**Ticket Sale project**

This project contains 3 Controllers, 5 Models and 6 Views which are set in 3 folders based on the action method names in the controllers associated with the razor views.

Project Output contains links for three pages named Home, Events, AboutUs page which as set in Layout.cshtml available in the shared View.

Home and AboutUs page are from 1 controller i.e. Home Controller and Model , Views associate with are Category and Index.cshtml , AboutUs.cshtml views respectively.

Events page as two controllers associated with it they are Cart and Events Controllers and remaining 4models and 4views.

Home page Just presents the details about the project and about us page contain id and Name the default route in program.cs is Homecontroller Index action method. The properties required for the Aboutus page are obtained from category model and binding it the aboutus.cshtml view with the action method invovked in the homecontroller i.e. AboutUs action method.

The main page of the project is Events.

Events

Events Controller

The EventsController in the provided ASP.NET Core MVC application handles the retrieval and display of event-related data to users. It contains two primary action methods: EventList and Details. The EventList method, decorated with the [ResponseCache(NoStore = true, Duration = 0)] attribute to prevent caching, fetches a list of events from an EventService. If no category ID is provided, or if the ID is "All", it retrieves all events. If a specific category ID is provided, it filters the events based on the category. This is done by first finding the corresponding category ID and then selecting events that match this category. The resulting list of events, along with categories and the current category ID, is encapsulated in a ListViewModel and returned to the view for rendering.

The Details method, also decorated with the same cache prevention attribute, retrieves details of a single event based on its ID. It instantiates the EventService, fetches the event using the provided ID, and returns the event object to the view. This allows users to see detailed information about a specific event. Both methods rely on the EventService to interact with the data source and ensure that the data presented to the user is current and not cached, promoting accurate and up-to-date information.

Models

EventServiceModel

The EventService as EventService class provides methods to handle events and categories. It initializes two lists: \_allEvents containing instances of the Events class, and \_allCategories containing instances of the Category class. The GetEvents method returns a specific event based on the provided event ID, the GetCategories method returns the list of all categories, and the GetAllEvents method returns the list of all events.

ListViewModel

The ListViewModel class serves as a view model that aggregates events and categories to be displayed in the view. It has three properties: Events, which holds a collection of Events objects; Categories, which holds a list of Category objects; and SelectedCategory, which stores the name of the currently selected category. The constructor of ListViewModel takes these three properties as parameters, allowing the controller to pass the necessary data to the view.

Views

The first view is a list view that uses the ListViewModel model. It starts with a link to view all events and then dynamically generates links for each event category available in Model.Categories. When a category link is clicked, it filters the events by that category. The view then displays a table with a header showing the selected category and a list of events under that category. Each event is listed with its title and a "View Details" link, which navigates to the details view for the selected event.

The second view is an event details view that uses the Events model. It displays detailed information about a single event, including an image, title, description, and ticket price formatted as currency. There are two links below the events : one to initiate the ticket purchasing process for the event by navigating to the Buy action of the Cart controller, and another to go back to the event list view. This allows users to easily navigate between the list of events and the details of a specific even.

Purchase Ticket Page

When the user select the event and wants to buy a ticket they click on the link present below particular event then it redirects into different page which has 7 input fields which as are of type textbox for eventname, selectdrop down for deliverymode, number text box for price, email for email, check box for senior discount , integer dropdown for number of tickets, submit button for purchase ticket button. When the submit button is clicked it submits to different page which explains that there is a FORM and 2 views named Buy and Confirmation associated with it.

The Controller is Cart Controller and Model is Buy Tickets Model. The Model contains all the properties and 2 Overloded Constructors and 2 methods to calculate AmountDue and Calculate Discount for senior citizens. The first constructor named as BuyTickets is used for Model binding it is the default constructor is called by default when page loads and the overloaded constructor has different signature such as eventname and ticketprice this values are set by the controller buy action method.

CalculateDiscount() method helps to calculate sales discount for seniors. CalculateAmountDue() method is used to calculate total amount due including if the user is senior the discount apply and if deliverymode is mail there is additional charge.

Cart Controller

It contains 2 action methods named Buy and Cofirmation.

Buy Action method is handles the initial request for the purchase page where it create instance for eventservice to get the eventdetails. Selectedevents helps to retrieve the event details by id using GetEvents() method which invovkes events list. Buyticket object help to set values to the parameters of the overloaded constructor in the model. It return the viewmodel for the buy.

Confrimation action method handles the form it has object of buytickets model which is validated to satisfy the properties of the model using ModelState.Isvalid and moves inside the if the block and object calls the CalculateAmountDue () method and return the view or else it returns and view of buyticket to user.

Views

Both views use BuyTicketModel

Buy View

The view uses buytickets model and Form and table tags to have a structured format for input fields in the BuyPage. And provides input for all properties of model and binds it to the input field of the view and 2 properties as readonly format as those input fields need to retrieve the date from previous selection of events and whose value are set by overloaded constructor.

Confirmation View

It dispalys details of the ticket for particular event priced including Order date and time.