AGRI ADVISOR

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

BY

T. Tej Mahendra	**(22501A05H9)**
Shaik Seema Jabeen	**(22501A05G6)**
P. Kamal Siddhardha	**(22501A05D5)**
Shaik Ruksana	**(22501A05F6)**

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 **“AGRI-ADVISOR”** is the bonafied work of **T. Tej Mahendra (22501A05H9), Shaik Seema Jabeen (22501A05G6), P. Kamal Siddhardha (22501A05D5), Shaik Ruksana (22501A05F6)** 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

Agri Advisor is an advanced mobile application designed to empower farmers by providing data-driven agricultural insights. Built using Java for Android and utilizing Firebase as the backend database, this app ensures seamless user experience and real-time data management. The system integrates user authentication, enabling farmers to register and log in securely. Once logged in, users can access weather updates based on their location, allowing them to plan farming activities effectively.

A key feature of Agri Advisor is the crop recommendation system, where farmers input soil and environmental attributes such as Nitrogen (N), Phosphorus (P), Potassium (K), humidity, temperature, and rainfall. This data is sent to a Flask API, which hosts a machine learning model trained to analyze the input and suggest the most suitable crop for cultivation. This feature enhances precision farming, ensuring that farmers maximize yield based on scientific analysis rather than guesswork.

Additionally, the application includes a crop information module, where users can browse, search, and view detailed descriptions of various crops. The information includes growth conditions, benefits, and farming techniques, aiding farmers in better crop management. The user interface is designed to be intuitive and easy to navigate, with a search bar for quick access to relevant crop data.

The app also features a user dashboard, where farmers can manage their personal details and track their interactions with the system. By integrating real-time data, machine learning, and cloud-based storage, Agri Advisor acts as a comprehensive digital assistant for farmers. It bridges the gap between technology and traditional farming, ensuring better resource utilization, improved yields, and sustainable agricultural practices.

2. INTRODUCTION

Agriculture is the backbone of many economies, providing food security and employment to millions. However, traditional farming methods often rely on experience rather than data-driven insights, leading to inefficiencies and lower yields. With advancements in technology and artificial intelligence, modern farming can now leverage machine learning, cloud computing, and real-time weather data to enhance productivity. Agri Advisor is a smart farming application designed to bridge the gap between technology and agriculture, empowering farmers with the tools they need to make informed decisions.

Agri Advisor is developed using Java for Android and Firebase as its database, ensuring real-time data storage and seamless user experience. The application features a secure user authentication system, allowing farmers to register and access personalized agricultural insights. A major highlight of the app is its crop recommendation system, where farmers input soil nutrients (N, P, K), temperature, humidity, and rainfall. The system then connects with a Flask API running a machine learning model, which analyzes the data and suggests the most suitable crop for cultivation. This feature optimizes resource utilization and enhances precision farming techniques.

Apart from recommendations, the app provides weather updates to help farmers plan their activities based on real-time climatic conditions. Additionally, a comprehensive crop information module enables users to explore different crops, their characteristics, growing conditions, and best farming practices. The searchable crop database ensures easy access to information, helping farmers make informed decisions.

The user dashboard acts as a centralized space where farmers can manage their profiles and track their interactions with the app. By combining AI-powered recommendations, cloud-based data storage, and intuitive user interface design, Agri Advisor aims to revolutionize modern farming, making agriculture smarter, more efficient, and sustainable.

3. OBJECTIVES AND SCOPE OF THE PROJECT

The primary objective of Agri Advisor is to develop an efficient and user-friendly Android-based agricultural assistance system that helps farmers make data-driven decisions. The platform aims to:

1. **Provide Data-Driven Crop Recommendations** – Implement a machine learning-based recommendation system that suggests the most suitable crop based on soil nutrients (N, P, K), temperature, humidity, and rainfall.

2. **Enhance Agricultural Decision-Making –** Equip farmers with real-time weather updates and detailed crop information to improve farm planning and productivity.

3. **Enable Seamless User Experience –** Develop an Android-based platform with Firebase integration for real-time data storage, user authentication, and secure access.

4. **Create an Interactive User Dashboard –** Provide farmers with a personalized dashboard where they can view weather conditions, crop recommendations, and their profile details.

5. **Implement Efficient Search and Filtering –** Allow users to easily search for crops, retrieve details, and filter relevant information for better accessibility.

Scope

The Agri Advisor application is designed to cater to farmers, and agricultural researchers who seek technology-driven insights for improving farming efficiency. The key areas of scope include:

- **User Registration & Authentication**: Secure login system allowing farmers to create profiles and access personalized agricultural data.

- **Crop Recommendation System**: Utilizing Flask API and machine learning models to suggest crops based on soil and climate conditions.

- **Weather Forecasting:** Integrating real-time weather updates to help farmers plan their agricultural activities.

- **Crop Information Database:** Providing a searchable and filterable database where users can learn about various crops, their characteristics, and best cultivation practices.

- **Scalability & Future Enhancements:** The application can be further expanded to include pest control guidance, irrigation planning, and AI-driven yield prediction for more agricultural support.

4. SOFTWARE USED – EXPLANATION

The **Agri Advisor** application is developed using a combination of **Android, Java, Firebase, Flask API, and various external APIs** to create a seamless and efficient agricultural advisory system.

Android

Android is an open-source mobile operating system developed by Google, designed primarily for touchscreen devices. It provides a robust platform for developing interactive, feature-rich mobile applications.

Usage in Agri Advisor

- The entire application is built on the Android platform, allowing farmers to access the system from their smartphones.
- Android provides a flexible UI framework using XML for designing layouts, ensuring a user-friendly experience.
- The app utilizes Android components such as Activities, Fragments, RecyclerView, SearchView, and ProgressBar to create an intuitive and responsive interface.

Java

Java is a high-level, object-oriented programming language widely used for Android app development due to its stability, security, and platform independence.

Usage in Agri Advisor

- Java is used to build the business logic of the app, handling user interactions, database connectivity, and API requests.
- It is used for implementing machine learning API calls, fetching weather data, and processing search/filtering operations.
- Java's exception handling ensures smooth performance even in case of network failures or invalid user inputs.

Firebase

Firebase is a cloud-based backend platform by Google that provides real-time database management, authentication, and cloud storage for mobile applications.

Usage in Agri Advisor

- Firebase Realtime Database is used to store and manage:
  - User details (registration, login authentication).

- Crop information (name, description, image URL).
- Search queries and recommendations.

- Firebase Authentication enables secure user login and registration, ensuring that only authenticated users can access personalized recommendations.
- Real-time synchronization allows instant updates, ensuring farmers always see the latest data without manually refreshing.

APIs (Application Programming Interfaces)

APIs allow applications to communicate with external services to fetch data dynamically.

Usage in Agri Advisor

- OpenWeather API: Fetches real-time weather data (temperature, humidity, rainfall) based on the user's location to help in agricultural planning.
- Firebase API: Used for storing and retrieving data efficiently in the cloud.
- Custom Flask API: Connects the Android app to the machine learning model for crop recommendation.

Development Tools

Code Editor: Android Studio is the official IDE for Android development, providing a powerful code editor, visual layout editor, and debugging tools. It supports Java and integrates seamlessly with Firebase, making it an ideal choice for developing feature-rich mobile applications. The IDE includes an emulator for testing applications across different devices and screen sizes, ensuring a smooth user experience. Android Studio also offers real-time error detection and intelligent code completion, helping developers write optimized and bug-free code.

Version Control System (VCS): GitHub is a widely used version control system that helps manage project files, track changes, and collaborate with team members. It enables developers to work on different features simultaneously using branches and ensures that all modifications are safely stored and documented. By utilizing GitHub, the project remains organized, with a history of code changes that can be reviewed and reverted if needed. This improves code stability and allows multiple developers to contribute efficiently.

5. PROPOSED MODEL

1. Overview

The proposed Crop Recommendation System is a mobile-based application designed to assist farmers in making data-driven decisions regarding crop selection. The system utilizes user input parameters such as nitrogen, phosphorus, potassium levels, temperature, humidity, and rainfall to predict the most suitable crop. It also integrates real-time weather updates and provides a database of crop details.

2. System Components

User Authentication and Profile Management

Users can register and log in to access the system. The profile stores essential details such as name, email, contact number, farming area, and land size, allowing personalized recommendations.

Weather Monitoring Module

The system fetches live weather updates based on the user's location, providing real-time data on temperature, humidity, and wind speed. This helps farmers plan their agricultural activities accordingly.

Crop Recommendation Engine

The recommendation engine, powered by a machine learning model, processes soil and climate parameters provided by the user. The model is hosted using Flask API, which communicates with the mobile application to deliver personalized crop suggestions.

Crop Information Database

A dedicated section allows users to browse different crops, view descriptions, and search for specific crops. The Firebase database stores information such as crop names, images, and growth requirements.

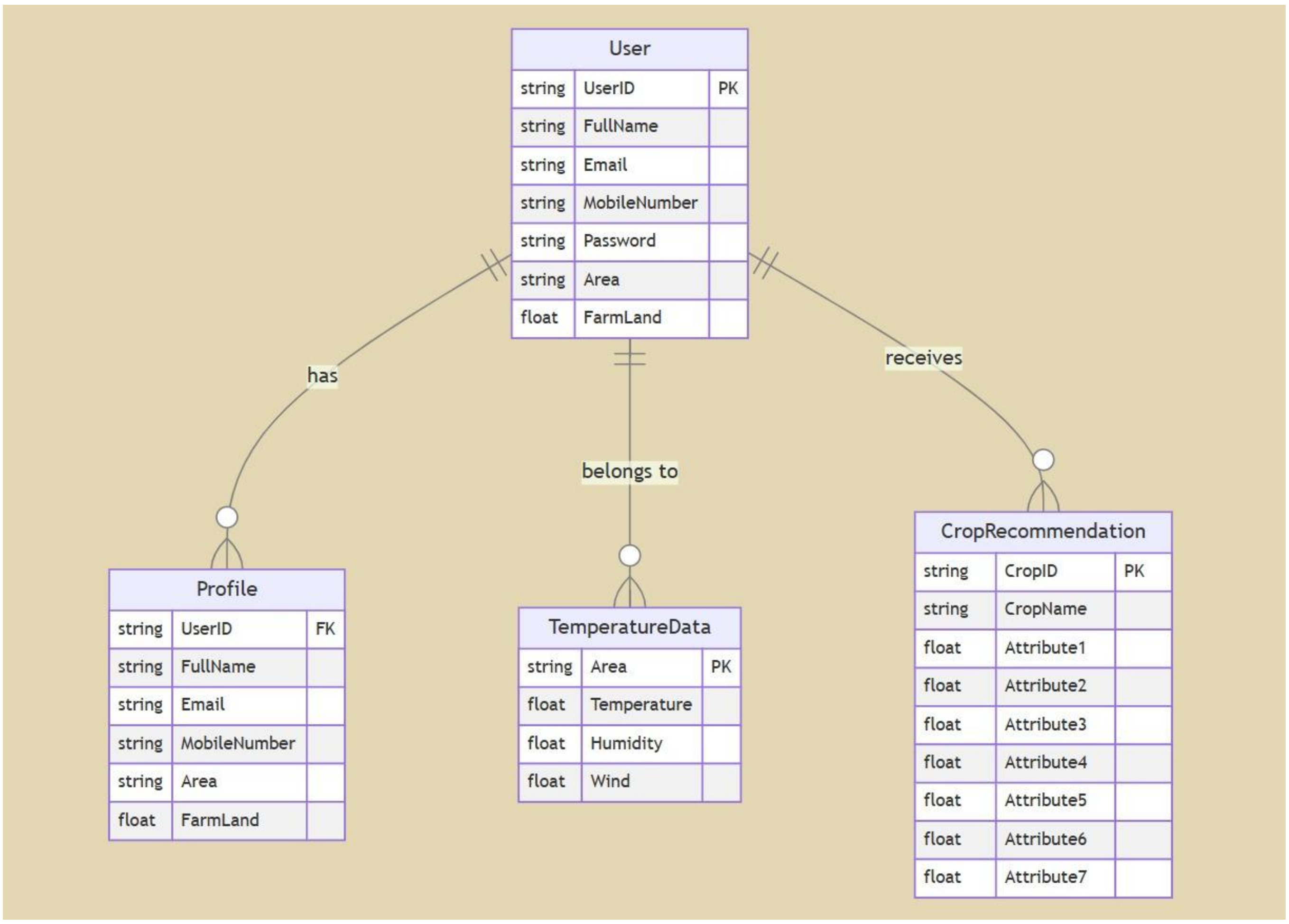
User Dashboard

The dashboard provides an overview of the user's details, previously recommended crops, and weather insights. It serves as a centralized hub for tracking past recommendations and managing user interactions.

3. Entity-Relationship Model

The database is structured with key entities:

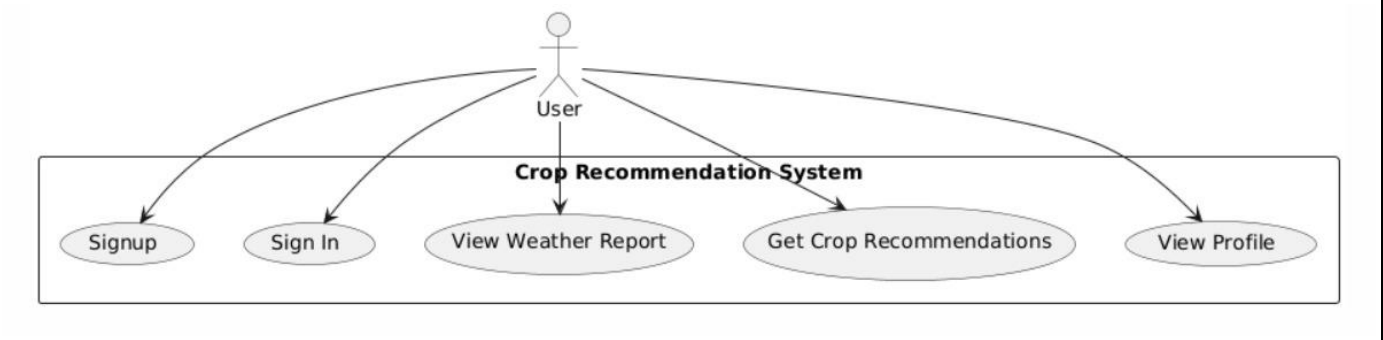
- User: Stores personal details and authentication credentials.
- Profile: Maintains user-specific information.
- TemperatureData: Logs weather conditions for different regions.
- CropRecommendation: Contains crop attributes required for the ML-based recommendation process.



4. Use Case Model

The system is designed to support key functionalities:

- Sign Up and Sign In for user authentication.
- View Weather Reports to access real-time environmental data.
- Get Crop Recommendations by submitting soil and climate attributes.
- View Profile to update personal details and track previous recommendations.



5. System Workflow

1. The user registers/logs in to the application.
2. The system retrieves weather updates based on the user's location.
3. The user enters soil and environmental attributes.
4. The ML model processes the data and suggests the best-suited crop.
5. The user can view detailed information on recommended crops.
6. All interactions are stored and accessible via the user dashboard.

6. Future Enhancements

The system can be expanded to include additional features such as yield prediction, automated irrigation suggestions, and pest detection using AI-based models. Integration with IoT devices for real-time soil monitoring could further enhance the precision of crop recommendations.

This proposed model outlines a robust, scalable, and user-friendly approach to improving agricultural decision-making using modern technology.

6. SAMPLE CODE

Github repository link: https://github.com/tej-mahender/Agri-Advisor

Folder Structure:

```
Agri-Advisor
+---app
|     +---main
|     |  |  AndroidManifest.xml
|     |  |
|     |  +---java
|     |  |  \---com
|     |  |      \---example
|     |  |          \---agri_advisor
|     |  |                  APIHandler.java
|     |  |                  CropDetailActivity.java
|     |  |                  CropDetailFragment.java
|     |  |                  CropFragment.java
|     |  |                  CropModel.java
|     |  |                  CropPagerAdapter.java
|     |  |                  DashboardActivity.java
|     |  |                  LoginActivity.java
|     |  |                  MyApp.java
|     |  |                  ProfileFragment.java
|     |  |                  RegisterActivity.java
|     |  |                  WeatherFragment.java
|     |  |                  YieldFragment.java
|     |  |
|     |  \---res
|     |      +---drawable
|     |      |      broken_clouds.png
|     |      |      clouds.png
|     |      |      crop.png
|     |      |      default_weather.png
|     |      |      error_image.png
|     |      |      ic_launcher_background.xml
|     |      |      ic_launcher_foreground.xml
|     |      |      img.png
|     |      |      img_1.png
|     |      |      img_2.png
|     |      |      input_bg.xml
|   |   |   mist.png
|   |   |   night_clear.png
|   |   |   placeholder_image.png
|   |   |   profile.png
|   |   |   rain_day.png
|   |   |   rain_night.png
|   |   |   scattered_clouds.png
|   |   |   shower_rain.png
|   |   |   snow.png
|   |   |   sunny.png
|   |   |   thunderstorm.png
|   |   |   weather.png
|   |   |   yield.png
|   |   |
|   |   +---layout
|   |   |   activity_crop_detail.xml
|   |   |   activity_dashboard.xml
|   |   |   activity_login.xml
|   |   |   activity_my_app.xml
|   |   |   activity_register.xml
|   |   |   fragment_crop.xml
|   |   |   fragment_crop_detail.xml
|   |   |   fragment_profile.xml
|   |   |   fragment_weather.xml
|   |   |   fragment_yield.xml
|   |   |   item_crop.xml
|   |   |
|   |   +---menu
|   |   |   bottom_nav_menu.xml
\---gradle
|  .| gitignore
|  .| build.gradle.kts
|  .| gradle.properties
|  .| gradlew
|  .| gradlew.bat
|  .| local.properties
|  .| settings.gradle.kts
|  .| libs.versions.toml
```

<u>LoginActivity.java</u>

```
package com.example.agri_advisor;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class LoginActivity extends AppCompatActivity {
    private FirebaseAuth mAuth;
    private EditText emailInput, passwordInput;
    private Button loginBtn, registerBtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        EdgeToEdge.enable(this);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.login), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
        mAuth = FirebaseAuth.getInstance();
        emailInput = findViewById(R.id.email);
        passwordInput = findViewById(R.id.password);
        loginBtn = findViewById(R.id.loginBtn);
        registerBtn = findViewById(R.id.registerBtn);

        loginBtn.setOnClickListener(v -> loginUser());
        registerBtn.setOnClickListener(v -> {
            startActivity(new Intent(LoginActivity.this, RegisterActivity.class));
        });
    }
        @Override
        protected void onStart() {
            super.onStart();
            if (mAuth.getCurrentUser() != null) {
                startActivity(new Intent(LoginActivity.this, DashboardActivity.class));
                finish();
            }
        }

    private void loginUser() {
        String email = emailInput.getText().toString().trim();
        String password = passwordInput.getText().toString().trim();

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

        mAuth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        FirebaseUser user = mAuth.getCurrentUser();
                        if (user != null) {
                                Toast.makeText(LoginActivity.this, "Login Successful",
Toast.LENGTH_SHORT).show();
                                startActivity(new Intent(LoginActivity.this, DashboardActivity.class));
                                finish();
                        }
                    } else {
                        Toast.makeText(LoginActivity.this, "Login Failed: " +
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                    }
                });
    }
}
```

DashboardActivity.java

```
package com.example.agri_advisor;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.fragment.app.Fragment;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.firebase.auth.FirebaseAuth;

public class DashboardActivity extends AppCompatActivity {
    private FirebaseAuth mAuth;
    private BottomNavigationView bottomNavigation;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dashboard);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_dashboard);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.dashboard), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });

        mAuth = FirebaseAuth.getInstance();
        bottomNavigation = findViewById(R.id.bottom_navigation);

        loadFragment(new WeatherFragment());

        bottomNavigation.setOnItemSelectedListener(item -> {
            Fragment selectedFragment = null;
            if (item.getItemId() == R.id.nav_weather) {
                selectedFragment = new WeatherFragment();
            } else if (item.getItemId() == R.id.nav_crop) {
                selectedFragment = new CropFragment();
            } else if (item.getItemId() == R.id.nav_yield) {
                selectedFragment = new YieldFragment();
            } else if (item.getItemId() == R.id.nav_profile) {
                selectedFragment = new ProfileFragment();
            }
            return loadFragment(selectedFragment);
        });
    }

    private boolean loadFragment(Fragment fragment) {
        if (fragment != null) {
            getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
fragment).commit();
```

```
            return true;
        }
        return false;
    }
}
```

CropFragment.java

```
package com.example.agri_advisor;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

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

public class CropFragment extends Fragment {
    private EditText inputN, inputP, inputK, inputTemp, inputHumidity, inputPh, inputRain;
    private TextView outputCrop;
    private Button btnPredict;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_crop, container, false);

        inputN = view.findViewById(R.id.input_nitrogen);
        inputP = view.findViewById(R.id.input_phosphorus);
        inputK = view.findViewById(R.id.input_potassium);
        inputTemp = view.findViewById(R.id.input_temperature);
        inputHumidity = view.findViewById(R.id.input_humidity);
        inputPh = view.findViewById(R.id.input_ph);
        inputRain = view.findViewById(R.id.input_rainfall);
        outputCrop = view.findViewById(R.id.output_crop);
        btnPredict = view.findViewById(R.id.btn_predict);
        btnPredict.setOnClickListener(view1 -> {
            try {
                Map<String, Object> cropData = new HashMap<>();
                cropData.put("N", Double.parseDouble(inputN.getText().toString()));
                cropData.put("P", Double.parseDouble(inputP.getText().toString()));
                cropData.put("K", Double.parseDouble(inputK.getText().toString()));
                cropData.put("temperature", Double.parseDouble(inputTemp.getText().toString()));
                cropData.put("humidity", Double.parseDouble(inputHumidity.getText().toString()));
                cropData.put("ph", Double.parseDouble(inputPh.getText().toString()));
                cropData.put("rainfall", Double.parseDouble(inputRain.getText().toString()));

                APIHandler.predictCrop(requireContext(), cropData, result -> outputCrop.setText(result));
            } catch (NumberFormatException e) {
                Toast.makeText(requireContext(), "Error: Enter valid numbers in all fields.",
Toast.LENGTH_SHORT).show();
            }
        });

        return view;
    }
}
```

YieldFragment.java

```
package com.example.agri_advisor;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.widget.SearchView;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import java.util.ArrayList;
import java.util.List;

public class YieldFragment extends Fragment {

    private RecyclerView recyclerView;
    private CropPagerAdapter adapter; // Correct Adapter
    private DatabaseReference databaseReference;
    private List<CropModel> cropList = new ArrayList<>();
    private SearchView searchView;
    private ProgressBar progressBar;
    private Button searchButton;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_yield, container, false);

        recyclerView = view.findViewById(R.id.recyclerView);
        searchView = view.findViewById(R.id.searchView);
        progressBar = view.findViewById(R.id.progressBar);
        searchButton = view.findViewById(R.id.searchButton);

        searchView.setIconifiedByDefault(false);
        searchView.clearFocus();

        recyclerView.setLayoutManager(new GridLayoutManager(getContext(), 2));
        adapter = new CropPagerAdapter(getContext(), cropList);
        recyclerView.setAdapter(adapter);

        databaseReference = FirebaseDatabase.getInstance().getReference("crops");

        loadCropsFromFirebase();

        searchButton.setOnClickListener(v -> filterCrops(searchView.getQuery().toString()));

        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                filterCrops(query);
                return false;
            }

            @Override
            public boolean onQueryTextChange(String newText) {
                filterCrops(newText);
                return false;
            }
        });

        return view;
    }

    private void loadCropsFromFirebase() {
        progressBar.setVisibility(View.VISIBLE);

        databaseReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                cropList.clear();
                for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                    CropModel crop = dataSnapshot.getValue(CropModel.class);
                    if (crop != null) {
                        Log.d("FirebaseData", "Loaded Crop: " + crop.getCropName());
                        cropList.add(crop);
                    }
                }
                adapter.updateList(cropList);
                progressBar.setVisibility(View.GONE);
            }
            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                progressBar.setVisibility(View.GONE);
                Toast.makeText(getContext(), "Failed to load crops: " + error.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
    }
    private void filterCrops(String query) {
        List<CropModel> filteredList = new ArrayList<>();
        for (CropModel crop : cropList) {
            if (crop.getCropName().toLowerCase().contains(query.toLowerCase())) {
                filteredList.add(crop);
            }
        }
        adapter.updateList(filteredList);
    }
}
```

7. RESULT/OUTPUT SCREEN SHOTS

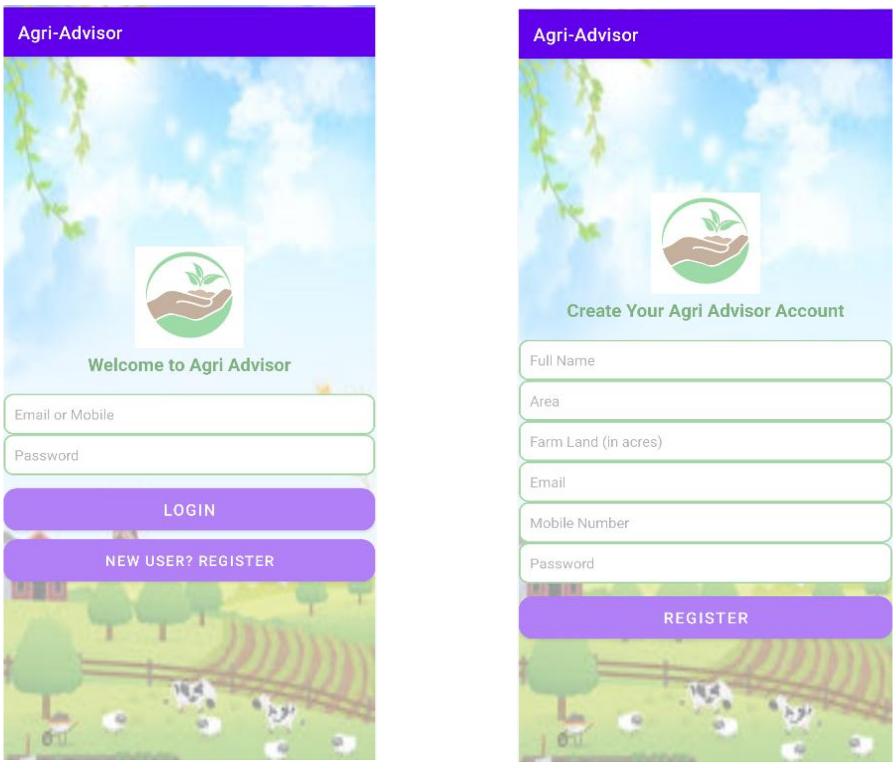
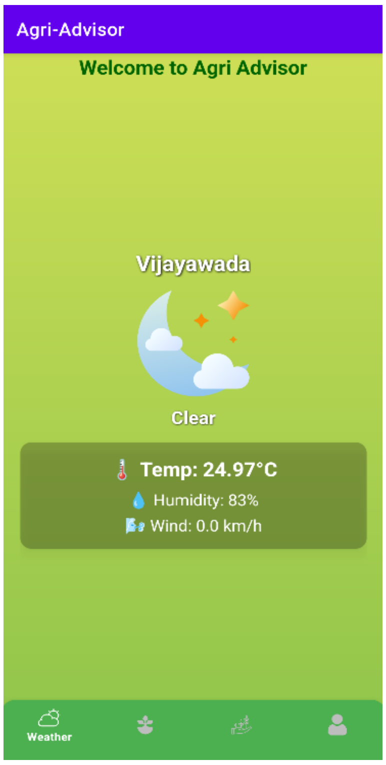


Figure 1: Sign-in and Sign-up page - The sign-in and sign-up pages of the Agri-Advisor app feature a bright farm-themed background with a blue sky and green fields. Both pages display a logo of a hand holding a plant, with input fields for login or registration, and prominent purple buttons for navigation.



Vijayawada

Clear

🌡 Temp: 24.97°C
💧 Humidity: 83%
🌬 Wind: 0.0 km/h

Figure 2: Weather details page - The weather details page of the Agri-Advisor app displays the current weather in Vijayawada, showing a clear night sky, temperature, humidity, and wind speed on a green gradient background.

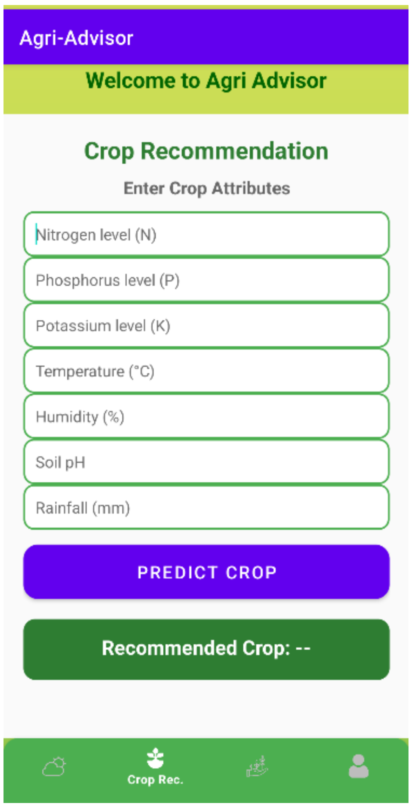


Figure 3: Crop Recommendation Page - The crop recommendation page allows users to input soil and environmental attributes like nitrogen, phosphorus, potassium, temperature, humidity, soil pH, and rainfall to predict the best-suited crop.

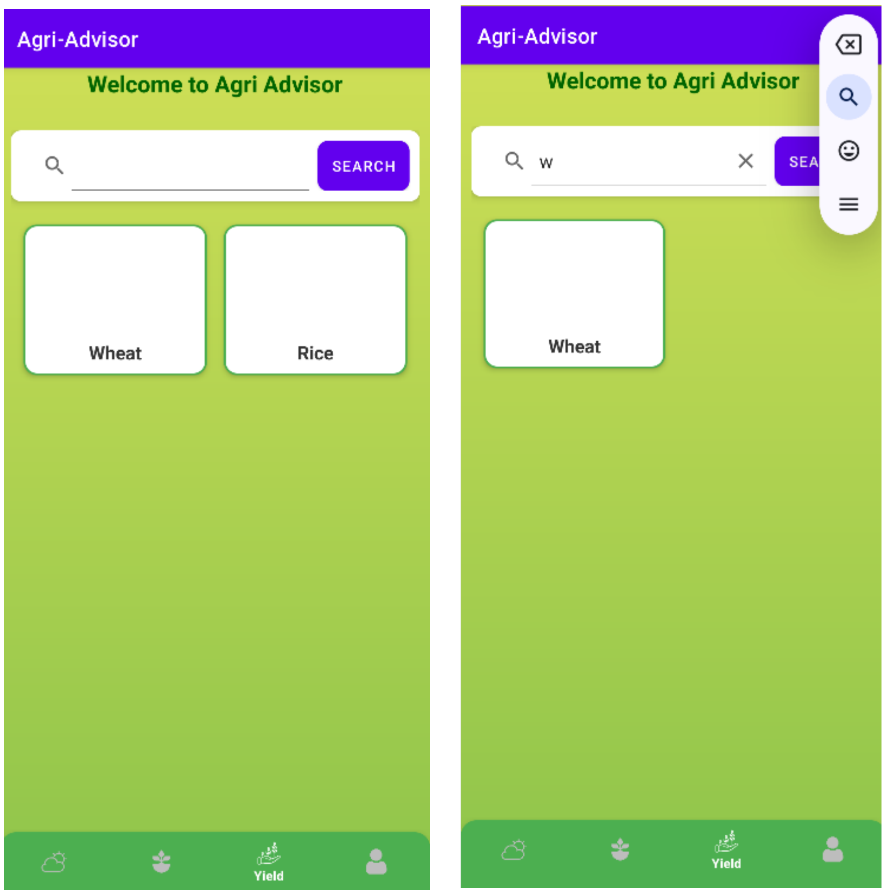
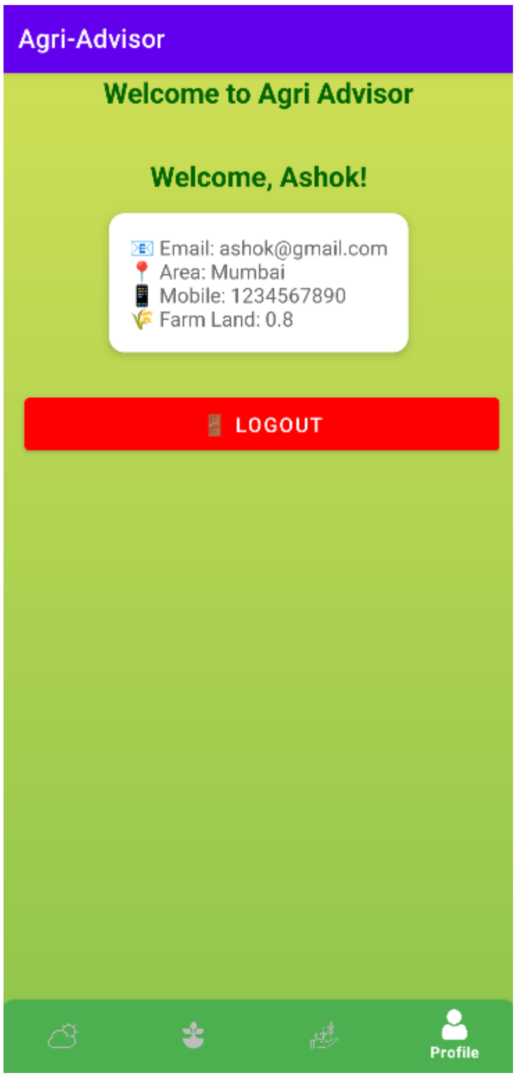


Figure 4: Crop details Page - A search interface that allows users to find crop details, with search results displayed as selectable cards.


Agri-Advisor

Welcome to Agri Advisor

Welcome, Ashok!

📧 Email: ashok@gmail.com
📍 Area: Mumbai
📱 Mobile: 1234567890
🌾 Farm Land: 0.8

Figure 5: Profile Page - This interface *di*splays user details including name, email, area, mobile number, and farm land size, with a red "Logout" button at the cente

8. CONCLUSION

Agri-Advisor is a comprehensive and user-friendly mobile application designed to assist farmers in making informed agricultural decisions. By integrating real-time weather updates, crop recommendations based on soil attributes, and a detailed crop information database, the system enhances agricultural productivity and efficiency. The application leverages Firebase for seamless data storage and retrieval, Flask API for machine learning-based crop prediction, and an intuitive Android interface for easy user interaction.

The platform provides a holistic solution for farmers, enabling them to access crucial weather data, receive personalized crop recommendations, and manage their profiles efficiently. With features like user authentication, dashboard access, and an integrated search function, Agri-Advisor ensures that farmers have all the necessary tools at their disposal for making data-driven decisions.

Looking ahead, Agri-Advisor has the potential for further advancements, such as integrating IoT-based soil monitoring, AI-driven pest detection, and automated irrigation recommendations. These enhancements would further empower farmers, making agriculture smarter and more sustainable. Overall, the project serves as a significant step towards digital transformation in farming, helping farmers optimize their resources, improve yield, and achieve better agricultural outcomes.

9. REFERENCES (WEB SITE URLS)

GitHub repository link: https://github.com/tej-mahender/Agri-Advisor

1. **Firebase Documentation: https://firebase.google.com/docs**
2. **Firebase Realtime Database: https://firebase.google.com/docs/database**
3. **Android Developer Guide: https://developer.android.com/guide**
4. **Android Studio: https://developer.android.com/studio**
5. **REST API with Flask: https://realpython.com/flask-restful-api/**