Abstract

The project is dedicated to developing an advanced agricultural monitoring system through the integration of a specialized robot equipped with various sensors and real-time weather data. In terms of software requirements, the system monitors water quality using sensors designed for agricultural applications, measuring parameters such as pH, electrical conductivity, dissolved oxygen, turbidity, and nutrients. For soil quality monitoring, sensors on the robot measure moisture, pH, and nutrient levels. Real-time weather data is also integrated to optimize resource use based on upcoming weather conditions. The agricultural robot gathers soil, water, and weather data, placing a significant emphasis on in-depth soil quality analysis. The project also involves monitoring crop health through the implementation of algorithm utilizes multispectral imagery processed with OpenCV such as plant stress, growth patterns, and overall vitality. These insights enable farmers to make informed decisions regarding crop management, contributing to improved agricultural practices and increased yields. This analysis informs adaptive measures for irrigation, nutrient delivery, offering actionable insights to farmers. Additionally, the system considers battery health ensuring sustained functionality and longevity. The iterative refinement of algorithms improves agricultural practices, enhancing efficiency and increasing yields. The hardware components, including ESP32, ESP32 CAM, solar panels, a robot chassis, motors, pumps, fans, electronic components, and a focus on battery health, collectively form a comprehensive solution for informed decision-making in agriculture.

Table of Contents

SR No	Content	Page No
1	Acknowledgment	4
2	Problem Definition	5-6
3	**Introduction**	7
3.1	Concept of Precision Agriculture	7-8
4	**Requirements**	
4.1	Software Requirements	9
4.2	Hardware Requirements	10-11
5	Timeline Chart	
6	**Literature Survey**	
6.1	*Flutter:* Introduction to Flutter Why use Flutter	12
6.2	*Firebase:* Introduction to Firebase Why to use Firebase	13
7	**Project Design**	
7.1	Use Case Diagram	
	Description	14
	Figure 1	15
7.2	Class Diagram	
	Description	16
	Figure 2,3,4,5	17-18
7.3	Sequence Diagram	
	Description	18-19
	Figure 6	20
7.4	Communication Diagram	
	Description	20-21
	Figure 7	21
7.5	Activity Diagram	
	Description	21-22
	Figure 8	22
7.6	State Chart Diagram	

	Description	22-23
	Figure 9	24
7.7	Component Diagram	
	Description	24-26
	Figure 10	26
7.8	Deployment Diagram	
	Description	26-28
	Figure 11	29
7.9	Flowchart	
	Description	29-30
	Figure 12	30
7.10	ER Diagram	
	Description	31-32
	Figure 13	32
8	Cocomo and Cocomo 2 Model	33-37
9	Features of the Project	38
10	Source Code	39 - 273
11	Snap Shots	274-276
12	Limitations of the Project	277
13	Applications of the Project	278-279
14	Future Scope of the Project	280
15	Conclusion	281
16	References	281

Acknowledgement

We are very thankful to our institute Shri Bhagubhai Mafatalal Polytechnic for approving our idea for the project in Mobile Application Development. WE had this topic in our mind as our final year project, as we have a great interest in application development.

We would also like to thank our project guide Mrs. Neeta Kadukar, to enlighten us with the knowledge of Mobile Application Development. The guidance was really helpful, information given to us helped in understanding the various concept. She has really encouraged us at each part of the development. Our guide has also provided the help in understanding and implementing some of the features in our project. Guide has also provided us with an immense knowledge for the document. We would like to give the credit to our project guide. We would be really thanking Mrs. Neeta Kadukar for each effort made to help us in our project.

We would really to thank our lab technician Mrs. Reshma Rane for providing us with all the needs to meet the technical requirements and to setup the technologies needed for our project. She was always there to solve the problem.

Problem Definition

Agricultural practices face challenges in implementing advanced monitoring systems, particularly integrating specialized robots with diverse sensors and real-time weather data, requiring precise sensor calibration and seamless data integration from various sources.

This project proposes a sophisticated agricultural monitoring system, integrating a robot, sensors, and real-time weather data, to optimize resource utilization, enhance crop management, and increase yields.

The solution offers real-time monitoring of water quality, soil parameters, and weather conditions, enabling farmers to make informed decisions for efficient resource utilization. It ensures precise sensor calibration and seamless integration of data from various sources.

The project prioritizes data accuracy, adaptability, and sustainability of hardware components, focusing on battery health and system longevity. It aims to enhance agricultural monitoring, improve crop yields, promote sustainable farming practices, and boost farm productivity.

Objective:

- **Optimizing Resource Utilization:** The primary objective is to develop an agricultural monitoring system that optimizes resource utilization by integrating specialized robots, sensors, and real-time weather data. This includes efficiently managing water usage, soil nutrients, and other inputs to enhance crop growth while minimizing waste.
- **Enhancing Crop Management**: Another objective is to improve crop management practices through real-time monitoring of key parameters such as water quality, soil conditions, and weather forecasts. This will enable farmers to make timely decisions regarding irrigation, fertilization, and pest control, leading to healthier crops and higher yields.
- **Increasing Yields:** The project aims to increase agricultural yields by providing farmers with accurate data and insights to optimize growing conditions. By leveraging advanced monitoring systems, farmers can identify and address issues such as nutrient deficiencies, water stress, or pest infestations early, thereby maximizing the productivity of their fields.

- **Ensuring Data Accuracy and Integration**: A crucial objective is to ensure the accuracy of sensor data and seamless integration of information from diverse sources. This involves precise sensor calibration, robust data processing algorithms, and compatibility with various data formats and communication protocols to provide reliable and actionable insights to farmers.
- **Promoting Sustainable Farming Practices**: The project seeks to promote sustainable farming practices by facilitating efficient resource management and reducing environmental impact. By monitoring soil health, water usage, and weather patterns in real-time, farmers can implement conservation strategies, minimize runoff, and reduce reliance on chemical inputs, thus contributing to long-term sustainability.
- **Boosting Farm Productivity**: Ultimately, the overarching objective is to boost farm productivity and profitability. By equipping farmers with advanced monitoring tools and decision support systems, the project aims to streamline operations, minimize risks, and maximize returns on investment, thereby ensuring the economic viability of agricultural enterprises.

Introduction

Precision agriculture (PA) is an approach to farming that utilizes technology and data to optimize various aspects of agricultural production. It involves the integration of advanced tools such as GPS, sensors, drones, robotics, and data analytics to gather, analyze, and act upon information about field variability, crop conditions, and environmental factors. The goal of precision agriculture is to make farming practices more efficient, sustainable, and profitable by precisely tailoring inputs and management practices to the specific needs of individual plants, sections of fields, or entire farming operations.

Key components and practices associated with precision agriculture:

1. **Remote Sensing**: Remote sensing technologies, such as satellite imagery, aerial drones, and ground-based sensors, are used to collect data on various parameters such as soil moisture, crop health, and weed infestation. These technologies provide detailed information about the spatial variability within fields, allowing farmers to make targeted management decisions.
2. **Global Positioning System (GPS):** GPS technology enables accurate mapping and geolocation of field boundaries, soil properties, and crop conditions. GPS-guided machinery, such as tractors and sprayers, can precisely navigate fields, apply inputs, and perform tasks according to predefined parameters, minimizing overlap and waste.
3. **Variable Rate Technology (VRT):** VRT allows farmers to apply inputs, such as fertilizers, pesticides, and irrigation water, at variable rates based on real-time data and field conditions. By adjusting application rates according to the specific needs of different areas within a field, VRT helps optimize resource utilization and improve crop yields while reducing environmental impact.
4. **Data Analytics and Decision Support Systems**: Advanced data analytics and decision support systems analyze large volumes of data collected from various sources to provide actionable insights and recommendations to farmers. These systems help farmers identify trends, patterns, and anomalies in crop performance, predict future outcomes, and make informed decisions about crop management practices.
5. **Automation and Robotics**: Automation technologies, including robotic systems and autonomous vehicles, are increasingly being used in precision agriculture to perform tasks such as planting, spraying, and harvesting with

precision and efficiency. These technologies reduce labor requirements, improve accuracy, and enable round-the-clock operations, particularly in large-scale farming operations.

6. **Environmental Monitoring and Sustainability:** Precision agriculture promotes environmental sustainability by minimizing the use of inputs, reducing soil erosion, conserving water resources, and minimizing the environmental impact of farming practices. By precisely targeting inputs and minimizing waste, precision agriculture helps mitigate the negative effects of agriculture on ecosystems and natural resources.

Requirements

Software Requirements:

1. **Flutter**:

- Flutter is used to develop the mobile application interface that interacts with the agricultural robot.
- It facilitates the creation of a user-friendly and visually appealing interface for farmers to access data, reports, and actionable insights.

2. **Firebase:**

- It will be utilized for various purposes, such as real-time data synchronization, user authentication, and cloud storage.
- Firebase can be employed to store and manage the collected soil, water, and weather data in the cloud for comprehensive analysis and historical tracking.

3. **Android Studio:**

- Android Studio is the integrated development environment used for Flutter app development. It provides tools for designing, coding, and testing the mobile application.

4. **Machine Learning:**

Machine learning is a crucial component for several aspects of the agricultural robot's functionality:

- Linear Regression: Used for trend analysis in water quality data over time.
- Isolation Forest or One-Class SVM: Applied for anomaly detection in water quality parameters.
- NDVI Calculation: Utilizes machine learning concepts for interpreting multispectral imagery and assessing crop health.

5. Cloud:

- Utilized for transmitting and storing collected soil, water, and weather data in a centralized system or cloud platform for comprehensive analysis.

Hardware Requirements:

1. ESP32: The ESP32 is a microcontroller that is likely used for controlling various aspects of the robot's functionality. It can handle sensor inputs, communicate with other components, and execute control algorithms.

2. ESP32 CAM: The ESP32 CAM is likely used for capturing images or videos, especially for tasks like soil and crop health monitoring. It may be integrated with other sensors for data collection.

3. SOLAR PANELS/S 10W: Solar panels are used to harness solar energy, providing a sustainable power source for the agricultural robot. The 10W rating indicates the power capacity of the solar panels.

4. ROBOT CHASIS / CAR CHASIS: The robot chassis or car chassis is the physical structure that holds and integrates all the electronic components. It provides a platform for mounting sensors, motors, and other hardware.

5. BO-GEARED MOTORS: Geared motors are likely used for driving the wheels of the robot. The gearbox helps control the speed and torque of the motors.

6. WHEELS: Wheels are essential for the robot's mobility. They are attached to the geared motors for movement.

7. BLOWER PUMP: The blower pump might be used for tasks related to irrigation or distributing substances in the agricultural field.

8. AIR BLOWER FAN: The air blower fan may be employed for various purposes, such as cooling components or creating airflow in specific areas.

9. SERVO MOTOR SG-90: The SG-90 servo motor is a small motor often used for precise control of mechanical components. It might be used for specific movements or adjustments in the robot.

10. LI-ION BATTERY PACK: The Li-ion battery pack serves as the power source for the robot, providing energy to all the electronic components.

11. TP5 100 CHARGING MODULE: The charging module is likely used for charging the Li-ion battery pack efficiently.

12. H-BRIDGE MOTOR CONTROLLER: H-Bridge motor controllers are essential for controlling the direction and speed of the motors.

13. GEAR MECHANISM: The gear mechanism may be used for specific functionalities, such as adjusting the movement or operation of certain components.

14. XL6009 BUCK BOOST CONVERTER: The buck-boost converter is likely used for regulating and maintaining a stable power supply to various components.

15. GSM MODULE: The GSM module enables communication over cellular networks, allowing the robot to transmit data or receive instructions remotely.

16. VEROBOARD: Veroboards are prototyping boards used for soldering and connecting electronic components during the development phase.

17. VARIOUS ELECTRONIC COMPONENTS: This category includes miscellaneous electronic components such as resistors, capacitors, sensors, etc., needed for the assembly and functioning of the robot.

Literature Survey

Flutter

Introduction to Flutter

Flutter is an open-source UI software development kit (SDK) developed by Google. It provides developers with a comprehensive framework for building high-quality, visually appealing applications for multiple platforms, including Android, iOS, web, and desktop. With Flutter, developers can write code once and deploy it across different platforms, saving time and effort. The hot reload feature allows for instant code changes, enabling rapid iteration and efficient development.

One of the key advantages of Flutter is its rich set of customizable widgets, which enable developers to create stunning user interfaces that adhere to the platform's design guidelines. Flutter's rendering engine ensures smooth and performant animations and transitions, delivering a native-like experience to users. Additionally, Flutter seamlessly integrates with platform-specific features and APIs, giving developers access to device capabilities such as camera, location, and sensors.

Flutter offers a powerful and efficient solution for cross-platform app development, combining the benefits of code reusability, fast development cycles, and excellent performance. With its extensive widget library, seamless integration with device features, and a vibrant community, Flutter empowers developers to create beautiful, feature-rich applications for a wide range of platforms.

Why Flutter?

Flutter is a cross-platform development tool that enables developers to write code once and deploy it on multiple platforms like iOS, Android, and web. It offers high-performance rendering, rich UI components, and a hot reload feature for real-time data visualization. Flutter's extensive widget library allows for visually appealing interfaces for agricultural monitoring systems.

Firebase

Introduction to Firebase

Firebase is a cutting-edge cloud-based platform that offers a wide range of services to support our project's development and backend needs. We have chosen Firebase due to its robust and scalable nature, which aligns perfectly with our goal of delivering a reliable and high-performing application. With Firebase, we can leverage its comprehensive suite of features, including user authentication, real-time database, cloud storage, cloud messaging, and hosting.

The user authentication feature provided by Firebase ensures secure and seamless authentication for our application. It offers a variety of authentication methods, such as email/password, social media logins, and phone number verification. This allows us to implement a secure and user-friendly login system while maintaining data privacy and access control.

Firebase's real-time database is another vital component for our project. It enables synchronized data updates across multiple devices in real-time, ensuring that users always have the latest information at their fingertips. This real-time functionality is crucial for features such as live chat, real-time collaboration, and dynamic content updates. Furthermore, Firebase's cloud storage feature provides a reliable and scalable solution for storing and retrieving user-generated content, such as images, videos, and files. This ensures that our application can efficiently handle large amounts of media files while maintaining optimal performance.

Why Firebase?

Firebase is a NoSQL database ideal for agricultural monitoring systems, offering real-time data storage, robust authentication, Cloud Functions, Hosting, and Cloud Storage solutions. Its scalability and reliability are built on Google's infrastructure, ensuring large data volumes are handled efficiently.

Project Design

1. Use Case Diagram:

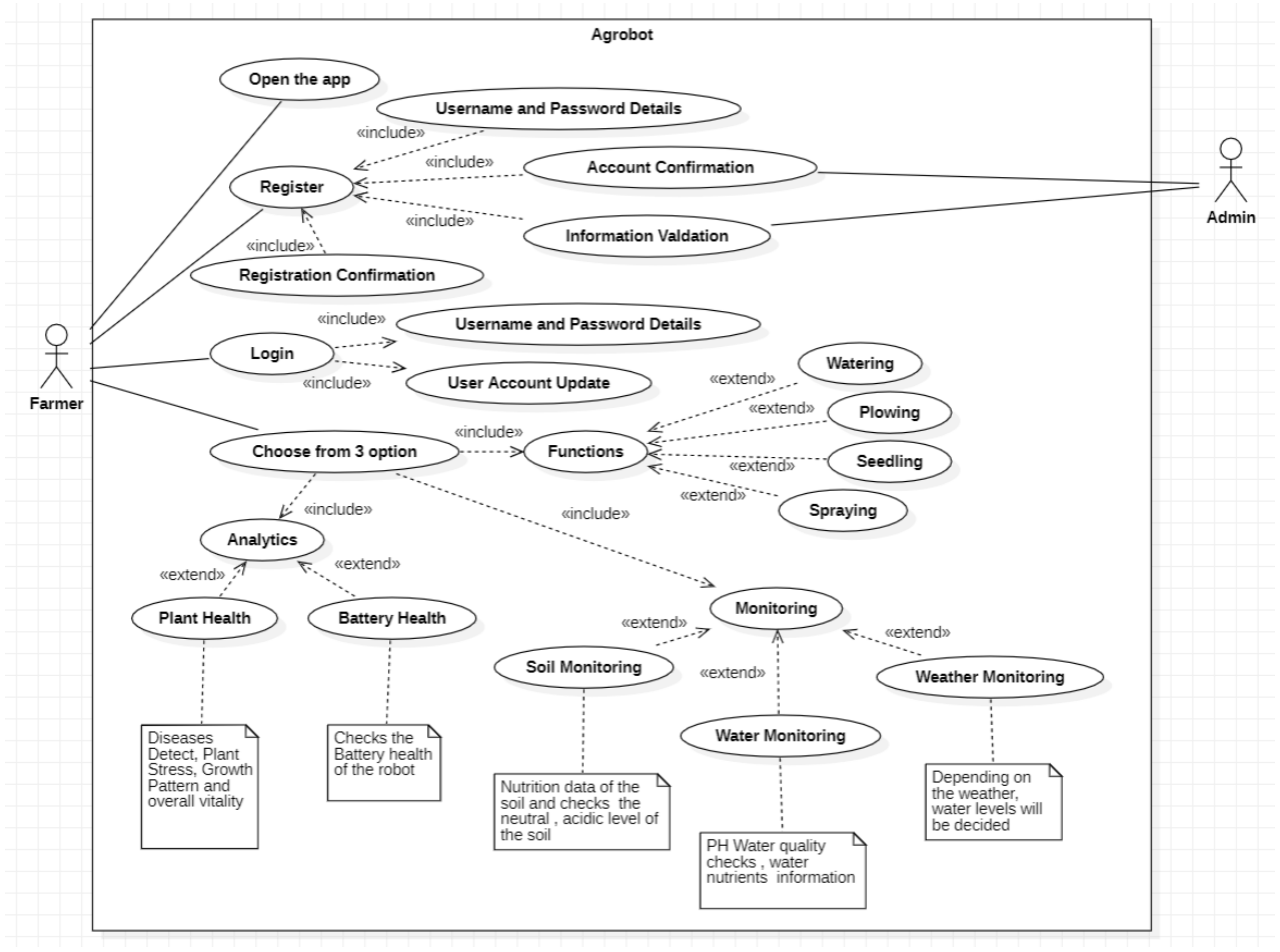


Figure 1

- **App**:
  - Represents the agricultural monitoring application.
  - Responsibilities include initiating data collection, displaying data to the Farmer, and managing user interactions.
- **Farmer**:
  - Represents the user (actor) who interacts with the application.
  - Initiates data collection, views data analysis results, and makes decisions based on insights.
- **Robot**:

    - Represents the specialized agricultural robot equipped with sensors for data collection.
    - Responsibilities include collecting soil, water, and weather data, performing soil quality and crop health analysis, and sending data to the App.
- **Agricultural Researcher:**
    - Represents another user (actor) who accesses the system for research purposes.
    - Responsibilities include requesting access to data, analyzing data for research, and performing research activities.
- **Admin**:
    - Represents the system administrator responsible for managing user access and system maintenance.
    - Responsibilities include approving access requests from the Agricultural Researcher and performing system maintenance tasks.
- **Data**:
    - Represents the collected agricultural data, including soil, water, weather, and crop health information.
    - Attributes may include parameters such as pH, moisture, temperature, etc.
- **DataAnalysis**:
    - Represents the analysis performed on the collected data.
    - Responsibilities include analyzing soil quality, crop health, and generating insights for decision-making.
- **UserInterface**:
    - Represents the interface elements used for user interaction within the application.
    - Responsibilities include presenting data to users and facilitating user input.
- **SystemMaintenance**:
    - Represents the component responsible for system maintenance activities.
    - Responsibilities include monitoring system performance and performing maintenance tasks as needed.

2. Class Diagram:

➢ **Actors**:

- **Farmer**:
  - Represents the user who interacts with the agricultural monitoring application to manage and monitor agricultural activities.
- **Agricultural Researcher**:
  - Represents another user who accesses the system for research purposes, such as analyzing historical data and performing research activities.
- **Admin**:
  - Represents the system administrator responsible for managing user access and performing system maintenance tasks.

➢ **Use Cases**:

- **Open App**:
  - Initiates the use of the agricultural monitoring application by the Farmer.
- **Initiate Data Collection**:
  - Farmer initiates the process of collecting soil, water, and weather data through the application.
- **View Data Analysis:**
  - Farmer views the analysis results of soil quality, weather conditions, and crop health displayed by the application.
- **Make Decisions:**
  - Farmer makes decisions regarding crop management, irrigation, and nutrient delivery based on the displayed data.
- **Request Research Access:**
  - Agricultural Researcher requests access to historical data and specific parameters for research purposes.
- **Approve Research Access:**
  - Admin approves the Agricultural Researcher's request for access to data.
- **Perform Research Analysis:**
  - Agricultural Researcher performs analysis on the data accessed for research purposes.
- **Monitor System Performance:**
  - Admin monitors system performance and performs maintenance tasks as needed.

- **Relationships**:
  - **Association**:
    - Each Use Case is associated with at least one actor, representing the actor's involvement in that particular use case.
  - **Include**:
    - Relationships between use cases to represent the inclusion of one use case within another. For example, "View Data Analysis" and "Make Decisions" may include "Open App" and "Initiate Data Collection."
  - **Extend**:
    - Relationships to represent optional or alternative behavior. For example, "Perform Research Analysis" may extend "View Data Analysis" to represent additional functionalities required for research purposes.

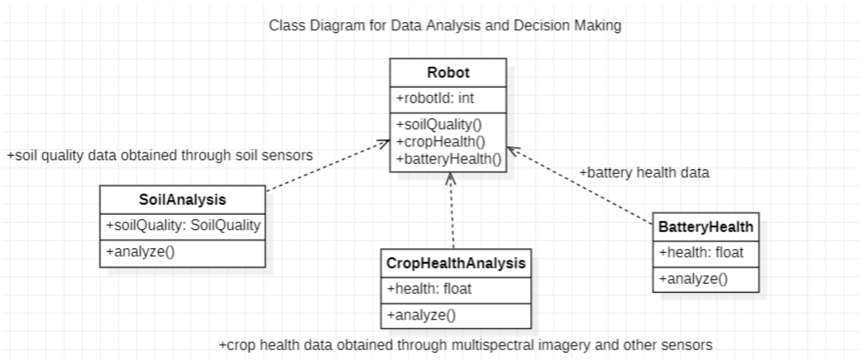

Class Diagram for Data Analysis and Decision Making
Robot
+robotId: int
+soilQuality()
+cropHealth()
+batteryHealth()
+soil quality data obtained through soil sensors
+battery health data
SoilAnalysis
+soilQuality: SoilQuality
+analyze()
CropHealthAnalysis
+health: float
+analyze()
BatteryHealth
+health: float
+analyze()
+crop health data obtained through multispectral imagery and other sensors

Figure 2

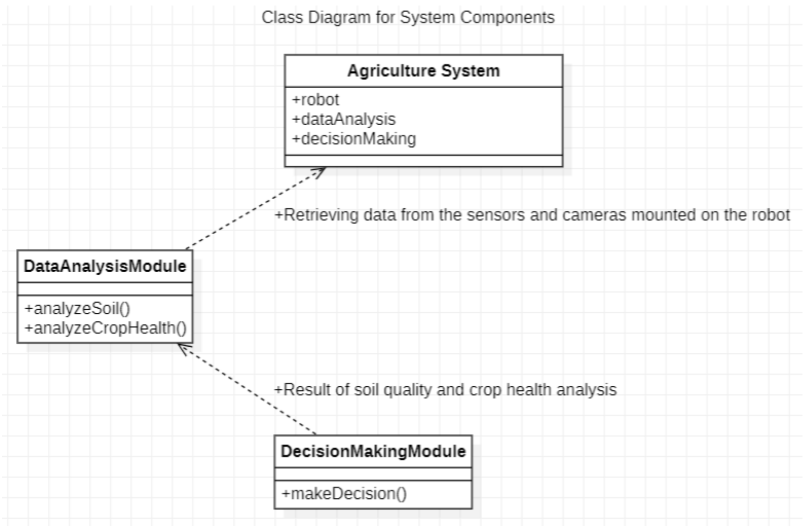

Class Diagram for System Components
Agriculture System
+robot
+dataAnalysis
+decisionMaking
+Retrieving data from the sensors and cameras mounted on the robot
DataAnalysisModule
+analyzeSoil()
+analyzeCropHealth()
+Result of soil quality and crop health analysis
DecisionMakingModule
+makeDecision()

Figure 3

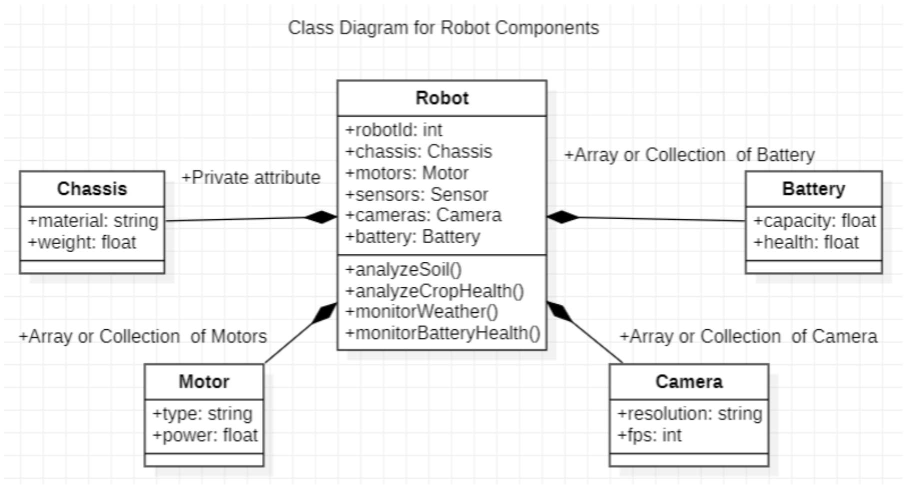


Figure 4

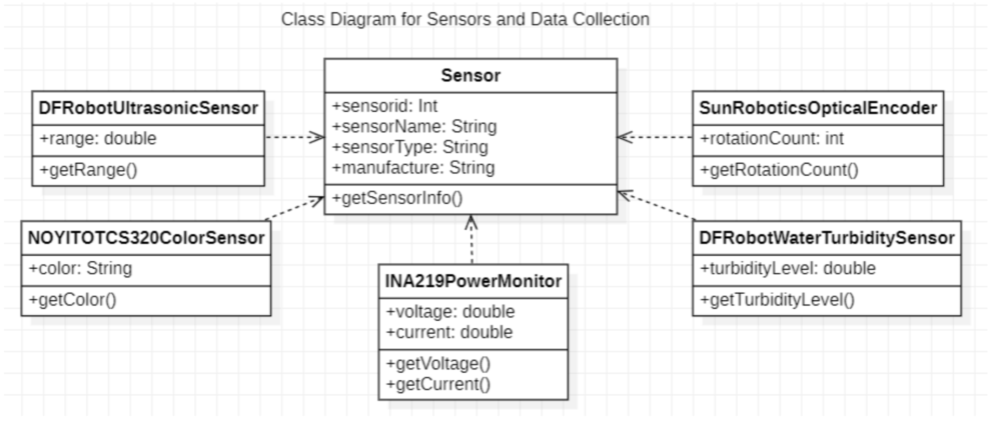


Figure 5

3. Sequence Diagram

- **Opening the App:**
  - The Farmer opens the agricultural monitoring application.
- **Data Collection Initiation:**
  - The Farmer initiates the data collection process through the app.
- **Requesting Data Collection:**
  - The App sends a request to the Robot to start data collection.
- **Data Collection by the Robot:**
  - The Robot begins collecting soil, water, and weather data.
  - It uses its sensors to measure parameters like pH, moisture, temperature, etc.
  - It also retrieves real-time weather data.
- **Analyzing Soil Quality:**
  - The Robot performs in-depth soil quality analysis.

    - It analyzes soil moisture, pH, and nutrient levels.
- **Analyzing Crop Health:**
    - The Robot processes multispectral imagery to analyze crop health.
    - It detects plant stress, growth patterns, and overall vitality.
- **Sending Data to the App:**
    - The Robot sends the collected data (soil, water, weather, and crop health) to the App.
- **Data Display and Analysis:**
    - The App displays the collected data to the Farmer.
    - The Farmer views insights regarding soil quality, weather conditions, and crop health.
- **Decision Making by the Farmer:**
    - Based on the displayed data, the Farmer makes decisions regarding crop management, irrigation, and nutrient delivery.
- **Researcher's Access:**
    - The Agricultural Researcher accesses the system for research purposes.
- **Research Data Request**:
    - The Researcher requests access to historical data and specific parameters for analysis.
- **Admin Approval:**
    - The Admin grants access to the requested data.
- **Access Granted:**
    - The Researcher receives access to the requested data.
- **Research Analysis:**
    - The Researcher analyzes the data for research purposes.
- **System Maintenance:**
    - The Admin monitors system performance and performs maintenance tasks as needed.

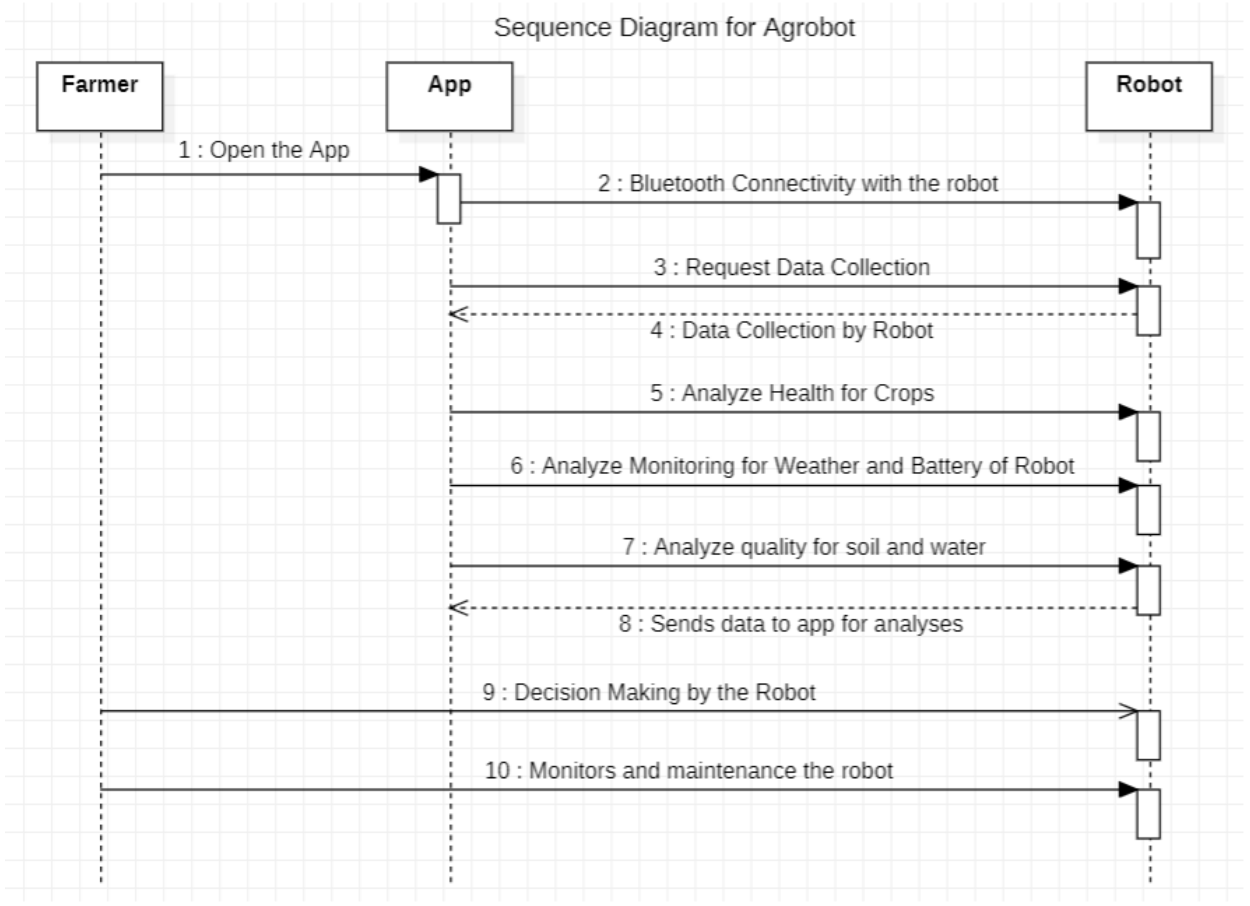

Sequence Diagram for Agrobot
Farmer
App
Robot
1 : Open the App
2 : Bluetooth Connectivity with the robot
3 : Request Data Collection
4 : Data Collection by Robot
5 : Analyze Health for Crops
6 : Analyze Monitoring for Weather and Battery of Robot
7 : Analyze quality for soil and water
8 : Sends data to app for analyses
9 : Decision Making by the Robot
10 : Monitors and maintenance the robot

Figure 6

4. Communication Diagram
   1) **Components**:
      - **Farmer**: Represents the user who interacts with the agricultural monitoring application.
      - **App**: Represents the agricultural monitoring application.
      - **Robot**: Represents the specialized agricultural robot equipped with sensors for data collection.
   2) **Communication Flows:**
      - **Farmer to App:**
        - The Farmer opens the application to initiate data collection and view analysis results.
      - **App to Robot:**
        - The App sends a request to the Robot to start data collection.
      - **Robot to App:**

- The Robot collects soil, water, and weather data and performs analysis.
- The Robot sends the collected data to the App.

- **App to Farmer:**
  - The App displays the collected data and analysis results to the Farmer.
  - The Farmer views the data and makes decisions based on the insights provided.

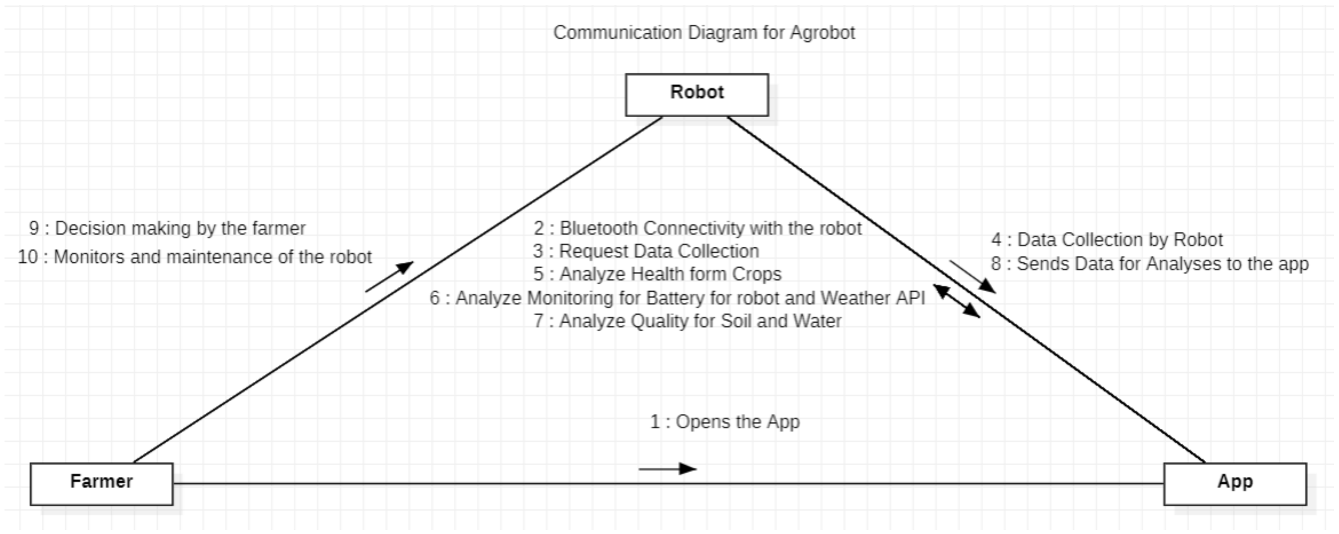

Communication Diagram for Agrobot
Robot
9 : Decision making by the farmer
10 : Monitors and maintenance of the robot
2 : Bluetooth Connectivity with the robot
3 : Request Data Collection
5 : Analyze Health form Crops
6 : Analyze Monitoring for Battery for robot and Weather API
7 : Analyze Quality for Soil and Water
4 : Data Collection by Robot
8 : Sends Data for Analyses to the app
1 : Opens the App
Farmer
App

Figure 7

5. Activity Diagram

- Opening the App:
  - The Farmer initiates the process by opening the agricultural monitoring application.
- Initiating Data Collection:
  - The Farmer triggers the process of data collection through the application.
- Data Collection by the Robot:
  - The Robot collects soil, water, and weather data.
  - It measures parameters like pH, moisture, temperature, etc.
- Analyzing Soil Quality:
  - The Robot performs soil quality analysis.
  - It analyzes soil moisture, pH, and nutrient levels.
- Analyzing Crop Health:
  - The Robot processes multispectral imagery to analyze crop health.

- It detects plant stress, growth patterns, and overall vitality.

- Displaying Data to the Farmer:
  - The App displays the collected data to the Farmer.
  - The Farmer views insights regarding soil quality, weather conditions, and crop health.
- Decision Making by the Farmer:
  - Based on the displayed data, the Farmer makes decisions regarding crop management, irrigation, and nutrient delivery.
- Research Analysis:
  - The Agricultural Researcher performs analysis on the data accessed for research purposes.
- System Maintenance:
  - The Admin monitors system performance and performs maintenance tasks as needed.

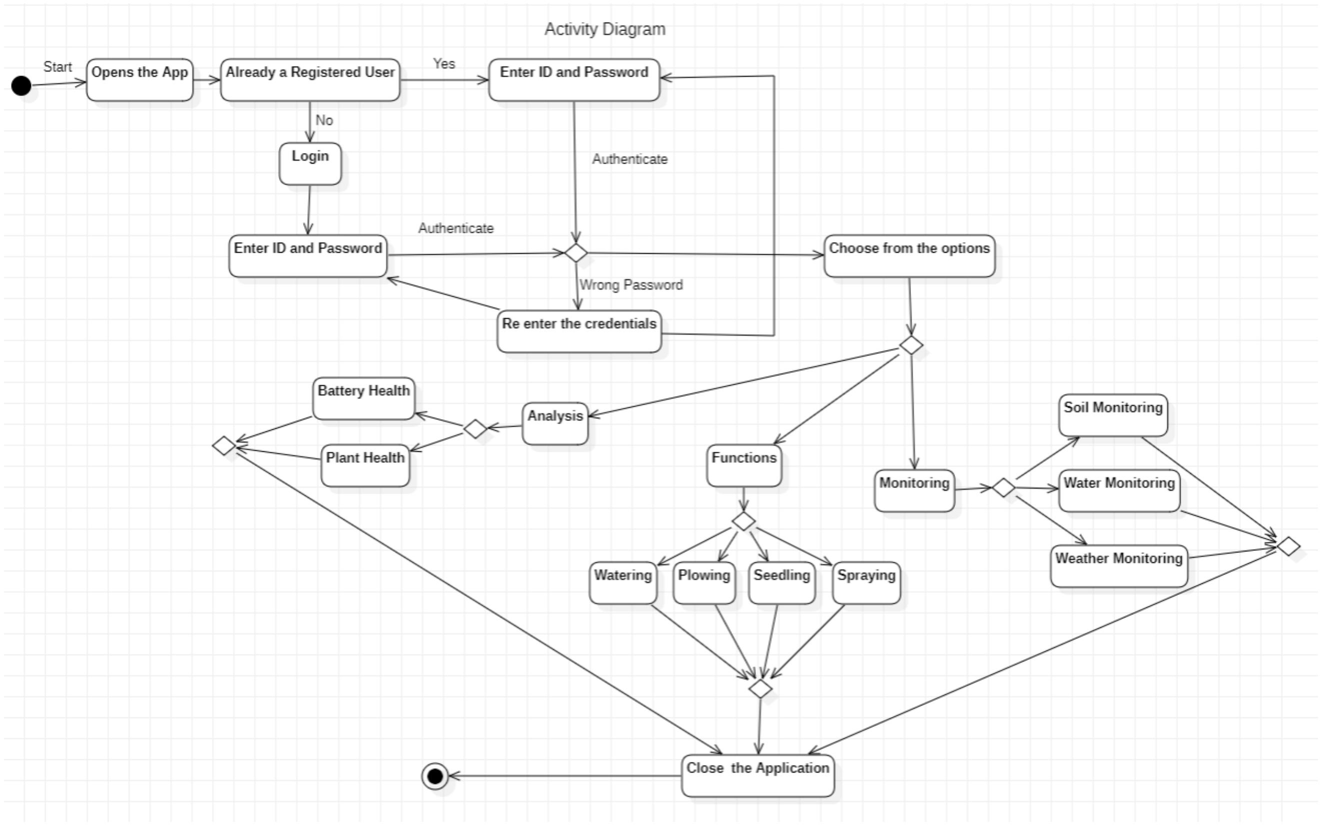


Figure 8

6. State Chart Diagram

1) **<u>States and Transitions</u>**:

   ➢ **Ready**:

        - Initial state of the system.
        - Transitions to "Collecting" upon the initiation of data collection.
    - **Collecting**:
        - State representing the collection of soil, water, and weather data by the Robot.
        - Transitions to "Analyzing" after data collection is complete.
    - **Analyzing**:
        - State representing the analysis of collected data.
        - Transitions to "Reporting" after analysis is complete.
    - **Reporting**:
        - State representing the generation of reports and insights.
        - No outgoing transitions as it is a final state.

2) **Events**:
    - **Data Collection**:
        - Triggered when the system receives a request to initiate data collection.
    - **Soil Data:**
        - Triggered when soil data collection is complete.
    - **Weather Data**:
        - Triggered when weather data collection is complete.
    - **Soil Quality**:
        - Triggered when soil quality analysis is complete.
    - **Water Data:**
        - Triggered when water data collection is complete.
    - **Plant Health:**
        - Triggered when plant health analysis is complete.

3) **Explanation**:
    - The system begins in the "Ready" state awaiting the initiation of data collection.
    - Upon receiving a data collection request, the system transitions to the "Collecting" state where soil, water, and weather data are collected.
    - After data collection is complete, the system transitions to the "Analyzing" state where the collected data is analyzed.
    - Once analysis is complete, the system transitions to the "Reporting" state where reports and insights are generated.
    - The system remains in the "Reporting" state until a new data collection request is received.

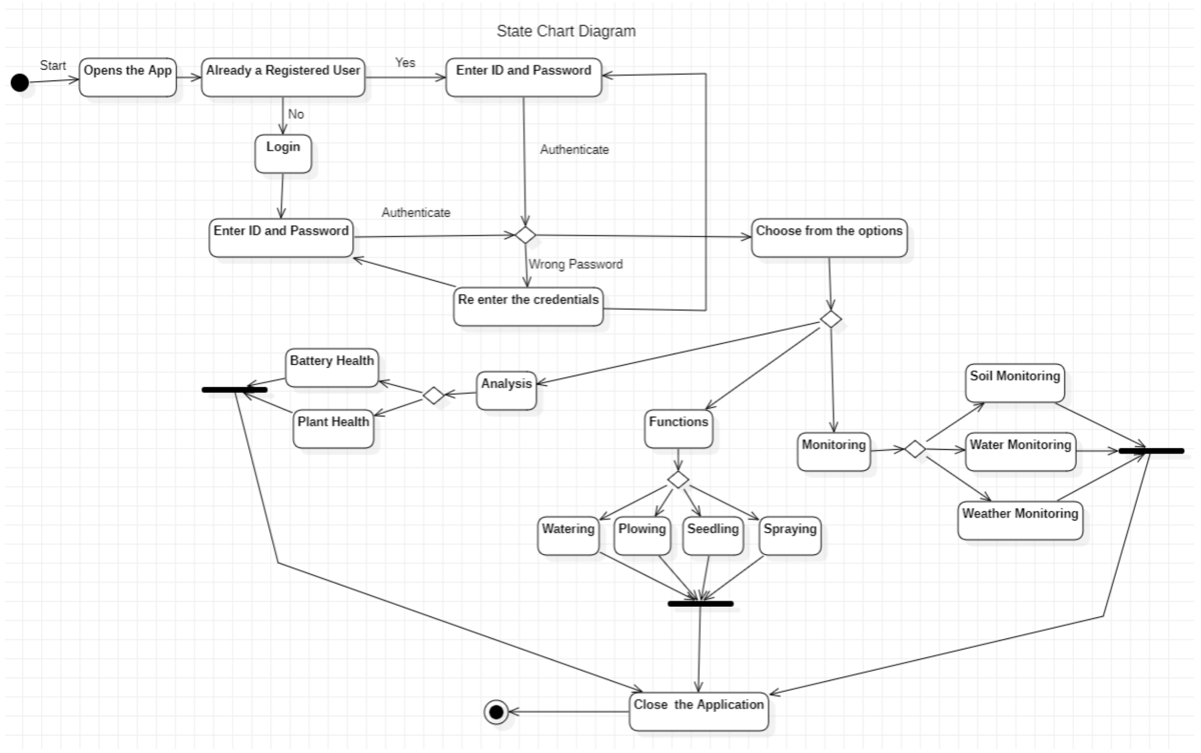


Figure 9

7. Component Diagram

1) **<u>Components</u>**:

- **App Component:**
  - Represents the agricultural monitoring application.
  - Responsibilities include user interface management, data processing, and communication with other components.
- **Robot Component:**
  - Represents the specialized agricultural robot equipped with sensors for data collection.
  - Responsibilities include sensor control, data collection, analysis, and communication with the App component.
- **Data Analysis Component:**
  - Represents the module responsible for analyzing collected data.
  - Responsibilities include analyzing soil quality, weather conditions, and crop health data received from the Robot component.
- **Data Storage Component:**
  - Represents the module responsible for storing collected data.

        - Responsibilities include storing soil, water, weather, and crop health data for future analysis and reference.
- **User Interface Component:**
    - Represents the module responsible for managing user interactions.
    - Responsibilities include displaying data analysis results to users and receiving user input.
- **Access Control Component:**
    - Represents the module responsible for managing user access to the system.
    - Responsibilities include user authentication, authorization, and access control policies.
- **System Maintenance Component:**
    - Represents the module responsible for monitoring system performance and performing maintenance tasks.
    - Responsibilities include system health monitoring, performance optimization, and software updates.

2) <u>**Dependencies**</u>:
- **App depends on Robot, Data Analysis, Data Storage, User Interface, and Access Control components:**
    - The App component interacts with the Robot component to initiate data collection and receive collected data.
    - It interacts with the Data Analysis component to process collected data and generate analysis results.
    - It interacts with the Data Storage component to store collected data for future reference.
    - It interacts with the User Interface component to display analysis results to users and receive user input.
    - It interacts with the Access Control component to manage user authentication and authorization.
- **Robot depends on Data Analysis and Data Storage components:**
    - The Robot component interacts with the Data Analysis component to perform data analysis.
    - It interacts with the Data Storage component to store collected data.
- **Data Analysis depends on Data Storage component:**

  - The Data Analysis component retrieves collected data from the Data Storage component for analysis.
- **User Interface depends on App and Access Control components:**
  - The User Interface component interacts with the App component to display analysis results and receive user input.
  - It interacts with the Access Control component to manage user authentication and authorization.
- **System Maintenance component may depend on other components for monitoring purposes:**
  - The System Maintenance component may interact with other components to monitor system performance and health.

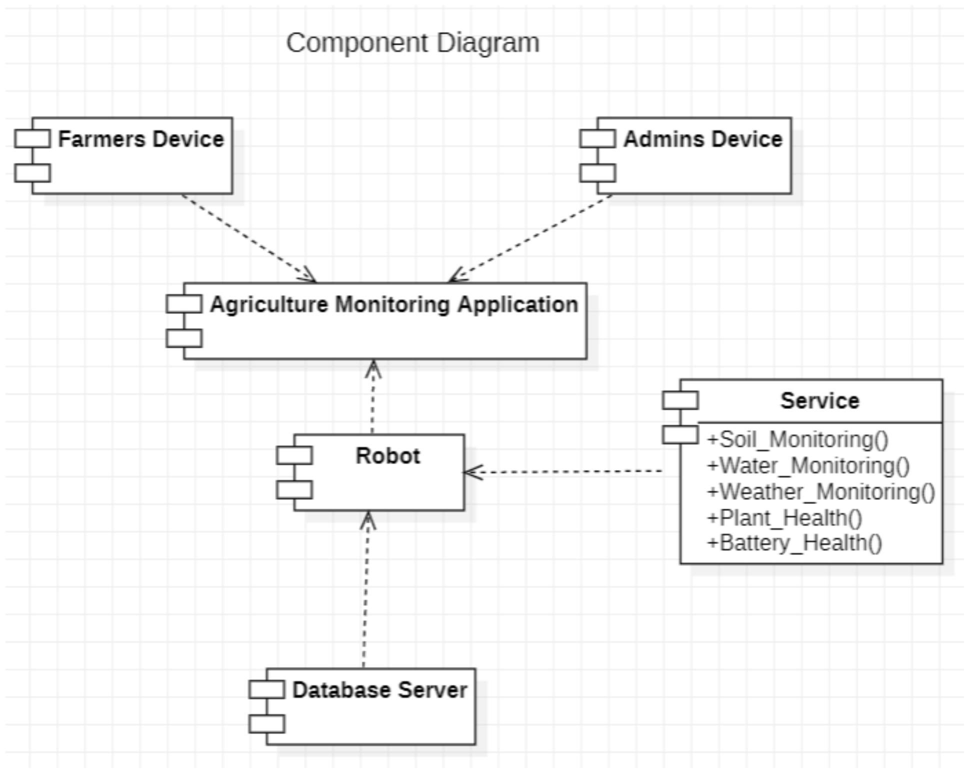


Figure 10

8. Deployment Diagram

1) **<u>Components</u>**:

- **Agricultural Monitoring Application:**
  - Represents the software application used by the Farmer, Agricultural Researcher, and Admin.
  - Installed on a server node for central access.
- **Robot**:

        - Represents the specialized agricultural robot equipped with sensors for data collection.
        - Installed on a physical robot chassis.
    - **Database Server**:
        - Represents the server hosting the database storing agricultural data.
        - Installed on a dedicated database server node.
    - **Weather Service:**
        - Represents the external service providing real-time weather data.
        - Accessed through the internet or a dedicated interface node.
2) **Nodes**:
    - **Farmer's Device:**
        - Represents the device used by the Farmer to access the agricultural monitoring application.
        - Can be a computer, tablet, or smartphone.
    - **Researcher's Device:**
        - Represents the device used by the Agricultural Researcher to access the agricultural monitoring application.
        - Similar to the Farmer's device.
    - **Admin's Device:**
        - Represents the device used by the Admin to manage the agricultural monitoring system.
        - Similar to the Farmer's device.
    - **Robot Chassis:**
        - Represents the physical chassis of the agricultural robot where sensors are installed.
        - Connected to the Robot component.
3) **Communication Paths**:
    - **Farmer's Device <-> Agricultural Monitoring Application:**
        - Represents the communication path between the Farmer's device and the deployed agricultural monitoring application on the server node.
    - **Researcher's Device <-> Agricultural Monitoring Application:**
        - Represents the communication path between the Agricultural Researcher's device and the deployed agricultural monitoring application on the server node.

- **Admin's Device <-> Agricultural Monitoring Application:**
  - Represents the communication path between the Admin's device and the deployed agricultural monitoring application on the server node.
- **Robot Chassis <-> Agricultural Monitoring Application:**
  - Represents the communication path between the physical robot chassis and the deployed agricultural monitoring application on the server node.
- **Weather Service <-> Agricultural Monitoring Application**:
  - Represents the communication path between the external weather service and the deployed agricultural monitoring application for accessing real-time weather data.

4) **Hardware Components:**
   - **Robot Chassis**: Represents the physical body of the agricultural robot.
   - Sensors: Include various sensors for measuring soil, water, and weather parameters, such as pH, moisture, temperature, etc.
   - **ESP32 and ESP32 CAM**: Microcontroller boards used for data collection and communication.
   - **Solar Panels**: Provide power to the system, ensuring continuous operation in remote locations.
   - **Motors, Pumps, Fans**: Mechanical components for controlling movement and operations of the robot.
   - **Battery**: Power source for the entire system, ensuring sustained functionality.

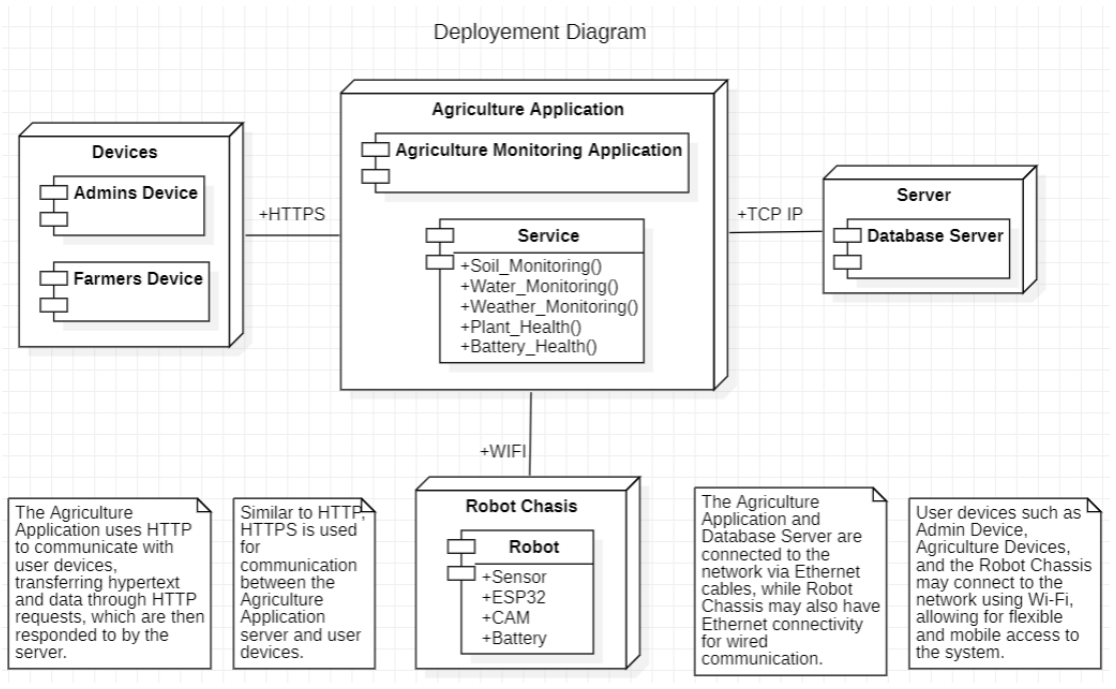

Deployement Diagram
Devices
Admins Device
Farmers Device
+HTTPS
Agriculture Application
Agriculture Monitoring Application
Service
+Soil_Monitoring()
+Water_Monitoring()
+Weather_Monitoring()
+Plant_Health()
+Battery_Health()
+TCP IP
Server
Database Server
+WIFI
Robot Chasis
Robot
+Sensor
+ESP32
+CAM
+Battery
The Agriculture Application uses HTTP to communicate with user devices, transferring hypertext and data through HTTP requests, which are then responded to by the server.
Similar to HTTP, HTTPS is used for communication between the Agriculture Application server and user devices.
The Agriculture Application and Database Server are connected to the network via Ethernet cables, while Robot Chassis may also have Ethernet connectivity for wired communication.
User devices such as Admin Device, Agriculture Devices, and the Robot Chassis may connect to the network using Wi-Fi, allowing for flexible and mobile access to the system.

Figure 11

9. Flowchart

1) <u>Opening the App</u>:

- **Start**: The flowchart begins with the start symbol, indicating the start of the process.
- **Farmer Opens App**: This step represents the Farmer initiating the agricultural monitoring application.
- **Initiate Data Collection**: The Farmer initiates the process of collecting soil, water, and weather data through the application.
- **Request Data Collection**: The App sends a request to the Robot to start data collection.

2) <u>Data Collection by the Robot</u>:

- **Collect Data**: The Robot begins collecting soil, water, and weather data using its sensors.
- **Analyze Soil Quality**: The Robot performs in-depth soil quality analysis.
- **Analyze Crop Health**: The Robot processes multispectral imagery to analyze crop health.

3) **<u>Sending Data to the App</u>**:

- **Send Data to App**: The Robot sends the collected data (soil, water, weather, and crop health) to the App.

4) **<u>Data Display and Analysis</u>**:

- **Display Data**: The App displays the collected data to the Farmer.
- **View Data Analysis**: The Farmer views insights regarding soil quality, weather conditions, and crop health.

5) **Decision Making by the Farmer**:
   - **Make Decisions**: Based on the displayed data, the Farmer makes decisions regarding crop management, irrigation, and nutrient delivery.
6) **Researcher's Access**:
   - **Access Request**: The Agricultural Researcher requests access to the system for research purposes.
   - **Admin Approval**: The Admin approves the Researcher's access request.
   - **Access Granted**: The Researcher receives access to the requested data.
   - **Research Analysis**: The Researcher analyzes the data for research purposes.
7) **System Maintenance**:
   - **Monitor System Performance**: The Admin monitors system performance and performs maintenance tasks as needed.

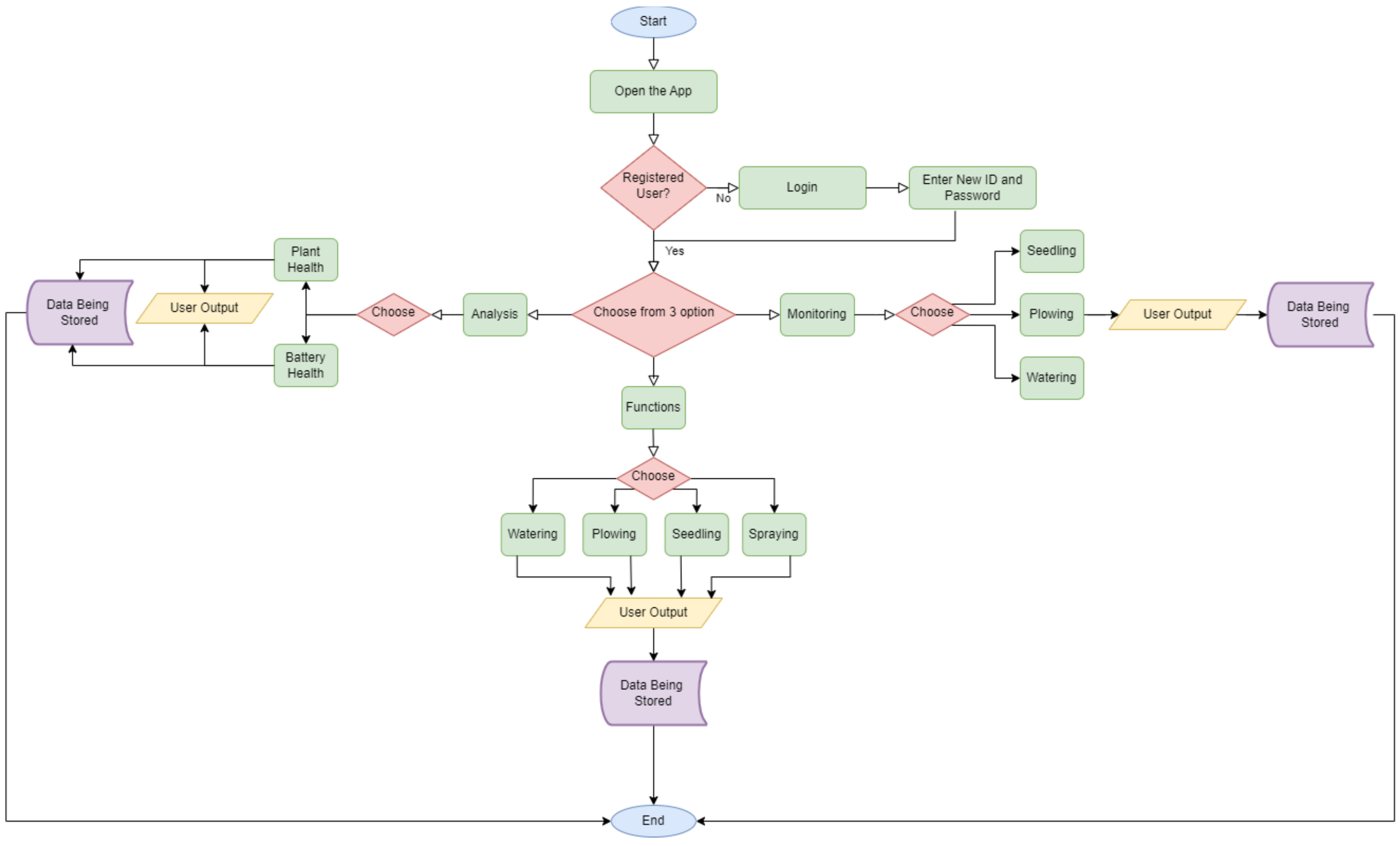


Figure 12

10. ER Diagram

1) **Entities**:

- **Farmer**:
  - Represents the users who interact with the agricultural monitoring system.
  - Attributes: FarmerID (Primary Key), Name, Email, etc.
- **Admin**:
  - Represents system administrators responsible for managing user access and system maintenance.
  - Attributes: AdminID (Primary Key), Name, Email, Role, etc.
- **Data**:
  - Represents the collected agricultural data.
  - Attributes: DataID (Primary Key), Timestamp, SoilMoisture, pHLevel, Temperature, etc.
- **Robot**:
  - Represents the specialized agricultural robot.
  - Attributes: RobotID (Primary Key).
- **Agricultural Data:**
  - Represents the collected agricultural data.
  - Attributes: DataID (Primary Key), SoilMoisture, pHLevel, Temperature.
- **Weather Service:**
  - Represents the external weather service.
  - Attributes: ServiceID (Primary Key), Name, URL.
- **Plant Health:**
  - Represents the health status of plants.
  - Attributes: PlantHealthID (Primary Key), HealthStatus, Timestamp.
- **Battery Health:**
  - Represents the health status of the battery.
  - Attributes: BatteryHealthID (Primary Key), HealthStatus, Timestamp.
- **Water Quality:**
  - Represents the quality of water.
  - Attributes: WaterQualityID (Primary Key), pHLevel, Timestamp.

2) **Relationships**:

- **Farmer Initiates Data Collection:**
  - One-to-Many relationship from Farmer to Data.
  - Each Farmer can initiate multiple data collection activities.
  - Example: A Farmer (entity) initiates the collection of soil, water, and weather data (entity).
- **Agricultural Researcher Analyzes Data:**
  - One-to-Many relationship from Agricultural Researcher to Data.
  - Each Agricultural Researcher can analyze multiple sets of collected data.
  - Example: An Agricultural Researcher (entity) analyzes historical data (entity) for research purposes.
- **Admin Manages Users:**
  - One-to-Many relationship from Admin to Farmer, Agricultural Researcher.
  - Each Admin can manage multiple users.

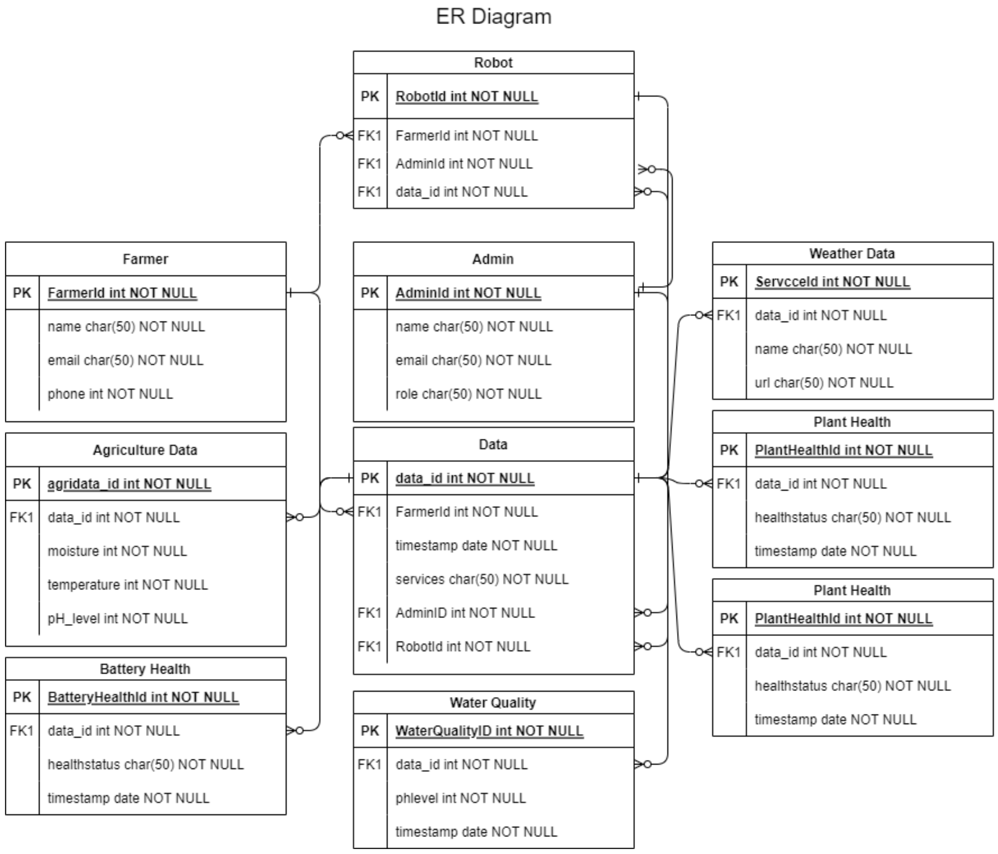


Figure 13

11. Estimation of the project using COCOMO/COCOCMO 2 approach

External Inputs (EI):

- External Inputs represent the user interactions that result in data entering the system from an external source.
- Examples include data entry forms, uploading files, or data imports from external systems.
- Each unique input type is counted separately.

External Outputs (EO):

- External Outputs represent the user interactions that result in data being sent out of the system.
- Examples include reports, screen displays, or data exports.
- Each unique output type is counted separately.

External Inquiries (EQ):

- External Inquiries represent the user interactions that involve both input and output but do not result in significant data processing.
- Examples include online queries or data lookups.
- Each unique inquiry type is counted separately.

Internal Logical Files (ILF):

- Internal Logical Files represent the logical files maintained within the system.
- These files are used and maintained by the software but are not directly visible to the user.
- Examples include databases, data files, or data structures used for storage within the application.
- Each unique logical file is counted separately.

External Interface Files (EIF):

- External Interface Files represent the logical files referenced by the software but are maintained by external systems.
- These files are used by the software to exchange data with external systems.
- Examples include files shared with other applications, databases managed by external systems, or web services.
- Each unique interface file is counted separately.

1. **Estimating Function Points (FP):**

- External Inputs (EI): Moderate complexity - 6
- External Outputs (EO): Moderate complexity - 5
- External Inquiries (EQ): Low complexity - 4
- Internal Logical Files (ILF): High complexity - 9
- External Interface Files (EIF): Moderate complexity – 6

   Total Function Points (FP) = EI + EO + EQ + ILF + EIF = 6 + 5 + 4 + 9 + 6 = 30 FP

2. **Applying COCOMO 2:**
   Using COCOMO 2, we can estimate the effort and schedule based on the estimated function points.
   - **Effort Estimation:**
     Using the Basic COCOMO model formula:
       - E = a * (KLOC)^b

     We'll solve for 'a' using the given effort (E) and project size (KLOC):
       - 4 PM = a * (0.03)^b

     Let's assume b = 1.15 (adjusted for the desired effort).
     Solving for 'a':
       - a = 4 / (0.03)^1.15 ≈ 261.47
   - **Schedule Estimation:**
     Using the Basic COCOMO model formula for schedule:
       - D = c * (E)^d

     Let's assume d = 0.35 (adjusted for the desired schedule).
     Solving for 'c':
       - 5 months = c * (4)^0.35
       - c = 5 / (4)^0.35 ≈ 2.42

Effort Estimation:

- E = 261.47 * (0.03)^1.15 ≈ 4 person-months

Schedule Estimation:

- D = 2.42 * (4)^0.35 ≈ 5 months

12. Test Cases

Functional Requirements:

Requirement:

1. **Monitoring Water Quality:**
   The system should accurately measure parameters such as pH, electrical conductivity, dissolved oxygen, turbidity, and nutrients using specialized sensors.

Test Cases:

1. **Test Case ID:** TC_WQ_01
   Test Case Description: Verify that the system accurately measures pH levels using the water quality sensor.
   Test Steps:
   - Submerge the water quality sensor in a solution with known pH.
   - Retrieve the measured pH value from the sensor.

   Expected Result:
   - The measured pH value matches the known pH value within an acceptable tolerance.
2. **Test Case ID**: TC_WQ_02
   Test Case Description: Verify that the system accurately measures electrical conductivity.
   Test Steps:
   - Submerge the water quality sensor in a solution with known electrical conductivity.
   - Retrieve the measured electrical conductivity value from the sensor.

   Expected Result:
   - The measured electrical conductivity value matches the known value within an acceptable tolerance.

Requirement:

1. **Monitoring Soil Quality**:
   The system should measure moisture, pH, and nutrient levels in the soil using sensors mounted on the agricultural robot.

Test Cases:

1. **Test Case ID**: TC_SQ_01
   Test Case Description: Verify that the system accurately measures soil moisture.
   Test Steps:
   - Insert the soil moisture sensor into soil with known moisture content.
   - Retrieve the measured moisture level from the sensor.

   Expected Result:
   - The measured moisture level matches the known moisture content within an acceptable tolerance.
2. **Test Case ID**: TC_SQ_02
   Test Case Description: Verify that the system accurately measures soil pH.
   Test Steps:
   - Insert the soil pH sensor into soil with known pH.
   - Retrieve the measured pH value from the sensor.

   Expected Result:
   - The measured pH value matches the known pH value within an acceptable tolerance.

Requirement:

1. **Crop Health Monitoring**:
   The system should analyze multispectral imagery to assess plant stress, growth patterns, and overall vitality.

Test Cases:

1. **Test Case ID**: TC_CHM_01
   Test Case Description: Verify that the system accurately detects plant stress in multispectral imagery.
   Test Steps:
   - Provide the system with multispectral imagery containing stressed plants.
   - Analyze the output of the system to identify stress markers.

   Expected Result:
   - The system correctly identifies stress markers in the multispectral imagery.

Non-Functional Requirements:

Requirement:

1. **Battery Health Monitoring:**
   The system should monitor battery health to ensure sustained functionality and longevity.

Test Cases:

1. **Test Case ID**: TC_BH_01
   Test Case Description: Verify that the system accurately monitors battery voltage.
   Test Steps:
   - Measure the voltage of the battery using a voltmeter.
   - Compare the measured voltage with the voltage reported by the system.

   Expected Result:
   - The reported voltage matches the measured voltage within an acceptable tolerance.

Features

1. **Real-Time Monitoring:**
   Continuous monitoring of water quality, soil parameters, and weather conditions provides farmers with real-time insights into the agricultural environment.
2. **Water Quality Analysis:**
   Precise measurement of pH, electrical conductivity, dissolved oxygen, turbidity, and nutrient levels allows for thorough water quality analysis, aiding in effective irrigation planning.
3. **Soil Quality Assessment:**
   Sensors measure soil moisture, pH, and nutrient levels, enabling farmers to assess soil health and make informed decisions regarding nutrient management and crop health.
4. **Weather Forecast Integration:**
   Real-time weather data integration assists in planning farming activities based on upcoming weather conditions, optimizing resource utilization.
5. **Crop Health Monitoring:**
   Implementation of the NDVI algorithm provides farmers with insights into crop health, including stress levels, growth patterns, and overall vitality, contributing to proactive crop management.
6. **Battery Health Monitoring:**
   Dedicated monitoring of the robot's battery health ensures sustained functionality and longevity, reducing downtime and maintenance requirements.

Source Code

1. **.dart_tools:**

dartpad:

```
// Flutter web plugin registrant file.
// Generated file. Do not edit.
// @dart = 2.13
// ignore_for_file: type=lint

import 'package:battery_plus_web/battery_plus_web.dart';
import 'package:file_picker/_internal/file_picker_web.dart';
import 'package:firebase_auth_web/firebase_auth_web.dart';
import 'package:firebase_core_web/firebase_core_web.dart';
import 'package:firebase_storage_web/firebase_storage_web.dart';
import 'package:fluttertoast/fluttertoast_web.dart';
import 'package:geolocator_web/geolocator_web.dart';
import 'package:image_picker_for_web/image_picker_for_web.dart';
import 'package:rive_common/rive_web.dart';
import 'package:shared_preferences_web/shared_preferences_web.dart';
import 'package:flutter_web_plugins/flutter_web_plugins.dart';

void registerPlugins([final Registrar? pluginRegistrar]) {
  final Registrar registrar = pluginRegistrar ?? webPluginRegistrar;
  BatteryPlusPlugin.registerWith(registrar);
  FilePickerWeb.registerWith(registrar);
  FirebaseAuthWeb.registerWith(registrar);
  FirebaseCoreWeb.registerWith(registrar);
  FirebaseStorageWeb.registerWith(registrar);
  FluttertoastWebPlugin.registerWith(registrar);
  GeolocatorPlugin.registerWith(registrar);
  ImagePickerPlugin.registerWith(registrar);
  RivePlugin.registerWith(registrar);
  SharedPreferencesPlugin.registerWith(registrar);
  registrar.registerMessageHandler();
}
```

extension_discovery:

{"version":2,"entries":[{"package":"agrobot","rootUri":"../","packageUri":"lib/"}]}

dart_plugin_registrant:

```
// Generated file. Do not edit.
// This file is generated from template in file
`flutter_tools/lib/src/flutter_plugins.dart`.
// @dart = 3.2

import 'dart:io'; // flutter_ignore: dart_io_import.
import 'package:geocoding_android/geocoding_android.dart';
import 'package:geolocator_android/geolocator_android.dart';
import 'package:image_picker_android/image_picker_android.dart';
import 'package:path_provider_android/path_provider_android.dart';
import 'package:shared_preferences_android/shared_preferences_android.dart';
import 'package:geocoding_ios/geocoding_ios.dart';
import 'package:geolocator_apple/geolocator_apple.dart';
import 'package:image_picker_ios/image_picker_ios.dart';
import 'package:path_provider_foundation/path_provider_foundation.dart';
import
'package:shared_preferences_foundation/shared_preferences_foundation.dart';
import 'package:battery_plus_linux/battery_plus_linux.dart';
import 'package:file_selector_linux/file_selector_linux.dart';
import 'package:image_picker_linux/image_picker_linux.dart';
import 'package:path_provider_linux/path_provider_linux.dart';
import 'package:shared_preferences_linux/shared_preferences_linux.dart';
import 'package:file_selector_macos/file_selector_macos.dart';
import 'package:geolocator_apple/geolocator_apple.dart';
import 'package:image_picker_macos/image_picker_macos.dart';
import 'package:path_provider_foundation/path_provider_foundation.dart';
import
'package:shared_preferences_foundation/shared_preferences_foundation.dart';
import 'package:file_selector_windows/file_selector_windows.dart';
import 'package:image_picker_windows/image_picker_windows.dart';
import 'package:path_provider_windows/path_provider_windows.dart';
import 'package:shared_preferences_windows/shared_preferences_windows.dart';

@pragma('vm:entry-point')
class _PluginRegistrant {

  @pragma('vm:entry-point')
  static void register() {
```

```
  if (Platform.isAndroid) {
    try {
      GeocodingAndroid.registerWith();
    } catch (err) {
      print(
        '`geocoding_android` threw an error: $err. '
        'The app may not function as expected until you remove this plugin from pubspec.yaml'
      );
    }

    try {
      GeolocatorAndroid.registerWith();
    } catch (err) {
      print(
        '`geolocator_android` threw an error: $err. '
        'The app may not function as expected until you remove this plugin from pubspec.yaml'
      );
    }

    try {
      ImagePickerAndroid.registerWith();
    } catch (err) {
      print(
        '`image_picker_android` threw an error: $err. '
```

```
          'The app may not function as expected until you remove this plugin from
pubspec.yaml'
        );
      }

      try {
        PathProviderAndroid.registerWith();
      } catch (err) {
        print(
          '`path_provider_android` threw an error: $err. '
          'The app may not function as expected until you remove this plugin from
pubspec.yaml'
        );
      }

      try {
        SharedPreferencesAndroid.registerWith();
      } catch (err) {
        print(
          '`shared_preferences_android` threw an error: $err. '
          'The app may not function as expected until you remove this plugin from
pubspec.yaml'
        );
      }

    } else if (Platform.isIOS) {
```

```
    try {
      GeocodingIOS.registerWith();
    } catch (err) {
      print(
        '`geocoding_ios` threw an error: $err. '
        'The app may not function as expected until you remove this plugin from
pubspec.yaml'
      );
    }

    try {
      GeolocatorApple.registerWith();
    } catch (err) {
      print(
        '`geolocator_apple` threw an error: $err. '
        'The app may not function as expected until you remove this plugin from
pubspec.yaml'
      );
    }

    try {
      ImagePickerIOS.registerWith();
    } catch (err) {
      print(
        '`image_picker_ios` threw an error: $err. '
```

```
          'The app may not function as expected until you remove this plugin from
pubspec.yaml'
        );
      }

      try {
        PathProviderFoundation.registerWith();
      } catch (err) {
        print(
          '`path_provider_foundation` threw an error: $err. '
          'The app may not function as expected until you remove this plugin from
pubspec.yaml'
        );
      }

      try {
        SharedPreferencesFoundation.registerWith();
      } catch (err) {
        print(
          '`shared_preferences_foundation` threw an error: $err. '
          'The app may not function as expected until you remove this plugin from
pubspec.yaml'
        );
      }

    } else if (Platform.isLinux) {
```

```
    try {
      BatteryPlusLinux.registerWith();
    } catch (err) {
      print(
        '`battery_plus_linux` threw an error: $err. '
        'The app may not function as expected until you remove this plugin from
pubspec.yaml'
      );
    }

    try {
      FileSelectorLinux.registerWith();
    } catch (err) {
      print(
        '`file_selector_linux` threw an error: $err. '
        'The app may not function as expected until you remove this plugin from
pubspec.yaml'
      );
    }

    try {
      ImagePickerLinux.registerWith();
    } catch (err) {
      print(
        '`image_picker_linux` threw an error: $err. '
```

```
          'The app may not function as expected until you remove this plugin from
pubspec.yaml'
        );
      }

      try {
        PathProviderLinux.registerWith();
      } catch (err) {
        print(
          '`path_provider_linux` threw an error: $err. '
          'The app may not function as expected until you remove this plugin from
pubspec.yaml'
        );
      }

      try {
        SharedPreferencesLinux.registerWith();
      } catch (err) {
        print(
          '`shared_preferences_linux` threw an error: $err. '
          'The app may not function as expected until you remove this plugin from
pubspec.yaml'
        );
      }

    } else if (Platform.isMacOS) {
```

```
  try {
    FileSelectorMacOS.registerWith();
  } catch (err) {
    print(
      '`file_selector_macos` threw an error: $err. '
      'The app may not function as expected until you remove this plugin from
pubspec.yaml'
    );
  }

  try {
    GeolocatorApple.registerWith();
  } catch (err) {
    print(
      '`geolocator_apple` threw an error: $err. '
      'The app may not function as expected until you remove this plugin from
pubspec.yaml'
    );
  }

  try {
    ImagePickerMacOS.registerWith();
  } catch (err) {
    print(
      '`image_picker_macos` threw an error: $err. '
```

```
        'The app may not function as expected until you remove this plugin from
pubspec.yaml'
      );
    }

    try {
      PathProviderFoundation.registerWith();
    } catch (err) {
      print(
        '`path_provider_foundation` threw an error: $err. '
        'The app may not function as expected until you remove this plugin from
pubspec.yaml'
      );
    }

    try {
      SharedPreferencesFoundation.registerWith();
    } catch (err) {
      print(
        '`shared_preferences_foundation` threw an error: $err. '
        'The app may not function as expected until you remove this plugin from
pubspec.yaml'
      );
    }

  } else if (Platform.isWindows) {
    try {
      FileSelectorWindows.registerWith();
    } catch (err) {
      print(
        '`file_selector_windows` threw an error: $err. '
        'The app may not function as expected until you remove this plugin from
pubspec.yaml'
      );
    }

    try {
      ImagePickerWindows.registerWith();
    } catch (err) {
      print(
        '`image_picker_windows` threw an error: $err. '
        'The app may not function as expected until you remove this plugin from
pubspec.yaml'
      );
    }

    try {
      PathProviderWindows.registerWith();
    } catch (err) {
      print(
        '`path_provider_windows` threw an error: $err. '
        'The app may not function as expected until you remove this plugin from
pubspec.yaml'
      );
    }

    try {
      SharedPreferencesWindows.registerWith();
    } catch (err) {
      print(
        '`shared_preferences_windows` threw an error: $err. '
        'The app may not function as expected until you remove this plugin from
pubspec.yaml'
      );
    }

  }
 }
}
```

package_config:

```
{
  "configVersion": 2,
  "packages": [
    {
      "name": "_flutterfire_internals",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/_flutterfire_intern
als-1.3.28",
      "packageUri": "lib/",
      "languageVersion": "3.2"
    },
    {
      "name": "animated_text_kit",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/animated_text_ki
t-4.2.2",
      "packageUri": "lib/",
      "languageVersion": "2.17"
    },
    {
      "name": "archive",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/archive-3.4.10",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "args",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/args-2.4.2",
      "packageUri": "lib/",
      "languageVersion": "2.19"
```

```
  },
  {
   "name": "async",
   "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/async-2.11.0",
   "packageUri": "lib/",
   "languageVersion": "2.18"
  },
  {
   "name": "awesome_dialog",
   "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/awesome_dialog-
3.2.0",
   "packageUri": "lib/",
   "languageVersion": "2.12"
  },
  {
   "name": "battery_plus",
   "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/battery_plus-
2.2.2",
   "packageUri": "lib/",
   "languageVersion": "2.12"
  },
  {
   "name": "battery_plus_linux",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/battery_plus_linu
x-1.2.0",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "battery_plus_macos",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/battery_plus_mac
os-1.1.1",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "battery_plus_platform_interface",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/battery_plus_plat
form_interface-1.2.2",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "battery_plus_web",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/battery_plus_web
-1.1.0",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "battery_plus_windows",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/battery_plus_win
dows-1.1.3",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "boolean_selector",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/boolean_selector-
2.1.1",
      "packageUri": "lib/",
      "languageVersion": "2.17"
    },
    {
      "name": "characters",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/characters-1.3.0",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "checked_yaml",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/checked_yaml-
2.0.3",
      "packageUri": "lib/",
      "languageVersion": "2.19"
    },
    {
      "name": "cli_util",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/cli_util-0.4.1",
      "packageUri": "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "clock",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/clock-1.1.1",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "collection",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/collection-
1.18.0",
      "packageUri": "lib/",
      "languageVersion": "2.18"
    },
    {
      "name": "convert",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/convert-3.1.1",
      "packageUri": "lib/",
      "languageVersion": "2.18"
    },
    {
      "name": "cross_file",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/cross_file-
0.3.4+1",
      "packageUri": "lib/",
      "languageVersion": "3.3"
    },
    {
      "name": "crypto",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/crypto-3.0.3",
      "packageUri": "lib/",
      "languageVersion": "2.19"
    },
    {
      "name": "dbus",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/dbus-0.7.10",
      "packageUri": "lib/",
      "languageVersion": "2.17"
    },
    {
      "name": "dio",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/dio-5.4.2+1",
      "packageUri": "lib/",
      "languageVersion": "2.15"
    },
    {
      "name": "fake_async",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/fake_async-
1.3.1",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "ffi",
      "rootUri": "file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/ffi-
2.1.2",
      "packageUri": "lib/",
      "languageVersion": "3.3"
    },
    {
      "name": "file",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/file-7.0.0",
      "packageUri":  "lib/",
      "languageVersion":  "3.0"
    },
    {
      "name": "file_picker",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/file_picker-
6.2.1",
      "packageUri":  "lib/",
      "languageVersion":  "2.19"
    },
    {
      "name": "file_selector_linux",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/file_selector_linu
x-0.9.2+1",
      "packageUri":  "lib/",
      "languageVersion":  "2.19"
    },
    {
      "name": "file_selector_macos",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/file_selector_mac
os-0.9.3+3",
      "packageUri":  "lib/",
      "languageVersion":  "2.19"
    },
    {
      "name": "file_selector_platform_interface",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/file_selector_plat
form_interface-2.6.2",
      "packageUri": "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "file_selector_windows",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/file_selector_win
dows-0.9.3+1",
      "packageUri": "lib/",
      "languageVersion": "2.19"
    },
    {
      "name": "firebase_auth",


      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/firebase_auth-
4.19.0",
      "packageUri": "lib/",
      "languageVersion": "2.18"
    },
    {
      "name": "firebase_auth_platform_interface",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/firebase_auth_pla
tform_interface-7.2.1",
      "packageUri": "lib/",
      "languageVersion": "2.18"
    },
    {
      "name": "firebase_auth_web",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/firebase_auth_we
b-5.11.0",
      "packageUri": "lib/",
      "languageVersion": "3.2"
    },
    {
      "name": "firebase_core",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/firebase_core-
2.28.0",
      "packageUri": "lib/",
      "languageVersion": "2.18"
    },
    {
      "name": "firebase_core_platform_interface",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/firebase_core_pla
tform_interface-5.0.0",
      "packageUri": "lib/",
      "languageVersion": "2.18"
    },
    {
      "name": "firebase_core_web",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/firebase_core_we
b-2.14.0",
      "packageUri": "lib/",
      "languageVersion": "3.2"
    },
    {
      "name": "firebase_storage",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/firebase_storage-
11.7.0",
      "packageUri": "lib/",
      "languageVersion": "2.18"
    },
    {
      "name": "firebase_storage_platform_interface",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/firebase_storage_
platform_interface-5.1.15",
      "packageUri": "lib/",
      "languageVersion": "2.18"
    },
    {
      "name": "firebase_storage_web",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/firebase_storage_
web-3.9.0",
      "packageUri": "lib/",
      "languageVersion": "3.2"
    },
    {
      "name": "fixnum",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/fixnum-1.1.0",
      "packageUri": "lib/",
      "languageVersion": "2.19"
    },
    {
      "name": "flutter",
      "rootUri": "file:///D:/Atharva/development/sdks/flutter/packages/flutter",
      "packageUri": "lib/",
      "languageVersion": "3.2"
    },
    {
      "name": "flutter_blue",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/flutter_blue-
0.8.0",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "flutter_blue_plus",
      "rootUri": 
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/flutter_blue_plus-
1.32.1",
      "packageUri": "lib/",
      "languageVersion": "2.15"
    },
    {
      "name": "flutter_bluetooth_serial",
      "rootUri": 
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/flutter_bluetooth
_serial-0.4.0",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "flutter_joystick",
      "rootUri": 
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/flutter_joystick-
0.0.4",
      "packageUri": "lib/",
      "languageVersion": "3.1"
    },
    {
      "name": "flutter_launcher_icons",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/flutter_launcher_i
cons-0.13.1",
      "packageUri": "lib/",
      "languageVersion": "2.18"
    },
    {
      "name": "flutter_lints",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/flutter_lints-
3.0.2",
      "packageUri": "lib/",
      "languageVersion": "3.1"
    },
    {
      "name": "flutter_local_notifications",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/flutter_local_noti
fications-8.2.0",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "flutter_local_notifications_platform_interface",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/flutter_local_noti
fications_platform_interface-4.0.1",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "flutter_plugin_android_lifecycle",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/flutter_plugin_an
droid_lifecycle-2.0.17",
      "packageUri": "lib/",
      "languageVersion": "2.19"
    },
    {
      "name": "flutter_spinkit",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/flutter_spinkit-
5.2.1",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "flutter_test",
      "rootUri": "file:///D:/Atharva/development/sdks/flutter/packages/flutter_test",
      "packageUri": "lib/",
      "languageVersion": "3.2"
    },
    {
      "name": "flutter_web_plugins",
      "rootUri":
"file:///D:/Atharva/development/sdks/flutter/packages/flutter_web_plugins",
      "packageUri":  "lib/",
      "languageVersion":  "3.2"
    },
    {
      "name": "fluttertoast",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/fluttertoast-
8.2.4",
      "packageUri":  "lib/",
      "languageVersion":  "2.12"
    },
    {
      "name": "font_awesome_flutter",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/font_awesome_fl
utter-10.7.0",
      "packageUri":  "lib/",
      "languageVersion":  "3.0"
    },
    {
      "name": "geocoding",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/geocoding-2.2.2",
      "packageUri":  "lib/",
      "languageVersion":  "2.17"
    },
    {
      "name": "geocoding_android",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/geocoding_andro
id-3.3.0",
      "packageUri": "lib/",
      "languageVersion": "2.17"
    },
    {
      "name": "geocoding_ios",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/geocoding_ios-
2.3.0",
      "packageUri": "lib/",
      "languageVersion": "2.17"
    },
    {
      "name": "geocoding_platform_interface",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/geocoding_platfo
rm_interface-3.2.0",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "geolocator",
      "rootUri": 
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/geolocator-
11.0.0",
      "packageUri": "lib/",
      "languageVersion": "2.15"
    },
    {
      "name": "geolocator_android",
      "rootUri": 
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/geolocator_andro
id-4.5.4",
      "packageUri": "lib/",
      "languageVersion": "2.15"
    },
    {
      "name": "geolocator_apple",
      "rootUri": 
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/geolocator_apple
-2.3.7",
      "packageUri": "lib/",
      "languageVersion": "2.15"
    },
    {
      "name": "geolocator_platform_interface",
      "rootUri": 
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/geolocator_platfo
rm_interface-4.2.2",
      "packageUri": "lib/",
      "languageVersion": "2.15"
    },
    {
      "name": "geolocator_web",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/geolocator_web-
3.0.0",
      "packageUri": "lib/",
      "languageVersion": "2.15"
    },
    {
      "name": "geolocator_windows",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/geolocator_wind
ows-0.2.3",
      "packageUri": "lib/",
      "languageVersion": "2.15"
    },
    {
      "name": "get",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/get-4.6.6",
      "packageUri": "lib/",
      "languageVersion": "2.15"
    },
    {
      "name": "graphs",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/graphs-2.3.1",
      "packageUri": "lib/",
      "languageVersion": "2.18"
    },
    {
      "name": "http",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/http-1.2.1",
      "packageUri": "lib/",
      "languageVersion": "3.3"
    },
    {
      "name": "http_parser",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/http_parser-
4.0.2",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "image",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/image-4.1.7",
      "packageUri": "lib/",
      "languageVersion": "2.15"
    },
    {
      "name": "image_picker",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/image_picker-
1.0.7",
      "packageUri": "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "image_picker_android",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/image_picker_an
droid-0.8.9+5",
      "packageUri": "lib/",
      "languageVersion": "3.1"
    },
    {
      "name": "image_picker_for_web",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/image_picker_for
_web-3.0.3",
      "packageUri": "lib/",
      "languageVersion": "3.3"
    },
    {
      "name": "image_picker_ios",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/image_picker_ios
-0.8.9+2",
      "packageUri":  "lib/",
      "languageVersion":  "3.2"
    },
    {
      "name": "image_picker_linux",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/image_picker_lin
ux-0.2.1+1",
      "packageUri":  "lib/",
      "languageVersion":  "2.19"
    },
    {
      "name": "image_picker_macos",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/image_picker_ma
cos-0.2.1+1",
      "packageUri":  "lib/",
      "languageVersion":  "2.19"
    },
    {
      "name": "image_picker_platform_interface",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/image_picker_pla
tform_interface-2.10.0",
      "packageUri":  "lib/",
      "languageVersion": "3.3"
    },
    {
      "name": "image_picker_windows",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/image_picker_wi
ndows-0.2.1+1",
      "packageUri": "lib/",
      "languageVersion": "2.19"
    },
    {
      "name": "intl",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/intl-0.19.0",
      "packageUri": "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "js",
      "rootUri": "file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/js-
0.7.1",
      "packageUri": "lib/",
      "languageVersion": "3.1"
    },
    {
      "name": "json_annotation",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/json_annotation-
4.8.1",
      "packageUri": "lib/",
      "languageVersion": "2.19"
    },
    {
      "name": "leak_tracker",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/leak_tracker-
10.0.0",
      "packageUri": "lib/",
      "languageVersion": "3.1"
    },
    {
      "name": "leak_tracker_flutter_testing",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/leak_tracker_flutt
er_testing-2.0.1",
      "packageUri": "lib/",
      "languageVersion": "3.1"
    },
    {
      "name": "leak_tracker_testing",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/leak_tracker_testi
ng-2.0.1",
      "packageUri": "lib/",
      "languageVersion": "3.1"
    },
    {
      "name": "lints",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/lints-3.0.0",
      "packageUri": "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "matcher",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/matcher-
0.12.16+1",
      "packageUri": "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "material_color_utilities",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/material_color_ut
ilities-0.8.0",
      "packageUri": "lib/",
      "languageVersion": "2.17"
    },
    {
      "name": "meta",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/meta-1.11.0",
      "packageUri":  "lib/",
      "languageVersion":  "2.12"
    },
    {
      "name": "mime",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/mime-1.0.5",
      "packageUri":  "lib/",
      "languageVersion":  "3.2"
    },
    {
      "name": "path",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/path-1.9.0",
      "packageUri":  "lib/",
      "languageVersion":  "3.0"
    },
    {
      "name": "path_provider",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/path_provider-
2.1.2",
      "packageUri":  "lib/",
      "languageVersion":  "3.0"
    },
    {
      "name": "path_provider_android",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/path_provider_an
droid-2.2.2",
      "packageUri":  "lib/",
      "languageVersion":  "3.0"
    },
    {
      "name": "path_provider_foundation",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/path_provider_fo
undation-2.3.2",
      "packageUri":  "lib/",
      "languageVersion":  "3.0"
    },
    {
      "name": "path_provider_linux",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/path_provider_lin
ux-2.2.1",
      "packageUri":  "lib/",
      "languageVersion":  "2.19"
    },
    {
      "name": "path_provider_platform_interface",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/path_provider_pl
atform_interface-2.1.2",
      "packageUri": "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "path_provider_windows",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/path_provider_wi
ndows-2.2.1",
      "packageUri": "lib/",
      "languageVersion": "2.19"
    },
    {
      "name": "percent_indicator",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/percent_indicator
-4.2.3",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "permission_handler",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/permission_handl
er-10.4.5",
      "packageUri": "lib/",
      "languageVersion": "2.15"
    },
    {
      "name": "permission_handler_android",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/permission_handl
er_android-10.3.6",
      "packageUri": "lib/",
      "languageVersion": "2.15"
    },
    {
      "name": "permission_handler_apple",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/permission_handl
er_apple-9.1.4",
      "packageUri": "lib/",
      "languageVersion": "2.15"
    },
    {
      "name": "permission_handler_platform_interface",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/permission_handl
er_platform_interface-3.12.0",
      "packageUri": "lib/",
      "languageVersion": "2.14"
    },
    {
      "name": "permission_handler_windows",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/permission_handl
er_windows-0.1.3",
      "packageUri":  "lib/",
      "languageVersion":  "2.12"
    },
    {
      "name": "petitparser",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/petitparser-
6.0.2",
      "packageUri":  "lib/",
      "languageVersion":  "3.2"
    },
    {
      "name": "platform",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/platform-3.1.4",
      "packageUri":  "lib/",
      "languageVersion":  "3.0"
    },
    {
      "name": "plugin_platform_interface",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/plugin_platform_
interface-2.1.8",
      "packageUri":  "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "pointycastle",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/pointycastle-
3.8.0",
      "packageUri": "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "protobuf",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/protobuf-2.1.0",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "rive",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/rive-0.12.4",
      "packageUri": "lib/",
      "languageVersion": "2.17"
    },
    {
      "name": "rive_common",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/rive_common-
0.2.8",
      "packageUri": "lib/",
      "languageVersion": "2.17"
    },
    {
      "name": "rxdart",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/rxdart-0.26.0",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "shared_preferences",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/shared_preferenc
es-2.2.2",
      "packageUri": "lib/",
      "languageVersion": "2.19"
    },
    {
      "name": "shared_preferences_android",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/shared_preferenc
es_android-2.2.1",
      "packageUri": "lib/",
      "languageVersion": "2.19"
    },
    {
      "name": "shared_preferences_foundation",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/shared_preferenc
es_foundation-2.3.5",
      "packageUri": "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "shared_preferences_linux",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/shared_preferenc
es_linux-2.3.2",
      "packageUri": "lib/",
      "languageVersion": "2.19"
    },
    {
      "name": "shared_preferences_platform_interface",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/shared_preferenc
es_platform_interface-2.3.2",
      "packageUri": "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "shared_preferences_web",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/shared_preferenc
es_web-2.3.0",
      "packageUri": "lib/",
      "languageVersion": "3.3"
    },
    {
      "name": "shared_preferences_windows",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/shared_preferenc
es_windows-2.3.2",
      "packageUri": "lib/",
      "languageVersion": "2.19"
    },
    {
      "name": "sky_engine",
      "rootUri":
"file:///D:/Atharva/development/sdks/flutter/bin/cache/pkg/sky_engine",
      "packageUri": "lib/",
      "languageVersion": "3.2"
    },
    {
      "name": "source_span",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/source_span-
1.10.0",
      "packageUri": "lib/",
      "languageVersion": "2.18"
    },
    {
      "name": "sprintf",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/sprintf-7.0.0",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "stack_trace",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/stack_trace-
1.11.1",
      "packageUri": "lib/",
      "languageVersion": "2.18"
    },
    {
      "name": "stream_channel",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/stream_channel-
2.1.2",
      "packageUri": "lib/",
      "languageVersion": "2.19"
    },
    {
      "name": "string_scanner",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/string_scanner-
1.2.0",
      "packageUri": "lib/",
      "languageVersion": "2.18"
    },
    {
      "name": "term_glyph",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/term_glyph-
1.2.1",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "test_api",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/test_api-0.6.1",
      "packageUri": "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "timezone",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/timezone-0.8.0",
      "packageUri": "lib/",
      "languageVersion": "2.12"
    },
    {
      "name": "typed_data",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/typed_data-
1.3.2",
      "packageUri": "lib/",
      "languageVersion": "2.17"
    },
    {
      "name": "upower",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/upower-0.7.0",
      "packageUri": "lib/",
      "languageVersion": "2.14"
    },
    {
      "name": "uuid",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/uuid-4.3.3",
      "packageUri": "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "vector_math",
```

"rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/vector_math-2.1.4",
      "packageUri": "lib/",
      "languageVersion": "2.14"
    },
    {
      "name": "vm_service",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/vm_service-13.0.0",
      "packageUri": "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "weather",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/weather-3.1.1",
      "packageUri": "lib/",
      "languageVersion": "2.17"
    },
    {
      "name": "web",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/web-0.5.1",
      "packageUri": "lib/",
      "languageVersion": "3.3"

```
    },
    {
      "name": "win32",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/win32-5.4.0",
      "packageUri": "lib/",
      "languageVersion": "3.3"
    },
    {
      "name": "xdg_directories",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/xdg_directories-
1.0.4",
      "packageUri": "lib/",
      "languageVersion": "3.0"
    },
    {
      "name": "xml",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/xml-6.5.0",
      "packageUri": "lib/",
      "languageVersion": "3.2"
    },
    {
      "name": "yaml",
      "rootUri":
"file:///C:/Users/DELL/AppData/Local/Pub/Cache/hosted/pub.dev/yaml-3.1.2",
      "packageUri": "lib/",
      "languageVersion": "2.19"
    },
    {
      "name": "agrobot",
      "rootUri": "../",
      "packageUri": "lib/",
      "languageVersion": "3.2"
    }
  ],
  "generated": "2024-04-08T02:57:04.974900Z",
  "generator": "pub",
  "generatorVersion": "3.3.0"
}
```

2. .idea:

libraries:

Dart_Package:

```
<component name="libraryTable">
  <library name="Dart Packages" type="DartPackagesLibraryType">
    <properties>
      <option name="packageNameToDirsMap">
        <entry key="async">
          <value>
            <list>
            <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/async-
2.11.0/lib" />
          </list>
        </value>
      </entry>
      <entry key="boolean_selector">
        <value>
          <list>
            <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/boolean_sel
ector-2.1.1/lib" />
          </list>
        </value>
      </entry>
      <entry key="characters">
        <value>
          <list>
            <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/characters-
1.3.0/lib" />
          </list>
        </value>
      </entry>
      <entry key="clock">
        <value>
          <list>
```

```
            <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/clock-
1.1.1/lib" />
          </list>
        </value>
      </entry>
      <entry key="collection">
        <value>
          <list>
            <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/collection-
1.18.0/lib" />
          </list>
        </value>
      </entry>
      <entry key="cupertino_icons">
        <value>
          <list>
            <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/cupertino_i
cons-1.0.6/lib" />
          </list>
        </value>
      </entry>
      <entry key="fake_async">
        <value>
          <list>
```

```
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/fake_async-
1.3.1/lib" />
            </list>
          </value>
        </entry>
        <entry key="flutter">
          <value>
            <list>
              <option
value="D:/Atharva/development/sdks/flutter/packages/flutter/lib" />
            </list>
          </value>
        </entry>
        <entry key="flutter_lints">
          <value>
            <list>
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/flutter_lints
-2.0.3/lib" />
            </list>
          </value>
        </entry>
        <entry key="flutter_test">
          <value>
            <list>
```

```
              <option
value="D:/Atharva/development/sdks/flutter/packages/flutter_test/lib" />
            </list>
          </value>
        </entry>
        <entry key="lints">
          <value>
            <list>
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/lints-
2.1.1/lib" />
            </list>
          </value>
        </entry>
        <entry key="matcher">
          <value>
            <list>
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/matcher-
0.12.16/lib" />
            </list>
          </value>
        </entry>
        <entry key="material_color_utilities">
          <value>
            <list>
```

```
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/material_co
lor_utilities-0.5.0/lib" />
            </list>
          </value>
        </entry>
        <entry key="meta">
          <value>
            <list>
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/meta-
1.10.0/lib" />
            </list>
          </value>
        </entry>
        <entry key="path">
          <value>
            <list>
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/path-
1.8.3/lib" />
            </list>
          </value>
        </entry>
        <entry key="sky_engine">
          <value>
            <list>
```

```
              <option
value="D:/Atharva/development/sdks/flutter/bin/cache/pkg/sky_engine/lib"  />
            </list>
          </value>
        </entry>
        <entry key="source_span">
          <value>
            <list>
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/source_span
-1.10.0/lib"  />
            </list>
          </value>
        </entry>
        <entry key="stack_trace">
          <value>
            <list>
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/stack_trace-
1.11.1/lib"  />
            </list>
          </value>
        </entry>
        <entry key="stream_channel">
          <value>
            <list>
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/stream_cha
nnel-2.1.2/lib" />
            </list>
          </value>
        </entry>
        <entry key="string_scanner">
          <value>
            <list>
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/string_scan
ner-1.2.0/lib" />
            </list>
          </value>
        </entry>
        <entry key="term_glyph">
          <value>
            <list>
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/term_glyph-
1.2.1/lib" />
            </list>
          </value>
        </entry>
        <entry key="test_api">
          <value>
            <list>
```

```
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/test_api-
0.6.1/lib" />
            </list>
          </value>
        </entry>
        <entry key="vector_math">
          <value>
            <list>
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/vector_mat
h-2.1.4/lib" />
            </list>
          </value>
        </entry>
        <entry key="web">
          <value>
            <list>
              <option
value="$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/web-
0.3.0/lib" />
            </list>
          </value>
        </entry>
      </option>
    </properties>
    <CLASSES>
    <root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/async-
2.11.0/lib" />

    <root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/boolean_
selector-2.1.1/lib" />

    <root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/character
s-1.3.0/lib" />

    <root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/clock-
1.1.1/lib" />

    <root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/collectio
n-1.18.0/lib" />

    <root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/cupertino
_icons-1.0.6/lib" />

    <root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/fake_asy
nc-1.3.1/lib" />

    <root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/flutter_li
nts-2.0.3/lib" />

    <root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/lints-
2.1.1/lib" />

    <root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/matcher-
0.12.16/lib" />
```

```
<root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/material_
color_utilities-0.5.0/lib" />

<root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/meta-
1.10.0/lib" />

<root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/path-
1.8.3/lib" />

<root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/source_s
pan-1.10.0/lib" />

<root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/stack_tra
ce-1.11.1/lib" />

<root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/stream_c
hannel-2.1.2/lib" />

<root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/string_sc
anner-1.2.0/lib" />

<root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/term_gly
ph-1.2.1/lib" />

<root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/test_api-
0.6.1/lib" />

<root
url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/vector_m
ath-2.1.4/lib" />
```

```
      <root url="file://$USER_HOME$/AppData/Local/Pub/Cache/hosted/pub.dev/web-0.3.0/lib" />
      <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/pkg/sky_engine/lib" />
      <root url="file://D:/Atharva/development/sdks/flutter/packages/flutter/lib" />
      <root url="file://D:/Atharva/development/sdks/flutter/packages/flutter_test/lib" />
    </CLASSES>
    <JAVADOC />
    <SOURCES />
  </library>
</component>
```

Dart_SDK:

```
<component name="libraryTable">
  <library name="Dart SDK">
    <CLASSES>
      <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-sdk/lib/async" />
      <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-sdk/lib/cli" />
      <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-sdk/lib/collection" />
      <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-sdk/lib/convert" />
      <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-sdk/lib/core" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/developer" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/ffi" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/html" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/indexed_db" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/io" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/isolate" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/js" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/js_interop" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/js_interop_unsafe" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/js_util" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/math" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/mirrors" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/svg" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/typed_data" />
    <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/web_audio" />
```

```
      <root url="file://D:/Atharva/development/sdks/flutter/bin/cache/dart-
sdk/lib/web_gl" />
    </CLASSES>
    <JAVADOC />
    <SOURCES />
  </library>
</component>
```

KotlinJavaRuntime:

```
<component name="libraryTable">
  <library name="KotlinJavaRuntime">
    <CLASSES>
      <root url="jar://$KOTLIN_BUNDLED$/lib/kotlin-stdlib.jar!/" />
      <root url="jar://$KOTLIN_BUNDLED$/lib/kotlin-reflect.jar!/" />
      <root url="jar://$KOTLIN_BUNDLED$/lib/kotlin-test.jar!/" />
    </CLASSES>
    <JAVADOC />
    <SOURCES>
      <root url="jar://$KOTLIN_BUNDLED$/lib/kotlin-stdlib-sources.jar!/" />
      <root url="jar://$KOTLIN_BUNDLED$/lib/kotlin-reflect-sources.jar!/" />
      <root url="jar://$KOTLIN_BUNDLED$/lib/kotlin-test-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

runConfigurations:

main_dart:

```
<component name="ProjectRunConfigurationManager">
  <configuration default="false" name="main.dart"
type="FlutterRunConfigurationType" factoryName="Flutter">
    <option name="filePath" value="$PROJECT_DIR$/lib/main.dart" />
    <method />
  </configuration>
</component>
```

misc:

```
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">
  <component name="ProjectRootManager">
    <output url="file://$PROJECT_DIR$/out" />
  </component>
</project>
```

modules:

```
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">
  <component name="ProjectModuleManager">
    <modules>
      <module fileurl="file://$PROJECT_DIR$/agrobot.iml"
filepath="$PROJECT_DIR$/agrobot.iml" />
```

```
    <module fileurl="file://$PROJECT_DIR$/android/agrobot_android.iml"
  filepath="$PROJECT_DIR$/android/agrobot_android.iml" />
    </modules>
   </component>
  </project>
```

workspace:

```
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">
  <component name="AutoImportSettings">
    <option name="autoReloadType" value="NONE" />
  </component>
  <component name="ChangeListManager">
    <list default="true" id="a5f0ad60-13ea-475c-a488-46a38e354b26"
name="Changes" comment="" />
    <option name="SHOW_DIALOG" value="false" />
    <option name="HIGHLIGHT_CONFLICTS" value="true" />
    <option name="HIGHLIGHT_NON_ACTIVE_CHANGELIST" value="false" />
    <option name="LAST_RESOLUTION" value="IGNORE" />
  </component>
  <component name="ProjectId" id="2b4TT8alk1sJzhm0dWFmp5yHIZO" />
  <component name="ProjectViewState">
    <option name="hideEmptyMiddlePackages" value="true" />
    <option name="showLibraryContents" value="true" />
  </component>
  <component name="PropertiesComponent">
```

```
    <property name="RunOnceActivity.OpenProjectViewOnStart" value="true" />
    <property name="RunOnceActivity.ShowReadmeOnStart" value="true" />
    <property name="RunOnceActivity.cidr.known.project.marker" value="true" />
    <property name="cidr.known.project.marker" value="true" />
    <property name="dart.analysis.tool.window.visible" value="false" />
    <property name="io.flutter.reload.alreadyRun" value="true" />
    <property name="settings.editor.selected.configurable"
value="AndroidSdkUpdater" />
    <property name="show.migrate.to.gradle.popup" value="false" />
  </component>
  <component name="RunManager">
    <configuration name="main.dart" type="FlutterRunConfigurationType"
factoryName="Flutter">
      <option name="filePath" value="$PROJECT_DIR$/lib/main.dart" />
      <method v="2" />
    </configuration>
  </component>
  <component name="SpellCheckerSettings" RuntimeDictionaries="0" Folders="0"
CustomDictionaries="0" DefaultDictionary="application-level"
UseSingleDictionary="true" transferred="true" />
  <component name="TaskManager">
    <task active="true" id="Default" summary="Default task">
      <changelist id="a5f0ad60-13ea-475c-a488-46a38e354b26" name="Changes"
comment="" />
      <created>1705473365075</created>
      <option name="number" value="Default" />
      <option name="presentableId" value="Default" />
```

```
     <updated>1705473365075</updated>
    </task>
    <servers />
  </component>
</project>
```

3. android

gradle:

```
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-7.5-all.zip
```

app:

AndroidManifest:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- The INTERNET  permission is required for development. Specifically,
         the Flutter tool needs it to communicate with the running application
         to allow setting breakpoints, to provide hot reload, etc.
    -->
    <uses-permission android:name="android.permission.INTERNET"/>
</manifest>
```

main:

AndroidManifest:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.agrobot">


    <!-- Required permissions for Location Access -->
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />


    <!-- Required permissions for battery access -->
    <uses-permission android:name="android.permission.BATTERY_STATS" />
    <uses-permission android:name="android.permission.INTERNET" />


    <!-- Permission for notification -->
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />


    <!-- Tell Google Play Store that your app uses Bluetooth LE
     Set android:required="true" if bluetooth is necessary -->
    <uses-feature android:name="android.hardware.bluetooth_le"
android:required="false" />


    <!-- New Bluetooth permissions in Android 12 -->
    <uses-permission android:name="android.permission.BLUETOOTH_SCAN"/>
    <uses-permission
android:name="android.permission.BLUETOOTH_CONNECT" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />


    <!-- legacy for Android 11 or lower -->
    <uses-permission android:name="android.permission.BLUETOOTH"
android:maxSdkVersion="31" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"
android:maxSdkVersion="31" />



    <application
        android:label="agrobot"
        android:name="${applicationName}"
        android:icon="@mipmap/launcher_icon">


        <receiver
android:name="com.dexterous.flutterlocalnotifications.ScheduledNotificationBootR
eceiver"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
                <action
android:name="android.intent.action.MY_PACKAGE_REPLACED" />
            </intent-filter>
        </receiver>
        <!-- Specify the custom notification service -->
        <service
android:name="com.dexterous.flutterlocalnotifications.services.NotificationService"
android:permission="android.permission.BIND_JOB_SERVICE"
android:exported="false" />


        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:launchMode="singleTop"
            android:theme="@style/LaunchTheme"

android:configChanges="orientation|keyboardHidden|keyboard|screenSize|smallestS
creenSize|locale|layoutDirection|fontScale|screenLayout|density|uiMode"
            android:hardwareAccelerated="true"
            android:windowSoftInputMode="adjustResize">
            <!-- Specifies an Android theme to apply to this Activity as soon as
                 the Android process has started. This theme is visible to the user
                 while the Flutter UI initializes. After that, this theme continues
                 to determine the Window background behind the Flutter UI. -->
            <meta-data
              android:name="io.flutter.embedding.android.NormalTheme"
              android:resource="@style/NormalTheme"
              />
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
          </intent-filter>
        </activity>
        <!-- Don't delete the meta-data below.
             This is used by the Flutter tool to generate GeneratedPluginRegistrant.java  --
>
        <meta-data
          android:name="flutterEmbedding"
          android:value="2" />
    </application>
</manifest>
```

java:

MainActivity:

```
package com.example.agrobot;
import io.flutter.embedding.android.FlutterActivity;
public class MainActivity  extends FlutterActivity  {
}
```

Io:

GeneratedPluginRegistrant:

```
package io.flutter.plugins;

import androidx.annotation.Keep;
import androidx.annotation.NonNull;
import io.flutter.Log;
import io.flutter.embedding.engine.FlutterEngine;


/**
 * Generated file. Do not edit.
 * This file is generated by the Flutter tool based on the
 * plugins that support the Android platform.
 */
@Keep
public final class GeneratedPluginRegistrant {
  private static final String TAG = "GeneratedPluginRegistrant";
  public static void registerWith(@NonNull FlutterEngine flutterEngine) {
    try {
      flutterEngine.getPlugins().add(new
dev.fluttercommunity.plus.battery.BatteryPlusPlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin battery_plus,
dev.fluttercommunity.plus.battery.BatteryPlusPlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new
com.mr.flutter.plugin.filepicker.FilePickerPlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin file_picker,
com.mr.flutter.plugin.filepicker.FilePickerPlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new
io.flutter.plugins.firebase.auth.FlutterFirebaseAuthPlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin firebase_auth,
io.flutter.plugins.firebase.auth.FlutterFirebaseAuthPlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new
io.flutter.plugins.firebase.core.FlutterFirebaseCorePlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin firebase_core,
io.flutter.plugins.firebase.core.FlutterFirebaseCorePlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new
io.flutter.plugins.firebase.storage.FlutterFirebaseStoragePlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin firebase_storage,
io.flutter.plugins.firebase.storage.FlutterFirebaseStoragePlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new
com.pauldemarco.flutter_blue.FlutterBluePlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin flutter_blue,
com.pauldemarco.flutter_blue.FlutterBluePlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new
com.lib.flutter_blue_plus.FlutterBluePlusPlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin flutter_blue_plus,
com.lib.flutter_blue_plus.FlutterBluePlusPlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new
io.github.edufolly.flutterbluetoothserial.FlutterBluetoothSerialPlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin flutter_bluetooth_serial,
io.github.edufolly.flutterbluetoothserial.FlutterBluetoothSerialPlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new
com.dexterous.flutterlocalnotifications.FlutterLocalNotificationsPlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin flutter_local_notifications,
com.dexterous.flutterlocalnotifications.FlutterLocalNotificationsPlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new
io.flutter.plugins.flutter_plugin_android_lifecycle.FlutterAndroidLifecyclePlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin flutter_plugin_android_lifecycle,
io.flutter.plugins.flutter_plugin_android_lifecycle.FlutterAndroidLifecyclePlugin",
e);
    }
    try {
      flutterEngine.getPlugins().add(new
io.github.ponnamkarthik.toast.fluttertoast.FlutterToastPlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin fluttertoast,
io.github.ponnamkarthik.toast.fluttertoast.FlutterToastPlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new
com.baseflow.geocoding.GeocodingPlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin geocoding_android,
com.baseflow.geocoding.GeocodingPlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new
com.baseflow.geolocator.GeolocatorPlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin geolocator_android,
com.baseflow.geolocator.GeolocatorPlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new
io.flutter.plugins.imagepicker.ImagePickerPlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin image_picker_android,
io.flutter.plugins.imagepicker.ImagePickerPlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new
io.flutter.plugins.pathprovider.PathProviderPlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin path_provider_android,
io.flutter.plugins.pathprovider.PathProviderPlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new
com.baseflow.permissionhandler.PermissionHandlerPlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin permission_handler_android,
com.baseflow.permissionhandler.PermissionHandlerPlugin", e);
    }
    try {
      flutterEngine.getPlugins().add(new app.rive.rive.RivePlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin rive_common, app.rive.rive.RivePlugin",
e);
    }
    try {
      flutterEngine.getPlugins().add(new
io.flutter.plugins.sharedpreferences.SharedPreferencesPlugin());
    } catch (Exception e) {
      Log.e(TAG, "Error registering plugin shared_preferences_android,
io.flutter.plugins.sharedpreferences.SharedPreferencesPlugin", e);
    }
  }
```

```
}
```

Res:

launch_background:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Modify this file to customize your launch splash screen -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@android:color/white" />

    <!-- You can insert your own image assets here -->
    <!-- <item>
        <bitmap
            android:gravity="center"
            android:src="@mipmap/launch_image" />
    </item> -->
</layer-list>
```

styles:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- Theme applied to the Android Window while the process is starting when the
OS's Dark Mode setting is off -->
    <style name="LaunchTheme"
parent="@android:style/Theme.Light.NoTitleBar">
        <!-- Show a splash screen on the activity. Automatically removed when
             the Flutter engine draws its first frame -->
        <item
name="android:windowBackground">@drawable/launch_background</item>
    </style>
    <!-- Theme applied to the Android Window as soon as the process has started.
         This theme determines the color of the Android Window while your
         Flutter UI initializes, as well as behind your Flutter UI while its
         running.

         This Theme is only used starting with V2 of Flutter's Android embedding. -->
    <style name="NormalTheme"
parent="@android:style/Theme.Light.NoTitleBar">
        <item name="android:windowBackground">?android:colorBackground</item>
    </style>
</resources>
```

debug:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- The INTERNET permission is required for development. Specifically,
         the Flutter tool needs it to communicate with the running application
         to allow setting breakpoints, to provide hot reload, etc.
    -->
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.BLUETOOTH_SCAN"/>
    <uses-permission
android:name="android.permission.BLUETOOTH_CONNECT" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

```
    <!-- legacy for Android 11 or lower -->
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
</manifest>
```

profile:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- The INTERNET  permission is required for development. Specifically,
         the Flutter tool needs it to communicate with the running application
         to allow setting breakpoints, to provide hot reload, etc.
    -->
    <uses-permission android:name="android.permission.INTERNET"/>
</manifest>
```

Google-services.json

```
{
  "project_info": {
    "project_number": "989810940058",
    "project_id": "agrobot-project",
    "storage_bucket": "agrobot-project.appspot.com"
  },
  "client": [
    {
      "client_info": {
```

```
      "mobilesdk_app_id": "1:989810940058:android:5a5dfb4d0c9d73b8ad2723",
      "android_client_info": {
        "package_name": "com.example.agrobot"
      }
    },
    "oauth_client": [],
    "api_key": [
      {
        "current_key": "AIzaSyAmv79_QQnO4sVo-8LSlU1NDE64AEPvoOE"
      }
    ],
    "services": {
      "appinvite_service": {
        "other_platform_oauth_client": []
      }
    }
  },
  {
    "client_info": {
      "mobilesdk_app_id": "1:989810940058:android:4db1790d50cca0d5ad2723",
      "android_client_info": {
        "package_name": "com.example.flutterbasics"
      }
    },
    "oauth_client": [],
      "api_key": [
        {
          "current_key": "AIzaSyAmv79_QQnO4sVo-8LSlU1NDE64AEPvoOE"
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_client": []
        }
      }
    }
  ],
  "configuration_version": "1"
}
```

4. build

app:

BuildConfig:

```
/** Automatically generated file. DO NOT MODIFY
 */
package com.example.agrobot;

public final class BuildConfig {
  public static final boolean DEBUG = Boolean.parseBoolean("true");
  public static final String APPLICATION_ID = "com.example.agrobot";
  public static final String BUILD_TYPE = "debug";
  public static final int VERSION_CODE = 1;
  public static final String VERSION_NAME = "1.0.0";
}
```

Output-medata.json

```
{
  "version": 3,
  "artifactType": {
    "type": "APK",
    "kind": "Directory"
  },
  "applicationId": "com.example.agrobot",
  "variantName": "debug",
  "elements": [
    {
      "type": "SINGLE",
      "filters": [],
      "attributes": [],
      "versionCode": 1,
      "versionName": "1.0.0",
      "outputFile": "app-debug.apk"
    }
  ],
  "elementType": "File"
}
```

RES:

Values:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="gcm_defaultSenderId"
translatable="false">989810940058</string>
    <string name="google_api_key"
translatable="false">AIzaSyAmv79_QQnO4sVo-
8LSlU1NDE64AEPvoOE</string>
    <string name="google_app_id"
translatable="false">1:989810940058:android:5a5dfb4d0c9d73b8ad2723</string>
    <string name="google_crash_reporting_api_key"
translatable="false">AIzaSyAmv79_QQnO4sVo-
8LSlU1NDE64AEPvoOE</string>
    <string name="google_storage_bucket" translatable="false">agrobot-
project.appspot.com</string>
    <string name="project_id" translatable="false">agrobot-project</string>
</resources>
```

Battery.java

```
/**
 * Automatically generated file. DO NOT MODIFY
 */
package dev.fluttercommunity.plus.battery;

public final class BuildConfig {
  public static final boolean DEBUG = Boolean.parseBoolean("true");
  public static final String LIBRARY_PACKAGE_NAME =
"dev.fluttercommunity.plus.battery";
  public static final String BUILD_TYPE = "debug";
}
```

output-medata.json

```
{
  "version": 3,
  "artifactType": {
    "type": "APK",
    "kind": "Directory"
  },
  "applicationId": "com.example.agrobot",
  "variantName": "debug",
  "elements": [
    {
      "type": "SINGLE",
      "filters": [],
      "attributes": [],
      "versionCode": 1,
      "versionName": "1.0.0",
      "outputFile": "app-debug.apk"
    }
  ],
  "elementType": "File"
}
```

File picker:

BuildConfig.java:

```
/**
 * Automatically generated file. DO NOT MODIFY
 */
package com.mr.flutter.plugin.filepicker;


public final class BuildConfig {
  public static final boolean DEBUG = Boolean.parseBoolean("true");
  public static final String LIBRARY_PACKAGE_NAME =
"com.mr.flutter.plugin.filepicker";
  public static final String BUILD_TYPE = "debug";
}
```

AndroidManifest:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mr.flutter.plugin.filepicker" >

    <uses-sdk
        android:minSdkVersion="16"
        android:targetSdkVersion="16" />

    <uses-permission
        android:name="android.permission.READ_EXTERNAL_STORAGE"
        android:maxSdkVersion="32" />
    <queries>
        <intent>
            <action android:name="android.intent.action.GET_CONTENT" />
            <data android:mimeType="*/*" />
        </intent>
    </queries>
</manifest>
```

firebase_auth:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="io.flutter.plugins.firebase.auth" >

    <uses-sdk
        android:minSdkVersion="19"
        android:targetSdkVersion="19" />

    <application>
        <service
android:name="com.google.firebase.components.ComponentDiscoveryService" >
            <meta-data

android:name="com.google.firebase.components:io.flutter.plugins.firebase.auth.Flut
terFirebaseAuthRegistrar"
                android:value="com.google.firebase.components.ComponentRegistrar" />
        </service>
    </application>

</manifest>
```

5. lib

reusable_widget:

```
import 'package:flutter/material.dart';
Image logoWidget(String imageName,double h,double w){
  return Image.asset(
    imageName,
    fit: BoxFit.fitWidth,
    width: w,
    height: h,
    color: Colors.white,
  );
}

TextField reusableTextField(String text,IconData icon, bool isPasswordType,
TextEditingController controller){
  return TextField(
    controller: controller,
    obscureText: isPasswordType,
    enableSuggestions: !isPasswordType,
    autocorrect: !isPasswordType,
    cursorColor: Colors.white,
    style: TextStyle(color: Colors.white.withOpacity(0.9)),
    decoration: InputDecoration(
      prefixIcon: Icon(
        icon,
        color:Colors.white70
      ),
    labelText: text,
    labelStyle: TextStyle(color: Colors.white.withOpacity(0.9)),
    filled: true,
    floatingLabelBehavior: FloatingLabelBehavior.never,
    fillColor: Colors.white.withOpacity(0.3),
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(30.0),
      borderSide: const BorderSide(width: 0,style:BorderStyle.none),
    ),
    ),
    keyboardType:
isPasswordType?TextInputType.visiblePassword:TextInputType.emailAddress,
  );
}


ButtonStyle buttonSecondary(Color color1, bool includeBorder,) {
  return ElevatedButton.styleFrom(
    minimumSize: const Size(300, 60),
    shape: RoundedRectangleBorder(
      borderRadius: const BorderRadius.all(Radius.circular(25)),
      side: includeBorder ? const BorderSide(color: Color(0xFF02AA6D),  width: 2) :
BorderSide.none,
    ),
    backgroundColor: color1,
  );
}

Container signinSignupScreenButton(BuildContext  context,bool isLogin,Function
onTap){

  return Container(
    width: MediaQuery.of(context).size.width,
    height: 50,
    margin: const EdgeInsets.fromLTRB(0,  10, 0, 20),
    child: ElevatedButton(
      style: buttonSecondary(Colors.white,false),
      onPressed: (){
        onTap();
      },
      child: Text(isLogin?'Login':'SignUp',style:  const TextStyle(color:
Colors.black87,fontWeight:  FontWeight.bold,fontSize:  16),),
    ),
  );
}
```

Battery_screen.dart

```
import 'dart:async';
import 'package:battery_plus/battery_plus.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_local_notifications/flutter_local_notifications.dart';
import 'package:permission_handler/permission_handler.dart';

class BatteryScreen extends StatefulWidget {
  final int batteryLevel;
  const BatteryScreen({super.key, required this.batteryLevel});

  @override
  State<BatteryScreen> createState() => _BatteryScreenState();
}

class _BatteryScreenState extends State<BatteryScreen> {
  int b=0;
  final FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin =
      FlutterLocalNotificationsPlugin();
  late Battery battery;
  int level = 100;
  BatteryState batteryState = BatteryState.full;
  late Timer timer;
  late StreamSubscription streamSubscription;

  @override
```

```
void initState() {
  b = widget.batteryLevel;
  super.initState();
  battery = Battery();
  initializeNotifications();
  requestNotificationPermissions();
  getBatteryPercentage();
  getBatteryState();
  timer = Timer.periodic(const Duration(seconds: 5), (timer) {
    getBatteryPercentage();
  });
  SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
}

void initializeNotifications() {
  const AndroidInitializationSettings initializationSettingsAndroid =
       AndroidInitializationSettings('@mipmap/ic_launcher');
  const InitializationSettings initializationSettings =
      InitializationSettings(android: initializationSettingsAndroid);
  flutterLocalNotificationsPlugin.initialize(initializationSettings);

  const AndroidNotificationChannel channel = AndroidNotificationChannel(
    'battery_channel',
    'Battery Notification',
    'Shows battery low notification',
      importance: Importance.max,
      playSound: true, // Enable sound for this channel
    );

    flutterLocalNotificationsPlugin
        .resolvePlatformSpecificImplementation<
            AndroidFlutterLocalNotificationsPlugin>()
        ?.createNotificationChannel(channel);

    // Set custom sound
    const AndroidNotificationDetails androidPlatformChannelSpecifics =
        AndroidNotificationDetails(
      'battery_channel',
      'Battery Notification',
      'Shows battery low notification',
      importance: Importance.max,
      priority: Priority.high,
      sound: RawResourceAndroidNotificationSound('assets/Sound/notification.mp3'),
// Specify the custom sound file
    );

    // ignore: unused_local_variable
    const NotificationDetails platformChannelSpecifics =
        NotificationDetails(android: androidPlatformChannelSpecifics);
  }
```

```
Future<void> requestNotificationPermissions() async {
 final status = await Permission.notification.request();
 if (status.isDenied) {
  // Handle the case if the user has denied permissions
 }
}

Future<void> showNotification() async {
 const AndroidNotificationDetails androidPlatformChannelSpecifics =
    AndroidNotificationDetails(
  'battery_channel',
  'Battery Notification',
  'Shows battery low notification',
  importance: Importance.max,
  priority: Priority.high,
 );
 const NotificationDetails platformChannelSpecifics =
    NotificationDetails(android: androidPlatformChannelSpecifics);

 await flutterLocalNotificationsPlugin.show(
  0,
  'Battery Low',
  'Please charge the robot!',
  platformChannelSpecifics,
  payload: 'battery_low',
  );
}

void getBatteryPercentage() async {
  int batteryLevel = b; // Manually set to 20 for testing
  setState(() {
    level = batteryLevel;
    if (level < 30) {
      showNotification();
    }
  });
}

void getBatteryState() {
  streamSubscription = battery.onBatteryStateChanged.listen((state) {
    setState(() {
      batteryState = state;
    });
  });
}

@override
void dispose() {
  SystemChrome.setPreferredOrientations([DeviceOrientation.landscapeLeft]);
  super.dispose();
  streamSubscription.cancel();
  timer.cancel();
}

Widget buildBattery(BatteryState state) {
  Color batteryColor = Colors.green;
  IconData batteryIcon = Icons.battery_full;

  if (level < 30) {
    batteryColor = Colors.red;
    batteryIcon = Icons.battery_alert;
  } else if (level < 80) {
    batteryColor = Colors.yellow;
    batteryIcon = Icons.battery_std;
  } else if (state == BatteryState.charging) {
    batteryColor = Colors.blue;
    batteryIcon = Icons.battery_charging_full;
  }

  return SizedBox(
    width: 200,
    height: 200,
    child: Icon(
      batteryIcon,
      size: 200,
      color: batteryColor,
    ),
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: SizedBox(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            const Text(
              'Robot Battery',
              style: TextStyle(fontSize: 50),
            ),
            buildBattery(batteryState),
            Text(
              '$level %',
              style: const TextStyle(color: Colors.black, fontSize: 25),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

Bluetooth.dart:

```
import 'dart:convert';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';
import 'package:fluttertoast/fluttertoast.dart';

class BluetoothApp extends StatefulWidget {
 const BluetoothApp({super.key});

 @override
 State<BluetoothApp> createState() => _BluetoothAppState();
}

class _BluetoothAppState extends State<BluetoothApp> {
 List<BluetoothDevice> devices = [];
 BluetoothConnection? connection;

 @override
```

```
  void initState() {
    super.initState();
    _startScan();
  }

  void _startScan() {
    FlutterBluetoothSerial.instance.getBondedDevices().then((List<BluetoothDevice>
bondedDevices) {
      setState(() {
        devices.addAll(bondedDevices);
      });
    });

    FlutterBluetoothSerial.instance.startDiscovery().listen((BluetoothDiscoveryResult
result) {
      setState(() {
        if (!devices.contains(result.device)) {
          devices.add(result.device);
        }
      });
    });
  }

  void _connectToDevice(BluetoothDevice  device) async {
    try {
      BluetoothConnection connection = await
BluetoothConnection.toAddress(device.address);
      setState(() {
        this.connection = connection;
      });
      if (kDebugMode) {
        print('Connected to ${device.name}');
      }
      Fluttertoast.showToast(
        msg: 'Connected to ${device.name}',
        toastLength: Toast.LENGTH_SHORT,
        gravity: ToastGravity.BOTTOM,
        backgroundColor: Colors.green,
        textColor: Colors.white,
      );
    } catch (e) {
      if (kDebugMode) {
        print('Error connecting to device: $e');
      }
      Fluttertoast.showToast(
        msg: 'Failed to connect to ${device.name}',
        toastLength: Toast.LENGTH_SHORT,
        gravity: ToastGravity.BOTTOM,
        backgroundColor: Colors.red,
        textColor: Colors.white,
      );
```

```
  }
}

void _sendMessageToConnectedDevice(String message) async {
  if (connection != null) {
    try {
      connection!.output.add(utf8.encode(message));
      await connection!.output.allSent;
      Fluttertoast.showToast(
        msg: 'Message sent to device',
        toastLength: Toast.LENGTH_SHORT,
        gravity: ToastGravity.BOTTOM,
        backgroundColor: Colors.blue,
        textColor: Colors.white,
      );
    } catch (e) {
      if (kDebugMode) {
        print('Error sending message: $e');
      }
      Fluttertoast.showToast(
        msg: 'Failed to send message to device',
        toastLength: Toast.LENGTH_SHORT,
        gravity: ToastGravity.BOTTOM,
        backgroundColor: Colors.red,
        textColor: Colors.white,
      );
    }
  } else {
    Fluttertoast.showToast(
      msg: 'No device connected',
      toastLength: Toast.LENGTH_SHORT,
      gravity: ToastGravity.BOTTOM,
      backgroundColor: Colors.red,
      textColor: Colors.white,
    );
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Bluetooth Classic Example'),
    ),
    body: Column(
      children: [
        Expanded(
          child: ListView.builder(
            itemCount: devices.length,
            itemBuilder: (context, index) {
```

```
        return ListTile(
          title: Text(devices[index].name.toString()),
          onTap: () {
            _connectToDevice(devices[index]);
          },
        );
      },
    ),
  ),
  ElevatedButton(
    onPressed: () {
      _sendMessageToConnectedDevice("0xP"); // For Plowing
    },
    child: Text('Plowing'),
  ),
  ElevatedButton(
    onPressed: () {
      _sendMessageToConnectedDevice("0xS"); // For Seeding
    },
    child: Text('Seeding'),
  ),
  ElevatedButton(
    onPressed: () {
      _sendMessageToConnectedDevice("0xW"); // For Watering
    },
          child: Text('Watering'),
        ),
        ElevatedButton(
          onPressed: () {
            _sendMessageToConnectedDevice("0xA"); // For All Above
          },
          child: Text('All Above'),
        ),
        ElevatedButton(
          onPressed: () {
            _sendMessageToConnectedDevice("0xR"); // For Spraying
          },
          child: Text('Spraying'),
        ),
      ],
    ),
  );
 }
}
```

Bluetooth_controller.dart

```
import 'package:flutter_blue_plus/flutter_blue_plus.dart';
import 'package:get/get.dart';
class BluetoothController extends GetxController {
 Stream<List<ScanResult>>? _scanResults;
```

```
  Future<void> scanDevices() async {
    // Starting scan
    await FlutterBluePlus.startScan(timeout: const Duration(seconds: 10));
    // Set scanResults stream
    _scanResults = FlutterBluePlus.scanResults;
    // Notify listeners that scan results are available
    update();
  }
  Stream<List<ScanResult>>? get scanResults => _scanResults;
}
```

ControlScreen.dart

```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';

class ControlScreen extends StatelessWidget {
  final BluetoothConnection connection;

  ControlScreen(this.connection);

  void _sendMessageToConnectedDevice(String message) async {
    try {
      connection.output.add(utf8.encode(message));
      await connection.output.allSent;
      print('Message sent to device: $message');
    } catch (e) {
      print('Error sending message: $e');
      // Handle message sending error
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Control Screen'),
      ),
      body: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          ElevatedButton(
            onPressed: () {
              _sendMessageToConnectedDevice("0xP"); // For Plowing
            },
            child: Text('Plowing'),
          ),
          ElevatedButton(
            onPressed: () {
```

```
          _sendMessageToConnectedDevice("0xS"); // For Seeding
        },
        child: Text('Seeding'),
      ),
      ElevatedButton(
        onPressed: () {
          _sendMessageToConnectedDevice("0xW"); // For Watering
        },
        child: Text('Watering'),
      ),
      ElevatedButton(
        onPressed: () {
          _sendMessageToConnectedDevice("0xA"); // For All Above
        },
        child: Text('All Above'),
      ),
      ElevatedButton(
        onPressed: () {
          _sendMessageToConnectedDevice("0xR"); // For Spraying
        },
        child: Text('Spraying'),
      ),
    ],
  ),
);
```

```
  }
}
```

Crop_screen.dart

```
import 'dart:io';

import 'package:animated_text_kit/animated_text_kit.dart';
import 'package:awesome_dialog/awesome_dialog.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_spinkit/flutter_spinkit.dart';
import 'package:agrobot/constants/constants.dart';
import 'package:image_picker/image_picker.dart';

import '../services/api_service.dart';

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<HomePage> {
 @override
void initState() {
 super.initState();
 SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
}

@override
void dispose() {
 SystemChrome.setPreferredOrientations([DeviceOrientation.landscapeLeft]);
 super.dispose();
}

final apiService = ApiService();
File? _selectedImage;
String diseaseName = '';
String diseasePrecautions = '';
bool detecting = false;
bool precautionLoading = false;

Future<void> _pickImage(ImageSource source) async {
 final pickedFile =
    await ImagePicker().pickImage(source: source, imageQuality: 50);
 if (pickedFile != null) {
   setState(() {
    _selectedImage = File(pickedFile.path);
    });
  }
}

detectDisease() async {
  setState(() {
    detecting = true;
  });
  try {
    diseaseName =
        await apiService.sendImageToGPT4Vision(image: _selectedImage!);
  } catch (error) {
    _showErrorSnackBar(error);
  } finally {
    setState(() {
      detecting = false;
    });
  }
}

showPrecautions() async {
  setState(() {
    precautionLoading = true;
  });
  try {
    if (diseasePrecautions == '') {
      diseasePrecautions =
          await apiService.sendMessageGPT(diseaseName: diseaseName);
    }
    _showSuccessDialog(diseaseName, diseasePrecautions);
  } catch (error) {
    _showErrorSnackBar(error);
  } finally {
    setState(() {
      precautionLoading = false;
    });
  }
}

void _showErrorSnackBar(Object error) {
  ScaffoldMessenger.of(context).showSnackBar(SnackBar(
    content: Text(error.toString()),
    backgroundColor: Colors.red,
  ));
}

void _showSuccessDialog(String title, String content) {
  AwesomeDialog(
    context: context,
    dialogType: DialogType.success,
    animType: AnimType.rightSlide,
    title: title,
    desc: content,
    btnOkText: 'Got it',
    btnOkColor: themeColor,
    btnOkOnPress: () {},
  ).show();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Column(
      children: <Widget>[
        const SizedBox(height: 20),
        Stack(
          children: [
            Container(
              height: MediaQuery.of(context).size.height * 0.23,
              width: double.infinity,
              decoration: BoxDecoration(
                borderRadius: const BorderRadius.only(
                  // Top right corner
                  bottomLeft: Radius.circular(50.0), // Bottom right corner
                ),
                color: themeColor,
              ),
            ),
            Container(
              height: MediaQuery.of(context).size.height  * 0.2,
              width: double.infinity,
              decoration: BoxDecoration(
                color: Colors.white,
                borderRadius: const BorderRadius.only(
                  // Top right corner
                  bottomLeft: Radius.circular(50.0), // Bottom right corner
                ),
                boxShadow: [
                  BoxShadow(
                    color: Colors.black.withOpacity(0.1),
                    // Shadow color with some transparency
                    spreadRadius: 1,
                    // Extend the shadow to all sides equally
                    blurRadius: 5,
                    // Soften the shadow
                    offset: const Offset(2, 2), // Position of the shadow
                  ),
                ],
              ),
              child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: <Widget>[
              ElevatedButton(
                onPressed: () {
                  _pickImage(ImageSource.gallery);
                },
                style: ElevatedButton.styleFrom(
                  backgroundColor: themeColor,
                ),
                child: Row(
                  mainAxisSize: MainAxisSize.min,
                  children: [
                    Text(
                      'OPEN GALLERY',
                      style: TextStyle(color: textColor),
                    ),
                    const SizedBox(width: 10),
                    Icon(
                      Icons.image,
                      color: textColor,
                    )
                  ],
                ),
              ),
              ElevatedButton(
                onPressed: () {
                  _pickImage(ImageSource.camera);
                },
                style: ElevatedButton.styleFrom(
                  backgroundColor: themeColor,
                ),
                child: Row(
                  mainAxisSize: MainAxisSize.min,
                  children: [
                    Text('START CAMERA',
                        style: TextStyle(color: textColor)),
                    const SizedBox(width: 10),
                    Icon(Icons.camera_alt, color: textColor)
                  ],
                ),
              ),
            ],
          ),
        ),
      ],
    ),
    _selectedImage == null
        ? Container(
            height: MediaQuery.of(context).size.height * 0.5,
            child: Image.asset('assets/images/pick1.png'),
          )
        : Expanded(
            child: Container(
              width: double.infinity,
              decoration:
                  BoxDecoration(borderRadius: BorderRadius.circular(20)),
              padding: const EdgeInsets.all(20),
              child: ClipRRect(
                borderRadius: BorderRadius.circular(20),
                child: Image.file(
                  _selectedImage!,
                  fit: BoxFit.cover,
                ),
              ),
            ),
          ),
    if (_selectedImage != null)
      detecting
          ? SpinKitWave(
              color: themeColor,
              size: 30,
            )
          : Container(
              width: double.infinity,
    padding:
      const EdgeInsets.symmetric(vertical: 0, horizontal: 20),
    child: ElevatedButton(
     style: ElevatedButton.styleFrom(
      backgroundColor: themeColor,
      padding: const EdgeInsets.symmetric(
        horizontal: 30, vertical: 15),
      // Set some horizontal and vertical padding
      shape: RoundedRectangleBorder(
       borderRadius:
         BorderRadius.circular(15), // Rounded corners
      ),
     ),
     onPressed: () {
      detectDisease();
     },
     child: const Text(
      'DETECT',
      style: TextStyle(
       color: Colors.white, // Set the text color to white
       fontSize: 16, // Set the font size
       fontWeight:
         FontWeight.bold, // Set the font weight to bold
      ),
     ),
              ),
             ),
        if (diseaseName != '')
         Column(
          children: [
           Container(
            height: MediaQuery.of(context).size.height * 0.2,
            padding: EdgeInsets.symmetric(vertical: 0, horizontal: 20),
            child: Column(
             mainAxisAlignment: MainAxisAlignment.center,
             children: [
              DefaultTextStyle(
               style: const TextStyle(
                 color: Colors.black,
                 fontWeight: FontWeight.w700,
                 fontSize: 16),
               child: AnimatedTextKit(
                 isRepeatingAnimation: false,
                 repeatForever: false,
                 displayFullTextOnTap: true,
                 totalRepeatCount: 1,
                 animatedTexts: [
                  TyperAnimatedText(
                   diseaseName.trim(),
                  ),
          ]),
       )
      ],
     ),
    ),
    precautionLoading
      ? const SpinKitWave(
        color: Colors.blue,
        size: 30,
       )
      : ElevatedButton(
        style: ElevatedButton.styleFrom(
         backgroundColor: Colors.blue,
         padding: const EdgeInsets.symmetric(
           horizontal: 30, vertical: 15),
        ),
        onPressed: () {
         showPrecautions();
        },
        child: Text(
         'PRECAUTION',
         style: TextStyle(
          color: textColor,
         ),
        ),
                        ),
                    ],
                  ),
                const SizedBox(height: 30),
              ],
            ),
          );
        }
      }
```

Device_list_screen.dart:

```
import 'package:agrobot/Screens/home.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';
import 'package:permission_handler/permission_handler.dart';

class DeviceListScreen extends StatefulWidget {
  const DeviceListScreen({super.key});

  @override
  State<DeviceListScreen> createState() => _DeviceListScreenState();
}

class _DeviceListScreenState extends State<DeviceListScreen> {
```

```
List<BluetoothDevice>  devices = [];
BluetoothConnection? connection;

@override
void initState() {
  super.initState();
  SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
  _requestPermissions();
}

@override
void dispose() {
  SystemChrome.setPreferredOrientations([DeviceOrientation.landscapeLeft]);
  super.dispose();
}

Future<void> _requestPermissions() async {
  // Request Bluetooth and location permissions
  Map<Permission, PermissionStatus> statuses = await [
    Permission.bluetooth,
    Permission.location,
  ].request();

  // Check if permissions are granted
  if (statuses[Permission.bluetooth] != PermissionStatus.granted ||
        statuses[Permission.location] != PermissionStatus.granted) {
          debugPrint(">>>>>>>>>>>Permission Not Granted");
      return;
    }

    // Permissions are granted, start scanning for devices
    _startScan();
  }

  void _startScan() {
    FlutterBluetoothSerial.instance.getBondedDevices().then((List<BluetoothDevice>
bondedDevices) {
      setState(() {
        devices.addAll(bondedDevices);
      });
    });

    FlutterBluetoothSerial.instance.startDiscovery().listen((BluetoothDiscoveryResult
result) {
      setState(() {
        if (!devices.contains(result.device)) {
          devices.add(result.device);
        }
      });
    });
  }
  void _connectToDevice(BluetoothDevice device) async {
    try {
      BluetoothConnection connection = await
BluetoothConnection.toAddress(device.address);
      setState(() {
        this.connection = connection;
      });
      Navigator.pushReplacement(
        // ignore: use_build_context_synchronously
        context,
        MaterialPageRoute(builder: (context)
=> ResponsiveJoystickExampleApp(connection: connection)),
      );
    } catch (e) {
      debugPrint('Error connecting to device: $e');
      // Handle connection error
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Device List'),
      ),
      body: ListView.builder(
        itemCount: devices.length,
        itemBuilder: (context, index) {
          return ListTile(
            title: Text(devices[index].name.toString()),
            onTap: () {
              _connectToDevice(devices[index]);
            },
          );
        },
      ),
    );
  }
}
```

Display.dart

```
import 'dart:typed_data';

import 'package:flutter/material.dart';
import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';

void main() {
  runApp(MyApp());
}
```

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Bluetooth Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: BluetoothScreen(),
    );
  }
}

class BluetoothScreen extends StatefulWidget {
  @override
  _BluetoothScreenState createState() => _BluetoothScreenState();
}

class _BluetoothScreenState extends State<BluetoothScreen> {
  BluetoothConnection? connection;

  @override
  void initState() {
    super.initState();
    connectToDevice();
  }


  Future<void> connectToDevice() async {
    List<BluetoothDevice> devices = [];
    try {
      devices = await FlutterBluetoothSerial.instance.getBondedDevices();
      // Assuming you have a bonded device and want to connect to the first one
      BluetoothDevice device = devices.first;
      BluetoothConnection newConnection = await
BluetoothConnection.toAddress(device.address);
      setState(() {
        connection = newConnection;
      });
      // Start listening to data
      connection!.input!.listen((Uint8List data) {
        String message = String.fromCharCodes(data);
        // Check the content of the message and redirect accordingly
        if (message.contains("important")) {
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) =>
ImportantMessageScreen(message)),
          );
        } else if (message.contains("urgent")) {
          Navigator.push(
            context,
            MaterialPageRoute(builder:  (context) => UrgentMessageScreen(message)),
          );
        } else {
          // Default screen for other messages
          Navigator.push(
            context,
            MaterialPageRoute(builder:  (context) => DefaultMessageScreen(message)),
          );
        }
      });
    } catch (error) {
      print('Error connecting to device: $error');
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Bluetooth Demo'),
      ),
      body: Center(
        child: connection == null
            ? const Text('Device Not Connected')
            : Text('Connected to device: $connection'),
      ),
    );
  }
}

class ImportantMessageScreen extends StatelessWidget {
  final String message;

  ImportantMessageScreen(this.message);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Important Message'),
      ),
      body: Center(
        child: Text(message),
      ),
    );
  }
}

class UrgentMessageScreen extends StatelessWidget {
  final String message;
  UrgentMessageScreen(this.message);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Urgent Message'),
      ),
      body: Center(
        child: Text(message),
      ),
    );
  }
}

class DefaultMessageScreen extends StatelessWidget {
  final String message;

  DefaultMessageScreen(this.message);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Default  Message'),
      ),
      body: Center(
        child: Text(message),
      ),
    );
  }
}
```

home.dart:

```
import 'dart:convert';
import 'dart:async';
import 'package:agrobot/Screens/login_screen.dart';
import 'package:agrobot/Screens/splash_screen.dart';
import 'package:agrobot/styles/energy.dart';
import 'package:agrobot/styles/crop.dart';
import 'package:agrobot/styles/soil.dart';
import 'package:agrobot/styles/water.dart';
import 'package:agrobot/styles/weather.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';
import 'package:flutter_joystick/flutter_joystick.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:shared_preferences/shared_preferences.dart';

class ResponsiveJoystickExampleApp extends StatefulWidget {
  final BluetoothConnection? connection;
  final double waterPercentage = 0.0;
  final double soilMoisture=0.0;
  final int batteryLevel=0;

  const ResponsiveJoystickExampleApp({super.key, this.connection});

  @override
  State<ResponsiveJoystickExampleApp> createState() =>
_ResponsiveJoystickExampleAppState();
}

class _ResponsiveJoystickExampleAppState extends
State<ResponsiveJoystickExampleApp> {

  double soilMoisture=0.0;
  double waterPercentage=0.0;
  int batteryLevel=0;

  StreamSubscription? _subscription;

    void sendMessageToDevice(String msg) {
```

```
    String message = msg;

    if (widget.connection != null) {
      widget.connection!.output.add(utf8.encode(message));
      widget.connection!.output.allSent.then((_) {
        // Message sent successfully
        debugPrint('Message sent: $message');
        _subscribeToData(); // Call subscribe method after message is sent
      }).catchError((error) {
        // Error occurred while sending message
        debugPrint('Error sending message: $error');
      });
    } else {
      // Connection is null, handle accordingly
      debugPrint('Bluetooth connection is not available');
    }
  }

void _subscribeToData() {
  // Cancel previous subscription if exists
  _subscription?.cancel();
  _subscription = widget.connection!.input!.listen((List<int> data) {
    final receivedString = utf8.decode(data);
    if(mounted){
      setState(() {
```

```
// Split the received string by commas
List<String> parts = receivedString.split(',');

// Extract data and parse if necessary
String waterPercentageString = parts[1];
double waterPercentage = double.tryParse(waterPercentageString) ?? 0.0;

String soilMoistureString = parts[0];
double soilMoisture = double.tryParse(soilMoistureString) ?? 0.0;

String batteryLevelString = parts[2];
int batteryLevel = int.tryParse(batteryLevelString) ?? 0;

// Pass the parsed data to the respective classes
if (waterPercentage >= 0.0 && waterPercentage <= 1.0) {
  Water(waterPercentage: waterPercentage);
}

if (soilMoisture >= 0.0 && soilMoisture <= 100.0) {
  Soil(soilMoisture: soilMoisture);
}

if (batteryLevel >= 0 && batteryLevel <= 100) {
  Energy(batteryLevel: batteryLevel);
}
```

```
          // Update the waterPercentage state variable if needed
          this.waterPercentage = waterPercentage;
          this.soilMoisture=soilMoisture;
          this.batteryLevel=batteryLevel;
        });
      }
    });
  }

    void sendMessage(String msg) {
    String message = msg;

    if (widget.connection != null) {
      widget.connection!.output.add(utf8.encode(message));
      widget.connection!.output.allSent.then((_) {
        // Message sent successfully
        debugPrint('Message sent: $message');
        _subscribeToData(); // Call subscribe method after message is sent
      }).catchError((error) {
        // Error occurred while sending message
        debugPrint('Error sending message: $error');
      });
    } else {
      // Connection is null, handle accordingly
    debugPrint('Bluetooth connection is not available');
  }
}

double _x = 0;
double _y = 0;

// Function to scale the joystick input to 0-255
int scaleTo255(double value) {
  // Map the value from -1 to 1 to -255 to 255
  return ((value + 1) / 2 * 255).round().clamp(0, 255);
}

void sendMessageToJoyStick(String message) {
  if (widget.connection != null) {
    widget.connection!.output.add(utf8.encode("$message\r\n"));
    widget.connection!.output.allSent.then((_) {
      debugPrint('Message sent: $message');
    });
  } else {
    debugPrint('Not connected to any device.');
  }
}

@override
```

```
Widget build(BuildContext context) {
 sendMessageToDevice("0xV");
 return LayoutBuilder(
  builder: (context, constraints) {
   return Scaffold(
    appBar: AppBar(
     backgroundColor: const Color(0xFFDAFFF2),
     title: const Text(
      'AgroBot',
      style: TextStyle(
       color: Color(0xFF006D44),
       fontFamily: 'UbuntuCondensed',
       fontWeight: FontWeight.bold,
      ),
     ),

     actions: <Widget>[
      IconButton(
       icon: const Icon(Icons.logout),
       onPressed: () {
        FirebaseAuth.instance.signOut().then((value) async {
         if (kDebugMode) {
          print("Signed Out");
         }
         var sharedPref = await SharedPreferences.getInstance();
```

```
                    sharedPref.setBool(SplashScreenState.keyLogin, false);
                    Navigator.push(
                        context, MaterialPageRoute(builder: (context) => const
LoginScreen()));
                  }).onError((error, stackTrace) {
                    if (kDebugMode) {
                      print("Error ${error.toString()}");
                    }
                  });
                },
              ),
            ],
          ),
          body: Row(
            children: [
              Expanded(
                flex: 2,
                child: SingleChildScrollView(
                  child: Column(
                    children: [
                      Container(
                        padding: const EdgeInsets.all(10),
                        alignment: Alignment.topLeft,
                        child: const Text(
                          'Functions',
                          style: TextStyle(
      fontSize: 30,
      fontFamily: 'UbuntuCondensed',
      color: Color(0xFF006D44),
    ),
  ),
),
Padding(
  padding: const EdgeInsets.only(left:  10),
  child: SizedBox(
    width: double.infinity,
    height: constraints.maxHeight  * 0.60,
    child: Wrap(
      spacing: 5,
      runSpacing: 10,
      alignment:  WrapAlignment.start,
      children: [
        ButtonDesign(
          text1:  'Plowing',
          iconData: FontAwesomeIcons.tractor,
          text2:  'Plowing Started',
          onPressed: () {
            sendMessage("0xA");
          },
        ),
        ButtonDesign(
      text1: 'Seedling',
      iconData: FontAwesomeIcons.seedling,
      text2: 'Seedling Started',
      onPressed: () {
        sendMessage("0xB");
      },
    ),
    ButtonDesign(
      text1: 'Watering',
      iconData: FontAwesomeIcons.droplet,
      text2: 'Watering Started',
      onPressed: () {
        sendMessage("0xC");
      },
    ),
    ButtonDesign(
      text1: 'All Above',
      iconData: FontAwesomeIcons.gears,
      text2: 'All 3 Above',
      onPressed: () {
        sendMessage("0xD");
      },
    ),
  ],
),
  ),
),
SizedBox(height: constraints.maxHeight * 0.05),
Container(
  padding: const EdgeInsets.all(20),
  alignment: Alignment.topLeft,
  child: const Text(
    'Monitoring',
    style: TextStyle(
      fontSize: 30,
      fontFamily: 'UbuntuCondensed',
      color: Color(0xFF006D44),
    ),
  ),
),
Padding(
  padding: const EdgeInsets.only(left: 10),
  child: SizedBox(
    width: double.infinity,
    height: constraints.maxHeight * 0.60,
    child: Wrap(
      spacing: 5,
      runSpacing: 10,
      alignment: WrapAlignment.start,
      children: [
                    Soil(soilMoisture: soilMoisture),
                    Water(waterPercentage: waterPercentage), // Passing the
connection
                    Energy(batteryLevel:batteryLevel),
                    const Weather(),
                  ],
                ),
              ),
            ),
            SizedBox(height: constraints.maxHeight * 0.10),
            Container(
              padding: const EdgeInsets.all(20),
              alignment: Alignment.topLeft,
              child: const Text(
                'Analytics',
                style: TextStyle(
                  fontSize: 30,
                  fontFamily: 'UbuntuCondensed',
                  color: Color(0xFF006D44),
                ),
              ),
            ),
            Padding(
              padding: const EdgeInsets.only(left: 10),
              child: SizedBox(
                width: double.infinity,
            height: constraints.maxHeight  * 0.40,
            child: const Wrap(
              spacing: 5,
              runSpacing: 10,
              alignment:  WrapAlignment.start,
              children: [
                Crop(),
              ],
            ),
          ),
        ),
      ],
    ),
  ),
),
Expanded(
  flex: 1,
  child: Column(
    mainAxisAlignment:  MainAxisAlignment.center,
    crossAxisAlignment:  CrossAxisAlignment.center,
    children: [
      SafeArea(
        child: Stack(
          children: [
            Align(
                    alignment: const Alignment(0, 0.8),
                    child: Joystick(
                      mode: JoystickMode.all,
                      listener: (details) {
                        // Update the x and y values when joystick moves
                        setState(() {
                          _x = details.x;
                          _y = details.y;
                        });

                        // Send the x and y values to the connected device
                        sendMessageToJoyStick("X: ${scaleTo255(_x)}, Y:
${scaleTo255(_y)}");
                      },
                    ),
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    ],
  ),
 );
},
    );
  }
}

class ButtonDesign extends StatefulWidget {
  final String text1;
  final IconData iconData;
  final String text2;
  final void Function() onPressed;

  const ButtonDesign({
    Key? key,
    required this.text1,
    required this.iconData,
    required this.text2,
    required this.onPressed,
  }) : super(key: key);

  @override
  State<ButtonDesign> createState() => _ButtonDesignState();
}

class _ButtonDesignState extends State<ButtonDesign> {
  bool _isClicked = false;
  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      style: ElevatedButton.styleFrom(
        minimumSize: const Size(80, 120),
        backgroundColor:!_isClicked ? const Color(0xFFDAFFF2) : const
Color(0xFF006D44),
      ),
      onPressed:() {
        setState(() {
          _isClicked = !_isClicked; // Toggle _isClicked state
        });
        widget.onPressed(); // Call the onPressed function provided by the parent
widget
      },
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          FaIcon(widget.iconData, size: 40,color:!_isClicked?const
Color(0xFF006D44): const Color(0xFFDAFFF2),),
          Text(widget.text1, style: TextStyle(fontSize: 20, fontFamily:
'UbuntuCondensed',color: !_isClicked ?const Color(0xFF006D44): const
Color(0xFFDAFFF2))),
        ],
      ),
    );
  }
```

```
}

void main() {
  runApp(const MaterialApp(
    home: ResponsiveJoystickExampleApp(),
  ));
}
```

home_screen.dart:

```
import 'package:agrobot/Screens/device_list_screen.dart';
import 'package:agrobot/Screens/login_screen.dart';
import 'package:agrobot/Screens/splash_screen.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:flutter/services.dart';

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key});
  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  void initState() {
    super.initState();
    SystemChrome.setPreferredOrientations([DeviceOrientation.landscapeLeft]);
  }

  @override
  void dispose() {
    SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
    super.dispose();
  }

  final double h=200;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: const Color(0xFFDAFFF2),
        title: const Text('AgroBot',style:
TextStyle(color:Color(0xFF006D44),fontFamily: 'UbuntuCondensed',fontWeight:
FontWeight.bold),),
        actions: <Widget>[
          IconButton(
            icon: const Icon(Icons.logout),
            onPressed: () {
            FirebaseAuth.instance.signOut().then((value) async {
            if (kDebugMode) {
              print("Signed Out");
            }
            var sharedPref= await SharedPreferences.getInstance();
            sharedPref.setBool(SplashScreenState.keyLogin,false);
            // ignore: use_build_context_synchronously
            Navigator.push(context,MaterialPageRoute(builder:  (context)=>const
LoginScreen()));
          }).onError((error, stackTrace) {
            if (kDebugMode) {
              print("Error ${error.toString()}");
            }
          });
            },
          ),
        ],
      ),
      body: const DeviceListScreen(),
    );
  }
}
```

Login_screen.dart

```
import 'package:agrobot/Screens/device_list_screen.dart';
import 'package:agrobot/Screens/signup_screen.dart';
import 'package:agrobot/Screens/splash_screen.dart';
import 'package:agrobot/reusable_widgets/resusable_widget.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:flutter/services.dart';

class LoginScreen extends StatefulWidget{
  const LoginScreen({super.key});

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen>{

  @override
  void initState() {
    super.initState();
    SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
  }
```

```
@override
void dispose() {
  SystemChrome.setPreferredOrientations([DeviceOrientation.landscapeLeft]);
  super.dispose();
}

final TextEditingController _passwordTextController=TextEditingController();
final TextEditingController _emailTextController=TextEditingController();

@override
Widget build(BuildContext context){
  return Scaffold(
    body:Container(
      width: double.infinity,
      height: double.infinity,
      decoration: const BoxDecoration(
        gradient: LinearGradient(
          colors: [Color(0xFF8711c1),Color(0xFF2472fc)],
          begin: Alignment.topCenter,
          end: Alignment.bottomCenter,
          )
          ),
          child: SingleChildScrollView(
            child: Padding(
              padding:
EdgeInsets.fromLTRB(20,MediaQuery.of(context).size.height*0.1,20,0),
                child: Column(
                  children: <Widget>[
                    logoWidget('assets/images/Without_BG.png',300,350),
                    const SizedBox(
                      height: 30,
                    ),
                    reusableTextField("Enter Email", Icons.person_outline, false,
_emailTextController),
                    const SizedBox(
                      height: 20,
                    ),
                    reusableTextField("Enter Password", Icons.lock_outline, true,
_passwordTextController),
                    const SizedBox(
                      height: 20,
                    ),
                    signinSignupScreenButton(context, true, (){
                      FirebaseAuth.instance.signInWithEmailAndPassword(
                        email: _emailTextController.text,
                        password: _passwordTextController.text).then((value) async {
                          if (kDebugMode) {
                            print("Login Successfully");
                          }
                          var sharedPref= await SharedPreferences.getInstance();


                          sharedPref.setBool(SplashScreenState.keyLogin,true);
                        Navigator.pushReplacement(context,MaterialPageRoute(builder:
(context)=>const DeviceListScreen()));

                      });
                    }),
                    const SizedBox(
                      height: 20,
                    ),
                    signupOptions(),
                  ],
                ),
              ),
            ),
          ),
        );
  }

  Row signupOptions(){
    return Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
      const Text("Don't have account? ",style: TextStyle(color: Colors.white70),),
      GestureDetector(
        onTap: (){
          Navigator.push(context,MaterialPageRoute(builder: (context)=> const
SignupScreen()));
```

```
        },
        child: const Text(
          'Sign Up',
          style:TextStyle(color: Colors.white,fontWeight: FontWeight.bold),
        ),
      )
      ],
    );
  }
}
```

Sample.dart

```
import 'package:agrobot/API/consts.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';

import 'package:geolocator/geolocator.dart';
import 'package:weather/weather.dart';

class SeedScreen extends StatefulWidget {
  const SeedScreen({Key? key}) : super(key: key);

  @override
  State<SeedScreen> createState() => _SeedScreenState();
}
```

```
class _SeedScreenState extends State<SeedScreen> {
  final WeatherFactory _wf = WeatherFactory(OPENWEATHER_API_KEY);

  Weather? _weather;

  @override
  void initState() {
    super.initState();
    SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
    _getLocationAndWeather();
  }

  @override
  void dispose() {
    SystemChrome.setPreferredOrientations([DeviceOrientation.landscapeLeft]);
    super.dispose();
  }

  Future<void> _getLocationAndWeather() async {
    Position position = await Geolocator.getCurrentPosition(
        desiredAccuracy: LocationAccuracy.high);
    _fetchWeather(position.latitude, position.longitude);
  }
  Future<void> _fetchWeather(double latitude, double longitude) async {
   Weather? weather = await _wf.currentWeatherByLocation(latitude, longitude);
   setState(() {
     _weather = weather;
   });
  }

  @override
 @override
 Widget build(BuildContext context) {
  String areaName = _weather?.areaName ?? "";
  return Scaffold(
   body: Column(
     children: [
      Text(areaName),
      Expanded(
        child: _buildUI(),
      ),
     ],
   ),
  );
 }
  Widget _buildUI() {
   if (_weather == null) {
     return const Center(
      child: CircularProgressIndicator(),
     );
   }

   return SingleChildScrollView(
      scrollDirection: Axis.vertical,
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.center,
        mainAxisSize: MainAxisSize.max,
        children: [
        Container(
        width: MediaQuery.of(context).size.width,  // Set width to full screen width
        height: MediaQuery.of(context).size.height * 0.5, // Set height to 50% of
 screen height
        child: _locationHeader(),
      ),
          SizedBox(
            height: MediaQuery.of(context).size.height * 0.08,
          ),
        ],
      ),
   );
  }

  Widget _locationHeader() {
   String areaName = _weather?.areaName ?? "";


   if(areaName=='Konkan Division'){
     return const Column(
      children: [
        Text("1.Rice",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
        Text("2.Coconut",style: TextStyle(fontSize: 15,fontWeight:
 FontWeight.bold),),
        Text("3.Cashew",style: TextStyle(fontSize: 15,fontWeight:
 FontWeight.bold),),
        Text("4.Mango",style: TextStyle(fontSize: 15,fontWeight:
 FontWeight.bold),),
        Text("5.Kokum",style: TextStyle(fontSize: 15,fontWeight:
 FontWeight.bold),),
      ],
     );
   }
   else if(areaName=='Pune Division'){
     return const Column(
      children: [
        Text("1.Sugarcane",style: TextStyle(fontSize: 15,fontWeight:
 FontWeight.bold),),
        Text("2.Grapes",style: TextStyle(fontSize: 15,fontWeight:
 FontWeight.bold),),
        Text("3.Wheat",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
        Text("4.Soybean",style: TextStyle(fontSize: 15,fontWeight:
 FontWeight.bold),),
        Text("5.Sorghum (Jowar)",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
      ],
    );
  }
  else if(areaName=='Nashik Division'){
    return const Column(
      children: [
        Text("1.Grapes",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
        Text("2.Onions",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
        Text("3.Tomatoes",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
        Text("4.Wheat",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
        Text("5.Sorghum (Jowar)",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
      ],
    );
  }
  else if(areaName=='Nagpur Division'){
    return const Column(
      children: [
        Text("1.Soybean",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
        Text("2.Cotton",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
        Text("3.Pigeon pea (Tur)",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
        Text("4.Sorghum (Jowar)",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
        Text("5.Maize",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
      ],
    );
  }
  else if(areaName=='Aurangabad Division'){
    return const Column(
      children: [
        Text("1.Grapes",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
        Text("2.Pomegranate",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
        Text("3.Soybean",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
        Text("4.Wheat",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
        Text("5.Sorghum (Jowar)",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
      ],
    );
  }
  else if(areaName=='Amravati Division'){
    return const Column(
      children: [
        Text("1.Cotton",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
        Text("2.Soybean",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
         Text("3.Sorghum (Jowar)",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
         Text("4.Pigeon pea (Tur)",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
         Text("5.Maize",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
       ],
     );
    }
    else{
      return const Text("",
      style:  TextStyle(fontSize: 20, fontWeight: FontWeight.bold),);
    }
  }

}
```

Screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_blue_plus/flutter_blue_plus.dart';


class Sample extends StatefulWidget {
  const Sample({super.key});

  @override
  State<Sample> createState() => _MyHomePageState();
```

```
}

class _MyHomePageState extends State<Sample> {
  List<BluetoothDevice> devicesList = [];

  @override
  void initState() {
    super.initState();
    startBluetoothScan();
  }

  void startBluetoothScan() async {
    // Start scanning
    await FlutterBluePlus.startScan(timeout: Duration(seconds: 4));

    // Listen to scanned devices
    FlutterBluePlus.scanResults.listen((List<ScanResult> scanResults) {
      // Update the devices list
      setState(() {
        devicesList.clear(); // Clear the previous list
        for (ScanResult scanResult in scanResults) {
          devicesList.add(scanResult.device);
        }
      });
    });
```

```
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('BLE Devices'),
    ),
    body: ListView.builder(
      itemCount: devicesList.length,
      itemBuilder: (BuildContext context, int index) {
        return ListTile(
          title: Text(devicesList[index].advName),
          subtitle: Text(devicesList[index].remoteId.toString()),
          // You can add more information if needed
        );
      },
    ),
  );
}

@override
void dispose() {
  FlutterBluePlus.stopScan();
  super.dispose();
  }
}
```

Signup_screen.dart

```
import 'package:agrobot/Screens/device_list_screen.dart';
import 'package:agrobot/Screens/splash_screen.dart';
import 'package:agrobot/reusable_widgets/resusable_widget.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:flutter/services.dart';

class SignupScreen extends StatefulWidget{
  const SignupScreen({super.key});

  @override
  State<SignupScreen> createState() => _SignupScreenState();
}

class _SignupScreenState extends State<SignupScreen>{

  @override
  void initState() {
    super.initState();
    SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
  }

  @override
  void dispose() {
    SystemChrome.setPreferredOrientations([DeviceOrientation.landscapeLeft]);
    super.dispose();
  }

final TextEditingController _passwordTextController=TextEditingController();
final TextEditingController _usernameTextController=TextEditingController();
final TextEditingController _emailTextController=TextEditingController();

  @override
  Widget build(BuildContext context){
    return Scaffold(
      extendBodyBehindAppBar: true,
      appBar: AppBar(
        backgroundColor: Colors.transparent,
        elevation: 0,
        title: const Text("Sign Up",style: TextStyle(fontSize: 24,fontWeight:
FontWeight.bold,color: Colors.white),),
      ),
      body:Container(
        width: double.infinity,
        height: double.infinity,
        decoration: const BoxDecoration(
         gradient: LinearGradient(
           colors: [Color(0xFF8711c1),Color(0xFF2472fc)],
           begin: Alignment.topCenter,
           end: Alignment.bottomCenter,
           )
           ),
           child: SingleChildScrollView(
             child: Padding(
               padding:
    EdgeInsets.fromLTRB(20,MediaQuery.of(context).size.height*0.1,20,0),
               child: Column(
                 children: <Widget>[
                   logoWidget('assets/images/Without_BG.png',300,350),
                   const SizedBox(
                     height: 30,
                   ),
                   reusableTextField("Enter  Username", Icons.person_outline, false,
    _usernameTextController),
                   const SizedBox(
                     height: 20,
                   ),
                   reusableTextField("Enter  Email", Icons.person_outline, false,
    _emailTextController),
                   const SizedBox(
                     height: 20,
                   ),
                   reusableTextField("Enter  Password", Icons.lock_outline, true,
    _passwordTextController),
                   const SizedBox(
                     height: 20,
                   ),
                   signinSignupScreenButton(context,  false, (){
                     FirebaseAuth.instance.createUserWithEmailAndPassword(
                       email: _emailTextController.text,
                       password: _passwordTextController.text
                     ).then((value) async {
                     if (kDebugMode) {
                       print("Created New Account");
                     }
                     var sharedPref= await SharedPreferences.getInstance();


                         sharedPref.setBool(SplashScreenState.keyLogin,true);
                           // ignore: use_build_context_synchronously
                           Navigator.pushReplacement(context,MaterialPageRoute(builder:
    (context)=>const DeviceListScreen()));
                     }).onError((error, stackTrace) {
                       if (kDebugMode) {
                         print("Error ${error.toString()}");
                       }
                     });
                   }),
                 ],
               ),
             ),
           ),
       ),
     );
 }
    }
```

Soil_screen.dart:

```
import 'package:agrobot/API/consts.dart';
import 'package:weather/weather.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:geolocator/geolocator.dart';
import 'package:percent_indicator/circular_percent_indicator.dart';

class SoilScreen extends StatefulWidget {
  final double soilMoisture; // Declare waterPercentage as final
  const SoilScreen({super.key, required this.soilMoisture}); // Fix super constructor
invocation

  @override
  State<SoilScreen> createState() => _SoilScreenState();
```

```
}

class _SoilScreenState extends State<SoilScreen> {
  double soilMoisture =0.0;
  final WeatherFactory _wf = WeatherFactory(OPENWEATHER_API_KEY);

  Weather? _weather;

  @override
  void initState() {
    super.initState();
    soilMoisture = widget.soilMoisture;
    SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
    _getLocationAndWeather();
  }

  @override
  void dispose() {
    SystemChrome.setPreferredOrientations([DeviceOrientation.landscapeLeft]);
    super.dispose();
  }

  Future<void> _getLocationAndWeather() async {
    Position position = await Geolocator.getCurrentPosition(
        desiredAccuracy: LocationAccuracy.high);
    _fetchWeather(position.latitude, position.longitude);
  }

  Future<void> _fetchWeather(double latitude, double longitude) async {
    Weather? weather = await _wf.currentWeatherByLocation(latitude, longitude);
    setState(() {
      _weather = weather;
    });
  }

  Color getColorAndPercentage(double percentage) {
    if (percentage < 0.3) {
      return Colors.red;
    } else if (percentage >= 0.3 && percentage <= 0.7) {
      return Colors.yellow;
    } else {
      return Colors.green;
    }
  }

  Widget _buildUI() {
    if (_weather == null) {
      return const Center(
        child: CircularProgressIndicator(),
      );
```

```
   }

   return Container(
         child: _locationHeader(),
   );
 }

 Widget _locationHeader() {
   String areaName = _weather?.areaName ?? "";

   if(areaName=='Konkan Division'){
     return const Column(
       children: [
         Text("1.Rice",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
         Text("2.Coconut",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
         Text("3.Cashew",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
         Text("4.Mango",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
         Text("5.Kokum",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
       ],
     );
   }
   else if(areaName=='Pune Division'){
     return const Column(
        children: [
         Text("1.Sugarcane",style: TextStyle(fontSize: 15,fontWeight: 
FontWeight.bold),),
         Text("2.Grapes",style: TextStyle(fontSize: 15,fontWeight: 
FontWeight.bold),),
         Text("3.Wheat",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
         Text("4.Soybean",style: TextStyle(fontSize: 15,fontWeight: 
FontWeight.bold),),
         Text("5.Sorghum (Jowar)",style: TextStyle(fontSize: 15,fontWeight: 
FontWeight.bold),),
        ],
      );
    }
    else if(areaName=='Nashik Division'){
      return const Column(
        children: [
         Text("1.Grapes",style: TextStyle(fontSize: 15,fontWeight: 
FontWeight.bold),),
         Text("2.Onions",style: TextStyle(fontSize: 15,fontWeight: 
FontWeight.bold),),
         Text("3.Tomatoes",style: TextStyle(fontSize: 15,fontWeight: 
FontWeight.bold),),
         Text("4.Wheat",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
         Text("5.Sorghum (Jowar)",style: TextStyle(fontSize: 15,fontWeight: 
FontWeight.bold),),
        ],
      );
    }
    else if(areaName=='Nagpur Division'){
      return const Column(
        children: [
          Text("1.Soybean",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
          Text("2.Cotton",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
          Text("3.Pigeon pea (Tur)",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
          Text("4.Sorghum (Jowar)",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
          Text("5.Maize",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
        ],
      );
    }
    else if(areaName=='Aurangabad Division'){
      return const Column(
        children: [
          Text("1.Grapes",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
          Text("2.Pomegranate",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
          Text("3.Soybean",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
          Text("4.Wheat",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
          Text("5.Sorghum (Jowar)",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
        ],
      );
    }
    else if(areaName=='Amravati Division'){
      return const Column(
        children: [
          Text("1.Cotton",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
          Text("2.Soybean",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
          Text("3.Sorghum (Jowar)",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
          Text("4.Pigeon pea (Tur)",style: TextStyle(fontSize: 15,fontWeight:
FontWeight.bold),),
          Text("5.Maize",style: TextStyle(fontSize: 15,fontWeight: FontWeight.bold),),
        ],
      );
    }
    else{
      return const Text("",
      style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),);
    }
  }

  @override
  Widget build(BuildContext context) {
     String areaName = _weather?.areaName ?? "";
      return Scaffold(
      body: Center(
child: Column(
   mainAxisAlignment: MainAxisAlignment.center,
   crossAxisAlignment: CrossAxisAlignment.center,
   children: [
    SizedBox(height: 100,),
    const Text(
     'Moisture',
     style: TextStyle(fontSize: 50),
    ),
    const SizedBox(height: 50),
    CircularPercentIndicator(
     animation: true,
     animationDuration: 1000,
     radius: 100,
     lineWidth: 20,
     percent: soilMoisture,
     progressColor: getColorAndPercentage(soilMoisture),
     backgroundColor: getColorAndPercentage(soilMoisture)
       .withOpacity(0.2),
     circularStrokeCap: CircularStrokeCap.round,
     center: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
       const Icon(
                IconData(0xf05a2, fontFamily: 'MaterialIcons'),
                size: 50,
                color: Colors.blue,
               ),
               Text(
                '${(soilMoisture * 100).toStringAsFixed(0)}%',
                style: const TextStyle(fontSize: 40),
               ),
              ],
             ),
            ),
            SizedBox(height: 50,),
            Text(areaName,style: TextStyle(fontSize: 25,fontWeight:
FontWeight.bold),),
            Expanded(
            child: _buildUI(),
            ),
           ],
          ),
       ),
     );
   }
}
```

Splash_screen.dart:

```
import 'dart:async';
import 'package:agrobot/Screens/device_list_screen.dart';
import 'package:agrobot/Screens/login_screen.dart';
import 'package:agrobot/Screens/signup_screen.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:shared_preferences/shared_preferences.dart';

class SplashScreen extends StatefulWidget{

  const SplashScreen({super.key});

  @override
  State<SplashScreen> createState() => SplashScreenState();
}

class SplashScreenState extends State<SplashScreen> {

  static const String keyLogin="Login";

  @override
  void initState() {
    super.initState();
    SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
    whereToGo();
  }
```

```
 @override
 void dispose() {
  SystemChrome.setPreferredOrientations([DeviceOrientation.landscapeLeft]);
  super.dispose();
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   body: Center(
    child: Container(
     color:const Color(0xFFDAFFF2),
     child: Image.asset('assets/images/LogoName.png'),
    ),
   ),
  );
 }
void whereToGo() async{
var sharedPref= await SharedPreferences.getInstance();
var isLoggedIn=sharedPref.getBool(keyLogin);
Timer(const Duration(seconds: 2),(){
 if(isLoggedIn!=null){
  if(isLoggedIn){
   Navigator.pushReplacement(context,MaterialPageRoute(builder:
(context)=>const DeviceListScreen()));
```

```
       }
    else{
      Navigator.pushReplacement(context,MaterialPageRoute(builder:
(context)=>const LoginScreen()));
      }
    } else{
      Navigator.pushReplacement(context,MaterialPageRoute(builder:
(context)=>const SignupScreen()));
    }
    }
  );
}
}
```

Water_screen.dart:

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:percent_indicator/circular_percent_indicator.dart';

class WaterScreen extends StatefulWidget {
  final double waterPercentage; // Declare waterPercentage as final
  WaterScreen({Key? key, required this.waterPercentage}) : super(key: key); // Fix
super constructor invocation


  @override
  State<WaterScreen> createState() => _WaterScreenState();
```

```
}

class _WaterScreenState extends State<WaterScreen> {
  double waterPercentage=0.0;

  @override
  void initState() {
    super.initState();
    waterPercentage = widget.waterPercentage;
    SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
  }

  @override
  void dispose() {
    SystemChrome.setPreferredOrientations([DeviceOrientation.landscapeLeft]);
    super.dispose();
  }
  Color getColorAndPercentage(double percentage) {
    if (percentage < 0.3) {
      return Colors.red;
    } else if (percentage >= 0.3 && percentage <= 0.7) {
      return Colors.yellow;
    } else {
      return Colors.green;
    }
```

```
}
@override
Widget build(BuildContext context) {
 return Scaffold(
  body: Container(
   width: double.infinity,
   height: double.infinity,
   child: Center(
    child: Column(
     mainAxisAlignment: MainAxisAlignment.center,
     crossAxisAlignment: CrossAxisAlignment.center,
     children: [
      const Text(
       'Water Quality',
       style: TextStyle(fontSize: 50),
      ),
      const SizedBox(height: 50),
      CircularPercentIndicator(
       animation: true,
       animationDuration: 1000,
       radius: 100,
       lineWidth: 20,
       percent: waterPercentage,
       progressColor: getColorAndPercentage(waterPercentage),
       backgroundColor: getColorAndPercentage(waterPercentage)
```

```
                .withOpacity(0.2),
              circularStrokeCap: CircularStrokeCap.round,
              center: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  const Icon(
                    IconData(0xf05a2, fontFamily: 'MaterialIcons'),
                    size: 50,
                    color: Colors.blue,
                  ),
                  Text(
                    '${(waterPercentage * 100).toStringAsFixed(0)}%',
                    style: const TextStyle(fontSize: 40),
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

Weather_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:intl/intl.dart';
import 'package:geolocator/geolocator.dart';
import 'package:weather/weather.dart';
import 'package:geocoding/geocoding.dart';
import 'package:agrobot/API/consts.dart';

class WeatherScreen extends StatefulWidget {
  const WeatherScreen({super.key});

  @override
  State<WeatherScreen> createState() => _WeatherScreenState();
}

class _WeatherScreenState extends State<WeatherScreen> {
  final WeatherFactory _wf = WeatherFactory(OPENWEATHER_API_KEY);

  Weather? _weather;
  String _location = 'Loading...';

  @override
  void initState() {
    super.initState();
    SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
    _getLocationAndWeather();
  }

  @override
  void dispose() {
    SystemChrome.setPreferredOrientations([DeviceOrientation.landscapeLeft]);
    super.dispose();
  }

  Future<void> _getLocationAndWeather() async {
    try {
      Position position = await Geolocator.getCurrentPosition(
          desiredAccuracy: LocationAccuracy.high);
      List<Placemark> placemarks = await placemarkFromCoordinates(
          position.latitude, position.longitude);
      setState(() {
        _location = '${placemarks.first.locality},
${placemarks.first.administrativeArea}';
      });
      _fetchWeather(position.latitude, position.longitude);
    } catch (e) {
      setState(() {
        _location = 'Error fetching location';
      });
    }
```

```
}

Future<void> _fetchWeather(double latitude, double longitude) async {
  Weather? weather = await _wf.currentWeatherByLocation(latitude, longitude);
  setState(() {
    _weather = weather;
  });
  _showPopupMessage(getIrrigationRecommendation());
}

String getIrrigationRecommendation() {
  if (_weather != null && _weather!.weatherDescription != null) {
    String weatherDescription = _weather!.weatherDescription!.toLowerCase();

    if (weatherDescription.contains('rain')) {
      return 'Dont Irrigate crops, Rainy conditions detected.';
    } else if (weatherDescription.contains('clear') ||
        weatherDescription.contains('sunny')) {
      return "Irrigate your crops. Sunny conditions.";
    } else if (weatherDescription.contains('smoke')) {
      return 'Irrigate your crops. Smoke detected.';
    } else if (weatherDescription.contains('cloud')) {
      return 'Monitor soil moisture. Cloudy conditions.';
    }
  }
```

```
 return 'No specific irrigation recommendation at the moment.';
}

void _showPopupMessage(String message) {
 showDialog(
  context: context,
  builder: (BuildContext context) {
   return AlertDialog(
    content: Text(message),
   );
  },
 );
 Future.delayed(const Duration(seconds: 10), () {
  Navigator.of(context).pop();
 });
}

@override
Widget build(BuildContext context) {
 return Scaffold(
  body: _buildUI(),
 );
}
```

```
Widget _buildUI() {
 if (_weather == null) {
  return const Center(
   child: CircularProgressIndicator(),
  );
 }

 return SizedBox(
  width: double.infinity,
  height: double.infinity,
  child: Column(
   mainAxisAlignment: MainAxisAlignment.center,
   crossAxisAlignment: CrossAxisAlignment.center,
   children: [
    Text(
     _location,
     style: const TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
    ),
    SizedBox(
     height: MediaQuery.of(context).size.height * 0.08,
    ),
    _dateTimeInfo(),
    SizedBox(
     height: MediaQuery.of(context).size.height * 0.05,
    ),
        _weatherIcon(),
        SizedBox(
          height: MediaQuery.of(context).size.height  * 0.02,
        ),
        _currentTemp(),
        SizedBox(
          height: MediaQuery.of(context).size.height  * 0.02,
        ),
        _extraInfo(),
        SizedBox(
          height: MediaQuery.of(context).size.height  * 0.02,
        ),
      ],
    ),
  );
}

Widget _dateTimeInfo() {
  DateTime  now = _weather!.date!;
  return Column(
    mainAxisAlignment:  MainAxisAlignment.center,
    crossAxisAlignment:  CrossAxisAlignment.center,
    children: [
      Text(
        DateFormat("EEEE").format(now),
        style: const TextStyle(fontWeight: FontWeight.w700,fontSize: 35),
      ),
      Text(
        DateFormat("d.M.y").format(now),
        style: const TextStyle(fontWeight: FontWeight.w700,fontSize: 35),
      ),
    ],
  );
}

Widget _weatherIcon() {
  return Column(
    mainAxisAlignment: MainAxisAlignment.center,
    crossAxisAlignment: CrossAxisAlignment.center,
    children: [
      Container(
        height: MediaQuery.of(context).size.height * 0.20,
        decoration: BoxDecoration(
          image: DecorationImage(
            image: NetworkImage(
                "http://openweathermap.org/img/wn/${_weather?.weatherIcon}@4x.png
"),
          ),
        ),
      ),
      Text(
        _weather?.weatherDescription ?? "",
        style: const TextStyle(color: Colors.black, fontSize: 20),
      ),
    ],
  );
}

Widget _currentTemp() {
  return Column(
    mainAxisAlignment: MainAxisAlignment.center,
    crossAxisAlignment: CrossAxisAlignment.center,
    children: [
      Text(
        "${_weather?.temperature?.celsius?.toStringAsFixed(0)} °C",
        style: const TextStyle(
            color: Colors.black, fontSize: 90, fontWeight: FontWeight.w500),
      ),
    ],
  );
}

Widget _extraInfo() {
  return Container(
    height: MediaQuery.of(context).size.height * 0.15,
    width: MediaQuery.of(context).size.width * 0.80,
decoration: BoxDecoration(
  color: Colors.deepPurpleAccent,
  borderRadius: BorderRadius.circular(20),
),
padding: const EdgeInsets.all(8.0),
child: Column(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    Text(
      "Max: ${_weather?.tempMax?.celsius?.toStringAsFixed(0)} °C",
      style: const TextStyle(color: Colors.white, fontSize: 15),
    ),
    Text(
      "Min: ${_weather?.tempMin?.celsius?.toStringAsFixed(0)} °C",
      style: const TextStyle(color: Colors.white, fontSize: 15),
    ),
    Text(
      "Wind: ${_weather?.windSpeed?.toStringAsFixed(0)} m/s",
      style: const TextStyle(color: Colors.white, fontSize: 15),
    ),
    Text(
      "Humidity: ${_weather?.humidity?.toStringAsFixed(0)} %",
      style: const TextStyle(color: Colors.white, fontSize: 15),
    ),
        ],
      ),
    );
  }
}

void main() {
  runApp(
    const MaterialApp(
    home:  WeatherScreen(),
  ));
}
```

Welcome_screen.dart

```
import 'package:agrobot/Screens/login_screen.dart';
import 'package:agrobot/Screens/signup_screen.dart';
import 'package:agrobot/styles/button2.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';

class WelcomeScreen extends StatefulWidget{

  const WelcomeScreen({super.key});
  @override
  State<WelcomeScreen> createState() => _WelcomeScreenState();
}

class _WelcomeScreenState extends State<WelcomeScreen> {

  @override
  void initState() {
    SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
    super.initState();
  }

  @override
  void dispose() {
    SystemChrome.setPreferredOrientations([DeviceOrientation.landscapeLeft]);
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Column(
          children: [
            SizedBox(
              width: 300,
              height: 300,
              child: Image.asset('assets/images/HelloScreen.png')
            ),
            const Text('Hello',style:TextStyle(fontSize:40,fontFamily:
'UbuntuCondensed',fontWeight: FontWeight.bold)),
            const SizedBox(
              width: 300,
              child: Text('Welcome to AgroBot, let us guide you through the farming
world',style: TextStyle(fontSize: 16),textAlign: TextAlign.center,)),
            const SizedBox(height:50),
            ElevatedButton(
              style:buttonSecondary(const Color(0xFF02AA6D),false),
              onPressed: (){
                Navigator.push(context,MaterialPageRoute(builder: (context)=> const
LoginScreen()));
              },
              child: const Text('Login',style: TextStyle(fontSize: 20,color: Colors.white),)
            ),
            const SizedBox(height:10),
            ElevatedButton(
              style:buttonSecondary(Colors.white,true),
              onPressed: (){
                Navigator.push(context, MaterialPageRoute(builder: (context)=>const
SignupScreen()));
              },
              child: const Text('SignUp',style:  TextStyle(fontSize:  20,color:
Colors.black),)
            ),
            const SizedBox(height:50),
            const Text('Join Us'),
            const SizedBox(height:10),
            const Row(
              mainAxisAlignment:  MainAxisAlignment.center,
              crossAxisAlignment:  CrossAxisAlignment.center,
              children: [
                FaIcon(FontAwesomeIcons.telegram,size:  40,color: Colors.blue,),
                SizedBox(width:15),
                FaIcon(FontAwesomeIcons.instagram,size:  40,color: Colors.black,),
                SizedBox(width:15),
                FaIcon(FontAwesomeIcons.twitter,size:  40,color: Colors.blue,),
                SizedBox(width:15),
                FaIcon(FontAwesomeIcons.google,size:  40,color: Colors.black,),
              ],
            ),
          ],
        )
      ),
    );
   }
}
```

Api_service.dart:

```
import 'dart:convert';
import 'dart:io';

import 'package:dio/dio.dart';

import '../constants/api_constants.dart';

class ApiService {
  final Dio _dio = Dio();

  Future<String> encodeImage(File image) async {
    final bytes = await image.readAsBytes();
    return base64Encode(bytes);
  }

  Future<String> sendMessageGPT({required String diseaseName}) async {
    try {
      final response = await _dio.post(
        "$BASE_URL/chat/completions",
        options: Options(
          headers: {
            HttpHeaders.authorizationHeader: 'Bearer $API_KEY',
            HttpHeaders.contentTypeHeader: "application/json",
          },
        ),
        data: {
          "model": 'gpt-3.5-turbo',
          "messages": [
            {
              "role": "user",
              "content":
                "GPT, upon receiving the name of a plant disease, provide three
precautionary measures to prevent or manage the disease. These measures should be
concise, clear, and limited to one sentence each. No additional information or
context is needed—only the three precautions in bullet-point format. The disease is
$diseaseName",
            }
          ],
        },
      );

      final jsonResponse = response.data;

      if (jsonResponse['error'] != null) {
        throw HttpException(jsonResponse['error']["message"]);
      }

      return jsonResponse["choices"][0]["message"]["content"];
    } catch (error) {
      throw Exception('Error: $error');
    }
```

```
}

Future<String> sendImageToGPT4Vision({
  required File image,
  int maxTokens = 50,
  String model = "gpt-4-vision-preview",
}) async {
  final String base64Image = await encodeImage(image);

  try {
    final response = await _dio.post(
      "$BASE_URL/chat/completions",
      options: Options(
        headers: {
          HttpHeaders.authorizationHeader: 'Bearer $API_KEY',
          HttpHeaders.contentTypeHeader: "application/json",
        },
      ),
      data: jsonEncode({
        'model': model,
        'messages': [
          {
            'role': 'system',
            'content': 'You have to give concise and short answers'
          },
          {
            'role': 'user',
            'content': [
              {
                'type': 'text',
                'text':
                    'GPT, your task is to identify plant health issues with precision.
Analyze any image of a plant or leaf I provide, and detect all abnormal conditions,
whether they are diseases, pests, deficiencies, or decay. Respond strictly with the
name of the condition identified, and nothing else—no explanations, no additional
text. If a condition is unrecognizable, reply with \'I don\'t know\'. If the image is not
plant-related, say \'Please pick another image\'',
              },
              {
                'type': 'image_url',
                'image_url': {
                  'url': 'data:image/jpeg;base64,$base64Image',
                },
              },
            ],
          },
        ],
        'max_tokens': maxTokens,
      }),
    );

    final jsonResponse = response.data;
```

```
      if (jsonResponse['error'] != null) {
        throw HttpException(jsonResponse['error']["message"]);
      }
      return jsonResponse["choices"][0]["message"]["content"];
    } catch (e) {
      throw Exception('Error: $e');
    }
  }
}
```

Styles:

Button1.dart:

```
import 'package:flutter/material.dart';

ButtonStyle buttonPrimary({required Size minimumSize}){
  return ElevatedButton.styleFrom(
  minimumSize: minimumSize,
  backgroundColor:const Color(0xFFDAFFF2),
  elevation: 0,
  shape: const RoundedRectangleBorder(
    borderRadius: BorderRadius.all(Radius.circular(20)),
  ),
  splashFactory: NoSplash.splashFactory,
);
}
```

Button2.dart:

```
import 'package:flutter/material.dart';

ButtonStyle buttonSecondary(Color color1, bool includeBorder,) {
  return ElevatedButton.styleFrom(
    minimumSize: const Size(300, 60),
    shape: RoundedRectangleBorder(
      borderRadius: const BorderRadius.all(Radius.circular(25)),
      side: includeBorder ? const BorderSide(color: Color(0xFF02AA6D), width: 2) :
BorderSide.none,
    ),
    backgroundColor: color1,
  );
}
```

Crop.dart:

```
import 'package:agrobot/Screens/crop_screen.dart';
import 'package:agrobot/styles/button1.dart';
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
class Crop extends StatelessWidget{
  const Crop({super.key});
  @override
  Widget build(BuildContext context){
    return ElevatedButton(
      style: buttonPrimary(minimumSize: const Size(110,120)),
      onPressed: (){
        Navigator.push(context,MaterialPageRoute(builder:(context)=>const
HomePage()));
      },
      child: const Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          FaIcon(FontAwesomeIcons.wheatAwn,size: 40,),
            Text('Crop Health',style: TextStyle(fontSize: 20,fontFamily:
'UbuntuCondensed')),
        ],
      ),
    );
  }
}
```

Energy.dart

```
import 'package:flutter/material.dart';
import 'package:agrobot/Screens/battery_screen.dart';
import 'package:agrobot/styles/button1.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';

class Energy extends StatefulWidget {
  final int batteryLevel;
  const Energy({super.key, required this.batteryLevel});

  @override
  State<Energy> createState() => _EnergyState();
}

class _EnergyState extends State<Energy> {
   late int batteryLevel;

   @override
  void initState() {
    batteryLevel = widget.batteryLevel;
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      style: buttonPrimary(minimumSize: const Size(120, 120)),
      onPressed: () {
        Navigator.push(context, MaterialPageRoute(builder: (context) =>
BatteryScreen(batteryLevel:widget.batteryLevel)));
      },
      child: const Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          FaIcon(FontAwesomeIcons.batteryQuarter, size: 40),
          Text('Battery', style: TextStyle(fontSize: 20, fontFamily:
'UbuntuCondensed')),
        ],
      ),
    );
  }
}
```

Soil.dart:

```
import 'package:agrobot/Screens/soil_screen.dart';
import 'package:flutter/material.dart';
import 'package:agrobot/styles/button1.dart';

class Soil extends StatefulWidget {
  final double soilMoisture; // Declare waterPercentage as final
  const Soil({super.key, required this.soilMoisture}); // Fix super constructor
invocation

  @override
  State<Soil> createState() => _SoilState();
}

class _SoilState extends State<Soil> {
  double soilMoisture=0.0;
  @override
  void initState() {
    soilMoisture = widget.soilMoisture;
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      style: buttonPrimary(minimumSize: const Size(120, 120)),
      onPressed: () {
        Navigator.push(context, MaterialPageRoute(builder: (context) =>
SoilScreen(soilMoisture: widget.soilMoisture)));
      },
      child: const Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(Icons.landscape, size: 40),
          Text('Soil', style: TextStyle(fontSize: 20, fontFamily: 'UbuntuCondensed')),
        ],
      ),
    );
  }
}
```

Solar.dart:

```
// import 'package:agrobot/Screens/solar_screen.dart';
import 'package:agrobot/styles/button1.dart';
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
class Solar extends StatelessWidget{
  const Solar({super.key});
  @override
  Widget build(BuildContext context){
    return ElevatedButton(
      style: buttonPrimary(minimumSize: const Size(120,120)),
      onPressed: (){
        // Navigator.push(context,MaterialPageRoute(builder: (context)=>const
SolarScreen()));
      },
      child: const Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          FaIcon(FontAwesomeIcons.solarPanel,size: 40,),
            Text('Solar',style: TextStyle(fontSize: 20,fontFamily: 'UbuntuCondensed')),
        ],
      ),
    );
  }
}
```

Water.dart:

```
import 'package:flutter/material.dart';
import 'package:agrobot/Screens/water_screen.dart';
import 'package:agrobot/styles/button1.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';

class Water extends StatefulWidget {
  final double waterPercentage; // Declare waterPercentage as final
  const Water({super.key, required this.waterPercentage}); // Fix super constructor
invocation

  @override
  State<Water> createState() => _WaterState();
}

class _WaterState extends State<Water> {
  late double waterPercentage;

  @override
  void initState() {
    waterPercentage = widget.waterPercentage;
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      style: buttonPrimary(minimumSize: const Size(120, 120)),
      onPressed: () {
        Navigator.push(context, MaterialPageRoute(builder: (context) =>
WaterScreen(waterPercentage: widget.waterPercentage)));
      },
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          FaIcon(FontAwesomeIcons.water, size: 40),
          Text('Water', style: TextStyle(fontSize: 20, fontFamily: 'UbuntuCondensed')),
        ],
      ),
    );
  }
}
```

Weather.dart:

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';
import 'package:agrobot/Screens/weather_screen.dart';
import 'package:agrobot/styles/button1.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
class Weather extends StatefulWidget {
  final BluetoothConnection? connection;
  const Weather({super.key, this.connection});
  @override
  State<Weather> createState() => _WeatherState();
}
class _WeatherState extends State<Weather> {
  // Function to send message to Bluetooth device
  void sendMessageToDevice(String msg) {
    // Replace 'your_message' with the actual message you want to send
    String message = msg;

    if (widget.connection != null) {
      widget.connection!.output.add(utf8.encode(message));
      widget.connection!.output.allSent.then((_) {
        // Message sent successfully
        debugPrint('Message sent: $message');
      }).catchError((error) {
        // Error occurred while sending message
        debugPrint('Error sending message: $error');
      });
    } else {
      // Connection is null, handle accordingly
      debugPrint('Bluetooth connection is not available');
    }
  }
  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      style: buttonPrimary(minimumSize: const Size(120, 120)),
      onPressed: () {
        sendMessageToDevice("Weather Data"); // Call function to send message when
button is pressed
        Navigator.push(context, MaterialPageRoute(builder: (context) => const
WeatherScreen()));
      },
      child: const Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          FaIcon(FontAwesomeIcons.cloudSunRain, size: 40),
          Text('Weather', style: TextStyle(fontSize: 20, fontFamily:
'UbuntuCondensed')),
        ],
      ),
    );
  }
}
```

Main.dart:

```
import 'package:agrobot/Screens/splash_screen.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: const FirebaseOptions(
      apiKey: 'AIzaSyAmv79_QQnO4sVo-8LSlU1NDE64AEPvoOE',
      appId: '1:989810940058:android:4db1790d50cca0d5ad2723',
      messagingSenderId: '989810940058',
      projectId: 'agrobot-project',
      storageBucket: 'gs://agrobot-project.appspot.com'
      )
  );
  runApp(const MyApp());
  SystemChrome.setPreferredOrientations([DeviceOrientation.landscapeLeft]);
}

  Color myColor=const Color(0xFF02AA6D);

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: const Color(0xFF02AA6D)),
        useMaterial3: true,
      ),

      home: const SplashScreen(),
    );
  }
}
```

6. linux

generated_plugin_registrant

```
//  Generated file. Do not edit.
// clang-format off
#include "generated_plugin_registrant.h"
#include <file_selector_linux/file_selector_plugin.h>

void fl_register_plugins(FlPluginRegistry* registry) {
  g_autoptr(FlPluginRegistrar) file_selector_linux_registrar =
      fl_plugin_registry_get_registrar_for_plugin(registry, "FileSelectorPlugin");
  file_selector_plugin_register_with_registrar(file_selector_linux_registrar);
}
```

main.cpp

```
#include "my_application.h"
int main(int argc, char** argv) {
  g_autoptr(MyApplication)  app = my_application_new();
  return g_application_run(G_APPLICATION(app),  argc, argv);
}
```

my_application.cpp

```
#include "my_application.h"

#include <flutter_linux/flutter_linux.h>
#ifdef GDK_WINDOWING_X11
#include <gdk/gdkx.h>
#endif

#include "flutter/generated_plugin_registrant.h"

struct _MyApplication  {
  GtkApplication  parent_instance;
  char** dart_entrypoint_arguments;
};

G_DEFINE_TYPE(MyApplication,  my_application,  GTK_TYPE_APPLICATION)
```

```
// Implements GApplication::activate.
static void my_application_activate(GApplication* application) {
  MyApplication* self = MY_APPLICATION(application);
  GtkWindow* window =
      GTK_WINDOW(gtk_application_window_new(GTK_APPLICATION(applicat
ion)));

  // Use a header bar when running in GNOME as this is the common style used
  // by applications and is the setup most users will be using (e.g. Ubuntu
  // desktop).
  // If running on X and not using GNOME then just use a traditional title bar
  // in case the window manager does more exotic layout, e.g. tiling.
  // If running on Wayland assume the header bar will work (may need changing
  // if future cases occur).
  gboolean use_header_bar = TRUE;
#ifdef GDK_WINDOWING_X11
  GdkScreen* screen = gtk_window_get_screen(window);
  if (GDK_IS_X11_SCREEN(screen)) {
    const gchar* wm_name = gdk_x11_screen_get_window_manager_name(screen);
    if (g_strcmp0(wm_name, "GNOME Shell") != 0) {
      use_header_bar = FALSE;
    }
  }
#endif
  if (use_header_bar) {
    GtkHeaderBar*  header_bar = GTK_HEADER_BAR(gtk_header_bar_new());
    gtk_widget_show(GTK_WIDGET(header_bar));
    gtk_header_bar_set_title(header_bar, "agrobot");
    gtk_header_bar_set_show_close_button(header_bar, TRUE);
    gtk_window_set_titlebar(window, GTK_WIDGET(header_bar));
  } else {
    gtk_window_set_title(window, "agrobot");
  }

  gtk_window_set_default_size(window, 1280, 720);
  gtk_widget_show(GTK_WIDGET(window));

  g_autoptr(FlDartProject) project = fl_dart_project_new();
  fl_dart_project_set_dart_entrypoint_arguments(project, self-
>dart_entrypoint_arguments);

  FlView* view = fl_view_new(project);
  gtk_widget_show(GTK_WIDGET(view));
  gtk_container_add(GTK_CONTAINER(window), GTK_WIDGET(view));

  fl_register_plugins(FL_PLUGIN_REGISTRY(view));

  gtk_widget_grab_focus(GTK_WIDGET(view));
}

// Implements GApplication::local_command_line.
```

```
static gboolean my_application_local_command_line(GApplication* application,
gchar*** arguments, int* exit_status) {
  MyApplication* self = MY_APPLICATION(application);
  // Strip out the first argument as it is the binary name.
  self->dart_entrypoint_arguments = g_strdupv(*arguments + 1);

  g_autoptr(GError) error = nullptr;
  if (!g_application_register(application, nullptr, &error)) {
    g_warning("Failed to register: %s", error->message);
    *exit_status = 1;
    return TRUE;
  }

  g_application_activate(application);
  *exit_status = 0;

  return TRUE;
}

// Implements GObject::dispose.
static void my_application_dispose(GObject* object) {
  MyApplication* self = MY_APPLICATION(object);
  g_clear_pointer(&self->dart_entrypoint_arguments, g_strfreev);
  G_OBJECT_CLASS(my_application_parent_class)->dispose(object);
}
```

```
static void my_application_class_init(MyApplicationClass*  klass) {
  G_APPLICATION_CLASS(klass)->activate  = my_application_activate;
  G_APPLICATION_CLASS(klass)->local_command_line  =
my_application_local_command_line;
  G_OBJECT_CLASS(klass)->dispose  = my_application_dispose;
}

static void my_application_init(MyApplication*  self) {}

MyApplication*  my_application_new() {
  return
MY_APPLICATION(g_object_new(my_application_get_type(),"application-id",
APPLICATION_ID,"flags",  G_APPLICATION_NON_UNIQUE,  nullptr));
}
```

7. macos

contents.json

```
{
  "images" : [
    {
      "size" : "16x16",
      "idiom" : "mac",
      "filename" : "app_icon_16.png",
      "scale" : "1x"
    },
    {
```

```
  "size" : "16x16",
  "idiom" : "mac",
  "filename" : "app_icon_32.png",
  "scale" : "2x"
},
{
  "size" : "32x32",
  "idiom" : "mac",
  "filename" : "app_icon_32.png",
  "scale" : "1x"
},
{
  "size" : "32x32",
  "idiom" : "mac",
  "filename" : "app_icon_64.png",
  "scale" : "2x"
},
{
  "size" : "128x128",
  "idiom" : "mac",
  "filename" : "app_icon_128.png",
  "scale" : "1x"
},
{
  "size" : "128x128",
  "idiom" : "mac",
  "filename" : "app_icon_256.png",
  "scale" : "2x"
},
{
  "size" : "256x256",
  "idiom" : "mac",
  "filename" : "app_icon_256.png",
  "scale" : "1x"
},
{
  "size" : "256x256",
  "idiom" : "mac",
  "filename" : "app_icon_512.png",
  "scale" : "2x"
},
{
  "size" : "512x512",
  "idiom" : "mac",
  "filename" : "app_icon_512.png",
  "scale" : "1x"
},
{
  "size" : "512x512",
  "idiom" : "mac",
      "filename" : "app_icon_1024.png",
      "scale" : "2x"
    }
  ],
  "info" : {
    "version" : 1,
    "author" : "xcode"
  }
}
```

8. test

widget_test.dart

```
// This is a basic Flutter widget test.
// To perform an interaction with a widget in your test, use the WidgetTester
// utility in the flutter_test package. For example, you can send tap and scroll
// gestures. You can also use WidgetTester to find child widgets in the widget
// tree, read text, and verify that the values of widget properties are correct.

import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

import 'package:agrobot/main.dart';

void main() {
  testWidgets('Counter increments smoke test', (WidgetTester tester) async {
    // Build our app and trigger a frame.
    await tester.pumpWidget(const MyApp());

    // Verify that our counter starts at 0.
    expect(find.text('0'), findsOneWidget);
    expect(find.text('1'), findsNothing);

    // Tap the '+' icon and trigger a frame.
    await tester.tap(find.byIcon(Icons.add));
    await tester.pump();

    // Verify that our counter has incremented.
    expect(find.text('0'), findsNothing);
    expect(find.text('1'), findsOneWidget);
  });
}
```

9. web

manifest.json

```
{
    "name": "agrobot",
    "short_name": "agrobot",
    "start_url": ".",
    "display": "standalone",
    "background_color": "#0175C2",
    "theme_color": "#0175C2",
    "description": "A new Flutter project.",
    "orientation": "portrait-primary",
    "prefer_related_applications": false,
    "icons": [
      {
        "src": "icons/Icon-192.png",
        "sizes": "192x192",
        "type": "image/png"
      },
      {
        "src": "icons/Icon-512.png",
        "sizes": "512x512",
        "type": "image/png"
      },
      {
        "src": "icons/Icon-maskable-192.png",
        "sizes": "192x192",
        "type": "image/png",
        "purpose": "maskable"
      },
      {
        "src": "icons/Icon-maskable-512.png",
        "sizes": "512x512",
        "type": "image/png",
        "purpose": "maskable"
      }
  ]
}
```

index:

```
<!DOCTYPE html>
<html>
<head>
  <!--
    If you are serving your web app in a path other than the root, change the
    href value below to reflect the base path you are serving from.

    The path provided below has to start and end with a slash "/" in order for
    it to work correctly.

    For more details:
    * https://developer.mozilla.org/en-US/docs/Web/HTML/Element/base

    This is a placeholder for base href that will be replaced by the value of
    the `--base-href` argument provided to `flutter build`.
  -->
  <base href="$FLUTTER_BASE_HREF">

  <meta charset="UTF-8">
  <meta content="IE=Edge" http-equiv="X-UA-Compatible">
  <meta name="description" content="A new Flutter project.">

  <!-- iOS meta tags & icons -->
  <meta name="apple-mobile-web-app-capable" content="yes">
  <meta name="apple-mobile-web-app-status-bar-style" content="black">
  <meta name="apple-mobile-web-app-title" content="agrobot">
  <link rel="apple-touch-icon" href="icons/Icon-192.png">

  <!-- Favicon -->
  <link rel="icon" type="image/png" href="favicon.png"/>

  <title>agrobot</title>
  <link rel="manifest" href="manifest.json">

  <script>
    // The value below is injected by flutter build, do not touch.
    const serviceWorkerVersion = null;
  </script>
  <!-- This script adds the flutter initialization JS code -->
  <script src="flutter.js" defer></script>
</head>
<body>
  <script>
    window.addEventListener('load', function(ev) {
      // Download main.dart.js
      _flutter.loader.loadEntrypoint({
        serviceWorker: {
          serviceWorkerVersion: serviceWorkerVersion,
        },
        onEntrypointLoaded: function(engineInitializer) {
          engineInitializer.initializeEngine().then(function(appRunner) {
            appRunner.runApp();
          });
        }
      });
    });
  </script>
</body>
</html>
```

10.windows

utils:

```
#include "utils.h"

#include <flutter_windows.h>
#include <io.h>
#include <stdio.h>
#include <windows.h>
```

```
#include <iostream>

void CreateAndAttachConsole() {
  if (::AllocConsole()) {
    FILE *unused;
    if (freopen_s(&unused, "CONOUT$", "w", stdout)) {
      _dup2(_fileno(stdout), 1);
    }
    if (freopen_s(&unused, "CONOUT$", "w", stderr)) {
      _dup2(_fileno(stdout), 2);
    }
    std::ios::sync_with_stdio();
    FlutterDesktopResyncOutputStreams();
  }
}

std::vector<std::string> GetCommandLineArguments() {
  // Convert the UTF-16 command line arguments to UTF-8 for the Engine to use.
  int argc;
  wchar_t** argv = ::CommandLineToArgvW(::GetCommandLineW(), &argc);
  if (argv == nullptr) {
    return std::vector<std::string>();
  }

  std::vector<std::string> command_line_arguments;
  // Skip the first argument as it's the binary name.
  for (int i = 1; i < argc; i++) {
    command_line_arguments.push_back(Utf8FromUtf16(argv[i]));
  }

  ::LocalFree(argv);

  return command_line_arguments;
}

std::string Utf8FromUtf16(const  wchar_t* utf16_string) {
  if (utf16_string == nullptr) {
    return std::string();
  }
  int target_length = ::WideCharToMultiByte(
      CP_UTF8, WC_ERR_INVALID_CHARS, utf16_string,
      -1, nullptr, 0, nullptr, nullptr)
    -1; // remove the trailing null character
  int input_length = (int)wcslen(utf16_string);
  std::string utf8_string;
  if (target_length <= 0 || target_length > utf8_string.max_size()) {
    return utf8_string;
  }
  utf8_string.resize(target_length);
  int converted_length = ::WideCharToMultiByte(
      CP_UTF8, WC_ERR_INVALID_CHARS, utf16_string,
      input_length, utf8_string.data(), target_length, nullptr, nullptr);
  if (converted_length == 0) {
    return std::string();
  }
  return utf8_string;
}
```

main:

```
#include <flutter/dart_project.h>
#include <flutter/flutter_view_controller.h>
#include <windows.h>

#include "flutter_window.h"
#include "utils.h"

int APIENTRY wWinMain(_In_ HINSTANCE instance, _In_opt_ HINSTANCE
prev, _In_ wchar_t *command_line, _In_ int show_command) {
  // Attach to console when present (e.g., 'flutter run') or create a
  // new console when running with a debugger.
  if (!::AttachConsole(ATTACH_PARENT_PROCESS) && ::IsDebuggerPresent())
{
    CreateAndAttachConsole();
  }
```

```
// Initialize COM, so that it is available for use in the library and/or
// plugins.
::CoInitializeEx(nullptr, COINIT_APARTMENTTHREADED);

flutter::DartProject project(L"data");

std::vector<std::string> command_line_arguments =
   GetCommandLineArguments();

project.set_dart_entrypoint_arguments(std::move(command_line_arguments));

FlutterWindow window(project);
Win32Window::Point origin(10, 10);
Win32Window::Size size(1280, 720);
if (!window.Create(L"agrobot", origin, size)) {
  return EXIT_FAILURE;
}
window.SetQuitOnClose(true);

::MSG msg;
while (::GetMessage(&msg, nullptr, 0, 0)) {
  ::TranslateMessage(&msg);
  ::DispatchMessage(&msg);
}
```

```
  ::CoUninitialize();
  return EXIT_SUCCESS;
}
```

flutter_window:

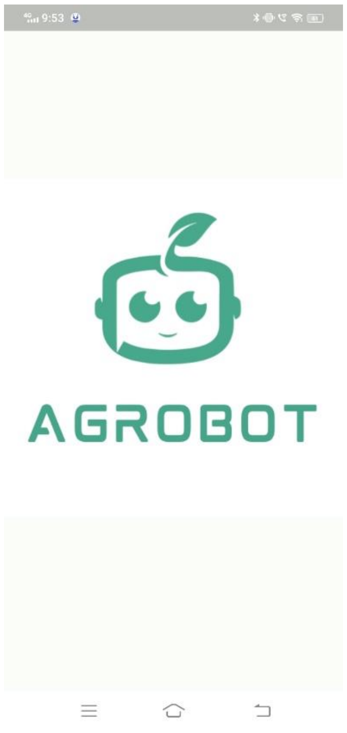
```
#include "flutter_window.h"

#include <optional>

#include "flutter/generated_plugin_registrant.h"

FlutterWindow::FlutterWindow(const flutter::DartProject& project)
    : project_(project) {}

FlutterWindow::~FlutterWindow() {}

bool FlutterWindow::OnCreate() {
  if (!Win32Window::OnCreate()) {
    return false;
  }

  RECT frame = GetClientArea();

  // The size here must match the window dimensions to avoid unnecessary surface
  // creation / destruction in the startup path.
  flutter_controller_ = std::make_unique<flutter::FlutterViewController>(
      frame.right - frame.left, frame.bottom - frame.top, project_);
  // Ensure that basic setup of the controller was successful.
  if (!flutter_controller_->engine() || !flutter_controller_->view()) {
    return false;
  }
  RegisterPlugins(flutter_controller_->engine());
  SetChildContent(flutter_controller_->view()->GetNativeWindow());

  flutter_controller_->engine()->SetNextFrameCallback([&]() {
    this->Show();
  });

  // Flutter can complete the first frame before the "show window" callback is
  // registered. The following call ensures a frame is pending to ensure the
  // window is shown. It is a no-op if the first frame hasn't completed yet.
  flutter_controller_->ForceRedraw();

  return true;
}

void FlutterWindow::OnDestroy() {
  if (flutter_controller_) {
    flutter_controller_ = nullptr;
  }
  Win32Window::OnDestroy();
}

LRESULT
FlutterWindow::MessageHandler(HWND hwnd, UINT const message,
                              WPARAM const wparam,
                              LPARAM const lparam) noexcept {
  // Give Flutter, including plugins, an opportunity to handle window messages.
  if (flutter_controller_) {
    std::optional<LRESULT> result =
        flutter_controller_->HandleTopLevelWindowProc(hwnd, message, wparam,
                                                      lparam);
    if (result) {
      return *result;
    }
  }

  switch (message) {
    case WM_FONTCHANGE:
      flutter_controller_->engine()->ReloadSystemFonts();
      break;
  }

  return Win32Window::MessageHandler(hwnd, message, wparam, lparam);
```

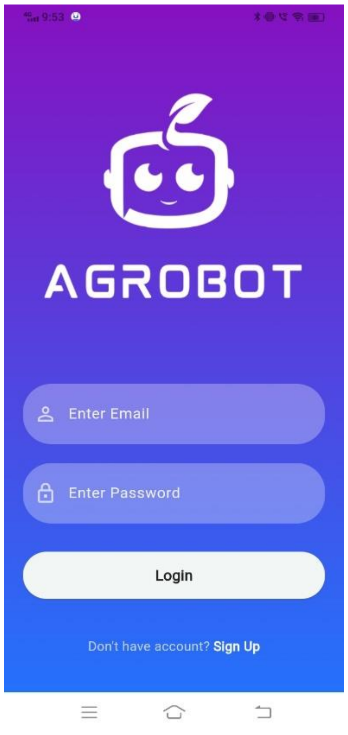
```
}
```

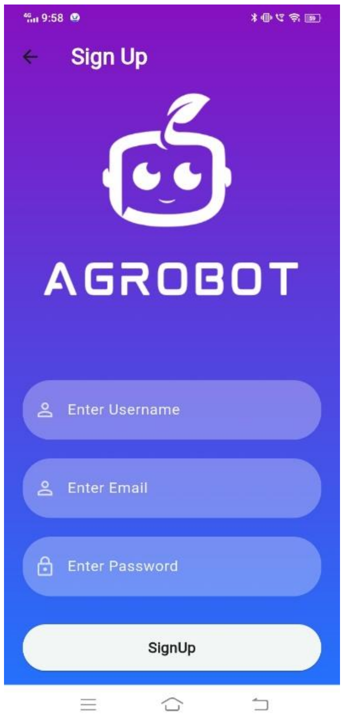
generated_plugin_registrant:

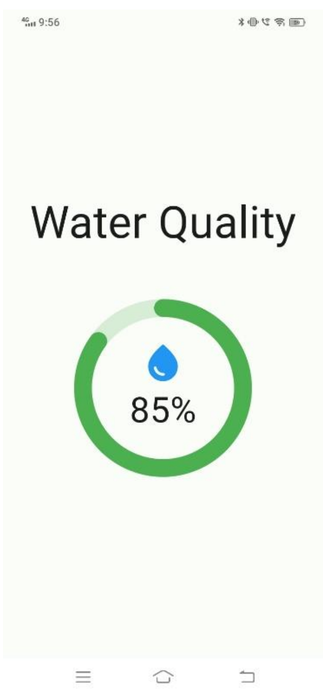
```
// Generated file. Do not edit.
// clang-format off
#include "generated_plugin_registrant.h"

#include <battery_plus_windows/battery_plus_windows_plugin.h>
#include <file_selector_windows/file_selector_windows.h>
#include <firebase_auth/firebase_auth_plugin_c_api.h>
#include <firebase_core/firebase_core_plugin_c_api.h>
#include <firebase_storage/firebase_storage_plugin_c_api.h>
#include <geolocator_windows/geolocator_windows.h>
#include <permission_handler_windows/permission_handler_windows_plugin.h>
#include <rive_common/rive_plugin.h>

void RegisterPlugins(flutter::PluginRegistry* registry) {
  BatteryPlusWindowsPluginRegisterWithRegistrar(
      registry->GetRegistrarForPlugin("BatteryPlusWindowsPlugin"));
  FileSelectorWindowsRegisterWithRegistrar(
      registry->GetRegistrarForPlugin("FileSelectorWindows"));
  FirebaseAuthPluginCApiRegisterWithRegistrar(
      registry->GetRegistrarForPlugin("FirebaseAuthPluginCApi"));
  FirebaseCorePluginCApiRegisterWithRegistrar(
      registry->GetRegistrarForPlugin("FirebaseCorePluginCApi"));
  FirebaseStoragePluginCApiRegisterWithRegistrar(
      registry->GetRegistrarForPlugin("FirebaseStoragePluginCApi"));
  GeolocatorWindowsRegisterWithRegistrar(
      registry->GetRegistrarForPlugin("GeolocatorWindows"));
  PermissionHandlerWindowsPluginRegisterWithRegistrar(
      registry->GetRegistrarForPlugin("PermissionHandlerWindowsPlugin"));
  RivePluginRegisterWithRegistrar(
      registry->GetRegistrarForPlugin("RivePlugin"));
}
```

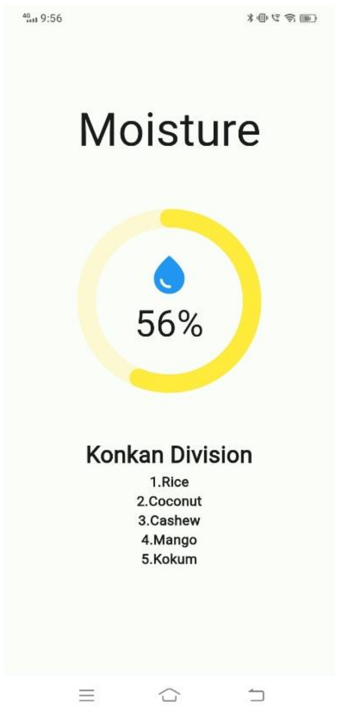
Output Snapshots



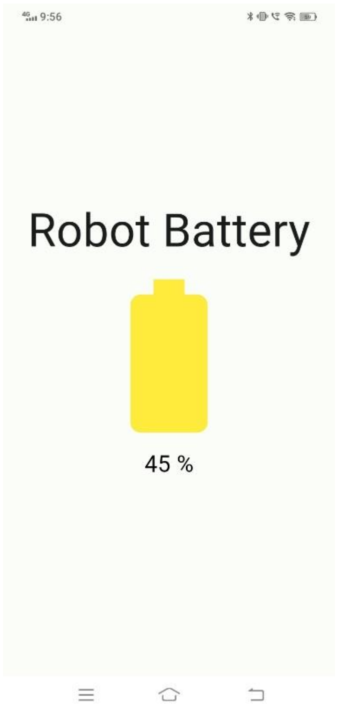


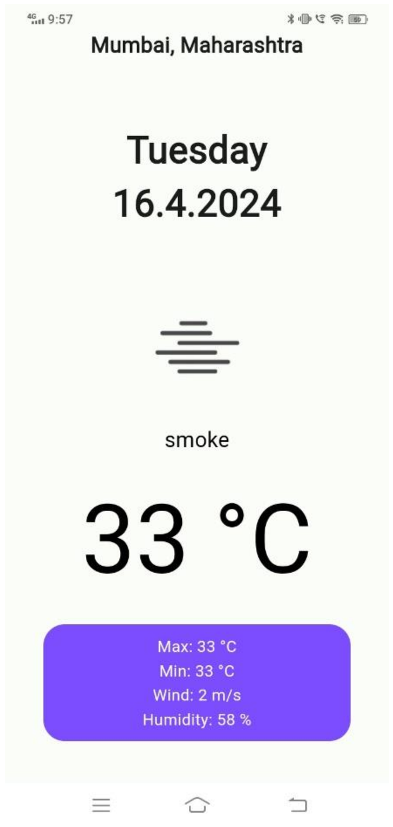






Konkan Division
1.Rice
2.Coconut
3.Cashew
4.Mango
5.Kokum





9:57

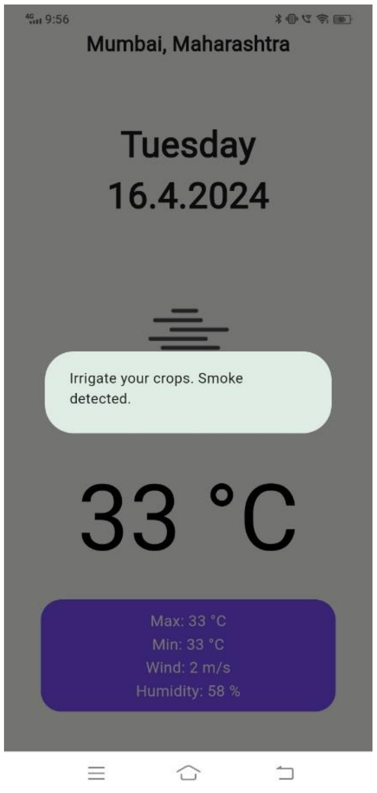
Mumbai, Maharashtra

Tuesday
16.4.2024

smoke

33 °C

Max: 33 °C
Min: 33 °C
Wind: 2 m/s
Humidity: 58 %



9:56

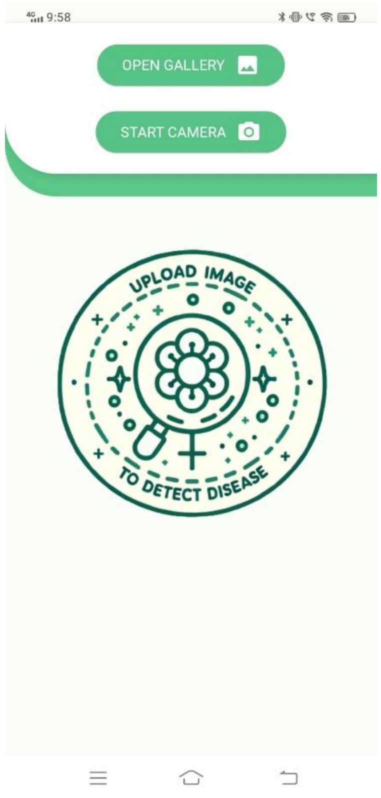
Mumbai, Maharashtra

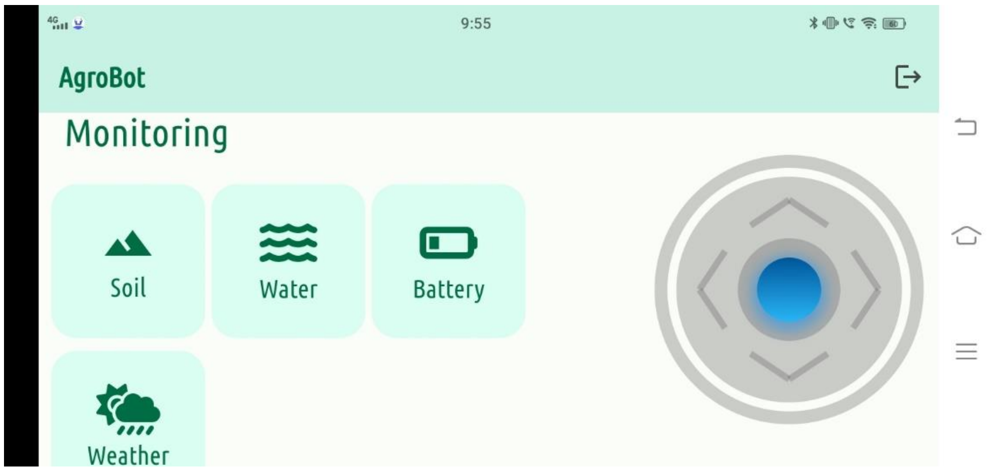
Tuesday
16.4.2024

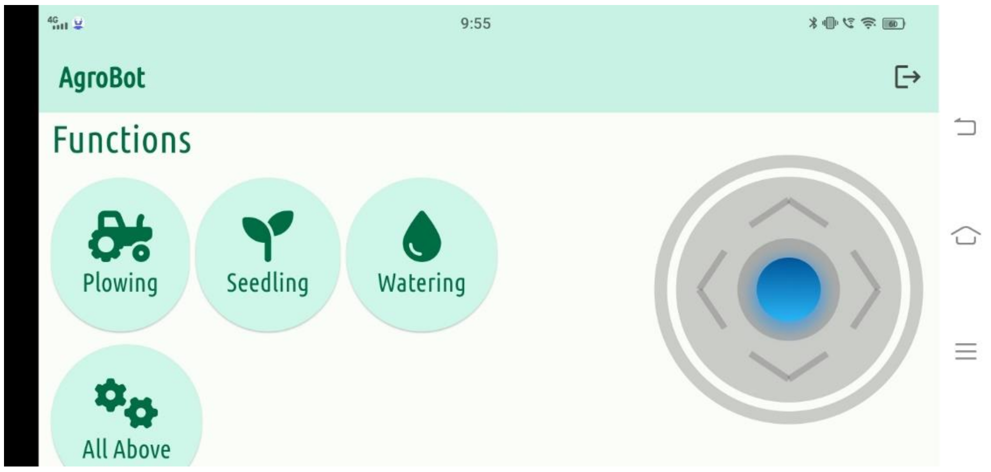
Irrigate your crops. Smoke detected.

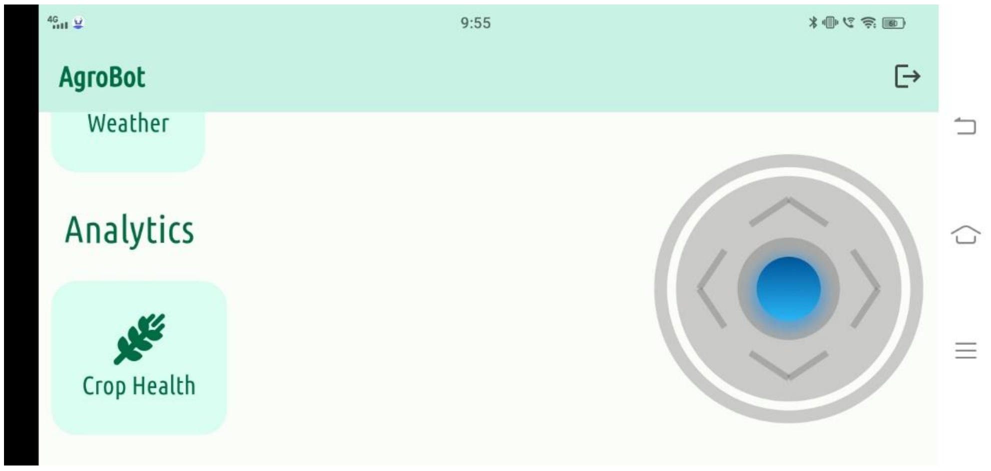
33 °C

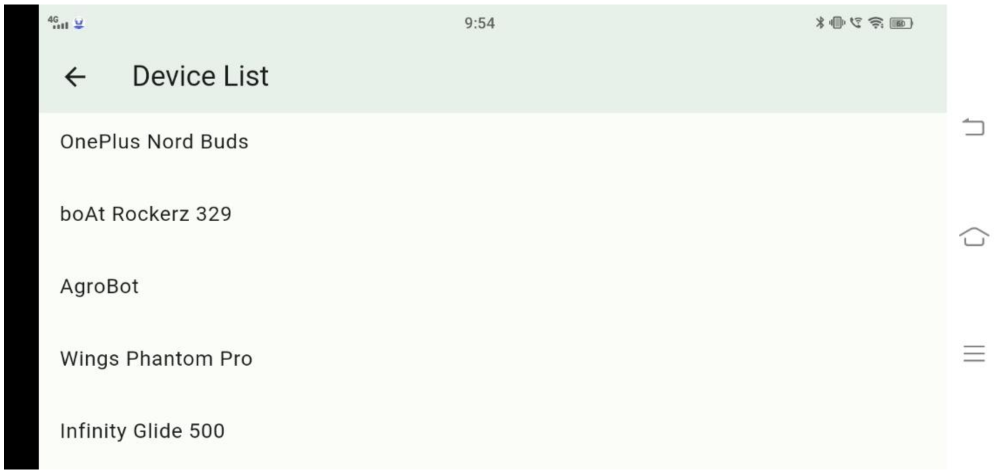
Max: 33 °C
Min: 33 °C
Wind: 2 m/s
Humidity: 58 %











OnePlus Nord Buds

boAt Rockerz 329

AgroBot

Wings Phantom Pro

Infinity Glide 500

Limitations

1. **Adaptability to Crop Varieties:**

Different crops may require different cultivation techniques. The robot's ability to adapt to various crop types and farming methods may be limited.

2. **Disease Recognition Limitations:**

While using the mobile app and camera for disease identification is innovative, the accuracy of disease detection may be influenced by factors such as lighting conditions, camera quality, and the diversity of diseases.

3. **Limited Manipulation Abilities:**

The robot's ability to perform more intricate tasks beyond plowing, seeding, and watering (e.g., pruning, harvesting delicate crops) may be limited, requiring additional human intervention for certain activities.

4. **Data Security and Privacy Concerns:**

Collecting and transmitting sensitive data through the mobile app may raise concerns about data security and privacy. Ensuring secure communication and storage of information is crucial.

5. **Initial Cost and Maintenance:**

The initial investment in such advanced robotics technology can be high. Additionally, regular maintenance and software updates are necessary to keep the system operational, adding to ongoing costs.

Applications

1. **Water Quality Monitoring:**

- The robot assesses water quality parameters such as pH, electrical conductivity (EC), dissolved oxygen (DO), turbidity, and nutrient levels.
- Applications include monitoring water sources for irrigation, ensuring that the water used for crops meets quality standards**.**

2. Soil Quality Monitoring:

- The robot collects data on soil moisture, pH, and nutrient levels using integrated soil sensors.
- It helps farmers understand the condition of the soil, enabling optimized irrigation and nutrient delivery.

3. Weather Monitoring and Forecasting:

- The robot integrates weather sensors to gather real-time weather data, including temperature, humidity, and rainfall.
- Farmers can plan agricultural activities based on accurate weather forecasts, optimizing resource utilization.

4. Crop Health Monitoring:

- The robot employs the Normalized Difference Vegetation Index (NDVI) algorithm to assess crop health using multispectral imagery.
- NDVI values indicate the density and health of vegetation, helping farmers identify stressed or unhealthy areas.

5. Plowing:

- The robot is likely equipped with a robust chassis, wheels, and motors capable of navigating through the agricultural field.
- It can autonomously perform plowing tasks, breaking and turning over the soil to prepare it for planting.

6. Seeding:

- The agricultural robot can carry and deploy seeding mechanisms as part of its functionality.

- It can autonomously distribute seeds across the plowed soil, ensuring even spacing and proper coverage for optimal crop growth.

7. Spraying:

- The robot is equipped with a blower pump and air blower fan, suggesting capabilities for spraying substances.
- It can be programmed to spray fertilizers, pesticides, or other agricultural inputs as needed for crop health.

8. Watering:

- The blower pump and air blower fan may also be used for irrigation purposes.
- The robot can water the crops as it moves through the field, ensuring that plants receive the appropriate amount of water for optimal growth.

Future Scope

1. **Global Adoption and Adaptation:**

The project could be adapted to suit different climates, crops, and farming practices globally. Collaboration with local agricultural experts can help customize the robot's functionalities to meet the specific needs of diverse farming environments.

2. **Economic Viability:**

Continuous efforts to improve cost-effectiveness and scalability can contribute to the economic viability of the project. Reducing initial investment costs and ensuring a favorable return on investment for farmers will be crucial for widespread adoption.

3. **Educational and Training Programs:**

Developing educational programs and training initiatives to help farmers understand and effectively use the technology can contribute to successful adoption. Providing support and resources for farmers transitioning to autonomous farming practices is essential.

4. **Precision Agriculture Techniques:**

The integration of precision agriculture techniques, such as variable rate technology and precision seeding, can improve the overall efficiency of the robot. It allows for targeted application of resources based on real-time data, optimizing yield and resource utilization.

5. **Autonomous Navigation and Coordination:**

Advancements in autonomous navigation systems can enable multiple robots to work collaboratively in the same field. This coordination can lead to increased efficiency and faster completion of tasks.

6. **Expandable Functionality:**

Future versions of the robot could incorporate additional functionalities such as harvesting, pruning, or pest control. This expansion of capabilities would make the robot more versatile and valuable for farmers.

Conclusion

A multifunctional agriculture robot with advanced technological features is revolutionizing modern farming practices. It allows farmers to remotely control plowing, seeding, and watering operations through a user-friendly mobile app, providing real-time monitoring of battery life, water quality, soil conditions, and weather updates. The app also incorporates a camera-based disease identification system, enhancing crop management precision and enabling proactive measures to safeguard crop health. The robot uses solar panels and a MPPT circuit to reduce environmental impact and ensure uninterrupted operation. This eco-friendly approach reduces environmental impact and ensures uninterrupted operation through renewable energy. The integration of this technology and sustainable power source promises to revolutionize farming methods, maximize yields, and contribute to a more sustainable and resilient future for the agriculture industry.

References

- https://www.researchgate.net/publication/283331223_Agriculture_monitoring_system_A_study
- https://www.sciencedirect.com/science/article/pii/S2666285X2100090X
- https://www.sciencedirect.com/science/article/pii/S2666154323003873
- https://ieeexplore.ieee.org/document/7831342
- https://www.researchgate.net/publication/372523093_Smart_Sensing_Technologies_for_Crop_Health_Monitoring_and_Disease_Detection