DECEMBER 8, 2021

AMAZOOM : AUTOMATED WAREHOUSE

CPEN 333 FINAL PROJECT

BOBBY SMITH, DAVID CHUNG, MATHEW PITHER, NICK VO

Executive Summary

Amazoom

Purpose of the Project

Amazoom warehouse real-time simulation system is designed to create a fully autonomous warehouse system.

The purposes of this project are as follows:

- Create a automated warehouse design
    - The central warehouse computer
    - The user-interface for the computer
    - The automation robots
    - The remote webserver that places orders
    - Delivery and restocking trucks

Method Used

Visual Studio has designed all autonomous system software.

The central warehouse computer is connected to the robots, delivery and restocking trucks, and a server that receives orders. The central computer automatically updates all the information. It allows the user to query the status of an order, check the number in stock of an item and get alerts about low-stock items.

The robots move around the warehouse to load items into trucks or stock in the warehouse. Each robot has a collision detection system that prevents colliding with robots to deliver items to a designated location safely.

Delivery and restocking trucks are parked at a specific location in the warehouse and load or unload the items. They report their arrival to the central computer and wait until the dock is open before they do their work.

Findings and Recommendation

Cost-effective and time-efficient suggestions include reaching out depending on the number of robots per the warehouse. When there were too many robots compared to loading dock size and warehouse size, the efficiency decreased. Our team recommends limiting the number of robots to get maximum efficiency in time and cost. Our team created a website for all UML, giving readers a more precise image.

Tech stack

- Visual Studio
    - Used for all coding
    - A consistent program for coding helped the user to understand the code and minimized possible bugs

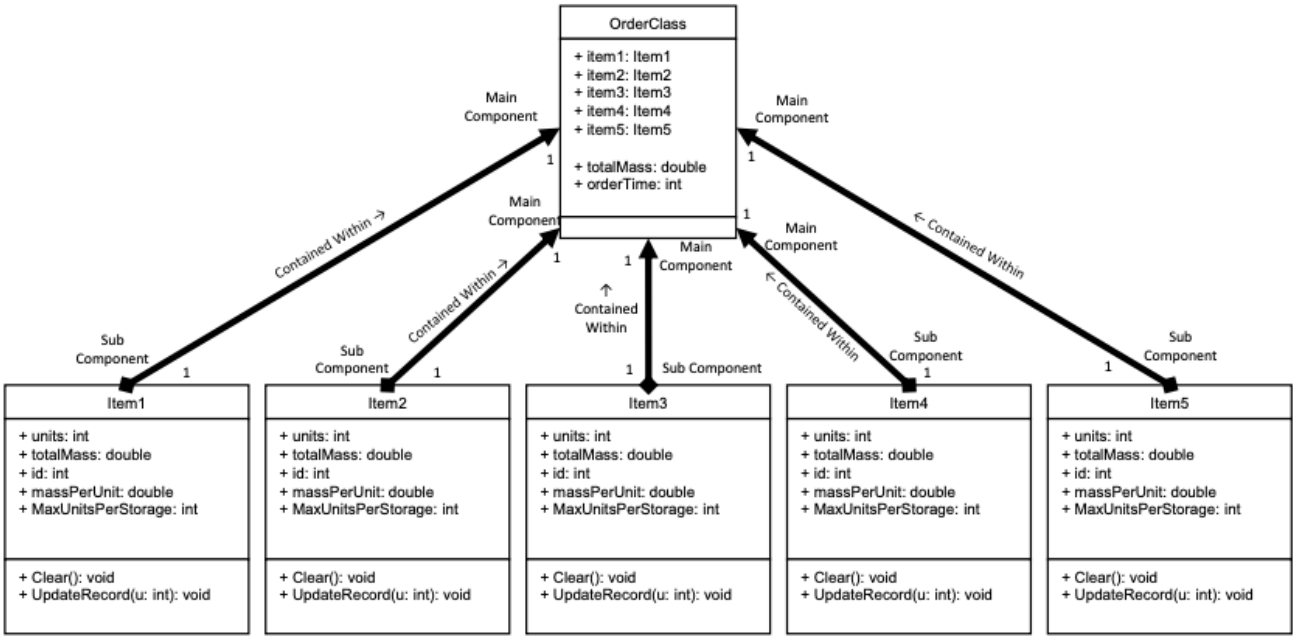
- UML Editor and Atom
    - Used for UML design website
    - Clear visual in one website to help understand and viewing the project design

UML

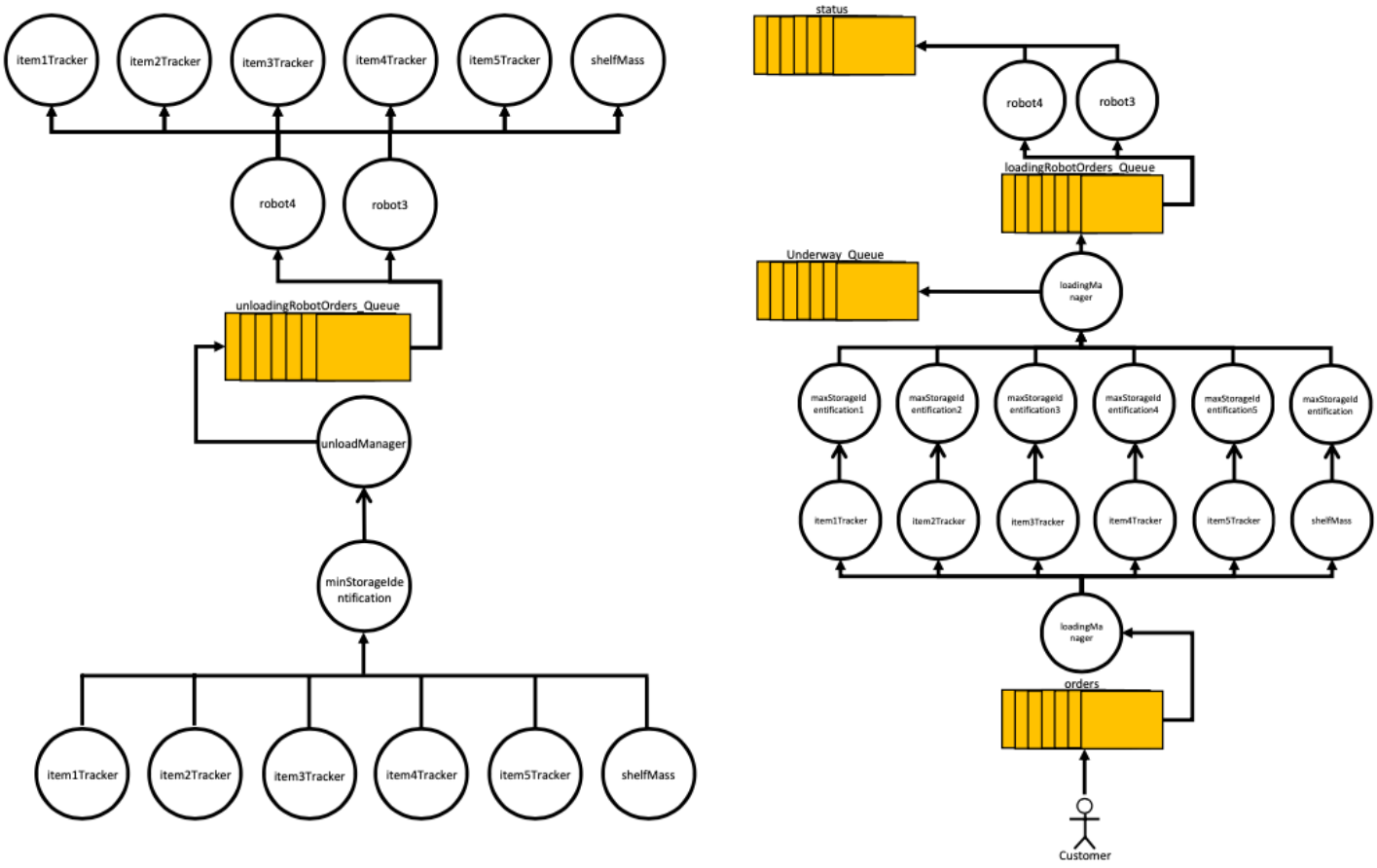
Our team created a website for all UML, giving readers a more precise image.

The link is https://nk-vo.github.io/Final/

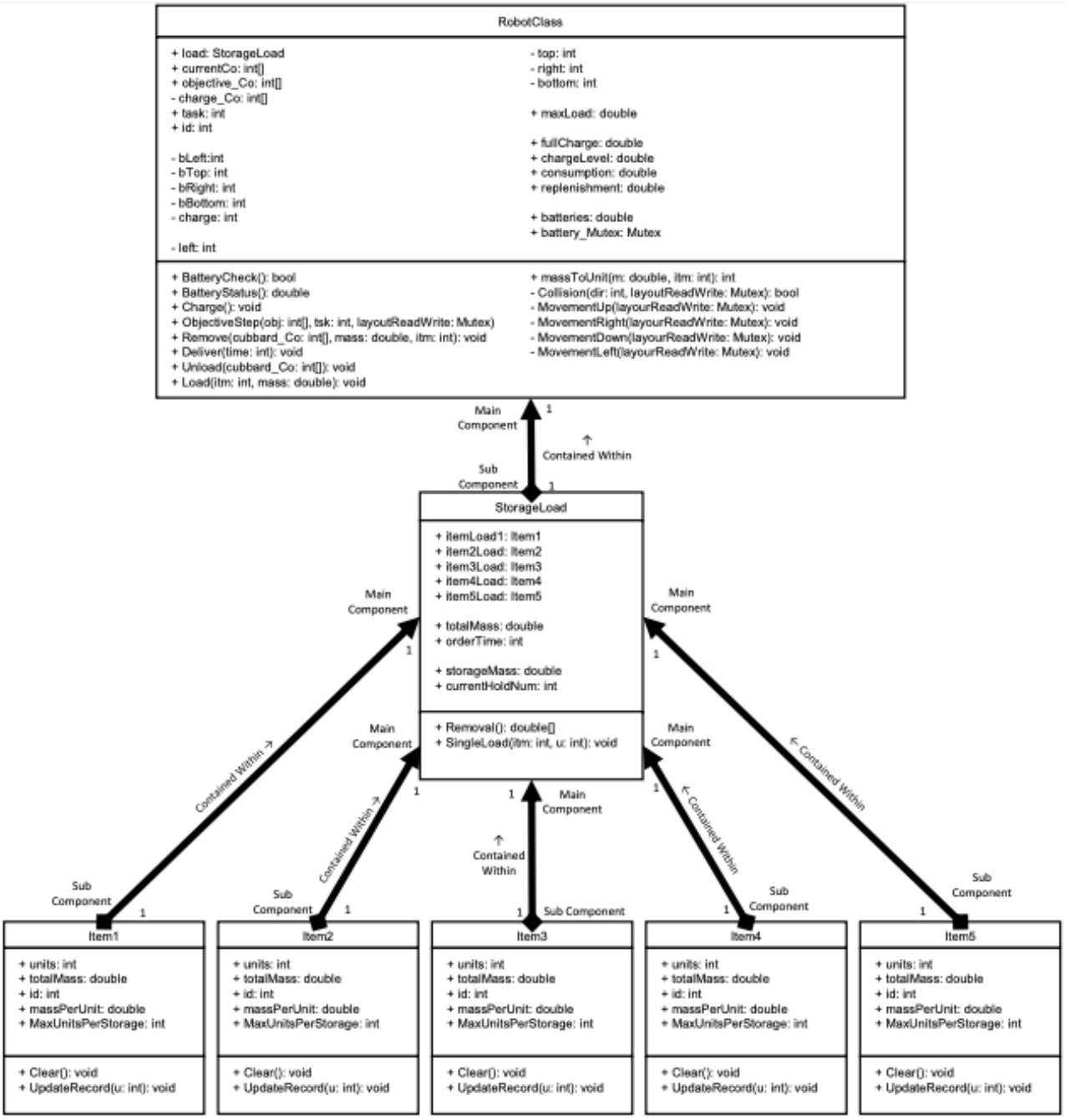
Order-Item Class Diagram



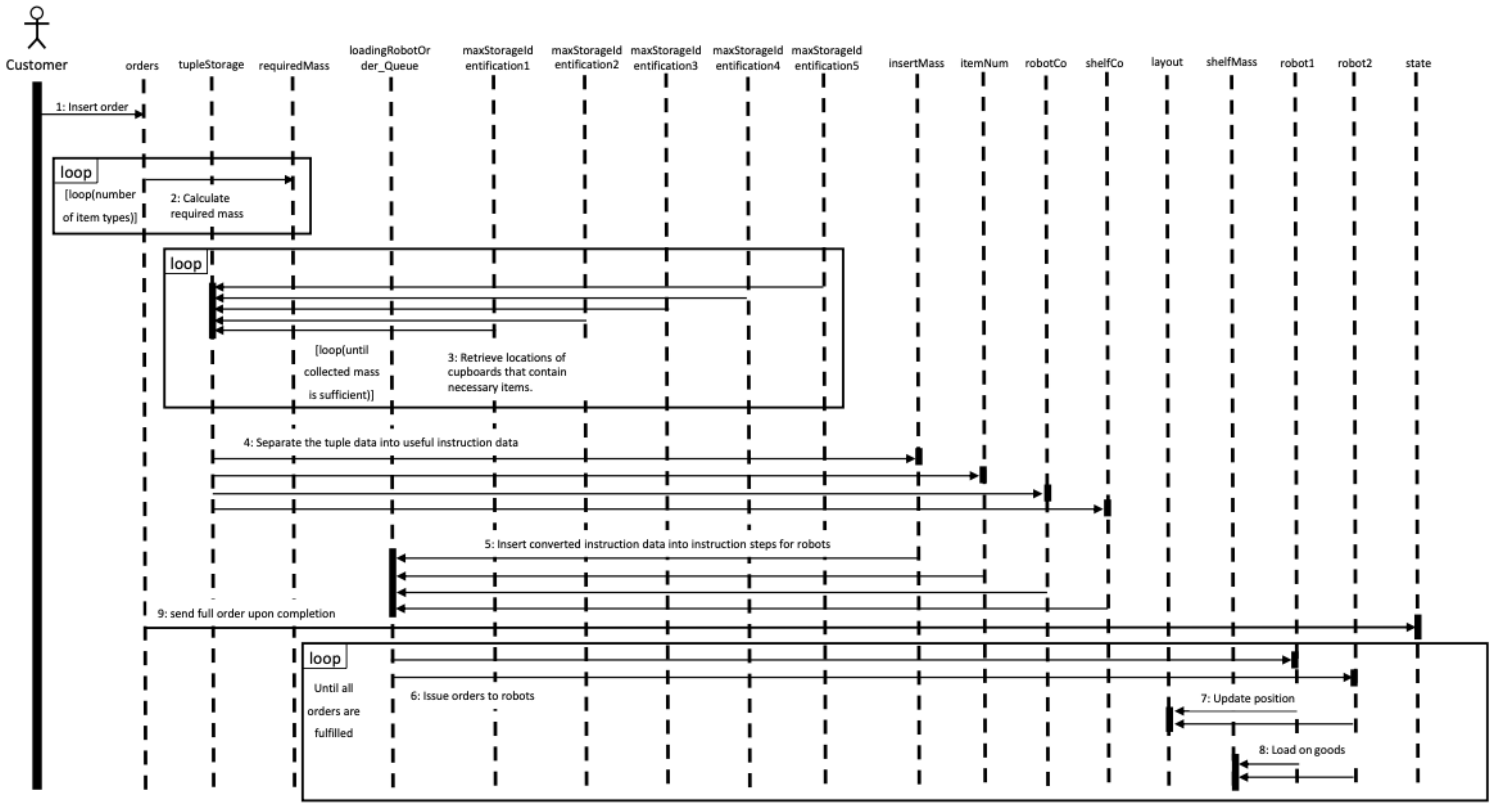
Robot-Item Object Interaction Diagram



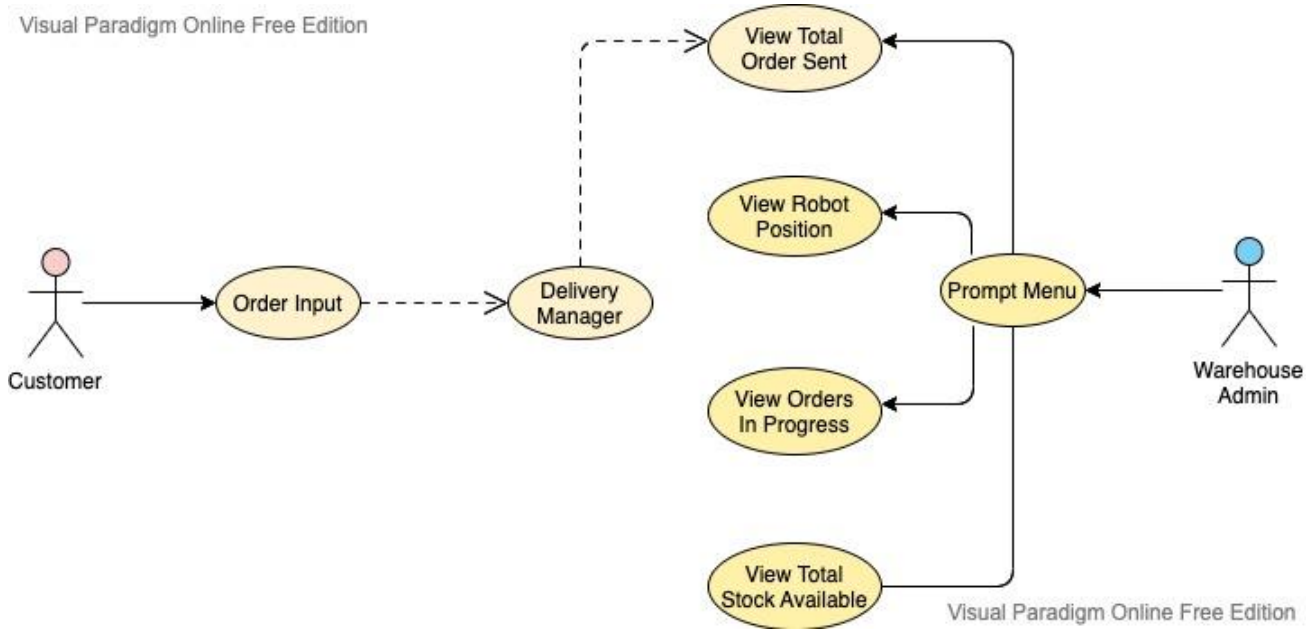
Robot-Storage-Item Class Diagram



Sequence Diagram



Use Case Structuring Template



User Story Diagram

Use-Case: Prompt Menu

Actor(s): Warehouse Administrator

Effects/Benefits:
Returns the program menu for various choices to be made.

1. Administrator opens program.
2. Administrator is prompted with menu of various functions.
3. If administrator wants to view the all the orders that have been fulfilled, then <<Scenario 1>>
4. If administrator wants to view all the orders that are in progress, then <<Scenario 2>>
5. If Administrator wants to view the layout of the warehouse and all the positions of the robots, then <<Scenario 3>>
6. If administrator wants to view the total stock of a specific item, then <<Scenario 4>>
7. Return to menu.
8. If administrator ends program: **end of Transaction**

Scenario 1: View Total Orders Sent
Scenario 2: View Orders In-Progress
Scenario 3: View Layout of Warehouse
Scenario 4: Total Stock of Warehouse

Use-Case: View Total Orders Sent

Actor(s): Warehouse Administrator

Effects/Benefits:
Returns the orders that have been sent out, the time-stamp of the completion, quantity of goods, time required to fulfil delivery.

1. Administrator is prompted with program menu.
2. Administrator enters command to view the orders that have been fulfilled at the warehouse.
3. Program returns information to administrator.
4. Administrator returns to main menu.

Use-Case: Input Order

Actor(s): Customer(s)

Effects/Benefits:
Customer inputs request for goods to be gathered and delivered from the warehouse.

1. Customer requests to input order.
2. Program returns query for customer to input quantity for each product.
3. Customer inputs quantity for each item.
4. Program returns a summary of the order, including the time required to deliver order and the total weight quantity of the order.
5. Order is relayed to the main queue or orders that need to be fulfilled.
6. Program returns to main menu.

Use-Case: View Orders In-Progress

Actor(s): Warehouse Administrator

Effects/Benefits:
Returns the orders that need to be fulfilled, the time needed to deliver goods, and their total quantity.

1. Administrator is prompted with program menu.
2. Administrator enters command to view the orders that need to be fulfilled at the warehouse.
3. Program returns information to administrator
4. Administrator returns to main menu.

Use-Case: View Layout of Warehouse

Actor(s): Warehouse Administrator

Effects/Benefits:
Generates an instantaneous view of the position of robots in warehouse

1. Administrator is prompted with program menu.
2. Administrator enters command to view the layout of the warehouse as well as the position of all the robots.
3. Program returns information to administrator
4. Administrator returns to main menu.

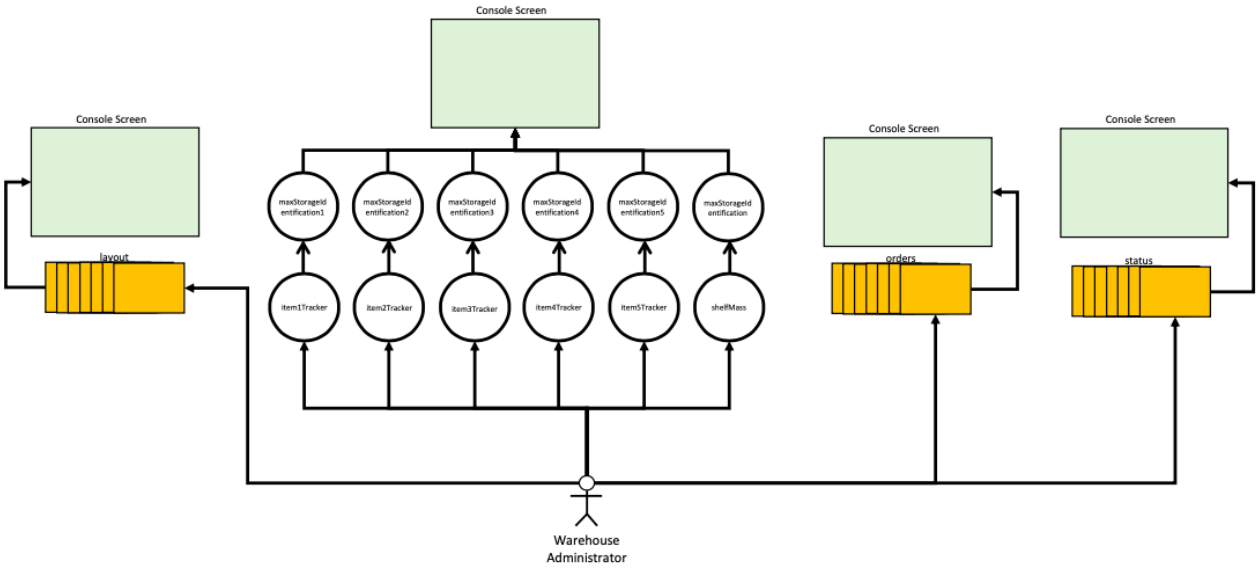
Use-Case: View Total Stock Available

Actor(s): Warehouse Administrator

Effects/Benefits:
Returns the total quantity of a specific stock on any shelf that contains the item.

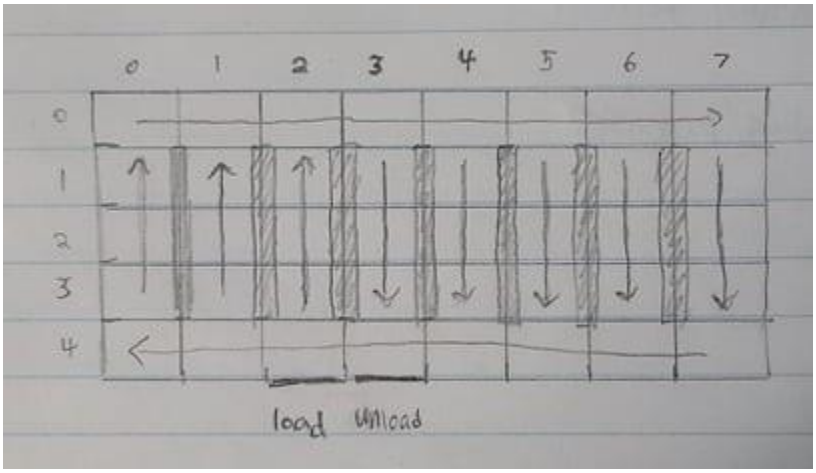
1. Administrator is prompted with program menu.
2. Administrator enters command specifying the quantity of each stock that they need to see.
3. Program returns information to administrator.
4. Administrator returns to main menu.

Warehouse Admin-Console Object Interaction Diagram



Justification for the design

The robot collision detection system makes the robot move in a clockwise direction on most occasions. In this method, the robot's efficiency may decrease in a small-sized warehouse. Still, the more straightforward method will provide less error and more efficiency in larger warehouses where more robots are moving in a larger area.



The arrow in the warehouse diagram shows the movement of robots.