FURNITURE WAREHOUSE SIMULATION SOFTWARE

Software Design Specification

Members: Denisa Apostol
Amira Cruceru
Emese Engedi
Iana Florea
Bogdan Pavel
Leonard Tudorache

Tutor: Chung Kuah

Table of Contents

Introduction 2

User Interface Design 2

Database Schema 3

Sequence Diagram 4

Class Diagram 5

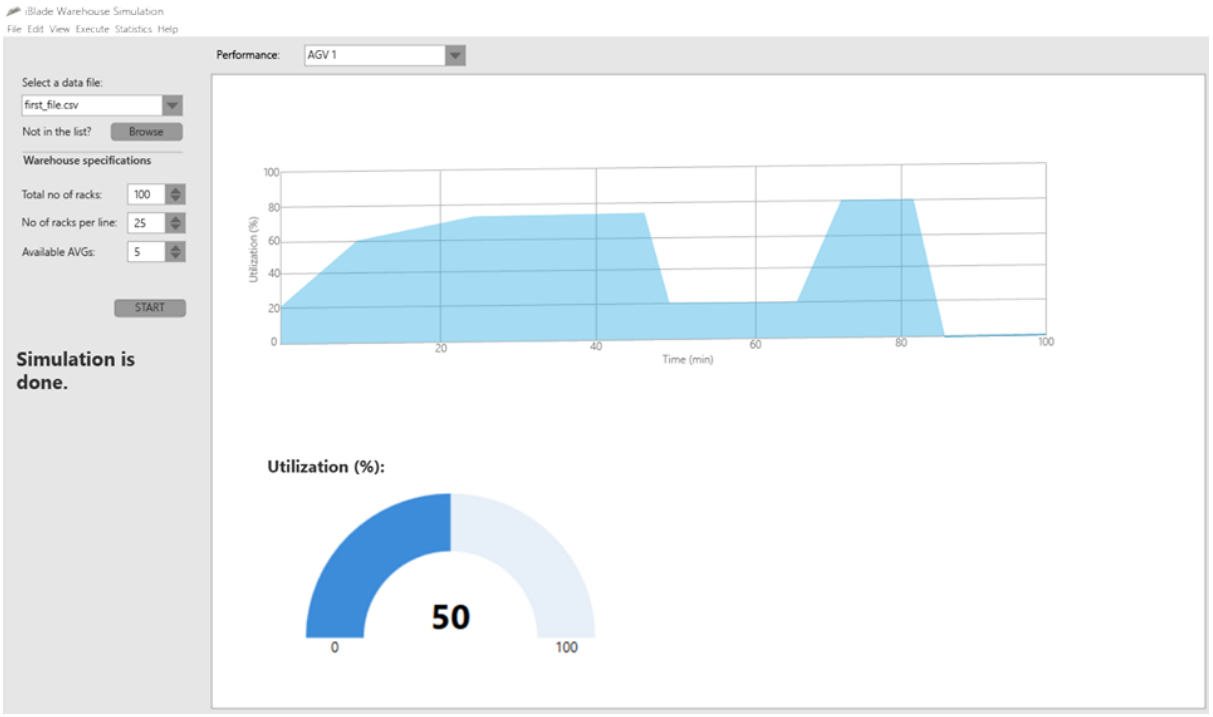
Introduction

The ‘iBlade Furniture Warehouse Simulation Software’ project is focused on creating a graphic simulation application, that will provide a 2D representation of a warehouse showing the objects and the outline of the premises. It will be possible for the user to change the layout of the warehouse, as well as the simulated orders even at run-time. The software helps to work out an optimal resource plan for a warehouse.

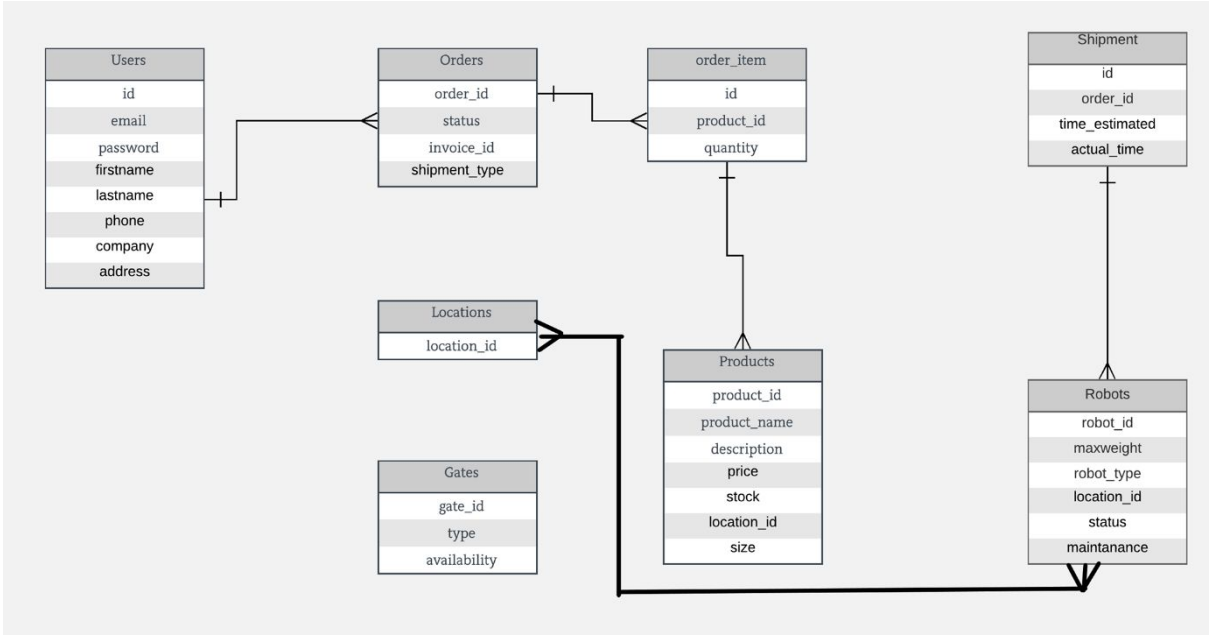
This design document details the models of technical implementation and provides an overview of the user interface. It is part of the documentation of the first iteration of the project - therefore far from being complete. All images seen here can also be found in the project’s GIT repository.

User Interface Design

The user interface will be realized in Windows Forms. We use user controls to provide modularity. The user will be able to navigate on the upper menu or through other controls, mostly placed on the left hand side of the software.

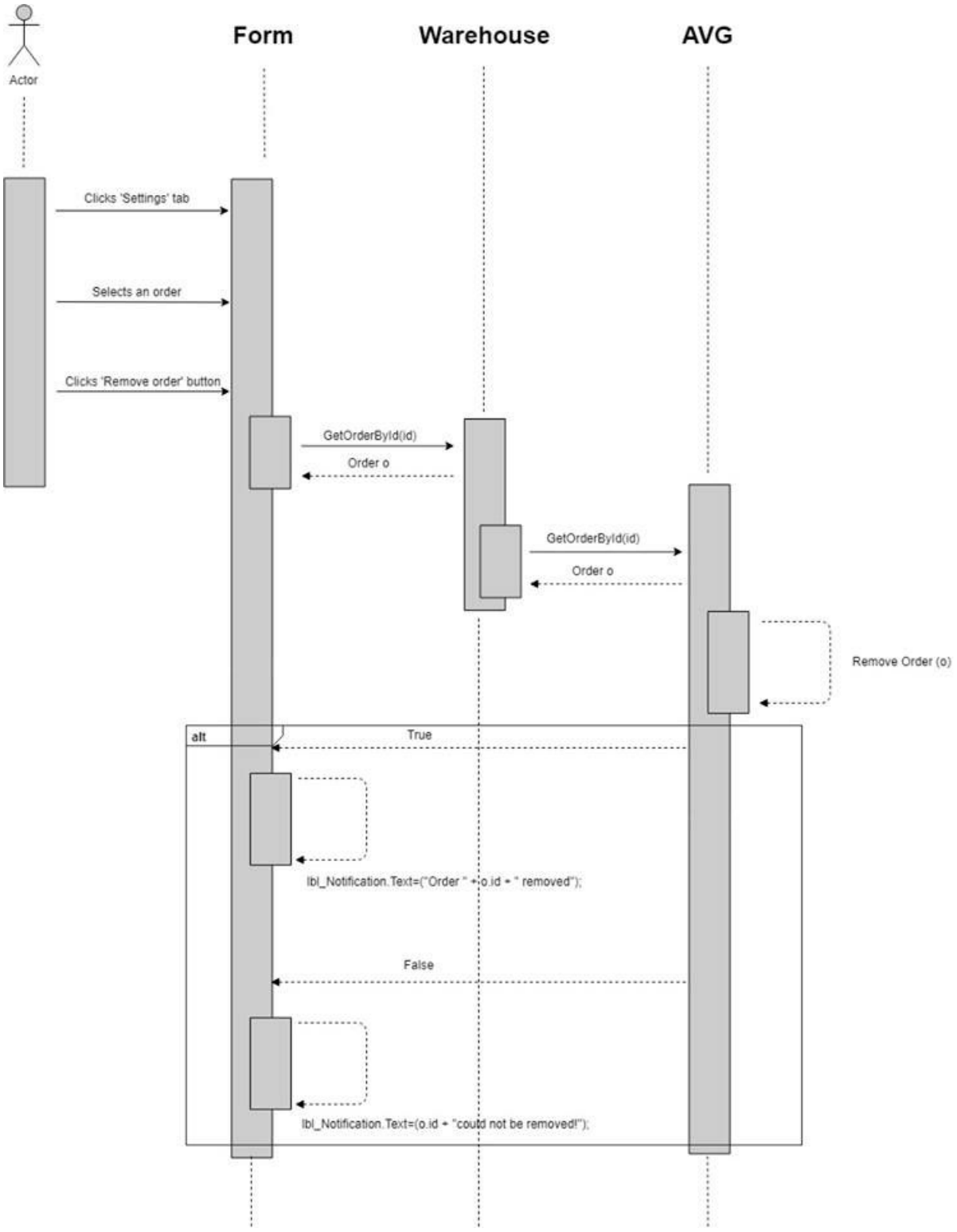


Database Schema



Sequence Diagram

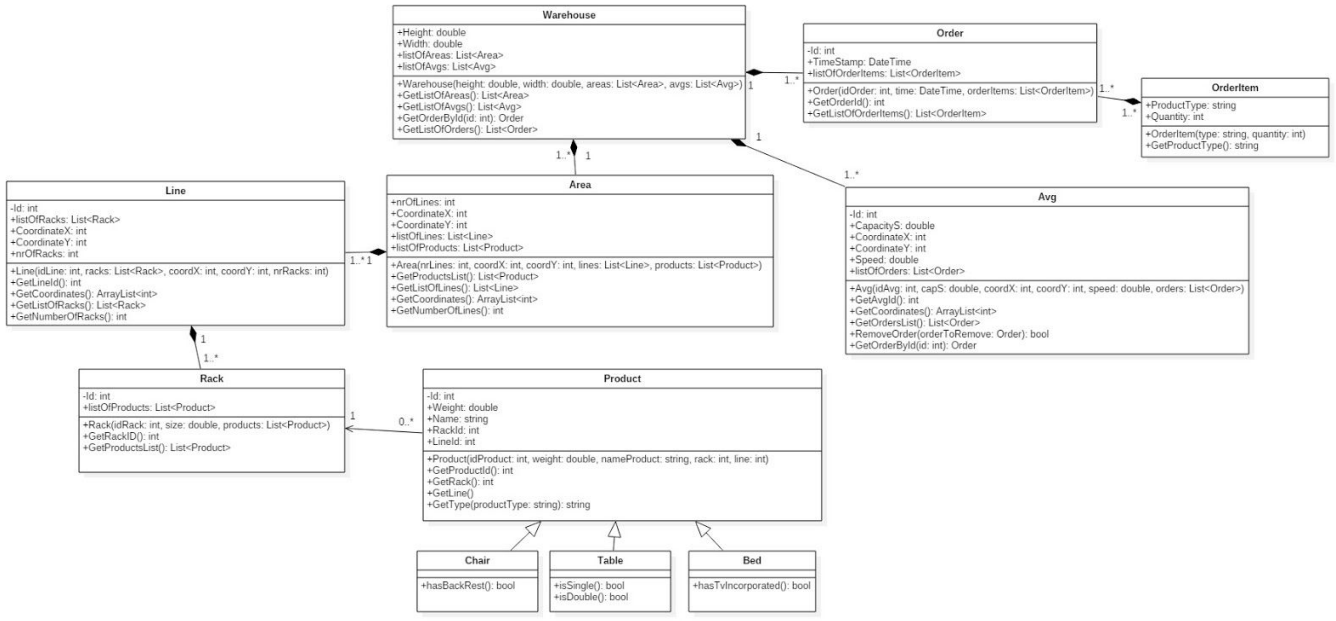
In this diagram, we show the process of removing a specific order from a AGV-s list while the simulation is running.



To remove an order the actor has to click the ‘Settings’, select an order and click ‘Remove order’ button. The form automatically accesses the ‘GetOrderById’ method (which has as parameter the id of the selected order) from the Warehouse class. To reach the AGV class, the Warehouse class uses the method ‘GetOrderById’. The AGV removes the Order object from its list of orders only if the order exists. In this case, the AGV returns True to the form. Automatically, the form shows in the lbl_Notification that the order was successfully removed. If the order does not exist, the AGV returns False to the form. Automatically, the form shows in the lbl_Notification that the order could not be removed.

Class Diagram

This class diagram shows the schematics of the classes and their relationships in the software.



Classes description

Line - Each line has an unique id, a position based on coordinates, a number of racks and a list of racks. The class has a constructor and methods to get the line id, coordinates, list of racks and number of racks.

Rack - Each rack has an unique id and a list of products from that racks. The class has a constructor and methods to get the rack id and the product list.

Warehouse - Each warehouse has a size (height, width), a list of areas of the warehouse and a list of AGV, a constructor and methods to get the list of areas, AGVs, to get the list of orders and search in the list an order, based on its id.

Area - Each area has a number of lines situated in that area, a position based on coordinates, a list of lines and products that can be found in that area. It has a constructor and methods to retrieve the products list, the lines list, the position and the number of lines.

Product - A product has an id, weight, name, rack id and line id which represent the rack id and the line id where that product is situated. The class has a constructor, a method to get the product id, the rack, the line where the product is situated and a method that returns the type of product such as Chair, Table, Bed etc. These types of products inherit everything that the Product class has have different attributes as well.

Order - An order has an unique id, a time stamp representing when the order was placed and a list of items from that order. It has a constructor, a method to get the id and a method to get the list of items from that order.

OrderItem - Each OrderItem object has a product type and a quantity. The class a constructor and a method to get the product type.

AGV - An AGV object has an id, a capacity which represents how much weight it can carry, a position based on coordinates, a speed of the AGV and a list of orders that an AGV needs to pick up from the warehouse. The class has a constructor, a method to get the id, the position and the list of orders. It also has a method to remove an order from the list based on an Order object which returns true if the order was successfully removed and false if the order was not found or could not be removed. The class has a method to get an Order object based on an id.

Relationships description

1. A line can have one or more racks and a rack can belong to only one line. The black diamond means that the rack cannot exist without the line it belongs to.

2. An area can have one or more lines and a line can belong to only one area. The black diamond means that a line cannot exist without the area it belongs to.
3. An rack can contain zero or more products and a product can be situated on one rack only. The dependency relationship means that although a product is dependent to a rack, the warehouse can still function without the product being placed on the rack.

4. The product can be of three types: Chair, Table and Bed. They are described as an enumeration.

5. A warehouse can contain one or more areas and an area can belong to only one warehouse. The black diamond means that an area cannot exist without the warehouse it belongs to.

6. A warehouse can have one or more orders and an order can belong to only one warehouse. The black diamond means that an order cannot exist without the warehouse it belongs to.

7. A warehouse can have one or more AGVs and an AGV can belong to only one warehouse. The black diamond means that an AGV cannot exist without the warehouse it belongs to.

8. An order can have one or more order items and an order item can belong to one or more orders. The black diamond means that an order cannot exist without at least an order item.