Write Up

- **Project Creation:**

You started by creating a new ASP.NET WebForms project in Visual Studio. You set up a connection to a SQL Server database named "School" on the local SQL Server instance ("SUNNYLAPPY\SQLEXPRESS").

- **Master Page (Main.master):**

You created a master page named "Main.master" using HTML and ASP.NET controls. The master page includes a header with a welcome message and a navigation menu using the ASP.NET Menu control. You used a SiteMapDataSource to provide a site map for navigation.

- **SiteMap:**

You defined a simple site map in "Web.SiteMap" to represent the hierarchical structure of your web application. It includes nodes for Home, Students, Subjects, and Classes

- **Student Page (Student.aspx):**

You created a web form for managing students ("Student.aspx") that uses the "Main.master" master page. The page includes a GridView control ("GridView1") to display data from the "Students" table in the database. The GridView is populated by the "BindGridView" method, which retrieves data from the "Students" table using a SQL query

- **Student Code-Behind (Student.aspx.cs):**

In the code-behind file, you wrote C# code to handle page events. The "Page_Load" event is used to bind data to the GridView when the page is loaded. The "GridView1_SelectedIndexChanged" event is available for handling row selection (though it's currently commented out).

• **Subjects Page (Subjects.aspx) and Code-Behind (Subjects.aspx.cs):**

You repeated a similar structure for managing subjects. The Subjects page ("Subjects.aspx") and code-behind file ("Subjects.aspx.cs") follow a pattern similar to the Student page.

• **Classes Page (Classes.aspx) and Code-Behind (Classes.aspx.cs):**

You did the same for managing classes, with the Classes page ("Classes.aspx") and code-behind file ("Classes.aspx.cs") mirroring the structure of the other pages.

• **Database Interaction:**

You used ADO.NET to interact with the SQL Server database. The connection string is hard-coded in each code-behind file, which may be a consideration for maintenance

• **Running the Application:**

After completing the development, you presumably ran the application to view and interact with the data through the web pages.

Step : Azure Virtual Machine (VM)

1. Go to the Azure Portal (https://portal.azure.com/).

2. Click "Create a resource" and search for "Virtual Machine."

3. Fill in the required details such as resource group, VM size, and hard drive type.

4. Provide administrator details like the administrator's name and password.

5. Click "Review + create" and, after validation succeeds, click "Create."

6. After the deployment succeeds, click "Go to resource," and then click "Connect." Download the RDP file and choose a file path for the download.

Step : Azure Web App

1. In the Azure Portal, create a Web App by selecting "Create a resource" and searching for "Web App."

2. Choose a resource group, provide the web app name, select the runtime stack, and choose the appropriate pricing plan for your subscription.

3. Click "Review and create," and then click "Create."

4. After the web app is created, click "Go to resource" and download the publish profile. Choose a path to save the file.

Step : Publishing

1. Back in Visual Studio, right-click on your project in the Solution Explorer and select "Publish."

2. Click "Import profile," browse for the publish profile file you downloaded in step 7, and click "Finish."

3. Click "Publish." This will publish the project to both the Azure VM and the Azure Web App.

Step : Accessing the Web App

1. Locate the downloaded RDP file for the Azure VM, open it, and provide the admin name and password.

2. Click "Yes" on the next dialog box, and it will load the Windows 10 VM.

3. In the VM's browser, copy the HTTP link from the publish page and paste it in the VM's browser. This will open the web app you created.

Conclusion

In conclusion, this process of creating an ASP.NET MVC web application, including defining model classes, controllers, and views. It further explained how

to deploy the application to both an Azure Virtual Machine (VM) and an Azure Web App. This multi-step approach allows for flexibility in hosting and testing the application, whether it's within the controlled environment of a VM or accessible globally through a web app. By following these steps, developers can efficiently build, deploy, and access their web applications, streamlining the development and deployment process.