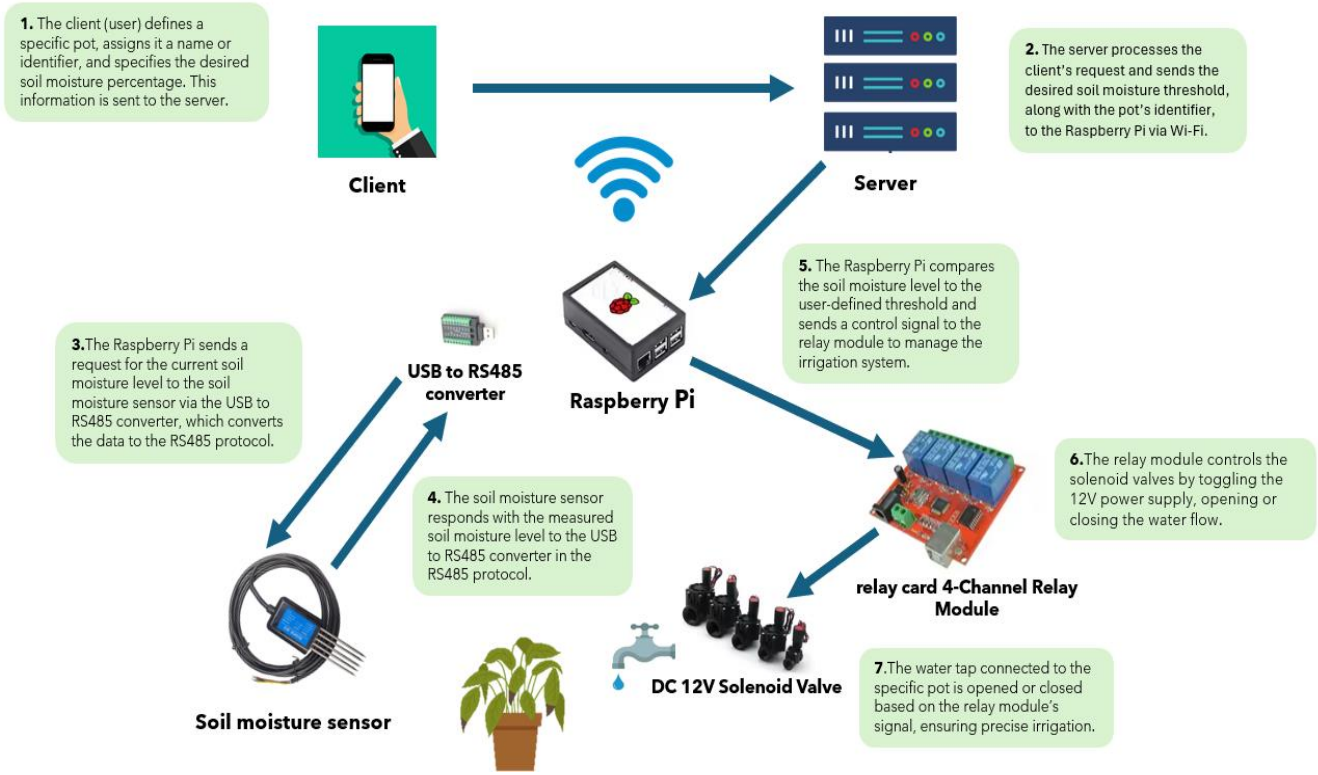
Smart Garden Project: Comprehensive Overview

Yael Yakobovich, Elizabeth Ashurov

Table of Contents

Project Overview

The Smart Garden Project is an innovative system designed to automate and optimize garden maintenance. Leveraging real-time sensor data, weather forecasts, and automated irrigation, this system ensures that plants receive the right amount of water and care at the right time. This project aims to simplify gardening for users while conserving water and promoting plant health.



Objective:

The primary goal of the Smart Garden Project is to provide an efficient, user-friendly solution for maintaining a garden. It is especially valuable for individuals with limited gardening experience or time. By integrating technology with gardening, the project **seeks to:**

1. Monitor environmental conditions in real-time.
2. Automate irrigation based on sensor data and weather forecasts.
3. Provide actionable insights and recommendations to users.
4. Enhance water conservation and plant care efficiency.

Key Features:

Real-Time Monitoring:

- Sensors measure soil moisture, temperature, light intensity, and humidity.
- Data is transmitted to a central server for analysis.

Automated Irrigation:

- The system activates water pumps based on soil moisture levels and weather data.
- Prevents overwatering and ensures optimal hydration.

Weather Integration:

- Fetches real-time weather forecasts from OpenWeatherMap.
- Adjusts irrigation schedules based on rainfall probability and temperature.

Mobile App Interface:

- Displays plant health and sensor data.
- Allows users to manually control irrigation and set thresholds.
- Provides a feature to upload a photo of a plant for identification.
- Identify the plant species and provide care recommendations, health status, and alerts about potential issues.

Historical Data and Trends:

- Tracks sensor data over time to provide insights into plant health and growth.

Functional Flow

1.Data Collection:

- Sensors measure environmental conditions (e.g., soil moisture).
- The Raspberry Pi reads sensor data.
- Data is processed locally and sent to the server for further analysis.

2. Weather Integration:

- The server fetches weather forecasts using the OpenWeatherMap API.
- Forecast data is combined with sensor readings to decide irrigation needs.

3. Decision-Making:

- The server evaluates the following conditions:
- Soil moisture below threshold.
- High temperature or dry air.
- Rainfall probability.
- Based on the analysis, the server sends irrigation commands to the Raspberry Pi.

4. Irrigation Control:

- The Raspberry Pi activates water pumps for specific durations.
- It monitors the soil moisture level during and after irrigation to ensure optimal watering.

5. User Interaction:

- The mobile app fetches real-time data from the server.
- Users can view plant status, trends, and system alerts.
- Manual control options allow users to override automatic decisions.
- Allows users to upload plant images and receive tailored insights, care tips, and notifications about potential issues such as dryness or diseases.

User Scenarios:

Scenario 1: Managing Soil Moisture

- The user receives an app alert: "Soil moisture is low. It is recommended to water the plant."
    - The user opens the app, views the soil moisture level, and manually activates irrigation.
    - Alternatively, the system automatically waters the plant and updates the user.

Scenario 2: Plant Identification

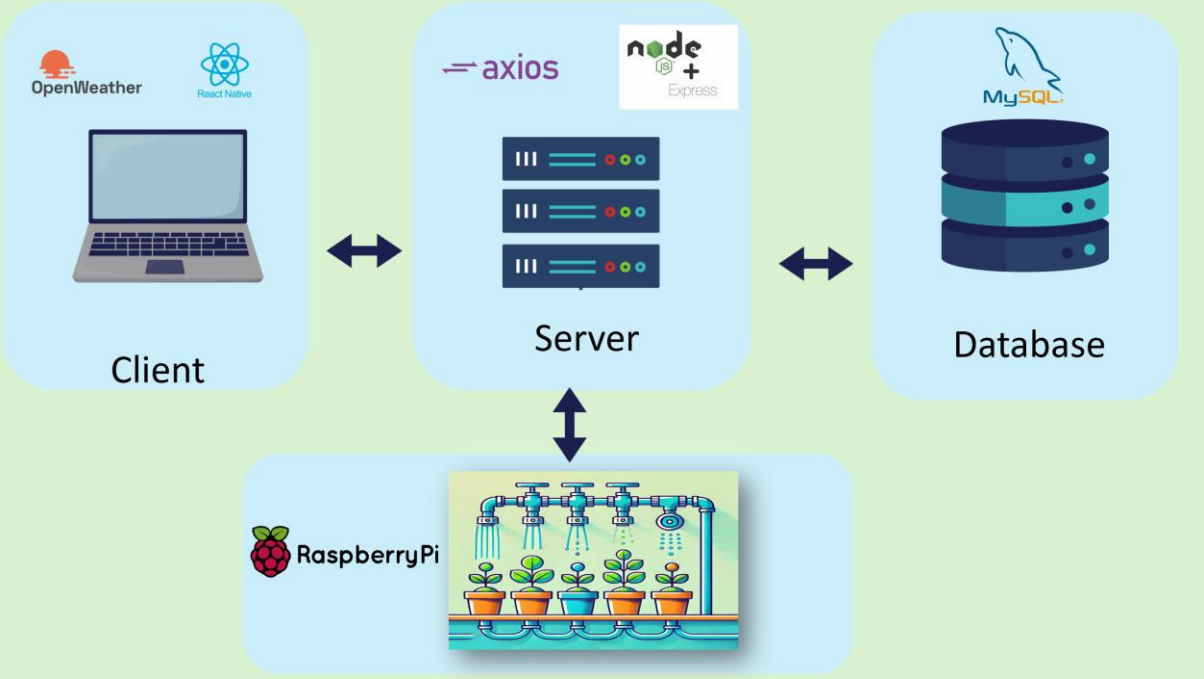
- The user uploads a photo of a plant using the app.
    - The app identifies the plant and provides customized recommendations, such as watering frequency, light needs, and soil requirements.

Scenario 3: Monitoring Weather and Making Decisions

- The app notifies the user of an upcoming rain forecast and pauses the irrigation schedule to conserve water.

High-Level Design

System Architecture



The system consists of the following key components:

Sensors:

- Soil Moisture Sensors: Measure the water content in the soil.

Raspberry Pi (Local Gateway):

- Acts as the central hub for collecting sensor data.
- Runs scripts to process data and interact with the server.
- Sends commands to actuators like water pumps.

Server:

- Handles data storage and processing.
- Fetches weather data from OpenWeatherMap.
- Provides an API for the mobile app to access plant data and control functions.

Mobile Application:

- Built using React Native for cross-platform compatibility.
- Displays real-time data and trends.

- Allows users to control irrigation and configure thresholds.

Database (MySQL):

- Stores sensor readings, user preferences, and historical data.
- Ensures data is accessible for trend analysis and decision-making.

Technology Stack

1.Hardware:

Raspberry Pi:

- Acts as the central controller.
- Handles sensor inputs and actuator outputs.

Sensors:

- Soil moisture sensor with RS485 interface(Model: JXBS-3001-LS RS485 Soil Moisture Sensor)

USB to RS485 converter

- Converts the computer's USB port to an RS485 port.

relay card 4-Channel Relay Module:

- Includes 4 relays to operate water taps or other devices.

DC 12V Solenoid Valve (Normally Closed):

- Controlled by DC 12V voltage.

power supply

- Model: DC Power Supply Adapter 12V/2A
- Suitable for operating the taps and relays.

2. Software:

Backend:

- Node.js + Express: Handles API requests and server logic.
- Python: Used on the Raspberry Pi for sensor control and communication.

  Database:
- MySQL: Stores sensor data, weather data, and user preferences.

Mobile App:

- React Native: Provides a cross-platform mobile interface.
- Axios: Handles API communication.

APIs:

- OpenWeatherMap: Fetches weather forecasts.
- Plant.id: Identifies plant species from user-uploaded photos and provides care recommendations and health assessments.

Smart Garden Project: SQL Database Structure

Below is the SQL representation of the database structure for the Smart Garden project, organized as tables and fields.

Users Table

- **Fields:**
    - user_id (Primary Key): Unique identifier for each user.
    - username: Name of the user.
    - password_hash: Encrypted password for secure login.
    - preferences: User-specific preferences (stored as JSON).

Plants Table

- **Fields:**
    - plant_id (Primary Key): Unique identifier for each plant.
    - name: Common or scientific name of the plant.
    - recommended_conditions: Ideal environmental parameters (stored as JSON).
    - user_id (Foreign Key): Links the plant to its owner.
    - valve_id (Foreign Key): Links the plant to its irrigation valve.

Sensor Data Table

- **Fields:**
    - sensor_id (Primary Key): Identifier for the sensor.
    - timestamp: Timestamp of data collection.
    - moisture: Soil moisture level.
    - plant_id (Foreign Key): Links the data to a specific plant.

Irrigation Commands Table

- **Fields:**
    - command_id (Primary Key): Unique identifier for each command.
    - timestamp: Time the command was issued.
    - valve_id (Foreign Key): The valve to activate.
    - plant_id (Foreign Key): Links the command to the specific plant being watered.
    - duration: Duration of watering in seconds.
    - status: Status of the command (e.g., executed, failed).

Irrigation Valves Table

- **Fields:**
    - valve_id (Primary Key): Unique identifier for the irrigation valve.
    - location: Location of the valve.
    - status: Status of the valve (e.g., active, inactive).

Alerts Table

- **Fields:**
    - alert_id (Primary Key): Unique identifier for the alert.
    - timestamp: Time the alert was generated.
    - alert_type: Type of alert (e.g., low moisture).
    - user_id (Foreign Key): Links the alert to a user.

Features of the Application

1. **Main Dashboard (Home):**
    - Provides an overview of plant conditions, including health status, moisture levels, and watering status.
    - Quick access to important notifications, such as irrigation activities or system alerts.
2. **Plant List (Plants):**
    - Displays a list of all plants in the system.
    - Each plant includes:
        - A photo, name, and health status indicator (Good/Moderate/Bad).
    - Clicking on a plant opens a detailed view with more information.
3. **Plant Details:**

    Provides a detailed view of a specific plant:

    - **Large photo of the plant.**
    - **Health status (Good/Moderate/Bad):** Based on current conditions compared to recommended thresholds.
    - **Current Conditions:**
        - **Moisture Level:** Current level compared to recommended range (fetched from a AI analysis).
        - **Sunlight Exposure:** Current hours vs. recommended daily sunlight.
        - **Temperature Range:** Current temperature vs. the plant's ideal range.
    - **Historical Data Graphs:** Displays trends in moisture, sunlight, and temperature over time.
    - **"Water Now" Button:** Allows manual irrigation for the specific plant.

4. **Notifications (in Home screen):**
    - List of recent system notifications, such as:
        - "The basil plant was watered for 5 minutes."
        - "Soil moisture is too low for the mint plant."
    - Push notifications to alert users directly on their devices.
5. **Upload Image for AI Analysis:**
    - **Image Upload Screen:**
        - Users can upload a plant photo for analysis.
    - **AI-Based Analysis:**
        - Identifies the plant type based on the uploaded image.
        - Detects potential problems (e.g., dry soil, yellowing leaves).
        - Provides actionable recommendations.
6. **Irrigation Management:**
    - **Automated Irrigation:**
        - Triggers based on sensor data, such as soil moisture and temperature.
    - **Manual Irrigation:**
        - Users can activate irrigation manually from the Plant Details screen.
        - Options to set the duration for irrigation.
7. **Graphs and Historical Data:**

    The historical data section will include the following graphs (displayed within the Plant Details page):

    - **Soil Moisture Levels:**
        - Displays changes in the soil moisture percentage over time for each plant.

- **Irrigation History:**
    - Visualizes irrigation activity, including Dates and times of irrigation and Duration of each watering session.

8. **User Settings:**
    - Personalization options:
        - Enable or disable notifications.
        - Update Plant Setting: Allows users to modify the soil moisture levels for individual plants
9. **Add New Plant:**
    - **Manual Entry:**
        - Users can manually input plant details, such as name, photo, and Ideal Moisture Percentage.
    - **Image Analysis:**
        - Upload a photo of the plant for AI-based identification.
        - Automatically fetch recommended conditions for the plant based on its type.

Steps for Building the Automated Irrigation System Application

Step 1: Setting Up the Raspberry Pi

The initial setup includes connecting a keyboard, monitor, and mouse to the Raspberry Pi for configuration purposes. The following tasks are completed during this phase:

1. **Operating System Installation:**
    - An operating system is downloaded and installed using the Raspberry Pi Imager tool.
2. **Network Configuration:**
    - The Raspberry Pi is connected to a Wi-Fi to enable software installation and communication with external systems.
3. **Environment Preparation:**
    - Python and necessary libraries are installed to support hardware integration and sensor data processing.

Step 2: Software and Hardware Integration

Sub-step 1: Connecting the RS485 Converter and Sensors

1. **RS485 Converter Setup:**
    - The RS485 converter is connected to the Raspberry Pi using the USB interface. A suitable driver for the converter is identified and installed to enable communication with the device.
2. **Sensor Integration:**
    - Sensors are wired to the RS485 converter.
3. **Data Reading Script:**
    - A Python script is developed to open the serial connection and read data from the sensors. The script processes the raw data received from the sensors and converts it into meaningful humidity measurements.

4. **Calibration**:
    - The calibration process adjusts the sensor readings to ensure they accurately reflect actual soil moisture levels.
    - This involves comparing the raw data to reference values and applying necessary corrections to achieve reliable measurements.

Sub-step 2: Connecting the Relay Module and Solenoid Valves

1. **Connecting the Relay Module to the Raspberry Pi:**
    - The relay module is connected to the Raspberry Pi via a USB port.
    - A suitable driver is identified and installed to enable the Raspberry Pi to communicate with the relay module through the USB interface.
    - The relay module is connected to an external 12V power supply to provide the necessary voltage for operating the solenoid valves.

2. **Connecting Solenoid Valves**
    - The solenoid valve is connected to the relay module and a 12V power supply to enable automated control.
3. **Control Script Development:**
    - A script is written to manage the opening and closing of valves based on input commands or sensor data.

Step 3: Developing Processes for Valve Operation and Humidity Measurement

A process is designed to manage valve operations based on real-time humidity measurements. This includes:

- Continuously monitoring humidity levels from sensors.
- Implementing thresholds to determine when to open or close the valves.
- Automating the irrigation process while ensuring efficient water usage.

Step 4: Server Integration with the Raspberry Pi

1. **Establishing Communication:**
    - The communication protocol: **WebSocket** is selected to enable data exchange between the Raspberry Pi and the server.
2. **Data Transmission:**
    - A script is developed to send sensor readings and operational logs from the Raspberry Pi to the server.
3. **Command Reception:**
    - The Raspberry Pi is configured to receive and execute commands sent by the server, such as opening or closing a specific valve.

Step 5: Developing the Irrigation Algorithm

1. **Algorithm Logic:**
    - An algorithm is implemented to determine the optimal irrigation schedule based on real-time sensor data, historical trends, and environmental factors (e.g., weather).
2. **Water Efficiency:**
    - The algorithm ensures efficient water use by calculating irrigation duration and frequency based on soil moisture levels.
3. Overcome sensor / valve malfunction including alerting and recovery.

Step 6: Building the Database multiple plants and history capture

1. **Database Design:**
    - A relational database is designed to store sensor data, valve operations, and other relevant information.
2. **Database Integration:**
    - Scripts are developed to log real-time data into the database and enable retrieval for analytics and decision-making.

Step 7: Extending the Server-to-Raspberry Pi

1. **Weather Data Functionality:**
    - The server retrieves real-time weather data from the **OpenWeather** API and sends it to the Raspberry Pi.
2. **Irrigation Scheduling:**
    - The server provides the Raspberry Pi with predefined irrigation schedules. These schedules dictate when irrigation should occur.
3. **Soil Moisture Updates:**
    - The Raspberry Pi periodically sends soil moisture data to the server.
    - The server processes and stores this data.
4. **Irrigation Duration Logging:**
    - After completing an irrigation session, the Raspberry Pi sends the irrigation duration for each plant to the server.

Step 8: Developing the Client-Server API

In this stage, the primary goal is to develop the server's ability to handle client requests and respond with the necessary data. This involves implementing server-side logic for each API endpoint without yet developing the client application. The steps include:

1. Defining API Endpoints:
    - Identify all the endpoints required for client-server communication, such as:
        - User Management: Registration, login, and session validation.
        - Plant Management: Adding plants, retrieving plant data, and updating settings.
        - Sensor Data: Fetching real-time sensor readings and historical data.
        - Irrigation Control: Sending irrigation commands and logging activities.
        - Notifications: Retrieving alerts and managing user preferences.
2. Server-Side Logic:

- Implement the logic for processing incoming requests:
    - Validate request parameters (e.g., user credentials, plant data).
    - Query or update the database as needed.
    - Generate appropriate responses for each endpoint, including error handling.

Step 9: Developing the Application

This stage focuses on building the client-side application that connects to the API developed in the previous step. The application serves as the user interface for managing and monitoring the automated irrigation system.

Development Highlights:

1. **First Feature: User Registration**
    - The initial feature implemented is user registration, ensuring secure and personalized access to the system.

Step-by-Step Workflow

1. **Manual Login:**
    - Users enter their email and password into the login form.
    - The credentials are sent to the server over a secure connection.
    - The server validates the email and password against the database.
    - Upon successful authentication, a session token (e.g., JWT) is generated and sent to the client.
2. **Login with Google (Optional):**
    - Users log in using their Google account.
    - The app redirects to Google's authentication page to retrieve an access token.
    - The server validates the token and either logs in the user or creates a new account if one does not exist.

3. **Session Management:**
    - Session tokens are used for all subsequent interactions with the server.
    - Tokens are validated with each request and expire after a defined period, requiring re-authentication.
4. **Error Handling:**
    - Invalid login attempts return appropriate error messages (e.g., "Invalid email or password").
    - Users can reset their password via a secure email link if needed.

2. **Core Development:**
    - Design and implement key screens:
        - **Dashboard:** Overview of plant conditions, system status, and alerts.
        - **Plant List:** Display all plants with images, names, and health statuses.
        - **Plant Details:** Detailed views with graphs, care recommendations, and irrigation history.
    - Integrate features such as:
        - Real-time notifications for soil moisture, irrigation status, and anomalies.
        - Manual and automated irrigation controls.
        - Graphical representation of historical data for plant metrics.

3. **Final Feature: Plant Identification Using AI**
    - Among the final features to be developed is plant identification using the **Plant.id API**, providing a sophisticated AI-powered analysis tool.

Step-by-Step Workflow

1. Image Upload:
    - Users upload a photo of the plant via the mobile app.
2. AI Analysis:

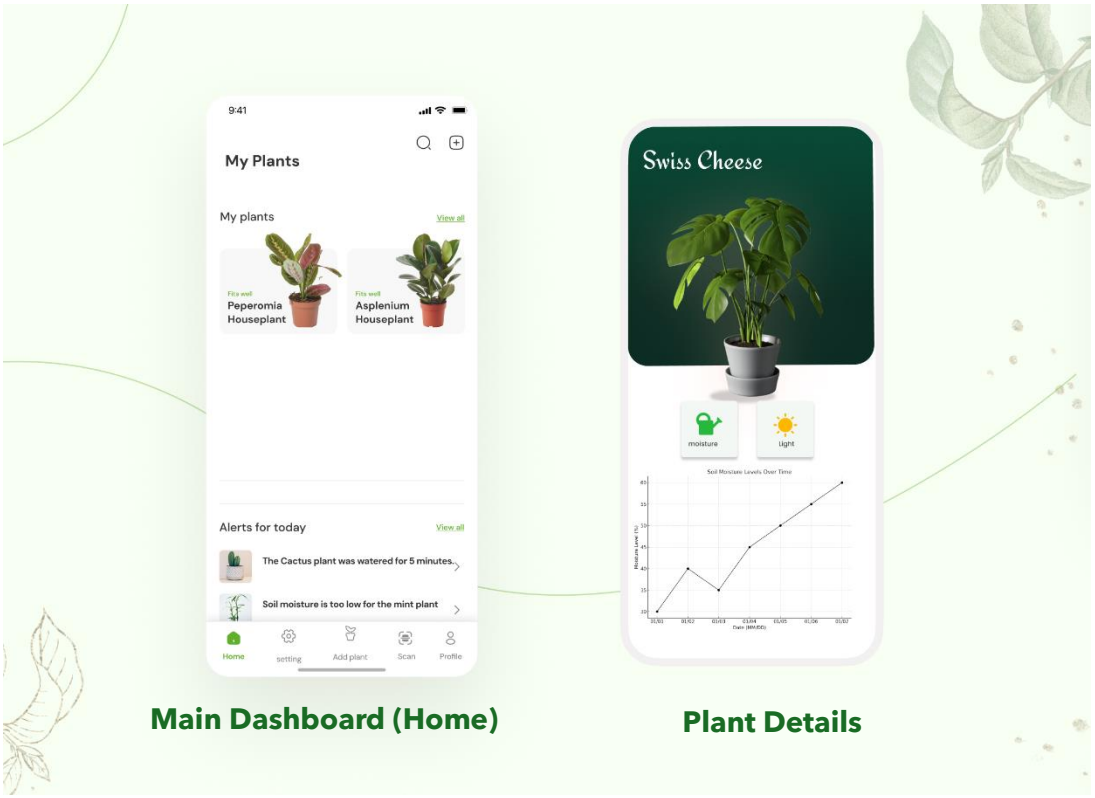
- The app sends the image to the Plant.id API.
- The API returns:
  - Plant species name (e.g., "Basil").
  - Recommended growing conditions:
    - Moisture range (e.g., 40%-70%).
    - Sunlight needs (e.g., Full Sun).
    - Temperature range (e.g., 18°C-25°C).

3. Fallback for Errors:
   - If the API fails to identify the plant:
     - The system prompts the user to manually enter the plant's name and care details.
4. Providing Insights:
   - Display tailored care tips based on plant type.
   - Highlight potential issues, such as dry leaves or inadequate sunlight.

Application Screens – Initial Sketches

The following images represent initial sketches of the application's design. These sketches outline the planned functionality and layout of key screens.

Main Dashboard (Home):



- **Key Buttons and Their Functions:**

1. **Settings Button:**
    - Opens the **User Settings** screen.
    - Allows users to adjust preferences such as enabling/disabling notifications or updating default moisture thresholds for plants.
2. **Add Plant Button:**
    - Opens the **Add Plant** screen.
    - Provides options to manually input plant details or upload a photo for AI-based identification.
3. **Scan Button:**
    - Triggers the **AI Plant Identification** feature.

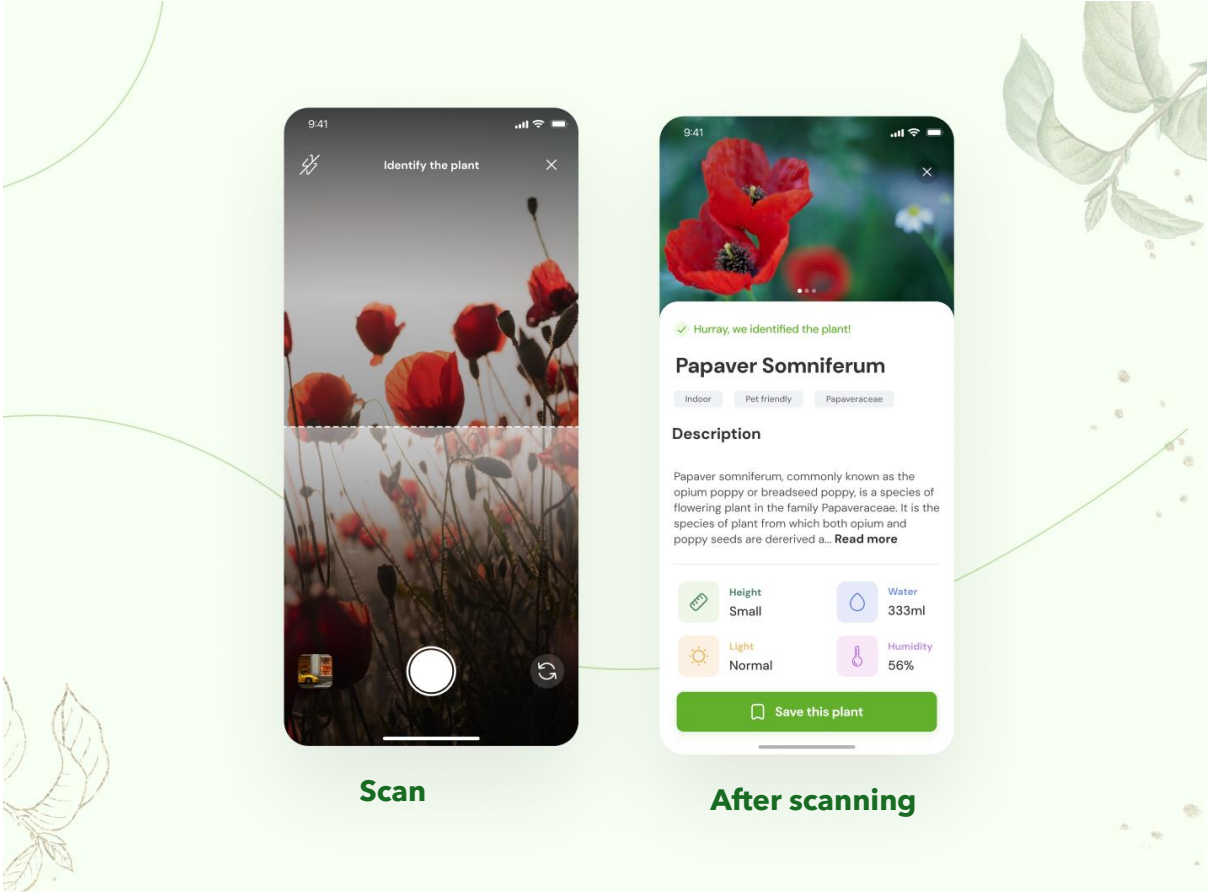
        - Allows users to take a photo for immediate analysis and identification.

- **Additional Features on the Dashboard:**
    - Quick access to plant health summaries.
    - Notifications area for critical system alerts.

Plant Details:

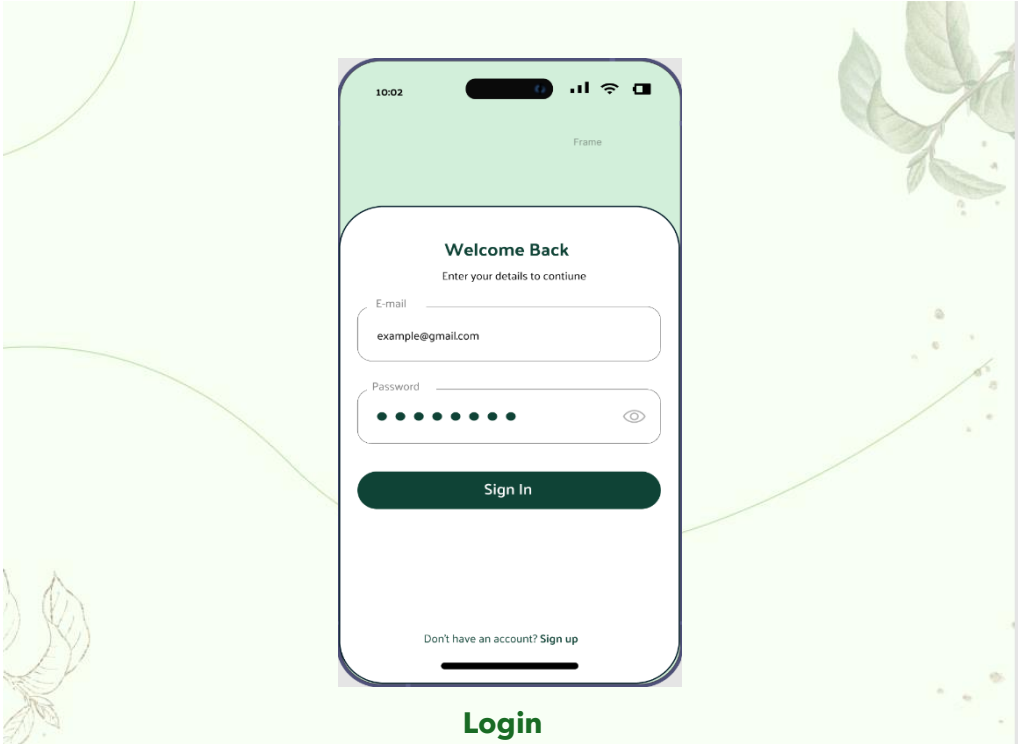
- **Description:**
  The Plant Details page offers an in-depth look at the selected plant's status.
  Key Features:
    - Displays the plant's photo.
    - Plant Health Status: The status is based on soil moisture and sunlight compared to recommended thresholds:
        - Green: Good status
        - Yellow: Moderate status
        - Red: Poor status
    - Includes graphs of historical data of soil moisture and irrigation activity.
    - Provides a "Water Now" button for manual irrigation.

Scan Screen: Plant Image Analysis



- **Description:**
  The **Scan Screen** allows users to take or upload a photo of a plant for AI-based analysis.
- **Key Features:**
  1. **Capture or Upload Image:** Take a photo with the camera or upload an existing image.
  2. **AI Analysis:** Identifies the plant type and detects potential issues.
  3. **Results:** Displays the plant type, care recommendations, and detected problems.

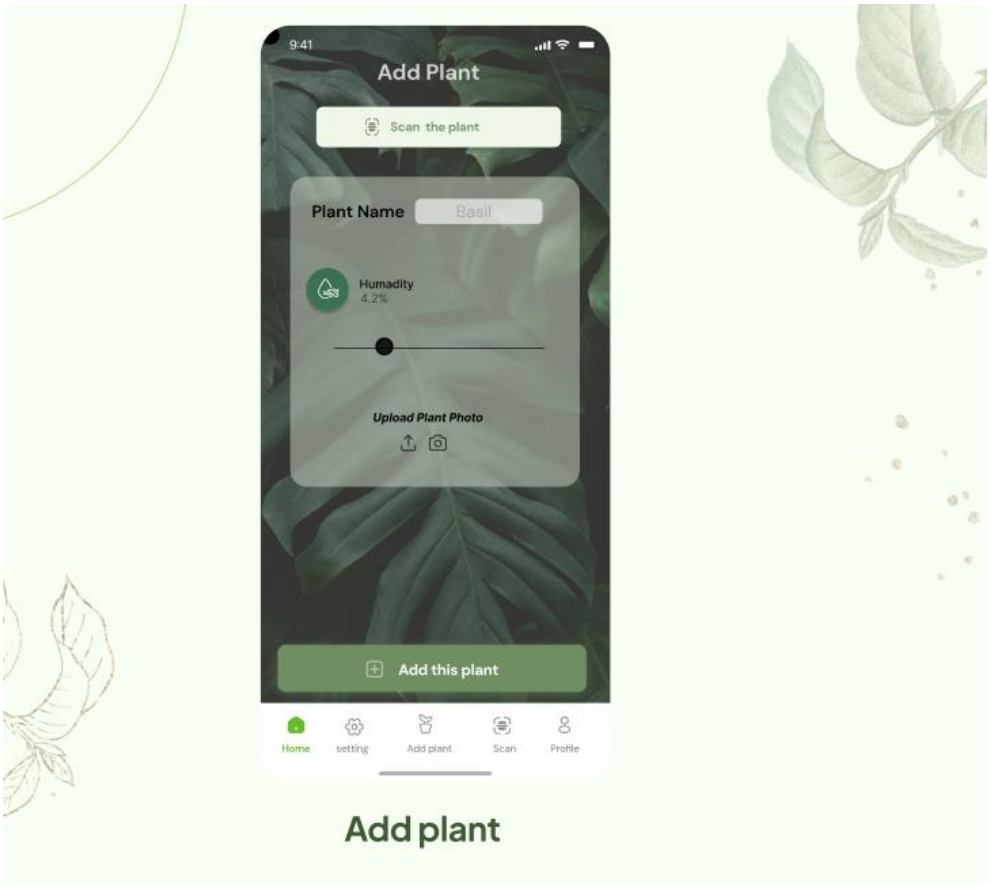
Login Screen



Login

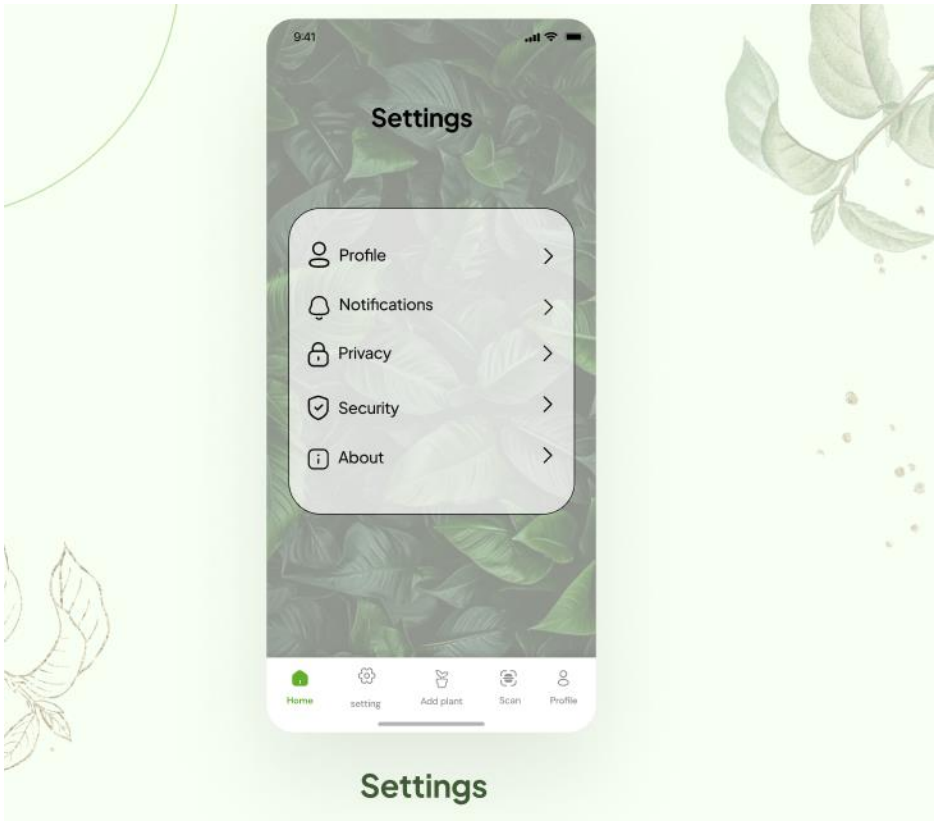
- **Description:**
  The Login screen enables secure access to the application for registered users.
  Key Features:
    - Fields for username and password input.
    - Login button to authenticate users and navigate to the dashboard.
    - Option to recover the password in case of forgotten.

Add Plant Screen



- **Description:**
  The Add Plant screen allows users to add new plants to their smart garden system easily. This screen integrates manual input and AI-based plant identification to provide a seamless user experience.
  Key Features:
    - Users can scan a plant using the camera to identify it and retrieve its name and care details.
    - A text field is provided for entering the plant's name manually if scanning is not used.
    - Users can adjust the target humidity level using a slider to match the plant's ideal requirements.
    - Users can upload a photo of the plant either from their gallery or by taking a new picture using the camera.
    - Finalizes the process and adds the plant to the system with the provided details.

Settings Screen



- **Description:**
  The Settings screen provides users with access to manage their account preferences, application configurations, and general information about the app.
  Key Features:
    - Allows users to view and edit their personal information, such as name and email.
    - Provides options to enable or disable various app notifications, including system alerts and plant health updates.
    - Includes settings for managing data sharing preferences and reviewing the app's privacy policy.
    - Offers features to update passwords, enable two-factor authentication, and review account activity for enhanced security.
    - Displays information about the application.