**Project Report**

1. **Introduction**

This Project has several warehouses located all around warehouses, each warehouse has different capacity.

However, there is no system connected all the warehouse for the smart system. We decided to design the program to make the warehouse system an automatic storage system by following the requirement design a program such that.

- When received an order, a robot will pick up an item from a warehouse and transfer.

- When received an order, the belt will output 1 item at a time.

- When received a command, a robot will store an item at a specific location.

Based on these requirements, we build a lot of storage space. It will support product relocation between warehouses. It is packed into the belt to move. And these program can store products from the outside to the warehouse or to remove items from the warehouse easily. So you can find the exact position of the product that you want to.

1. **Requirement Analysis**

**Functional Requirement and specification**

From the overview we will be divided into three parts: belt, warehouses, input & output Command

1. **Warehouse Specifications requirement**

A row is a 2-dimensional grid, each space in a grid is used to store an item, each warehouse has a robot to pick up and store items. There are 5 warehouses, warehouse1 connects with a **conveyor belt**, Warehouse 2 and Warehouse 3, Warehouse 2 connects with Warehouse 4 and 5, and There is a robot running around to transfer items from a warehouse to a conveyor belt. The conveyor belt can hold up to 10 items. All warehouses can storage 9675 products. Warehouse 1, 2, and 3 have 5 rows of 10x10 grid. Can storage 500x3 = 1500 products. Warehouse 4 has 7 rows of 5x5 grid. Can storage 175 products. Warehouse 5 has 20 rows of 20x20 grid. Can storage 8000 products.

**2.Belt**

**2.2 Product ID requirement**

Each product has a unique id in a form of 4 characters: x y z

* x represents a type of the item. It has a value of A to Y.
* y represents a row number of the item. It has a value of 1 to 5.
* z represents a row number of the item. It has a value of 00 to 99.

**2.3 Input Command requirement**

There are several commands we can give to the system. The commands have following formats.

* 0XXXX Retrieve a product id XXXX
* 1XXXX Store a product id XXXX
* 2XY00 Sort Warehouse X at row Y
* 30000 Retrieve a product from the conveyor belt
* 40000 Output information of all warehouses
* 5XXXX Search for a product ID XXXX
* 9XXXXYYYY Manually put a product id XXXX at position YYYY

**2.3.1 Output command requirement - Retrieving command: 0XXXX**

**If the system can operate successfully, the system should print out the following statements in this order:**

* Retrieving Successfully.

**If belt is full, the system should print out following statements:**

* Belt is full. Cannot retrieve the product

**If slot is empty, the system should print out following statements:**

* Slot is empty. Cannot retrieve the product.

**- Storing command: 1XXXX**

**If the system can operate successfully, the system should print out the following statements in this order:**

* Moving from Belt to A
* Moving from A to C
* Storing a product id XXXX in warehouse C: row y slot z
* Moving from C to A
* Moving from A to Start
* Storing Successfully!

**If slot is unavailable, the system should print out following statements in this order:**

* Slot is occupied. Can’t store the product.

**- Sorting command: 2XY00**

**If the system can operate successfully, the system should print out the following statements in this order:**

* Sorting process for warehouse A is complete.

**- Retrieving from the belt command: 30000**

**If the system can operate successfully, the system should print out the following statements in this order:**

* Retrieve a product with id XXXX from the belt.
* The belt now has x products on the line.

**If there is nothing on the belt, the system should print out following statements in this order:**

* The belt is empty. Cannot retrieve the product from the belt.

**- Warehouse Information: 40000**

**If the system can operate successfully, the system should print out the following statements in this order:**

* Warehouse A
* Numbers of Rows: 5
* Numbers of total products: 8
* Product in row 1: id A100, C108, E111 o Product in row 2: id –
* Product in row 3: id L355 o Product in row 4: id Q450 o Product in row 5: id U500, W501,

**- Searching for a product: 5XXXX**

**If the system can operate successfully, the system should print out the following statements in this order:**

* Found the product at XXXX.

**If the system cannot find the product, the system should print out following statements in this order:**

* Product not found.

**- Manually moving the product: 9XXXXYYYY**

**If the system can operate successfully, the system should print out the following statements in this order:**

* Move product XXXX to YYYY

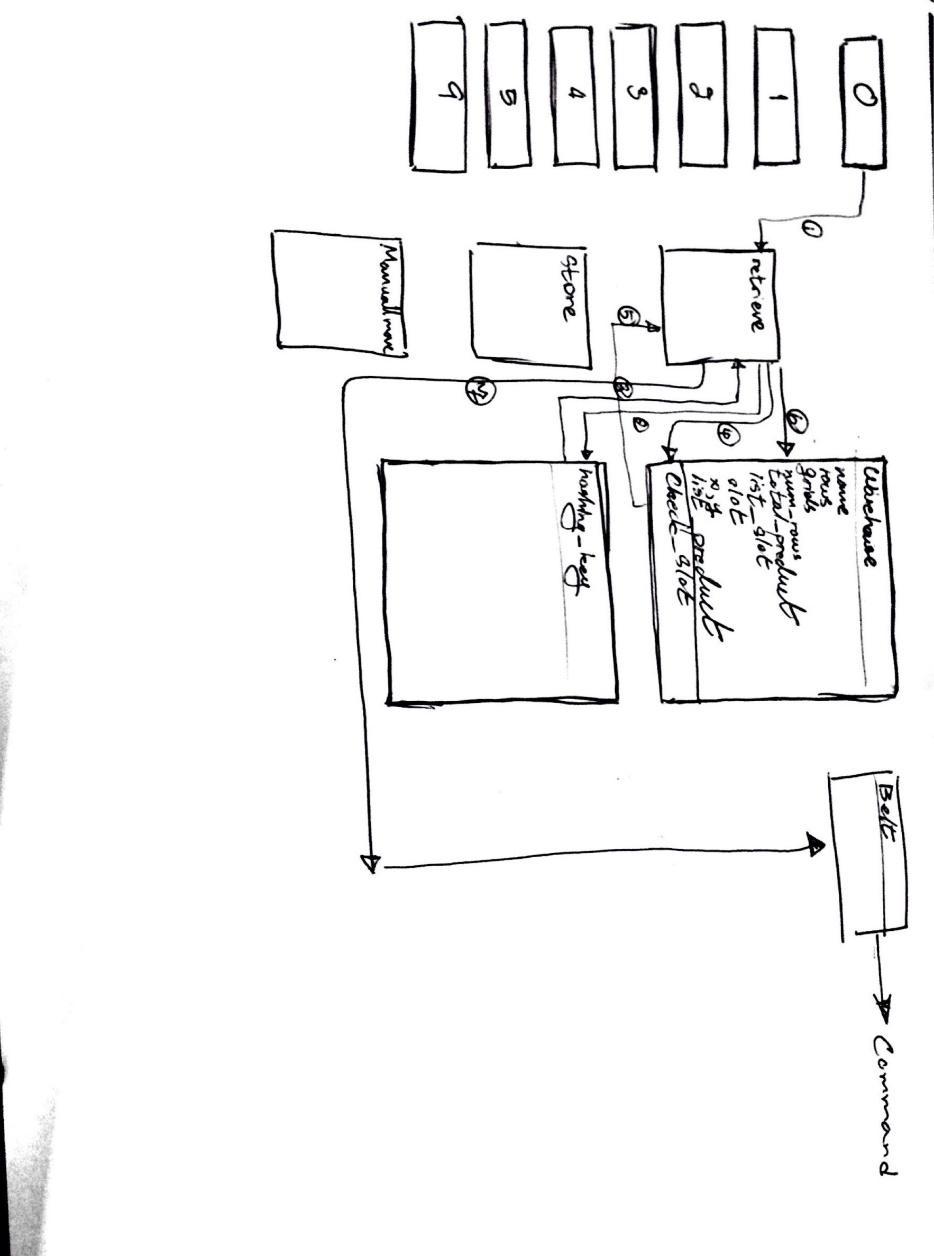
**If the slot is occupied, the system should print out following statements in this order:**

* Slot is occupied. Failed to move.

**Non-Functional Requirement**

**Rehash key**

It’s key to solve how to searching, re-check, retrieving, storage, Manually



(Concept how to use rehash to retrieving product and storage)

Normally when we need find some of product We will need to find every warehouse by warehouse to find it and it take more time, but if we use Rehash we build key shortcut to find all the warehouse by searching in key shortcut in easily and faster than normally way,

Key shortcut it will store the current product id + position. each warehouse is a grid map to find it more convenient to search where it is, (order of one)

**Extra Code:**

**InputComman: 60000**

Show all Product key in all warehouse

**InputComman: 70000**

Show in side of all warehouse

**Design UML**