Department: **FEST** Program: **BS (CS)**

**Full Stack Development**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Announced date: 30/12/2024 | | | | Due Date: 22/1/2025 | Total Marks = 11 |
| **Complex Computing Problem (CCP)** | | | | | |
| **Mapped CLO** | **SDG** | **Complex Problem Solving Mapped** | | | |
| CLO3 | 4 & 9 | WP1 (Range of Conflicting Requirements)  WP2 (Depth of Analysis required)  WP3 (Depth of Knowledge required) | | | |

**Problem Statement:**

Develop a **full-stack web application** for an online bookstore to manage book inventories, orders, and user accounts. The system should provide separate interfaces for **customers** and **admins**. Customers can browse books, place orders, and view order history, while admins can manage books and monitor sales.

### **Functional Requirements:**

1. **Customer Interface:**
   * **Browse Books:**
     + View book details: Name, Author, Price, Category, and Availability.
   * **Order Placement:**
     + Add books to the cart and place orders.
   * **Order History:**
     + View previously placed orders.
2. **Admin Interface:**
   * **Book Management:**
     + Add, update, or delete books.
   * **Sales Overview:**
     + View daily, weekly, and monthly sales.
3. **Authentication and Authorization:**
   * User authentication for customers and admins.
   * Role-based access: Admin vs. Customer.
4. **Search and Filter Books:**
   * Search by book name or author.
   * Filter by category and price range.
5. **Responsive Design:**
   * Ensure the application is mobile-friendly using Bootstrap.

### **Constraints/Assumptions:**

1. Use **React** for the front-end and **Laravel** for the back-end.
2. Database: **MySQL** or **PostgreSQL**.
3. API Integration: Use Laravel APIs for fetching data dynamically.
4. Ensure modular architecture with reusable components (e.g., BookList, Cart).

### **Identification of Areas Where Computational/Modern Tools Are Required:**

1. **Development Environment:**
   * Tools: Visual Studio Code, Node.js, Laravel, Postman.
2. **Front-End Development:**
   * React for UI, Bootstrap for responsive design.
3. **Back-End Development:**
   * Laravel for APIs, Eloquent ORM for database management.
4. **Testing:**
   * Tools: Jest (front-end), PHPUnit (back-end).

### **Expected Outcomes:**

1. A functional web application with the following:
   * Customer and admin interfaces.
   * Real-time book browsing and ordering.
   * Secure login and role-based access control.
2. RESTful APIs for data exchange between front-end and back-end.
3. A responsive, user-friendly UI.

**Deliverables**

### **Deliverables:**

**Report (must include):**

1. **Introduction:**
   * Objective and scope of the web application.
2. **Designed Model:**
   * ER Diagram and architecture overview.
   * Mockups for customer and admin interfaces.
3. **Methodology:**
   * Tools and frameworks used.
   * Steps for implementing APIs, authentication, and UI.
4. **Result and Discussion:**
   * Screenshots of the developed app showing key features.
   * Explanation of app functionality.
   * Link to the app code on GitHub.

### **Additional Challenges (Optional):**

1. **Real-Time Notifications:**
   * Use WebSockets for notifying admins about new orders.
2. **Graphical Sales Dashboard:**
   * Integrate charts for sales data visualization.
3. **Unit and Integration Testing:**
   * Write test cases for critical components.
4. **Advanced Filtering:**
   * Implement dynamic filters with AJAX.

s

|  |  |  |
| --- | --- | --- |
| **CCP Attributes mapped** | | |
| **Attributes of Complex Problem Solving** | | **Justification** |
| **WP1** | Range of Conflicting Requirements | Balancing features like real-time ordering with a clean, responsive UI design. |
| **WP2** | Depth of Analysis required | Requires analyzing and designing RESTful APIs, database relationships, and flows. |
| **WP3** | Depth of Knowledge required | Involves integrating advanced front-end and back-end technologies. |