**Step 1**: Creating an ASP.NET MVC project to demonstrate built-in user authentication

* Open Visual Studio.
* From the top menu, select File->New->Project.
* In Create A New Project screen, select ASP.NET Core Web Application from the list of available project types and click on Next.
* Enter Project Name as PracticeProjectManageSchoolData and click on Create.
* From the list of project sub-types, choose Web Application (Model-View-Controller) and uncheck Configure for HTTPS.
* Click on Change Authentication and from the authentication types choose Individual User Accounts.
* Click on Create.
* This will create the files for an ASP.NET MVC project with built-in User Authentication.

**Step 2**: Building the project

* From the top menu, choose Build->Build Solution.
* If any compile errors are shown, fix them as required.

**Step 3**: Publishing and running the project

* From the top menu, select Debug->Start Without Debugging.
* This will execute the program in the default browser.
* In the homepage, click on the Register Link at the top right.
* Fill in the registration form and click Register.
* It will initially show an exception. Click on the Apply Migration button.
* Refresh the page. You should be logged in now.
* Clicking on your email id at the top will open a dashboard page, which lets you configure settings and change your data.

**Step 4**: Creating a Student table in a database

* Open SQL Server Management Studio. In the login screen, make a note of the Server Name value as you will need to put in your ASP.NET application.
* In Object Explorer, right click Databases and choose New Database.
* Enter Database name as School and click Ok.
* In Object Explorer, expand School->Tables. Right click Tables and choose New->Table.
* For the first row enter ID as Column Name, int as Data Type, and uncheck Allow Nulls.
* In Column Properties, go to Identity Specification and expand it. Double click Is Identity to make it Yes.
* For the next row enter Name as Column Name, varchar(100) as Data Type, and uncheck Allow Nulls.
* For the next row enter Address as Column Name, varchar(100) as Data Type, and uncheck Allow Nulls.
* For the next row enter Email as Column Name, varchar(75) as Data Type, and uncheck Allow Nulls.
* For the next row enter Class as Column Name, varchar(5) as Data Type, and uncheck Allow Nulls.
* Click on the x icon to close the table grid tab. Click Yes on the save dialog.
* For Enter a name put Student and press Ok.
* In Object Explorer, expand School->Tables->Student. Right click Tables and choose Edit Top 200 Rows.
* Add in a few rows of data with random values.
* Close the Management Studio.

**Step 5**: Creating a Class table in a database

* In Object Explorer, expand School->Tables. Right click Tables and choose New->Table.
* For the first row enter ID as Column Name, int as Data Type, and uncheck Allow Nulls.
* In Column Properties, go to Identity Specification and expand it. Double click Is Identity to make it Yes.
* For the next row enter Name as Column Name, varchar(5) as Data Type, and uncheck Allow Nulls.
* Click on the x icon to close the table grid tab. Click Yes on the save dialog.
* For Enter a name put Class and press Ok.
* In Object Explorer, expand Class->Tables->Class. Right click Tables and choose Edit Top 200 Rows.
* Add in a few rows of data with random values.
* Close the Management Studio.

**Step 6**: Creating a Subject table in a database

* In Object Explorer, expand School->Tables. Right click Tables and choose New->Table.
* For the first row enter ID as Column Name, int as Data Type, and uncheck Allow Nulls.
* In Column Properties, go to Identity Specification and expand it. Double click Is Identity to make it Yes.
* For the next row enter Name as Column Name, varchar(50) as Data Type, and uncheck Allow Nulls.
* Click on the x icon to close the table grid tab. Click Yes on the save dialog.
* For Enter a name put Subject and press Ok.
* In Object Explorer, expand Class->Tables->Class. Right click Tables and choose Edit Top 200 Rows.
* Add in a few rows of data with random values.
* Close the Management Studio.

**Step 7**: Adding EntityFramework using NuGet

* In the top menu, go to Tools->Nuget Package Manager->Package Manager Console.
* In the console, type install-package entityframework and press Enter.

**Step 8**: Add connection string for school database to secrets.json using Visual Studio

**Step 9**: Edit Program.cs to buse builder configuration to use the school connection string and add a dbcontext using the school connection string.

**Step 10:** Using EntityFramework to generate models and dbcontext

* In the Solution Explorer, right click the project name and choose Add->New Item.
* Under Visual C#, select Data and choose EF Core Database First Wizard. Then click Add.
* In the next screen, select your school database and EF Core version. Then click OK.
* In the next screen select all three tables. Then click OK.
* Enter Context name as SchoolContext. Select use DataAnnotation Attributes. Then click OK.
* This will create all the data context and entity files for the three tables.

**Step 11**: Creating StudentsController to auto-create the views for CRUD operations

* In Solution Explorer, expand Controllers, right click Controllers and choose Add->Controller.
* From the list of types, choose MVC Controller with views, using Entity Framework and click Add.
* In the next screen, choose Model Class as Student, Data Context Class as SchoolContext, Controller Name as StudentsController and click Add.
* Choose layout page at: ~/Views/Shared/\_Layout.cshtml
* This will generate the code for the Controller for all Student CRUD operations.

**Step 12**: Creating ClassesController to auto-create the views for CRUD operations

* In Solution Explorer, expand Controllers, right click Controllers and choose Add->Controller.
* From the list of types, choose MVC Controller with views, using Entity Framework and click Add.
* In the next screen, choose Model Class.cs as class, Data Context Class as SchoolContext, Controller Name as ClassesController and click Add.
* Choose layout page at: ~/Views/Shared/\_Layout.cshtml
* This will generate the code for the Controller for all Class CRUD operations.

**Step 13**: Creating SubjectsController to auto-create the views for CRUD operations

* In Solution Explorer, expand Controllers, right click Controllers and choose Add->Controller.
* From the list of types, choose MVC Controller with views, using Entity Framework and click Add.
* In the next screen, choose Model Subject as Class, Data Context Class as SchoolContext, Controller Name as SubjectsController and click Add.
* Choose layout page at: ~/Views/Shared/\_Layout.cshtml
* This will generate the code for the Controller for all Class CRUD operations.

**Step 14:** Building the project

* From the top menu, choose Build->Build Solution.
* If any compile errors are shown, fix them as required.

**Step 15:** Publishing and running the project

* From the top menu, select Debug->Start Without Debugging.
* This will execute the program in the default browser.
* To see the student pages, go to the url: <http://localhost:xxxx/students>.
* To see the classes pages, go to the url: http://localhost:xxxx/classes.
* To see the subjects pages, go to the url: <http://localhost:xxxx/subjects>.

**Step 16:** Pushing the code to your GitHub repositories

* Open your command prompt and navigate to the folder where you have created your files.
* cd <folder path>
* Initialize your repository using the following command:
* git init
* Add all the files to your git repository using the following command:
* git add .
* Commit the changes using the following command:
* git commit -m “Changes have been committed.”
* Push the files to the folder you created initially using the following command:
* git push -u origin master