TECHNICAL_DOCUMENTATION

" AgriSense: Leveraging IoT for Precision Farming"

Developing an IoT-based system for precision farming, integrating soil moisture sensors, weather data, automated irrigation, and crop health analytics.

Xu Kaiyang

Fort Hays State University

August 22, 2023

TECHNICAL_DOCUMENTATION

1. Introduction:

AgriSense is an advanced agricultural system that leverages the power of IoT to facilitate precision farming. This documentation provides a technical overview of the system, its architecture, codebase, dependencies, and other crucial elements necessary for developers and system administrators to understand, modify, and maintain the system.

2. System Architecture:

AgriSense is built on a client-server model. The front-end is developed using [****] while the back-end is powered by [****]. Data is stored in a [****] database.

- Front-end: Handles user interactions, displays real-time data, and offers intuitive controls for users.

- Back-end: Processes data, interacts with the database, and sends information to the front-end.

- Database: Stores user data, soil moisture readings, weather patterns, and other relevant information.

3. Code Structure:

The codebase is organized into the following directories:

- /app: Main application folder containing all essential files.

- /app/models: Contains data models and database interactions.

- /app/routes: Contains all routes and controllers.

- /config: Configuration files for the application.

- /tests: Unit and integration tests.

4. Dependencies:

AgriSense relies on several third-party libraries and frameworks. Some of the primary dependencies include:

- [****]

- [****]

- [****]

5. Installation and Configuration:

1. Clone the repository:

    git clone [https://github.com/XuKaiyang1/2023U_CSCI441_VB_Software-Engineering_Xu-Kaiyang]

2. Navigate to the project directory:

    cd path_to_project_directory

3. Install dependencies:

    pip install -r requirements.txt

4. Set up the database:

    python manage.py migrate

5. Run the server:

    python manage.py runserver

6. Database Schema:

The main database tables and their relationships are as follows:

- User: Stores user information.

    - Fields: UserID, Username, Password, Email, DateJoined

    - Relations: One-to-many with SoilMoistureReadings

- SoilMoistureReadings: Stores soil moisture data.

    - Fields: ReadingID, UserID, Timestamp, MoistureLevel

    - Relations: Many-to-one with User

- WeatherPatterns: Stores weather data.

    - Fields: WeatherID, Timestamp, Temperature, Humidity, Rainfall

    - Relations: None

7. Troubleshooting and FAQs:

- Q: I'm getting a database connection error. What should I do?

  A: Ensure that your database server is running and that you have correctly set the database connection parameters in the config file.

- Q: The front-end isn't displaying the latest data. Why?

  A: This could be due to caching. Try clearing your browser cache or ensure that the back-end is sending the updated data.

- Q: How do I add a new dependency to the project?

  A: Use pip to install the dependency and then update the `requirements.txt` file.

(Note: Add any other frequently asked questions or common issues that developers might face.)