TRIBHUVAN UNIVERSITY

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Kritipur, Kathmandu, Nepal

Object-Oriented Software Engineering
Case Study on:
E-Commerce System (Online Book Store)

Submitted By:
Priya Shrestha
Roll No: 24
MSc.CSIT First Semester

Submitted To:
Prof. Dr. Subarna Shakya
TU, CDCSIT

Date: 2082/02/30

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who contributed to the successful completion of this case study.

First and foremost, I extend my heartfelt thanks to **Prof. Dr. Subarna Shakya** for her invaluable guidance, encouragement, and insightful feedback throughout the development of this case study.

I am also deeply grateful to the **Head of Department, Sarbin Sayami**, for his continuous support and leadership, which greatly facilitated the progress of this work.

My sincere appreciation goes to my **peers and classmates** for their constructive suggestions, collaborative spirit, and encouragement during various stages of this case study.

Lastly, I would like to acknowledge the various online resources and research materials that provided a strong foundation for understanding Object-Oriented Software Engineering principles and their practical application in developing an e-commerce system.

ABSTRACT

The Online Bookstore System, through its integration of a microservices-ready architecture and a rich frontend experience, delivers a cohesive platform for browsing, purchasing, and managing books. By modularizing core functions—catalog management, user authentication, shopping cart operations, order processing, payment handling, and review management—and leveraging a Spring Boot backend with a React frontend, the system ensures scalability, maintainability, and responsive performance. [1] A robust RESTful API layer and relational database schema facilitate real-time search and secure transaction processing, while integrated analytics on user behavior and purchase history enable personalized recommendations and inventory forecasting. Client-side tokenization and TLS/AES encryption enforce stringent security and privacy standards, empowering users with trust and control over their data. Overall, this system not only enhances customer engagement through intuitive interfaces and tailored suggestions but also streamlines administrative workflows and data-driven decision-making for sustainable growth.

Keywords: Online Bookstore System, Microservices Architecture, Spring Boot, React, RESTful API, Secure Payments, Inventory Forecasting.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

The **Online Book Store System** is a comprehensive software application designed to streamline the process of browsing, purchasing, and managing books for customers. This system offers essential functionalities, including searching for books, adding books to the cart, completing secure purchases, and managing customer accounts. Additionally, the system provides administrators with tools to manage inventory, track sales, update book listings, and organize promotions. This report, based on Object-Oriented Analysis and Design (OOAD) principles, provides a detailed overview of the system's requirements and design considerations, aiming to offer an efficient, user-friendly solution for both customers and store administrators. [3]

In today's fast-evolving digital marketplace, where online book shopping has become increasingly popular, an effective and reliable e-commerce platform is essential. Such a system not only ensures a smooth, convenient purchasing experience for book buyers but also optimizes operational tasks for administrators. The **Online Book Store System** aims to provide a dynamic and secure environment for browsing and buying books while offering robust management capabilities. This report employs Unified Modeling Language (UML) diagrams to present a thorough analysis and design of the Online Book Store System, ensuring a clear, well-structured approach to meeting the needs of both customers and administrators.

1.1 Problem Domain

- ❖ In current era, the hassles of moving from bookshop to the other and not finding the books we need is frustrating and time/resources consuming.
- ❖ Moving towards automation, the manual record keeping process is a hectic one which sometimes results in data loss.
- ❖ This system solves the issues of having to waste time and other precious resources in search of a book via online platform.
- ❖ This system also organizes events such as book reading, book signing, book clubs, etc., which enhances the reading experiences.

1.2 Scope

- ❖ The project will cover essential features of an online bookstore, including user registration and login, book browsing and search, shopping cart management, order placement, and product shipping.
- ❖ Advanced functionalities such as online payment integration, email marketing, multi-store management, promotion handling, and multi-channel support are beyond the scope of this study and will not be implemented.

CHAPTER 2: OBJECTIVE OF CASE STUDY

The primary objective of this case study is to apply Object-Oriented Analysis and Design (OOAD) principles and Unified Modeling Language (UML) diagrams to comprehensively model and analyze the Online Bookstore System. This includes:

- ❖ To provide users with access to a comprehensive collection of over 1,000 books from any location at any time.
- ❖ To ensure efficient data processing, minimizing response time and enhancing user experience.
- ❖ To offer a platform where users can express their opinions through reviews and ratings.
- ❖ To safeguard user information by preventing unauthorized access and ensuring data security.
- ❖ To guarantee immediate availability of requested information to users upon demand.
- ❖ To implement a simple, reliable, and secure payment integration that enables users to purchase or access books with ease.

CHAPTER 3: DESCRIPTION ABOUT YOUY CASE STUDY

The case study focuses on an **Online Bookstore System**, which is a critical software application in the domain of digital commerce. This system is designed to facilitate the browsing, selection, and purchasing of books for customers through an online interface. It encompasses a wide array of functionalities, including:

- ❖ **User Registration and Login:** Allowing users to create accounts and access personalized features securely.
- ❖ **Book Browsing and Search:** Enabling customers to explore books based on categories such as genre, title, author, or keywords.
- ❖ **Shopping Cart Management:** Allowing users to add, update, or remove books before completing their purchases.
- ❖ **Order Placement:** Facilitating the process of confirming and placing book orders based on user selections.
- ❖ **Shipping Management:** Handling the logistics related to packaging and delivery of ordered books to the customer.
- ❖ **Wishlist Feature:** Enabling users to mark books for future purchase consideration.
- ❖ **Inventory Management:** Managing book stock levels, availability status, and catalog updates in real time.

The system caters to diverse user roles, including:

- ❖ **Customers:** Who use the system to browse, search for, and purchase books online.
- ❖ **Administrators:** Who manage book inventory, update product information, and oversee orders and logistics.
- ❖ **Support Agents:** Who may assist customers with inquiries, order issues, or account-related concerns.

The system is designed to manage complex interactions between various components such as users, book records, orders, and logistics workflows, while ensuring **data accuracy, security, and scalability**. It aims to provide a smooth, efficient, and user-friendly platform that enhances the experience of purchasing books online and streamlines bookstore operations.

CHAPTER 4: THE REQUIREMENT MODEL

4.1 Requirement Analysis

The **Online Bookstore System (OBS)** is a web-based application designed to provide users with a convenient and intuitive platform for purchasing books online. Its primary goal is to ensure a smooth, secure, and efficient transaction process between customers and the bookstore.
OBS offers a wide range of features, including user registration, login, book browsing, Wishlist creation, shopping cart functionality, and a secure checkout process. These features work together to deliver a complete and user-friendly online shopping experience tailored specifically to book lovers.
The requirements for the Online Bookstore System are categorized into Functional and Non-Functional requirements.

4.1.1 Functional Requirements:

For Customer

- ❖ **Login and Register:** Users must be able to register for an account, log in securely using their credentials, and manage their profile information (e.g., name, email, password).
- ❖ **Explore Categories**: Customers must be able to browse various book categories (e.g., fiction, non-fiction, academic) to discover books of interest.
- ❖ **Search Books**: Users must be able to search for books by title, author, category, or keyword to quickly locate specific items in the catalog.
- ❖ **Check Review**: Customers must be able to view ratings and reviews posted by other users to help make informed purchasing decisions.
- ❖ **Add to Wishlist:** Customers must be able to save books to a Wishlist for potential future purchases without needing to add them to the cart immediately.
- ❖ **Add to Cart**: Users must be able to add selected books to a shopping cart for purchase, with options to modify quantities or remove items before checkout.
- ❖ **Checkout Cart:** Users must be able to proceed to checkout, where they confirm selected items, provide delivery details, and prepare for payment.
- ❖ **Payment:** The system must securely handle payment processing using various methods (e.g., credit/debit cards, UPI, or online wallets) and confirm the order once payment is successful.
- ❖ **Logout**: Customers must be able to securely log out of their accounts to protect their personal and transactional data.

For Admin

- ❖ Login
- ❖ Add, update, or remove product (book) details.
- ❖ Manage Customers accounts and user data.
- ❖ View and manage customer orders (order status, payment verification, delivery updates).

4.1.2 Non-Functional Requirements:

These define the quality attributes of the system:

- ❖ **Performance**: The system must efficiently handle multiple users browsing, searching, and purchasing books simultaneously without noticeable delays.
- ❖ **Reliability**: The system must operate reliably with minimal downtime and ensure accurate order processing and data integrity.
- ❖ **Security**: The system must enforce secure login, data encryption, and protect sensitive user and payment information against unauthorized access.
- ❖ **Scalability**: The system must be able to scale to support a growing number of users, products, and transactions as the business expands.
- ❖ **Usability**: The system must offer an intuitive, easy-to-use interface for both customers and administrators, accessible across devices and browsers.

CHAPTER 5: THE ANALYSIS MODEL

The analysis model focuses on understanding the system's functionalities from the user's perspective, primarily through **Use Case Diagrams**.

5.1 Introduction to Use Case Diagrams

Use case diagrams visually represent the system's functionality and interactions between users (actors) and the system. They provide a high-level view of the system's behavior from an external perspective, facilitating communication among stakeholders, developers, and designers.

5.2 Component of Use Case Diagrams

- ❖ **Actors**: Entities interacting with the system (users, external systems). Depicted as stick figures outside the system boundary.
- ❖ **Use Cases**: Specific functionalities or features of the system from a user's perspective. Depicted as ovals within the system boundary.
- ❖ **Relationships**: Connections between actors and use cases, showing who performs what.

5.3 Role of Use Case Diagrams in OOAD

Use case diagrams serve as blueprints for system behavior, guiding requirements analysis and implementation. They help in eliciting, organizing, and prioritizing requirements, defining system boundaries, and validating functionalities.

5.4 Actors in Online Bookstore System

● **Customer (Primary Actor):** Interacts with the system to browse, purchase, and manage books.

- ❖ **Regular Customer**
- ❖ **Premium Member**

Privileges may vary based on membership level (e.g., discounts, early access to sales).

● **Admin (Secondary Actor):** Oversees and manages the entire bookstore system.

- ❖ Adds/updates/removes books
- ❖ Manages users and categories
- ❖ Handles orders and inventory

● **Delivery Staff (Secondary Actor):** Handles logistics and delivery processes.

- Views assigned orders
- Updates delivery status (e.g., shipped, delivered)

5.5 Scenarios

- ❖ Users register, log in, and manage their profile to access personalized features of the bookstore.
- ❖ Customers explore book categories or use the search feature to find books by title, author, or keyword.
- ❖ Users add books to their Wishlist or shopping cart, update quantities, and prepare for purchase.
- ❖ Customers proceed to checkout, enter delivery details, complete secure payments, and receive order confirmations.
- ❖ Admins log in to manage books, monitor customer activity, handle orders, and update catalog or system data.

5.6 Use Case Diagram

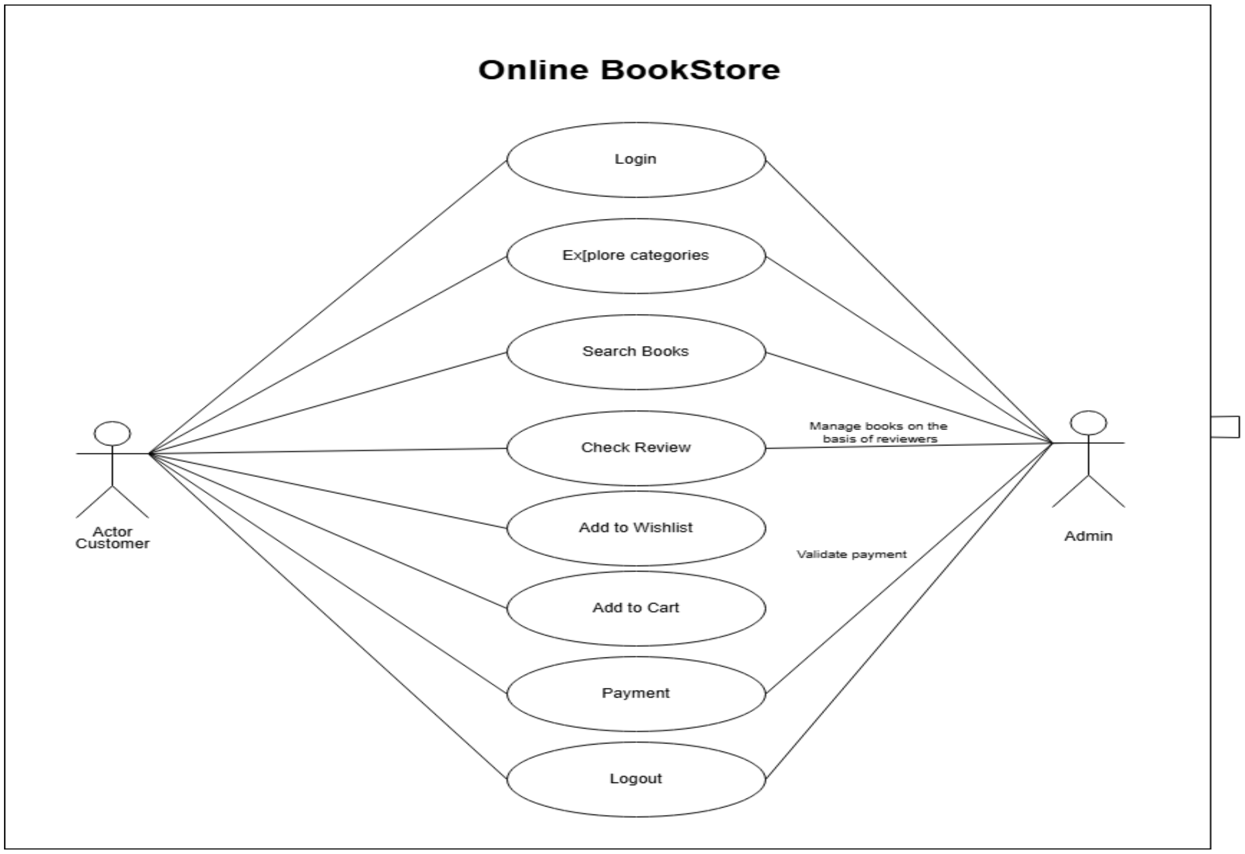

Figure1: Usecase diagram

6. THE DESIGN MODEL

The design model details the system's structure and behavior using various UML diagrams, including Class Diagrams, Activity Diagrams, Sequence Diagrams, and State Machine Diagrams.

6.1 Class Diagram

6.1.1 Introduction to Class Diagrams

Class diagrams are foundational elements in OOAD, providing a visual representation of a system's static structure. They show the system's entities, their attributes, relationships, and behaviors, serving as blueprints for software systems. [2]

6.1.2 Significance of Class Diagrams

They define the building blocks and their interconnections, giving insights into the system's architecture, and facilitating communication and code generation.

6.1.3 Components of Class Diagrams

- ❖ **Classes**: Fundamental entities encapsulating data (attributes) and behaviors (methods). ● Attributes: Properties or characteristics of a class.
- ❖ **Methods**: Operations or behaviors objects can perform.
- ❖ **Relationships**: Connections between classes (associations, aggregations, compositions, generalizations, dependencies).
- ❖ **Multiplicity**: Specifies the cardinality of relationships (e.g., one-to-one, one-to-many).

6.1.4 Methodologies for creating Class Diagrams

- ❖ **Requirements Analysis**: Identifying entities, attributes, and behaviors from system requirements.
- ❖ **Domain Modeling**: Mapping real-world concepts to classes.
- ❖ **Incremental Refinement**: Iterative evolution based on feedback and changing requirements.

6.1.4 Class Diagram Representation

The class diagram for the online bookstore system illustrates user functionalities such as registration, login, browsing books, adding books to the cart, placing orders, reviewing books, and managing payments. The diagram consists of classes such as **User**, **Book**, **Order**, **Cart**, **Payment**, **Review**, and **Admin**, demonstrating the relationships between users, books, orders, payments, and system-generated reviews aimed at enhancing the online shopping experience.

The class diagram represents:

1. **Generalization:** Between User (super class) and customer/Admin (child classes).
2. **Simple Associations**: Between Customer and Cart class, Customer and Order class and Order and Book class.
3. **Aggregation**: Between Cart and Book class and Order and Book class.
4. **Dependencies**:

- ❖ Payment class depends on Order class.
- ❖ Cart class depends on Book class.
- ❖ Order class depends on Payment class.

5. **Multiplicity**:
   - ❖ * means many.
   - ❖ 1 means one

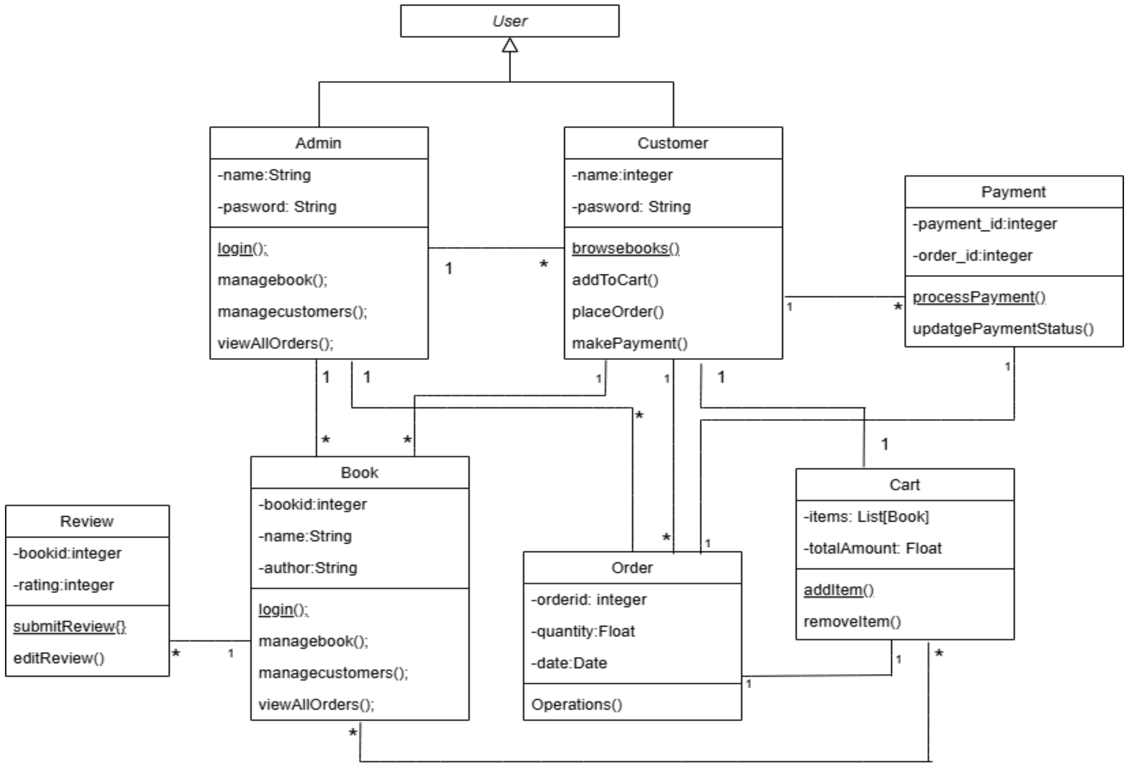

Figure2: Class Diagram

6.2 Activity Diagram

6.2.1 Introduction to Activity Diagrams

Activity diagrams are UML flowcharts depicting the flow of activities and actions within a system or process. They are "behavior diagrams" illustrating what should happen in the modeled system, often used for business process modeling or describing steps of a use case.

This activity diagram captures the core customer journey from browsing books to purchasing and optionally reviewing them. The flow follows a logical sequence: browsing, adding to the cart, ordering, paying, and finally, writing a review if the customer desires.

6.2.2 Activity Diagram for Online Bookstore System

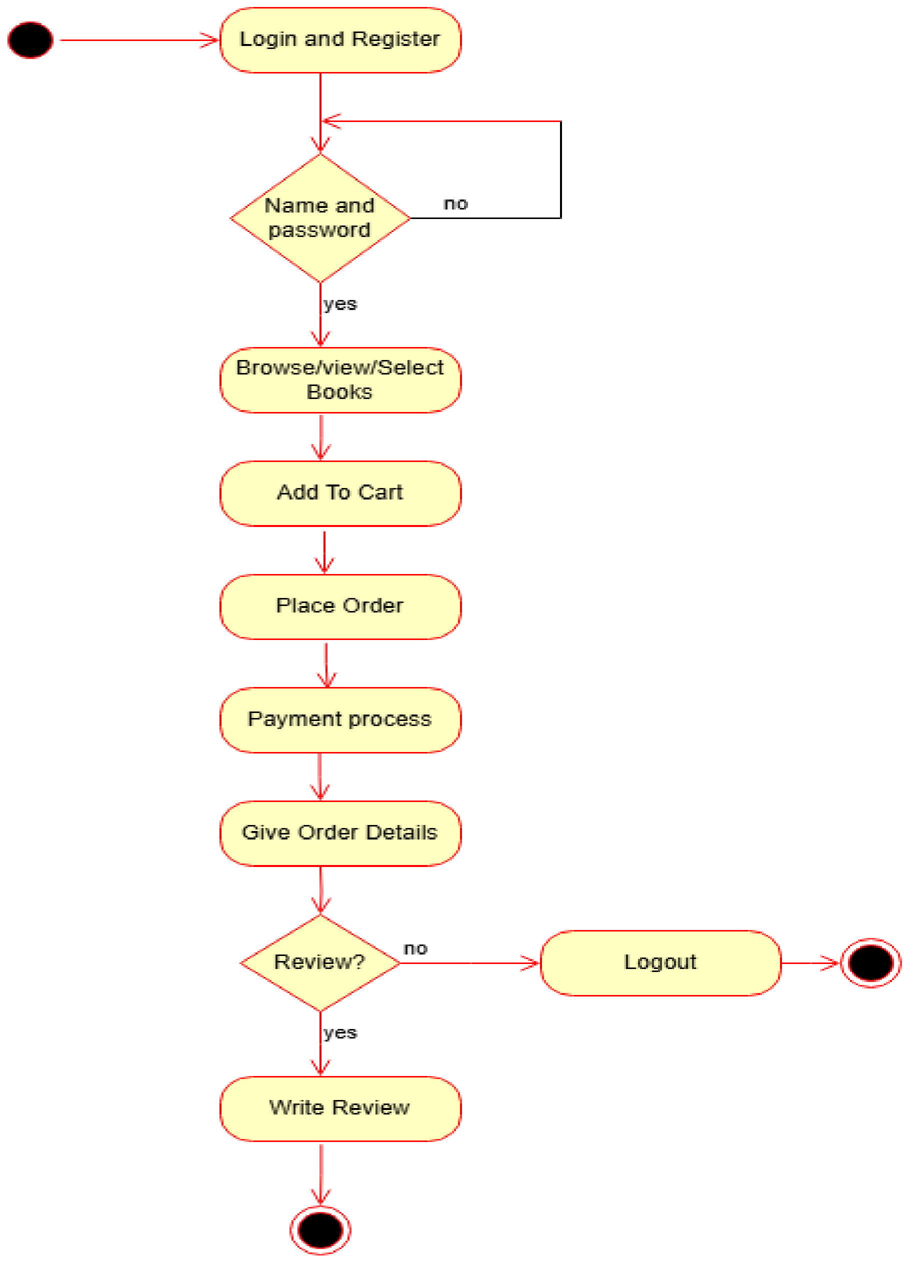

Figure3: Activity Diagram

6.3 Sequence Diagram

6.3.1 Introduction to Sequence Diagrams

Sequence diagrams are UML diagrams used to visually represent the interactions and flow of messages between objects or components within a system over time. They focus on depicting the chronological sequence of interactions, showing how objects collaborate to accomplish specific tasks.

6.3.2 Key Elements and Characteristics

- **Lifelines**: Represent participating objects or components.
- **Messages:** Depict communication between objects (synchronous, asynchronous, self-referential).
- Used by software developers and business professionals to understand requirements.
- Includes Actors, Lifelines, Activations, Call messages, Return messages, Self messages, etc.

6.3.3 Sequence Diagram Representation

In the **Online Bookstore System**, a **user** first registers and logs into the system. After logging in, the user browses through the available books across various categories (e.g., fiction, non-fiction, science, etc.). The system displays the books and their details, such as title, price, and availability. The user then selects books to add to their shopping cart. Once the books are added, the user proceeds to checkout, where they enter their **payment details** and the system processes the transaction through a **payment gateway**. Upon successful payment, the system confirms the order and updates the **inventory** to reflect the purchase. Additionally, the user has the option to generate **reports** related to their purchase history, including total spending, most purchased genres, or order status.

The system's sequence diagram includes:

- 5 Lifelines
- 16 activations
- Many call and return messages

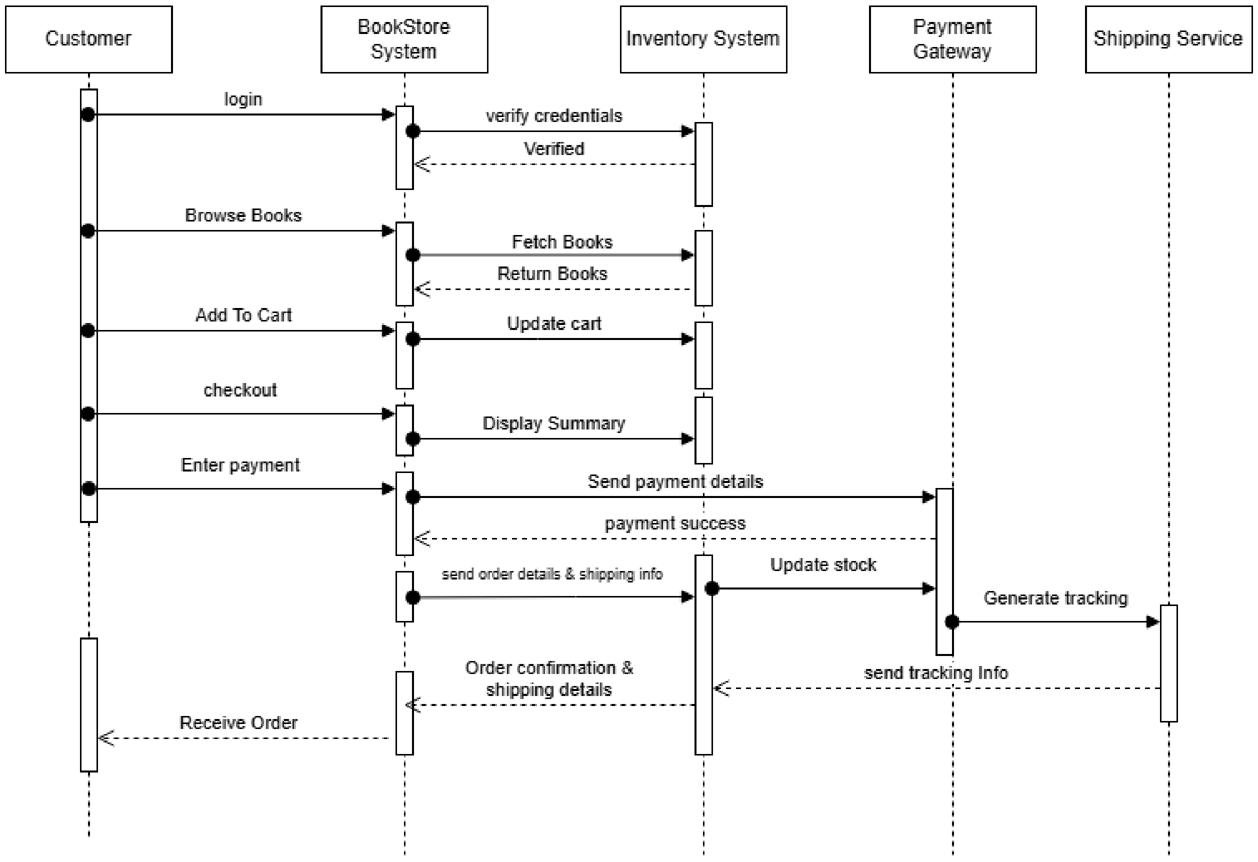

Figure4: Sequence Diagram

6.4 State Machine Diagram

6.4.1 Introduction to State Machine Diagrams

State machine diagrams (or state diagrams) are UML diagrams that depict the various states an object or system can transition through in response to events during its lifecycle. They graphically represent the dynamic behavior, illustrating states, transitions, and events that drive the system's behavior.

6.4.2 Usefulness and Representation

States represent conditions or modes of an object; transitions define movement between states, triggered by events or conditions. They are useful for modeling complex behaviors, object lifecycles, and workflows.

6.4.3 State Machine Diagram Representation

- ❖ Describes the behavior of various states within an Online Bookstore System.
- ❖ Includes each required state with its transition.
- ❖ Features entry-exit actions.

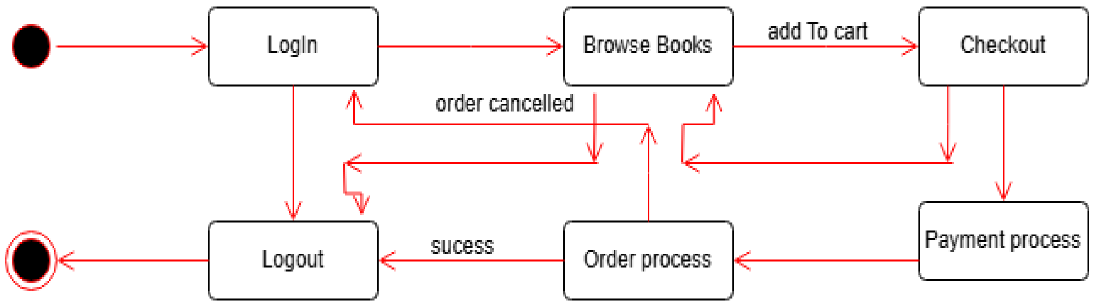

Figure5: State Machine Diagram

6.5 Component Diagram

A component diagram of an online bookstore represents the modular parts of the system and their interrelationships, showcasing how the software components interact to fulfill the system's functionality. Key components typically include the **User Interface** (for browsing, searching, and purchasing books), **Authentication Module** (for login and registration), **Catalog Management** (to handle book listings and details), **Shopping Cart** (to manage user-selected items), **Order Processing** (to handle order placement and status tracking), **Payment Gateway** (to process payments securely), and **Database Access Layer** (to interact with the backend database for storing user data, book inventory, and transactions). These components are connected through interfaces, showing how they communicate to deliver the complete online bookstore experience.

A **component diagram** consists of:

1. **Components** – Modular parts of the system (e.g., Login, Payment).
2. **Interfaces** – Points of communication between components; can be **provided** (lollipop) or **required** (socket).
3. **Dependencies** – Dashed arrows showing one component depends on another.

4. **Connectors** – Lines linking interfaces to show communication.
5. **Packages** (optional) – Groups of related components for organization.

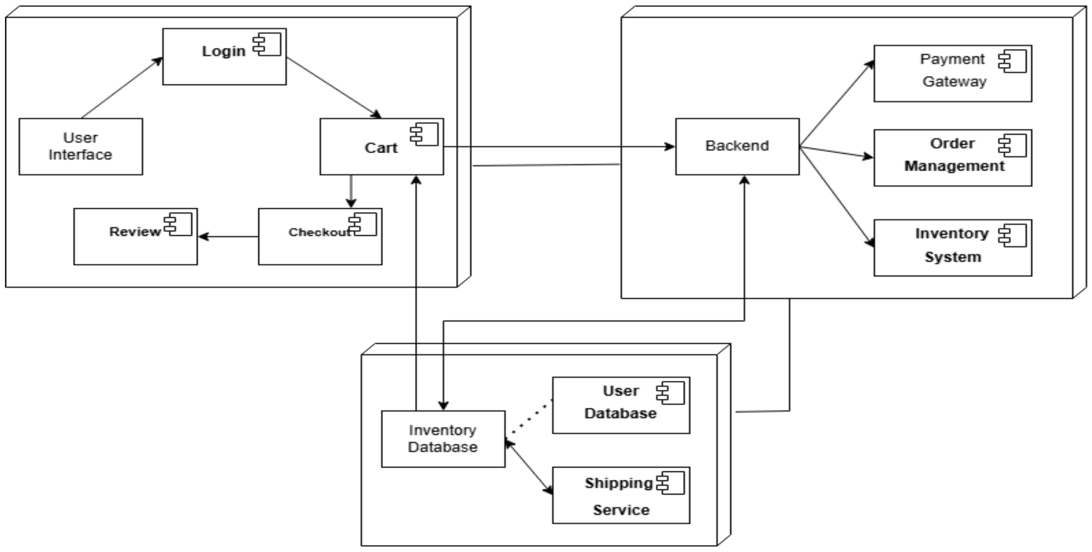

Figure6: Component Diagram

6.6 Deployment Diagram

A deployment diagram for an online bookstore illustrates the physical arrangement of the system's components: users access the bookstore via client devices (such as PCs or smartphones) using web browsers, which send requests over the internet to a web server hosting the bookstore application. This web server processes user interactions and communicates with a dedicated database server that stores all data including user accounts, books, and orders. The client and web server connect through HTTP/HTTPS, while the web server and database server interact via database queries, showing how the software is distributed across hardware to support seamless online shopping.

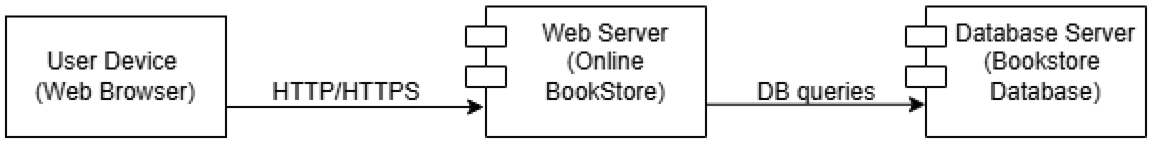
User Device (Web Browser)
HTTP/HTTPS
Web Server (Online BookStore)
DB queries
Database Server (Bookstore Database)

Figure7: Deployment Diagram

7. Implementation Model

The provided case study report primarily focuses on the analysis and design phases of the **Online Bookstore System** using UML diagrams. It does not contain an "Implementation Model" section or actual code. Typically, an Implementation Model for the Online Bookstore System would detail:

- ❖ **Programming Languages**: The specific languages used, such as Java for backend development or JavaScript/TypeScript for frontend development.
- ❖ **Frameworks and Libraries**: The frameworks or libraries utilized, such as Spring Boot for the backend API, React or Angular for the frontend, and Hibernate for ORM.
- ❖ **Database**: The chosen database system, such as MySQL or PostgreSQL, including its schema design for managing users, books, orders, and payments.
- ❖ **Deployment Environment**: How the system is deployed, for example, on cloud platforms like AWS or Azure, or on on-premise servers.
- ❖ **Key Code Snippets**: Examples of critical code logic, architectural patterns (like MVC), and integration points such as payment gateway APIs.
- ❖ **Testing Strategy**: How the system is tested, including unit testing for individual components, integration testing to ensure modules work together, and system testing for end-to-end functionality.

To complete this section, actual development and coding would be required based on the design documented in the preceding sections.

8. Conclusion and Recommendation

8.1 Conclusion

In conclusion, this Object-Oriented Analysis and Design (OOAD) report for the **Online Bookstore System** offers a thorough and structured depiction of the system's architecture, core functionalities, and interactions. By extensively employing various UML diagrams—including class diagrams, use case diagrams, activity diagrams, sequence diagrams, and state machine diagrams—the critical components and behaviors of the system have been clearly captured and effectively illustrated.

The class diagrams provide a solid foundation for understanding the system's static structure, detailing key entities such as Book, User, Order, Shopping Cart, Payment, Catalog, and Review, along with their attributes, relationships, and responsibilities. Use case diagrams present a comprehensive overview from the users' perspective, outlining essential functionalities like browsing, purchasing, and account management, thereby ensuring alignment with stakeholder needs. Activity diagrams demonstrate the dynamic flow of important processes such as order placement and payment handling, helping to optimize system workflows. Meanwhile, state machine diagrams model the lifecycle and state transitions of vital entities like orders and payments, contributing to a robust system design and validation. [1]

Overall, this report exemplifies a methodical and well-organized approach to system modeling, utilizing UML notations to offer valuable insights into the system's requirements, design, and future implementation. This foundational work is vital for building a reliable, efficient, and user-friendly online bookstore system that effectively serves the needs of customers, administrators, and other stakeholders.

8.2 Recommendation

Based on the detailed analysis and design presented, the following recommendations are made for the successful development and future enhancement of the Online Bookstore System:

- ❖ **Use Agile Sprints:** Deliver features in 2–4-week iterations to adapt quickly and gather feedback.
- ❖ **Microservices Architecture:** Build services (catalog, orders, payments) as independent Spring Boot components, containerized with Docker.
- ❖ **Comprehensive Testing:** Combine unit, integration, and end-to-end tests (JUnit, Jest, Cypress) to ensure reliability.
- ❖ **Security & Compliance:** Integrate PCI-compliant payment gateways, encrypt data (TLS/AES-256), and follow GDPR/CCPA.
- ❖ **Responsive UX:** Implement a mobile-first React frontend with accessibility best practices.
- ❖ **Scalable Infrastructure:** Orchestrate with Kubernetes, monitor with Prometheus/ELK, and plan for future mobile app and recommendation-engine integrations.

9. REFERENCES

[1] V. Paradigm, " UML Tool and Resources. Retrieved from https://www.visual-paradigm.com/," 2024.

[2] C. Larman, "An Introduction to Object-Oriented Analysis and Design and Iterative Development," (3rd ed.)Pearson Education., 2004.

[3] M. Fowler, "UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd ed.)," (3rd ed.). Addison-Wesley, 2003.

[4] Baeldung, "Spring Boot Tutorials and Guides. Retrieved from https://www.baeldung.com/spring-boot," 2024.

[5] F. Inc., "A JavaScript Library for Building User Interfaces. Retrieved from https://reactjs.org/," 2024.