Assignment Basic Level

B1. What is model?

- ⇨ Model is represents the shape of data.
- ⇨ Model object store data retrieved from the database.
- ⇨ Model is represents shape of the data as public properties and business logic as methods.

B2. What is routing in MVC?

- ⇨ Routing is responsible for handle the route our application.
- ⇨ Set the route first page execute.
- ⇨ Routing is a pattern matching system.
- ⇨ Routing is maps incoming request form browser to a particular resource (controller and action methods)

B3. What is the difference between TempData, ViewData, and ViewBag?

⇨

TempData	ViewData	ViewBag
It is a key-value dictionary collection.	It is a key-value dictionary collection.	It is a type object.
It helps to maintain data one controller to another controller or one action to another action.	Used to pass data from controller to corresponding view.	Used to pass data from controller to corresponding view.
It is derived from TempDataDictionary class.	It is derived from ViewDataDictionary class.	It is a dynamically container.
TempData required typecasting for complex data type and check for null value to avoid error.	ViewData required typecasting for complex data type and check for null value to avoid error.	ViewBag is does not require typecasting for complex data type.

B4. What is difference between MVC and Web Forms?

⇨

ASP .Net MVC	ASP .Net Web Forms
It is a lightweight and follow MVC (Model, View, and Controller) pattern based development.	It is work on event based.
MVC uses Html controls it uses Html helper.	Web form uses server side controls.
MVC does not maintain the viewstate, so pages are lightweight.	Web form uses server side controls so it maintain the viewstate at client side
MVC has route based URL means URL is divided into controller and event.	Web form work on file based URL means URL file must be physically available.
MVC uses Layouts for Consistent look and feel.	Web from is uses master pages for consistent look and feel.
MVC has partial view for code reusability.	Web form is a user control for code re-usability.
In MVC view and logic are kept separately not tightly coupled.	In web form, web form (ASPX) UI are related code behind (ASPX.cs) for logic, it is tightly coupled.

B5. What is session? What is the default time for session?

- ⇨ Session is state management Technique. Created user in login a user can perform any authorized task until the session in alive.
- ⇨ Session is valid for all request, not for a single redirect.
- ⇨ It is also require type casting for get data and check for null value to avoid error
- ⇨ The default time for session: 20 minutes.
- ⇨ This time is increase and decrease.
- ⇨ Session has performance drawback because it slow down the application that's why it is not recommended to always use session, session can be used according to the situation.

Assignment Intermediate Level

I1. What is Partial View in MVC? With example.

- ⇨ Partial view in MVC is special view which renders a portion of view content.
- ⇨ It is just like control of a web form application.
- ⇨ Partial can be reusable in multiple views.
- ⇨

I2. What is the difference between View and Partial View?

- ⇨

View	Partial View
View contains the layout page.	Partial view does not contain the layout page.
_viewstart page is rendered before any view is rendered.	Partial view does not check for layout page.
View may have markup tags like html, body, head, title, etc.	The partial view is specially designed to render within the view and as a result it does not contain any markup.

- ⇨ Partial view is more lightweight than the view. We can also pass a regular view to the RenderPartial method.

I3. Explain the concept of MVC Scaffolding?

- ⇨ MVC Scaffolding is automatic generate code in controller.
- ⇨ Add scaffolding to your project when you want quickly add code that interacts with data models.
- ⇨ Scaffolding can reduce the time to developer.

I4. How to change time of session?

- ⇨ Change time of session add sessionState time out in web.config file under the system.web section.
- ⇨ <sessionState timeout = “60” mode = “InProc”>
- ⇨ Default time is 20 minutes. Replace 60 minutes set time as you wish.
- ⇨ Second method is session start method. Set the timeout property of session to the required.
- ⇨ Session.Timeout=60;

I5. What is query string? What are disadvantages of query string?

- ⇨ Querystring is use to transfer information from one page to another page through the URL.
- ⇨ QueryString is attached to the URL with "?".
- ⇨ Disadvantages :
- ⇨ All the attributes and values are visible to the end user. Therefore, they are not secure.
- ⇨ There is a limit to URL length of 255 characters.

I6. What is cookie? What are limitations for cookie?

- ⇨ Cookies is one of the state management technique, so that we can store information in browser for late use.
- ⇨ Cookies are easy to use. Use both request and response object are use throw a cookies collection.
- ⇨ No server resource are require as they store the client cookies in browser.
- ⇨ There are two type of cookies:
- ⇨ 1. Temporary cookies save in browser memory. The expire of temporary cookies by default time 30 minutes.
- ⇨ 2. Persistent cookies are save in physical memory. To define lifetime of persistent cookies. It can set expire time.

- ⇨ Limitation:
- ⇨ The main drawback is the privacy for most users.
- ⇨ The cookie enabled web browsers keep track of all the websites you have visited.
- ⇨ Cookies can be tampered and thus creating a security hole.

I7. Create one example to store data in session and show on other view.

⇨ Login view page.

```
@using (Html.BeginForm())
{
   @Html.AntiForgeryToken()
   <div class="form-horizontal">
      <h4>Login Page</h4>
      <hr />

      <div class="form-group">
         @Html.LabelFor(model => model.username, htmlAttributes: new {
@class = "control-label col-md-2" })
         <div class="col-md-10">
            @Html.TextBox("user", "", new { @class = "form-control",
@placeholder = "Enter Username", @onchange = "setPass()" })
            <p style="color:red">@ViewBag.invaliduser</p>
         </div>
      </div>

      <div class="form-group">
         @Html.LabelFor(model => model.password, htmlAttributes: new {
@class = "control-label col-md-2" })
         <div class="col-md-10">
            @Html.TextBox("pass", "", new { @class = "form-control", @type =
"password", @placeholder = "Enter Username" })
            <p style="color:red">@ViewBag.invalidpass</p>
         </div>
      </div>
      <div class="form-group">
         <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Log In" class="btn btn-default" />
         </div>
      </div>
   </div>
}
```

⇨ Controller page:

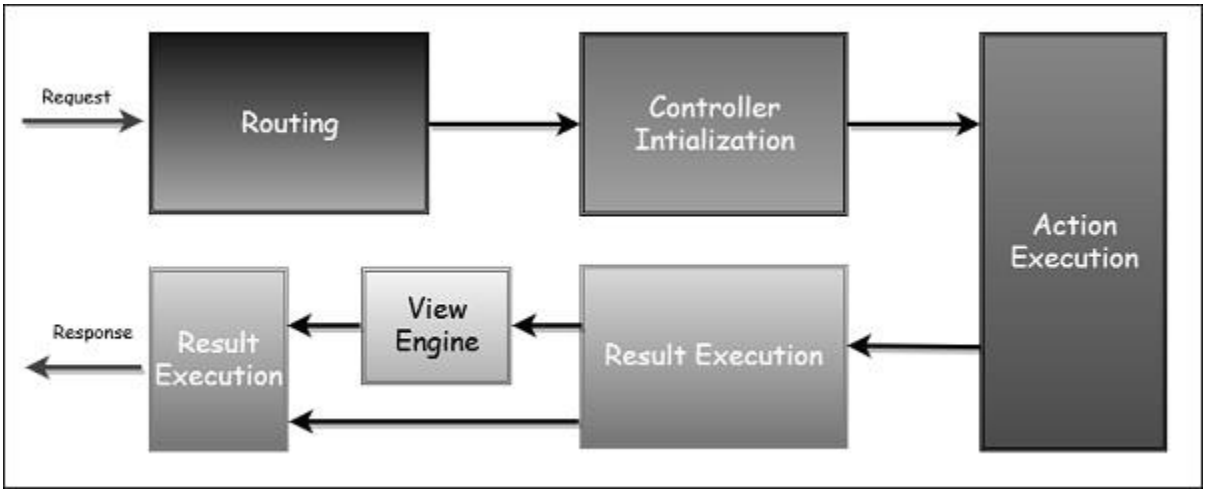
```
[HttpPost]
        public ActionResult login(FormCollection fc)
        {
            var username = fc["user"].Trim();
            var userpassword = fc["pass"].Trim();

            HttpCookie userInfo = new HttpCookie("UserInformation");
            userInfo["COusername"] = username;
            userInfo["COpassword"] = userpassword;
            userInfo.Expires = DateTime.Now.AddMonths(1);
            Response.Cookies.Add(userInfo);

            var lg = con.tblLogins.FirstOrDefault();

            if (lg.username == username || lg.password == userpassword)
            {
                Session["Suser"] = username;
                return RedirectToAction("displayStudent");

            }
            else
            {
                ViewBag.invaliduser = "* Invalid username...!!!";
                ViewBag.invalidpass = "* Invalid password...!!!";
            }
            return View();
        }
```

⇨ Get session values:

```
@if (Session["Suser"] != null)
        {
            <text>Welcome @Session["Suser"].ToString()</text>
        }
        else
        {
            Url.Action("login", "Test");
        }
```

Assignment Advanced Level

A1. Explain MVC application life cycle.



- ⇨ The **routing** module is responsible for matching the incoming URL to routes we define in our application.
- ⇨ All routes have an associated **route handler** with them and this is **entry point** to the **MVC framework**.

- ⇨ The MVC framework handles **converting** the route data to concrete controller that can handle request.

- ⇨ After the controller has been created, the next major step **Action execution.**
- ⇨ A controller call action invoker finds and select action methods to invoke controller.

- ⇨ After our action result are prepared, then next stage, which is **Result Execution**. MVC separates declaring the result form executing the result.

- ⇨ If results view type, the view engine will be call and response for find and return view.

- ⇨ If the result is not view type, the action results will execute on it is own.

- ⇨ This results execution is what generate actual response to the original request.

A2. List out different return types of a controller action method.

- ⇨ System.Object
- ⇨ System.Web.Mvc.ActionResult
- ⇨ System.Web.Mvc.ContentResult
- ⇨ System.Web.Mvc.EmptyResult
- ⇨ System.Web.Mvc.FileResult
- ⇨ System.Web.Mvc.HttpStatusCodeResult
- ⇨ System.Web.Mvc.JavaScriptResult
- ⇨ System.Web.Mvc.JsonResult
- ⇨ System.Web.Mvc.RedirectResult
- ⇨ System.Web.Mvc.RedirectToRouteResult
- ⇨ System.Web.Mvc.ViewResultBase

A3. What are filters in MVC?

- ⇨ Filters in MVC there might be situation where you need to implementation **some functionality before and after the execution** of action methods.
- ⇨ In such situation, need to use **Filters**.
- ⇨ There are 5 type of Filters in MVC:
    1. Authentication filters
    2. Authorization filters
    3. Action filters
    4. Result filters
    5. Exception filters
- ⇨ All type of filters use in an action method, their sequence of execution.
- ⇨ We can use in-built filters and custom filters in MVC.
- ⇨ We can use filters at the 3 level:
    1. Action level
    2. Controller level
    3. Application level

A4. What are HTML helpers in MVC?

- ⇨ HTML helpers are methods.
- ⇨ These returns html string.
- ⇨ These are used on view.
- ⇨ In simple terms these are c# methods which are used to return HTML.
- ⇨ Using HTML helpers you can render a TextBox, a TextArea, Image tag etc.
- ⇨ MVC has built in HTML helpers.
- ⇨ We can create custom helper also.
- ⇨ Using HTML helpers a view can show model properties and can generate html as per the types of properties.

⇨ **Type of HTML helpers:**

1. Inline HTML helpers
2. Built-in HTML helpers
   - Standard HTML helpers
   - Strongly typed HTML Helpers
   - Template HTML helpers
3. Custom HTML helpers

A5. Differences between Razor and ASPX View Engine in MVC?

⇨

Razor View engine	ASPX View Engine
Razor view engine is an advanced view engine that was introduced with mvc3. It is not a new language but it is new markup syntax.	Aspx view engine is a default view engine for the asp.net mvc that is included with asp.net mvc from beginning.
The namespace for Razor engine System.Web.Razor	The namespace for aspx engine System.Web.Mvc.WebFormViewEngine
The file extensions use with razor engine are .cshtml (for c#) and vbhtml (for VB) extension for views, partial views, layout pages.	The file extension use with ASPX engine are .aspx for views, .ascx for partial views, .master for layout/master pages.
Razor has new and advance syntax that are compact, expressive and reduces typing.	ASPX engine has the same syntax like asp.net web form uses for .aspx pages.
Razor syntax easy to learn and clean than aspx syntax.	Aspx syntax are borrowed from asp.net web forms syntax that mixed with html and sometime make a view messy.
Razor uses @ symbol to make code @Html.ActionLink(“Create new”,”create”)	ASPX use <% and %> symbol to make code <%: Html.ActionLink(“Create new”,”create” %>
Razor engine slow compare to ASPX engine.	ASPX engine faster than Razor engine.

A6. Pass one values from one view to another view using query string.
View 1:

```
<script>
    function send() {
        var name = $("#txtname").val();
        window.location.href = '/Home/queryString2/?name=' + name;
    }
</script>
<h2>queryString1</h2>

@using (Html.BeginForm())
{
    <div class="form-horizontal">
            <div class="form-group">
                <div>
                    <label>Name</label>
                </div>
                <div class="col-md-10">
                    @Html.TextBox("txtname", "",new { @class = "form-control", @type =
"text" })
                </div>
            </div>
            <br />
            <input type="button" id="btn1" value="Send" onclick="send()" class="btn btn-
success"/>

    </div>
}
```

View 2:

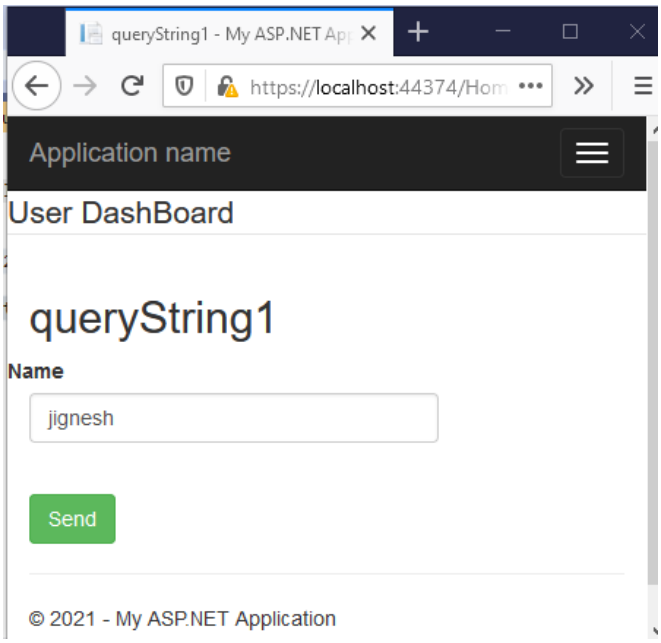
```
       <h2>queryString2</h2>

       @Request.QueryString["name"]
```

Controller :

```
public ActionResult queryString1()
        {
            return View();
        }

        public ActionResult queryString2()
        {
            return View();
        }
```

Output:

View1



View2