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

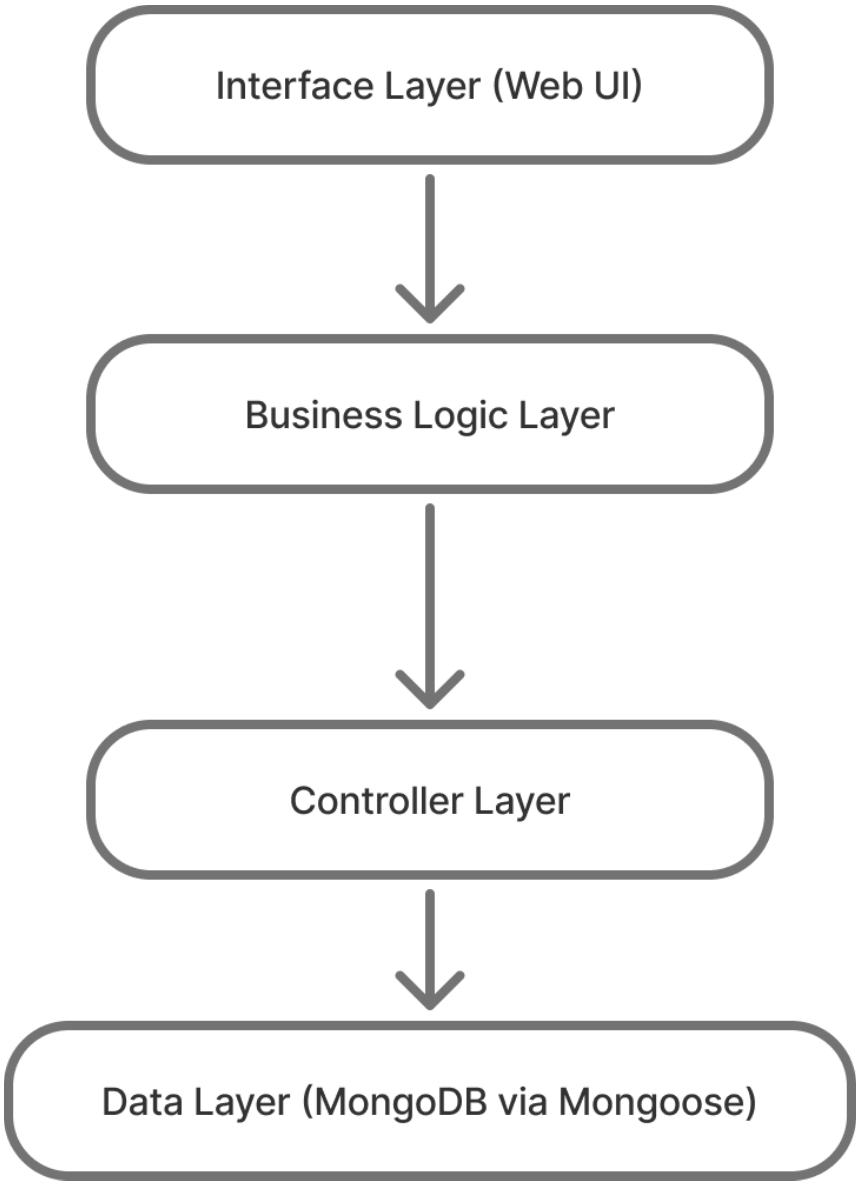
The primary objectives of the project are:

- 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 the 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

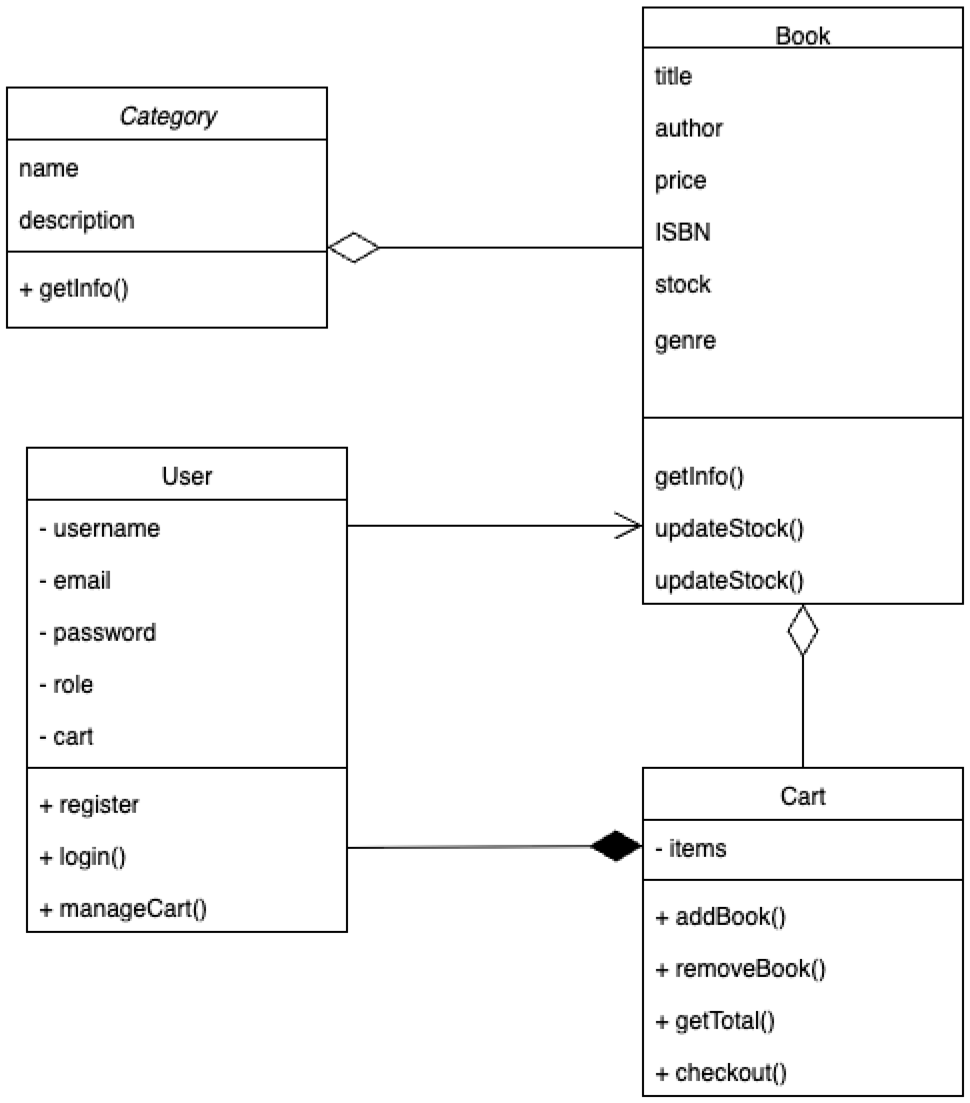
3.1 Architectural Overview

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, and 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
```

3.2.2 User Class

Responsibilities:

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

Attributes:

```
username: String
email: String
password: String
```

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 } ]
```

3.3 Data Layer (MongoDB + Mongoose)

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

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

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	Node.js
Database	MongoDB
ODM Tool	Mongoose
Version Control	Git, GitHub
Editor	Visual Studio Code
Frontend UI	HTML/CSS/JS or React

5. Team Structure and Roles

Role	Responsibilities
Backend Dev 1 **(Ghislaine)**	Develop User, `Book`, and `Category` classes, define schemas, routes and documentation, and work on the project proposal document.
Backend Dev 2 **(Samuel)**	Develop Cart class, define schema, route and documentation, implement throttling, and work on the system architecture document.
Backend Dev 3 **(Okeke)**	Test the backend system, create the UML diagram, implement documentation and work on the project proposal document.
Frontend Dev **(Enzo)**	Create a basic UI for browsing books and managing the cart, implement business logic, and work on the system architecture document.

6. Deliverables

- Modular JavaScript backend using OOP principles
- MongoDB integration with Mongoose
- Data for books and categories
- Simple frontend
- Full project documentation:
    - UML class diagram
    - README
    - Setup instructions

- GitHub repository with version-controlled source code
- Demo video walkthrough

7. 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.