College: Engineering and Information Technology

Department: Information Technology

INT305 Fundamentals of Software Engineering
Fall 2024-25_1
Assignment: Group Project

WeAgri - Smart Agricultural System

Prepared by:

Amnah Khaled – 202210779

Hala Renan – 202211790

Huda Mohammed Bilal – 202211270

Lubna Sher Aslam - 202120102

Nouf Abdullah - 202111359

Supervised by:

Dr. Shafiz Mohd Yusof

Table of Contents

Introduction

In this era, where drastic climate change is affecting the livelihoods of many people, especially farmers, we proposed an idea to create software that targets some of the UN Sustainable Development Goals. Our software targets mainly farmers and wishes to make their lives easier by providing them with different kinds of services and tools but with sustainability in mind. It provides services such as providing information about crops, connecting farmers with local businesses and many more. It wishes to provide an interface to sensors which manage many different tasks. By using technology, we can help farmers increase their productivity, improve their daily lives and build a more sustainable system.

The Idea of our software

Our software aims to provide a platform for farmers to assist them in farming by acquiring them with the necessary knowledge and tools, giving them guidance with each step and connecting them with local suppliers. By providing farmers with an AI chat tool, they can easily clear their confusion or doubts regarding farming if any. They can even request a farming specialist for extra help. Our platform supports sustainable practices by providing farming materials on rent for farmers if they are willing. It also aims to provide farmers with a platform where they can sell their harvested crops once it's ready. This software can be accessed by all types of people, but the set of functions differs for different kinds of users. It aims to connect local farming businesses together by giving them a platform on which they can advertise their businesses. This helps farmers connect with the local suppliers and use low cost materials.

Our software wishes to become a platform that can provide volunteering opportunities for unemployed individuals who are interested in assisting farmers or are interested in other related professions. Users who have large quantity of leftover / close to expiry food can access our platform and request for food waste pickup team, and we as the organization can recycle the food by providing it to the interested farmers for the purpose of composting. Our Software also wishes to provide an interface for farmers who request sensors for farming. It can help farmers to control their water sprinklers and check the surrounding weather conditions if they are connected to the internet. Our sensors are charged by solar cells which are sources of renewable energy. By implementing all these various functionalities in our software, we wish to make the lives of farmers and other individuals easier.

Our Software's Purpose

After proposing our idea, we discovered that the market lacks such software that could specifically help with farming needs, the absence of specialized software forces many farmers to depend on conventional farming techniques, which may not be as efficient or productive in the current era of data-driven agriculture. Such issues can be fixed by creating an innovative and high tech-driven software.

With this software, we are enabling farmers and those with interest in farming to transition from traditional, time consuming, tiresome, and inefficient farming techniques. Modern day farmers are faced with challenges including reduced input, unfavourable climate change, and the rising global call for organic farming. Farmers require solutions to organize their activities, find reliable information and make decisions faster.

The main purpose of this software is to make the farming process smooth and less time consuming by increasing efficiency and profitability and promoting sustainability. The software also aims at bringing a change to the daily routines of farmers by providing important tools and resources. It supports informative decision making by helping regulate tasks, managing farming resources, creating schedules, and giving access to information whether through experts or AI. This gives farmers extra time to help expanding their operations and achieve higher global market standards.

The software focuses on boosting productivity, connecting and strengthen direct links with farmers and local markets, by encouraging farmers to sell their products within their communities. This results in depending less on imported goods, by making sure that fresh, locally grown products are always available, which enlarges income and growth potentiality.

The software also creates new work opportunities. As the farming operation increases, the demand for transportation, supplies maintenance services and supplies from local businesses increase as well. By creating an interface that joins farmers with these service providers, the software encourages partnerships and creates jobs within the local economy.

The software encourages sustainable and eco-friendly farming practices by reducing and recycling food waste, by collecting it and making a fertilizer by composting it. This fertilizer enhances the quality of the crop as well as the nutrients on the soil. It is fully natural and has no harmful chemicals in it.

Our software and the UN Sustainable Development Goals

Our program is designed to directly address some of the United Nations Sustainable Development Goals by improving agricultural practices, boosting farming scarcity, promoting sustainability and creating new economic opportunities in the United Arab Emirates. Some of the specific Sustainable Development Goals our program promotes and will cover are:

Sustainable Development Goal 2: Zero hunger

Our program contributes to food security by providing tools that help farmers increase yields. With tools like locust and weather alerts and advice from agricultural professionals, we assist farmers in reducing crop risks and guaranteeing improved food production and accessibility in the United Arab Emirates. As we all are aware, locally produced farming goods take time and a lot of effort to grow in the climatic conditions of UAE. Our software targets to reduce these farming crises as well as rely less on import goods and increase the productivity of locally produced goods. The approach will give UAE's export business a boost as well.

Sustainable Development Goal 12: Responsible consumption and production

In the United Arab Emirates, our initiative promotes sustainable farming practices by permitting food waste to be recycled and encouraging composting. It connects local markets with farmers, enables not only them but also consumers, to reuse waste and organic materials, reduce waste and encourage responsible consumption.

Sustainable Development Goal 13: Climate Action

Our software provides climate-related information, such as weather alerts and locust alerts, that help farmers adapt to the effects of climate changes in the UAE. By improving their capacity to respond to these challenges, the program helps the resilience of agricultural systems, by recovering quickly from difficulties/ toughness, in areas affected by climate variability.

Sustainable Development Goal 8: Work growth and Economic expansion

Although the remarkable development in the agricultural sector UAE has made, there still lay some underlying challenges that need to be addressed other than agricultural production. Through our program, local business owners and individuals with no jobs can participate in agricultural services and support. By creating these economic opportunities, we help drive sustainable economic growth within UAE as well as promote and target decent work in rural and agricultural communities.

These challenges are all further aggravated by UAE's reliance on imported food, with 80-90% of the food being imported. In some areas of the UAE managing water resources efficiently is crucial, and innovative systems are essential for improving food security and reducing dependence on imports. Using our idea and approach, we support the UAE's agricultural sector in overcoming these challenges in all parts of the country while contributing to broader the UN goals.

Related statistics

- FAS Post Dubai reports that due to the arid climate, approximately 80% of the UAE's agricultural products are imported. The UAE imported US$16 billion worth of agricultural goods in 2021, a 1.2% increase over 2020. (UAE-Northeast-2023, n.d.)
- (Grocery retail industry sales reached US$19.6 billion in 2022, up 5.4% over the previous year. Imports of consumer-oriented agricultural products increased to US$13.2 billion in 2022, an 18.5% increase over 2021 due to the economic recovery spurred by a revival in tourism, government initiatives promoting food security and trade, supply chain stabilization post COVID-19 disruptions, strong hiring, and trade pacts.) (UAE-2022, n.d.)
- Supply shortages, rising gas prices and global inflationary pressures pose major challenges to the UAE retail industry in 2022. Notably, the cost of gas, electricity and other fuels in the UAE increased 2.9% with a surge in inflation from 0.2% to 4.8 % in 2021 and 2022, respectively, driven by increases in housing, transportation, and global food prices. As a result, the cost of food for domestic use increased by 8.3%. (UAE-Northeast-2023, n.d.)

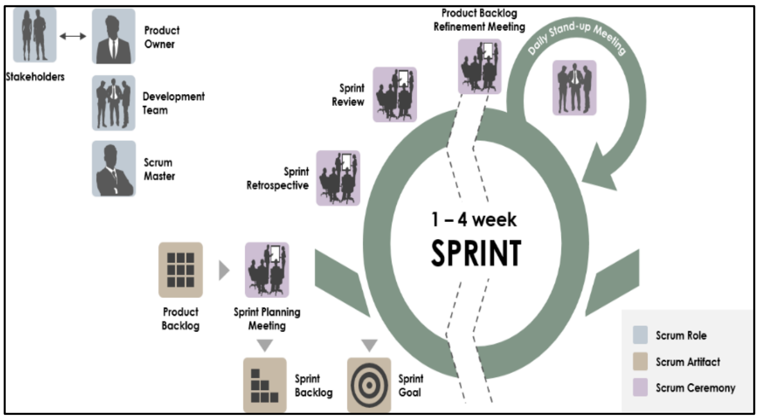
By promoting sustainable agricultural implementations and increasing access to resources as well as creating other opportunities, we can easily move towards decreasing some of these problems along with the underlying challenges and advance the broader goals of the United Nations.

Software Development Life Cycle Planning model

Scrum

The Software development life cycle model that we will use to build our project is Scrum. Scrum is one of the Agile Process Models. Agile Process Models are useful for breaking large complex projects into smaller parts. In Agile (Scrum), the priority is given to customer collaboration which is done by discussing any changes and getting feedback from them after every small work product is created. It focuses on making sure that the software is functioning well. Each change is monitored and properly implemented. The work is divided among the team, and the team reviews their progress at the end of a certain period. Extensive and regular communication is done with the stakeholders and the product owner.

In Scrum, Sprints are short 1-4 weeks period, where you do daily meetings, and discuss what needs to be done and analyze what was done. Before starting the sprint, you create a plan for it. At the end of the sprint, you review your progress and your work product and your retrospect on what needs improvement for the next sprint. A small work product is created at the end of the sprint. Sprint Backlog – document (to – do list) that stores the requirement / functions that need to be accomplished in each sprint.



Why are we using Scrum?

The purpose of using scrum as our developmental model is because of having flexible requirements and product. Since the stakeholders are open to changes and are not very specific to what they require, scrum can be ideal in this situation. Also, we don’t wish to follow a traditional approach to building software and not being able to go back to fix any mistake in each step. We chose scrum because it is iterative as well. When looking at the above-mentioned advantages, we decided that scrum is the most suitable for developing our project. But also keeping in mind and being aware of its disadvantages as well.

The Advantages and Disadvantages of Scrum

Advantages	Disadvantages
Focus on collaborating and communicating with the stake holders and developers. (Everyone is involved)	It is difficult for everyone to devote/commit their time to the project. Constant meetings, for example standup meetings may make the team members feel fatigued and less productive.
Adapts to changing requirements and priorities throughout software development.	With more team members, scrum might prove difficult to implement, as it is difficult to handle and manage large numbers of people. Increase in the number of team members might increase conflicts between the members, increase complexity and decrease collaboration.
A working version (not with full functionalities) model of the system is developed earlier compared to some of the SDL models. This helps the developers to improve and work on any changes to the product as the customers give their feedback.	Since Scrum is a flexible model that adapts well to changing requirements throughout the project, it is not efficient to implement it to projects that have fixed requirements.
Since Scrum involves short iterative sprints with regular inspection, any risks or issues can be detected earlier and can be managed.	Having a model that is very flexible and frequently getting feedback from the customer can become a boon for the developers as well, if the developers are not able to put clear boundaries (scope). Going over the project boundaries (Scope creep) can result in: ■ Delays in Projects ■ Increased Costs ■ Less focus on Quality ■ Extending the timeline.

The Project’s Schedule

Sprint 1 (Week 1 + 2)	Sprint 2 (Week 3 + 4)	Sprint 3 (Week 5 - 8)	Sprint 4 (Week 9 - 12)
Create a list of stakeholders who are involved and communicate with them.	Create and discuss Scenario based, class-based, and behavioral-based analysis models	Developers need to build different prototypes which need discussed with the stakeholders	The formal technical review will review the product and check the quality of the product, fix any bugs.
Try to understand the nature of the problem and inquire about it	Try to negotiate with the stakeholders by generating a realistic set of requirements	Select the prototype which fulfills all the needs of the stakeholders and is realistic and achievable by the developers	Check if all the requirements are implemented and functioning properly
Address/Define the problem we need to solve and understand how the product fits into the business needs	Classify, organize and define relationships between the requirements	Add the core/important + additional functionalities	Check if the stakeholder is satisfied with the product
Identify the goals and create objectives	Review and correct the organized requirements if needed	Test and check if the product is functioning well	Deploying the Final Product and monitoring it’s functioning
Allocate needed resources	Order the requirements based on the stakeholders’ priorities	Create user interfaces, written documentation, and a manual on how to use the product	Implement and manage the changes and create a documentation of it and create traceability tables.

The Project’s Schedule (in detail)

This Project will be a top priority for our company, as it wishes to accomplish the SDG goals for the UN. In Order to accommodate this project within our company, we plan to finish it in 4 sprints spanning over 12 weeks. Our Companies Goals are to maintain top notch quality and to make sure each need of the customers is fulfilled in time but also keeping aware of the project’s boundaries. Keeping these goals in mind, we define the following schedule.

- Time/duration – 12 weeks
- Total sprints – 4 Sprints

Sprint 1 – Week 1 + Week 2

Goal for Week 1: Get to know your stakeholders

Key tasks / sprint backlog:

- Create a list of stakeholders who are involved
- Try to understand the nature of the problem
- Try to inquire about the problem

Products achieved: List of stakeholders and their directories

Goal for Week 2: Collaborate with the customer

Key tasks / sprint backlog:

- Communicate with the customer
- Address/Define the problem we need to solve
- Identify the goals and create objectives
- Understand how the product fits into the business needs
- Allocate needed resources

Products achieved: A written document of requirements

Sprint 2 – Week 3 + Week 4

Goal for Week 3: Create Analysis Models

Key tasks / sprint backlog:

- Create Scenario based analysis models
- Create class-based analysis models
- Create behavioral-based analysis models

Products achieved: Refined Lists of Analysis Models

Goal for Week 4: Refine/Finalize Requirements

Key tasks / sprint backlog:

- Discuss the analysis models created in Week 3 with the stakeholders
- Try to negotiate with the stakeholders by generating a realistic set of requirements
- Classify and organize the requirements
- Define relationships between the requirements
- Review and correct the organized requirements if needed
- Order the requirements based on the stakeholders' priorities

Products achieved: List of clear, organized and distinct set of requirements

Sprint 3 – Week 5 + Week 6 and Week 7 + Week 8

Goal for Week 5 and week 6: Create a prototype

Key tasks / sprint backlog:

- Developers need to build different prototypes
- Discuss the prototype with the stakeholders
- Select the prototype which fulfills all the needs of the stakeholders and is realistic and achievable by the developers

Products achieved: A refined prototype

Goal for Week 7 and week 8: Finalize the product and test it

Key tasks / sprint backlog:

- Add the core/important functionalities
- Add additional functionalities
- Test and check if the product is functioning well

- Create written documentation of the product
- Create user interfaces for the product
- Create a manual/guide on how to use the product

Products achieved: A fully functional product along with its guide

Sprint 4 - Week 9 + Week 10 and Week 11 + Week 12

Goal for Week 9 and 10: Check our products' quality

Key tasks / sprint backlog:

- The formal technical review will review the product
- Check if the product meets the required quality
- Check if the product is free of bugs
- Check if all the requirements are implemented and functioning properly
- Check if the stakeholder is satisfied with the product
- Deploying the Final Product

Products achieved: Deployment of a Quality assured product

Goal for Week 11 and 12: Implement any changes + create additional documents

Key tasks / sprint backlog:

- Implement any changes in the product
- Manage the changes and create a documentation of it
- Fix any issues
- Create traceability tables (which will help us with any projects in the future)
- Monitor the use and functioning of our product

Products Achieved: Traceability tables

How much will the Project cost?

Tasks	Cost
Designing and implementing the user interface	$10,000 / 40,000 AED
Developing the software – programming it + testing	$35,000 / 140,000 AED
Managing + Monitoring the Project	$12,000 / 48,000 AED
Reviewing the Project (Formal technical review team)	$13,800 / 55,200 AED
Risk Assessment	$10,900 / 43,600 AED
Other resources	Approx. $5000 / 20,000 AED
Total	Approx. $86,700 346,800 AED

Our goal is to be realistic and budget friendly since the project is for the UN and to make the farmers' jobs easier. So, to build this software it will take approximately $86,700 which is 346,800 dirhams.

System Requirements and Analysis (Modelling)

Requirements Document

1. Functional Requirements:
    1. Log in to account / Create new account
    2. Request for farming services:
        - Request for guidance
        - Request a farming schedule
        - Request for renting farming materials
        - Request for sensor.
    3. Request for general services:
        - Request a food waste pick up team
        - Request for volunteering opportunities
        - Request for purchasing crops
        - Request for marketing their business
    4. Can sell harvests
    5. Give Advice and Guidance to Farmers
    6. Allow/Deny requests
    7. Manage Users and Farming Specialists Information
    8. Sent notifications about alerts to farmers
    9. Sent notification to drivers for any pick-up or deliveries.
    10. Sensors Check/Display Weather data
        - Check/Display temperature
        - Check/Display Humidity
        - Check/Display Wind speed
        - Check/Display the surrounding ph. level
    11. Control Water Sprinklers
    12. Display Battery Level

2. Non – Functional Requirements
    1. Easier navigation
    2. User friendly
    3. Protection of personal data
    4. Double authentication when logging in
    5. Quick response time
    6. Quickly fix any issues
    7. Implement and manage changes
    8. Users can customize settings to individual preferences
    9. Display information to users based on their region
    10. Adjust to users' language and region

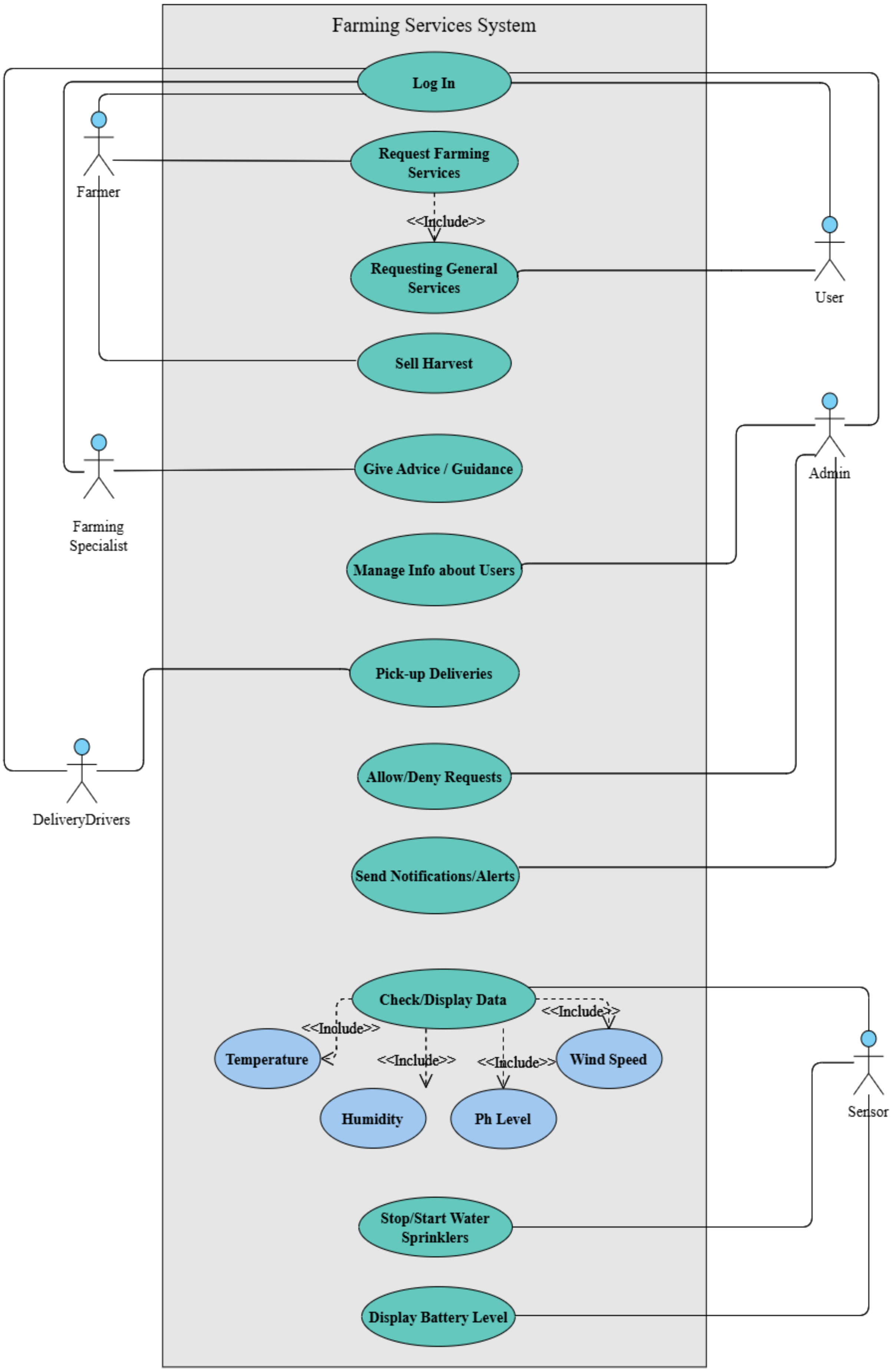
3. Actors Involved
    1. Farmers
    2. Users
    3. Admin
    4. Farming Specialist
    5. Delivery Drivers
    6. Sensor

UML Diagrams

Use Case Diagram

Our software is used by Users, Farmers, Farming specialists, drivers, Admin and a sensor which is an outside entity interacting and sending data to our system. Except for sensors (outside entity), all the actors need to log in to the system. Admins manage information about all the users. They can even send notifications about any alerts to farmers. Farmers are special types of users. They can request general services and farming services, but Users can only request general services. Farmers can also sell their harvest once the crops are harvested. (All farmers are users but not all users are farmers). A Farming Specialist uses the system to give advice or guidance to any farmer who requested it. Delivery Drivers are the ones who are responsible for delivering and picking up any orders. They use the system to check their deliveries that need to be made. A sensor is an external entity that uses our system as an interface for the farmers. Sensors can check display temperature, humidity, Ph level and the wind speed. They can even control the water sprinklers located on a farm. Sensors are charged by Solar cells. They can use the system to display their battery levels.

(For the sake of clarity and to not make our diagram cluttered, we didn't list each of the general and farming services in the use case diagram)

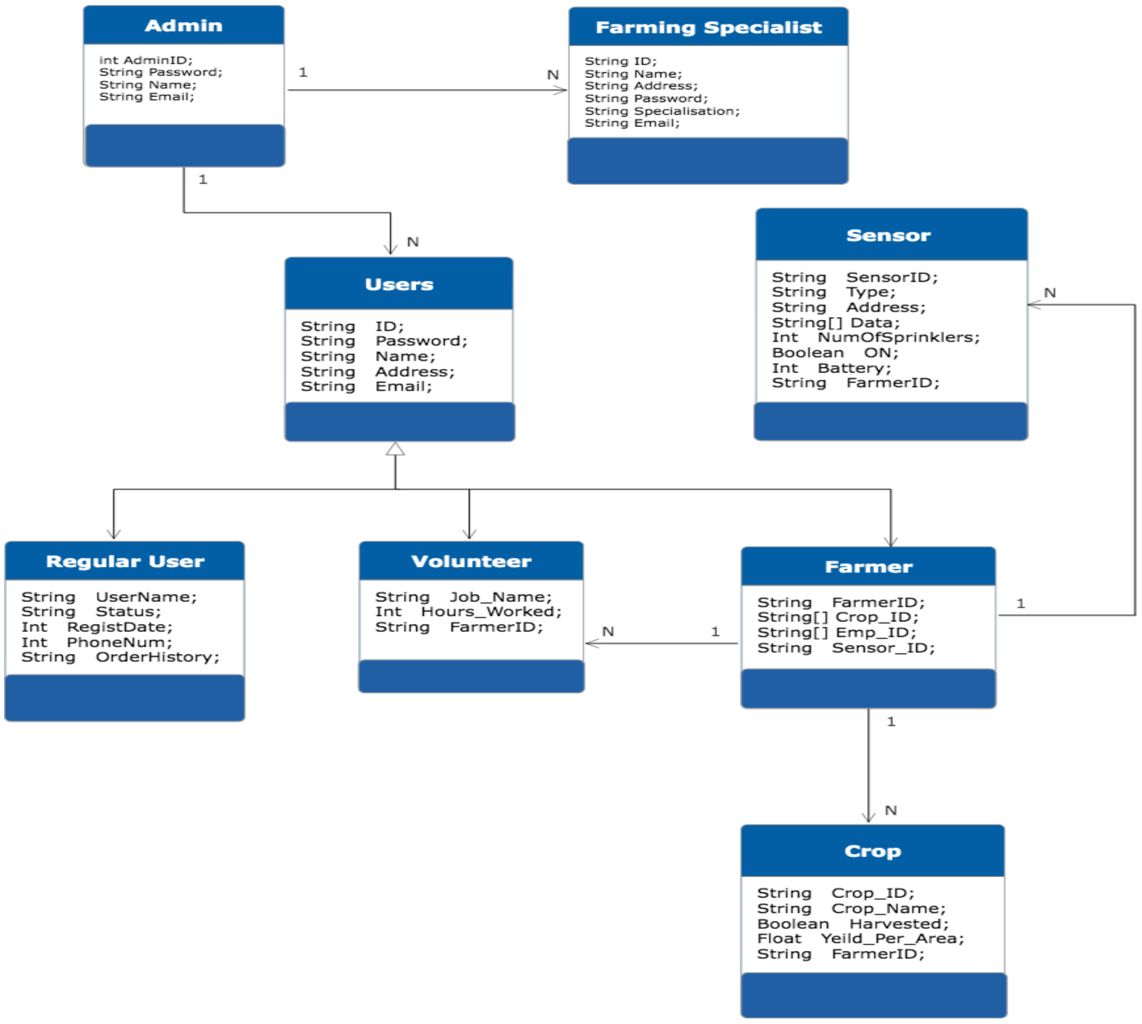

Farming Services System
Log In
Request Farming Services
<<Include>>
Requesting General Services
Sell Harvest
Give Advice / Guidance
Manage Info about Users
Pick-up Deliveries
Allow/Deny Requests
Send Notifications/Alerts
Check/Display Data
<<Include>>
<<Include>>
<<Include>>
<<Include>>
Temperature
Humidity
Ph Level
Wind Speed
Stop/Start Water Sprinklers
Display Battery Level
Farmer
Farming Specialist
DeliveryDrivers
User
Admin
Sensor

Class UML Diagram

The requests from users on our software are managed my admins. An admin is identified by name and email. An admin can manage 1 or more than one farming specialist. A farming specialist is identified by its name, address and what they specialize in.

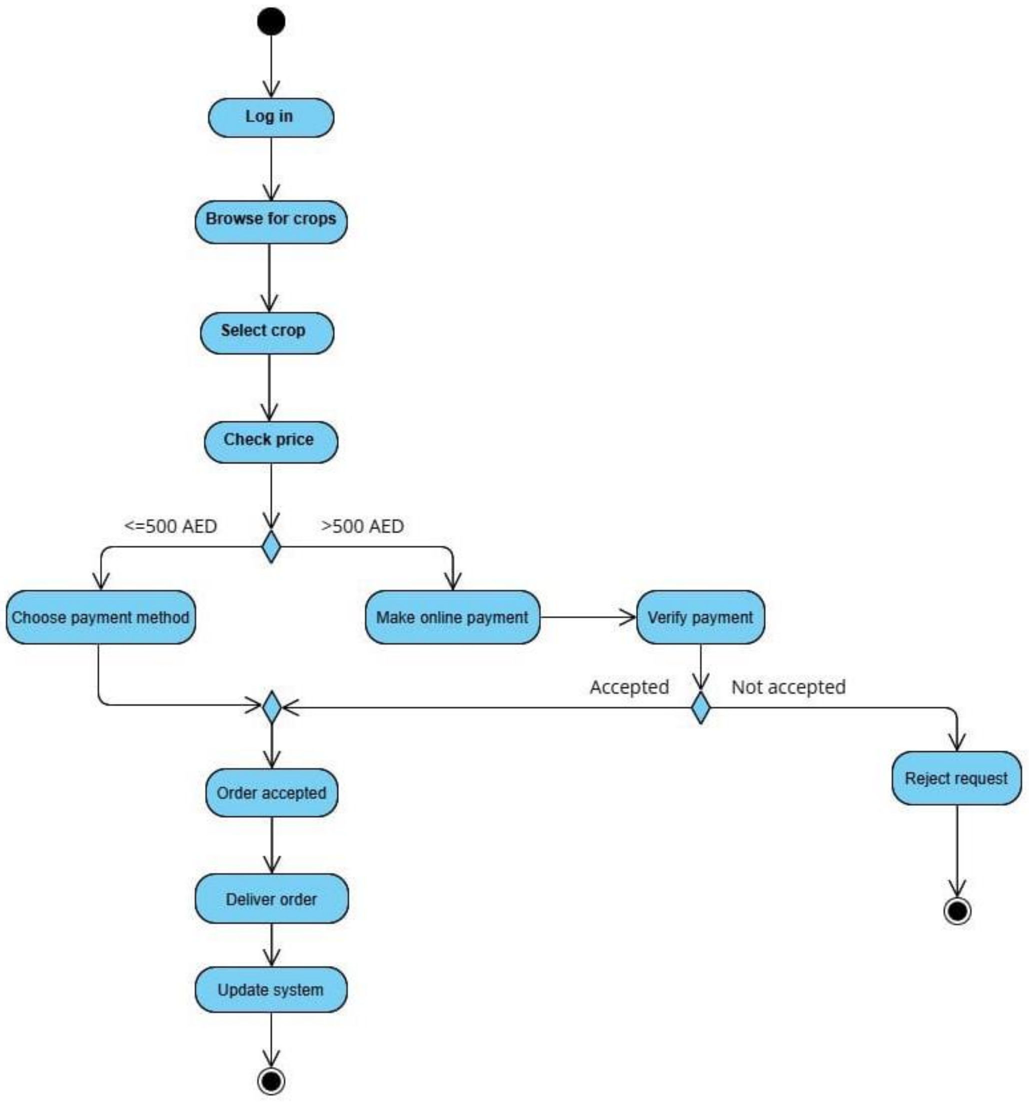
An Admin can also manage many users. Users can be of 3 types. We have regular user, volunteers and farmers. Regular users can request for general services. Once a regular user is registered, it is assigned a unique username, and the date of registration is noted. A user's status determines if he/she is still an active user or not. Volunteers are those users who are unemployed and are interested in working or are willing in helping a farmer in any tasks. Their total hours of work, the ID of the farmer they work for and the job they do is noted. A Farmer can have many volunteers working for them. Farmers are those special users who are working as a farmer and cultivating and harvesting different kinds of crops. A farmer can cultivate many crops. Each crop has a unique ID. Each of them is identified by their names, the total area they occupy (yield per area) and if they are harvested yet or not.

To make farming easier and less time consuming, a Farmer can have many sensors. A sensor is identified by a unique ID and the specification or type of sensor it is. It contains different pieces of information about the weather. Sensors can switch ON/OFF the water sprinklers. It is charged by solar cells and its battery is displayed.


Admin
int AdminID;
String Password;
String Name;
String Email;
1
N
Farming Specialist
String ID;
String Name;
String Address;
String Password;
String Specialisation;
String Email;
1
N
Users
String ID;
String Password;
String Name;
String Address;
String Email;
Sensor
String SensorID;
String Type;
String Address;
String[] Data;
Int NumOfSprinklers;
Boolean ON;
Int Battery;
String FarmerID;
N
Regular User
String UserName;
String Status;
Int RegistDate;
Int PhoneNum;
String OrderHistory;
Volunteer
String Job_Name;
Int Hours_Worked;
String FarmerID;
N
1
Farmer
String FarmerID;
String[] Crop_ID;
String[] Emp_ID;
String Sensor_ID;
1
1
N
Crop
String Crop_ID;
String Crop_Name;
Boolean Harvested;
Float Yeild_Per_Area;
String FarmerID;

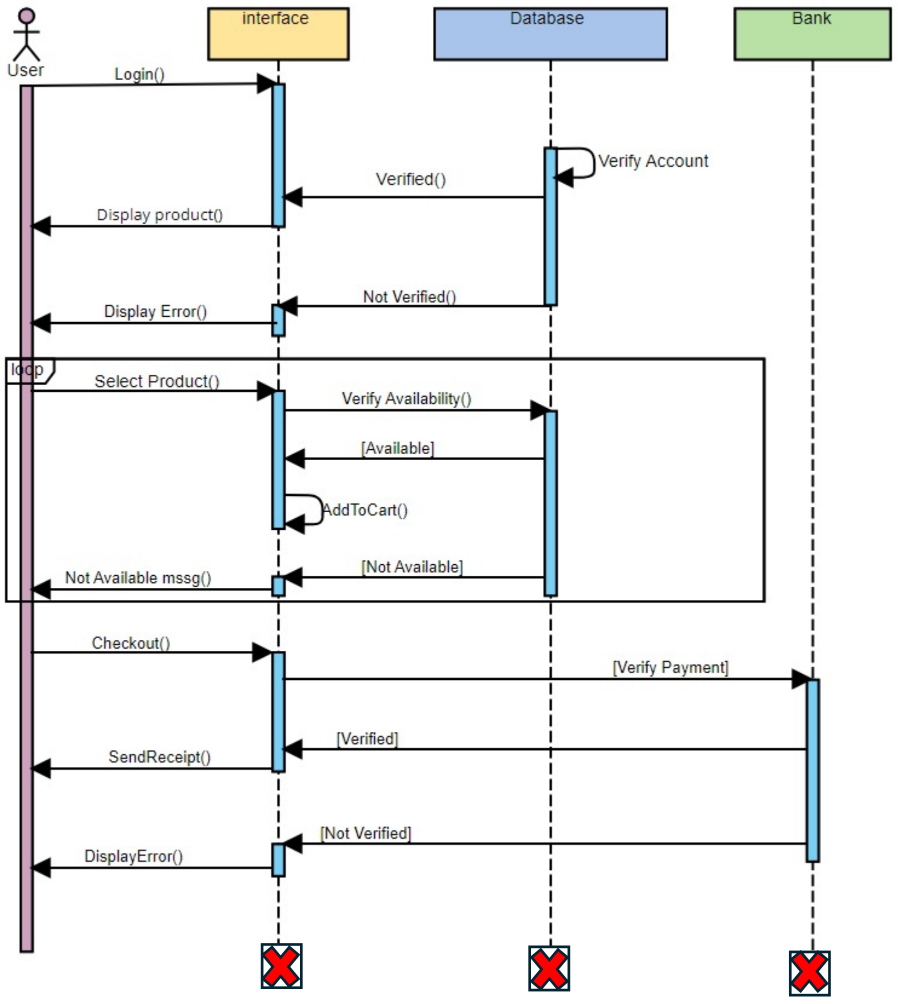
Activity Diagram

When the user opens our app to purchase a product, the system requests the customer to log in, then the customer will browse for crops, selects all the crops they wish to purchase and proceed to the transaction. The system will check the total price of all the crops added to the cart. If the total price is less than or equal to 500 AED, the user can choose a preferred payment method. After the confirmation, the system will accept the order. We will deliver the order to the customer, after the order has been delivered and received by the customer, the system will update the crop and purchase details. On the other hand, if the total cost of the purchase is greater than 500 AED, the customer's only option is to make an online payment, after the data is entered the system will verify the payment. If its accepted, it will go through the same process mentioned earlier, otherwise the system will reject the request.



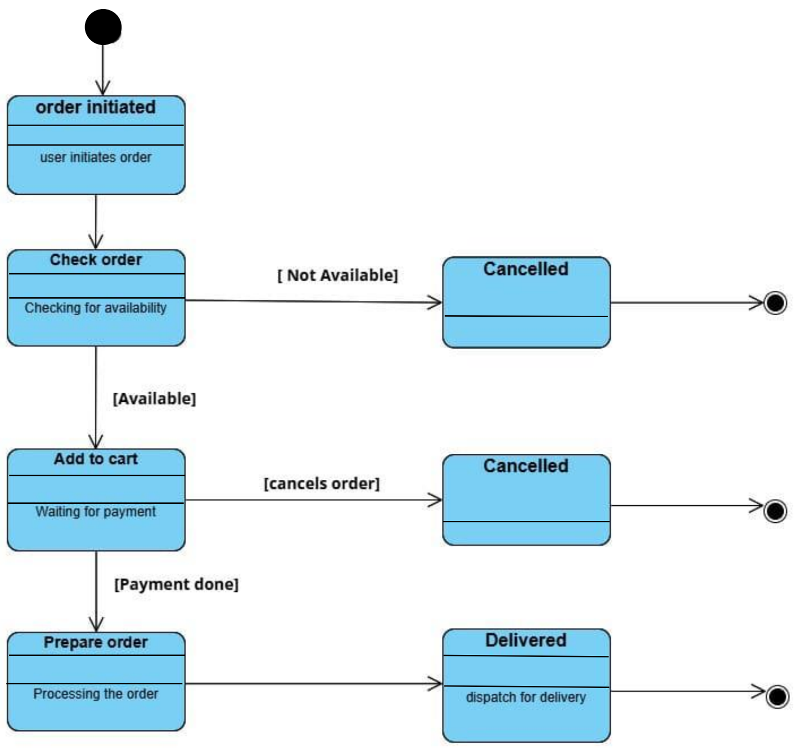
Sequence Diagram

The user starts by logging-in to the system, the system is going to identify with the database to verify the username and password. If the account is not verified, an error message is going to be displayed, and the customer can re-enter the account details. If the account is verified the products menu is displayed and the customer can select the needed product. After selecting the product, the system is going to verify if the chosen quantity is available or not. If available, it is added to the cart. If it is not available, a message noting unavailability is going to be displayed and the item won't be added to the cart. This procedure is going to be repeated with all the chosen products until the user chooses to proceed to check-out. When the customer proceeds to check out and pay, the system is going to verify the payment with the bank, if the payment is verified the receipt is going to be sent to the user. If it is not verified, an error message is going to be sent.



State Diagram

This is a state diagram of an order made by a customer. After logging into the system, the customer will initiate the order. Once it is initiated, the order is checked by the admin for availability. If the contents of the order are not available, the order will be automatically cancelled. If everything is available, it will be added to the cart waiting (on hold) for the customer to finish with the payment procedure. In case the customer does not pay on time or decides to cancel the order, the order will be cancelled. After payment is done (processed and accepted), the order will be prepared (collected and packaged) and delivered to the customer by the delivery team.



System Design Modeling

Let’s describe our system in terms of the Four design specifications:

I) Data/Design Class Realization

For design class realization we convert the class diagram into a design class realization by describing the classes in terms of data structures.

The entity Admin can be described as a class with 4 attributes AdminId, Password, name and email. All of them can be represented as a string. The relationship between the admin and farming specialist can be represented as a List of farming_specialist objects. The relationship between the admin and the users can be represented as a List of User objects. The above two relationships may result in 2 methods, managing_farming_specialist() and managing_users().

The entity User can be described as a class with 5 attributes ID, Password, Name, Address and Email. All of them will be represented as a string. It also has AdminId which is a string and a result of the relationship between User and Admin. The class User is a superclass and has 3 subclasses: Regular_User, Volunteer, and Farmer.

The entity Reguler_User can be described as a sub class inheriting all the attributes from User. The attributes of Regular User are Username, Status, RegistDate, PhoneNum, OrderHistory. Username, Status can be represented as a string. OrderHistory can be represented as a list. RegistDate can be represented as a Date type. PhoneNum can be represented as an Integer.

The entity Farmer can be described as a sub class inheriting all the attributes from User. The attributes of Farmer are FarmerId, CropId, EmpId, SensorId. FarmerId can be represented as a string. The farmer has one to many relationship with the Volunteer, Sensor and Crops. These relationships can be represented as CropId, EmpId, SensorId, which are all List of Strings.

The entity Volunteer can be described as a sub class inheriting all the attributes from User. The attributes of Volunteer are JobName, HoursWorked, FarmerID. JobName can be represented as a list of strings, HoursWorked is an integer. FarmerID can be represented a list of Strings.

The entity Sensor can be described as a class. The attributes of Sensor are SensorId (String), Type (String) , Address (String), Data (Dictionary with keys representing the type of data and the values representing the value of the data, for example temperature (key) and 39.5 C (Value)), NumberOfSprinklers (Integer), ON (Boolean), Battery (Integer), FarmerID (String).

The entity Crop can be described as a class. The attributes of Crop are CropId (String), Crop_Name (String), Harvested (Boolean), Yield_Per_Area (Float), FarmerID (String).

The entity Farming_Specialist can be described as a class. The attributes of Farming_Specialist are ID (String), Password (String), Name (String), Address (String), Specialization (String), Email (String).

II) Architectural Design - Systems Architecture

Layered Architecture

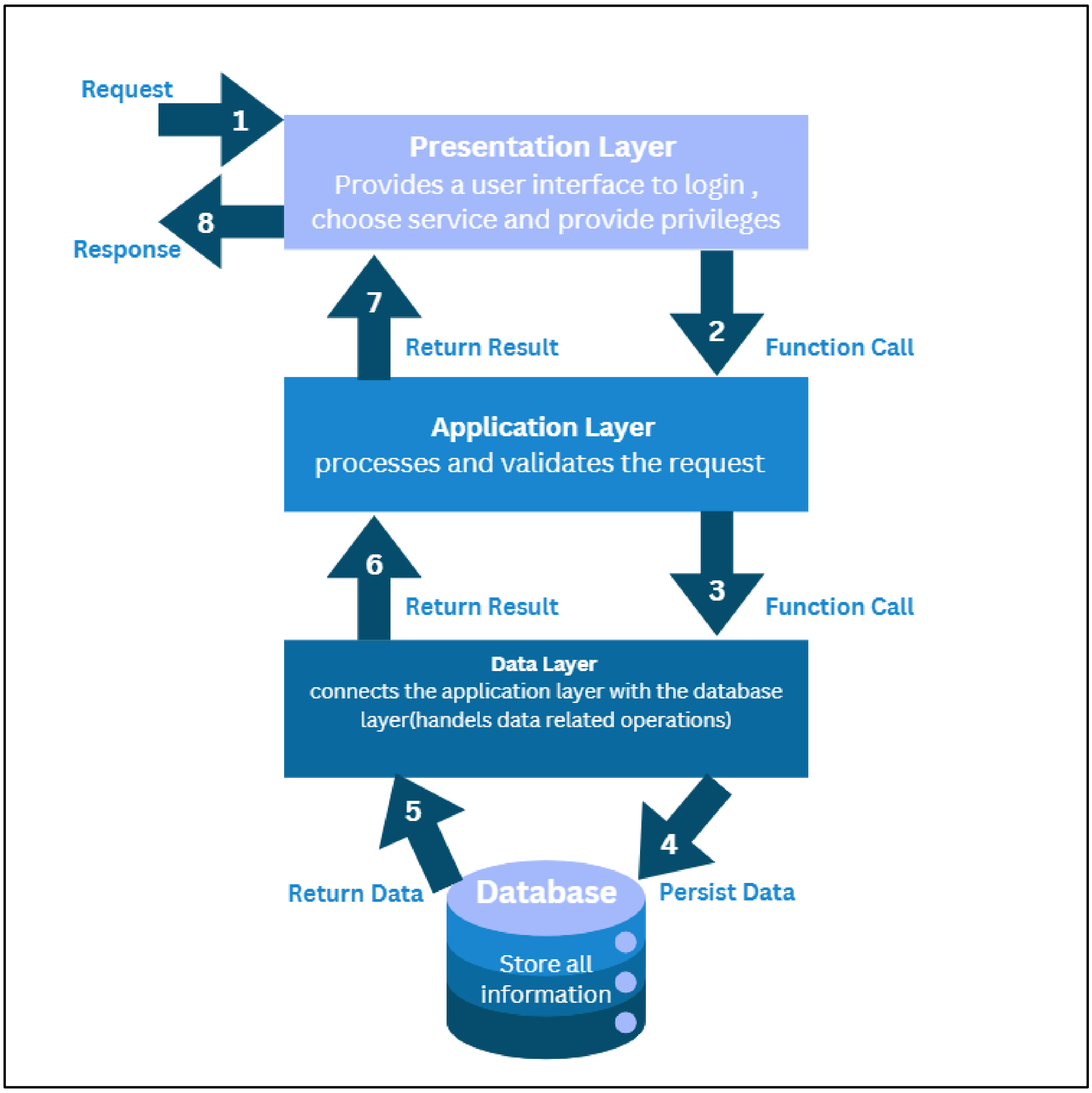
In layered architecture, we have multiple layers. Each layer is designed to accomplish different tasks. The flow of layers is from high level language to lower-level language (machine/coding).

1. First Layer – Presentation layer:
    - This layer provides interaction between the user and the software
    - Based on the user, the interface is refined.
    - For example, when a user logs in, the user will be provided with a menu with different services based on their identification. (regular/farmer/volunteer)
2. Second Layer – Application Layer
    - This layer processes and validates the request made by the users in the first layer.
    - Let's say that a user has requested to purchase a crop, now all the checking if the crop is available, validating the payment, etc. is done in this layer.
3. Third layer – Data Layer
    - This layer is responsible for connecting the application with the database for any required information. It makes sure that the data is stored in a repository that is durable and can exist for a long time.

- For example, to check if a crop is available, it will fetch the data from the database.

4. Fourth Layer – Database
   - This layer is responsible for storing information related to the system.

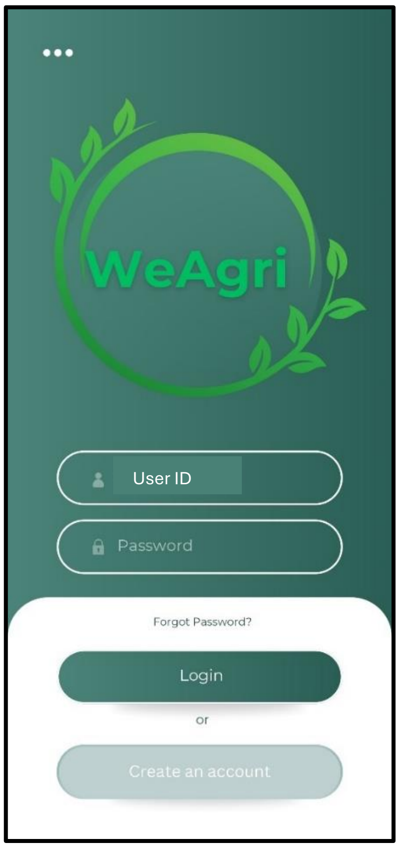
Here is a diagram of our system's architecture and the flow of a process.



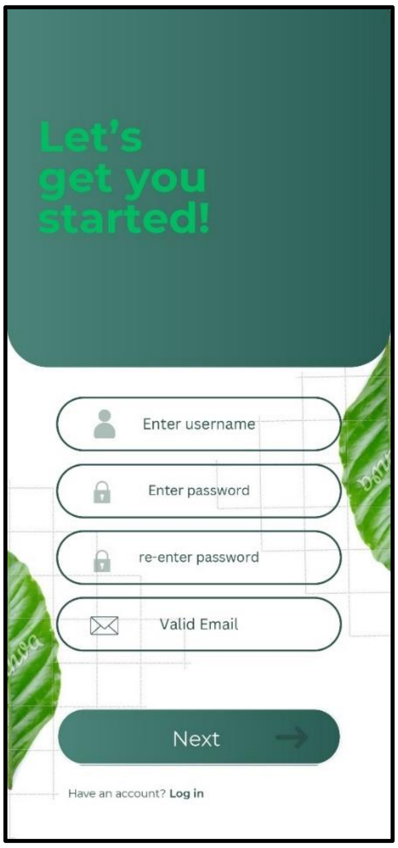
Request
1
Presentation Layer
Provides a user interface to login , choose service and provide privileges
8
Response
7
Return Result
2
Function Call
Application Layer
processes and validates the request
6
Return Result
3
Function Call
Data Layer
connects the application layer with the database layer(handels data related operations)
5
Return Data
4
Persist Data
Database
Store all information

III) The Interface Design

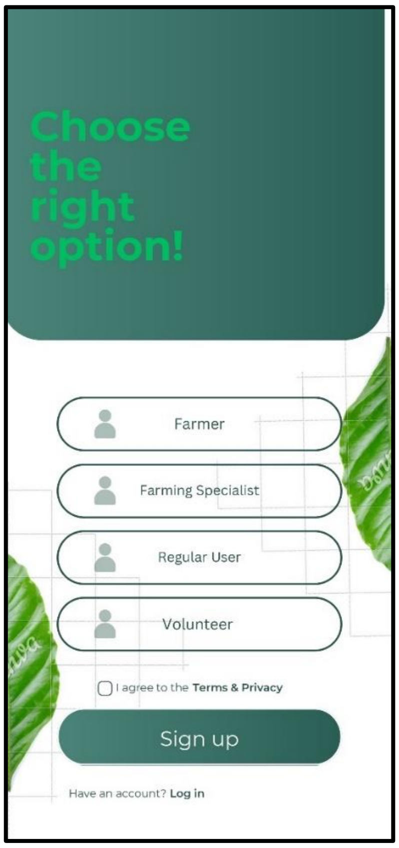
Here is a touch and feel of the interface of our software.



(1)



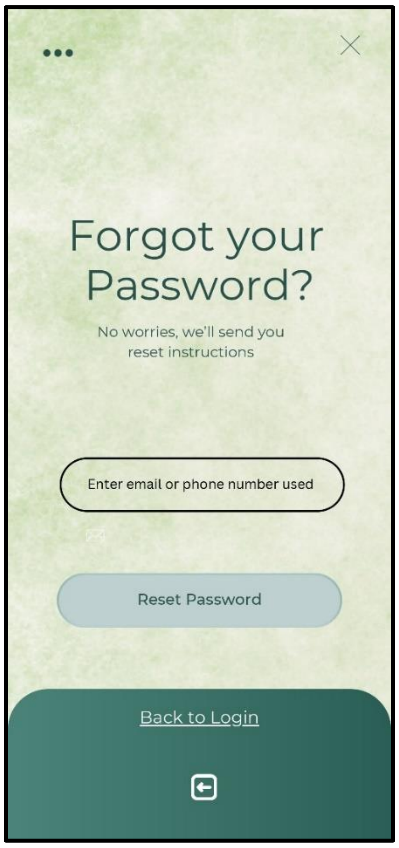
(2)



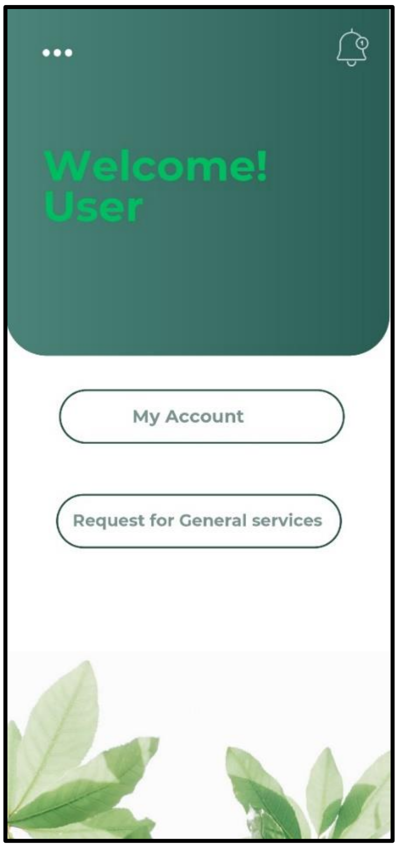
(3)

The image (1) shows the first interface that appears when we open our app. We are asked to sign in using user ID and password. When we press Login, the following interfaces according to our ID follow up: Interfaces (5) to (8).

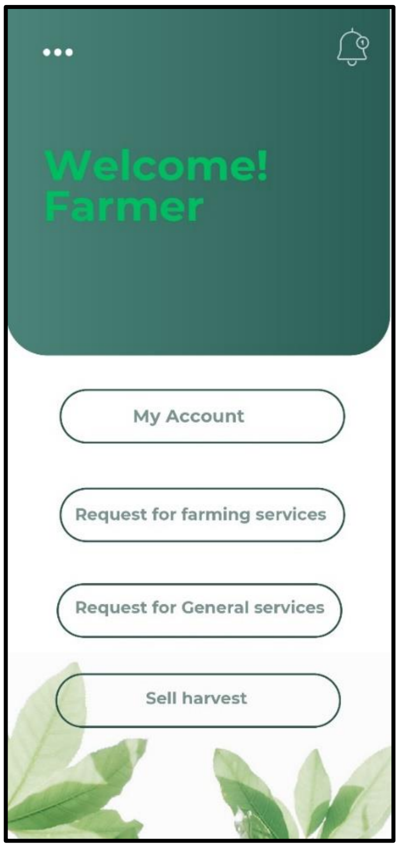
In case we are a new user, we can press on Create an account and the following interface (image (2)) should be visible. After creating our username and password the image (3) should follow up. Here we have the option to select our role.


Forgot your Password?
No worries, we'll send you reset instructions
Enter email or phone number used
Reset Password
Back to Login

(4)

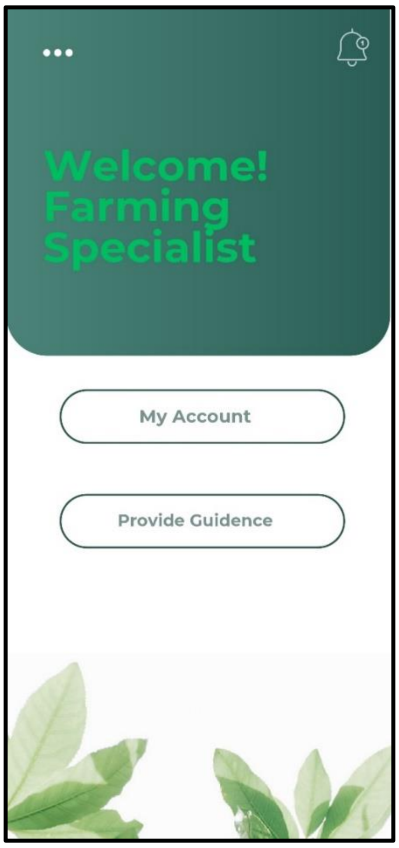


(5)

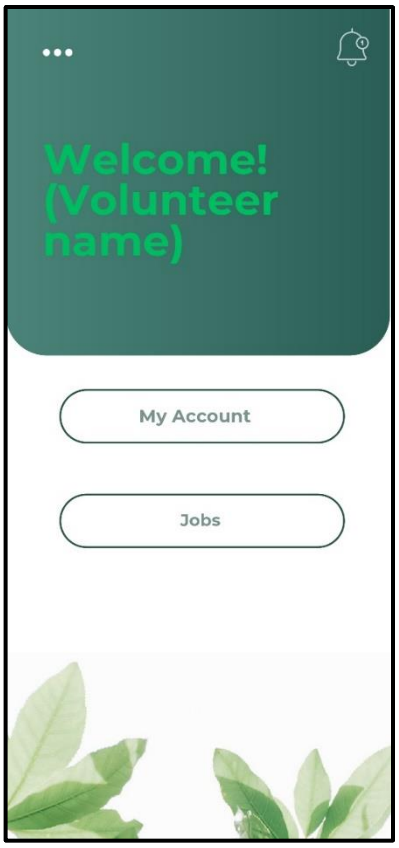


(6)

The interface (image (4)) is displayed in case the user has forgotten his/her password, so they have the option to rest the password.

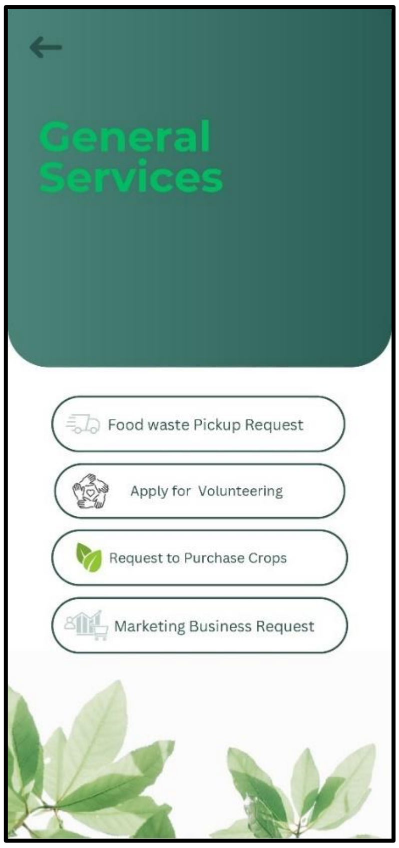


(7)

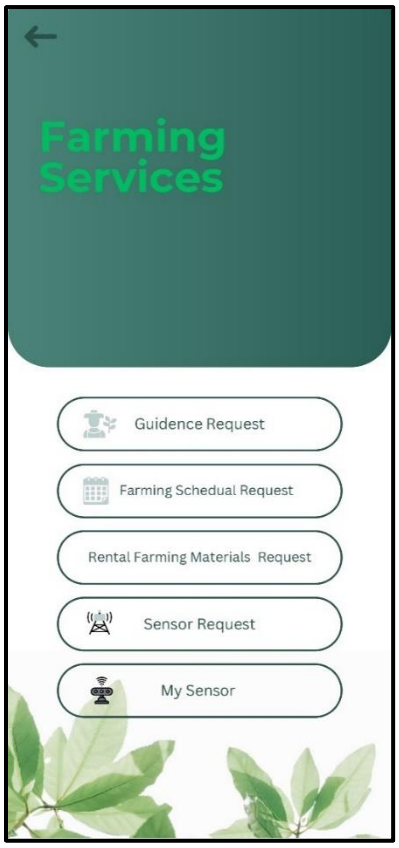


(8)

The interfaces (image (5) to (8)) are displayed according to the type of users.


General Services
Food waste Pickup Request
Apply for Volunteering
Request to Purchase Crops
Marketing Business Request

(9)

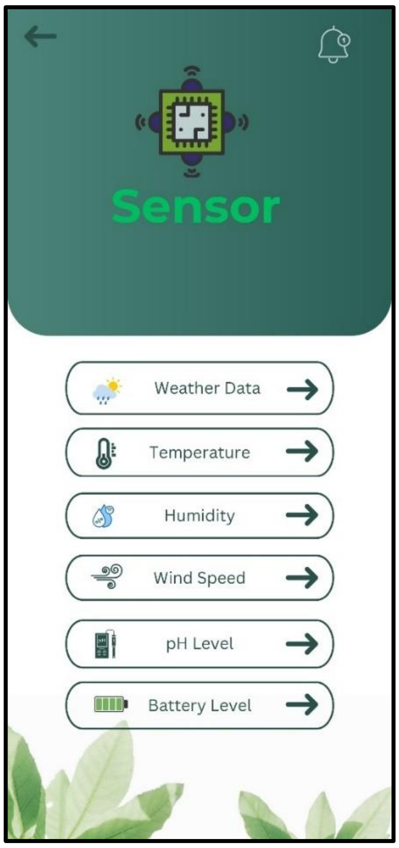


(10)

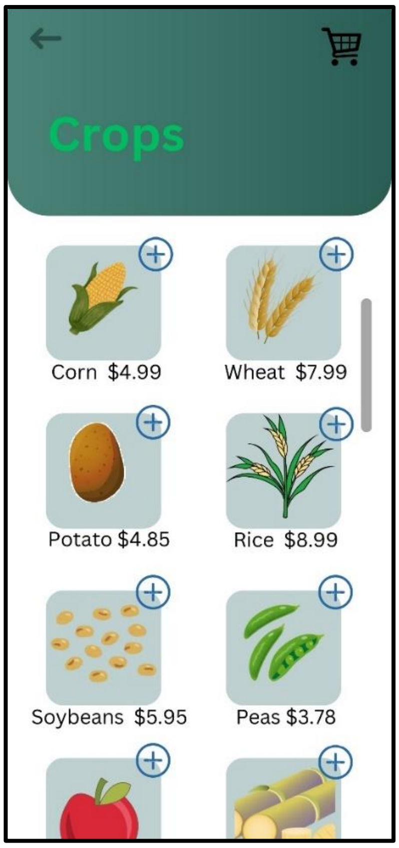
The interfaces (image (9) and (10)) show how the options for different services are displayed for general and farming services.

The interface (image (11)) shows how our sensor dashboard is shown to the farmers. They have the option to view different types of weather data.

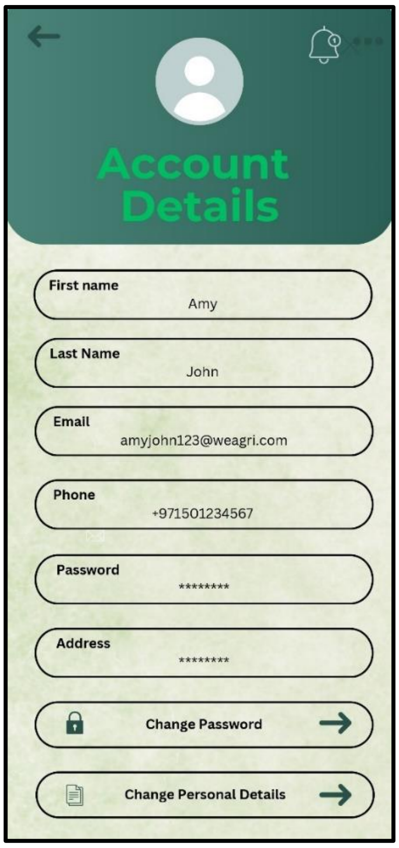
The interface (image (12)) is displayed when we click on the 'Request to Purchase Crops' service in general services. Different types of crops with their prices available are displayed. We can add a particular crop to our cart by clicking/pressing on the + symbol. For example, if we click on + symbol on Rice, 1Kg of Rice will be added to the cart.



(11)



(12)


Account Details

First name
Amy

Last Name
John

Email
amyjohn123@weagri.com

Phone
+971501234567

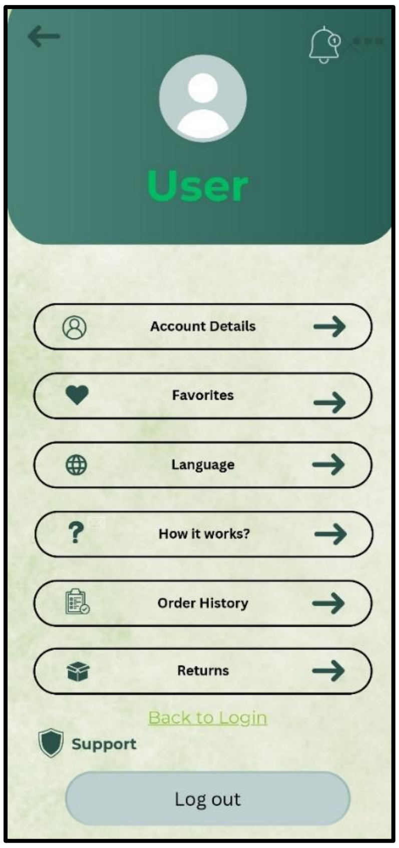
Password
****

Address
****

Change Password

Change Personal Details

(13)


User

Account Details

Favorites

Language

How it works?

Order History

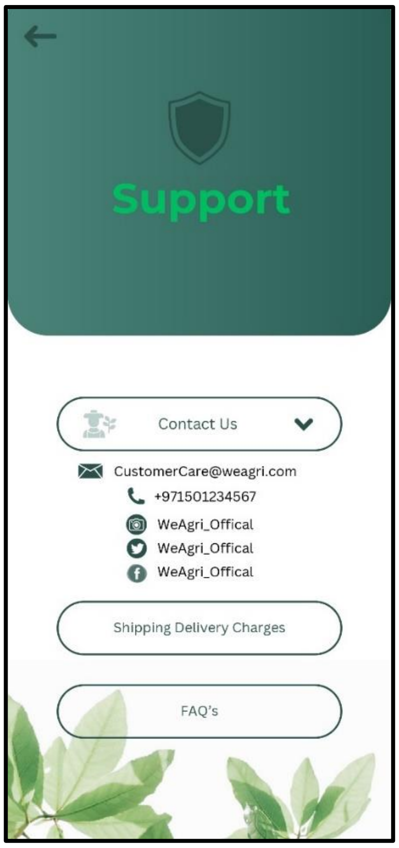
Returns

Back to Login

Support

Log out

(14)


Support

Contact Us

CustomerCare@weagri.com

+971501234567

WeAgri_Offical

WeAgri_Offical

WeAgri_Offical

Shipping Delivery Charges

FAQ's

(15)

The interface (image (13) and (14)) displays all the information related to that user, like their name, password, etc. It gives them the option to change their password or any password details. They have the option to view their order history, language they want to use the app in, etc.

The interface (image (15)) is shown if a user wants to know or contact our systems officials. The information about Email ID, Phone number and other social media handles are displayed in Support. If anyone wants to know the delivery charges or have any questions can check out the support page.

IV) Component-level Design

We have divided our system into the following components.

We have 3 main components Front-End Component, Back-End and the Data-Base Component.

The Front-End component has the following components in it:

- Admin Management: Manages all the interactions related to Admin and helps manage the system. This component uses functionalities offered by the Service/Request Management component.
- Farmer Management: Manages all the interactions related to Farmer and helps manage farmers accounts. Manage the services provided to farmers. This component uses functionalities offered by the Sensor Dashboard (Since Farmers use sensors to track and observe their environment) and Crop Management component (Farmers need to keep track of the crops they are harvesting).
- Sensor Dashboard: Provides functionalities that help display and manages interactions of farmers to the dashboard. This component uses functionalities offered by the sensor management component.
- User Management: Manages all the interactions related to Users (volunteers included). Manages the services provided to the Users. This component uses functionalities offered by the Order Component.

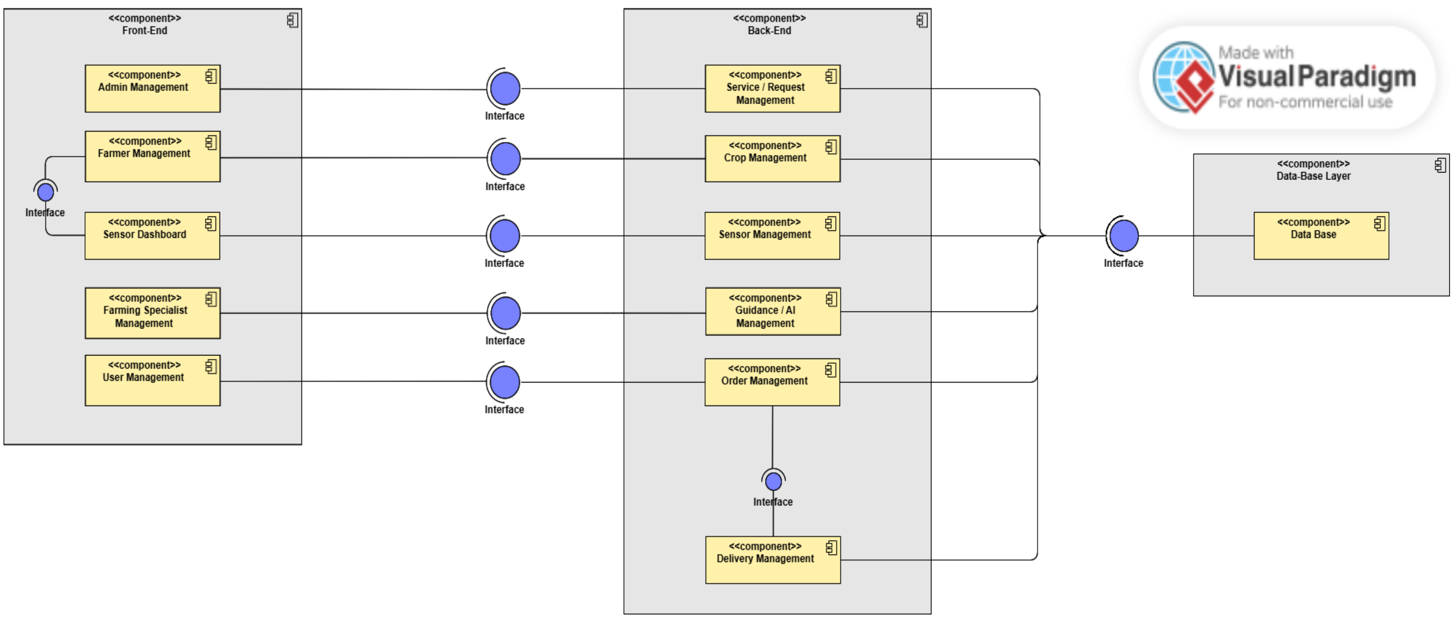
The Back End Component:

- Service/Request Management: Provides all the functionalities needed to manage different types of services for different types of users. This component is required by Admin since, admins manage requests by approving or denying it.
- Crop Management: Provides all the functionalities needed to manage information about different types of crops harvested or still in progress by farmers. This component is required by Farmer management since farmers need to store information about their crops.
- Sensor Management: Provides all the functionalities needed to manage sensors. Functionalities that help determine the data like PH value, Humidity, etc. This component is required by sensor dashboard.

- Guidance/AI Management: Provides all the functionalities needed to manage the AI tool that supports farmers by providing information. This component is required by Farming Specialist Management.
- Order Management: Provides all the functions needed to manage the order made by the Users by saving their orders by managing the inventory of the crops. This component is required by User Management
- Delivery Management: Provides all the functions needed to manage the deliveries of the orders made by the users. This component is required by Order Management

Data-Base Layer:

- Data Base: Stores information needed by all the components to function properly. It is accessed by all the components of the back end.

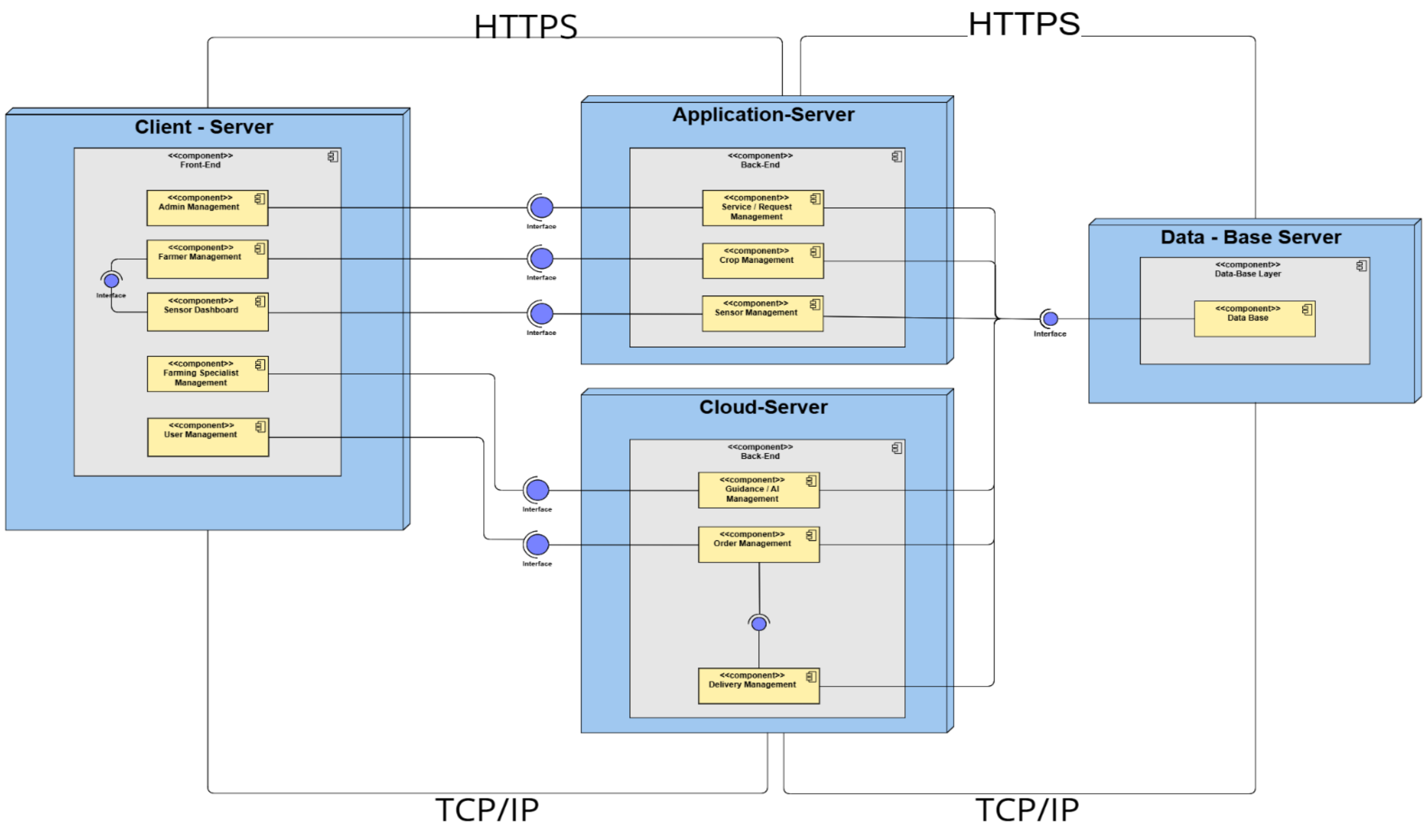


<<component>> Front-End
<<component>> Admin Management
<<component>> Farmer Management
<<component>> Sensor Dashboard
<<component>> Farming Specialist Management
<<component>> User Management
Interface
<<component>> Back-End
<<component>> Service / Request Management
<<component>> Crop Management
<<component>> Sensor Management
<<component>> Guidance / AI Management
<<component>> Order Management
<<component>> Delivery Management
<<component>> Data-Base Layer
<<component>> Data Base
Made with Visual Paradigm For non-commercial use

The Deployment Diagram of our System

Our software requires the following hardwares to function:

- Client-Server Node:
  This server is hosted on the clients end by providing the clients with the interface of our software. It contains all the components of the Front-End Component. It is connected to the application server and the cloud-server by HTTPS and TCP/IP connection.
- Application-Server Node: This server is required by the back - end of the application and it is connected to the Data-Base Server via the HTTPS connection. It contains the Service/Request Management, Crop Management and the Sensor management components.
- Cloud-Server Node: This server contains components that need to be hosted on Cloud for faster and more secure connection. It contains the Guidance/AI management, Order Management and Delivery Management Component. It is connected to the Data-Base Server via the TCP/IP connection.
- Data-Base Server Node: This node stores the database component; it stores all the data required by the back-end components.


HTTPS
HTTPS
Client - Server
<<component>> Front-End
<<component>> Admin Management
<<component>> Farmer Management
Interface
<<component>> Sensor Dashboard
<<component>> Farming Specialist Management
<<component>> User Management
Application-Server
<<component>> Back-End
<<component>> Service / Request Management
<<component>> Crop Management
<<component>> Sensor Management
Interface
Interface
Interface
Data - Base Server
<<component>> Data-Base Layer
<<component>> Data Base
Interface
Cloud-Server
<<component>> Back-End
<<component>> Guidance / AI Management
<<component>> Order Management
<<component>> Delivery Management
Interface
Interface
TCP/IP
TCP/IP

Architectural Design Tasks

Task 1: Representing the System in Context – Architectural Context Diagram (ACD)

The above Architectural Context Diagram represents our system in context, let's describe it.

The Actors in our diagram are:

- User (includes farmer + regular user + volunteer)
- Farming Specialist

These Actors use our system to achieve their goals.

The Superordinate Entity is:

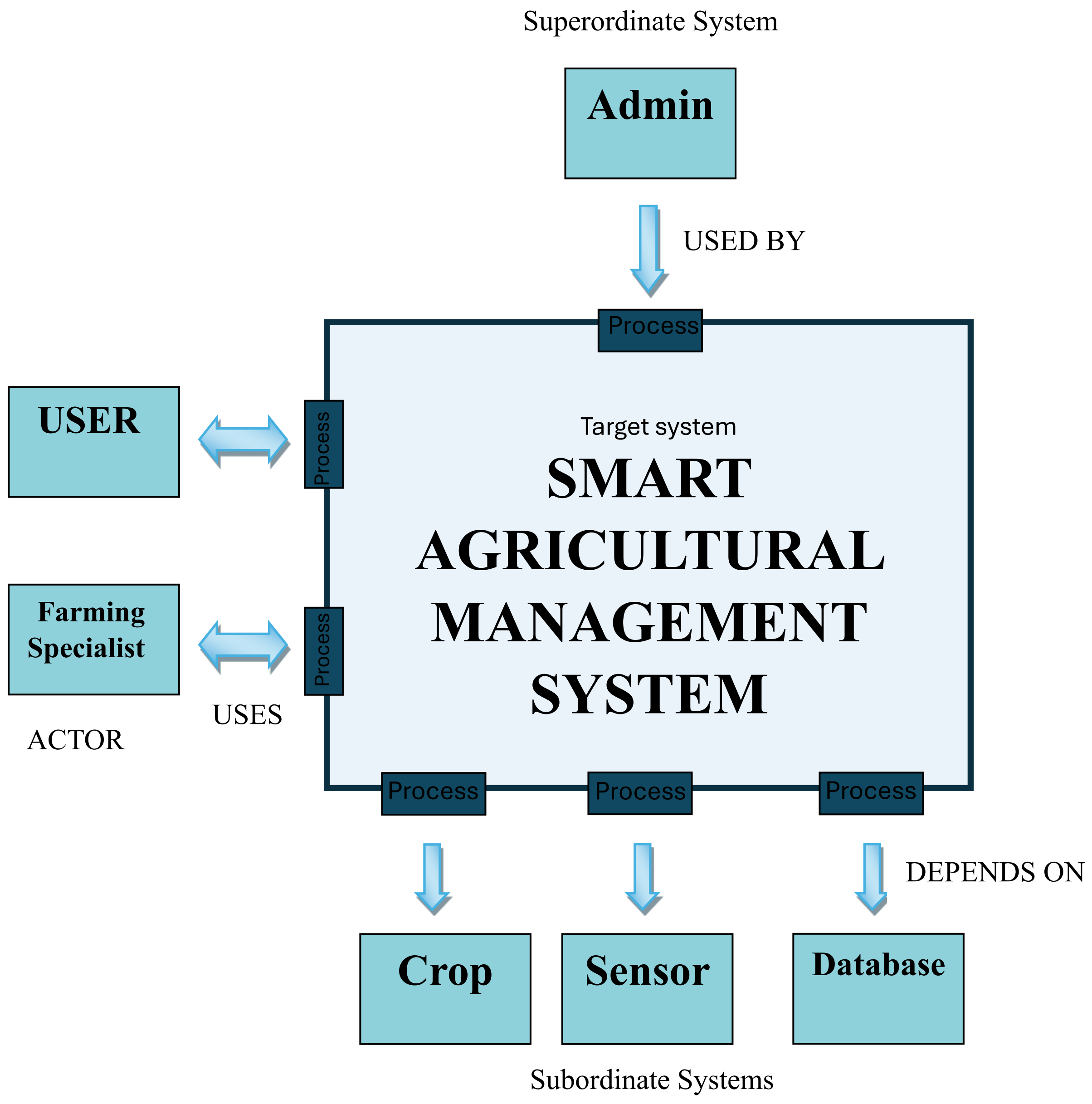
- Admin

Admin can be considered as an actor as well, in fact it is an actor, but the reason we consider it as a superordinate system is because it manages and controls the functioning of our system. Which is why the functioning of an admin is used by our system.

The Subordinate Systems are:

- Crop
- Sensor
- Database

Our System depends on the subordinate systems to process are require data. Sensor processes and provides weather data. Data-Base stores all the data required by our system. Crop is a special system; in fact, our whole system depends on crop information. Since it is very important, we specifically mentioned even though database also stores information about crops.



Superordinate System
Admin
USED BY
Process
USER
Process
Target system
SMART AGRICULTURAL MANAGEMENT SYSTEM
Farming Specialist
Process
USES
ACTOR
Process
Process
Process
DEPENDS ON
Crop
Sensor
Database
Subordinate Systems

Task 2: Defining Archetype

Looking at our software, here are some of the archetypes that we have defined that determine our system's behavior and can be reused to build similar types of software.

- System_Manager: This role is assigned to the Admin. As the name suggests, this role manages the system by monitoring its behavior and requests made by the users. Also managing and maintaining information about different kinds of users.
- Purchaser: This role can be assigned to any kind of user (volunteer, farmer, reguler_user), as they request/purchase for any kinds of available service.
- Helper: This role can be assigned to the farming specialist as they assist the farmers regarding any concern. It can even be assigned to volunteers to help farmers with any tasks related to farming.

Task 3: Refining the architecture into components

This task has already been described earlier when we created the component diagram (page 29-30).

Task 4: Describing instantiations of the system

Just like how we create instances which are objects of a class, the same way we define instances of our architecture. The deployment diagram created and described earlier (page 31) is one way of understanding the instantiation of our architecture, as the deployment diagram converts the logical details of our system into physical details and describes to us how our system can be implemented in a particular environment.

The Design Choices of our System

While designing a software, it is important define some design choices which includes Quality guidelines, design concepts, etc. that guide and shape our design modelling process. Here are few design modelling principles and design concepts that have been given extra attention while designing our software. Many of the design concepts also align with the quality guidelines that guides and highlights the quality of our software.

- Design Modelling Concepts and Principles

1. Modularity

   Modularity is the concept of dividing the software into individual components that are each designed to perform a specific function, this makes the functioning of the software simpler and easier and helps find us any fault in the software easily. By dividing our software into different components like crop management, farmer management, sensor management etc, we can ensure that even if one component is malfunctioning it doesn't affect the functioning of the other components. This concept also helps us achieve the 7th principle of the Design Modelling that states that components should be loosely coupled.

2. Functional Independence

   Functional Independence is a design concept that focuses on making sure that each component is defined to perform a single task, and no component need to depend on other components. Without Functional independence, if one component is modified then the changes may be reflected in the other components as well which could give us unwanted results. For instance, if we make any changes to the sensor dashboard, it would be unacceptable to see the crop management functioning differently. This concept perfectly aligns with the 8th principle of the Design Modelling that states that the Component-level design should be functionally independent.

3. Interfaces

   Principle number 5 and 6 states that the interfaces should be designed with care and the user interface design should be according to the needs of the user. Our software's interface is designed in a way that makes it easy for the user to navigate, which also aligns with one of the 8 quality guidelines. Since our software will be used mostly by farmers, its interface has been designed with different types of icon that can help understand its meaning, and farmers can easily use it. Even the sensors dashboard is designed with large icons and readings. Our interfaces can be easily maintained and changed if needed in the future.

4. Separation of Concerns

   Our system is divided into separate well-defined components that each focus on a specific task. This makes the system more flexible and easier to maintain. This concept is quite like modularity and functional independence. In fact, all 3 of these design concepts go hand in hand.

Here are few of the 5 architectural considerations that have been given extra attention while building the architecture of our system.

1. Economy

   This consideration makes sure that the design of the architecture should be simple and not complicated. Our software doesn't include any unnecessary features, it only includes those features that are required by our stakeholders. So, we made sure to not go outside the scope of our system when designing its architecture. As unnecessary feature adds unnecessary complexity, which might make it difficult for users to understand and use, so we made sure to avoid adding extra not so useful details. Each feature or component in our software has a purpose.

2. Visibility

   By documenting each architectural decisions clearly, it makes it easier for any developers new to our system to easily grasp the idea and the reason for such decisions made. Our system's architecture decisions should be transparent and obvious to the developers and stakeholders. developers and stakeholders should understand the decisions as to why for example the Farmer Management component was added, or why was layered architecture chosen, etc.

3. Symmetry

   Our System's architecture is well balanced and consistent. Consistency in our system helps keeping the system simple and easy to navigate. The consistency is provided to elevate our user's experience in using our software. Our software's interface specifically is consistent in keeping up with needs of the user. Consistency also signifies that we have made sure to focus on keeping the quality of our top-notch.

As mentioned before, our priority to maintain the quality of our system is very high, therefore, while designing the system's architecture and design models, all the 8 quality guidelines have been kept in mind. Some of these quality guidelines have been given extra importance. Our System's architecture has been built, with components that are readable and understandable for the constructors of our system. They include all the requirements explicitly mentioned the stakeholders such as, our system should be able to help admin in managing requests, help any user in requesting for the mentioned services, etc. In fact, the components have been designed in a way that they can be reused for any other similar systems as well specially the sensor management component. We have implemented the layered architecture to build our system. The layered architecture is one of the most common architectural styles. Each class or a component in our systems has been constructed using well defined data structures. The data, architecture, interfaces and the component have been distinctly represented.

Construction: Validation, Verification and Testing

Black Box Testing

We performed Black Box Testing on our software. We implemented the Black-Box testing technique by defining test cases that focus on analyzing how our software behaves when we enter incorrect or correct data. We performed Equivalence partitioning by defining valid and invalid values for the attribute Yield_Per_Area for the class crop.

Invalid	Valid	Invalid
< 5	>=5 and <= 1000	>1000

Our software should be able to differentiate between the valid and invalid values and respond accordingly. The test cases for it are defined in T12, T13 and T14. Here are all the test cases we have defined to test the functioning of our software.

Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass /Fail
T1	Check admin/user/farming specialist login with valid id and password	Go to the app and login with valid ID and Password.	ID: Admin101 Password: 8A87n5	Able to Log In	As Expected	Pass
T2	Check admin/user/farming specialist login with invalid id and password	Go to the app and login with ID and Invalid Password.	ID: Admin101 Password: 8A87N5	Not Able to Log In	As Expected	Pass
T3	Check if each service is accessible to the required users and is functioning and displayed properly and it is easy to navigate	Go to the app and login with valid ID and Password. Then, Navigate the service menu.	ID: User101 Password: 87dsF9	Able to navigate the service menu	As Expected	Pass
T4	Check if the inventory of crops is being updated	Go to the app and login with valid Farmer ID and password and select the 'Sell Harvest' option and add the crop details and price. Go back and select	ID: User101 Password: 87dsF9 *Adding Crop Information* Crop_ID: Rice569 Crop_Name: Rice Quantity: 20Kg Price: 30 Aed/Kg	Able to See and purchase any new crops added into the list of crops	As Expected	Pass

		on purchase a crop service.				
T6	Check if user purchases are functioning properly	Go to the app and login with valid user ID and password and select the request to purchase a crop and add a crop to the cart and proceed with the payment and add card details.	ID: User101 Password: 87dsF9 *Enter Payment Details* Card Number: 1234 5678 9123 4567 Card Expiry Date: 02/2028 CVV: ***	Able to Purchase a crop without any difficulty or error	As Expected	Pass
T7	Check if valid messages are displayed if Admin rejects a request	Go to the app and login with valid admin ID and password. Then reject a service request made by a user with ID: User101. Log in to the app with a user ID and password who tried to request for that service and check if you received any messages regarding that service.	*Admin Log In to reject service* ID: Admin101 Password: 8A87n5 *Users log in to account and requests for a certain service* ID: User101 Password: 87dsF9	User is not able to proceed with a request if admin rejects it and user should be able to receive a message stating that the request has been rejected.	As Expected	Pass
T8	Ensure that data is stored without any modification in the database	Go to App and Create an account with valid User Information. (Admin should go to Database and check the new entry and make sure it's the same)	*User Creates an Account* ID: User108 Pwd: 9J5D61 Name: Jacob Address: ******* PhoneNo: +971568492432	Data is stored as it is entered	As Expected	Pass
T9	Check if the sensor dashboard is displayed properly to the farmers	Go to the App and login with valid Farmer ID and password. Go the farming services option and go to my sensor and analyze the dashboard	ID: Farmer101 Password: 87ds55	Farmers should be able to view the sensor dashboard if they have purchased the sensor service	As Expected	Pass

T10	Ensure each user of the software has the least privilege required to accomplish a task (Security)	Log In with a user ID and password and just navigate the app	ID: User108 Pwd: 9J5D61	Users should not be able to view other users' data or change any information.	As Expected	Pass
T12	Check if valid error messages are displayed if a farmer enters a float value < 5 for yield_per_area	Log in with a Farmer ID and password and enter crop details and make sure to enter Yield_Per_Area as -5	ID: Farmer101 Password: 87ds55 *Entering Crop Details like name, id etc.* Yield_Per_Area = -5	The system should not accept -5 as Yield_Per_Area since it is an invalid value as it is less than 5.	As Expected	Pass
T13	Check if it is accepted when Farmer enters a float value >= 5 and <= 1000 for yield_per_area	Log in with a Farmer ID and password and enter crop details and make sure to enter Yield_Per_Area as 100	ID: Farmer101 Password: 87ds55 *Entering Crop Details like name, id etc.* Yield_Per_Area = 100	The system should be able to accept 100 as Yield_Per_Area since it is a valid value as it is between 5 and 1000.	As Expected	Pass
T14	Check if valid error messages are displayed if Farmer enters a float value > 1000 for yield_per_area	Log in with a Farmer ID and password and enter crop details and make sure to enter Yield_Per_Area as 1050	ID: Farmer101 Password: 87ds55 *Entering Crop Details like name, id etc.* Yield_Per_Area = 1050	The system should not accept 1050 as Yield_Per_Area since it is an invalid value as it is greater than 1050.	As Expected	Pass

Thankfully, we have passed all the test cases, which tell us that our software is performing remarkably well.

Analyzing the Strength and Weaknesses of Our System

Strengths	Weaknesses
The software ensures separate privileges for different types of users (least privilege)	The AI chatbot is less efficient and informative
The software is easy to navigate and use	The software cannot help customers to track their orders
The software meets all the needs of the stakeholders	The software needs to be connected to Wi-Fi to be able to use it
The software is easy to maintain	The software might face difficulty in handling large number of users

Future Recommendations

- The AI chat-bot can be improved a lot and updated regularly to be more informative and useful.
- The software can be further improved by implementing predictive AI models that can help farmers by predicting the sales they will make each month by selling their harvest.
- The software can implement functionalities that can help customers who have purchased some items to track their order.
- Large amounts of data storage can help the software to handle large numbers of users worldwide.

Conclusion

Finally, the story of our software is coming to an end. From building the idea of our software to analyzing the strengths and weaknesses of our software, full attention has been given to each step.

Our software, even though it may seem complicated, has made it simple to understand. Quality has been put to mind, from the moment of choosing agile as the software developmental model, as agile gives extra care and attention to risks. In the requirements modeling phase, all the requirements have been clearly defined. All the UML diagrams we have created help us understand our software in detail.

In the design modeling stage, we have described our system in extra detail, by describing its architecture and its component and deployment diagrams that help us understand how it can be implemented in the real world.

In the testing phase, we chose black box testing to help make sure that our software is functioning as it is made to function and it's not giving us any unwanted results and thankfully all our test cases have passed the test.

Hopefully, our stakeholders such as Dr SAMY and the UN are satisfied with our software and appreciate the effort put into building it.

References

https://sdgs.un.org/goals

https://www.undp.org/sustainable-development-goals

https://www.indeed.com/career-advice/career-development/disadvantages-of-scrum

https://teamhood.com/agile/scrum-advantages-disadvantages/

https://jdi.group/software-development-cost-breakdown/

https://successive.tech/blog/software-development-cost/

https://www.techtarget.com/searchsoftwarequality/definition/agile-software-development

https://www.simplilearn.com/tutorials/agile-scrum-tutorial/what-is-agile

https://intranetssn.github.io/www.ssn.net/twiki/pub/CseIntranet/CseBCS6403/PressmanBook.pdf

Tools used to build our models and interfaces:

https://www.canva.com/

https://www.visual-paradigm.com/

Cited Statistics

UAE-2022. (n.d.). From https://www.foodexport.org/export-insights/market-country-profiles/united-arab-emirates/#:~:text=FAS%20Post%20Dubai%20reports%20that,a%201.2%25%20increase%20from%202020.

UAE-Northeast-2023. (n.d.). From https://www.foodexport.org/export-insights/market-country-profiles/united-arab-emirates/#:~:text=FAS%20Post%20Dubai%20reports%20that,a%201.2%25%20increase%20from%202020.