Warehouse Management and Pathfinder Tool

URL:
http://ec2-18-221-234-28.us-east-2.compute.amazonaws.com/SoftwareEngin eering/Warehouse.html

Software Engineering - Group #12

Michael Ehnes, Liam Flaherty, Mitchell Hoffmann, Parker Jones, Christian Reed, Skyler Todd

Table of Contents

2. Customer Statement of Requirements
4. Glossary of Terms
5. System Requirements
6. Effort Estimation
10. Traceability Matrix
11. Use Cases
16. Domain Analysis
17. Interaction Diagrams
21. Class Diagram and Interface Specification
24. System Architecture
28. Algorithms and Data Structures
29. UI Design and Implementation
30. Design of Tests
31. History, Current, and Future Work
32. References

Summary of Changes

1. Changed some words and phrasing in the CSR
2. Added Terms to the Glossary
3. Changed the Class Diagram
4. Changed the UI Design graphics and description
5. Added References
6. Removed the idea of using a pathing AI

Customer Statement of Requirements

The working environment of a warehouse or a large store is complex and arduous. Many products exist in a large warehouse, such as that of Amazon or Walmart, and it can be difficult to locate and ship items in an efficient and timely manner. We would like to request a software that is: easily accessible, easy to use, and responsive. We require a product that will allow us to reduce the amount of time and effort it takes to perform our jobs adequately. Not only do we require a application to perform these tasks but also make it user friendly.

We would like to see a software that can give us a competitive edge and be cheap to use and maintain. This application must be easily accessible on all devices. We would also like to be able to access this application through a web browser. This would make it easier for us to gain access to this application over various ways so we would not be dependent of specific hardware. Especially for a smaller business like us, we can may not necessarily be able to provide all employees specific hardware for this software we want. That is why I would like to have as many options as possible for the application to run on.

This application must be well constructed and easy to interpret at first glance. This would include buttons and layout being thoughtfully designed so we can clearly understand where all the features are. We request a software that is friendly for newer employees so we can minimize training and specialization for this tool. The interface must be simple and smart, buttons being easy to click and items easy to search. The interface must also have a graphical reaction to each action that is done so we know what is interactable. The interface should also contain a map of the warehouse or store so employees/customers know exactly where to look when requesting an items location. The map should be based on some sort of grid so it is easy for us to interpret. We do not want anything overly intricate, just something that provides a basic scale to represent our warehouse shelves. We would also like some sort of key for the map so we can easily understand what each part of the map can represent. We would also like to request a way for employees to comment about the map so managers will be able to fix any problem that employees can run into. These problems could include items being stocked in the wrong area, damaged goods, broken tools, warnings, or notes for the department. These comments should have features such as being anonymous comments and who can see these comments. That way some employees can leave comments for themselves and for other employees or managerial staff. We would like an application with a decisive interface that will make it easy for our employees to understand the application and the features it can provide.

The application must have all the basic functions that most warehouse management software should provide. These include an item catalog, item search, and item tracking. We would also like to request a possible way to search items and be able to pull information like quantity, location and serial number. The search feature of the application must include every item that is housed in the warehouse and the quantity of the item. There must be a feature that removes the item when it has been taken out of the warehouse, and a feature that can allow the user to add more stock to the item list. The application should also be able to warn employees when an item is low in stock. This warning should be customizable based on a numerical amount of that item. For example we could receive a warning about item A when we have 5 of it left in the warehouse and get another warning for item B when we have 20 of it left in the warehouse. A comprehensive list of items must be easy to access and amend. After we search for an object we can have the application map the requested item on the interface and provide not only a written but a visual location of the item. This feature must not only be able to map out single items but be able to map out multiple items. A map of the warehouse must be easy to access after a search of an item(s) is done, and must give the quickest route to the item(s). With all these features not only should the application be an effective tool, but should be easily accessible for employees. In short we require an application that can manage items in our warehouse, map out items for employees to find, and be easy to understand.

Glossary of Terms

Database - a structured set of data held in a computer, especially one that is accessible in various ways.

Browser - a program with a graphical user interface for displaying HTML files, used to navigate the World Wide Web.

Application - Computer software designed to perform a group of coordinated functions, tasks, or activities for the benefit of the user.

Interface - a device or program for connecting two items of hardware or software so that they can be operated jointly or communicate with each other.

GUI (Graphical User Interface) - is a user interface that includes graphical elements, such as windows, icons and buttons.

End User – The user who uses a particular product.

Authentication – the process in which a database or system can verify or authenticate a user's identity

System Requirements

Functional Requirements

Label	Weight	Description
1. REQ-Search	1	The ability to search for and find items and their quantity
2. REQ-Map	2	A map and simple directions to where the item is located
3. REQ-List	3	A comprehensive list of items and the quantity
4. REQ-Add	4	The ability to add items and their quantity to the stock
5. REQ-Remove	5	The ability to remove items from stock and auto subtract quantity

Nonfunctional Requirements

6. The application must be able to handle a large amount of items to be searched through.
7. The search function must be quick and efficient with no errors, likely utilizing binary search.
8. The application must have a button to quickly return to the initial page to search for further items.
9. The database and server holding the list of items and the source of the pages must not be reachable from any of the application pages.
10. Every page must have the same aesthetically pleasing look with simple colors and easily readable text from any device.
11. The application must correctly list the item, its quantity, and where the item is located without anyerror.
12. The application must have a consistently reliable and easy to read interface between multiple platforms such as computers, phones, tablets, etc.

On Screen Appearance Requirements

13. Easily clickable buttons with easy to read text
14. Easy to use search function that quickly finds all items
15. Interface that reacts to actions to show that the application is responsive

Effort Estimation using Use Case Points

a. Use Case Points

Projects with many complicated requirements take more effort to design and implement than projects with few simple requirements. In addition, the effort depends not only on inherent difficulty or complexity of the problem, but also on what tools the develo pers employ and how skilled the developers are.

The formula for calculating UCP is composed of three variables:

1. Unadjusted Use Case Points (UUCP), which measures the complexity of the functional requirements.
2. The Technical Complexity Factor (TCF), which measures the complexity of the nonfunctional requirements.
3. The Environment Complexity Factor (ECF), which assesses the development team's
4. experience and their development environment. UCP = UUCP×TCF×ECF

Actor Type	**Description of how to recognize the actor type**	**Weight**
Simple	The actor is another system which interacts with our system through an API.	1
Average	The actor is a person interacting through a text based user interface.	2
Complex	The actor is a person interacting via a graphical user interface.	3

UAW= 3x2+2x1+1x1 = 9

Use Case Category	Description	Weight
Simple	Simple user interface. Up to one participating actor (plus initiating actor). Number of steps for the success scenario: ≤3. If presently available, its domain model includes≤ 3 concepts.	5
Average	Moderate interface design. Two or more participating actors. Number of steps for the success scenario: 4 to 7. If presently available, its domain model includes between 5 and 10 concepts.	10
Complex	Complex user interface or processing. Three or more participating actors. Number of steps for the success Scenario: 7. If available, its domain model includes ≥ 10 concepts	15

The UUCW is calculated by tallying the use cases in each category, multiplying each count by its specified weighting factor, and then adding the products.

Use Case	Description	Category	Weight
UC1	Complex user interface. Infinite steps for completion depending on list size. Two participating Actors (Visitor, Processor)	Average	10
UC2	Simple user interface. 3 steps for success	Simple	5

	scenario. 3 participating actors (Visitor, Database, Processor)		
UC3	Complex user interface. 6 Steps for success scenario. 2 Participating actors (Visitor, Database)	Complex	15
UC4	Simple User interface. 3 steps for success scenario. 2 participating actors (Visitor and Database)	Simple	5

UUCW = 2x5 + 1 x 10+ 1x15 = 35

UUCP = 35 + 9 = 44

Technical Factor	Description	Weight	Perceived Complexity	Factor
T1	Distributed System	2	2	4
T2	Performance Objectives	1	1	1
T3	End-user efficiency	1	1	1
T4	Complex internal Processing	1	1	1
T5	Reusable design or code	1	2	2
T6	Easy to install	0.5	1	.5
T7	Easy to use	0.5	1	.5

T8	Portable	2	3	6
T9	Easy to change	1	3	3
T10	Concurrent Use	1	1	1
T11	Special Security Features	1	1	1
T12	Provides direct access for third parties	1	1	1
T13	Special user training facilities are required	1	1	1
			Total	23

.6+(.01x23) = .83

This will cause a reduction of the UCP by 17%.

Environmental Factor	Description	Weight	Perceived Impact	Total
E1	Beginner Familiarity with the UML-based development	1.5	3	4.5
E2	Application Problem Experience	0.5	4	2
E3	Paradigm Experience	1	2	2
E4	Lead Analyst Capability	0.5	5	2.5
E5	Motivation	1	3	3
E6	Stable	2	2	4

	Requirements			
E7	Part time staff	-1	0	0
E8	Difficult Programming Language	-1	2	-2
			total	16

1.4+(-.03x16) = 1.07

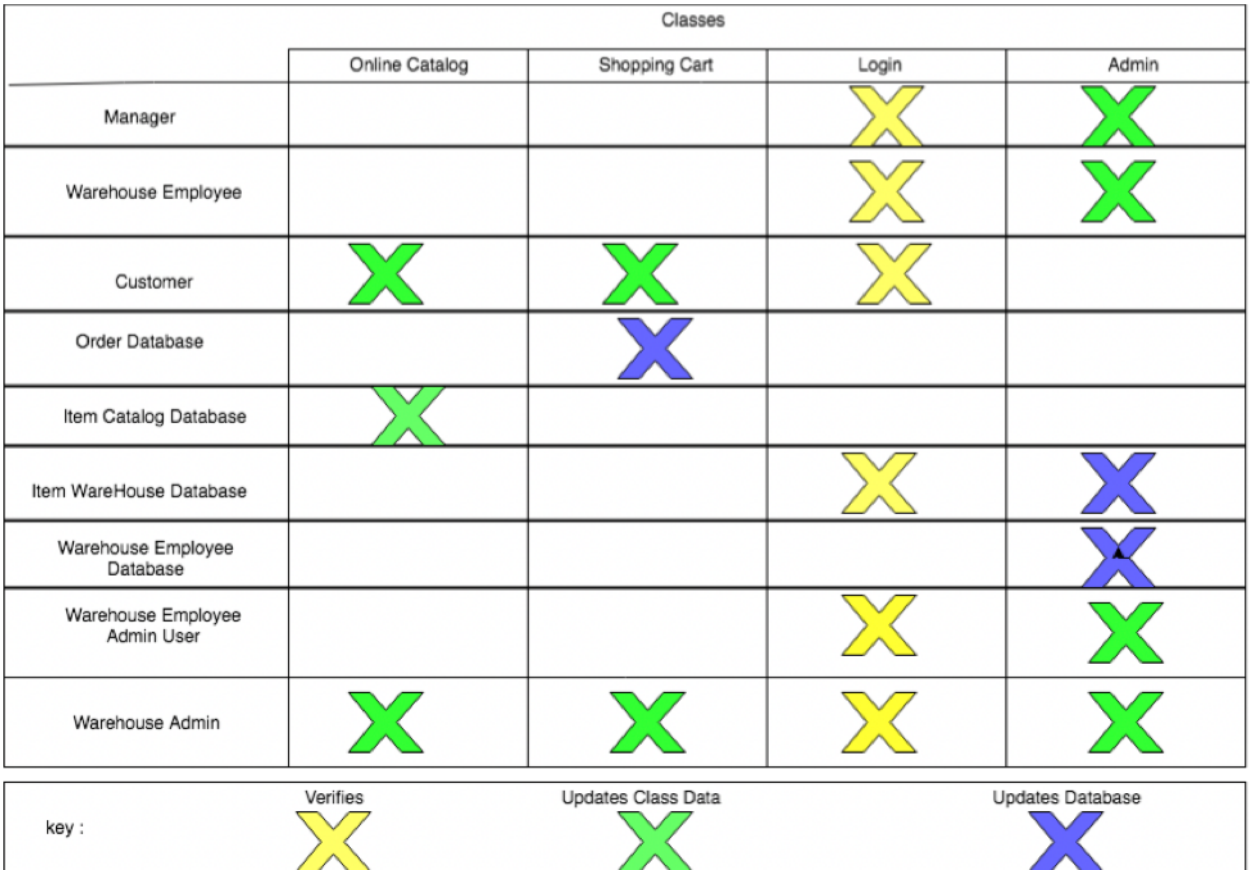
Results in a decrease of the UCP by 8%

44x.83x1.07=39 Case points

PF = 15
39x15 = 585 hours

Traceability Matrix

C. Traceability Matrix
The Traceability matrix is meant to represent what aspect of external sources will be updated or notified by each class structure. The Manager, Warehouse Employee, Warehouse Admin, and Customer are human resources and users of our application. In the traceability matrix they are meant to show their current or supposed access to each class. Manager and Employee can use the online catalog but the Admin class provides them with all the tool necessary so they do not need it. Customers are only able to login, view the online catalog and purchase goods. Their orders are sent out to the orders database table. The rest of the database tables that will be updated by each class. There will be a total of 4 different database tables : Item Catalog - for all items in every warehouse, Item Warehouse Database - for all items in the particular warehouse, Warehouse Employee Database - stores employee information and Admin access, Order database - stores orders that need to be distributed to individual warehouses for item processing. Item Catalog updates the Online Catalog class. Warehouse Employee Database check user input from the Login class and is updated by the Admin Class. Item Warehouse Database gets updated by the Admin class. Order Database gets updated by the Shopping Cart class.



Casual Description

For Adding and Removing Items this use case is responds when the customer interacts with ordering items. The customer has the ability to add and remove items from there shopping queue. This use case is similar to most online shopping as it gives the user the ability to order items directly from the warehouse. The managers can also use this process when adding or removing items from the warehouse. Collecting Items is a use case that happens when a customer adds or removes items from a list. The list is a collection of items that relate to the items in the warehouse. This list can then be processed and sent to the warehouse so the warehouse workers can process the order. For Manager's Layout the management level user has the ability to map out the warehouse in a scale map. The manager-level user can then have the ability to edit the map and designate where each individual item it located with the store. Updating information can then be done when the user logs into the app and changes their personal information. This can be down when the user send a change request that relates to information such as address, ordering information, or other various personal information.

Use Case 1 - Adding / Removing Item	
Initiating Actor	Customer
Actors Goal	To add or remove items from their shopping list from their handheld device.
Pre-Conditions	Actor has logged into the app and the search GUI is showing
Post-Conditions	Once the user has selected all the items they wish to purchase there will be their list and a subtotal at the bottom of the page.
Flow Of Events	1) User logs into app. 2) User selects the menu tab. 3) User selects the search tab. 4) User searches for items and adds them to the list (repeat this step for as many items as the user wants). 5) If the user wishes to remove an item they can search it and press remove from list. 6) User presses the finished button and a finalized list with its subtotal populates the page.

Use Case 2 - Collecting Items	
Initiating Actor	Customer
Actors Goals	To follow set path and retrieve items on list
Pre-Conditions	User has added items to a list
Post-Conditions	User will have acquired desired items on their list

Flow Of Events	1) Once user has created their list, they press the go button on the bottom 2) The map will tell the user where to go from the entrance to their first item. 3) Once user has retrieved their item, they will press the “next item” button. 4) The map will then show the user how to get to the next item from their last items location. 5) User will repeat step four until all desired items have been acquired.

Use Case 3 - Managers Layout	
Initiating Actor	Manager
Actors Goals	Create and populate a map, or change existing map.
Pre-Conditions	Manager has logged in as a manager
Post-Conditions	The store map will be complete with shelf locations and items that populate the shelves

Flow Of Events	1) Manager drags and drops shelf locations from a toolbox displayed in the GUI 2) If there was a previously saved store layout the page will be populated with the appropriate layout and the manager can continue to execute the remaining steps 3) Once the manager has set up the store layout, they may add items onto a shelf selecting that shelf. They may also add extra information during this step, such as price and quantity. 4) After the manager has completely set up the store, they will click save and the information about the layout will be sent to a database.

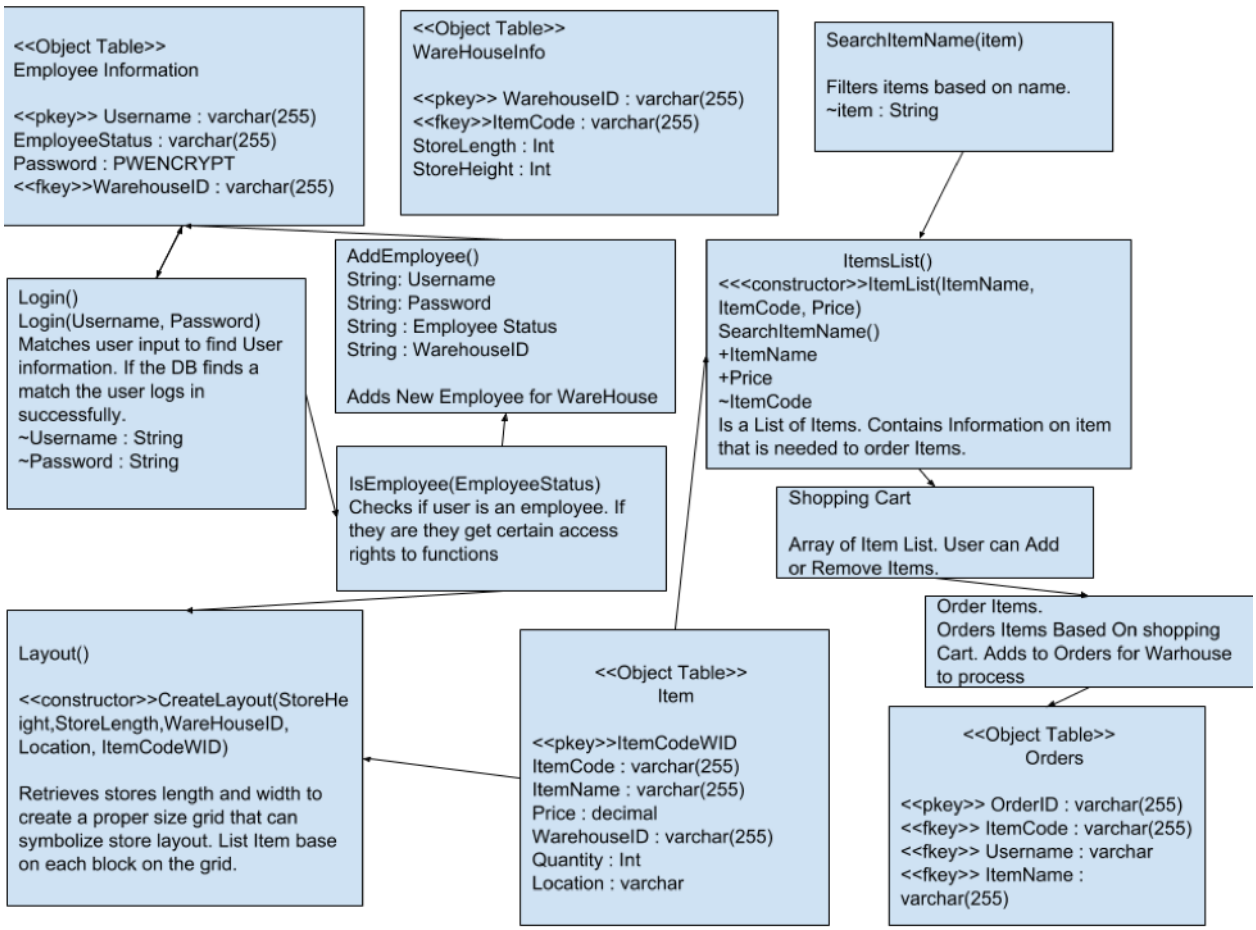
Use Case 4 - User Updates information	
Initiating Actor	Customer
Actors Goals	Update accounts details.
Pre-Conditions	User has made an account, and has logged in.
Post-Conditions	Users details will be changed to what they desire.

Flow Of Events	1) User selects the drop down menu button. 2) User selects gear box at the bottom of the menu drop down. 3) Now a page will be populated with the current information of the customer. 4) User can now update information such as birthday. 5) Once user has made the appropriate changes, they can click save and the updated information. 6) Updated information is sent to database.

Domain Analysis

5A Domain Model

UML Diagram



Interaction Diagrams

The Interaction Diagrams needed little change because our plan of work is mostly to polish to this point. Our application is still planned to do the same things just with a little more polishing for the final demo.

Figure 1 - Use Case 1

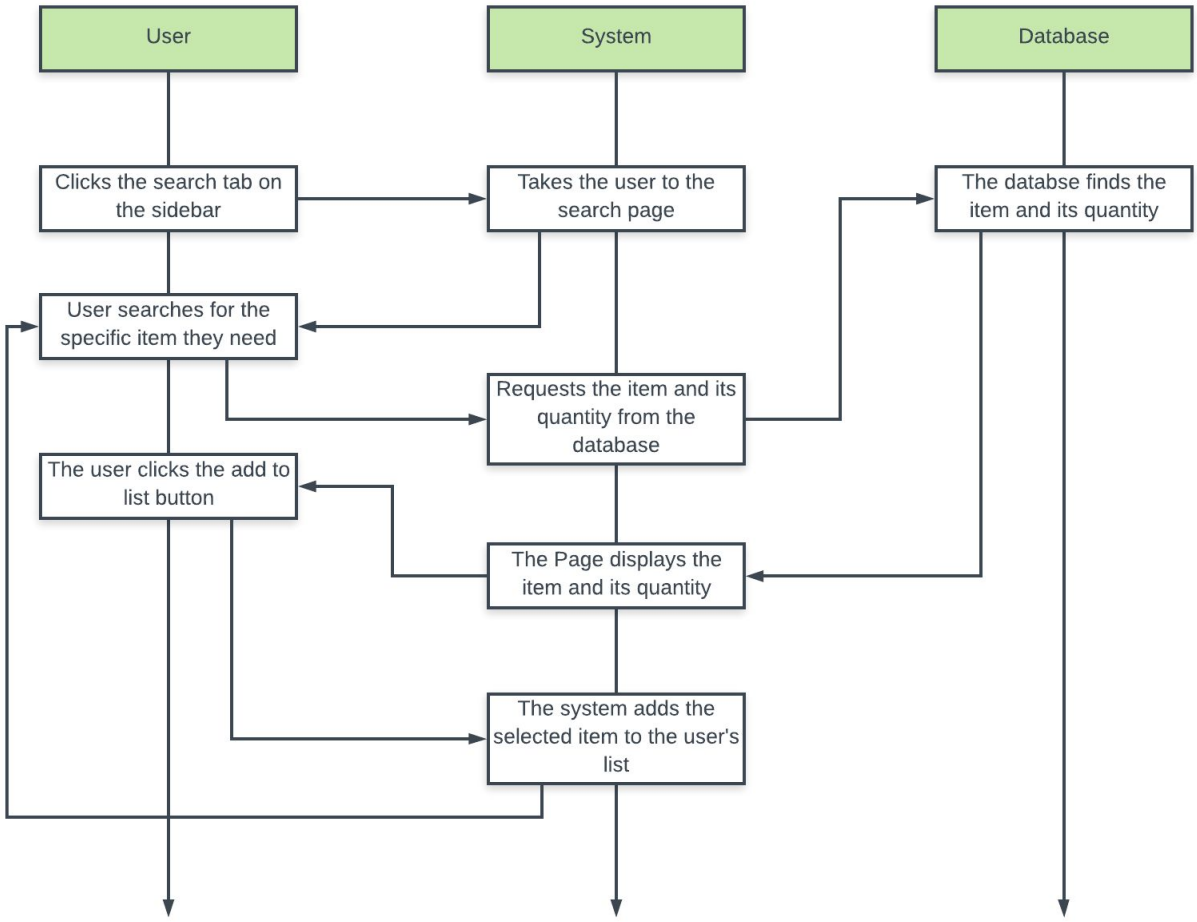


Figure 1 shows the model of use case 1 which is the user adding an item to their list. First the user will click on the sidebar menu button and will click on the “search” tab. This will take them to the search page which then the user will search for the item they want. The system will request the item from the database and the database will send the item information and quantity back to the user. The user will then add the item to their personal list by clicking the “add item” button. The user can repeat this process for as many items as they need.

Figure 2 - Use Case 2

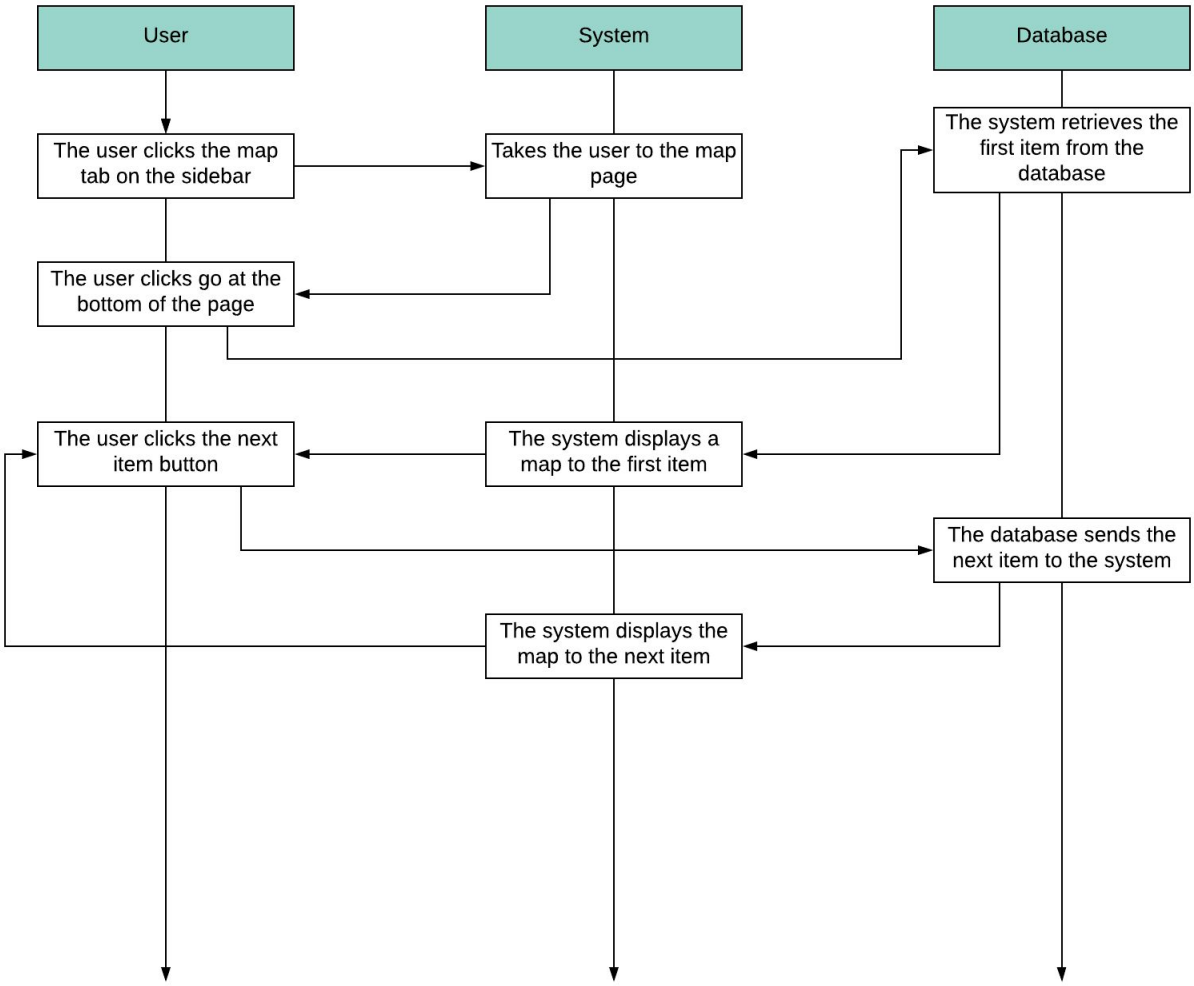


Figure 2 shows the model for use case 2 which is the user using the map to find their items. First the user will click the sidebar menu button and click the “map” tab. This will take them to the map page which then the user will use the on screen map to find the easiest path to their item. The user will click the “go” button at the bottom of the page which will retrieve the first item in the list from the database and display the map for the item in the map display box. Once the user has found their item they can click the “next” button and the system will display the map for the next item in the database’s user item list. The user can repeat the next item step until they have found all of the items in their list.

Figure 3 - Use Case 3

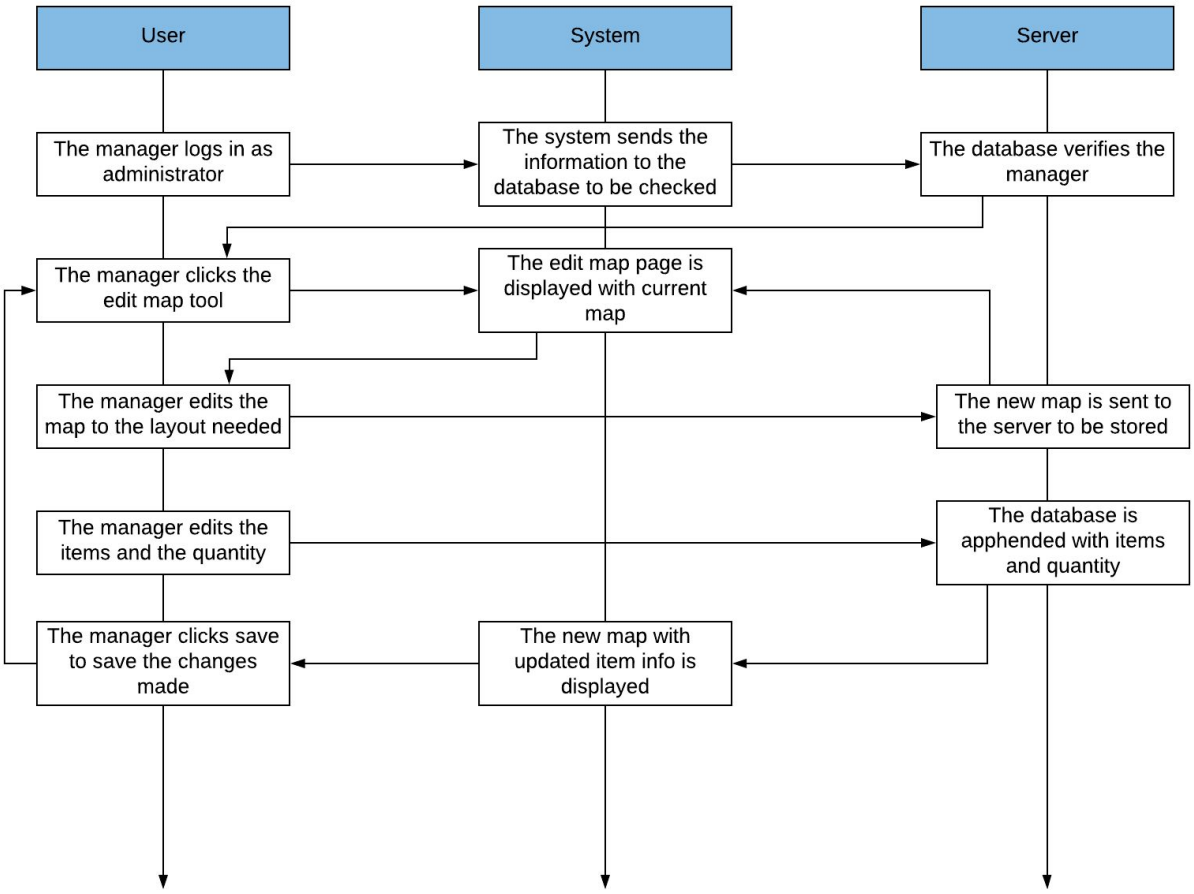


Figure 3 shows the model for use case 3 which is the manager editing and saving the layout for the base map. First the manager will log in as an administrator, which will be verified by the database. Then the manager will click the “edit map” tool to edit the base map that the user sees. The manager can then edit the map to the layout desired with the given tools. When the layout is to the manager’s liking, the manager will click the “save” button which will send the new map to the server to replace the old one. The manager can then edit the items, their quantity, and where they are place in the store. The manager can click the save button one final time and the final map with all of the updated items will be ready to be used by the user.

Figure 4 - Use Case 4

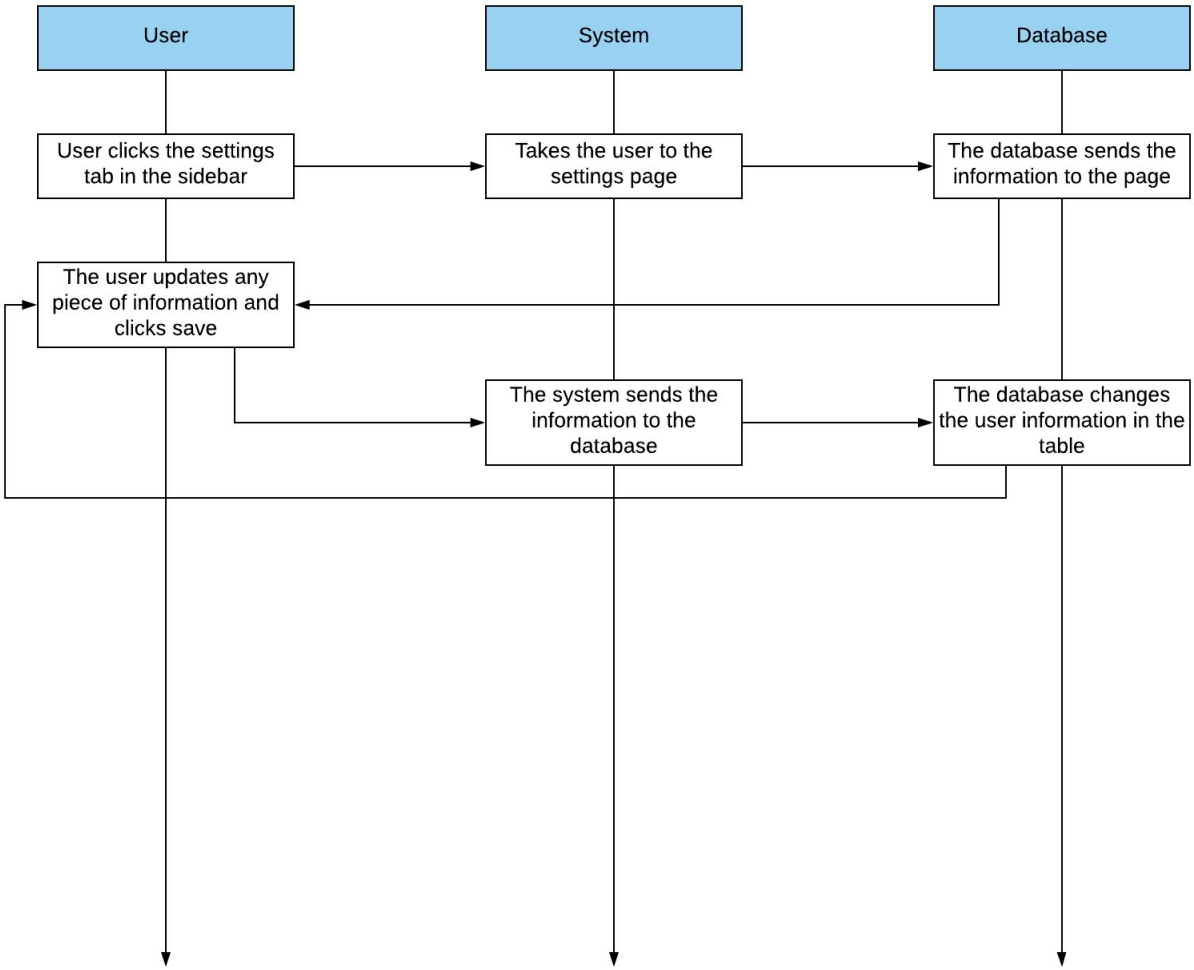


Figure 4 shows the model for use case 4 which is the user editing their personal information. The user will click on the “settings” tab in the sidebar which will take them to the corresponding page. The system retrieves the information from the database and displays it on the page. The user can update any information they need to, and will then save that information, which will be sent to the database and saved. The user can repeat this process when need be.

Class Diagram and Interface Specification

Traceability Matrix

Test Case Number	Test Case	Steps	Test Data	Expected Results
1	Login	Enter Username Enter Password	User = user password = pass	Moves to Splashpage & creates Session Login
2	Logout	Click Logout		Ends Login Session Redirects to Login Page
3	Search	Enter Text in search bar	itemName = USB	Returns list of items that contain USB in their name
4	Add Item to Cart	Click Add Item button next to selected item	ItemName = USB ItemCost = 30.00	Adds item to Cart and generates total price. Creates Session for items.
5	Remove Item from Cart	Click Remove item next to items in shopping cart page.	ItemCode = Y1234 ItemQuantity = 1	Removes item from Cart Session
6	Order Items	Click Order Items in shopping cart Page	$_SESSION['Cart']	Sends Array of items to the Database to be processed
7	Add Employee	Go to Admin Page. Enter Information to add Employee.	EmployeeID= test1 EmployeeName= test WarehouseID= Temp WarehouseName = Temp AdminStatus=Admin	Sends data to add user to database so employees can login to website and process orders
8	Add Warehouse	Enter Warehouse ID, Warehouse name, Length, and Width of warehouse	WarehouseID= Temp WarehouseName = Temp Width = 150 Length= 75	Sends warehouse data to sql server to be added and be process for user later
9	Get Warehouse Map	Process a grid layout of the warehouse automatically maping each item to the grid that they are located in		Retrieves warehouse information from Session and retreive all items located in the warehouse along width the warehouses dimensions
10	Add Items to Warehouse Grid	User clicks on a block in the warehouse grid where they enter the item information to be added in that grid spot	ItemName = Usb ItemCode = Y1234 Quantity = 10 Cost = 15.00	Adds Item to designated spot in the warehouse to be loaded later. It adds it to the warehouseItems table and Items table
11	Process Orders	Process orders by highlighting spots in the warehouse grid where the item is located. Then it will remove the items from the warehouse Inventory table	Click button next to designated order	Highlights items in grid. Removes items from warehouseItems table and Items table when it is finished.

Design Patterns: (What needs Improvement?)

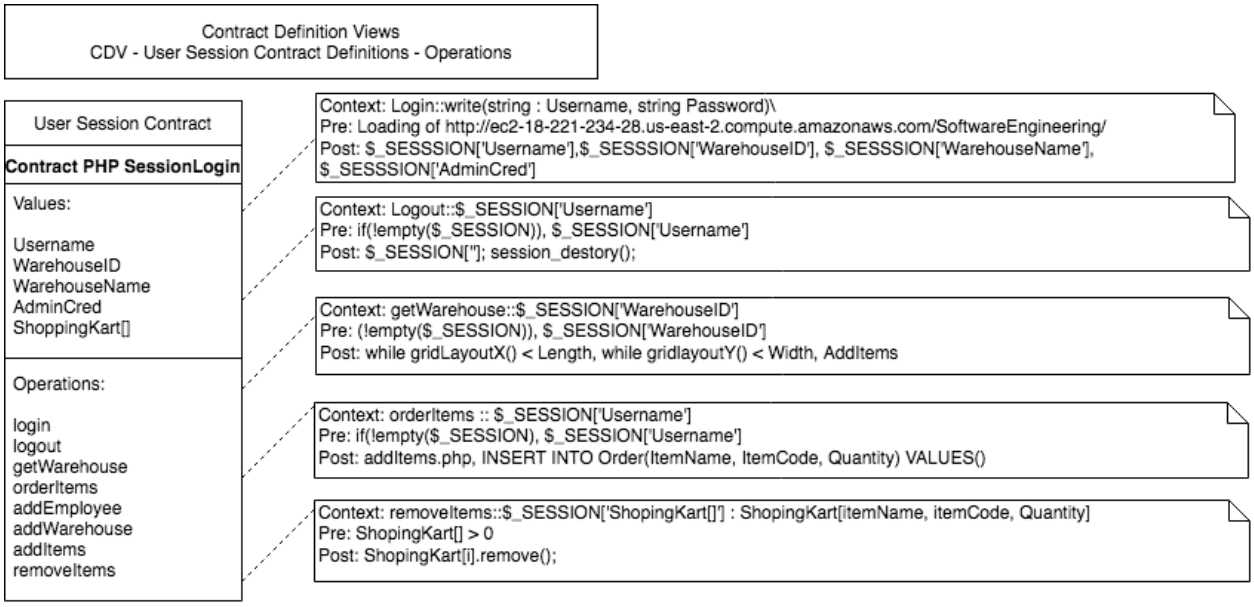
Behavioral	Structural	Creational
Strategy Delegation of Time Skills: Design, Logic, and Leadership Mediation	Delegation Planning Ahead Research Writing Skills Commitment	Communication of ideas Balancing reality and goals Conceptual Design

Behavioral: To improve our behavioral skills we should of communicated as a team better. If we did a better job at communicating we could of understood each others ideas better. We should of strategized and looked ahead as software engineering for a website is difficult to formalized into a readable formats such as OCL and UML. We as a team needed to delegate more time to planning ahead and getting work done early. If we communicated our skills and had a more centralized leadership we believe our group work would have had more structure.

Structural: In hindsight we believe we should of improved in structure the most. Delegating time management and planning for each individual on deadlines would of helped us dramatically. We had issues were we certain parts of our website needed to be done before others' could start on their parts but we had to wait for the other to finish their job.

Creational: I believe, we as a group, were strong in out creational aspirations and views for what we wanted to develop. I do believe we should have balanced realities and goals since creating a pathing AI for our website was just not possible with everyone's' work load. If we conceptualized a more realistic approach and understood our capabilities we could of focus of the website and have a website with a few structural sound functions instead of a lot of functions with a weak design.

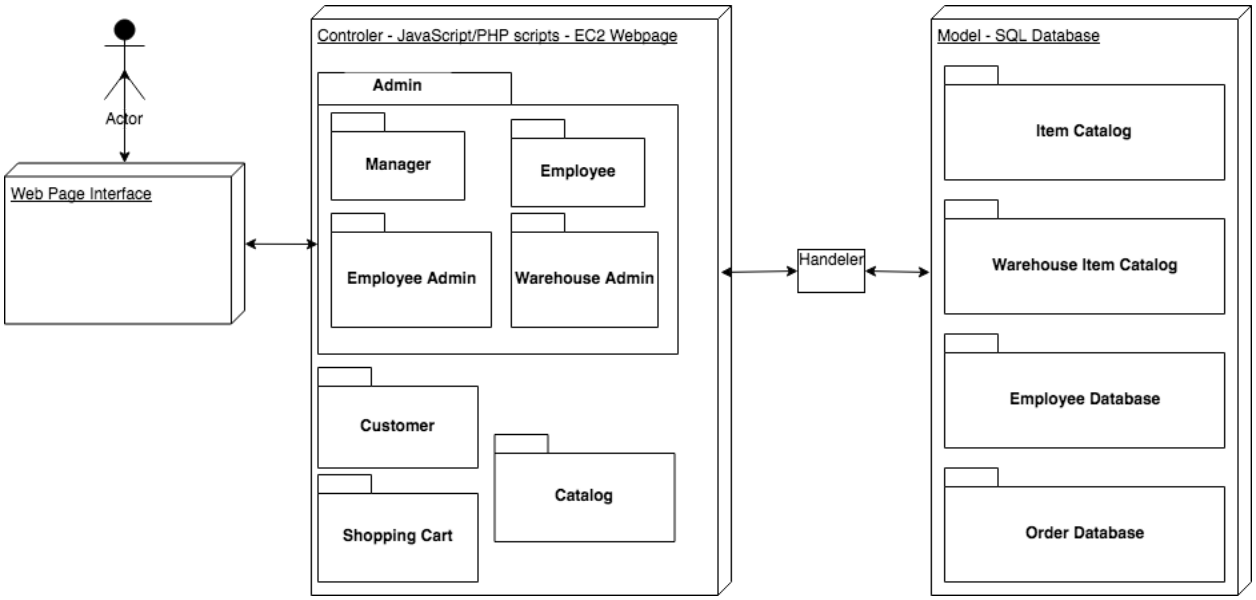
OCL Contract



System Architecture

a.) Our architectural style is Event-driven or implicit invocation. The main system (server) relies on inputs from the user (client) in order to determine the action that should be taken. This is true for almost all actions except for pre programmed actions that require no user intervention such as displaying the login screen initially and if the user is already logged in, redirecting to the main page.

b.)



c.) Since our application is web based, that implies that there will be two main actors in a connection. The server and the client. The server contains the database of items created with SQL and the server side php scripts to fulfill client requests. The client will be any device utilizing the services of the server. The client will utilize a clean user interface in order to create requests and receive data from the server. This data will mostly consist of item information that is relevant while shopping such as: item name, price, quantity, and physical location. Well will also implement the EC2 amazon web servers for our controller.

d.) Our application requires persistent data storage to hold the information on each item located in the store. It does this by utilizing a relational database (SQL) located on a remote server.

<<Object Table>> Item Catalog	<<Object Table>> WareHouse Item Info	<<Object Table>> Orders	<<Object Table>> Employee Database
<<pkey>>ItemCodeWID ItemCode : varchar(255) ItemName : varchar(255) Price : decimal WarehouseID : varchar(255) Quantity : Int Location : varchar	<<pkey>> WarehouseID : varchar(255) <<fkey>>ItemCode : varchar(255) StoreLength : Int StoreHeight : Int	<<pkey>> OrderID : varchar(255) <<fkey>> ItemCode :varchar(255) <<fkey>> Username : varchar <<fkey>> ItemName:varchar(255)	<<pkey>> Username : varchar(255) EmployeeStatus : varchar(255) Password : PWENCRYPT <<fkey>>WarehouseID : varchar(255)

e.) The network protocol utilized by our server system is the TCP/IP suite of protocols. More specifically, to find the SQL server the UDP protocol is used. Communication between SQL server and client uses TCP protocol. The client utilizes both HTTP protocol, to connect to the web server, and TCP/IP to transfer data back and forth between the SQL server and the end user.

f.) Our system is event-driven as specified in part a.) under the System Architecture header. Users may generate web application actions in an arbitrary order minus some required initial steps such as logging in. Our system is not time dependent and has no concern for real time. Our system also does not utilize threads or concurrent processing as it is not required for our project.

g.) Our system relies on quite a numerous amount of hardware since it requires a client and server, which is usually two different machines. The server requirements are as follows:

PHP 5.5+ require at least Windows 2008/Vista, or 2008r2, 2012, 2012r2, 2016 or 7, 8, 8.1, 10. Either 32-Bit or 64-bit (aka X86 or X64. PHP does not run on Windows RT/WOA/ARM). As of PHP 7.2.0 Windows 2008 and Vista are no longer supported. PHP requires the Visual C runtime(CRT). Many applications require that so it may already be installed. PHP 5.5 and 5.6 require VC CRT 11 (Visual Studio 2012)

Since SQL will also be present on the server, the system requirements for that are:

Memory Requirements: a minimum of 1 GB, but optimally your machine should have 4 or more GB of memory as database size increases to ensure optimal performance.

Processor Speed: a minimum of 64 bit processor and clock speed of 1.4 GHz is required, but the preferred is a 64 bit processor that is 2.0 GHz or faster.

Processor Types: The list of acceptable processor types are: AMD Opteron, AMD Athlon 64, Intel Xeon with Intel EM64T support, Intel Pentium IV with EM64T support

Disk Space Requirements:

Database Engine and data files, Replication, Full-Text Search, and Data Quality Services	1480 MB
Database Engine (as above) with R Services (In-Database)	2744 MB
Database Engine (as above) with PolyBase Query Service for External Data	4194 MB
Analysis Services and data files	698 MB
Reporting Services	967 MB
Microsoft R Server (Standalone)	280 MB
Reporting Services - SharePoint	1203 MB
Reporting Services Add-in for SharePoint Products	325 MB

Data Quality Client	121 MB
Client Tools Connectivity	328 MB
Integration Services	306 MB
Client Components (other than SQL Server Books Online components and Integration Services tools)	445 MB
Master Data Services	280 MB
SQL Server Books Online Components to view and manage help content*	27 MB
All Features	8030 MB

The system requirements for the client are significantly less. The only client requirement is a functional operating system and web browser. Chrome seems to be the biggest memory hog of all web browsers currently so we will use that as a baseline for our client requirements. Those requirements are, Google Chrome will run on computers equipped with a Pentium 4 processor or higher, which encompasses most machines manufactured since 2001. The computer must have approximately 100MB of free hard drive space and 128MB of RAM. The oldest version of Windows supported by Chrome is Windows XP with Service Pack 2 installed. Chrome also runs on computers with Windows Vista or Windows 7 installed.

Algorithms and Data Structures

A.) Algorithms

Most of our application does not rely on algorithms as is more dependant on our database structure. We plan on introducing two quality algorithms to our application. The first algorithm will be implementing store layout to match and fit the webpage. When we return the store width and height form out database we plan on creating a GUI that can change with each different store layout. We plan on use a basic mathematical formula where each 5 ft squared unit of the layout will become a squared grid on the GUI. We will set a max width and length for the GUI and have the squared grids resize accordingly.

E.g. Store Layout: Width = 250ft Length = 400 → 50x80 grid is then created and each grids size will be adjusted accordingly to the allocated size for the GUI

GUI length = Length / 5
GUI Width = Length /5

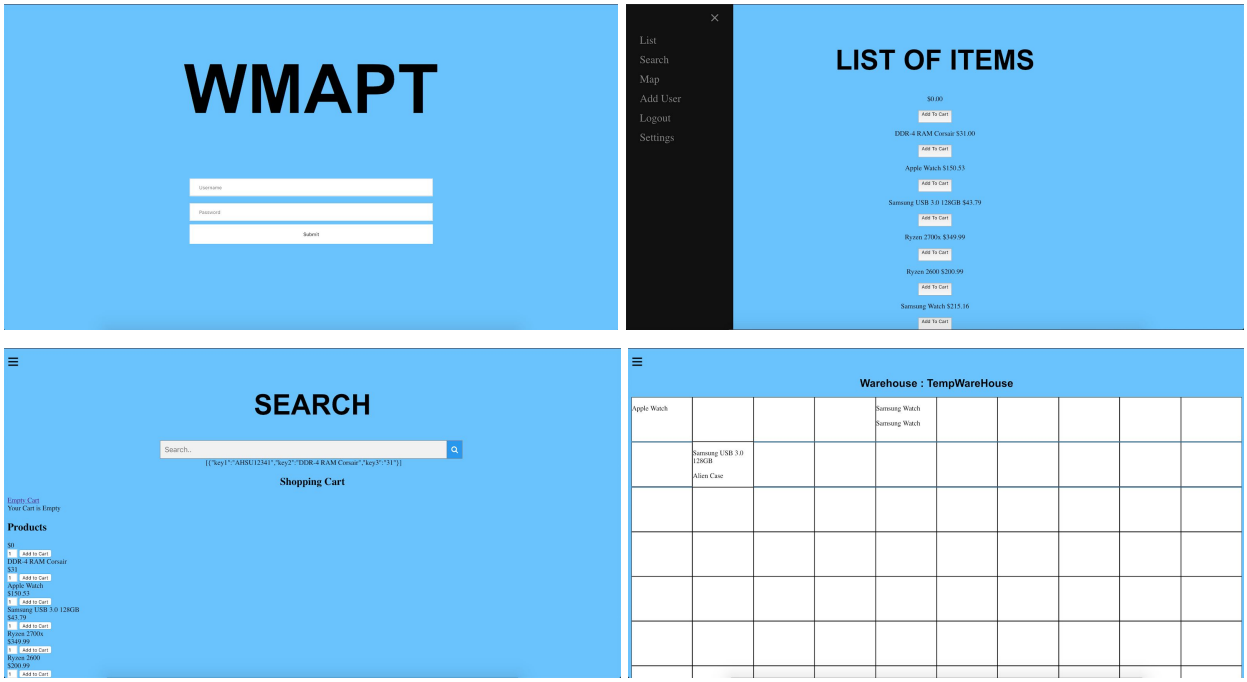
After that we plan on introducing a system to implement an algorithm to find any item in that warehouse in the fastest time. We will do this by getting the items location on the grid and the location of the employee. The location of the employee can be put anywhere on the GUI map. From there we will use Dijkstra's algorithm to find the best available path for the employee to take. The GUI will then display the route the employee should take.

B.) Data Structures

Since we will be mainly using PHP and Javascript for our controller's data structures we will not have any complex data structures there. Mostly we will create Objects such as an Item List that contain two strings and an integer. These Objects include ItemList[String, String, int], ItemLocation[int, int], and StoreLayout[int, int].

UI Design and Implementation

The UI design has changed considerably from its initial concept. The splash-page has become a comprehensive list of the items in the database with a shopping cart at the bottom of the page. The Search page also has a list of items below the search bar. These items are the ones that can be added to the shopping cart. The Search in its current configuration only lists off the items that have been searched for, they have to be added using the buttons below the results. The map is similar to what we initially proposed and still consists of a grid with items that can be used. Much of the design still stands as it was originally proposed, including the color, layout, the sidebar.



Design of Test

Test: TC01
Function Tested: Login(String, String): String

Call Function Pass	User gains Access to Administrative Rights
Call Function Fail	A String containing null is returned and the user is redirected to the login page

Test: TC03
Function Tested: AddItem(String, String): ItemList

Call Function Pass	User is notified that the item was correctly added to the Database and the webpage will reload with an update item list array
Call Function Fail	User is notified that the item was not correctly added to the Database and will send back a warning message

Test: TC05
Function Tested: RemoveItem(String, String): ItemList

Call Function Pass	User is notified that the item was correctly removed from the Database and the webpage will reload with an update item list array
Call Function Fail	User is notified that the item was not correctly removed from the Database

Test: TC06
Function Tested: GetItemList(): ItemList

Call Function Pass	User gets a webpage with all items loaded
Call Function Fail	User is notified that the webpage can not currently load any items from the database

C.)Testing Strategy

We plan on testing each of these functions separately. We will conduct each individual test and examine every possible outcome. We will make sure the user will not have much text input to prevent an even wider margin of error.

History of Work

As a design group we have tried to stay close to our deadlines but we have definitely not met some of them as we had in Report 1 or 2. The login page and the design of the webpage were finished pretty soon in the production of the software. We were definitely ambitious with some of the deadlines. The creation of items and stock were not completed until around the 20th of November which is about a month past the deadline we originally expected. The biggest factor that led into these missed deadlines were the complexity of the tasks we are trying to accomplish. We did not expect some of the tasks to be as difficult as they happened to be. As in both schedules, we expected the application to be completed by the 24th of November which has already passed. Now it is looking more like near the 13th of December.

Current Status

The current status of the application is nearly complete. The map is running though not completely functional. The list of items is up and items can be placed into a shopping cart to be checked out. Still to be worked on is the pathway on the map and the settings page for the user.

The key accomplishments on the project are:

- A working map with Items and their quantity present
- A working login and logout system for users
- A working shopping cart and add to cart mechanism
- A pathway on the map

Future Work

Future work on this project would consist of refining all of the components and enabling them to work across multiple systems. We would need to make the UI a little easier to use and nicer to look at. Presumably all bugs would be squashed with extensive future work and the application would work to the full extent we planned. The primary focus with future work would be to enable access to more that just admins and basic users and to enable the app to work across multiple warehouse environments with separate logins.

References

- Decker, Fred. “Google Chrome Software Requirements.” Small Business - Chron.com, Chron.com, 26 Oct. 2016, smallbusiness.chron.com/google-chrome-software-requirements-48820.html.
- “Install Requirements - Manual.” Php, php.net/manual/en/install.windows.requirements.php.
- MashaMSFT. “Hardware and Software Requirements for Installing SQL Server 2016.” 2016 | Microsoft Docs, docs.microsoft.com/en-us/sql/sql-server/install/hardware-and-software-requirements-for-installing-sql-server?view=sql-server-2017#pmosr.
- http://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf
- “GUI.” *Tech Terms*, techterms.com/definition/gui.
- “Web Browser.” *Tech Terms*, techterms.com/definition/web_browser.
- “Application.” *Tech Terms*, techterms.com/definition/application.
- “Interface.” *Tech Terms*, techterms.com/definition/interface.
- “Database.” *Tech Terms*, techterms.com/definition/database.