PROJECT REPORT

Titled: ApnaDukaan

Problem Statement:

E-Commerce Site for Local Sellers

1. **Tech Stack: React + Spring Boot + MySQL**
2. **Modules: Seller Portal, Customer, Product Catalog, Cart**
3. **Use Case: Empowers local sellers to manage their store online**
4. **Evaluation: Cart mechanics, order workflow, seller CRUD, secure checkout flow**

Prepared By:

Name: Deepanshu Mandhyan

UID: 23BCS12327

Section: KRG_3A

1. Overview

ApnaDukaan is a community-driven e-commerce platform designed to empower local sellers by providing them with a simple and reliable way to take their businesses online. Unlike mainstream e-commerce giants that often impose high fees and complex onboarding processes, ApnaDukaan focuses on ground-level vendors, shopkeepers, and small business owners who struggle to establish a digital presence. The platform offers two primary interfaces: a Seller Portal for product management, inventory, and order handling, and a Customer Portal for browsing, cart management, and secure checkout. Built on a modern tech stack of React.js (frontend), Spring Boot (backend), and MySQL (database), ApnaDukaan ensures scalability, security, and ease of use. By bridging the gap between traditional markets and the digital economy, it creates opportunities for local businesses to thrive while offering customers the convenience of shopping from trusted local sellers.

2. Problem Statement

Despite the rapid growth of e-commerce, small and local sellers continue to face challenges in adopting digital platforms due to high costs, limited technical knowledge, and the dominance of large-scale vendors. Traditional shopkeepers and ground-level businesses often lack access to affordable, user-friendly tools to showcase their products online and connect with a broader customer base. This creates a digital divide where local sellers are unable to compete effectively, while customers have limited access to nearby, trusted vendors through online channels. ApnaDukaan aims to solve this problem by providing a dedicated e-commerce solution that is simple, affordable, and tailored to the needs of local sellers.

3. Objectives

- Enable **local sellers** to register, list products, manage inventory, and process customer orders through a simple seller portal.
- Provide customers with a **user-friendly platform** to browse products, add items to cart, and complete secure checkout.
- Ensure **reliable order workflow and secure transactions** using a robust backend with scalable architecture.
- Promote **local commerce and digital inclusion** by bridging the gap between traditional markets and online shopping.

3. Requirements Specification

3.1 Functional Requirements

- Sellers must be able to **register, log in, and manage their store** (add/edit/delete products, track inventory, view orders).
- Customers must be able to **browse products, add items to cart, place orders, and make secure payments**.
- The system should handle the **complete order workflow** – from cart management to order confirmation.
- Admin should be able to **monitor users, sellers, and transactions** to ensure transparency and compliance.

3.2 Non-Functional Requirements

- The platform must ensure **security** for user data and transactions (authentication, encryption).
- The system should be **scalable** to handle an increasing number of sellers, customers, and products.
- The application should provide a **responsive and user-friendly interface** across devices.
- The system should maintain **high availability and reliability**, ensuring minimal downtime.

4. System Architecture and Design

The project will be built using a decoupled architecture with React.js for the frontend and Java Spring Boot for the backend.

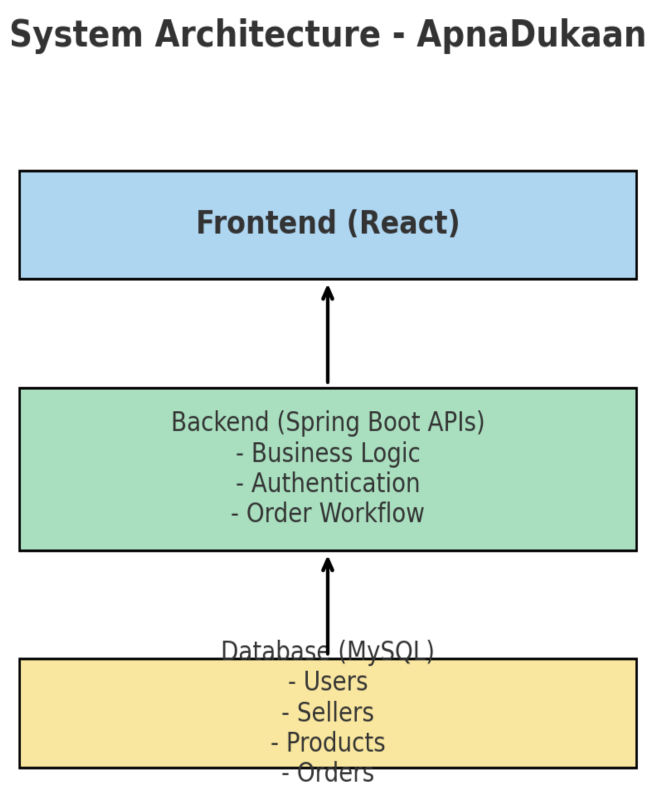
4.1 Technology Stack Components

- **Frontend:** React (with React Router for navigation, Axios/Fetch for API calls, simple CSS/Bootstrap/Tailwind for styling).
- **Backend:** Spring Boot (REST APIs, Spring Security with JWT for authentication, Spring Data JPA for ORM).

- **Database:** MySQL (all data storage – users, sellers, products, orders, payments)

4.2 System Architecture Diagram

1. **Presentation Layer (React):**
   - Customer UI – browse products, manage cart, place orders.
   - Seller UI – manage products, track orders, update inventory.
2. **Application Layer (Spring Boot):**
   - Handles business logic, authentication, order workflow, and secure transactions.
3. **Data Layer (MySQL):**
   - Stores all persistent data – users, sellers, products, orders, payments.



.

5. Scalability Plan

Database Scaling (MySQL)

- Use **proper indexing** on frequently queried fields (e.g., product_id, seller_id, order_id).
- Implement **read replicas** for handling customer-heavy traffic (browsing/catalog search).
- Use **partitioning/sharding** if product and order data grows very large.

Application Scaling (Spring Boot)

- Deploy Spring Boot as a **stateless service** so multiple instances can run in parallel.
- Use a **load balancer** (e.g., Nginx/HAProxy) to distribute incoming requests.
- Cache frequently accessed data (e.g., product catalog) in memory using Spring Cache.

Frontend Scaling (React)

- Host the React app on a **CDN or static hosting service** (Netlify, Vercel, or S3 + CloudFront) for fast global access.
- Implement **lazy loading & code splitting** to improve page performance.

Workflow Optimization

- For heavy catalog searches, integrate **server-side pagination** and filtering in APIs.
- For checkout, ensure **transactional consistency** using database transactions to avoid stock/order mismatches.

Growth-Ready Deployment

- Start with a **monolithic Spring Boot + MySQL** setup for simplicity.
- As traffic grows, separate **seller services, order services, and payment services** into modules or microservices.
- Migrate to **cloud deployment (AWS/GCP/Azure)** for auto-scaling infrastructure when required.

6. Conclusion

ApnaDukaan provides a simple yet powerful platform to bring local sellers online, enabling them to manage stores digitally while offering customers a secure and seamless shopping experience. Built with **React, Spring Boot, and MySQL**, it ensures reliability, scalability, and ease of use. The project not only addresses current needs but also holds strong potential for future growth, supporting the vision of a digitally connected local marketplace.