Project name: Warehouse Management System
Members of Team: Sowjanya Achar, Aparajita Singh, Nivetha Kesavan

GitHub

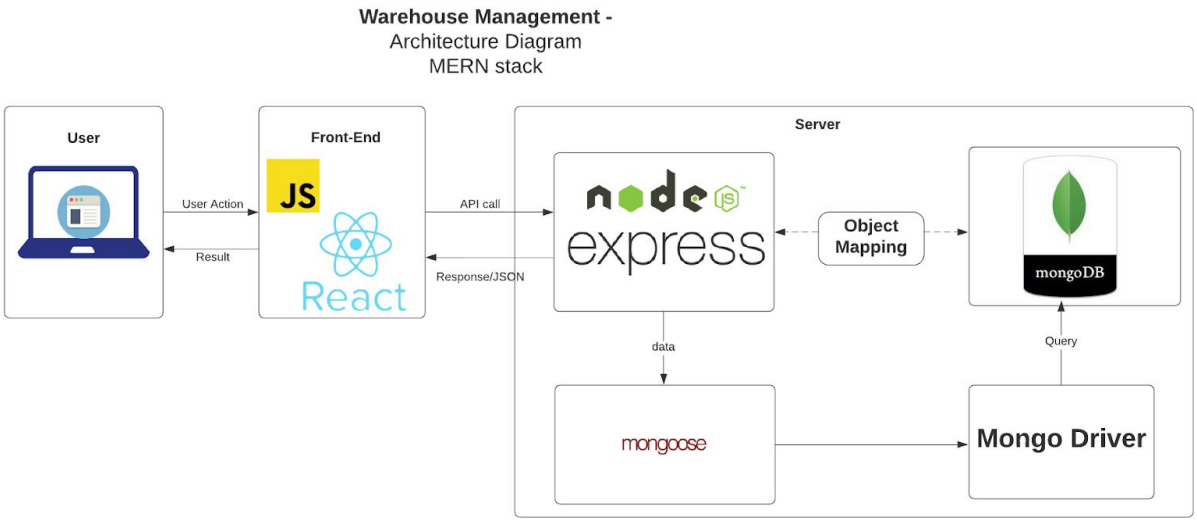
https://github.com/SowjanyaAchar/ooad-wms

Overview and Objective:

Warehouse management is extremely important in manufacturing today. Our project goal was to implement a warehouse management system that helps control and manage the day-to-day operations in a warehouse. WMS guides inventory receiving and put away optimizes picking and shipping of orders and advises on inventory replenishment.

Final State of System:

We used the MERN stack to implement our WMS system.



Features Implemented:

Our system is designed to have 3 roles: Admin, Seller, and Client. We have implemented the below functionality for the roles.

1) Client :

- Register and Login
- Search for a Product
- Get all the Products
- Add to cart
- View cart
- Delete cart
- Place order
- Cancel order
- Make Payment

2) Warehouse Administrator:
    - Add sellers
    - View All Clients
    - Add Admins
    - Delete any type of user
    - View Orders
    - Search for a Product
    - Check Total Sales

3) Seller:
    - Add Product
    - Delete Product

Testing the system:

We wrote a python script test.py to test our rest endpoints with different payload using the python requests library

Patterns Implemented:

MVC Pattern:
We have written Routes which act as Controllers. These controllers interact with Models (It consists of the Schema of our database, which is used for handling the database logic.) which calls DAO methods for MongoDB. The Rest APIs act as our view.

Data Access Object Pattern: DAO pattern is used to separate the data persistence logic in a separate layer. This allows our API to isolate how the low-level operations to access the

database is done. We have used Mongoose which is an Object Data Modeling (ODM) library for MongoDB and Node.**js**. This lets us create objects which are analogous to Table in the database, making our code (node.js) interact with these objects hence providing abstraction defined as DAO.

Strategy Pattern: To implement the login functionality we used Passport which is Express-compatible authentication middleware for Node.js.
It supports a wide range of providers (Facebook, Twitter, Google, etc.) each implemented as a Strategy. The Passport.js library is available as an npm package. The user of the library can decide which npm package to install for his particular use case.

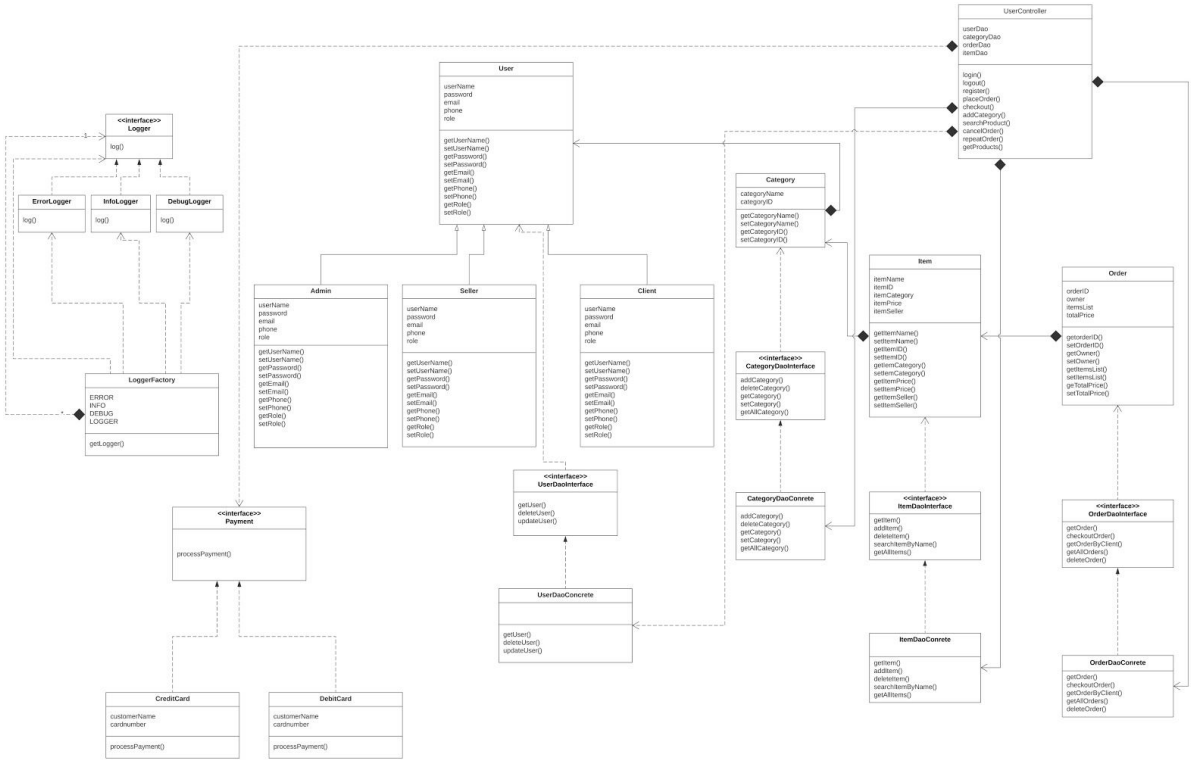
Features Not Implemented:

We have not implemented the following :

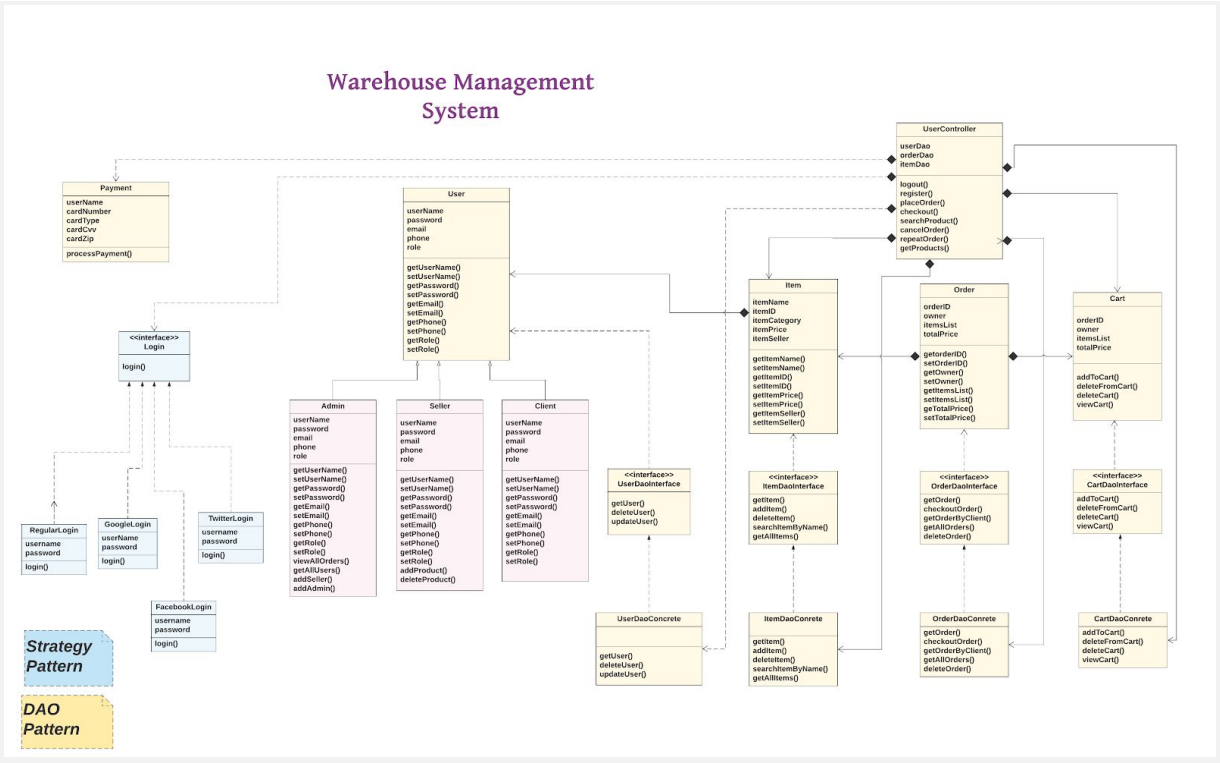
- Front End using React, due to time constraints.
- Factory Pattern - We performed debugging using the error statements of Node.js with API testing using Insomnia instead of using Logger Interface. Therefore, we did not use LoggerInterface in which we had intended to use the Factory Pattern.
- In our first draft, we had a use case for the seller to add/delete categories and products. That made less sense as we proceeded with the project. It made more sense to have the seller have the functionality to add and delete products.
- In the final version, we do not have ‘see previous order’ and ‘repeat previous order’ functionalities for the client.

Class diagram:

The class diagram included in Project 4:



Current Class Diagram:



Changes to the system:

We initially planned to use only one login option where the users need to enter their username and password they used upon registering. Later, we implemented another login as a strategy pattern with options to log in using Facebook, Twitter, and Google along with the initially planned one. This allowed us to implement the Strategy Pattern in there.

As mentioned in the previous section, we didn't use the Logger Interface. We instead used the error statements from Node.js with Insomnia API calls.

Earlier, we did not account for Cart in the DAO pattern. We have included it in the final version.

A clear statement of what code in the project is original vs. what code you used from other sources:

Citations for codes and tutorials:

1. Tutorial: https://www.youtube.com/watch?v=arkYCiCP6-g&t=6s
2. Tutorial for Login: https://www.youtube.com/watch?v=iX8UhDOmkPE&t=43s
3. The Add to cart logic was derived from https://github.com/ivan3123708/fullstack-shopping-cart
4. Mongoose command referred from https://mongoosejs.com/docs/
5. https://www.geeksforgeeks.org/mern-stack/
6. Login using Passport referred from http://www.passportjs.org/
7. https://github.com/mohamedsamara/mern-ecommerce/blob/master/server/routes/api/brand.js

Tools Used:

1. Webstorm: IDE for development https://www.jetbrains.com/webstorm/
2. Insomnia: API testing https://insomnia.rest/

Key design process elements and challenges:

1. Requirement analysis:
   This was the first step in our project. In requirement analysis, we defined software requirements and system requirements. We went on to list some constraints and assumptions we were making in the system.
   In our warehouse management system, we have three types of users - client, seller, and admin. After defining the users, we described the different use cases for each type of user.
2. System Design:
   In system design, we first thought of the data storage part of our application. We decided to use a MongoDB database for data storage. For the backend framework, we decided to use Express and for the Javascript Runtime Environment, we used Node.js. We decided to test our API routes using the Insomnia Rest client.
3. Plan final iteration:
   After project 5, we planned a few things for the next iteration. While we implemented most of the functionalities, we were unable to implement React to create the front end due to time constraints. This design process element did not go as planned.

4. OOAD with Javascript: Adopting Design patterns with node.js was one of the challenges we faced. Working on java and python in most of the assignments we were familiar with patterns, but it was challenging to show clear pattern use using MERN stack. We weren't familiar with mongoose as an ORM during phase Project 4, later when we did use it we were able to implement DAO pattern successfully.