Project Documentation

PREPARED BY
Prof. Naimish Vadodariya
Computer Science & Engineering – 5th Semester
Darshan University
A.Y. - 2025-26

Table of Contents

1. Project Goals & Objectives

Overview

- This project aims to develop a basic management system that enables users to efficiently manage data related to various core entities of the application.
- The application should support full CRUD operations, session and role-based access control, input validation, and filtering across all functional modules.
- Role-based access will determine what actions can be performed by different types of users (e.g., admin, manager, staff, customer).
- The backend will be developed using ASP.NET Core Web API to handle all business logic and data operations securely.

Example for an Online Bookstore Management System:

- This project aims to develop an Online Bookstore Management System to manage books, authors, categories, customers, and orders efficiently.
- The system will include complete CRUD operations, user authentication, role-based access control, and input validation across all modules.
- Different user roles such as **Admin**, **Customer**, and **Staff** will have different access levels—ensuring only authorized actions are allowed per user type.
- All backend operations will be handled through a secure ASP.NET Core Web API.

Goals

- **Design a Normalized Database Structure:**
  - Create normalized tables for core entities.
  - Establish one-to-many and many-to-many relationships where needed, including a structure for storing user roles and permissions.
- **Implement CRUD & Authentication Functionality:**
  - Develop Create, Read, Update, and Delete operations for all major modules via Web API.
  - Implement secure user authentication (login, logout, registration) and role-based access controls.
- **Ensure Data Integrity and Validation:**
  - Apply validations for all required fields and enforce consistency in the database.
  - Prevent unauthorized data changes through access checks based on user roles.
- **Develop an Intuitive User Interface:**
  - Build a user-friendly frontend that adapts visibility and accessibility of features based on user roles.
  - Enable role-specific dashboards and menu options.
- **Encourage Good Software Development Practices:**
  - Apply clean coding principles, separation of concerns, and RESTful API conventions.
  - Structure the codebase to support scalability, maintainability, and secure role-based logic.

Example for an Online Bookstore Management System:

- **Design a Normalized Database Structure:**
  - Create normalized tables for books, authors, categories, customers, orders, and reviews.

- Implement many-to-many relationships (e.g., books ↔ authors) and one-to-many (e.g., category → books), including role and user mapping tables.

- **Implement CRUD & Authentication Functionality:**
  - Use ASP.NET Core Web API to implement CRUD for all modules (Books, Categories, Orders, etc.).
  - Authenticate users with login, registration, and logout features.
  - Control access based on roles (e.g., only Admins can manage books; Customers can view and place orders).
- **Ensure Data Integrity and Validation:**
  - Validate inputs like ISBN, email, price, and quantity on both frontend and backend.
  - Prevent unauthorized data operations by enforcing role-based restrictions on APIs.
- **Develop an Intuitive User Interface:**
  - Design a user-friendly frontend that changes dynamically based on the logged-in user's role.
  - Admins see full management tools; customers see product listings, order history, and checkout pages.
- **Encourage Good Software Development Practices:**
  - Implement RESTful API design, modular coding, and layered architecture.
  - Use role-based middleware and services to maintain clean separation of concerns.

Objectives

- **Entity Management:**
  - Enable users to create, view, update, and delete records for all major entities involved in the system.
  - Capture both essential and additional information as required for each entity.
- **Category/Group Management:**
  - Manage groupings or classifications and associate them with primary entities.
  - Access permissions to modify these should be role-restricted (e.g., only admin).
- **User Registration and Record Management:**
  - Enable registration, profile updates, and logical deletion of user-related records.
  - Only authorized roles should access or modify sensitive user data.
- **Scheduling or Transaction Handling:**
  - Allow scheduling or transactional processes to be created and managed based on role (e.g., customer books, admin approves).
  - Role filters should determine who can view, edit, or cancel a transaction.
- **Entity Relationship Mapping:**
  - Maintain complex mappings (e.g., many-to-many) between core entities with update permissions restricted by role.
- **Data Integrity and Validation:**
  - Validate user input on both client and server sides.
  - Use role-based restrictions to prevent unauthorized data tampering.

Example for an Online Bookstore Management System:

- **Book Management:**
  - Admins can add, update, view, or delete books, including metadata like title, author, category, price, and stock.
  - Customers and Staff can only view book details based on role permissions.

- **Author and Category Management:**
    - Admins manage categories and authors.
    - Books can be associated with multiple authors and one category using relationship tables.
- **Customer Registration and Profile Management:**
    - New customers can register, update their profile, and view their personal order history.
    - Only Admins can view all customer data..
- **Order Placement and Tracking:**
    - Customers can place, update, or cancel their own orders.
    - Staff can view all orders for processing, and Admins can update order statuses.
- **Book-Author Association:**
    - Admins manage many-to-many relationships between books and authors.
    - Only authorized roles can add or remove these associations.
- **Data Integrity and Validation:**
    - Validate formats like ISBN, email, and phone.
    - Use role-based checks to avoid privilege misuse (e.g., a customer trying to modify stock levels).

2. Scope of the Project

- **Entity and Category Management**
    - Users with admin or manager roles can create, update, or delete records.
    - Category/group tagging should be managed by privileged users only.
    - Mapping relationships between entities should respect role permissions.
- **User or Record Management**
    - Basic users can view or update their own records.
    - Admin-level users can view, approve, or deactivate other users and perform soft deletes.
- **Transaction or Event Scheduling**
    - Regular users can create or request transactions (e.g., bookings, orders).
    - Admin or staff can confirm, reschedule, or cancel based on permissions.
    - Role restrictions should govern visibility of event details and access to actions.
- **Dashboard and Summary Views**
    - Dashboard content will differ by role (e.g., admin sees full analytics, user sees personal activity).
    - Summary cards and filters must only show data that the role has permission to access.
- **Input Validation and Error Handling**
    - All roles should experience consistent and helpful feedback for invalid inputs.
    - Unauthorized actions must trigger proper role-based error messages and redirections.

Example for an Online Bookstore Management System:

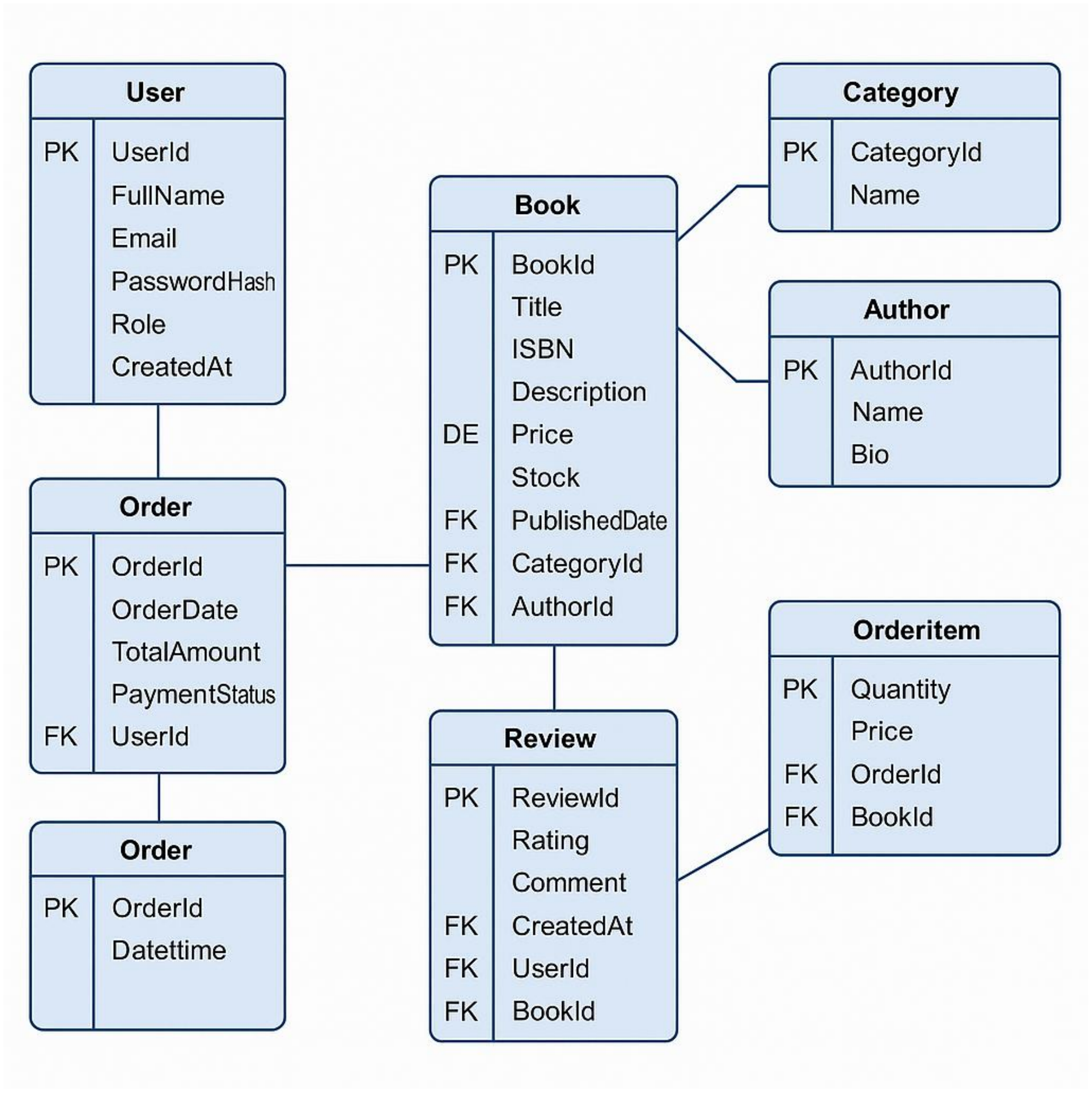
- **Book and Category Management**
    - Admins can add, update, or delete books, categories, and author profiles.
    - Customers can view books and filter by category or author.
    - Staff may have read-only access to books for stock review purposes
- **Customer and Order Management**
    - Customers can manage their personal profiles and view their orders.
    - Staff can view all orders to handle dispatching.

- Admins have full control over all users and order records
- **Order Scheduling and Processing**
  - Customers place and manage orders.
  - Staff updates order status to "Packed", "Dispatched", or "Delivered".
  - Admin can override or cancel orders if required
- **Dashboard and Reports**
  - Admins see analytics like sales reports, top-selling books, and user metrics.
  - Staff sees pending orders and stock alerts.
  - Customers see their latest orders and recommendations.
- **Input Validation and Error Handling**
  - All roles experience user-friendly validation feedback.
  - Unauthorized actions return role-specific error messages (e.g., "Admin access required" or "Permission denied").

3. Project Timeline

Week No.	Week Date	Task List
1	16/06/2025 - 22/06/2025	Requirement Gathering, Project Scope, and Feasibility Analysis
2	23/06/2025 - 29/06/2025	Design Database Schema, Create EF Core Models, Define Entities & Relationships
3	30/06/2025 - 06/07/2025	Set up Web API Project, Configure EF Core with Database, Scaffold Basic CRUD for Main Module
4	07/07/2025 - 13/07/2025	Implement CRUD APIs for Additional Modules/Tables
5	14/07/2025 - 20/07/2025	Add Validation, Data Annotations, Business Logic Layer
6	21/07/2025 - 27/07/2025	Implement JWT Authentication and Authorization
7	28/07/2025 - 03/08/2025	Implement Role-Based Access Control in APIs
8	04/08/2025 - 10/08/2025	Add Advanced API Features: Filtering, Sorting, Pagination
9	11/08/2025 - 17/08/2025	Consume Web API in Client Application — Part 1 (Setup, Basic Data Retrieval, Display)
10	18/08/2025 - 24/08/2025	Consume Web API in Client Application — Part 2 (Role-Based UI, Secure Calls with JWT, Error Handling)
11	25/08/2025 - 31/08/2025	Implement Extra Functionalities (Logging, Caching, Notifications, Export Data, etc.)
12	01/09/2025 - 07/09/2025	Evaluation

Sample ER Diagram for Online Bookstore Management System


User
PK UserId
FullName
Email
PasswordHash
Role
CreatedAt
Order
PK OrderId
OrderDate
TotalAmount
PaymentStatus
FK UserId
Order
PK OrderId
Datettime
Book
PK BookId
Title
ISBN
Description
DE Price
Stock
FK PublishedDate
FK CategoryId
FK AuthorId
Review
PK ReviewId
Rating
Comment
FK CreatedAt
FK UserId
FK BookId
Category
PK CategoryId
Name
Author
PK AuthorId
Name
Bio
Orderitem
PK Quantity
Price
FK OrderId
FK BookId