**.Net Web Developer**

**Project 1: Online Learning Platform with ASP.NET Core**

* **Objective**: Create a web application for managing online courses and student enrollments.
* **Description**: Develop a web-based platform that allows users to browse courses, enroll, and track their progress. Use ASP.NET Core for the backend and implement user authentication and role-based access control.
* **Technologies to use**: ASP.NET Core, MVC, Entity Framework, User Authentication.

**Week 1: Initial Setup and Basic Course Listings**

* **Set up project environment:** Install and configure ASP.NET Core with MVC and Entity Framework.
* **Database design:** Create the initial database schema for users, courses, and enrollments using Entity Framework.
* **Develop course listing:** Implement a basic page where users can browse courses, with details like course description and instructor information.
* **User authentication setup:** Integrate basic authentication with user registration and login using Identity.

**Deliverables:**

* ASP.NET Core project set up.
* Basic database schema created using Entity Framework.
* Course listing page with basic navigation.
* User authentication system (login and registration).

**Week 2: Course Enrollment and Role-Based Access Control**

* **Course enrollment functionality:** Implement the ability for authenticated users to enroll in courses.
* **Role-based access control:** Set up roles (e.g., student, instructor, admin) with different permissions and access levels.
* **Admin dashboard:** Develop a basic admin dashboard for course and user management + Frontend functions.
* **Testing:** Test user registration, role-based access control, and course enrollment functionality.

**Deliverables:**

* Working course enrollment feature.
* Roles and permissions established.
* Admin dashboard for managing users and courses.
* Tested and functional user roles and course enrollment.

**Week 3: Progress Tracking and User Profiles**

* **Progress tracking:** Implement a system that allows users to track their progress in enrolled courses.
* **User profiles:** Create profile pages for users to view their personal information and track their enrolled courses and progress.
* **Search and filtering:** Add search functionality to allow users to filter and find specific courses.
* **Testing:** Ensure that progress tracking works and all users can view their profiles correctly.

**Deliverables:**

* Progress tracking system for enrolled courses.
* User profile pages with course tracking information.
* Search and filtering functionality.
* Completed testing for profile pages and progress tracking.

**Week 4: Final Testing, UI Enhancements, and Deployment**

* **UI enhancements:** Improve the overall user interface using CSS and JavaScript for better UX.
* **Final testing:** Conduct thorough testing of all functionalities (course enrollment, progress tracking, admin management).
* **Deployment:** Deploy the platform on a hosting service (Azure, AWS, or local IIS).
* **Documentation:** Complete project documentation, including setup instructions and user manual.

**Deliverables:**

* Fully functional and tested online learning platform.
* Deployed platform with live URL.
* Complete project documentation.

**Project 2: Library Management System with ASP.NET Core**

* **Project Title**: " Library Management System with ASP.NET Core"
* **Objective**: Create a library management system for tracking book checkouts, returns, and penalties for late returns.
* **Description**: Create a library management system for tracking book checkouts, returns, and penalties for late returns. The system will include a search feature for available books, as well as member management. SQL Server will store book and member details. ASP.NET Core with C# will process transactions, while the frontend will use HTML, CSS, and JavaScript to manage the user interface.
* **Technologies to use**: ASP.NET Core, MVC, Entity Framework, User Authentication.

**4-Week Plan:**

**Week 1: Initial Setup and Database Design**

* **Set up project environment:** Install and configure ASP.NET Core with Entity Framework and SQL Server.
* **Design database schema:** Create the database schema for books, members, checkouts, returns, and penalties.
* **Basic book listing:** Implement a page where librarians can view all available books, including details like author, genre, and availability status.
* **User authentication setup:** Implement basic user authentication for both librarians and members.

**Deliverables:**

* ASP.NET Core project set up.
* Database schema designed and implemented with Entity Framework.
* Book listing page with basic functionality.
* User authentication (librarians and members).

**Week 2: Checkout, Returns, and Penalty Calculation**

* **Checkout functionality:** Implement the ability for librarians to record book checkouts by members, including due dates.
* **Return system:** Implement a feature to record book returns and calculate penalties for overdue books.
* **Penalty system:** Automatically calculate penalties based on the number of overdue days.
* **Testing:** Test book checkouts, returns, and penalty calculations for various scenarios.

**Deliverables:**

* Working book checkout and return system.
* Penalty calculation functionality.
* Tested and functional checkout and return system.

**Week 3: Member Management and Book Search**

* **Member management:** Develop a system to manage member information (e.g., registration, contact details, borrowing history).
* **Book search functionality:** Implement search and filtering features to help librarians find available books by title, author, or genre.
* **Member portal:** Create a portal for members to view their borrowed books, due dates, and any penalties.
* **Testing:** Ensure the member management and book search functionalities work as expected.

**Deliverables:**

* Member management system.
* Book search and filtering functionality.
* Member portal with borrowing history and penalties.
* Completed testing for member and search features.

**Week 4: Final Testing, UI Enhancements, and Deployment**

* **UI enhancements:** Refine the user interface for a better user experience, incorporating responsive design with CSS and JavaScript.
* **Final testing:** Test all features (checkouts, returns, penalties, member management) to ensure everything works correctly.
* **Deployment:** Deploy the library management system to a hosting platform (Azure, AWS, or local IIS).
* **Documentation:** Write detailed project documentation, including a setup guide and user manual.

**Deliverables:**

* Fully functional and tested library management system.
* Deployed system with a live URL.
* Complete project documentation.

**Project 3: Hospital Management System with ASP.NET Core**

**Objective:**

Create a web application to manage hospital appointments, patient records, and staff scheduling.

**Description:**

The hospital management system will allow administrators to manage doctors, patients, and appointment schedules. The system will provide separate access for hospital staff (doctors, nurses, and administrators) and patients, with role-based access control. Patients can log in to view their appointments, doctors’ details, and medical records, while administrators can manage staff schedules and patient data. Doctors will have access to patient records and appointments.

**Technologies to use:**

ASP.NET Core, MVC, Entity Framework, SQL Server, User Authentication.

**4-Week Plan:**

**Week 1: Setup and Patient Management**

* **Set up project environment:** Install and configure ASP.NET Core with MVC and Entity Framework.
* **Design database schema:** Create the database schema for patients, staff, appointments, and medical records.
* **Patient management:** Implement a page where administrators can add, edit, and delete patient details.
* **User authentication setup:** Implement user authentication with role-based access (patients, doctors, administrators).

**Deliverables:**

* ASP.NET Core project set up.
* Database schema for patients, appointments, and staff created using Entity Framework.
* Basic patient management page implemented.
* User authentication and role-based access implemented.

**Week 2: Appointment Scheduling and Staff Management**

* **Appointment management:** Implement functionality for creating and managing appointments for patients with doctors.
* **Staff management:** Create a system for managing staff members, their schedules, and their roles within the hospital.
* **Role-based access control:** Set up access permissions so that only administrators can manage appointments and staff schedules.
* **Testing:** Test the patient and staff management systems.

**Deliverables:**

* Functional appointment scheduling system.
* Staff management system for managing doctor schedules.
* Role-based access implemented for different user types.
* Tested patient and staff management functionality.

**Week 3: Medical Records and Notifications**

* **Medical record management:** Implement a system where doctors can view and update patient medical records.
* **Notifications:** Set up email or SMS notifications for patients regarding their appointments.
* **Testing:** Ensure that the medical records system works seamlessly and notifications are sent properly.

**Deliverables:**

* Functional medical record management system.
* Notifications (email/SMS) for appointment reminders.
* Completed testing of medical record management and notifications.

**Week 4: Final Testing, UI Enhancements, and Deployment**

* **UI enhancements:** Improve the user interface to make it more intuitive and responsive.
* **Final testing:** Test all features (patient management, appointment scheduling, medical records, staff management) to ensure they work smoothly.
* **Deployment:** Deploy the hospital management system to a hosting platform.
* **Documentation:** Write detailed project documentation, including setup and user guides.

**Deliverables:**

* Fully functional and tested hospital management system.
* Deployed system with live URL.
* Complete project documentation.

**Project 4: Inventory Management System with ASP.NET Core**

**Objective:**

Create a web-based inventory management system to track products, suppliers, and stock levels.

**Description:**

The inventory management system will allow businesses to manage their products, suppliers, and inventory levels. Users will be able to add, edit, and delete products, categorize items, and track stock quantities. The system will feature a low-stock alert system that notifies the administrator when product levels fall below a specified threshold. The backend will be developed using ASP.NET Core with Entity Framework for database operations, and the frontend will use MVC with user authentication for secure access.

**Technologies to use:**

ASP.NET Core, MVC, Entity Framework, SQL Server, User Authentication.

**4-Week Plan:**

**Week 1: Setup and Product Management**

* **Set up project environment:** Install and configure ASP.NET Core with MVC and Entity Framework.
* **Database schema design:** Create the database schema for products, suppliers, categories, and stock levels.
* **Product management page:** Implement functionality for adding, editing, and deleting products, including assigning categories and suppliers.
* **User authentication setup:** Implement basic authentication with role-based access for administrators and staff.

**Deliverables:**

* ASP.NET Core project set up.
* Database schema designed for products, suppliers, and stock.
* Basic product management page implemented.
* User authentication with role-based access control.

**Week 2: Supplier Management and Stock Tracking**

* **Supplier management:** Implement a page where administrators can manage supplier details (e.g., contact info, supplier products).
* **Stock tracking:** Create functionality to track product stock levels and display real-time stock counts.
* **Low-stock alerts:** Implement a system that notifies the administrator when stock levels are low.
* **Testing:** Test product and stock management functionalities.

**Deliverables:**

* Supplier management system implemented.
* Real-time stock tracking for inventory.
* Low-stock alert system working.
* Tested product and stock management.

**Week 3: Reporting and Dashboard**

* **Inventory reporting:** Implement reporting features to allow administrators to generate reports on stock levels, products, and suppliers.
* **Dashboard:** Develop a dashboard to give an overview of the inventory system, including stock levels, low-stock items, and product categories.
* **Testing:** Test reporting and dashboard functionalities to ensure accuracy.

**Deliverables:**

* Inventory reporting functionality implemented.
* Admin dashboard with real-time stock and product overviews.
* Tested reporting and dashboard features.

**Week 4: Final Testing, UI Enhancements, and Deployment**

* **UI enhancements:** Improve the user interface with better navigation, responsiveness, and modern design elements.
* **Final testing:** Conduct full testing of all functionalities, including product, stock, and supplier management.
* **Deployment:** Deploy the system on a hosting platform (Azure, AWS, or local IIS).
* **Documentation:** Complete the documentation for the project, including a setup guide and user instructions.

**Deliverables:**

* Fully functional and tested inventory management system.
* Deployed system with a live URL.
* Complete project documentation.

**Project 5: Real Estate Property Management System with ASP.NET Core**

**Objective:**

Create a web application for managing real estate properties, tenants, and leases.

**Description:**

This project focuses on building a property management system that helps real estate companies manage their properties, tenants, and lease agreements. Property managers can use the system to add new properties, track tenant information, and manage lease terms. Tenants can log in to view their lease details, make rent payments, and report issues with their properties. The system will feature user authentication with role-based access control, ensuring that property managers and tenants have different permissions.

**Technologies to use:**

ASP.NET Core, MVC, Entity Framework, SQL Server, User Authentication.

**4-Week Plan:**

**Week 1: Setup and Property Management**

* **Set up project environment:** Install and configure ASP.NET Core with MVC and Entity Framework.
* **Database design:** Create the database schema for properties, tenants, and lease agreements.
* **Property management page:** Implement functionality for adding and managing property details (e.g., location, size, price).
* **User authentication setup:** Implement user authentication with role-based access for property managers and tenants.

**Deliverables:**

* ASP.NET Core project set up.
* Database schema for properties, tenants, and leases created.
* Property management page implemented.
* User authentication with role-based access control.

**Week 2: Tenant Management and Lease Agreements**

* **Tenant management:** Create functionality for managing tenant information (e.g., personal details, lease term, payment history).
* **Lease agreements:** Implement functionality to create, edit, and terminate lease agreements.
* **Testing:** Test tenant management and lease agreement features to ensure accuracy.

**Deliverables:**

* Tenant management system implemented.
* Lease agreement management functionality.
* Tested tenant and lease management system.

**Week 3: Rent Payments and Reporting**

* **Rent payments:** Implement a rent payment system for tenants, allowing them to make payments online.
* **Financial reporting:** Develop reporting features for property managers to view rent payments, outstanding payments, and tenant activity.
* **Testing:** Ensure that the rent payment system and reporting features work as expected.

**Deliverables:**

* Functional rent payment system for tenants.
* Financial reporting system for property managers.
* Tested rent payment and reporting features.

**Week 4: Final Testing, UI Enhancements, and Deployment**

* **UI enhancements:** Improve the user interface with better navigation and a responsive design.
* **Final testing:** Conduct thorough testing of all features (property management, tenant management, payments, leases).
* Deployment: Deploy the real estate management system to a hosting platform.
* Documentation: Complete project documentation, including setup and user manuals.

Deliverables:

* Fully functional and tested real estate management system.
* Deployed system with a live URL.
* Complete project documentation.