MOBILE APPLICATION DEVELOPMENT LAB
MINI PROJECT REPORT

205001096
Selvendran MV

205001100
Shivcharan T

205001129
Vishal

CoNOTE

Problem Statement:

Our mini mobile application project focuses on developing a Collaborative Notes Management App with permissions. The main purpose of this app is to enable users to store and manage their notes efficiently and share it. Each user has a login(in-app or Google). Each note comprises a title and a description. The application provides a user-friendly interface for users to easily view all their stored notes. Furthermore, users have the flexibility to modify existing notes and delete them as necessary, offering a comprehensive and straightforward solution for note organization and management. Another feature is that users can share the app with others and give permissions to either read or write in the same note.

Functional Requirements:

Functional requirements include the features that the system provides to the user.

1. Login/Register Page:

- Users should be able to create new accounts or log in securely to the system.

- They can use Google sign-in or in-app sign-in.

- The registration process should include necessary user information and validation checks.

- Passwords should be securely stored using encryption techniques.

- Users should receive appropriate error messages for unsuccessful login attempts.

2. Homepage:

- Upon successful login, users are directed to a personalized homepage.

- The homepage contains the notes with small preview of the content

- Quick access links to essential features, such as creating new notes or accessing shared projects, should be available.

3. Notes Page:

- Users can create, view, and edit their own notes in a clean and intuitive interface.

4. Share Notes:

- Users should be able to share their notes with other users.
- Access permissions (view-only or edit) can be specified when sharing notes.
- Shared notes should appear in the recipient's interface in real-time or with minimal delay.

5. Edit Permissions (Read Only/Write):

- Note owners should have the ability to set permissions for collaborators on a per-user basis.
- Collaborators with "write" permissions can edit the content of shared notes.
- Collaborators with "read-only" permissions can view the notes but not make changes.

Non-Functional Requirements:

These are requirements that are not functional in nature. Specifically, these are the constraints that the system must work within.

- **UI/UX:**
  The application must provide a clear user experience to enable seamless use.

- **Performance:**
  - The system should provide responsive interfaces with low latency, even under peak usage conditions.

- Notes retrieval and editing should occur swiftly, providing a seamless user experience.

- **Scalability:**
  - The system architecture should be scalable to accommodate a growing number of users, notes, and concurrent activities.

.

Functionalities:

-Login/Register Module:

This module facilitates user authentication and registration processes. Users can securely log in with their credentials or create a new account by providing necessary information. The module includes password encryption for security and ensures a smooth and safe onboarding experience for users.

-Note Module:

The Note module allows users to create, view, and edit their notes. Users can employ rich text formatting, embed images and media, and organize their notes efficiently. The module also includes features such as search, sort, and categorization to enhance note organization. Revision history is maintained for tracking changes made to notes over time.

-Share Module:

This module enables users to share their notes with specific individuals or groups. It supports both internal and external collaboration by providing a mechanism to invite collaborators. Users can set access permissions (view-only or edit) when sharing notes, and shared notes are updated in real-time or with minimal delay in the recipients' interfaces.

-Addition Module:

The Addition Module allows users to add new content, notes, or elements to the system. This includes creating new notes, adding collaborators to shared notes, and incorporating new features or functionalities to enhance the user experience. It ensures a straightforward process for users to expand and enrich their work within the system.

-Deletion Module:

This module handles the removal of content or elements from the system. Users can delete individual notes, remove collaborators from shared notes, or delete unnecessary data. The Deletion Module should include confirmation mechanisms to prevent accidental data loss and ensure intentional removal.

-Permission Module:

The Permission Module allows note owners to manage access control for collaborators. Users can set permissions on a per-user basis, specifying whether collaborators have

read-only or write/edit access. The module ensures that collaboration remains secure and transparent by notifying users when permission settings are modified.

-Signout Module:

The Signout Module provides users with a secure way to log out of the system. Upon signing out, the user's session is terminated, enhancing security by preventing unauthorized access to the account. It ensures that users can confidently conclude their sessions and protect their data from unauthorized access

System Design:

1.Architecture:

The architecture allows for collaborative and real-time note creation and sharing while ensuring data consistency and security across multiple devices.

1. Client-Side (Mobile App):
   - User Interface (UI): The mobile app's UI allows users to interact with the application, create, edit, and view notes.

   - Authentication Module: Handles user authentication, login, and registration, often utilizing third-party authentication services like Google or Facebook.

- Note Management Module: Manages the creation, editing, and deletion of notes locally before syncing with the server.

2. Server-Side:
- Authentication Server: Manages user authentication, validating user credentials and generating access tokens for secure communication.

- Application Server: Handles business logic, note creation, modification, and deletion requests. Manages user permissions and collaborations.

3. Database Management:
- Real-time Database: Stores note data and user information in a real-time database, allowing instant updates and synchronization across multiple devices.

4. Third-Party Services:
- Authentication Providers: Integrates with third-party authentication services (e.g., Google, Facebook) for user convenience and security.

5. Security Measures:
- Access Controls: Implements role-based access controls to manage user permissions and restrict access to sensitive data.

2. Database Design:

The Firebase Realtime Database is a cloud-hosted NoSQL database that lets you store and sync data between your users in real time. Storage for Firebase lets you upload and share user generated content, such as images and video, which allows you to build rich media content into your apps. This is used to store images of users and recipes.

3. Collections:

1. User:
    - User Name
    - Email
    - Password

2. Note:
    - Note Id
    - Note title
    - Note content
    - Note user permissions

3. Permission:
    - Permission Type
    - Users associated

Technologies used:

1. Android Studio (Java):

    The application is built for the Android platform using Java within the Android Studio environment. Android Studio provides a comprehensive integrated

development environment (IDE) for Android app development, facilitating the creation of user-friendly and efficient Android applications.

2. Firebase (Authentication, Real-time Database, Storage):
   Firebase is employed to enhance the app's functionality. It serves as a comprehensive backend solution, handling user authentication securely, providing a real-time database for dynamic data updates, and offering storage for multimedia content. Firebase simplifies the development process by offering reliable and scalable cloud services.

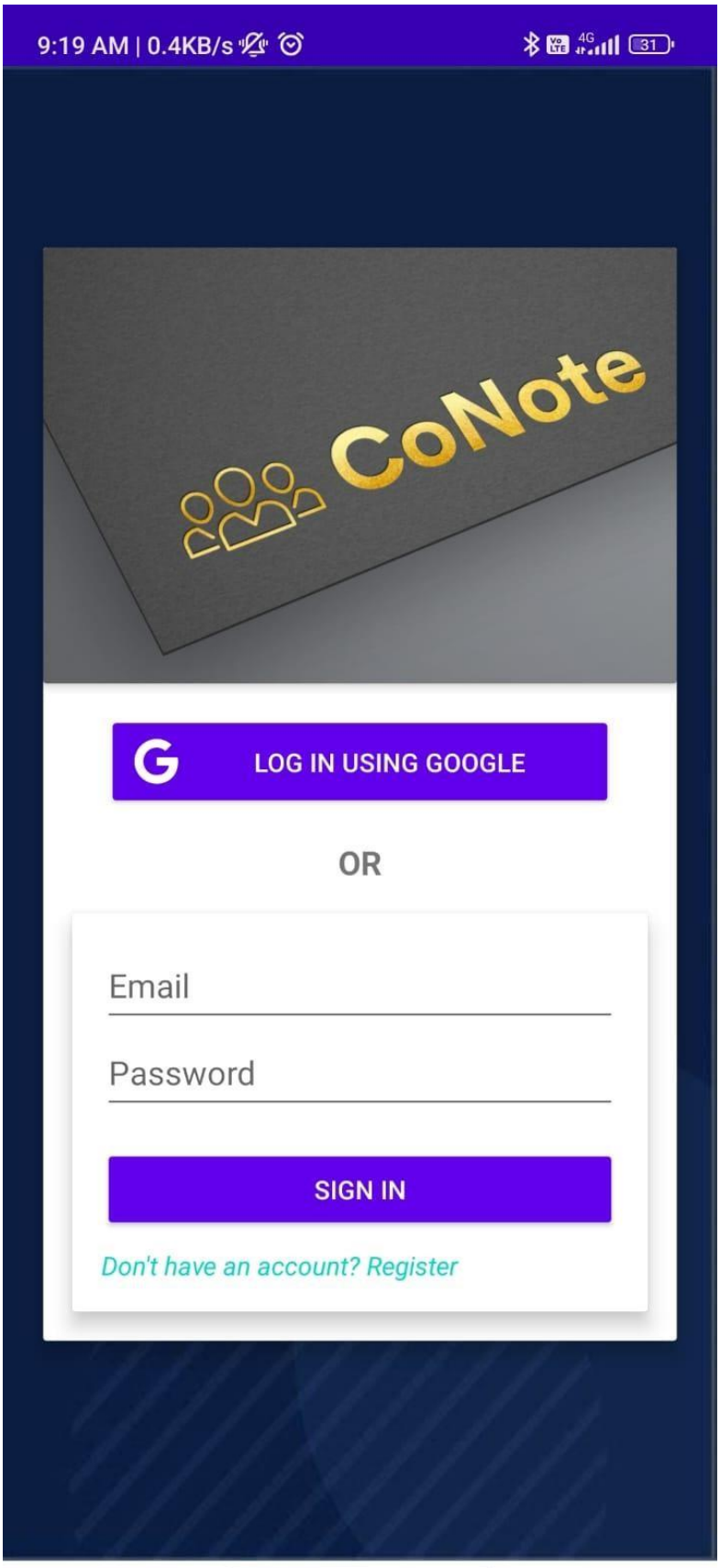
3. RecyclerView:
   The app utilizes the RecyclerView component to dynamically display lists of notes. RecyclerView efficiently handles large data sets, optimizing the user interface by recycling views and improving performance. This ensures a smooth and responsive user experience when navigating through a collection of notes.

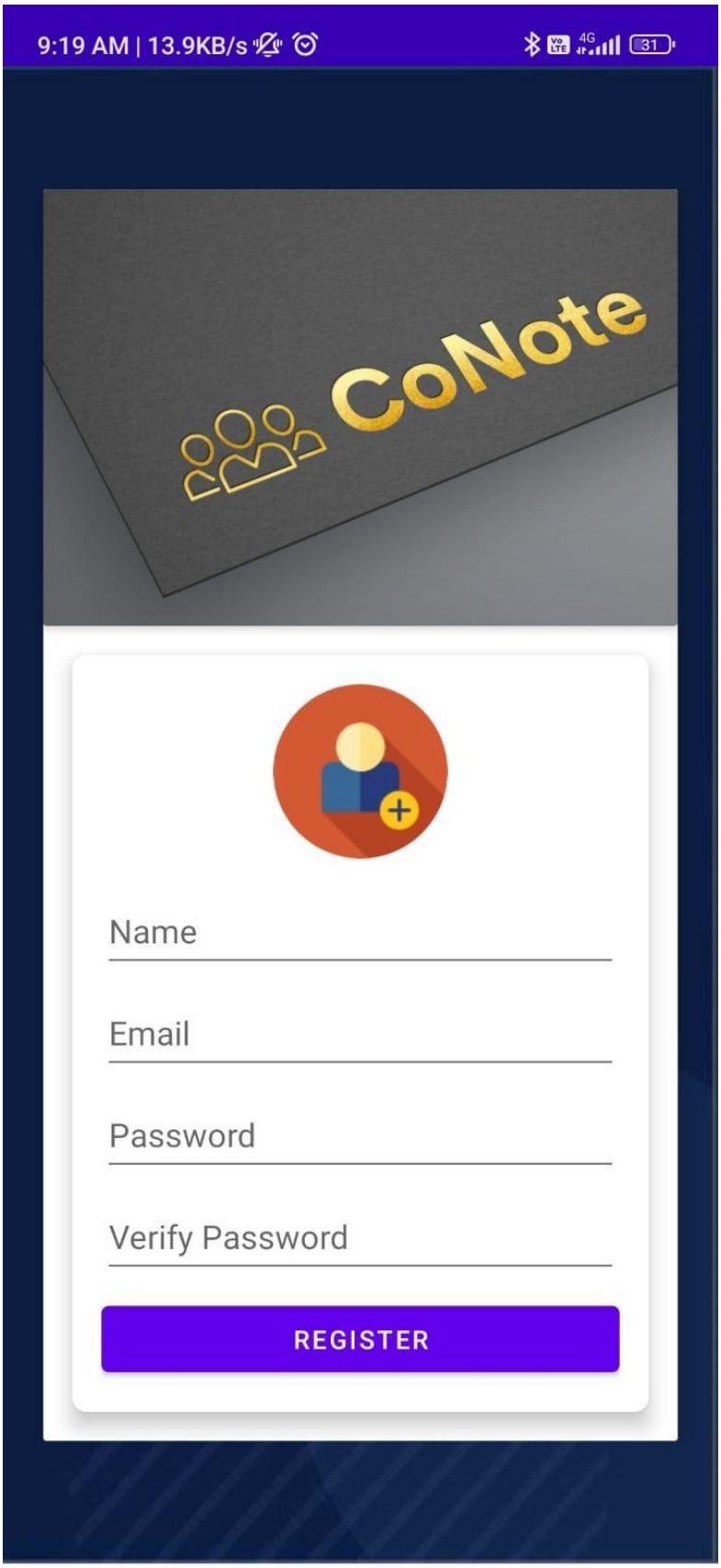
4. Custom Adapter Classes:
   Custom adapter classes are implemented to seamlessly bridge the gap between the app's data source and the user interface. These adapters define how data is presented in predefined layouts/views, enhancing the flexibility and customization of the app's visual representation. Custom adapters contribute to a cohesive and visually appealing user experience.

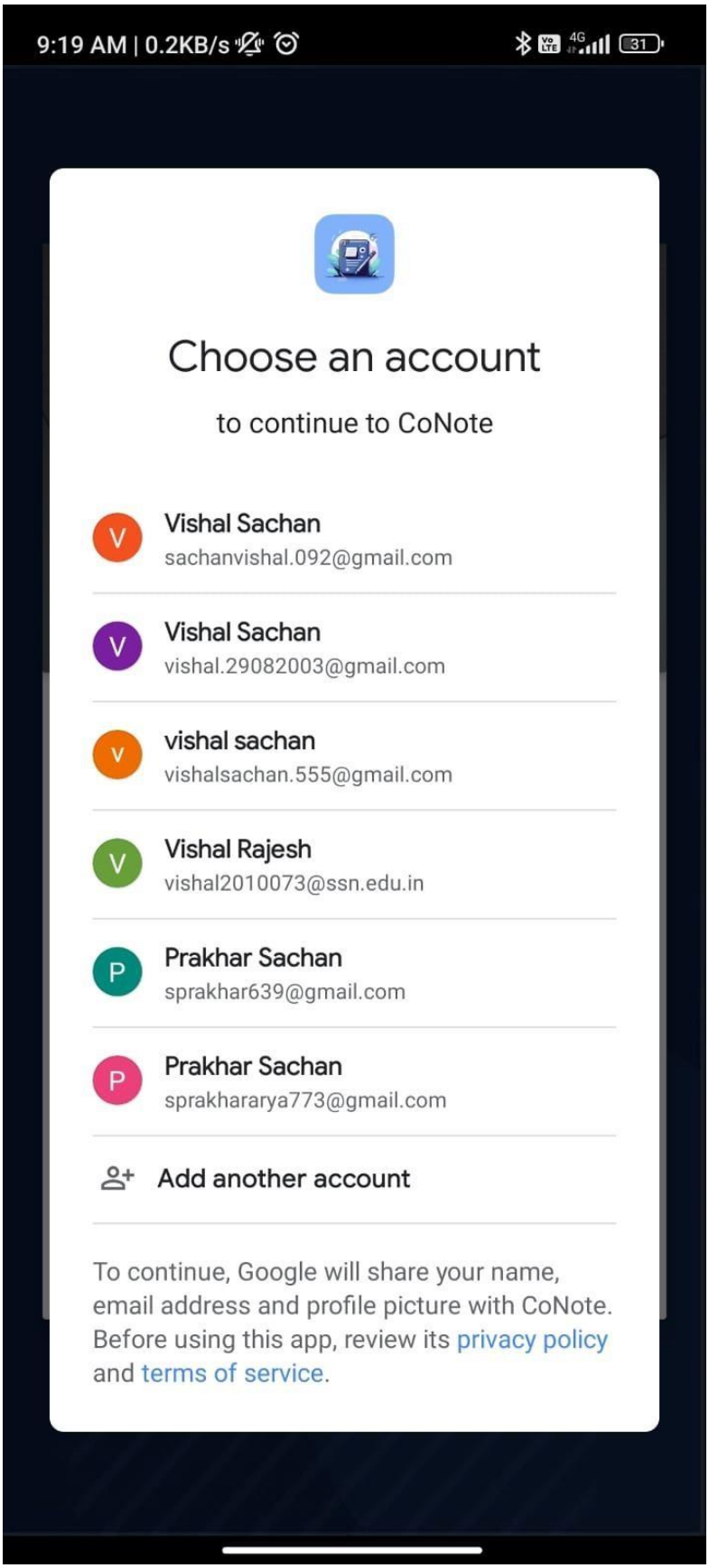
5. Google Login Services:

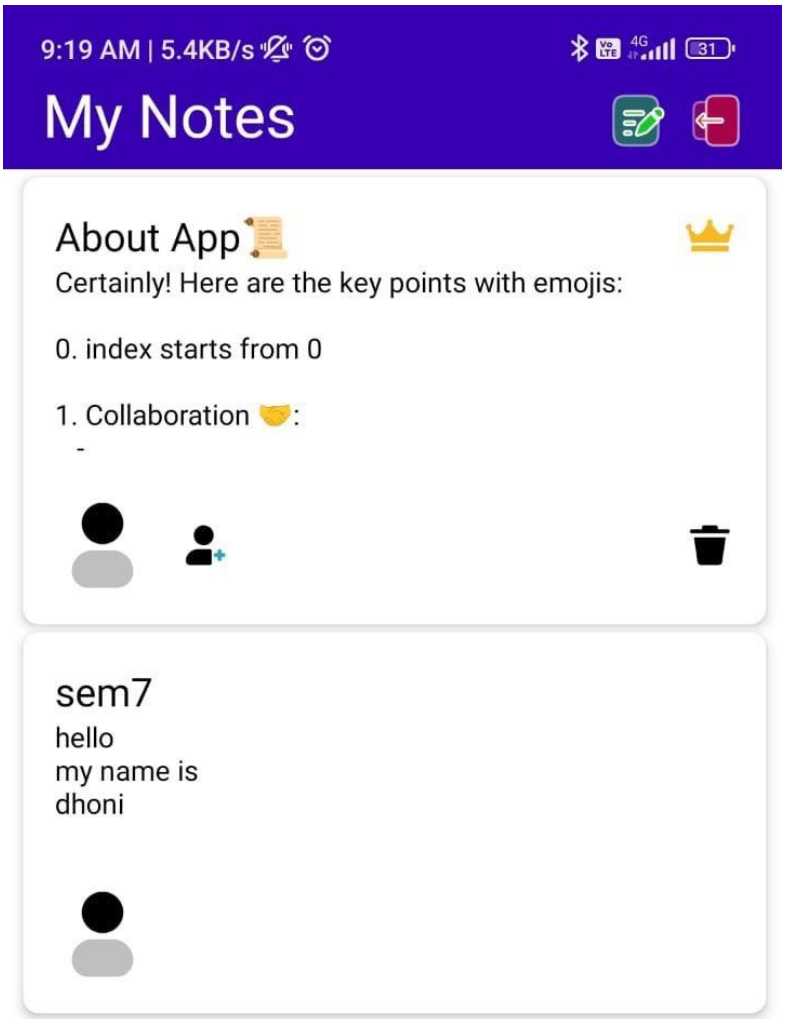
   The app integrates Google login services to streamline the authentication process. This feature enables users to effortlessly log in and sign up using their Google credentials. Leveraging Google login services enhances user convenience, potentially increasing user adoption and engagement while maintaining a secure authentication mechanism.

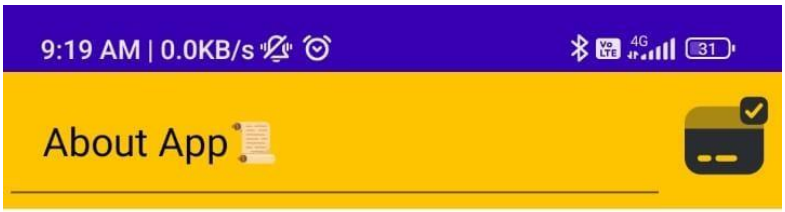
Output Screenshots:


9:19 AM | 0.4KB/s
CoNote
LOG IN USING GOOGLE
OR
Email
Password
SIGN IN
Don't have an account? Register


9:19 AM | 13.9KB/s
CoNote
Name
Email
Password
Verify Password
REGISTER


9:19 AM | 0.2KB/s
Choose an account
to continue to CoNote
Vishal Sachan
sachanvishal.092@gmail.com
Vishal Sachan
vishal.29082003@gmail.com
vishal sachan
vishalsachan.555@gmail.com
Vishal Rajesh
vishal2010073@ssn.edu.in
Prakhar Sachan
sprakhar639@gmail.com
Prakhar Sachan
sprakhararya773@gmail.com
Add another account
To continue, Google will share your name, email address and profile picture with CoNote. Before using this app, review its privacy policy and terms of service.


9:19 AM | 5.4KB/s
My Notes
About App
Certainly! Here are the key points with emojis:
0. index starts from 0
1. Collaboration :
-
sem7
hello
my name is
dhoni



Certainly! Here are the key points with emojis:

0. index starts from 0

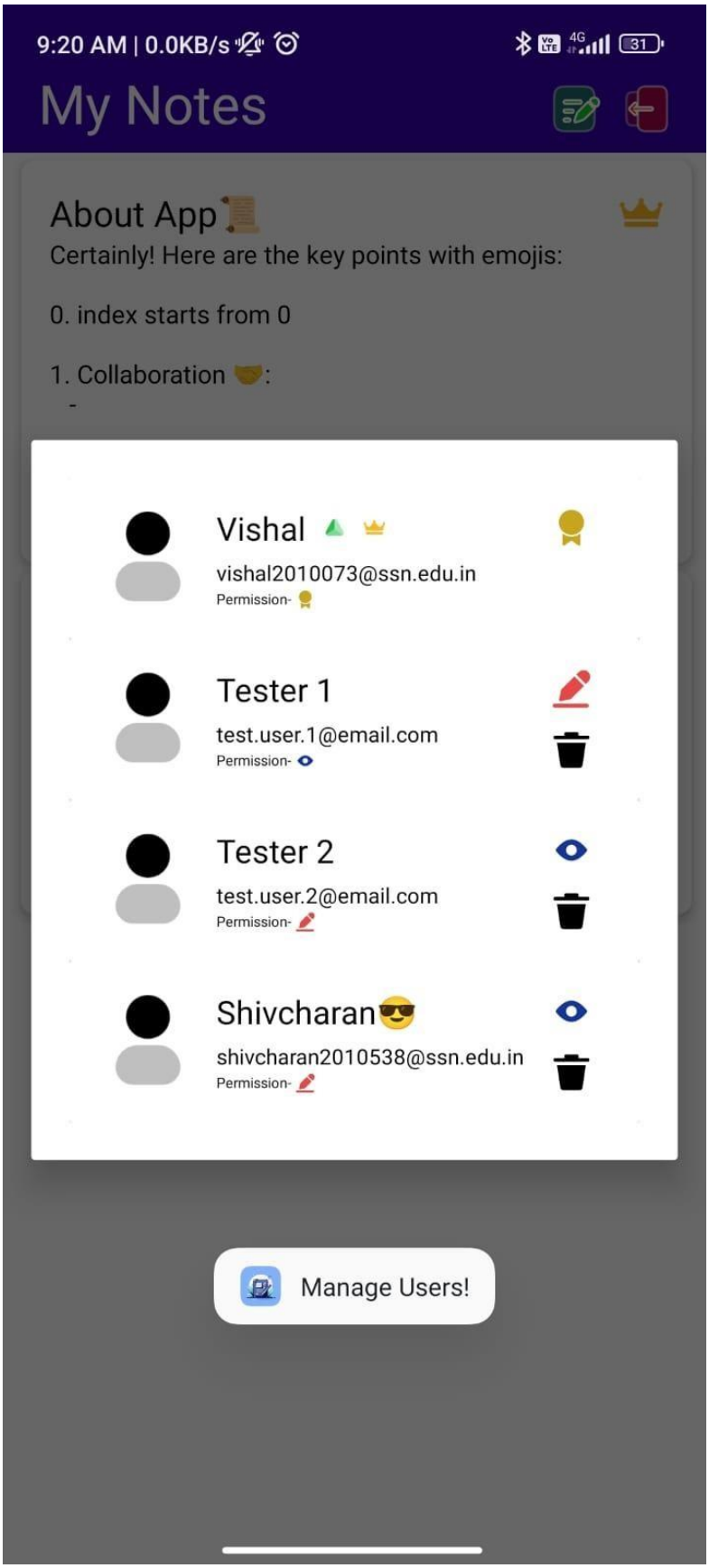
1. Collaboration 🤝:
   - Create and share notes for seamless teamwork.

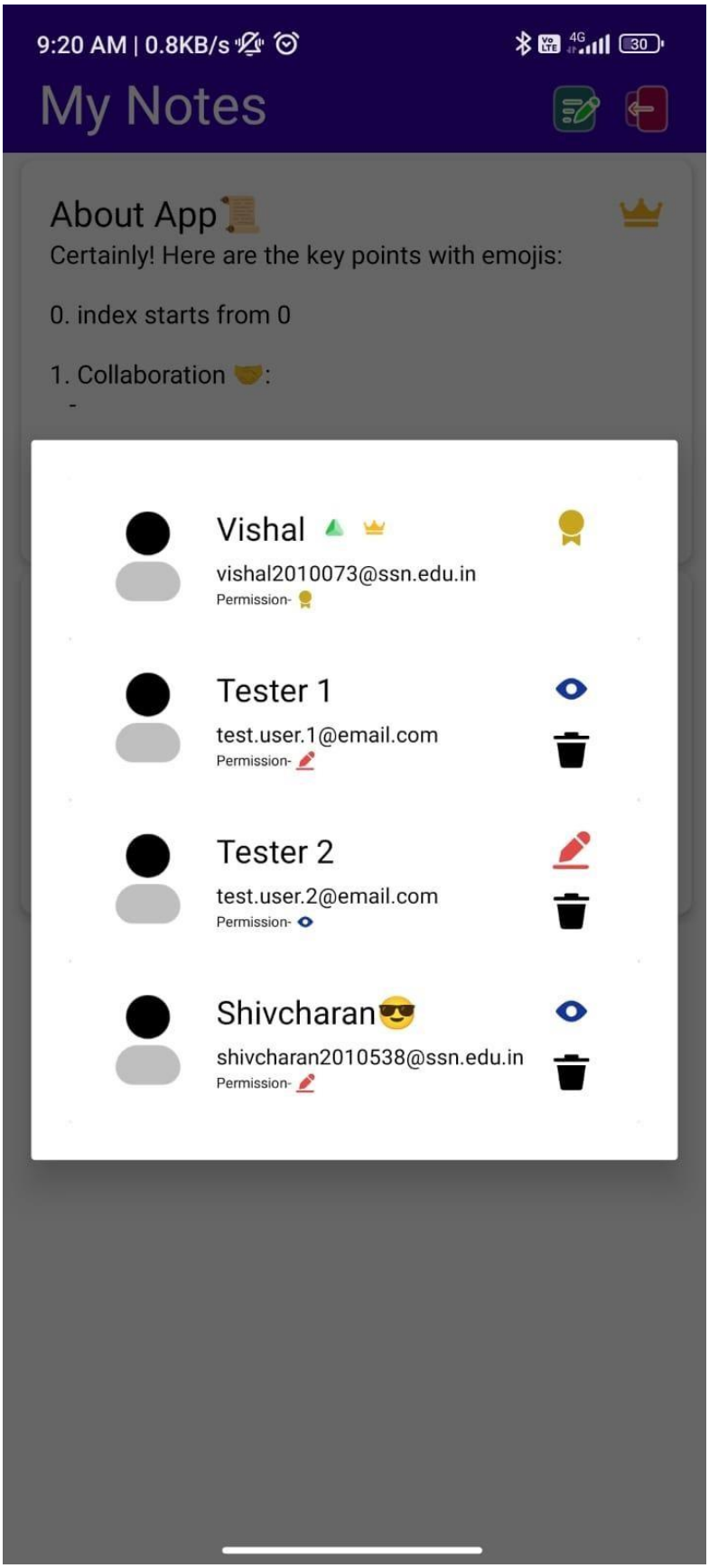
2. Permissions 🔒:
   - Set "View Only" or "Edit" permissions for collaborators.

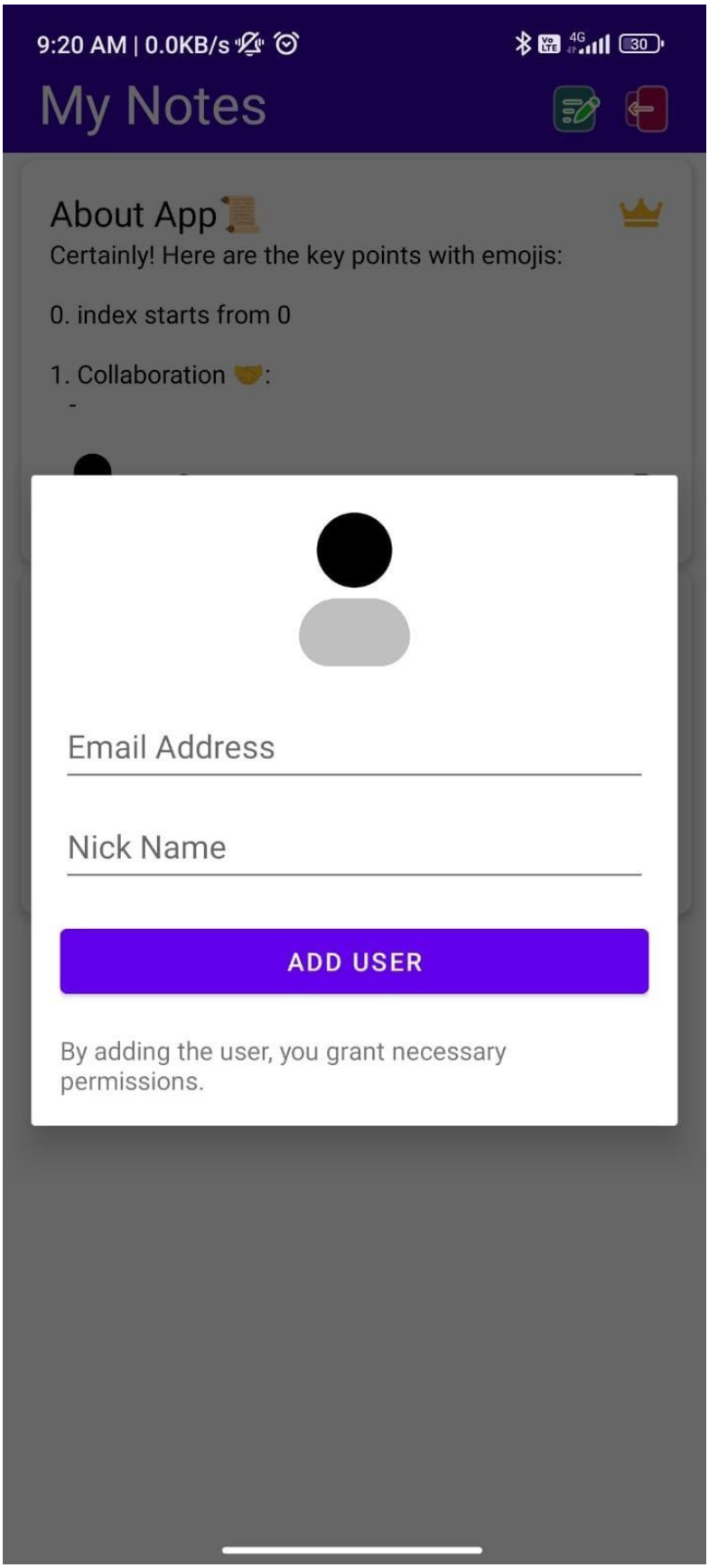
3. Real-time Editing 🔄:
   - Simultaneously collaborate on notes in real-time.

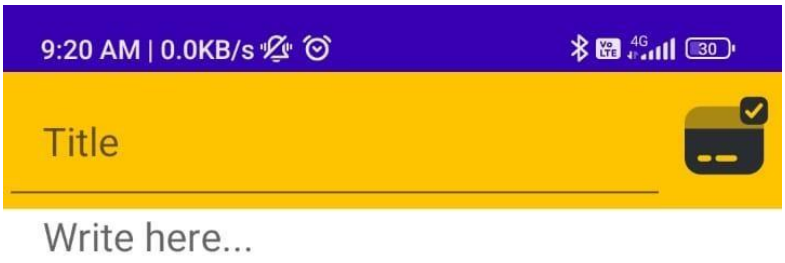
4. Version History 📚:
   - Track changes and revert to previous versions.

5. User-Friendly Interface 🖥️:
   - Intuitive design for easy adoption and efficient use.


9:20 AM | 0.0KB/s
My Notes
About App
Certainly! Here are the key points with emojis:
0. index starts from 0
1. Collaboration :
-
Vishal
vishal2010073@ssn.edu.in
Permission-
Tester 1
test.user.1@email.com
Permission-
Tester 2
test.user.2@email.com
Permission-
Shivcharan
shivcharan2010538@ssn.edu.in
Permission-
Manage Users!


9:20 AM | 0.8KB/s
My Notes
About App
Certainly! Here are the key points with emojis:
0. index starts from 0
1. Collaboration :
-
Vishal
vishal2010073@ssn.edu.in
Permission-
Tester 1
test.user.1@email.com
Permission-
Tester 2
test.user.2@email.com
Permission-
Shivcharan
shivcharan2010538@ssn.edu.in
Permission-


9:20 AM | 0.0KB/s
My Notes
About App
Certainly! Here are the key points with emojis:
0. index starts from 0
1. Collaboration
Email Address
Nick Name
ADD USER
By adding the user, you grant necessary permissions.


9:20 AM | 0.0KB/s
Title
Write here...

Best Practices:

- The code written to develop the application is concise and easily understandable.
- Each intent layout is provided with its own java file.
- The attributes of the database are clearly defined.
- The colour template used is in such a way that it becomes easier to use the application.
- Proper naming convention is used to define variables and functions.

Learning Outcomes:

- We learnt how to use android studio to create applications.
- We learnt how to create a database to store and display the information accordingly.
- We learnt how to use different concepts involved in android development to build the application.
- We learnt how to handle different layouts to provide a seamless experience.
- We learnt how to design the different pages using the underlying xml file.

Code:
MainActivity.java

```
package com.example.project_7;

import androidx.activity.result.ActivityResult;
import androidx.activity.result.ActivityResultCallback;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.IntentSenderRequest;
import androidx.activity.result.contract.ActivityResultContract;
import androidx.activity.result.contract.ActivityResultContracts;

import androidx.appcompat.app.AppCompatActivity;

import android.annotation.TargetApi;
import android.app.Activity;
import android.content.Intent;
import android.content.IntentSender;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.example.project_7.model.UserModel;
import com.google.android.gms.auth.api.identity.BeginSignInRequest;
import com.google.android.gms.auth.api.identity.BeginSignInResult;
import com.google.android.gms.auth.api.identity.Identity;
import com.google.android.gms.auth.api.identity.SignInClient;
import com.google.android.gms.auth.api.identity.SignInCredential;
import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
```

```
import com.google.android.gms.auth.api.signin.GoogleSignInClient;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;
import com.google.android.gms.common.api.ApiException;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import org.jetbrains.annotations.NotNull;

import java.io.Externalizable;
import java.io.Serializable;
import java.util.Map;
import java.util.Optional;
import java.util.concurrent.atomic.AtomicBoolean;

public class MainActivity extends AppCompatActivity {
    private static final int REQ_ONE_TAP = 2;  // Can be any integer unique to the
     Activity.
    private boolean showOneTapUI = true;
    private SignInClient oneTapClient;
    private BeginSignInRequest signUpRequest;

    GoogleSignInClient mGoogleSignInClient;
    private static int RC_SIGN_IN = 100;

    FirebaseDatabase database = FirebaseDatabase.getInstance();
    DatabaseReference usersRef = database.getReference("Users");

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

        // Configure sign-in to request the user's ID, email address, and basic
        // profile. ID and basic profile are included in DEFAULT_SIGN_IN.
        GoogleSignInOptions gso = new
      GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                .requestEmail()
                .build();

        // Build a GoogleSignInClient with the options specified by gso.
        mGoogleSignInClient = GoogleSignIn.getClient(this, gso);

        // Check for existing Google Sign In account, if the user is already signed in
```

```
 // the GoogleSignInAccount will be non-null.

 Button button = findViewById(R.id.button);
 button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
       signIn();
    }
 });

 TextView register = findViewById(R.id.registerTextView);
 register.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
       Intent intent = new Intent(MainActivity.this, RegisterUser.class);
       startActivity(intent);
    }
 });


 Button signIn = findViewById(R.id.signInButton);
 signIn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
       EditText emailEditText = findViewById(R.id.emailEditText);
       EditText passwordField = findViewById(R.id.passwordEditText);
       String email = emailEditText.getText().toString();
       String password = passwordField.getText().toString();

       usersRef.get().addOnCompleteListener(new
OnCompleteListener<DataSnapshot>() {
          @TargetApi(Build.VERSION_CODES.N)
          @Override
          public void onComplete(@NotNull Task<DataSnapshot> task) {
             if (!task.isSuccessful()) {
                Toast.makeText(MainActivity.this, "Error Getting Data!",
Toast.LENGTH_SHORT).show();
             }
             else {
                Map<String, Object> registeredUsers = (Map<String, Object>)
task.getResult().getValue();
                if(registeredUsers != null) {
                   registeredUsers.forEach((key, userData) -> {
                      Optional.ofNullable((Map<String, Object>) userData)
                            .map(userMap -> {
                               String emailId = (String) userMap.get("email");
                               String pass = Optional.ofNullable((String)
userMap.get("password")).orElse(null);
```

```
                                if (emailId != null && emailId.equals(email)) {
                                    if (pass!= null && !pass.isEmpty() &&
    pass.equals(password)) {
                                        SharedPreferences preferences =
    getSharedPreferences("user_prefs", MODE_PRIVATE);
                                        SharedPreferences.Editor editor =
    preferences.edit();
                                        editor.putString("email", email);
                                        editor.apply();

                                        Intent intent = new Intent(MainActivity.this,
    MyNotes.class);
                                        startActivity(intent);
//                                        Toast.makeText(MainActivity.this, "Signed
    In!", Toast.LENGTH_SHORT).show();
                                        finish();
                                        return false;
                                    } else {
                                        Toast.makeText(MainActivity.this, "Wrong
    Password!", Toast.LENGTH_SHORT).show();
                                        return false;
                                    }
                                }
                                return true; // Indicate that the iteration should
    continue
                            })
                            .orElseGet(() -> {
                                // Handle the case where userData is not a Map
                                Toast.makeText(MainActivity.this, "Invalid User
    Data!", Toast.LENGTH_SHORT).show();
                                return false; // Indicate that the iteration should
    continue
                            });
                    });

                } else {
                    Toast.makeText(MainActivity.this, "Email Not Registered!",
    Toast.LENGTH_SHORT).show();
                }
            }
        }

    });
  });

}
```

```
    private void signIn() {
        Intent signInIntent = mGoogleSignInClient.getSignInIntent();
        startActivityForResult(signInIntent, RC_SIGN_IN);
    }
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        // Result returned from launching the Intent from
      GoogleSignInClient.getSignInIntent(...);
        if (requestCode == RC_SIGN_IN) {
            // The Task returned from this call is always completed, no need to attach
            // a listener.
            Task<GoogleSignInAccount> task =
      GoogleSignIn.getSignedInAccountFromIntent(data);
            handleSignInResult(task);
        }
    }

    private void handleSignInResult(Task<GoogleSignInAccount> completedTask)
      {
        try {
            GoogleSignInAccount acct =
      completedTask.getResult(ApiException.class);
            if (acct != null) {
                SharedPreferences preferences = getSharedPreferences("user_prefs",
      MODE_PRIVATE);
                SharedPreferences.Editor editor = preferences.edit();
                editor.putString("email", acct.getEmail());
                editor.apply();
                registerUserFromGoogleSignIn(acct.getGivenName(), acct.getEmail());
//                Toast.makeText(MainActivity.this, "Signed In!",
      Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(getApplicationContext(), MyNotes.class);
                startActivity(intent);
                finish();
            } else {
                Toast.makeText(MainActivity.this, "Not Signed In!",
      Toast.LENGTH_SHORT).show();
            }
        } catch (ApiException e) {
            // The ApiException status code indicates the detailed failure reason.
            // Please refer to the GoogleSignInStatusCodes class reference for more
      information.
            Log.d("Sign In Error :", e.getMessage());
        }
    }
```

```
    private void registerUserFromGoogleSignIn(String name, String email) {
        usersRef.get().addOnCompleteListener(new
    OnCompleteListener<DataSnapshot>() {
            @TargetApi(Build.VERSION_CODES.N)
            @Override
            public void onComplete(@NotNull Task<DataSnapshot> task) {
                if (!task.isSuccessful()) {
                    Toast.makeText(MainActivity.this, "Error Getting Data!",
    Toast.LENGTH_SHORT).show();
                }
                else {
                    Map<String, Object> registeredUsers = (Map<String, Object>)
    task.getResult().getValue();
                    AtomicBoolean alreadyRegistered = new AtomicBoolean(false);
                    if(registeredUsers != null) {
                        registeredUsers.forEach((key, userData) -> {
                            String emailId = (String) ((Map<String, Object>)
    userData).get("email");
                            if(emailId.equals(email)) {
                                alreadyRegistered.set(true);
//                                Toast.makeText(MainActivity.this, "Already Registered -
    "+email, Toast.LENGTH_SHORT).show();
                                return;
                            }
                        });
                    }
                    if(!alreadyRegistered.get()){
                        registerUser(name, email);
                    }
                }
            }

            private void registerUser(String name, String email) {
                UserModel user = new UserModel(name, email);
                usersRef.push().setValue(user);
            }
        });
    }

}
//
//oneTapClient = Identity.getSignInClient(this);
//        signUpRequest = BeginSignInRequest.builder()
//
    .setGoogleIdTokenRequestOptions(BeginSignInRequest.GoogleIdTokenReq
    uestOptions.builder()
//        .setSupported(true)
```

```
//        // Your server's client ID, not your Android client ID.
//        .setServerClientId(getString(R.string.client_id))
//        // Show all accounts on the device.
//        .setFilterByAuthorizedAccounts(false)
//        .build())
//        .build();
//
//
//        ActivityResultLauncher<IntentSenderRequest> activityResultLauncher =
//        registerForActivityResult(new
    ActivityResultContracts.StartIntentSenderForResult(),
//        new ActivityResultCallback<ActivityResult>() {
//@Override
//public void onActivityResult(ActivityResult result) {
//        if(result.getResultCode() == Activity.RESULT_OK) {
//        try {
//        SignInCredential credential =
    oneTapClient.getSignInCredentialFromIntent(result.getData());
//        String idToken = credential.getGoogleIdToken();
//        if (idToken !=  null) {
//        // Got an ID token from Google. Use it to authenticate
//        // with your backend.
//
//        Log.d("TAG", "Got ID token.");
//
//        String email = credential.getId();
//        Toast.makeText(getApplicationContext(), "Email: "+email,
    Toast.LENGTH_SHORT).show();
//


//        }
//        } catch (ApiException e) {
//        // ...
//        //Caller has been temporarily blocked due to too many canceled sign-in
    prompts.
//        // dial this in ur android device ->  *#*#66382723#*#*
//        e.printStackTrace();
//        }
//        }
//        }
//        });
//
//        Button button = findViewById(R.id.button);
//        button.setOnClickListener(new View.OnClickListener() {
//@Override
//public void onClick(View view) {
//        oneTapClient.beginSignIn(signUpRequest)
//        .addOnSuccessListener(MainActivity.this, new
```

```
    OnSuccessListener<BeginSignInResult>() {
//@Override
//public void onSuccess(BeginSignInResult result) {
//        IntentSenderRequest intentSenderRequest =
//        new
    IntentSenderRequest.Builder(result.getPendingIntent().getIntentSender()).buil
    d();
//        activityResultLauncher.launch(intentSenderRequest);
//        }
//        })
//        .addOnFailureListener(MainActivity.this, new OnFailureListener() {
//@Override
//public void onFailure( Exception e) {
//        // No Google Accounts found. Just continue presenting the signed-out UI.
//        Log.d("TAG", e.getLocalizedMessage());
//        }
//        });
//        }
//        });
```

Activity.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:background="@drawable/background"
    tools:context=".MainActivity">

    <androidx.cardview.widget.CardView
        android:id="@+id/signInCard"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="100dp"
        android:layout_marginStart="22dp"
        android:layout_marginEnd="22dp"
        app:cardElevation="8dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_horizontal"
    android:orientation="vertical">

    <androidx.cardview.widget.CardView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:cardElevation="4dp">

        <!-- ImageView to hold the image -->
        <ImageView
            android:id="@+id/cardImageView"
            android:layout_width="match_parent"
            android:layout_height="240dp"
            android:scaleType="centerCrop"
            android:src="@drawable/app_name"/>

    </androidx.cardview.widget.CardView>

    <Button
        android:id="@+id/button"
        style="@style/Widget.AppCompat.Button"
        android:layout_width="280dp"
        android:layout_height="54dp"
        android:layout_marginTop="16dp"
        android:elevation="@dimen/cardview_compat_inset_shadow"
        android:text="Log In Using Google"
        app:icon="@drawable/ic_google"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="parent" />

    <TextView
        android:id="@+id/termsTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="OR"
        android:textSize="18sp"
        android:textStyle="bold" />

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
android:layout_margin="16dp"
app:cardElevation="8dp">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <!-- Username Field -->
    <EditText
        android:id="@+id/emailEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Email"
        android:inputType="text"
        android:minHeight="48dp" />

    <!-- Password Field -->
    <EditText
        android:id="@+id/passwordEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Password"
        android:inputType="textPassword"
        android:minHeight="48dp" />

    <!-- Sign In Button -->
    <Button
        android:id="@+id/signInButton"
        style="@style/Widget.AppCompat.Button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="Sign In" />

    <!-- Register TextView -->
    <TextView
        android:id="@+id/registerTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="Don't have an account? Register"
        android:textColor="?attr/colorAccent"
        android:textStyle="italic" />

</LinearLayout>
```

```
            </androidx.cardview.widget.CardView>
        </LinearLayout>
    </androidx.cardview.widget.CardView>
</androidx.constraintlayout.widget.ConstraintLayout>
```