Warehouse Management System

PennWest California

Specification Document

CSC 4900 – Senior Project 1

Group Members:

Brian Colditz

Shakear Wilson

Ty Kress

William Serowik

Instructor Comments/Evaluation

Table of Contents

Abstract

The Warehouse Management System intends to help warehouse managers always track and manage their inventory; this includes ongoing shipments, deliveries, and supplies within the warehouse. While existing systems are available, they are less affordable and rely on manual or outdated systems to track stock, leading to inefficiencies, errors, and increased operational costs. The team aims to develop a warehouse management system to simplify inventory tracking and management. It will focus on user-friendliness, compatibility, and practical use by warehouse managers and workers. Implementing this system will allow these businesses to streamline operations, reduce errors, and increase productivity. This document will show the development management systems and the project plan.

Description of the Document

Purpose and Use:

This document outlines the scope and detailed requirements for the project, covering aspects such as product functionality and system needs. It will serve as a reference for both the developers and the client to establish the terms of their agreement. The client reserves the right to dispute the document's contents to ensure the product meets its desired specifications. Once the terms are accepted, this document will serve as a binding contract.

Intended Audience:

The intended audience for this document includes the development team and the client. The client will review the document to verify that it fulfills all their software requirements. If the client disagrees with any specifications in the document, revisions should be made until mutual agreement. This alignment is crucial, as the development team will rely on this document as a guide while building the final project.

System Description

Overview:

The objective of our WMS is to create a system that has a user-friendly interface, simplifies management, and is compatible with most devices. This creates an intuitive, responsive design for warehouse managers and workers, minimizing the learning curve. It will include QR and Barcode scanning for easy product management and auditing. The project will utilize Bootstrap as the front-end framework to build a responsive interface, and AJAX and JavaScript will be used to improve user interaction. For our simplified management, a streamlined warehouse operation will be implemented, introducing role-based access to ensure data integrity and secure management of products. This project will be able to handle automating inventory updates, stock management, and reporting.

Additionally, the system will contain compatibility features that ensure cross-platform compatibility. These features can be achieved using Bootstrap to create a responsive design that works across desktop and mobile. A Docker environment is used to simplify deployment and ensure consistent performance across different systems. All WMS data must be stored, so

backend development with PHP will use object-oriented features for modular design and maintainability, utilizing MySQL for data storage and retrieval.

Environment and Constraints:

End User Profile:

The end-user profile for the Warehouse management system primarily includes warehouse managers and employees who need a simple and effective tool to help streamline inventory tracking and operations. As they oversee overall operations, warehouse managers need real-time visibility into inventory levels, shipments, and deliveries. They require tools for job assignment, stock auditing, report generation, and monitoring employee activity. The primary goal for warehouse managers is to increase operational efficiency, reduce errors, and maintain data accuracy.

Warehouse workers are another critical user group. They manage everyday responsibilities such as inventory shipping, stocking, and receiving. These users need a simple user interface to scan QR codes and barcodes and get updates on task assignments and stock locations. Employees would require the basic technical abilities to operate and use mobile devices or scanners with little instruction. They aim to do assignments precisely and on time while avoiding unnecessary complexity.

User Interaction:

To access the Warehouse Management System (WMS), users must have a user that is a warehouse manager to create an account for them by providing basic details, such as a username, password, and a valid email address. After the account is set up, the user will be prompted to log

in using their credentials to access the platform securely. As a second layer of security, users must enable two-factor authentication (2FA), which involves receiving a unique code via an authenticator app. This step is critical to ensuring secure access and preventing unauthorized individuals from reaching sensitive data within the system. After successful login and 2FA setup, the user lands on the main dashboard.

The main dashboard is a central hub where users can quickly view key metrics, inventory levels, and recent activities. To navigate through the application, users can use the sidebar; a menu that links to various sections of the WMS, such as Inventory, Shipments, Orders, and Reports. Selecting a section redirects the user to the chosen area, where they can view and manage related data. For example, in the Inventory section, users can view a detailed list of items, their current stock levels, and relevant details like SKU numbers and storage locations. The sidebar allows users to switch between sections easily, ensuring efficient navigation and ease of use across the system.

Users interact with tools designed for efficient stock control and data modification when managing inventory. Within the Inventory section, they can add new items, update stock quantities, or adjust item details as needed. For example, when receiving a new shipment, users can increase the stock level of items and add relevant information, such as batch numbers or expiration dates. Additionally, the system would allow users to scan barcodes or QR codes to streamline updates, making item identification quick and accurate. Users can also generate reports, track inventory turnover, and identify low-stock items, which supports dynamic restocking and efficient inventory management. This organized, secure, and accessible setup ensures users can monitor and control inventory effectively within the WMS.

Hardware Constraints:

To use the application, the end user must have internet access with either a PC or a mobile device. This is necessary to interact with the app and access the product information in real time. The app relies on internet connectivity to pull the most up-to-date data for each product, so it will not function properly without a stable connection.

In addition to internet access, the user will need a device capable of scanning QR codes or barcodes. This allows them to identify specific product details by scanning the corresponding code. A smartphone or tablet with a built-in scanner should be sufficient, but depending on the scale of their operation, users may also need a dedicated barcode scanner.

Another essential tool is a thermal printer. This will print product tags necessary for labeling the items. Thermal printers are ideal for this purpose because of their efficiency and low cost. The user must ensure the printer is compatible with their device and appropriately connected to generate the tags. Without these tools, the app's full functionality will be limited.

Software Constraints:

The software constraints are limited to internet access and a web browser. Any device that runs a modern web browser like Chrome, Safari, or another equivalent browser will work. The application will also work on mobile devices that run modern web browsers. If the end user is not hosting WMS on a local server, any device attempting to access WMS will require an internet connection to access the site.

Time Constraints:

The time constraints for this project will be on an academic schedule, meaning that we will have a whole college semester, which is approximately fifteen weeks, to complete this project. All group members will dedicate several hours per week to developing the project. Because we are on an academic schedule and have other commitments such as jobs and classes, we cannot dedicate our full time and attention to the project. It is expected that the members will set up time to work on the project's development and record updates that were made. As a team, we plan to properly manage the time given to execute the features presented in this document.

Cost Constraints:

The cost constraints for this project will be based on the need to deliver the necessary functionality while keeping the projects affordable for small businesses. Using free or open-source tools like Docker, PHP, AJAX, and Bootstrap reduces overall development costs. Hardware requirements such as PCs, mobile devices, and barcode scanners are estimated to cost between $100 and $1,000 per unit, depending on the scale of the warehouse. We will also need a domain to host our website, which could cost anywhere from $5-$50 a year.

Testing and quality control could require additional funding. Again, barcode and QR scanners cost around $50-$200 each. A testing environment, such as a temporary warehouse rental or a collaboration with a local business, could add additional expenses. Additionally, creating training material and other documentation could cost an additional $500, depending on whether the material is developed internally or outsourced, and the level of detail needed.

Maintenance, upgrades, and bug patches are ongoing expenses estimated at around 10-20% of the initial development cost per year. In contrast, many competing WMS systems charge between $500 and $1000 per facility per month or $100 and $200 per user. By focusing on cost-

effectiveness, this project seeks to provide a scalable and efficient solution aimed at smaller businesses while avoiding the high recurring costs characteristic of current systems.

Other Concerns:

The end-user must have Wi-Fi access and a device to access the website. This will allow the best utilization of all the functions of the WMS. If the user is unable to connect to the website, they will be unable to use any CRUD functionality and will not have access to the site. The user will need to understand that they will have to memorize their username and password to log in.

Acceptance Test Criteria:

Testers:

The warehouse management system's team members will be the project's leading testing and quality assurance team. Every member will conduct individual testing along with group gatherings for tests, including the hardware, software, and website interface and functionality. Intense testing will be done every time a member plans to add functionality to the project to ensure that each new portion of the project integrates successfully into the project. We plan on managing the project using a waterfall method, with our phases including requirements, design, implementation, testing, deployment, and maintenance.

Criteria for User Acceptance:

User Creation:

All user accounts must first be created by an authorized administrator, who inputs the user’s details, such as name, email address, and role, into the system. Once the account is created, the

system sends a verification email containing a secure, time-sensitive link to the user's registered email address. The user must click this link to verify their account and set a secure password, completing the activation process. This ensures that only verified users can access the WMS, with all actions logged for administrative auditing and security compliance.

User Authentication:

The login process will require users to provide a valid email address and a secure password. The system will validate these credentials against the database to ensure they are correct and associated with an active account. This initial step ensures that only authorized personnel can attempt to access the WMS.

Session Management:

Upon successful login, the system will establish a secure user session using technologies such as secure cookies and token-based authentication. This prevents unauthorized access to user accounts by ensuring that session tokens are valid and non-transferable.

Access to System Features:

Once logged in, users will have access to the full suite of functionalities offered by the WMS, tailored to their role-based permissions. The system will ensure that users can perform tasks such as inventory management, product audits, and QR code or barcode scanning without encountering barriers or performance issues.

User Feedback and Error Handling:

The system will include detailed error messages and guidance during the login process. For example, users will receive prompts if their email or password is incorrect, their email has not been verified, or their account has been locked due to multiple failed login attempts.

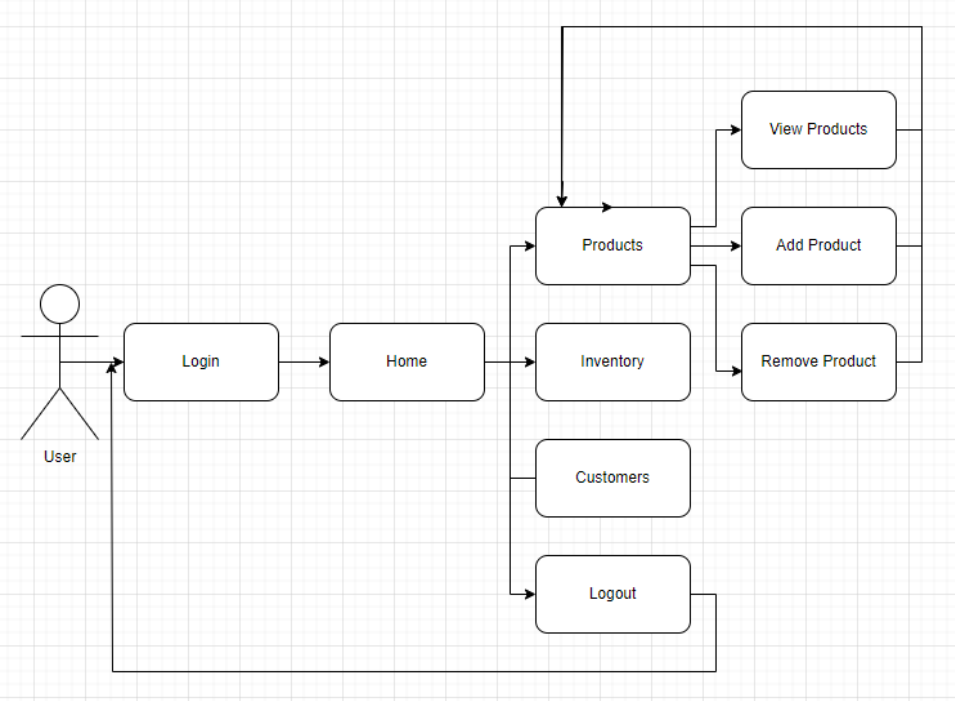
Integration of Separate Parts and Installation:

Integrating separate parts and installation to a WMS topically involves many steps, including Data Mapping and Transformation, that ensure data formats from the systems align. This is crucial for mapping SKU codes and inventory identifiers between systems. Lastly, Configuration and Customization will align with WMS processes, such as CRUD (create, read, update, delete), which configures workflows, user permissions, and data dashboards on the site. These steps ensure the WMS is effectively integrated and operational with all the necessary components to streamline warehouse activities.

System Modeling

Functional: Use Cases & Scenarios

The “add product” use case enables an employee or manager to add a new product to the WMS database. The user will log in to the WMS webpage, navigate the sidebar to the products dropdown, then the add product option, and use the interface to input the required information.



Normal Scenario:

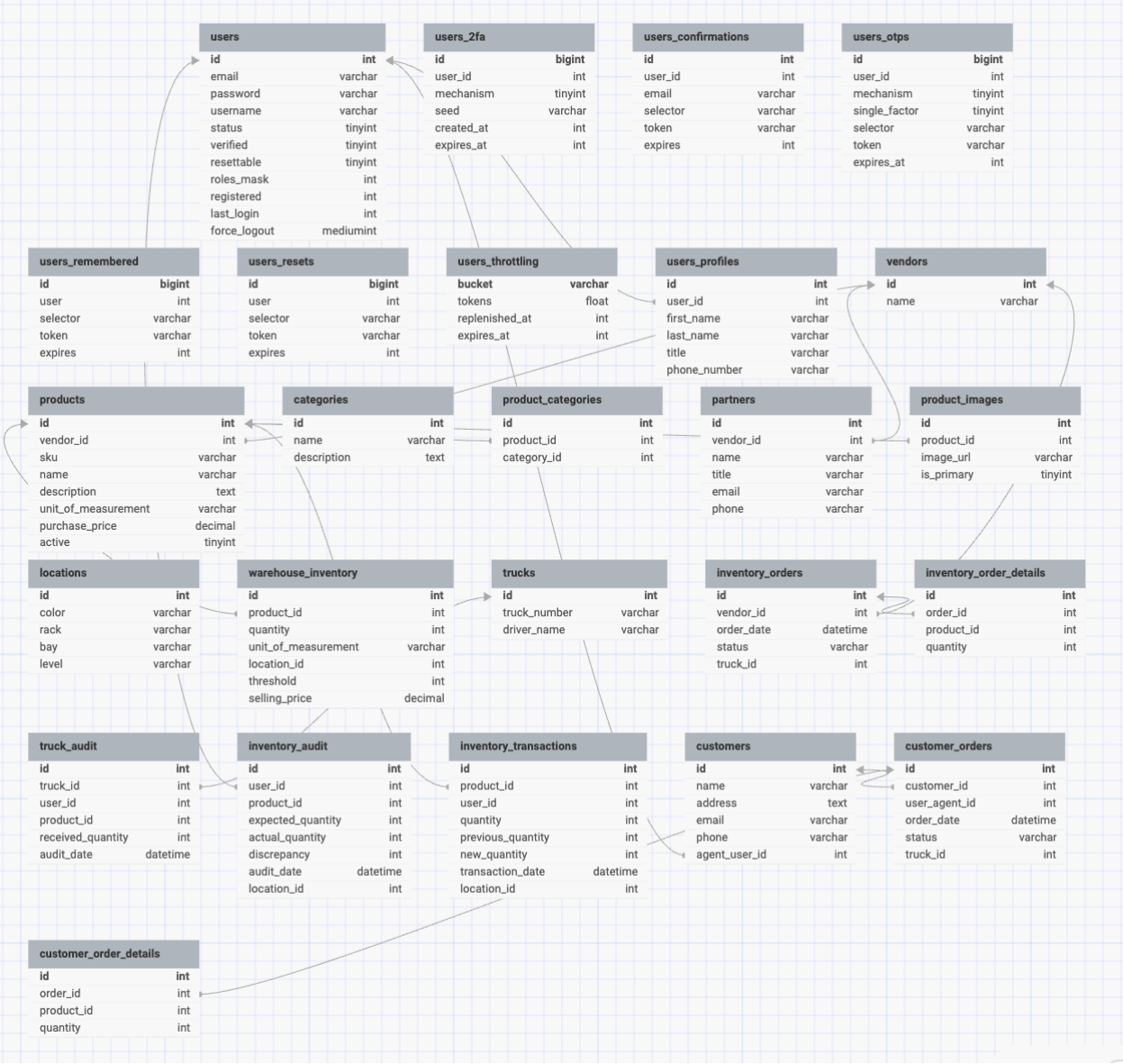
1. The user, employee, or manager goes to the webpage.
2. Enter username and password.
3. WMS's authentication component checks if the username and password match the username and password in the database.
4. WMS finds they match and lets the user access their account dashboard.
5. The user navigates the sidebar to the product dropdown.
6. The user selects the Add Product tab.
7. WMS opens the add product page.
8. The user adds the required data and clicks Add.
9. WMS checks all the necessary information was entered and is not a duplicate.
10. WMS adds the new product to the product database.
11. WMS lets the user know the data was added successfully.

Exception Scenario:

1. The user, employee, or manager goes to the webpage.
2. Enter username and password.
3. WMS's authentication component checks if the username and password match the username and password in the database.
4. WMS finds they match and lets the user access their account dashboard.
5. The user navigates the sidebar to the product dropdown.
6. The user selects the Add Product tab.
7. WMS opens the add product page.
8. The user adds the required data and clicks Add.
9. WMS checks all the necessary information was entered and is not a duplicate.
10. WMS finds the product is duplicated and lets the user know it is a duplicate.
11. WMS does not add the product to the database.

Entity: Class Diagrams

In WMS, each entity class corresponds to a table in the database. Each attribute corresponds to a column in the table. Below are WMS's entity class diagrams.



Class Descriptions

Below is a list of 3 entity classes, their descriptions, and their attributes. The other courses shown are structured similarly.

Users class:

The users class stores users' information, which is used in authentication. The attributes are id, email, password, username, status, verified, resettable, roles_mask, registered, last_login, and force_logout.

- **Id:** will store a unique combination of integers that will be used to identify the User throughout the web application. No user will be able to change this.
- **Email:** will store the User's email address.
- **Password:** This field stores a unique password the User will use to log into WMS. If the password is forgotten, the User can change it.
- **Username:** This field stores a unique name the User will use to log in to WMS. The User can change the username.
- **Status:** will store a tinyint to track whether a user is active.
- **Verified:** I will store a tinyint to indicate whether a user's email has been verified.
- **Resettable:** will store a tinyint to indicate whether a user can reset their password.
- **Roles_mask:** will store an int to indicate what role a user has been assigned to limit their capabilities.
- **Registered:** will store an int to
- **Last_login:** will store an int that will contain the date and time data of their most recent login
- **Force_logout:** This function stores a medium containing the date and time a user will be forced to log out if there is no activity.

Users_profiles class:

The users_profiles class stores users' personal information. It uses a foreign key user_id linked to the Id attribute in the users class. The attributes are id, user_id, first_name, last_name, title, and phone_number.

- **Id:** will store an int value to identify each user profile uniquely.
- **User_id:** will store an int value linked to the user class id attribute to link users to their profiles.
- **First_name:** will store a varchar value of the user's first name.
- **Last_name:** will store a varchar value of the user's last name.
- **Title:** will store a varchar value of the user's title within a company.
- **Phone_number:** will store a varchar value of the user's phone number.

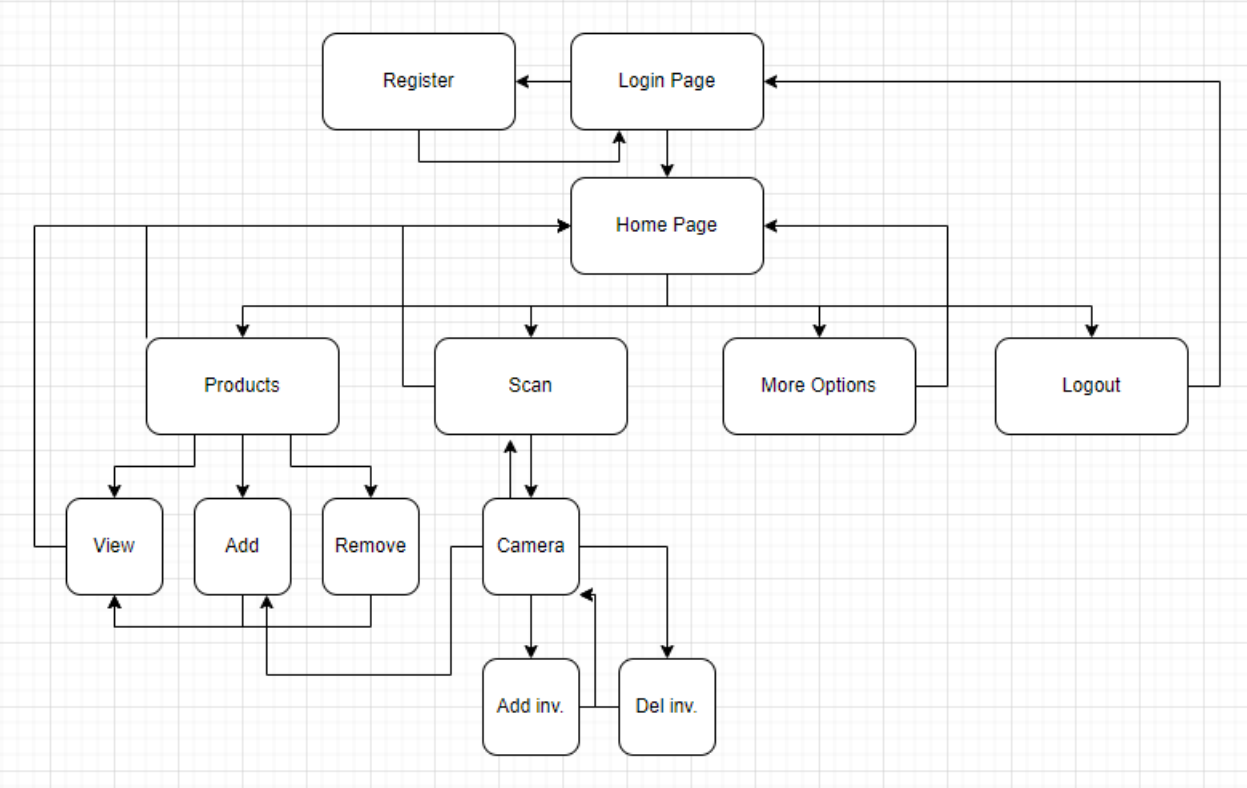
Products class:

The products class will store data relevant to any product that goes through the warehouse. This class uses a foreign key vendor_id linked to the Id attribute in the vendors class. The data in this class will be used for various calculations. The attributes are id, vendor_id, sku, name, description, unit_of_measurement, purchase_price, and active.

- **Id:** will store an int value to identify each product uniquely.
- **vendor_id:** will store an int value linked to the vendor's class id attribute to identify the product's vendors.
- **Sku:** will store a varchar value of a product's stock keeping unit.
- **Name:** will store a varchar of the product's name
- **Description:** will store a text value of a description of the product.

- **Unit_of_measurement:** will store a varchar value of the product's standard unit of measurement.
- **Purchase_price:** will store a decimal value of the product's purchase price from a vendor.
- **Active:** will store a tinyint value to indicate whether a vendor is selling a product.

Dynamic: Statechart



States

- The first state is when a user visits the web page, they are brought to the login page.
- From here, they can log in.

- Each state is a page. When the user registers, they are taken to a registration page where they enter account information, which is saved to the user database.
- The user will then be returned to the login page.
- Once the user logs in with their email and password, they will be taken to their homepage.
- For simplicity, examples of each possible state are not shown because they will all follow a similar pattern.
- Once on the homepage, a user will have many options and can navigate to many pages/states.
- For instance, the user can select the products dropdown, giving them the power to manage their products by navigating to the Products view and adding or removing pages. If they were to navigate to the add or remove states, they would be taken to a page where they would input product data and then click add/remove a product. The system would inform them if it failed, and they would stay in that state until they corrected the issue and resubmitted. The system will notify and redirect them to the view state if it is successful.
- On the view page, the user can view a complete list of products, sort and search the list, and revert to the home page when they leave.
- The user could also select the scan state to open their camera app to scan QR codes.
- Each code scanner will notify the user if an item is added/removed from the inventory and reopen the camera. If the product does not exist, they will be routed to the product add page.
- The user can also access the logout button, which will sign them out of their account and redirect them to the logout page.

- Many other options will be available that will follow a similar state flow to that of the product pages.

Events

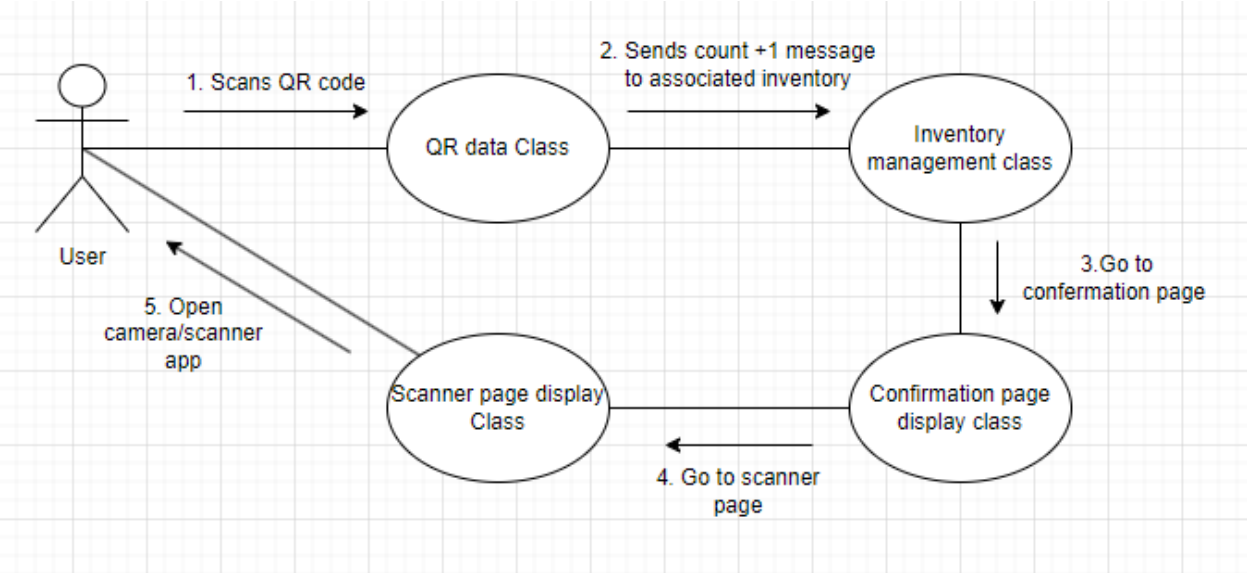
A user can create an event in many different ways while using WMS. They can filter and search for a product on the product view page, which displays results based on their search criteria. They can add/remove a product, which will add/remove a product to/from the database. Users can scan products into and out of the system. They can also edit their user data.

Transitions

A transition occurs every time a user selects an option on the homepage. On the login page, the user can click the register button. A transition will occur where the user will be taken to the registration page, where they will be prompted to enter information. After clicking register, they will initiate another transfer back to the login page, where they will be asked to enter their information. Once they click login in another transfer, they are moved to the homepage where their options grow vastly, and each result in another transfer to a new page where an action can be performed.

Collaboration Diagram

The added scanned inventory use case is shown in the collaboration diagram below.



1. The added scanned inventory use case is realized following these steps.
2. The employee scans a QR code on an item to be added to the inventory.
3. The QR data class identifies the item scanned and sends an add-on inventory message to the inventory management class.
4. The inventory management class adds one and redirects the employee from the camera/scanner app to a confirmation page.
5. The confirmation page is displayed until the employee taps the screen or clicks anywhere. Then, they are transferred to the scanner page.
6. The employee clicks the scan-in button to transfer to the camera/scanner app to continue scanning items.

Components / Tools Needed

This project will require several programs. Docker will be used to run a local environment to view the site as we work on it (2). Composer will manage software versions for libraries that are being used to ensure we are all working on the same code and package versions

(5). The GitHub repository will be used to share and track versions of code, what everyone is working on, and what needs to be worked on so everyone can code efficiently and not overlap (1). Lastly, everyone will need a coding environment such as Visual Studio Code or an equivalent code editor to write and edit their code. Once the project is complete, a server and domain will be necessary to operate the program and store the database in the long term.

Citations

About github and git. GitHub Docs. (n.d.). https://docs.github.com/en/get-started/start-your-journey/about-github-and-git

Manuals. Docker Documentation. (2024, October 24). https://docs.docker.com/manuals/

MySQL 8.4 Reference Manual. MySQL. (n.d.). https://dev.mysql.com/doc/refman/8.4/en/

Numeric Data Types. SAP. (n.d.). https://help.sap.com/docs/SAP_SQL_Anywhere/93079d4ba8e44920ae63ffb4def91f5b/81f712a96ce2101483c5ee3b141a30cc.html

What is composer?. DigitalOcean. (n.d.). https://www.digitalocean.com/community/tutorials/what-is-composer

Appendix: Glossary of Terms

AJAX - stands for Asynchronous JavaScript and XML. It is a set of web development techniques using various technologies to create dynamic web applications.

Bootstrap – a front-end framework for building responsive, mobile-first websites using pre-designed CSS, JavaScript, and HTML components.

CRUD - (create, read, update, delete) - The functionality that will be implemented to add, edit, and delete rows of data in tables.

Docker Environment – a lightweight containerized platform that allows developers to create, deploy, and run applications in isolated environments, ensuring consistency across different systems.

Interface – a physical connection between two computer systems, conversational syntax, a format for logical messages passed between the systems, and data – encoding structure understood by both systems.

JavaScript – an object-oriented computer programming language commonly used to create interactive effects within web browsers.

Mediumint – A MySQL data type that stores integer values that fit in a maximum of 3 Bytes (3).

MySQL – an open-source relational database management system that uses Structured Query Language (SQL) to store, retrieve and manipulate data.

PHP – a free, open-source scripting language used to create dynamic web pages and applications.

SKU - (stock keeping unit) - a unique alphanumeric code that retailers use to identify and track products.

Template – a form, mold or pattern used as a guide to make something.

Tinyint - A MySQL data type that stores integer values that fit in a maximum of 1 Bytes (3).

QR - (Quick response) - a two-dimensional barcode that can be scanned with a smartphone to access information.

Varchar - are variable-length strings. The length can be specified as a value from 0 to 65,535 (3).

WMS - (warehouse management system) - a software application that helps companies manage their warehouses operations.

Appendix: Team Details

The leader of this documentation was Ty Kress. All members took part in proofreading, formatting, and grammar evaluations. Each member of the team completed their section as well as helped to add to other members sections. The requirement document was created by all the following individual efforts:

Brian was responsible for Purpose and Use, Intended Audience, System Description Overview, Time Constraints, Cost Constraints, and class diagram.

Shakear was responsible for Abstract, Description of the Document, End User Profile, User Interaction, Hardware Constraints, Software Constraints, and Other Concerns.

Ty was responsible for System Modeling Functional: Use Cases & Scenarios, Entity: Class Diagrams, Class name / description / type Attributes, Dynamic: Statechart States, Events, and Transitions.

William was responsible for Acceptance Test Criteria, Testers, Criteria for User Acceptance, Integration of Separate Parts and Installation, Collaboration Diagrams, and Components / Tools Needed.

Other contributions that were coordinated by group members consisted of:

- Discord meetings/discussion
- In person discussion
- Formatting and planning outline of document
- Meeting at Writing Center
- Feedback and Proofreading

Appendix: Workflow Authentication

I, Brian Colditz, agree with the details defined in this document that represent functional requirements of WMS. Also, I agree that the work that was done as stated by this document

Signature:  Date: 11/18/2024

I, Ty Kress, agree with the details defined in this document that represent functional requirements of WMS. Also, I agree that the work that was done as stated by this document

Signature:  Date: 11/18/2024

I, Shakear Wilson, agree with the details defined in this document that represent functional requirements of WMS. Also, I agree that the work that was done as stated by this document

Signature: Date: 11/18/2024

I, William Serowik, agree with the details defined in this document that represent functional requirements of WMS. Also, I agree that the work that was done as stated by this document

Signature: William Serowik Date: 11/18/2024

Appendix: Writing Center report

The following notes are related to our 11/18/2024 12:30 PM EST appointment with Tatiana Kostovny.

Grammar:

- Commas: When using a phrase to describe a term, be sure to put it between two commas. There was a sentence that included a phrase describing the length of a college semester which lacked the second comma.

- Colon: I noticed that there was a sentence where you put a comma instead of a colon before listing items in a series. A colon would make the sentence a lot easier to read.

- Hyphen: There was a word where you did not need a hyphen. I added a comment where this occurred.

- Run-on: I noticed a run-on sentence that could be separated using a comma or splitting into two sentences.

Format:

- Items in a series: I noticed that items in a series are used several times throughout the document, specifically in the abstract. While this isn't necessarily a bad thing, switching up the sentence formatting can make it more interesting to read.

- Flow/Wording: There are multiple points throughout the document where sentences don't flow when reading it. I suggest changing the wording to make it more understandable.

- Wording: The phrase "such as" is used a lot. Again, this isn't a bad thing, but it's always a good idea to use a variety of vocabulary.