# Project Title:

E-Commerce.

## Description of the Project:

Our E-Commerce Platform is a web application designed to facilitate online buying and selling of products. It provides a user-friendly interface for both administrators and regular users. The platform allows users to browse products, make purchases, and manage their orders. Administrators have additional privileges to manage products, categories, and orders.

## Functional Requirements of the Project:

**Admin Features:**

1. Manage Products:

* Add new products with details such as name, description, price, and image.
* Update existing product information.
* Delete products from the inventory.

1. Manage Categories:

* Add new product categories.
* Update existing category names.
* Delete categories.

1. Manage Orders:

* View and update order status.
* View order details including products, quantities, and total amount.

**User Features:**

1. Browse Products:

* View a list of available products with details.
* Filter products by category or search by keyword.
* View individual product details.

1. Make Orders:

* Add products to the shopping cart.
* Proceed to checkout and place orders.
* View order history and details.

## Non-Functional Requirements of the Project:

**Performance:**

The application should have fast response times for user interactions such as browsing products, adding items to the cart, and processing orders.

Response times should be consistent, even during peak traffic periods.

Page load times should be optimized to provide a smooth user experience.

**Scalability:**

The system should be able to handle increasing numbers of concurrent users and growing product catalogs.

Horizontal scalability should be supported to accommodate increased traffic by adding more server instances or containers.

**Security:**

User authentication and authorization mechanisms should be robust to prevent unauthorized access to sensitive data and actions.

Passwords should be securely hashed and stored in the database.

Communication between the client and server should be encrypted using HTTPS to protect against eavesdropping and data tampering.

Input validation and sanitization should be implemented to prevent common security vulnerabilities.

**Reliability:**

The application should be highly available, with minimal downtime for maintenance or upgrades.

Failover mechanisms should be in place to handle server or component failures gracefully.

Data integrity should be maintained through proper error handling and transaction management.

**Usability:**

The user interface should be intuitive and easy to navigate, catering to users with varying levels of technical expertise.

Clear feedback should be provided for user actions such as form submissions and error messages.

Accessibility standards should be followed to ensure the application is usable by individuals with disabilities.

**Maintainability:**

Code should be well-structured and modular, following best practices and design patterns.

Documentation should be comprehensive, covering both code documentation and user manuals.

Automated testing should be implemented to ensure code quality and facilitate future changes without introducing regressions.

**Compatibility:**

The application should be compatible with a wide range of web browsers and devices, including desktops, laptops, tablets, and smartphones.

Responsive design techniques should be employed to ensure optimal viewing and interaction across different screen sizes and resolutions.

**Performance Monitoring and Logging:**

Monitoring tools should be in place to track application performance metrics such as response times, throughput, and error rates.

Logging should be implemented to record important events and transactions for troubleshooting and auditing purposes.

Alerts should be configured to notify administrators of critical issues or anomalies in real-time.

## Architecture Scope of the Project:

**Frontend: React.**

The frontend of the application is developed using React.js, a popular JavaScript library for building user interfaces.

React components are responsible for rendering UI elements and handling user interactions.

Fetch API is used to communicate with the backend server to fetch data and perform CRUD operations.

**Backend: Spring Boot.**

The backend of the application is built using Spring Boot, a Java-based framework for building web applications.

Spring Boot provides features such as RESTful APIs, dependency injection, and security.

Spring Data JPA is used for data persistence, allowing easy interaction with the database.

Spring Security is employed to manage authentication and authorization, ensuring secure access to resources based on user roles.

**Database**:

The application utilizes a relational database (PostgreSQL) to store product information, user data, and order details.

Hibernate ORM is used for object-relational mapping, simplifying database interactions and ensuring portability across different database providers.

**Deployment:**

Docker containers are being used for packaging the application, providing consistency and scalability across different environments.

The application can be deployed on cloud platforms such as AWS, Azure, or Heroku.