SOFTWARE REQUIREMENTS SPECIFICATIONS

On

RETAIL- APPLICATION MANAGEMENT SYSTEM

(Urban Cartel)

Submitted by:

Aastha Bajaj

Divyansh Nama

Manshi Mahato

Prathik Sajjan

TABLE OF CONTENTS

1. INTRODUCTION **1**

1.1 Problem Statement 1

1.1.1 Purpose 1

1.1.2 Scope 1

1.1.3 Definitions and Acronyms 1

1.1.4 References 2

2. FUNCTIONAL REQUIREMENTS **3**

3. NON-FUNCTIONAL REQUIREMENTS **5**

4. USE CASE ANALYSIS **7**

5. SEQUENCE DIAGRAM **8**

6. ENTITY ATTRIBUTES **9**

1. INTRODUCTION

1.1 Problem Statement

1.1.1 Purpose

The primary objective is to provide an e-commerce platform for users to browse products, place orders, make payments, and track shipments.

1.1.2 Scope

It covers various modules like order management, payment processing, user authentication, and shipment tracking, ensuring that both customers and admins have a smooth experience.

1.1.3. Definitions, Acronyms, and Abbreviations

- **API(Application Programming Interface)** :API allows different software applications to communicate with each other. In your retail app, the API connects the frontend with the backend microservices, enabling features like placing orders, processing payments, and retrieving shipment details.
- **JWT (JSON Web Token)**:JWT is a compact, URL-safe token used for securely transmitting information between the frontend and backend. In your project, JWT is used for user authentication, ensuring that only authorized users can access the API and perform actions such as viewing order history or updating account details.
- **Microservices**: Microservices refer to an architectural style in which the application is divided into small, independently deployable services. Each service handles a specific business function (e.g., order management, payment processing) and can be updated or scaled independently.
- **REST (Representational State Transfer)**:REST is a set of architectural principles for designing networked applications. In your retail app, REST is used for communication

between the frontend and the backend microservices via HTTP requests (e.g., GET, POST, PUT, DELETE).

- **CI/CD**:
  CI/CD automates the process of integrating code changes and deploying them to production. In your retail app, CI/CD ensures that new features or bug fixes are automatically tested and deployed, leading to faster and more reliable releases.
- **RDBMS**:
  An RDBMS, like AWS RDS, manages data in tables with relationships between them. In your project, entities such as Order, OrderItem, and PaymentMethod are stored in a relational database.

1.4**References**

The retail application uses API gateway for interconnecting various microservices with our client and is also making use of JWT Tokens for Authorization of user using various hashing and Encryption algorithms.

2. FUNCTIONAL REQUIREMENTS

2.1 User Registration and Login

- **Registration**: Users must be able to create an account by providing personal details such as name, email, and password.
- **Login**: Users can log in using their email and password, and their identity is verified using JWT-based authentication

2.2 Product Catalog

- **Browse Products**: Users should be able to browse products in search bar.
- **View Product Details**: Users can view detailed information about a product, including images, descriptions, specifications, and reviews.

2.3 Shopping Cart Management

- **Add to Cart**: Users can add products to their shopping cart.
- **View Cart**: Users can view all items added to their shopping cart, including total price and item details.
- **Update Cart**: Users should be able to update the quantity of items or remove products from the cart.

2.4 Order Management

- **Place an Order**: Users can place an order by selecting products from their cart and providing delivery details.
- **Order Confirmation**: Once an order is placed, the system sends a confirmation with order details.
- **Order History**: Users can view their past orders.

2.5 Payment Processing

- **Payment Gateway Integration**: The system integrates with third-party payment gateways to securely process payments using credit/debit cards, digital wallets, or other payment methods.

2.6 User Profile Management

- **View Profile**: Users can view their personal information, such as name, email, address, and payment details.
- **Manage Addresses**: Users can add, edit, or remove delivery addresses in their profile.

2.7 Admin Management

- **Inventory Management**: Admins can manage products, including adding new products, updating product details, and managing stock levels.
- **Order Management**: Admins can view and update order statuses (e.g., pending, shipped, canceled).
- **User Management**: Admins can manage user accounts, including creating, updating users.

2.8 Notifications

- **Notifications**: Users will receive the notifications after order is confirmed and after signing up for the first time .

2.9 Security and Access Control

- **Role-Based Access Control (RBAC)**: Different roles (e.g., customer, admin) have specific permissions to perform actions such as managing orders or editing products.

3. NON-FUNCTIONAL REQUIREMENTS

3.1 Performance

- **Response Time**: The system must respond to user actions (e.g., placing an order, browsing products) within 2 seconds.
- **Data Processing**: The system should process user requests (e.g., payment, order placement) within a maximum of 5 seconds, even during peak traffic.

3.2 Security

- **Authentication**: The system must implement JWT-based authentication for all API requests to ensure that only authorized users can access the system.
- **Role-Based Access Control (RBAC)**: Admin users and regular users must have access to different sets of functionalities to protect sensitive operations (e.g., admin-only actions like product management).

3.3 Usability

- **User-Friendly Interface**: The UI must be intuitive, with minimal learning required for new users to navigate through the application.

3.4 Maintainability

- **Modular Design**: The system should be built using a modular architecture (e.g., microservices) to ensure that updates or bug fixes can be applied to individual components without affecting the entire system.
- **Code Documentation**: All code must be well-documented to facilitate easy updates and maintenance by other developers.

4. USE CASE ANALYSIS

Name	Login
Actors	Customer/Admin
Preconditions	Existence of database and correctly installed web-based server.
Description	When the application is accessed via a web browser, the user is presented with a login page. Customers and admins can log in using their credentials, which are validated by the system.

Name	Manage Users
Actors	Admin
Preconditions	Existence of database and correctly installed web-based server.
Description	Admins can manage user accounts, including creating new accounts, updating user details, deactivating accounts, and setting permissions. This ensures proper access control and user management.

Name	Manage Products
Actors	Admin
Preconditions	Existence of database and correctly installed web-based server.
Description	Admins can manage the product catalog by adding new products, updating existing product information, removing outdated products, and managing product categories.

Name	Process Orders
Actors	Admin
Preconditions	Existence of database and correctly installed web-based server.
Description	Admins handle order processing, including updating order statuses (e.g.,

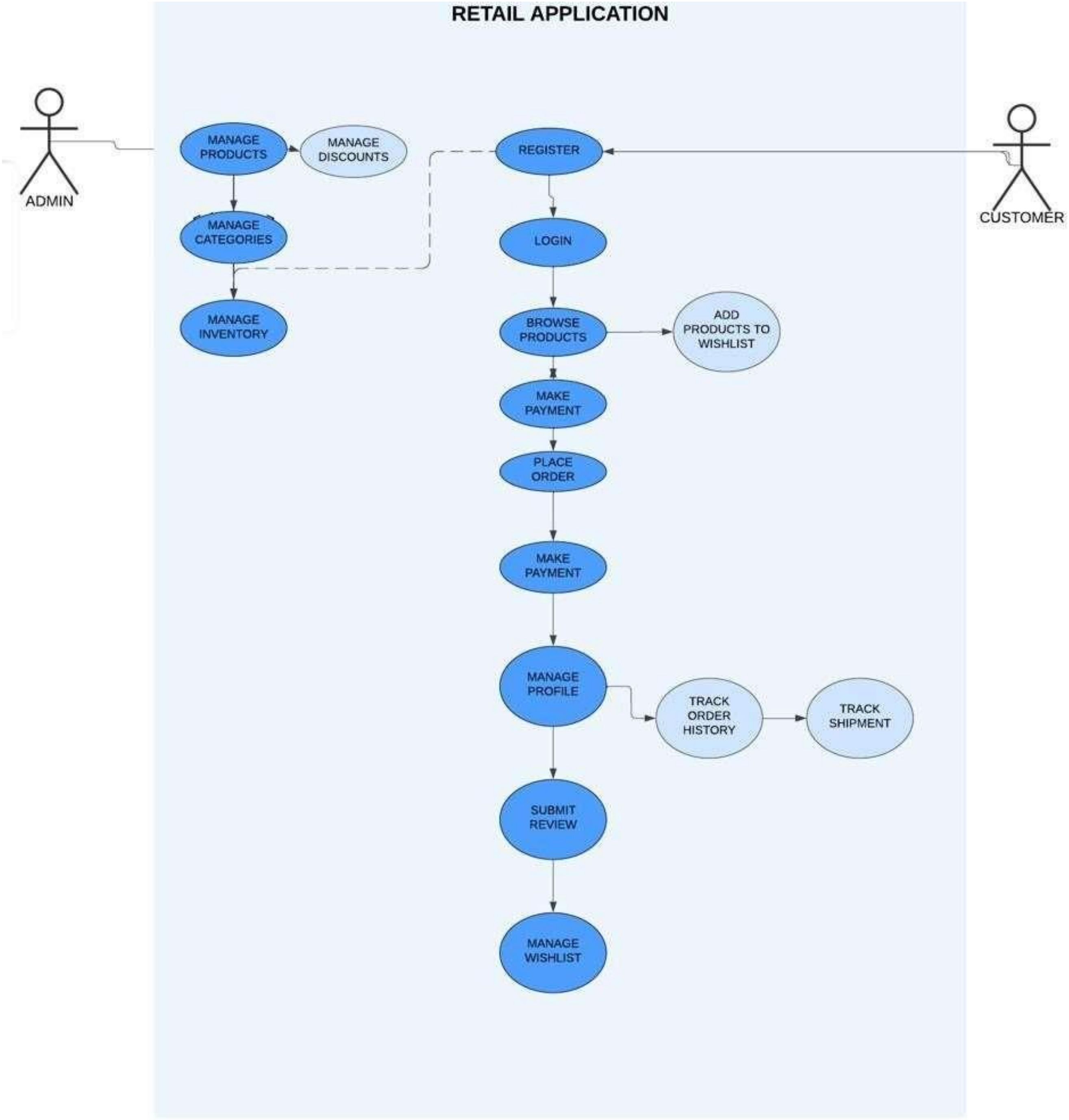
	from pending to shipped), managing returns, and handling customer queries related to orders.

Name	Search Products
Actors	Customer
Preconditions	Existence of database and correctly installed web-based server.
Description	Customers can search for products using keywords, filters (such as price range, brand), and sorting options (e.g., by popularity, price). This helps customers find the products they are interested in quickly.

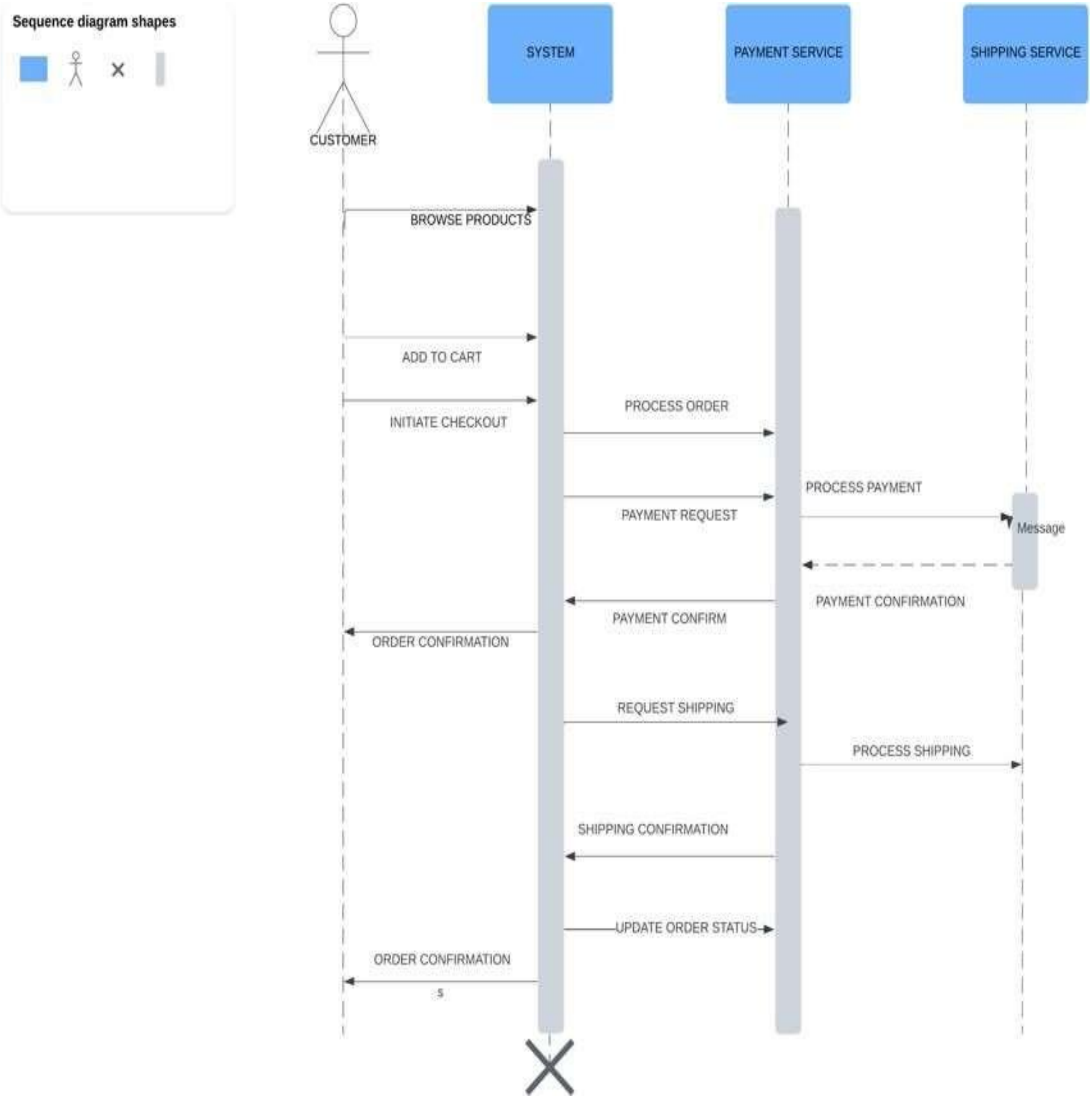
Name	Add Reviews
Actors	Customer
Preconditions	Existence of database and correctly installed web-based server.
Description	Customers can add Reviews for every product and also can see existing reviews.

Name	Manage Wishlist
Actors	Customer
Preconditions	Existence of database and correctly installed web-based server.
Description	Customers can add an item to Wishlist, see the Wishlist, delete item from Wishlist and also can move the items from Wishlist to the cart.

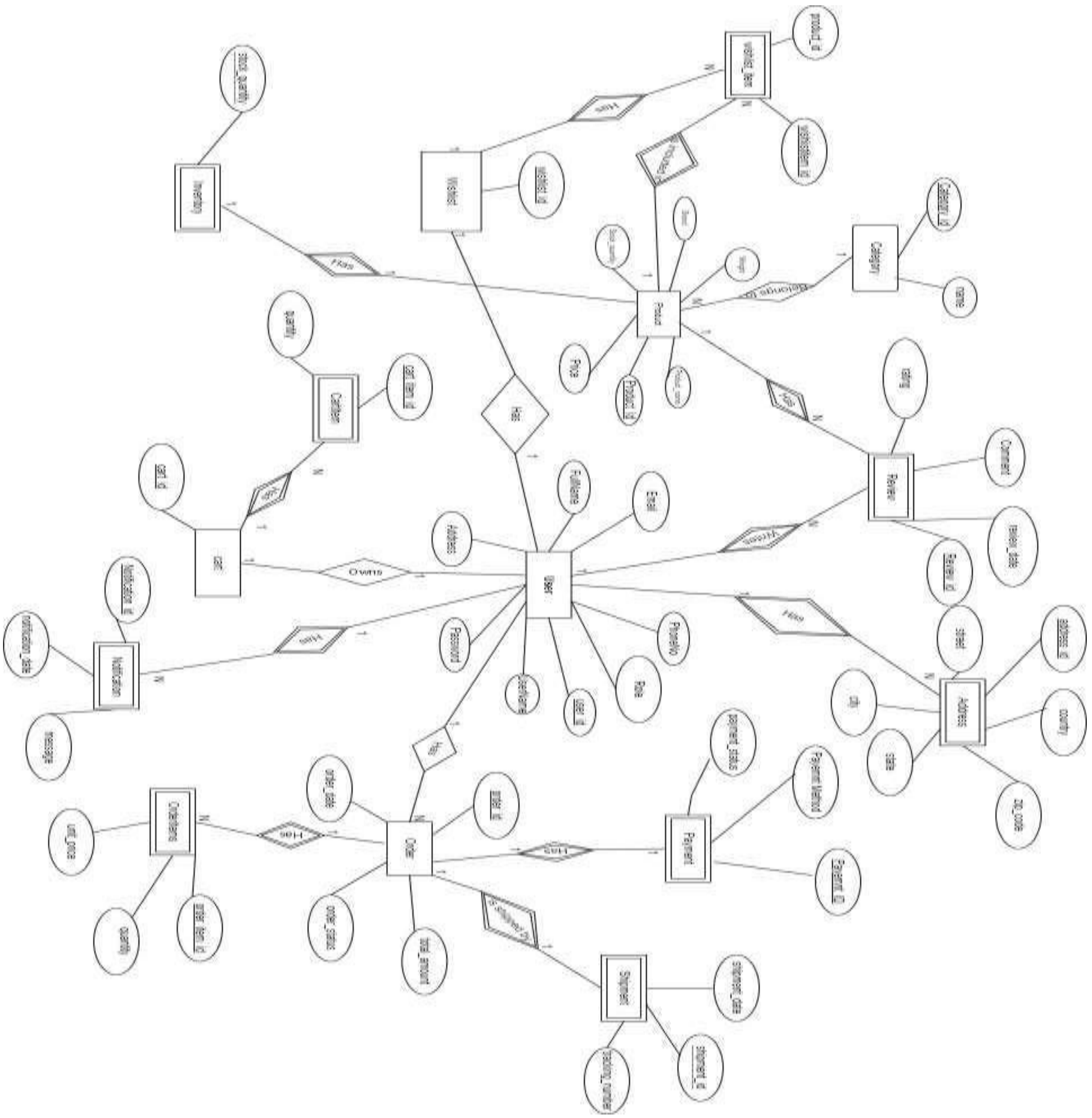
5. USE CASE DIAGRAM



6. SEQUENCE DIAGRAM



7. ENTITY-RELATIONSHIP DIAGRAM



product_id
wishlist_item
wishlistitem_id
Has
Includes
Wishlist
wishlist_id
stock_quantity
Inventory
Has
Product
Category
Category_id
name
Belongs to
Price
Product_id
rating
Review
Comment
review_date
Review_id
Writes
Has
quantity
Cartitem
cart_item_id
cart_id
cart
Has
Owns
User
Address
Fullname
Email
Password
Username
user_id
Role
PhoneNo
Notification
Notification_id
notification_date
message
Has
Has
street
address_id
city
Address
country
state
zip_code
Has
order_date
order_id
Order
Orderitems
unit_price
quantity
order_item_id
Has
Has
Payment
payment_status
Payment Method
Payment_ID
order_status
total_amount
Shipment
shipment_date
shipment_id
tracking_number
1
N