**Mentor Evaluation #1**

**FARMezy, CPG-106**

**Dated: 25 May 2021**

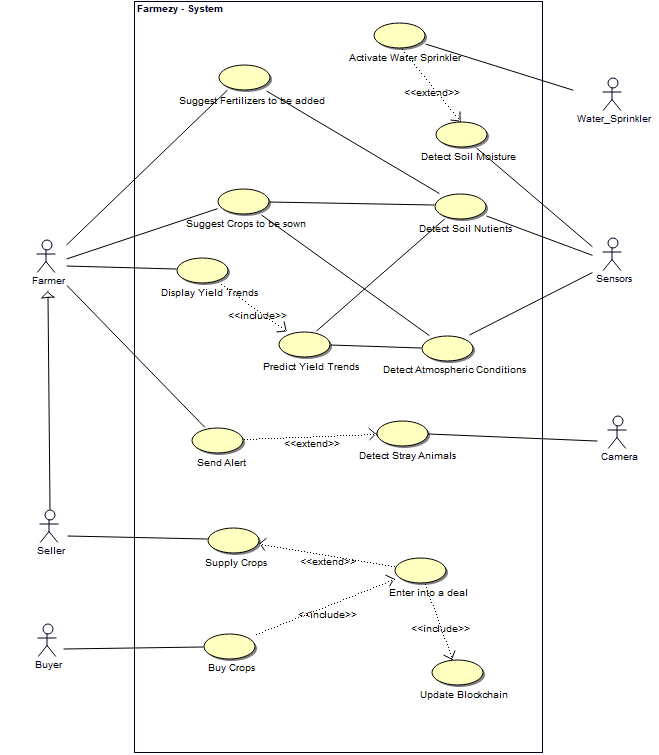
**Dr. Neeraj Kumar**

**INDEX**

Contents of this document.

|  |  |  |
| --- | --- | --- |
| S.No. | Topic | Page |
| 2. | Use Case Diagram & Descriptions (14) | 2-16 |
| 3. | List of Tasks & Subtasks (14) | 17-18 |
| 7.1. | Functional Requirements (14) | 19-20 |
| 7.2. | Non-Functional Requirements (5) | 20-21 |

**2. Use Case Model**



**2.1 Use Case #1 (Detect Soil Moisture)**

**Preconditions -**

1. Power should be available to the system.
2. All the connections in the system should be tight.
3. Sensors should be active and working.

**Post conditions -**

1. Activate Sprinkler Flag would be updated.
2. Activate Water Sprinkler use case will be triggered.

**Extends -** none

**Includes -** none

**Flow of Events**

1. **Basic Flow** 
   1. System triggers the soil moisture sensor.
   2. System retrieves the response from the soil moisture sensor.
   3. System updates the response in the database.
   4. System computes the need of activating the water sprinkler.
   5. System updates the activate sprinkler flag.
   6. (Extension Point - Activate Water Sprinkler)
   7. Repeat steps ‘a’ to ‘f’ after every 5 minutes.
   8. The use case ends.
2. **Alternative Flow**

b1. System fails to retrieve response from soil moisture sensor

1. System prompts the farmer about auto irrigation failure.
2. System lodges a complaint to the customer service for assistance.

f1. System fails to trigger Activate Water Sprinkler

1. System prompts the farmer about auto irrigation failure.
2. System lodges a complaint to the customer service for assistance.

**2.2 Use Case #2 (Activate Water Sprinkler)**

**Preconditions -**

1. Detect Soil Moisture use case should be triggered.

**Post conditions -**

1. Water Sprinkler would be activated as per the value of Activate Sprinkler Flag.

**Extends -** Detect Soil Moisture

**Includes -** none

**Flow of Events**

1. **Basic Flow** 
   1. System retrieves the value of activate sprinkler flag.
   2. System turns on the water sprinkler if the value of activate sprinkler flag is high.
   3. System turns off the water sprinkler if the value of activate sprinkler flag is low.
   4. Repeat steps ‘a’ to ‘c’ after every 5 minutes.
   5. The use case ends.
2. **Alternative Flow**

b1. System fails to turn on the water sprinkler due to connection issue

1. System prompts the farmer about auto irrigation failure.
2. System lodges a complaint to the customer service for assistance.

b2. Water Sprinkler is turned on but the pump turns off due to inadequate water.

1. System prompts the farmer about Inadequate Water Supply.

**2.3 Use Case #3 (Detect Soil Nutrients)**

**Preconditions -**

1. Power should be available to the system.
2. All the connections in the system should be tight.
3. Sensors should be active and working.

**Post conditions -**

1. Value recorded by the soil nutrient sensors will be stored in the database

**Extends -** none

**Includes -** none

**Flow of Events**

1. **Basic Flow** 
   1. System triggers the soil nutrient sensors.
   2. System retrieves the response from the soil nutrient sensors.
   3. System updates the response in the database.
   4. Repeat steps ‘a’ to ‘c’ after every 4 hours.
   5. The use case ends.
2. **Alternative Flow**

b1. System fails to retrieve response from soil nutrient sensor

1. System prompts the farmer about Nutrient Detection Failure.
2. System lodges a complaint to the customer service for assistance.

**2.4 Use Case #4 (Detect Atmospheric Conditions)**

**Preconditions -**

1. Power should be available to the system.
2. All the connections in the system should be tight.
3. Sensors should be active and working.

**Post conditions -**

1. Value recorded by temperature and humidity sensors will be stored in the database

**Extends -** none

**Includes -** none

**Flow of Events**

1. **Basic Flow**
2. System triggers the temperature and humidity sensors.
3. System retrieves the response from the temperature and humidity sensors.
4. System updates the response in the database.
5. Repeat steps ‘a’ to ‘c’ after every 30 minutes.
6. The use case ends.
7. **Alternative Flow**

b1. System fails to retrieve response from temperature sensor

1. System prompts the farmer about Temperature Sensor Failure.
2. System lodges a complaint to the customer service for assistance.

b2. System fails to retrieve response from humidity sensor

1. System prompts the farmer about Humidity Sensor Failure.
2. System lodges a complaint to the customer service for assistance.

**2.5 Use Case #5 (Detect Stray Animals)**

**Preconditions -**

1. Power should be available to the system.
2. All the connections in the system should be tight.
3. Camera should be active and working.

**Post conditions -**

1. Stray animal detected flag would be updated.
2. Send Alert use case will be triggered.

**Extends -** none

**Includes -** none

**Flow of Events**

1. **Basic Flow**
2. System triggers the camera.
3. System retrieves the response image from the camera.
4. System updates the response image in the database.
5. System detects the presence of stray animals in the image using object detection algorithms.
6. System updates the stray animal detected flag.
7. (Extension Point - Send Alert)
8. Repeat steps ‘a’ to ‘f’ after every 30 seconds.
9. The use case ends.
10. **Alternative Flow**

b1. System fails to retrieve response from Camera

1. System prompts the farmer about Stray Animal Detection failure.
2. System lodges a complaint to the customer service for assistance.

f1. System fails to trigger Send Alert

1. System prompts the farmer about Stray Animal Detection failure.
2. System lodges a complaint to the customer service for assistance.

**2.6 Use Case #6 (Send Alert)**

**Preconditions -**

1. Detect Stray Animals use case should be triggered.

**Post conditions -**

1. Alert would be sent to the farmer as per the value of the Stray animal detected Flag.

**Extends -** Detect Stray Animals

**Includes -** none

**Flow of Events**

1. **Basic Flow** 
   1. System retrieves the value of the Stray animal detected flag.
   2. System sends an alert to the farmer if the value of the Stray animal detected flag is high.
   3. Repeat steps ‘a’ to ‘b’ after every 2 minutes.
   4. The use case ends.
2. **Alternative Flow**

b1. System fails to send alert due to connection issue

1. System prompts the farmer about Stay animal detection failed.
2. System lodges a complaint to the customer service for assistance.

**2.7 Use Case #7 (Display Yield Trends)**

**Preconditions -**

1. Power should be available to the system.
2. Detect Soil Nutrients use case should be triggered.
3. Detect Atmospheric Conditions use case should be triggered.

**Post conditions -**

1. Yield Trends would be displayed to the farmer in the form of a visually readable graph.

**Extends -** none

**Includes -** Predict Yield Trends

**Flow of Events**

1. **Basic Flow**
2. Farmer clicks on the Display Yield Trends Option
3. System triggers Predict Yield Trends use case.
4. System fetches the output of the predict yield trends use case from the database.
5. System displays the output in a visually readable graph.
6. The use case ends.
7. **Alternative Flow**

c1. System fails to fetch the output from the database.

1. System prompts the farmer about Database Connection Failure Error
2. System lodges a complaint to the customer service for assistance.

**2.8 Use Case #8 (Predict Yield Trends)**

**Preconditions -**

1. Power should be available to the system.
2. Detect Soil Nutrients use case should be triggered.
3. Detect Atmospheric Conditions use case should be triggered.
4. Display Yield Trends use case should be triggered.

**Post conditions -**

1. Yield Trends would be stored in the system database.

**Extends -** none

**Includes -** none

**Flow of Events**

1. **Basic Flow**
2. System fetches the values recorded by the Soil Nutrient Sensors and temperature and moisture sensors from the database.
3. System runs ML models to predict the yield trends.
4. Predicted trends are stored in the database.
5. The use case ends
6. **Alternative Flow**

a1. System fails to fetch the output from the database.

1. System prompts the farmer about Database Connection Failure Error.
2. System lodges a complaint to the customer service for assistance.

**2.9 Use Case #9 (Suggest Fertilizers to be added)**

**Preconditions -**

1. Power should be available to the system.
2. Detect Soil Nutrients use case should be triggered.
3. Detect Atmospheric Conditions use case should be triggered.

**Post conditions -**

1. Suggestions for Fertilizers to be added will be stored in the database and displayed to the farmer

**Extends -** none

**Includes -** none

**Flow of Events**

1. **Basic Flow** 
   1. Farmer clicks on the Suggest Fertilizers to be added option.
   2. System fetches the values recorded by the Soil Nutrient Sensors and temperature and moisture sensors from the database.
   3. System runs ML models to predict the fertilizers to be added.
   4. Suggestions for fertilizers to be added will be stored in the database.
   5. Final Output will be displayed to the farmer.
   6. The use case ends.
2. **Alternative Flow**

b1. System fails to fetch the output from the database.

1. System prompts the farmer about Database Connection Failure Error.
2. System lodges a complaint to the customer service for assistance.

**2.10 Use Case #10 (Suggest Crops to be sown)**

**Preconditions -**

1. Power should be available to the system.
2. Detect Soil Nutrients use case should be triggered.
3. Detect Atmospheric Conditions use case should be triggered.

**Post conditions -**

1. Suggestions for crops to be sown will be stored in the database and displayed to the farmer

**Extends -** none

**Includes -** none

**Flow of Events**

1. **Basic Flow** 
   1. Farmer clicks on the Suggest Crops to be sown option.
   2. System fetches the values recorded by the Soil Nutrient Sensors and temperature and moisture sensors from the database.
   3. System runs ML models to predict crops to be sown.
   4. Suggestions for crops to be sown will be stored in the database.
   5. Final Output will be displayed to the farmer.
   6. The use case ends.
2. **Alternative Flow**

b1. System fails to fetch the output from the database.

1. System prompts the farmer about Database Connection Failure Error.
2. System lodges a complaint to the customer service for assistance.

**2.11 Use Case #11 (Buy Crops)**

**Preconditions -**

1. Power should be available to the system.
2. Network Connection should be available.
3. Buyer should be registered

**Post conditions -**

1. Buyer and Seller will enter a deal, and the Enter into a Deal use case will be triggered.

**Extends -** none

**Includes -** Enter into a Deal

**Flow of Events**

1. **Basic Flow** 
   1. Buyer clicks on the Buy Crops option.
   2. System fetches the list of all available farmers who are willing to sell.
   3. Buyer chooses the preferred seller from the list.
   4. Buyer and Seller Enter a deal, and Enter into a Deal use case is triggered.
   5. The use case ends.
2. **Alternative Flow**

b1. System fails to fetch the list of available sellers from the database.

1. System prompts the buyer about Database Connection Failure Error.
2. System lodges a complaint to the customer service for assistance.

**2.12 Use Case #12 (Enter into a deal)**

**Preconditions -**

1. Power should be available to the system.
2. Network Connection should be available.
3. Buyer should be registered

**Post conditions -**

1. Buyer will get the supply of crops if the deal is successful, and Supply Crops use case will be triggered.
2. Update Blockchain use case will be triggered.

**Extends -** none

**Includes -** Update Blockchain

**Flow of Events**

1. **Basic Flow** 
   1. Buyer offers his final price for the crops.
   2. Seller chooses to accept or reject the offer.
   3. If the seller chooses to accept the offer, supply crops use case is triggered.
   4. (Extension Point- Supply Crops)
   5. Update blockchain use case is triggered.
   6. The use case ends.
2. **Alternative Flow**

b1. System fails to complete the deal due to network failure.

1. System prompts the buyer and seller about Network Connection Failure Error.
2. System returns to the homescreen for both buyer and seller.

**2.13 Use Case #13 (Supply Crops)**

**Preconditions -**

1. Power should be available to the system.
2. Network Connection should be available.
3. Buyer should be registered
4. Deal between buyer and seller should be successful.

**Post conditions -**

1. Buyer will get the delivery of crops, and the confirmation for the same would be stored in the database.
2. Seller will get the decided amount of money as stored by the system in the blockchain, and confirmation for the same would be stored in the database.

**Extends -** Enter into a deal

**Includes -** none

**Flow of Events**

1. **Basic Flow**
2. Buyer is prompted to ensure the delivery of the crops.
3. Buyer chooses yes or no, and this is stored as a confirmation in the database.
4. Seller is prompted to ensure the credit of the amount as decided during the deal.
5. Seller chooses yes or no, and this is stored as a confirmation in the database.
6. This is further validated by an authorised officer.
7. Use case ends.
8. **Alternative Flow**

b1. Buyer fails to choose yes or no due to network failure.

1. System prompts the buyer about Network Connection Failure Error.
2. Buyer is prompted again to ensure the delivery of the crops.

b2. Seller fails to choose yes or no due to network failure.

1. System prompts the seller about Network Connection Failure Error.
2. Seller is prompted again to ensure the credit of the amount as decided during the deal.

**2.14 Use Case #14 (Update Blockchain)**

**Preconditions -**

1. Power should be available to the system.
2. Network Connection should be available.
3. Enter into a deal should be triggered.

**Post conditions -**

1. All the information regarding the deal including the offered price, and acceptance status will be updated in the blockchain.

**Extends -** none

**Includes -** none

**Flow of Events**

1. **Basic Flow**
   1. System creates a hashnode comprising all the details of the deal including the offered price and acceptance status of the seller.
   2. System adds the hashnode into the blockchain.
   3. Hashnode gets validated by all the occurrences of the blockchain.
   4. Use case ends
2. **Alternative Flow**

b1. System fails to add the hashnode to the blockchain due to network failure.

1. System saves that copy of the hashnode locally and adds it to the blockchain as soon as the network is available on a priority basis.

**3. To identify the set of complete tasks and subtasks of the project.**

1. **Project Proposal & Literature Survey**
   1. Idea Discussion With Mentor
   2. Feasibility Evaluation
   3. Literature Survey
   4. Project Proposal
   5. Proposal Acceptance
2. **Designing Model & Sensor Testing**
   1. Designing Model
   2. Simulating Model
   3. Acquiring Sensors
   4. Testing Sensors
   5. Mentor Approval For Design
3. **Hardware Interfacing**
   1. Acquiring Circuit Components
   2. Assembling Circuit
   3. Programming Source Code For Raspberry Pi
   4. Testing WiFi Module & Raspberry Pi
   5. Testing Camera Unit
   6. Testing Assembled Circuit
   7. Improvising Circuit Design As Needed
4. **ML Modelling**
   1. Understanding Project Requirements
   2. Elaborating Required Features
   3. Mentor Discussion For Guidance
   4. Deciding Appropriate ML Model
   5. Programming Source Code
   6. Implementing ML Model
   7. Debugging & Dry Run Of Test Cases
   8. Improvising ML Model As Needed
5. **Mobile Application**
   1. Understanding Mobile Application Requirements
   2. Categorising Functional & Non-Functional Requirements
   3. Deciding Appropriate Development Framework
   4. Programming Source Code
   5. Deciding Appropriate Backend Solution
   6. Designing Relational Database Management Schema
   7. Programming Source Code
   8. Debugging & Dry Run Of Test Cases
   9. Improvising Mobile App & Database As Needed
6. **Learning & Implementing Object Detection using Neural Networks**
   1. Learning About Neural Networks
   2. Learning About Object Detection
   3. Mentor Discussion For Guidance
   4. Programming Source Code
   5. Debugging & Dry Run Of Test Cases
   6. Improvising Object Detection Model As Needed
7. **Integration**
   1. Linking Mobile App To Database
   2. Linking Raspberry Pi To Database
   3. Linking ML Models To Database
   4. Linking Object Detection Model To Database
   5. Integrating Software & Hardware Components
8. **Testing and Debugging**
   1. Testing Responsiveness Of Integrated System
   2. Debugging & Dry Run Of Test Cases
   3. Integrated System Feature Evaluation
   4. Improvising Integrated System As Needed
   5. Mentor Approval For Integrated System
9. **Learning Blockchain**
   1. Learning About Blockchain
   2. Understanding Key Benefits Of Blockchain
   3. Mentor Discussion For Guidance
   4. Deciding Appropriate Development Platform
   5. Designing Blockchain
10. **Implementing and integration**
    1. Programming Source Code
    2. Implementing Blockchain
    3. Integrating Blockchain Into Mobile App
11. **Testing and Debugging**
    1. Debugging & Dry Run Of Test Cases
    2. Blockchain Feature Evaluation
    3. Improvising Blockchain As Needed
12. **Performing Modifications**
    1. Identifying Strengths & Weaknesses
    2. Improvising Project As Needed
    3. Mentor Approval For Project
13. **Result Evaluation**
    1. Team Evaluation
    2. Mentor Evaluation
    3. Panel Evaluation
    4. Submission Of Final Project
14. **Final Report**
    1. Compiling All Documents Into Single Report
    2. Mentor Approval For Report
    3. Submission Of Final Report

**7. To identify all the requirements of the project and to classify them into Functional and Non-Functional requirements.**

**7.1 Functional Requirements**

The functional requirements of this project are listed below :

**F1:** The system will trigger soil moisture sensor every 5 minutes to compute the need of irrigation, and in turn activation of water sprinkler.

**F2:** The system will check the activate-sprinkler-flag value every 5 minutes to toggle the state of the water sprinkler according to need.

**F3:** The system will trigger the soil nutrient sensor every 4 hours to update soil nutrient value in the database.

**F4:** The system will trigger the temperature and humidity sensors every 30 minutes to update atmospheric conditions in the database.

**F5:** The system will trigger the camera every 30 seconds to capture an image of the field, and in turn detect the presence of stray animals, and toggle the stray-animals-detect-flag value.

**F6:** The system will check the stray-animals-detect-flag value every 2 minutes to conditionally send an alert to the user in danger.

**F7:** The system will trigger #F8: Predict Yield Trends use case, and in turn fetch data of predicted trends from the database, to display the yield trends in a visually readable graph, whenever the user clicks on the said option.

**F8:** The system will fetch data of sensors from the database, to run ML models to predict the yield trends and upload to the database, whenever the user clicks on the said option.

**F9:** The system will fetch data of sensors from the database, to run ML models to predict and display the suggested fertilisers for adding to soil, whenever the user clicks on the said option.

**F10:** The system will fetch data of sensors from the database, to run ML models to predict and display the suggested crops for sowing, whenever the user clicks on the said option.

**F11:** Buyer and Seller will enter a deal, and the #F12: Enter into a Deal use case will be triggered.

**F12:** Buyer will get the supply of crops after a successful deal is made, and #F13: Supply Crops use case and #F14: Update Blockchain use case will be triggered.

**F13:** Buyer will get the delivery of crops, Seller will get the decided amount of money, and confirmation will be stored in the database.

**F14:** The system will create a hashnode containing all details of the deal, to add it into the blockchain and validate by all the occurrences.

**7.2 Non-Functional Requirements**

**7.2.1 External Interface Requirements**

**A- User Interfaces**

The User Device Requirements are listed below :

● Android v8.0 and higher

● iOS v11.0 and higher

● RAM 3GB and higher

● Stable internet connection

**B- Hardware Interfaces**

**B1- FARMezy Hardware Unit**

● Raspberry Pi Microcontroller

● Various Sensors

● Camera Unit

**B2- Software Development PC**

● Processor : Intel i7 9th gen 9750H CPU @ 2.60 GHz (6 cores, 12 threads, L3 cache 12 MB)

● RAM : 16 GB DDR4 @ 2667 MHz

● Storage : 256 GB SSD NVME + 1 TB HDD SATA

● Graphics : NVIDIA GeForce GTX 1650 - 4 GB GDDR5

● OS : Microsoft Windows 10 Home Version 20H2 Build 19042.985

**C- Software Interfaces**

**C1- Front end** = (app) Flutter Framework with Dart programming language

**C2- Back end** = (app) FireBase + (ML models) Python + Blockchain + (rasp pi) Ubuntu

**7.2.2 Performance Requirements**

● All the user details are stored locally also, so user details and processed information are always available.

● Fetching and storing data from the database should be done in less than 2 seconds.

● All the processing should be done at the database (server) side to achieve peak efficiency.

**7.2.3 Safety and Security Requirements**

● Secured access of farmer records

● Adding and removal of farmers is accessible by administrator only

● All the data to be sent via HTTP protocol for better encryption

● Backup of farmer database at regular intervals to avoid any data loss

● User details that are stored locally are to be encrypted to avoid data leak

● Strict adherence to authorised blockchain tampering

● Hardware Unit installed in a secured/concealed spot

**7.2.4 Software Quality Attributes**

● While sending data to the database it should be stored locally first, so in case internet service gets disrupted, data can be uploaded again without any delay.

● User device details to be registered to avoid hacking

**7.2.5 Database Requirements**

All the data for farmer records will be stored in a relational database schema. For the time being, the servers of FireBase will be used.

----------------------------------------------------------- End of Document -----------------------------------------------------------