Report # 2
Group 5 C++ WMS

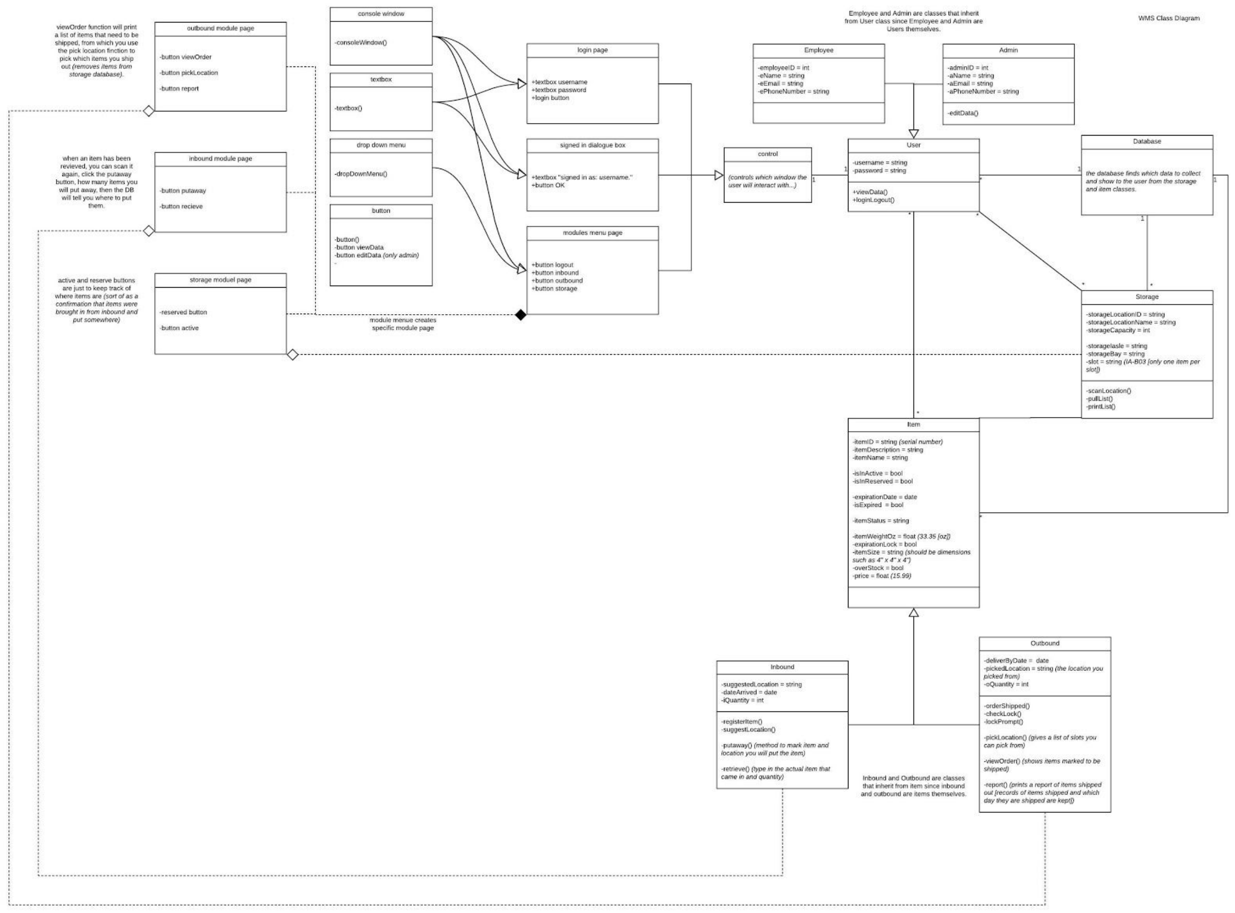
Table of Continents:

Group Member Contributions:

- **Alejandro**
    - I set the table of continents to be automatic and worked on some of the formatting of the document. I also worked on the hardware requirements section and helped work on subsystem design. I worked on statusInfo, active, and reserved on the data types and operation signatures section. I helped work on the design and optimization of the class diagram.
    - I worked on the data structures section and helped revise it, copied the list of test cases (that were already described in the testcase), helped update the use case list to be more accurate, and worked on the plan for testing non-functional dependencies.
- **Sela**
    - I worked on persistent data storage.Helped on the design of subsystems.
- **Andy**
    - I worked on the network protocol section. I helped work on the design of the subsystem. I also worked on explaining the architectural design.
- **Tony**
    - I helped on the design of the subsystems. I worked on storage, item, outbound and inbound data types, attributes, functions and operation signatures. I also worked on the mapping subsystems section.
    - Helped on the plan for testing non-functional requirements table.

Class Diagram and Interface Specifications:

Class Diagram:


viewOrder function will print a list of items that need to be shipped, from which you use the pick location finction to pick which items you ship out *(removes items from storage database).*
outbound module page
-button viewOrder
-button pickLocation
-button report
when an item has been revieved, you can scan it again, click the putaway button, how many items you will put away, then the DB will tell you where to put them.
inbound module page
-button putaway
-button recieve
active and reserve buttons are just to keep track of where items are *(sort of as a confirmation that items were brought in from inbound and put somewhere)*
storage moduel page
-reserved button
-button active
console window
-consoleWindow()
textbox
-textbox()
drop down menu
-dropDownMenu()
button
-button()
-button viewData
-button editData *(only admin)*
login page
+textbox username
+textbox password
+login button
signed in dialogue box
+textbox "signed in as: *username.*"
+button OK
modules menu page
+button logout
+button inbound
+button outbound
+button storage
module menue creates specific module page
control
(controls which window the user will interact with...)
Employee and Admin are classes that inherit from User class since Employee and Admin are Users themselves.
WMS Class Diagram
Employee
-employeeID = int
-eName = string
-eEmail = string
-ePhoneNumber = string
Admin
-adminID = int
-aName = string
-aEmail = string
-aPhoneNumber = string
-editData()
User
-username = string
-password = string
+viewData()
+loginLogout()
Database
the database finds which data to collect and show to the user from the storage and item classes.
Storage
-storageLocationID = string
-storageLocationName = string
-storageCapacity = int
-storageIasle = string
-storageBay = string
-slot = string *(IA-B03 [only one item per slot])*
-scanLocation()
-pullList()
-printList()
Item
-itemID = string *(serial number)*
-itemDescription = string
-itemName = string
-isInActive = bool
-isInReserved = bool
-expirationDate = date
-isExpired = bool
-itemStatus = string
-itemWeightOz = float *(33.35 [oz])*
-expirationLock = bool
-itemSize = string *(should be dimensions such as 4" x 4" x 4")*
-overStock = bool
-price = float *(15.99)*
Inbound
-suggestedLocation = string
-dateArrived = date
-iQuantity = int
-registerItem()
-suggestLocation()
-putaway() *(method to mark item and location you will put the item)*
-retrieve() *(type in the actual item that came in and quantity)*
Outbound
-deliverByDate = date
-pickedLocation = string *(the location you picked from)*
-oQuantity = int
-orderShipped()
-checkLock()
-lockPrompt()
-pickLocation() *(gives a list of slots you can pick from)*
-viewOrder() *(shows items marked to be shipped)*
-report() *(prints a report of items shipped out [records of items shipped and which day they are shipped are kept])*
Inbound and Outbound are classes that inherit from item since inbound and outbound are items themselves.
1
1
1
1
1
1
*
*
*
*
*
*
*

Data types and Operation Signature:

Class name = Admin
This is a class that is tied to the user Admin

Attributes

- adminID: Unique ID of an admin
- aName: Name of the admin
- aEmail: Email that belongs to the admin
- aPhoneNumber: phone number that belongs to the admin

Functions

- editData(): This function will provide access to edit data

Class name = Employee
This is a class that will list attributes of the employee

Attributes

- employeeID: Unique ID of the employee
- eName: Name of the employee
- eEmail: Email that belongs to the employee
- ePhoneNumber: phone number that belongs to the employee

Class name = User
This class holds information that is inherited by the employee and admin

Attributes

- username: Unique username a user will use to log in
- password: Password that a user will need to log in

Functions

- viewData(): This function will provide the users access to view data in the GUI
- loginLogout(): This function will take care of the functionality of logging in and logging out

---

Class name = Item
the item table will show all descriptive traits of a certain product

Attributes

- itemID: each item received will have its unique item ID
- itemDescription: a string that will provide a short description of
  the item
- ItemName: the name of the item
- isInActive: Boolean data type will display a Y or N if the
  item is stored in the active table
- isInReserve: Boolean data type will display a ‘Y’ or ‘N’ depending if the item is stored in the reserve table
- expirationDate: the date the item expires

- isExpired: display Boolean depending if the product has expired
- itemStatus: shows the current stage of the item e.g “In Receiving”
- itemWeightOz: Weight of the item in ounces
- expirationLock: displays if the item has a lock due to being expired
- itemSize: will list the dimension of the item, format will be same as 4’ x 4’ x 4’
- overStock: will list Y or N if the item is in overstock, if it does not fit in active
- price: price of the item as a float

Class name = Outbound
This class will display all of the functions and attributes associated with the process of preparing an item to be shipped out (Exit the warehouse).

Attributes

- deliverByDate: suggested deadline of an item, item expected to be delivered by this date
- pickedLocation: the name of the location the user has picked from
- oQuantity: how many items the user has pulled

Functions

- orderShipped(): This function will pull a query of all the items that have being successfully shipped out
- checkLock(): Before an item is shipped out, this function will make sure that the items are free of locks
- lockPrompt(): This function will prompt a warning message if it identifies an item with a current lock
- pickLocation(): This function will get an order and display a list of slots that the user can pick from
- viewOrder(): This function will display in list view the orders that are ready to be picked

---

Class name = Inbound
This class will take care of the steps of receiving and registering a product. (Incoming to the warehouse)

Attributes

- suggestedLocation: the name of the location where the product will be stored next
- dateArrived: The date the item was checked in
- iQuantity: Indicates the quantity of an item being received

Functions

- registerItem(): This function will make sure to save the new Item entry into the database
- suggestLocation(): Depending on the space a newly registered item takes, it will suggest where the item should be stored.

Class name = Storage
This class will list the attributes and functions related to where the item is stored. (Where the item lives in the warehouse)

Attributes

- storageLocationID: unique ID with the specific name of the item's storage location. For example Isle 2 Bay 04 Slot 1
- storageLocationName: The category title of the storage location
- storageCapacity: Number that lists what is the maximum that location can hold
- storageAisle: Isle number of where the item is stored
- storageBay: Bay string of where the item is stored
- slot: Only one item fits in a slot, this will display the number of slot where the item is located.

Functions

- scanLocation(): This function will scan the storage locations for availability. It will be looking at quantities.
- pullList(): This function will pull a list and display the fields in the storage table.

printList(): This function will display and print a list of what is stored in the storage table, * May not need this function

Class name = Admin
This is a class that is tied to the user Admin

Attributes

- adminID: Unique ID of an admin
- aName: Name of the admin
- aEmail: Email that belongs to the admin
- aPhoneNumber: phone number that belongs to the admin

Class name = inbound module page
This is a class that triggers inbound related actions after button is pressed in UI

Actions

- button putaway: After this button is pressed a user is ready to putaway received items
- button receive: After this button is pressed a user is ready to receive items

Class name = Outbound Module
This is a class that triggers outbound related actions after button is pressed in UI

Actions

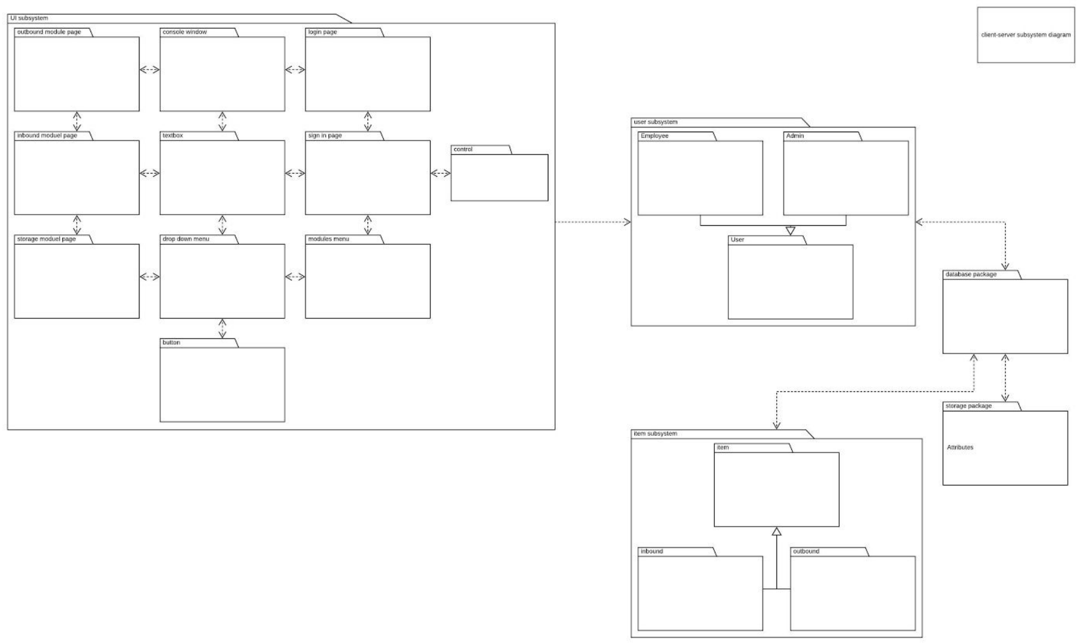
- button viewOrder: This button is a trigger to populate a list of orders that need to be shipped out
- button pickLocation: Once this button is hit the user is ready to pick items for shipment
- button report: This button is a trigger to generate a report of items already shipped

System Architecture and System Design:

Architectural Design:

We are using the client-server. There will be one server and a couple clients. The clients will communicate with the server by sending requests to get what they need. The Ui subsystem goes to the user subsystem. The user subsystem and the item subsystem go to the database package. The database package goes to the storage package.

Identify Subsystems:



This is our UML package diagram for our subsystems based off of the server-client architecture model.

Mapping Subsystems to hardware:

Hardware Type 1: Computer with Monitor, what the employee will use to view UI (Client)

Subsystems: User Subsystem, UI Subsystem, Storage package and Item Subsystem

---

Hardware Type 2: Scanner / Portable Device, device that will register the item (Client)

Subsystems: User Subsystem and item Subsystem

Hardware Type 3: Database Server (Server)

Subsystems: Database Subsystem, Storage package and Item Subsystem

----

Persistent Data Storage:

- Knowing the amount of products we have in stock such as inbound, and outbound is important to us, so we wanted to keep track of all of our data, including employees' information, and inventory.
- To save some space and resource, any inventory that has left the warehouse or has been sold for more than 12 months will be removed from the database.
- We will store our data inside the database for faster access.
- To access the data, we will be using MySQL.
- Depending on the constrain of our resource and how much data we will have, we might be able to store our data in flat file instead.

Network Protocol:

- The network protocol that will be used will be the transmission control protocol, and internet protocol. It will use an internet connection to connect the devices being used.
- Hypertext Transfer Protocol(HTTP) will be used to establish the connection.
- HTTP allows for the use of mobile devices for the warehouse management system.
- Using HTTP should allow for the different devices to use the same network, such as the mobile devices.

Hardware Requirements:

Server Requirements: *(support up to 100 terminals)*

- CPU: Intel Xeon processor 3 GHz or greater (4+ cores).
- RAM: 16 GB.
- 1 Gbps or faster network interface.
- 100 GB (for Operating System, Application, Logs, etc.).
- 1 TB or more (for Database).
- 500 GB (for case files).
- Access to the internet to host.

Computer requirements:

- RAM: 1 GB
- LCD Color Display (Monitor Resolution 1024 × 768 or higher).

- OS: Windows 7 or above.
- Keyboards/mice for data entries.
- Internet access (to connect to server).
- Portability (laptop).
- Battery life lasting at least 8 ½ hours (to cover one of the staff's full shift) or higher.

Scanner Requirements:

- WiFi Radio Module (works up to 100m from WiFi access point).
- Micro USB Recharging.
- Battery lasting at least 8 ½ hours or higher.
- Integrated Barcode Scanner (1D / 2D barcodes supported).
- LCD Display (Screen Resolution 640 × 480 or higher).
- Shock Resistance.
- Dust and Splash Resistance.

Algorithms and Data Structures:

Algorithms:

We are not using a mathematical model for our program.

Data Structures:

- Hash Tables: we will use hash tables to store our numeric and character data.
- Arrays: we will use arrays to handle variable values quicker.
- Linked lists: to organize and display the values entered by the user.

User Interface Design and Implementation:

We decided to keep our user interface the same as our prototype. the reason we were not making any changes is because we want our WMS to be as user friendly as possible for our customers yet effective at the same time. We don't want to confuse user by adding too many buttons or too less. All the buttons that appear on the user interface should be clear to the user. Users should be able to look at each button and determine what the button is supposed to do, whether the user needs to log in, looks for inventory or what needs to be restocked etc…

We want to keep our UI mainly simple to use, so we will not be focusing on the aesthetics yet. The reason why is because we do not want to confuse the employees using the WMS. A Lot of the actions will be repetitive and would like our system to be fast, effective and not use up many resources just running the UI alone.

In the future, when we update our software, one of our possible options is that we might be adding more buttons or search box for keywords on our interface as we think if it is necessary. We will also be adding more colors to the background of pages, this will be done to help the user differentiate in which module they are currently in.

Plan for Testing Non-Functional Requirements:

Nonfunctional requirement	How we will test the requirement	Expected Result
The interface should be easy for the users to operate.	We will get someone inexperienced with the ui to try it out and see how comfortably they work with it.	Volunteer will complete one UI use case in less than 30 seconds.
The database search queries should not take more than 5 seconds.	We will run common search queries and see how long they take and see if any queries need revising.	Queries will return in < 5 seconds.
The system and all the data should all be backed up.	We will check the database backup manager and see that it is backed up.	Back up system will have the same data as primary within 30 seconds of a primary save.
The systems should be able to handle multiple users at once.	We will log on to the system and see if any errors or lag occurs.	System will handle at least 3 users at one time.
The system should be operable on multiple different devices.	We will try to log onto the system with different types of devices and see how the system handles it.	Simultaneously update data from database tool and UI within 1 minute.
A system update will be outside of company operating hours and should not take more than 3 hours to complete.	We will perform an update and check the log to see how long it will take.	Successfully update system in < than 3 hours.