Group 10 Book Store System Architecture

1. Project Summary

This document outlines the finalised proposal and comprehensive system architecture for the *Online Bookstore Simulation* project. The system is designed using object-oriented programming (OOP) principles in JavaScript, with MongoDB as the data persistence layer. The simulation models core e-commerce functionalities such as browsing inventory, managing a shopping cart, and completing a purchase.

2. System Objectives

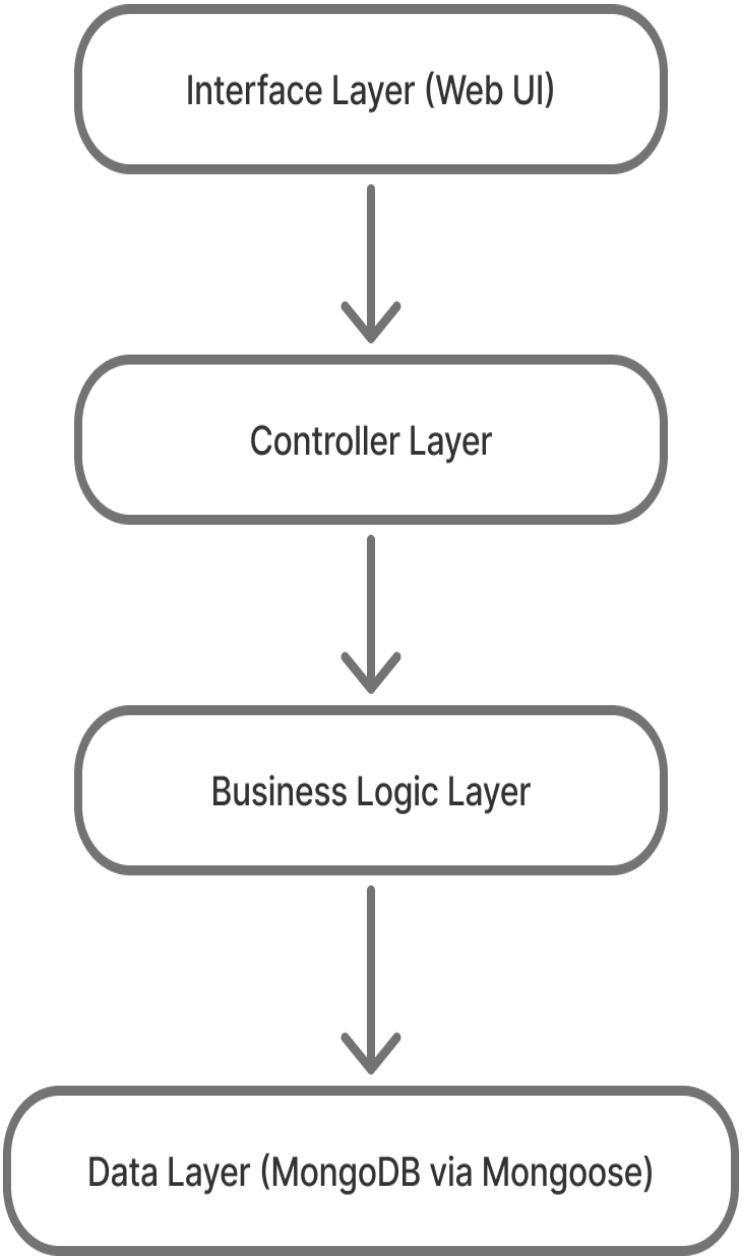
The primary objectives of the project are:

- To simulate an online bookstore backend using JavaScript classes and objects.
- To model real-world entities (Books, Users, Cart, Store) using OOP principles.
- To persist book inventory using MongoDB.
- To enable users to interact with the system through cart management and purchasing logic.
- To demonstrate modular software architecture and clean code practices.

3. System Architecture

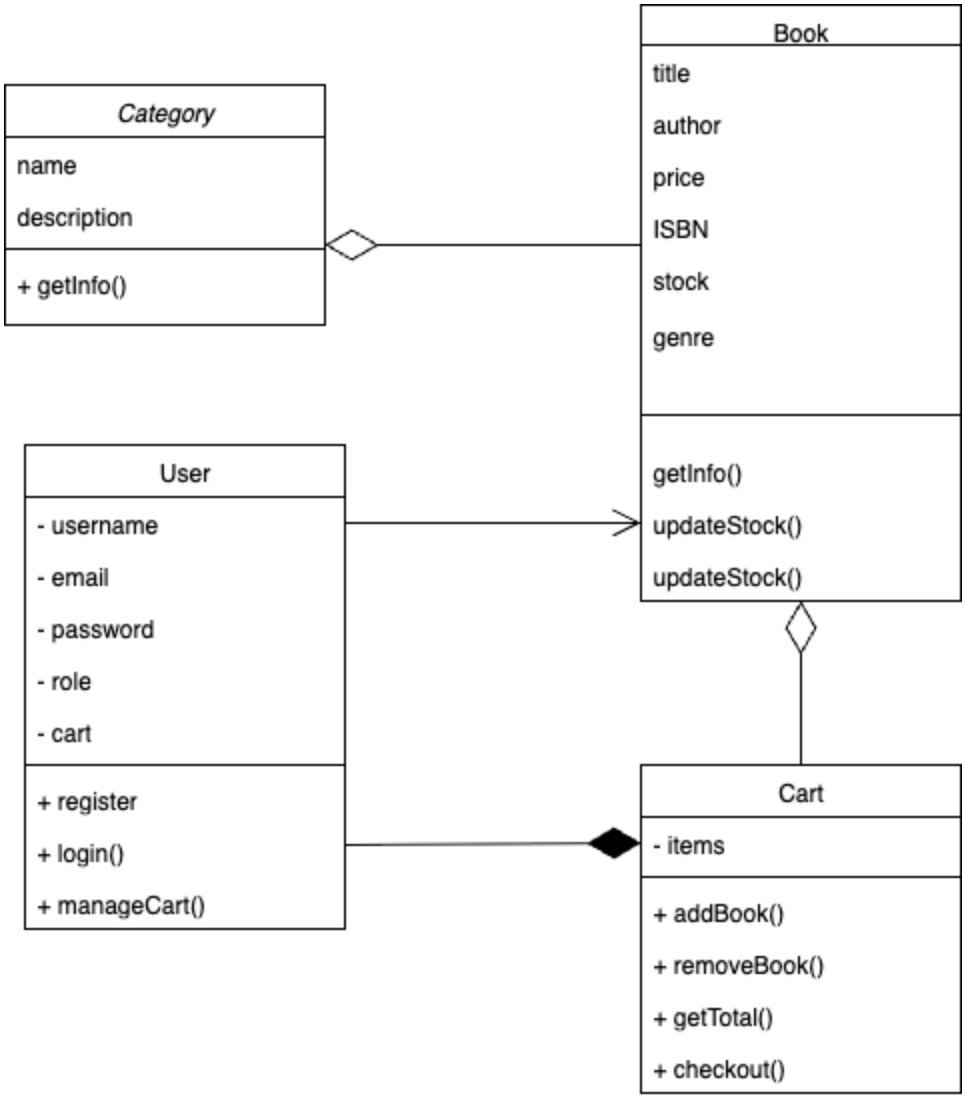
3.1 Architectural Overview

The system follows a modular, layered architecture, composed of the following main layers:



Each component is decoupled for maintainability, clarity, and reusability.

3.2 Core Components



3.2.1 Book Class

Responsibilities:

- Represents a single book instance.
- Contains book-specific data: title, author, price, genre, stock, ISBN.
- Used as the atomic unit of all cart and inventory interactions.

Attributes:

```
title: String
author: String
```

```
price: Number
genre: String
isbn: String
stock: Number
```

Methods:

- `getDetails()`: Returns full book metadata.
- `updateStock(quantity)`: Adjusts stock level.
- `isInStock()`: Returns true if stock > 0.

3.2.2 User Class

Responsibilities:

- Models user behaviour and identity.
- Connects user to their shopping cart.

Attributes:

```
username: String
email: String
password: String
cart: Cart (composition)
```

Methods:

- `addToCart(book, quantity)`
- `removeFromCart(book)`
- `viewCart()`

- `checkout()`

3.2.3 Cart Class

Responsibilities:

- Manages a collection of books selected by the user.
- Handles pricing, quantities, and total computation.

Attributes:

`items: [ { book: Book, quantity: Number } ]`

Methods:

- `addBook(book, quantity)`
- `removeBook(book)`
- `getTotal()`
- `clearCart()`

3.2.4 Store Class

Responsibilities:

- Acts as a gateway between the user and backend services.
- Manages database interactions.
- Facilitates business workflows (e.g., purchasing).

Methods:

- `loadBooks()`: Fetches books from MongoDB.
- `updateBookStock(bookId, quantity)`: Writes updated stock values to DB.
- `processPurchase(user)`: Validates cart, updates stock, clears cart.

3.3 Data Layer (MongoDB + Mongoose)

The database will store book inventory. Each document in the `books` collection contains:

```
{
  "_id": ObjectId,
  "title": "Example Book",
  "author": "Author Name",
  "price": 15.99,
  "genre": "Fiction",
  "isbn": "123-4567890123",
  "stock": 10
}
```

Mongoose will provide schema definitions and abstraction for query operations.

3.4 Frontend Layer

A minimal web-based UI (HTML/CSS/JS) developed to:

- Display a list of available books.
- Allow users to interact with their cart.
- Simulate the checkout process.

This frontend will interact with the backend using a REST-like structure.

4. Technologies Used

Layer	Technology
Programming Language	JavaScript (ES6+)
Backend Runtime	Node.js
Database	MongoDB
ODM Tool	Mongoose
Version Control	Git, GitHub
Editor	Visual Studio Code
Frontend UI	HTML/CSS/JS or React

5. Project Timeline

Day	Milestone
1	● Finalize project plan, define schema, design class/UML diagrams. ● Implement core classes: Category, Book, and User
2	● Integrate MongoDB and seed data for books and categories ● Develop cart logic and checkout workflow
3	● Conduct testing, fix bugs, validate functionality
4	● Wrap-up: documentation, optional UI, prepare demo & submission

6. Team Structure and Roles

Role	Responsibilities
Backend Dev 1 **(Ghislaine)**	Develop `Book` and `Category` classes, define schemas, seed MongoDB; assist with testing and documentation
Backend Dev 2 **(Samuel)**	Build `User` class and registration/login logic; cart reference; assist with testing and documentation
Backend Dev 3 **(Okeke)**	Build `Cart` and `Store` classes, implement purchase logic and interactions with database; assist with testing and documentation
Frontend Dev **(Enzo)**	Create a basic UI for browsing books and managing cart; assist with testing and documentation

7. Deliverables

- Fully functional backend simulation using OOP in JavaScript.
- MongoDB database integration with CRUD operations on books.
- Basic UI for user interactions.
- Well-documented source code (README and Swagger docs).

- GitHub repository containing all project files and version history.

8. Conclusion

This project encapsulates the foundational principles of software design: modularity, abstraction, maintainability, and scalability. By leveraging JavaScript's OOP capabilities in combination with MongoDB, we deliver a clean simulation that could serve as a base for real-world e-commerce systems. The final product will be extensible, testable, and educationally valuable as a practical example of full-stack design using object-oriented programming.