كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

جامعة حلوان
HELWAN UNIVERSITY

Software Engineering-2 Cover Sheet

Project Title: ……

Row Number (in PDF): …… Time: ……

#	Member Name (printed in Arabic)	Member Id (printed)	Handwritten signature
1	محمد احمد محمد عبد المجيد	20210737	
2	محمد عادل محمد عبد الحليم	20210804	
3	محمد سمير سعد يوسف	20210781	
4	محمد رمضان جمعه سليمان	20210776	
5	محمد طارق عبد الفتاح محمد	20210800	
6			
7			

Project Requirements(grades)

SRS	2	
SDD	2	
Validation	3	
OCL	4	
AOP	4	
Microservices	1	
Cloud	4	

_____
20

Contents

Introduction

Our E-Commerce Platform offers a comprehensive solution for both administrators and customers, enabling seamless management and shopping experiences.

Administrators have full control over the product inventory and user management. They can easily update product details, delete products as necessary, manage user accounts by deleting them when required, and retrieve a list of all users for administrative purposes. Additionally, administrators have the capability to search for specific users by their unique identifiers.

Customers benefit from a user-friendly interface that allows them to interact with the platform effortlessly. They can create and manage their shopping carts, view the contents of their carts at any time, add products to their carts with ease, update their cart contents as needed, and remove items from their carts when desired. Furthermore, customers have access to the entire product catalog, enabling them to browse and search for products by name. Once satisfied with their selections, customers can proceed to create orders and view the status of their orders.

With these features, our E-Commerce Platform provides a robust and efficient solution for both administrators and customers, enhancing the overall shopping experience for all users involved.

Functionality

The Stock Management System provides functionalities to address various inventory management needs for businesses with Admin and Supervisor roles.

Admin Functionalities:

- **User Management:**
  - Get All Users
  - Search user By Id.
  - Delete Account.
- **Product Management:**
  - Add new products with descriptions and unique identifiers.
  - Edit existing product information (e.g., name, description).
  - Delete products that are no longer relevant.

    - Assign product quantities to specific warehouses, defining initial stock levels.
    - View product stock reports across all warehouses, providing insights into overall inventory levels.

- **Security:** The system should implement user authentication and authorization to restrict access based on user roles (Admin vs. Supervisor). Only authorized users should be able to perform specific actions.
- **Data Validation:** Ensure data entered by users is valid (e.g., positive stock quantities, unique product identifiers).
- **Logging and Auditing:** Maintain logs of user actions (e.g., product additions, stock updates) for traceability and potential analysis.

Non-Functionality

Non-functional requirements encompass the broader characteristics of the system, focusing on how it behaves rather than its specific features. Here's an outline of some key non-functional requirements:

- **Performance:** The system should respond to user actions and data requests in a timely manner, especially for critical tasks like stock updates and low-stock reporting.

- **Scalability:** The system should be designed to accommodate future growth, potentially with an increasing number of products, warehouses, and users.

- **Availability:** The system should be highly available, minimizing downtime and ensuring continuous inventory management. Consider backup and recovery mechanisms for data security.

- **Usability:** The user interface should be intuitive and easy to use for both Admins and Supervisors, regardless of their technical expertise.

- **Security:** The system should prioritize data security by protecting user information, product details, and stock levels from unauthorized access. Implement strong encryption mechanisms and access controls.

- **Maintainability:** The code should be well-structured, documented, and modular for easier maintenance and future updates.

System Architecture Overview

The e-commerce platform is designed as a scalable and resilient system that caters to the needs of both administrators and customers. The architecture follows a microservices-based approach, allowing for modularity, flexibility, and ease of maintenance.

- **Presentation Layer:** Responsible for presenting the user interface to both administrators and customers.
- Developed using modern web technologies like HTML, CSS, and JavaScript.
- Provides intuitive interfaces for browsing products, managing carts, and placing orders.
- **Application Layer:** Houses the core business logic of the e-commerce platform.
- Comprised of various microservices responsible for different functionalities:
- Product Service: Handles product-related operations such as CRUD operations, search, and retrieval.
- User Service: Manages user accounts, authentication, and authorization.
- Cart Service: Facilitates cart management, including adding, updating, and deleting items.
- Order Service: Handles order creation, management, and fulfillment.
- Each microservice is deployed independently, promoting scalability and fault isolation.
- **Database Layer:**
- Stores and manages the persistent data required by the application.
- Utilizes a combination of relational and NoSQL databases to cater to different data requirements:
- Product Database: Stores product information such as name, price, and availability.
- User Database: Manages user accounts, authentication details, and preferences.
- Order Database: Stores order details, including items purchased, quantities, and status.
- Employing database sharding and replication to ensure data availability and scalability.
- **Integration Layer:**
- Facilitates communication between different components of the system.
- Utilizes RESTful APIs for inter-service communication, enabling loose coupling and flexibility.
- Implements message queues and event-driven architecture for asynchronous processing and decoupling of services.

- **Security Layer:**
- Implements robust authentication and authorization mechanisms to safeguard user accounts and sensitive information.
- Utilizes HTTPS protocol for secure communication between clients and servers.
- Implements role-based access control (RBAC) to restrict access to certain functionalities based on user roles.
- Incorporates encryption techniques to protect data at rest and in transit.
- **Infrastructure Layer:** Provides the underlying infrastructure necessary to deploy and operate the e-commerce platform.
- Utilizes cloud services such as AWS, Azure, or GCP for scalable compute, storage, and networking resources.
- Implements containerization using Docker for packaging applications and dependencies.
- Orchestrates containerized applications using Kubernetes for efficient deployment, scaling, and management.
- Incorporates monitoring and logging solutions for real-time performance monitoring, troubleshooting, and analysis.

Technology Stack:

Backend:

- Framework: Spring Boot
- Programming Language: Java
- Database: MySQL or MariaDB
- RESTful API: Spring Web or Spring MVC for building API endpoints
- Database Access: Spring Data JPA for seamless database interaction
- Authentication and Authorization: Spring Security for managing user authentication and authorization

Frontend:

- Framework: React
- Programming Languages: JavaScript, HTML, CSS
- State Management: Redux or React Context API for managing application state
- UI Component Library: Material-UI or Bootstrap for pre-built UI components and styling

Additional Technologies:

- API Documentation: Swagger or Springfox for generating API documentation

- Development Tools: IntelliJ IDEA or Eclipse (IDE), npm (Node Package Manager) for frontend dependencies, Git for version control
- Deployment: Docker for containerization, Jenkins or GitLab CI/CD for continuous integration and deployment

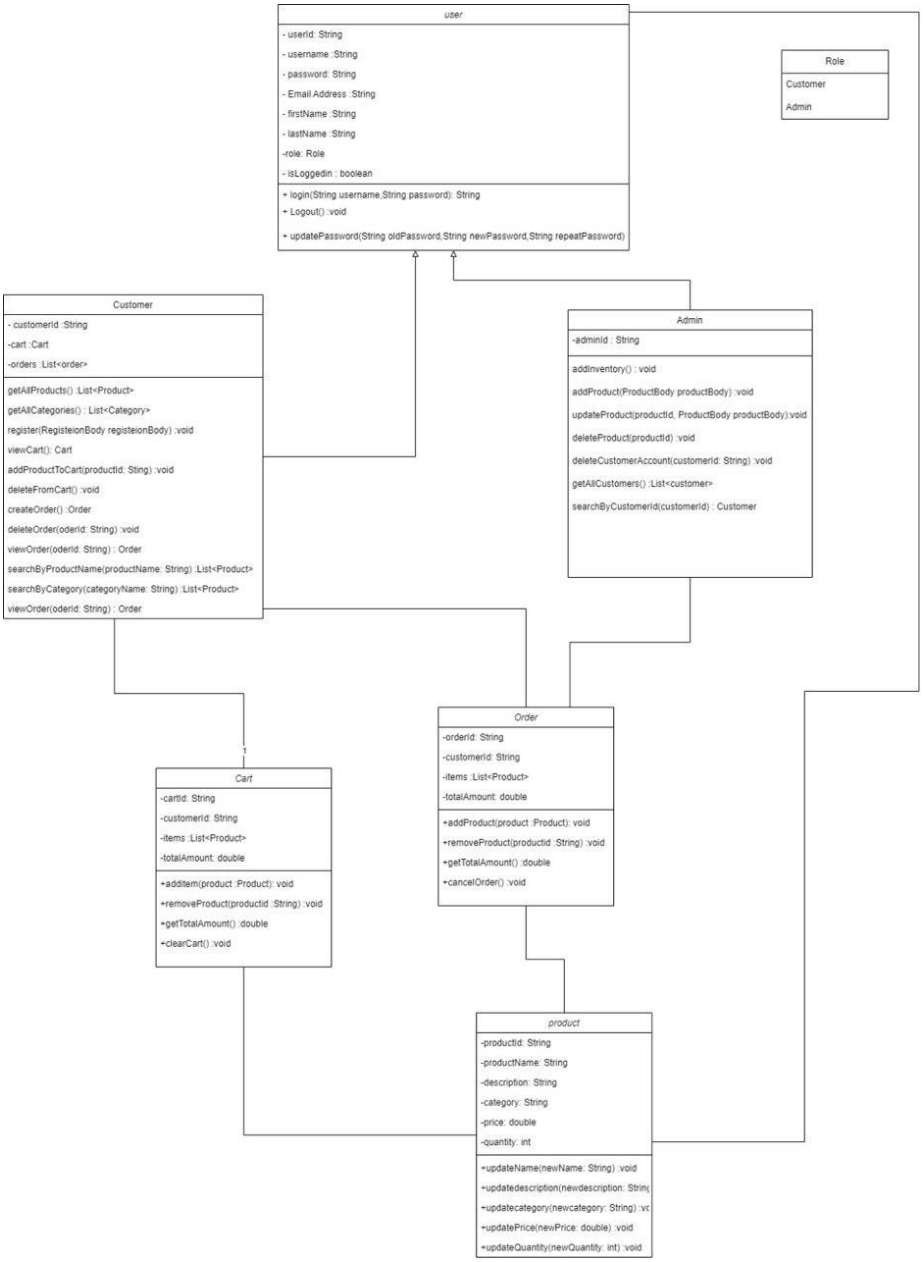
API Structure:

- Authentication Endpoints: /api/auth/login (POST), /api/auth/logout (POST)
- User Endpoints: /api/users (GET, POST, PUT, DELETE)
- Product Endpoints: /api/products (GET, POST, PUT, DELETE)
- Warehouse Endpoints: /api/warehouses (GET, POST, PUT, DELETE)
- Stock Level Endpoints: /api/stock-levels (GET, PUT)

Diagrams

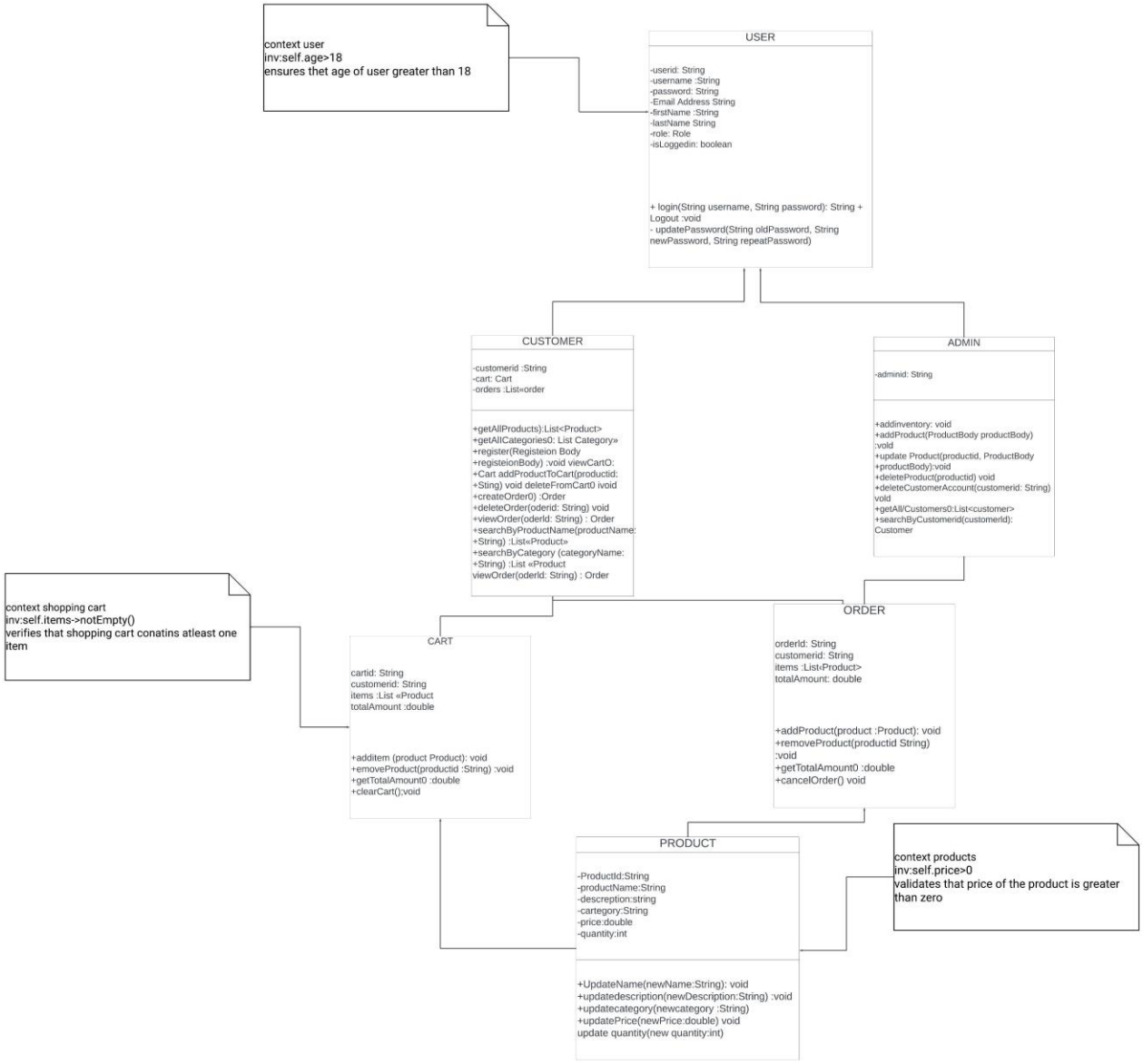
Class Diagram:

Class diagrams are like blueprints for object-oriented software. They show the building blocks (classes) and how they connect (relationships) to create the overall system. They help developers understand and design the software effectively.



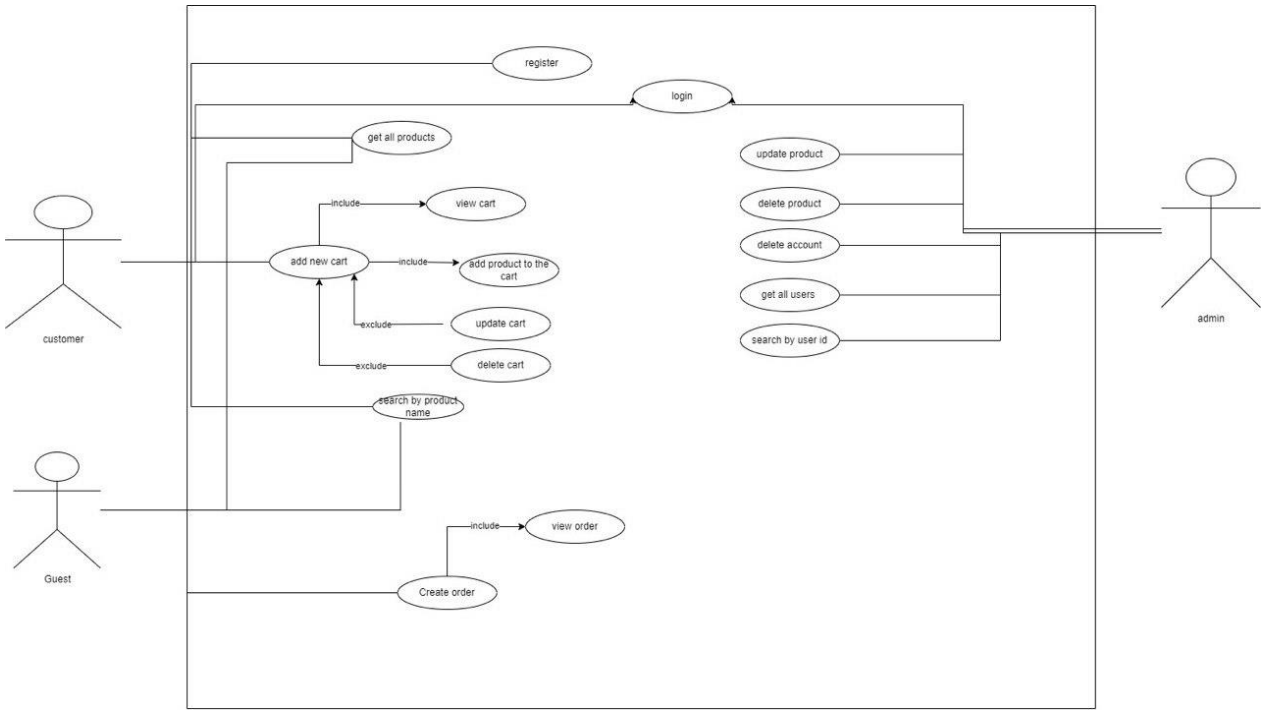
OCL Diagram:

a language for adding rules to UML models. It lets you define restrictions on data (like product stock must be non-negative) to ensure a well-designed system.



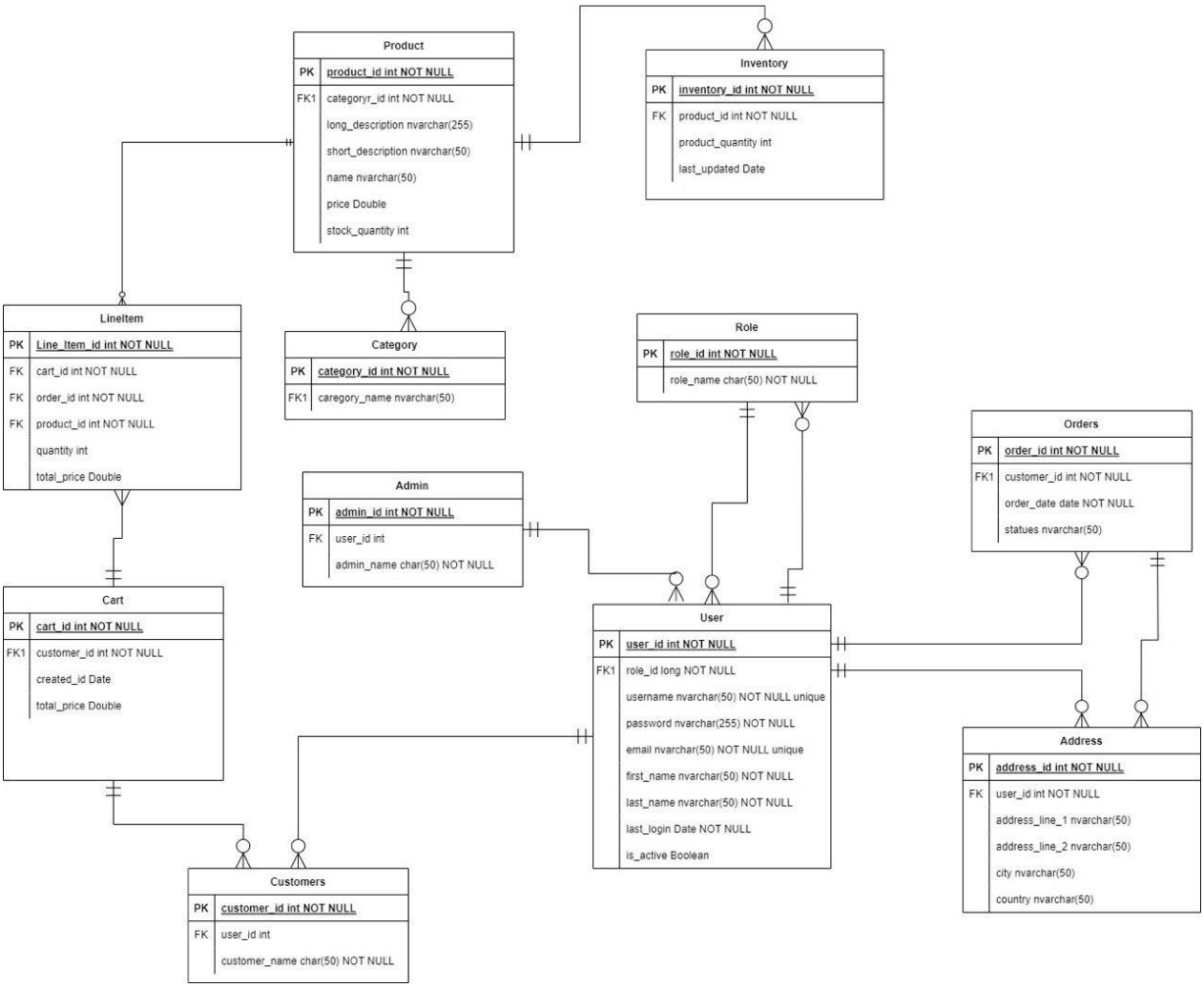
Use case Diagram:

Use cases are short stories explaining how users interact with a system to achieve goals. They help ensure the system meets user needs.



ERD Diagram:

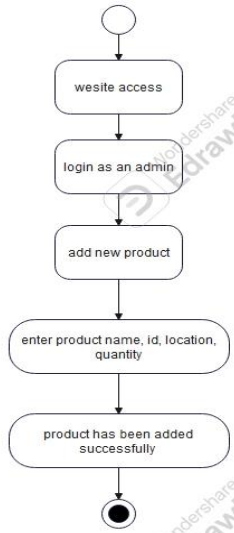
Blueprints for data! They show the main things (entities) in a system and how they connect (relationships) to store information effectively.



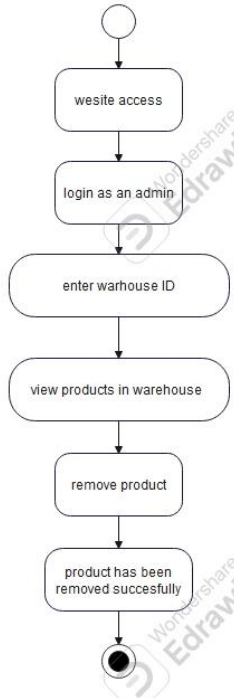
Activity Diagrams:

Flowcharts on steroids! They visually depict the flow of actions within a system, including decisions, steps, and potential alternative paths. They help understand the overall process and how different parts interact.

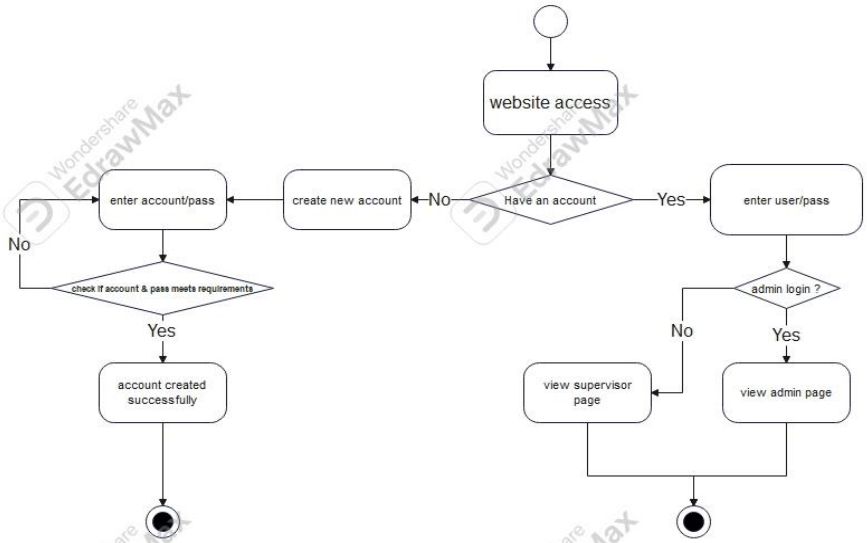
1-Add product diagram:



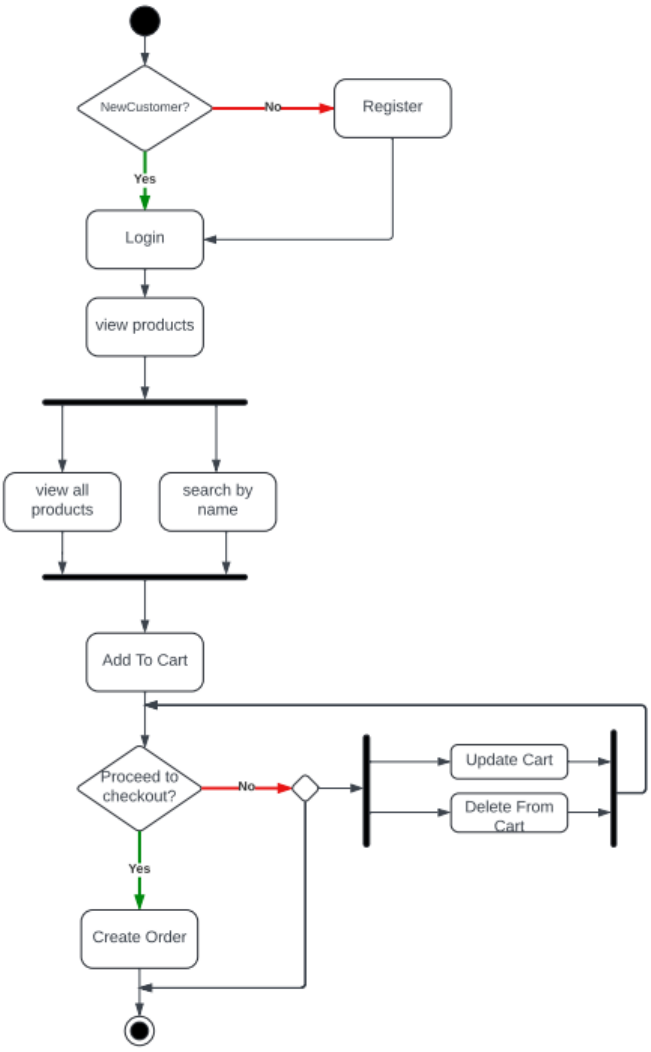
4-Delete product diagram:



5-Login & Signup diagram:



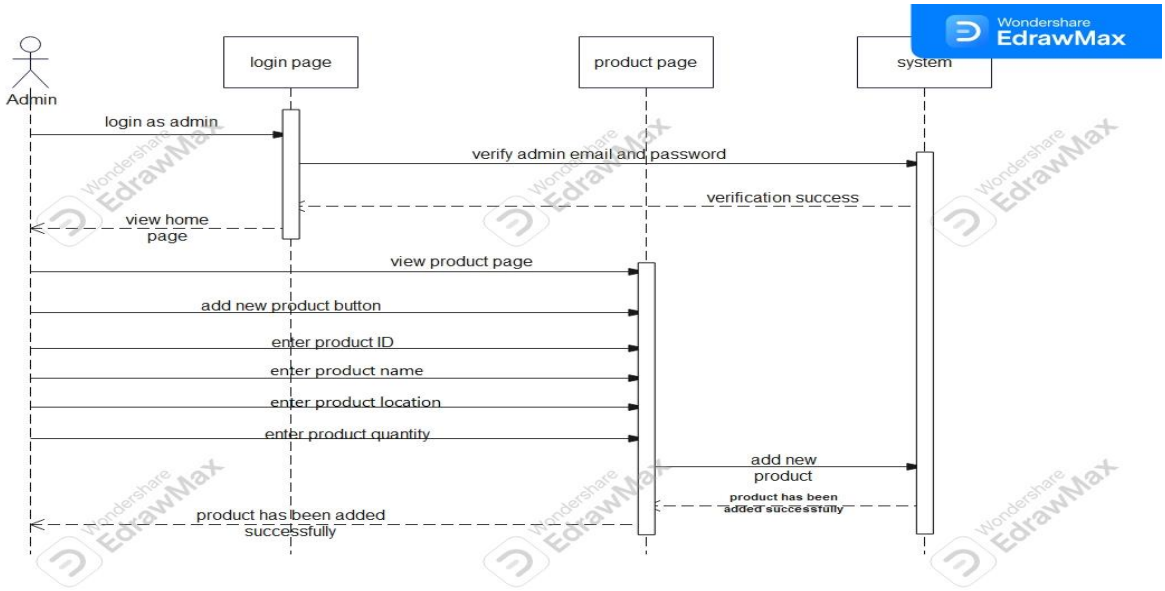
6-User Activity diagram:


NewCustomer?
No
Register
Yes
Login
view products
view all products
search by name
Add To Cart
Proceed to checkout?
No
Update Cart
Delete From Cart
Yes
Create Order

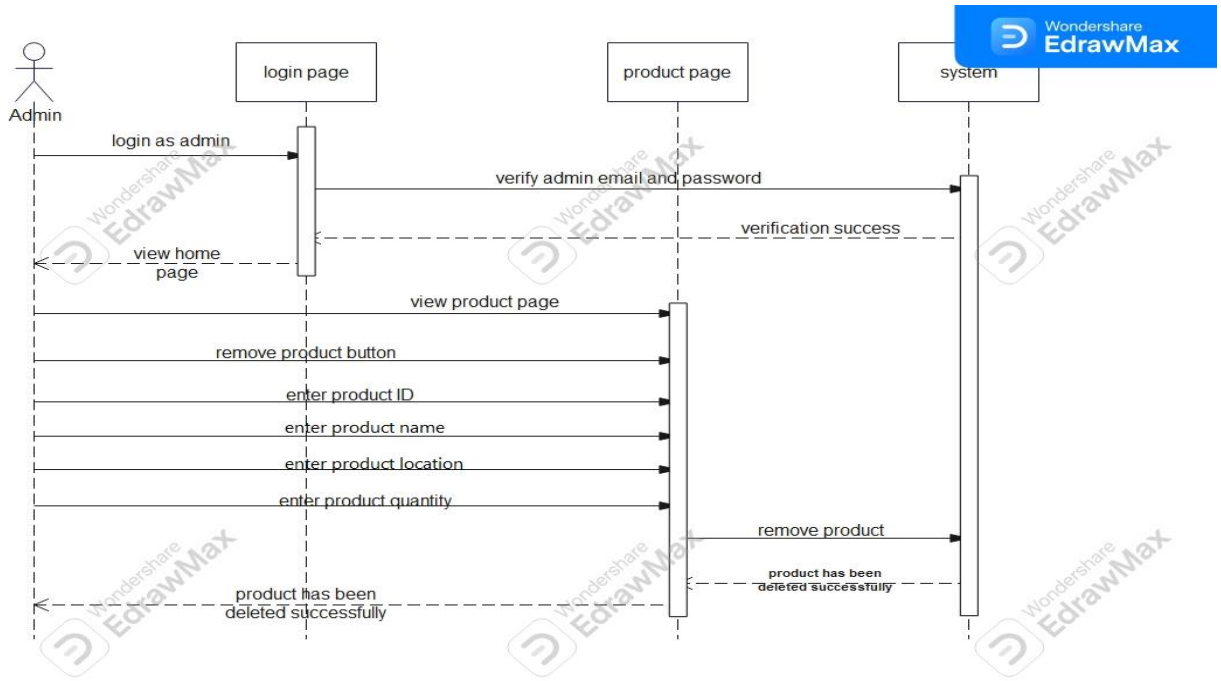
Sequence Diagrams:

Conversations in action! They show how objects in a system communicate with each other in a specific sequence, like a conversation, to achieve a task. They visualize the messages exchanged between objects and the order in which they occur.

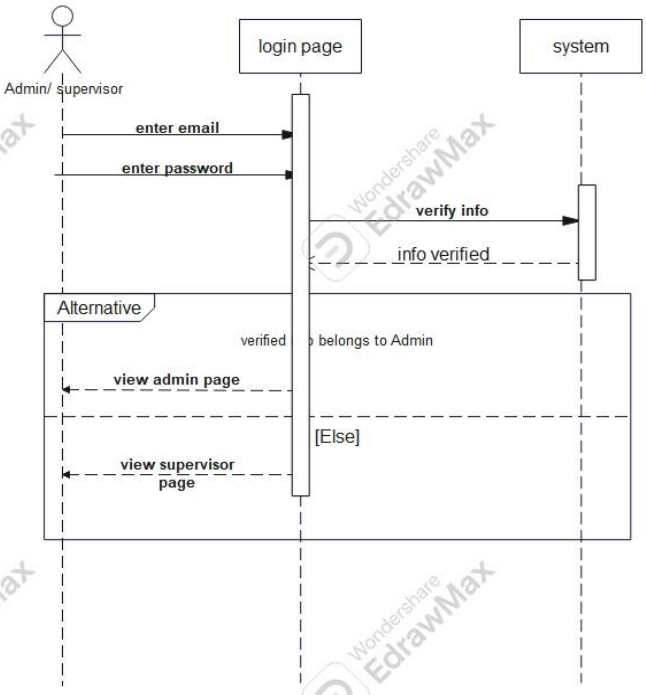
1- Add product diagram:



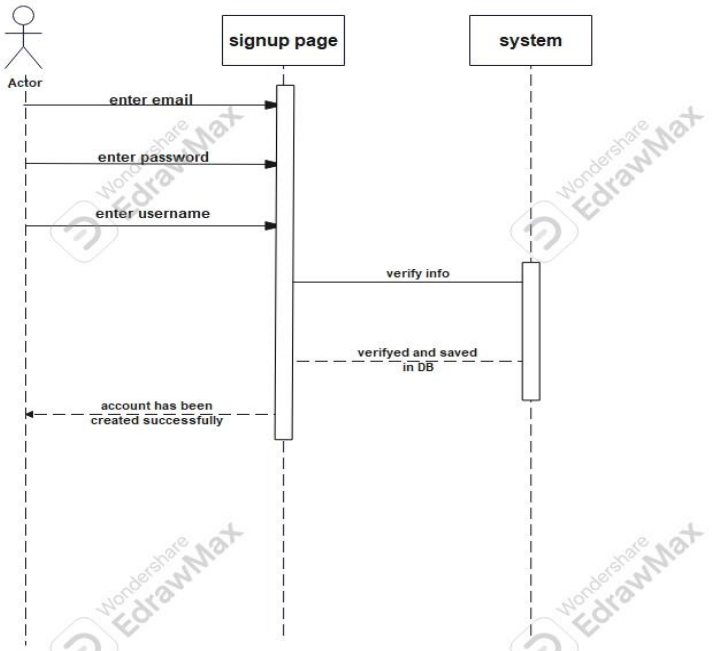
2- Delete product diagram:



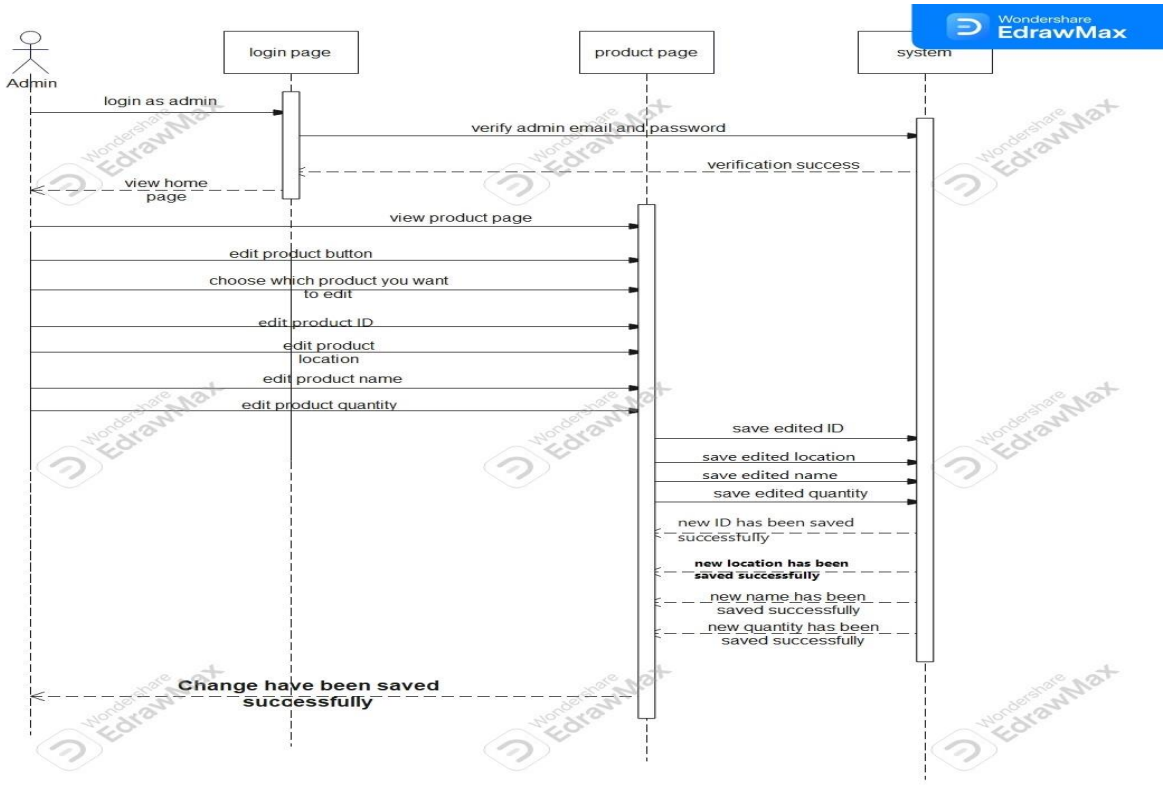
3- Login diagram:



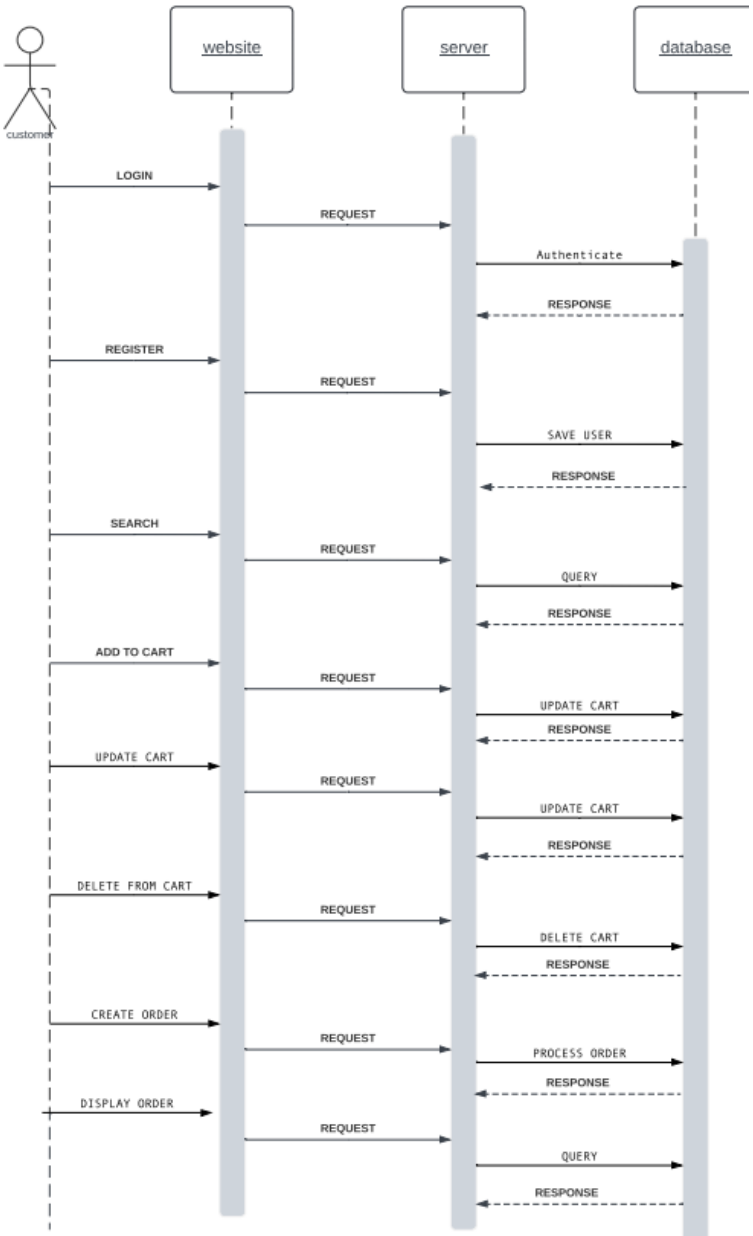
4- Signup diagram:



5- Edit product diagram:



6- User Activity diagram:


website
server
database
customer
LOGIN
REQUEST
Authenticate
RESPONSE
REGISTER
REQUEST
SAVE USER
RESPONSE
SEARCH
REQUEST
QUERY
RESPONSE
ADD TO CART
REQUEST
UPDATE CART
RESPONSE
UPDATE CART
REQUEST
UPDATE CART
RESPONSE
DELETE FROM CART
REQUEST
DELETE CART
RESPONSE
CREATE ORDER
REQUEST
PROCESS ORDER
RESPONSE
DISPLAY ORDER
REQUEST
QUERY
RESPONSE