Project Design Documents

PRODUCT MANAGEMENT SYSTEM

1. Project Overview:

Organizations experience difficulties in inventory and warehouse management as they look for proper problems related to the monitoring of products, order placement, and management of stock level there are a variety of these issues, the Product Management System project has features that improve the usability, increase the rate of operations, and support the decision making. Firms need a Product Management System which is an application to manage a firm. Product/stock operation more effectively. It gives real-time exposure to commodity and inventory levels.

1.2 Problem Statement:

Paper-based processes and integrated systems have unique challenges such as

result in poor stock management, slow order delivery, and inadequate reliance on stock status reports. Business performance. The executive team is aware of the significant gap in the development of the warehouse management system that can enhance the efficiency of the warehouse, and the workflow contributes to the growth of the firm.

To aid in the achievement of the project goal, the identify the potential stakeholders with relation to the project and what they stand to gain interest:

- **Managers**

  In other words, managers need to be aware directly of inventory status at all times products, capacity, positions, and overall movement within a warehouse. This will enable them to make informed decisions to improve functionality and prevent all kinds of problems that arise.

- **Workers:**

Employees need an easy-to-use tool to help them with the many streams of workflows from receiving and placing items in their store to ordering. Thus the focus here is on selection, packaging, and shipping. This will allow them to meet orders easily properly and efficiently, all the time maintaining accurate stock counts.

- **Process Executives:**
  Evaluations and business insight professionals' work is expected to contain high levels of reporting and operation execution intelligence to track and evaluate key performance indicators that will help determine what areas need attention By keeping track of improvement, decisions are made based on data that will help enhance the entire business operations.
- **Distribution Companies**

  Our client's logistics partners have to be able to interact with the PMS to retrieve current order and shipping data. This will enable them to deliver the products more often and at a more accurate time hence improving the client experience.

- **End Users:**

  Customers expect timely order delivery, precise order fulfillment, and the delivery speed of transparent order tracking. These needs are going to have to be met to sustain a brand image and customer loyalty is a critical component of a favorable brand image.

1.3 Problem Solution:

To overcome these problems and satisfy the various stakeholders, we suggest an effective computerized and centralized management of stock in the warehouse. Supply Chain Management. The system will in a way provide tracking to foreign inventory on a real-time basis levels, locations, and statuses, this helps to ensure that the firm has the right stock at different levels, locations, and status ranges levels and reduces stock-outs.

- **Distribution Center Management:**

  The rationale for the recommendations made here is based on the expectation that overall organizational operations will be made more efficient automation of critical

warehouse operations including receiving, put-away, order fresh picking, packing as well as shipping.

- **Connectivity Solutions**
  The solution will integrate well with the client's current enterprise thus increasing efficiency. Different levels of resource planning (ERP), order management, and logistics systems will help organizations achieve visibility and integration across the different links in the supply chain.
- **Data Analysis**
  Complete coverage and analytics of the business environment capacities will be created to provide operation management specialists with intelligence help in the decision-making process at the strategic level.
- **Interface Design**
  Easy and intuitive website and application interfaces will be developed allowing Users of the information provided by this application including warehouse employees, managers, and customers complete their jobs.

  When applied, the current Product management problem will be solved by implementing a stock management system for our client management challenges alongside putting themselves in a strategic development frame competitiveness in the most competitive sector of the retail industry that is operating online.

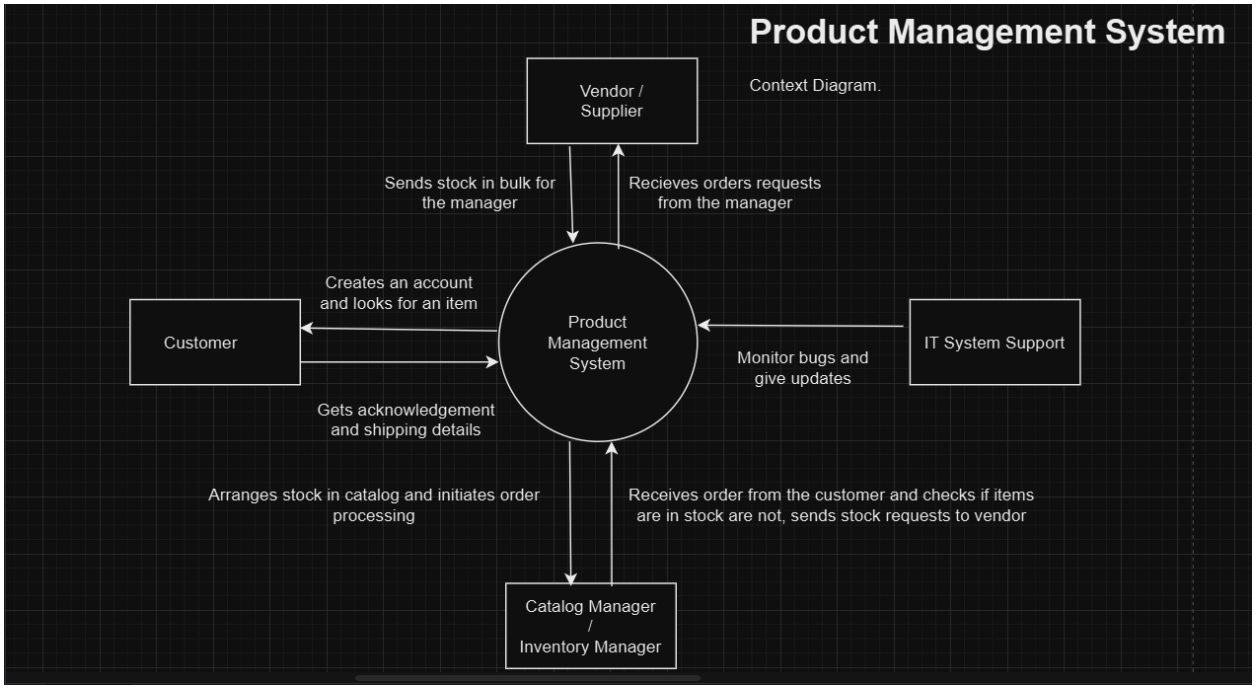
1.4 General model

1.4.1 Context diagram

Context diagram provides a broad view with regards to Product Management.

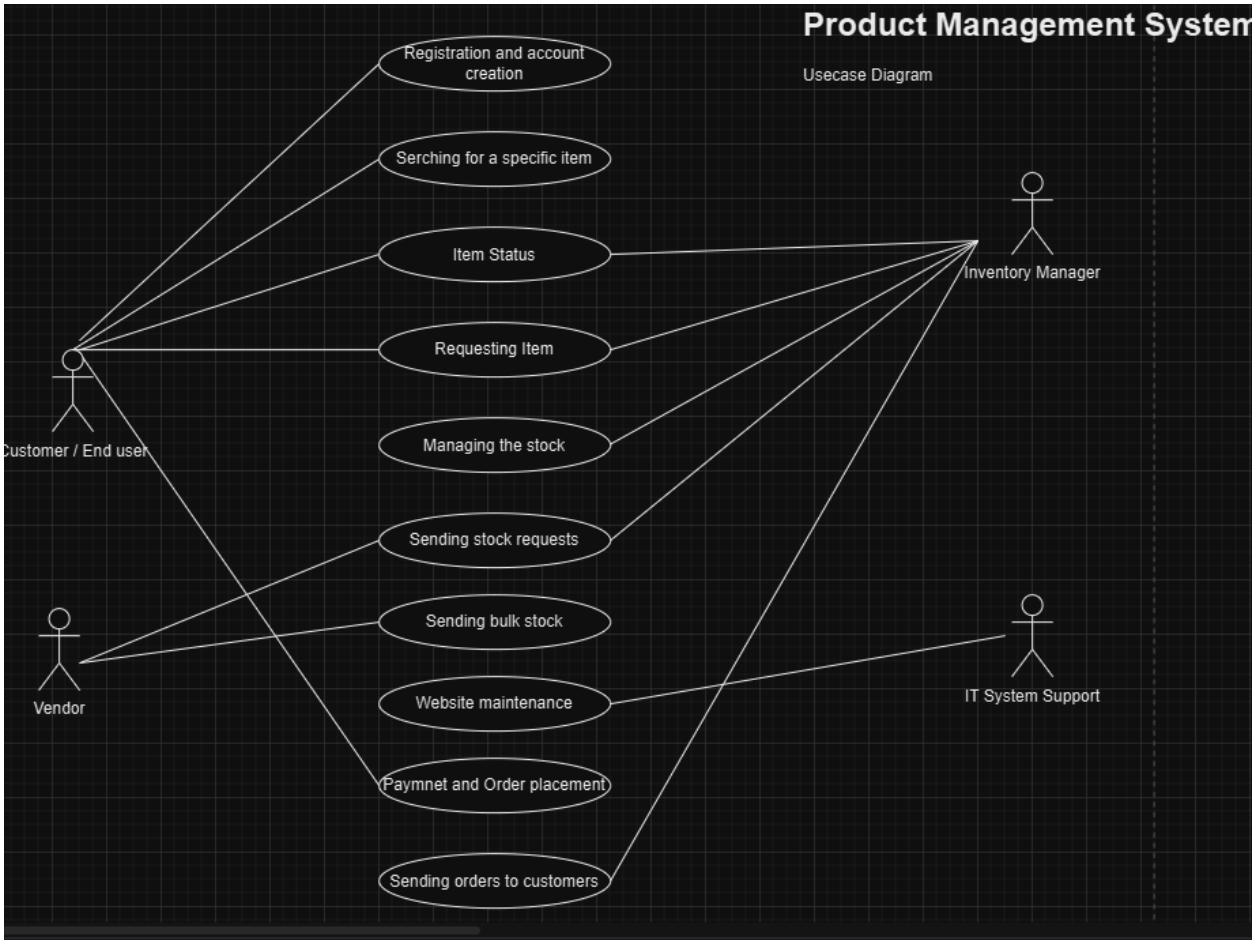
System and its relationships with the other large organizations. It also represents the customer interacting to interact with the system for navigating and viewing the order status of items. Using the, The Warehouse Manager used a system to deal with merchandise, locate the purchase order, and update the stock data. The Vendor uses the system to get restocking alerts and then influences the changes in-stock inventory information. It defines the scope of the system and how the information will be moving in the process between the system and its stakeholders, and

the more chance there is for a more examination of what is going on.



1.4.2 Use case diagram

Using a use case diagram, the architectural plan represents the product management system's significant parts and interactions. It tells the applicability of the basic use cases such as Order Products. Products Inventory, Control Products, Show Purchase Order, Open Purchase Order, Update, information about; Stock Status Report,& Stock Details; Restocking alerts. These examples list out the basic characteristics of the system in addition to the relations between the Products. New classes of actors that are obvious in the development are the Manager, Vendor, and Customer actors. In the other graphic, this is a brief explanation of the system's scope and functioning.



1.4.3 User Stories

1. As a Customer, I would want to search and order products, so that I can get the items I want online effortlessly.

2. As a Manager, I would want to manage and take over the control of products and their details, so that prominent data on products is displayed for the buyers.

3. As a vendor, I want to be able to manage all purchases, so that I can provide all the needed products to the end users as per their needs.

4. As a customer, I would like to keep track of my orders, so that I am always updated about my delivery status.

5. As a manager, I want to be able to create and access product status reports, so that I can oversee inventory levels.

6. As a Developer, I want to enhance mobile usage and ensure usability testing, so that users can access from mobile devices smoothly.

7. As a manager, I want to be able to update products' stock status details, so that the inventory is kept up-to-date.

8. As a vendor, I would like to be able to view purchase orders and history, so that I can easily understand and process them.

9. As a manager, I would like to be able to receive alerts automatically if products get out of stock so that I can prevent late delivery of products.

2. Architectural Overview:

Model View Controller Architecture:

1. Product Management View:

This view provides the product's personnel with an opportunity to work with and control inventory. This includes:

- **List available stock:**

  Describes the available products and their name, quantity in stock, and their position inside.

  The warehouse.

- **Browse and choose Functionality:**

  Customers can look for a product or a part number in the computer or they can categorically search by selecting parameters such as category or location.

- **Add or Delete Products:**

It enables a user to introduce new products into the inventory or demotic existing products in it.

- **Update the Product details:**

  Enables modification of product-related specifications including number, location, and other relevant

  attributes.

  Referring to GP_Data_Model:

  The View only receives information from the Controller and it has to show information to the user.

  The Model, namely, GP_Data_Model, concerns goods and quantities and other inventory details.

- **Order Processing View:**

  This view manages the customers' orders and their delivery. This includes:

- **Order details:**

  Shows a wealth of information about a certain order, product, quantity, and much more.

  Customer information.

- **Order Processing Actions:**

  Enables warehouse employees to change an order status P (processed, shipped) and handle inventory accordingly.

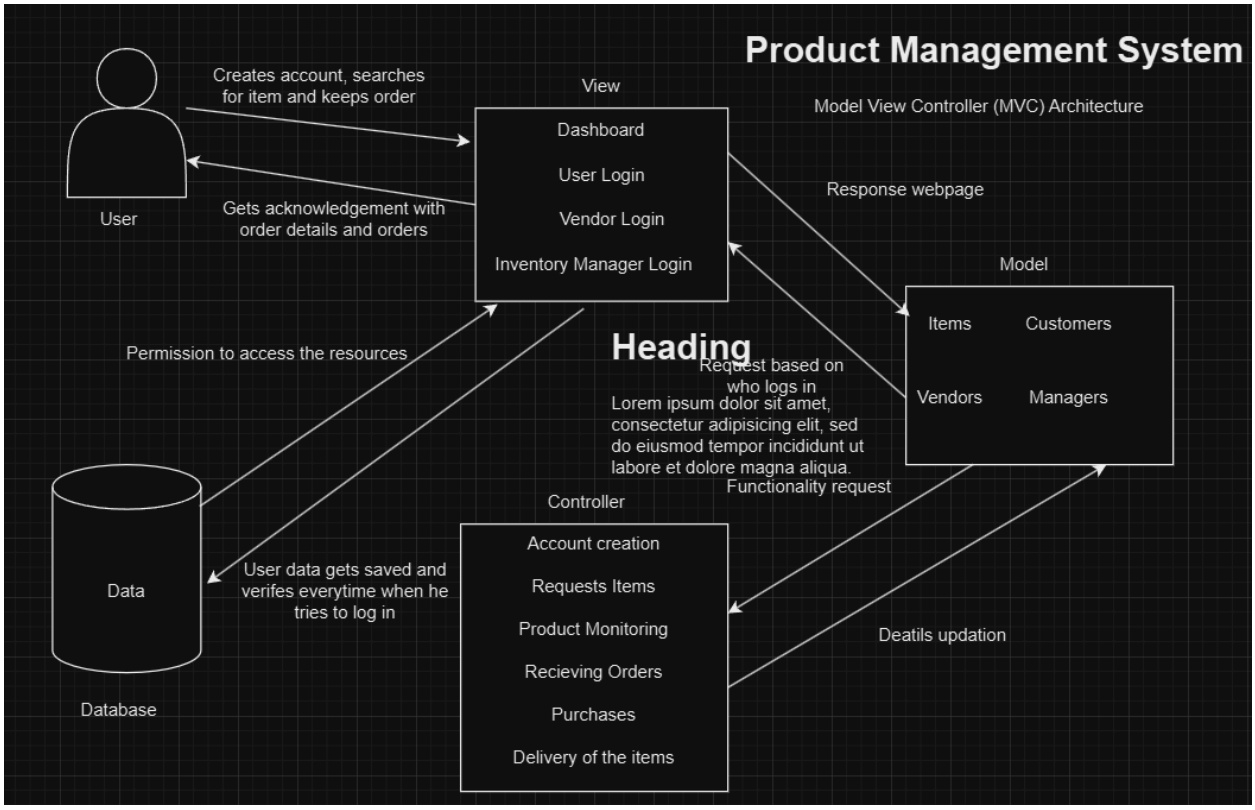
  Referring to GP_Data_Model:

  The View interacts with the Controller regarding the customer's actions regarding the order

  fulfillment.

  The Model (GP_Data_Model) performs requests on orders, products, and customer

information.

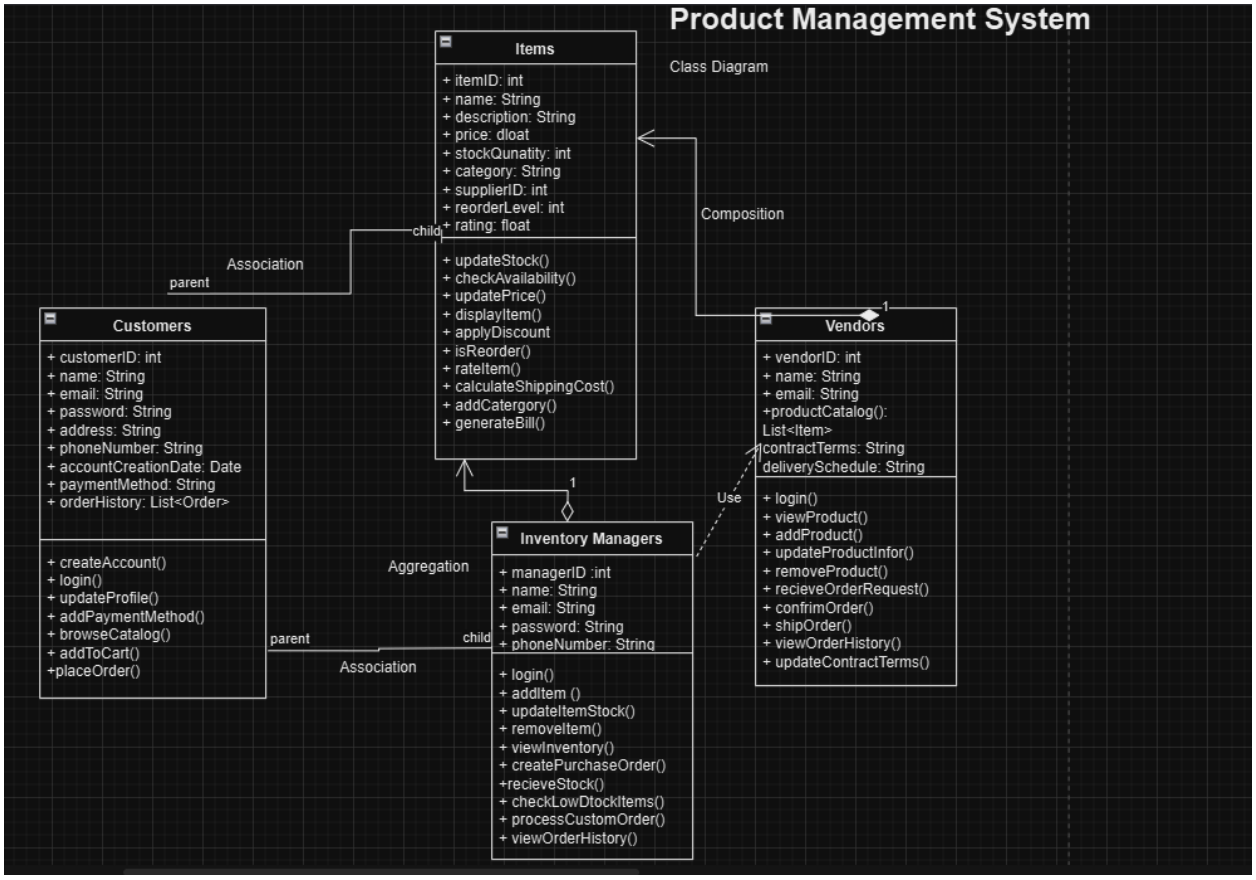


2.1.Subsystem Architecture:

UML Diagrams

Class Diagram:

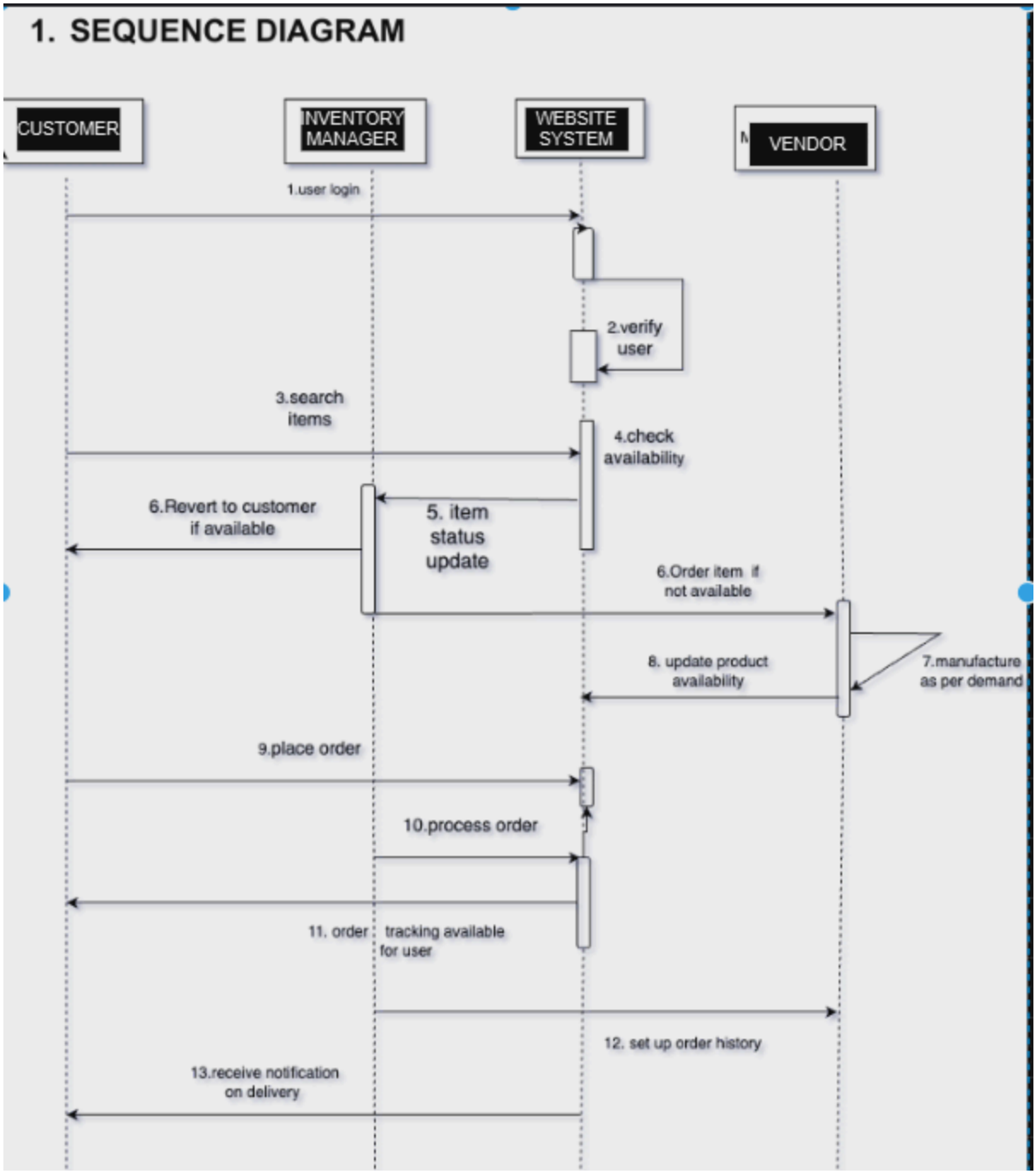
On the class diagram, one can see relationships and interactions that exist between the Key constituents of PMS as the central part of the Product Management System. The important classes are As we can see, ‘Warehouse Stock Management System’ and ‘Warehouse Manager’ all have values of their properties and which also have methods related to that particular type of object. Schmid et al., the paper presents an architectural overview of this diagram in high-level information dissemination of the system depending on the structure and data flow between the numerous entities for getting a global vision of the system design and functionality.



Sequence Diagram:

The Sequence diagram shows how work is done in warehouse stock management

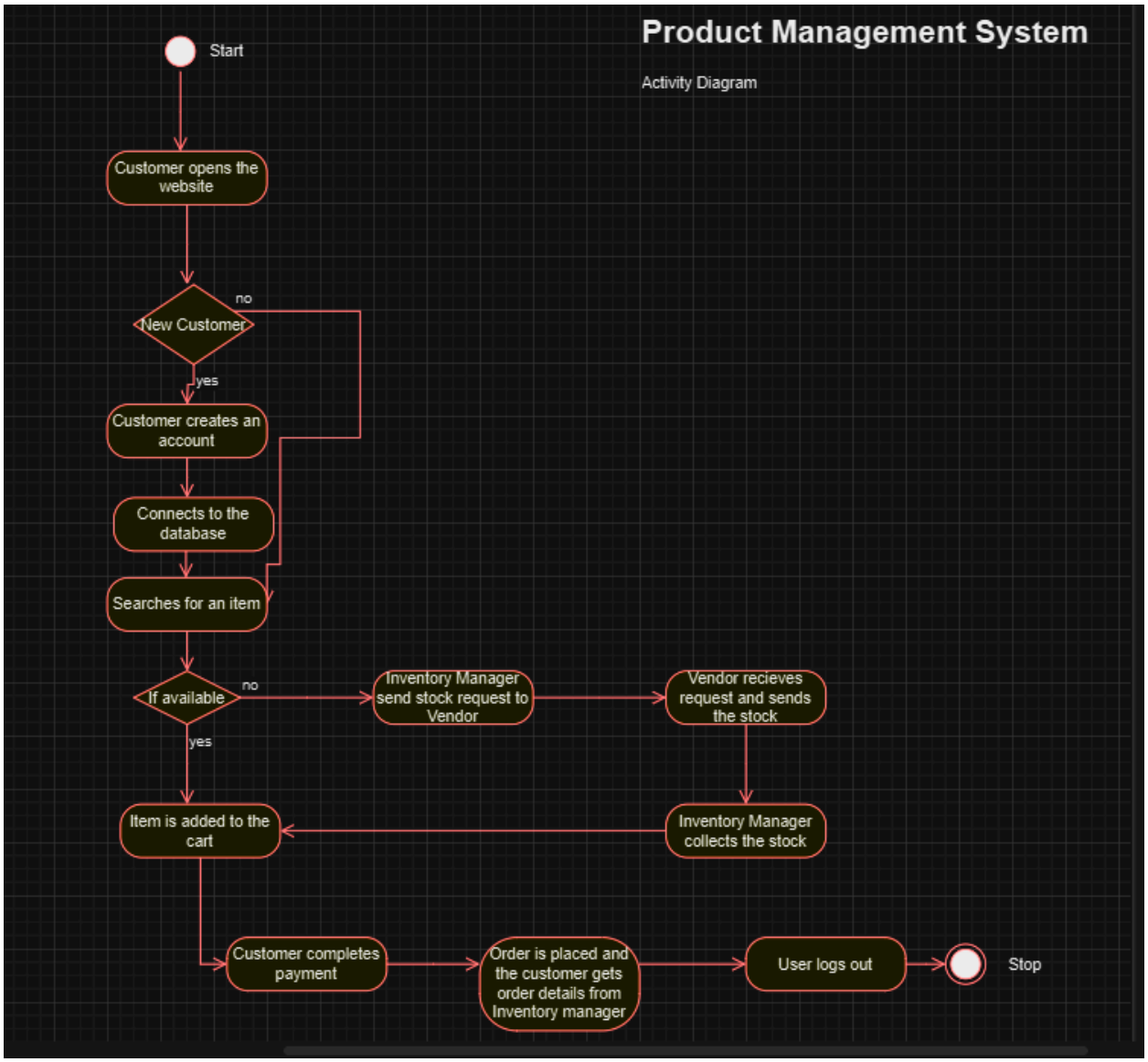
system. The roles start with the customer for registration and login while the next flow to different pages of the website goes to the next role which is the warehouse manager, the warehouse stock management system, and the vendor for browsing WG. According to the level of interactivity, the cascaded Web 2.0 technologies include information products, adding products to the cart, availability checks, and payment. The order that the customer places caps the process. The diagram indicates the subsequent sequence of activities that customers undertake when purchasing the product through the Internet warehouse stocks.



Activity Diagram:

It describes the process of working in warehouse stock management according to the activity diagram system. Starting from the user registration/Login to the User interface for browsing products and adding them to the cart and providing their availability, and the payment for the products. The procedure ends at the time when the order is placed. The figure displays all the

consecutive phases that are involved Within the computer-aided consumption of the product while purchasing the said product on the Internet.



Package diagram

- **Product System**:

  The bundle of services that covers warehouse management is known as an umbrella package feature.

- **Inventory Control:**

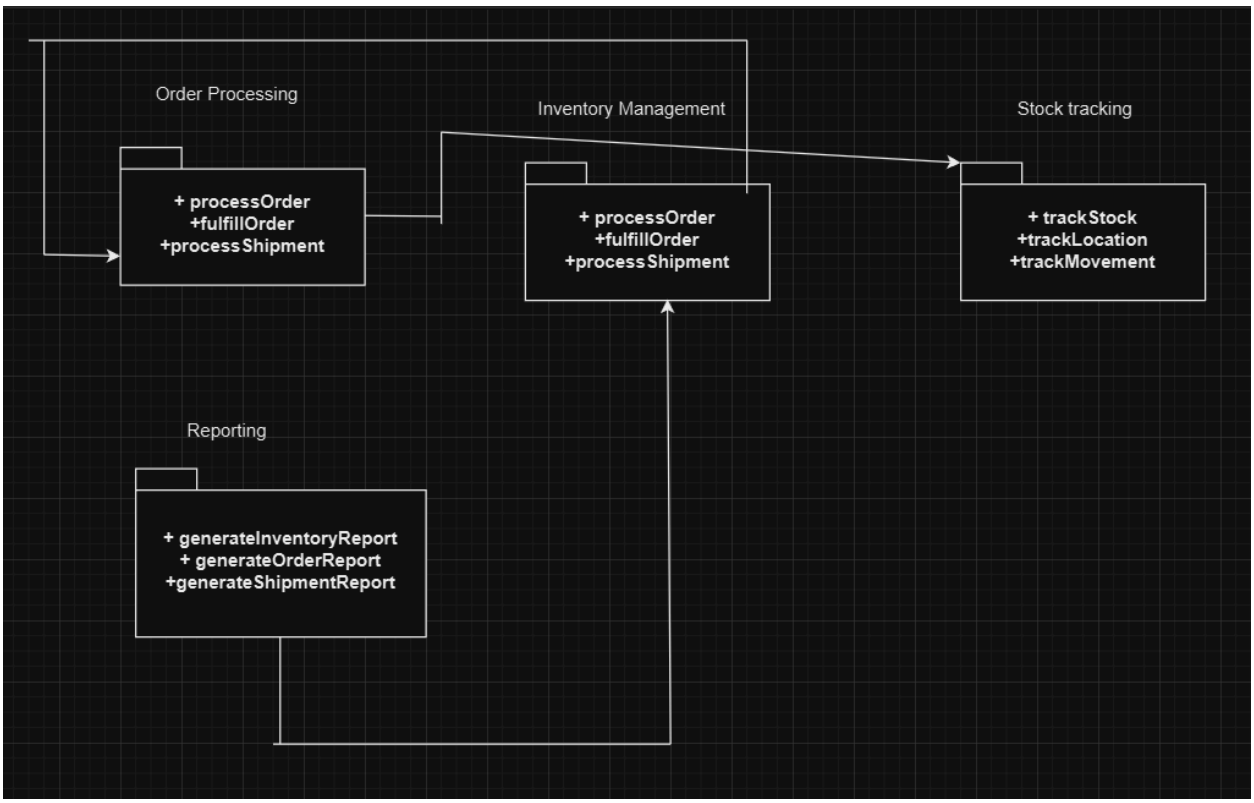
  This package deals with the stock in the warehouse. It manages stock levels including putting in new products, changing quantities, and even withdrawing items from the stock.

- **Order Fulfillment**:

  This package deals with orders which have been dispatched to the warehouse. It offers utility features for input of orders, order fulfillment, control/status of orders, and order management returns.

- **Reporting Suite:**

Serves for export and prints out the current status of the warehouse stock system. It assists customers in generating several sorts regarding stock balances, order delivery completion percentages, shipment statuses, and all other useful value metrics. All of these are packaged in such a way that related functionality is included in one package, enhancing modularity throughout the system. The term helps in maintaining the sustainability of the system. The relations and consequent interdependency between these programs are integrated into the general process of functioning of the warehouse inventory system.



2.2.Dependency Architecture

Database ER Diagram

A dependency Architecture for a warehouse stock management system has four layers:

Application-specific, application-generic, middleware, and system-software.

- **Domain-Specific Layer:**

  This layer includes software components, which are specific to it. Product management system. It also has such functions as inventory coordination of all activities, managing the orders, monitoring the delivery time, and preparing the reports.

- **Core Application Layer:**

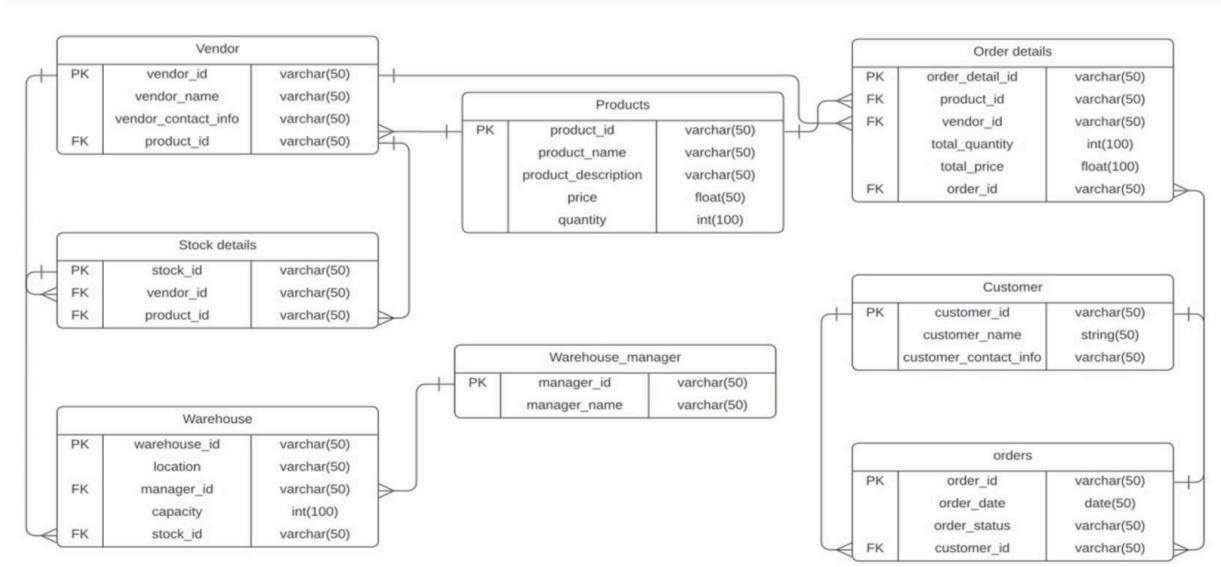
  This layer includes the components and libraries that are reusable in the framework and are not inherent to WMS solutions but are utilized to construct and maintain their functionality. It can include modules that help to exercise user authentication, databases, etc logging, and error handling.

- **Service Layer:**

  It links the application-specific and system software layers to the real systems interface layers. That is why some of the middleware components such as message brokers, API gateways, and integration frameworks are the link between the subsystems of the system to enable sound communication.

- **Operating System Layer:**

  This layer consists of software and systems upon which the warehouse stock management system is based. It has a built-in operating system, a database management system, a web server, and other system-level applications that are needed by the program to run.



2.3 Persistent Data Storage

This section identifies the key information that must be stored in the system and the chosen storage strategy. For the Product Management System (PMS), a relational database is used to store structured data related to products, inventory, orders, and user details.

Database Schema

- **Tables and Columns:**
    - **Products:** `ProductID`, `ProductName`, `Category`, `StockQuantity`, `Price`, `Location`
    - **Orders:** `OrderID`, `CustomerID`, `OrderDate`, `ShippingDate`, `OrderStatus`
    - **Users:** `UserID`, `Username`, `PasswordHash`, `Role` **(e.g., Manager, Vendor)**
    - **Stock Movements:** `MovementID`, `ProductID`, `ChangeAmount`, `Date`, `Reason`

Data will be stored in SQL format, leveraging database management systems like MySQL or PostgreSQL for consistency and reliability.

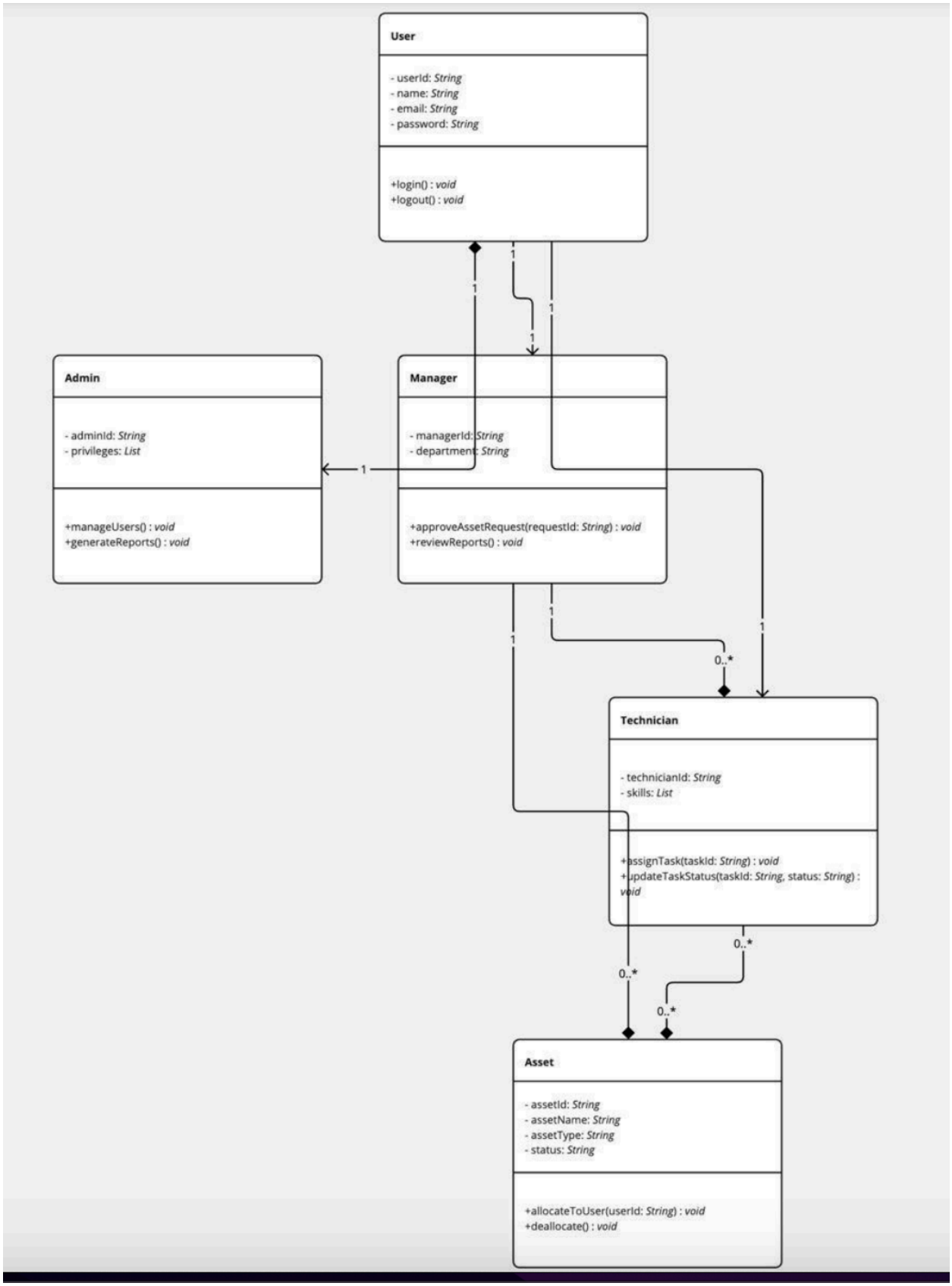
2.4 Global Control Flow

The PMS employs a **mixed control flow model**:

- **Event-driven**: The system waits for user interactions, such as product searches, order placements, and stock updates. This allows for asynchronous handling of user actions.
- **Time dependency** : A product management system may have timer-controlled actions in terms of Automated reminders for inventory restocking or product launches,Scheduled reports on product performance, Notifications for overdue tasks or milestones.
- **Concurrency**: Multi-threading is applied in the backend for handling simultaneous operations, especially for order processing and inventory management. The `Order Management Module` and `Stock Update Module` operate with distinct threads and use synchronization techniques like **mutex locks** to avoid data inconsistencies.

3 Detailed System Design

3.1 Static View

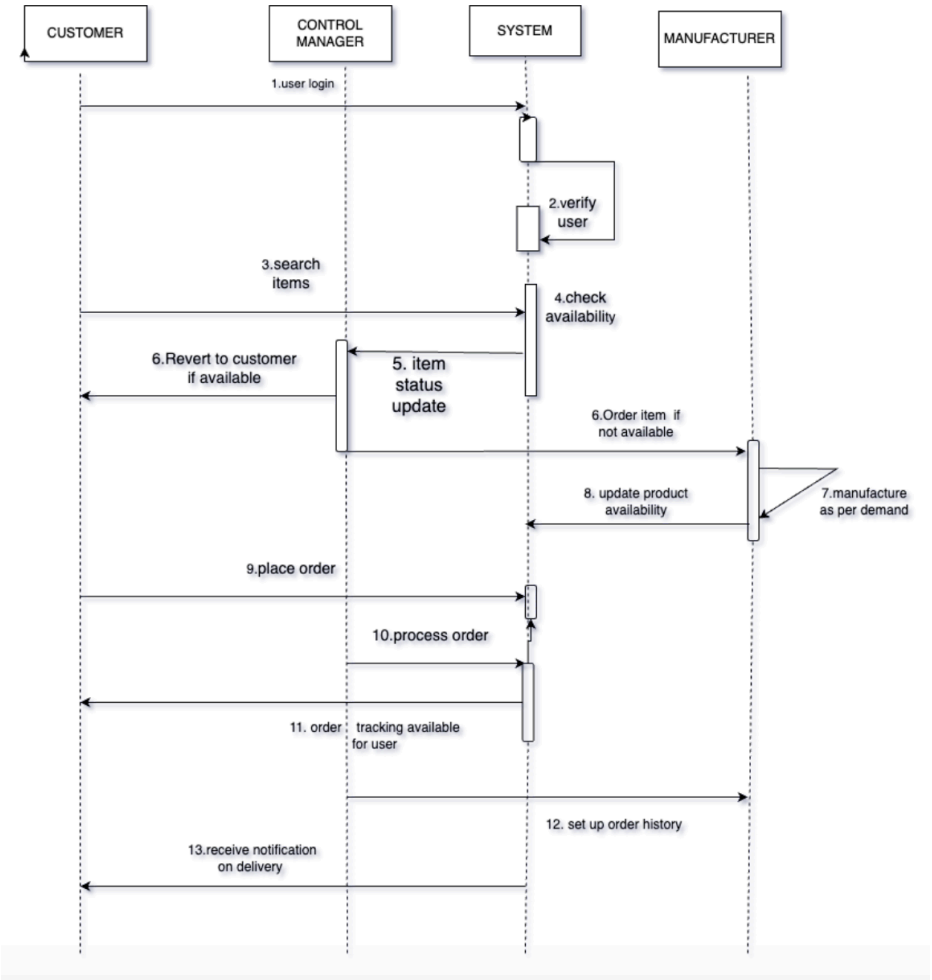

User
- userId: String
- name: String
- email: String
- password: String
+login() : void
+logout() : void
Admin
- adminId: String
- privileges: List
+manageUsers() : void
+generateReports() : void
Manager
- managerId: String
- department: String
+approveAssetRequest(requestId: String) : void
+reviewReports() : void
Technician
- technicianId: String
- skills: List
+assignTask(taskId: String) : void
+updateTaskStatus(taskId: String, status: String) : void
Asset
- assetId: String
- assetName: String
- assetType: String
- status: String
+allocateToUser(userId: String) : void
+deallocate() : void
1
0..*

The static structure of the system is illustrated through UML Class Diagrams. Key classes include:

- Product Manager: Manages product listings and inventory updates.
- OrderProcessor: Handles order placements, updates, and fulfillment.
- UserAuthenticator: Manages login and authentication processes.
- ReportGenerator: Compiles inventory and order status reports.

Each class includes attributes for data storage and methods for interaction with other components, maintaining a high level of cohesion and loose coupling for modularity.

3.2 Dynamic View



The UML Sequence Diagrams describe the flow of operations for essential functionalities, such as:

Placing an Order:

- The customer initiates an order.
- OrderProcessor verifies product availability.
- UserAuthenticator confirms user credentials.
- Inventory is updated via the Product Manager.

Stock Updates:

- Automatic alerts are triggered when stock levels fall below a threshold, as shown by the interaction between ProductManager and ReportGenerator.

Design Rationale:

The justification for the design choices made throughout a project's development is provided by the design rationale. Understanding the rationale behind specific decisions and how they support the overarching objectives is beneficial.

Key Evaluation Criteria:

Every component of the diagram should be explained in detail in the justification. It should explain the interrelationships between various parts and the rationale behind particular architectural decisions. Every element of the justification ought to be closely connected to the matching section of the diagram. This guarantees that the diagram and reasoning complement one another and offer a thorough understanding of the design. Every design choice should have a clear and logical justification in the rationale. These justifications ought to be founded on sensible ideas, including technical viability, user requirements, financial limitations, or performance criteria. Every effort should be made to support decisions with evidence. This could include citations to related initiatives, industry norms, factual information, or professional judgments.

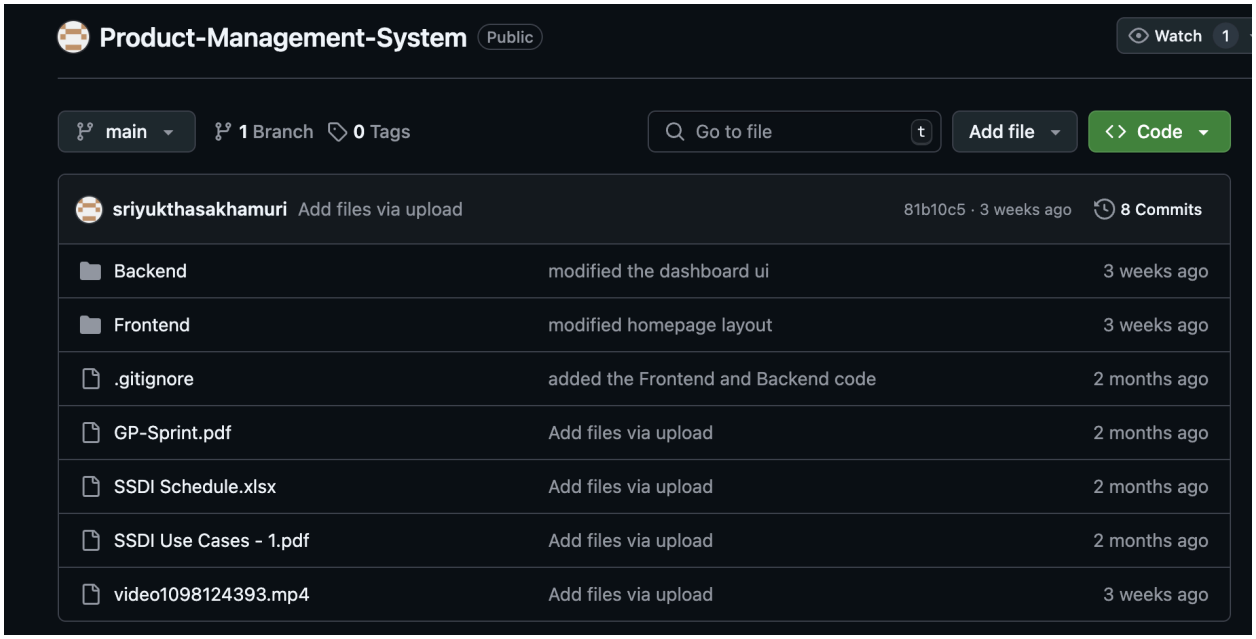
Consideration of Non-Functional Requirements:

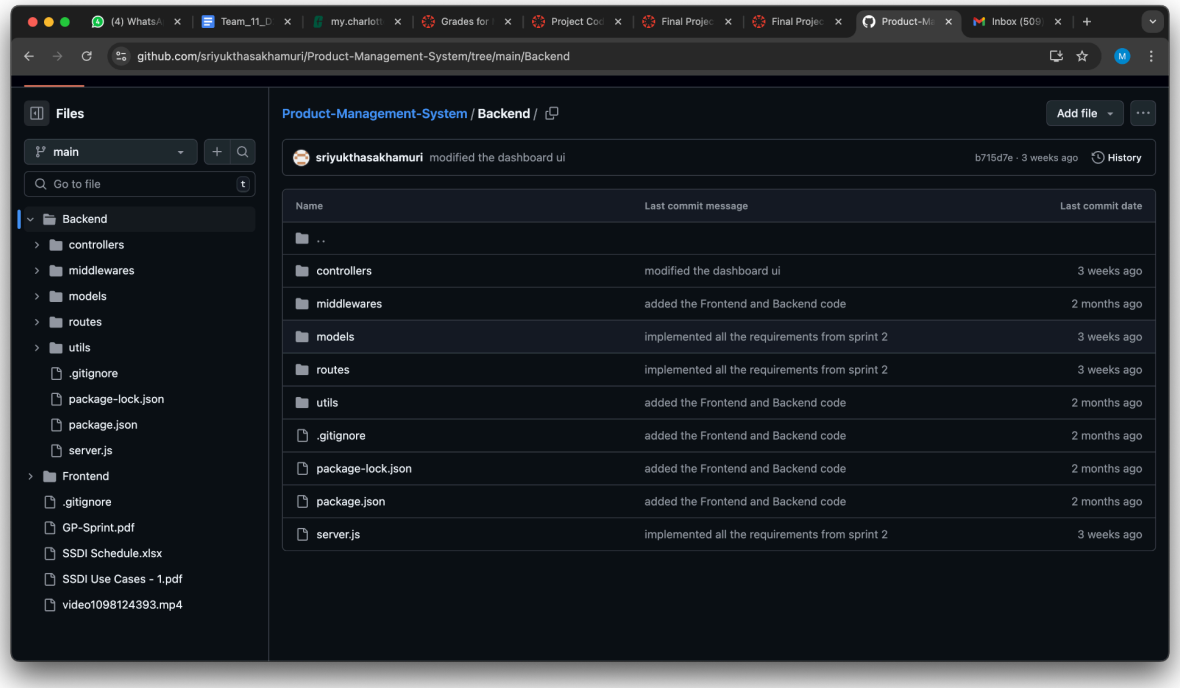
The justification ought to discuss how the design satisfies performance standards including scalability, efficiency, and speed. It should describe how the design guards against unwanted access and guarantees data privacy. User experience, accessibility, and ease of use should all be taken into account. The justification ought to clarify how the design facilitates future updates and simple maintenance.

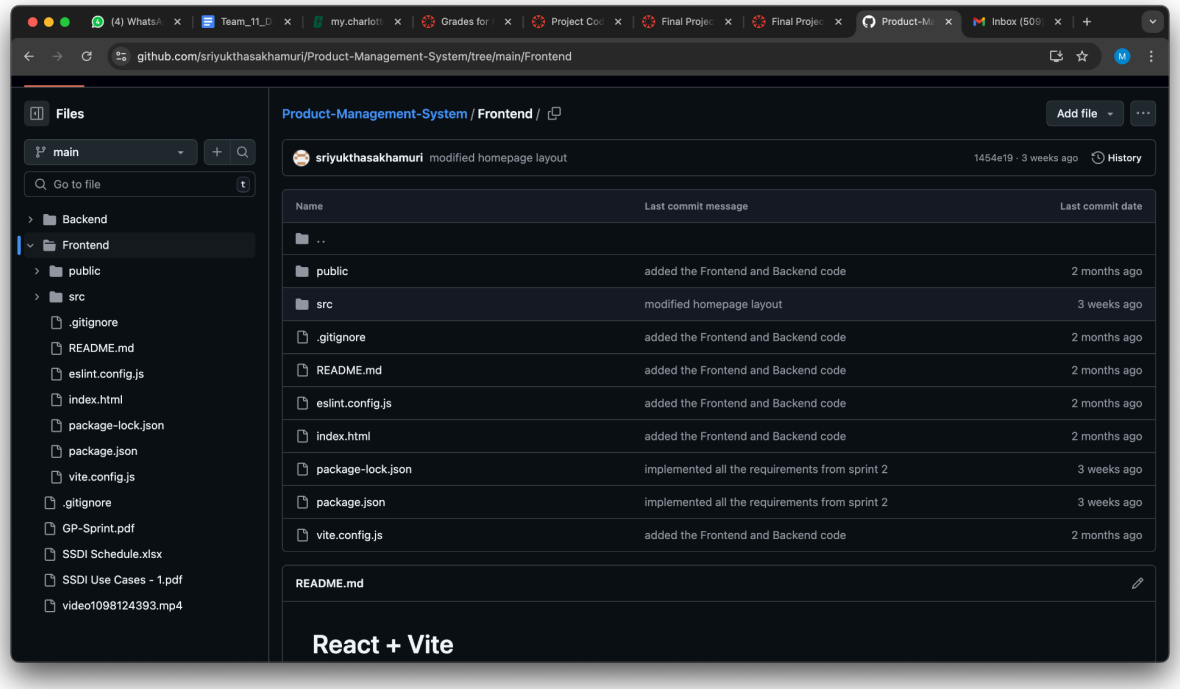
Consideration of Alternative Design Choices:

Any other design possibilities that were taken into consideration should be described in the justification. This indicates that before deciding on the final design, the designers considered a number of different strategies. The justification should outline the advantages and disadvantages of each option. This clarifies the rationale for the ultimate decision. Any trade-offs that were made, such as weighing complexity against maintainability or performance against cost, should be justified.

CODE / PROJECT STRUCTURE:

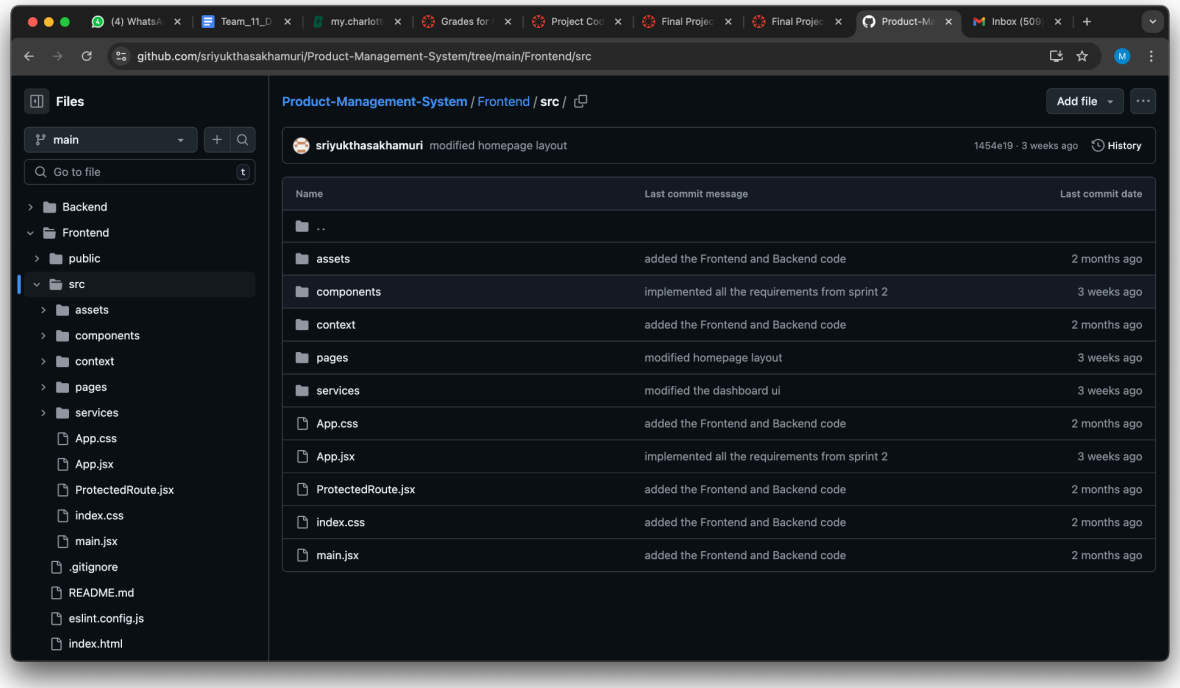


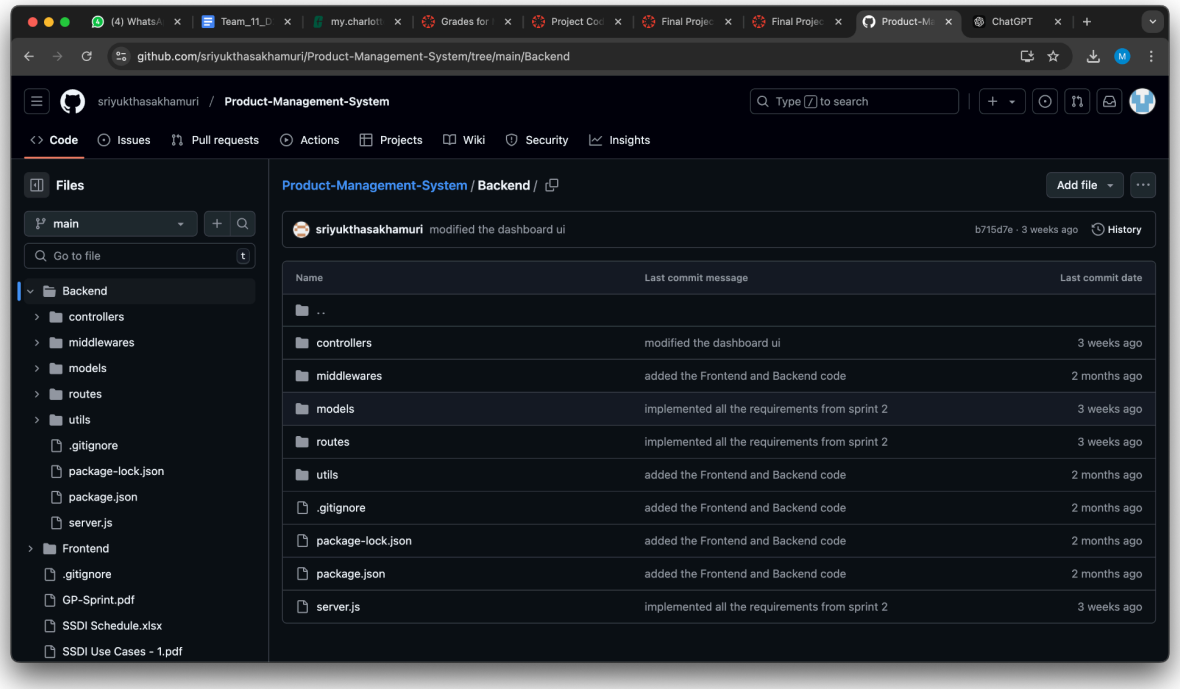

github.com/sriyukthasakhamuri/Product-Management-System/tree/main/Backend
Files
main
Go to file
Backend
controllers
middlewares
models
routes
utils
.gitignore
package-lock.json
package.json
server.js
Frontend
.gitignore
GP-Sprint.pdf
SSDI Schedule.xlsx
SSDI Use Cases - 1.pdf
video1098124393.mp4
Product-Management-System / Backend /
Add file
sriyukthasakhamuri modified the dashboard ui
b715d7e · 3 weeks ago
History
Name
Last commit message
Last commit date
..
controllers modified the dashboard ui 3 weeks ago
middlewares added the Frontend and Backend code 2 months ago
models implemented all the requirements from sprint 2 3 weeks ago
routes implemented all the requirements from sprint 2 3 weeks ago
utils added the Frontend and Backend code 2 months ago
.gitignore added the Frontend and Backend code 2 months ago
package-lock.json added the Frontend and Backend code 2 months ago
package.json added the Frontend and Backend code 2 months ago
server.js implemented all the requirements from sprint 2 3 weeks ago


github.com/sriyukthasakhamuri/Product-Management-System/tree/main/Frontend
Files
main
Go to file
Backend
Frontend
public
src
.gitignore
README.md
eslint.config.js
index.html
package-lock.json
package.json
vite.config.js
.gitignore
GP-Sprint.pdf
SSDI Schedule.xlsx
SSDI Use Cases - 1.pdf
video1098124393.mp4
Product-Management-System / Frontend /
Add file
sriyukthasakhamuri modified homepage layout
1454e19 · 3 weeks ago
History
Name
Last commit message
Last commit date
..
public added the Frontend and Backend code 2 months ago
src modified homepage layout 3 weeks ago
.gitignore added the Frontend and Backend code 2 months ago
README.md added the Frontend and Backend code 2 months ago
eslint.config.js added the Frontend and Backend code 2 months ago
index.html added the Frontend and Backend code 2 months ago
package-lock.json implemented all the requirements from sprint 2 3 weeks ago
package.json implemented all the requirements from sprint 2 3 weeks ago
vite.config.js added the Frontend and Backend code 2 months ago
README.md
React + Vite

index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```





[View our repository]

Features

- Product Catalog Management: Perform Create, Read, Update, and Delete (CRUD) operations on products, including attributes like name, description, price, quantity, category, and images.

- Inventory Control: Monitor stock levels, receive restocking alerts, and manage stock status reports.

- Order Processing: Handle customer orders efficiently, including order placement, tracking, and fulfillment.

- User Roles: Support for various stakeholders such as Managers, Workers, Process Executives, Distribution Companies, and End Users, each with specific functionalities.

- Data Analytics: Generate reports and analytics to aid in decision-making processes at strategic levels.

Installation

1. Clone the Repository:

```bash
git clone https://github.com/sriyukthasakhamuri/Product-Management-System.git
```

2. Navigate to the Project Directory:

```bash
cd Product-Management-System
```

3. Install Dependencies:

   - For the backend:

   ```bash
   cd server
   gradle clean build
   ```

   - For the frontend:

   ```bash
   cd client
   npm install
   ```

4. Set Up the Database:

   - Ensure PostgreSQL is installed and running.
   - Create a database named `postgres`.
   - Update the database configuration in `application.properties` with your database credentials.

Usage

1. Start the Backend Server:

   - Manually:

   ```bash
   cd server
   gradle bootRun
   ```

   - Using Docker:

   ```bash
   docker build -t pms-image .
   docker run --name pms-server --link postgres -p 8080:8080 -d pms-image
   ```

2. Start the Frontend Application:

   ```bash
   cd client
   npm start
   ```

3. Access the Application:

   - Open your web browser and navigate to `http://localhost:3000`.

Contributing

We welcome contributions to enhance the Product Management System. Please follow these steps:

1. Fork the repository.

2. Create a new branch (`git checkout -b feature/YourFeature`).

3. Commit your changes (`git commit -m 'Add YourFeature'`).

4. Push to the branch (`git push origin feature/YourFeature`).

5. Open a Pull Request.