CDAC MUMBAI

Module: MS .NET

Assignment-2 (Basic MVC Architecture)

---

1. Online Bookstore Application

Problem Statement:
Create an ASP.NET MVC Core application for an online bookstore.

- **Architecture**: Implement the standard MVC architecture.
- **Folder Structure**: Ensure proper organization of `Models`, `Views`, and `Controllers`.
- **Controllers and Actions**:
  - Create a `BookController` with actions for listing books (`HttpGet`) and adding a new book (`HttpPost`).
  - Use the `NoAction` attribute to exclude a private helper method from being invoked.
- **Views**: Create Razor views to display books and a form for adding a new book.
- **Models and Validation**:
  - Define a `Book` model with properties like `Title`, `Author`, `Price`, and `PublishedDate`.
  - Use **Data Annotations** for validation (e.g., `[Required]`, `[Range]`).

2. Student Registration Portal

Problem Statement:
Develop a student registration portal using ASP.NET MVC Core.

- **Controllers and Actions**:
  - Create a `StudentController` with actions for registering a student and listing all students.
  - Use `HttpGet` for the registration form and `HttpPost` to submit data.
- **Views**:
  - Create a Razor view for the registration form.
  - Use **HTML Helper Functions** for input fields (`TextBoxFor`, `DropDownListFor`).
- **Models and ViewModel**:
  - Create a `Student` model with properties like `Name`, `Age`, `Class`, and `Email`.
  - Create a `StudentViewModel` to include additional data like a list of available classes.

3. Contact Us Page with Validation

Problem Statement:
Build a "Contact Us" page for a website using ASP.NET MVC Core.

- **Controllers and Actions**:
  - Create a `ContactController` with actions to display the contact form and process the submission.

- **Views**:
  - Use Razor views to design the contact form.
  - Include fields for `Name`, `Email`, `Subject`, and `Message`.
- **Validation**:
  - Use **Data Annotations** for validation, ensuring required fields and proper email formatting.
  - Display validation errors using Razor.
- **ViewBag**:
  - Use `ViewBag` to display a confirmation message upon successful submission.

4. Product Management Dashboard

Problem Statement:
Create a product management dashboard to add, edit, and list products.

- **Controllers and Actions**:
  - Create a `ProductController` with actions for CRUD operations.
  - Use appropriate `HttpGet` and `HttpPost` attributes.
- **Models and Views**:
  - Define a `Product` model with properties like `ID`, `Name`, `Price`, and `Stock`.
  - Create Razor views for listing products and forms for adding/editing products.
- **ViewBag**:
  - Use `ViewBag` to display dynamic headings (e.g., "Edit Product" or "Add New Product") in views.

5. User Login and Registration System

Problem Statement:
Develop a user authentication system for login and registration.

- **Controllers and Actions**:
  - Create an `AccountController` with actions for `Register`, `Login`, and `Logout`.
- **Models**:
  - Create `User` and `LoginViewModel` classes.
- **Views**:
  - Use Razor to create login and registration forms.
  - Include validations using **Data Annotations** for fields like `Email`, `Password`, and `ConfirmPassword`.
- **Validation**: Ensure all fields are validated and display errors using Razor.