

CPEN 333: System Software Engineering

Amazoom Automated Warehouse

December 8 2021

Oon Chu Lip 94755584
Anoush Sepehri 66723735
Gurmehak Pannu 87263455
Hayato Terao 23227598

Executive Summary

Attached is the documentation for the proposed Amazoom Warehouse Automation System. The system can be broken down into three components: the customers (client), server, and warehouses. The customer uses an intuitive user interface for ordering items. Within the customer interface, the user can see all available products, select certain items, and submit an order. Once the order is submitted, the customer can keep track of whether the order has been accepted, currently being processed, or is out for delivery. The server acts as the interface between the customers and the warehouses. The server keeps track of all warehouses and customers actively connected and dispatches order requests to the correct warehouse. The warehouse includes a user interface for the corresponding manager to monitor. In the user interface, the manager can keep track of orders that are accepted, actively being processed, or delivered, as well as the status of robots in the warehouse. The user interface also provides a tracker for the current inventory on the shelves, as well as the estimated inventory based on the trucks that have been ordered and are on their way to the docks.

The proposed Amazoom automation system provides several advantages when implemented. The customer is provided with real-time information of the available items and their inventory to order. The server operates completely autonomously and directs customer's orders to the appropriate warehouse. The server can keep track of any number of customers, as well as any number of warehouses to achieve this. In the warehouse automation program, a dynamic number of robots can be selected. If many orders are requested, the manager can increase the number of robots. If there are not many orders requested, the manager can reduce the number of active robots, thus improving efficiency. The user interface also provides a comprehensive list of information regarding active robots, their statuses (i.e. picking up item or dropping off item), their function (i.e. restocking shelves or preparing a delivery), as well as what they are currently carrying. An animated layout of the warehouse is also presented to monitor their position in real-time. Lastly, the inventory is autonomously monitored and anytime a product becomes low in stock, restocking trucks with that product are automatically sent to the docks. This eliminates the need for the warehouse managers to monitor inventory and always ensures that the warehouse is stocked with enough products.

The proposed warehouse automation program for Amazoom can be integrated into existing warehouses to greatly improve operation. Deliveries, warehouse inventories, autonomous robots can all be monitored by a warehouse manager through a simple and intuitive graphical user interface. The provided server/client software also makes it extremely easy for the customer to order items and will thus bring in more revenue compared to competitors. Lastly, the final system can easily be scaled up to accommodate multiple warehouses and multiple customers placing orders at the same time.

Table of Content

High-level Overview	1
Use Cases	2
Checking warehouse status	2
Adding product to warehouse	2
Removing product from warehouse	2
Adding and removing robot to/from warehouse	3
Restocking a product	3
Processing an order	3
Updating the status of an order	3
Ordering a product	4
User Interface	5
Technology Stack	8
Communication Protocol	9
Warehouse Logic	11
Robot Logic	12
Class Diagram	13
Test Plan	14

High-level Overview

Customers connect to the server through a customer user interface. They place orders that are sent to a server which controls which warehouse will receive the order. Depending on the products in the order, a warehouse in the network will receive it (or part of it) and begin autonomously preparing it through the use of Amazoom robots. Order statuses are constantly being relayed back to the customer when the order has been accepted, in progress, and out for delivery. The high-level overview of the system is shown in Fig. 1.

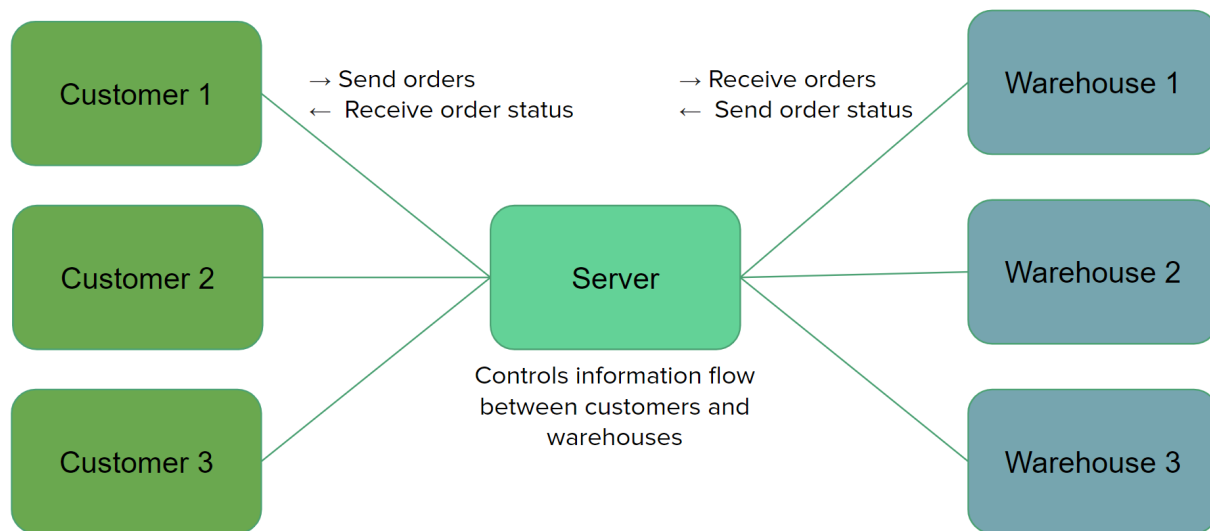


Figure 1: Object Interaction Diagram

Use Cases

The use cases for the automated warehouse system are described below and shown in Fig. 2.

Checking warehouse status

Benefit	<ul style="list-style-type: none">• Manager can check the status of the warehouse including the inventory, number of robots, each robot's status, and order histories
Interaction	<ul style="list-style-type: none">• Manager opens the admin user interface• Central computer sends information of the inventory, robot status, and list of processed orders to the UI
Effects	<ul style="list-style-type: none">• Admin UI is updated

Adding product to warehouse

Benefit	<ul style="list-style-type: none">• Customer gets the newest information of all available products
Interaction	<ul style="list-style-type: none">• Manager enters name, weight, and volume of product to be added• Warehouse checks to see if a product of same name does not exist in the warehouse• If not, adds the new product to the inventory database, assign a shelf to the new product, and places an order for restocking the new product• Send updated inventory information to the customers
Effects	<ul style="list-style-type: none">• Inventory database of the warehouse is updated• Shelf database of the warehouse is updated• Inventory database of the server is updated

Removing product from warehouse

Benefit	<ul style="list-style-type: none">• Customer gets the newest information of all available products
Interaction	<ul style="list-style-type: none">• Manager selects the product to removed from the list of products• Remove the product from the inventory database• Send updated inventory information to the client
Effects	<ul style="list-style-type: none">• Inventory database of the warehouse is updated• Inventory database of the server is updated

Adding and removing robot to/from warehouse

Benefit	<ul style="list-style-type: none">• Number of robots in the warehouse is changed to control operation speed and efficiency
Interaction	<ul style="list-style-type: none">• Manager selects to add or remove a robot from a warehouse• If add, adds a new robot to the warehouse• If remove, waits until there is one robot not working on a task and removes from the warehouse
Effects	<ul style="list-style-type: none">• List of robots in central computer is updated

Restocking a product

Benefit	<ul style="list-style-type: none">• Product is restocked
Interaction	<ul style="list-style-type: none">• Central computer checks if there are any shelves that can stock more of the product assigned to the shelf• If there is, requests a restocking truck with specified numbers of specified products• Robot moves products from restocking truck to the shelves
Effects	<ul style="list-style-type: none">• Inventory database in the server is updated• Shelf database of the warehouse is updated

Processing an order

Benefit	<ul style="list-style-type: none">• Customer receives the products ordered
Interaction	<ul style="list-style-type: none">• Warehouse checks if the order can be fulfilled• If it can, send commands to robot and request delivery truck• Robots pick up specified numbers of specified products from the shelves and transport them to the delivery truck• Dispatch delivery truck when robot is done stocking
Effects	<ul style="list-style-type: none">• Shelf database of the warehouse is updated• Order list of the warehouse is updated

Updating the status of an order

Benefit	<ul style="list-style-type: none">• Customer receives the status of an order
Interaction	<ul style="list-style-type: none">• Warehouse checks if the order received can be fulfilled• If it can, update to "Accepted". Else, update to "Denied"• Warehouse gives commands to the robots and updates the status to "In progress"• When delivery truck dispatches, update to "Delivered"
Effects	<ul style="list-style-type: none">• Order list of the warehouse is updated

Ordering a product

Benefit	<ul style="list-style-type: none"> Customer receives the ordered product if accepted
Interaction	<ul style="list-style-type: none"> Customer enters the name and quantity of the product to order Customer places an order If order can be fulfilled by any of the available warehouses, order is accepted by the warehouse and the product is sent to the customer Update inventory status of the warehouse
Effects	<ul style="list-style-type: none"> Inventory status of the applicable warehouse is updated Order list of the warehouse is updated Robots in warehouses moves to pick the products from the shelves A delivery truck is ordered to deliver the products to the customer

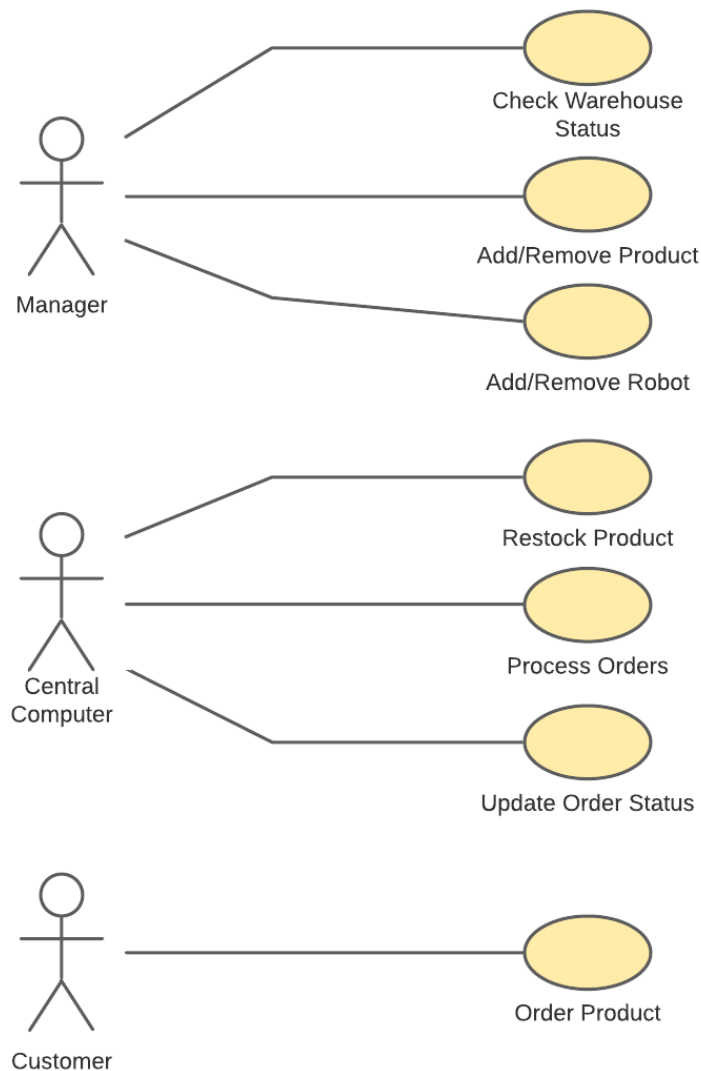
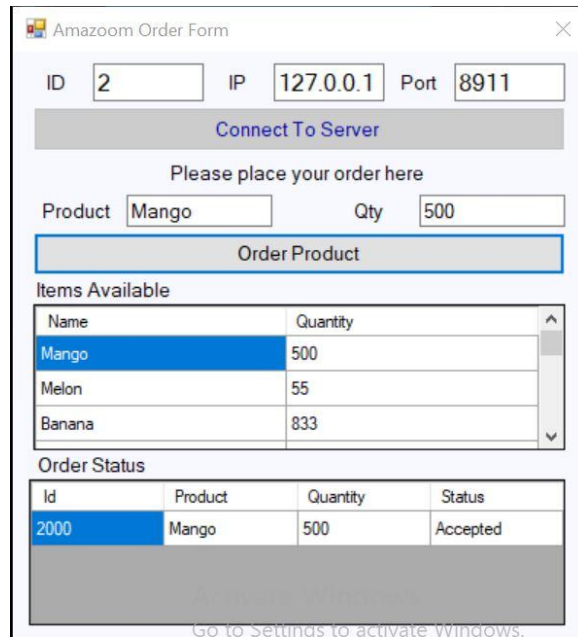


Figure 2: Use Case Diagram

User Interface

The user interface for the customer is shown in Fig. 3. Through this interface, the customer can connect to the server and place an order of a product with an associated quantity. Furthermore, the customer can observe the items available with their maximum quantities and the history statuses of all previous orders.



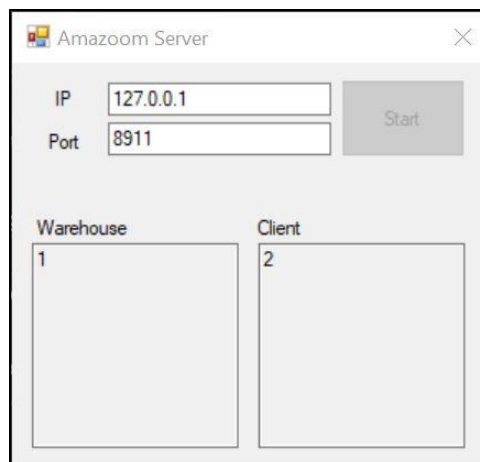
The screenshot shows a window titled "Amazoom Order Form". It contains input fields for "ID" (value 2), "IP" (value 127.0.0.1), and "Port" (value 8911). Below these is a "Connect To Server" button. A section titled "Please place your order here" contains "Product" (value Mango) and "Qty" (value 500) fields, followed by an "Order Product" button. Below this is a table titled "Items Available" with columns "Name" and "Quantity". The table lists Mango (500), Melon (55), and Banana (833). At the bottom is a table titled "Order Status" with columns "Id", "Product", "Quantity", and "Status". It shows a single order with Id 2000, Product Mango, Quantity 500, and Status Accepted. A footer text reads "Go to Settings to activate Windows."

Name	Quantity
Mango	500
Melon	55
Banana	833

Id	Product	Quantity	Status
2000	Mango	500	Accepted

Figure 3: Customer Interface

The interface for the server is shown in Fig. 4. Through this interface, the admin can set up the server. Moreover, the admin can observe which warehouses and clients are currently connected to this server.



The screenshot shows a window titled "Amazoom Server". It contains input fields for "IP" (value 127.0.0.1) and "Port" (value 8911), followed by a "Start" button. Below these are two large rectangular areas labeled "Warehouse" and "Client". The "Warehouse" area contains the number 1, and the "Client" area contains the number 2.

Figure 4: Server Interface

The manager interface for the warehouse is shown in Fig. 5 . When the warehouse is started, the manager can choose the dimension of the warehouse, the number of docks and the number of robots. The manager can also choose a file to load the initial inventory. After the warehouse is started, the manager can connect to the server.

Within the 'Robots' table, the manager can observe the IDs of the robots deployed, which products are being carried by each robot and how many of each product. The manager can also observe the statuses of the robots through their colors. White indicates the robot is inactive; red indicates the robot is en route to picking up product (from truck/shelf); yellow indicates the robot is en route to dropping off product (to truck/shelf); green indicates the robot is going back home. Besides, the manager can add or remove robots depending on the number of orders being processed.

Within the 'Inventory' table, the manager can observe the products available along with their estimated and actual quantities. The difference between estimated and actual quantities is that the former describes the products being ordered/delivered but have not reached the warehouse/customer while the latter describes the products currently on the shelves. The manager can also observe the statuses of the products through their colors. Green indicates the estimated quantity matches the actual quantity; red indicates estimated quantity is more than actual quantity (product is scheduled for restocking); yellow indicates the estimated quantity is less than actual quantity (product is scheduled for delivery). Also, the manager can add or remove products within the inventory.

Within the 'Orders' table, the manager can observe the IDs of all of the previous orders, the statuses of each order (Denied, Accepted, In Progress, Delivered), which products are ordered and how many of each product. The difference between Accepted and In Progress statuses is that the former describes that a delivery truck is scheduled for docking while the latter describes that a robot is dispatched for carrying the order. For testing purposes, the manager can submit a random order to verify the functionality of the warehouse.

Within the map, the manager can observe the location of active robots and the location of docks. The manager can also observe the type of the robots through their colors. Red indicates the robot is assigned for restocking while orange indicates the robot is assigned for delivery. The robot moves in the clockwise direction to prevent collision. The top-left location of the map is the home for the robots (imagine there is a door on the left of the home cell that opens to the parking space of the robots). The robot begins from home, picks up product from the truck or shelf, drops off product to the truck or shelf and goes back to home.

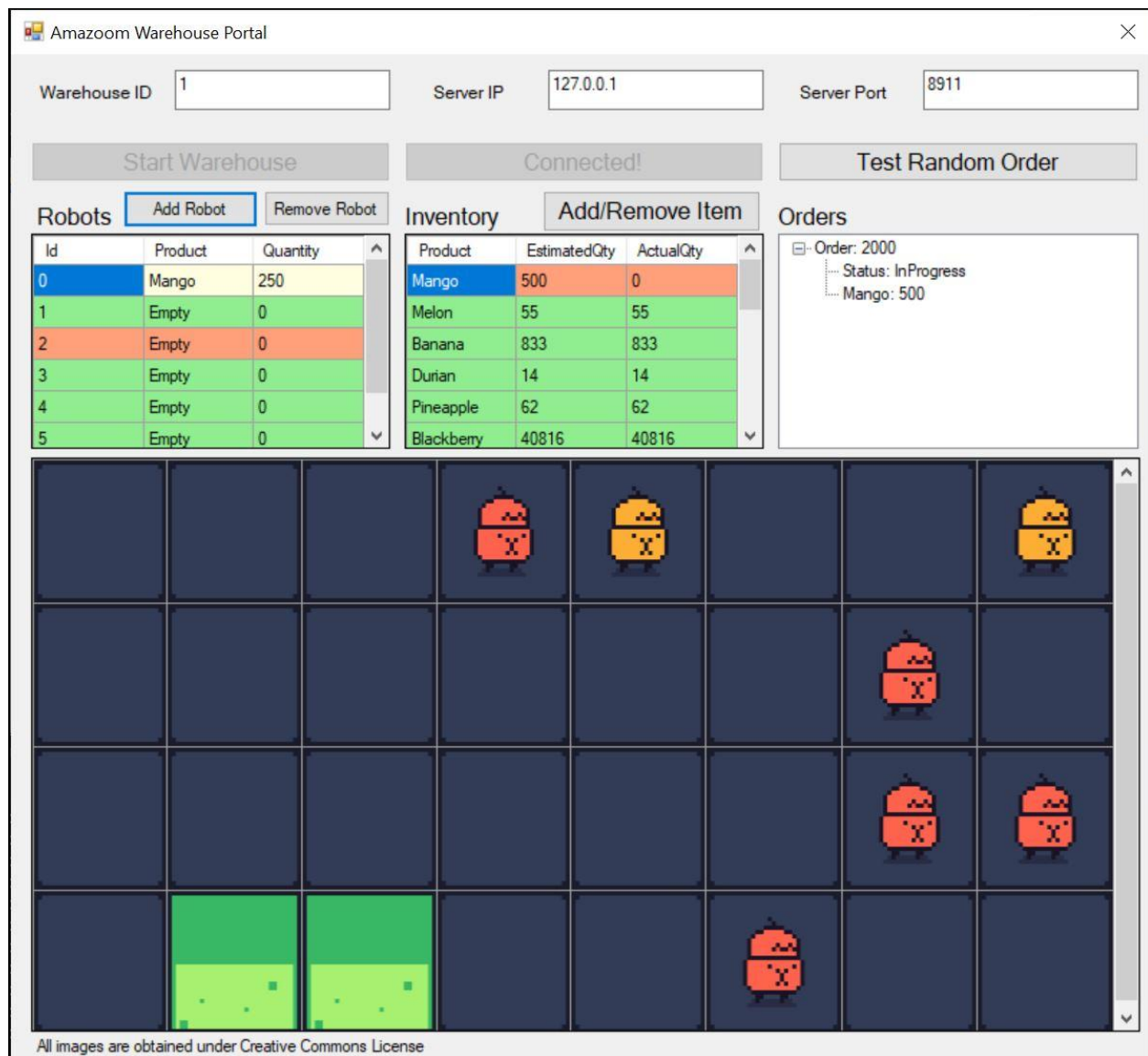


Figure 5: Warehouse Interface

Technology Stack

All codes are written in C# with the .NET framework. The .NET framework can be used to develop codes for a variety of applications including systems we developed for the Amazoom automation system. It can also run in multiple OS so it is highly flexible.

SimpleTCP was used for server/client socket communication. One of the reasons for choosing SimpleTCP over ASP.NET or other framework is that the team was more familiar with SimpleTCP and given the time constraint, the team decided to pursue working with SimpleTCP. Furthermore, by working with lower-level socket communication code with SimpleTCP, the team decided that this will allow the team to modify the code with higher flexibility.

Windows Form was used to design the user interfaces for the manager and the customers. It is very intuitive to use and design a UI with Windows Form and because all team members had experience using it, the team decided to use Windows Form.

Communication Protocol

The communication between warehouse, server and client (customer) is achieved through TCP/IP. In our case, the TCP/IP server is the server while the TCP/IP client is the warehouse and client. Using simpleTCP package, we can use C# String to communicate between warehouse, server and client. To differentiate different types of message send, we develop a communication protocol with the message format as shown.

“Command/Sender/Receiver/Payload\r”

where,

Command: type of payload
Sender: sender of payload
Receiver: receiver of payload
Payload: content of message

Whenever a field is not required, we use ‘-’ to denote such a field as null. Using this communication protocol, the warehouse and client can send messages to the server. The server will interpret the message and then broadcast a new message to all of its TCP clients (both warehouse and client). All of the TCP clients filter the message based on its receiver to decide who should interpret the message. The sequence diagram for the communication between warehouse, server and client is shown in Fig. 6 with Table 1 describing each of the communication protocols.

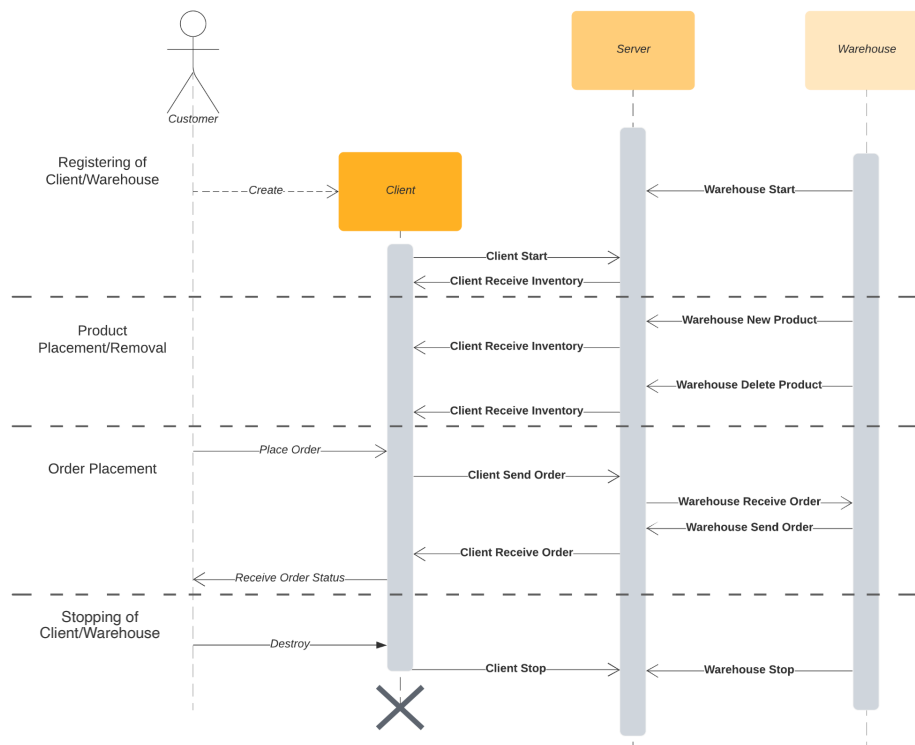


Figure 6: Sequence Diagram for Communication between Warehouse, Server and Client

Table 1: Communication protocol

Title	Direction	Format
Client Start	Client->Server	" ClientStart /[Client ID]/-/" Server registers new client. Client receives inventory from server through 'Client Receive Inventory'.
Warehouse Start	Warehouse->Server	" WarehouseStart /[Warehouse ID]/-/Product*Quantity,...," Server registers new warehouse and associated warehouse inventory. Clients receive updated inventory from the server through 'Client Receive Inventory'.
Client Receive Inventory	Server->All	" InventoryToClient /-/Product*Quantity,...," All clients receive inventory from the server whenever a warehouse is added/removed or product is added/removed within a warehouse.
Warehouse New Product	Warehouse->Server	" WarehouseNewProduct /[Warehouse ID]/-/Product*Quantity" Server receives added product from warehouse. Clients receive updated inventory from the server through 'Client Receive Inventory'.
Warehouse Delete Product	Warehouse->Server	" WarehouseDeleteProduct /[Warehouse ID]/-/Product*Quantity" Server receives deleted product from warehouse. Clients receive updated inventory from the server through 'Client Receive Inventory'.
Client Delete Product	Server->All	" ClientDeleteProduct /-/Product*Quantity" All clients receive deleted products from the server.
Client Send Order	Client->Server	" OrderFromClient /[Order ID]/-/Product*Quantity" Server receives a new order from the client which will be dispatched to the appropriate warehouse based on the ordered product.
Warehouse Receive Order	Server->All	" OrderToWarehouse /[Order ID]/[Warehouse ID]/Product*Quantity" Warehouse receives a new order from the server where the ordered product exists in the inventory of the warehouse.
Warehouse Send Order	Warehouse->Server	" OrderFromWarehouse /-/[Order ID]/Status" Server receives order status from the warehouse which will be dispatched to all of the clients.
Client Receive Order	Server->All	" OrderToClient /-/[Order ID]/Status" Client receives order status from the server.
Client Stop	Client->Server	" ClientStop /[Client ID]/-/" Server unregisters client.
Warehouse Stop	Warehouse->Server	" WarehouseStop /[Warehouse ID]/-/" Server unregisters warehouse. Clients receive updated inventory from the server through 'Client Receive Inventory'.

Warehouse Logic

The logic for delivery and restocking in the warehouse is shown in Fig. 7 and Fig. 8 respectively.

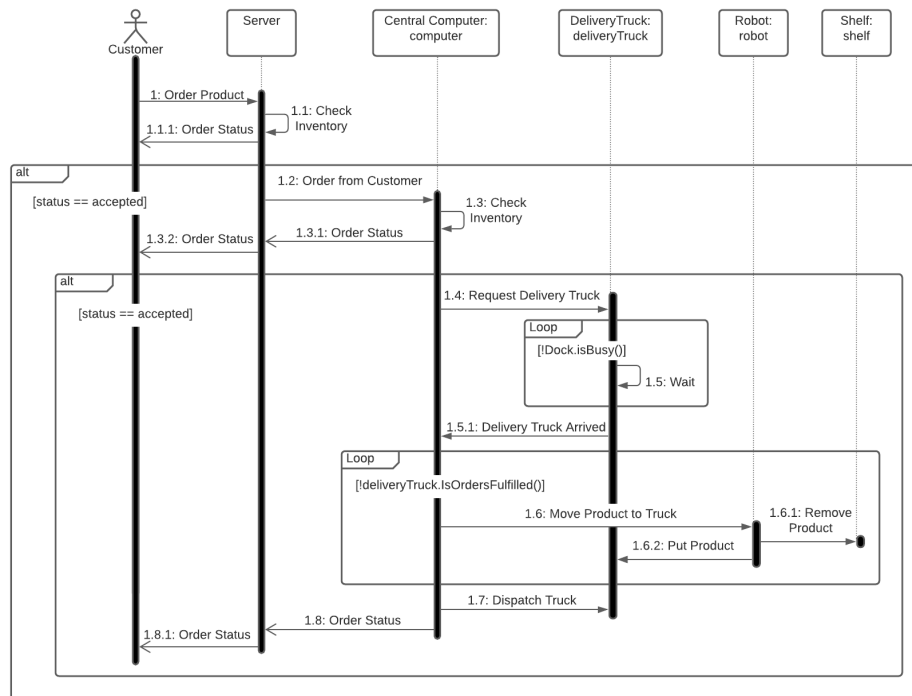


Figure 7: Sequence Diagram for Delivery

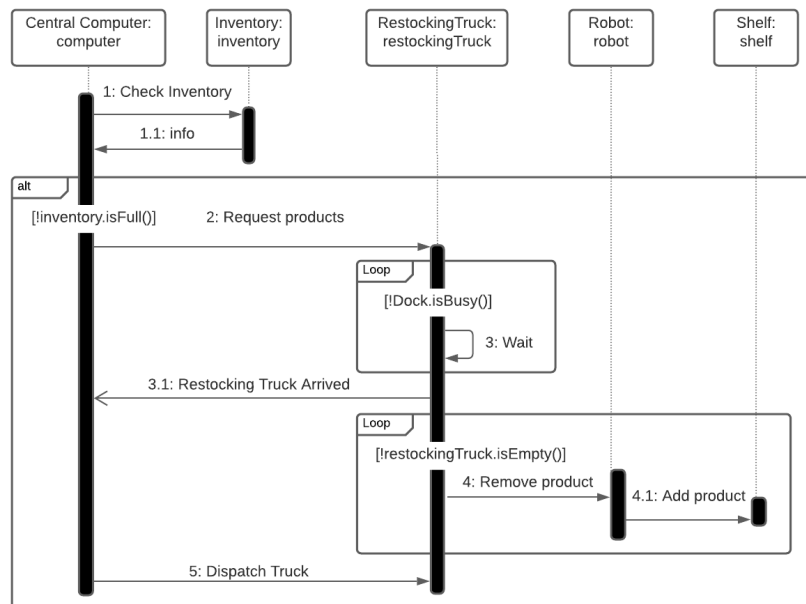


Figure 8: Sequence Diagram for Restocking

Robot Logic

The logic for robots in the warehouse is shown in Fig. 9.

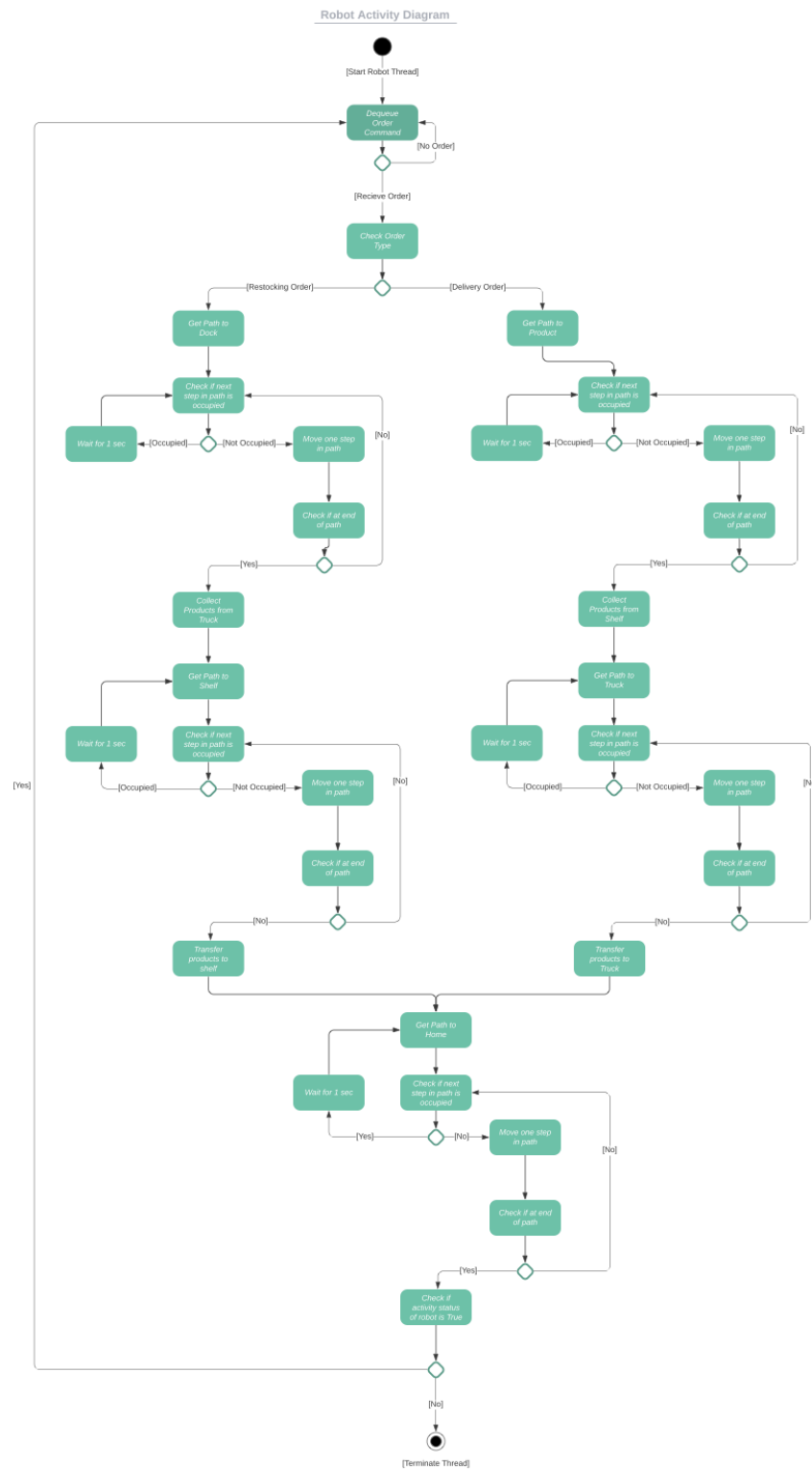


Figure 9: Activity Diagram for Robot in Warehouse

Class Diagram

The class diagram for the warehouse is shown in Fig. 10.

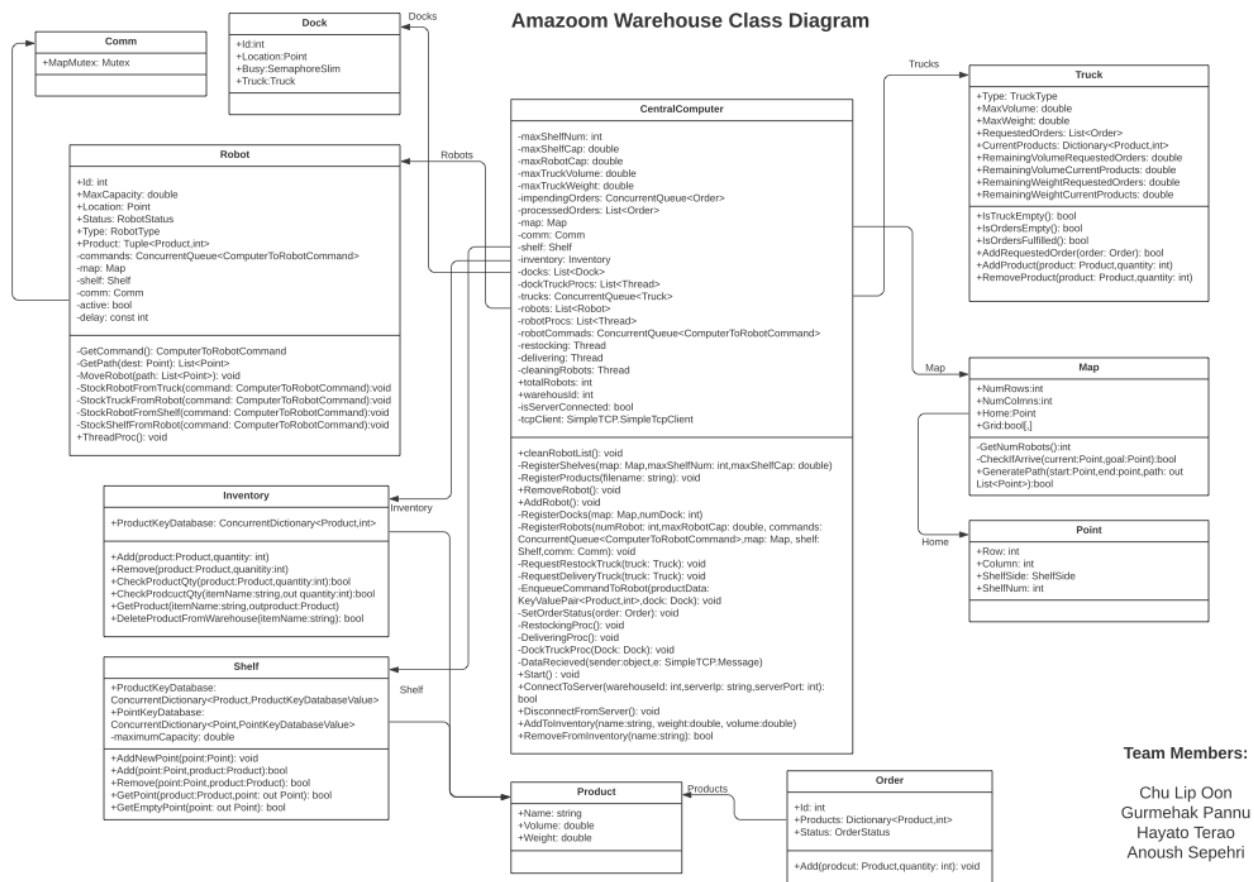


Figure 10: Class Diagram for Warehouse

Test Plan

Due to time constraint, the team decided to perform system-level integration testing instead of module-level unit testing as we predict most errors would occur during integration. The test cases for warehouse, server and client are described in Tables 2, 3 and 4 respectively.

Table 2: Test Cases for Warehouse Functions

ID	Objective and Procedures	Status
1.1	<p><i>Warehouse Initialization</i></p> <ul style="list-style-type: none"> A. Start the warehouse with the following parameters: <ul style="list-style-type: none"> a. Size x: 3, Size y: 3, Number of docks: 1, Number of robots: 1 B. Start the warehouse with inventory named 'testProduct.txt'. C. Verify the warehouse generated is accurate. D. Verify the inventory is fully restocked after a while. E. Restart the warehouse with the following parameters: <ul style="list-style-type: none"> a. Size x: 8, Size y: 4, Number of docks: 2, Number of robots: 4 F. Start the warehouse with inventory named 'testProduct.txt'. G. Verify the warehouse generated is accurate. H. Verify the inventory is fully restocked after a while. 	Passed December 7 2021
1.2	<p><i>Inventory Initialization and Restocking Process</i></p> <ul style="list-style-type: none"> A. Continue from 1.1.E, start the warehouse with the inventory named 'fruitsProduct.txt' with 19 different types of products of various weights. B. Verify the inventory generated is accurate. C. Verify the inventory is initially empty after the warehouse is started. D. Verify the inventory is fully restocked after a while. 	Passed December 7 2021
1.3	<p><i>Ordering Process</i></p> <ul style="list-style-type: none"> A. Continue from 1.2.D, press the 'Test Random Order' button which generates an order with a random number of products with a random number of quantities. Such an order may be 'Denied' due to exceeding the weight of the truck. Repeat until such order is 'Accepted'. B. Verify the quantities of the products within the inventory are reduced. C. Verify the order is 'InProgress' then 'Delivered'. D. Verify the inventory is fully restocked after a while. 	Passed December 7 2021
1.4	<p><i>Add and Remove Robot</i></p> <ul style="list-style-type: none"> A. Continue from 1.2.D, increase the number of robots from 4 to 6. B. Verify the number of robots is accurate. C. Press the 'Test Random Order' button to generate an order. D. During ordering and restocking, increase the number of robots from 6 to 8 and back to 6. E. Verify the number of robots is accurate during this process. F. After the robot has fully restocked the inventory and is inactive, decrease the number of robots from 6 to 4. G. Verify the number of robots is accurate. 	Passed December 7 2021
1.5	<p><i>Add and Remove Item</i></p> <ul style="list-style-type: none"> A. Continue from 1.2.D, add a new product with the following parameter, <ul style="list-style-type: none"> a. Name: MiracleBerry, Weight: 0.1kg, Volume: 0.01m3 B. Verify this product is added to the inventory and has initial quantity of zero. C. Verify this product is fully restocked after a while. D. Remove this product from the inventory. E. Verify this product is removed from the inventory. 	Passed December 7 2021

Table 3: Test Cases for Server Functions

ID	Objective and Procedure	Status
2.1	<p><i>Warehouse Registration</i></p> <ul style="list-style-type: none"> A. Start the server with the following parameters: <ul style="list-style-type: none"> a. IP: 127.0.0.1, Port: 8911 B. Start a warehouse with the following parameters: <ul style="list-style-type: none"> a. Size x: 8, Size y: 4, Number of docks: 2, Number of robots: 4 b. Start the warehouse with inventory named 'fruitsProduct.txt'. C. Connect the warehouse to the server with ID of 1. D. Verify the warehouse is registered by the server. E. Start another warehouse with the following parameters: <ul style="list-style-type: none"> a. Size x: 6, Size y: 3, Number of docks: 1, Number of robots: 2 b. Start the warehouse with inventory named 'testProduct.txt'. F. Connect the warehouse to the server of ID of 2. G. Verify the warehouse is registered by the server. H. Closes the warehouses. I. Verify the warehouses are no longer registered by the server. 	Passed December 7 2021
2.2	<p><i>Client Registration</i></p> <ul style="list-style-type: none"> A. Continue from 2.1.1G, start and connect a client to the server with ID of 1. B. Verify the client is registered by the server. C. Start and connect another client to the server of ID of 2. D. Verify the client is registered by the server. E. Closes the clients. F. Verify the clients are no longer registered by the server. 	Passed December 7 2021

Table 4: Test Cases for Client Functions

ID	Objective and Procedure	Status
3.1	<p><i>Inventory Display</i></p> <ul style="list-style-type: none"> A. Continue from 2.2.D, we have two warehouses with unique products and two clients connected to the server. B. Verify both the clients display combined inventory information from both of the warehouses 	Passed December 7 2021
3.2	<p><i>Ordering Process</i></p> <ul style="list-style-type: none"> A. Continue from 3.1.B, order the following product from Client 1. <ul style="list-style-type: none"> a. Name: Mango, Quantity: 500 (Warehouse 1) B. Verify the order status is 'Accepted', 'InProgress' and then 'Delivered'. C. Ensure Client 2 does not receive any order statuses. D. Order the following product from Client 2. <ul style="list-style-type: none"> a. Name: FryingPan, Quantity: 27 (Warehouse 2) E. Verify the order status is 'Accepted', 'InProgress' and then 'Delivered'. F. Ensure Client 1 does not receive any order statuses. G. Repeat this process again after both of the warehouses have fully restocked. 	Passed December 7 2021