University of British Columbia

Electrical and Computer Engineering

CPEN 333- Final Project

Project Design Document

Amazoom

Jake An

Ahmed Tameem

Zi Tan

Will Chaba

Table of Contents

1. Executive Summary

Amazoom, the world's largest internet based retailer, is showing interest in automating their warehouses to cut costs. Our firm specializes in designing robots for automation and presents to Amazoom an automatic warehouse system with friendly graphical user interfaces to assist warehouse managers.

Our system intends each warehouse to utilize its own central computer that will automatically assign trucks and robots to a new order once placed by a customer. The central computer is thread safe giving our system the advantage of using threads to handle multiple orders simultaneously. The warehouse central computer also keeps track of warehouse inventory and item locations via built in item and inventory manager classes.

Our firm has built a website to serve as the UI for warehouse managers to interact with. Our website uses the System.Runtime.Remoting namespace with underlying TCP protocol to communicate with each warehouse's central computer to retrieve important information for managers to know such as inventory lists and order statuses. The web server is a remote server capable of serving multiple warehouses. This way, warehouse management can be centralized to one remote web server instead of warehouses having their own individual server.

One of our key design goals is to incorporate flexibility in our system. We seek to achieve this goal by providing warehouse managers the tools to customize their warehouse through the website. From the website, warehouse managers will be able to add or remove robots and change the layout of the warehouse.

2. Design Philosophy

An agile development approach was taken for designing the system. First, a simple working system was designed and then more complex elements were added. Three sprints were completed for this project.

The first sprint's goals were building a backend with one robot that loads items onto one truck (unloading functionality was not implemented for this sprint).

The second sprint's goals were building a front-end prototype (that is not connected to the backend yet) and upgrading the back-end such that it can handle an arbitrary amount of robots and trucks (with both loading and unloading implemented).

The third sprint's goals were integrating the back-end with the finished front-end and implementing multi-threading. Moreover, the system was upgraded such that it can handle an arbitrary number of clients and warehouses.

3a. User Stories

The Central Computer (New order):

1. The user submits a new order to the computer
2. The system notify the web server if the order items are in the database
3. If the system cannot find the items in the database then <scenario 1>
4. The system adds order to bottom of the order queue
5. The system notifies the web server to update inventory list to reflect sale

Scenario 1:

- The system notify the web server to update the order cannot be fulfilled

The Central Computer (Restock):

1. The user notifies to the computer that a restock truck has arrived
2. The user notifies to the computer of the contents inside the truck
3. The system plans the route and location for new items
4. The system notify the web server to update inventory for addition of new items
5. If the restock truck still has contents inside then <scenario 1>
6. The user notifies to the computer that the restock truck has left
7. The system updates the location of items inside the warehouse because items were put in
8. The system updates that one loading bay spot is available

Scenario 1:

- Restock truck stays in loading bay

The Central Computer (Delivery):

1. The user notifies to the computer that a delivery truck has arrived
2. The system retrieves the order query from the web server
3. The system assigns orders starting from the top of the queue to the delivery truck
4. If the next order will surpass the delivery truck cargo limit then <scenario 1>
5. The user notifies to the computer that the delivery truck has left
6. The system updates that one loading spot is available
7. The system updates the location of items inside the warehouse because items were taken out
8. The system notifies the web server to update the order queue to remove the orders in the truck

Scenario 1:

- Check if the next order in the queue is able to fit inside the truck. Repeat until no order can fit into the truck.

The Central Computer (Warehouse Manager):

1. The user notifies the system to access the user interface
2. The systems notifies web server for item inventory and checks if any are low on stock
3. The system notifies the user if any items are low on stock
4. If the user wants to access the order status then <scenario 1>
5. If the user wants to access the item stock then <scenario 2>

Scenario 1:

- The system requests the web server for the order status

Scenario 2:

- The system requests the web server for the inventory status of the item.

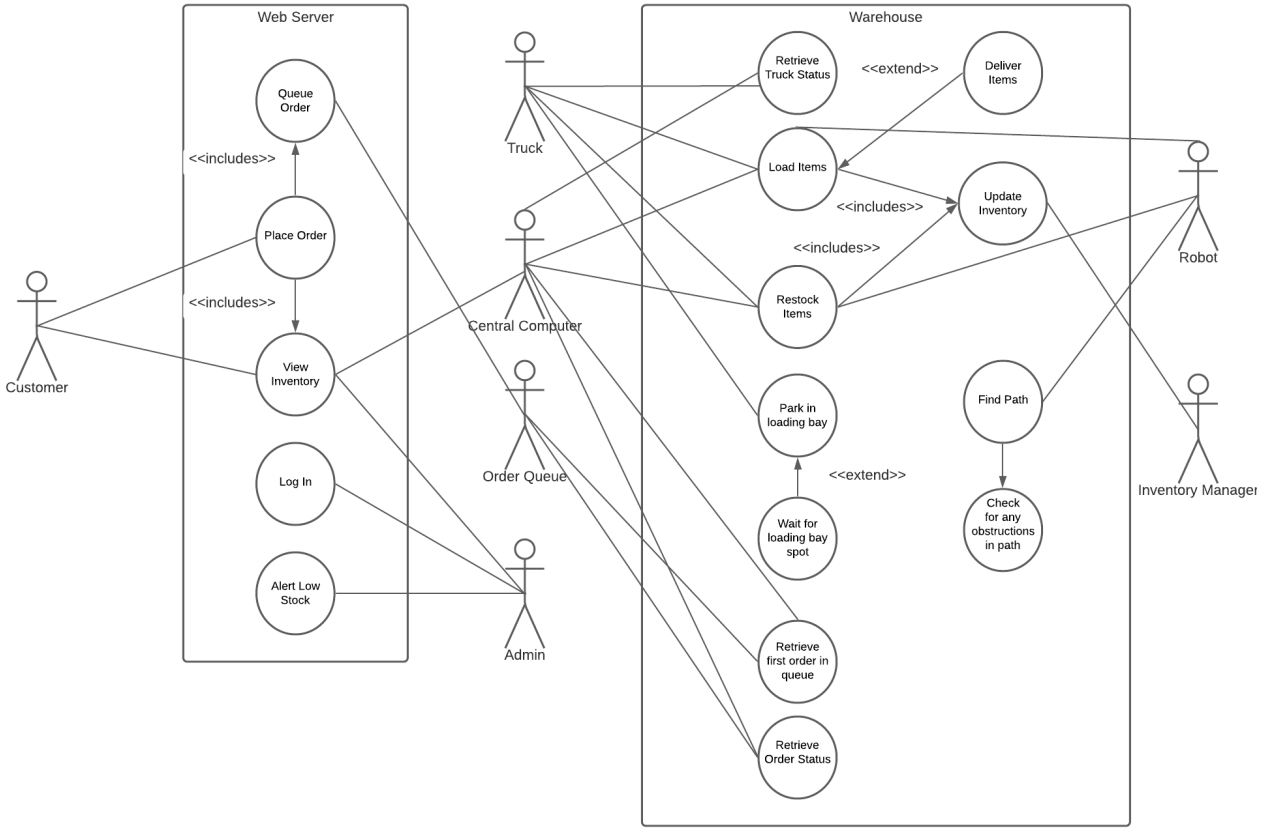
Delivery Truck:

1. The user notifies the system that the truck is ready to be docked
2. If there is no available dock space then <scenario 1>
3. The user notifies the system that the truck is docked and is ready for delivery/restock
4. The system notifies the truck that it can leave
5. The user notifies the system that the truck has left

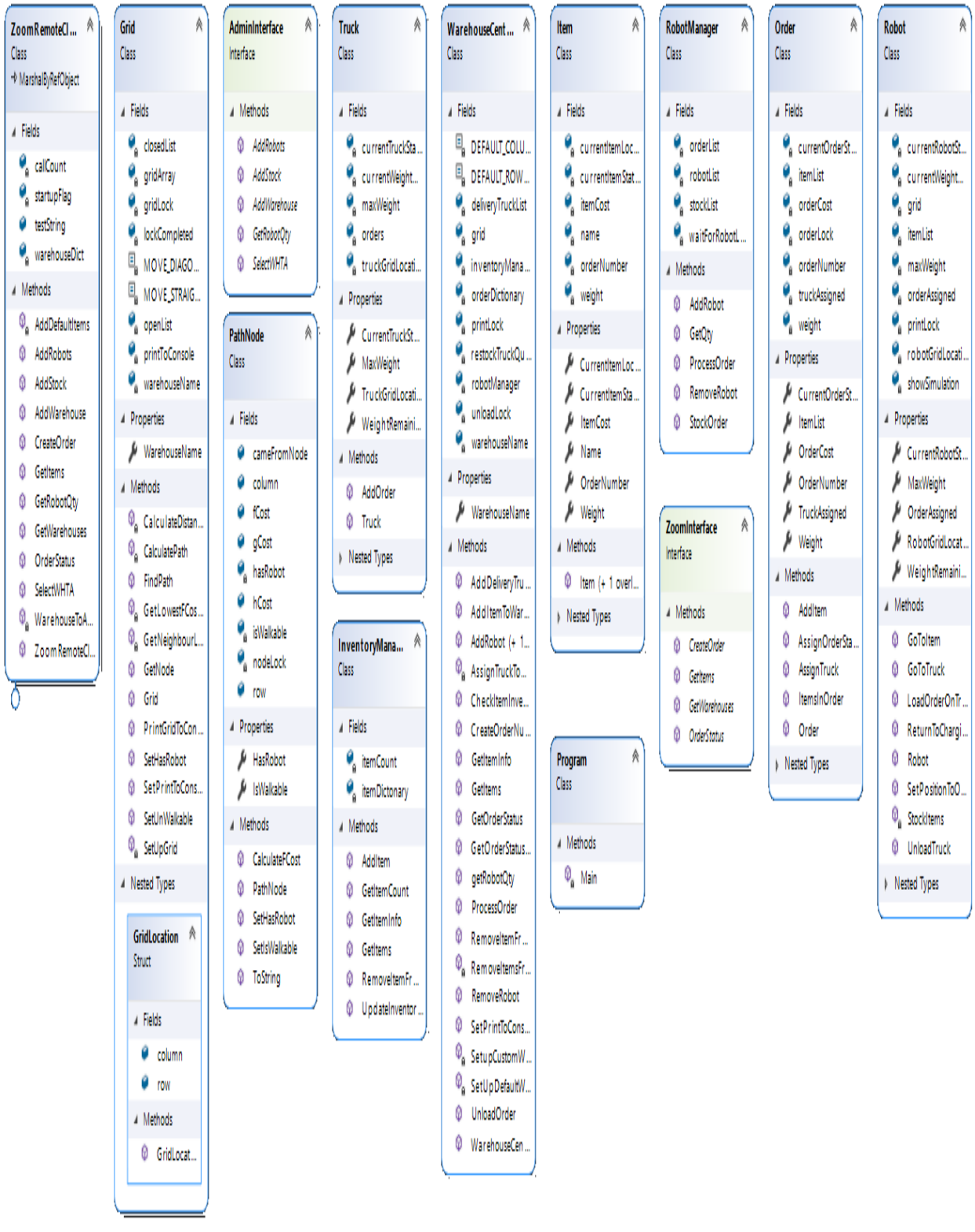
Scenario 1:

- Wait until dock space is available.

3b. Use Case Diagram

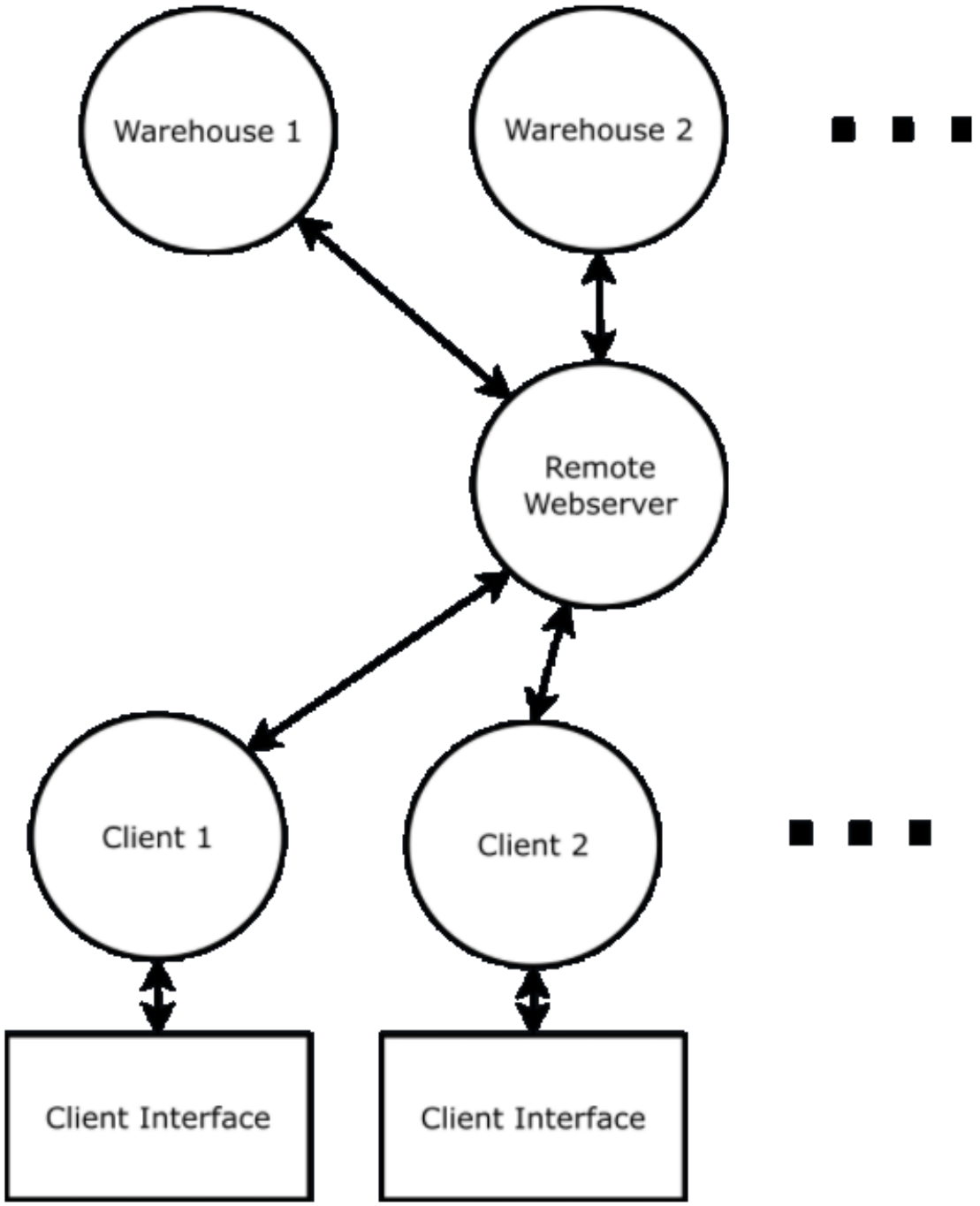


3c. Class Diagram

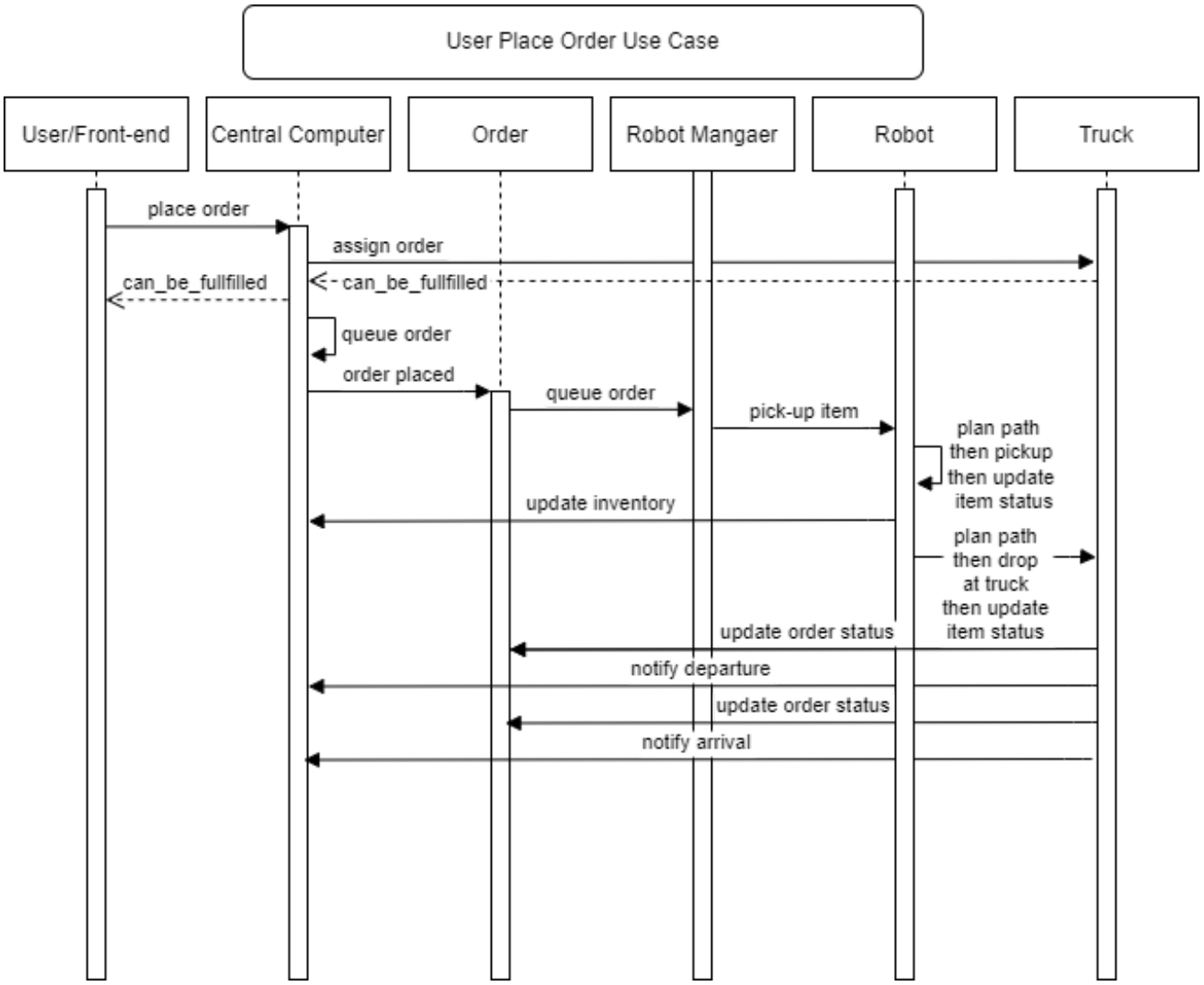


*See GitHub Repository for full quality Image

3d. Object Interaction Diagram



3e. Sequence Diagram


User Place Order Use Case
User/Front-end
Central Computer
Order
Robot Mangaer
Robot
Truck
place order
assign order
can_be_fullfilled
can_be_fullfilled
queue order
order placed
queue order
pick-up item
plan path
then pickup
then update
item status
update inventory
plan path
then drop
at truck
then update
item status
update order status
notify departure
update order status
notify arrival

4. Specifications

Front End:

- ASP.NET MVC framework
- System.Remoting.Remoting class with underlying TCP for communication
- Supports multiple clients
- Supports multiple warehouses
- Warehouse shapes can be specified by the warehouse admin

Back End:

- Written in C#
- Multi-threaded
- Object oriented approach
- Uses A* pathfinding
- Supports a variable number of robots