



**INSTITUTE FOR ADVANCED COMPUTING AND
SOFTWARE DEVELOPMENT, AKURDI, PUNE**

**“ShopStack
E-Commerce website”**

PG-DAC March 2025

Submitted By:

Group No: 107

Roll No.	Name of Student
252143	Dewanshu Misra
252227	Yash Bhangale
252157	Narayan Kande

Mrs. Geeta
Project Guide

Mr. Prashant Deshpande
Centre Coordinator

ABSTRACT

We extend our heartfelt gratitude to **Mrs. Geeta**, our project guide, for her invaluable guidance, continuous support, and insightful feedback throughout the course of this project. Her mentorship played a crucial role in shaping our understanding and approach.

We are also sincerely thankful to **Mr. Prashant Deshpande**, Centre Coordinator, for providing us with the necessary infrastructure, resources, and a conducive environment to carry out our work effectively.

Special thanks to our **peers and families**, whose constant encouragement, patience, and motivation helped us stay focused and dedicated. Their belief in us has been a source of strength during every phase of this journey.

Lastly, we would like to express our appreciation to everyone who directly or indirectly contributed to the successful completion of this project.

ACKNOWLEDGEMENT

I take this occasion to thank God, almighty for blessing us with his grace and taking our endeavor to a successful culmination. I extend my heartfelt thanks to our esteemed guide, **Mrs. Geeta** for providing me with the right guidance and advice at the crucial juncture and showing me the right way. I sincerely thank our respected Centre Co-Ordinator, **Mr.Prashant Deshpande**, for allowing us to use the available facilities. I would also like to thank the other faculty members at this occasion. Last but not least, I would like to thank my friends and family for the support and encouragement they have given me during our work.

Dewanshu Misra (252143)

Yash Bhangale (252227)

Narayan Kande(252157)

Table of Contents

Sr.No	Description	Page No.
1	Introduction	1
2	SRS	4
3	Diagrams	9
3.1	ER Diagram	9
3.2	Use Case Diagram	10
3.3	Data Flow Diagram	11
3.4	Activity Diagram	12
3.5	Class Diagram	15
3.6	Component Diagram	16
3.7	Sequence Diagram	17
4	Database Design	18
5	Snapshots	22
6	Conclusion	33
7	References	34

1. INTRODUCTION

1. Introduction

ShopStack provides an all-in-one online e-commerce platform that enables seamless interaction between sellers and customers, supported by an admin backend for system oversight. The application ensures secure access through role-based privileges, allowing each user type to perform relevant operations efficiently.

1.1 Purpose

The purpose of **ShopStack** is to digitize, automate, and streamline the online shopping process, reducing manual effort and delays while maintaining transparency and control. It bridges the gap between sellers and customers through a user-friendly interface, robust backend services, and secure transaction handling.

1.2 Scope

ShopStack is a web-based application compatible with both desktop and mobile browsers. It supports:

- Role-based authentication and authorization using Spring Security
- RESTful APIs with Swagger documentation
- Microservices architecture for modularity, scalability, and maintainability

1.3 Objectives

- ☐ Implement **Role-Based Access Control (RBAC)** for **CUSTOMER**, **SELLER**, and **ADMIN**
- ☐ Secure user authentication using **BCrypt** password encryption
- ☐ Provide interactive API documentation using **Swagger**
- ☐ Utilize **microservices** for product, order, payment, and user management modules
- ☐ Enable containerization using **Docker** for seamless deployment
- ☐ Maintain clean separation of concerns using **DTOs**, **Entities**, **Repositories**, **Services**, and

Controllers

2.Modules and Functionalities

2.1 User Module

Features:

- User Registration & Login
- Profile Management
- Address Book
- Wishlist

2.2 Cart Module

• Features:

- Add/Remove Items
- Update Quantity
- View Total Price

2.3 Order Module

• Features:

- Place Order
- Track Status
- Cancel/Return

• Order Statuses:

- Pending
- Confirmed
- Shipped
- Delivered
- Cancelled

2.4 Payment Module

• Features:

- Secure Checkout

- Multiple Payment Options
- Refunds & Receipts
- **Supported Methods:**
 - Credit/Debit Cards
 - UPI
 - Net Banking

🔒 Security & Compliance

- **Security Measures:**
 - HTTPS Encryption
 - JWT Authentication
 - Role-Based Access Control
- **Compliance:**
 - GDPR
 - PCI-DSS (for payments)

ADMIN Privileges:

- View, update, delete all users
- View, update, delete all items and orders
- Monitor system activity (optional future scope)

2.6 Deployment and Documentation

- **Swagger UI** for REST API testing and documentation
- **Postman Collection** for API testing
- **Docker** used for containerization (backend, frontend, database)
- **MySQL** as the database

2. SOFTWARE REQUIREMENT SPECIFICATION

2. SOFTWARE REQUIREMENT SPECIFICATION

2.1 Functional Requirements

1) User Module

- Register/Login with email and password
- Manage profile and addresses
- Add/remove items from wishlist

2) Cart Module

- Add products to cart
- Update quantity
- View total price
- Remove items

3) Order Module

- Place orders from cart
- Track order status
- Cancel or return orders

4) Payment Module

- Secure checkout
- Multiple payment options (UPI, cards, net banking)
- Generate receipts
- Handle refunds

2.2 Non-Functional Requirements

Performance

- API response time should be **under 2 seconds** for 95% of requests under normal load conditions
- Backend operations optimized using **pagination, indexed database queries**, and efficient data structures

Scalability

- Designed to support **10,000+ concurrent users**
- Stateless REST APIs make it cloud- and container-ready
- Easily deployable across multiple environments using **Docker**

Security

- Passwords are stored securely using **BCrypt hashing**
- APIs are protected using **JWT (JSON Web Tokens)** or **Spring Security**
- Role-based access controls prevent unauthorized actions
- Input validation and error handling to prevent SQL injection and XSS

Maintainability

- Codebase follows **modular architecture** (Controller → Service → Repository → DTOs → Entities)
- Follows **SOLID** principles and best practices
- Cleanly separated concerns for better testing and debugging
- Detailed comments and JavaDoc for future maintainers

Usability

- Simple and intuitive user interface (React)
- Responsive design for desktop and mobile browsers
- Clear error messages and validations to guide users

Portability & Deployment

- Containerized using **Docker** to ensure consistency across development, testing, and production environments
- Backend, frontend, and MySQL can run as separate containers
- Compatible with CI/CD tools for automated deployment

2.3 Non-Functional Requirements for ShopStack

1. Performance

- **Response Time:**
The system shall respond to user actions within 2 seconds under normal operating conditions.
- **Scalability:**
The system shall handle an increasing number of users and transactions without performance degradation. It should support at least 10,000 concurrent users.
- **Throughput:**
The system shall process at least 100 transactions per second during peak usage times.

2. Reliability

- **Availability:**
The system shall have an uptime of 99.9% over a 12-month period, ensuring high availability for users.
- **Fault Tolerance:**
The system shall continue to operate in the event of hardware or software failures, with minimal disruption to users.
- **Error Handling:**
The system shall gracefully handle errors and provide meaningful feedback to users when issues occur.

3. Usability

- **User Interface:**
The system shall have a user-friendly interface that is easy to navigate, with clear instructions and minimal learning curve.

4. Maintainability

- **Modularity:**
The system shall be designed in a modular fashion, allowing for easy updates and enhancements to individual components without affecting the entire system.
- **Code Quality:**
The system shall follow coding best practices, with well-documented, clean, and

maintainable code.

- **Testing:**

The system shall undergo thorough testing, including unit tests, integration tests, and user acceptance tests, to ensure quality and stability.

- **Other Requirements:**

Hardware and Network Interfaces:

Back-end Server Configuration:

- Intel Pentium-IV Processor
- 8 GB RAM

Front-end Client Configuration:

- AMD RYZEN 5 Processor
- 128 MB SDRAM
- 10 GB Hard Disk Drive
- 104 Keys Keyboard
- PS2 Mouse with pad

Software Interfaces:

Software configuration for back-end Services:

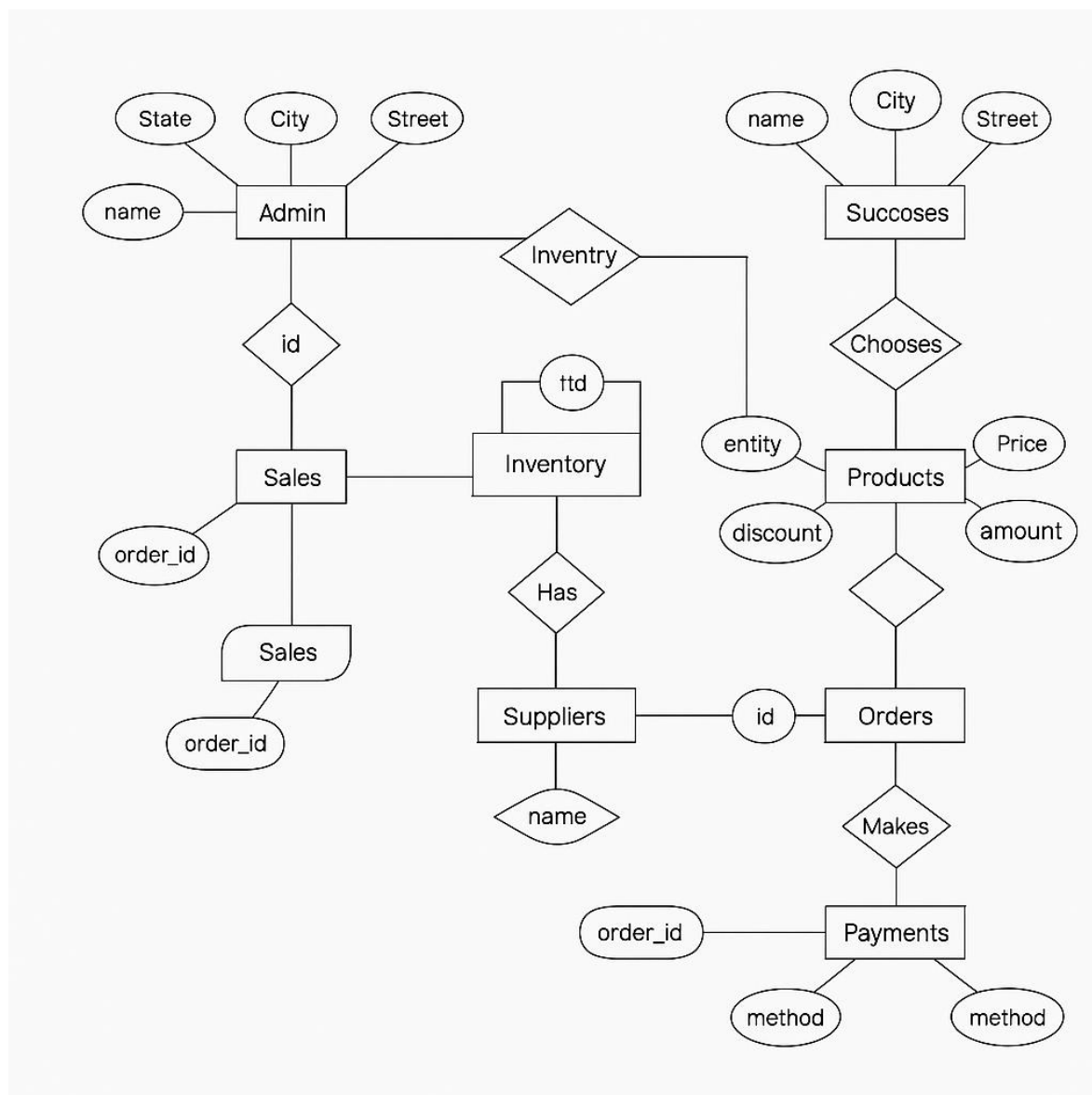
- Java EE
- Spring Boot, JPA, Razor Pay, Spring Authentication
- MySQL
- STS 3.9.18

Software configuration for front-end Services:

- ReactJS, Redux
- HTML, CSS, JS
- Bootstrap
- VS Code

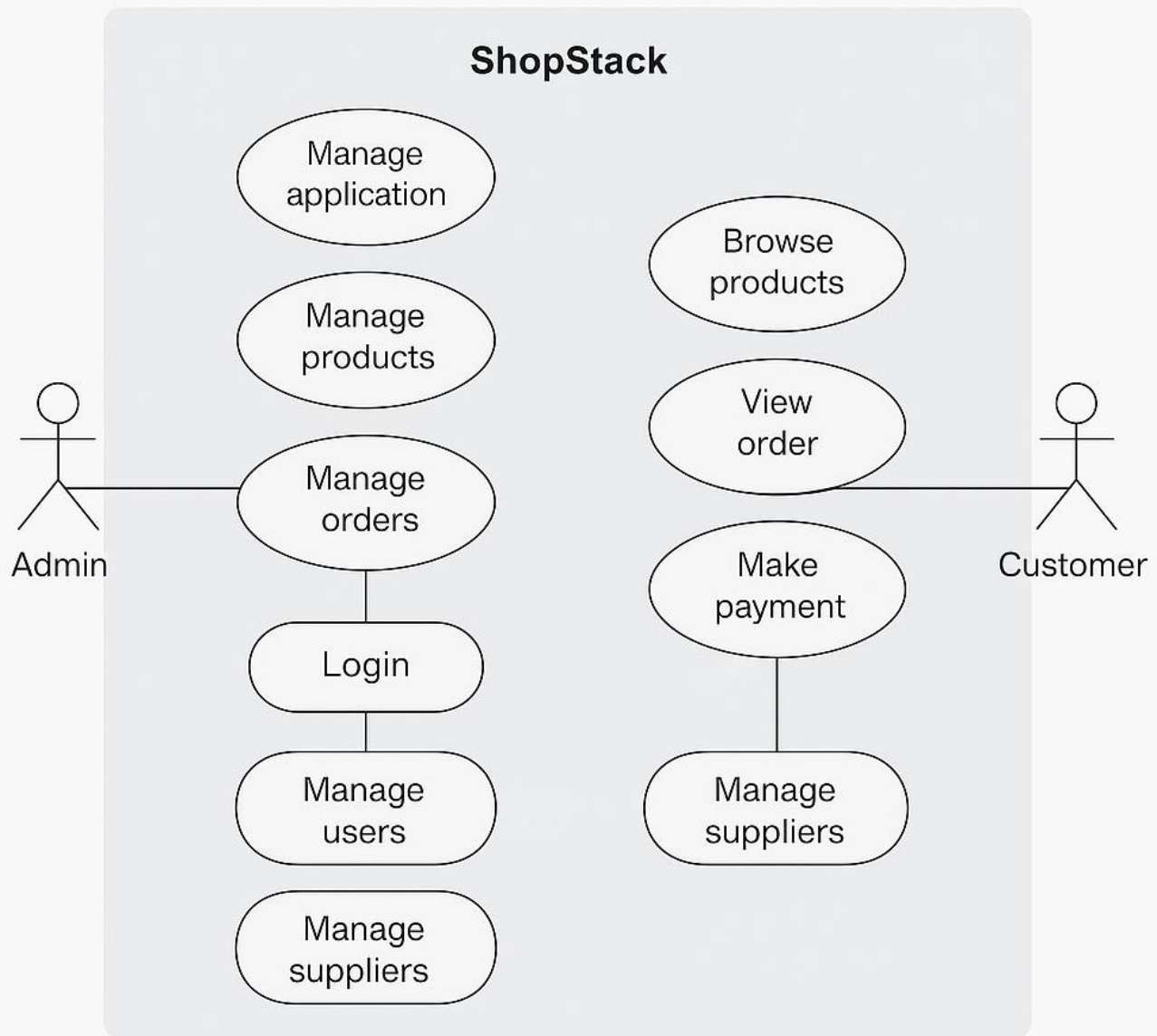
3. DIAGRAMS

3.1 Entity Relationship Diagram:



Er diagram of ShopStack

3.2 Use Case Diagram:



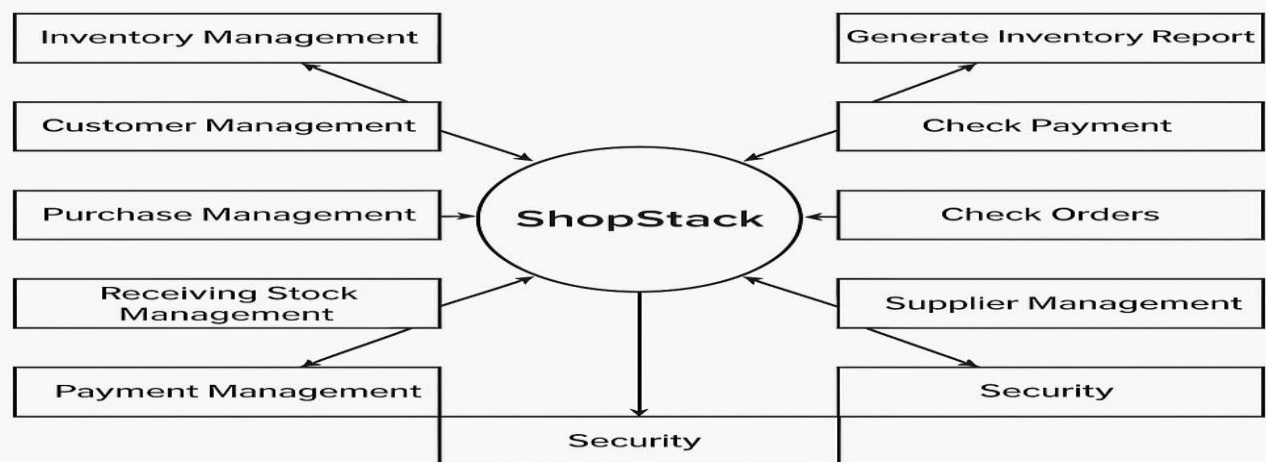
Use Case Diagram for ShopStack

3.3 Data Flow Diagram:

DFD Level 0:

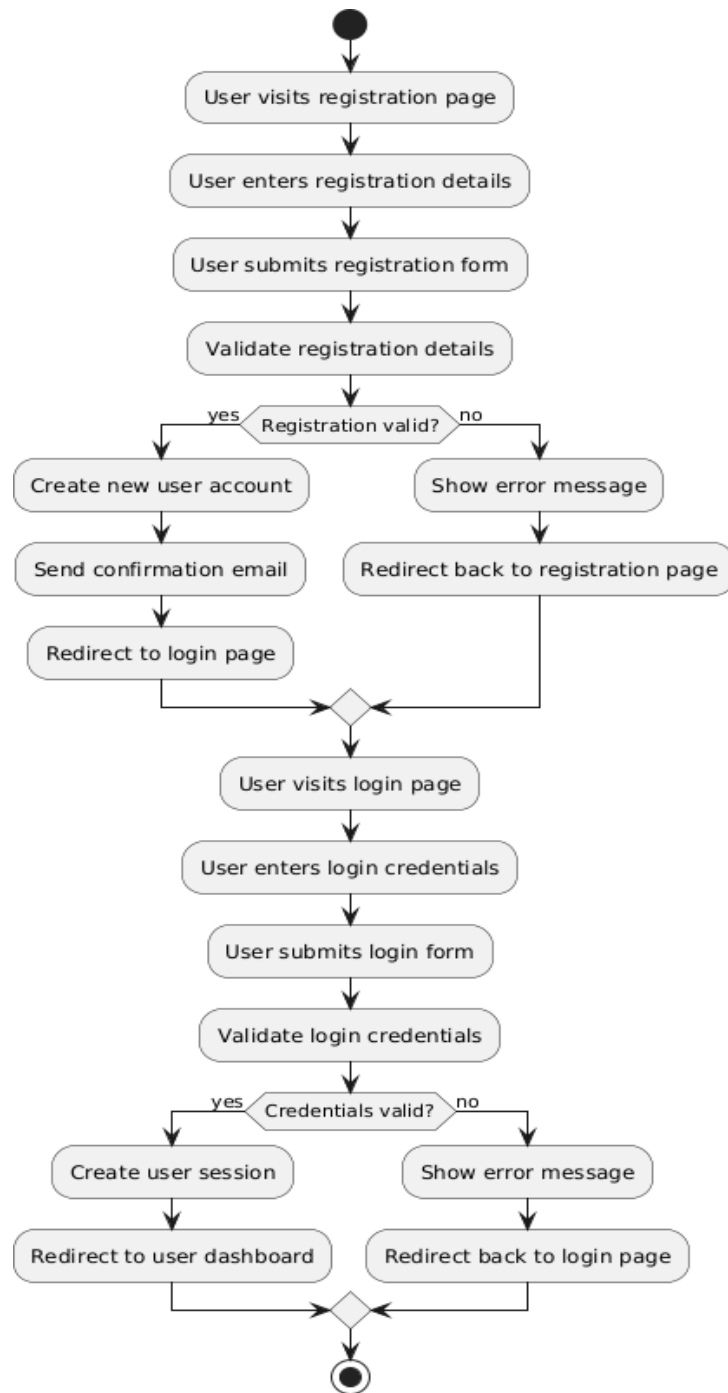


DFD level 1 :



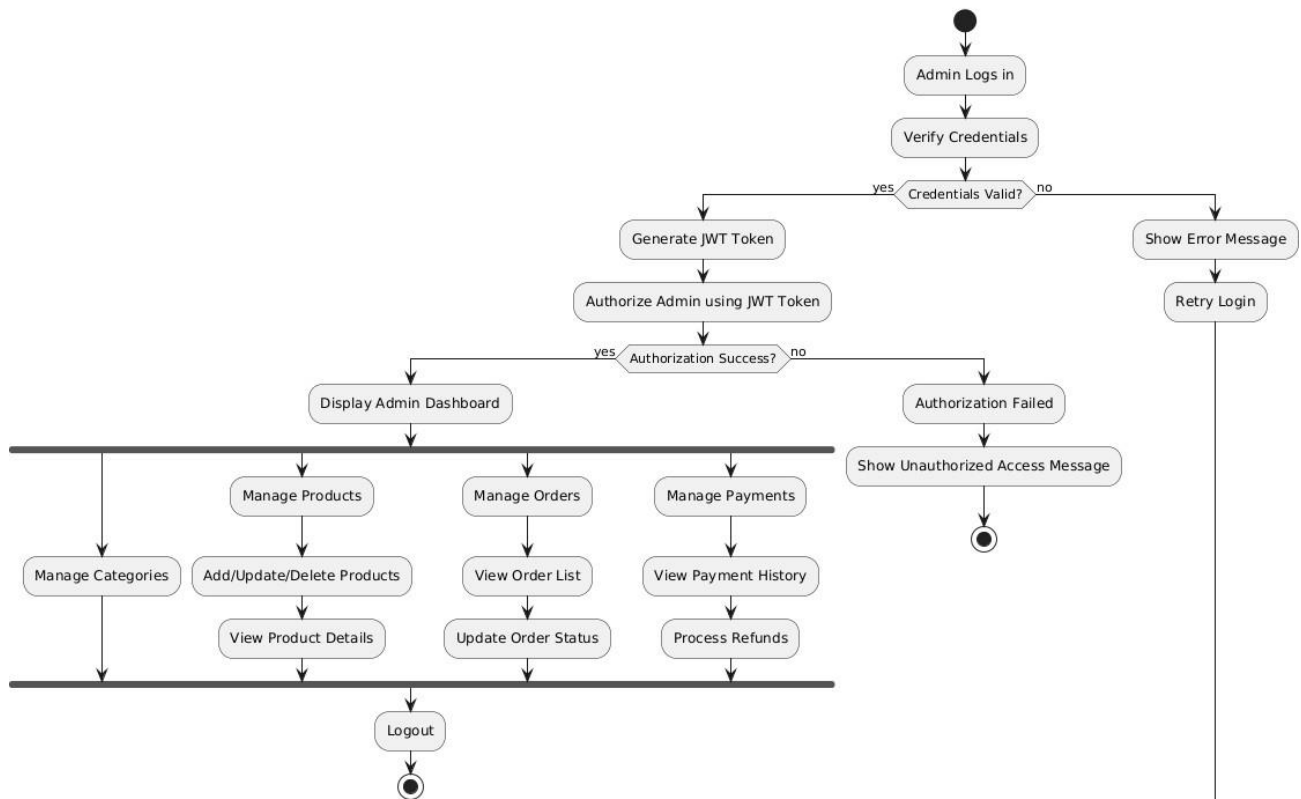
3.4 Activity Diagram :

1. login Activity Diagram



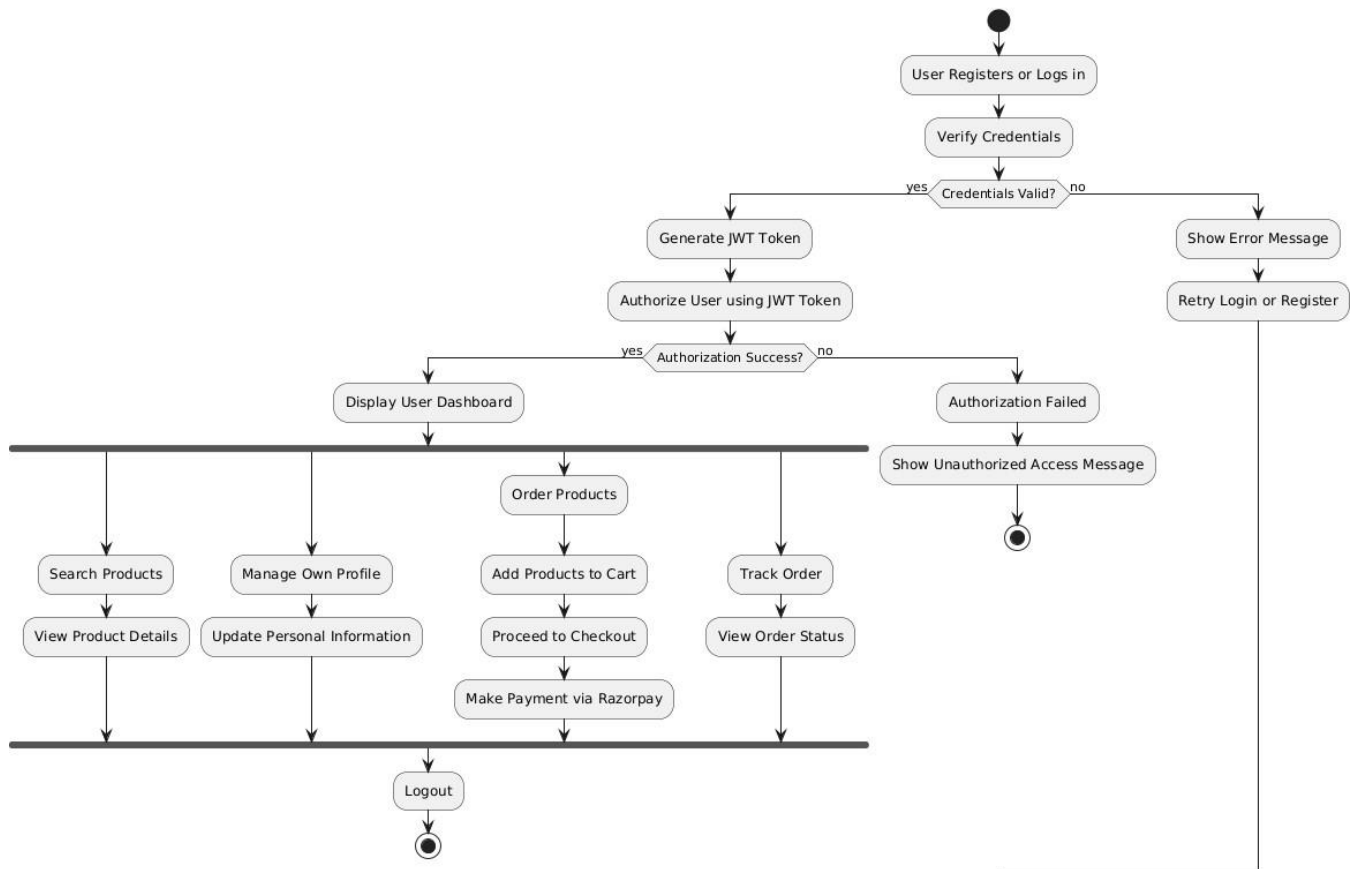
2. Admin Activity Diagram:

Admin Activity Diagram for Inventory SellMart

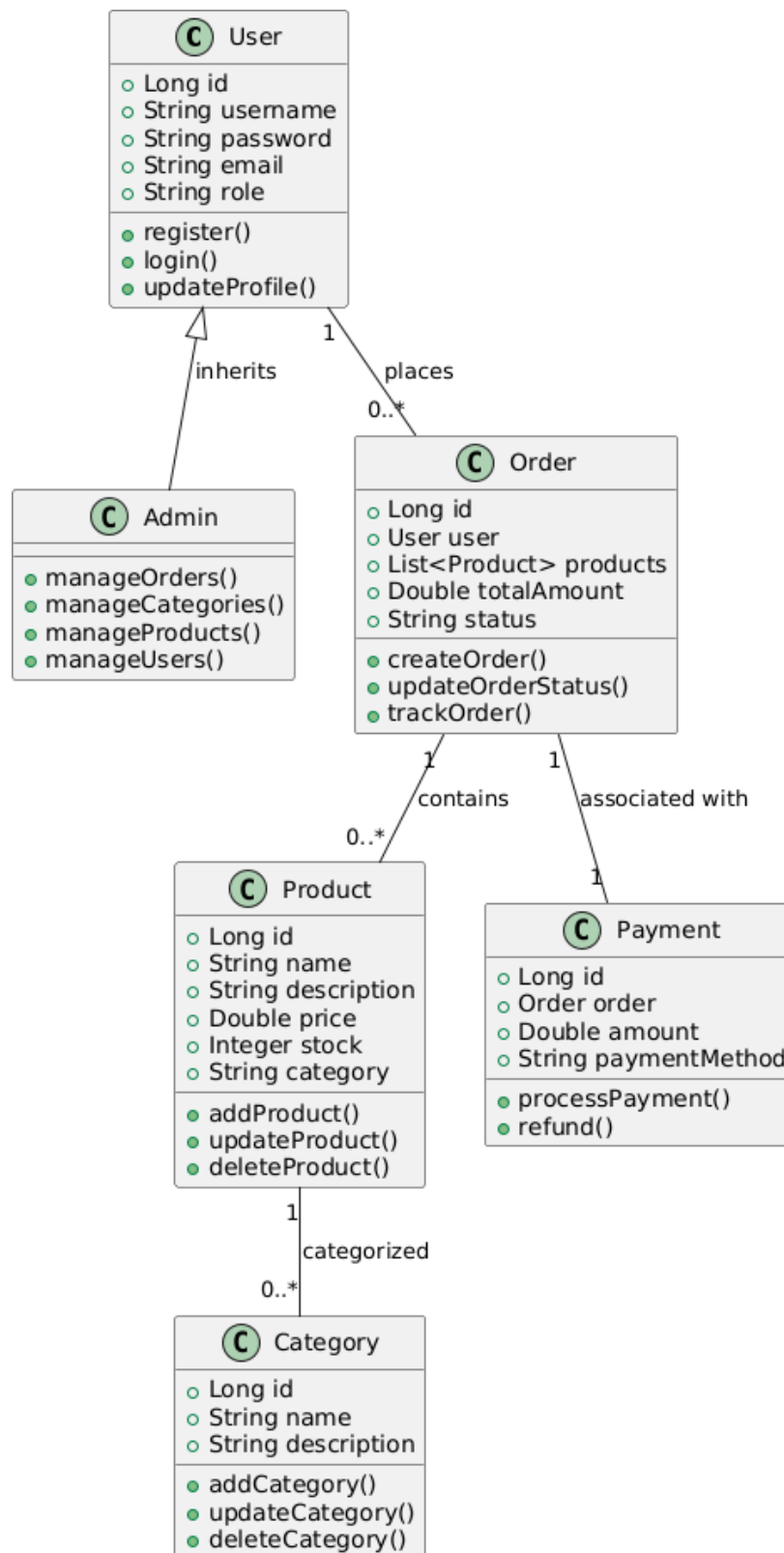


3. User Activity Diagram

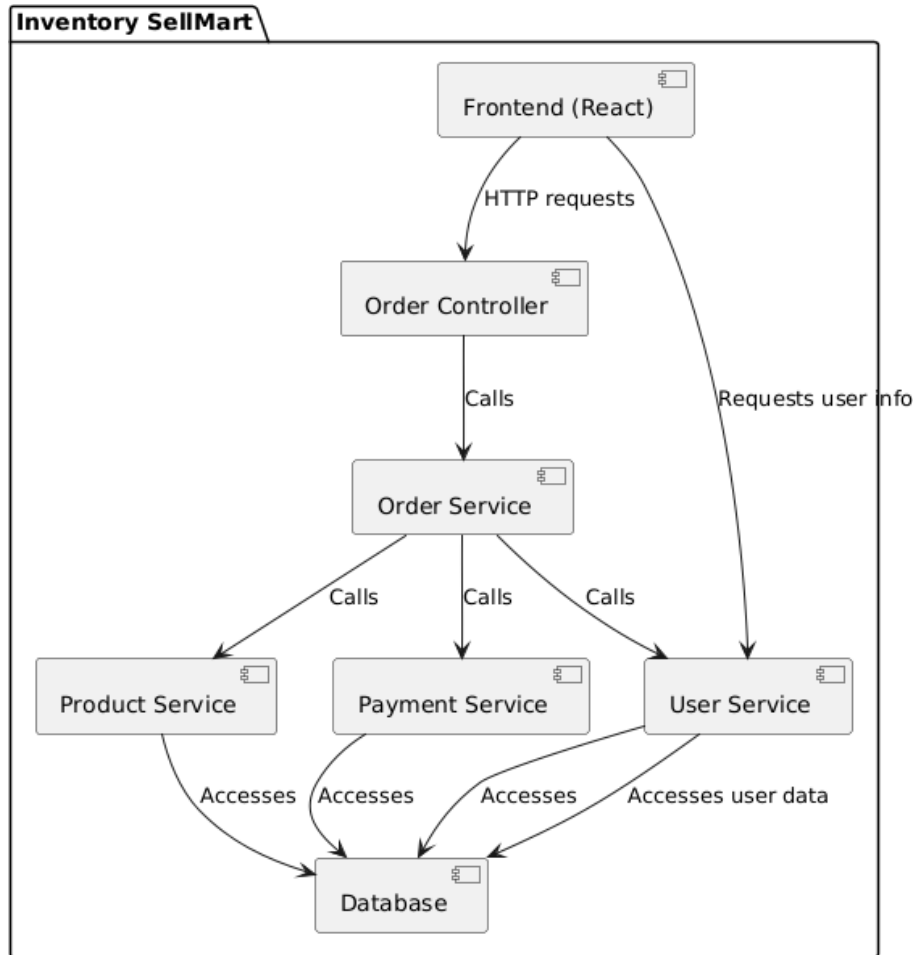
User Activity Diagram for Inventory SellMart



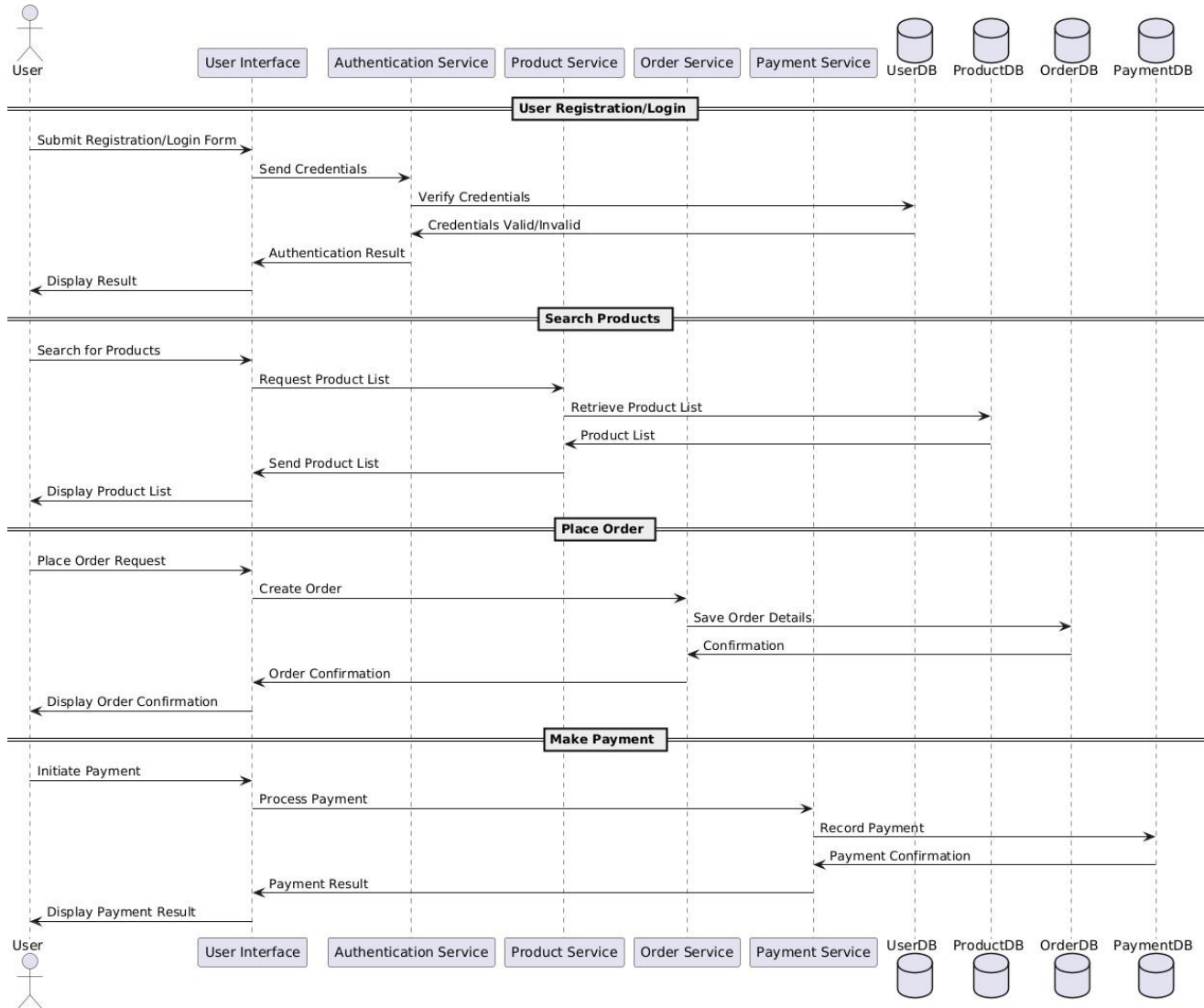
3.5 Class Diagram:



3.6 Component Diagram:

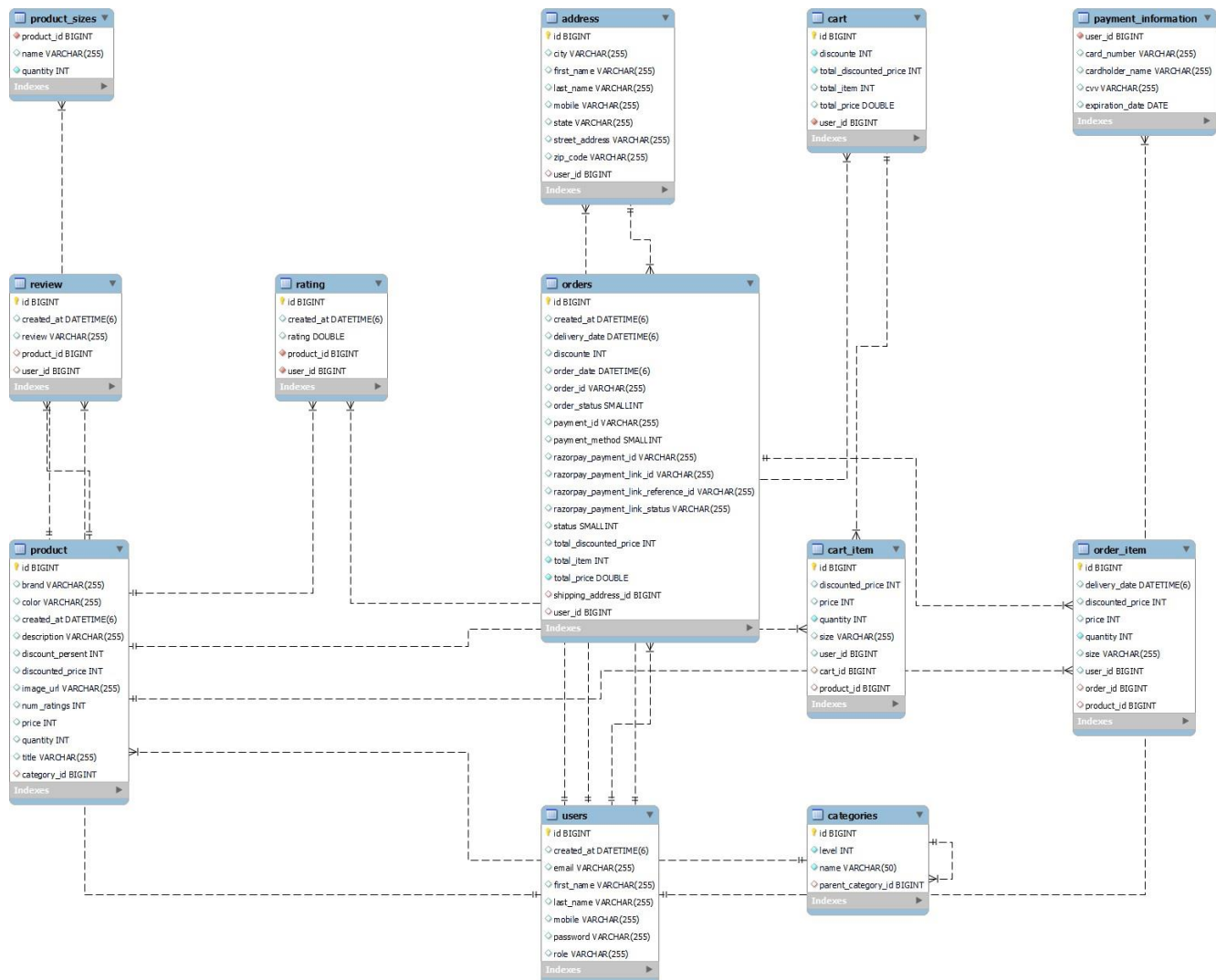


3.7 Sequence Diagram



4. DATABASE DESIGN

4.1 Design:



4.2 Tables:

The following table structures depict the database design.

Field	Type	Description
userId	UUID	Unique user identifier
name	String	Full name
email	String	Email address
password	Hashed	Encrypted password
addresses	Array	List of saved addresses

Table 1: user

Field	Type	Description
orderId	UUID	Unique order identifier
userId	UUID	Linked user
cartId	UUID	Snapshot of cart
status	String	Current order status

Table 2: orders

Field Name	Data Type	Description
productId	UUID / String	Unique identifier for each product
name	String	Name of the product
description	Text	Detailed description of the product
category	String	Product category (e.g., Electronics, Fashion)
brand	String	Brand or manufacturer
price	Number	Selling price

Table 3: Products

categoryId	UUID / String	Unique identifier for each category
name	String	Name of the category (e.g., Electronics, Fashion)
description	Text	Optional description of the category

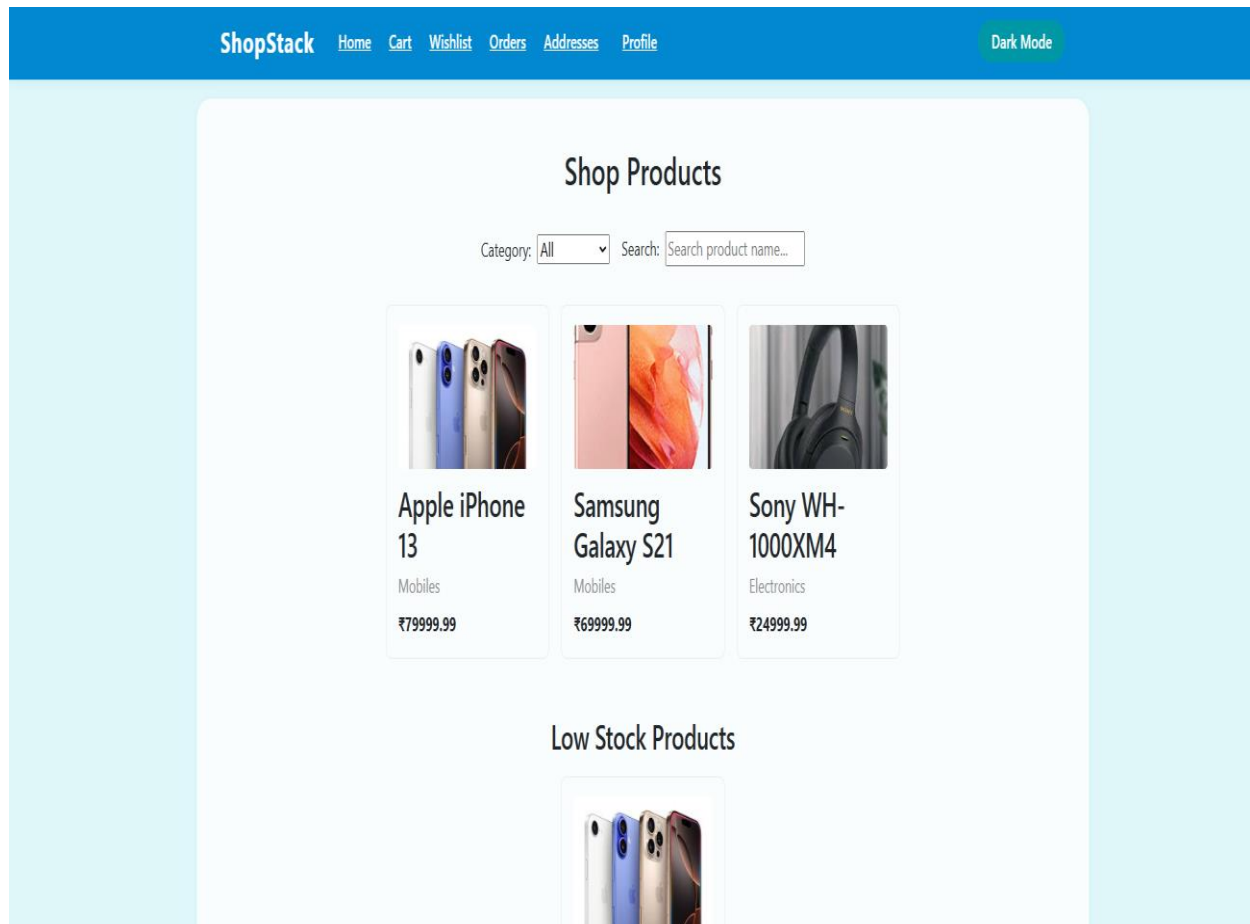
Table 4: Categories

Field	Type	Description
paymentId	UUID	Unique payment identifier
orderId	UUID	Linked order
amount	Number	Paid amount
method	String	Payment method used
status	String	Success/Failed/Pending

Table 5 : payment information

5. SNAPSHOTS

Home Page:



Login:

ShopStack

[Home](#)

[Cart](#)

[Wishlist](#)

[Orders](#)

[Addresses](#)

[Profile](#)

Dark Mode

My Profile

Username: user31

Email: user31@mail.com

Mobile No:

Edit Profile

Logout

Registration:**ShopStack**[Home](#)[Cart](#)[Wishlist](#)[Orders](#)[Addresses](#)[Login](#)[Register](#)[Dark Mode](#)**ShopStack**

Register

Already have an account? [Login](#)

Filtered Products:**ShopStack**[Home](#)[Cart](#)[Wishlist](#)[Orders](#)[Addresses](#)[Profile](#)[Dark Mode](#)

Shop Products

Category: MobilesSearch: 

Apple iPhone 13

Mobiles

₹79999.99



Samsung Galaxy S21

Mobiles

₹69999.99


Low Stock Products



Search Products:


ShopStack [Home](#) [Cart](#) [Wishlist](#) [Orders](#) [Addresses](#) [Profile](#) [Dark Mode](#)

Category: Mobiles Search:



Apple iPhone 13
Mobiles
₹79999.99

Low Stock Products



Apple iPhone 13

6. CONCLUSION

ShopStack successfully implements a secure, scalable, and user-friendly online e-commerce system that streamlines the buying and selling process for both customers and sellers. The system features:

- **Robust authentication and role-based access control** using Spring Security, ensuring data security and user privacy.
- **Modular microservices architecture** with RESTful APIs developed using Spring Boot and tested through Postman and Swagger, enabling seamless integration and future extensibility.
- **User-friendly React frontend**, delivering a responsive and intuitive interface for customers and sellers alike.
- **MySQL** as the backend database to ensure reliable data storage and fast retrieval.

Together, these features make **ShopStack** a modern, end-to-end solution for online retail management and e-commerce automation.

7. REFERENCES

1. <https://docs.spring.io/spring-security/reference/>
2. <https://docs.spring.io/spring-boot/index.html/>
3. <https://www.amazon.com/Agile-Software-Development-Principles-Patterns/dp/0135974445>
4. Garcia-Molina, H., Ullman, J. D., & Widom, J. (2008). *Database Systems: The Complete Book*. Prentice Hall.
5. Codd, E. F. (1970). *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM, 13(6), 377-387.
6. Fowler, M. (2003). *Patterns of Enterprise Application Architecture*. Addison-Wesley.
7. Evans, E. (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.
8. Soni, D., Nord, R. L., & Hofmeister, C. (1995). *Software Architecture in Industrial Applications*. Proceedings of the 17th International Conference on Software Engineering, 196-207.