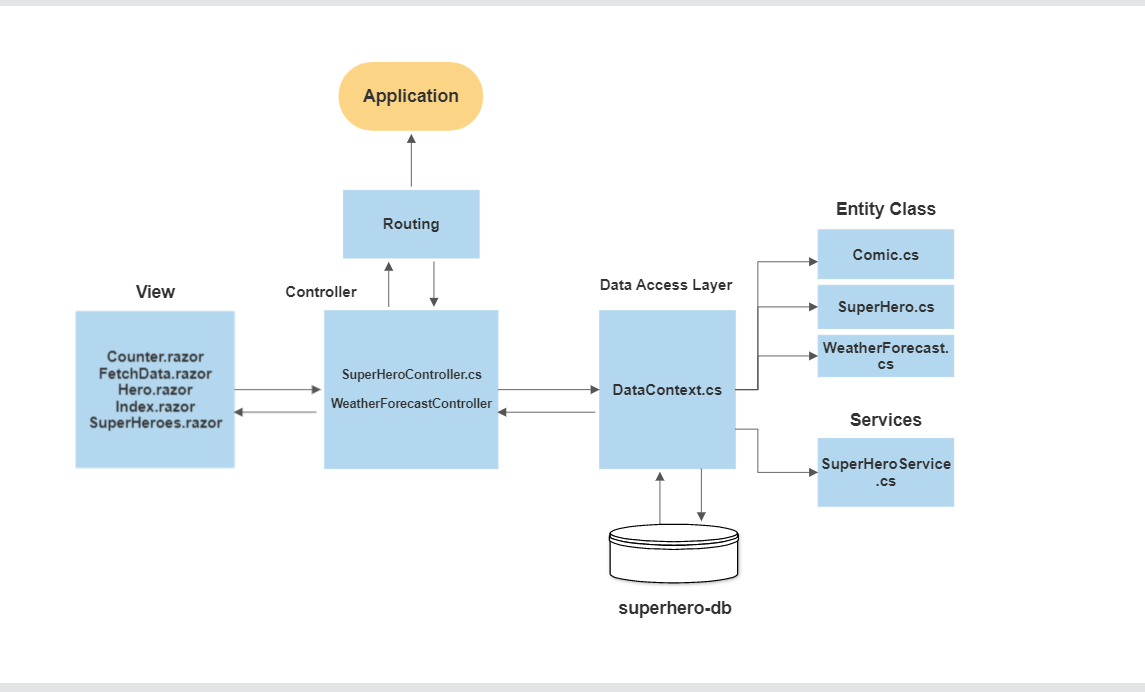
**View:** 

Provide a simple and clean syntax for combining HTML and C# code.

A Razor page is a standalone web page that contains both HTML markup and C# code, and is typically used to display and manipulate data. Razor Pages are a useful alternative to traditional MVC controllers that can contain code to redirect the client to a different URL or action method.

This is often used to implement login functionality or to redirect the client to a different page after a

form submission.(Model-View-Controller) applications particularly for smaller or simpler applications, or for developers who prefer a more intuitive and straightforward syntax.

* **Counter.razor:** Contain Html code and C# code for count increment.
* **FetchData.razor:** Contain Html code for weatherforecast table and c# method for add weather forecast data from **WeatherForecast.cs** class
* **SuperHeroes.razor:** Contain Html table for creating new user and c# code for performing the save and delete user.

**Routing:**

Routing in MVC is the process of determining which controller action to execute in response to an incoming request, based on the URL of the request and the routes defined in the application's routing table or configuration.

The **Route** attribute is an attribute used in ASP.NET MVC and ASP.NET Core to specify the route for a controller or action. When applied to a controller class, the Route attribute specifies the base route for all actions in the controller. When applied to an action method, the **Route** attribute specifies the route for that specific action.

[Route("api/[controller]")]

[ApiController]

public class SuperHeroController : ControllerBase

{

//action method go here

}

In this example, the **Route** attribute specifies that all action methods in the SuperHeroController controller should be accessible via the /api/SuperHero route. For example, the List action of the SuperHeroController would be accessible at /api/SuperHero/list.

public async Task<ActionResult<List<SuperHero>>> GetSuperHeroes()

{

//action logic goes here

}

public async Task<ActionResult<SuperHero>> GetSingleHero(int id)

{

}

**‘GetSingleHero’** action will be accessible at the /api/SuperHero/{id} route, where {id} is an integer parameter.

**Controller:**

* **SuperHeroesController :**

Controller contain Action methods:These are methods that are called in response to specific

requests from clients. Each action method typically represents a specific task or operation that the controller can perform.

In SuperHeroesController collect data list from **SuperHero.cs** Entity class.

Controllers can contain code to redirect the client to a different URL or action method.

This is often used to implement login functionality or to redirect the client to a different page after a form submission.

* **WeatherForecastController:**

‘WeatherForecastController’ this controller is used to display weather forecast information to users.

The **‘WeatherForecastController** ’ has a dependency on an **IWeatherForecastService** interface, which is used to retrieve weather forecast data from some external source. I.e WeatherForecast.cs.

‘**IWeatherForecastService** ’ is used to retrieve WeatherForecast data and then pass the data to the view for rendering.

**Data Access Layer :**

* **DataContext.cs:**

DataContext.cs class contains the Dbset of SuperHeroes and comics Classes.

And The Index action method retrieves the database list from SuperHeroes and comics classes.and perform the operation on that database.

* **Entity Classes :**

1. Comics.cs
2. SuperHero.cs
3. WeatherForecast.cs

* **Services:**

**SuperHeroService.cs:**

The ‘SuperHeroServices’ class provides a **‘CreateHero’** async method to create user data from an external API.

The **‘SuperHeroServices’** class depends on an **‘HttpClient’** object, which it uses to make HTTP requests to the API.

The **‘SuperHeroServices’** class implements the **‘ISuperHeroServices’** interface, which defines the contract for the service. This allows the service to be easily injected into other parts of the application, such as a controller, using dependency injection.

A SuperHeroservice is a class that provides a service or set of related services to other classes in the application. Services are typically used to encapsulate business logic for Create new User form and perform the operation on that form Ex. Create Delete, Update Or other reusable functionality that can be used by multiple parts of the application.

**Database (superhero-db):**

In superhero-db is used to store and retrieve data that is used by the application.

Database is totally considered to be part of Model Layer.

Model layer communicates with the database through the Data Access Layer.

i.e, DataContext.cs.