**Applied Software Project Report**

By

Vinay Goel

**A Master’s Project Report submitted to Scaler Neovarsity - Woolf in partial fulfillment of the requirements for the degree of Master of Science in Computer Science**

March, 2025



**Scaler Mentee Email ID:** goel.vinay5@gmail.com

**Thesis Supervisor:** Naman Bhalla

**Date of Submission:** 15/03/2025

**Certification**

I confirm that I have overseen / reviewed this applied project and, in my judgment, it adheres to the appropriate standards of academic presentation. I believe it satisfactorily meets the criteria, in terms of both quality and breadth, to serve as an applied project report for the attainment of Master of Science in Computer Science degree. This applied project report has been submitted to Woolf and is deemed sufficient to fulfill the prerequisites for the Master of Science in Computer Science degree.

Naman Bhalla

…………………

Project Guide / Supervisor

**DECLARATION**

I confirm that this project report, submitted to fulfill the requirements for the Master of Science in Computer Science degree, completed by me from 15/01/2024 to 26/06/2024, is the result of my own individual endeavor. The Project has been made on my own under the guidance of my supervisor with proper acknowledgement and without plagiarism. Any contributions from external sources or individuals, including the use of AI tools, are appropriately acknowledged through citation. By making this declaration, I acknowledge that any violation of this statement constitutes academic misconduct. I understand that such misconduct may lead to expulsion from the program and/or disqualification from receiving the degree.

**Vinay Goel**

**Vinay Goel Date: 15 March 2025**

**ACKNOWLEDGMENT**

**I want to express gratitude to myself who helped, inspired or motivated me to complete the program and earn the Master’s degree**

**Table of Contents**

[**List of Tables 6**](#_9nnr2lniv90f)

[**List of Figures 7**](#_ju1gc9w3iuai)

[**Applied Software Project 8**](#_b4cf8683b1wd)

[Abstract 8](#_sj7c7bghlznr)

[Project Description 8](#_1z5fx61h0cc)

[Requirement Gathering 9](#_joagy45av5k0)

[Class Diagrams 9](#_nvf4h831fm8o)

[Database Schema Design 9](#_ydqs8nkbe6m9)

[Feature Development Process 11](#_p6mfl8dwb9sy)

[Deployment Flow 12](#_2mk44ad33gi)

[Technologies Used 12](#_wn68bn10ag78)

[Conclusion 13](#_4yf46wt6rx84)

[**References 14**](#_z0iyzog9l959)

## List of Tables

**(To be written sequentially as they appear in the text)**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Title** | **Page No.** |
| **1** |  |  |
| **2** |  |  |

## List of Figures

**(List of Images, Graphs, Charts sequentially as they appear in the text)**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| **1** |  |  |
| **2** |  |  |

## Applied Software Project

### Abstract

### This project describes the design and implementation of a scalable and high-performance e-commerce platform based on a microservices architecture. The main goal is to give users an uninterrupted online shopping experience with high availability, security, and best performance. The platform includes key e-commerce features such as user management, browsing the product catalogue, shopping cart management, order processing, and secure payment processing.

### In order to improve system scalability and efficiency, the architecture utilizes new cloud-based technologies like load balancers for request distribution, an API Gateway for routing, and Kafka for asynchronous messaging. The backend services make use of relational (MySQL) and NoSQL (MongoDB) databases to store structured data and flexible data, respectively. Elasticsearch is incorporated in the product catalogue service to support real-time search functionality with features such as typo correction. Also, Redis caching preloads highly accessed data to minimize latency when retrieving shopping carts.

### This system is architecturally designed to sustain high traffic loads and deliver a stable user experience via secure authentication, effective session management, and live order tracking. Independent scaling of services is made possible by the microservices-based model, making modifications and future development easy. Integrating contemporary software engineering practices and distributed computing concepts, this project helps shape the field of scalable and reliable e-commerce solutions. The suggested architecture can be utilized across sectors in order to enhance online retail businesses, streamline order fulfilment processes, and increase customer interaction through targeted shopping experiences.

### Project Description

* 1. Introduction

Digital commerce has revolutionized the interaction between consumers and businesses, with e-commerce sites becoming a core component of contemporary trade. The goal of this project is to create a scalable and feature-loaded e-commerce website that facilitates effortless online transactions. The site will include features like user authentication, browsing of a product catalogue, management of shopping carts, order placement, and secure payment.

* 1. Objectives

The major objectives of the project are:

* To develop and deploy a secure multi-user e-commerce platform.
* To achieve a secure and scalable microservices architecture.
* To include search and filtering functionalities for improved shopping.
* To implement an efficient order management process, including notifications and tracking.
* To make database performance efficient through Redis caching and Elasticsearch search indexing.
* To enable secure payments through payment gateway integration.
  1. Project Flow

The development process is in accordance with agile methodologies with iterative enhancements. Following is the development life cycle:

+-

**Figure 2.3.1**: Project Development Process

* 1. System Architecture

The system is based on a microservices architecture to maintain modularity and scalability. Following is a simplified diagram of the system architecture.

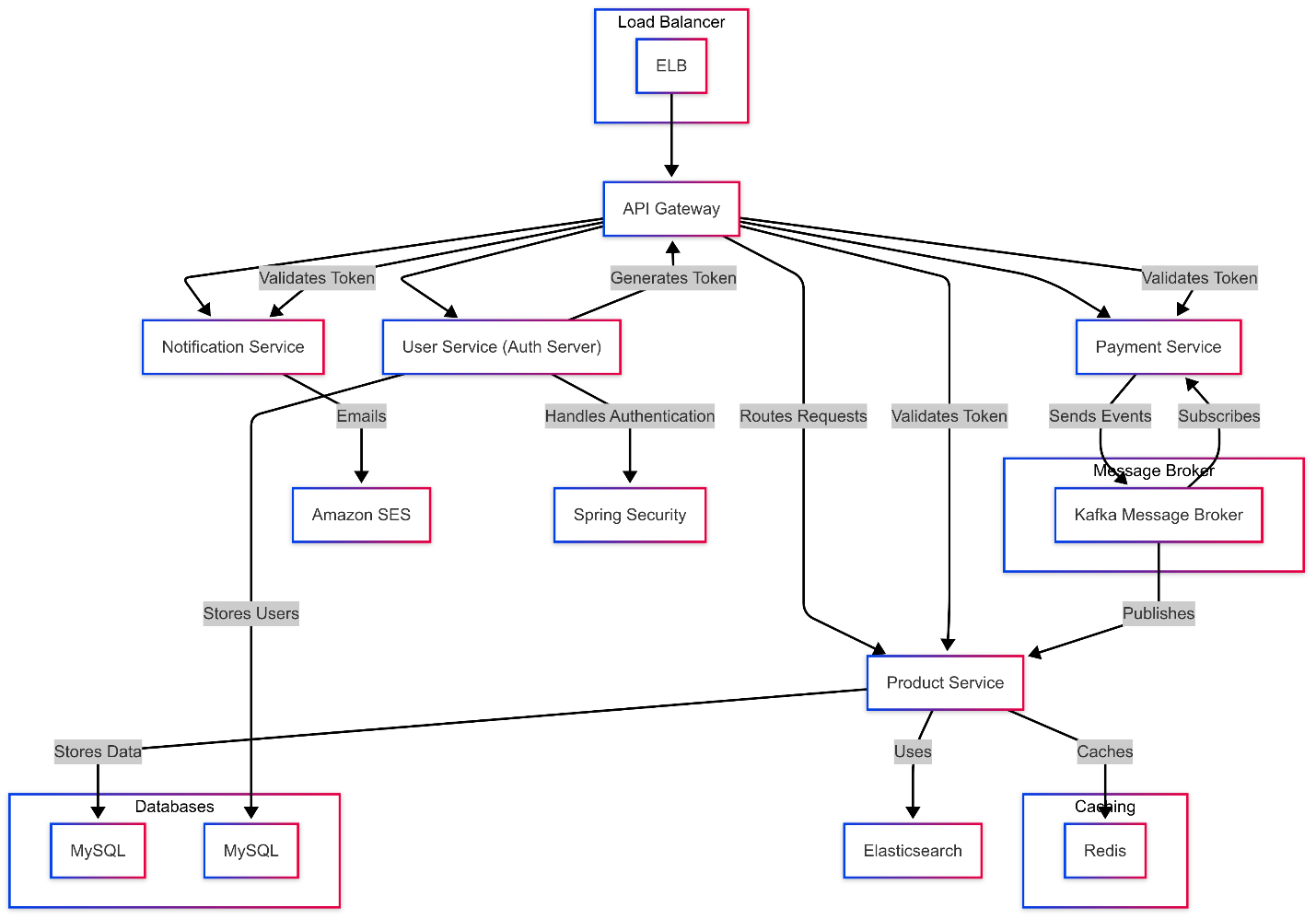


Figure 2.4.2: E-commerce System Architecture

* 1. Relevance and Real-world Applications

This e-commerce platform is applicable to businesses that want an online presence and an automated order processing system. It is beneficial:

* Retail companies: Allowing global access and efficient sales.
* Customers: Providing a seamless, tailored shopping experience.
* Logistics: Enabling order tracking and fulfilment.
* Data Analytics: Capturing customer insights through tracking user behaviour.
  1. Conclusion

The project will transform e-commerce by creating an effective, secure, and easy-to-use shopping platform. The microservices architecture allows scalability, which enables it to accommodate future development. Through the application of current cloud-based technologies, this project will make a large contribution to internet retailing, logistics, and customer experience optimization.

### Requirement Gathering

* 1. Functional Requirements
     1. User Management

**User Registration**

* The system shall allow new users to register using **email**
* User account information should be securely stored following **industry-standard encryption**.

**Secure Login**

* The system shall support **secure authentication** via email-password credentials

**Profile Management**

* Users shall have the ability to **view, update, and manage** their personal information, including email, name, and contact details.
* The system shall enforce **input validation and verification mechanisms** to prevent unauthorized modifications.

**Password Management**

* Users must be able to **reset their passwords securely**.
* The system shall enforce **strong password policies** (e.g., minimum length, special characters).
  + 1. Product Catalogue

**Product Browsing**:

* Users should have the ability to explore products across various categories for easy discovery.

**Product Details**:

* Each product page should display essential details, including images, descriptions, specifications, and other relevant attributes.

**Product Search**:

* Users should be able to search products using keywords.
  + 1. Cart & Checkout

**Add to Cart:**

* Users should be able to select and add products to their shopping cart for future purchases.

**Cart Review:**

* Users should have the ability to review selected items, including quantity, price breakdown, and total cost before proceeding to checkout.

**Checkout**:

* The platform should provide a seamless checkout experience, allowing users to specify delivery details and select preferred payment methods.

* + 1. Order Management

**Order Confirmation:**

* After making a purchase, users should receive a confirmation with order details.

**Order History:**

* Users should be able to view their past orders.

**Order Tracking:**

* Provide users with a way to track their order's delivery status.
  + 1. Payment

**Multiple Payment Options:**

* The system should support multiple payment methods, including credit/debit cards, online banking, and other widely used payment gateways.

**Secure Payment Processing:**

* Transactions should be processed securely, ensuring the confidentiality and integrity of users' financial data.

**Payment Receipt:**

* After a successful transaction, users should receive a digital receipt confirming their payment.
  + 1. Authentication

**Secure Authentication:**

* Ensure that user data remains private and secure during login and throughout their session.

**Session Management:**

* Users should remain logged in for a specified duration or until they decide to log out.
  1. Non-Functional Requirements

**Security**

* Implement **Spring Security** for authentication and authorization.
* Encrypt sensitive data such as passwords using **BCrypt hashing**.
* Ensure secure authentication mechanisms, including **OAuth2 and JWT-based authorization**.
* Implement **role-based access control (RBAC)** to restrict unauthorized access to sensitive user data.
* Apply **rate limiting** to prevent brute-force login attacks.
* Apply **IP whitelisting** to prevent unauthorized access.
* Prevent **SQL injection, XSS, and CSRF** attacks.
* Payment data must be encrypted and comply with **PCI-DSS** security standards.

**Performance**

* Ensure the services can handle **at least 1000 concurrent user requests** with minimal latency.
* Optimize database queries and use **caching (e.g., Redis)** for frequently accessed user data.
* Implement asynchronous processing where necessary using **messaging queues such as** **Kafka or RabbitMQ** to prevent blocking operations.
* Product service must support **full-text search** and filtering using **Elasticsearch**.

**Scalability**

* Deploy in a **containerized environment (Docker, Kubernetes)** for auto-scaling capabilities.

**Availability & Reliability**

* Maintain **99.9% uptime** with load-balanced, multi-instance deployment.
* Implement **circuit breakers** and **failover mechanisms** to handle service failures gracefully.
* Provide **automated recovery mechanisms** for unexpected crashes.
* Order and Payment Service must ensure **ACID compliance** for all financial transactions using a reliable RDBMS
* Implement **idempotency mechanisms** to prevent duplicate order processing.

**Logging**

* Enable **logging** for tracking user authentication and activity.

**Maintainability**

* Implement **automated unit and integration tests** to ensure system stability.
* Maintain **well-documented API specifications** using OpenAPI/Swagger.
* Unit and integration tests should achieve at least **80% code coverage**.
* Implement CI/CD pipelines using GitHub Actions/Jenkins for automated deployments.
* Code should adhere to industry best practices such as **SOLID principles and Design Patterns.**
* All microservices must be deployed using Kubernetes (K8s) and Docker.
* Services should support multi-region deployment for disaster recovery and high availability.
* Database backups must be automated daily with retention for 30 days.

**Monitoring**

* Use **structured logging (ELK Stack, Prometheus, or Grafana)** for system monitoring.
* Services must expose health check endpoints (/health, /metrics) for monitoring via **Prometheus**.
  1. Use Case Diagram

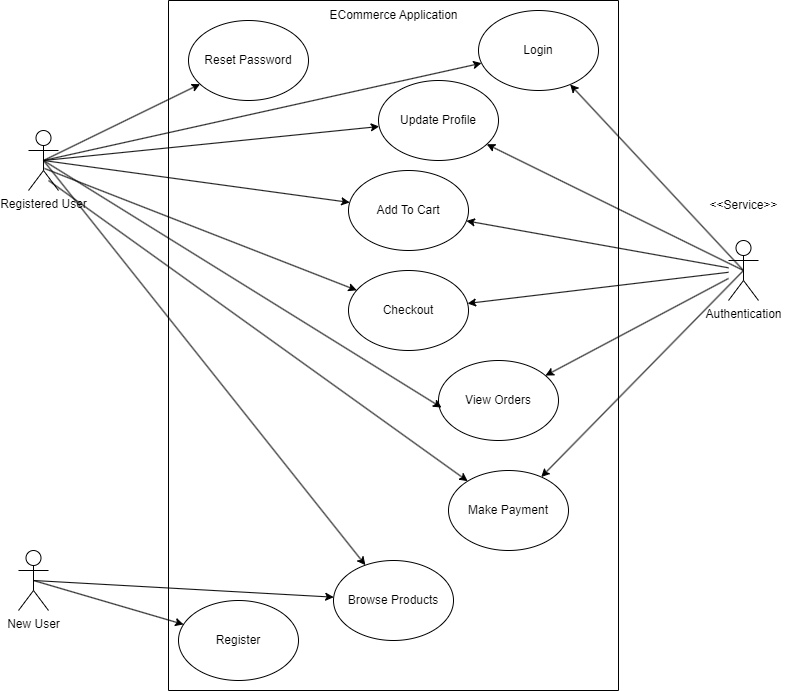


Figure 3.1.1: E-commerce Use Case Diagram

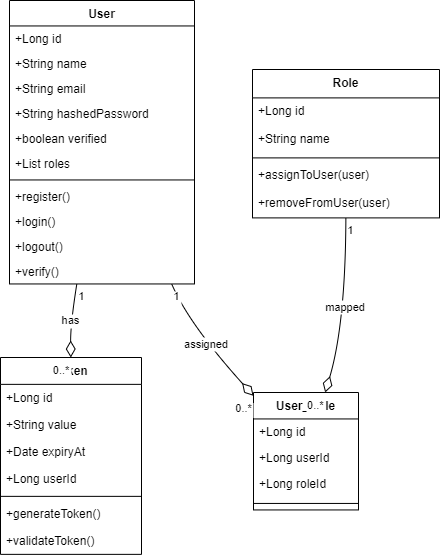
* 1. Features

**Table 3.4:** Features Of E-Commerce Application

|  |  |  |
| --- | --- | --- |
| **Service** | **Feature** | **Description** |
| **User Management** | **User Registration** | Allows new users to create an account using their email or social media profiles.  POST /register: Register a new user |
|  | **User Login** | Enables users to securely log in using their credentials.  POST /login: Log in the user |
|  | **Profile Management** | Users can view, update, and manage their personal details.  GET /profile: Retrieve user profile  PUT /profile: Update profile |
|  | **Password Reset** | Provides users with a secure way to reset their password via email verification. |
| **Product Catalogue** | **Product Browsing** | Enables users to browse products across different categories.  GET /products: Get products by category |
|  | **Product Details** | Displays product images, descriptions, specifications, and relevant details.  GET /products/{id}: Get detailed view of a product |
|  | **Product Search** | Allows users to search for products using keywords.  GET /search?query=keyword: Full-text search on product catalog |
| **Cart & Checkout** | **Add to Cart** | Users can add selected products to their cart for future purchase.  POST /cart: Add item to cart |
|  | **Cart Review** | Users can view their selected items, adjust quantities, and check the total cost.  GET /cart: Retrieve cart contents  DELETE /cart/{itemId}: Remove item from cart |
|  | **Checkout Process** | Facilitates a seamless checkout experience, including delivery and payment selection. |
| **Order Management** | **Order Confirmation** | Users receive an order confirmation with all relevant details.  POST /orders: Place an order |
|  | **Order History** | Users can view a list of their past purchases.  GET /orders: View order history |
|  | **Order Tracking** | Provides real-time tracking updates for orders in transit.  GET /orders/{id}: Get order status |
| **Payment** | **Multiple Payment Methods** | Supports credit/debit cards, online banking, and other payment gateways. |
|  | **Secure Transactions** | Ensures encrypted and secure payment processing.  POST /payment: Process payment |
|  | **Payment Receipt** | Generates digital receipts after successful transactions. |
| **Authentication** | **User Authentication** | Implements secure login mechanisms to protect user data. |
|  | **Session Management** | Manages user sessions with defined expiration and logout options. |

### Class Diagrams

User Management



### Database Schema Design

* 1. Tables

User Management

User

* + Id
  + Name
  + Email (Unique)
  + hashedPassword
  + roles
  + verified
  + Primary Key (Id)

Role

* + Id
  + Name
  + Primary Key (Id)

Token

* Id
* Value
* User\_Id
* ExpiryAt
* Primary Key (Id)

User\_Role

* Id
* User\_id
* Role\_Id
  1. Foreign Keys

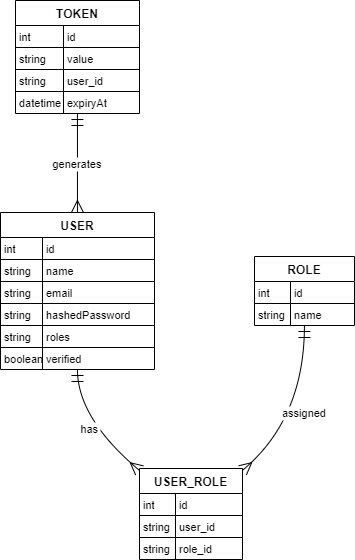
User Management

* Token(User\_Id) refers User(Id)
* User\_Role(User\_Id) refers Users(Id)
* User\_Role(Role\_Id) refers Role(Id)
  1. Cardinality of Relations

User Management

* Between Users and Roles -> m:m
* Between Users and Tokens -> 1:m
  1. Diagrams

User Management



### Feature Development Process

Pick One key feature - Talk about its development process, implementation and performance optimisation / metric optimisation achieved…

For example, ‘Book a seat’ feature in developing ‘BookMyShow’ app

Elaborate the request flow to backend

* 1. API Request Payload
  2. Service which picks the request
  3. Flow of MVC architecture

Explain the performance improvement / metric optimization achieved.

For example,

* Used Cache to reduce API Response time by X seconds…
* Optimized Query Response time by using Indexing…

Benchmarking of response time without the optimisation and post the optimisation

### Deployment Flow

Explain how the deployment will work via AWS (Describe the below) -

* EC2
* VPC
* Security Groups
* RDS
* Cache
* Managed Infra / Elastic Beanstalk

Use diagrams, images to explain better

### Technologies Used

Kafka, MySQL, Springboot, Cloud etc…

* For each key technology used in building the project,
  + Detail and describe each of them
  + Elaborate how they can be used in real life
  + Provide example of real-life applications using them

Use diagrams, images to explain better

**Tip** - Use the internet to improve your project but DO NOT PLAGIARIZE - Include proper references if you are quoting articles from the internet

### Conclusion

The Conclusion should include some key points as elaborated below -

* Key Takeaways: Highlight the important concepts and technologies learned from doing the Project
* Practical Applications : Significance of technologies with their real-world applications
* Limitations : Limitations of the technologies, cost implications and suggestions for improvement

## References

Include the websites or works or the list of works referred to in a text or consulted by you for writing this report

1. Name of the Website, Date and time of referring to the Website, Name of the Author, Title/Topic
2. Author Name, Title / Topic, Research Paper Name / Book Name, Year of Publication

Format Guidelines

1. Detailed and Elaborate report of 40 pages at least is expected
2. Margins - Every page of your document must meet the margin requirements of 1.25 inches on the left and right, and 1 inch on the top and bottom.
3. Font:
   1. Style: Times New Roman,
   2. Font Size:14 (For Headings), 12 (For body text) in black colored text.
   3. All text must be the same justification, like left justified or fully justified.
4. Line Spacing:
   1. Body of the text: 1.5
   2. List of Tables/graphs/charts/bibliography: single line.
5. Alignment:
   1. Title page: Centre
   2. Chapter Heading: Centre
   3. Subheading: Left
   4. Body of the text: Justify
6. Titles: All titles and subtitles should be in bold. All tables/graphs/charts/figures should have appropriate titles.
7. Numbering of the tables, charts, graphs should be in the following fashion: Second table/graph/chart in the second chapter should be numbered as Table/graph/chart no. 2.02; where the first digit stands for chapter no. and digits after (.) stands for number of table/graph/charts in that chapter. Same numbering should be followed for all other chapters.