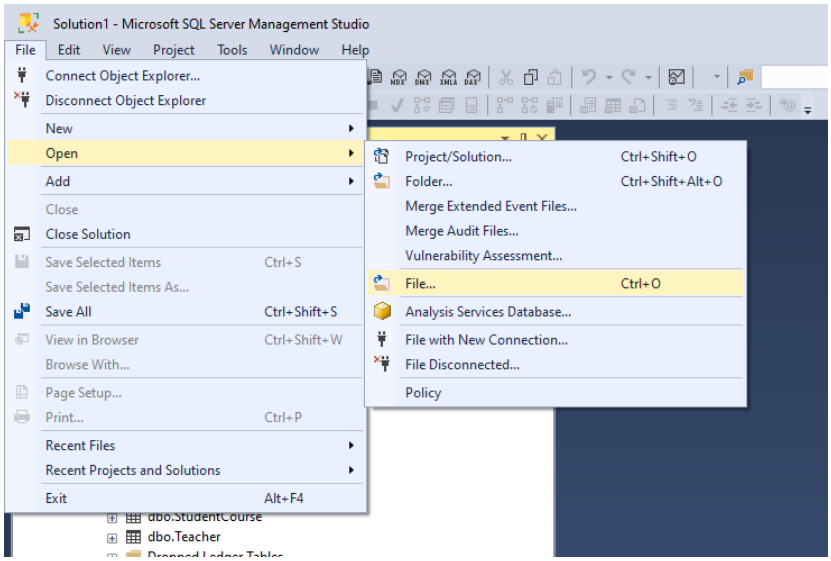
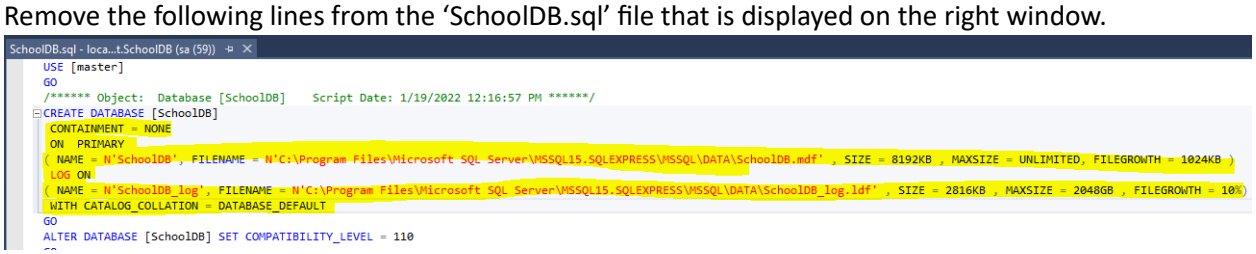
Modifying StudentMobileApp + WebAPI Project with Entity Framework Core

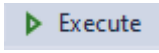
Set up SchoolDB

1) In SQL Management Studio, Select *File->Open->File->SchoolDB.sql* from **starter** folder

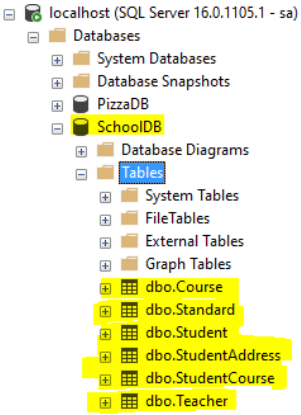


2) Remove the following lines from the 'SchoolDB.sql' file that is displayed on the right window.



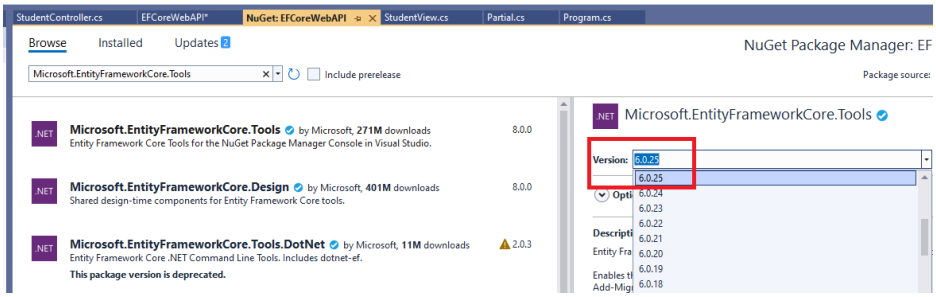
3) Select 

or F5 to run the script. Right click on **Databases** folder to refresh the databases in **Object Explorer**. The database should be created successfully.



4) Explore the database to understand the tables, fields and data available.

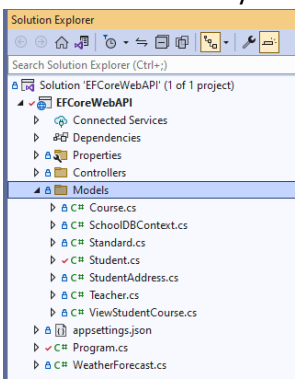
5) In Visual Studio, select *File->Open Project/Solution->EFCoreWebAPI->EFCoreWebAPI.sln* from **starter** folder.
6) Right click on the project in *Solution Explorer->Manage NuGet Packages*. Search for the 2 packages below and install them. Ensure that the correct version is selected.
    - Microsoft.EntityFrameworkCore.SqlServer (6.0.25)
    - Microsoft.EntityFrameworkCore.Tools (6.0.25)



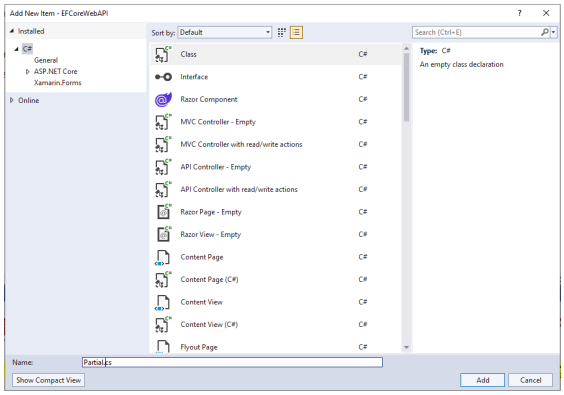
7) In **Package Manager Console**, run the following command to generate the entity model from the database.

    Scaffold-DbContext "Server=<destination address>;Database=SchoolDB;User Id=<your user id>;Password=<your password>" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models

8) The following class should be generated in the project for the entity model. Do not make any edits to these classes as they are auto generated.



9) Right click on **Models** folder in *Solution Explorer->Add->New Item->Class*. Name the class 'Partial.cs'



10) Repeat Step 9 to add another class 'StudentView.cs'

11) In ‘Partial.cs’, replace with the following codes.

```
namespace EFCoreWebAPI.Models
{
    10 references
    public partial class Student
    {
        0 references
        public string ModifiedStudentName   // property
        {
            get { return "Student " + this.StudentName; }   // get method
        }
    }
}
```

12) In ‘StudentView.cs’, replace with the following codes. You can also refer to the codes from ‘*code snippets.txt’* in **starter** folder.

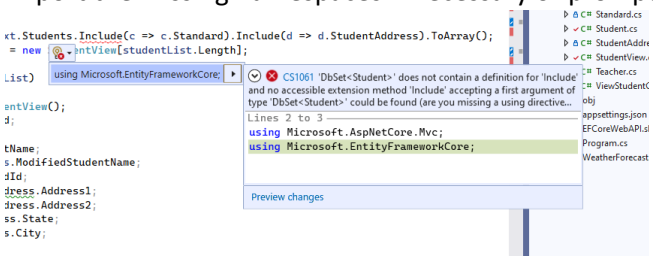
```
namespace EFCoreWebAPI.Models
{
    0 references
    public class StudentView
    {
        0 references
        public int StudentId { get; set; }

        private string? studentName;
        0 references
        public string? ModifiedStudentName { get; set; }
        private int? standardId;
        private string? address2;
        0 references
        public string? StudentName
        {
            get
            {
                return studentName;
            }
            set
            {
                if (value == null)
                {
                    studentName = "";
                }
                else
                {
                    studentName = value;
                }
            }
        }
        0 references
        public int? StandardId
        {
            get
            {
                return standardId;
            }
            set
            {
                if (value == null)
                {
                    standardId = 0;
                }
                else
                {
                    standardId = value;
                }
            }
        }
        0 references
        public string Address1 { get; set; }
        0 references
        public string? Address2
        {
            get
            {
                return address2;
            }
            set
            {
                if (value == null)
                {
                    address2 = "";
                }
                else
                {
                    address2 = value;
                }
            }
        }
        0 references
        public string State { get; set; }
        0 references
        public string City { get; set; }

    }
}
```

13) Right click on **Controllers** folder in *Solution Explorer->Add->Controller->API Controller with read/write action.* Name the controller class ‘StudentController.cs’

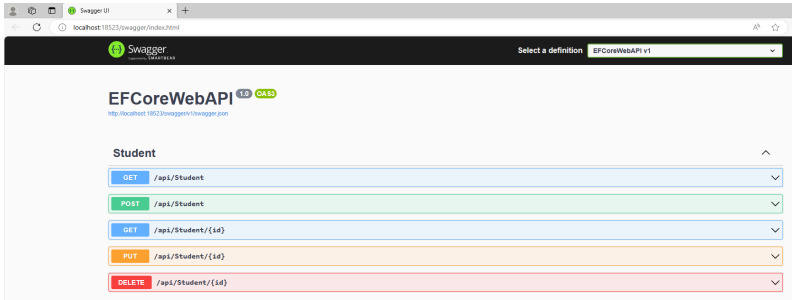
14) Replace **Get()** method in 'StudentController.cs' with the following codes. The method returns a list of students stored in the database.

```
//get student list
// GET: api/<StudentController>
[HttpGet]
0 references
public IActionResult Get()
{
    using (SchoolDBContext context = new SchoolDBContext())
    {
        try
        {
            Student[] studentList = context.Students.Include(c => c.Standard).Include(d => d.StudentAddress).ToArray();
            StudentView[] studentListView = new StudentView[studentList.Length];
            int i = 0;
            foreach (Student s in studentList)
            {
                StudentView sv = new StudentView();
                sv.StudentId = s.StudentId;

                sv.StudentName = s.StudentName;
                sv.ModifiedStudentName = s.ModifiedStudentName;
                sv.StandardId = s.StandardId;
                sv.Address1 = s.StudentAddress.Address1;
                sv.Address2 = s.StudentAddress.Address2;
                sv.State = s.StudentAddress.State;
                sv.City = s.StudentAddress.City;
                studentListView[i] = sv;
                i = i + 1;
            }
            return Ok(studentListView);
            //return Ok(context.Students.Include(c => c.Standard).Include(d => d.StudentAddress).ToArray());
        }
        catch (Exception)
        {
            return NotFound("An error has occured. Unable to load list of students");
        }
    }
}
```

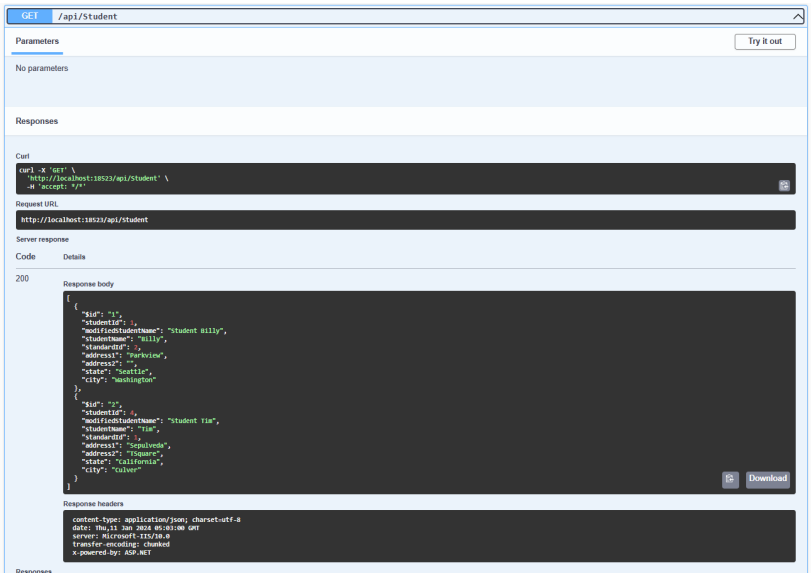
15) Import the missing namespaces if necessary or prompted.



16) Select IIS Express or **F5** to run the project on default browser. It will be automatically deployed on IISExpress. Take note of the port number in the URL as well and record it somewhere.



17) Let's test the **Get** method. It should return you a list of students available from the database in JSON.



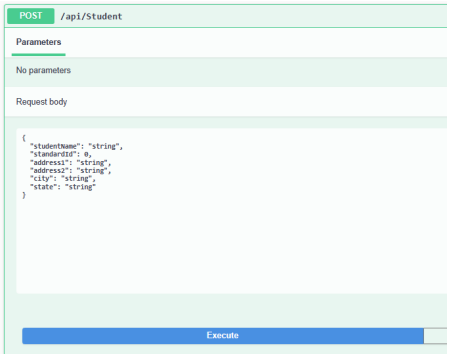
18) Replace **Post()** method in 'StudentController.cs' with the following codes. The method add a new student provided in the parameter to the database.

```
//Add new student
// POST api/<StudentController>
[HttpPost]
public IActionResult Post([FromBody] PostStudentBody newStudentPost)
{
    using (SchoolDBContext context = new SchoolDBContext())
    {
        try
        {
            Student newStudent = new Student();
            newStudent.StudentName = newStudentPost.StudentName;
            newStudent.StandardId = newStudentPost.StandardId;
            context.Students.Add(newStudent);
            context.SaveChanges();
            StudentAddress newStudentAddress = new StudentAddress();
            newStudentAddress.StudentId = newStudent.StudentId;
            newStudentAddress.Address1 = newStudentPost.Address1;
            newStudentAddress.Address2 = newStudentPost.Address2;
            newStudentAddress.City = newStudentPost.City;
            newStudentAddress.State = newStudentPost.State;
            newStudent.StudentAddress = newStudentAddress;
            context.SaveChanges();
            return Ok(newStudent.StudentId);
        }
        catch (Exception)
        {
            return BadRequest("Unable to add student. Check request body is in correct format.");
        }
    }
}
```

19) The Post() method requires a new class for the request body. Add the following **PostStudentBody** class at the bottom of **StudentController** class.

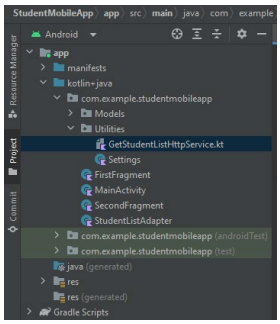
```
public class PostStudentBody
{
    public string? StudentName { get; set; }
    public int? StandardId { get; set; }

    public string Address1 { get; set; }
    public string? Address2 { get; set; }
    public string City { get; set; }
    public string State { get; set; }

    public PostStudentBody()
    {

    }
}
```

20) Test the **Post()** method. It should add a new student to the database based on your provided values in the request body.

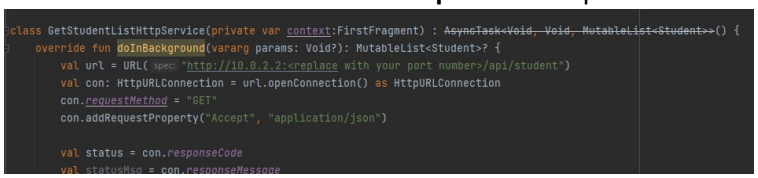


21) In Android Studio, Select *Open->StudentMobileApp* from **starter** folder

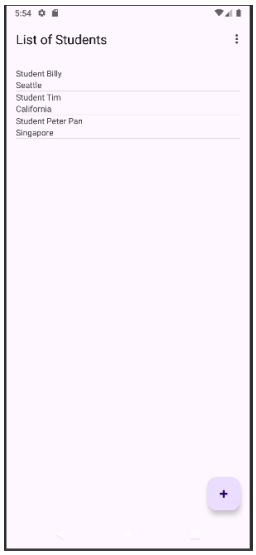
22) In the project explorer, double click on the file *app->kotlin+java->Utilities->GetStudentListHttpService.kt* to view the source codes.



23) Look for the class **GetStudentListHttpService**. Replace the following URL with your own port number.



24) Select  to run the mobile app. It should call the web api and display a list of students on the first screen.



25) Add a new **AddStudentHttpService** in the same file. It should be after the previous class. This class is required to call the Web API to add a new student to the database.

```
class AddStudentHttpService(private var context: SecondFragment, private var newStudent:Student) : AsyncTask<Void, Void, Boolean>() {
    override fun doInBackground(vararg params: Void?): Boolean {
        val url = URL( spec: "http://10.0.2.2:1600/api/student")
        val con: HttpURLConnection = url.openConnection() as HttpURLConnection
        con.requestMethod = "POST"
        con.setRequestProperty("Content-Type", "application/json; utf-8");
        con.setRequestProperty("Accept", "application/json");
        val json = Json.encodeToString(newStudent)
        val os: OutputStream = con.outputStream
        os.write(json.toByteArray())
        os.flush()
        os.close()
        Log.d( tag: "Http Service Tag-post",  msg: "response code "+con.responseCode);

        con.disconnect();
        return con.responseCode == 200
    }

    override fun onPreExecute() {
        super.onPreExecute()
        // ...
    }

    override fun onPostExecute(result: Boolean) {
        super.onPostExecute(result)
        if(result)
        {
            context.onAddorUpdateCompleted();
        }
        else
        {
            context.onAddorUpdateError();
        }
    }
}
```

26) In ‘SecondFragment.kt’, add the following codes to call **AddStudentHttpService.** This will call the Web API to add a new student based on the values provided when the submit button is clicked.

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    (requireActivity() as MainActivity).binding.fab.isVisible=false
    if(Settings.AddStudent)
    {
        (requireActivity() as MainActivity).title = "Add New Student"
        binding.studentNameET.setText("")
        binding.gradeSpin.setSelection(0)
        binding.address1ET.setText("")
        binding.address2ET.setText("")
        binding.stateET.setText("")
        binding.cityET.setText("")

        binding.studentNameET.isEnabled=true
        binding.gradeSpin.isEnabled=true
        binding.address1ET.isEnabled=true
        binding.address2ET.isEnabled=true
        binding.stateET.isEnabled=true
        binding.cityET.isEnabled=true
        binding.editBtn.isVisible=false
        binding.submitBtn.isVisible=true

        binding.submitBtn.setOnClickListener { view ->
            val newStudentObj = Student()
            newStudentObj.studentName = binding.studentNameET.text.toString()
            newStudentObj.standardId = binding.gradeSpin.selectedItemPosition
            newStudentObj.address1 = binding.address1ET.text.toString()
            newStudentObj.address2 = binding.address2ET.text.toString()
            newStudentObj.state = binding.stateET.text.toString()
            newStudentObj.city = binding.cityET.text.toString()
            AddStudentHttpService( context: this,newStudentObj).execute()
        }
    }
```

27) You might need to resolve any unresolved references. In the following case, you import the class as stated in the prompt.

```
AddStudentHttpService(this,newStudentObj).execute()
Unresolved reference: AddStudentHttpService
Import class 'AddStudentHttpService'  Alt+Shift+Enter   More actions...  Alt+Enter
```

28) Test and run the application. You should be able to add a new student via the mobile app with the screens provided.

Challenge Yourself

Based on what you have learnt so far, try implementing the Update and Delete functions in the mobile app as well as the Web API. This will complete the full CRUD functions for the project.