Let me summarise the highlights of the lectures covered in this section. In those lectures I have discussed

Project file is now largely trimmed with only a few necessary things like the core framework. All the files that were listed in the earlier version of the core framework have been done

away with. This project file holds the reference to the nuget packages and updates them as and when there is a change in the nuget package.

Next we discussed the Program file. It is where the infrastructure for the project is created.Program file starts the asp.net core application as a console application and is the entry point to

the application. The Main method calls another method that builds the webhost or web server on top of the consiole application through a concept called Generic Host Concept

The Program File then calls the Startup file

Startup file or the Startup class is a place where the all the custom and built -in services are registered and the middleware pipeline is built. The Startup class has two methods :

ConfigureServices() : It is this method where all the services, both framework built-in and custom services are registered. For example:

services.AddRazorPages();

services.AddTransient<ToDoService>();

Configure(): It is the place where all the important middleware (called middleware pipeline) is configured. Some of the important ones are static middleware that serves all the static

assets like html, css, images and the routing middleware that does the routing! There is also a developer exception page middleware that provides a different exception page in development

than when it is running on production.

Ok, @Page directive is the first directive on a Razor Page which indicates that it is a razor page. @Model directive is followed by the PageModel class for the razor page (that is derived from a base PageModel class)

which indicates that it is an MVC action that can handle a request directly without a controller action.

We have also seen that the PageModel class has page handler methods that sound similar to the HTTP GET/POST verbs : OnGet(), OnPost()

OnGet() page handler method comes with the default template and if it returns a void, the html response returned by the server to the browser, is the page itself. Otherwise the OnGet() handler can also return an IActionResult

type that may redirect the browser to another page.

The razor page with the .cshtml extension derives its power from the fact that it can accomodate both the static and dynamic html and also ordinary C# code. The PageModel class can be thought of as analogous to the

code behind file in the earlier Web Form applications in the legacy .Net Framework and maintains the separation of logic from the presentation