WhatsApp Clone

A Project Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

Name: CHANDAN SHARMA Roll Number – 457

Under the esteemed guidance of

Prof. Randeep Singh Ghai

Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

GURU NANAK KHALSA COLLEGE

OF

ARTS, SCIENCE & COMMERCE

(Autonomous) MATUNGA,

MUMBAI – 400 019

MAHARASHTRA AY 2023 –

2024

GURU NANAK KHALSA COLLEGE OF ARTS, SCIENCE & COMMERCE

(Autonomous)

MATUNGA, MUMBAI, MAHARASHTRA – 400 019

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled, "**WhatsApp Clone**", is bonafied work of **Chandan Sharma** bearing **Seat No: 457** submitted in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide

Coordinator

External Examiner

DEPT. OF INFORMATION TECHNOLOGY
G. N. KHALSA COLLEGE (AUTONOMOUS)
MATUNGA, MUMBAI-400 019

Date:

College Seal

EXAMINED

18 APR 2024

ACKNOWLEDGEMENT

I would like to express my thanks to the people who have helped me most throughout my project. I am grateful to my **Prof. Randeep Singh Ghai & Mrs. Pinky Panda** for nonstop support for the project. I can't say thank you enough for him tremendous support and help.

I owe my deep gratitude to our HOD of Information Technology Department **Mrs. Pinky Panda** who took keen interest on our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

At last but not the least I want to thank all of my friends who helped/treasured me out in completing the project, where they all exchanged their own interesting ideas, thoughts and made this possible to complete my project with all accurate information. I wish to thank my parents for their personal support or attention who inspired/encouraged me to go my own way.

DECLARATION

I hereby declare that the project entitled, "**WhatsApp Clone**" done a**t Guru Nanak Khalsa College**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university. The project is done in partial fulfilment of the requirements for the award of degree of

BACHELOR OF SCIENCE (INFORMATION

TECHNOLOGY) to be submitted as final semester project as part of our curriculum.

CHANDAN SHARMA

TABLE OF CONTENTS

Software and Broad Area of Application

Languages used-	JAVA, XML.
Tools used-	Android Studio
Technologies used-	Microsoft Windows 11

Area of Applications	Social media application

WhatsApp Clone

The WhatsApp Clone App Project aims to develop a messaging application inspired by the popular WhatsApp platform, offering seamless communication, multimedia sharing, and real-time messaging between users. The application will be designed for Android and will leverage key features to provide a familiar and user-friendly experience, similar to WhatsApp, while also addressing certain shortcomings and incorporating innovative features.

Features:

- **User Registration:** Allow users to sign up with their phone numbers, email, or social media accounts and implement secure authentication methods.
- **Real-time Messaging:** Enable users to send and receive text messages in real-time, create groups, and engage in group conversations.
- **User Profile Customization:** Allow users to customize their profiles by adding a profile picture, status, and other personal details.
- **Notification System:** Implement a comprehensive notification system to keep users informed about new messages, calls, and other important activities.

Advantages:

Seamless Communication: The WhatsApp Clone App will provide users with a reliable and efficient platform for real-time communication, making it easy to connect with friends and family.

User-Friendly Interface: The app's intuitive and user-friendly interface will offer a familiar experience to WhatsApp users, reducing the learning curve.

Disadvantages:

Competition: The messaging app market is highly competitive, with several well-established platforms, which may pose challenges in acquiring a significant user base.

User Retention: Maintaining user engagement and retention can be challenging, given the abundance of messaging apps available to users.

Infrastructure Costs: To support real-time messaging and media sharing, the project will require a robust infrastructure, potentially incurring significant operational costs.

Software Requirements:

Development Platforms: Android Studio (for Android) Realtime Messaging: Firebase Cloud Messaging (FCM) for real-time communication.

Push Notifications: Firebase Cloud Messaging (FCM) or OneSignal for push notifications.

References:

Firebase Cloud Messaging (FCM) - https://firebase.google.com/docs/cloud-messaging

WhatsApp - https://www.whatsapp.com/

CHAPTER 2: SURVEY OF TECHNOLOGIES

2.1 Introduction

The WhatsApp Clone App Project aims to develop a messaging application inspired by the popular WhatsApp platform, offering seamless communication, multimedia sharing, and real-time messaging between users. The application will be designed for Android and will leverage key features to provide a familiar and user-friendly experience, similar to WhatsApp, while also addressing certain shortcomings and incorporating innovative features.

2.3 Survey in the market

Conducting a survey in the market is crucial to understanding the preferences and expectations of potential users outside of the company. This survey will provide valuable insights into the messaging app landscape, competitor analysis, and user demands. Here are some key points to consider during the

market survey:

Target Audience: Define the target audience for the messaging app. Identify the demographics, age groups, and professions that are likely to use such an application.

Messaging App Usage: Determine which messaging apps are currently popular among the target audience. Understand the reasons behind their popularity and what features users find most appealing

User Behavior: Study how users interact with messaging apps. Identify how frequently they use these apps, the nature of their communications, and the types of content they commonly share.

Pain Points with Existing Apps: Explore the common issues users face with current messaging apps. This could include privacy concerns, lack of certain features, or user interfacerelated problems.

Feature Prioritization: Ask users about the features they consider most important in a messaging app. This could include end-to-end encryption, group chats, voice and video calling, multimedia sharing, etc. **User Interface Preferences:** Gather feedback on user interface preferences and design elements that users find visually appealing and easy to use.

Market Gaps: Identify any specific gaps in the messaging app market that the new WhatsApp Clone App can address. Look for unmet needs or areas where competitors are lacking.

Brand Perception: Understand how users perceive the WhatsApp brand and whether it influences their willingness to try a similar messaging app.

Monetization Strategies: Assess users' willingness to pay for premium features or subscription plans, as well as their acceptance of ads or other monetization methods.

Cross-Platform Support: Determine the importance of cross-platform compatibility among users who use multiple devices and operating systems.

Security Concerns: Gauge users' level of concern regarding the security and privacy of their communications on messaging apps.

Innovative Features: Ask users about any innovative or unique features they would like to see in a messaging app that could set it apart from existing options.

Marketing and Outreach: Gather insights into how users discover new apps and what marketing strategies would be most effective in promoting the WhatsApp Clone App.

By conducting a comprehensive market survey, the development team can gain valuable feedback and data to refine the messaging app's features, user experience, and overall value proposition. The goal is to create a product that resonates with the target audience and stands out in the competitive messaging app market.

2.4 Detailed explanation of one solution in market

One solution in the messaging app market that has gained significant popularity and success is "Telegram."

Telegram is a cloud-based instant messaging app that was launched in

2013 by Pavel Durov and his brother Nikolai Durov. It is known for its focus on security, speed, and user privacy. Here's a detailed explanation of some key features and aspects of Telegram:

Security and Privacy: One of Telegram's main selling points is its strong emphasis on security and privacy. It uses end-to-end encryption for secret chats, ensuring that only the intended recipients can read the messages. Additionally, Telegram's server-

client encryption also ensures that messages are encrypted in transit and stored in an encrypted format on their servers.

Cloud-Based: Telegram's cloud-based infrastructure allows users to access their messages and media from multiple devices seamlessly. This is a significant advantage for users who switch between phones, tablets, and computers regularly.

User Interface: Telegram boasts a clean and intuitive user interface, making it easy for users to navigate and use the app efficiently. The app offers a range of customization options, allowing users to personalize their chat backgrounds, themes, and even create custom stickers.

Group Chats and Channels: Telegram supports large group chats with thousands of members. Channel creation is another standout feature that allows users to broadcast messages to an unlimited number of subscribers. This feature has been widely adopted by news organizations, businesses, and community groups.

File Sharing and Multimedia: Telegram enables users to share various file types, including documents, images, videos, and more. The app supports media compression to reduce file sizes while maintaining quality. **Bots and APIs:** Telegram provides an open API and platform for developers to create bots that can perform various tasks within the app. These bots can be used for automated customer support, news updates, productivity tools, and more.

Voice and Video Calls: Telegram expanded its features to include voice and video calling, making it a more comprehensive communication platform.

Cross-Platform Support: Telegram is available on multiple platforms, including Android, iOS, Windows, macOS, and web browsers.

This ensures users can communicate seamlessly across different devices.

Speed: Telegram is known for its fast and efficient message delivery, making real-time communication a smooth experience even in lowbandwidth conditions.

Open Source: Telegram's client-side code is open-source, allowing security researchers to inspect the code and verify its encryption protocols.

Stickers and GIFs: Telegram offers an extensive library of stickers and GIFs, allowing users to express themselves creatively during conversations. Telegram's commitment to user privacy, security, and feature-rich functionality has led to its popularity, especially among users who prioritize these aspects in a messaging app. The platform has grown to millions of active users worldwide, making it one of the significant competitors in the messaging app market.

2.5 Proposed System, its advantages / disadvantages

The proposed WhatsApp Clone App aims to develop a messaging application inspired by the popular WhatsApp platform, offering seamless communication, multimedia sharing, and real-time messaging between users. The application will be designed for Android and will leverage key features to provide a familiar and user-friendly experience, similar to WhatsApp, while also addressing certain shortcomings and incorporating innovative features.

Advantages:

Familiar User Experience: By emulating the user experience of WhatsApp, the app can attract users who are already familiar with the layout and functionalities, making it easier for them to adapt to the new platform.

Seamless Communication: The app's real-time messaging capabilities ensure smooth and instantaneous communication, enabling users to engage in conversations without delays.

Multimedia Sharing: Supporting various file types, such as images, videos, and documents, facilitates easy multimedia sharing among users, enhancing collaboration and interaction.

Cross-Platform Compatibility: Designing the app for Android ensures that users can communicate seamlessly across different Android devices, providing convenience and accessibility.

Disadvantages:

Competition: The messaging app market is highly competitive, with several well-established platforms, which may pose challenges in acquiring a significant user base.

User Retention: Maintaining user engagement and retention can be challenging, given the abundance of messaging apps available to users.

Infrastructure Costs: To support real-time messaging and media sharing, the project will require a robust infrastructure, potentially incurring significant operational costs.

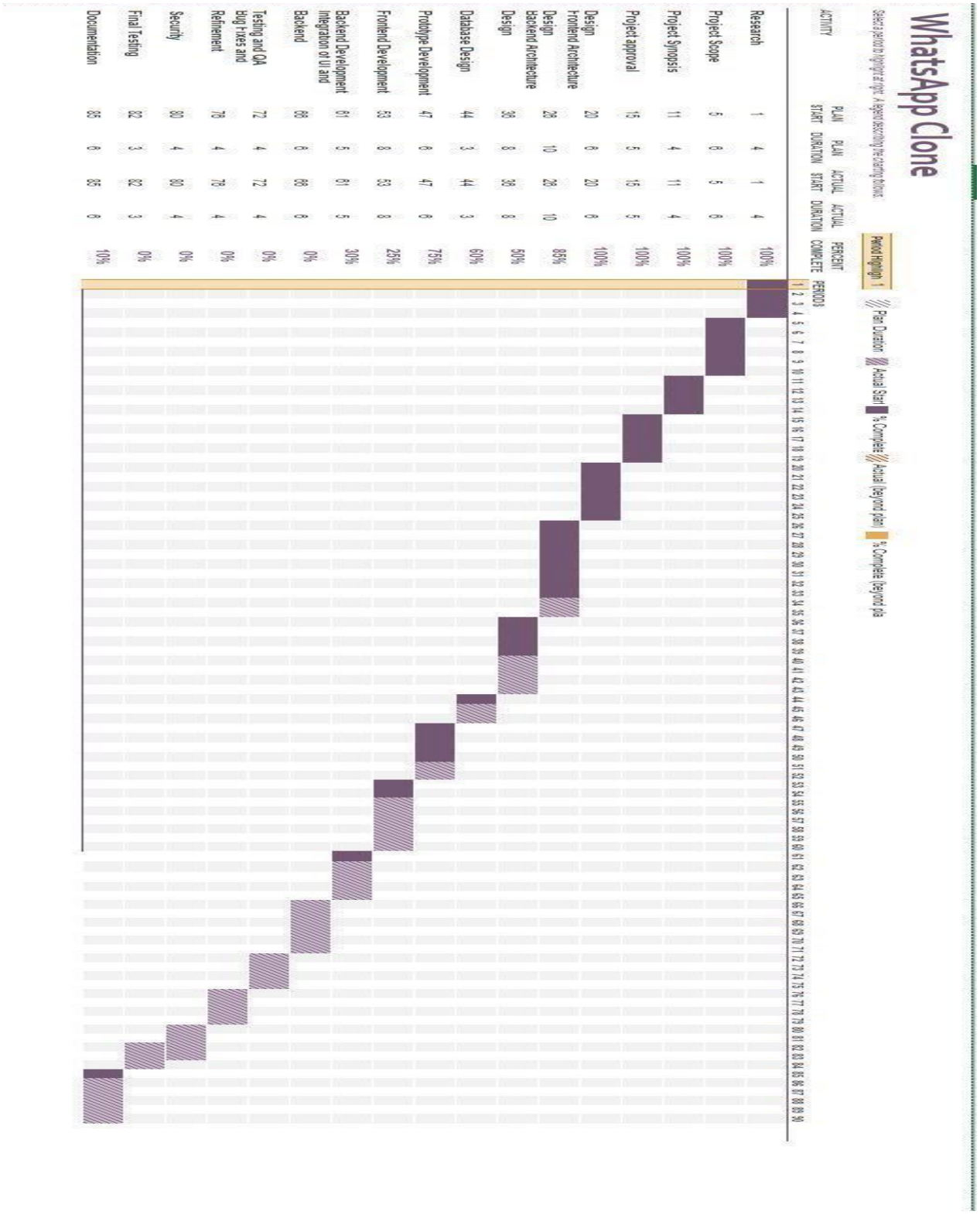
Software Requirements: Development Platforms: Android Studio
(for Android)

Real-time Messaging: Firebase Cloud Messaging (FCM) for real-time communication.

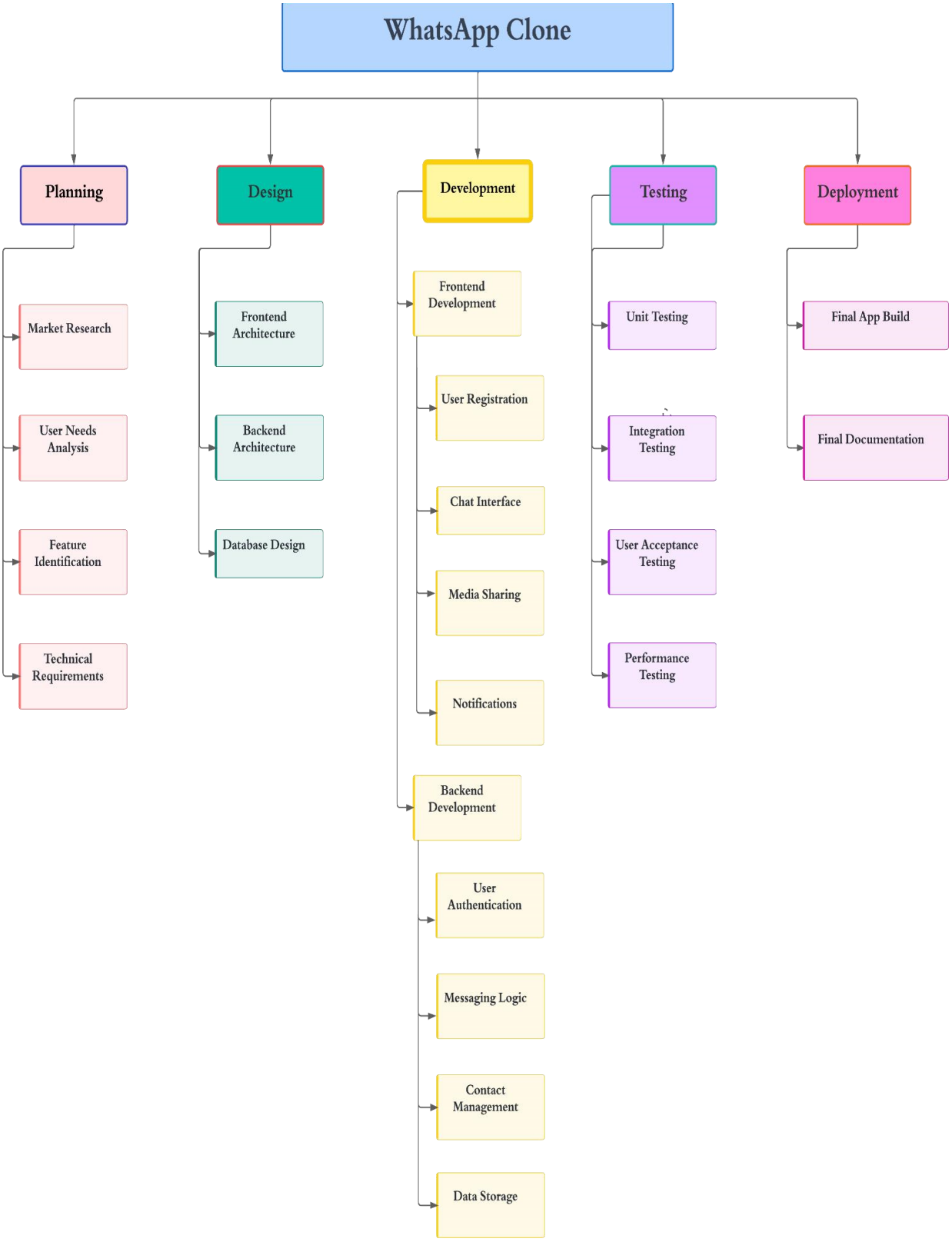
Push Notifications: Firebase Cloud Messaging (FCM) or OneSignal for push notifications.

CHAPTER 3: REQUIREMENTS

GANTT CHART



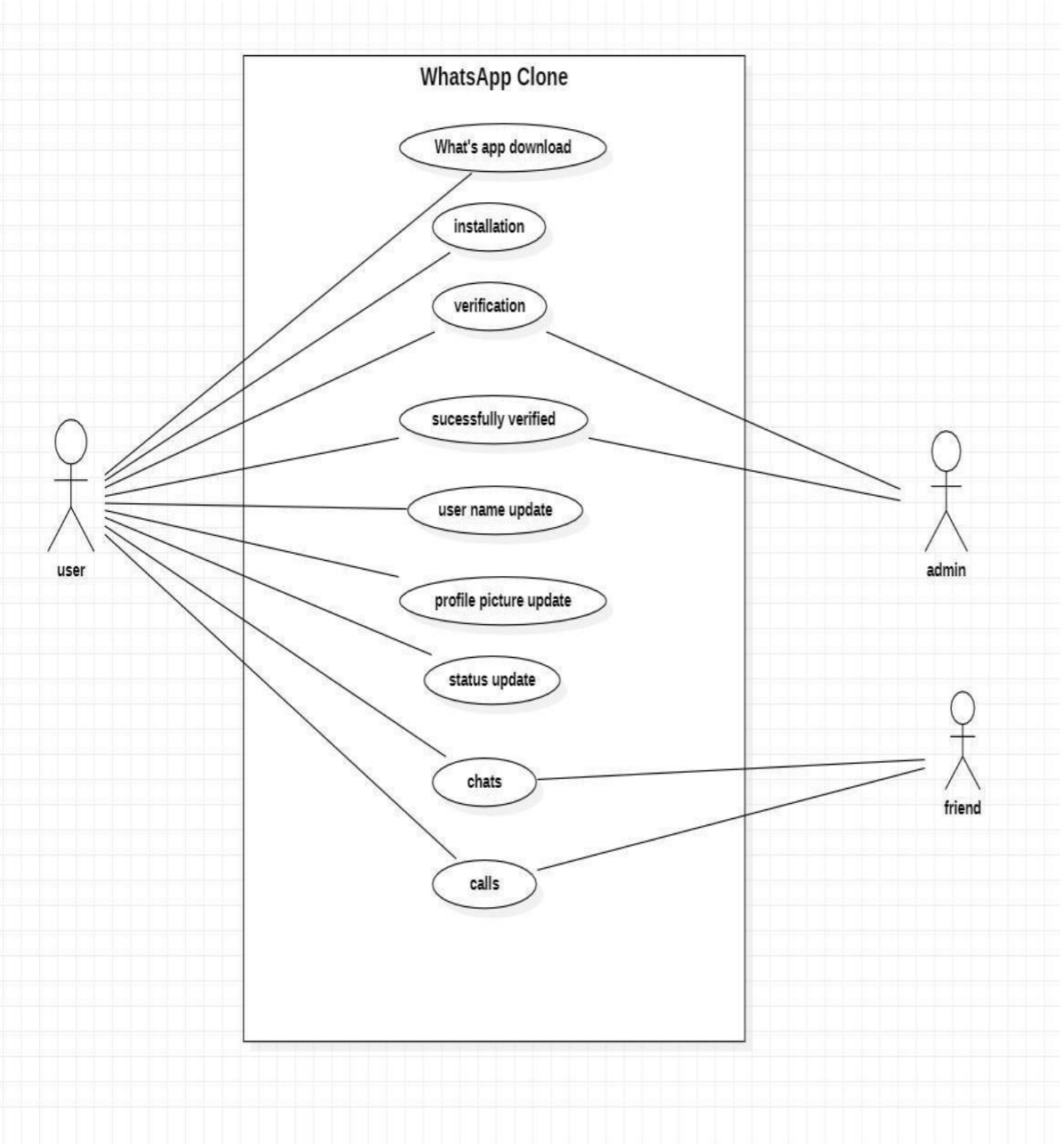
WBS CHART


WhatsApp Clone
Planning
Design
Development
Testing
Deployment
Market Research
User Needs Analysis
Feature Identification
Technical Requirements
Frontend Architecture
Backend Architecture
Database Design
Frontend Development
User Registration
Chat Interface
Media Sharing
Notifications
Backend Development
User Authentication
Messaging Logic
Contact Management
Data Storage
Unit Testing
Integration Testing
User Acceptance Testing
Performance Testing
Final App Build
Final Documentation

CHAPTER 4: DESIGN

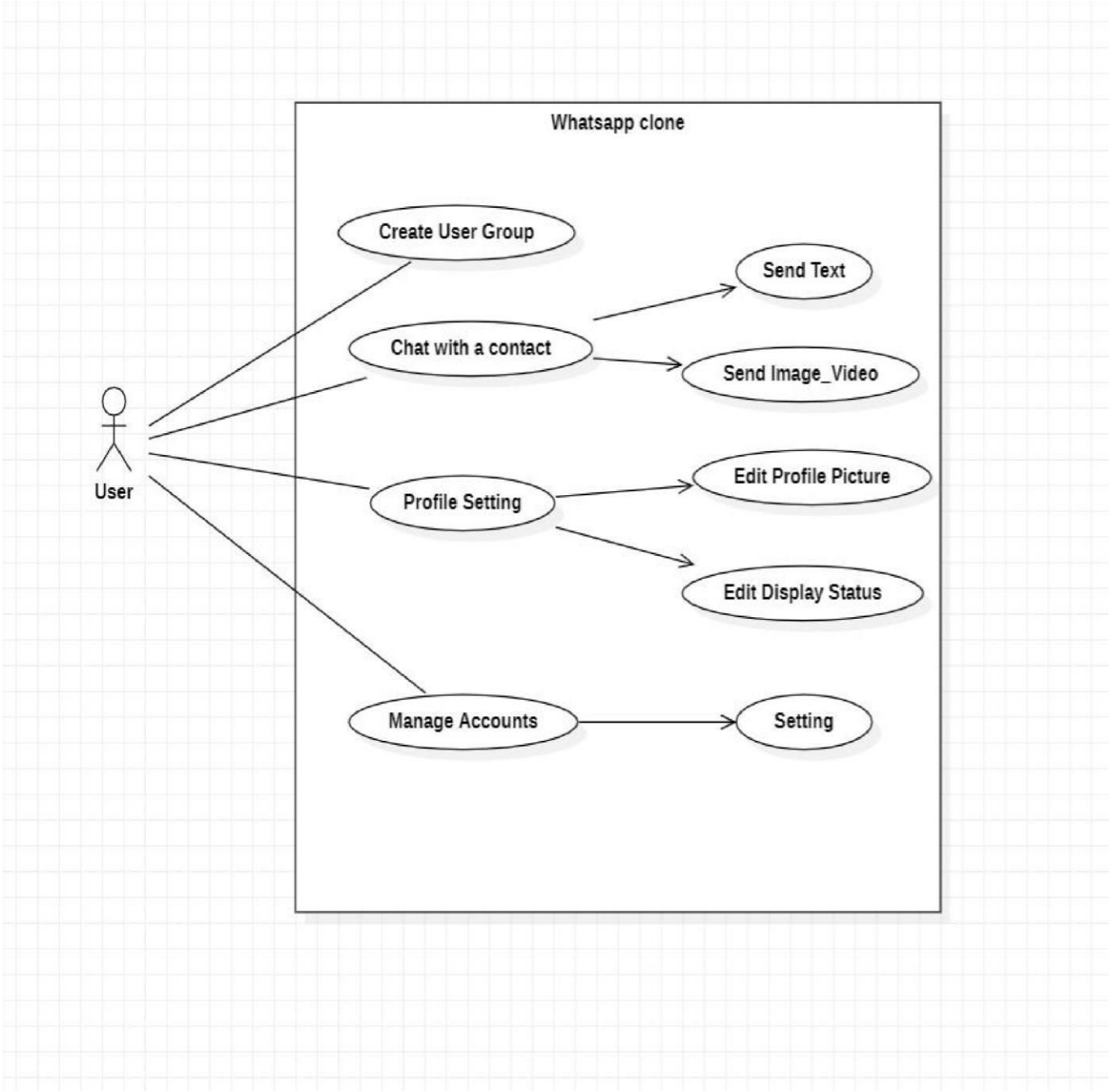
USE CASE DIAGRAM

1



In this use case Diagram the user will first download the app after that he will enter his email id and password for user verification than Verify it successfully from the admin. After that user will update his profile picture in the same way status Than the user chat with his friend and call.

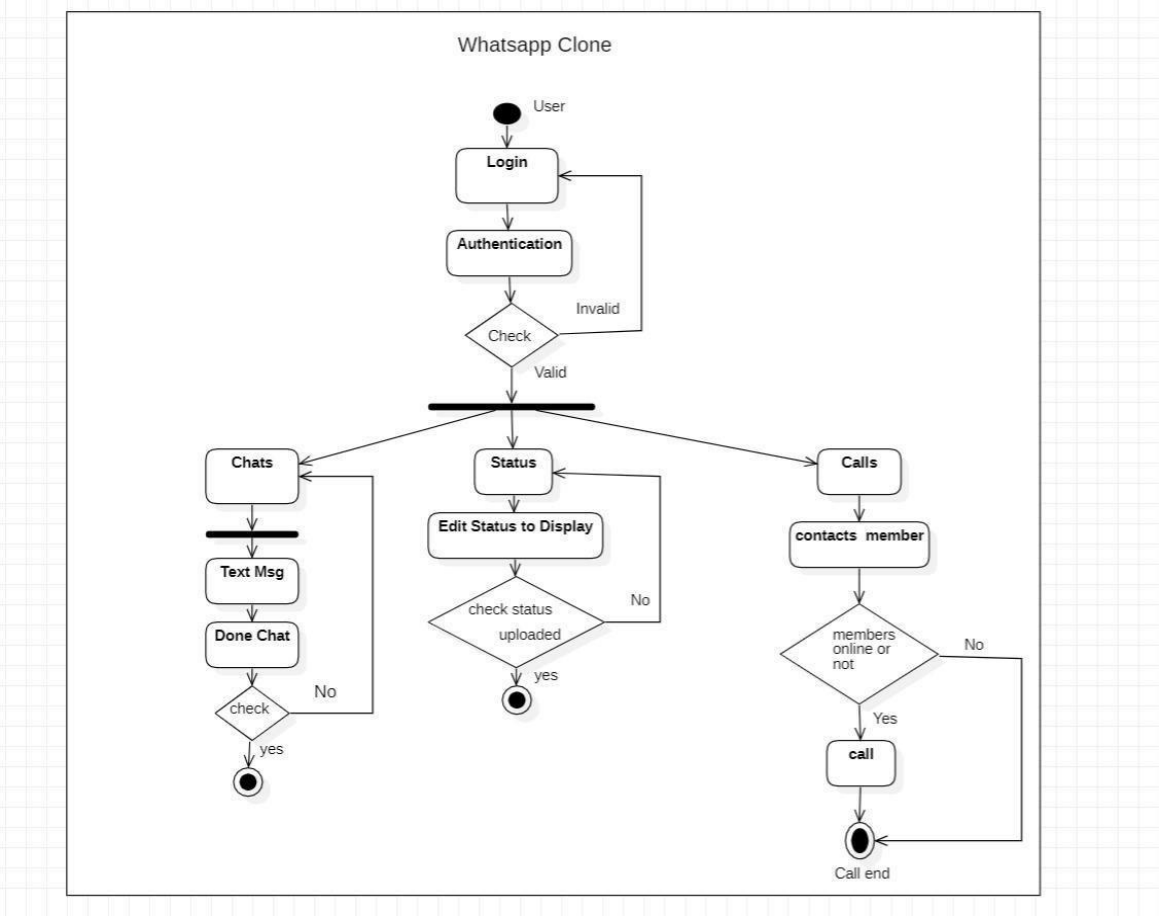
USE CASE DIAGRAM



In this use case Diagram the user have the power to create his own group, They can chat with hist contact members in the text format or they can send the media also. User can Manage his profile Picture and they manage also his account.

ACTIVITY DIAGRAM

1)



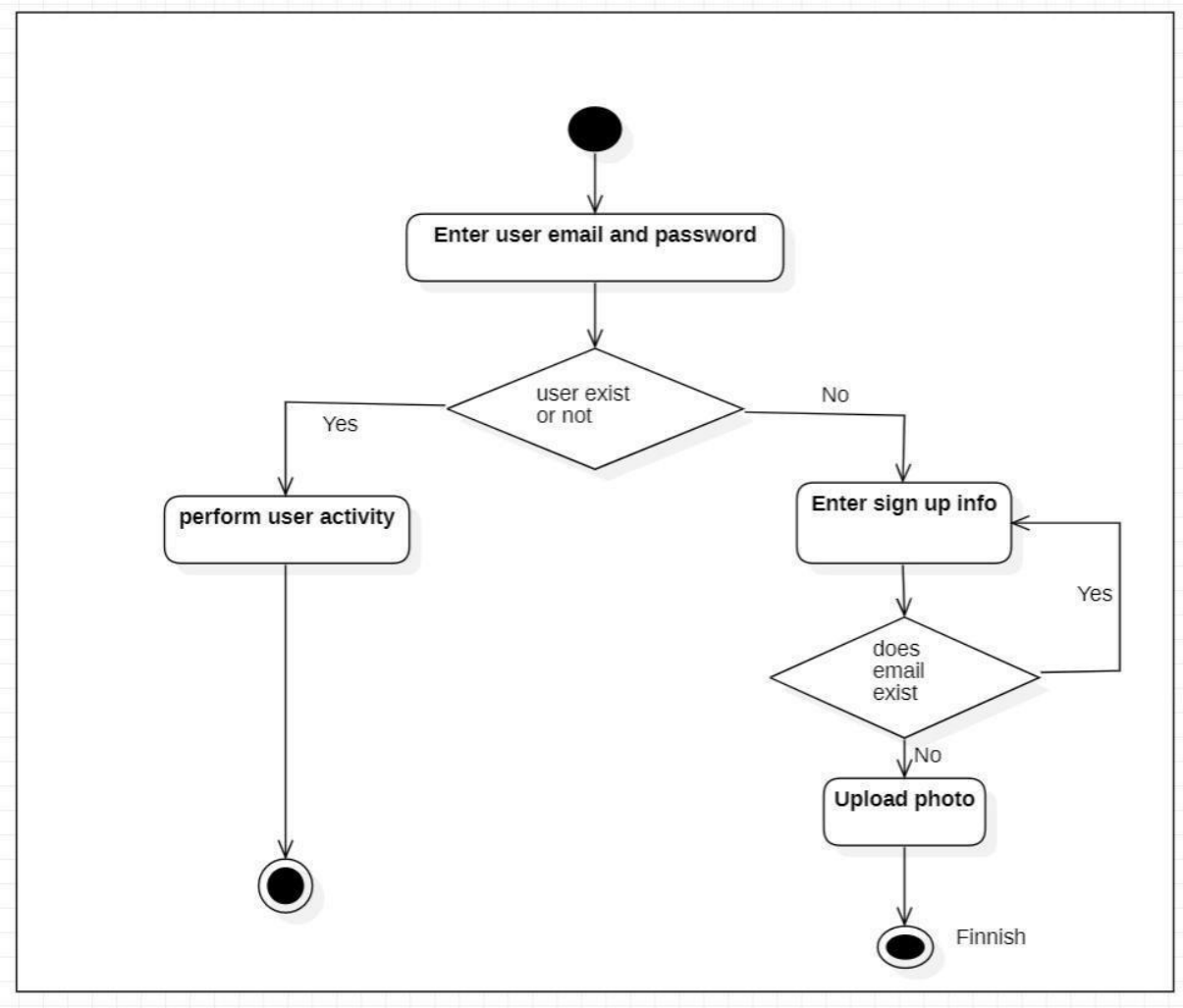
In this Activity Diagram, when user login the app email id and password than the user authentication check in valid or valid. If user authentication is invalid than user again go to the login activity than once again user inter email id and password than user authentication become valid than user have three interface Chats, Status, Calls. User can be Chat his contact members when chat his done or not check if no then chat continue if yes chat become Finnish.

User can be Update his status. User edit his status than her check the status is update or not if no than go to the first state if yes then Finnish.

After user call some contact members they will check member are online or not if No the call has been end if yes then call ring.

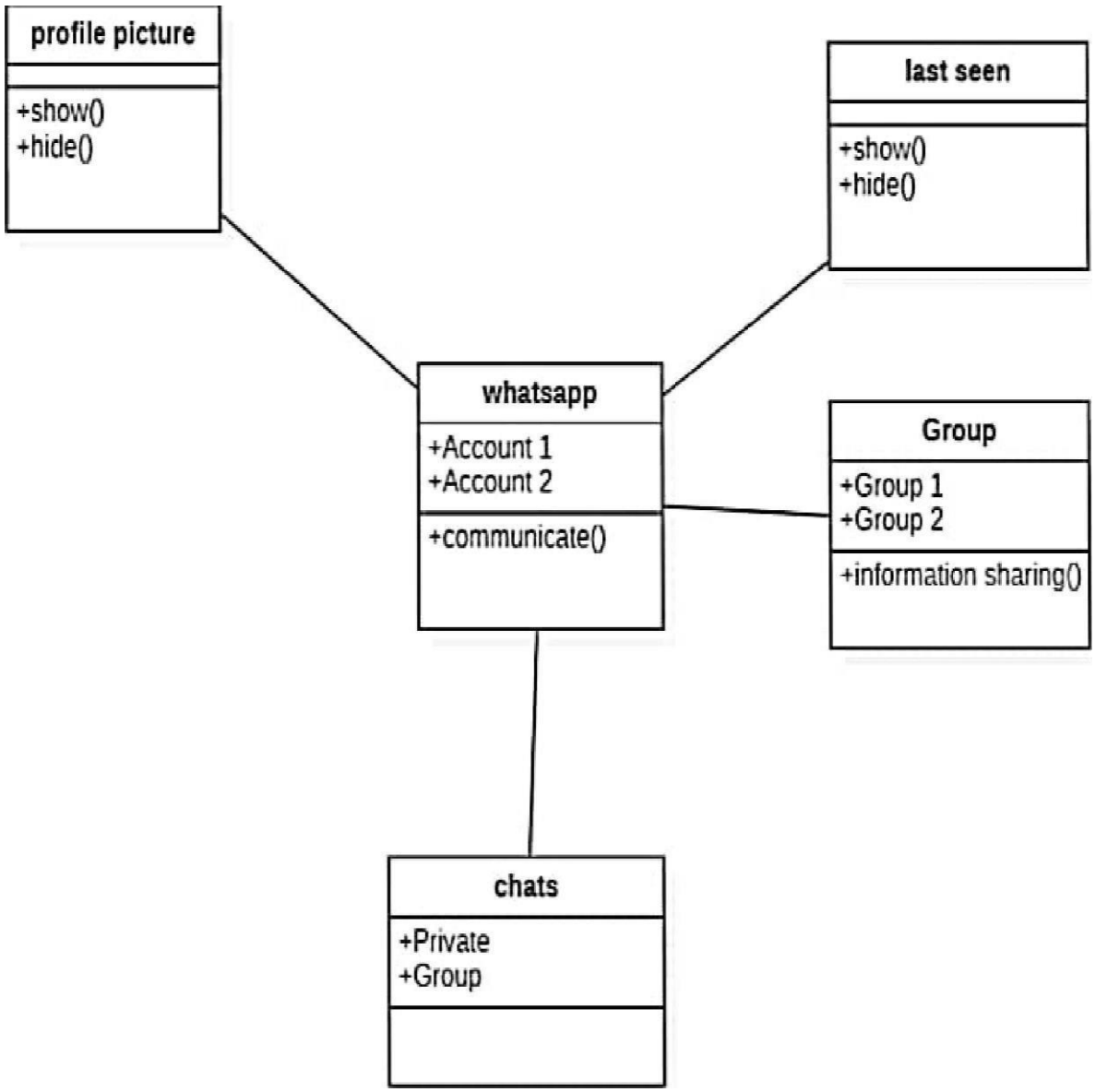
ACTIVITY DIAGRAM

2)

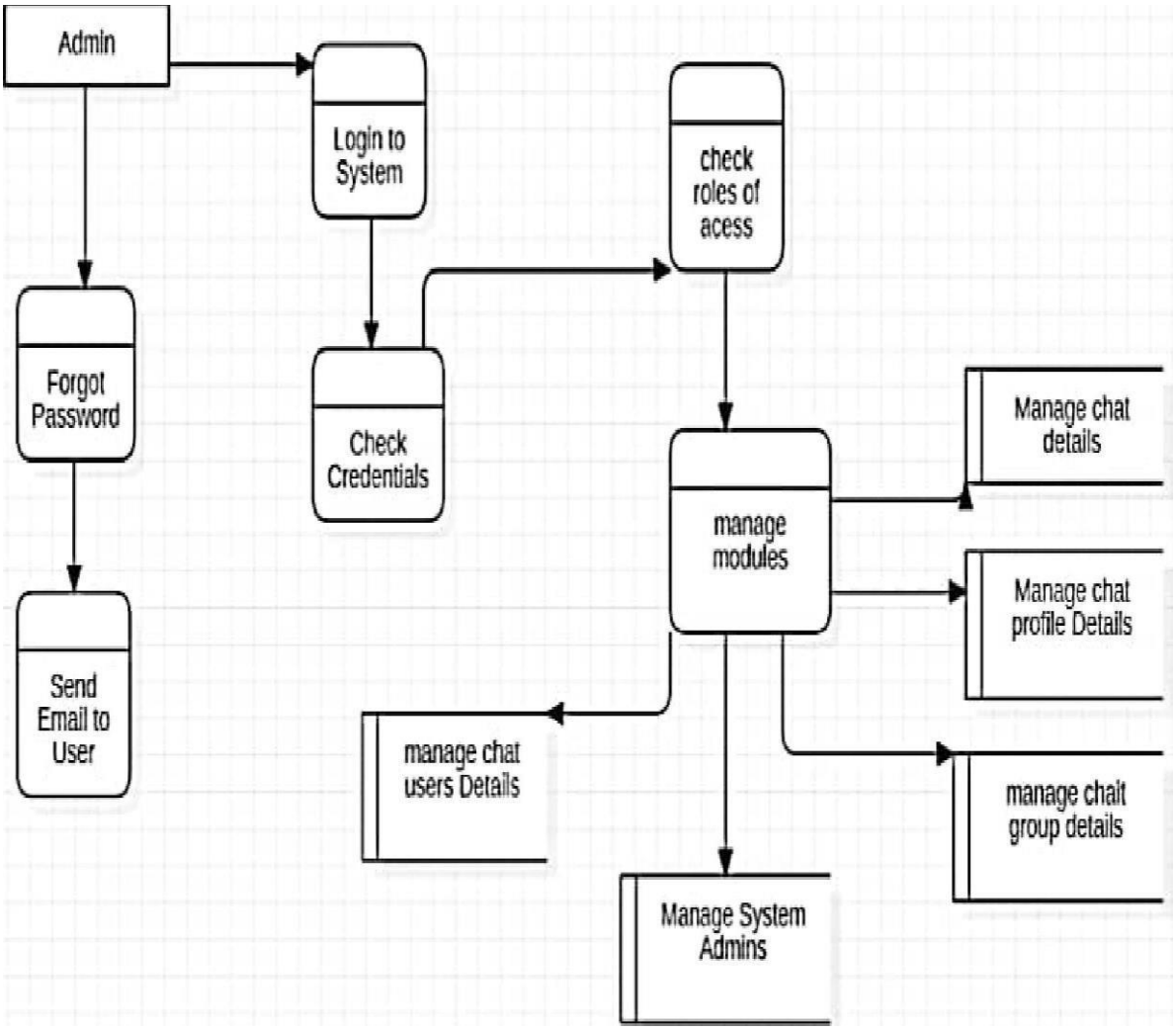


In this Activity Diagram, when the user Enter the email id and password then check the user exist or not if yes than perform user activity if yes than enter the Sign up info if sign up info already exist than re-enter sign up info if no than upload photo and perform user activity.

CLASS DIAGRAM



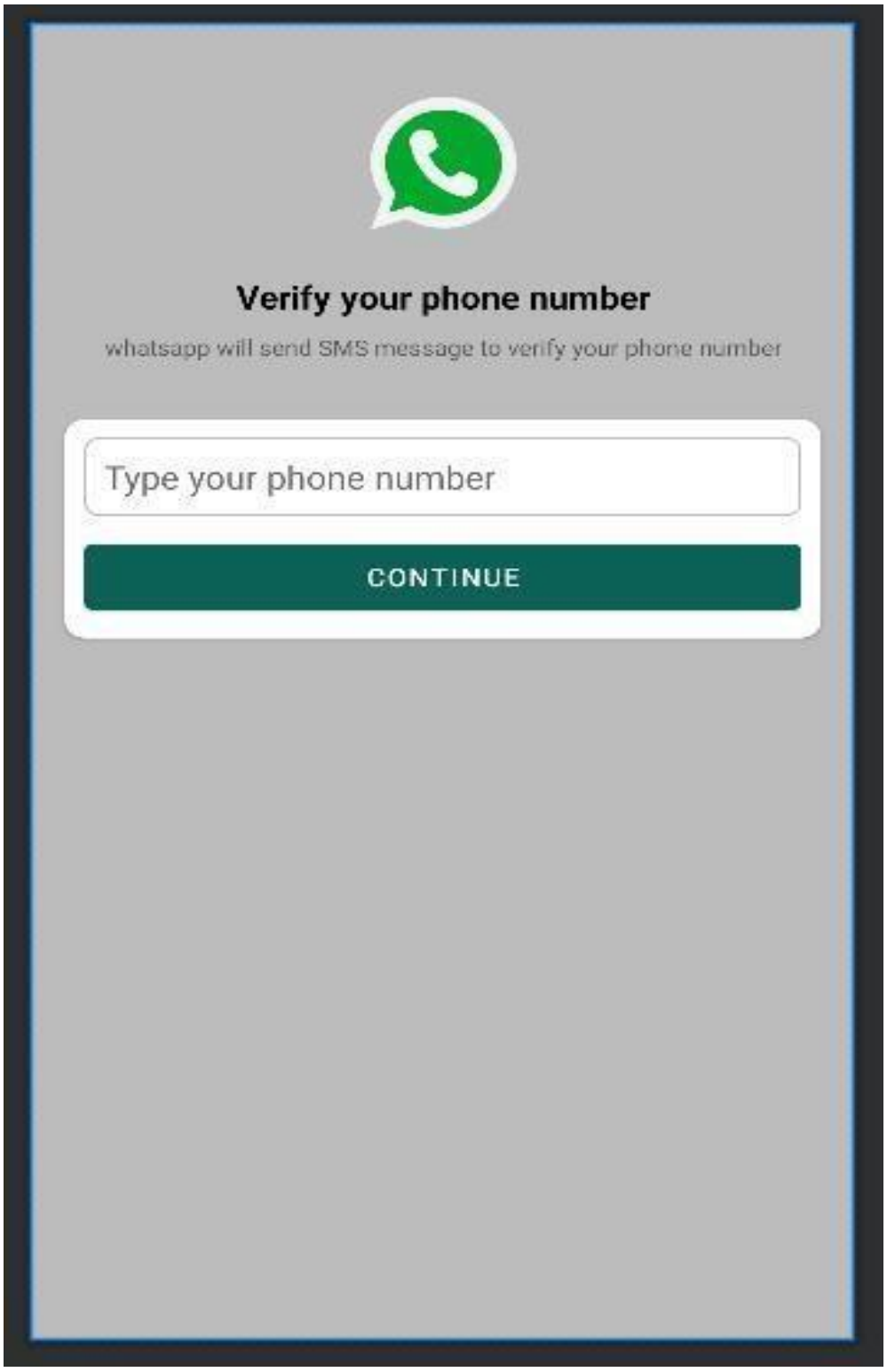
DFD DAIGRAM



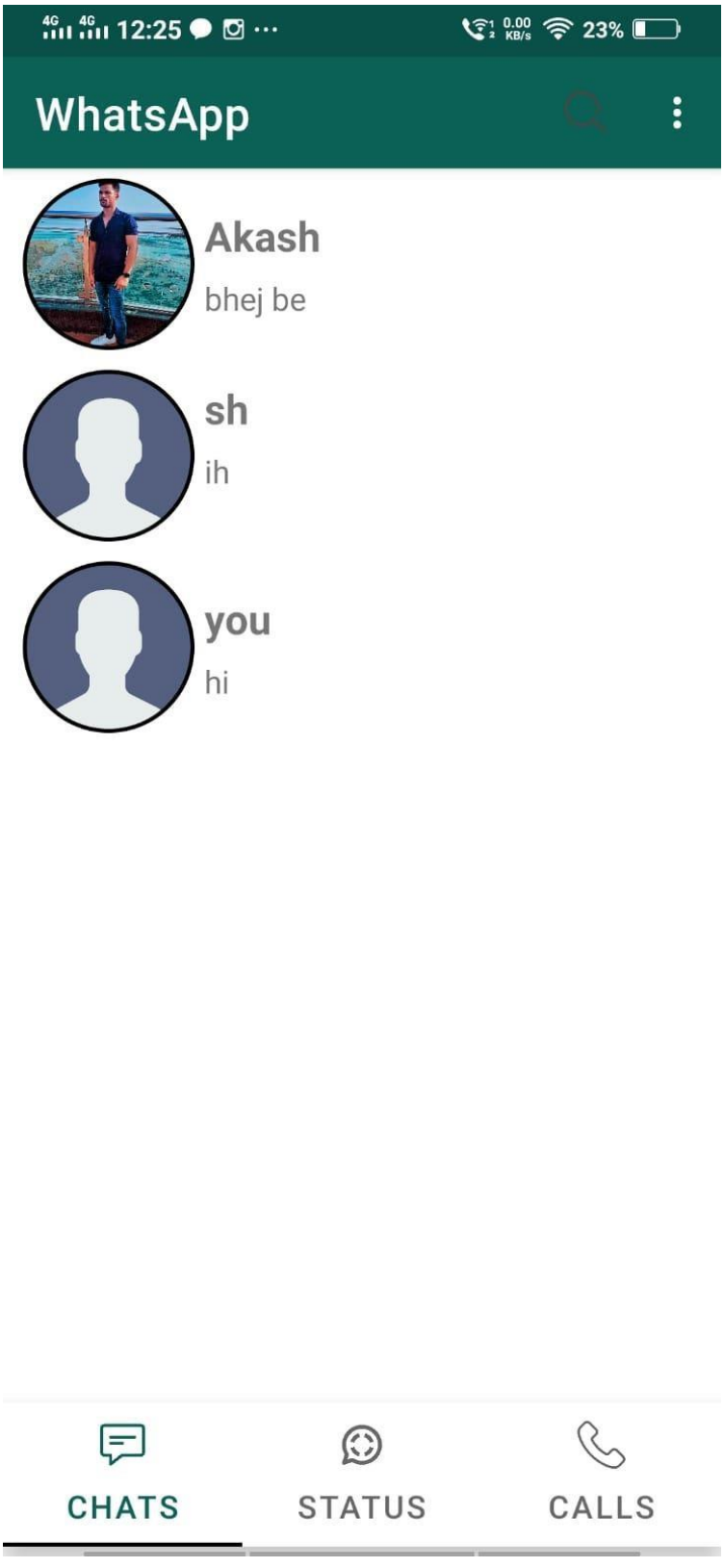
CHAPTER 5: FINAL GUI

WHATSAPP GUI DESIGNS

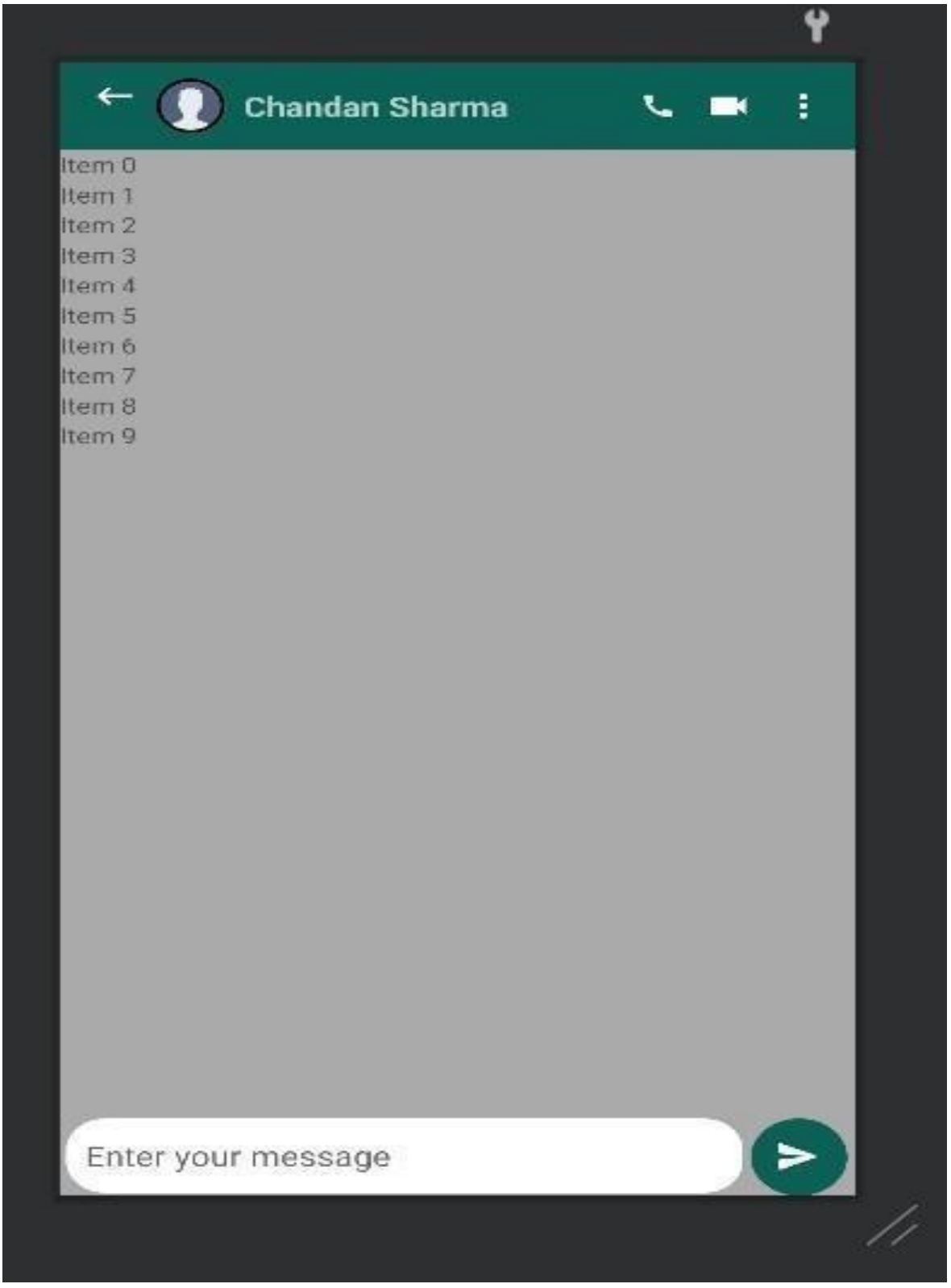
1) SIGNUP

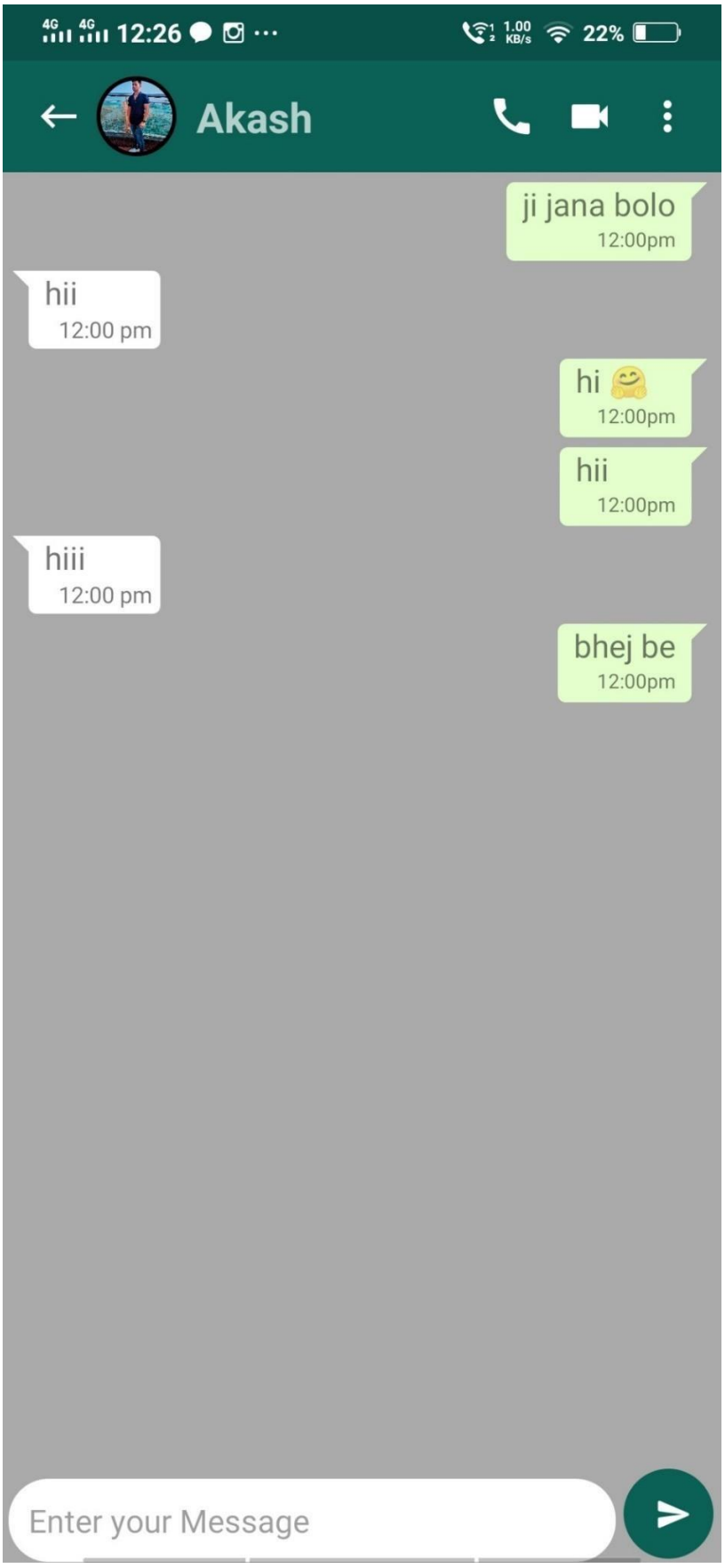


Main interface

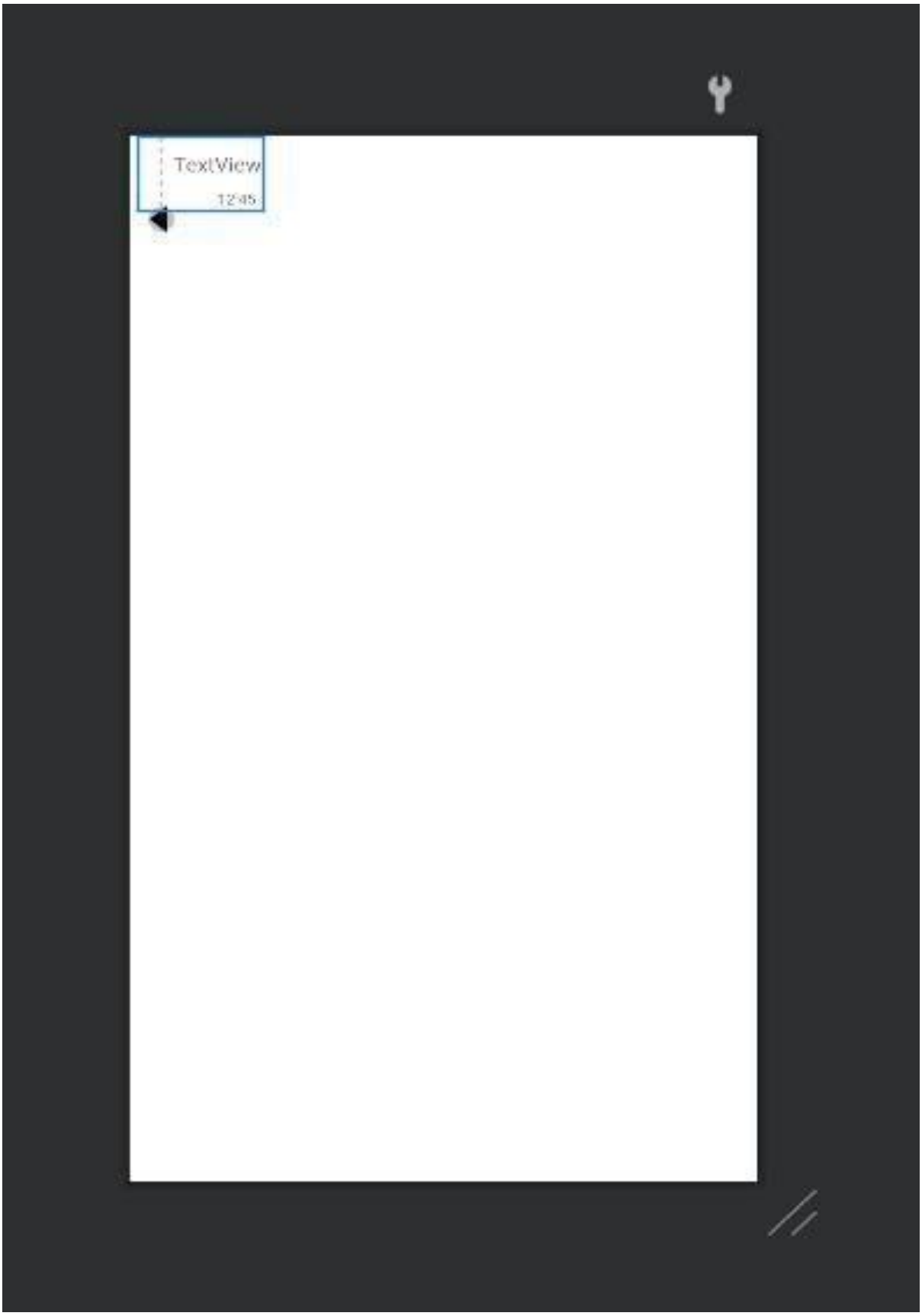

12:25
0.00 KB/s
23%
WhatsApp
Akash
bhej be
sh
ih
you
hi
CHATS
STATUS
CALLS

2) CHATS

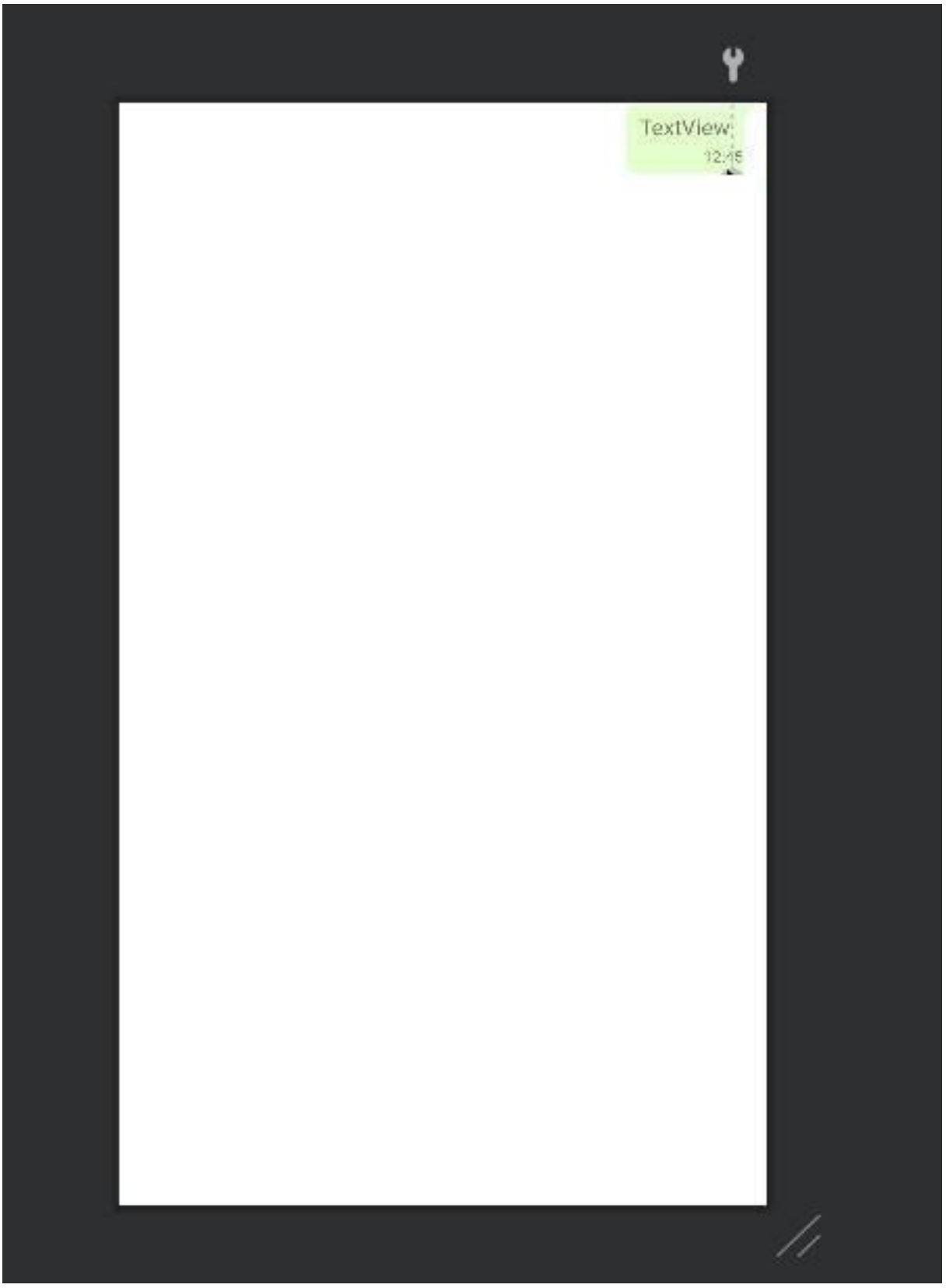

Chandan Sharma
Item 0
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 9
Enter your message


12:26
1.00 KB/s
22%
Akash
ji jana bolo
12:00pm
hii
12:00 pm
hi
12:00pm
hii
12:00pm
hiii
12:00 pm
bhej be
12:00pm
Enter your Message

4) RECIVER

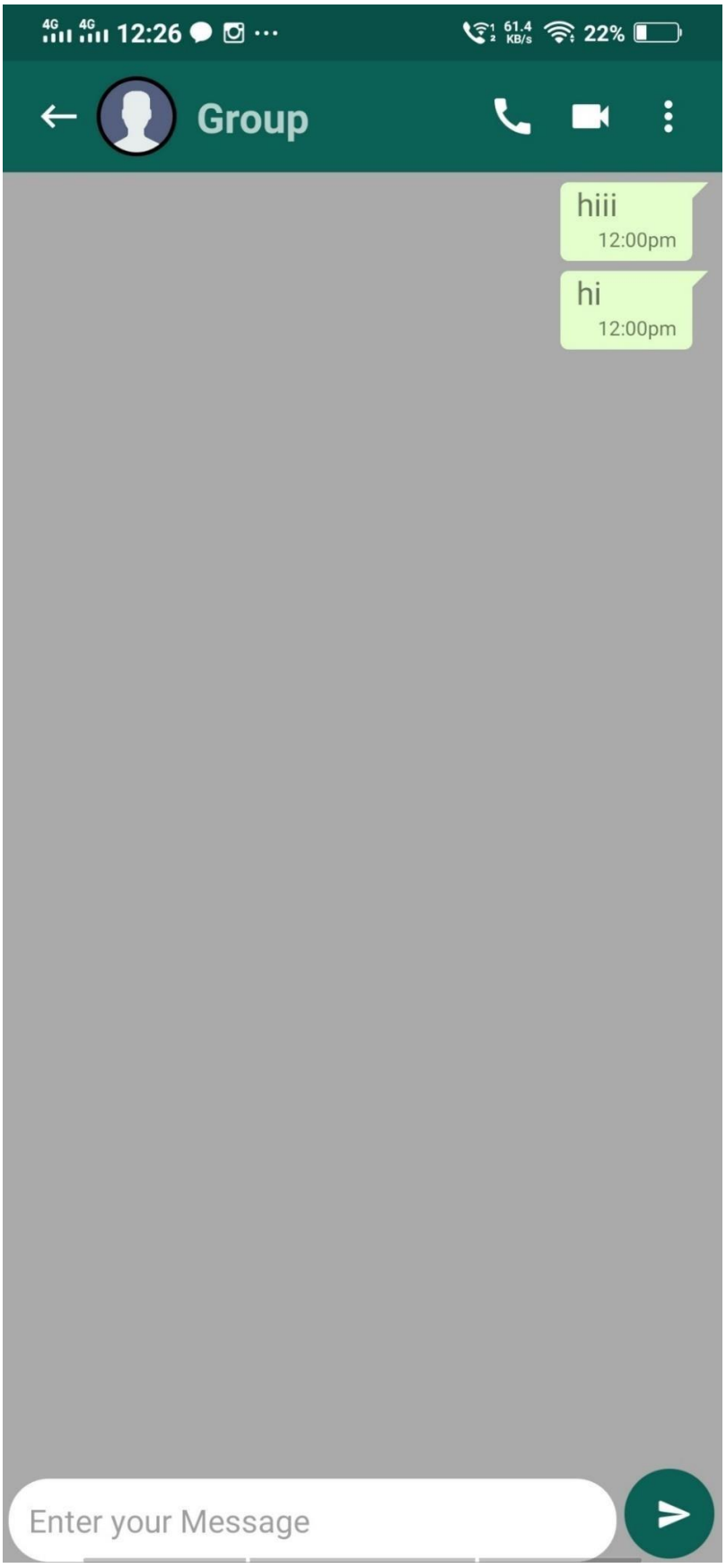


5) SENDER

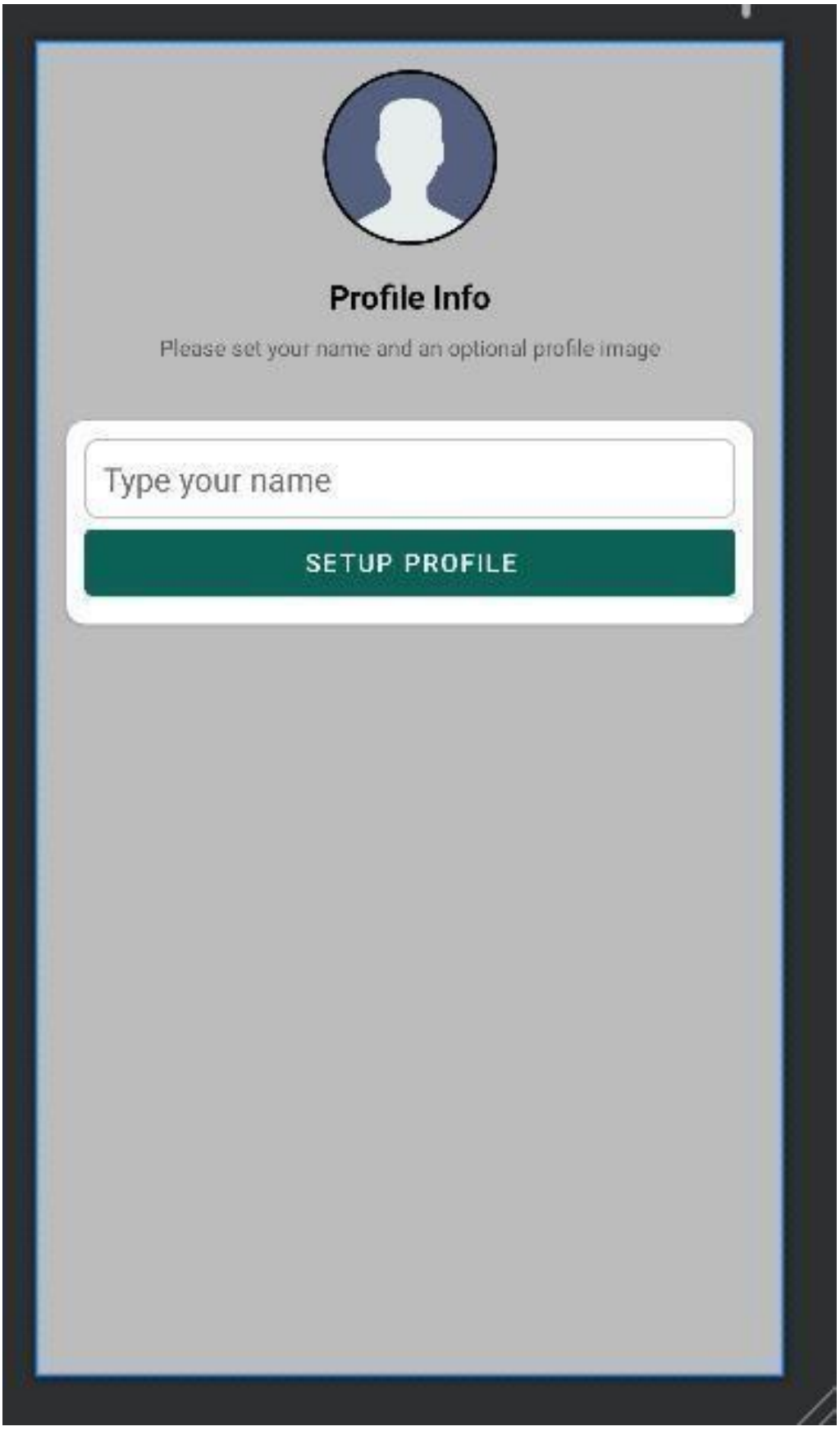


6)GROUP CHAT


Chandan Sharma
Item 0
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 9
Enter your message



7} SET PROFILE


Profile Info
Please set your name and an optional profile image
Type your name
SETUP PROFILE

CHAPTER 6: IMPLEMENTATION AND TESTING

6.1 CODING DETAILS

OtpVerifyActivity.class

```
package com.ck.WhatsApp.Activities; import
androidx.annotation.NonNull; import
androidx.appcompat.app.AppCompatActivity;
import android.app.ProgressDialog; import
android.content.Intent; import android.os.Bundle; import
android.view.View; import android.widget.Toast; import
com.ck.WhatsApp.Activities.SetupProfileActivity; import
com.ck.WhatsApp.databinding.ActivityOtpVerifyBinding; import
com.google.android.gms.tasks.OnCompleteListener; import
com.google.android.gms.tasks.Task; import
com.google.firebase.FirebaseException; import
com.google.firebase.auth.AuthResult; import
com.google.firebase.auth.FirebaseAuth; import
com.google.firebase.auth.PhoneAuthCredential; import
com.google.firebase.auth.PhoneAuthOptions; import
com.google.firebase.auth.PhoneAuthProvider; import
java.util.concurrent.TimeUnit;

public class OtpVerifyActivity extends AppCompatActivity {

   private ActivityOtpVerifyBinding binding;
   FirebaseAuth auth;
   String verificationId;
ProgressDialog dialog;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);        binding =
ActivityOtpVerifyBinding.inflate(getLayoutInflater());
setContentView(binding.getRoot());
```

```
        dialog = new ProgressDialog(this);
dialog.setMessage("Sending OTP...");
dialog.setCancelable(false);        dialog.show();


        getSupportActionBar().hide();

        auth = FirebaseAuth.getInstance();

        String phoneNumber =
getIntent().getStringExtra("phoneNumber");
// Ensure phoneNumber starts with "+91"
if (!phoneNumber.startsWith("+91")) {
phoneNumber = "+91" + phoneNumber;
        }
        binding.phoneLbl.setText("Verify " + phoneNumber);

        PhoneAuthOptions options =
PhoneAuthOptions.newBuilder(auth)
                .setPhoneNumber(phoneNumber)
                .setTimeout(60L, TimeUnit.SECONDS)
                .setActivity(OtpVerifyActivity.this)
                .setCallbacks(new
PhoneAuthProvider.OnVerificationStateChangedCallbacks() {
                    @Override
                    public void onVerificationCompleted(@NonNull
PhoneAuthCredential phoneAuthCredential) {}


                    @Override
                    public void onVerificationFailed(@NonNull
FirebaseException e) {}


                    @Override
                    public void onCodeSent(@NonNull String verifyId,
@NonNull PhoneAuthProvider.ForceResendingToken
forceResendingToken) {
                        super.onCodeSent(verifyId, forceResendingToken);
dialog.dismiss();                        verificationId = verifyId;
```

```
            }
}).build();
      PhoneAuthProvider.verifyPhoneNumber(options);

      binding.continueBtn.setOnClickListener(new
View.OnClickListener() {
         @Override          public
void onClick(View v) {
            if(binding.otpView.getText().toString().trim().isEmpty()){
               Toast.makeText(OtpVerifyActivity.this, "OTP is not
Valid", Toast.LENGTH_SHORT).show();
            } else {
               if(verificationId != null){
String otp =
binding.otpView.getText().toString().trim();

                  PhoneAuthCredential credential =
PhoneAuthProvider.getCredential(verificationId, otp);


FirebaseAuth.getInstance().signInWithCredential(credential).addOn
CompleteListener(new OnCompleteListener<AuthResult>() {
                     @Override
                     public void onComplete(@NonNull Task<AuthResult>
task) {
                        if(task.isSuccessful()){
                           Toast.makeText(OtpVerifyActivity.this, "Login
successful", Toast.LENGTH_SHORT).show();
                           Intent intent = new
Intent(OtpVerifyActivity.this, SetupProfileActivity.class);

intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
startActivity(intent);
                        } else {
                           Toast.makeText(OtpVerifyActivity.this, "OTP is
not valid", Toast.LENGTH_SHORT).show();
```

```
                    }
                }
            });
        }
    }
  });
 }
}
    /* // editTextInput();


binding.tvMobile.setText(String.format("+91%S",getIntent().getStrin
gExtra("phone")));
    verificationId = getIntent().getStringExtra("verificationId");
binding.tvResendOtp.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
          Toast.makeText(OtpVerifyActivity.this, "OTP Send
Successfully", Toast.LENGTH_SHORT).show();

        }
    });
    binding.btnVerify.setOnClickListener(new
View.OnClickListener() {

        @Override          public void onClick(View v) {
binding.progressBarVerify.setVisibility(View.VISIBLE);
binding.btnVerify.setVisibility(View.INVISIBLE);
if(binding.tvOtp.getText().toString().trim().isEmpty()){
            Toast.makeText(OtpVerifyActivity.this, "OTP is not
Valid", Toast.LENGTH_SHORT).show();
          }else{
            if(verificationId != null){
              String code = binding.tvOtp.getText().toString().trim();
```

```
                PhoneAuthCredential credential =
PhoneAuthProvider.getCredential(verificationId, code);
                FirebaseAuth
                    .getInstance().signInWithCredential(credential)
.addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                  @Override
                  public void onComplete(@NonNull Task<AuthResult>
task) {
                    if(task.isSuccessful()){

//binding.progressBarVerify.setVisibility(View.VISIBLE);

//binding.btnVerify.setVisibility(View.INVISIBLE);

                        Intent intent = new
Intent(OtpVerifyActivity.this,MainActivity.class);

intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FL
AG_ACTIVITY_NEW_TASK);
                        startActivity(intent);
                      }else {
                        //
binding.progressBarVerify.setVisibility(View.GONE);

//binding.btnVerify.setVisibility(View.VISIBLE);
                        Toast.makeText(OtpVerifyActivity.this, "OTP is
not valid", Toast.LENGTH_SHORT).show();
                      }
                    }
                  });
              }
            }
          }
});
```

```
   }*/
  /* private void editTextInput() {
       binding.tvOtp.addTextChangedListener(new TextWatcher() {
           @Override
           public void beforeTextChanged(CharSequence s, int start, int
count, int after) {

}

           @Override
           public void onTextChanged(CharSequence s, int start, int
before, int count) {

}

           @Override
           public void afterTextChanged(Editable s) {

           }
       });
   }*/
//}
```

SendOtpActivity.class

```
package com.ck.WhatsApp.Activities;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent; import
android.os.Bundle;
import android.view.View;

import com.ck.WhatsApp.databinding.ActivitySendOtpBinding;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.PhoneAuthProvider;

public class SendOtpActivity extends AppCompatActivity {

   private       ActivitySendOtpBinding       binding;
private FirebaseAuth auth;
   private     PhoneAuthProvider.OnVerificationStateChangedCallbacks
mCallbacks;

   @Override     protected void onCreate(Bundle
savedInstanceState) {         super.onCreate(savedInstanceState);
binding = ActivitySendOtpBinding.inflate(getLayoutInflater());
setContentView(binding.getRoot());         auth =
FirebaseAuth.getInstance();         if(auth.getCurrentUser() != null){
          Intent intent = new
Intent(SendOtpActivity.this,MainActivity.class);
          startActivity(intent);
finish();
       }
       getSupportActionBar().hide();



       binding.continueBtn.setOnClickListener(new
View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {

            Intent intent = new
Intent(SendOtpActivity.this,OtpVerifyActivity.class);

intent.putExtra("phoneNumber",binding.phoneBox.getText().toString(
)
);
            startActivity(intent);

        }
});

   }
}
     /* binding.btnGetOtp.setOnClickListener(new
View.OnClickListener() {
        @Override           public void
onClick(View v) {
            if(binding.etPhone.getText().toString().trim().isEmpty()){
               Toast.makeText(SendOtpActivity.this, "Invalid phone
Number", Toast.LENGTH_SHORT).show();

            }else
if(binding.etPhone.getText().toString().trim().length()!=10 ){
               Toast.makeText(SendOtpActivity.this, "Type valid phone
Number", Toast.LENGTH_SHORT).show();
            }else                {
otpSend();
            }
        }
      });
   }
   private void otpSend() {
```

```
        binding.progressBar.setVisibility(View.VISIBLE);
binding.progressBar.setVisibility(View.INVISIBLE);

        mCallbacks = new
PhoneAuthProvider.OnVerificationStateChangedCallbacks()             {
@Override
            public void onVerificationCompleted(@NonNull
PhoneAuthCredential credential) {
            }

            @Override
            public void onVerificationFailed(@NonNull FirebaseException
e) {
              binding.progressBar.setVisibility(View.GONE);
binding.btnGetOtp.setVisibility(View.VISIBLE);
                Toast.makeText(SendOtpActivity.this,
e.getLocalizedMessage(), Toast.LENGTH_SHORT).show();
            }

            @Override              public void onCodeSent(@NonNull
String verificationId,
                                   @NonNull
PhoneAuthProvider.ForceResendingToken token) {

              binding.progressBar.setVisibility(View.GONE);
binding.btnGetOtp.setVisibility(View.VISIBLE);
                Intent intent = new
Intent(SendOtpActivity.this,OtpVerifyActivity.class);
                startActivity(intent);

intent.putExtra("phone",binding.etPhone.getText().toString().trim());
intent.putExtra("verificationId",verificationId);
            }
        };
        PhoneAuthOptions options =
```

```
            PhoneAuthOptions.newBuilder(mAuth)
.setPhoneNumber("+91"+
binding.etPhone.getText().toString().trim())          // Phone number to
verify
                    .setTimeout(60L, TimeUnit.SECONDS) // Timeout and
unit
                    .setActivity(this)                // (optional) Activity for
callback binding
                    // If no activity is passed, reCAPTCHA verification can
not be used.
                    .setCallbacks(mCallbacks)         //
OnVerificationStateChangedCallbacks
                    .build();
        PhoneAuthProvider.verifyPhoneNumber(options);
    }
}*/
```

Test Cases

• Test cases using phone to sing in

Test No	Test case	Expected Result	Pass/Fail
1	Opening the WhatsApp app	Show the login page	pass
2	Enter valid mobile number	Text box contain number	pass
3	Check the mobile number is valid or not	Show error massage	pass
4	Check if mobile number is valid than sand the OTP one the number	Go to the OTP verify page	Pass
5	Enter the OTP in the text box	Text box Desplayed and contain the OTP	pass
6	Enter OTP is Valid or not if valid	Loding the page	Pass
7	After loding the page go to the browser and its check you are human or not and varify it	Automatic verify	pass
8	After varify is done than go to the home page	Show the home page	pass

Test-case of home page

Test No	Test case	Expected Result	Pass/Fail
1	Verify that home page is displayed after login or not.	User will directed to home page	pass
2	logOut to the Whatsapp after click logOut	Loding and succsesfull logOut	pass

Chat Detail Activity

Test no	Test case	Expected Result	Pass/Fail
1	UI Elements Test	the activity initializes without crashing and Firebase instances are initialized properly	Pass

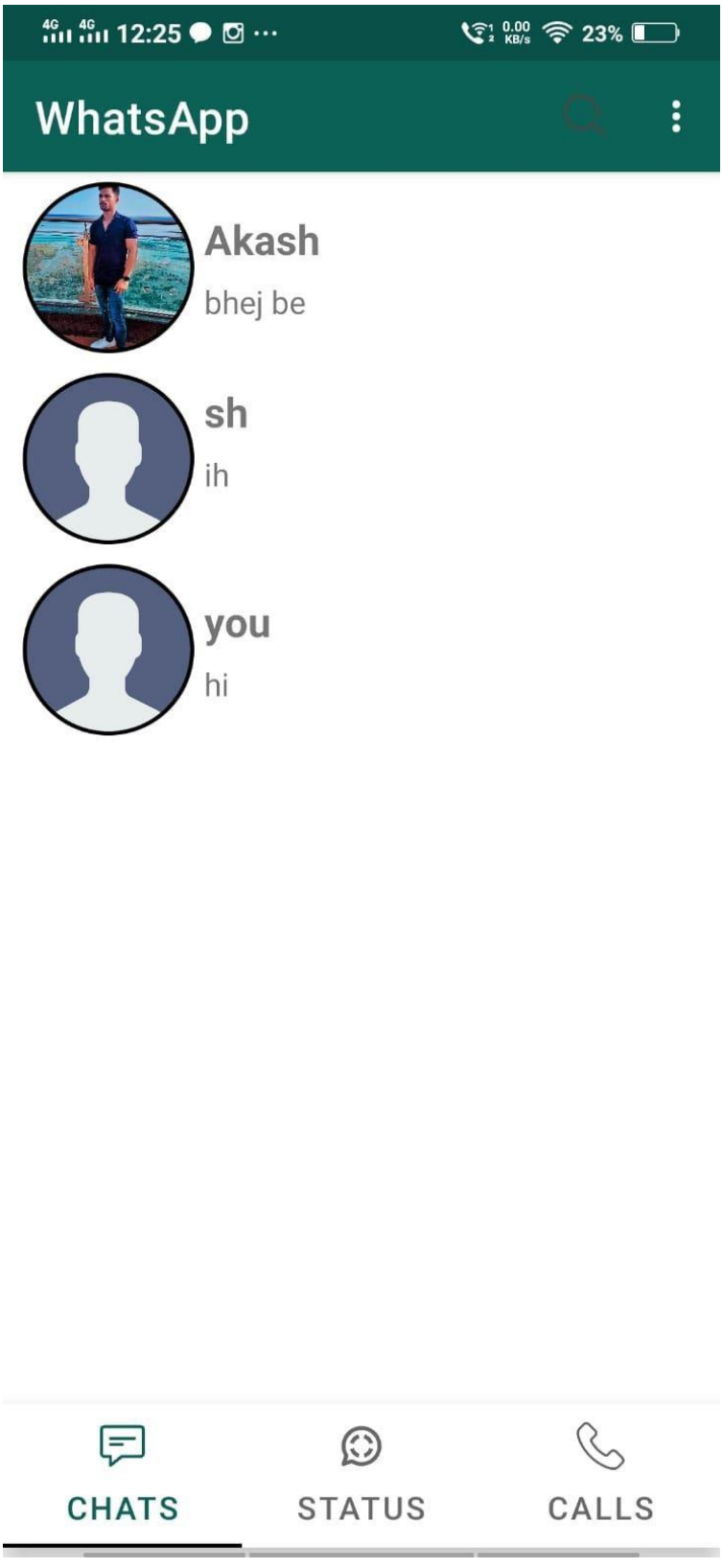
2	Data Retrieval Test	Verify that sender and receiver IDs are retrieved correctly from the intent, and messages are fetched from the Firebase Realtime Database.	pass
3	Sending Message Test	Test sending a message and ensure it appears in both the sender's and receiver's rooms in the Firebase Realtime Database.	pass
4	Activity Navigation Test	Verify that clicking the back arrow button navigates back to the Main Activity.	pass
5	Performance Test	Test the performance of loading and sending messages, especially with a large number of messages or high network latency.	pass

Setup Profile Activity

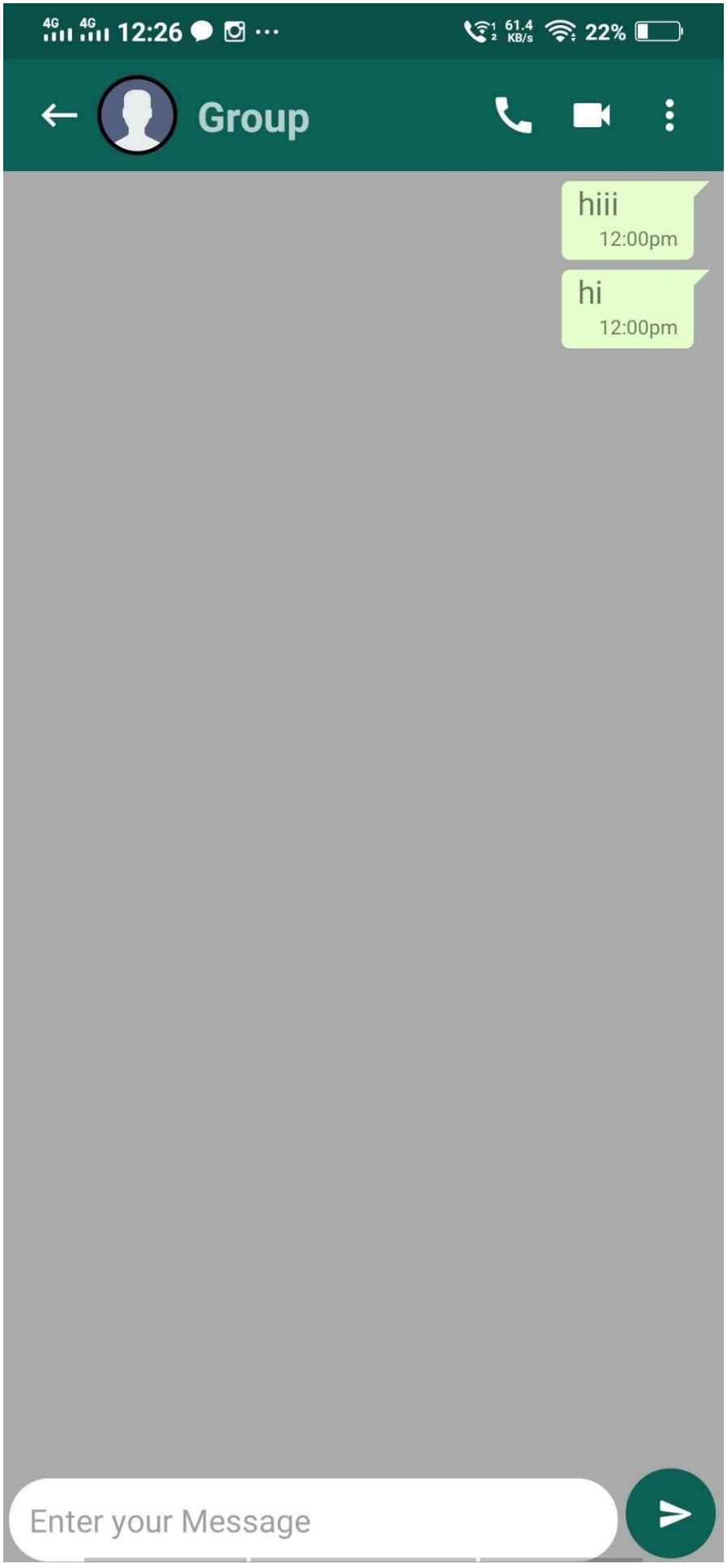
Test no	Test case	Expected Result	Pass/fail
1	UI Elements Test	Verify that UI elements like the profile picture ImageView, name EditText, and continue button are displayed correctly.	pass

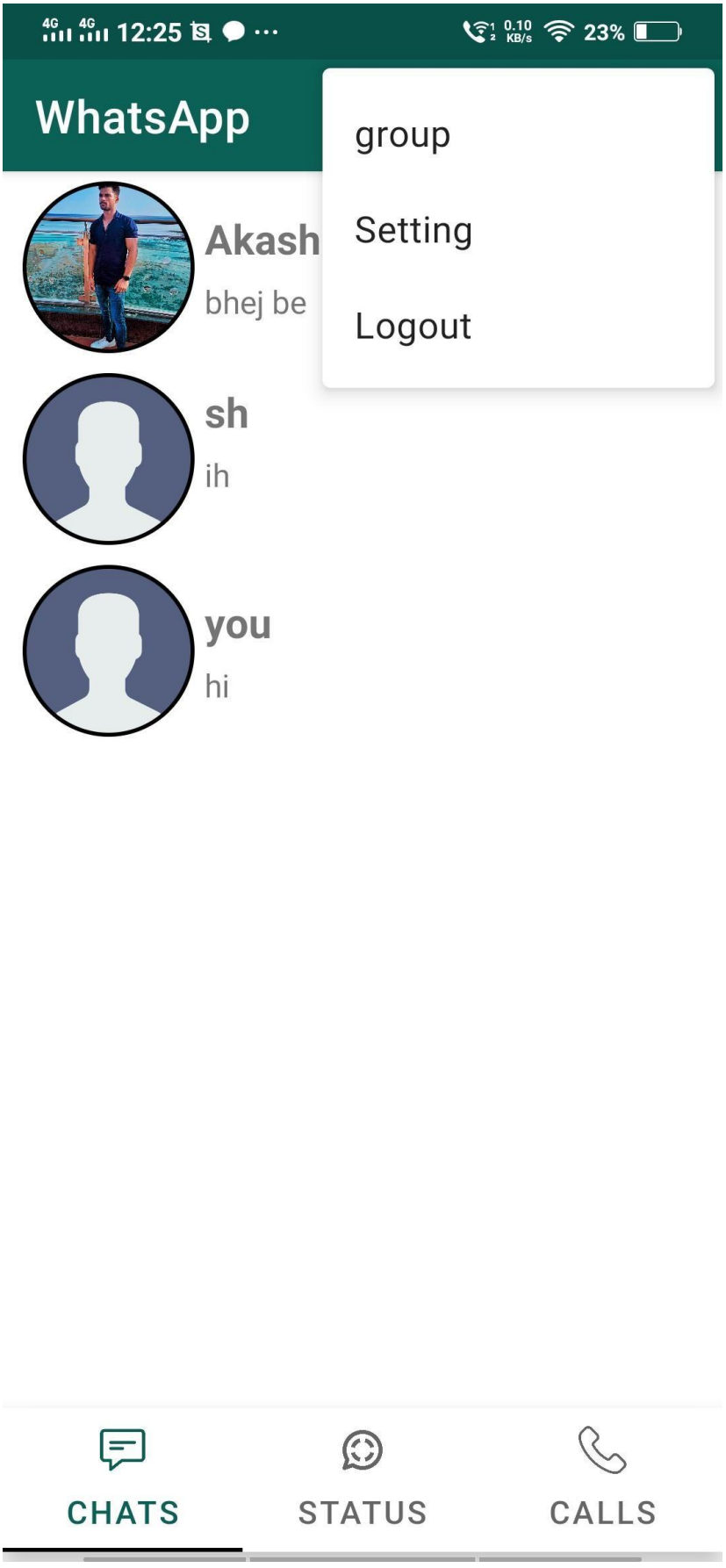
2	Profile Picture Selection Test	Test selecting a profile picture by clicking on the profile picture ImageView and choosing an image from the gallery.	pass
3	Profile Name Validation Test	Test entering a profile name and ensure that it's validated properly.	pass
4	Profile Creation Test	Test creating a profile by entering a name and optionally selecting a profile picture, then clicking the continue button.	pass
5	Profile Picture Upload Test	Test uploading a profile picture to Firebase Storage when a picture is selected.	pass
6	Profile Creation Without Picture Test	Test creating a profile without selecting a profile picture.	pass
7	Activity Navigation Test	Verify that after creating a profile, the user is redirected to the MainActivity.	pass

6.3 RESULTS


12:25
23%
WhatsApp
Akash
bhej be
sh
ih
you
hi
CHATS
STATUS
CALLS


12:26
1.00 KB/s
22%
Akash
ji jana bolo
12:00pm
hii
12:00 pm
hi
12:00pm
hii
12:00pm
hiii
12:00 pm
bhej be
12:00pm
Enter your Message


12:26
61.4 KB/s
22%
Group
hiii
12:00pm
hi
12:00pm
Enter your Message


12:25
0.10 KB/s
23%
WhatsApp
group
Setting
Logout
Akash
bhej be
sh
ih
you
hi
CHATS
STATUS
CALLS

CHAPTER 7: COCLUSION AND FUTURE ENHANCEMENTS

Conclusion:

The WhatsApp project demonstrates the implementation of essential features for a messaging application using Firebase as the backend. Key functionalities such as user authentication, profile setup, real-time messaging, and image uploading have been successfully implemented. The project provides a solid foundation for building a functional chat application.

Future Enhancements:

1.Enhanced User Experience (UX):

- Implement features like message status indicators (e.g., delivered, read) to enhance user engagement and interaction.
- Integrate typing indicators to show when the other user is typing a message.
- Implement push notifications to notify users of new messages even when the app is in the background.

2.Security Enhancements:

- Implement end-to-end encryption for messages to ensure the privacy and security of user communications.
- Add two-factor authentication (2FA) for enhanced account security.

3.Rich Media Support:

- Extend the messaging functionality to support sharing of media files such as images, videos, documents, and voice messages.
- Implement image and video compression to optimize storage usage and reduce bandwidth consumption.

4.Group Chat Functionality:

- Implement group chat functionality to allow users to create and participate in group conversations.
- Add features like group admin controls, group invites, and group Profile management.

5. User Profile Enhancements:

- Allow users to customize their profiles with additional information such as status messages, profile themes, and emojis.
- Implement profile editing functionality to enable users to update their profile information after the initial setup.

6. Search and Filter Functionality:

- Implement search functionality to allow users to search for specific messages or contacts within the app.
- Add filtering options to organize messages by date, sender, or keyword.

7. Localization and Internationalization:

- Localize the app to support multiple languages and regions to cater to a diverse user base.

- Implement internationalization best practices to ensure proper formatting of dates, times, and currencies.

8. Offline Support:

- Implement offline mode functionality to allow users to access and read their messages even when they are not connected to the internet.
- Implement local caching of messages to improve app performance and responsiveness.

9. Analytics and Insights:

- Integrate analytics tools to track user engagement, app usage patterns, and performance metrics.
- Use insights gathered from analytics to optimize user experience, prioritize feature development, and identify areas for improvement.

10.Continuous Testing and Bug Fixes:

- Establish a robust testing strategy with automated tests for critical functionalities to ensure app stability and reliability.
- Regularly monitor and address user-reported bugs and issues to provide a seamless user experience.

By implementing these enhancements, the WhatsApp project can evolve into a feature-rich messaging platform that offers a seamless user experience, robust security, and scalability to support a growing user base.