Gardian - Automated Garden Watering System

Bi Bianca Ciobanu

TABLE OF CONTENTS

- User Story #8 – Notifications and Alerts

Interface description

  - 1. Login Page
  - 2. Registration Page
  - 3. Home Page
  - 4. Plant Recognition Page
  - 5. Plant Profile Management Page
  - 6.Notifications and Alerts Panel

Protocol, Application Data Structure and Message Flow

- Protocol
- Application Data Structure and Message Flow:

Scalability and Security Considerations

  - Machine Specifications
  - Maximum Load
  - Security Measures

Diagrams

- System Architecture Diagram
- Use Case Diagram
- Entity-Relationship Diagram

Prototype

Introduction

App description

The Automated Garden Watering web application is a smart tool designed to help users effectively care for their plants by integrating Arduino technology and user-friendly interfaces. It offers features such as user authentication for secure plant data storage, real-time soil moisture monitoring, comprehensive environmental tracking, automated and manual water pump control, and personalized plant care based on plant

recognition. Users can create detailed profiles for each plant, set watering schedules, and receive notifications for critical conditions, ensuring optimal plant health and maintenance with minimal effort.

Used Technologies

The project is build on Windows 11 and it has the following integrated technologies:

Software

Frontend:

- **ReactJS v18.18.0**: Main framework for building the user interface.
- **Axios**: For making HTTP requests to the backend.
- **Material-UI**: Component libraries for UI design.

Backend:

- **Spring Boot v3.3.2**: Main framework for developing the backend.
- **Spring Data JPA**: For data persistence and database interaction.
- **Spring Security**: For securing the application.
- **WebSockets**: For real-time communication between the server and client.
- **Maven 4.0.0**: Build automation tools.

Database:

- **MySQL Workbench 8.0**

Integration and Communication:

- **REST APIs**: For standard HTTP communication between the frontend and backend.
- **WebSockets**: For real-time, bidirectional communication between the frontend, backend, and ESP32.
- **Postman**: For API testing.
- **Arduino IDE 2.2.1:** for integration of client side communication and sensor data reading.

Hardware

- *ESP32*
- Soil moisture sensor
- Temperature and humidity sensor
- Light sensor

By integrating these technologies, a real-time monitoring and control to ensure plants get the right amount of water will be achieved. While ReactJS and Material-UI provide a dynamic user interface for managing the garden system, Spring Boot ensures a reliable backend. Spring Data JPA handles database interactions, and WebSockets enable real-time communication between the server, client, and ESP32. MySQL Workbench manages the database, ensuring efficient data storage and retrieval. These technologies together create a complex and efficient automated plant watering system.

App Functionalities

User Authentication

Allows user to keep all information regarding its plants stored in one place.

Soil Moisture Monitoring

Involves using soil moisture sensors to measure the moisture levels in the soil. The ESP32 sends this data to the backend server periodically, allowing for real-time monitoring.

Environmental Monitoring

Extends the first functionality by measuring temperature, humidity, and light levels, sending this data to the backend for comprehensive environmental tracking.

Water Pump Control

Based on the moisture levels or direct commands from the server, the ESP32 can activate or deactivate the water pump. This control also includes a manual override function, enabling users to turn the pump on or off via commands from the server.

Plant recognition based on plant name

The application includes a plant recognition algorithm where users can input the name of a plant and retrieve personalized watering information from a database. This

database stores information about various plants, including optimal watering requirements, frequency, and other care instructions.

Profiles for different plants

Users can create and store profiles for different plants in their garden, including details such as plant name, watering frequency, preferred sunlight, and soil type. Each plant profile integrates sensor data to monitor and display real-time information about the plant's status, such as soil moisture levels and temperature.

Scheduling functionalities

Allow automated watering at specific times, giving users a scheduling interface to set their watering preferences.

Notifications and Alerts

Inform users of critical conditions, such as extremely low or high soil moisture levels, or other sensory dysfunctions and connectivity issues within the ESP32 device.

User Stories

User Story #1 – User Authentication

As a user,

I want to be able to authenticate into the application,

So that I can securely access and manage all information regarding my plants in one place.

Acceptance Criteria:

1. **AC#1 – Registration Page**
    - **Scenario 1:** Successful Registration

- **Given that** I am a new user,
- **When** I navigate to the registration page,
- **Then** I should be able to create a new account by filing the form having the following fields:

Field Name	Field Type	Mandatory/Optiona...	Default Valu
username	Text Min 6 characters Max 30 characters	Mandatory	usename

password	Text+Numbers At least 3 digits Min 8 characters Max 30 characters, Case sensitive	Mandatory	password
password confirmation	Same value as in the "password" field	Mandatory	confirm pass
email	Free Text Max 30 characters, Email format	Mandatory	email

    - **Scenario 2:** Registration with Existing Email
- **Given** that I am a new user,
- **When** I navigate to the registration page,
- **And** I enter an email that is already registered,
- **Then** I should receive an error message indicating that the email is already in use.

Field Name	Field Type	Error
Email	Free text	Already registered in the system

    - **Scenario 3:** Password Mismatch
- **Given** that I am a new user,
- **When** I enter a password and a password confirmation that do not match,
- **Then** I should receive an error message indicating that the passwords do not match.
    - **Scenario 4**: Username too short
- **Given** that I am a new user,
- **When** I enter an username that is shorter than 6 characters,
- **Then** I should receive an error indicating that the username is too short.
    - **Scenario 5**: Password too short
- **Given** that I am a new user,
- **When** I enter a password that is shorter than 8 characters,
- **Then** I should receive an error indicating that the password is too short.
    - **Scenario 6**: Password doesn't contain enough requested characters

- **Given** that I am a new user,
- **When** I enter a password that contains less than 3 digits,
- **Then** I should receive an error indicating that the password must contain at least 3 digits.

1. **AC#2 – Login Page**
    - **Scenario 1:** Successful Login

- **Given** that I am a registered user,
- **When** I navigate to the login page,
- **And** I enter my username and password,
- **Then** I should be able to log in successfully.
    - **Scenario 2:** Incorrect Password
- **Given** that I am a registered user,
- **When** I enter an incorrect password,
- **Then** I should receive an error message indicating that the password is incorrect.

Field Name	Field Type	Validation/Requirements
Password	Text+Numbers	Does not match the registered username's password

    - **Scenario 3:** Non-Existent Username
- **Given** that I am a registered user,
- **When** I enter a non-existent username,
- **Then** I should receive an error message indicating that the username is not found.

Field Name	Field Type	Validation/Requirements
Username	Free text	Not registered in the system

User Story #2 – Soil Moisture Monitoring

As a user,

I want to monitor soil moisture levels in real-time,

So that I can ensure my plants are being watered adequately.

Acceptance Criteria:

1. **AC#1 – Real-Time Moisture Data**
    - **Scenario 1:** Successful Data Display

- **Given** that I have soil moisture sensors set up,
- **When** the sensors measure soil moisture,
- **Then** the data should be sent to the backend server and displayed on my plant profile in real-time.

    - **Scenario 2:** Sensor Malfunction
- **Given** that I have soil moisture sensors set up,
- **When** the sensors malfunction or lose connection,
- **Then** I should receive an alert indicating the issue.

User Story #3 – Environmental Monitoring

As a user,

I want to monitor temperature, humidity, and light levels,

So that I can track the environmental conditions affecting my plants.

Acceptance Criteria:

1. **AC#1 – Environmental Data Collection**
    - **Scenario 1:** Successful Data Display
- **Given** that I have environmental sensors set up,
- **When** the sensors measure temperature, humidity, and light levels,
- **Then** the data should be sent to the backend server and displayed on my plant profile.
    - **Scenario 2:** Sensor Malfunction
- **Given** that I have environmental sensors set up,
- **When** the sensors malfunction or lose connection,
- **Then** I should receive an alert indicating the issue.

User Story #4 – Water Pump Control

As a user,

I want to control the water pump based on soil moisture levels or direct commands,

So that I can automate or manually manage the watering of my plants.

Acceptance Criteria:

1. **AC#1 – Automated Pump Control**
    - **Scenario 1:** Automatic Activation

- **Given** that the soil moisture levels are below a certain threshold,
- **When** the system detects low moisture,
- **Then** the ESP32 should activate the water pump automatically.
    - **Scenario 2:** No Activation
- **Given** that the soil moisture levels are above the threshold,
- **When** the system detects adequate moisture,
- **Then** the ESP32 should not activate the water pump.

2. **AC#2 – Manual Pump Control**
    - **Scenario 1:** Manual Override
- **Given** that I am on the dashboard,
- **When** I click the manual override button,
- **Then** I should be able to turn the water pump on or off directly from the server.
    - **Scenario 2:** Manual Control Failure
- **Given** that I am on the dashboard,
- **When** I click the manual override button and there is a connectivity issue,
- **Then** I should receive an error message indicating the failure.

User Story #5 – Plant Recognition

As a user,

I want to recognize plants by their name and get personalized watering information,

So that I can ensure each plant receives appropriate care.

Acceptance Criteria:

1. **AC#1 – Plant Recognition Input**
    - **Scenario 1:** Successful Recognition
- **Given** that I am on the plant recognition page,
- **When** I input a plant name,
- **Then** the system should retrieve and display personalized watering information from the database.
    - **Scenario 2:** Unrecognized Plant
- **Given** that I am on the plant recognition page,
- **When** I input a plant name that is not in the database,
- **Then** the system should display a message indicating that the plant is not recognized.

User Story #6 – Plant Profiles

As a user,

I want to create and store profiles for different plants,

So that I can monitor their status and manage their watering needs more effectively.

Acceptance Criteria:

1. **AC#1 – Create Plant Profile**
    - **Scenario 1:** Successful Profile Creation

- **Given** that I am on the plant profiles page,
- **When** I add a new plant profile,
- **Then** I should be able to enter details such as plant name, watering frequency, preferred sunlight, and soil type.

Field Name	Field Type	Validation/Requirements
Plant Name	Free text	Required
Watering Frequency	Dropdown	Required, options (Daily, Weekly, Bi-weekly, Monthly)
Preferred Sunlight	Dropdown	Required, options (Full Sun, Partial Sun, Shade)
Soil Type	Dropdown	Required, options (Loamy, Sandy, Clay, Silt)

    - **Scenario 2:** Missing Information

- **Given** that I am on the plant profiles page,
- **When** I add a new plant profile but omit the required information(see Scenario 1),
- **Then** I should receive an error message indicating the missing fields.

Field Name	Field Type	Validation/Requirements
Plant Name	Free text	Missing value
Watering Frequency	Dropdown	Missing value
Preferred Sunlight	Dropdown	Missing value
Soil Type	Dropdown	Missing value

    - **Scenario 3**: Plant recognized

- **Given** that I am creating a plant profile,

- **When** I press the "Fill fields automatically" button,
- **And** I have already used the plant recognition feature,
- **Then** I should be able to get the field information be filled automatically for the recognized plant.

2. **AC#2 – View Plant Profiles**
    - **Scenario 1:** Successful Profile View
- **Given** that I have created plant profiles,
- **When** I view the plant profiles page,
- **Then** I should see a list of all my plants with their profiles and current status based on sensor data.
    - **Scenario 2:** No Profiles Created
- **Given** that I am on the plant profiles page,
- **When** there are no profiles created,
- **Then** I should see a message indicating that no plant profiles are available.

User Story #7 – Scheduling Watering

As a user,
I want to set automated watering schedules,
So that my plants are watered at specific times without manual intervention.
Acceptance Criteria:

1. **AC#1 – Set Watering Schedule**
    - **Scenario 1:** Successful Schedule Set
- **Given** that I am on the scheduling page,
- **When** I set a new watering schedule,
- **Then** I should be able to specify the days and times for the watering to occur.
    - **Scenario 2:** Conflict in Schedule
- **Given** that I am on the scheduling page,
- **When** I set a new watering schedule that conflicts with an existing one,
- **Then** I should receive a message indicating the conflict and be prompted to resolve it.

1. **AC#2 – Automated Watering**
    - **Scenario 1:** Scheduled Watering
- **Given** that I have set a watering schedule,

- **When** the specified time is reached,
- **Then** the system should automatically activate the water pump according to the schedule.
    - **Scenario 2:** Schedule Not Executed
- **Given** that I have set a watering schedule,
- **When** the specified time is reached and the system fails to activate the water pump,
- **Then** I should receive an alert indicating the issue.

User Story #8 – Notifications and Alerts

As a user,

I want to receive notifications and alerts for critical conditions,

So that I can take timely actions to address any issues with my plants.

Acceptance Criteria:

1. **AC#1 – Soil Moisture Alerts**
    - **Scenario 1:** Low Moisture Alert
- **Given** that the soil moisture levels are critically low,
- **When** such a condition is detected,
- **Then** the system should send me a notification or alert.
    - **Scenario 2:** High Moisture Alert
- **Given** that the soil moisture levels are critically high,
- **When** such a condition is detected,
- **Then** the system should send me a notification or alert.

1. **AC#2 – Sensor Dysfunction Alerts**

- **Scenario 1:** Sensor Connectivity Issue
    - **Given** that there is a dysfunction or connectivity issue with the ESP32 device,
    - **When** such an issue is detected,
    - **Then** the system should send me an alert to inform me of the problem.
- **Scenario 2:** Sensor Data Issue
    - **Given** that the sensor data is not being transmitted correctly,
    - **When** such an issue is detected,
    - **Then** the system should send me an alert to inform me of the problem.

Interface description

1. Login Page

- **Role:** Allows users to authenticate into the application to access and manage plant information.
- **Elements:**
  - Username input field.
  - Password input field.
  - Login button.
  - Link to the registration page for new users.
  - Forgot password link.

2. Registration Page

- **Role:** Enables new users to create an account in the application.
- **Elements:**
  - Username input field.
  - Email address input field.
  - Password input field.
  - Confirm password input field.
  - Register button.
  - Link to the login page for existing users.

3. Home Page

- **Role:** Centralizes all plant profiles and other functionalities of the application.
- **Elements:**
  - Panels for viewing and managing each plant profiles.
  - Option to add new plant profiles.
  - Panel for setting automated watering schedules.
  - Notifications and alerts for critical conditions or sensor malfunctions.
  - Plant data statistics

4. Plant Recognition Page

- **Role:** Allows users to input a plant's name to receive personalized watering information from an integrated database.

- **Elements:**
  - Input field for entering the plant's name.
  - Search button to fetch plant information.
  - Search results and displayed information about the selected plant.

5. Plant Profile Management Page

- **Role:** Provides functionality for users to view, and manage profiles for different plants in their garden.
- **Elements:**
  - Plant profile with details such as name, watering frequency, sunlight preferences, and soil type.
  - Panel for monitoring soil moisture levels.
  - Panel for monitoring environmental conditions (temperature, humidity, light).
  - Manual control button for the water pump.
  - Edit and delete functionalities for existing plant profiles.

6.Notifications and Alerts Panel

- **Role:** Displays real-time notifications and alerts about critical conditions or system updates.
- **Elements:**
  - Options to configure notification preferences.
  - List of recent notifications.
  - Details of each notification with timestamp and type (e.g., sensor alert, watering schedule update).

Protocol, Application Data Structure and Message Flow

Protocol

The application uses HTTP as the standard protocol for communication between the frontend and backend components. The backend communicates with the ESP32 devices using WebSockets for real-time sensor data exchange.
Port Matrix:

- **Frontend to Backend:** HTTP (Port 80 for HTTPS)
- **Backend to ESP32 Devices:** WebSockets (Port 80 for WebSockets over WS/WSS)

Frontend and Backend Communication

- **HTTP(S) Requests**: The frontend, built using ReactJS, will use Axios to send HTTP(S) requests to the backend for data retrieval and updates. This ensures that data such as user authentication, plant profiles, sensor readings, and control commands are securely transmitted.
- **REST APIs**: The backend, developed with Spring Boot, exposes RESTful APIs that handle various CRUD operations. These APIs are secured using Spring Security to ensure that only authenticated and authorized users can access or modify data.

Real-time Updates

- **WebSockets**: For real-time communication, WebSockets are used alongside HTTP(S). This is particularly important for real-time sensor data monitoring and instant notifications. The backend maintains WebSocket connections to push updates to the frontend without the need for repeated HTTP requests.

Hardware Communication

- **ESP32 Integration**: The ESP32 microcontroller, equipped with sensors, sends data to the backend over HTTP(S). The Arduino IDE is used to program the ESP32 to make HTTP(S) requests to the backend, ensuring that sensor data such as soil moisture, temperature, and humidity are securely transmitted. Additionally, control commands (e.g., to activate the water pump) are sent from the backend to the ESP32 via HTTP(S).

Application Data Structure and Message Flow:

1. User Authentication

   Log in: User enters their email and password to log in.

   i. **Request**: `POST /login`

   ii. **Body**:

   ```
   { "username": "user12",
   "password": "password123" }
   ```

   iii**. Status Code:** 200 OK

2. Plant Profile Management

   Viewing plant profiles: User navigates to their plant profiles page.

   i. **Request**: `GET /plant_profiles`

ii. **Response**:


```
 {
    "profile_id": 1,
    "plant_id": 101,
    "plantName": "Tomato",
    "user_id": 1,
    "location": "Greenhouse",
    "last_watered": "2024-07-15T0700Z",
    "soil_moisture": 45.0,
    "light_level": 700.0,
    "temperature": 22.0,
    "humidity": 60.0
  },
  {
    "profile_id": 2,
    "plant_id": 102,
    "plantName": "Basil",
    "user_id": 1,
    "location": "Kitchen",
    "last_watered": "2024-07-16T0700Z",
    "soil_moisture": 50%,
    "light_level": 500.0,
    "temperature": 24.0,
    "humidity": 55.0
  }
```


iii. **Status Code:** 200 OK

Adding a new plant profile: User adds a new plant to their garden.

i. **Request**: `POST /plant_profiles/add`

ii. **Body**:

```
{ "plantName": "Tomato",
  "location": "Greenhouse",
 "wateringFrequency": "Daily",
 "preferredSunlight": "Full Sun",
 "soilType": "Loamy" }
```

iii. **Response**: Returns the newly created plant profile.


```
{
```

```
  "profile_id": 3,
  "plant_id": 103,
  "plantName": "Tomato",
  "user_id": 1,
  "location": "Greenhouse",
  "last_watered": null,
  "soil_moisture": null,
  "light_level": null,
  "temperature": null,
  "humidity": null
}
```

iv. **Status Code:** 201 Created

3. Soil Moisture and Environmental Monitoring

Retrieving sensor data: Backend requests the latest sensor data from an ESP32 device.

i. **Request**: `GET /plant_profiles/Tomato/sensors_data`

ii. **Response**: Returns current sensor data (soil moisture, temperature, humidity, light levels) from the ESP32.

```
{
"soil_moisture": 45.0,
"light_level": 700.0,
"temperature": 22.0,
"humidity": 60.0

}
```

ESP32 sending sensor data: ESP32 sends updated sensor data to the backend.

i. **WebSocket Message**: `ws://backend/sensor/update`

ii. **Message Body**:

```
{
"soilMoisture": 45,
"temperature": 22,
"humidity": 60,
"lightLevel": 700 }
```

4. Water Pump Control

Automated watering based on moisture level: ESP32 activates water pump based on soil moisture data.

i. **Request**: `POST/plant_profiles/Tomato/pump/control`

ii. **Body**:

```
{ "sensorId": "12345",
 "action": "activate" }
```

iii. **Response**: Confirms activation or deactivation of the water pump.

```
{
  "status": "success",
  "message": "Water pump activated"
}
```

Manual override: User manually activates or deactivates the water pump from the frontend.

i. **Request**: `POST/pump/manual`

ii. **Body**: `{ "action": "deactivate" }`

iii. **Response**: Confirms the manual action taken on the water pump.

```
{
  "status": "success",
  "message": "Water pump deactivated"
}
```

5. Plant Recognition and Personalized Watering Information

Plant recognition by name: User inputs the name of a plant to get personalized watering information.

i. **Request**: `POST /plants/recognize`

ii. **Body**: `{ "plantName": "Tomato" }`

iii. **Response**: Returns personalized watering instructions and care tips for the plant from the database.

```
{
  "plant_id": 101,
  "wateringFrequency": "Daily",
  "preferredSunlight": "Full Sun",
  "soilType": "Loamy",
```

```
  "careTips": "Water daily in the morning, ensure full sunlight exposure."
}
```

6. Scheduling Functionalities

   Setting up a watering schedule: User sets a specific time for automated watering.

   i. **Request**: `POST/plant_profiles/Tomato/schedule/set`

   ii. **Body**: `{ "plantId": "12345", "next_watering_date": "2024-07-16T0700Z", "status":"Pending..."}`

   iii. **Response**: Confirms the scheduling setup for the plant.

```
{
  "status": "success",
  "message": "Watering schedule set for plant ID 12345"
}
```

7. Notifications and Alerts

   Receiving alerts: Backend sends alerts to the user for critical conditions.

   i. **WebSocket Message**: `ws://frontend/alerts`

   ii. **Message Body**:

```
{ "alertType": "LowSoilMoisture",
"message": "Soil moisture is below the threshold for Tomato plant." }
```

Scalability and Security Considerations

Machine Specifications

CPU: 2-4 processors

RAM: 8-16 GB RAM

Storage: 100GB SSD

Network: High-speed internet connection, 500 Mbps

Maximum Load

Requests Per Second (RPS): Between a minimum of 100 and a maximum of 300

- This range ensures that the server can handle frequent requests from multiple users, including sensor updates and user interactions.

Simultaneous Users: 2000 simultaneous users

Number of Simultaneous Sensor Data Updates: Minimum 500, maximum 1500

Number of Active WebSocket Connections: 1000

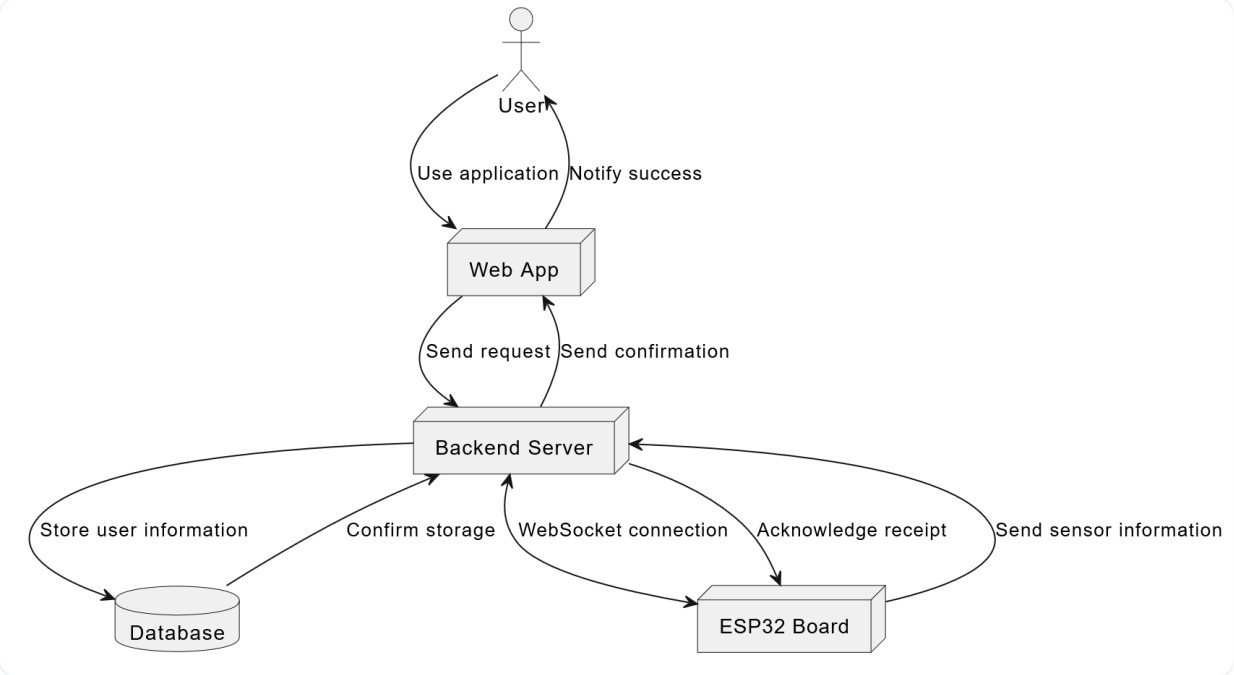
- WebSockets are used for real-time communication between the ESP32 devices and the backend server, allowing for instant updates and commands. The system can support up to 1000 active connections at any given time.

Security Measures

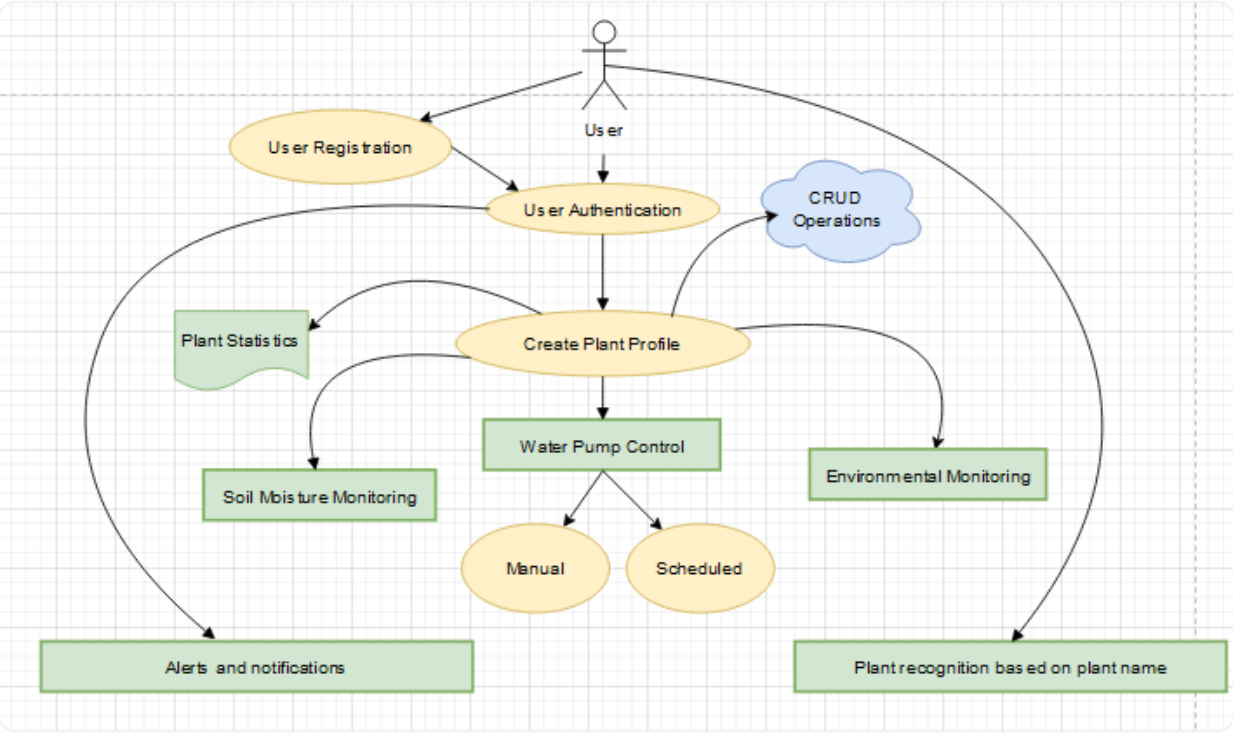
1. **Authentication and Authorization:**
    - **JWT (JSON Web Tokens):**
        - **Usage:** JWT is used for secure user authentication and authorization.
        - **Implementation:** On successful login, the server issues a JWT token that the client must include in the headers of subsequent requests to access protected resources.
        - **Advantages:** JWT tokens are stateless, reducing server load and making it easier to scale the application.
    - **OAuth2:**
        - **Usage:** For third-party authentication and secure API access.
2. **Data Encryption:**
    - **HTTPS (SSL/TLS):**
        - **Usage:** All data transmitted between the client and server is encrypted using HTTPS to protect against eavesdropping and man-in-the-middle attacks.
    - **Encryption of Sensitive Data:**
        - **Usage:** Sensitive data such as passwords and personal user information are encrypted before being stored in the database.

Diagrams

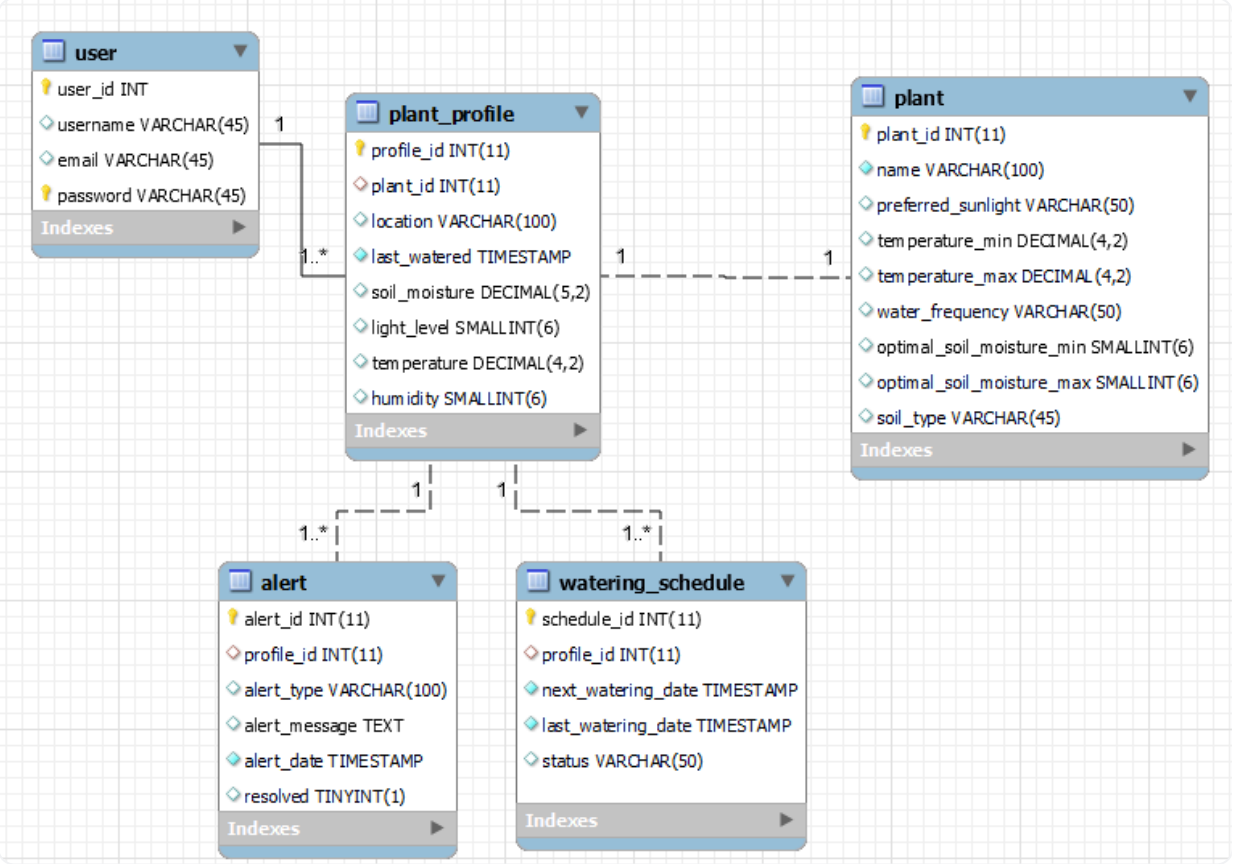
System Architecture Diagram



Use Case Diagram

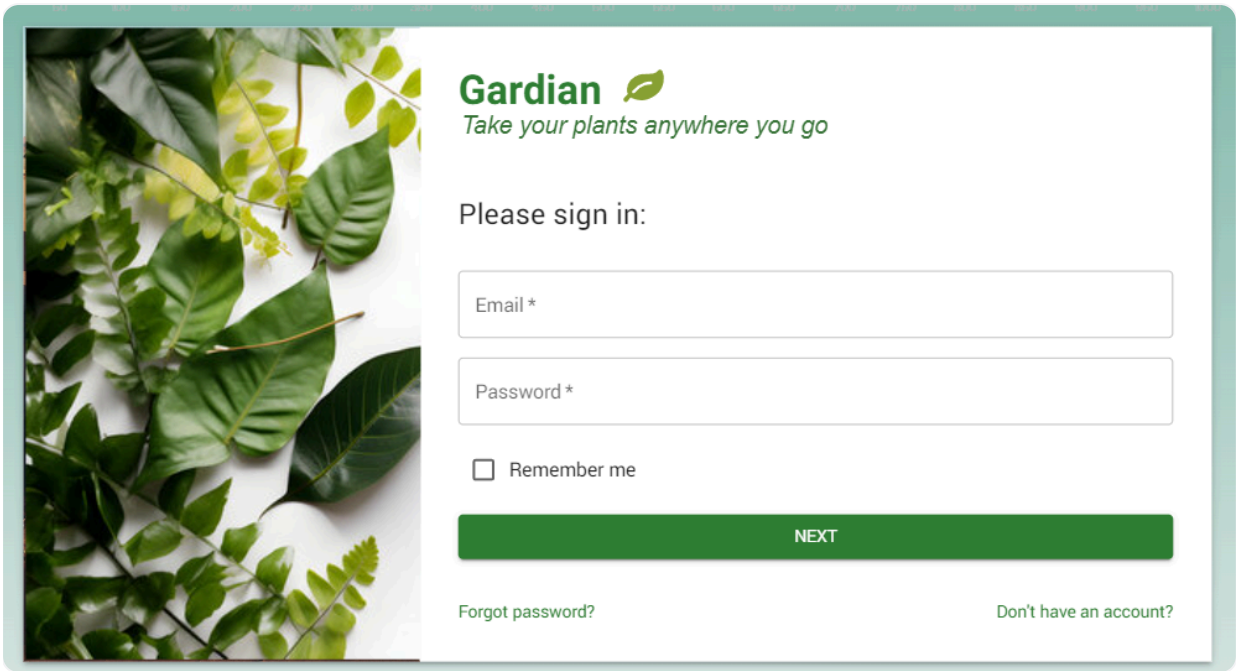


Entity-Relationship Diagram

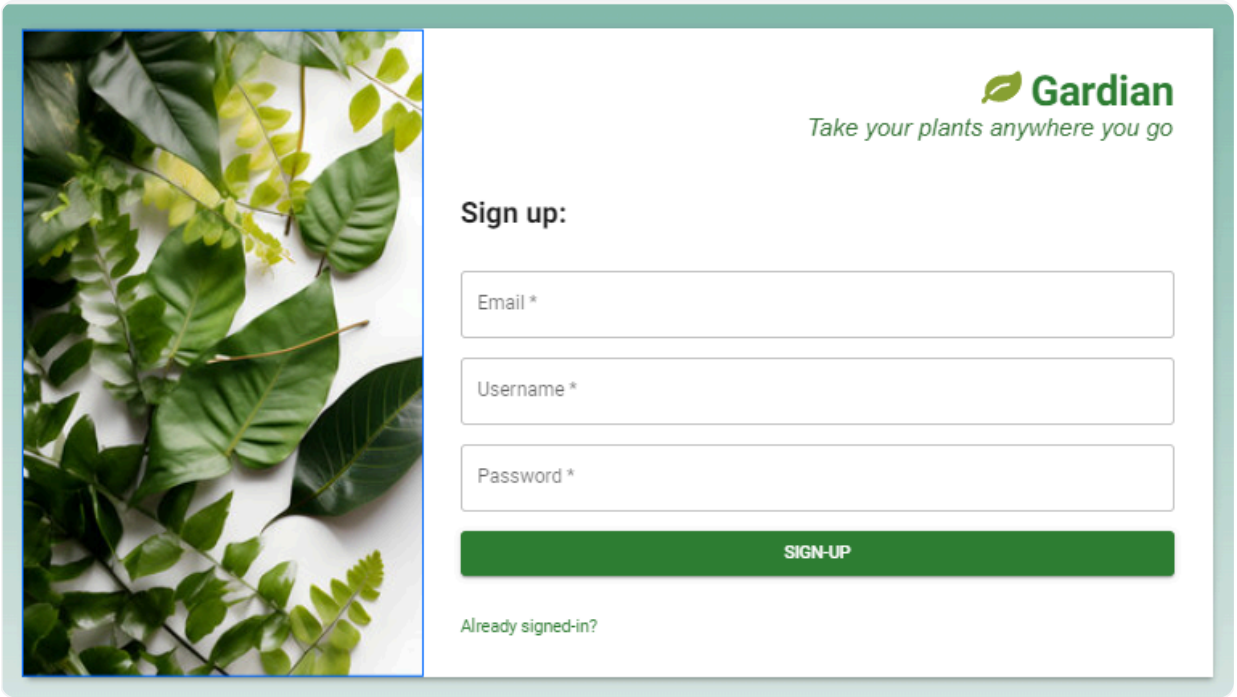


Prototype

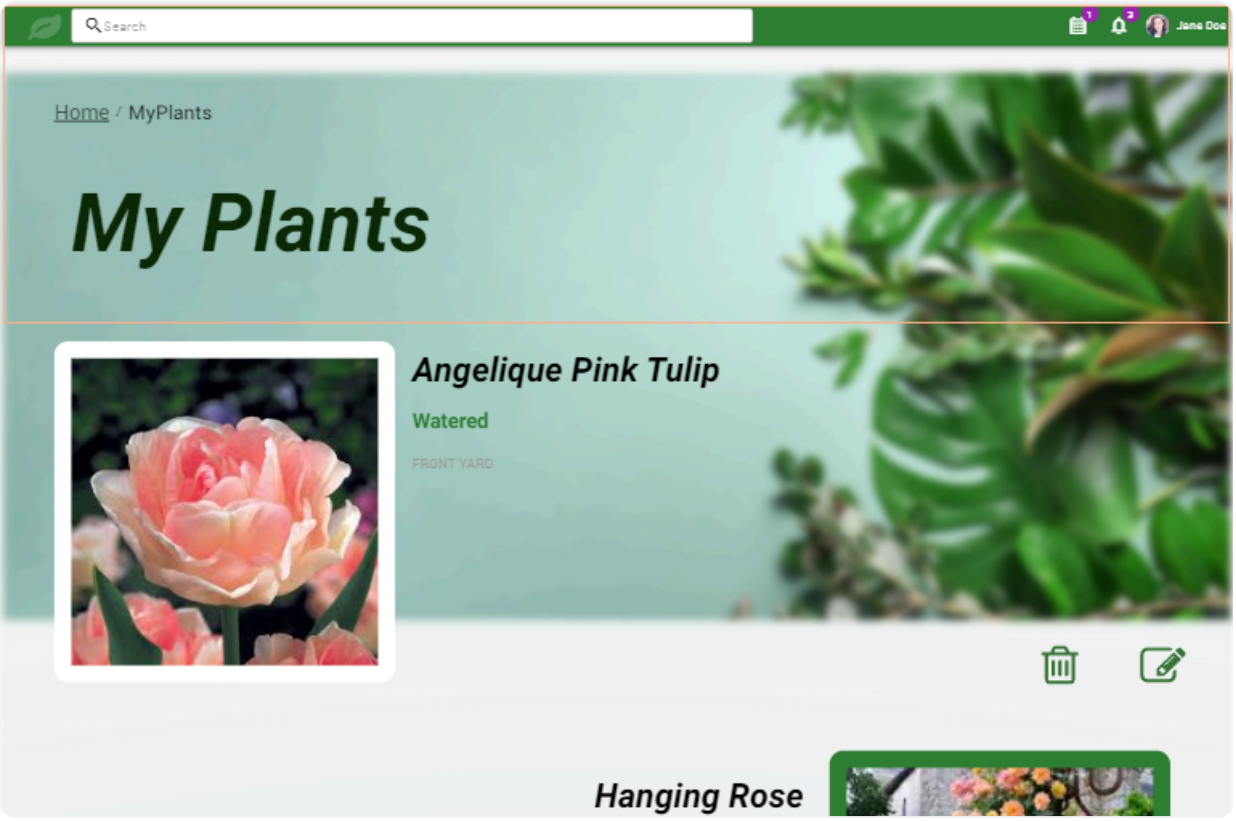
Sign in form:



Sign up form:



My Plants Profiles page:



Watering In Progress...

FRONT YARD

Cactus

Too much water!

BALCONY