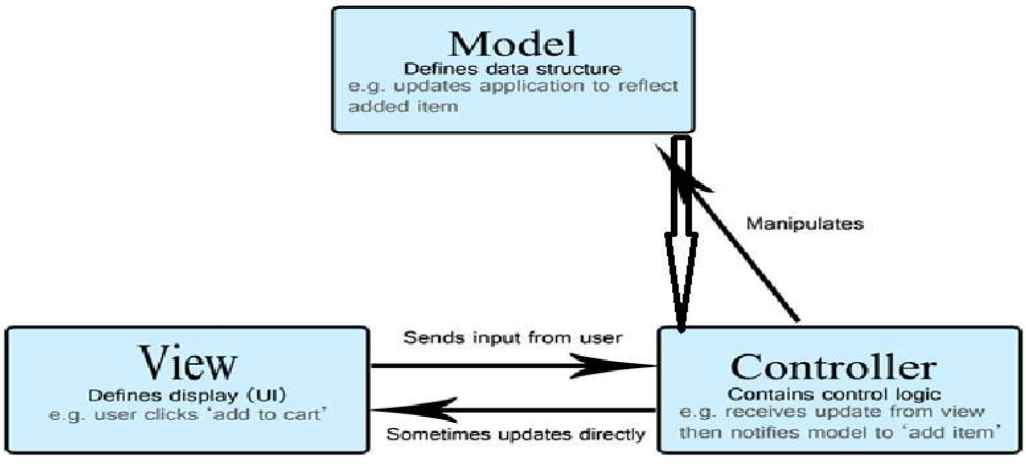
The flow of how ASP.NET MVC code works would be as follows:

1. A client (such as a web browser) sends an HTTP request to the server. The request includes information such as the URL, HTTP method (GET, POST, etc.), and any data that needs to be sent to the server.
2. The server receives the request and passes it to the ASP.NET runtime. The ASP.NET runtime is responsible for processing the request and generating a response.
3. The ASP.NET runtime uses routing rules to determine which code should handle the request. In an MVC application, routing rules are used to match the requested URL to a specific controller and action method.
4. Once the appropriate controller and action method have been identified, the ASP.NET runtime passes the request to the controller. The controller is responsible for handling the request and generating a response.
5. The controller interacts with the model to retrieve or process data. The model represents the data, it can be a simple class or a more complex system such as a database or a web service.
6. Once the controller has the necessary data, it passes the data to the view. The view is responsible for displaying the data to the user. The view can be an HTML template or a Razor file.
7. The view generates the HTML or other content that will be sent back to the client as a response.
8. The ASP.NET runtime takes the response generated by the view and sends it back to the client.



Views are typically organized into folders based on their purpose and relationship to the controllers and models. The standard folder structure for views in an ASP.NET MVC application is as follows:

1. Views: This is the top-level folder that contains all of the views for the application.
    a. Views/{ControllerName}: This folder contains views that are specific to a particular controller. Each controller will have its own folder within the Views folder, and views that are associated with that controller will be placed in that folder.
    b. Views/Home: This folder contains views that are specific to the HomeController.
2. Views/Shared: This folder contains views that are shared across multiple controllers and actions. Examples of views that are typically placed in the Shared folder include the layout view, error views, and partial views.

The separation of views into different folders makes it easy to locate the views that are associated with a specific controller or action and also make it easy to manage and maintain the views. The Shared folder contains views that are shared across multiple controllers and actions, such as layout view, error views, and partial views. This separation allows for easy modification and management of views.

Created database and the code to generate models automatically using Data first approach :

Scaffold-DbContext "Server=(localdb)\mssqllocaldb;Database=TestManagementDB;Trusted_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models

The MVC pattern can utilize a view model to separate the concerns of the user interface from the underlying data and business logic. In the MVC pattern, a view model is typically a C# class that is used to hold data that is required by a view. The view model is used to prepare the data for display in the view, and it can also handle any data manipulation or validation that needs to be done in response to user input. It's true that view model is not a mandatory part of the MVC pattern, but using it can make it easier to manage the data being passed between the controller and the view, and can help to keep the views more focused on presentation.