UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Software Engineering for IoT

Project Documentation: Crop Harvest Optimization System

Submitted to:
Professor Davide Di Ruscio

Submitted by:
Minase Mengistu (297864)

January , 2025

Contents

1 Introduction

In modern agriculture, maximizing crop yield while minimizing resource usage and environmental impact is a pressing challenge. Farmers require accurate and timely data to make informed decisions about planting, watering, fertilizing, and harvesting their crops. This is especially true in the face of increasing climate variability, which makes traditional farming methods less reliable. The Crop Harvest Optimization System is designed to address these challenges by leveraging advanced technology. The system integrates sensor data, weather forecasts, and GDD calculations into a unified platform that empowers farmers with actionable insights. By tracking GDD, a measure of heat accumulation crucial for crop development, the system predicts the optimal harvest time for various crops, ensuring maximum yield and quality. Key features of the system include:

- Actual Temperature Monitoring: IoT sensors provide continuous data on temperature data.
- Weather Forecast Integration: External APIs deliver 10-day weather forecasts, enabling predictive decision-making.
- GDD Calculation: The system automates the calculation of cumulative GDD values, tailored to each crop's unique growth requirements.
- User-Friendly Dashboard: An intuitive interface displays farm, field, and sensor data, offering clear insights at a glance.

This document details the design, implementation, and functionality of the system. It also highlights the technologies used, such as FastAPI, Angular, and Azure Table Storage, chosen for their scalability, performance, and reliability. By offering a modular and extensible architecture, the system provides a robust foundation for modern farm management.

2 Functional Requirements

- Farm and Field Management: Users can create and manage farms and their corresponding fields.
- Sensor Integration: The system supports the addition of sensors for monitoring temperature, humidity, and other environmental parameters.
- GDD Calculation: Automatically calculates cumulative GDD values based on sensor data and external weather API.
- Weather Forecast Integration: Retrieves and displays 7-day weather forecasts, including temperature and humidity data.
- Dashboard: Provides a user-friendly interface for viewing farm and field data, sensor readings, and GDD trends.

- User Authentication: Auth0 integration ensures secure user authentication and role-based access control.
- Data Storage: Persist farm, field, sensor, and GDD data in Azure Table Storage and MySQL.

3 Non-Functional Requirements

- Reliability
  - Azure Storage ensures reliability through data partitioning and replication across multiple geographic locations, enabling fault tolerance, scalability, and continuous availability even during unexpected failures.
  - The MQTT broker ensures real-time messaging between sensors and the data processor.
- Security
  - Secure all API endpoints and the dashboard with Auth0.
- Maintainability
  - Modular code structure using FastAPI, Angular, and Docker ensures ease of updates and debugging.
  - Comprehensive logging and monitoring are implemented.

4 Technologies Used

The Crop Harvest Optimization System is built using a modern and scalable tech stack, ensuring high performance, reliability, and ease of maintenance:

- Messaging: Utilizes MQTT with Eclipse Mosquitto for real-time communication between system components.
- Backend API: Developed with FastAPI and Python, leveraging SQLAlchemy for efficient database interactions.
- Frontend Dashboard: Built using Angular, providing a responsive and intuitive user interface for managing farms, fields, and sensors.
- Database: Combines MySQL for relational data storage and Azure Table Storage for scalable handling of large, time-series datasets from sensors and forecasts.
- Authentication: Secured with Auth0, offering robust authentication and authorization mechanisms.
- Containerization: Employs Docker and Docker Compose for easy deployment, scalability, and consistent development environments.
- Weather API Integration: Integrates the Met.no LocationForecast API for reliable and accurate weather data to enhance GDD predictions.

This technology stack ensures the system is modular, extensible, and capable of handling complex agricultural optimization tasks efficiently.

5 System Architecture

Figure 1 Shows the architecture of the system developed. As depicted in the figure, the system is composed of seven interconnected components each with their own specific functionality. Each component's functionality is detailed as follows:

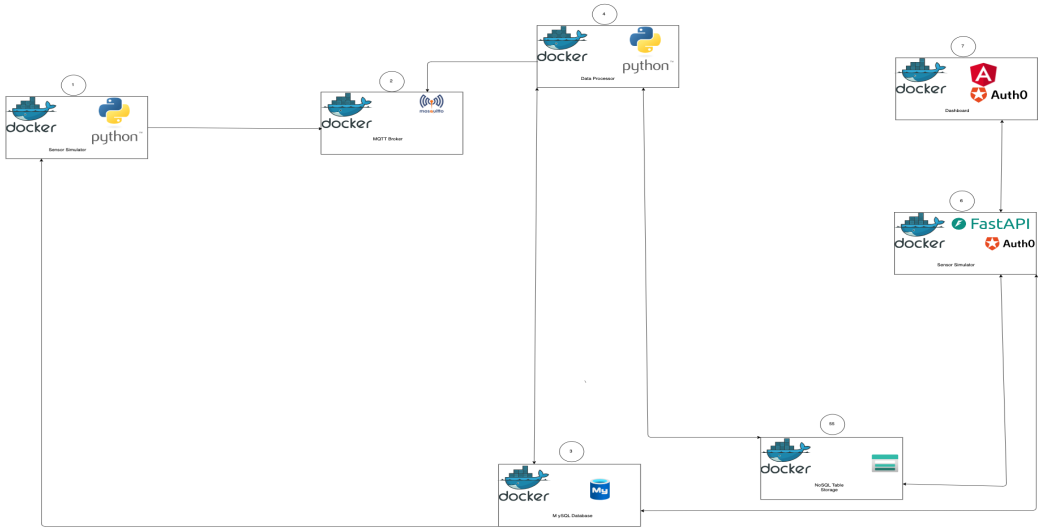

Figure 1: System Architecture Diagram

5.1 Sensor Simulator

The sensor simulator is a vital component of the Crop Harvest Optimization System, generating realistic temperature data (-10°C to 35°C) from virtual IoT sensors and publishing it to an MQTT broker. This allows for testing and functionality validation without physical sensors. It also simulates historical data for the past 20 days, integrates with a MySQL database for sensor configurations, and supports real-time publishing using the MQTT publish/subscribe model.

MQTT topics are structured hierarchically as farms/FarmId/fields/FieldId, enabling targeted subscriptions for specific farms or fields. For example, data from farm ID 1 and field ID 2 is published to farms/1/fields/2. This format ensures scalability and easy system extension.

The workflow involves connecting to the MQTT broker, fetching sensor data from the database, generating readings (timestamp, temperature, sensor ID), and publishing the data to structured MQTT topics for seamless integration with the system.

5.2 MQTT broker

The MQTT broker facilitates real-time communication between the sensor simulator and the data processor components. It uses Eclipse Mosquitto for lightweight messaging and is deployed in Docker, configured with ports 1883 for MQTT communication and 9001 for WebSocket communication.

Its primary role is to enable reliable data transfer across components while ensuring data integrity through the Quality of Service (QoS=1) protocol. This setup ensures seamless and dependable message delivery within the system.

5.3 MySQL Database

As shown in Figure 2, the database design for the Crop Harvest Optimization System is organized hierarchically to reflect real-world farm operations and ensure efficient data management. Users are linked to farms they manage or monitor, and each farm is further divided into fields. Fields host sensors that monitor environmental data and crop growth metrics, with all entities connected through clearly defined relationships.

The schema supports modularity and scalability, enabling the system to handle a growing number of users, farms, fields, and sensors. This relational structure ensures robust and efficient data queries, facilitating smooth system operations and future extensibility.

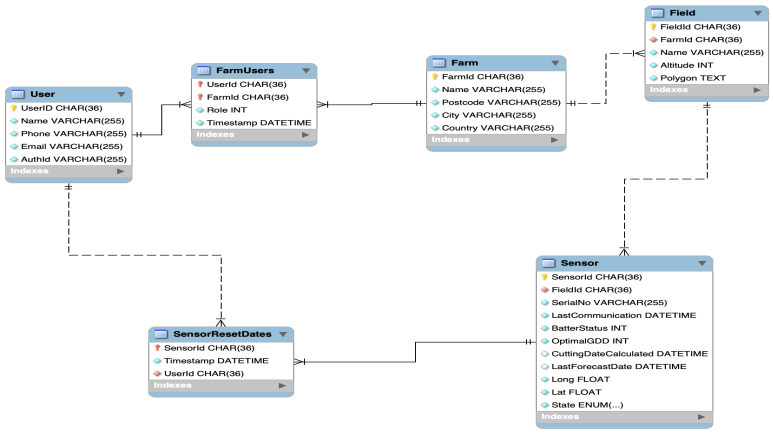

Figure 2: Database Model Diagram

5.4 Data Processor

The data processor is a crucial component of the Crop Harvest Optimization System, responsible for ingesting real-time sensor data, integrating weather forecasts, and performing key computations such as Growing Degree Days (GDD) and optimal crop cutting dates. It collects raw sensor data from an MQTT broker, decodes the messages, and stores them in Azure Table Storage for processing and analysis. The processor also

integrates weather data from external APIs, using the geographic coordinates of sensors to fetch location-specific forecasts, which are stored for enhancing GDD predictions.

The processor calculates cumulative GDD values by combining real-time sensor data with forecasted weather conditions. Based on these calculations and the optimal GDD thresholds set for crops, it determines the optimal cutting dates for each sensor. These dates are then updated in the MySQL database for further system use.

Designed for scalability and reliability, the processor operates on a modular workflow, integrating MQTT for data ingestion, Azure Table Storage for efficient storage, and MySQL for maintaining metadata and calculated results. Its robust error-handling mechanisms ensure uninterrupted operations and accurate computations, making it a vital part of the system's decision-making process.

5.5 NoSQL Table Storage

The Weather Data Storage Component is designed to manage large volumes of time-series data from IoT sensors and weather forecasts, leveraging NoSQL (Azure Table Storage) for scalability and efficiency. This approach enables fast read and write operations, making it ideal for handling continuous data streams from sensors and external APIs.

The component integrates actual data from sensors, capturing real-time environmental conditions, and forecast data from APIs, including predicted parameters like temperature and humidity. These are stored in the weatherdata table, using PartitionKey for sensor identification and RowKey for timestamps. GDD values, essential for crop growth tracking, are calculated and stored in the gdddata table.

By using NoSQL, the system efficiently manages large-scale, time-sensitive data while ensuring scalability to handle growing sensor networks and forecast demands. This component seamlessly integrates with the Data Processor and REST API, providing actionable insights and maintaining high performance even under heavy data loads

5.6 REST API

The REST API Component, built with FastAPI, serves as the primary communication interface between the backend and the dashboard, enabling secure and efficient management of farms, fields, sensors, and related data. Its design ensures high performance, scalability, and seamless integration with MySQL, Azure Table Storage, and the dashboard.

Key features include robust security through Auth0 for authentication and authorization, ensuring secure access to resources. RESTful endpoints provide functionalities such as fetching farm data, creating new farms and fields, managing sensors, and retrieving detailed GDD and weather forecasts. Notable endpoints include:

- `GET: /farms/farmdashboard` for retrieving user-specific farm data and

- `GET: /fields/fielddashboard{field_id}` for accessing field-specific details like GDD and forecast data.

The API's swagger documentation page, as depicted in Figure 3, can be accessed at `http://localhost:8000/docs/`.

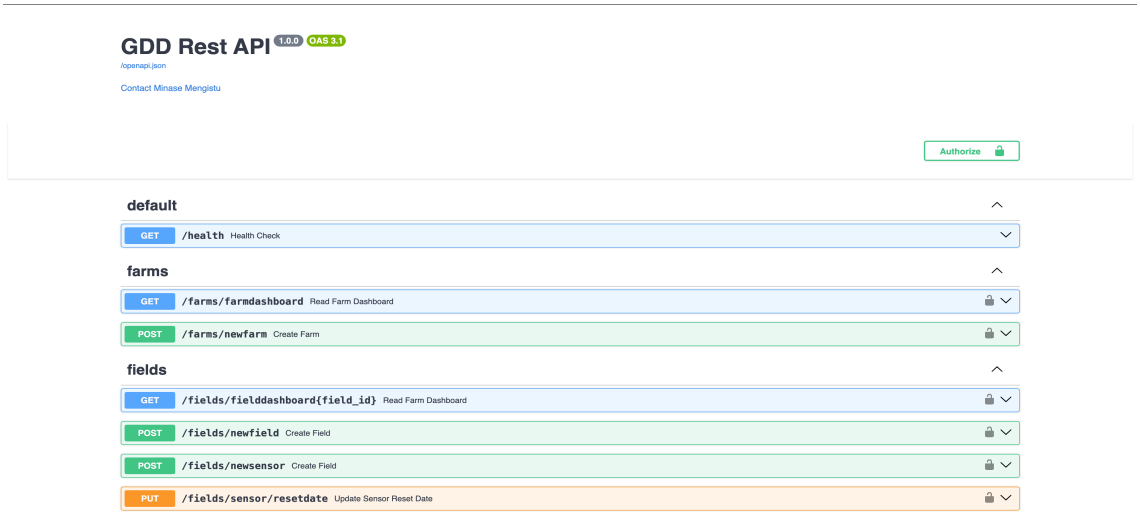

Figure 3: The API's Swagger documentation page

All API endpoints are protected by role-based access control, allowing users to access only data relevant to their farms and fields. This ensures secure and efficient operations while maintaining data integrity across the system. The REST API is integral to facilitating communication and delivering actionable insights to the dashboard.

5.7 Dashboard

The Dashboard Component provides an intuitive and user-friendly interface for monitoring and managing farms, fields, and sensors. Below are screenshots illustrating key features and functionality.

The welcome page greets users and prompts them to create their first farm if none exist, As shown in figure 3.

Users can create a new farm as shown in the figure 4.

Once inside a farm users can create a field in a farm using the create new field form as shown in figure 5.

A field can consist of multiple sensors, in the current implementation users can add only one sensor in a field, where users can add a sensor in a field using the form shown in figure 6.

The farm dashboard shows some stats of the fields in a selected farm. These stats consist of the current GDD of the fields and the predicted cutting dates of each Field as shown in figure 7.

The fields dashboard shows a detailed view of a given field such as the current and Optimal GDD of the Field, The sensor of the field. A seven-day forecast of temperature, GDD and Humidity is also displayed. The predicted optimal cutting date is also shown as shown in figure 8.

Users can also change the sensor reset date. The sensor reset date is the starting date where the GDD and Cutting date are calculated. This date basically represents the date where the crop is planted in the field. Figure 9 shows the form to change the sensor reset date.

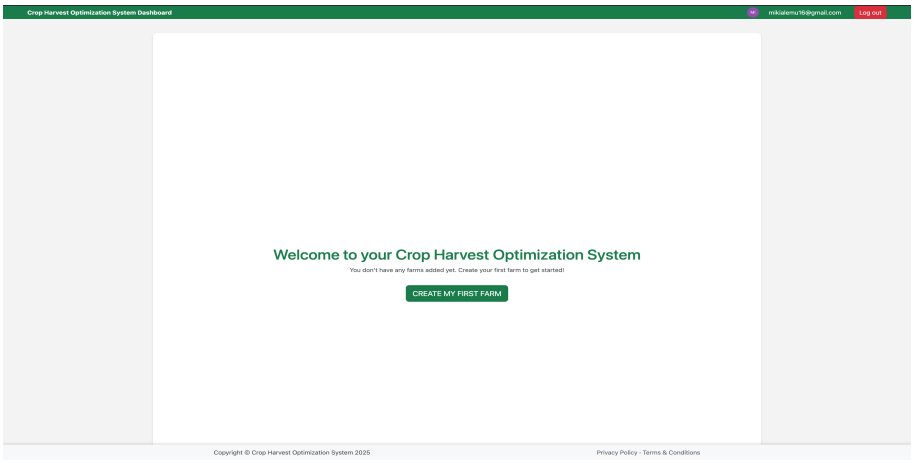

Figure 4: Dashboard: Welcome page

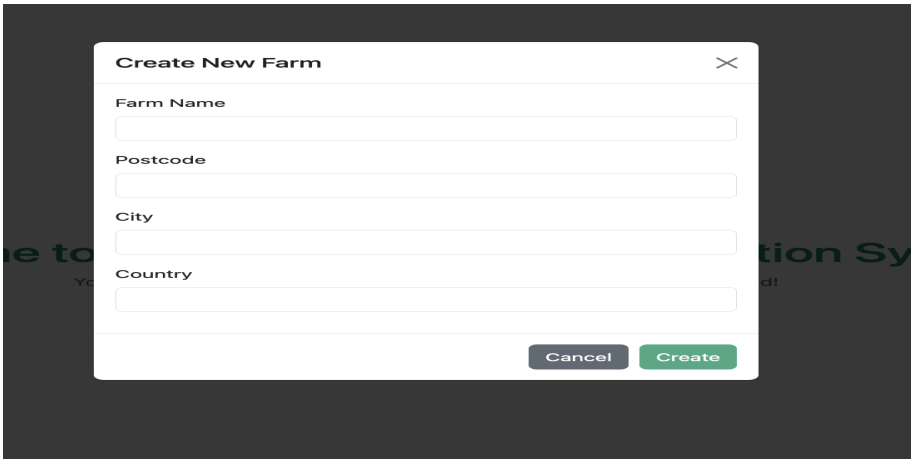

Figure 5: Dashboard: Create New Farm

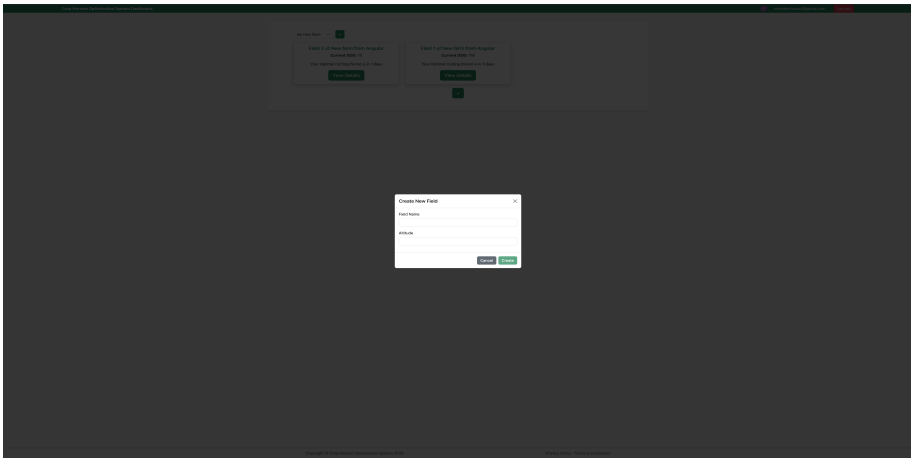

Figure 6: Dashboard: Create New Field

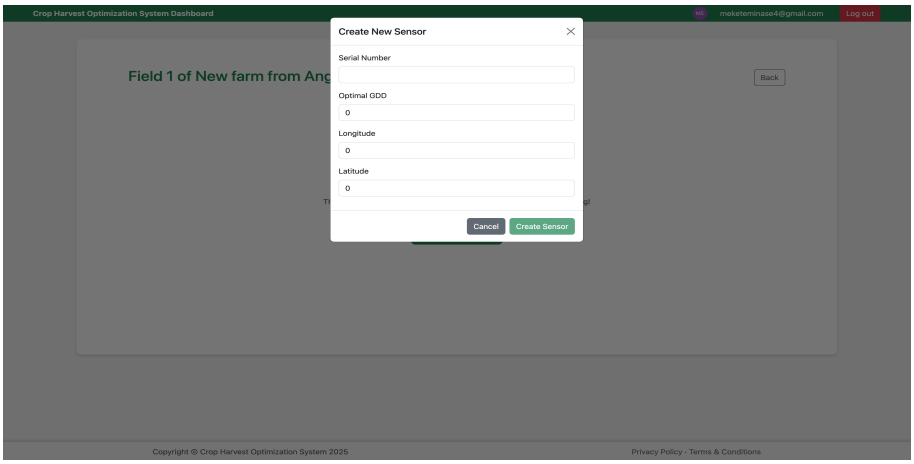

Figure 7: Dashboard: Create New Field

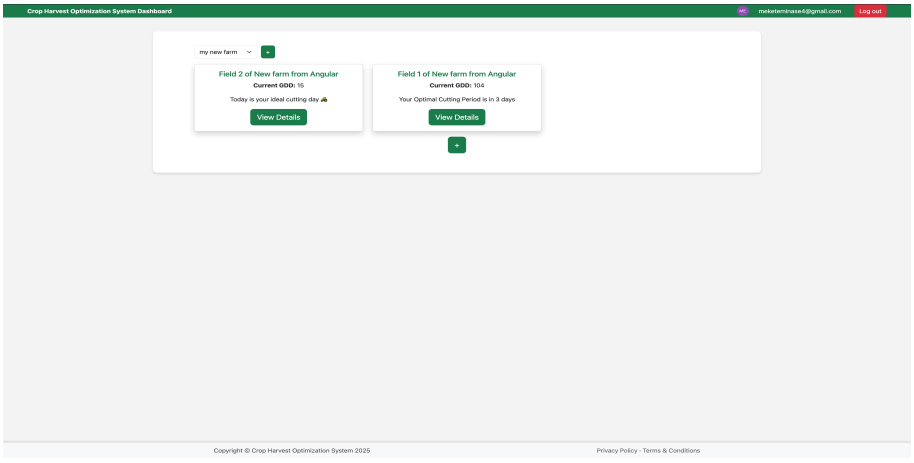

Figure 8: Farm Dashboard

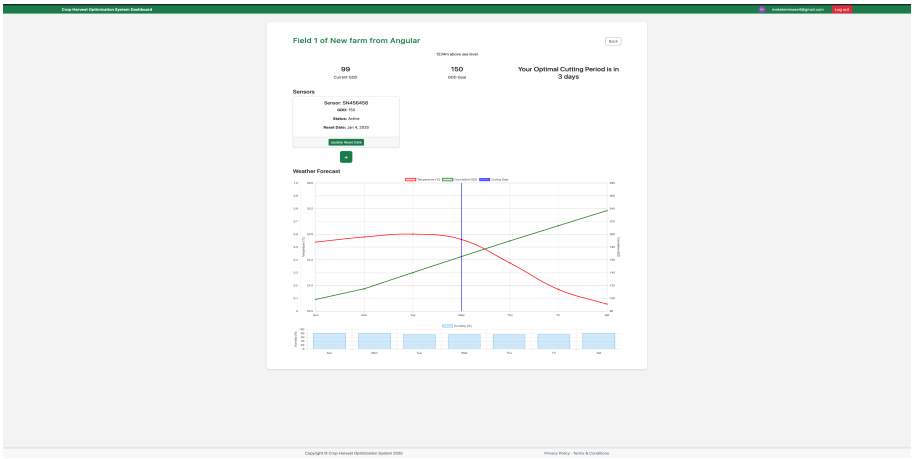

Figure 9: Field Dashboard

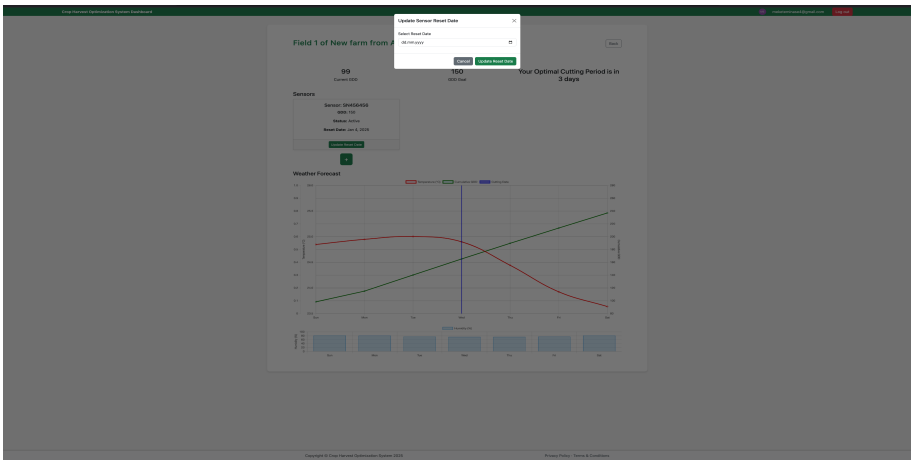

Figure 10: Dashboard: Sensor Reset date

6 Conclusion

The Farm Advisor System is a comprehensive solution for modern farm management. By integrating sensor data, weather forecasts, and GDD calculations, the system enables informed decision-making for farmers. The modular design ensures ease of maintenance, and the use of cloud technologies ensures scalability and reliability. Future improvements may include adding support for more sensor types and advanced analytics.