# **Project Documentation: Online Learning Platform with ASP.NET Core**

## **1. Project Overview**

### **Objective:**

The objective of this project is to build a web-based application that enables users to manage online courses and enrollments efficiently. Users will be able to browse available courses, enroll in them, and track their progress. The platform will be secured with user authentication and implement role-based access control to ensure different levels of access for different user roles.

### **Key Features:**

* Course browsing and enrollment.
* Student progress tracking.
* Role-based access control for different users.
* User authentication and profile management.

## **2. Functional Requirements**

### **User Roles:**

* **Students**:
  + Can browse courses, view course details, and enroll.
  + Can track their progress in the courses they have enrolled in.
* **Instructors**:
  + Can create and manage course content.
  + Can track the enrollments and progress of students in their courses.
* **Admins**:
  + Can manage the platform, including creating or deleting courses and managing users.
  + Have full control over all resources, including courses and user permissions.

### **Features:**

1. **Course Catalog**:
   * Users can view available courses, including title, description, instructor name, and course duration.
2. **User Authentication**:
   * A login system with secure password handling and validation.
   * Role-based access, ensuring students, instructors, and admins have access to appropriate features.
3. **Enrollment System**:
   * Students can enroll in available courses.
   * Enrollment status is updated and tracked per student.
4. **Progress Tracking**:
   * Students can see their learning progress (completed modules, quizzes, etc.) in enrolled courses.
   * Instructors can view the progress of students enrolled in their courses.
5. **Admin Dashboard**:
   * Admins can add/edit/delete courses and manage user roles (e.g., promoting a user to an instructor or admin).
   * Overview of platform statistics (number of courses, enrollments, active users).

## **3. System Design and Architecture**

### **3.1 Application Architecture:**

* **Backend**:
  + ASP.NET Core will be used to handle all business logic, routing, and database interaction.
  + Controllers will manage HTTP requests, interact with services, and render appropriate views.
* **Frontend**:
  + Razor Pages will be used to render dynamic HTML content.
  + Forms will allow users to input data, such as login credentials and course enrollment actions.
* **Database**:
  + The platform will use **Entity Framework Core** to manage interactions with the underlying SQL Server database.
  + The database schema will contain tables for Users, Roles, Courses, Enrollments, and Progress.

### **3.2 Key Components:**

* **Model-View-Controller (MVC)**:
  + **Model**: Defines the data structure (Courses, Users, Enrollments).
  + **View**: Defines the UI that users interact with, written in Razor.
  + **Controller**: Handles input, processes it using services, and returns the appropriate response.

## **4. Technologies Stack**

### **Backend:**

* **ASP.NET Core**:  
  The core framework for building the web application, handling business logic, and processing requests.

### **Frontend:**

* **Razor Pages (ASP.NET Core MVC)**:  
  Used to generate dynamic pages and interact with the backend logic.

### **Data Access:**

* **Entity Framework Core**:  
  An Object-Relational Mapping (ORM) framework that provides database interaction with SQL Server. This allows for easy querying and database management through models.

### **Database:**

* **SQL Server**:  
  The relational database management system used to store user data, courses, enrollments, and progress tracking.

### **Security:**

* **ASP.NET Identity**:  
  A membership system that provides user authentication, password recovery, and role-based access control.

## **5. Database Design**

### **Tables:**

* **Users**:
  + UserID (Primary Key)
  + Name
  + Email
  + PasswordHash
  + RoleID (Foreign Key)
* **Roles**:
  + RoleID (Primary Key)
  + RoleName (Student, Instructor, Admin)
* **Courses**:
  + CourseID (Primary Key)
  + Title
  + Description
  + InstructorID (Foreign Key)
* **Enrollments**:
  + EnrollmentID (Primary Key)
  + CourseID (Foreign Key)
  + UserID (Foreign Key)
  + EnrollmentDate
* **Progress**:
  + ProgressID (Primary Key)
  + EnrollmentID (Foreign Key)
  + CompletedModules
  + TotalModules

## **6. User Authentication and Authorization**

### **User Authentication:**

* **ASP.NET Identity**:  
  This will manage user sign-in and authentication, including password hashing and recovery.

### **Role-Based Access Control (RBAC):**

* Role assignments are handled through the Identity system.
  + **Students** have access to view and enroll in courses.
  + **Instructors** can manage courses they created.
  + **Admins** can manage the entire platform, including user roles and courses.

## **8. Security Considerations**

### **8.1 Authentication:**

* Secure authentication will be implemented using ASP.NET Identity with hashed passwords and validation checks for secure user access.

### **8.2 Authorization:**

* Only authenticated users with specific roles (student, instructor, or admin) can perform role-specific actions. For example, only instructors can create courses, and only admins can delete them.

### **8.3 Data Protection:**

* All sensitive user information (e.g., passwords) will be encrypted.
* Role-based access control will ensure that users cannot access resources they are not authorized to manage.

## **9. Future Improvements**

### **9.1 Additional Features:**

* **Quiz and Assessment Modules**: Allow students to complete quizzes to assess their knowledge.
* **Messaging**: Enable students to communicate with instructors through the platform.
* **Discussion Forums**: Add a discussion section for each course where students can collaborate.

### **9.2 Scalability:**

* The application can be scaled by using cloud-based hosting (e.g., Azure) and additional caching mechanisms for performance improvements.

## **10. Conclusion**

This project provides a robust solution for managing online courses and enrollments, ensuring secure user authentication and clear role-based access control. The use of ASP.NET Core and Entity Framework ensures a modern, scalable architecture that can be expanded as needed.

This documentation outlines the key features, system architecture, and technical requirements for the platform, serving as a roadmap for development and future enhancements.