**PullaiahgariTicketSales**

1.This Application follows Model-View-Controller (MVC) pattern, The application enables users to browse events, read event details, and make ticket purchases.

2.This application consists of 5 Models, 3 Controllers & their respective views. Here is the detailed explanation for all of them:

**Models:**

These classes contain the data and business logic of the application. The following items are included:

* **BuyTickets:** This model covers procuring a ticket, storing information on the event, customer details, and the computed total cost of the order. The system contains data annotations to validate input and ways for calculating discounts and the total amount payable.
* **Category & Event:** Primary models that describe categories & events, respectively.
* **EventsService:** A class that serves as a data store for events and categories, providing methods for retrieving data. For the production environment, it is common to replace this with a database connection.
* **ListViewModel:** The *EventList* view uses a composite model that stores data, containing events, categories, and selected category information.

**Views:**

The control of displaying the user interface by utilizing Razor syntax and ASP.NET Core Tag Helpers:

* **Buy.cshtml:** Provides a form for people to fill out with their details and buy tickets. It uses ASP.NET Core Tag Helpers to connect form fields to the *BuyTickets* model, which makes it easier for the view and controller to share data.
* **Confirmation.cshtml:** Shows a short confirmation page summarizing the ticket buy when it's done successfully.
* **Details.cshtml:** Provides thorough information about a chosen event together with a ticket pricing and a buying option.
* **EventList.cshtml:** Utilizes data from the *ListViewModel* to display a list of events that are currently available, with the option to filter by category.
* **About.cshtml & Index.cshtml:** They function as static content views that offer fundamental information about the application and its creator.

**Controllers:**

Manage the application's logic and facilitate interaction between models and views:

**1.CartController:**

* The function *Buy(int id)* retrieves the event with the specified ID, creates a *BuyTickets* object, and displays the *Buy* view for purchasing tickets.
* The *Confirmation* function in the *BuyTickets* model validates the user input, computes the total cost, and shows the Confirmation view if the purchase is completed successfully.

**2.EventsController:**

* *EventList(string id = "All")*: Gathers events according to the selected category or all events. It builds the *EventList* view and a *ListViewModel*.
* Renders the Details view and retrieves a particular event with *Details(int id)*.

**3.HomeController:**

* Return the corresponding static views for Index and About.

**Data Flow & Request Processing :**

* **User Interaction:** The user initiates an action, such as clicking a link or submitting a form.
* **Request Routing:** The application determines the appropriate controller and action method based on the user's request.
* **Controller Processing:** The controller retrieves data from models, performs any necessary logic (calculations, validations), and selects the appropriate view.
* **View Generation:** The view renders the HTML for the page, incorporating the data provided by the controller.
* **Response and Display:** The generated HTML is sent back to the user's browser, which then displays the resulting page.