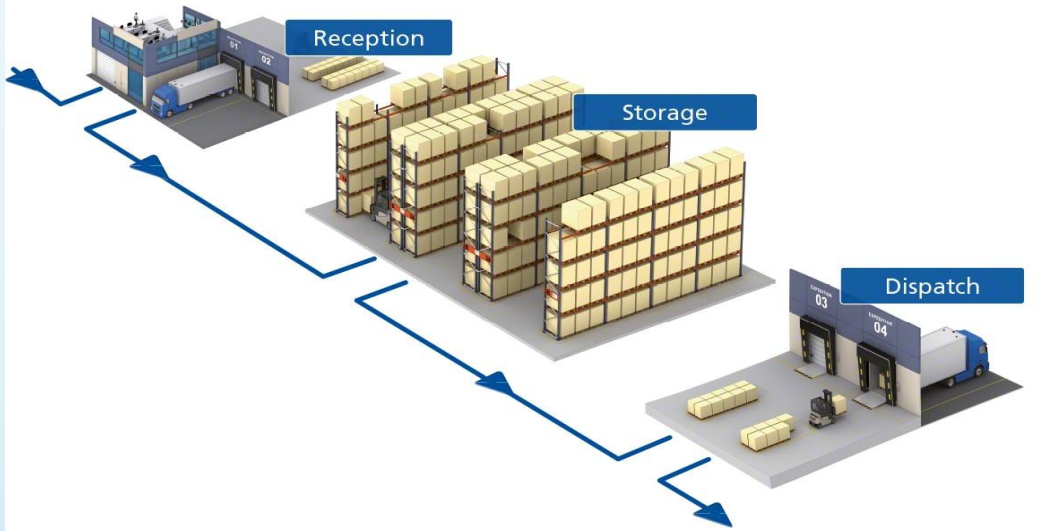
WAREHOUSE MANAGEMENT SYSTEM

TEAM NO :20

Manju Priya .P (2022506016)
Meenakshi .C (2022506097)
Nethra V.S (2022506115)


Reception
Storage
Dispatch


Warehouse Management System
Inventory Management
Receiving & Put Away
Reporting Management
Wi-Fi & Bar code system
Picking & Packing
Location Management

AIM:

The aim of this project is to develop a comprehensive Warehouse Management System (WMS) that efficiently manages warehouse operations such as adding, removing, and updating products, managing employee and customer information, and facilitating product orders by using the basic C++ concepts and Data Structures.

OBJECTIVES :

1 .Inventory Management:

To implement functionality for adding, removing, and updating product details in the warehouse.

2 . Role-based Access:

To provide distinct functionalities based on the user role (Admin, Employee, Customer).

3 . Order Processing:

To enable customers to view products and place orders, ensuring real-time inventory updates.

4 . Employee Management:

To allow the admin to manage employee information including adding, removing, and updating employee details.

5 . Customer Management:

To facilitate customer management by allowing the admin to add, remove, and update customer details.

6 . Product Stack and Queue Management:

To implement stack and queue structures for managing product storage in a more organized manner.

7 . Reporting:

To generate and display detailed reports on warehouse inventory, employee, and customer information.

ALGORITHM :

Import libraries

-iostream
-vector
-string
-algorithm
-memory
-stack-queue

class Item:

method Item(name, cost, quantity):
set name, cost, quantity

method display():

print name, cost, quantity

method order(qty):

```
if quantity >= qty:
  reduce quantity by qty
  return true
else:
  return false
```

method updateCost(newCost):

```
set cost to newCost
```

method updateQuantity(newQuantity):

```
set quantity to newQuantity
```

method getName():

```
return name
```

method getCost():
return cost

method getQuantity():
return quantity

operator *(qty):
return cost * qty

class Warehouse<T>:
method addItem(item):
add item to items list

method removeItem(itemName):
remove item from items list if name matches itemName

method displayItems():
iterate over items:
display on each item

method orderItem(itemName, qty):
iterate over items:
if item name matches itemName:
call order with qty on item
return true if successful, else false

method updateItem(itemName, newCost, newQuantity):
iterate over items:
if item name matches itemName:
call updateCost and updateQuantity on item

method getItems():
return items

class User:
method User(name):
set name

method displayRole():
abstract method

method getName():
return name

class Customer(User):
method Customer(name):
call User constructor with name

method displayRole():
print customer role

method viewProducts(warehouse):
call displayItems on warehouse

method orderItem(warehouse, itemName, qty):
iterate over items in warehouse:
if item name matches itemName:
calculate order cost
if ordering is successful:
print order details
else:
print order failure

class Employee(User):
method Employee(name):
call User constructor with name

method displayRole():
print employee role

method addItemToWarehouse(warehouse, item):
add item to warehouse

method removeItemFromWarehouse(warehouse, itemName):
remove item from warehouse if name matches itemName

method updateItemInWarehouse(warehouse, itemName, newCost, newQuantity):
update item in warehouse if name matches itemName

class Manager(Employee):
method Manager(name):
call Employee constructor with name

method displayRole():
print manager role

method generateReport(warehouse):
print warehouse report by calling displayItems on warehouse

class Admin(Manager):
method Admin(name):
call Manager constructor with name

method displayRole():
print admin role

method addEmployee(employees, employee):
add employee to employees list

method removeEmployee(employees, employeeName):
remove employee from employees list if name matches employeeName

method updateEmployee(employees, employeeName, newName):
update employee name in employees list if name matches employeeName

method addCustomer(customers, customer):
add customer to customers list

method removeCustomer(customers, customerName):
remove customer from customers list if name matches customerName

method updateCustomer(customers, customerName, newName):
update customer name in customers list if name matches customerName

method displayEmployees(employees):
    iterate over employees and print each name

method displayCustomers(customers):
    iterate over customers and print each name

class Stack<T>:
  method push(item):
      push item to stack

  method pop():
      if stack is not empty:
          pop item from stack
      else:
          print stack is empty

method top():
if stack is not empty:
return top item
else:
throw error

method display():
create a temporary stack
iterate over temporary stack and print each item

method isEmpty():
return if stack is empty

class Queue<T>:
method enqueue(item):
enqueue item to queue

method dequeue():

```
 if queue is not empty:
    dequeue item from queue
 else:
     print queue is empty
```

method front():

```
 if queue is not empty:
    return front item
 else:
    throw error
```

method display():

```
   create a temporary queue
   iterate over temporary queue and print each item
```

method isEmpty():
return if queue is empty

function main():
create warehouse, employees, customers, and admin objects
create itemStack and itemQueue objects

while true:
display welcome message
get roleChoice from user

if roleChoice is 1:
get admin password
if password is correct:
while true:
display admin menu
based on user choice, perform admin actions

```
if adminChoice is 1:
    display add employee form
    get employee details
     add employee

else if adminChoice is 2:
    display remove employee form
    get employee name
    remove employee

else if adminChoice is 3:
    display update employee form
    get employee name
    get new name
    update employee name
```

```
else if adminChoice is 4:
    display add customer form
     get customer details
     add customer

else if adminChoice is 5:
     display remove customer form
      get customer name
      remove customer

else if adminChoice is 6:
      display update customer form
      get customer name
      get new name
      update customer name
```

```
        else if adminChoice is 7:
            display employees list

        else if adminChoice is 8:
            display customers list

        else if adminChoice is 9:
            display warehouse items

        else if adminChoice is 10:
            break

        else:
            print invalid choice

else:
    print authentication failed
```

else if roleChoice is 2:

get employee name
if employee exists:
while true
display employee menu
based on user choice, perform employee actions

if employeeChoice is 1:
display add product form
get product details
add product to warehouse

else if employeeChoice is 2:
display remove product form
get product name
remove product from warehouse

```
else if employeeChoice is 3:
   display update product form
   get product name
   get new cost and quantity
   update product in warehouse

 else if employeeChoice is 4:
    display warehouse items

 else if employeeChoice is 5:
     display customers list

else if employeeChoice is 6:
   while true:
       display product stack menu
       based on user choice, perform stack actions
```

```
if stackChoice is 1:
        display push product form
        get product details
        push product to stack

 else if stackChoice is 2:
        pop product from stack

 else if stackChoice is 3:
        display stack items

else if stackChoice is 4
         break

 else:
        print invalid choice
```

```
else if employeeChoice is 7:
        while true:
            display product queue menu
            based on user choice, perform queue actions

            if queueChoice is 1:
               display enqueue product form
               get product details
               enqueue product to queue

             else if queueChoice is 2:
               dequeue product from queue

             else if queueChoice is 3:
               display queue items
```

```
            else if queueChoice is 4:
                    break

            else:
                    print invalid choice

        else if employeeChoice is 8:
                break

        else:
                print invalid choice

else:
        print employee not found

else if roleChoice is 3:
        get customer name
```

```
if customer exists:
        while true:
            display customer menu
            based on user choice, perform customer actions

    if customerChoice is 1:
        display warehouse items

else if customerChoice is 2:
        display products list
        get product name and quantity
        order product

    else if customerChoice is 3:
         break
```

```
            else:
                print invalid choice

            else:
                print customer not found

    else if roleChoice is 4:
        print exit message
        break

    else:
        print invalid choice


end function main
```

OUTPUT:


WELCOME TO WAREHOUSE MANAGEMENT SYSTEM
SELECT YOUR ROLE:
1. ADMIN
2. EMPLOYEE
3. CUSTOMER
4. EXIT
ENTER YOUR CHOICE (1,2,3,4):1
WELCOME SEETHA!!!
ENTER YOUR PASSWORD (ADMIN PASSWORD): jeep21sa
Authentication failed. Sorry!

```
                                    ---------------------------------------
                                    WELCOME TO WAREHOUSE MANAGEMENT SYSTEM
                                    ---------------------------------------


SELECT YOUR ROLE:
-----------------
1. ADMIN
2. EMPLOYEE
3. CUSTOMER
4. EXIT

ENTER YOUR CHOICE (1,2,3,4):1
WELCOME SEETHA!!!
ENTER YOUR PASSWORD (ADMIN PASSWORD): apple7

Admin Menu:
1. Add Employee
2. Remove Employee
3. Update Employee
4. Add Customer
5. Remove Customer
6. Update Customer
7. Display Employee Details
8. Display Customer Details
9. Display Product Details
10. Back
Enter your choice: 1
Enter employee name: Seenu
Seetha added employee: Seenu
```

```
Enter your choice: 7
Employees:
Ramesh
Bindhu
Seenu

Admin Menu:
1. Add Employee
2. Remove Employee
3. Update Employee
4. Add Customer
5. Remove Customer
6. Update Customer
7. Display Employee Details
8. Display Customer Details
9. Display Product Details
10. Back
Enter your choice: 8
Customers:
Ammu
Praveen

Admin Menu:
1. Add Employee
2. Remove Employee
3. Update Employee
4. Add Customer
5. Remove Customer
6. Update Customer
7. Display Employee Details
8. Display Customer Details
```

```
Enter your choice: 3
Enter employee name: Seenu
Enter new name: Ramamurthi
Seetha updated employee name to: Ramamurthi

Admin Menu:
1. Add Employee
2. Remove Employee
3. Update Employee
4. Add Customer
5. Remove Customer
6. Update Customer
7. Display Employee Details
8. Display Customer Details
9. Display Product Details
10. Back
Enter your choice: 7
Employees:
Ramesh
Bindhu
Ramamurthi

Admin Menu:
1. Add Employee
2. Remove Employee
3. Update Employee
4. Add Customer
5. Remove Customer
6. Update Customer
7. Display Employee Details
8. Display Customer Details
```

```
Enter your choice: 4
Enter customer name: Kishore
Seetha added customer: Kishore

Admin Menu:
1. Add Employee
2. Remove Employee
3. Update Employee
4. Add Customer
5. Remove Customer
6. Update Customer
7. Display Employee Details
8. Display Customer Details
9. Display Product Details
10. Back
Enter your choice: 8
Customers:
Ammu
Praveen
Kishore

Admin Menu:
1. Add Employee
2. Remove Employee
3. Update Employee
4. Add Customer
5. Remove Customer
6. Update Customer
7. Display Employee Details
8. Display Customer Details
9. Display Product Details
```

```
Enter your choice: 2
Enter employee name: Ramamurthi
Seetha removed employee: Ramamurthi

Admin Menu:
1. Add Employee
2. Remove Employee
3. Update Employee
4. Add Customer
5. Remove Customer
6. Update Customer
7. Display Employee Details
8. Display Customer Details
9. Display Product Details
10. Back
Enter your choice: 7
Employees:
Ramesh
Bindhu

Admin Menu:
1. Add Employee
2. Remove Employee
3. Update Employee
4. Add Customer
5. Remove Customer
6. Update Customer
7. Display Employee Details
8. Display Customer Details
9. Display Product Details
10. Back
```

```
Enter your choice: 10

SELECT YOUR ROLE:
-----------------
1. ADMIN
2. EMPLOYEE
3. CUSTOMER
4. EXIT

ENTER YOUR CHOICE (1,2,3,4):2
Enter employee name: Bindhu

Employee Menu:
1. Add Product
2. Remove Product
3. Update Product
4. Display Product Details
5. Display Customer Details
6. Manage Product Stack
7. Manage Product Queue
8. Back
Enter your choice: 1
Enter item name: carrots
Enter item cost (per kg ($)) : 20
Enter item quantity (kg) : 500
Bindhu added item to warehouse: carrots

Employee Menu:
1. Add Product
2. Remove Product
3. Update Product
```

```
Enter your choice: 1
Enter item name: Rice
Enter item cost (per kg ($)) : 32
Enter item quantity (kg) : 789
Bindhu added item to warehouse: Rice

Employee Menu:
1. Add Product
2. Remove Product
3. Update Product
4. Display Product Details
5. Display Customer Details
6. Manage Product Stack
7. Manage Product Queue
8. Back
Enter your choice: 1
Enter item name: Wheat
Enter item cost (per kg ($)) : 55
Enter item quantity (kg) : 1289
Bindhu added item to warehouse: Wheat

Employee Menu:
1. Add Product
2. Remove Product
3. Update Product
4. Display Product Details
5. Display Customer Details
6. Manage Product Stack
7. Manage Product Queue
8. Back
```

```
Enter your choice: 4
Warehouse Inventory:
Name: carrots, Cost (per kg): $20, Quantity: 500 kg
Name: Notebook, Cost (per kg): $20, Quantity: 25 kg
Name: Rice, Cost (per kg): $32, Quantity: 789 kg
Name: Wheat, Cost (per kg): $55, Quantity: 1289 kg

Employee Menu:
1. Add Product
2. Remove Product
3. Update Product
4. Display Product Details
5. Display Customer Details
6. Manage Product Stack
7. Manage Product Queue
8. Back
Enter your choice: 2
Enter item name: Notebook
Bindhu removed item from warehouse: Notebook

Employee Menu:
1. Add Product
2. Remove Product
3. Update Product
4. Display Product Details
5. Display Customer Details
6. Manage Product Stack
7. Manage Product Queue
8. Back
```

```
Enter your choice: 4
Warehouse Inventory:
Name: carrots, Cost (per kg): $20, Quantity: 500 kg
Name: Rice, Cost (per kg): $32, Quantity: 789 kg
Name: Wheat, Cost (per kg): $55, Quantity: 1289 kg

Employee Menu:
1. Add Product
2. Remove Product
3. Update Product
4. Display Product Details
5. Display Customer Details
6. Manage Product Stack
7. Manage Product Queue
8. Back
Enter your choice: 6

Product Stack Menu:
1. Push Product to Stack
2. Pop Product from Stack
3. Display Stack
4. Back
Enter your choice: 3
Name: Wheat, Cost (per kg): $55, Quantity: 1289 kg
Name: Rice, Cost (per kg): $32, Quantity: 789 kg
Name: carrots, Cost (per kg): $20, Quantity: 500 kg
```

```
Enter your choice: 8

SELECT YOUR ROLE:
-----------------
1. ADMIN
2. EMPLOYEE
3. CUSTOMER
4. EXIT

ENTER YOUR CHOICE (1,2,3,4):3
Enter customer name: Ammu
Customer Menu:
1. View Products
2. Order Product
3. Back
```

```
Enter your choice: 2
Enter item name: Rice
Enter quantity (kg) to order: 5
Order Cost: $160
Thanks Ammu!!! You ordered 5 kg of Rice

Customer Menu:
1. View Products
2. Order Product
3. Back
Enter your choice: 4
Invalid choice. Please try again.

Customer Menu:
1. View Products
2. Order Product
3. Back
Enter your choice: 3

SELECT YOUR ROLE:
-----------------
1. ADMIN
2. EMPLOYEE
3. CUSTOMER
4. EXIT

ENTER YOUR CHOICE (1,2,3,4):4


...Program finished with exit code 0
Press ENTER to exit console.
```

•PROBLEMS FACED:

- In the part of inheritance we have faced some errors.After the modification made by us makes us easy to resolve the issue.

- Actually its so easy to use the basic C++ Concepts but the aim of using it in meaningful way its will be difficult.

Conclusion :

This Warehouse Management System provides a robust and efficient way to handle warehouse operations. By leveraging object-oriented principles and role-based access, the system ensures smooth and organized management of products, employees, and customers. The integration of stack and queue data structures further enhances the flexibility and functionality of the system.