UNIVERSITY OF THE PUNJAB

BS (SE) Fall 2022 Morning

Web Engineering

Lab 04

Time Duration: 2 hr, Total Marks: 40

Objective:

The purpose of this lab is to provide hands-on experience with **Model-View-Controller (MVC)** architecture in ASP.NET Core. You will learn how to:

1. Understand the MVC design pattern.
2. Create controllers, views, and models.
3. Use ViewBagand ViewModelfor data passing.
4. Handle form submissions using **GET** and **POST** methods.
5. Display data in an HTML table.
6. Understand the role of HTML forms in MVC applications.

Marks Division:

- **Controller Implementation** – 10 marks
- **Views Implementation** – 5 marks for each view
- **Model Implementation** – 5 marks for each model

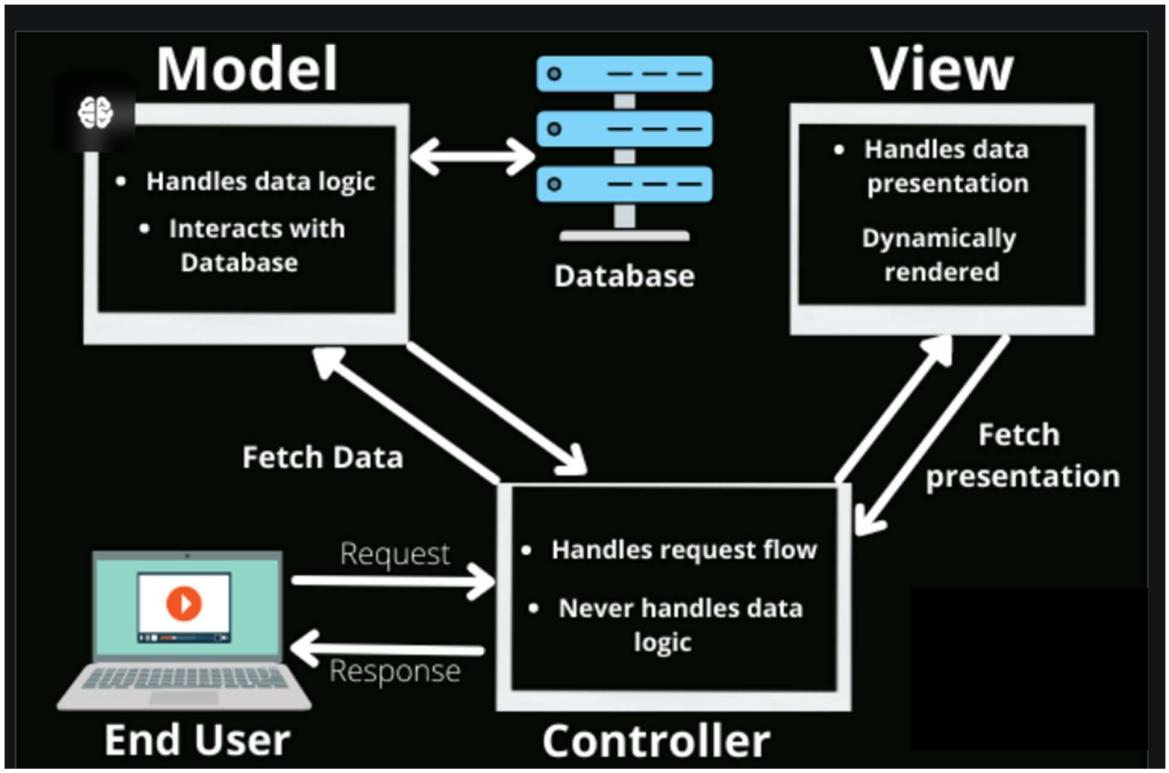
1. MVC Overview

The **Model-View-Controller (MVC)** pattern is a software architecture pattern used for developing web applications. It separates the application into three interconnected components:

1. **Model**: Represents the data and the business logic of the application.
2. **View**: Represents the UI (User Interface) of the application. It is responsible for displaying data.
3. **Controller**: Acts as an intermediary between the Model and the View. It handles user input, updates the Model, and selects the appropriate View for display.

MVC Diagram:

Below is a basic representation of how MVC works:



In this diagram:

- The **User Request** (typically from a browser) triggers the Controller to process the data.
- The **Model** contains the business logic and data storage (often a database).
- The **View** is used to display the data to the user in a formatted manner.

Create a Basic MVC Project

Objective:

- **Set up a new MVC project** using ASP.NET Core in Visual Studio.
- Create **Controllers**, **Views**, and **Models** for basic interaction.

Steps:

1. Open **Visual Studio** and create a new ASP.NET Core project using the **Model-View-Controller (MVC)** template.
2. Configure the project name (e.g., **MvcLab**) and set up the required configurations.
3. Run the project and access the default home page to ensure it is working.

Task 1: Create a Controller with Views

Objective:

- Create a **Home Controller** with multiple actions to demonstrate the **View** and **Controller** interaction.

Steps:

1. In the **Controllers** folder, add a new **HomeController** class.
2. Define basic actions like Indexand Aboutin the controller.
3. Create corresponding views for these actions inside the **Views** folder:
    - Views/Home/Index.cshtml
    - Views/Home/About.cshtml
4. Use **ViewData** or **ViewBag** to pass data from the controller to the view.

Expected Outcome:

- You will be able to define multiple actions in the controller and render views based on the requests.

Task 2: Using ViewModels

Objective:

- Use **ViewModel** to pass multiple pieces of data from the controller to the view.

Steps:

1. Create a **HomeViewModel** class that holds different properties (like a title and a list of items).
2. In the controller, instantiate this ViewModel and pass it to the view.
3. In the view, use **@model** to access the data passed through the ViewModel and display it.

Expected Outcome:

- A well-structured way to pass complex data objects from the controller to the view using ViewModels.

Task 3: Understand the Use of GET and POST in Forms

Objective:

- Understand the use of **GET** and **POST** methods in MVC forms.

Steps:

1. Create a form in the view with input fields like **username** and **email (**handle the input validations correctly**)**.
2. Set the **method** attribute of the form to **POST** so that when the form is submitted, it sends data to the server.
3. In the controller, add two methods:
    - One to **GET** the form.
    - One to **POST** the form data and process it.

Expected Outcome:

- You will gain a clear understanding of **GET** and **POST** and how they affect the form submission process.

Task 4: Creating a Product List with a Table

Objective:

- Display a **list of products** in an HTML table, showcasing data passed from the controller to the view.

Steps:

1. Create a **Product** model with properties like **ProductName** and **Price**.
2. Create a controller action that fetches a list of products from the model.
3. In the view, loop through the product list and display the data in an HTML table.

Expected Outcome:

- You will understand how to pass collections of data from the controller to the view and display them in an HTML table.

Task 5: Order Item Form in MVC

Objective:
Create an Order Item form using the MVC architecture.

Instructions:

1. **Model (OrderItem Model):**
   Create a class `OrderItem` in the `Models` folder with the following properties:
   - `FoodItem` (string)
   - `Quantity` (int)
     Add a constructor to initialize the properties.
2. **Controller (OrderController):**

- Create a controller named `OrderController` in the `Controllers` folder.
- Define a `GET` action named `Order` that returns the view with an empty `OrderItem` model.
- Define a `POST` action to handle the form submission and display a confirmation message.

3. **View (Order View - Order.cshtml):**
   Create the view `Order.cshtml` for displaying the form to place an order using basic HTML structure.