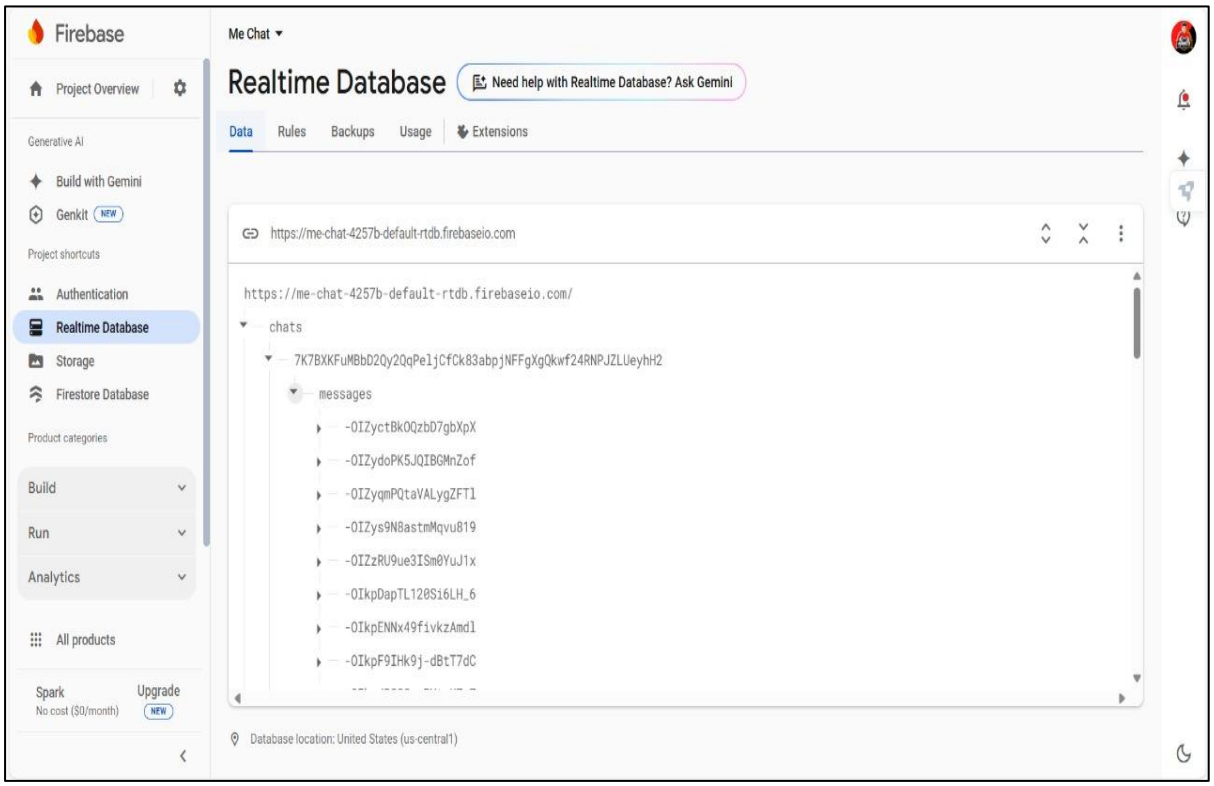
        - Chat messages and media references.
    - Data is retrieved efficiently to provide personalized experiences and seamless access to chat history.
6. **Profile Management**:
    - Users can update their profile information, including personal details and preferences.
    - Updated data is stored in the Firebase database and reflected across the application.
7. **Logout Process**:
    - Users can log out of the application, which terminates their session securely.
    - This ensures that user data remains protected when not in use.
8. **System Integration**:
    - **Firebase** is used for:
        - Authentication (user verification).
        - Database management (storing messages, user preferences, and profile data).
    - **Cloudinary** is integrated for media handling, including upload, storage, and retrieval of images and other media files.

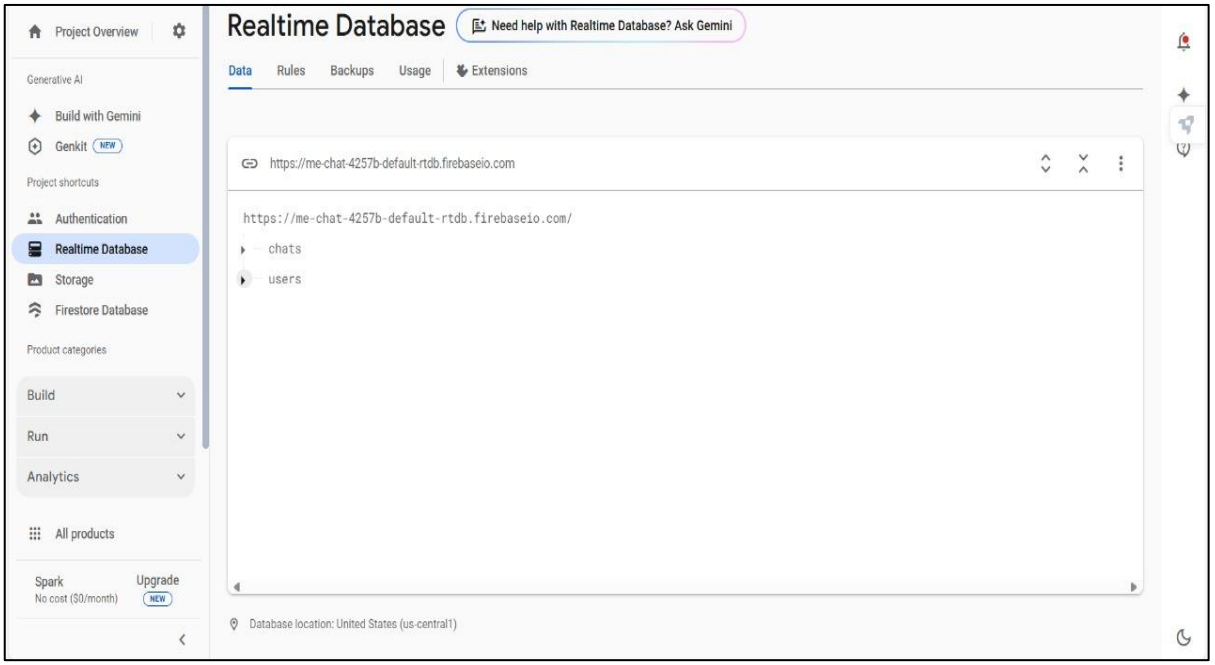
The Level 1 DFD for "MeChat" demonstrates a well-structured flow of data between user interactions, authentication, message handling, media management, and data storage. The integration of Firebase and Cloudinary ensures a secure, efficient, and scalable system. This diagram serves as a foundational reference for understanding the application’s architecture and guiding further development and documentation efforts.

Firebase Schemas

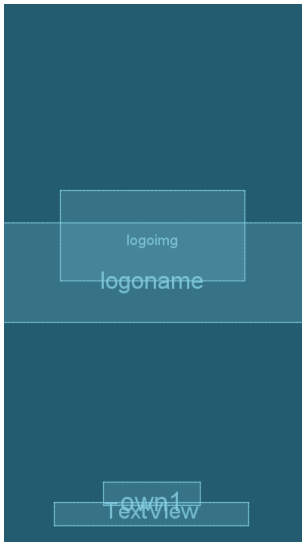


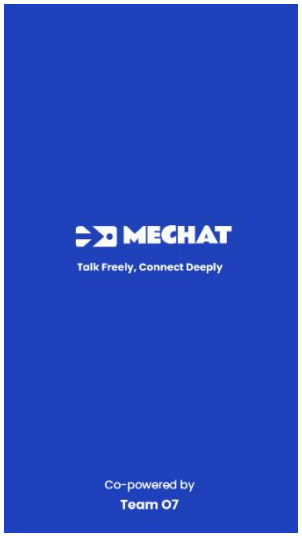


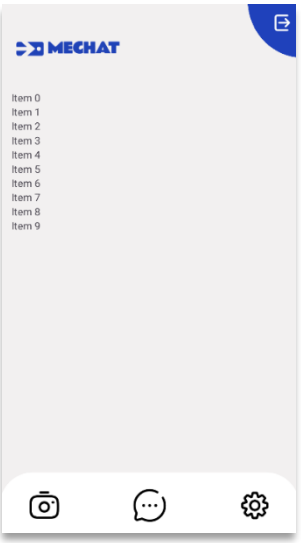

Firebase
Me Chat
Project Overview
Realtime Database
Need help with Realtime Database? Ask Gemini
Generative AI
Build with Gemini
Genkit
NEW
Project shortcuts
Authentication
Realtime Database
Storage
Firestore Database
Product categories
Build
Run
Analytics
All products
Spark
No cost ($0/month)
Upgrade
NEW
Data
Rules
Backups
Usage
Extensions
https://me-chat-4257b-default-rtdb.firebaseio.com
https://me-chat-4257b-default-rtdb.firebaseio.com/
chats
7K7BXKFuMBbD2Qy2QqPeljCfCk83abpjNFFgXgQkwf24RNPJZLUeyhH2
messages
-OIZyctBkOQzbD7gbXpX
-OIZydoPK5JQIBGMnZof
-OIZyqmPQtaVALygZFTl
-OIZys9N8astmMqvu819
-OIZzRU9ue3ISm0YuJ1x
-OIkpDapTL120Si6LH_6
-OIkpENNx49fivkzAmdl
-OIkpF9IHk9j-dBtT7dC
Database location: United States (us-central1)

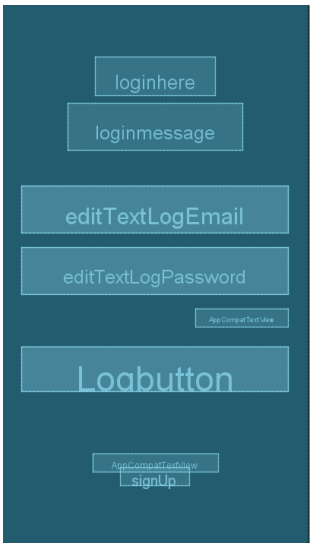

Project Overview
Realtime Database
Need help with Realtime Database? Ask Gemini
Generative AI
Build with Gemini
Genkit
NEW
Project shortcuts
Authentication
Realtime Database
Storage
Firestore Database
Product categories
Build
Run
Analytics
All products
Spark
No cost ($0/month)
Upgrade
NEW
Data
Rules
Backups
Usage
Extensions
https://me-chat-4257b-default-rtdb.firebaseio.com
https://me-chat-4257b-default-rtdb.firebaseio.com/
chats
users
Database location: United States (us-central1)

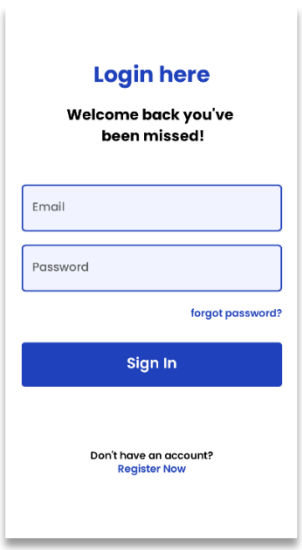
Application Screen Overview (Developer Perspective)

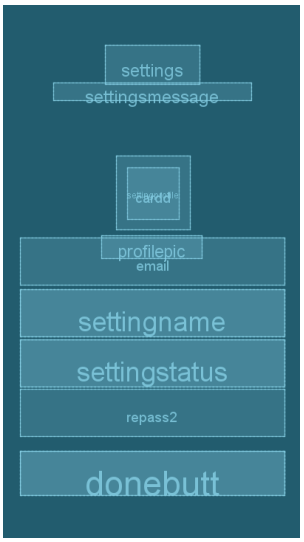


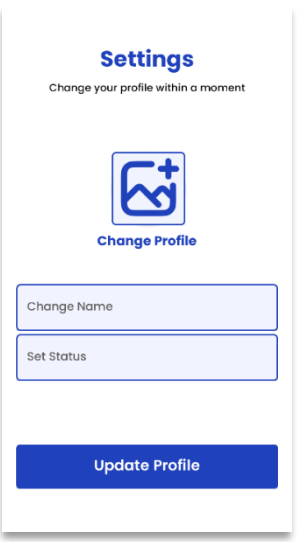


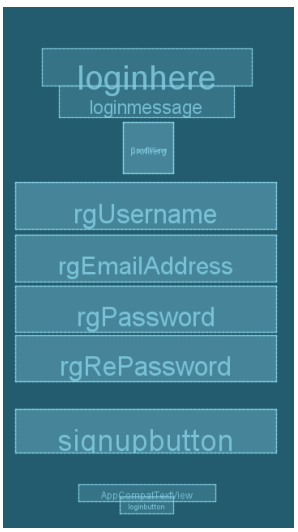


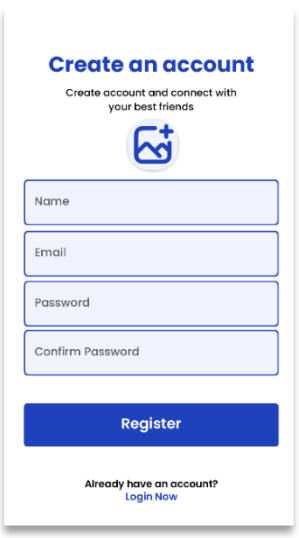


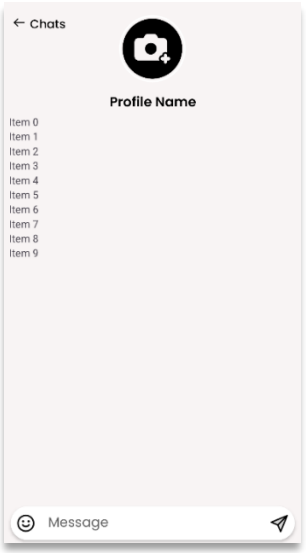




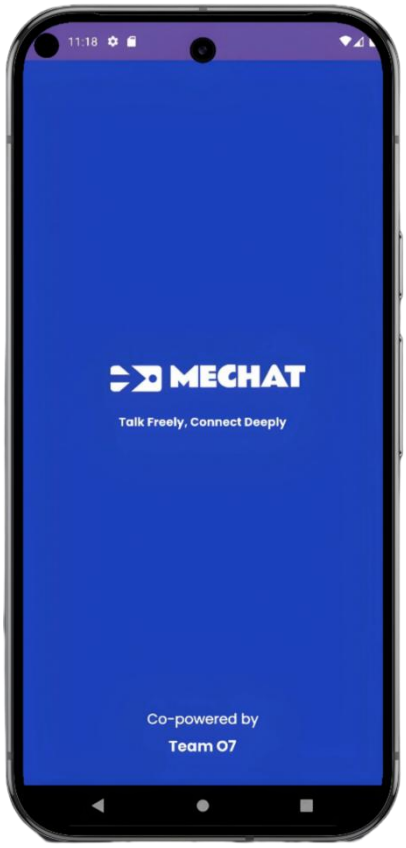


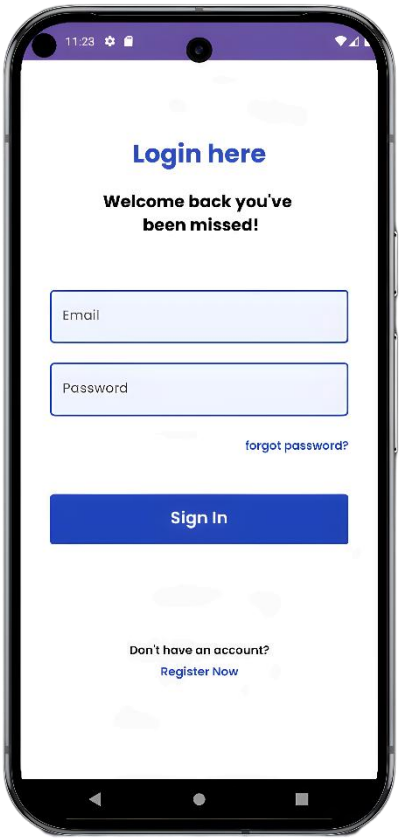


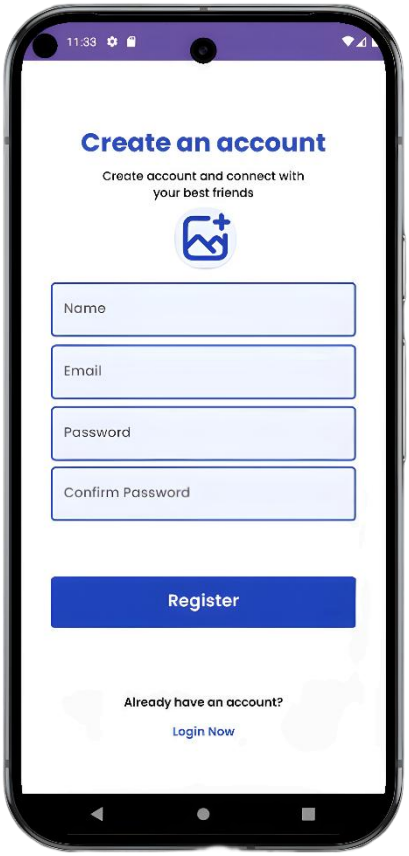


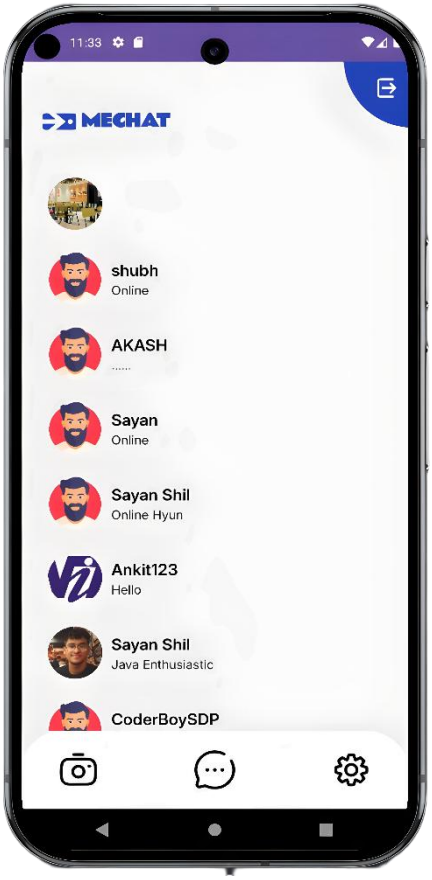


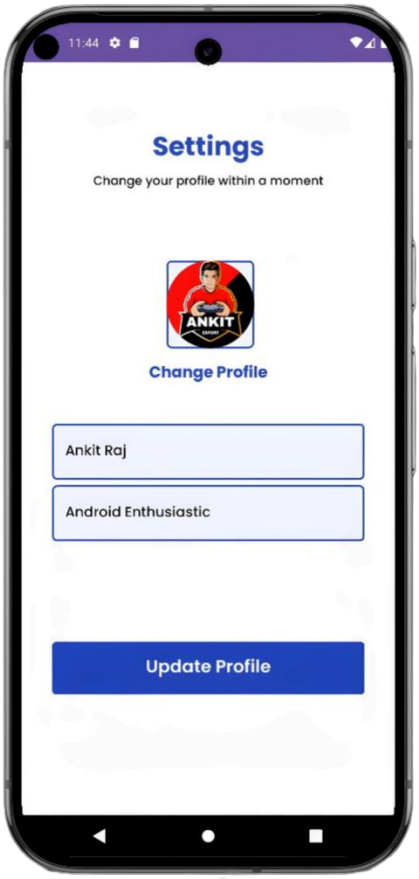
Application Screen Overview (User Perspective)

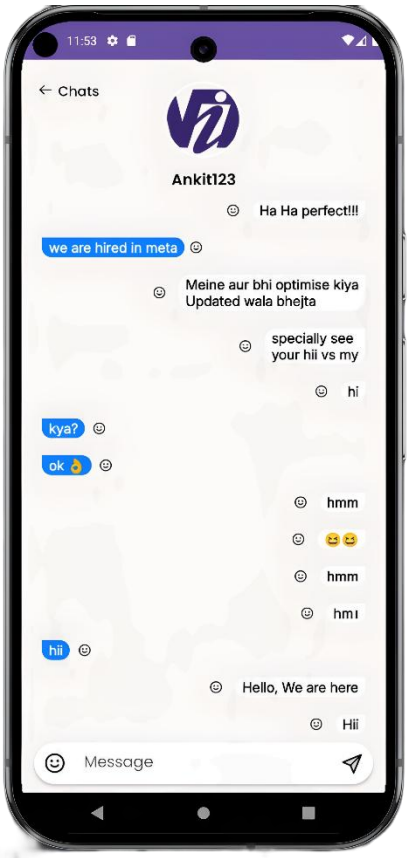






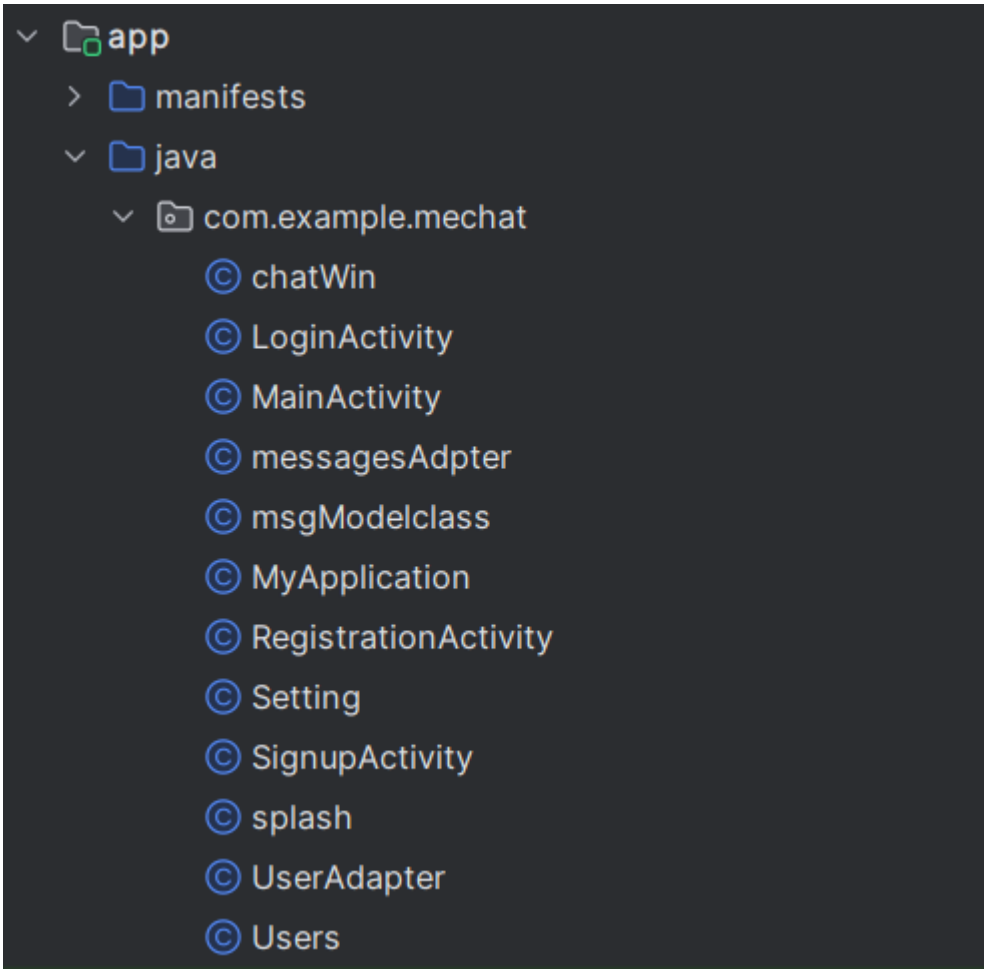






Coding Standard Followed & Screenshots

APP SCREENSHOT



SPLASH SCREEN:

CODE:

```
package com.example.mechat;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class splash extends AppCompatActivity {
    ImageView logo;
    TextView name,own_1,own_2;
    Animation topAnim,bottomAnim;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
        logo = findViewById(R.id.logoimg);
        name = findViewById(R.id.logoname);
        own_1 = findViewById(R.id.own1);
        own_2 = findViewById(R.id.own2);
        topAnim = AnimationUtils.loadAnimation(this,R.anim.top_animation);
        bottomAnim =
AnimationUtils.loadAnimation(this,R.anim.bottom_animation);
        logo.setAnimation(topAnim);
        name.setAnimation(bottomAnim);
        own_1.setAnimation(bottomAnim);
        own_2.setAnimation(bottomAnim);
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent = new Intent(splash.this,
MainActivity.class);
                startActivity(intent);
```

```
                finish();
            }
        },4000);
    }
}
```


11:18
MECHAT
Talk Freely, Connect Deeply
Co-powered by
Team 07

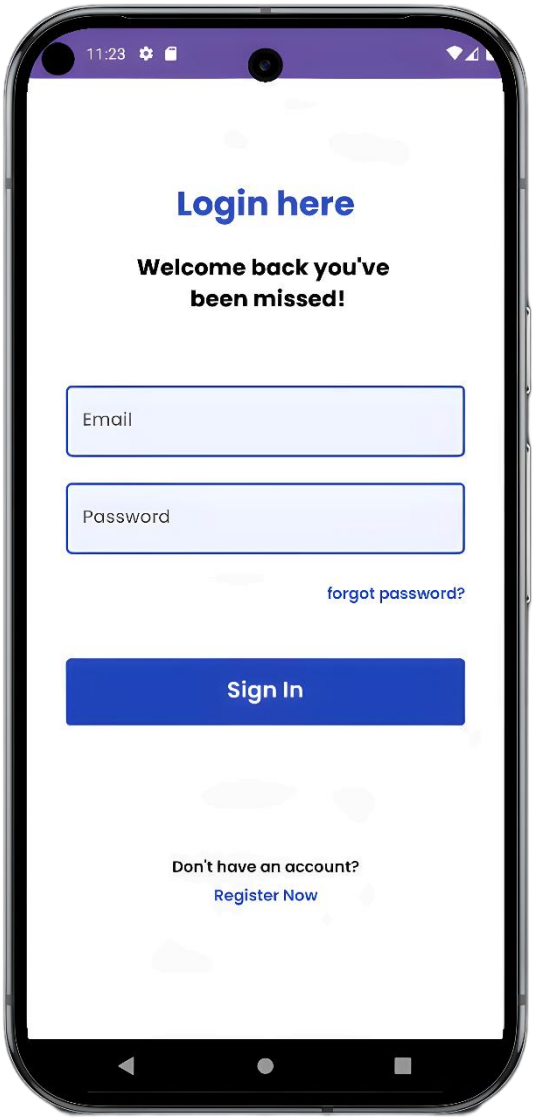
LOGIN PAGE:

CODE:

```
package com.example.mechat;

import android.annotation.SuppressLint;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
public class LoginActivity extends AppCompatActivity {

 Button button;
    TextView signup;
    EditText email,password;
    FirebaseAuth auth;
    String emailPattern = "[a-zA-Z0-9._-]+@[a-z]+\\.+[a-z]+";
    android.app.ProgressDialog progressDialog;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        progressDialog = new ProgressDialog(this);
        progressDialog.setMessage("Please wait");
        progressDialog.setCancelable(false);
```

```
        auth = FirebaseAuth.getInstance();
        button = findViewById(R.id.Logbutton);
        email = findViewById(R.id.editTextLogEmail);
        password = findViewById(R.id.editTextLogPassword);
        signup = findViewById(R.id.signUp);
        signup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new
Intent(LoginActivity.this,RegistrationActivity.class);
                startActivity(intent);
                finish();
            }
        });
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String Email = email.getText().toString().trim();
                String Password = password.getText().toString().trim();

                if((TextUtils.isEmpty(Email))){
                    progressDialog.dismiss();
                    Toast.makeText(LoginActivity.this,"Enter the
Email",Toast.LENGTH_SHORT).show();
                }else if(TextUtils.isEmpty(Password)){
                    progressDialog.dismiss();
                    Toast.makeText(LoginActivity.this,"Enter the
Password",Toast.LENGTH_SHORT).show();
                }else if(!Email.matches(emailPattern)){
                    progressDialog.dismiss();
                    email.setError("Invalid Email Address");
                }else if(Password.length()<6){
                    progressDialog.dismiss();
                    password.setError("Password must be greater than 6
characters");
                    Toast.makeText(LoginActivity.this, "Password Needs To
Be Longer Than Six Characters", Toast.LENGTH_SHORT).show();
                }else{
                    progressDialog.show();

auth.signInWithEmailAndPassword(Email,Password).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult>
task) {
                            if(task.isSuccessful()){
```

```
                                try{
                                    Intent intent = new
Intent(LoginActivity.this,MainActivity.class);
                                    startActivity(intent);
                                    finish();
                                }catch (Exception e){

Toast.makeText(LoginActivity.this,e.getMessage(),Toast.LENGTH_SHORT).show()
;
                                }
                            }else{

Toast.makeText(LoginActivity.this,task.getException().getMessage(),Toast.LE
NGTH_SHORT).show();
                            }
                        }
                    });             }
    });}}}
```



SIGN UP PAGE:

CODE:

```
package com.example.mechat;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import com.cloudinary.android.MediaManager;
import com.cloudinary.android.callback.UploadCallback;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import java.util.HashMap;
import java.util.Map;
import de.hdodenhof.circleimageview.CircleImageView;
import com.cloudinary.android.callback.ErrorInfo;

public class RegistrationActivity extends AppCompatActivity {
    TextView loginBut;
    EditText rg_username, rg_password, rg_repassword, rg_email;
    Button rg_signup;
    CircleImageView rg_profileImg;
    FirebaseAuth auth;
    Uri imageUri;
    String imageUriString;
    FirebaseDatabase database;
    ProgressDialog progressDialog;

    private static final String DEFAULT_IMAGE_URL =
"https://res.cloudinary.com/your_cloud/image/upload/v1738928649/Man.png";
```

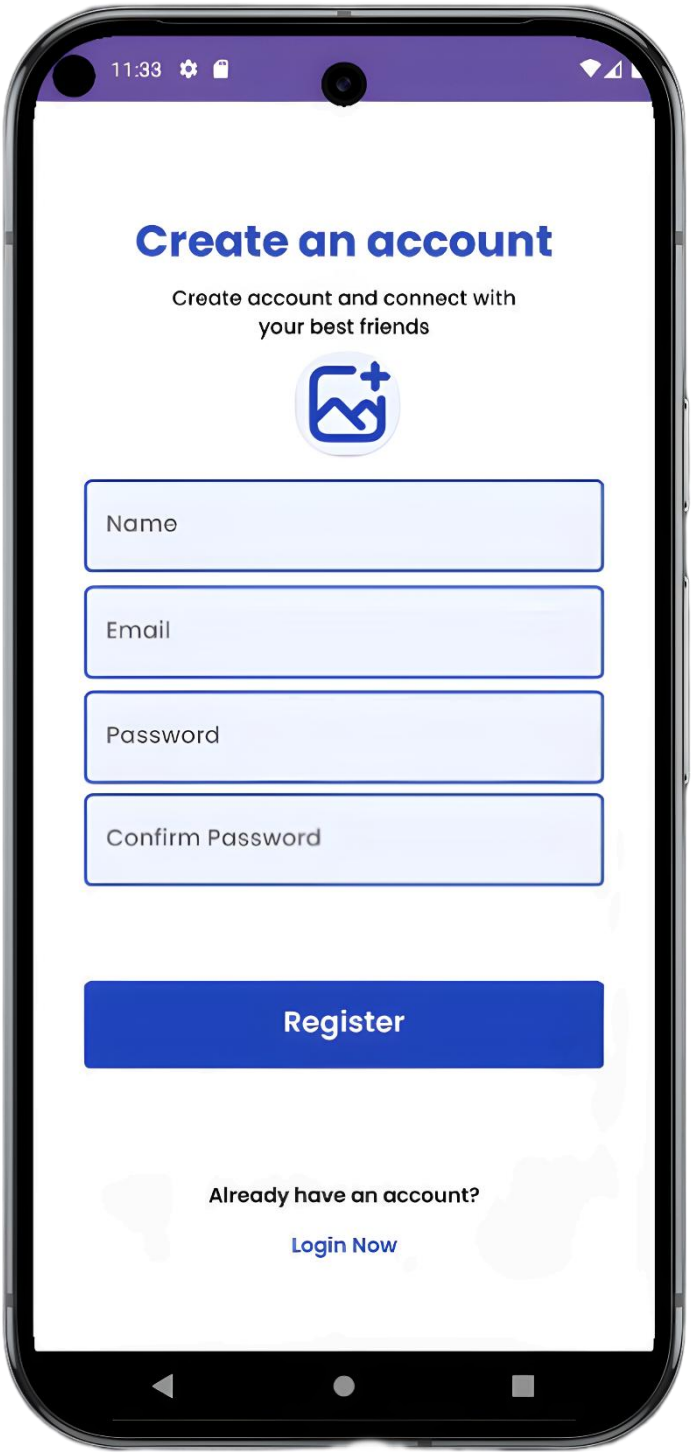
```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registration);

        progressDialog = new ProgressDialog(this);
        progressDialog.setMessage("Establishing your Account");
        progressDialog.setCancelable(false);

        // ☑ Removed initConfig() because Cloudinary is initialized in
MyApplication.java

        loginBut = findViewById(R.id.loginbutton);
        rg_username = findViewById(R.id.rgUsername);
        rg_email = findViewById(R.id.rgEmailAddress);
        rg_password = findViewById(R.id.rgPassword);
        rg_repassword = findViewById(R.id.rgRePassword);
        rg_signup = findViewById(R.id.signupbutton);
        rg_profileImg = findViewById(R.id.profilerg);

        auth = FirebaseAuth.getInstance();
        database = FirebaseDatabase.getInstance();

        rg_profileImg.setOnClickListener(v -> {
            Intent intent = new Intent();
            intent.setType("image/*");
            intent.setAction(Intent.ACTION_GET_CONTENT);
            startActivityForResult(Intent.createChooser(intent, "Select
Picture"), 10);
        });

        rg_signup.setOnClickListener(v -> registerUser());

        loginBut.setOnClickListener(v -> {
            Intent intent = new Intent(RegistrationActivity.this,
LoginActivity.class);
            startActivity(intent);
            finish();
        });
    }

    private void registerUser() {
        String name = rg_username.getText().toString().trim();
        String email = rg_email.getText().toString().trim();
        String password = rg_password.getText().toString().trim();
```

```
        String confirmPassword =
rg_repassword.getText().toString().trim();
        String status = "Online";

        if (TextUtils.isEmpty(name) || TextUtils.isEmpty(email) ||
TextUtils.isEmpty(password) || TextUtils.isEmpty(confirmPassword)) {
            progressDialog.dismiss();
            Toast.makeText(this, "Please enter valid information",
Toast.LENGTH_SHORT).show();
            return;
        }

        auth.createUserWithEmailAndPassword(email,
password).addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                String id = task.getResult().getUser().getUid();
                DatabaseReference reference =
database.getReference().child("users").child(id);

                if (imageUri != null) {
                    uploadToCloudinary(id, name, email, password, status,
reference);
                } else {
                    saveUserToDatabase(id, name, email, password,
DEFAULT_IMAGE_URL, status, reference);
                }
            } else {
                Toast.makeText(this, "Registration Failed: " +
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void uploadToCloudinary(String id, String name, String email,
String password, String status, DatabaseReference reference) {
        if (imageUri == null) {
            Toast.makeText(this, "No image selected!",
Toast.LENGTH_SHORT).show();
            return;
        }

        progressDialog.setMessage("Uploading Image...");
        progressDialog.show();

        MediaManager.get().upload(imageUri)
```

```
                .option("resource_type", "image")
                .callback(new UploadCallback() {
                    @Override
                    public void onStart(String requestId) {
                        // ☑ Called when the upload starts
                        progressDialog.setMessage("Uploading Image...");
                    }

                    @Override
                    public void onProgress(String requestId, long bytes,
long totalBytes) {
                        // ☑ Called while uploading
                        int progress = (int) ((bytes * 100) / totalBytes);
                        progressDialog.setMessage("Uploading... " +
progress + "%");
                    }

                    @Override
                    public void onSuccess(String requestId, Map
resultData) {
                        // ☑ Called when upload is successful
                        if (resultData != null &&
resultData.get("secure_url") != null) {
                            imageUriString =
resultData.get("secure_url").toString();
                            saveUserToDatabase(id, name, email, password,
imageUriString, status, reference);
                        } else {
                            Toast.makeText(RegistrationActivity.this,
"Upload failed: No URL returned!", Toast.LENGTH_SHORT).show();
                        }
                        progressDialog.dismiss();
                    }

                    @Override
                    public void onError(String requestId, ErrorInfo error)
{
                        // ☑ Handle Cloudinary upload failure
                        progressDialog.dismiss();
                        Toast.makeText(RegistrationActivity.this, "Image
Upload Failed: " + error.getDescription(), Toast.LENGTH_SHORT).show();
                        saveUserToDatabase(id, name, email, password,
DEFAULT_IMAGE_URL, status, reference);
                    }
```

```
                    @Override
                    public void onReschedule(String requestId, ErrorInfo
error) {
                        // ☑ Called if upload is rescheduled (e.g., poor
network)
                        Toast.makeText(RegistrationActivity.this, "Upload
Rescheduled: " + error.getDescription(), Toast.LENGTH_SHORT).show();
                    }
                }).dispatch();
    }


    private void saveUserToDatabase(String id, String name, String email,
String password, String imageUrl, String status, DatabaseReference
reference) {
        Users user = new Users(id, name, email, password, imageUrl,
status);
        reference.setValue(user).addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                Toast.makeText(RegistrationActivity.this, "User Registered
Successfully!", Toast.LENGTH_SHORT).show();
                navigateToMain();
            } else {
                Toast.makeText(RegistrationActivity.this, "Error in
Creating the User", Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void navigateToMain() {
        progressDialog.show();
        Intent intent = new Intent(RegistrationActivity.this,
MainActivity.class);
        startActivity(intent);
        finish();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == 10 && data != null && data.getData() != null) {
            imageUri = data.getData();
            rg_profileImg.setImageURI(imageUri);
        }
```

```
    }
}
```


11:33
Create an account
Create account and connect with your best friends
Name
Email
Password
Confirm Password
Register
Already have an account?
Login Now

MAIN ACTIVITY:

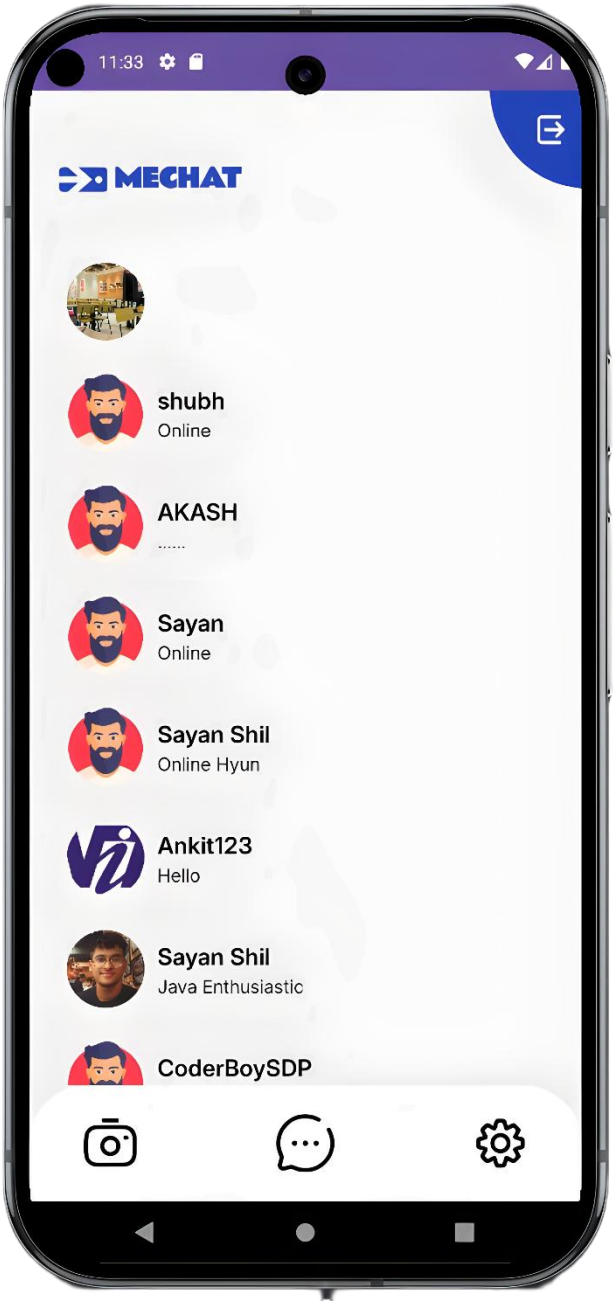
CODE:

```
package com.example.mechat;

import android.app.Dialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {
    FirebaseAuth auth;
    RecyclerView mainUserRecyclerView;
    UserAdapter adapter;
    FirebaseDatabase database;
    List<Users> usersArrayList;
    ImageView imglogout, cumbut, setbut;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        auth = FirebaseAuth.getInstance();
        if (auth.getCurrentUser() == null) {
            Toast.makeText(this, "User not authenticated!",
```

```
Toast.LENGTH_SHORT).show();
            startActivity(new Intent(MainActivity.this,
LoginActivity.class));
            finish();
            return;
        }

        database = FirebaseDatabase.getInstance();
        DatabaseReference reference =
database.getReference().child("users");

        usersArrayList = new ArrayList<>();
        adapter = new UserAdapter(this, usersArrayList);

        imglogout = findViewById(R.id.logoutimg);
        mainUserRecyclerView = findViewById(R.id.mainUserRecyclerView);
        mainUserRecyclerView.setLayoutManager(new
LinearLayoutManager(this));
        mainUserRecyclerView.setAdapter(adapter);

        setbut = findViewById(R.id.settingBut);
        cumbut = findViewById(R.id.camBut);

        //  ◇  Handle Logout Click
        imglogout.setOnClickListener(v -> {
            Dialog dialog = new Dialog(MainActivity.this, R.style.dialoge);
            dialog.setContentView(R.layout.dialog_layout);
            dialog.setCancelable(false);

            Button yes = dialog.findViewById(R.id.yesbnt);
            Button no = dialog.findViewById(R.id.nobnt);

            yes.setOnClickListener(v1 -> {
                dialog.dismiss();
                FirebaseAuth.getInstance().signOut();
                Intent intent = new Intent(MainActivity.this,
LoginActivity.class);
                startActivity(intent);
                finish();
            });

            no.setOnClickListener(v1 -> dialog.dismiss());

            dialog.show();
        });
```

```
        //  ◇  Open Settings Activity
        setbut.setOnClickListener(v -> {
            Intent intent = new Intent(MainActivity.this, Setting.class);
            startActivity(intent);
        });

        //  ◇  Open Camera Safely
        cumbut.setOnClickListener(v -> {
            Intent intent = new
Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
            if (intent.resolveActivity(getPackageManager()) != null) {
                startActivityForResult(intent, 10);
            } else {
                Toast.makeText(MainActivity.this, "No camera app
available", Toast.LENGTH_SHORT).show();
            }
        });

        //  ◇  Load Users from Firebase
        reference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                usersArrayList.clear();
                for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                    if (dataSnapshot.exists()) {
                        Users user = dataSnapshot.getValue(Users.class);
                        usersArrayList.add(user);
                    }
                }
                adapter.notifyDataSetChanged();
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Toast.makeText(MainActivity.this, "Database error: " +
error.getMessage(), Toast.LENGTH_SHORT).show();
            }
        });
    }

    //  Handle Camera Result
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == 10 && resultCode == RESULT_OK) {
```

```
            Toast.makeText(this, "Photo captured successfully!",
Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "Camera action canceled",
Toast.LENGTH_SHORT).show();
        }
    }
}
```



SETTING:

CODE:

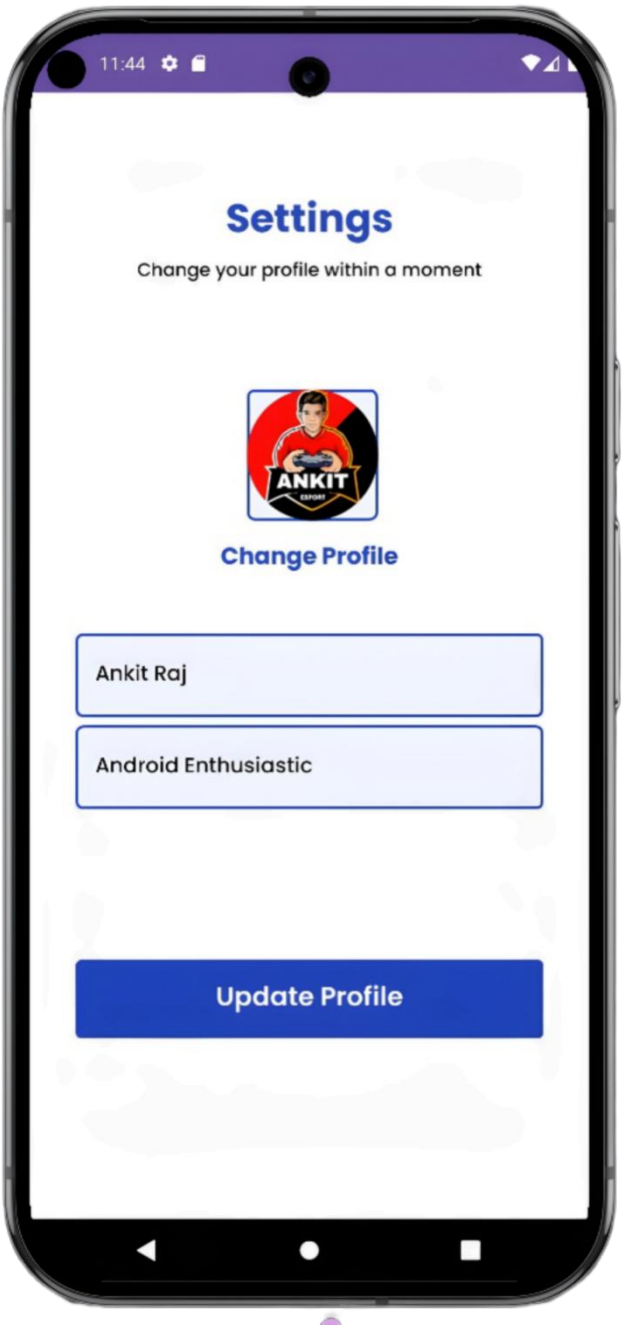
```
package com.example.mechat;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.app.ProgressDialog;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import com.cloudinary.android.MediaManager;
import com.cloudinary.android.callback.ErrorInfo;
import com.cloudinary.android.callback.UploadCallback;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.squareup.picasso.Picasso;

import java.util.HashMap;
import java.util.Map;

public class Setting extends AppCompatActivity {
    ImageView setprofile;
    EditText setname, setstatus;
    Button donebut;
    FirebaseAuth auth;
    FirebaseDatabase database;
    Uri setImageUri;
    ProgressDialog progressDialog;
    String email, password;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_setting);
```

```
        auth = FirebaseAuth.getInstance();
        database = FirebaseDatabase.getInstance();

        setprofile = findViewById(R.id.settingprofile);
        setname = findViewById(R.id.settingname);
        setstatus = findViewById(R.id.settingstatus);
        donebut = findViewById(R.id.donebutt);

        progressDialog = new ProgressDialog(this);
        progressDialog.setMessage("Saving...");
        progressDialog.setCancelable(false);

        DatabaseReference reference =
database.getReference().child("users").child(auth.getUid());

        reference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                email = snapshot.child("mail").getValue(String.class);
                password =
snapshot.child("password").getValue(String.class);
                String name =
snapshot.child("username").getValue(String.class);
                String profile =
snapshot.child("profilePic").getValue(String.class);
                String status =
snapshot.child("status").getValue(String.class);

                setname.setText(name);
                setstatus.setText(status);

                if (profile != null && !profile.isEmpty()) {
                    Picasso.get().load(profile).into(setprofile);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Log.e("Firebase", "Error fetching data: " +
error.getMessage());
            }
        });

        // Select Image from Gallery
        setprofile.setOnClickListener(view -> {
            Intent intent = new Intent();
            intent.setType("image/*");
            intent.setAction(Intent.ACTION_GET_CONTENT);
```

```
            startActivityForResult(Intent.createChooser(intent, "Select
Picture"), 10);
        });

        // Save Data on Button Click
        donebut.setOnClickListener(view -> {
            progressDialog.show();
            String name = setname.getText().toString();
            String status = setstatus.getText().toString();

            if (setImageUri != null) {
                uploadImageToCloudinary(name, status, reference);
            } else {
                updateUserProfile(name, status, null, reference);
            }
        });
    }

    // ☑ Handle Image Selection from Gallery
    @Override
    protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == 10 && data != null) {
            setImageUri = data.getData();
            setprofile.setImageURI(setImageUri);
        }
    }

    // ☑ Upload Image to Cloudinary
    private void uploadImageToCloudinary(String name, String status,
DatabaseReference reference) {
        progressDialog.setMessage("Uploading Image...");
        progressDialog.show();

        MediaManager.get().upload(setImageUri)
                .callback(new UploadCallback() {
                    @Override
                    public void onStart(String requestId) {
                        Log.d("Cloudinary", "Upload started...");
                    }

                    @Override
                    public void onProgress(String requestId, long bytes,
long totalBytes) {
                        Log.d("Cloudinary", "Uploading: " + (bytes * 100 /
totalBytes) + "%");
                    }
```

```
                    @Override
                    public void onSuccess(String requestId, Map resultData)
{
                        String imageUrl =
resultData.get("secure_url").toString();
                        Log.d("Cloudinary", "Image uploaded: " + imageUrl);

                        updateUserProfile(name, status, imageUrl,
reference);
                    }

                    @Override
                    public void onError(String requestId, ErrorInfo error)
{
                        progressDialog.dismiss();
                        Toast.makeText(Setting.this, "Image Upload
Failed!", Toast.LENGTH_SHORT).show();
                        Log.e("Cloudinary", "Upload error: " +
error.getDescription());
                    }

                    @Override
                    public void onReschedule(String requestId, ErrorInfo
error) {
                        Log.e("Cloudinary", "Upload rescheduled: " +
error.getDescription());
                    }
                })
                .dispatch();
    }

    // Update User Profile in Firebase

    private void updateUserProfile(String name, String status, @Nullable
String imageUrl, DatabaseReference reference) {
        progressDialog.setMessage("Saving Data...");
        progressDialog.show();

        Map<String, Object> userData = new HashMap<>();
        userData.put("username", name);
        userData.put("status", status);
        if (imageUrl != null) {
            userData.put("profilePic", imageUrl);
        }

        reference.updateChildren(userData).addOnCompleteListener(task -> {
            progressDialog.dismiss();
            if (task.isSuccessful()) {
                Toast.makeText(Setting.this, "Profile Updated!",
```

```
Toast.LENGTH_SHORT).show();
                startActivity(new Intent(Setting.this,
MainActivity.class));
                finish();
            } else {
                Toast.makeText(Setting.this, "Error Saving Data!",
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```



CHAT WINDOW:

CODE:

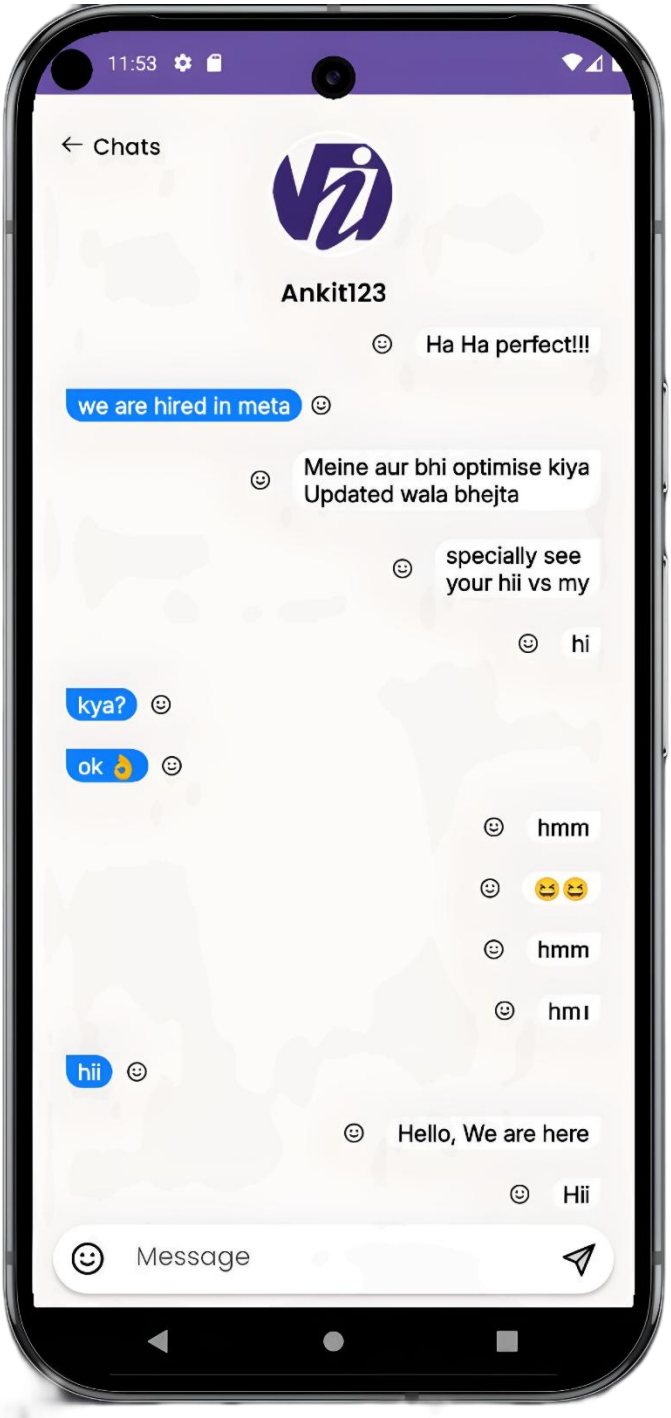
```
package com.example.mechat;

import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.AppCompatEditText;
import androidx.appcompat.widget.AppCompatImageView;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.squareup.picasso.Picasso;

import java.util.ArrayList;
import java.util.Date;

import de.hdodenhof.circleimageview.CircleImageView;

public class chatWin extends AppCompatActivity {
    String reciverimg, reciverUid, reciverName, SenderUID;
    CircleImageView profile;
    TextView reciverNName;
    FirebaseDatabase database;
    FirebaseAuth firebaseAuth;
    public static String senderImg;
    public static String reciverImg;
```

```
    AppCompatImageView sendbtn;
    AppCompatEditText textmsg;

    String senderRoom, reciverRoom;
    RecyclerView messageAdpter;
    ArrayList<msgModelclass> messagesArrayList;
    messagesAdpter mmessagesAdpter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_chat_win);

        database = FirebaseDatabase.getInstance();
        firebaseAuth = FirebaseAuth.getInstance();

        // Initialize Views
        sendbtn = findViewById(R.id.sendbtnn);
        textmsg = findViewById(R.id.textmsg);
        reciverNName = findViewById(R.id.recivername);
        profile = findViewById(R.id.profileimgg);
        messageAdpter = findViewById(R.id.msgadpter);

        // Get Intent Data
        reciverName = getIntent().getStringExtra("nameee");
        reciverimg = getIntent().getStringExtra("recieverImg");
        reciverUid = getIntent().getStringExtra("uid");

        Log.d("Chat", "Receiver Name: " + reciverName);
        Log.d("Chat", "Receiver Image: " + reciverimg);

        // Set Receiver Name
        if (reciverName != null) {
            reciverNName.setText(reciverName);
        } else {
            reciverNName.setText("Unknown User");
        }

        // Set Profile Image with Picasso
        if (reciverimg != null && !reciverimg.isEmpty()) {

Picasso.get().load(reciverimg).placeholder(R.drawable.man).into(profile);
        } else {
            profile.setImageResource(R.drawable.man);
        }
```

```
        // RecyclerView Setup
        messagesArrayList = new ArrayList<>();
        LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(this);
        messageAdpter.setLayoutManager(linearLayoutManager);

        mmessagesAdpter = new messagesAdpter(chatWin.this,
messagesArrayList);
        messageAdpter.setAdapter(mmessagesAdpter);

        SenderUID = firebaseAuth.getUid();
        senderRoom = SenderUID + reciverUid;
        reciverRoom = reciverUid + SenderUID;

        DatabaseReference chatreference =
database.getReference().child("chats").child(senderRoom).child("messages")
;

        chatreference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                messagesArrayList.clear();
                for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                    msgModelclass messages =
dataSnapshot.getValue(msgModelclass.class);
                    messagesArrayList.add(messages);
                }
                mmessagesAdpter.notifyDataSetChanged();

                // Scroll to the last message after data update

messageAdpter.scrollToPosition(mmessagesAdpter.getItemCount() - 1);
                Log.d("Chat", "Messages updated: " +
messagesArrayList.size());
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Log.e("Chat", "Database Error: " + error.getMessage());
            }
        });

        // ☑ Send Message on Button Click
        sendbtn.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View view) {
                String message = textmsg.getText().toString().trim();

                if (message.isEmpty()) {
                    Toast.makeText(chatWin.this, "Enter a message first!",
Toast.LENGTH_SHORT).show();
                    return;
                }

                textmsg.setText(""); // Clear input box
                Date date = new Date();
                msgModelclass messagess = new msgModelclass(message,
SenderUID, date.getTime());

                Log.d("Chat", "Sending message: " + message);

                // Send message to Firebase

database.getReference().child("chats").child(senderRoom).child("messages")
                        .push().setValue(messagess)
                        .addOnCompleteListener(new
OnCompleteListener<Void>() {
                            @Override
                            public void onComplete(@NonNull Task<Void>
task) {
                                if (task.isSuccessful()) {
                                    Log.d("Chat", "Message sent
successfully");

                                    // Also save to receiver's chat room

database.getReference().child("chats").child(reciverRoom).child("messages"
)
                                            .push().setValue(messagess)
                                            .addOnCompleteListener(new
OnCompleteListener<Void>() {
                                                @Override
                                                public void
onComplete(@NonNull Task<Void> task) {
                                                    Log.d("Chat", "Message
added to receiver's chat room");
                                                }
                                            });
```

```
                                } else {
                                    Log.e("Chat", "Failed to send
message", task.getException());
                                }
                            }
                        });
                }
            });
    }
}
```



OTHER CODES:

MESSAGES ADAPTER:

CODE:

```
package com.example.mechat;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.widget.AppCompatImageView;
import androidx.appcompat.widget.AppCompatTextView;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.auth.FirebaseAuth;

import java.util.ArrayList;

import de.hdodenhof.circleimageview.CircleImageView;

public class messagesAdpter extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
    private Context context;
    private ArrayList<msgModelclass> messagesList;
    private static final int MSG_TYPE_SENDER = 1;
    private static final int MSG_TYPE_RECEIVER = 2;

    public messagesAdpter(Context context, ArrayList<msgModelclass>
messagesList) {
        this.context = context;
        this.messagesList = messagesList;
    }

    @Override
    public int getItemViewType(int position) {
        if
```

```
(messagesList.get(position).getSenderId().equals(FirebaseAuth.getInstance(
).getUid())) {
            return MSG_TYPE_SENDER;
        } else {
            return MSG_TYPE_RECEIVER;
        }
    }

    @NonNull
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        if (viewType == MSG_TYPE_SENDER) {
            View view =
LayoutInflater.from(context).inflate(R.layout.sender_layout, parent,
false);
            return new SenderViewHolder(view);
        } else {
            View view =
LayoutInflater.from(context).inflate(R.layout.reciever_layout, parent,
false);
            return new ReceiverViewHolder(view);
        }
    }

    @Override
    public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder,
int position) {
        msgModelclass message = messagesList.get(position);

        if (holder.getItemViewType() == MSG_TYPE_SENDER) {
            ((SenderViewHolder)
holder).msgsendertyp.setText(message.getMessage());
        } else {
            ((ReceiverViewHolder)
holder).recivertextset.setText(message.getMessage());
        }
    }

    @Override
    public int getItemCount() {
        return messagesList.size();
    }

    static class SenderViewHolder extends RecyclerView.ViewHolder {
```

```
        AppCompatTextView msgsendertyp;
        AppCompatImageView profilerggg;

        public SenderViewHolder(@NonNull View itemView) {
            super(itemView);
            msgsendertyp = itemView.findViewById(R.id.msgsendertyp);
            profilerggg = itemView.findViewById(R.id.profilerggg);
        }
    }

    static class ReceiverViewHolder extends RecyclerView.ViewHolder {
        AppCompatTextView recivertextset;
        AppCompatImageView pro;

        public ReceiverViewHolder(@NonNull View itemView) {
            super(itemView);
            recivertextset = itemView.findViewById(R.id.recivertextset);
            pro = itemView.findViewById(R.id.pro);
        }
    }
}
```

MESSAGE MODEL CLASS:

CODE:

```
package com.example.mechat;

public class msgModelclass {
    String message,senderId;
    long timeStamp;

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public msgModelclass() {
    }

    public msgModelclass(String message, String senderId, long timeStamp)
{
        this.message = message;
        this.senderId = senderId;
        this.timeStamp = timeStamp;
    }

    public String getSenderId() {
        return senderId;
    }

    public void setSenderId(String senderId) {
        this.senderId = senderId;
    }

    public long getTimeStamp() {
        return timeStamp;
    }

    public void setTimeStamp(long timeStamp) {
        this.timeStamp = timeStamp;
    }
}
```

MY APPLICATION:

CODE:

```
package com.example.mechat;

import android.app.Application;
import com.cloudinary.android.MediaManager;
import java.util.HashMap;
import java.util.Map;

public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        // ☑ Cloudinary Initialization (Runs Once for the Whole App)
        Map<String, String> config = new HashMap<>();
        config.put("cloud_name", "decujmqwp");
        config.put("api_key", "322447726646421");
        config.put("api_secret", "bJ9GVHsCMmarUYTbuTTmWxJDtMY");
        MediaManager.init(this, config);

    }
}
```

USER ADAPTER:

CODE:

```
package com.example.mechat;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.squareup.picasso.Picasso;

import java.util.List;

import de.hdodenhof.circleimageview.CircleImageView;

public class UserAdapter extends
RecyclerView.Adapter<UserAdapter.ViewHolder> {
    private Context context; //  ◇  Added context
    private List<Users> usersList;

    public UserAdapter(Context context, List<Users> usersList) { //  ◇
Fixed constructor
        this.context = context;
        this.usersList = usersList;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.user_item,
parent, false);
        return new ViewHolder(view);
    }

    @Override
```

```
    public void onBindViewHolder(@NonNull ViewHolder holder, int position)
{
        Users user = usersList.get(position);
        holder.username.setText(user.getUsername());
        holder.userstatus.setText(user.getStatus());

        // Handle image loading with a placeholder & error image
        Picasso.get()
                .load(user.getProfilePic())
                .placeholder(R.drawable.man)  // Set a default image
                .error(R.drawable.man)        // Set an error image
                .into(holder.userimg);

        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(context, chatWin.class); //  ◇
Fixed
                intent.putExtra("nameee", user.getUsername());
                intent.putExtra("recieverImg", user.getProfilePic());
                intent.putExtra("uid", user.getUserId());
                context.startActivity(intent); //  ◇  Fixed
            }
        });
    }

    @Override
    public int getItemCount() {
        return usersList.size();
    }

    public static class ViewHolder extends RecyclerView.ViewHolder {
        CircleImageView userimg;
        TextView username, userstatus;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            userimg = itemView.findViewById(R.id.userimg);
            username = itemView.findViewById(R.id.username);
            userstatus = itemView.findViewById(R.id.userstatus);
        }
    }
}
```

USERS:

CODE:

```
package com.example.mechat;

public class Users {
    private String profilePic, mail, username, password, userId,
lastMessage, status;

    // ☑ Default Constructor (Required for Firebase)
    public Users() {
    }

    // ☑ Properly Assign Data in Constructor
    public Users(String id, String name, String email, String password,
String imageUri, String status) {
        this.userId = id;
        this.username = name;
        this.mail = email;
        this.password = password;
        this.profilePic = imageUri;
        this.status = status;
        this.lastMessage = ""; // Default empty message
    }

    // ☑ Getters & Setters
    public String getProfilePic() { return profilePic; }
    public void setProfilePic(String profilePic) { this.profilePic =
profilePic; }

    public String getMail() { return mail; }
    public void setMail(String mail) { this.mail = mail; }

    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }

    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }

    public String getUserId() { return userId; }
    public void setUserId(String userId) { this.userId = userId; }

    public String getLastMessage() { return lastMessage; }
```

```
    public void setLastMessage(String lastMessage) { this.lastMessage =
lastMessage; }

    public String getStatus() { return status; }
    public void setStatus(String status) { this.status = status; }
}
```

Future Scope and Further Enhancements:

1. **Group Chat Functionality**: Implement the ability for users to create and participate in group chats, allowing multiple users to communicate in a single conversation.
2. **Voice and Video Calls**: Add voice and video calling capabilities for richer, real-time communication.
3. **Web Version**: Develop a web-based version of the app to allow users to access their messages and profiles from desktops or laptops.
4. **End-to-End Encryption**: Enhance security by implementing end-to-end encryption, ensuring that only users involved in the conversation can read the messages.
5. **Premium Features**: Introduce in-app purchases or subscription models, offering advanced features like custom themes, extra storage, or an ad-free experience.
6. **AI Integration**: Add features like smart replies, message filtering, or chatbots to improve user experience and engagement.
7. **Multi-Language Support**: Expand the app to support multiple languages, making it more accessible to a global audience.
8. **Push Notification**: Allow users to push notification settings, giving them control over alerts for new messages or updates.

Conclusion:

This messaging app project successfully addresses the core needs of modern communication by providing a secure, efficient, and intuitive platform for users. By leveraging Firebase for real-time data synchronization and secure authentication, the app ensures a seamless user experience with minimal latency in message delivery and storage. The focus on data privacy and security is paramount, with the use of Firebase's secure database and authentication mechanisms, which offer both scalability and robust protection for user information.

The app's simplicity, combined with essential features such as profile management, real-time chatting, and a camera integration for profile picture updates, ensures users can communicate effectively while maintaining control over their personal data. The inclusion of a logout feature offers an additional layer of security, ensuring that users can easily protect their accounts when not in use.

In conclusion, this messaging app has the potential to evolve into a robust and scalable communication platform. By continuously improving the app's features and prioritizing user privacy, security, and experience, the app can meet the needs of a diverse and growing user base, positioning itself as a competitive player in the messaging app market. With a solid foundation, future enhancements, and careful attention to user feedback, the app can become a reliable and trusted communication tool for users worldwide.

References and Bibliography

1. **Android Java Development**
   - Android Developers. (n.d.). Official documentation on developing Android apps using Java. Retrieved from: https://developer.android.com/docs
   - **Java Programming**: Oracle. (n.d.). Official documentation for Java programming language. Retrieved from: https://docs.oracle.com/javase/
2. **Firebase Documentation**
   - Firebase Authentication. (n.d.). Firebase Authentication for securing login, signup, and user management. Retrieved from: https://firebase.google.com/docs/auth
   - Firebase Firestore. (n.d.). Firebase Firestore for storing and syncing user data. Retrieved from: https://firebase.google.com/docs/firestore
   - Firebase Cloud Storage. (n.d.). Firebase Storage for uploading and managing profile pictures. Retrieved from: https://firebase.google.com/docs/storage
3. **Android Camera Integration**
   - Android Developers. (n.d.). Using the Camera API in Android. Retrieved from: https://developer.android.com/reference/android/hardware/camera2/package-summary
   - Android Camera Intent. (n.d.). Retrieved from: https://developer.android.com/guide/topics/media/camera
4. **Real-time Database with Firebase**
   - Firebase Realtime Database. (n.d.). Storing and syncing data in real-time with Firebase. Retrieved from: https://firebase.google.com/docs/database
5. **Security Best Practices for Android**

- Android Security Best Practices. (n.d.). Official guide for implementing security measures in Android apps. Retrieved from: https://developer.android.com/topic/security

6. **UI/UX Design for Android Apps**
   - Material Design Guidelines. (n.d.). Google's official Material Design guidelines for creating intuitive, visually appealing Android apps. Retrieved from: https://material.io/design
   - Gradient Using by
   - https://webgradients.com/
7. **Best Practices for Java Development**
   - Effective Java (3rd Edition) by Joshua Bloch.
     This book provides excellent practices and tips for writing efficient and maintainable Java code.
     ISBN: 978-0134685991
8. **Android App Testing**
   - Android Testing Documentation. (n.d.). Official Android documentation for unit testing, UI testing, and debugging. Retrieved from: https://developer.android.com/training/testing
9. **Data Privacy and Security Guidelines**
   - General Data Protection Regulation (GDPR). (n.d.). Retrieved from: https://gdpr.eu/
10. **Java Programming Resources**
    - Java Documentation for Android. (n.d.). Java programming resources specifically for Android development. Retrieved from: https://www.javaworld.com/

These references provide a solid foundation for understanding the tools, technologies, and best practices used in the development of the messaging app, ensuring it meets both technical and security standards.