E-Commerce Platform with AI-Powered Recommendation System

Project Report Submitted in partial fulfillment for the degree of  
MASTER IN COMPUTER APPLICATION (MCA)  
Kurukshetra University

Submitted By: Ravi Kumar Sohal  
STUDENT ID: 11097325

ENROLLMENT #: 24DEOJULYMCA00442  
MCA 3rd Semester

Guide: \_\_\_\_\_\_\_\_\_\_\_  
Qualification: MCA (with 3+ years IT experience)  
Kurukshetra University

# Acknowledgement

I would like to express my sincere gratitude to my project guide for their continuous support, patience, and motivation throughout this project.   
Their immense knowledge and valuable guidance have been a source of inspiration and helped me in successfully completing this work.  
  
I also thank my faculty members and peers at Kurukshetra University who provided valuable feedback during the development phase.   
Finally, I extend heartfelt thanks to my family and friends for their encouragement and constant support during this project.

# Certificate

This is to certify that this project, entitled “**E-Commerce Platform with AI-Powered Recommendation System**,” submitted in partial fulfillment of the degree of MASTER IN COMPUTER APPLICATION (MCA) to the Kurukshetra University by Mr. **Ravi Kumar Sohal**, STUDENT ID: **11097325** & ENROLLMENT #: **24DEOJULYMCA00442,** is authentic work carried out by him under my guidance.

The matter embodied in this project work has not been submitted earlier for the award of any degree or diploma to the best of my knowledge and belief.

**Signature of the Student:** \_\_\_\_\_\_\_\_\_\_\_\_

**Signature of the Guide:** \_\_\_\_\_\_\_\_\_\_\_\_

Date: \_\_\_\_\_\_\_\_\_\_\_\_

# Synopsis / Abstract

Title of the Project:

E-Commerce Platform with AI-Powered Recommendation System

Problem Statement:  
Existing e-commerce platforms often struggle with providing personalized shopping experiences due to a one-size-fits-all approach to product displays. This leads to information overload, where users are presented with a vast catalogue of products, many of which are irrelevant to their interests. The absence of a tailored recommendation system results in lower customer engagement, missed up-selling and cross-selling opportunities, and a sub-optimal user experience.

Why Chosen:

This project directly addresses a crucial and highly relevant topic in modern technology: e-commerce and AI/machine learning.

It allows you to apply theoretical knowledge from multiple academic disciplines—web development, database management, and data science—into a single, integrated application. Completing this project will demonstrate your ability to work with a full-stack architecture and a diverse technology stack, including gaining hands-on expertise in:

**Frontend Development:** Building a dynamic user interface with Angular, HTML, CSS, and JavaScript.

**Backend Development:** Creating robust REST APIs and managing business logic using the industry-standard Java Spring Boot framework.

**Database Management:** Designing a relational database schema and interacting with it using Spring Data JPA.

**Machine Learning:** Implementing a recommendation model in R and deploying it as an API, tackling real-world challenges like the cold start problem and data sparsity.

**System Integration:** Learning how to connect and make different services communicate seamlessly.

Objective & Scope:

The core objective of this project is to build an e-commerce platform that goes beyond basic functionality by integrating an AI-powered recommendation engine. The system aims to enhance the user experience and increase sales by providing personalized product suggestions based on user behaviour. The scope of the project covers three main areas: a responsive frontend built with Angular for user interaction and admin management, a robust backend developed with Java Spring Boot to handle all e-commerce business logic and APIs, and a separate R service for the recommendation engine that will analyze user data to generate and serve intelligent product recommendations.

Methodology:

This project will be developed using a modular architectural approach to ensure scalability and maintainability. Each core component—frontend, backend, and recommendation engine—will be developed as a separate service.

The system is built using Java Spring Boot (backend), Angular 20 (frontend), MySQL (database), and R microservice for recommendations.

APIs ensure secure communication. Authentication and authorization are handled using JWT with Spring Security.

The frontend, built with Angular 20, focuses on a component-based design for a seamless user experience.

All services will be integrated and tested for seamless communication and functionality, with version control managed via Git and GitHub to facilitate collaborative development.

Hardware & Software:

**Hardware:** i5/i7 processor, 8GB+ RAM, 250GB+ storage.

**Software:**

Java 21, Spring Boot, Spring Security

Angular, Node.js (for frontend build)

MySQL Database & R (Recommendation Engine)

Testing:

Unit testing, Integration testing, and UAT (User Acceptance Testing) were performed.   
JUnit and Postman were used to validate APIs.

Contribution:

The project demonstrates integration of web development and AI, aligning with real-world industry practices.   
It provides a blueprint for scalable, secure, and intelligent e-commerce platforms.

# Main Report

## Objective & Scope of the Project

The primary objective of this project is to design and develop a fully functional e-commerce platform with a key focus on integrating an AI-powered recommendation system. The goal is to move beyond a static product catalog by leveraging user behavior data to provide personalized product suggestions. This personalization aims to enhance the user shopping experience, increase customer engagement, and drive sales through more effective up-selling and cross-selling. The project also includes a comprehensive admin module for efficient management of products, orders, and user data.

The project's scope is divided into three main components:

Frontend Web Portal: A responsive and intuitive user interface built with Angular 20, allowing customers to browse products, add items to a shopping cart, manage their profiles, and complete a simulated checkout process. The admin dashboard provides statistical insights and management tools for products, categories, orders, and users.

Backend E-commerce Core: A robust backend system developed using the Java Spring Boot framework. This module will handle all core e-commerce functionalities, including user authentication and authorization (for both customers and admins), product and order management, shopping cart logic, and data persistence with a MySQL database. It will expose a set of REST APIs to be consumed by the frontend.

AI Recommendation Engine: A separate service built in the R language that analyzes user interactions (e.g., views, purchases) stored in the database. This engine will use machine learning models to generate personalized product recommendations, which will be served to the frontend via a dedicated API. The scope includes addressing the "cold start problem" by providing initial recommendations for new users or products.

## Theoretical Background

This project is built upon the theoretical foundations of several key computer science and data science domains. The e-commerce platform relies on principles of web development and software engineering, including the Model-View-Controller (MVC) architectural pattern, which provides a clear separation of concerns for the frontend (View), backend logic (Controller), and data management (Model). The RESTful API design is crucial for enabling stateless and scalable communication between the frontend and backend. The core innovation of the project—the AI recommendation system—is grounded in the field of machine learning. Specifically, it will apply collaborative filtering, an algorithm that predicts a user's interest in products by collecting preferences from many users. This method addresses the challenge of data sparsity by identifying patterns in user behavior, such as items frequently purchased together, to generate accurate and personalized product suggestions.

## Definition of Problem

The primary problem this project addresses is the lack of personalization in conventional e-commerce platforms. Many online retail websites present a static, one-size-fits-all product catalog to every user, regardless of their individual preferences or past behavior. This approach often leads to customer frustration and information overload, where users must sift through countless irrelevant products to find what they're looking for. This results in a suboptimal shopping experience, lower user engagement, and missed opportunities for businesses to increase sales through up-selling and cross-selling. To solve this, there is a clear need for a system that can intelligently analyze user behavior and dynamically recommend products, creating a more engaging and efficient shopping journey for each customer.

## System Analysis & Design vis-a-vis User Requirements

The success of this e-commerce platform hinges on a thoughtful analysis of user needs, followed by a robust system design that translates those needs into a functional and reliable application. This section outlines the requirements from the perspective of two key user types—customer and admin—and details how the proposed system architecture is designed to meet each of these requirements.

User Requirements

This project is designed to cater to two distinct user groups, each with a unique set of needs and functionalities.

Customer User Requirements

* **Authentication & Profile Management:**
  + Ability to register for a new account.
  + Securely log in and log out of the system.
  + Manage personal details, shipping addresses, and billing addresses in their profile.
* **Product Interaction:**
  + Browse and search for products by category or keywords.
  + View detailed product information, including descriptions, prices, and stock status.
  + Receive personalized product recommendations on their dashboard and product pages.
* **Shopping Cart & Checkout:**
  + Add products to a virtual shopping cart.
  + View and modify cart contents (e.g., change quantities, remove items).
  + Proceed to a simulated checkout process to complete an order.
* **Order Management:**
  + View their order history and status.

Admin User Requirements

* **Dashboard & Analytics:**
  + Access a dedicated dashboard to view shop statistics.
  + See a list of low-stock products to manage inventory.
* **Content Management:**
  + Add, edit, and delete product categories.
  + Add, edit, and delete products, including details, images, and pricing.
* **Order & User Management:**
  + View and manage all customer orders.
  + View and manage all registered user accounts.

A diagram of a product

AI-generated content may be incorrect.

Figure User Activity Diagram

System Analysis

The analysis phase deconstructs the user requirements to define the technical functionalities and data models required to support them.

Customer-facing Functionality: The need for browsing, searching, and viewing products dictates the creation of a Product entity with fields for name, description, price, and stock. The shopping cart requires a Cart and CartItem model. User authentication and profile management necessitate a User entity with secure password storage and fields for personal and address details. The order process requires Order and OrderItem entities.

Admin-facing Functionality: The requirements for an admin dashboard and management tools necessitate dedicated Admin APIs and a robust database schema to store and retrieve data for products, categories, orders, and users. The "low-stock" feature requires a query that filters products based on a stock count threshold.

Recommendation Engine: The core requirement for AI-driven recommendations is the most complex. It requires a dedicated data model to capture and store UserInteractions (e.g., product views, purchases). This data will be the input for the machine learning model, which will then process it to generate a list of recommended product IDs. This necessitates a separate service with a dedicated API endpoint.

System Design

Based on the analysis, a three-tier architecture with a dedicated machine learning service is the ideal design choice. This modular approach ensures that each component can be developed, scaled, and maintained independently.

**A diagram of a diagram

AI-generated content may be incorrect.**

Figure End to End System Flow

* Tier 1: Frontend (Angular 20)
  + The user interface is built as a single-page application (SPA) using Angular 20. It's responsible for the user experience, including data display, form handling, and user interaction.
  + It communicates with the backend exclusively via RESTful APIs using Angular's HttpClient module.
* Tier 2: Backend (Java Spring Boot)
  + This layer acts as the business logic and data access tier.
  + It receives requests from the frontend, processes them (e.g., user authentication, validating cart items), and interacts with the database.
  + Crucially, it acts as a central hub, routing API calls to other services. When the frontend requests recommendations, the Spring Boot backend will make a request to the R ML Service.
  + Spring Security will be configured to handle authentication and enforce role-based access control for public, customer, and admin-specific endpoints.
* Tier 3: Database (MySQL)
  + A centralized MySQL database stores all persistent data for the application, including products, categories, users, orders, and crucially, user interaction data for the recommendation engine.
* Separate Service: Recommendation Engine (R Language)
  + This is an independent service that exposes its own REST API.
  + It connects to the MySQL database to pull the latest UserInteraction data.
  + It runs the recommendation algorithms using libraries like recommenderlab and returns a list of recommended products.

Mapping Requirements to Design

| **User Requirement** | **System Analysis** | **System Design Solution** |
| --- | --- | --- |
| **User Registration & Login** | User data model, password hashing, and secure API endpoints. | Angular forms connect to Spring Boot's /auth endpoints. Spring Security handles secure authentication and token generation. |
| **Browse & Search Products** | Product data model; API for product listing and search. | Angular displays products from a /products API call to the Spring Boot backend, which queries the MySQL products table. |
| **AI-powered Recommendations** | UserInteraction data model; a separate ML service API. | A dedicated R service processes UserInteraction data from MySQL. The Spring Boot backend calls the R service's API and returns recommendations. |
| **Shopping Cart & Checkout** | Cart and OrderItem models; APIs to add, remove, and update items. | Angular components manage cart state and interact with the /cart and /checkout endpoints in the Spring Boot backend. |
| **Admin Dashboard & Analytics** | APIs for aggregated data; queries for low-stock items. | Secure Spring Boot APIs (/admin/dashboard, /admin/low-stock) provide data to the Angular admin dashboard. |
| **Product & Category Management (Admin)** | Full CRUD (Create, Read, Update, Delete) functionality for Product and Category entities. | Angular admin forms send requests (POST, PUT, DELETE) to secure Spring Boot endpoints (/admin/products, /admin/categories). |
| **Order & User Management (Admin)** | Full CRUD functionality for Order and User entities. | Admin-specific APIs in Spring Boot (/admin/orders, /admin/users) allow the admin to view and modify data. |

## System Planning (PERT Chart)

The Program Evaluation and Review Technique (PERT) Chart is a project management tool used to visualize the timeline, dependencies, and critical path for a project. It provides a structured approach to planning by breaking down the entire project into a series of individual tasks, estimating the time required for each, and identifying the sequence in which they must be completed.

Project Task Breakdown & Dependencies:

Phase 1: Initial Planning & Setup

**Task 1.1: System & Database Design (1 week):** Define the architecture, database schema (tables for products, users, orders, etc.), and API endpoints. Dependencies: None.

**Task 1.2: Environment Setup (2 days):** Install and configure development environments (IDEs, Node.js, Spring Boot, MySQL, R) and set up the Git repository. Dependencies: Task 1.1.

Phase 2: Backend Development (Java Spring Boot)

**Task 2.1: User Management API (3 days):** Develop APIs for user registration, authentication, login, and profile management. Dependencies: Task 1.2.

**Task 2.2: Product & Category API (2 days):** Create APIs for managing products and categories (CRUD operations). Dependencies: Task 2.1.

**Task 2.3: Cart & Order Management API (7 Days):** Develop the core e-commerce logic for adding items to the cart, simulated checkout, and order history. Dependencies: Task 2.2.

**Task 2.4: Admin Dashboard API (2 Days):** Implement APIs for fetching shop statistics and low-stock product lists. Dependencies: Task 2.3.

Phase 3: Frontend Development (Angular 20)

**Task 3.1: Core UI/UX Design (2 weeks):** Design the overall user interface for both the customer and admin portals. Dependencies: Task 1.1.

**Task 3.2: User Interface Implementation (2 days):** Develop the login, registration, and user profile pages. Dependencies: Task 2.1, 3.1.

**Task 3.3: Product Catalog & Cart UI (4 days):** Build the product browsing pages, search functionality, and shopping cart interface. Dependencies: Task 2.2, 3.2.

**Task 3.4: Admin Dashboard UI (3 days):** Create the dashboard with product management, order history, and statistical views. Dependencies: Task 2.4, 3.3.

Phase 4: Recommendation Engine & Integration

**Task 4.1: Data Collection Logic (1 day):** Implement backend logic to capture user interactions (views, purchases) and store them in the database. Dependencies: Task 2.2.

**Task 4.2: Recommendation Engine Model (5 days):** Develop and train the recommendation model in R using *recommenderlab*. Dependencies: Task 4.1.

**Task 4.3: ML API Development & Deployment (1 days):** Use plumber to expose the R model as a RESTful API. Dependencies: Task 4.2.

**Task 4.4: Frontend & Backend Integration (1 day):** Connect the frontend to the backend APIs and the backend to the R ML API. Dependencies: Task 2.4, 3.3, 4.3.

**Task 4.5: Final Testing & Deployment (1 day):** Conduct comprehensive testing of all components and deploy the application. Dependencies: Task 3.4, 4.4.

## Process Logic of Each Module

**1. User Management Module**

**Purpose**: Handles customer and admin accounts, registration, login, authentication, and profile management.

**Logic Flow**:

* User visits registration/login page.
* On registration, frontend validates input (email, password rules, etc.).
* Backend (Spring Boot) checks if user already exists in the database (MySQL).
* If new, user details are hashed (for password security) and stored.
* For login, credentials are validated; JWT token is generated for secure session.
* Authenticated users can update profile details (name, address, preferences).
* Admin users are differentiated by role flag (e.g., role=admin).

A diagram of a company

AI-generated content may be incorrect.

Figure : Authentication Flow

A diagram of a product

AI-generated content may be incorrect.

Figure Customer User Activity Diagram

A diagram of a server

AI-generated content may be incorrect.

Figure User Controller Flow

2.Product Management Module

**Purpose**: Manages the catalog of products, their categories, stock, and details.

**Logic Flow**:

* Admin adds/updates/deletes products via admin dashboard.
* Product details (name, description, category, price, stock, image URL) are validated on frontend.
* Backend stores/updates product records in MySQL database.
* When customers browse, frontend fetches product list via REST APIs.
* Category filters and search queries are applied at the backend using SQL queries.
* Product stock is updated automatically after each successful order.

A diagram of a product

AI-generated content may be incorrect.

Figure Product Controller Flow

3. **Category Management Module**

**Purpose**: Organizes products into categories for better browsing and search.

**Logic Flow**:

* Admin creates categories (e.g., Electronics, Clothing, Home Appliances).
* Each category is assigned a unique ID in the database.
* Products are linked to categories using foreign keys.
* When customers browse, frontend displays categories with associated products.
* API endpoints fetch category-wise products for filtering and recommendations.

A diagram of a company

AI-generated content may be incorrect.

Figure Category Controller Flow

4. **Shopping Cart Module**

**Purpose**: Allows customers to add/remove items before checkout.

**Logic Flow**:

* Customer browses and clicks “Add to Cart.”
* Frontend stores cart items temporarily (local state/session storage).
* On login, cart is synced with backend (saved in MySQL with user ID).
* Quantity updates trigger recalculation of cart totals.
* If product is out of stock, backend prevents adding to cart.
* Cart details are fetched during checkout.

A screenshot of a computer

AI-generated content may be incorrect.

Figure Cart Controller Flow

5. **Order Management Module**

**Purpose**: Handles order placement, processing, status updates, and history.

**Logic Flow**:

* Customer confirms cart → proceeds to checkout.
* Backend verifies stock availability.
* If available, order record is created in MySQL with “Pending” status.
* Stock is reduced accordingly.
* Admin can update order status (Shipped, Delivered, Cancelled).
* Customers can view past orders from “My Orders” section.

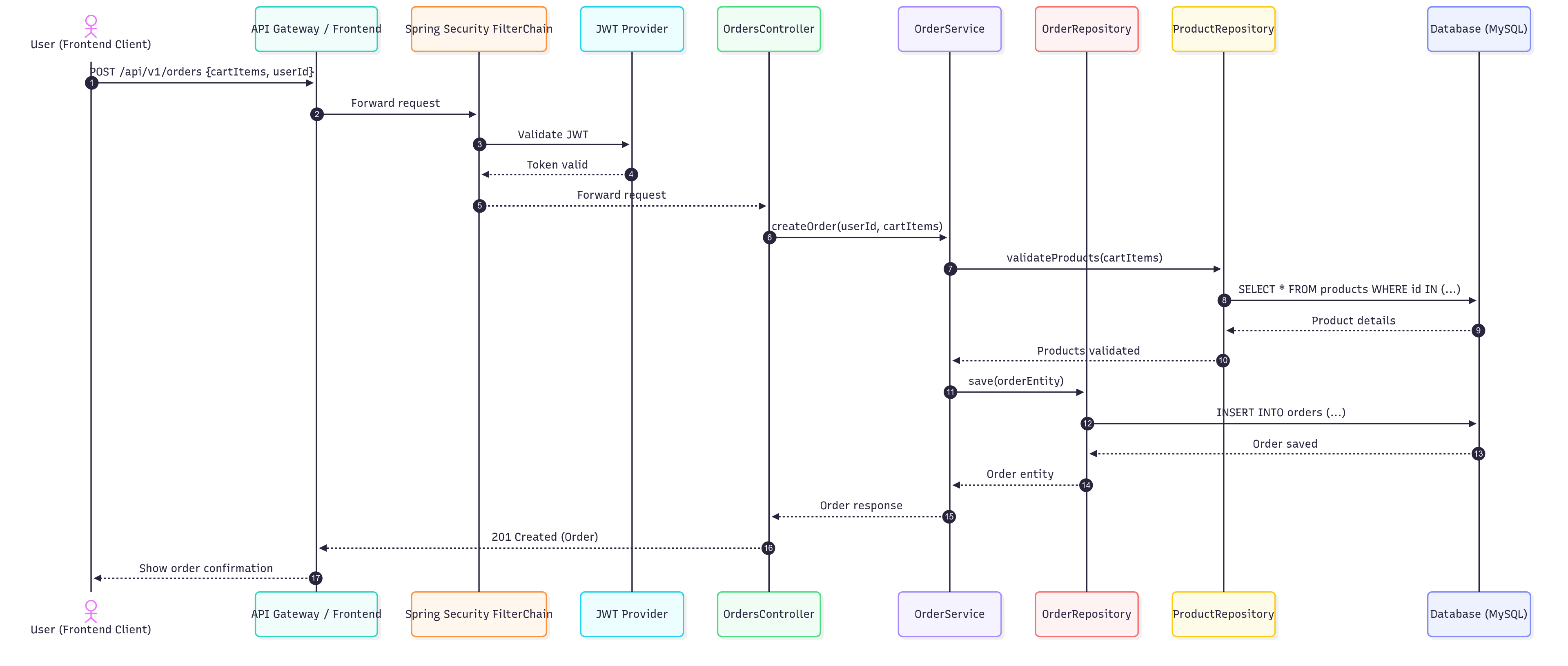


Figure Order Controller Flow

6. **Recommendation Engine Module (AI Microservice)**

**Purpose**: Suggests products to customers based on browsing history, purchase patterns, and categories.

**Logic Flow**:

* Frontend sends user ID or browsing context to backend.
* Backend calls **R microservice (via REST API)** with user data.
* Microservice applies recommendation algorithm (e.g., collaborative filtering, content-based filtering).
* Recommended product IDs are returned to backend.
* Backend fetches product details from database.
* Frontend displays “Recommended for You” section.

A diagram of a company

AI-generated content may be incorrect.

Figure Recommendation API Controller Flow

7. **Admin Dashboard Module**

**Purpose**: Provides admins with control over users, products, orders, and categories.

**Logic Flow**:

* Admin logs in (role-based authentication).
* Dashboard shows summary: total users, products, pending orders, sales reports.
* Admin can manage CRUD operations on products, categories, and users.
* Order management: update status, cancel/refund orders.
* Reports generation: Sales trends, most sold products, inventory levels.

A diagram of a company

AI-generated content may be incorrect.

Figure Admin User Flow

8. **Search and Filter Module**

**Purpose**: Enables customers to find products quickly.

**Logic Flow**:

* Customer enters keyword in search bar.
* Frontend sends query to backend API.
* Backend executes SQL LIKE query or full-text search.
* Results are sorted by relevance and sent back.
* Filters (price range, categories, popularity) refine results.

9. **Notification & Alert Module**

**Purpose**: Keeps users updated on order status and promotions.

**Logic Flow**:

* Backend triggers notification when an event occurs (order placed, shipped, delivered).
* Notification stored in MySQL and displayed on user dashboard.
* Email notifications may be sent (optional).
* Admin receives low-stock alerts.

## Methodology Adopted, System Implementation & Hardware/Software Used

The project is developed on a standard system using:

- Hardware: Standard Mac Air with 8 GB RAM, Apple M1 Chipset,   
- Backend Software: Java 21.0.5, Spring Boot v3.5.4, Spring v6.2.9, Apache Maven 3.9.11, Tomcat Server 10.1.43, MySQL,

- Frontend Software: Angular 20.2.1, on Node JS 22,17,0, typescript 5.8.3

- ML Engine: R 4.5.1, RStuio Postman, R Studio 2025.05.1,

- Database: MySQL 9.3.0

- IDE: 1.103.2, Mermaid & Figma

- Version Control: git version 2.48.1

## System Maintenance & Evaluation

The system is designed to be scalable, maintainable, and secure. Future improvements include advanced ML-based recommendations.

## Cost and Benefit Analysis

**Cost:** Development time, infrastructure (hosting, DB), and testing tools.  
**Benefit:** Real-world industry-level platform, reusable modules, customer engagement, and learning advanced technologies.

## Detailed Life Cycle of the Project

Requirements → Design → Development → Testing → Deployment → Maintenance

A diagram of software development life cycle

AI-generated content may be incorrect.

Figure SDLC Lifecycle

## ERD & DFD

Entity Relationship Diagram and Data Flow Diagrams describe the structure and data flow of the system.

A diagram of a computer

AI-generated content may be incorrect.

Figure ERD Diagram

## Input and Output Screen Design

Frontend screens include Login, Product Listing, Cart, Checkout, Admin Dashboard. Outputs include invoices, reports, and recommendations.

## Testing

Unit testing with JUnit, API testing with Postman, and manual user testing for validation.

## User/Operational Manual

Includes access rights, security measures, admin controls, backup procedures, and user guidelines.

# Annexures

## Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name** | **Column Name** | **Data Type** | **Description** |
| addresses | id | bigint | Primary key, unique identifier for the address. |
|  | city | varchar(50) | The city of the address. |
|  | country | varchar(50) | The country of the address. |
|  | created\_at | datetime(6) | Timestamp of when the address record was created. |
|  | postal\_code | varchar(20) | The postal code of the address. |
|  | state | varchar(50) | The state of the address. |
|  | street | varchar(100) | The street of the address. |
|  | updated\_at | datetime(6) | Timestamp of the last update to the address record. |
| cart\_items | id | bigint | Primary key, unique identifier for the cart item. |
|  | discount\_at\_addition | decimal(38,2) | The discount applied to the product at the time it was added to the cart. |
|  | price\_at\_addition | decimal(38,2) | The price of the product when it was added to the cart. |
|  | quantity | int | The number of units of the product in the cart. |
|  | total | decimal(38,2) | The total price for this item, including quantity and discount. |
|  | cart\_id | bigint | Foreign key referencing the carts table. |
|  | product\_id | bigint | Foreign key referencing the products table. |
| carts | id | bigint | Primary key, unique identifier for the cart. |
|  | created\_at | datetime(6) | Timestamp of when the cart was created. |
|  | total\_amount | decimal(38,2) | The total price of all items in the cart. |
|  | updated\_at | datetime(6) | Timestamp of the last update to the cart. |
|  | user\_id | bigint | Foreign key referencing the users table. |
| categories | id | bigint | Primary key, unique identifier for the category. |
|  | created\_at | datetime(6) | Timestamp of when the category was created. |
|  | description | varchar(500) | A description of the category. |
|  | name | varchar(100) | The name of the category. |
|  | updated\_at | datetime(6) | Timestamp of the last update to the category. |
| order\_items | id | bigint | Primary key, unique identifier for the order item. |
|  | discount | decimal(38,2) | The discount applied to the product at the time of purchase. |
|  | price | decimal(38,2) | The price of the product at the time of purchase. |
|  | quantity | int | The number of units of the product in the order. |
|  | total | decimal(38,2) | The total price for this item in the order. |
|  | order\_id | bigint | Foreign key referencing the orders table. |
|  | product\_id | bigint | Foreign key referencing the products table. |
| orders | id | bigint | Primary key, unique identifier for the order. |
|  | created\_at | datetime(6) | Timestamp of when the order record was created. |
|  | order\_date | datetime(6) | The date and time the order was placed. |
|  | status | enum | The current status of the order. |
|  | total\_amount | decimal(38,2) | The total cost of the order. |
|  | updated\_at | datetime(6) | Timestamp of the last update to the order record. |
|  | shipping\_address\_id | bigint | Foreign key referencing the addresses table for the shipping address. |
|  | user | bigint | Foreign key referencing the users table. |
| products | id | bigint | Primary key, unique identifier for the product. |
|  | created\_at | datetime(6) | Timestamp of when the product record was created. |
|  | description | varchar(500) | A detailed description of the product. |
|  | discount | decimal(38,2) | The discount percentage on the product. |
|  | image\_url | varchar(255) | The URL of the product's image. |
|  | name | varchar(100) | The name of the product. |
|  | price | decimal(38,2) | The price of the product. |
|  | stock\_quantity | int | The current stock level of the product. |
|  | updated\_at | datetime(6) | Timestamp of the last update to the product record. |
|  | category\_id | bigint | Foreign key referencing the categories table. |
| user\_interactions | id | bigint | Primary key, unique identifier for the interaction. |
|  | interaction\_type | enum | The type of interaction. |
|  | timestamp | datetime(6) | The date and time the interaction occurred. |
|  | product\_id | bigint | Foreign key referencing the products table. |
|  | user\_id | bigint | Foreign key referencing the users table. |
| users | id | bigint | Primary key, unique identifier for the user. |
|  | created\_at | datetime(6) | Timestamp of when the user account was created. |
|  | email | varchar(200) | The user's email address (unique). |
|  | password | varchar(255) | The user's hashed password. |
|  | phone\_number | varchar(15) | The user's phone number. |
|  | updated\_at | datetime(6) | Timestamp of the last update to the user account. |
|  | username | varchar(100) | The user's username (unique). |
|  | billing\_address\_id | bigint | Foreign key referencing the addresses table for the billing address. |
|  | shipping\_address\_id | bigint | Foreign key referencing the addresses table for the shipping address. |
|  | role | varchar(10) | The user's role. |
|  | firstname | varchar(100) | The user's first name. |
|  | lastname | varchar(100) | The user's last name. |

## List of Abbreviations

API: Application Programming Interface  
DBMS: Database Management System  
ERD: Entity Relationship Diagram  
DFD: Data Flow Diagram  
JWT: JSON Web Token

## References / Bibliography

1. Spring Boot Documentation - <https://spring.io/projects/spring-boot>
2. Java Documentation: <https://docs.oracle.com/en/java/javase/24/>
3. Node.JS Documentation: <https://nodejs.org/docs/latest/api/>
4. Angular Documentation - <https://angular.io/docs>
5. MySQL Documentation - <https://dev.mysql.com/doc/>
6. AI Recommendation Techniques
   1. *Recommender Systems: The Textbook.* Springer Author: Aggarwal, C. C.
7. R Language Documentation – <https://r4ds.hadley.nz/intro.html>
   1. Book: R for Data Science Author: Hadley Wickham, Mine Cetinkaya-Rundel & Garrett Grolemund
8. R Package recommenderlab Documentation - <https://cran.r-project.org/web/packages/recommenderlab/index.html>
9. R Package plumber Documentation: <https://www.rplumber.io/>
10. GitHub Documentation - <https://docs.github.com/>