IT314 Software Engineering

Group 25

Warehouse Management System

Software Requirements Specification

Team Members

Name	Id
Mohammad Tejabwala	202001406
Aniruddh Damor	202001255
Jaimin Rathwa	202001423
Mehul Bhuva	202001437
Aastha Shetty	202001260
Prarthee Desai	202001257
Nisarg Patel	202001436
Arth Detroja	202001274
Rushabh Patel	20200141
Gaurang Ganvit	202001247

Table of Contents

Warehouse management for food and other goods

1. Introduction

The Software Requirements Specification (SRS) introduction overviews the entire SRS, including its purpose, scope, definitions, acronyms, abbreviations, references, and a general overview of the SRS. This document aims to thoroughly analyze and provide a detailed understanding of the **Warehouse Management System** by defining the problem statement. Additionally, the document focuses on the capabilities required by stakeholders and their needs while outlining the high-level features of the product. More detailed requirements for the **Warehouse Management System** are provided later in the document.

1.1 Purpose

This document aims to gather and analyze various ideas for defining the system and its customer requirements. It also aims to predict and organize the product's usage to understand the project better. This document outlines concepts that may be developed later and documents ideas that are being considered but may still need to be implemented.
This SRS document provides a detailed overview of our software product, including its parameters and goals. It describes the target audience, user interface, hardware and software requirements, and how the client, team, and audience perceive the product and its functionality. This document also aids designers and developers in the software delivery lifecycle (SDLC) processes.

1.2 Features of the project solution:

- Authorized access
- Advanced reservation for goods' storage
- Track of storage process
- Tracking storage life of goods
- Stores the necessary details of the warehouse - location, capacity, etc.
- Finds the crop eviction rate for nearby warehouse

1.3 Functional Requirements

1. Warehouse Management System (WMS)

- Store warehouse details, including name, location, storage capacity, type of goods stored, etc.
- Keep track of warehouse storage capacity and availability in real-time
- Update the status of each warehouse in real-time as crops are stored or removed
- Generate reports on warehouse usage over time, including which crops are stored and when.
- Provide a reservation system for farmers to book storage space in advance
- Store information about the types of crops stored in each warehouse and their storage life

2. Farmer Management System (FMS)

- Store farmer details, including name, location, type of crops they grow, etc.
- Keep track of the crops each farmer has stored and their storage life
- Provide recommendations to farmers on which crops to grow based on the storage capacity and availability of nearby warehouses
- Allow farmers to search for nearby warehouses based on their location and the type of crops they need to store
- Provide farmers with information about the eviction rate of each warehouse
- Store information about future reservations made by farmers

3. Integration of WMS and FMS

- Provide farmers with real-time information about the storage capacity and availability of each warehouse

● Allow seamless communication between the WMS and FMS

● Enable farmers to make reservations for storage space directly from the FMS

● Automatically update the status of each warehouse in the WMS as reservations are made and crops are stored

● Store information about the types of crops stored in each warehouse and their storage life

❖ Non-Functional Requirements

1. The system should have a user-friendly interface, with clear and concise instructions, and intuitive navigation.

2. The system should be secure and protect sensitive information, such as farmer and warehouse details, from unauthorized access.

3. The system should be scalable to accommodate future growth and increased usage, including the ability to store and manage more warehouses and farmers.

4. The system should provide fast and responsive real-time updates to farmers about the storage capacity and availability of nearby warehouses.

5. The system should be highly available and reliable, ensuring that farmers have access to the information they need when they need it.

6. The system should be compatible with different platforms and devices, such as laptops, smartphones, and tablets, and should be accessible through a web browser.

7. The system should provide reports on the usage of each warehouse over time, including which crops are stored and when, and the eviction rate of each warehouse.

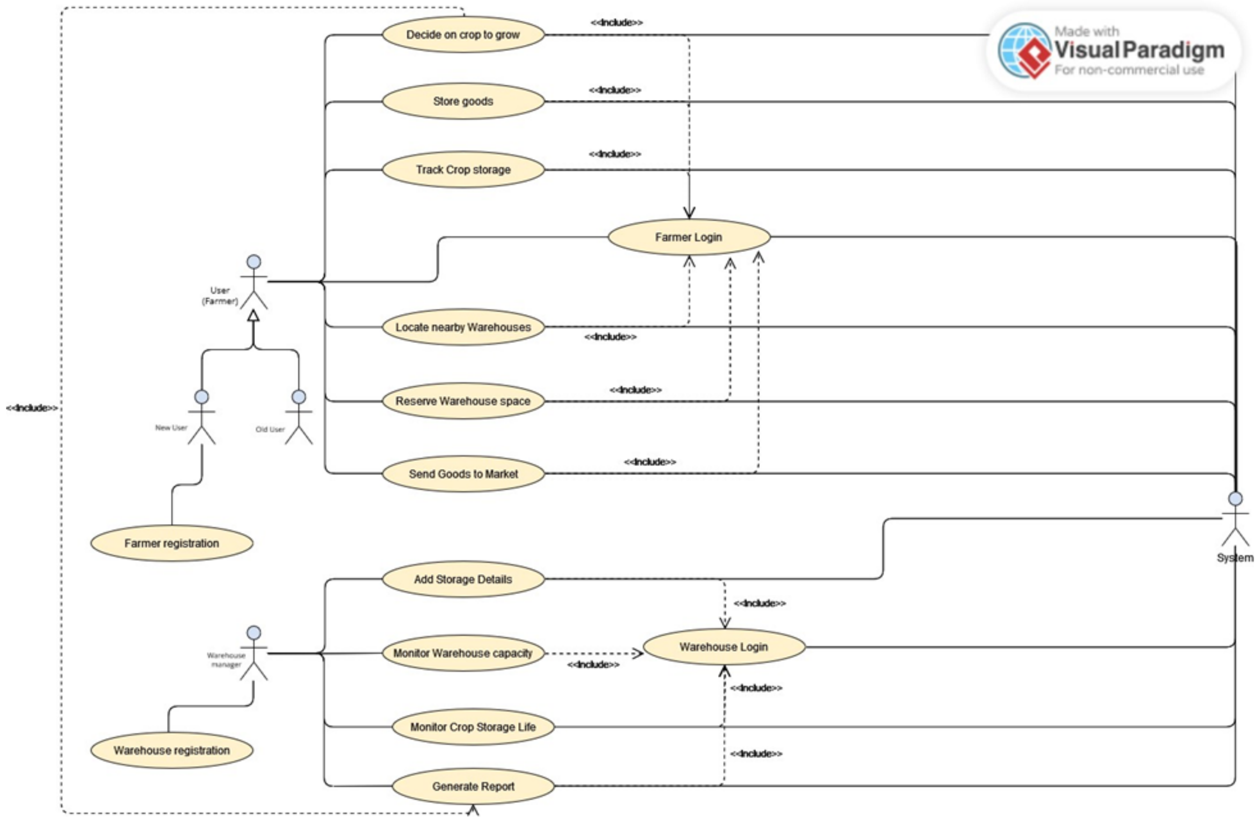
8. The system should have a robust data backup and recovery system to ensure that data is protected in case of a system failure.

9. The system should have error handling mechanisms in place to ensure that errors and exceptions are handled gracefully and do not impact the operation of the system.

❖ Tools

- HTML
- CSS
- Django
- MongoDB
- Javascript

❖ Use Case Diagram



❖ Process Model

Agile software development is a model that emphasizes collaboration, flexibility, and rapid iteration. It's well suited to the problem statement of designing a system for warehouses and farmers, as the requirements may change and evolve as the development progresses. Here's how the agile model can be used for this problem statement:

- Requirements Gathering: The development team will work closely with stakeholders, such as farmers and warehouse managers, to gather requirements and understand the needs of the system. This may involve conducting user interviews, creating prototypes, and conducting user testing.

- Sprint Planning: The development team will break down the requirements into small, manageable tasks and prioritize them based on importance and dependencies. These tasks will be organized into sprints, or short development cycles, that typically last two to four weeks.

- Development and Testing: During each sprint, the development team will work on completing the tasks assigned to them. This may involve writing code, conducting unit tests, and performing integration testing. The development team will also regularly conduct code reviews to ensure the quality of the code and catch any potential issues early on.

- Sprint Review: At the end of each sprint, the development team will present their progress to stakeholders and receive feedback. This is an opportunity for stakeholders to provide feedback and suggest any changes that need to be made.

- Sprint Retrospective: After the sprint review, the development team will reflect on the sprint and identify areas for improvement. This may involve changes to processes, tools, or the development team's composition.

- Repeat: The process will repeat, with the development team using the feedback and insights from the sprint review to make improvements and move forward. This iteration continues until the system is fully developed and meets the requirements of the stakeholders.

The agile model allows for frequent feedback and iteration, making it ideal for the problem statement of designing a system for warehouses and farmers. By breaking down the development into small sprints and working closely with stakeholders, the development team can ensure that the system is developed to meet the needs of the users and is delivered in a timely and efficient manner.

❖ User Stories:

Front of the card	Back of the card
- As a warehouse owner, I should be able to register into the system with details such as name, location, storage capacity, type of goods stored so that the farmer can find and see details of the warehouse	- The system must enforce data validation rules to ensure that required fields are entered and that the data is in the correct format - The system must be accessible to farmers, who can view warehouse details and search for warehouses based on their needs and preferences
- As a warehouse owner, I should be able to login to the system so that I can manage and update my warehouse details.	- The system must validate the warehouse login credentials - The system should only allow other operations of the warehouse after the login is successful

- As a warehouse owner, I want to update the status of my warehouse as crops are added or removed in real time so that the farmers can know about the storage space that is available in the warehouse	-The warehouse owner should be able to update the warehouse inventory status in real-time as crops are added or removed. -The system should be able to automatically adjust the inventory levels and available storage space after any changes are made. -The system should display the current inventory status and available storage space in real-time, to allow farmers to make informed decisions. -The farmer should be able to view the updated inventory status and available storage space before making a reservation for storing their crops.

- As a warehouse owner I should be able to generate reports of warehouse current goods storage so that I can be able to know the current status of the warehouse.	-The system should allow the warehouse owner to generate a report of the current goods storage in the warehouse. -The report should display the total amount of goods stored in the warehouse, broken down by type of goods. -The report should show the current capacity utilization of the warehouse, indicating how much space is available for additional storage. -The report should display the age and condition of the goods currently stored in the warehouse, to ensure that the warehouse is not overstocked with expired or damaged goods.
- As a warehouse owner, I should be able to provide a reservation system for the farmers so that they can book storage space in advance.	-The system should not allow for reservations that exceed the storage capacity of the warehouse. -The system should take the start date and end date for storage.

- As a farmer, I should be able to register into the system with details such as name, location, email, number so that I can utilize the system for tracking crops and finding warehouses for storage.	- The system should validate the input fields given by the farmer during the registration - The system should not register an already registered user. - The system should keep Email unique in the system
- As a farmer, I should be able to login into the system so that I can access the system and do the needful tasks.	- The system must validate the farmer login credentials - The system should only allow other operations by the farmer, after the login is successful
- As a farmer I should be able to keep track of the goods that I have stored so that I can get a better understanding of it.	- The system should allow the farmer to view the details of their stored goods, including the type of goods, quantity, location, and date of storage.

	- The farmer should be able to search for specific goods or filter the goods based on different criteria, such as storage location or date of storage.
- As a farmer I should be able to make a reservation about the storage space in advance so that I do not need to worry about getting space for my goods when I reach the warehouse.	-The system should allow the farmer to view the available storage space at the warehouse and reserve the required amount of space. -The system should display the availability of storage space in real-time, to prevent double booking or overbooking of storage space.
- As a farmer, I should be able to receive alerts when my crops are approaching their storage life limit so that I can sell or use them before they go bad.	-The system should track the storage life limit of the crops stored by the farmer in the warehouse, based on their type, condition, and storage environment. -The farmer should be able to set up alerts or notifications for specific crops, with a configurable threshold for when the alerts should be triggered.

- As a farmer, I should be able to cancel or modify my reservation so that I can manage my crops storage.	-The system should display all the future reservations that are made by the farmer. - If modifying, they should check if the reservation does not exceed the storage capacity of the warehouse. - If modifying, the system confirms the modification and updates the reservation details.
- As a farmer, I should be able to find nearby warehouses so that I can find warehouses to store my goods.	- The system should take input the location from which the farmer wants to find nearby warehouses. - The system should take input the distance of the radius of the circle that may contain the warehouse

- As a farmer, I should be able to get recommendations from the system about the types of crops to grow so that I can optimize my yield and minimize wastage.	- The system should take into consideration the current goods already stored by the warehouses in the system and future reservations to give recommendations about the crops to grow

- As a user, I want a user-friendly interface so that I can have clear instructions and intuitive navigation.	- The user interface should be clear, simple, and intuitive, with easy-to-understand instructions and labeling - The system should use consistent and familiar design patterns and navigation elements, so that users can easily understand how to interact with the interface - The system should use appropriate fonts, colors, and visual elements to make the interface aesthetically pleasing and easy to read - The system should provide feedback to users when they perform actions, so that they know that the system is processing their requests

	- The system should be accessible to users with different levels of technical proficiency and different physical abilities, with support for assistive technologies and other accessibility features - The system should be tested with real users to ensure that it is easy to use and meets their needs
- As a user, I want the system to be secure so that I can protect my sensitive information from unauthorized people.	- The system should check that a user logged in is not a robot or any computerized application. - The system should use strong encryption and secure communication protocols to protect sensitive information from unauthorized access or interception - The system should enforce access controls and authentication mechanisms to ensure that only authorized users can access sensitive information

- As a user, I want the system to be scalable so that it can accommodate future growth and increased usage.	- Ensure that the system architecture and infrastructure can handle future growth and increased usage without compromising performance, reliability, or security. - Design the system to be modular and easily extendable to accommodate new features or functionality.
- As a user, I want the system to be fast and responsive so that it can provide real time updates to farmers about the storage capacity and availability of nearby warehouses	- The system should be able to provide real-time updates about storage capacity and availability of nearby warehouses to farmers, with a response time of no more than 2 seconds - The system should be optimized for high performance, with minimal latency and delays when processing requests and handling data - The system should be able to handle a high volume of concurrent user requests without experiencing performance issues or slowdowns

	- The system should use caching and other optimization techniques to reduce the load on the server and improve response times
- As a user, I want that the system is highly available and reliable, so that I can ensure that I have access to the information whenever I need it	- The system should be available and accessible to users at all times, with minimal downtime or interruptions - The system should be designed with high availability and fault tolerance in mind, with redundant systems and failover mechanisms to minimize the impact of any failures or outages - The system should be able to recover quickly and automatically from any failures or outages, without requiring manual intervention - The system should provide users with clear and timely notifications in the event of any planned or unplanned downtime or maintenance

- As a user, I want the system to be compatible with different platforms and devices, such as laptops, smartphones, and tablets, and should be accessible through a web browser so that I can access the system from any device.	- The system should be compatible with different platforms and devices, such as laptops, smartphones, and tablets - The system should be accessible through a web browser on any device with an internet connection - The user interface should be responsive and adaptive, adjusting to the screen size and resolution of the device being used to access the system - The system should be designed to provide a consistent user experience across different platforms and devices, with all features and functionality available on each platform

	- The system should be tested on different devices and browsers to ensure compatibility and functionality

- As a user, I want the system to provide reports on the usage of each warehouse over time, including which crops are stored and when, and the eviction rate of each warehouse.	- Design a reporting module that allows farmers to generate usage reports for each warehouse over a specified time period. - The system should store historical data on crop storage and eviction rates for each warehouse to enable accurate reporting. - The reports should be customizable and allow the farmer to filter and sort the data based on various criteria, such as crop type, storage period, or warehouse location.
- As a user, I want the system to have a robust data backup and recovery system so that data is protected in case of system failure.	- The backup system should be designed to take regular, automated backups of all system data, including user information and other critical data - The backup system should be securely stored off-site, preferably in a separate geographic location to ensure that data is protected in case of natural disasters or other emergencies - The recovery system should be tested regularly to ensure that backups can be restored

	quickly and efficiently in case of system failure or other disasters
- As a user, I want the system to have error handling mechanisms in place so that errors and exceptions are handled gracefully and do not impact the operation of the system.	- The system should be designed with robust error handling mechanisms to handle errors and exceptions gracefully - Error messages should be clear and informative, providing users with a clear understanding of the issue and any steps that can be taken to resolve it - The system should log errors and exceptions, providing developers and system administrators with the information they need to diagnose and resolve issues - Error handling mechanisms should not impact the overall operation of the system, and should be designed to minimize any impact on users - The system should be tested thoroughly to ensure that all possible errors and exceptions are handled correctly, and that error messages are clear and informative

	- Error handling mechanisms should be reviewed and updated regularly to ensure that they remain effective and up-to-date with any changes to the system or underlying technologies

❖ Product Backlog:

Sprint 1: Login and Registration of Farmer and Warehouse:

1. As a warehouse owner, I should be able to register into the system with details such as name, location, storage capacity, type of goods stored so that the farmer can find and see details of the warehouse

2. As a warehouse owner, I should be able to login to the system so that I can manage and update my warehouse details.

3. As a farmer, I should be able to register into the system with details such as name, location, email, number so that I can utilize the system for tracking of crops and finding warehouses for storage.

4. As a farmer, I should be able to login into the system so that I can access the system and do the needful tasks.

Sprint 2: Warehouse and Farmer storage updates

1. As a warehouse owner, I want to update the status of my warehouse as crops are added or removed in real time so that the farmers can know about the storage space that is available in the warehouse

2. As a warehouse owner I should be able to generate reports of warehouse current goods storage so that I can be able to know the current status of the warehouse.

3. As a farmer I should be able to keep track of the goods that I have stored so that I can get a better understanding of it

4. As a farmer, I should be able to receive alerts when my crops are approaching their storage life limit so that I can sell or use them before they go bad.

Sprint 3: Warehouse Reservations

1. As a warehouse owner, I should be able to provide a reservation system for the farmers so that they can book storage space in advance.

2. As a farmer I should be able to make a reservation about the storage space in advance so that I do not need to worry about getting space for my goods when I reach the warehouse.

3. As a farmer, I should be able to cancel or modify my reservation so that I can manage my crops storage.

Sprint 4: Nearby warehouses and Crops recommendations

1. As a farmer, I should be able to find nearby warehouses so that I can find warehouses to store my goods.

2. As a farmer, I should be able to get recommendations from the system about the types of crops to grow so that I can optimize my yield and minimize wastage.

❖ **<u>Domain Analysis Model</u>**

Introduction: The domain for the system is Providing warehouse information to farmers. The motive behind developing such a system is the food being wasted annually is due to lack of proper storage facilities. This number keeps on increasing and something needs to be done to control this,so using this system we can decrease the wastage of food.

Boundary objects:

- Login / Registration Page

- Login Page eventually provides users to log in/ register and manage their personal information and manage their content,So In some sense Login Page is providing a visual boundary between user and system.

- Users and warehouse managers will have separate Login / Registration pages to create boundaries between users and managers.

● Payment Interface

- Payment interface restricts users to access the services without completing the payment procedure, thus it acts as a boundary object.

Entity objects:

1. **Farmer** : Farmers can utilize the system for tracking of crops and finding warehouses for storage.

2. **Warehouse manager**: Warehouse manager can update the status of their warehouse as crops are added or removed in real time so that the farmers can know about the storage space that is available in the warehouse.

3. **Warehouse**: A warehouse is used to store the crops,so that farmers can store their crops and make more profit from that and decrease the wastage of crops.

4. **Reservation** : It contains information about farmer’s advance reservation for their crops in particular warehouses.

Control objects:

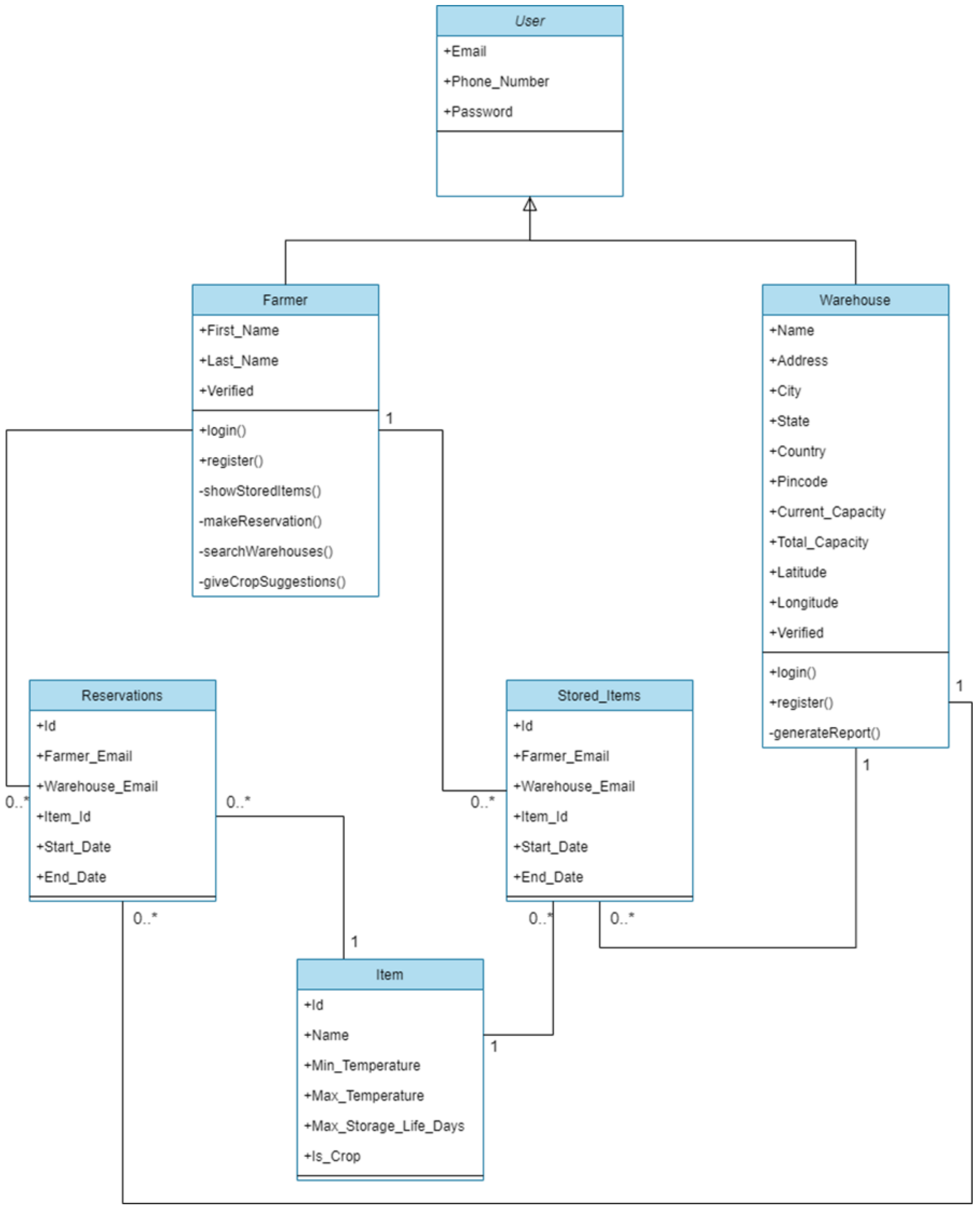
1. **Registration as farmer**: It controls the flow of registration data to the database.

2. **Login as farmer**: It controls the flow of validation of user information from the database.

3. **Reservation** : It controls the flow of reservation while reserving the warehouse space.

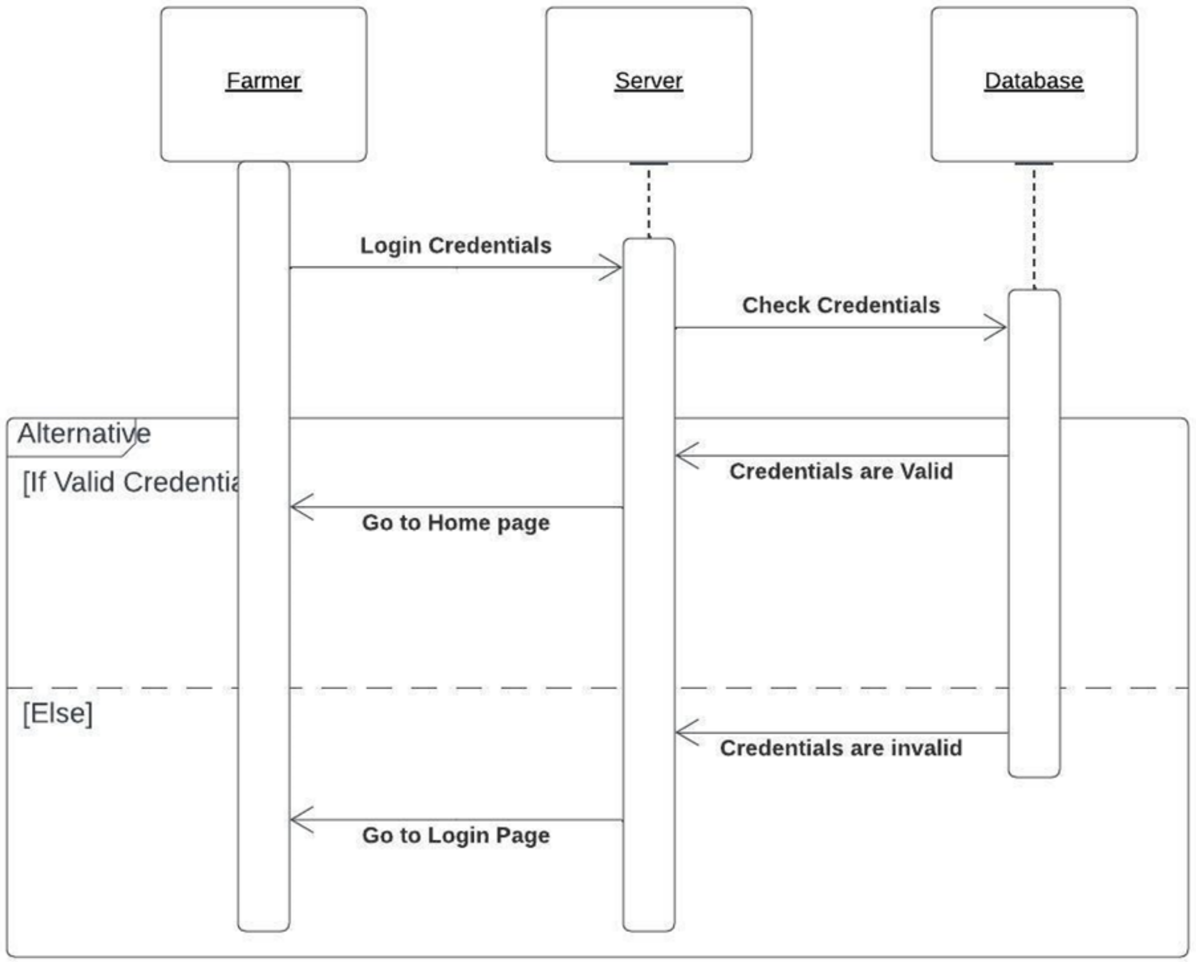
4. **Reservation into warehouse**: It controls the flow of user information to the database.

❖ **Class Diagram**

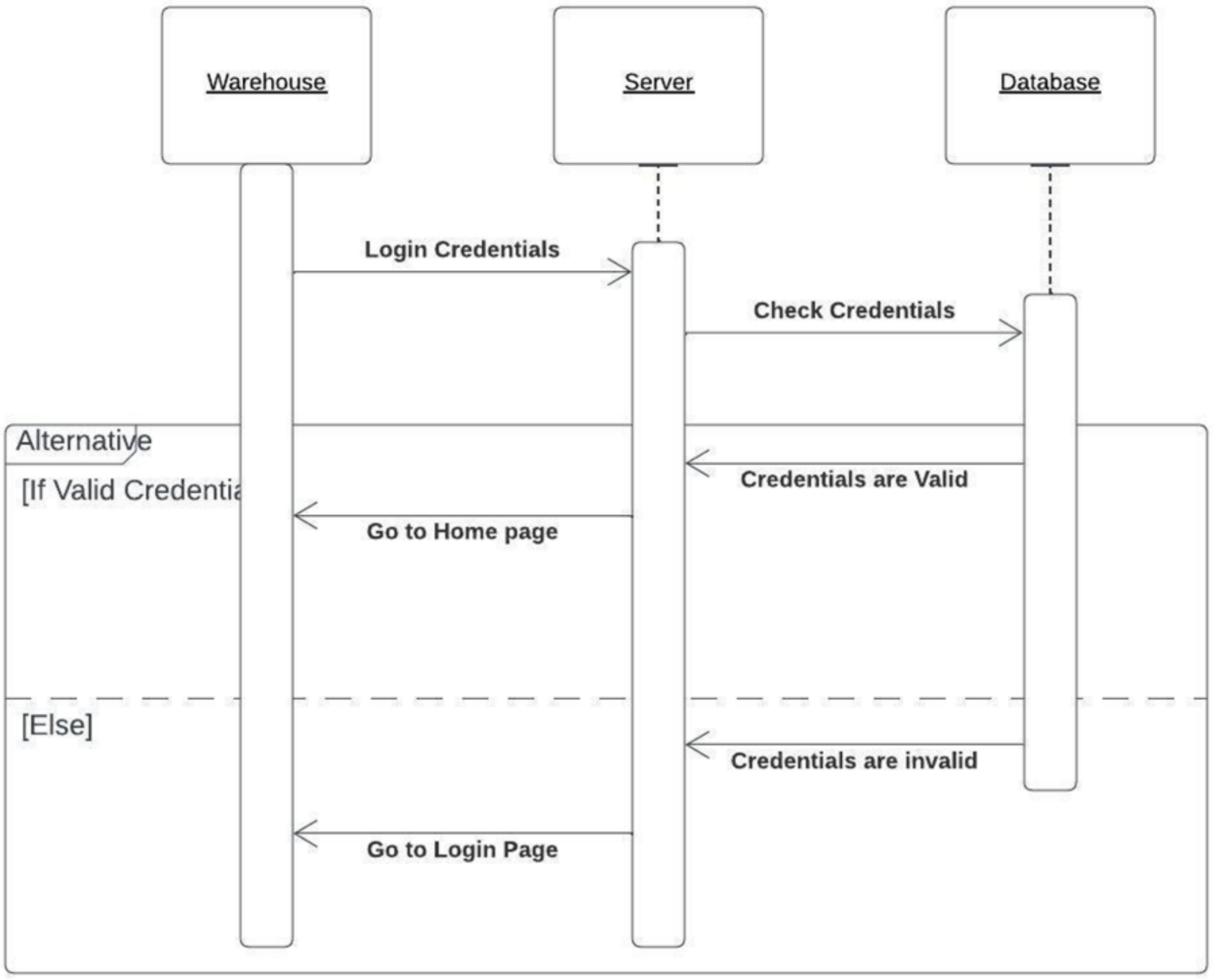

User
+Email
+Phone_Number
+Password
Farmer
+First_Name
+Last_Name
+Verified
+login()
+register()
-showStoredItems()
-makeReservation()
-searchWarehouses()
-giveCropSuggestions()
Warehouse
+Name
+Address
+City
+State
+Country
+Pincode
+Current_Capacity
+Total_Capacity
+Latitude
+Longitude
+Verified
+login()
+register()
-generateReport()
Reservations
+Id
+Farmer_Email
+Warehouse_Email
+Item_Id
+Start_Date
+End_Date
Stored_Items
+Id
+Farmer_Email
+Warehouse_Email
+Item_Id
+Start_Date
+End_Date
Item
+Id
+Name
+Min_Temperature
+Max_Temperature
+Max_Storage_Life_Days
+Is_Crop
1
0..*
0..*
0..*
0..*
0..*
0..*
1
1
1
1

❖ **<u>Sequence Diagram</u>**

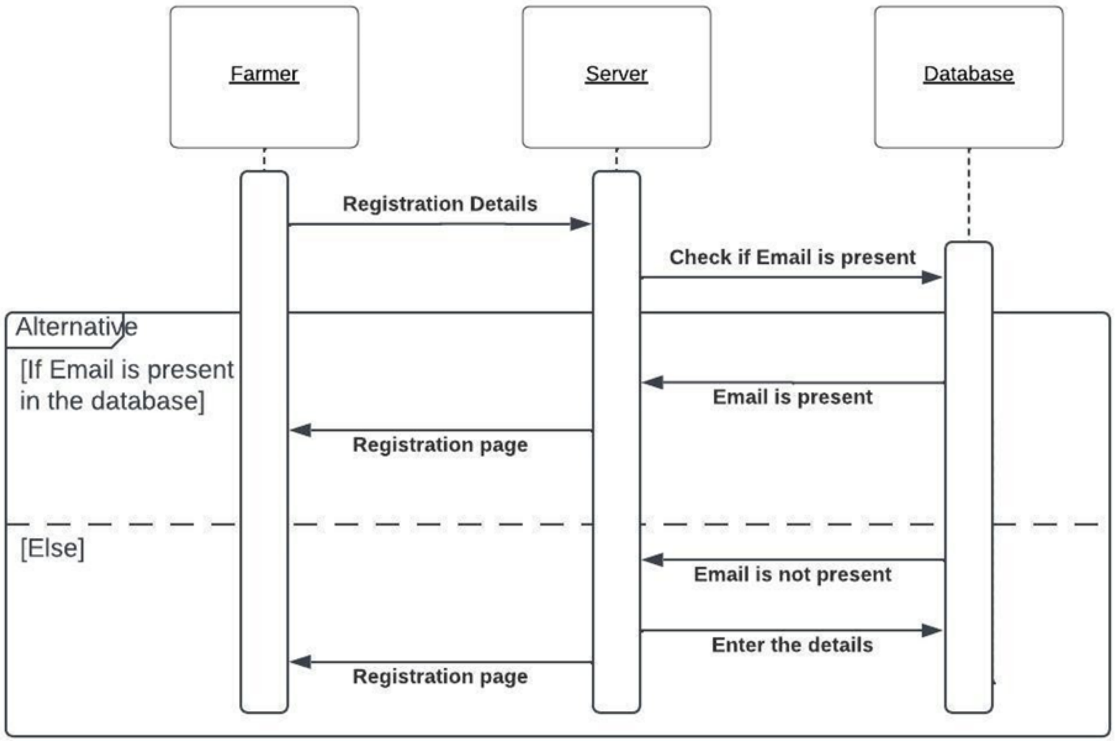
1. Farmer Login



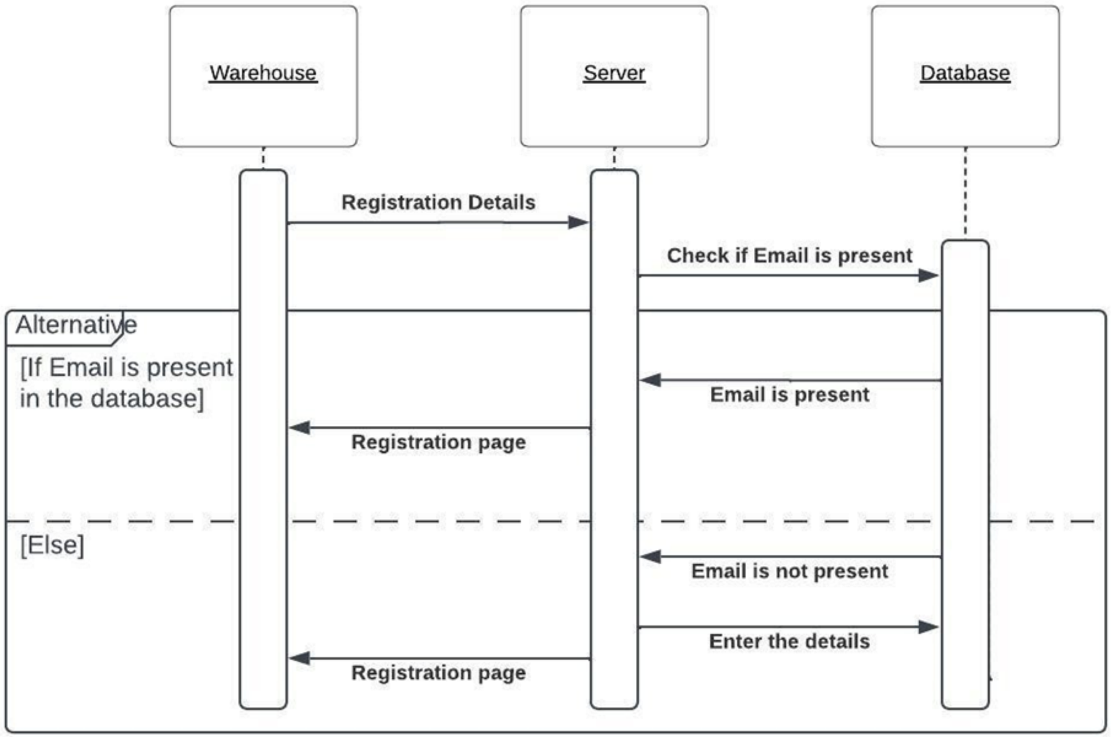
2. Warehouse Login



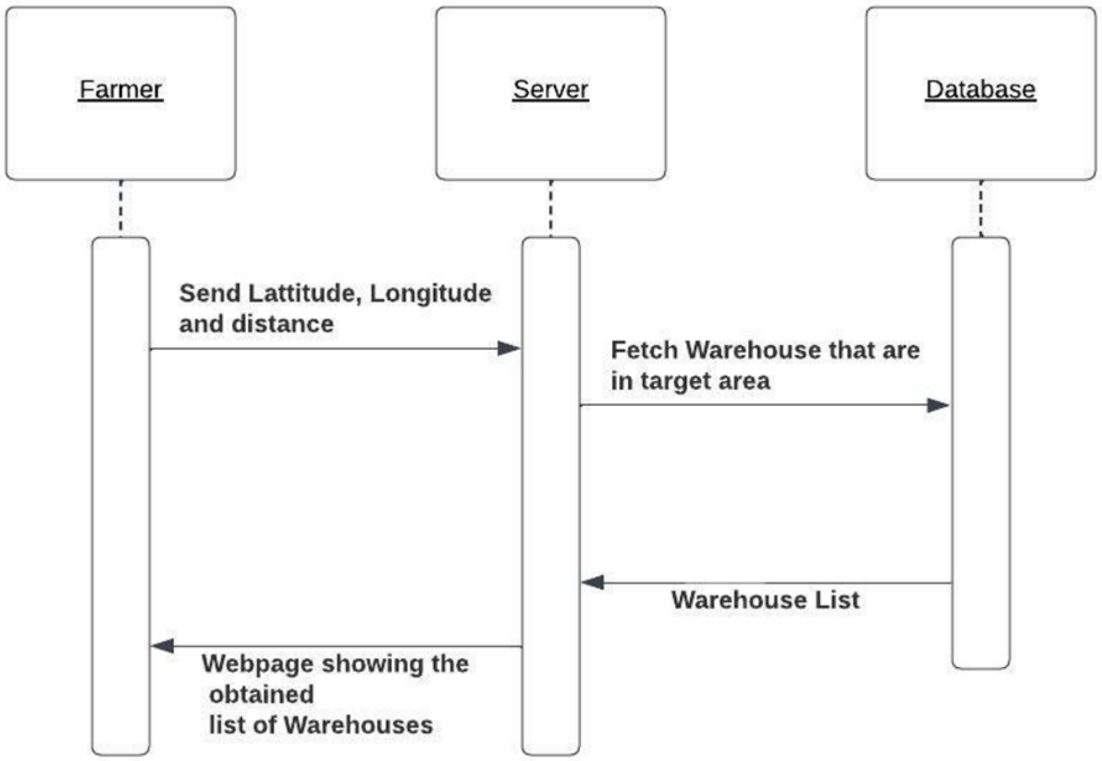
3. Farmer Registration



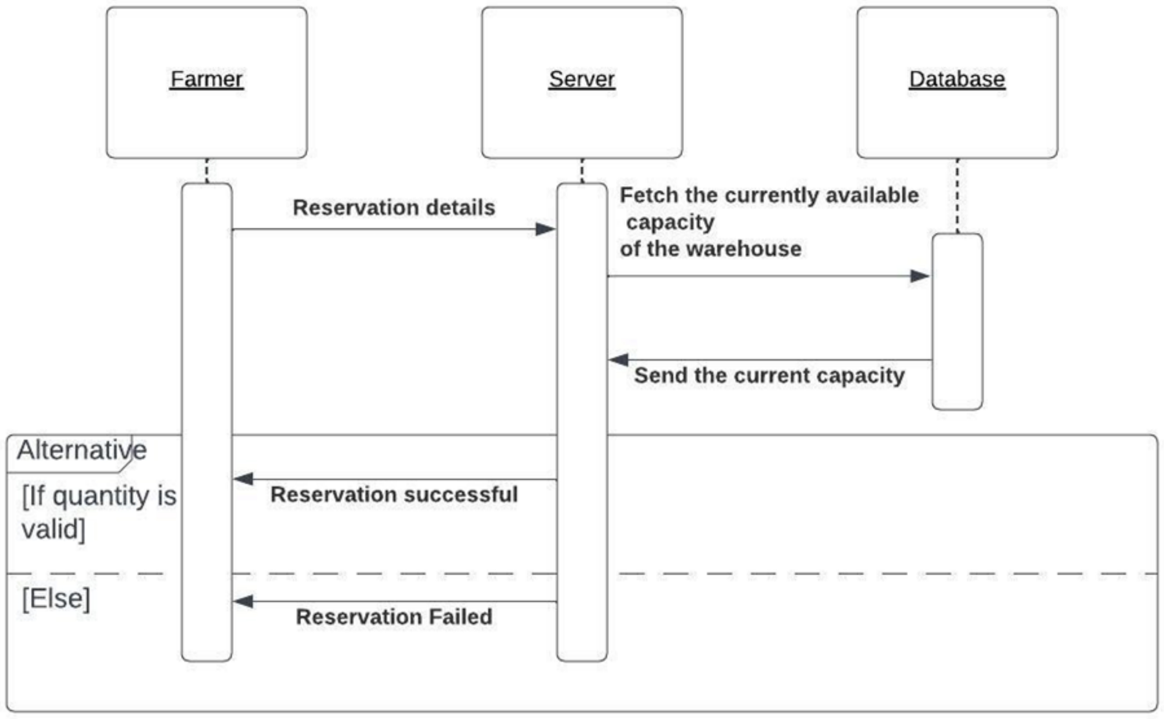
4. Warehouse Registration



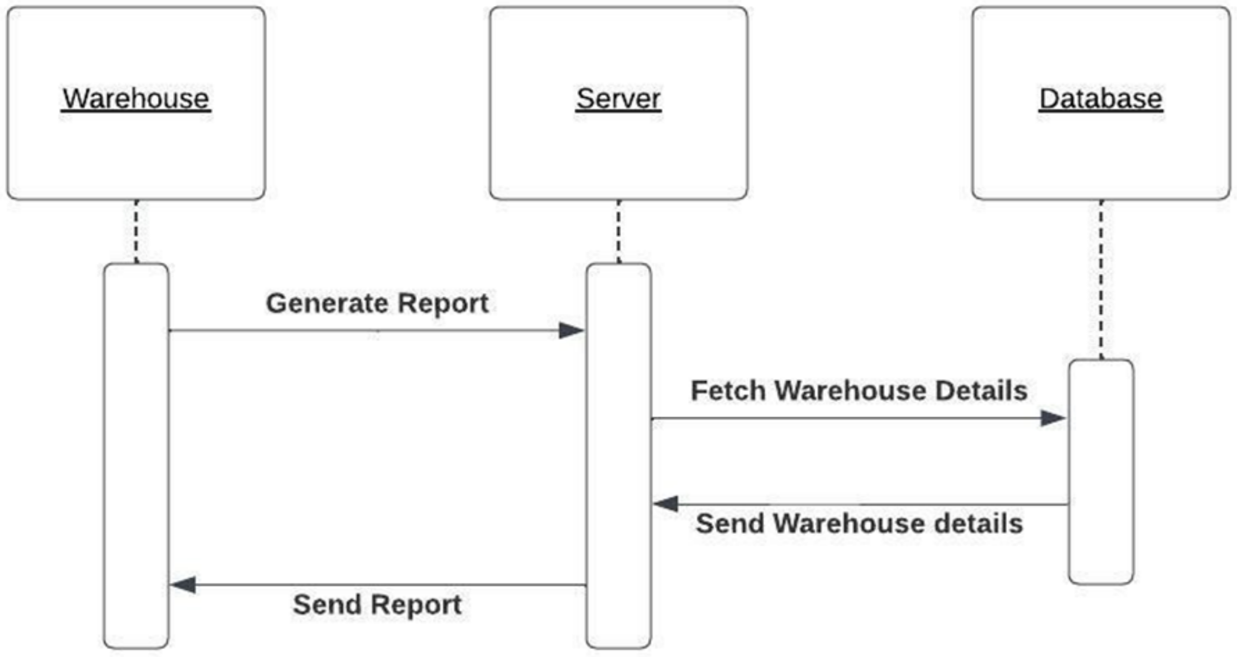
5. To find a nearby warehouse


Farmer
Server
Database
Send Lattitude, Longitude and distance
Fetch Warehouse that are in target area
Warehouse List
Webpage showing the obtained list of Warehouses

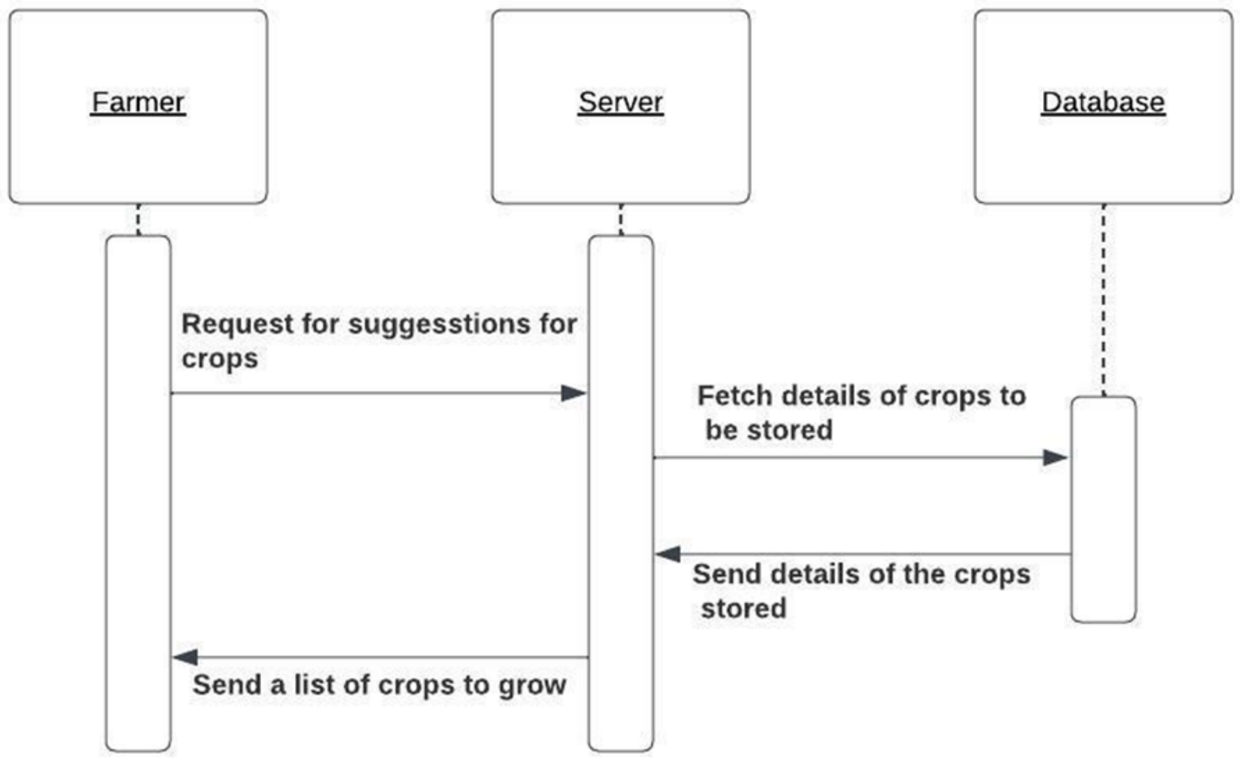
6. Farmer Reservation of the Warehouse based on availability



7. Report generation by warehouse



8. Get suggestions from the warehouse data as per what the farmer should grow more.


Farmer
Server
Database
Request for suggesstions for crops
Fetch details of crops to be stored
Send details of the crops stored
Send a list of crops to grow

❖ **Design Goals**

The design goals of our project typically are:

1. ***Scalability:*** The WMS should be able to handle large volumes of data, transactions, and users as the warehouse grows over time. The system should be designed to accommodate future expansion and handle increased workload without compromising performance or stability.

2. ***Flexibility:*** The WMS should be flexible enough to adapt to different warehouse configurations, processes, and requirements. It should be configurable to support various types of warehouses, such as fulfillment centers, distribution centers, and third-party logistics providers.

3. ***Efficiency:*** The WMS should optimize warehouse operations by automating routine tasks, streamlining processes, and minimizing unnecessary manual effort. It should facilitate efficient order picking, packing, shipping, receiving, and inventory management, resulting in improved productivity and reduced operational costs.

4. ***Accuracy:*** The WMS should provide real-time visibility and accurate tracking of inventory movements, locations, and quantities to ensure accurate inventory

management. It should also support barcode scanning, or other technologies for accurate and reliable data capture.

5. ***Security:*** The WMS should ensure the confidentiality, integrity, and availability of warehouse data. It should have robust authentication, authorization, and access control mechanisms to protect against unauthorized access, data breaches, and other security risks.

6. ***User-friendliness:*** The WMS should have an intuitive and user-friendly interface that is easy to use for warehouse personnel, including warehouse workers, supervisors, and managers. It should provide clear visibility into warehouse operations, enable quick decision-making, and support user customization based on roles and responsibilities.

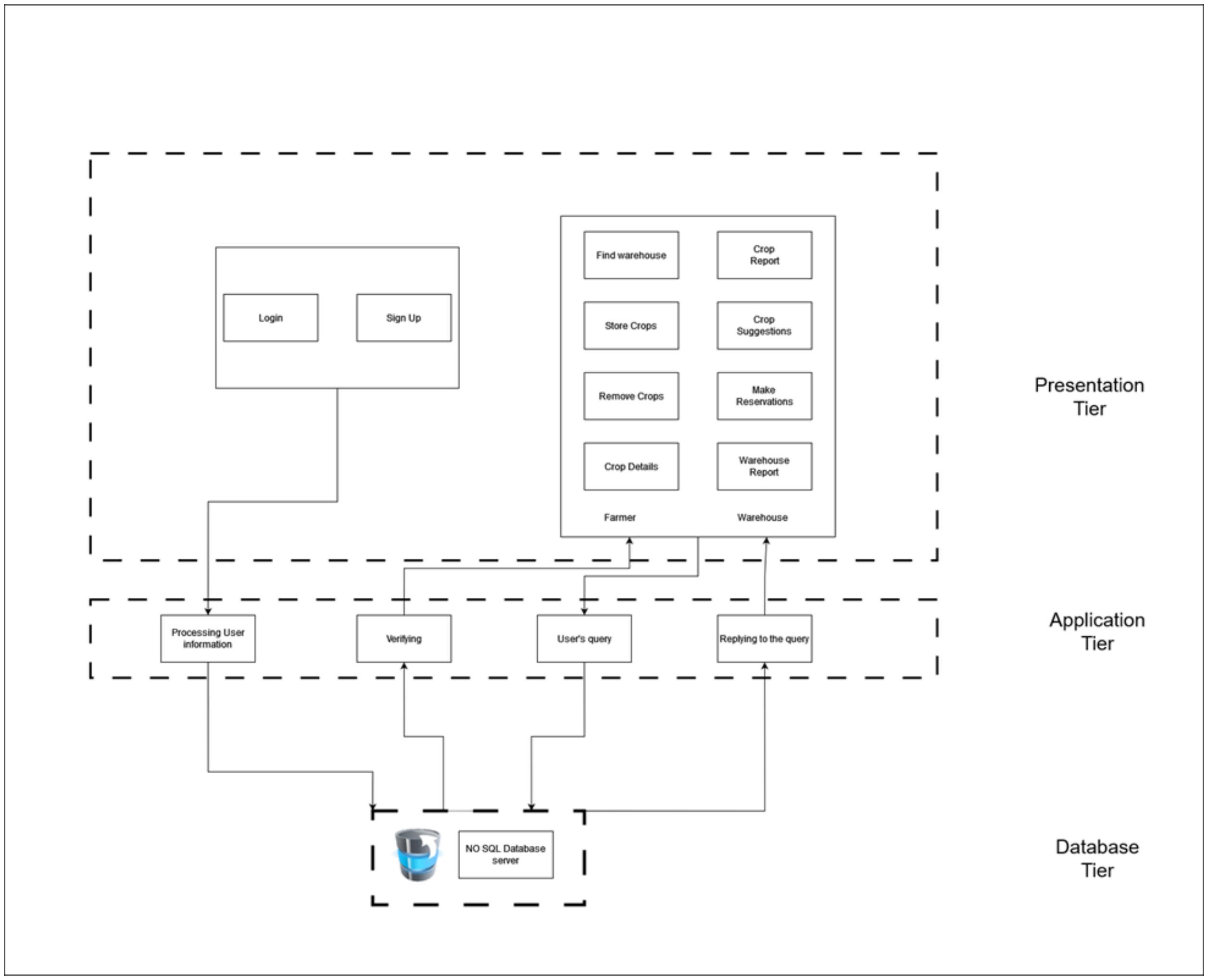
7. ***Reliability and fault tolerance:*** The WMS should be designed to operate reliably and be fault-tolerant to ensure continuous warehouse operations. It should have backup and recovery mechanisms in place to protect against data loss or system failures.

8. ***Supportability and maintainability:*** The WMS should be designed for ease of support and maintenance, including robust logging, monitoring, and debugging capabilities. It should also have a well-defined support process and documentation to enable efficient issue resolution and system maintenance.

9. ***Reporting and analytics:*** The WMS should provide comprehensive reporting and analytics capabilities to enable data-driven decision-making. It should generate standard and custom reports, provide real-time dashboards, and support data analysis for performance measurement, trend analysis, and strategic planning.

By incorporating these design goals into the development of our project, we can create a robust, scalable, efficient, and user-friendly solution that meets the specific needs of a warehouse operation.

❖ **High Level System Design**



In a Django web application, the presentation layer is responsible for managing the user interface and functions, the application layer handles all the logic within the Django framework, and the database layer utilizes MongoDB for data storage and retrieval.

❖ **<u>Architecture : N- tier architecture</u>**

A software architectural approach that divides an application into logical layers and physical tiers is known as N-tier architecture. Each layer is responsible for a distinct task and might be hosted on separate computers or clusters. This separation of concerns enables scalability, manageability, adaptability, and security.

A three-tier architecture is a sort of n-tier architecture in which an application is divided into three logical layers and physical tiers. These are the tiers:

The presentation layer: The presentation layer is the topmost layer and deals with user interaction. The presentation layer of a farmer warehouse management system might be a web or mobile application that allows farmers to engage with the system. The presentation layer can enable user identification, warehouse inventory display, product addition and removal, and report generation.

Business Logic Layer: The business logic layer is the middle layer and contains the core functionalities of the application. In a farmer warehouse management system, the business logic layer can perform tasks such as inventory management, order management, and handle business rules and calculations.

Data Storage Layer: The data storage layer is the bottom layer and is responsible for managing data storage and retrieval. In a farmer warehouse management system, the data storage layer can consist of a database that stores all the information related to the warehouse, including inventory levels, product details, and customer orders.

By using a 3-tier architecture in a farmer warehouse management system, you can achieve better code maintainability, scalability, and security. It also allows for easy upgrades and modifications to specific layers without affecting the entire system.