

CMPE2002 Assignment 1



SEPTEMBER 26

Smart Farming and Agriculture Management App
Authored by: Olimar Ramilo

Table of Contents

Project Overview	3
Work Breakdown Structure.....	5
Functional and Non-Functional Requirements.....	6
UML Use Case Diagram	7
UML Sequence Diagram	7
UML Activity Diagram	8
UML State Diagram	9
Agile/Scrum Planning.....	11

Project Overview

The Smart Farming and Agriculture Management app is a comprehensive solution aimed at revolutionizing traditional farming practices by leveraging technology to enhance crop management, improve soil health, and provide farmers with real-time data and actionable insights. This project report outlines the process of developing the app, starting from requirements elicitation to defining functional and non-functional requirements.

Elicitation Techniques

Background Reading

In gathering requirements for the Smart Farming and Agriculture Management App, we used the background reading technique to learn from existing research and industry insights. We read academic papers, industry reports, and online materials about modern farming and technology. This helped us understand the challenges and solutions in farming. We then used this knowledge to create the app's requirements, especially for monitoring crops and soil, and AI recommendations. This process ensured our app followed the best practices and latest trends in agriculture.

Interviews

We conducted interviews with farmers, agricultural experts, and potential app users. We asked about their farming practices, challenges, and tech expectations. These interviews helped us identify key app requirements like real-time crop monitoring and user-friendly design, aligning the app with user needs.

Knowledge Elicitation – Protocol Analysis

In our project, we used a method where people talk about what they're doing and thinking as they work on tasks. We recorded these conversations, called 'think-aloud protocols,' and studied them to understand how they think while doing their tasks. We didn't watch farmers and experts directly, but we had conversations with them where they talked about their farming practices and what they need from technology. This helped us understand how they make decisions and guided the requirements for our app.

Rationale

In the context of this project, we employed background reading, interviews, and protocol analysis as our key requirement elicitation techniques. These techniques collectively formed the foundation for understanding the agricultural landscape, gathering insights from stakeholders, and delving into cognitive processes to shape the Smart Farming and Agriculture Management App effectively. Interviews were found to be the most effective elicitation technique as they provided valuable insights into our potential app users and how they

would integrate modern technology into their daily workflow. These direct conversations with farmers, agricultural experts, and potential users enabled us to gain a deep understanding of their challenges, decision-making processes, and specific needs within the agriculture sector. Through interviews, trust was established, and candid feedback was obtained, which played a critical role in shaping the app to meet the real-world requirements and expectations of our target audience. To model our project requirements effectively, our primary actors include farmers (as users), the weather system, the emergency system, and the AI system. We employed various visualization techniques such as swim lane diagrams, use case diagrams, and state diagrams to depict how these actors interact within the system.

Work Breakdown Structure

The WBS table below identifies the tasks and sub-tasks of the Smart App Farming Project:

Method/Model	Story Points	Sprint/Iteration
1. Requirement Analysis and Project Scope		
1.1 Define Project Scope	1	1
1.2 Identify Stake holders	2	1
1.3 Project planning	2	2
2. Requirements Elicitation		
2.1 Background Reading	3	2
2.2 Interviews	3	2
2.3 Knowledge Elicitation Protocol Analysis	3	2
3. System Design		
3.1 Architectural Database Design	4	3
3.2 Database Design	3	3
3.3 User Interface Design	3	4
4. Development		
4.1 User Registration and Login	5	4
4.2 User Profile Management	4	5
4.3 Farm Dashboard	6	5
4.3.1 Crop Health Monitoring	6	6
4.3.1 Soil Quality Analysis	5	6
4.3.1 AI Recommendation System	7	7
4.3.1 Progress Reporting	6	8
4.3.1 Crop Trend Analysis	5	9
4.3 Weather Alerts		10
4.4 Emergency Alerts	3	10
4.5 2FA Security Implementation	4	11
4.6 Encryption Implementation	4	11
4.7 Pest Control	6	12
4.8 Irrigation Management	5	13
5. Testing		
5.1 Unit Testing	3	14
5.2 Integration Testing	4	14
5.3 UI Testing	3	15
5.4 Bug fixing	4	15
6. Deployment		
6.1 Server setup	3	16
6.2 App deployment	4	16
6.3 Documentation	2	16

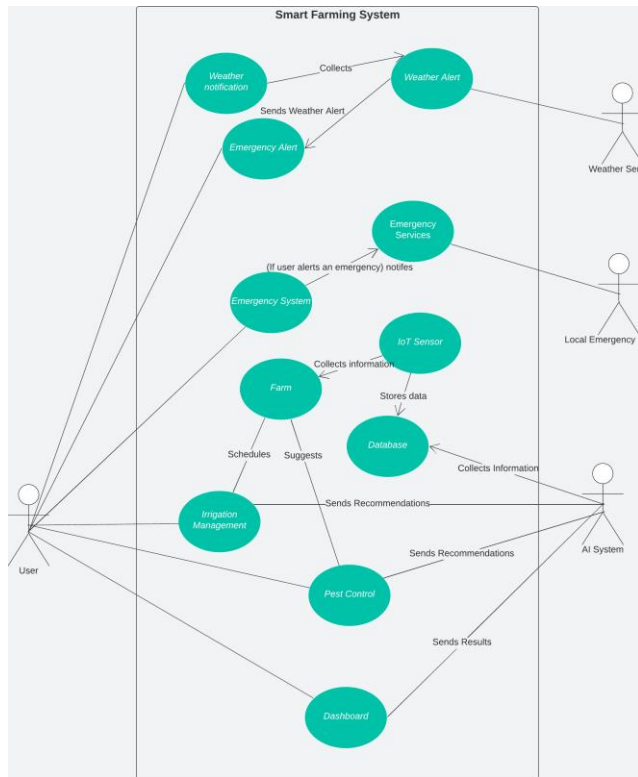
Functional and Non-Functional Requirements

Below are tables containing the functional and non-functional requirements of the features and necessities required to build this application.

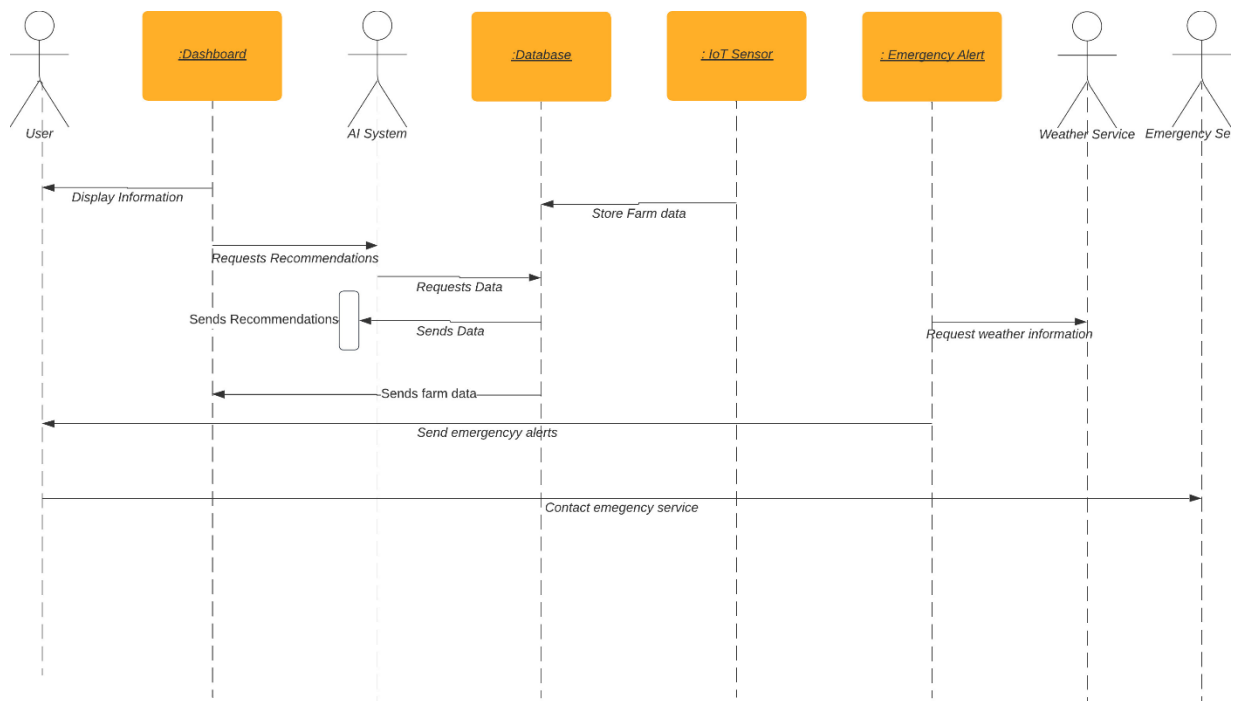
Functional Requirements
1. Users must be able to register and log into their accounts
2. Users must be able to monitor crop health in real time
3. The app should provide AI-based recommendations for their farm
4. Users need access to progress reports for their farm
5. Crop trend data should be available for decision-making
6. Users should be able to analyse the soil of their crops
7. Users should receive weather alerts and forecasts relevant to their farm location
8. Users must receive emergency alerts related to their farms
9. The app should provide recommendations for pest control and fertilization
10. Users should be able to schedule irrigation based on current and historical data
11. The app should be accessible on both mobile and desktop devices
12. The app should have a friendly user-interface that requires minimal interactions

Non-Functional Requirements
1. The app must perform efficiently and scale with large amount of data
2. Response times for critical features should be fast
3. The app should accommodate for increasing number of users and farms
4. The app should accommodate for increasing amount of user traffic into the server during peak hours
5. Data privacy and standards should be followed and be compliant
6. User data must be securely stored and protected from unauthorized access
7. The app must have an user-friendly design
8. The app should recover gracefully after failures
9. The must should be available to users 24/7 with minimal downtime
10. Regular data backups should be performed to ensure data integrity and facilitate recovery in case of data loss
11. Provide a comprehensive user and developer documentation
12. The app should be easy to integrate with external data sources and services such as weather forecasts

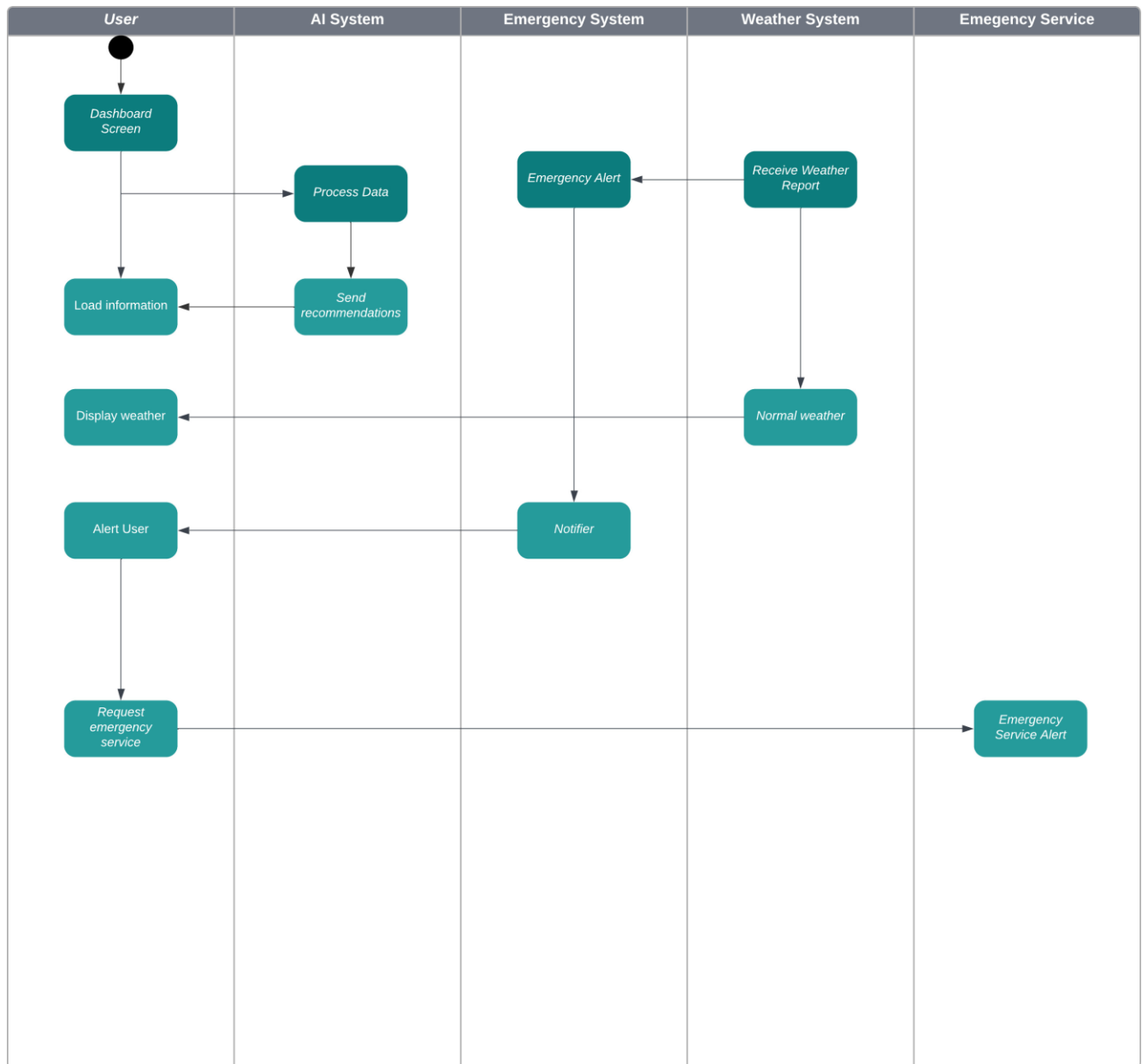
UML Use Case Diagram



UML Sequence Diagram

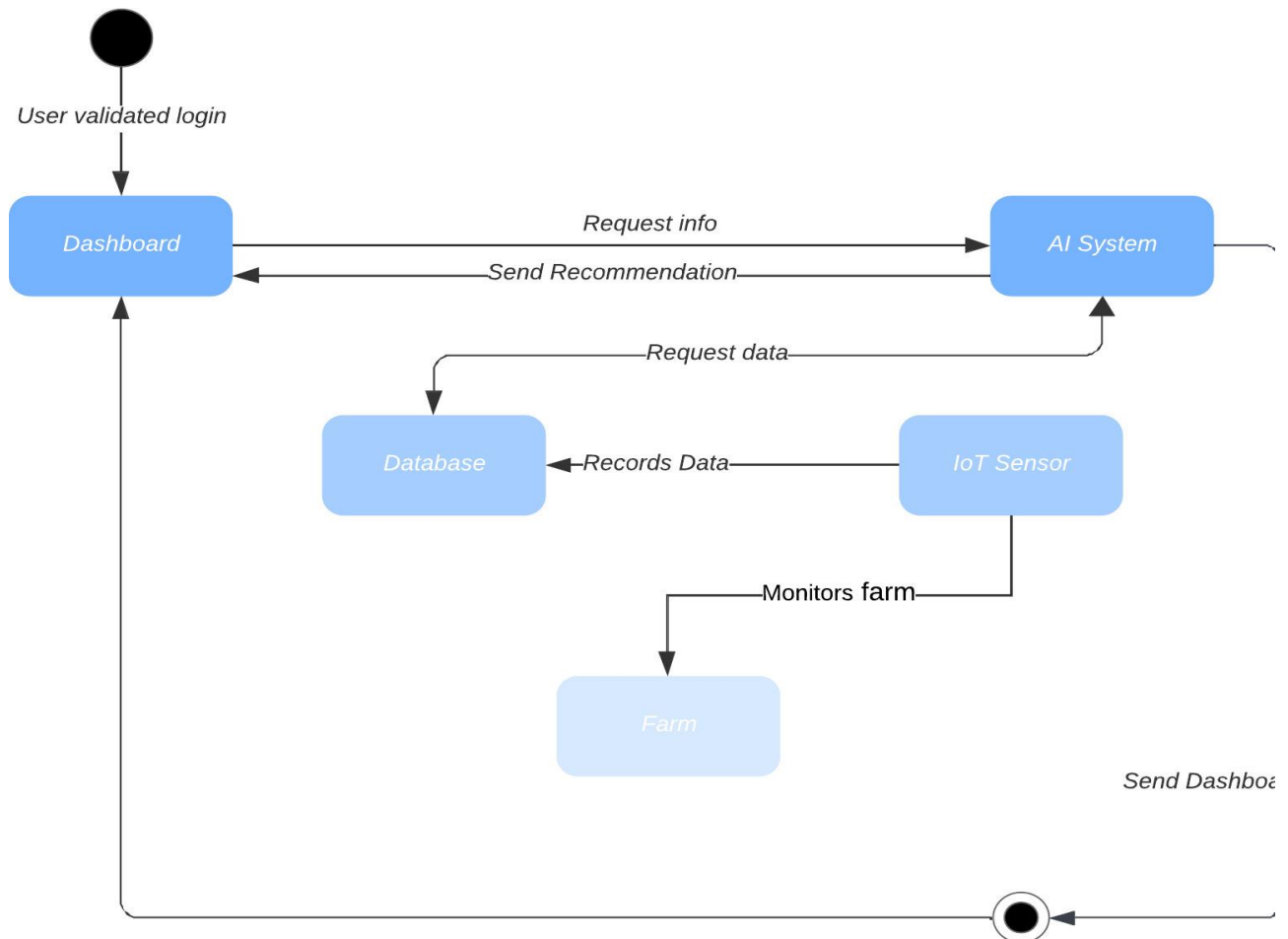


UML Activity Diagram

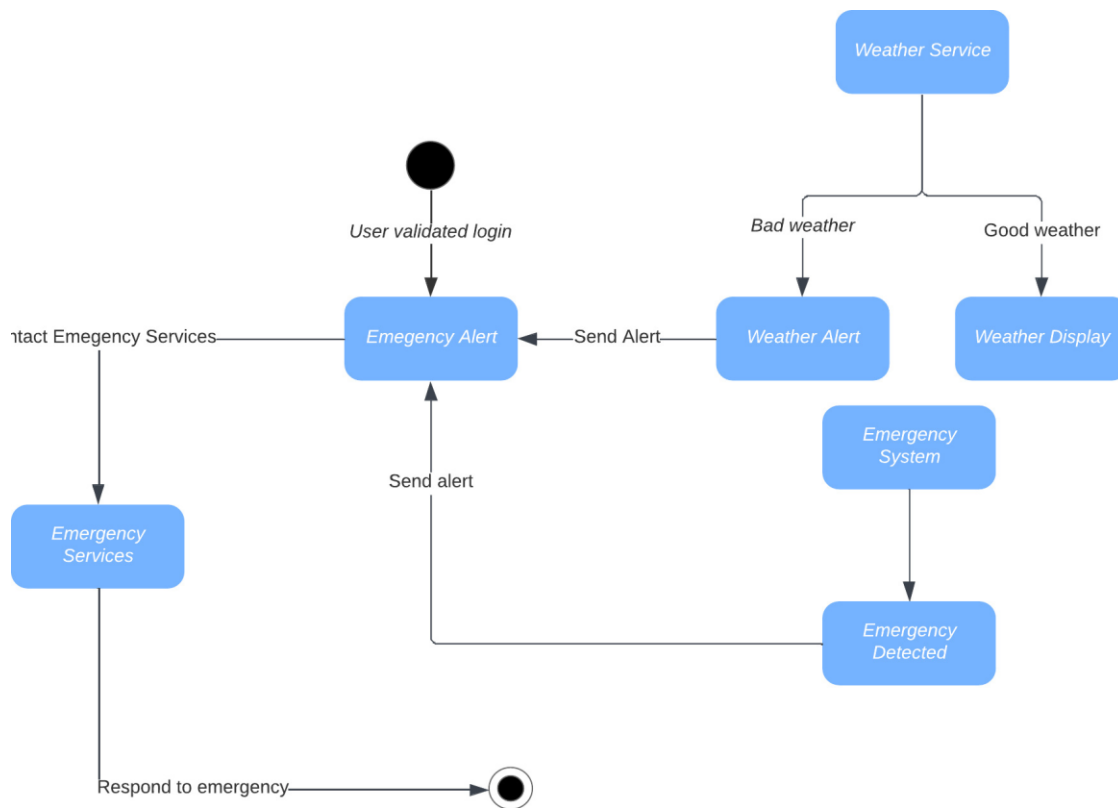


UML State Diagram

Below is the state diagram for the Dashboard feature. Assuming user has successfully validated the login.



State diagram for Weather/Emergency Systems



Agile/Scrum Planning

The burn-up chart below provides a visual representation of the planned vs actual progress of the tasks from the Work Breakdown structure provided above in 16 sprints/iterations.

