WEATHER APP

BACHELOR OF

TECHNOLOGY IN

COMPUTER SCIENCE AND ENGINEERING

BY

Vanamatla Anitha **(22501A05J0)**

Vemulapalli Gnaneswari **(22501A05J3)**

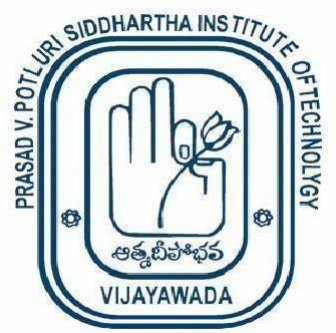
Tatiparti Meghana **(22501A05H8)**

Veeranki Revanth **(23505A0517)**

Under the Guidance of

Mr. Michael Sadgun Rao Kona,

Assistant Professor



PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Permanently affiliated to JNTU: Kakinada, Approved by AICTE)

(An NBA & NAAC A++ accredited and ISO 9001:2015 certified institution)

Kanuru, Vijayawada-520007

2024-25

PRASAD V POTLURI

SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Permanently affiliated to JNTU :: Kakinada, Approved by AICTE)

(An NBA & NAAC A++ accredited and ISO 9001:2015 certified institution)

Kanuru, Vijayawada – 520007

CERTIFICATE

This is to certify that the project report title **"Weather App"** is the bonafide work of **Vanamatla Anitha (22501A05J0), Vemulapalli Gnaneswari (22501A05J3), Tatiparti Meghana(22501A05H8), Veeranki Revanth(23505A0517)** in partial fulfilment of completing the Academic project in Mobile App Development (20SA8651) during the academic year 2024-25.

Signature of the Incharge **Signature of the HOD**

INDEX

1.ABSTRACT

The **Weather App** is a mobile application designed to provide real-time weather updates for any entered city using the **OpenWeatherMap API**. The app features a clean and interactive user interface built with **Android Studio and Java**, allowing users to search for weather conditions by city name. Key weather details such as **temperature, humidity, and weather conditions** are displayed in a structured format.

The app integrates **Retrofit**, a popular HTTP client for API requests, ensuring fast and reliable data retrieval. A **navigation drawer** enhances usability, enabling users to access different sections such as **Profile, Settings, and Logout**. User authentication is managed through a **sign-in/sign-up system**, and an **SQLite database** is used for storing user details.

To improve user experience, the app features **dynamic weather icons**, real-time search functionality, and error handling for invalid city inputs. The design follows **modern UI/UX principles**, ensuring responsiveness across different screen sizes. Future improvements may include **extended forecasts, location-based weather retrieval, offline weather caching, and push notifications** for weather alerts. This app serves as a **reliable, user-friendly weather tracking tool**, making weather data easily accessible for users worldwide.

2.SDG Justification

The **Sustainable Development Goals (SDGs)** offer a framework for addressing global challenges. Below is a justification of how a **Weather App** can align with several SDGs:

1.SDG 3: Good Health and Well-being

The app provides real-time weather data, helping users prepare for extreme conditions, reducing health risks like heat strokes and respiratory issues.

2.SDG 11: Sustainable Cities and Communities

By offering weather data, the app aids communities in responding to extreme events like floods and storms, supporting sustainable urban planning.

3.SDG 13: Climate Action

The app raises awareness of climate change by providing real-time climate updates, empowering users to take climate-resilient actions.

4.SDG 9: Industry, Innovation, and Infrastructure

Weather Apps foster innovation, providing accurate data to industries like agriculture, construction, and transportation for better planning.

5.SDG 2: Zero Hunger

Weather data, such as rainfall and temperature forecasts, supports farmers in crop planning and reducing crop failure risks.

6.SDG7:Affordable and Clean Energy

The app helps users optimize energy use by providing weather forecasts, promoting solar energy use and reducing reliance on non-renewable sources.

3. INTRODUCTION

Weather conditions significantly impact our daily routines, travel, agriculture, and overall lifestyle. Access to **real-time weather updates** is crucial for planning and decision-making. The **Weather App** is developed to provide users with instant and accurate weather information for any city by integrating the **OpenWeatherMap API**. This Android-based mobile application is designed to fetch live weather data, including **temperature, humidity, longitude, latitude, and weather conditions**, enhancing user convenience.

The app is built using **Android Studio and Java**, leveraging **Retrofit** for efficient API communication. When a user enters a city name, the app fetches real-time weather data and displays it in an easy-to-read format. A **navigation drawer** enhances user experience, offering quick access to key sections such as **Profile, Settings, and Logout**. Users can sign in or sign up to personalize their experience, with their details stored in an **SQLite database** for easy management.

The application follows a **modular and scalable design**, allowing smooth performance even with continuous API requests. The UI is **simple yet interactive**, ensuring that users can effortlessly obtain weather updates. Additionally, features such as **profile management** enable users to edit their details, ensuring a personalized experience.

Future improvements for the Weather App include **extended forecasts, real-time weather alerts, and GPS-based location tracking** for automatic weather updates. Furthermore, features like **push notifications for severe weather conditions** will be integrated to provide users with timely warnings.

This Weather App is an **efficient, lightweight, and user-friendly solution** for obtaining instant weather updates. It is designed to serve individuals who need quick weather information while ensuring **accuracy, reliability, and ease of use**. As technology advances, further enhancements will make the app more intelligent and responsive, catering to the evolving needs of users worldwide.

4. OBJECTIVES AND SCOPE OF THE PROJECT

Objectives:

The primary goal of this project is to develop a mobile-based Weather App that provides users with realtime weather updates for any city they search. The app aims to offer a seamless user experience while ensuring accurate and up-to-date weather information. The key objectives of the project are:

1. **Real-time Weather Fetching** – Implement OpenWeatherMap API to display temperature, humidity, and weather conditions.
2. **User Authentication** – Enable users to Sign in and Sign up securely to personalize their experience.
3. **Profile Management** – Allow users to view and update their name, email, and password.
4. **Data Storage** – Use SQLite database to store user credentials and profile details.
5. **Seamless Navigation** – Integrate a navigation drawer to access Home, Profile, Settings, and Logout.
6. **Efficient API Communication** – Implement Retrofit library for fast and optimized API calls.
7. **User-friendly UI** – Ensure a responsive and visually appealing interface for easy usability.
8. **Performance Optimization** – Reduce loading times and optimize data usage for smooth operation.
9. **Future Enhancements** – Provide scope for additional features like extended forecasts, location tracking, and weather alerts

Scope of the Project:

This project is focused on building a functional and user-friendly Android-based Weather App that will benefit different users:

1. **General Users** – Anyone looking for quick and accurate weather updates.
2. **Travelers** – People who want to check the weather conditions before traveling.
3. **Students & Researchers** – Users who need **climate data** for studies and projects.
4. **Outdoor Enthusiasts** – Hikers, cyclists, and athletes who rely on weather forecasts.

The app ensures real-time weather tracking, personalized user experience, and secure authentication. With future scalability options, it is designed to be efficient, reliable, and easy to use, making it an essential tool for weather monitoring.

5. SOFTWARE USED

In the development of the WeatherApp, the following software technologies were utilized to ensure a responsive, efficient, and user-friendly experience:

Frontend Technologies • XML (Extensible Markup Language)

XML was used to design the user interface (UI) for the Android application. It defined the layout of various UI elements such as text fields, buttons, navigation menus, and weather icons, ensuring a structured and interactive design.

- **Java**

Java was used as the primary programming language to implement the app's logic and functionality.It ensured smooth communication between the UI components and backend services, handling user interactions and API calls efficiently.

- **Android Studio**

Android Studio served as the main IDE for developing and testing the WeatherApp. It provided various debugging tools, emulators, and performance optimization features, ensuring an efficient development process.

Backend Technologies • Open Weather Map API

Open Weather Map API was used to fetch real-time weather data such as temperature, humidity, wind speed, and weather conditions based on the entered city name. It provided accurate and up-to-date weather information, enhancing the app's reliability.

- **SQLite (Local Database)**

SQLite was used for storing user data such as login credentials and profile details. It ensured that the app functions even without an internet connection, maintaining a seamless user experience.

API Integration • Retrofit (REST API Library)

Retrofit was used to handle API requests and parse responses efficiently. It allowed seamless communication between the frontend and OpenWeatherMap API, ensuring smooth data retrieval.

- **GSON (JSON Parser)**

GSON was used for converting JSON responses from the weather API into Java objects, making data handling easier.

Design and Documentation Tools •

Canva

Canva was used for designing UI elements, icons, and app prototypes. It ensured a visually appealing user interface that aligns with modern app design principles.

- **Microsoft Word**

Microsoft Word was used for documenting the project, writing reports, and preparing structured documentation.
It ensured clarity and proper record-keeping of the project's development process.

Version Control and Deployment

GitHub

- GitHub was used for version control, collaboration, and code management.
- It ensured that all changes in the app were tracked, stored, and managed efficiently, allowing future updates and bug fixes.

6. PROPOSED MODEL

The WeatherApp follows a well-structured model designed to provide real-time weather updates, seamless navigation, and an enhanced user experience. The model ensures that users receive accurate weather forecasts while maintaining data security, offline access, and user-friendly interaction.

1. User Management & Authentication

- Secure login and registration system using SQLite (with Firebase authentication as a future enhancement).
- User credentials (email, password) are securely stored, ensuring authorized access.
- Profile management allows users to update their name, email, and password with real-time reflection of changes.

2. Weather Data Retrieval & Display

- Users can search for a city to get real-time weather details.
- The app fetches weather data using OpenWeatherMap API via Retrofit.
- Displayed data includes:
  - City Name
  - Temperature (°C or °F based on preference) ○ Humidity levels ○ Wind Speed
- Dynamic UI updates ensure that changes reflect instantly.

3. Navigation & Profile Management

- The app includes a Navigation Drawer for easy access to:
  - Home – Main weather display page. ○ Profile – Displays user details (name, email) with an option to edit.
  - Settings – (Future enhancement: theme selection, units preference, notifications).
  - Logout – Securely logs out the user.
- Intuitive interface with smooth navigation ensures an engaging user experience.

4. Offline Support & Data Persistence

- The app stores the last fetched weather data in SQLite database.
- Users can view the last retrieved weather report even without an internet connection.
- This feature ensures accessibility in case of network issues.

5. API Integration & Error Handling

- Uses Retrofit to fetch weather details from OpenWeatherMap API.
- JSON response is processed using GSON for seamless data conversion.
- Error handling mechanisms include:
  - Invalid city name alerts. ○ Internet connectivity checks.
  - Loading indicators for real-time API responses.

6. User-Friendly Interface & Customization

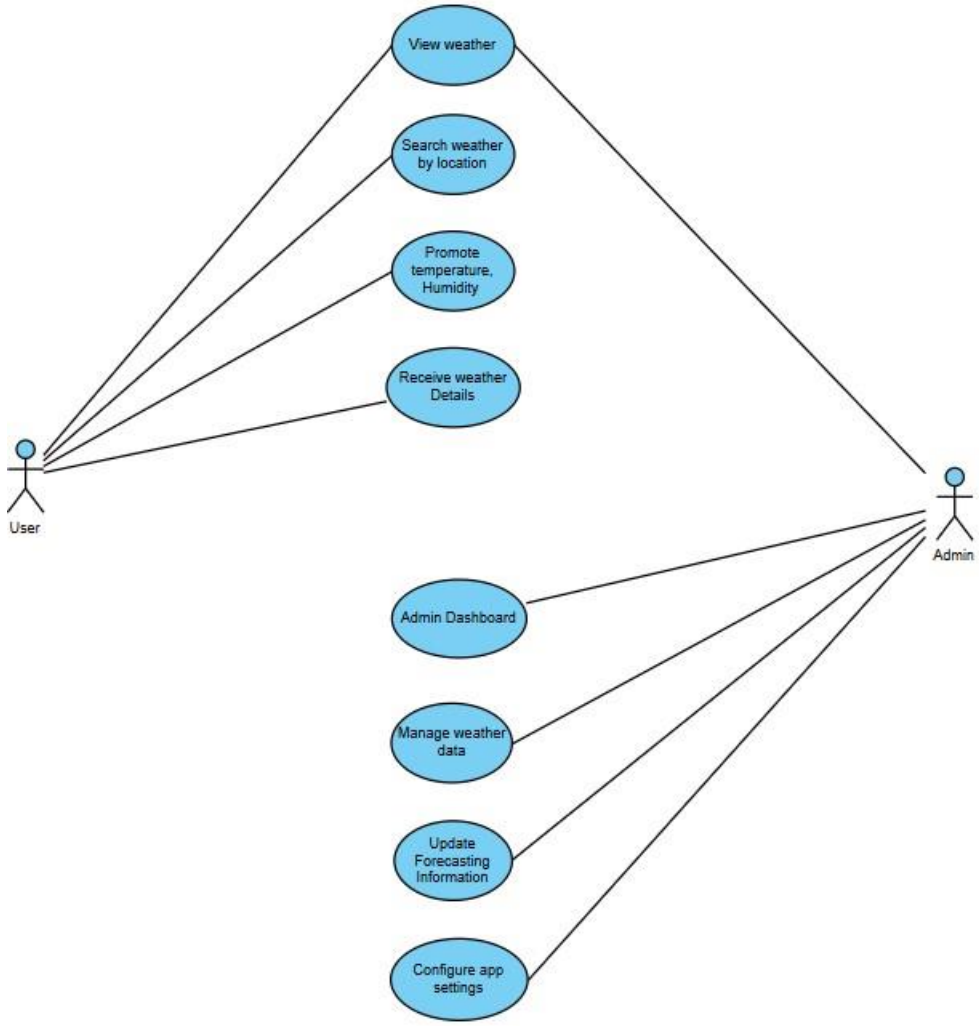
- Material Design principles ensure a visually appealing UI.

- Simple layout with clear labels and icons for easy interaction.
- Future enhancements include:
  - Theme selection (light/dark mode) ○ Language preferences
  - Push notifications for severe weather alerts

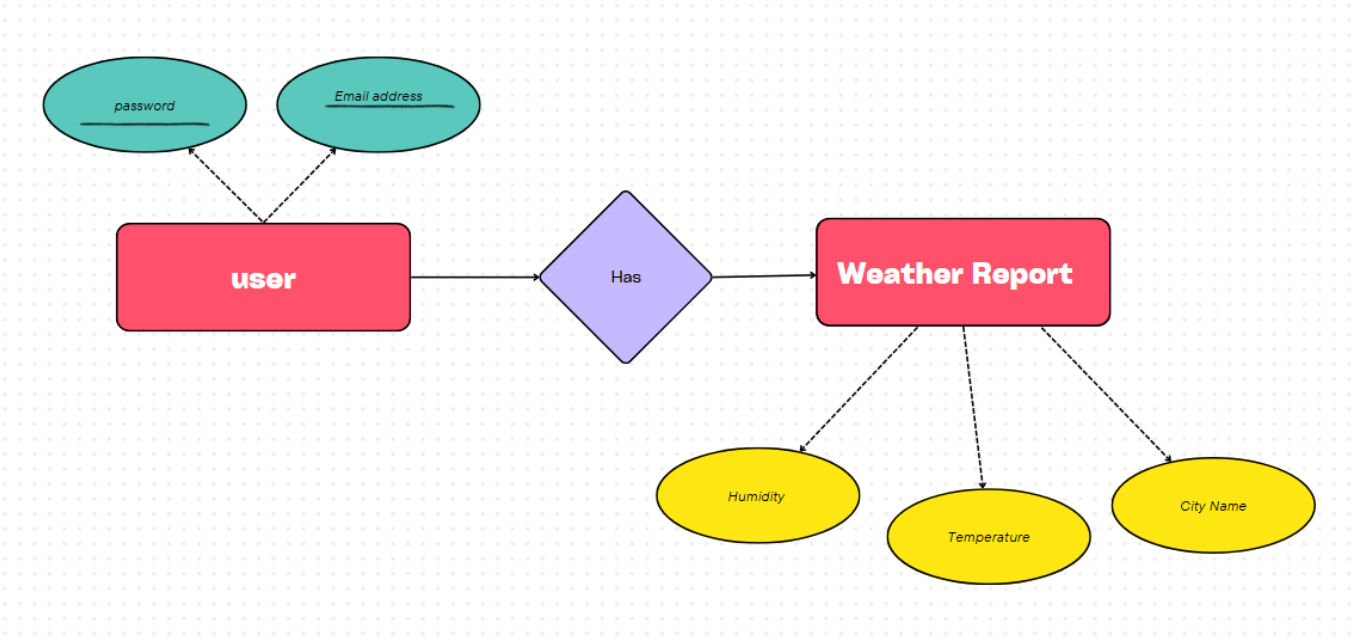
7. Real-Time Data Updates & Expansion Scope

- The app is scalable and can be expanded to include advanced features, such as:
  - GPS-based automatic location detection. ○ Hourly and weekly weather forecasts.
  - AI-powered weather predictions using machine learning models.

Use Case Diagram:



Entity-Relationship (ER) Diagram:

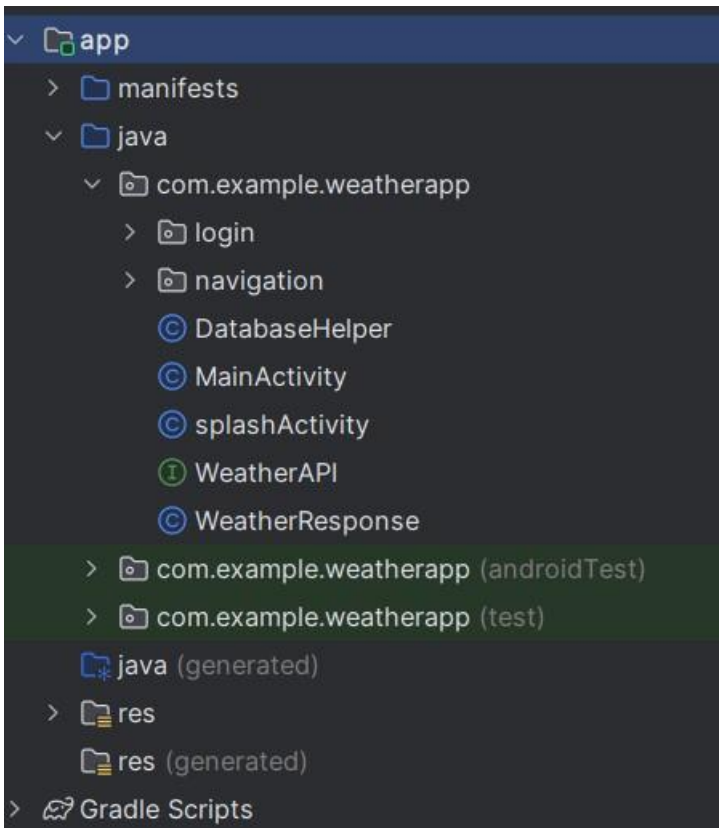


7. SAMPLE CODE

GitHub repository link:

https://github.com/Anithavanamatla/weatherapp

Folder Structure:



Program:

//SplashActivity.java

```
package com.example.weatherapp;

import android.content.Intent;
import android.os.Bundle; import
android.os.Handler;
import androidx.appcompat.app.AppCompatActivity;

import com.example.weatherapp.login.SignInActivity;

public class splashActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);        setContentView(R.layout.activity_splash);

       new Handler().postDelayed(() -> {
```

```
        Intent intent = new Intent(splashActivity.this, SignInActivity.class);
startActivity(intent);            finish();
    }, 3000); // 3 seconds delay
  }
}
```

//Login

//a) SignInActivity.java

```
package com.example.weatherapp.login;

import android.content.Intent; import
android.database.Cursor; import android.os.Bundle;
import android.view.View; import
android.widget.Button; import
android.widget.EditText; import
android.widget.TextView; import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import com.example.weatherapp.DatabaseHelper;
import com.example.weatherapp.MainActivity; import
com.example.weatherapp.R; import
com.example.weatherapp.login.SignUpActivity;
import com.example.weatherapp.navigation.ProfileActivity;

public class SignInActivity extends AppCompatActivity {
private EditText edtEmail, edtPassword;    private Button
btnSignIn;    private TextView createAccount;
  private DatabaseHelper dbHelper;

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

    // Initialize UI components        edtEmail =
findViewById(R.id.edtEmail);       edtPassword =
findViewById(R.id.edtPassword);       btnSignIn =
findViewById(R.id.btnSignIn);       createAccount =
findViewById(R.id.createAccount);       dbHelper = new
DatabaseHelper(this);

    // Login Button Click
    btnSignIn.setOnClickListener(view -> loginUser());

    // Redirect to Signup Page
    createAccount.setOnClickListener(view -> {
```

```
        Intent intent = new Intent(SignInActivity.this, SignUpActivity.class);
startActivity(intent);
    });
  }

  private void loginUser() {
    String email = edtEmail.getText().toString().trim();
    String password = edtPassword.getText().toString().trim();

    if (email.isEmpty() || password.isEmpty()) {
      Toast.makeText(this, "Please enter both email and password", Toast.LENGTH_SHORT).show();
return;
    }

    // Validate credentials
    if (dbHelper.checkUser(email, password)) {
      Toast.makeText(this, "Login Successful!", Toast.LENGTH_SHORT).show();

      // Pass email to MainActivity
      Intent mainIntent = new Intent(SignInActivity.this, MainActivity.class);
mainIntent.putExtra("EMAIL", email);          startActivity(mainIntent);

      finish(); // Close login activity
} else {
      Toast.makeText(this, "Invalid email or password", Toast.LENGTH_SHORT).show();
    }
  }

}
```

//b)SignUpActivity.java

```
package com.example.weatherapp.login;

import android.content.Intent;
import android.os.Bundle; import
android.view.View; import
android.widget.Button; import
android.widget.EditText; import
android.widget.TextView; import
android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import com.example.weatherapp.DatabaseHelper; import
com.example.weatherapp.R;
import com.example.weatherapp.login.SignInActivity;

public class SignUpActivity extends AppCompatActivity {
```

```
    private EditText edtEmail, edtPassword;
private Button btnSignUp;     private
TextView AlreadyAccount;
    private DatabaseHelper dbHelper;

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

        // Initialize UI components         edtEmail =
findViewById(R.id.edtEmail);         edtPassword =
findViewById(R.id.edtPassword);         btnSignUp =
findViewById(R.id.btnSignUp);
        AlreadyAccount = findViewById(R.id.AlreadyAccount);
dbHelper = new DatabaseHelper(this);

        // SignUp Button Click
        btnSignUp.setOnClickListener(view -> registerUser());

        // Redirect to Login Page
        AlreadyAccount.setOnClickListener(view -> {
            Intent intent = new Intent(SignUpActivity.this, SignInActivity.class);
startActivity(intent);
            finish();
        });
    }

    private void registerUser() {
        String email = edtEmail.getText().toString().trim();
        String password = edtPassword.getText().toString().trim();

        if (email.isEmpty() || password.isEmpty()) {
            Toast.makeText(this, "Please fill in all fields", Toast.LENGTH_SHORT).show();
return;
        }

        if (dbHelper.checkEmailExists(email)) {
            Toast.makeText(this, "Email already registered! Try logging in.",
Toast.LENGTH_SHORT).show();
            return;
        }

        boolean insert = dbHelper.insertUser(email, password);
if (insert) {
```

```
            Toast.makeText(this, "Signup Successful! Please Login.", Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(SignUpActivity.this, SignInActivity.class);
startActivity(intent);
            finish(); // Close signup activity
} else {
            Toast.makeText(this, "Signup Failed! Try Again.", Toast.LENGTH_SHORT).show();
        }
    }
}
//Navigation
```

1) ProfileActivity.java

```
package com.example.weatherapp.navigation;

import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle; import
android.widget.Button; import
android.widget.ImageView; import
android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.example.weatherapp.DatabaseHelper; import
com.example.weatherapp.R;
import com.example.weatherapp.login.SignInActivity;

public class ProfileActivity extends AppCompatActivity {
private TextView txtUserName, txtUserEmail;     private
Button btnEditProfile, btnLogout;     private ImageView
btnBack;     private DatabaseHelper dbHelper;
    private String loggedInEmail = "";

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

        // Initialize UI components
        txtUserName = findViewById(R.id.txtUserName);
txtUserEmail = findViewById(R.id.txtUserEmail);
btnEditProfile = findViewById(R.id.btnEditProfile);         btnLogout
= findViewById(R.id.btnLogout);         btnBack =
findViewById(R.id.btnBack);

        // Initialize database helper
```

```
        dbHelper = new DatabaseHelper(this);

        // Get logged-in email from Intent         Intent intent
= getIntent();         if (intent != null &&
intent.hasExtra("EMAIL")) {             loggedInEmail =
intent.getStringExtra("EMAIL");
        }

        // Fetch and display user details
fetchUserDetails();

        // Edit Profile Button
        btnEditProfile.setOnClickListener(view -> {
            Intent editIntent = new Intent(ProfileActivity.this, EditProfileActivity.class);
editIntent.putExtra("EMAIL", loggedInEmail);
            startActivityForResult(editIntent, 1);
        });

        // Logout Button
        btnLogout.setOnClickListener(view -> {
            Toast.makeText(ProfileActivity.this, "Logged Out!", Toast.LENGTH_SHORT).show();
startActivity(new Intent(ProfileActivity.this, SignInActivity.class));             finish();
        });

        // Back Button
        btnBack.setOnClickListener(view -> finish());
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
super.onActivityResult(requestCode, resultCode, data);         if (requestCode ==
1 && resultCode == RESULT_OK) {             fetchUserDetails(); // Refresh
updated data
        }
    }

    private void fetchUserDetails() {
        Cursor cursor = dbHelper.getUserData(loggedInEmail);
if (cursor != null && cursor.moveToFirst()) { int nameIndex
= cursor.getColumnIndex("name");
            int emailIndex = cursor.getColumnIndex("email");

            if (nameIndex != -1 && emailIndex != -1) {
String userName = cursor.getString(nameIndex);
loggedInEmail = cursor.getString(emailIndex);
```

```
            txtUserName.setText(userName);
            txtUserEmail.setText(loggedInEmail);
        }
        cursor.close();
} else {
        Toast.makeText(this, "Failed to load profile details", Toast.LENGTH_SHORT).show();
    }
  }

  @Override    protected void
onResume() {
super.onResume();
    fetchUserDetails();
  }
}
```

2)Settings.java

```
package com.example.weatherapp.navigation;

import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle; import
android.widget.Button; import
android.widget.EditText; import
android.widget.ImageView; import
android.widget.Switch;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.example.weatherapp.DatabaseHelper; import
com.example.weatherapp.R;
import com.example.weatherapp.login.SignInActivity;

public class settingsActivity extends AppCompatActivity {
private EditText etUserName, etUserEmail;    private
Switch switchNotifications;    private Button
btnSaveSettings, btnLogout;    private ImageView
btnBack;    private DatabaseHelper dbHelper;    private
String loggedInEmail;

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

```
        // Initialize UI components
        etUserName = findViewById(R.id.etUserName);
etUserEmail = findViewById(R.id.etUserEmail);
switchNotifications = findViewById(R.id.switchNotifications);
btnSaveSettings = findViewById(R.id.btnSaveSettings);
btnLogout = findViewById(R.id.btnLogout);        btnBack =
findViewById(R.id.btnBack);

        // Initialize database helper
        dbHelper = new DatabaseHelper(this);

        // Get logged-in email from Intent        Intent
intent = getIntent();        if (intent != null &&
intent.hasExtra("EMAIL")) {
            loggedInEmail = intent.getStringExtra("EMAIL");
        } else {
            loggedInEmail = "";
        }

        // Fetch and display user details
loadUserData();

        // Save button click
        btnSaveSettings.setOnClickListener(view -> saveUserSettings());

        // Logout button
        btnLogout.setOnClickListener(view -> {
            Toast.makeText(settingsActivity.this, "Logged Out!", Toast.LENGTH_SHORT).show();
startActivity(new Intent(settingsActivity.this, SignInActivity.class));            finish();
        });

        // Back button
        btnBack.setOnClickListener(view -> finish());
    }

    private void loadUserData() {
        Cursor cursor = dbHelper.getUserData(loggedInEmail);
if (cursor != null && cursor.moveToFirst()) { int nameIndex
= cursor.getColumnIndex("name");
            int emailIndex = cursor.getColumnIndex("email");

            if (nameIndex != -1) {
                etUserName.setText(cursor.getString(nameIndex));
            } else {
                Toast.makeText(this, "Column 'name' not found!", Toast.LENGTH_SHORT).show();
```

```
            }

            if (emailIndex != -1) {
                etUserEmail.setText(cursor.getString(emailIndex));
            } else {
                Toast.makeText(this, "Column 'email' not found!", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(this, "User not found!", Toast.LENGTH_SHORT).show();
        }
        if (cursor != null) {
cursor.close();
        }
    }


    private void saveUserSettings() {
        String newUserName = etUserName.getText().toString().trim();
        String newUserEmail = etUserEmail.getText().toString().trim();

        if (newUserName.isEmpty() || newUserEmail.isEmpty()) {
            Toast.makeText(this, "Fields cannot be empty", Toast.LENGTH_SHORT).show();
return;
        }

        boolean isUpdated = dbHelper.updateUser(loggedInEmail, newUserName, newUserEmail);
if (isUpdated) {
            Toast.makeText(this, "Settings updated!", Toast.LENGTH_SHORT).show();
loggedInEmail = newUserEmail;
        } else {
            Toast.makeText(this, "Update failed!", Toast.LENGTH_SHORT).show();
        }
    }
}
```

3) EditProfileActivity.java

```
package com.example.weatherapp.navigation;

import android.content.Intent; import
android.database.Cursor; import
android.os.Bundle; import
android.widget.Button; import
android.widget.EditText; import
android.widget.ImageView; import
android.widget.Switch;
import android.widget.Toast;
```

```
import androidx.appcompat.app.AppCompatActivity;

import com.example.weatherapp.DatabaseHelper; import
com.example.weatherapp.R;
import com.example.weatherapp.login.SignInActivity;

public class settingsActivity extends AppCompatActivity {
private EditText etUserName, etUserEmail;     private
Switch switchNotifications;     private Button
btnSaveSettings, btnLogout;     private ImageView
btnBack;     private DatabaseHelper dbHelper;
   private String loggedInEmail;

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

       // Initialize UI components
       etUserName = findViewById(R.id.etUserName);
etUserEmail = findViewById(R.id.etUserEmail);
switchNotifications = findViewById(R.id.switchNotifications);
btnSaveSettings = findViewById(R.id.btnSaveSettings);
btnLogout = findViewById(R.id.btnLogout);
       btnBack = findViewById(R.id.btnBack);

       // Initialize database helper
dbHelper = new DatabaseHelper(this);

       // Get logged-in email from Intent         Intent intent
= getIntent();         if (intent != null &&
intent.hasExtra("EMAIL")) {             loggedInEmail =
intent.getStringExtra("EMAIL");
       } else {
          loggedInEmail = "";
       }

       // Fetch and display user details
loadUserData();

       // Save button click
       btnSaveSettings.setOnClickListener(view -> saveUserSettings());

       // Logout button
       btnLogout.setOnClickListener(view -> {
```

```
        Toast.makeText(settingsActivity.this, "Logged Out!", Toast.LENGTH_SHORT).show();
startActivity(new Intent(settingsActivity.this, SignInActivity.class));            finish();
    });

    // Back button
    btnBack.setOnClickListener(view -> finish());
  }

  private void loadUserData() {
    Cursor cursor = dbHelper.getUserData(loggedInEmail);
if (cursor != null && cursor.moveToFirst()) {            int
nameIndex = cursor.getColumnIndex("name");            int
emailIndex = cursor.getColumnIndex("email");

      if (nameIndex != -1) {
        etUserName.setText(cursor.getString(nameIndex));
      } else {
        Toast.makeText(this, "Column 'name' not found!", Toast.LENGTH_SHORT).show();
      }

      if (emailIndex != -1) {
        etUserEmail.setText(cursor.getString(emailIndex));
      } else {
        Toast.makeText(this, "Column 'email' not found!", Toast.LENGTH_SHORT).show();
      }
    } else {
      Toast.makeText(this, "User not found!", Toast.LENGTH_SHORT).show();
    }
    if (cursor != null) {
cursor.close();
    }
  }


  private void saveUserSettings() {
    String newUserName = etUserName.getText().toString().trim();
    String newUserEmail = etUserEmail.getText().toString().trim();

    if (newUserName.isEmpty() || newUserEmail.isEmpty()) {
      Toast.makeText(this, "Fields cannot be empty", Toast.LENGTH_SHORT).show();
return;
    }

    boolean isUpdated = dbHelper.updateUser(loggedInEmail, newUserName, newUserEmail);
if (isUpdated) {
```

```
            Toast.makeText(this, "Settings updated!", Toast.LENGTH_SHORT).show();
loggedInEmail = newUserEmail;
        } else {
            Toast.makeText(this, "Update failed!", Toast.LENGTH_SHORT).show();
        }
    }
}
```

//MainActivity.java

```
package com.example.weatherapp;

import android.content.Intent;
import android.os.Bundle; import
android.view.MenuItem; import
android.view.View; import
android.widget.Button; import
android.widget.EditText; import
android.widget.ImageView; import
android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull; import
androidx.appcompat.app.ActionBarDrawerToggle; import
androidx.appcompat.app.AppCompatActivity; import
androidx.drawerlayout.widget.DrawerLayout;

import com.bumptech.glide.Glide;
import com.example.weatherapp.login.SignInActivity; import
com.example.weatherapp.navigation.ProfileActivity; import
com.example.weatherapp.navigation.settingsActivity; import
com.google.android.material.navigation.NavigationView;

import retrofit2.Call; import
retrofit2.Callback; import
retrofit2.Response; import
retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class MainActivity extends AppCompatActivity {
    private EditText edtCityName;
private Button btnGetWeather;     private
TextView txtCity, txtTemperature,
txtHumidity;     private ImageView
imgWeatherIcon;     private
DrawerLayout drawerLayout;     private
NavigationView navigationView;
    private ActionBarDrawerToggle toggle;
```

```
   private static final String API_KEY = "c9b54fe4ef77b6e128c14763e6cadfdd";
private static final String BASE_URL = "https://api.openweathermap.org/data/2.5/";

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

       // Initialize UI Elements
       edtCityName = findViewById(R.id.edtCityName);
btnGetWeather = findViewById(R.id.btnGetWeather);
txtCity = findViewById(R.id.txtCity);        txtTemperature =
findViewById(R.id.txtTemperature);        txtHumidity =
findViewById(R.id.txtHumidity);        imgWeatherIcon =
findViewById(R.id.imgWeatherIcon);        drawerLayout =
findViewById(R.id.drawer_layout);        navigationView =
findViewById(R.id.navigation_view);

       // Setup Navigation Drawer
       toggle = new ActionBarDrawerToggle(this, drawerLayout, R.string.open, R.string.close);
drawerLayout.addDrawerListener(toggle);
       toggle.syncState();
       getSupportActionBar().setDisplayHomeAsUpEnabled(true);

       navigationView.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
           @Override
           public boolean onNavigationItemSelected(@NonNull MenuItem item) {
int id = item.getItemId();

               if (id == R.id.nav_home) {
                   Toast.makeText(MainActivity.this, "Home Clicked", Toast.LENGTH_SHORT).show();
               } else if (id == R.id.nav_profile) {
                   // Open ProfileActivity with User Email
                   Intent profileIntent = new Intent(MainActivity.this, ProfileActivity.class);
                   profileIntent.putExtra("EMAIL", "user@example.com");
startActivity(profileIntent);                } else if (id ==
R.id.nav_settings) {
                   Intent settingIntent = new Intent(MainActivity.this, settingsActivity.class);
startActivity(settingIntent);
               } else if (id == R.id.nav_logout) {
                   Toast.makeText(MainActivity.this, "Logged Out!", Toast.LENGTH_SHORT).show();
startActivity(new Intent(MainActivity.this, SignInActivity.class));                finish();
} else {
```

```
                return false;  // Handle unexpected items
            }

            drawerLayout.closeDrawers();
return true;
        }
    });

    // Button Click to Fetch Weather
    btnGetWeather.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String city = edtCityName.getText().toString().trim();
if (!city.isEmpty()) {                  fetchWeatherData(city);
            } else {
                Toast.makeText(MainActivity.this, "Please enter a city name",
Toast.LENGTH_SHORT).show();
            }
        }
    });
  }

  private void fetchWeatherData(String city) {
    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build();

    WeatherAPI weatherAPI = retrofit.create(WeatherAPI.class);
    Call<WeatherResponse> call = weatherAPI.getWeather(city, API_KEY, "metric");

    call.enqueue(new Callback<WeatherResponse>() {
        @Override
        public void onResponse(Call<WeatherResponse> call, Response<WeatherResponse> response)
{
            if (response.isSuccessful() && response.body() != null) {
WeatherResponse weather = response.body();                  txtCity.setText(" ⚲
City: " + weather.getName());
                txtTemperature.setText(" 🌡 Temperature: " + weather.getMain().getTemp() + "°C");
txtHumidity.setText(" 💧 Humidity: " + weather.getMain().getHumidity() + "%");
                // Load weather icon using Glide
                String iconUrl = "https://openweathermap.org/img/wn/" +
weather.getWeather().get(0).getIcon() + "@2x.png";
                Glide.with(MainActivity.this).load(iconUrl).into(imgWeatherIcon);
            } else {
```

```
                Toast.makeText(MainActivity.this, "City not found", Toast.LENGTH_SHORT).show();
            }
        }

        @Override
        public void onFailure(Call<WeatherResponse> call, Throwable t) {
            Toast.makeText(MainActivity.this, "Error fetching data", Toast.LENGTH_SHORT).show();
        }
    });
}

@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    return toggle.onOptionsItemSelected(item) || super.onOptionsItemSelected(item);
}
}
```

//WeatherResonse.java

```
package com.example.weatherapp;

import java.util.List;

public class WeatherResponse {
private Coord coord;    private
List<Weather> weather;    private
Main main;    private Wind wind;
private Clouds clouds;    private
Sys sys;
    private String name;

    public Coord getCoord() { return coord; }
    public List<Weather> getWeather() { return weather; }
public Main getMain() { return main; }    public Wind
getWind() { return wind; }    public Clouds getClouds()
{ return clouds; }    public Sys getSys() { return sys; }
    public String getName() { return name; }

    public static class Coord {
        private double lon;        private
double lat;        public double getLon() {
return lon; }
        public double getLat() { return lat; }
    }
```

```
   public static class Weather {
private String main;        private
String description;
       private String icon;

       public String getMain() { return main; }        public
String getDescription() { return description; }        public
String getIcon() { return icon; }
   }

   public static class Main {
private double temp;        private
double feels_like;        private
double temp_min;        private
double temp_max;        private
int pressure;
       private int humidity;

       public double getTemp() { return temp; }
public double getFeelsLike() { return feels_like; }
public double getTempMin() { return temp_min; }
public double getTempMax() { return temp_max; }
public int getPressure() { return pressure; }
       public int getHumidity() { return humidity; }
   }

   public static class Wind {
private double speed;        private
int deg;
       private double gust;

       public double getSpeed() { return speed; }
public int getDeg() { return deg; }        public
double getGust() { return gust; }
   }

   public static class Clouds {
private int all;
       public int getAll() { return all; }
   }
   public static class Sys {
private String country;
private long sunrise;
       private long sunset;
```

```
        public String getCountry() { return country; }
public long getSunrise() { return sunrise; }         public
long getSunset() { return sunset; }
    }
}
```

//WeatherApi.java

```
package com.example.weatherapp;

import retrofit2.Call; import
retrofit2.http.GET;
import retrofit2.http.Query;

public interface WeatherAPI {
    @GET("data/2.5/weather")
    Call<WeatherResponse> getWeather(
@Query("q") String city,
            @Query("appid") String apiKey,
            @Query("units") String units // Use "metric" for Celsius, "imperial" for Fahrenheit
    );
}
```

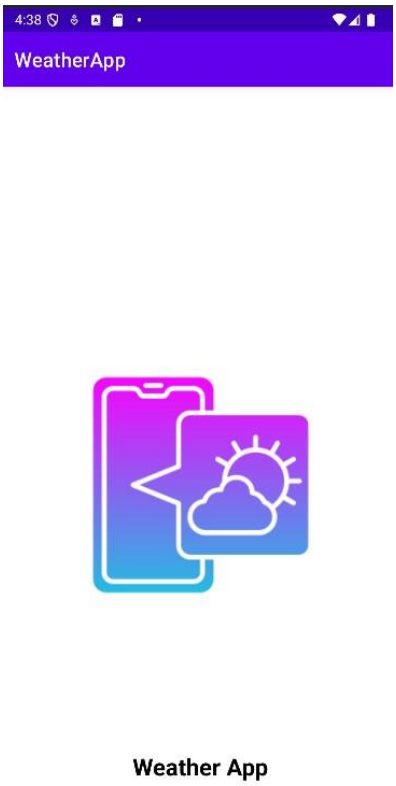
//DatabaseHelper.java package

```
com.example.weatherapp;

import android.content.ContentValues; import
android.content.Context; import
android.database.Cursor; import
android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper {
private static final String DATABASE_NAME = "users.db";
private static final int DATABASE_VERSION = 1;     private
static final String TABLE_USERS = "users";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override public void
    onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE users (id
    INTEGER PRIMARY KEY
    AUTOINCREMENT, name TEXT, email
    TEXT UNIQUE, password TEXT)");
    }
```

```
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
db.execSQL("DROP TABLE IF EXISTS " + TABLE_USERS);        onCreate(db);
    }

    public boolean insertUser(String email, String password) {
        SQLiteDatabase db = this.getWritableDatabase();
ContentValues values = new ContentValues();
        values.put("email", email);        values.put("password",
password);        long result = db.insert(TABLE_USERS, null,
values);        return result != -1; // Returns true if insertion is
successful    }

    public boolean checkUser(String email, String password) {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery("SELECT * FROM " + TABLE_USERS + " WHERE email=? AND
password=?", new String[]{email, password});        boolean exists = cursor.getCount() > 0;
cursor.close();        return exists;
    }
    public boolean checkEmailExists(String email) {
SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery("SELECT * FROM " + TABLE_USERS + " WHERE email=?", new
String[]{email});
        boolean exists = cursor.getCount() > 0;
cursor.close();        return exists;
    }
    public Cursor getUserData(String email) {
        SQLiteDatabase db = this.getReadableDatabase();
        return db.rawQuery("SELECT name, email FROM users WHERE email = ?", new String[]{email});
    }
    public boolean updateUser(String oldEmail, String newUserName, String newUserEmail, String
newPassword) {
        SQLiteDatabase db = this.getWritableDatabase();
ContentValues contentValues = new ContentValues();
contentValues.put("name", newUserName);
contentValues.put("email", newUserEmail);
        if (!newPassword.isEmpty()) {
            contentValues.put("password", newPassword);
        }

        int result = db.update("users", contentValues, "email = ?", new String[]{oldEmail});
return result > 0;
    }

    public boolean updateUserName(String email, String newName) {
```

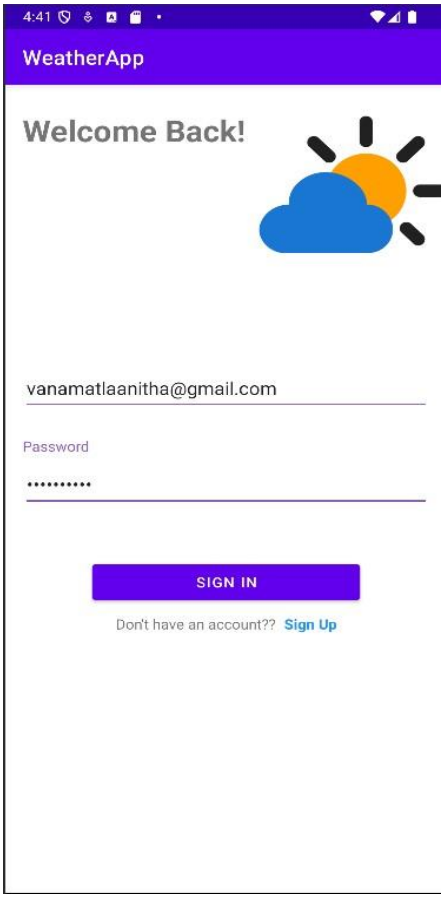
```
        SQLiteDatabase db = this.getWritableDatabase();
ContentValues values = new ContentValues();
values.put("name", newName);

        int result = db.update("users", values, "email=?", new String[]{email});
return result > 0; // Returns true if update is successful
    }
    public boolean updateUser(String oldEmail, String newName, String newEmail) {
        SQLiteDatabase db = this.getWritableDatabase();
ContentValues contentValues = new ContentValues();
contentValues.put("name", newName);
        contentValues.put("email", newEmail);

        long result = db.update("users", contentValues, "email=?", new String[]{oldEmail});
return result != -1;  // Returns true if updated, false otherwise
    }
}
```

8.RESULT/OUTPUT SCREENS



Splashpage – opening page

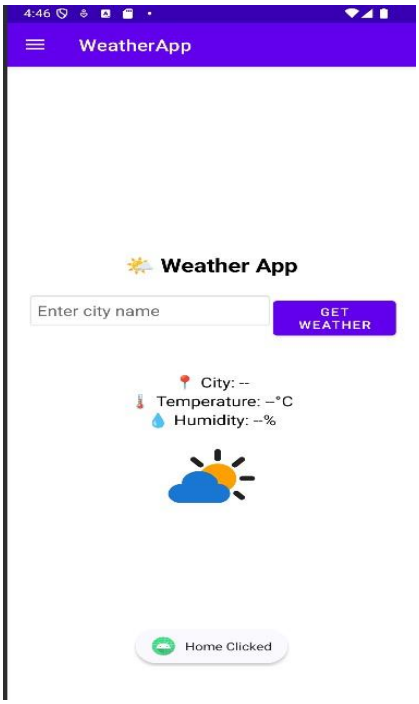
SignUp - Creating account in app



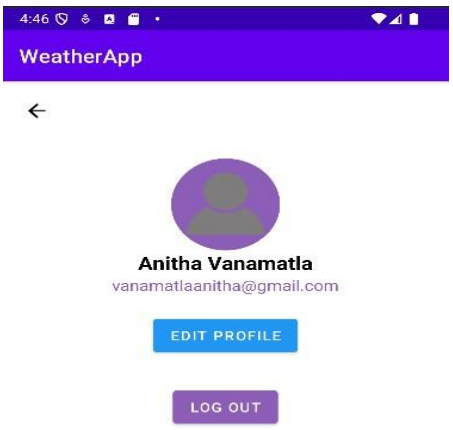
SignIn – Login into App

Home page after sign in

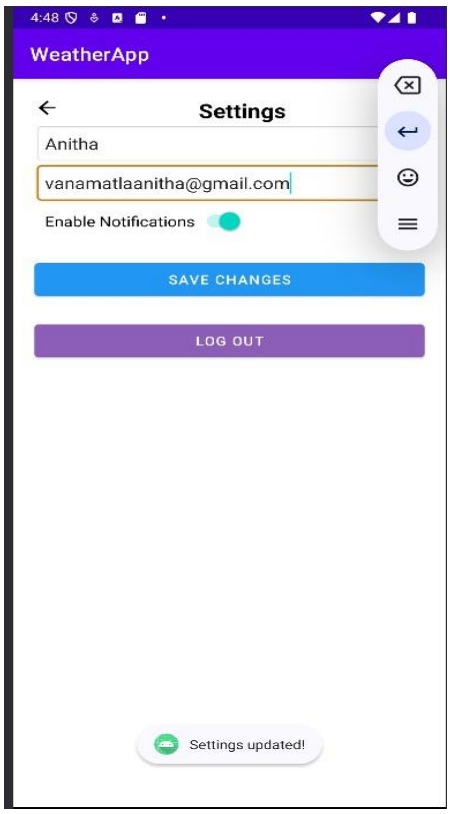
Navigation bar



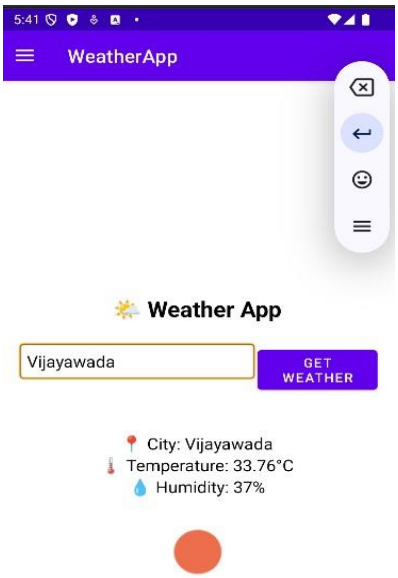
When i clicked on Home icon



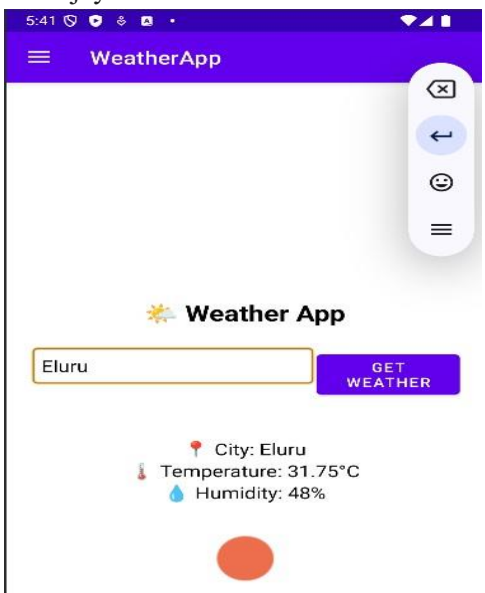
When I clicked on Profile icon



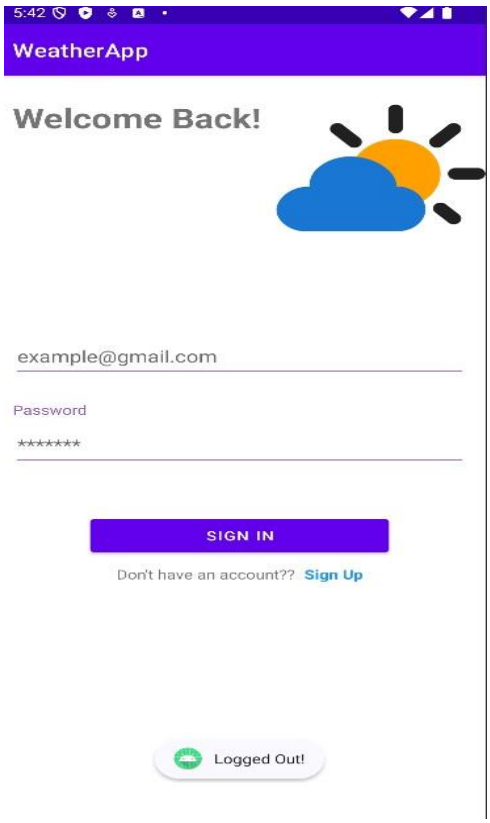
when I clicked on Settings icon



when I search the weather details for Vijayawada



When I enter Eluru City , it shows Eluru details



When I clicked on Logout Icon Weather

9.CONCLUSION & FUTURE ENHANCEMENTS

This project successfully implements real-time weather data retrieval for a specified city using the OpenWeatherMap API in an Android application. The app provides users with key weather information, including temperature, humidity, and a dynamically loaded weather icon.

By integrating Retrofit, the app ensures efficient and seamless API communication, fetching real-time data in a structured manner. The use of Glide enhances the user experience by dynamically loading weather condition icons based on API responses. Additionally, robust error handling ensures that users receive meaningful feedback, such as alerts when a city is not found or network issues occur.

The interface is designed to be intuitive, allowing users to simply enter a city name and retrieve weather updates instantly. The implementation follows best coding practices, ensuring modularity, readability, and maintainability. Key enhancements include improved UI feedback, accurate API response handling, and enhanced user interaction.

This project can be further expanded by adding forecast features, GPS-based location detection, and caching mechanisms for offline support. Overall, this application provides a solid foundation for weather-based services and showcases the effective use of modern Android development techniques.

10. REFERENCES (WEB SITE URLS)

GitHub repository links:
https://github.com/Anithavanamatla/weatherapp

1. Expo Docs - https://expo.dev

2. Mongoose Docs - https://mongoosejs.com

3. Eas Docs - https://docs.expo.dev/build/introduction

4. Android Studio Docs - https://developer.android.com/develop

5.API Key - https://openweathermap.org/api