VISVESHVARAYA TECHNOLOGICAL UNIVERSITY – BELAGAVI

MOBILE APPLICATION DEVELOPMENT

MINI-PROJECT REPORT

"Chit Chat"

Submitted in partial fulfilment for the requirement of MAD mini-Project Sixth Semester, BE, CS&E.

Has been carried out by:

SEEMA R **3PG20CS039**

Under the Guidance of

PROF. INDIRA

PROF. MALTESH KAMATAR

Department Of Computer Science and Engineering

Bellary V.V. Sangha's

PROUDHADEVARAYA INSTITUTE OF TECHNOLOGY

T.B. Dam, Hosapete-583225

Vijayanagar Dist. Karnataka

Bellary V.V. Sangha's

PROUDHADEVARAYA INSTITUTE OF TECHNOLOGY

T.B. Dam, Hosapete-583225

Vijayanagar Dist. Karnataka

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify Ms. ***SEEMA R (3PG20CS039)*** studying in sixth semester **B.E, CS&E Dept**, has successfully submitted the project entitled "**CHIT CHAT"** towards the fulfilment for the requirements of **Mobile Application Development Lab(18CSMP68),** VTU, Belagavi during the academic year 2022- 2023.

ACKNOWLEDGEMENT

As we had started this project with the sense of learning something on the topic " *CHIT CHAT* ".

The euphoria and satisfaction that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible constant guidance and encouragement crowned our efforts with success.

First of all, we thank ***Visveswaraya Technological University***, ***Belagavi*** for providing this opportunity to implement this project.

We consider ourselves proud to be a part of "***P.D.I. T***" family, the institution which stood by our way in all endeavours.

We also take this opportunity to express our gratitude to ***Dr.ROHIT***, ***PRINCIPAL*** for providing us environment to do the project.

We sincerely thank ***Ms. Parvati Kadli, HOD Dept of CSE, PDIT***, and Teaching and Non-Teaching staff, without whose moral support this project would not have been successful.

We would like to express our heartfelt gratitude towards our inspiring Guide's ***Prof. Indira and Prof. Malatesh Kamatar*** whose firm belief in our capabilities and moral support to bring our potential to the forefront has played a major role in accomplishment of this project.

Last but not the least; we would like to thank our God, Parents, Relatives and Friends for their support and suggestions without which we couldn't have achieved this success.

PROJECT ASSOCIATES:

SEEMA R **3PG20CS039**

ABSTRACT

The Chit Chat App is an Android application developed to provide users with a platform for casual and interactive conversations. This report presents an overview of the Android development process for the Chit Chat App, highlighting the key components, features, and technical aspects involved.

The report aimed to create a user-friendly, engaging, and reliable chat application. It outlines the development team's methodologies, including agile practices and iterative development cycles, to ensure efficient progress and effective collaboration.

The Chit Chat App Android Development Report provides a comprehensive overview of the development process, highlighting the app's objectives, architecture, UI/UX design, testing, and challenges. The report serves as a valuable resource for understanding the technical aspects involved in developing a feature-rich chat application for the Android platform.

CONTENTS

CHAPTER 1

INTRODUCTION

Mobile application development is the process of creating software applications that run on a mobile device, and a typical mobile application utilizes a network connection to work with remote computing resources.

A mobile app developer uses programming languages and development skills to create, test, and develop applications on mobile devices. They work in popular operating system environments like iOS and Android and often take into account UI and UX principles when creating applications. Extensible Markup Language (XML) is used in android programming to create the screens of the application. Whatever the use sees on the screen is all the work of XML. Here is listed an example of XML code creating a screen with the picture.

The Chit Chat App is an Android application designed to provide a platform for users to engage in real-time conversations with friends, family, and other app users. This project report provides an overview of the development process, including the objectives, features, technologies used, challenges faced, and future recommendations for the Chit Chat App. In this project XML is used for the android screen design of the application and the java for the defining the background activity of the application.

CHAPTER 2

LITERATURE SURVEY

Literature Review Mobile Assisted Language Learning (MALL) Other than the Computer Assisted Language Learning (CALL), Mobile Assisted Language Learning (MALL) is rapidly gaining users.

Mobile devices such as hand phones, personal digital assistant (PDA) and tablet PC have been perceived as the ideal devices for individualized informal learning. Kukulska-Hulme (Researcher) (2009) made a comparison between MALL and CALL by pointing out that MALL enables a new approach of learning process. She further adds that MALL enables a new way of continuous learning as learning is not restricted within the four walls of a classroom but beyond. Viberg and Gronlund (Professors) (2012) identify three key terms related to MALL. First is about the technological concepts of learning where the concept of MALL is detached from the well-known learning theories, such as constructivism and collaborative learning.

Second is about techno-centered concept which focuses on technology as the communication means between learner and content, and between teacher and students. Third, it looks into contrasting the contexts of formal and informal learning and how the integration of mobile technologies could aid language learning in various situations.

CHAPTER 3

PROBLEM STATEMENT

- Develop an intuitive and user-friendly Android application for real-time chatting.
- Enable users to create profiles, add friends, and initiate private conversations.
- Implement push notifications to alert users about new messages.
- Provide a secure and reliable platform for exchanging text messages.
- Optimize performance to ensure smooth user experience.

CHAPTER-4

REQUIREMENTS AND SPECIFICATION

HARDWARE REQUIREMENTS

- 1 gigahertz (GHz) or faster 32-bit (x86) or 64-bit (x64) processor.
- 1 gigabyte (GB) RAM (32-bit) or 2 GB RAM (64-bit)
- 16 GB available hard disk space (32-bit) or 20 GB (64-bit)
- DirectX 9 graphics device with WDDM 1.0 or higher driver.

SOFTWARE REQUIREMENTS

- Operating system: WINDOWS 7,8,8.1,10,11.
- Programming language: Java.
- Android Studio.
- Visual Studio Code.

CHAPTER 5

ABOUT MOBILE APPLICATION DEVELOPMENT

Mobile application development is the process of creating software applications that run on a mobile device, and a typical mobile application utilizes a network connection to work with remote computing resources. Hence, the mobile development process involves creating installable software bundles (code, binaries, assets, etc.) , implementing backend services such as data access with an API, and testing the application on target devices.

Mobile Applications and Device Platforms

There are two dominant platforms in the modern smartphone market. One is the iOS platform from Apple Inc. The iOS platform is the operating system that powers Apple's popular line of iPhone smartphones. The second is Android from Google. The Android operating system is used not only by Google devices but also by many other OEMs to build their own smartphones and other smart devices.

Although there are some similarities between these two platforms when building applications, developing for iOS vs. developing for Android involves using different software development kits (SDKs) and different development toolchain. While Apple uses iOS exclusively for its own devices, Google makes Android available to other companies provided they meet specific requirements such as including certain Google applications on the devices they ship. Developers can build apps for hundreds of millions of devices by targeting both of these platforms.

Alternatives for Building Mobile Apps

There are four major development approaches when building mobile applications

- Native Mobile Applications
- Cross-Platform Native Mobile Applications
- Hybrid Mobile Applications
- Progressive Web Applications

Each of these approaches for developing mobile applications has its own set of advantages and disadvantages. When choosing the right development approach for their projects, developers consider the desired user experience, the computing resources and native features required by the app, the development budget, time targets, and resources available to maintain the app.

Native Applications

Native mobile applications are written in the programming language and frameworks provided by the platform owner and running directly on the operating system of the device such as iOS and Android.

Cross-Platform Applications

Cross-platform native mobile applications can be written in variety of different programming languages and frameworks, but they are compiled into a native application running directly on the operating system of the device.

Hybrid-Web Applications

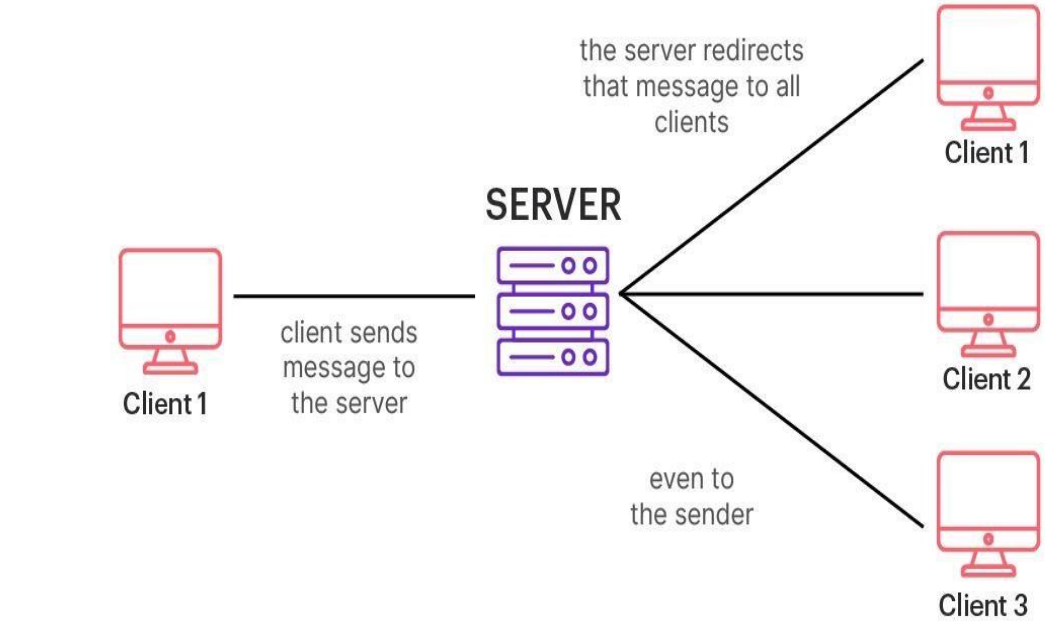
Hybrid mobile applications are built with standard web technologies - such as JavaScript, CSS, and HTML5 - and they are bundled as app installation packages. Contrary to the native apps, hybrid apps work on a 'web container' which provides a browser runtime and a bridge for native device APIs via Apache Cordova.

Progressive Web Applications

PWAs offer an alternative approach to traditional mobile app development by skipping app store delivery and app installations. PWAs are web applications that utilize a set of browser capabilities - such as working offline, running a background process, and adding a link to the device home screen - to provide an 'app like' user experience.

CHAPTER 6

DESIGN AND IMPLEMENTATION



FLOW CHART

CHAPTER 7

SOURCE CODE

Main Activity:

```
package com.example.chatapp.activities;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent; import
android.graphics.Bitmap; import
android.graphics.BitmapFactory; import
android.os.Bundle; import
android.util.Base64; import
android.view.View;
import android.widget.Toast;


import com.example.chatapp.adapter.RecentConversationAdapter;
import com.example.chatapp.databinding.ActivityMainBinding;
import com.example.chatapp.listeners.ConversionListener; import
com.example.chatapp.models.ChatMessage; import
com.example.chatapp.models.User; import
com.example.chatapp.utilities.Constants; import
com.example.chatapp.utilities.PreferenceManager; import
com.google.firebase.firestore.DocumentChange; import
com.google.firebase.firestore.DocumentReference; import
com.google.firebase.firestore.EventListener; import
com.google.firebase.firestore.FieldValue; import
com.google.firebase.firestore.FirebaseFirestore; import
com.google.firebase.firestore.QuerySnapshot; import
com.google.firebase.messaging.FirebaseMessaging;

import java.util.ArrayList; import
java.util.Collections; import
java.util.HashMap; import
java.util.List;
import java.util.Queue;

public class MainActivity extends AppCompatActivity implements ConversionListener {

    public ActivityMainBinding binding;     private
PreferenceManager preferenceManager;     private
List<ChatMessage> conversations;     private
RecentConversationAdapter conversationAdapter;
    private FirebaseFirestore database;

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
setContentView(binding.getRoot());
        preferenceManager = new PreferenceManager(getApplicationContext());
init();         loadUserDetails();         getToken();         setListeners();
listenConversations();
    }

    private void init() {
        conversations = new ArrayList<>();         conversationAdapter = new
RecentConversationAdapter(conversations, this);
binding.conversationsRecyclerView.setAdapter(conversationAdapter);
database = FirebaseFirestore.getInstance();
    }

    private void setListeners() {
        binding.imageSignout.setOnClickListener(v -> signOut());
binding.fabNewChat.setOnClickListener(v ->
                startActivity(new Intent(getApplicationContext(), UserActivity.class)));
    }

    private void loadUserDetails() {
binding.textName.setText(preferenceManager.getString(Constants.KEY_NAME));         byte[]
bytes = Base64.decode(preferenceManager.getString(Constants.KEY_IMAGE),
Base64.DEFAULT);
        Bitmap bitmap = BitmapFactory.decodeByteArray(bytes, 0, bytes.length);
binding.imageProfile.setImageBitmap(bitmap);
    }

    private void showToast(String message) {
        Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT).show();
    }

    private void listenConversations() {
        database.collection(Constants.KEY_COLLECTION_CONVERSATIONS)
                .whereEqualTo(Constants.KEY_SENDER_ID,
preferenceManager.getString(Constants.KEY_USER_ID))
                .addSnapshotListener(eventListener);
        database.collection(Constants.KEY_COLLECTION_CONVERSATIONS)
                .whereEqualTo(Constants.KEY_RECEIVER_ID,
preferenceManager.getString(Constants.KEY_USER_ID))
                .addSnapshotListener(eventListener);
    }

    private final EventListener<QuerySnapshot> eventListener = (value, error) -> {
if(error != null) {             return;
        }
        if(value != null) {
            for(DocumentChange documentChange : value.getDocumentChanges()) {
```

```
            if(documentChange.getType() == DocumentChange.Type.ADDED) {
                String senderId =
documentChange.getDocument().getString(Constants.KEY_SENDER_ID);
                String receiverId =
documentChange.getDocument().getString(Constants.KEY_RECEIVER_ID);
                ChatMessage chatMessage = new ChatMessage();
                chatMessage.senderId = senderId;
chatMessage.receiverId = receiverId;
                if(preferenceManager.getString(Constants.KEY_USER_ID).equals(senderId)) {
chatMessage.conversionImage =
documentChange.getDocument().getString(Constants.KEY_RECEIVER_IMAGE);
chatMessage.conversionName =
documentChange.getDocument().getString(Constants.KEY_RECEIVER_NAME);
chatMessage.conversionId =
documentChange.getDocument().getString(Constants.KEY_RECEIVER_ID);
                }
else {
                    chatMessage.conversionImage =
documentChange.getDocument().getString(Constants.KEY_SENDER_IMAGE);
chatMessage.conversionName =
documentChange.getDocument().getString(Constants.KEY_SENDER_NAME);
chatMessage.conversionId =
documentChange.getDocument().getString(Constants.KEY_SENDER_ID);
                }
                chatMessage.message =
documentChange.getDocument().getString(Constants.KEY_LAST_MESSAGE);
chatMessage.dateObject =
documentChange.getDocument().getDate(Constants.KEY_TIMESTAMP);
                conversations.add(chatMessage);
            }
            else if(documentChange.getType() == DocumentChange.Type.MODIFIED) {
for(int i = 0; i < conversations.size(); i++) {                    String senderId =
documentChange.getDocument().getString(Constants.KEY_SENDER_ID);
                    String receiverId =
documentChange.getDocument().getString(Constants.KEY_RECEIVER_ID);
                    if(conversations.get(i).senderId.equals(senderId) &&
conversations.get(i).receiverId.equals(receiverId)) {
                        conversations.get(i).message =
documentChange.getDocument().getString(Constants.KEY_LAST_MESSAGE);
conversations.get(i).dateObject =
documentChange.getDocument().getDate(Constants.KEY_TIMESTAMP);
break;
                    }
                }
            }
        }
        Collections.sort(conversations, (obj1, obj2) -> obj2.dateObject.compareTo(obj1.dateObject));
conversationAdapter.notifyDataSetChanged();
        binding.conversationsRecyclerView.smoothScrollToPosition(0);
binding.conversationsRecyclerView.setVisibility(View.VISIBLE);
binding.progressBar.setVisibility(View.GONE);
```

```
        }
    };

    private void getToken() {
        FirebaseMessaging.getInstance().getToken().addOnSuccessListener(this::updateToken);
    }

    private void updateToken(String token) {
        FirebaseFirestore database = FirebaseFirestore.getInstance();
DocumentReference documentReference =
                database.collection(Constants.KEY_COLLECTION_USERS).document(
preferenceManager.getString(Constants.KEY_USER_ID)
                );
        documentReference.update(Constants.KEY_FCM_TOKEN, token)
                .addOnFailureListener(e -> showToast("Unable to update token"));
    }

    private void signOut() {
showToast("Signing Out...");
        FirebaseFirestore database = FirebaseFirestore.getInstance();
DocumentReference documentReference =
                database.collection(Constants.KEY_COLLECTION_USERS).document(
preferenceManager.getString(Constants.KEY_USER_ID)
                );
        HashMap<String, Object> updates = new HashMap<>();
updates.put(Constants.KEY_FCM_TOKEN, FieldValue.delete());
documentReference.update(updates)
.addOnSuccessListener(unused -> {
preferenceManager.clear();
                    startActivity(new Intent(getApplicationContext(), SignInActivity.class));
finish();
                })
                .addOnFailureListener(e -> showToast("Unable to sign out"));
    }

    @Override
    public void onConversionClicked(User user) {
        Intent intent = new Intent(getApplicationContext(), ChatActivity.class);
intent.putExtra(Constants.KEY_USER, user);
        startActivity(intent);
    }
```

BUILD CONFIG

```
package com.example.chatapp;

public final class BuildConfig {
  public static final boolean DEBUG = Boolean.parseBoolean("true");
public static final String APPLICATION_ID = "com.example.chatapp";
public static final String BUILD_TYPE = "debug";   public static final
```

```
int VERSION_CODE = 1;   public static final String VERSION_NAME
= "1.0";
}
```

SIGN UP ACTIVITY

```
public class SignUpActivity extends AppCompatActivity {

   private ActivitySignUpBinding binding;
private PreferenceManager preferenceManager;
private String encodedImage;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
      binding = ActivitySignUpBinding.inflate(getLayoutInflater());
setContentView(binding.getRoot());
      preferenceManager = new PreferenceManager(getApplicationContext());
setListeners();
   }

   private void setListeners() {
      binding.textSignIn.setOnClickListener(v -> onBackPressed());
binding.buttonSignUp.setOnClickListener(v -> {
         if(isValidSignUpDetails()) {
            signUp();
         }
      });
      binding.layoutImage.setOnClickListener(v      ->      {
Intent intent = new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
         intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
pickImage.launch(intent);
      });
   }

   private void showToast(String message) {
      Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT).show();
   }

   private void signUp() {
loading(true);
      FirebaseFirestore database = FirebaseFirestore.getInstance();
HashMap<String, Object> user = new HashMap<>();
user.put(Constants.KEY_NAME, binding.inputName.getText().toString());
user.put(Constants.KEY_EMAIL, binding.inputEmail.getText().toString());
user.put(Constants.KEY_PASSWORD, binding.inputPassword.getText().toString());
user.put(Constants.KEY_IMAGE,encodedImage);
database.collection(Constants.KEY_COLLECTION_USERS)
            .add(user)
            .addOnSuccessListener(documentReference -> {
loading(false);
               preferenceManager.putBoolean(Constants.KEY_IS_SIGNED_IN, true);
```

```
                preferenceManager.putString(Constants.KEY_USER_ID, documentReference.getId());
preferenceManager.putString(Constants.KEY_NAME, binding.inputName.getText().toString());
                preferenceManager.putString(Constants.KEY_IMAGE, encodedImage);
Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
startActivity(intent);
            })
            .addOnFailureListener(exception -> {
loading(false);
                showToast(exception.getMessage());
            });
    }

    private String encodedImage(Bitmap bitmap) {
        int previewWidth = 150;
        int previewHeight = bitmap.getHeight() + previewWidth / bitmap.getWidth();
        Bitmap previewBitmap = Bitmap.createScaledBitmap(bitmap, previewWidth, previewHeight, false);
        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
previewBitmap.compress(Bitmap.CompressFormat.JPEG, 100,
byteArrayOutputStream);        byte[] bytes = byteArrayOutputStream.toByteArray();
return Base64.encodeToString(bytes, Base64.DEFAULT);
    }

    private final ActivityResultLauncher<Intent> pickImage = registerForActivityResult(
new ActivityResultContracts.StartActivityForResult(),            result -> {
            if(result.getResultCode() == RESULT_OK) {
if (result.getData() != null) {
                    Uri imageUri = result.getData().getData();
try {
                        InputStream inputStream = getContentResolver().openInputStream(imageUri);
                        Bitmap bitmap = BitmapFactory.decodeStream(inputStream);
binding.imageProfile.setImageBitmap(bitmap);
binding.textAddImage.setVisibility(View.GONE);
encodedImage = encodedImage(bitmap);                        } catch
(FileNotFoundException e) {                        e.printStackTrace();
                    }
                }
            }
        }
    );

    private Boolean isValidSignUpDetails() {
if(encodedImage == null) {
showToast("Select profile image");
            return false;
        }
        else if(binding.inputName.getText().toString().trim().isEmpty()) {
showToast("Enter Name");            return false;
        }
        else if(binding.inputEmail.getText().toString().trim().isEmpty()) {
showToast("Enter email");
            return false;
        }
        else if(!Patterns.EMAIL_ADDRESS.matcher(binding.inputEmail.getText().toString()).matches()) {
showToast("Enter a valid email");
            return false;
```

```
        }
        else if(binding.inputPassword.getText().toString().trim().isEmpty()) {
showToast("Enter password");            return false;
        }
        else if(binding.inputPassword.getText().toString().trim().length()<8) {
showToast("Minimum length must be 8 characters");            return false;
        }
        else if(binding.inputConfirmPassword.getText().toString().trim().isEmpty()) {
showToast("Confirm your password");
            return false;
        }
else
if(!binding.inputPassword.getText().toString().equals(binding.inputConfirmPassword.getText().toString()))  {
showToast("Password should be same");            return false;
        }
else {
return
true;
        }
    }

    private void loading(Boolean isLoading) {
        if(isLoading) {
            binding.buttonSignUp.setVisibility(View.INVISIBLE);
binding.progressBar.setVisibility(View.VISIBLE);
        }
else {
            binding.progressBar.setVisibility(View.INVISIBLE);
            binding.buttonSignUp.setVisibility(View.VISIBLE);
        }
    }
}
```

SIGN IN ACTIVITY:

```
ackage com.example.chatapp.activities;

import android.content.Intent;
import android.os.Bundle; import
android.util.Patterns; import
android.view.View;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.example.chatapp.databinding.ActivitySignInBinding;
import com.example.chatapp.utilities.Constants; import
com.example.chatapp.utilities.PreferenceManager; import
com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
```

```
public class SignInActivity extends AppCompatActivity {

    private ActivitySignInBinding binding;
private PreferenceManager preferenceManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        preferenceManager = new PreferenceManager(getApplicationContext());
if(preferenceManager.getBoolean(Constants.KEY_IS_SIGNED_IN)) {
Intent intent = new Intent(getApplicationContext(), MainActivity.class);
startActivity(intent);
            finish();
        }
        binding = ActivitySignInBinding.inflate(getLayoutInflater());
setContentView(binding.getRoot());
        setListeners();
    }

    private void setListeners() {
        binding.textCreateNewAccount.setOnClickListener(v ->
                startActivity(new Intent(getApplicationContext(), SignUpActivity.class)));
binding.buttonSignIn.setOnClickListener(v -> {
            if(isValidStringInDetails()) {
                signIn();
            }
        });
    }

    private void signIn() {
loading(true);
        FirebaseFirestore database = FirebaseFirestore.getInstance();
database.collection(Constants.KEY_COLLECTION_USERS)
                .whereEqualTo(Constants.KEY_EMAIL, binding.inputEmail.getText().toString())
                .whereEqualTo(Constants.KEY_PASSWORD, binding.inputPassword.getText().toString())
                .get()
                .addOnCompleteListener(task -> {
                  if(task.isSuccessful() && task.getResult() != null
&& task.getResult().getDocuments().size() > 0) {
                        DocumentSnapshot documentSnapshot = task.getResult().getDocuments().get(0);
preferenceManager.putBoolean(Constants.KEY_IS_SIGNED_IN, true);
                        preferenceManager.putString(Constants.KEY_USER_ID, documentSnapshot.getId());
preferenceManager.putString(Constants.KEY_NAME,
documentSnapshot.getString(Constants.KEY_NAME));
                        preferenceManager.putString(Constants.KEY_IMAGE,
documentSnapshot.getString(Constants.KEY_IMAGE));
                        Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                        startActivity(intent);
```

```
                }
else {
                    loading(false);
                    showToast("Unable to sign in");
                }
            });
    }

    private void loading(Boolean isLoading) {
        if(isLoading) {
            binding.buttonSignIn.setVisibility(View.INVISIBLE);
binding.progressBar.setVisibility(View.VISIBLE);
        }
else {
            binding.progressBar.setVisibility(View.INVISIBLE);
binding.buttonSignIn.setVisibility(View.VISIBLE);
        }
    }

    private void showToast(String message) {
        Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT).show();
    }

    private Boolean isValidStringInDetails() {
if(binding.inputEmail.getText().toString().trim().isEmpty()) {            showToast("Enter
email");
            return false;
        }
        else if(!Patterns.EMAIL_ADDRESS.matcher(binding.inputEmail.getText().toString()).matches())
{
            showToast("Enter a valid email");
            return false;
        }
        else if(binding.inputPassword.getText().toString().trim().isEmpty()) {
showToast("Enter password");            return false;
        }
else {
            return true;
        }
    }

}
```

USER ACTIVITY

```
public class UserActivity extends AppCompatActivity implements UserListener {

    private ActivityUserBinding binding;     private
PreferenceManager preferenceManager;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        binding = ActivityUserBinding.inflate(getLayoutInflater());
setContentView(binding.getRoot());
        preferenceManager = new PreferenceManager(getApplicationContext());
setListeners();        getUsers();
    }

    private void setListeners() {
        binding.imageBack.setOnClickListener(v -> onBackPressed());
    }

    private void getUsers() {
        loading(true);
        FirebaseFirestore database = FirebaseFirestore.getInstance();
        database.collection(Constants.KEY_COLLECTION_USERS)
                .get()
                .addOnCompleteListener(task -> {
loading(false);
                    String currentUserId = preferenceManager.getString(Constants.KEY_USER_ID);
if(task.isSuccessful() && task.getResult() != null) {                    List<User> users = new
ArrayList<>();
                        for (QueryDocumentSnapshot queryDocumentSnapshot : task.getResult()) {
if(currentUserId.equals(queryDocumentSnapshot.getId())) {                            continue;
                        }
                        User user = new User();
                        user.name = queryDocumentSnapshot.getString(Constants.KEY_NAME);
user.email = queryDocumentSnapshot.getString(Constants.KEY_EMAIL);
user.image = queryDocumentSnapshot.getString(Constants.KEY_IMAGE);
user.token = queryDocumentSnapshot.getString(Constants.KEY_FCM_TOKEN);
user.id = queryDocumentSnapshot.getId();                        users.add(user);
                    }
                    if(users.size() > 0) {
                        UsersAdapter usersAdapter = new UsersAdapter(users, this);
binding.usersRecyclerView.setAdapter(usersAdapter);
binding.usersRecyclerView.setVisibility(View.VISIBLE);
                    }
                    else {
                        showErrorMessage();
                    }
}
                else {
                    showErrorMessage();
                }
            });
    }

    private void showErrorMessage() {
        binding.textErrorMessage.setText(String.format("%s", "No user available"));
binding.textErrorMessage.setVisibility(View.VISIBLE);
```

```
    }

    private void loading(Boolean isLoading) {
        if(isLoading) {
            binding.progressBar.setVisibility(View.VISIBLE);
        }
else {
            binding.progressBar.setVisibility(View.INVISIBLE);
        }
    }

    @Override
    public void onUserClicked(User user) {
        Intent intent = new Intent(getApplicationContext(), ChatActivity.class);
intent.putExtra(Constants.KEY_USER, user);
        startActivity(intent);
        finish();
    }
}
```

CHAT ACTIVITY

```
public class ChatActivity extends AppCompatActivity {

    private ActivityChatBinding binding;
private User receiverUser;
    private List<ChatMessage> chatMessages;
private ChatAdapter chatAdapter;     private
PreferenceManager preferenceManager; private
FirebaseFirestore database;
    private String conversionId = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        binding = ActivityChatBinding.inflate(getLayoutInflater());
setContentView(binding.getRoot());
        setListeners();
loadReceiverDetails();
        init();
        listenMessages();
    }

    private void init() {
        preferenceManager = new PreferenceManager(getApplicationContext());
chatMessages = new ArrayList<>();         chatAdapter = new ChatAdapter(
                chatMessages,
                getBitmapFromEncodedString(receiverUser.image),
preferenceManager.getString(Constants.KEY_USER_ID)
        );
```

```
        binding.chatRecyclerView.setAdapter(chatAdapter);
database = FirebaseFirestore.getInstance();
    }

    private void sendMessage() {
        HashMap<String, Object> message = new HashMap<>();
message.put(Constants.KEY_SENDER_ID,
preferenceManager.getString(Constants.KEY_USER_ID));
message.put(Constants.KEY_RECEIVER_ID, receiverUser.id);
        message.put(Constants.KEY_MESSAGE, binding.inputMessage.getText().toString());
message.put(Constants.KEY_TIMESTAMP, new Date());
        database.collection(Constants.KEY_COLLECTION_CHAT).add(message);
if(conversionId != null) {
            updateConversion(binding.inputMessage.getText().toString());
        }
else {
            HashMap<String, Object> conversion = new HashMap<>();
conversion.put(Constants.KEY_SENDER_ID,
preferenceManager.getString(Constants.KEY_USER_ID));
conversion.put(Constants.KEY_SENDER_NAME,
preferenceManager.getString(Constants.KEY_NAME));
conversion.put(Constants.KEY_SENDER_IMAGE,
preferenceManager.getString(Constants.KEY_IMAGE));
            conversion.put(Constants.KEY_RECEIVER_ID, receiverUser.id);
conversion.put(Constants.KEY_RECEIVER_NAME, receiverUser.name);
conversion.put(Constants.KEY_RECEIVER_IMAGE, receiverUser.image);
            conversion.put(Constants.KEY_LAST_MESSAGE, binding.inputMessage.getText().toString());
conversion.put(Constants.KEY_TIMESTAMP, new Date());            addConversion(conversion);
        }
        binding.inputMessage.setText(null);
    }

    private void listenMessages() {
        database.collection(Constants.KEY_COLLECTION_CHAT)
.whereEqualTo(Constants.KEY_SENDER_ID,
preferenceManager.getString(Constants.KEY_USER_ID))
                .whereEqualTo(Constants.KEY_RECEIVER_ID, receiverUser.id)
                .addSnapshotListener(eventListener);
        database.collection(Constants.KEY_COLLECTION_CHAT)
.whereEqualTo(Constants.KEY_SENDER_ID, receiverUser.id)
                .whereEqualTo(Constants.KEY_RECEIVER_ID,
preferenceManager.getString(Constants.KEY_USER_ID))
                .addSnapshotListener(eventListener);
    }

    private final EventListener<QuerySnapshot> eventListener = (value, error) -> {
if(error != null) {
            return;
        }
        if(value != null) {
            int count = chatMessages.size();
```

```
            for(DocumentChange documentChange : value.getDocumentChanges()) {
if(documentChange.getType() == DocumentChange.Type.ADDED) {
                    ChatMessage chatMessage = new ChatMessage();
                    chatMessage.senderId =
documentChange.getDocument().getString(Constants.KEY_SENDER_ID);
chatMessage.receiverId =
documentChange.getDocument().getString(Constants.KEY_RECEIVER_ID);
chatMessage.message =
documentChange.getDocument().getString(Constants.KEY_MESSAGE);
chatMessage.dateTime =
getReadableDateTime(documentChange.getDocument().getDate(Constants.KEY_TIMESTAMP));
chatMessage.dateObject =
documentChange.getDocument().getDate(Constants.KEY_TIMESTAMP);
                    chatMessages.add(chatMessage);
                }
            }
            Collections.sort(chatMessages, (obj1, obj2) -> obj1.dateObject.compareTo(obj2.dateObject));
if(count == 0) {
                chatAdapter.notifyDataSetChanged();
            }
else {
                chatAdapter.notifyItemRangeInserted(chatMessages.size(), chatMessages.size());
binding.chatRecyclerView.smoothScrollToPosition(chatMessages.size() - 1);
            }
            binding.chatRecyclerView.setVisibility(View.VISIBLE);
        }
        binding.progressBar.setVisibility(View.GONE);
        if(conversionId == null) {
            checkForConversion();
        }
    };


    private Bitmap getBitmapFromEncodedString(String encodedImage) {
byte[] bytes = Base64.decode(encodedImage, Base64.DEFAULT);        return
BitmapFactory.decodeByteArray(bytes, 0, bytes.length);
    }

    private void loadReceiverDetails() {
        receiverUser = (User) getIntent().getSerializableExtra(Constants.KEY_USER);
binding.textName.setText(receiverUser.name);
    }

    private void setListeners() {
binding.imageBack.setOnClickListener(v -> onBackPressed());
binding.layoutSend.setOnClickListener(v -> sendMessage());
    }

    private String getReadableDateTime(Date date) {
        return new SimpleDateFormat("MMMM dd, yyyy - hh:mm a", Locale.getDefault()).format(date);
```

```
    }

    private void addConversion(HashMap<String, Object> conversion) {
        database.collection(Constants.KEY_COLLECTION_CONVERSATIONS)
                .add(conversion)
                .addOnSuccessListener(documentReference -> conversionId = documentReference.getId());
    }

    private void updateConversion(String message) {
        DocumentReference documentReference =

database.collection(Constants.KEY_COLLECTION_CONVERSATIONS).document(conversionId);
documentReference.update(
                Constants.KEY_LAST_MESSAGE, message,
                Constants.KEY_TIMESTAMP, new Date()
        );
    }

    private void checkForConversion() {
if(chatMessages.size() != 0) {
checkForConversionRemotely(
                    preferenceManager.getString(Constants.KEY_USER_ID),
receiverUser.id
            );

            checkForConversionRemotely(
receiverUser.id,
                    preferenceManager.getString(Constants.KEY_USER_ID)
            );
        }
    }

    private void checkForConversionRemotely(String senderId, String receiverId) {
database.collection(Constants.KEY_COLLECTION_CONVERSATIONS)
                .whereEqualTo(Constants.KEY_SENDER_ID, senderId)
                .whereEqualTo(Constants.KEY_RECEIVER_ID, receiverId)
                .get()
                .addOnCompleteListener(conversionOnCompleteListener);
    }

    private final OnCompleteListener<QuerySnapshot> conversionOnCompleteListener = task -> {
        if (task.isSuccessful() && task.getResult() != null && task.getResult().getDocuments().size() > 0) {
            DocumentSnapshot documentSnapshot = task.getResult().getDocuments().get(0);
conversionId = documentSnapshot.getId();
        }
    };
```

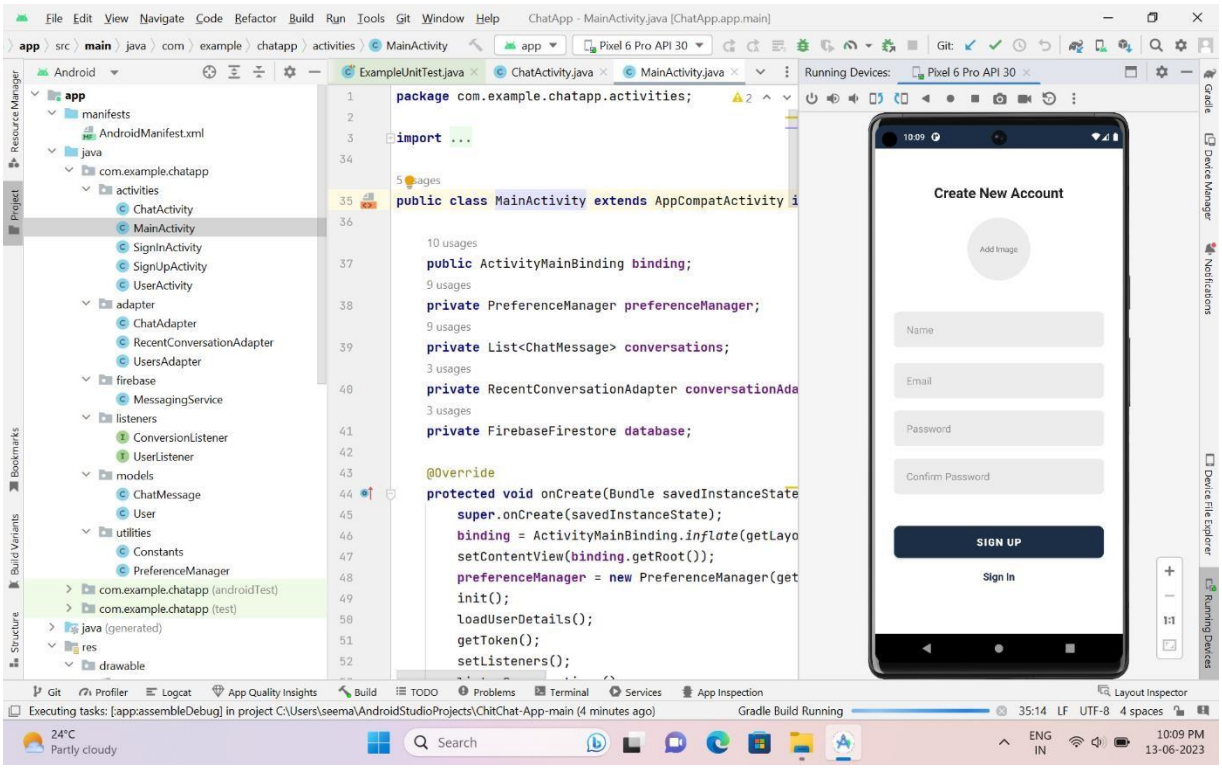
Workspace.xml:

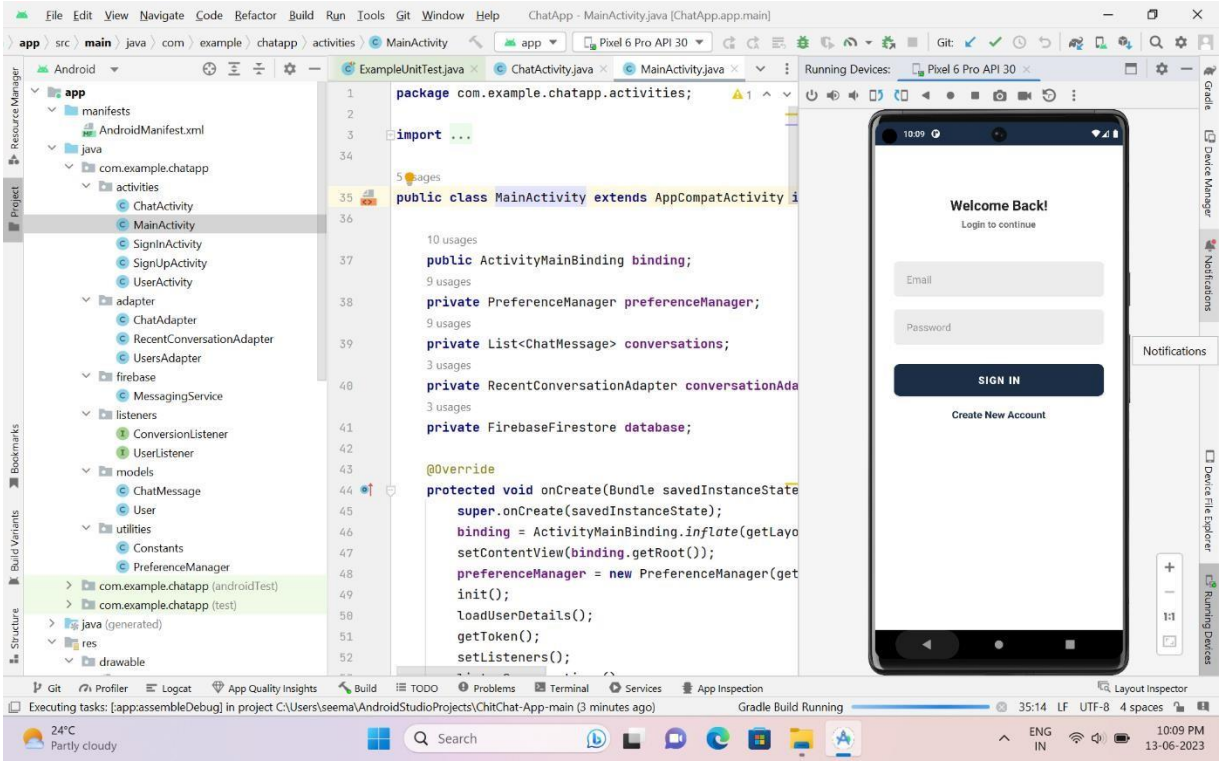
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"     android:layout_height="match_parent"
android:animateLayoutChanges="true"
android:background="@color/primary"
tools:context=".activities.ChatActivity">
  </component>

<View
   android:id="@+id/headerBackground"
android:layout_width="match_parent"     android:layout_height="0dp"
android:background="@color/primary"
   app:layout_constraintBottom_toTopOf="@id/viewSupporter"
app:layout_constraintTop_toTopOf="parent"/>

<TextView
   android:id="@+id/textName"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_marginStart="@dimen/_8sdp"
android:layout_marginEnd="@dimen/_8sdp"
   android:ellipsize="end"
android:gravity="center"     android:maxLines="1"
android:textColor="@color/white"
android:textSize="@dimen/_14ssp"
   android:textStyle="bold"
   app:layout_constraintBottom_toBottomOf="@id/imageBack"
app:layout_constraintEnd_toStartOf="@id/imageInfo"
app:layout_constraintStart_toEndOf="@id/imageBack"
app:layout_constraintTop_toTopOf="@id/imageBack"/>

<ProgressBar
   android:id="@+id/progressBar"
   android:layout_width="@dimen/_25sdp"
android:layout_height="@dimen/_25sdp"
app:layout_constraintBottom_toBottomOf="@id/viewBackground"
app:layout_constraintEnd_toEndOf="@id/viewBackground"
app:layout_constraintStart_toStartOf="@id/viewBackground"
   app:layout_constraintTop_toTopOf="@id/viewBackground"/>
```

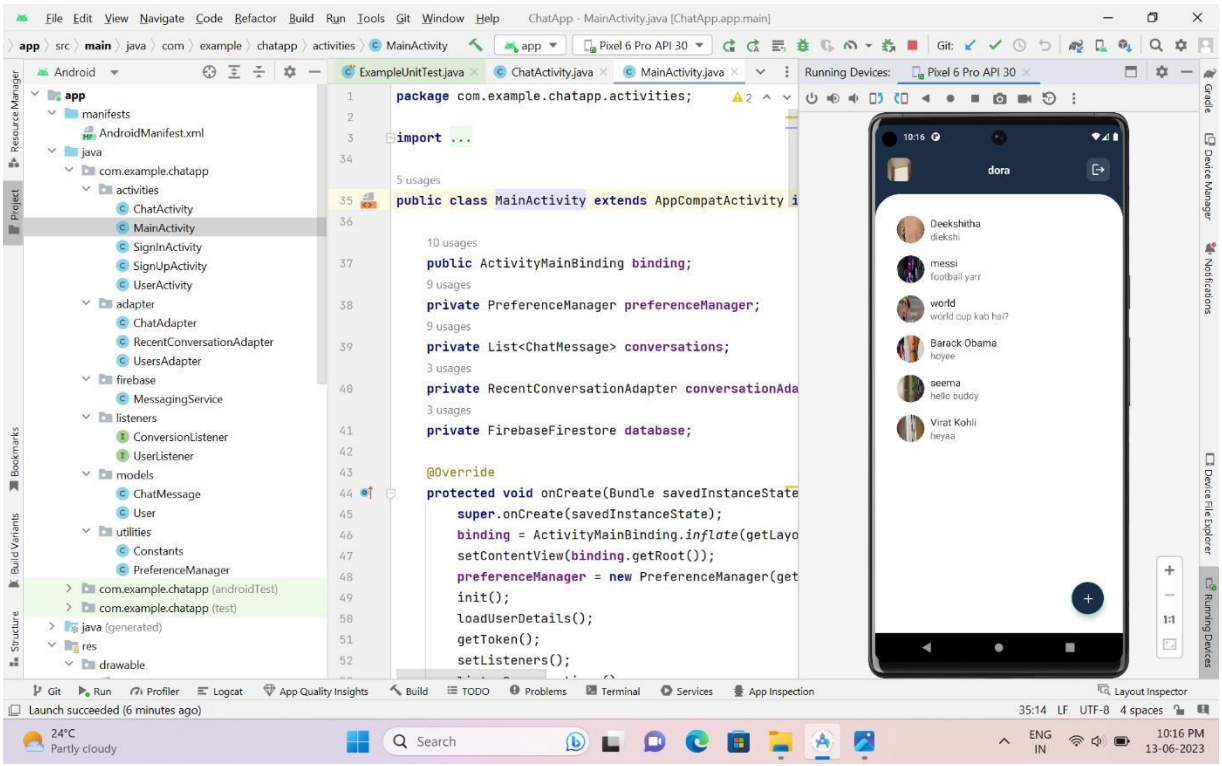
```
<FrameLayout     android:id="@+id/layoutSend"
android:layout_width="@dimen/_40sdp"
android:layout_height="@dimen/_40sdp"
android:layout_marginEnd="@dimen/_16sdp"
android:layout_marginBottom="@dimen/_12sdp"
android:background="@drawable/background_chat_input"
app:layout_constraintBottom_toBottomOf="parent"
   app:layout_constraintEnd_toEndOf="parent">

   <androidx.appcompat.widget.AppCompatImageView
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_marginStart="@dimen/_4sdp"
android:padding="@dimen/_8sdp"         android:src="@drawable/ic_send"
      android:tint="@color/white"/>

<EditText     android:id="@+id/inputPassword"
android:layout_width="match_parent"
android:layout_height="@dimen/_45sdp"
android:layout_marginTop="@dimen/_16sdp"
android:background="@drawable/background_input"
   android:hint="@string/password"
android:imeOptions="actionDone"
android:importantForAutofill="no"
android:inputType="textPassword"
android:paddingStart="@dimen/_16sdp"
android:paddingEnd="@dimen/_16sdp"
android:textColor="@color/primary_text"
   android:textSize="@dimen/_13ssp" />

<FrameLayout
   android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="@dimen/_20sdp"
   android:animateLayoutChanges="true">

<androidx.appcompat.widget.AppCompatImageView
   android:id="@+id/imageBack"
android:layout_width="@dimen/_30sdp"
android:layout_height="@dimen/_30sdp"
android:layout_marginStart="@dimen/_16sdp"
android:layout_marginTop="@dimen/_12sdp"
android:background="@drawable/background_icon"
android:padding="@dimen/_6sdp"     android:src="@drawable/ic_back"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
android:tint="@color/white"/>
```

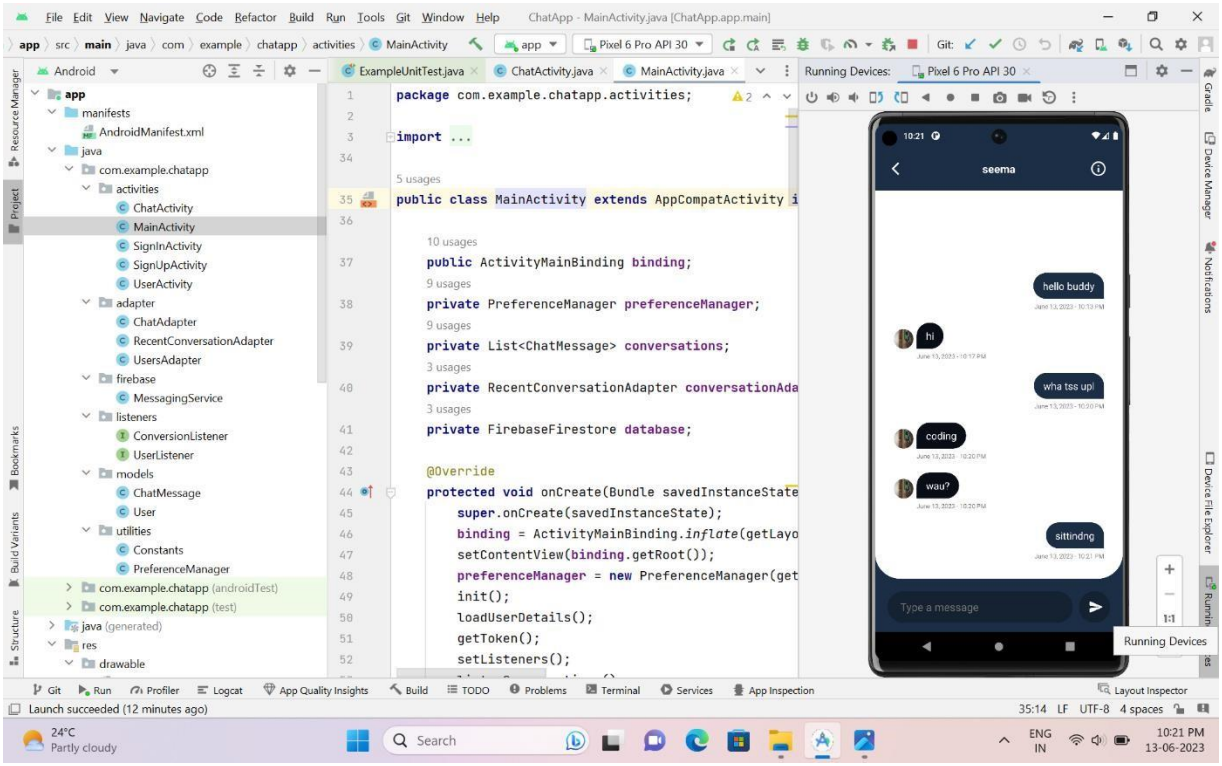
CHAPTER 8

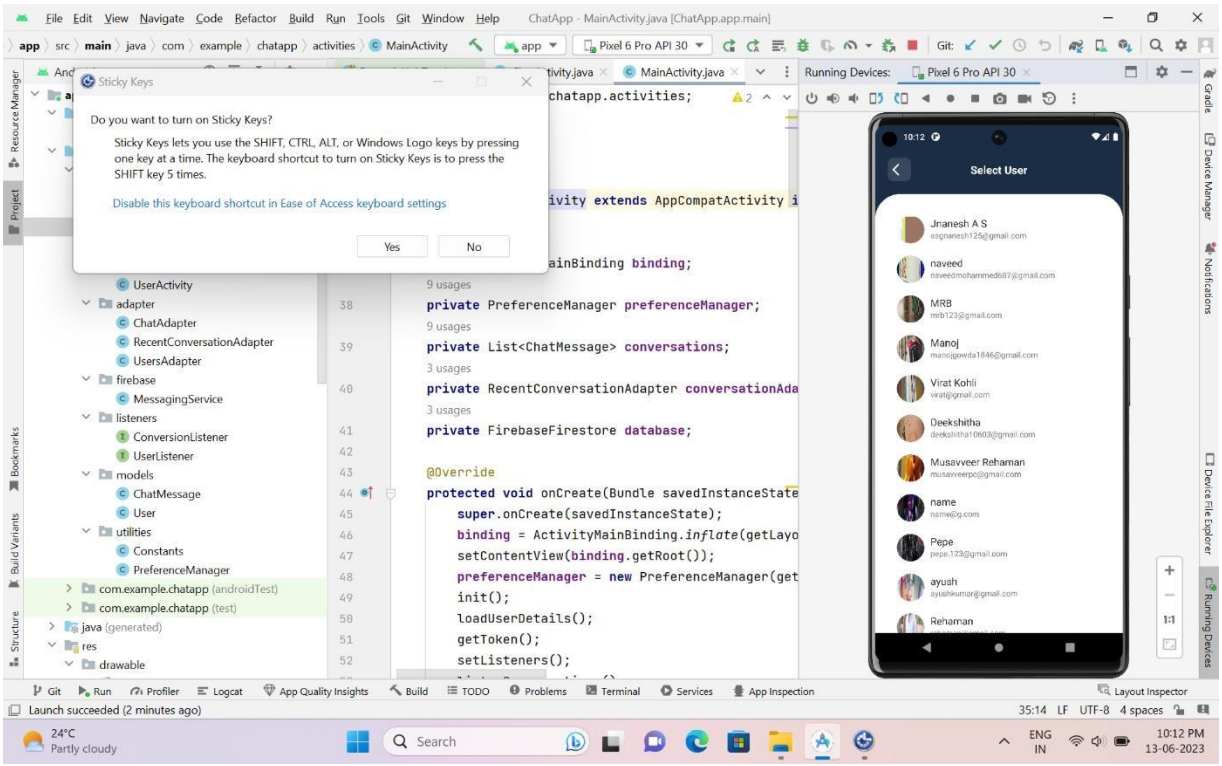
SNAPSHOTS






File Edit View Navigate Code Refactor Build Run Tools Git Window Help
ChatApp - MainActivity.java [ChatApp.app.main]
app src main java com example chatapp activities MainActivity
Pixel 6 Pro API 30
Android
ExampleUnitTest.java
ChatActivity.java
MainActivity.java
Running Devices:
Pixel 6 Pro API 30
app
manifests
AndroidManifest.xml
java
com.example.chatapp
activities
ChatActivity
MainActivity
SignInActivity
SignUpActivity
UserActivity
adapter
ChatAdapter
RecentConversationAdapter
UsersAdapter
firebase
MessagingService
listeners
ConversionListener
UserListener
models
ChatMessage
User
utilities
Constants
PreferenceManager
com.example.chatapp (androidTest)
com.example.chatapp (test)
java (generated)
res
drawable
package com.example.chatapp.activities;
import ...
5 usages
public class MainActivity extends AppCompatActivity i
10 usages
public ActivityMainBinding binding;
9 usages
private PreferenceManager preferenceManager;
9 usages
private List<ChatMessage> conversations;
3 usages
private RecentConversationAdapter conversationAda
3 usages
private FirebaseFirestore database;
@Override
protected void onCreate(Bundle savedInstanceState
super.onCreate(savedInstanceState);
binding = ActivityMainBinding.inflate(getLayo
setContentView(binding.getRoot());
preferenceManager = new PreferenceManager(get
init();
loadUserDetails();
getToken();
setListeners();
10:16
dora
Deekshitha
diekshi
messi
football yarr
world
world cup kab hai?
Barack Obama
hoyee
seema
hello buddy
Virat Kohli
heyaa
Git Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection Layout Inspector
Launch succeeded (6 minutes ago)
35:14 LF UTF-8 4 spaces
24°C
Partly cloudy
Search
ENG
IN
10:16 PM
13-06-2023


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
ChatApp - MainActivity.java [ChatApp.app.main]
app src main java com example chatapp activities MainActivity
Pixel 6 Pro API 30
Android
ExampleUnitTest.java
ChatActivity.java
MainActivity.java
Running Devices:
Pixel 6 Pro API 30
app
manifests
AndroidManifest.xml
java
com.example.chatapp
activities
ChatActivity
MainActivity
SignInActivity
SignUpActivity
UserActivity
adapter
ChatAdapter
RecentConversationAdapter
UsersAdapter
firebase
MessagingService
listeners
ConversionListener
UserListener
models
ChatMessage
User
utilities
Constants
PreferenceManager
com.example.chatapp (androidTest)
com.example.chatapp (test)
java (generated)
res
drawable
package com.example.chatapp.activities;
import ...
5 usages
public class MainActivity extends AppCompatActivity i
10 usages
public ActivityMainBinding binding;
9 usages
private PreferenceManager preferenceManager;
9 usages
private List<ChatMessage> conversations;
3 usages
private RecentConversationAdapter conversationAda
3 usages
private FirebaseFirestore database;
@Override
protected void onCreate(Bundle savedInstanceState
super.onCreate(savedInstanceState);
binding = ActivityMainBinding.inflate(getLayo
setContentView(binding.getRoot());
preferenceManager = new PreferenceManager(get
init();
loadUserDetails();
getToken();
setListeners();
10:21
seema
hello buddy
hi
wha tss up!
coding
wau?
sittindng
Type a message
Running Devices
Git Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection Layout Inspector
Launch succeeded (12 minutes ago)
35:14 LF UTF-8 4 spaces
24°C
Partly cloudy
Search
ENG
IN
10:21 PM
13-06-2023


Sticky Keys
Do you want to turn on Sticky Keys?
Sticky Keys lets you use the SHIFT, CTRL, ALT, or Windows Logo keys by pressing one key at a time. The keyboard shortcut to turn on Sticky Keys is to press the SHIFT key 5 times.
Disable this keyboard shortcut in Ease of Access keyboard settings
Yes
No
Select User
Jnanesh A S
naveed
MRB
Manoj
Virat Kohli
Deekshitha
Musavveer Rehaman
name
Pepe
ayush
Rehaman

CHAPTER 9

CONCLUSION

- The Chit Chat App has been successfully developed for the Android platform. The primary objective of the project was to create an interactive and engaging chat application that provides users with an enjoyable communication experience.

- The core functionality of the Chit Chat App, which includes real-time messaging and chat features, has been implemented effectively.

- The app's layout and design elements are visually appealing, contributing to a positive user experience.

- Additionally, they implemented a robust search functionality, enabling users to easily find and connect with their friends and contacts within the app.

- The Chit Chat App has undergone extensive testing to ensure its stability, reliability, and security. The team addressed any identified bugs or glitches and made necessary improvements to optimize performance.

CHAPTER 10

BIBILOGRAPHY

1. Google Developer Training, "Android Developer Fundamentals Course – Concept Reference", Google Developer Training Team, 2017.
2. Erik Hellman, "Android Programming – Pushing the Limits", 1st Edition, Wiley India Pvt Ltd, 2014.
3. Dawn Griffiths and David Griffiths, "Head First Android Development", 1st Edition, O'Reilly SPD Publishers, 2015.
4. J F DiMarzio, "Beginning Android Programming with Android Studio", 4th Edition, Wiley India Pvt Ltd, 2016. ISBN-13: 978-8126565580
5. Anubhav Pradhan, Anil V Deshpande, "Composing Mobile Apps" using Android, Wiley 2014, ISBN: 978-81-265-4660-2
6. www.developer.android.com
7. https://en.wikipedia.org/wiki/Mobile-assisted_language_learning
8. https://www.oracle.com/java/technologies/downloads/
9. www.android.com
10. www.androidhive.info
11. www.androidcompare.com
12. www.tutorialpoints.com

CHAPTER - 11

GROUP MEMBERS

NAME : SEEMA R

USN : 3PG20CS039

SEM : 6th sem

BRANCH : CSE

E-Mail I'd : seemar2411@gmail.com