IT314 Software Engineering

Group 25

Name	Student Id
TEJABWALA MOHAMMAD JABIRBHAI	202001406
PRARTHEE BHAVINBHAI DESAI	202001257
AASTHA JAGDISH SHETTY	202001260
PATEL NISARG NAGINBHAI	202001436
DETROJA ARTH JITESHBHAI	202001274
BHUVA MEHULKUMAR VISABHAI	202001437
RUSHABH MAHESH PATEL	202001419
JAIMIN RATHWA (GR)	202001423
GANVIT GAURANG SHANKARBHAI	202001247
DAMOR ANIRUDDH RATANBHAI	202001255

Lab 6

❖ Domain Analysis Model

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:

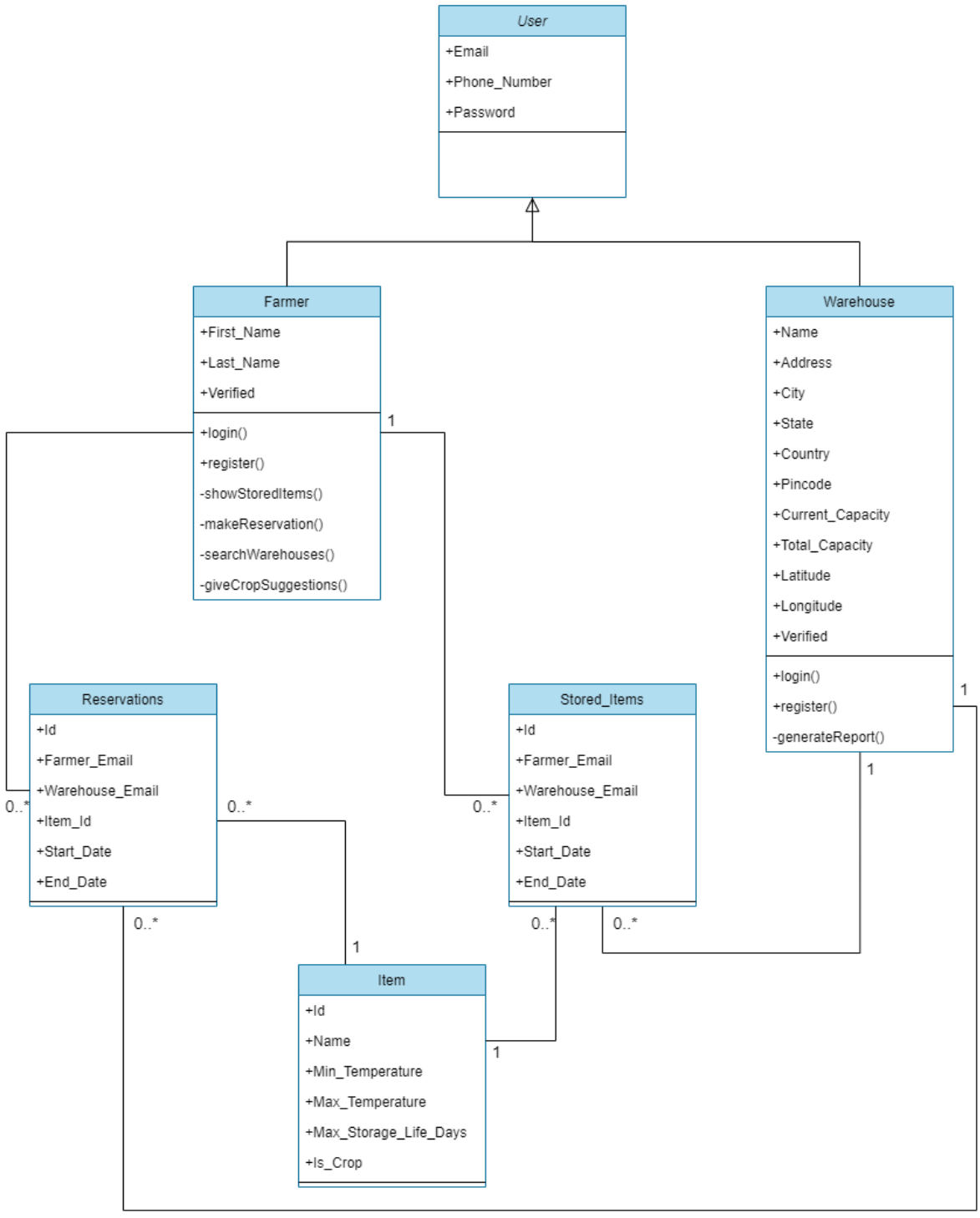
1. **Registration as farmer**: It controls the flow of registration data to the database.

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.

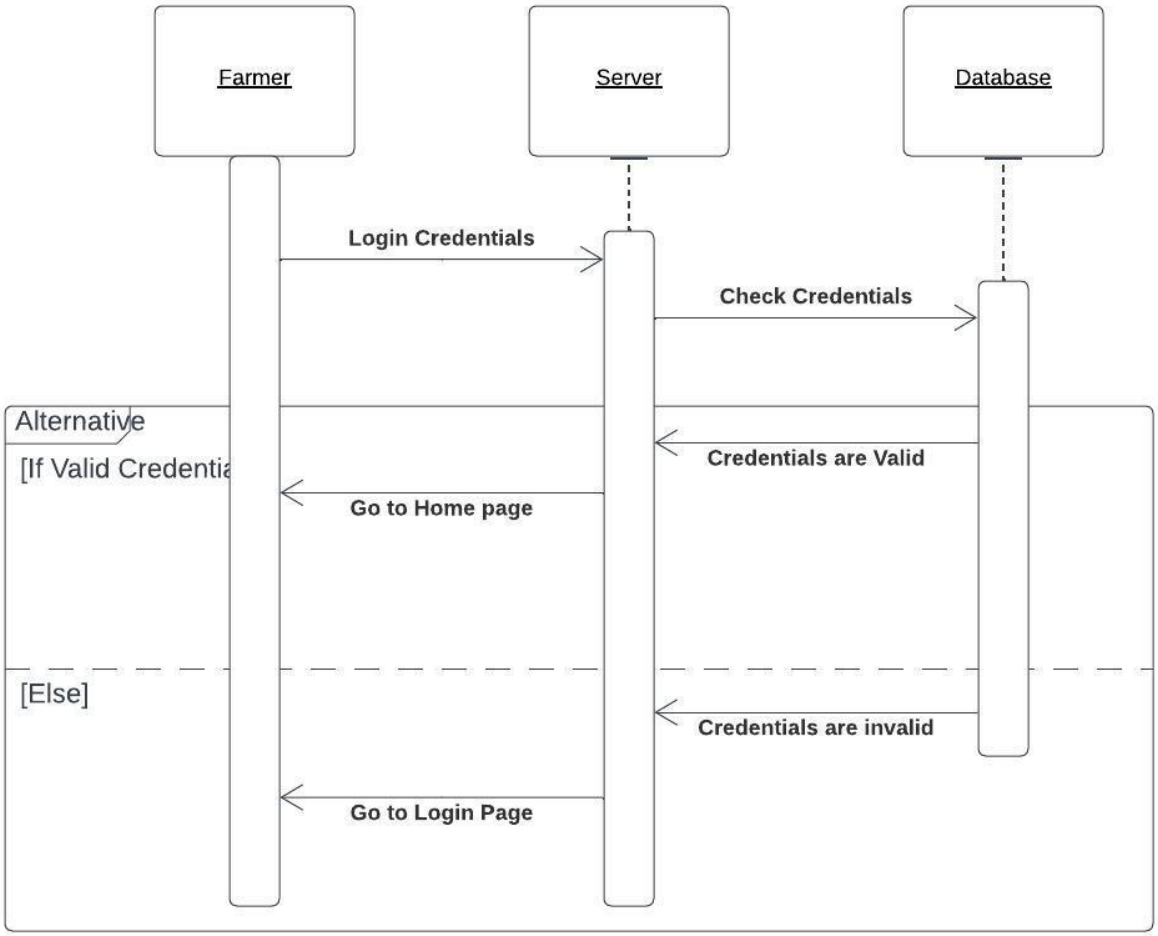
4. **Reservation into warehouse**: It controls the flow of user information to the database.

❖ Class Diagram

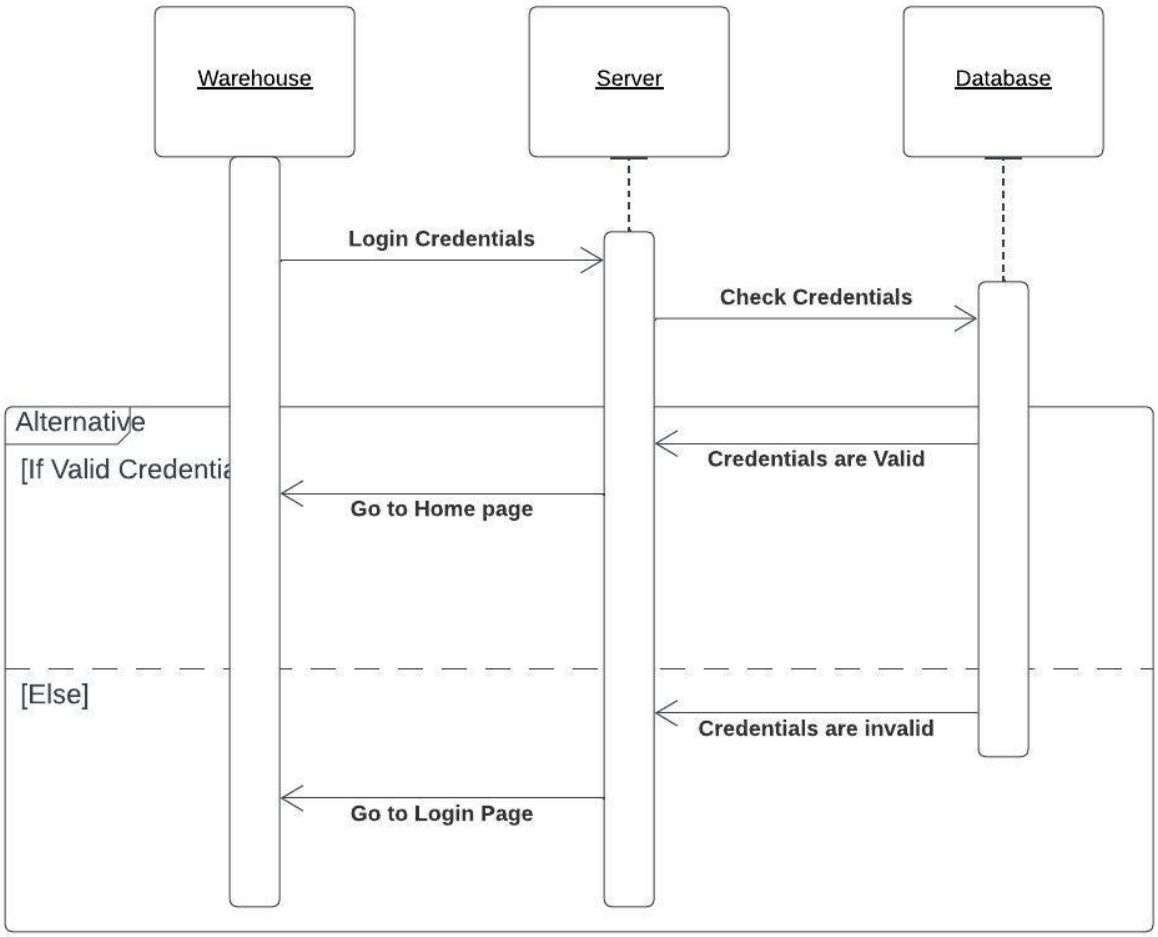


❖ Sequence Diagram

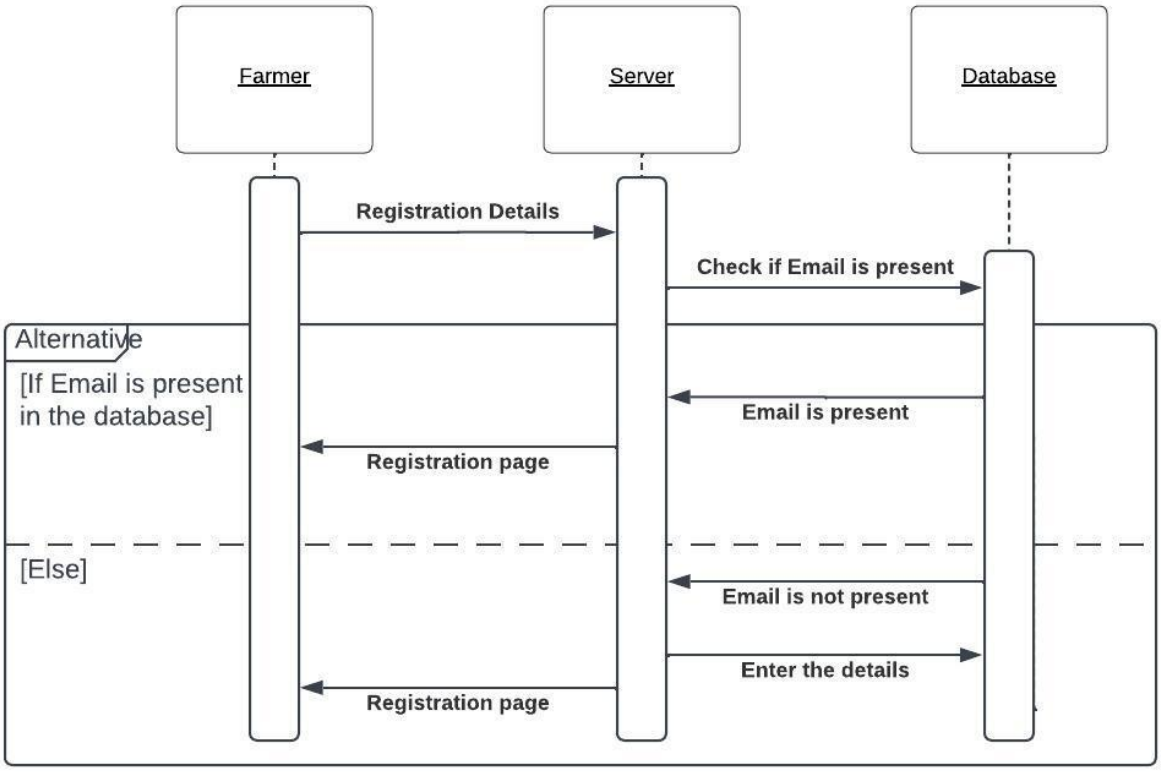
1. Farmer Login



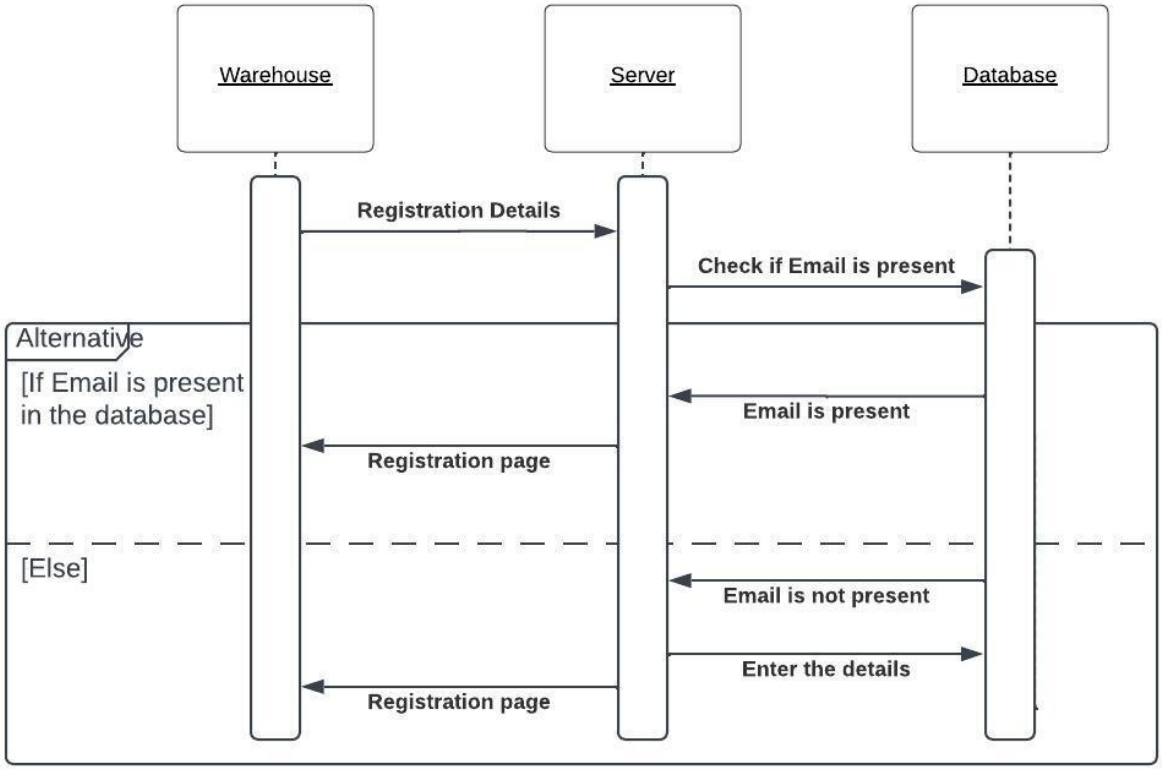
2. Warehouse Login



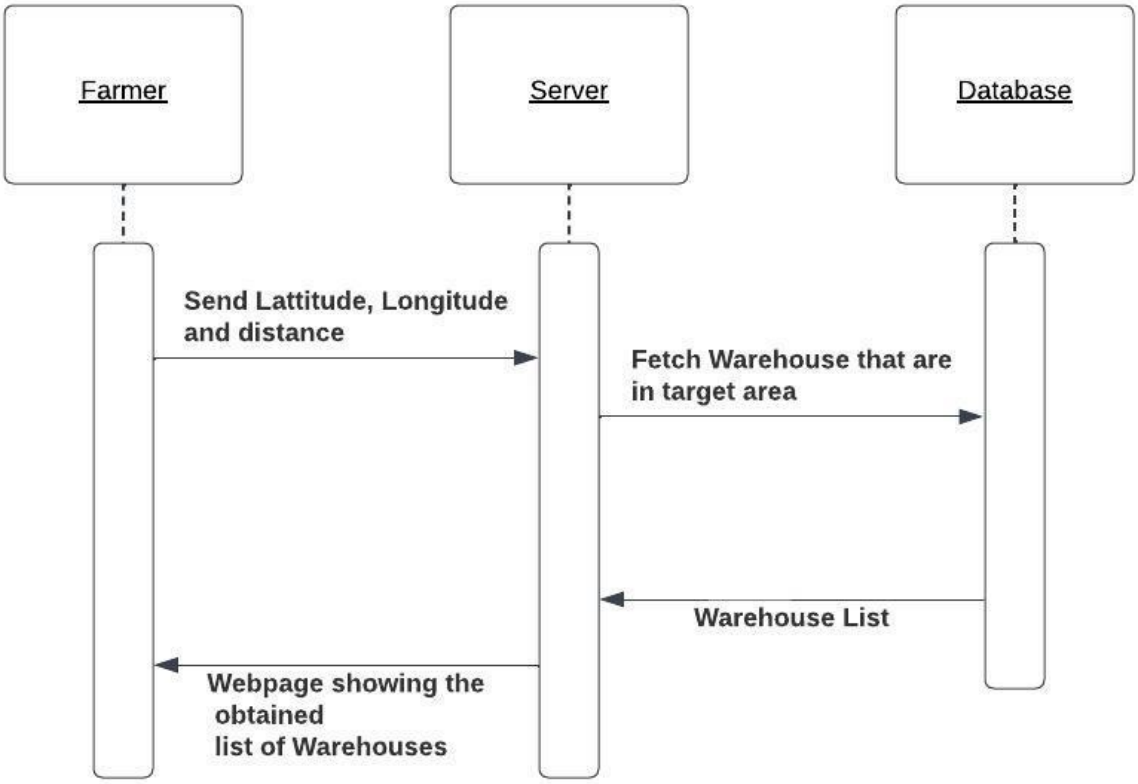
3. Farmer Registration



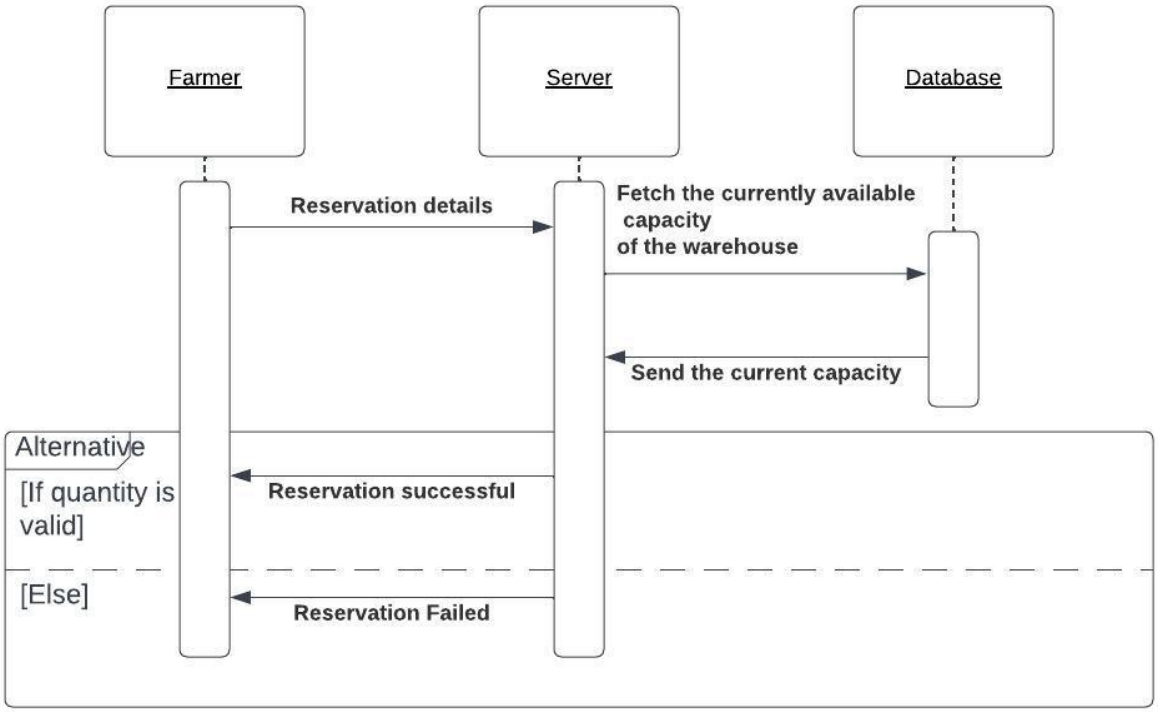
4. Warehouse Registration



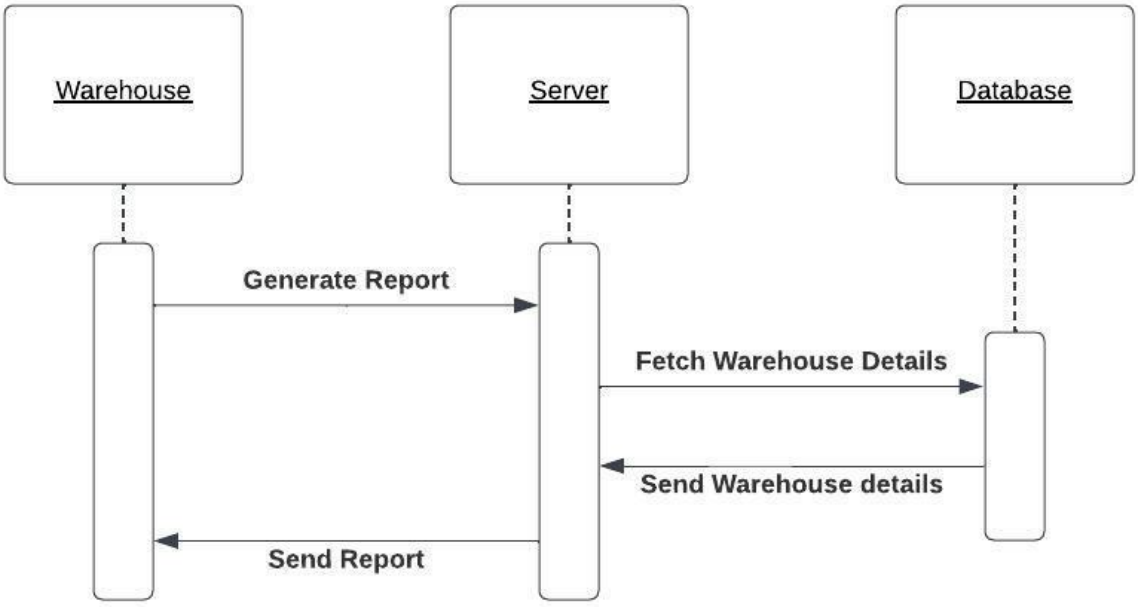
5. To find a nearby warehouse



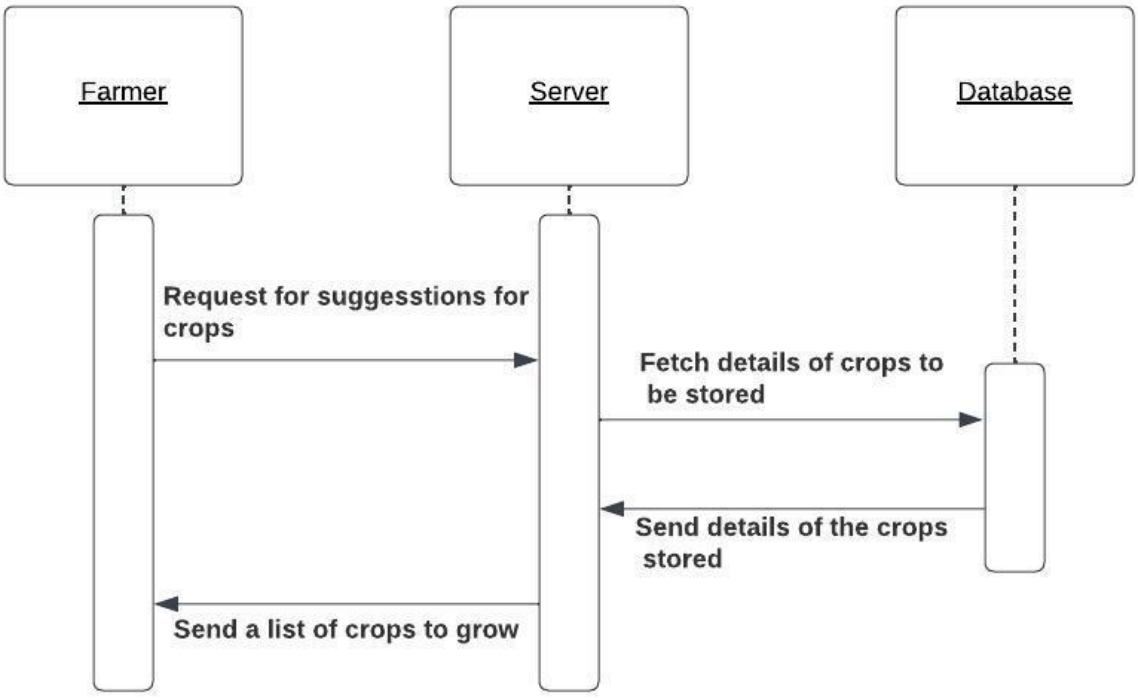
6. Farmer Reservation of the Warehouse based on availability



7. Report generation by warehouse



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



❖ 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.

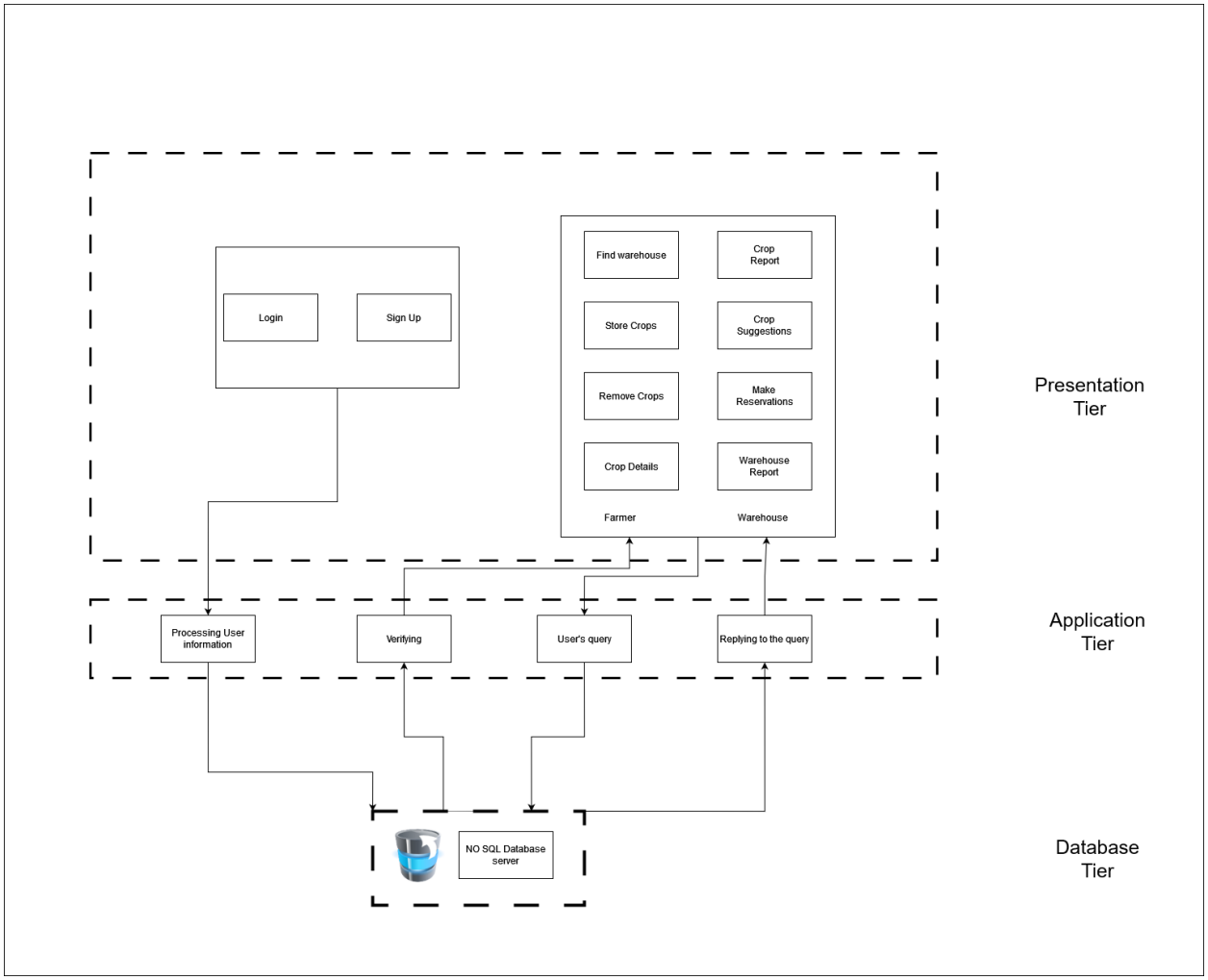
7. ***Reliability and fault tolerance:*** The WMS should be designed to operate reliably and be fault-tolerant to ensure continuous warehouse operations. It should have backup and recovery mechanisms in place to protect against data loss or system failures.

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.

❖ Architecture : N- tier architecture

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.