Car Repository - Insert

Description

Note :

EFCore.csproj file is given for your reference in the code template. Do not make any changes here.

Objective: This exercise aims to introduce learners to developing a C# application utilizing Entity Framework Core to store car details in a SQL Server database.

Scenario:

Sam, a skilled software developer, recently joined a prestigious car company. As part of his responsibilities, he has been assigned the task of creating an advanced application for adding car details to the company's database.

Help them to create a **C#** application to store details in the **SQLSERVER** database and use the **entity framework core** to connect to the database, which provides the following functionalities.

Functionalities:

- Add car details to the database.

Requirements:

1. In the class **Car,** implement the below given public properties.

Data Type	Property Name
int	Id

string	Brand
string	Model
double	Price

Add the below data annotation attributes :

- **Key** attribute - which is used to make the **Id** property a primary key in the **Car** class.
- **DatabaseGeneratedOption.None** attribute - which is used to make the **Id** property a non-identity primary key in the **Car** class.

2. In the class **Car Context,** implement the below given property, method and also inherit the class **DbContext**.

Data Type	Property Name
DbSet<Car>	Cars

Method	Description
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)	This method is used to configure the database connection for the **DbContext**.

appsettings.json:

```
{
  "ConnectionStrings": {
   "DefaultConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial Catalog=StudentDB;Integrated Security=True;"
  }
}
```

This file is already provided for you. Use the connection string in your context class **OnConfiguring** method to configure the database.

The below-given reference code is the part of the CarContext.cs

3. In the class **CarRepository,** implement the below given public method.

Method	Description
public bool AddCar(Car car)	This method is used to add the given car details to the database. If the car details are added successfully, then return **true**.

4. In the class **Program - Main()** method,

-- Get the all input values from the user.

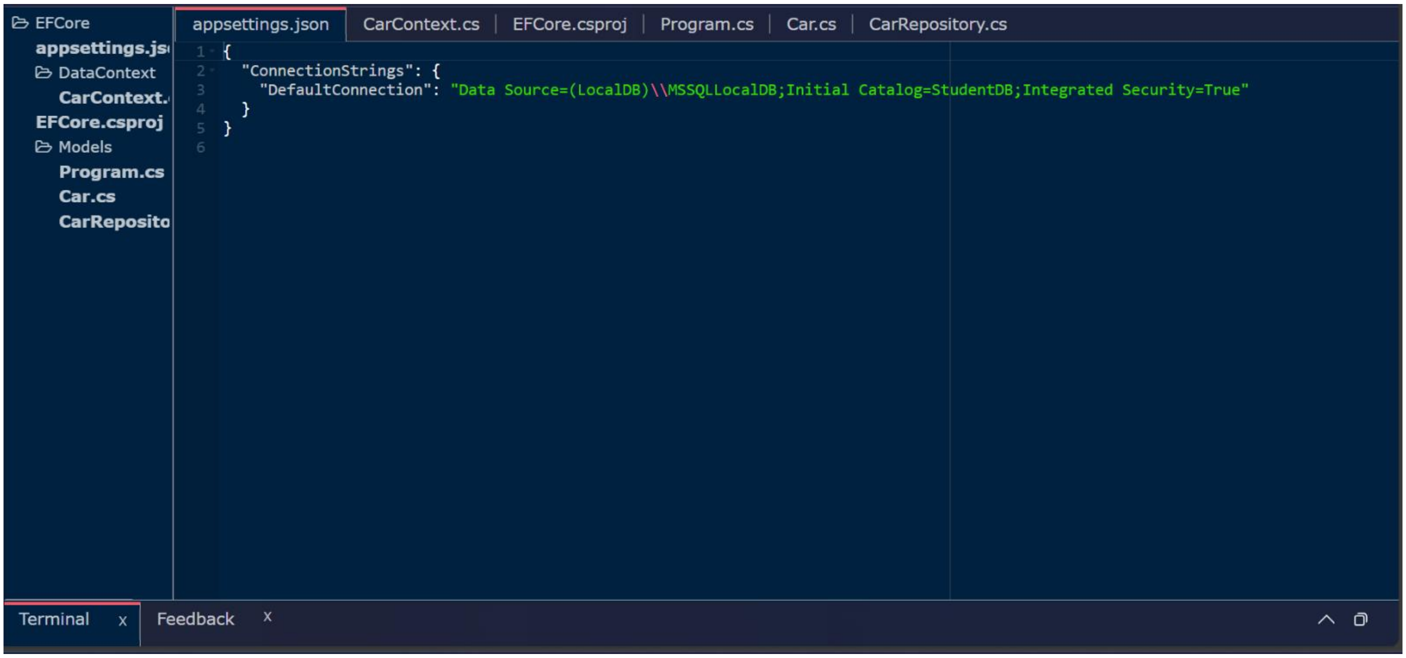
-- Call the AddCar method and print **Details Added Successfully.**

Note:

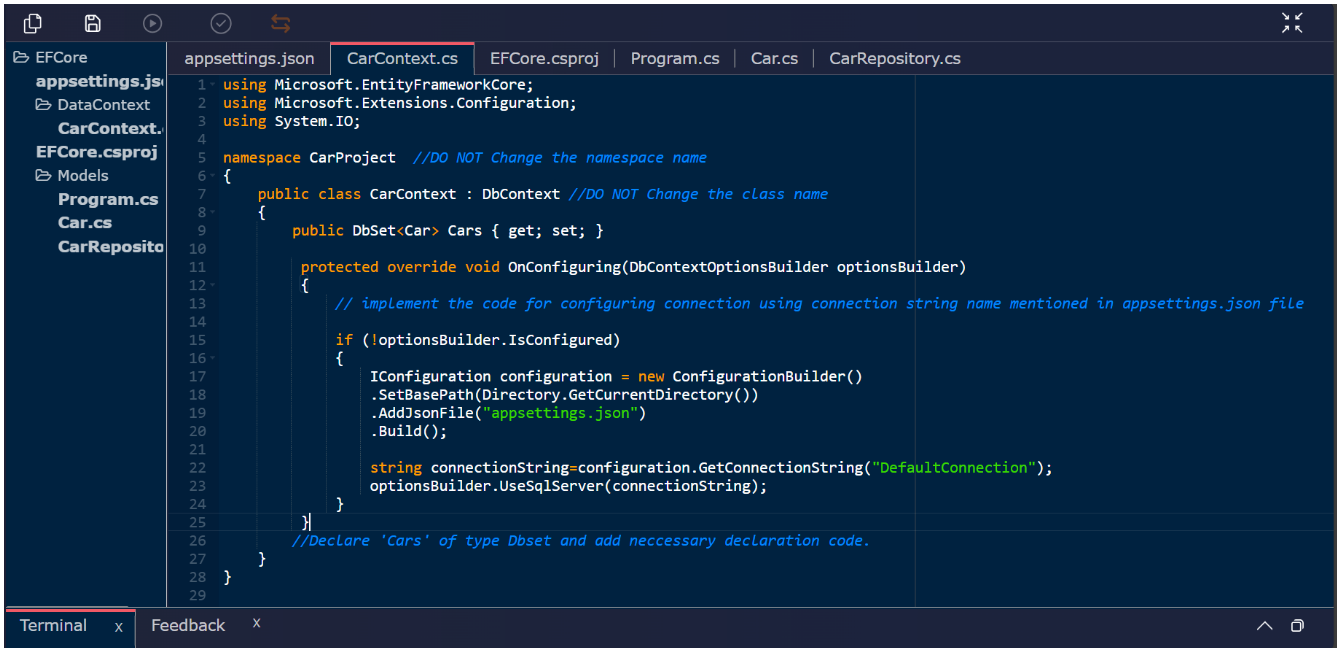
- Keep the properties, methods and classes as **public.**
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code template.

Sample Input/output:

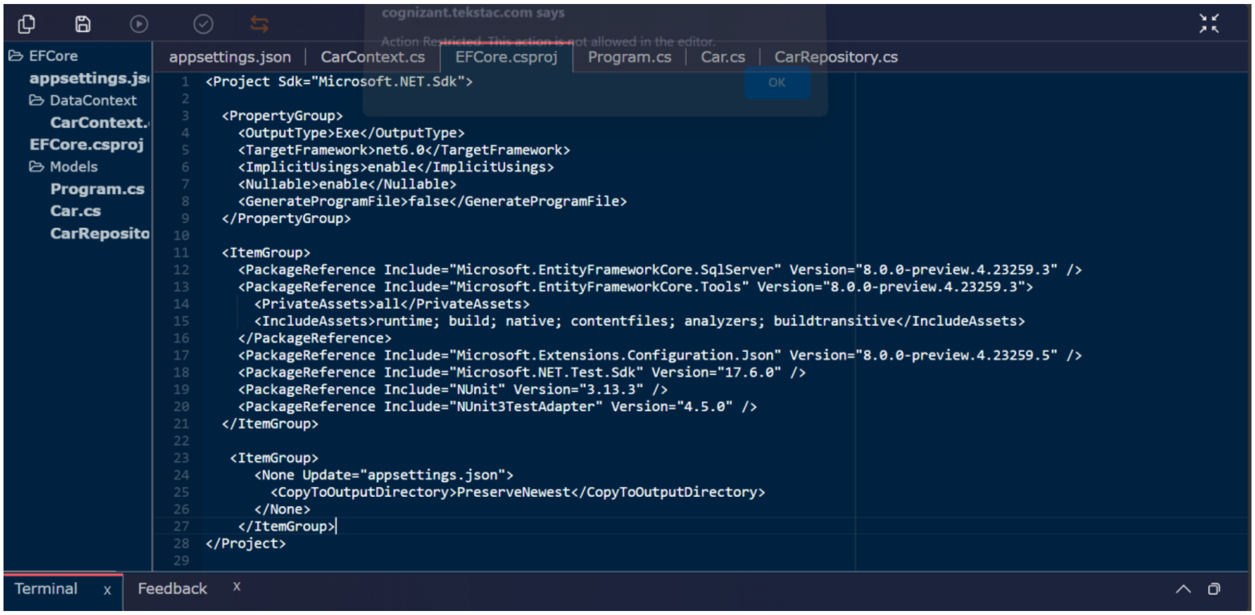
```
Enter car id
1
Enter car brand
Hyundai
Enter car model
Hyundai Elantra
Enter car price
1800000

Details Added Successfully
```



```json
{
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial Catalog=StudentDB;Integrated Security=True"
  }
}
```



```csharp
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using System.IO;

namespace CarProject  //DO NOT Change the namespace name
{
    public class CarContext : DbContext //DO NOT Change the class name
    {
        public DbSet<Car> Cars { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            // implement the code for configuring connection using connection string name mentioned in appsettings.json file

            if (!optionsBuilder.IsConfigured)
            {
                IConfiguration configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("appsettings.json")
                .Build();

                string connectionString=configuration.GetConnectionString("DefaultConnection");
                optionsBuilder.UseSqlServer(connectionString);
            }
        }
        //Declare 'Cars' of type Dbset and add neccessary declaration code.
    }
}
```

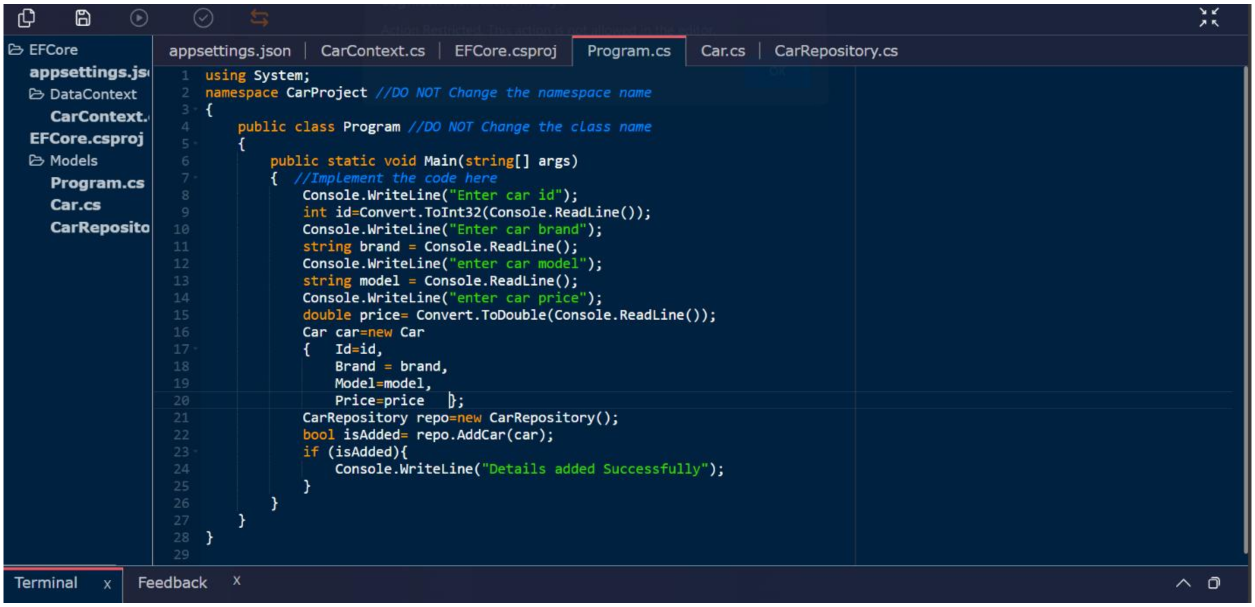


cognizant.tekstac.com says

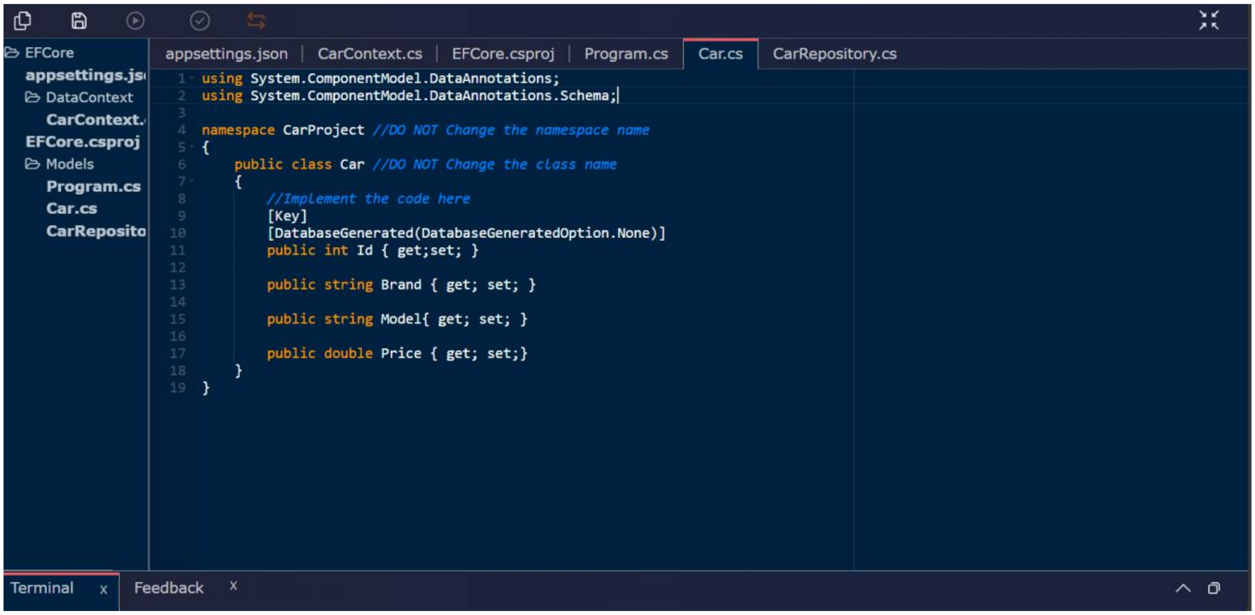
Action Restricted. This action is not allowed in the editor.

OK

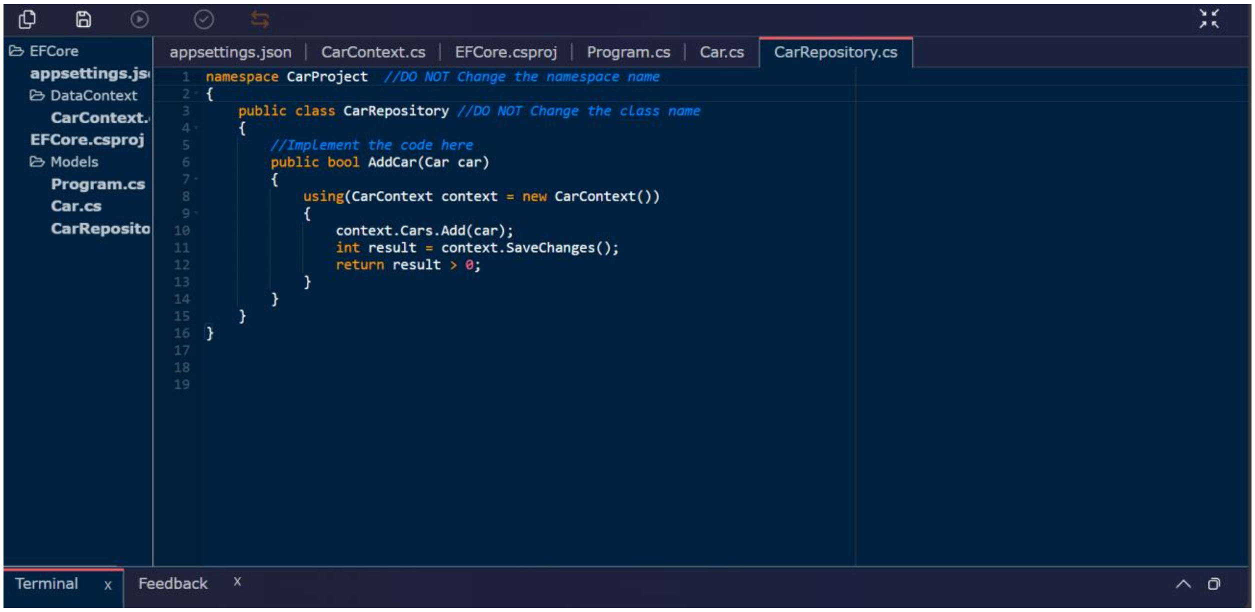
```xml
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net6.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <GenerateProgramFile>false</GenerateProgramFile>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.0-preview.4.23259.3" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="8.0.0-preview.4.23259.3">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.Extensions.Configuration.Json" Version="8.0.0-preview.4.23259.5" />
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.6.0" />
    <PackageReference Include="NUnit" Version="3.13.3" />
    <PackageReference Include="NUnit3TestAdapter" Version="4.5.0" />
  </ItemGroup>

   <ItemGroup>
      <None Update="appsettings.json">
        <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
      </None>
    </ItemGroup>
</Project>
```



```csharp
using System;
namespace CarProject //DO NOT Change the namespace name
{
    public class Program //DO NOT Change the class name
    {
        public static void Main(string[] args)
        {  //Implement the code here
            Console.WriteLine("Enter car id");
            int id=Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter car brand");
            string brand = Console.ReadLine();
            Console.WriteLine("enter car model");
            string model = Console.ReadLine();
            Console.WriteLine("enter car price");
            double price= Convert.ToDouble(Console.ReadLine());
            Car car=new Car
            {   Id=id,
                Brand = brand,
                Model=model,
                Price=price   };
            CarRepository repo=new CarRepository();
            bool isAdded= repo.AddCar(car);
            if (isAdded){
                Console.WriteLine("Details added Successfully");
            }
        }
    }
}
```



```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace CarProject //DO NOT Change the namespace name
{
    public class Car //DO NOT Change the class name
    {
        //Implement the code here
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int Id { get;set; }

        public string Brand { get; set; }

        public string Model{ get; set; }

        public double Price { get; set;}
    }
}
```



```
namespace CarProject  //DO NOT Change the namespace name
{
    public class CarRepository //DO NOT Change the class name
    {
        //Implement the code here
        public bool AddCar(Car car)
        {
            using(CarContext context = new CarContext())
            {
                context.Cars.Add(car);
                int result = context.SaveChanges();
                return result > 0;
            }
        }
    }
}
```

Car Repository - Eager Loading

Description

Note :

EFCore.csproj file is given for your reference in the code template. Do not make any changes here.

Objective: This exercise aims to introduce learners to the concept of eager loading in Entity Framework Core and demonstrate its usage to optimize data retrieval by efficiently loading related data along with the main entities in a single query.

Scenario:

Sam has been assigned the new task of creating an advanced application for retrieving car and make details from the company's database.

Help them create a **C#** application to retrieve details from the **SQLSERVER** database and use the **entity framework core** to connect to the database, which provides the following functionalities.

Functionalities:

- Retrieving car and make details from the database using **Eager Loading.**

Requirements:

1. In the class **Car,** implement the below given public properties.

Data Type	Property Name
int	Id

string	Model
int	Year
int	MakeId
Make	Make

Add the below data annotation attributes :

- **Key** attribute - which is used to make the **Id** property a primary key in the **Car** class.
- **DatabaseGeneratedOption.None** attribute - which is used to make the **Id** property a non-identity primary key in the **Car** class.

2. In the class **Make,** implement the below given public properties.

Data Type	**Property Name**
int	Id
string	Name

Add the below data annotation attributes :

- **Key** attribute - which is used to make the **Id** property a primary key in the **Make** class.
- **DatabaseGeneratedOption. None** attribute - which is used to make the **Id** property a non-identity primary key in the **Make** class.

3. In the class **Car Context,** implement the below given property, method and also inherit the class **DbContext**.

Data Type	Property Name
DbSet<Car>	Cars
DbSet<Make>	Makes

Method	Description
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)	This method is used to configure the database connection for the **DbContext**.

appsettings.json:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial Catalog=StudentDB;Integrated Security=True;"
  }
}
```

This file is already provided for you. Use the connection string in your context class **OnConfiguring** method to configure the database.

The below-given reference code is the part of the CarContext.cs

4. In the class **CarRepository,** implement the below given public method.

Method	**Description**
public static IEnumerable<Car> GetAllCarsWithMake(CarContext context)	This method is used to get the cars with make details from the database. **Hint**: Use **Include** method (Eager Loading Concept).

4. In the class **Program - Main()** method,

-- Call the GetAllCarsWithMake method and display the result as per the sample output.

5. The "**GetMyExpression**" method is for testing your LINQ QUERY EXPRESSION. So fill your query expression in the space holder provided. ONLY THE QUERY EXPRESSION Nothing more needs to be implemented in this method.

The below sample data is already available in the database table **Makes**.

Id	**Name**
101	Toyota
102	Ford
103	Honda
104	BMW

The below sample data is already available in the database table **Cars**.

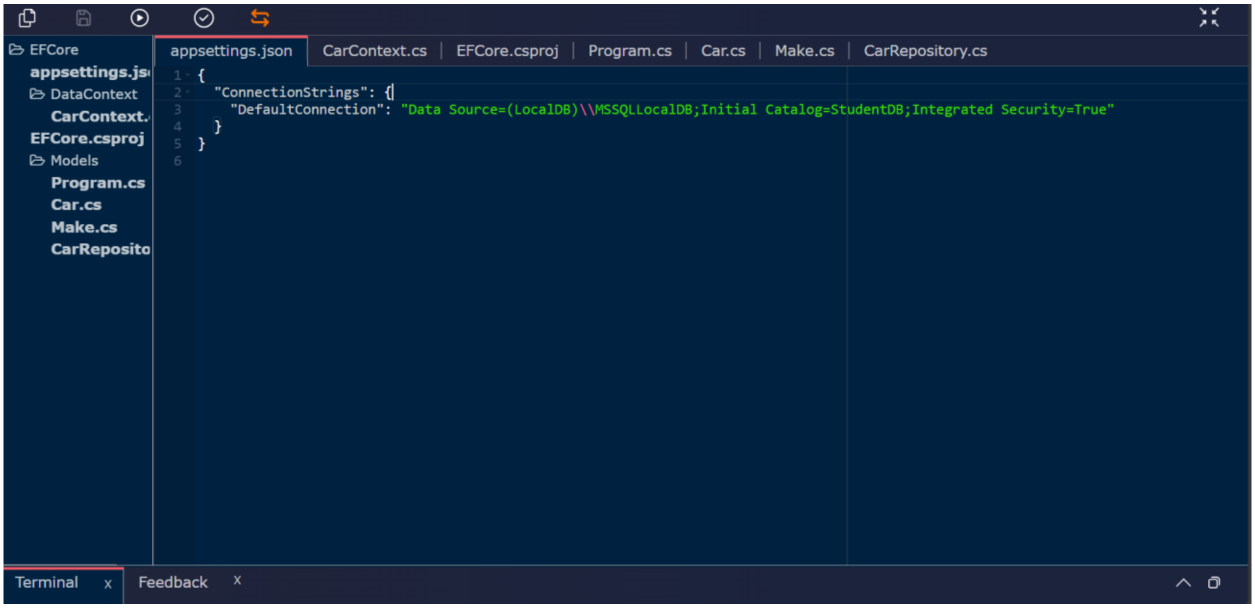
Id	**Model**	**Year**	**MakeId**
1	Camry	1999	101
2	Mustang	2002	102
3	Sienna	2000	101
4	Civic	1998	103
5	X3	2002	104

Note:

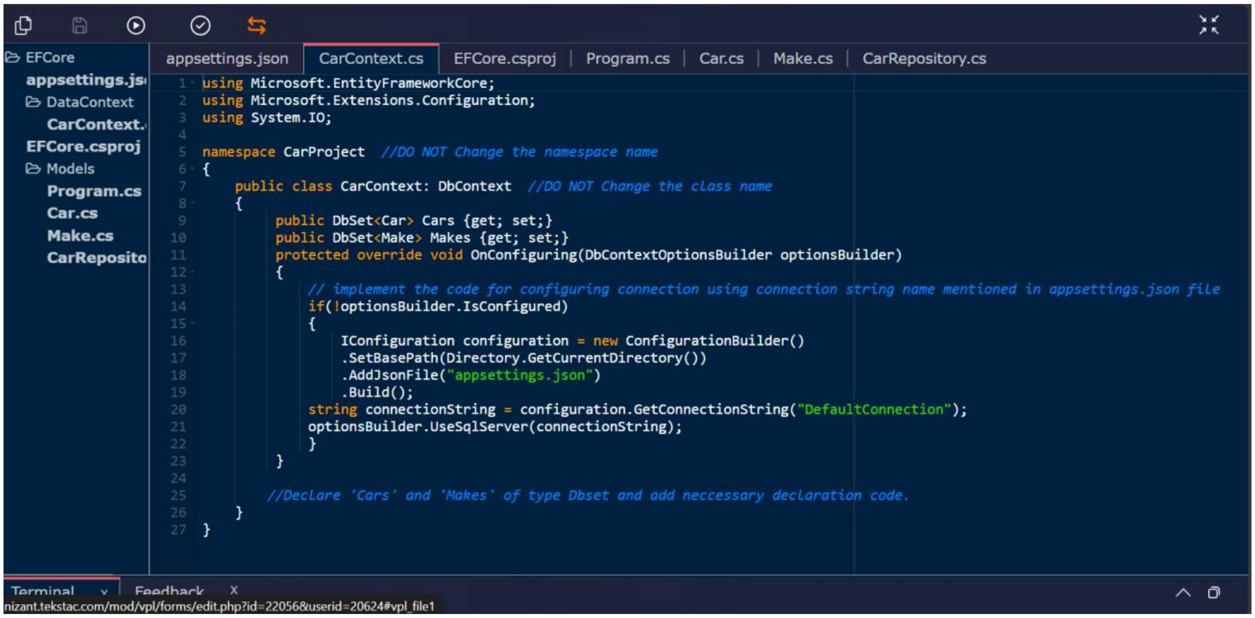
- Keep the properties, methods and classes as **public.**
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code template.

Sample Output :

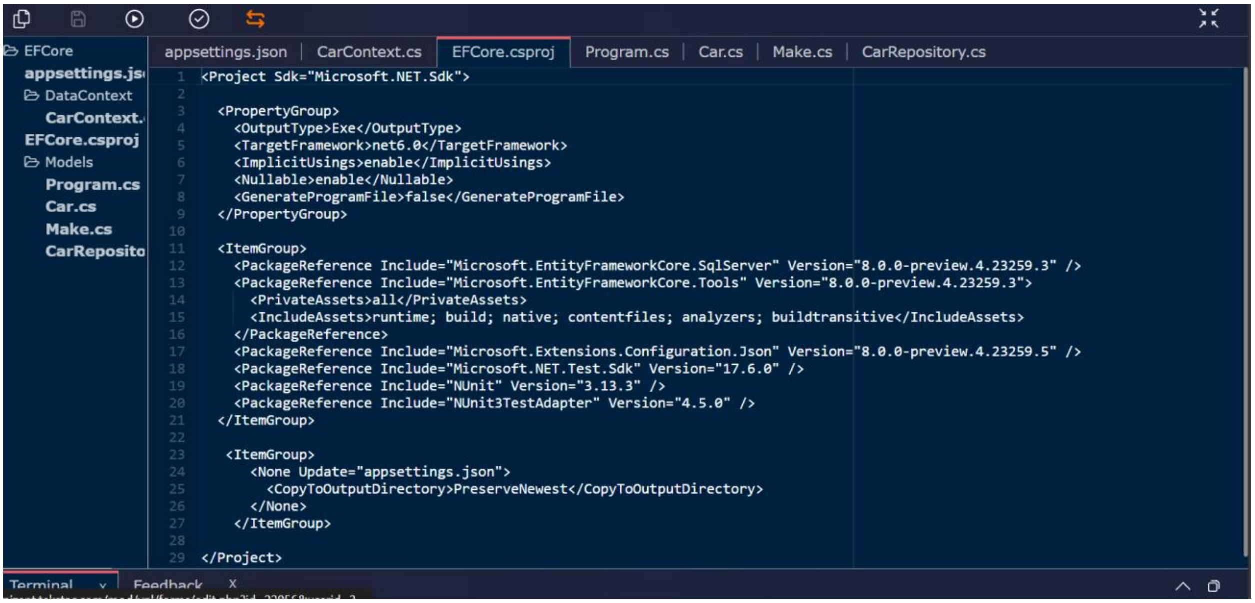
```
Car Id: 1,     Make: Toyota,     Model: Camry,       Year: 1999

Car Id: 2,     Make: Ford,        Model: Mustang,   Year: 2002

Car Id: 3,     Make: Toyota,     Model: Sienna,      Year: 2000

Car Id: 4,     Make: Honda,      Model: Civic,        Year: 1998

Car Id: 5,     Make: BMW,        Model: X3,            Year: 2002
```



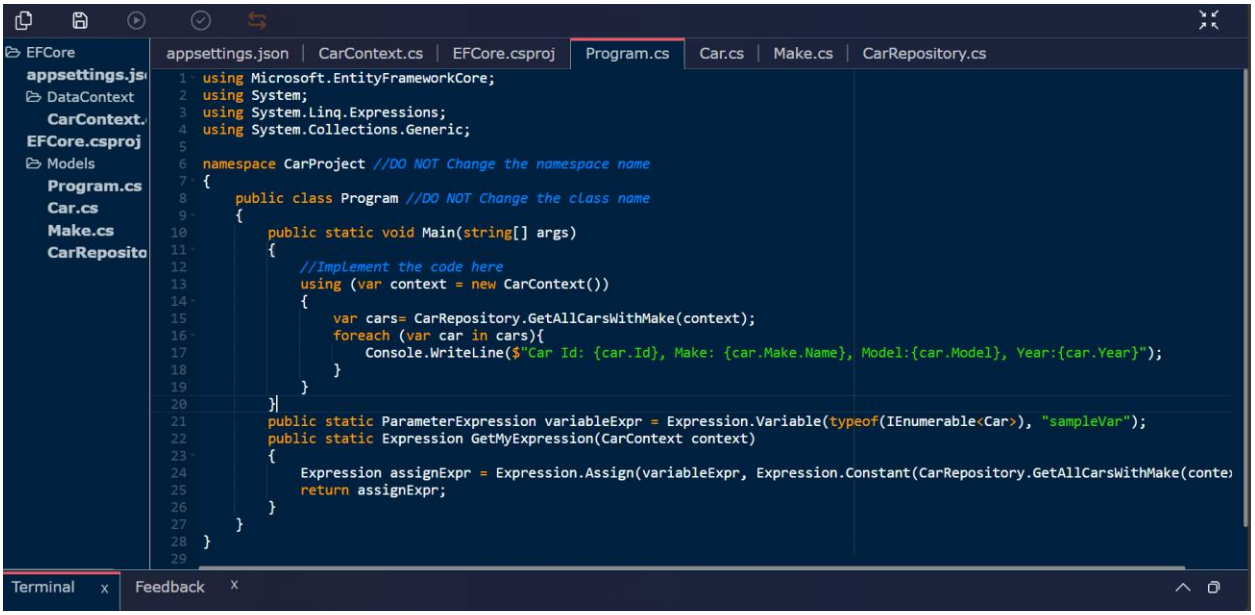
```json
{
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial Catalog=StudentDB;Integrated Security=True"
  }
}
```



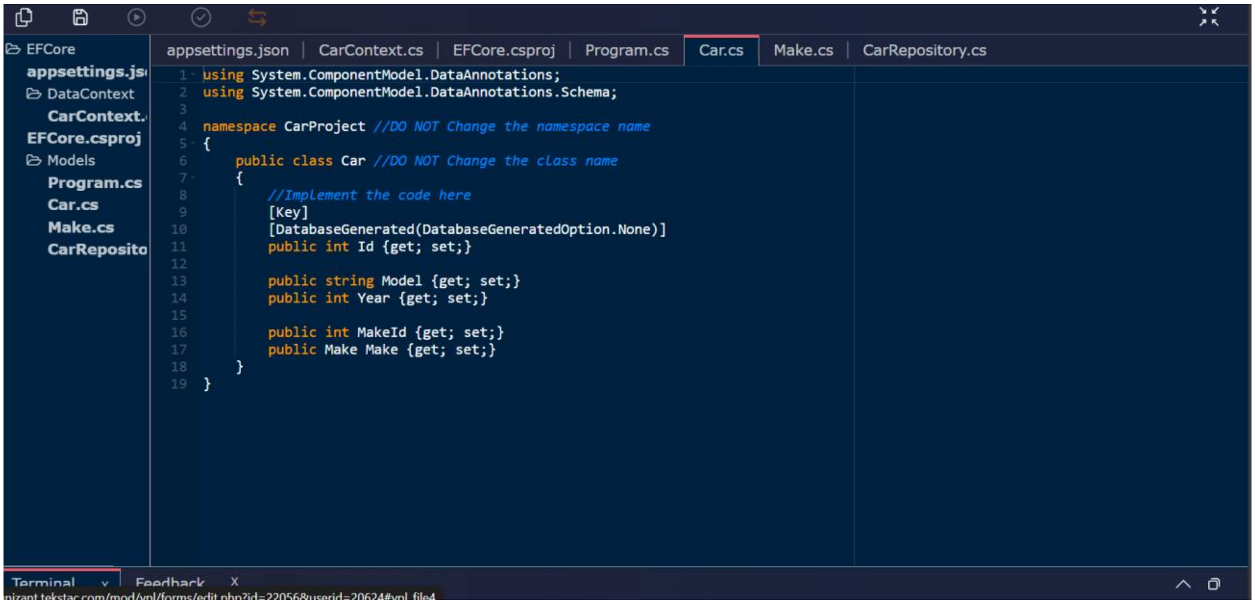
```csharp
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using System.IO;

namespace CarProject  //DO NOT Change the namespace name
{
    public class CarContext: DbContext  //DO NOT Change the class name
    {
         public DbSet<Car> Cars {get; set;}
         public DbSet<Make> Makes {get; set;}
         protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
         {
             // implement the code for configuring connection using connection string name mentioned in appsettings.json file
             if(!optionsBuilder.IsConfigured)
             {
                 IConfiguration configuration = new ConfigurationBuilder()
                 .SetBasePath(Directory.GetCurrentDirectory())
                 .AddJsonFile("appsettings.json")
                 .Build();
             string connectionString = configuration.GetConnectionString("DefaultConnection");
             optionsBuilder.UseSqlServer(connectionString);
             }
         }

        //Declare 'Cars' and 'Makes' of type Dbset and add neccessary declaration code.
    }
}
```



```xml
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net6.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <GenerateProgramFile>false</GenerateProgramFile>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.0-preview.4.23259.3" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="8.0.0-preview.4.23259.3">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.Extensions.Configuration.Json" Version="8.0.0-preview.4.23259.5" />
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.6.0" />
    <PackageReference Include="NUnit" Version="3.13.3" />
    <PackageReference Include="NUnit3TestAdapter" Version="4.5.0" />
  </ItemGroup>

   <ItemGroup>
      <None Update="appsettings.json">
        <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
      </None>
    </ItemGroup>

</Project>
```

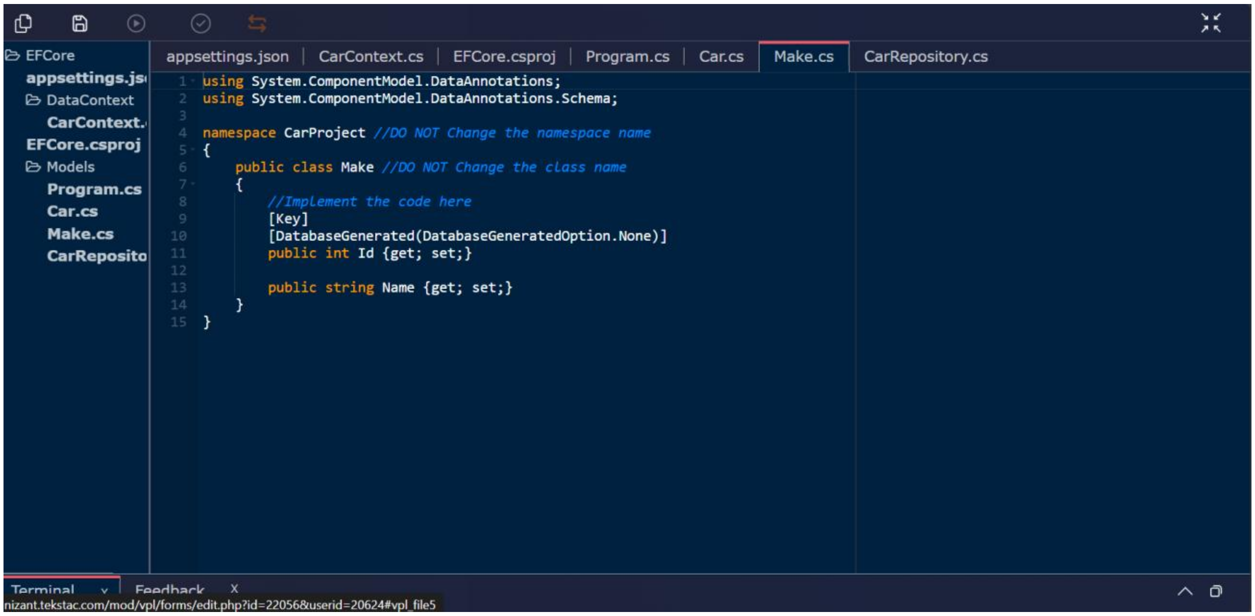


```csharp
using Microsoft.EntityFrameworkCore;
using System;
using System.Linq.Expressions;
using System.Collections.Generic;

namespace CarProject //DO NOT Change the namespace name
{
    public class Program //DO NOT Change the class name
    {
        public static void Main(string[] args)
        {
            //Implement the code here
            using (var context = new CarContext())
            {
                var cars= CarRepository.GetAllCarsWithMake(context);
                foreach (var car in cars){
                    Console.WriteLine($"Car Id: {car.Id}, Make: {car.Make.Name}, Model:{car.Model}, Year:{car.Year}");
                }
            }
        }
        public static ParameterExpression variableExpr = Expression.Variable(typeof(IEnumerable<Car>), "sampleVar");
        public static Expression GetMyExpression(CarContext context)
        {
            Expression assignExpr = Expression.Assign(variableExpr, Expression.Constant(CarRepository.GetAllCarsWithMake(conte
            return assignExpr;
        }
    }
}

```

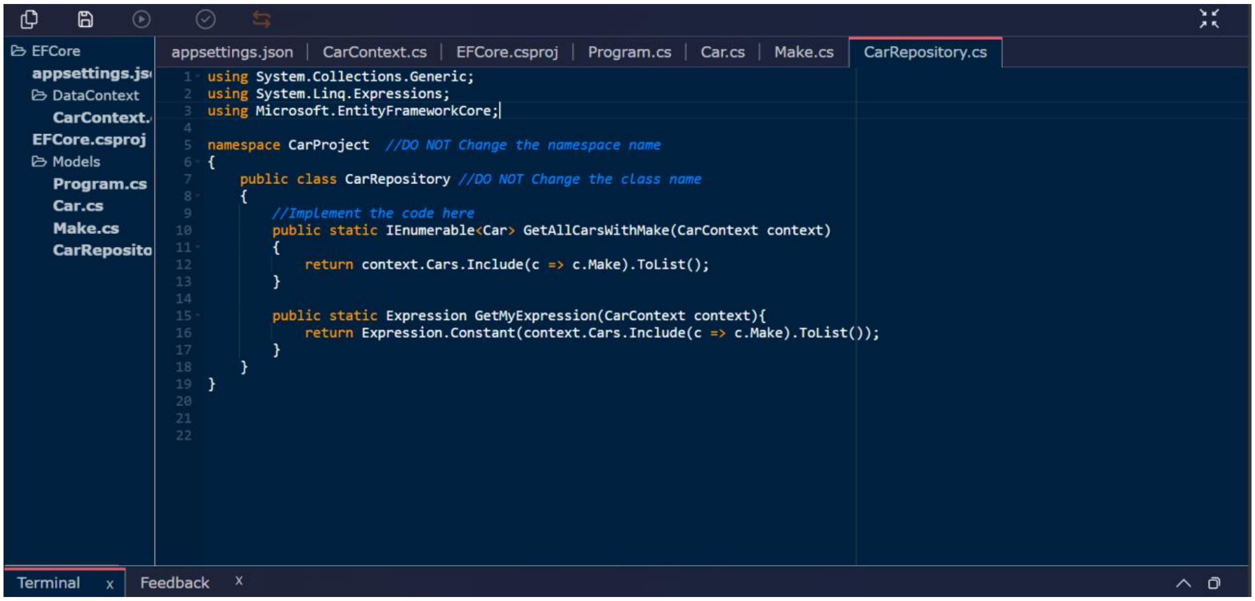


```csharp
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace CarProject //DO NOT Change the namespace name
{
    public class Car //DO NOT Change the class name
    {
        //Implement the code here
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int Id {get; set;}

        public string Model {get; set;}
        public int Year {get; set;}

        public int MakeId {get; set;}
        public Make Make {get; set;}
    }
}
```

nizant.tekstac.com/mod/vpl/forms/edit.php?id=22056&userid=20624#vpl_file4



```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace CarProject //DO NOT Change the namespace name
{
    public class Make //DO NOT Change the class name
    {
        //Implement the code here
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int Id {get; set;}

        public string Name {get; set;}
    }
}
```



```
using System.Collections.Generic;
using System.Linq.Expressions;
using Microsoft.EntityFrameworkCore;

namespace CarProject  //DO NOT Change the namespace name
{
    public class CarRepository //DO NOT Change the class name
    {
        //Implement the code here
        public static IEnumerable<Car> GetAllCarsWithMake(CarContext context)
        {
            return context.Cars.Include(c => c.Make).ToList();
        }

        public static Expression GetMyExpression(CarContext context){
            return Expression.Constant(context.Cars.Include(c => c.Make).ToList());
        }
    }
}
```

Car Repository - Lazy Loading

Description

Note :

EFCore.csproj file is given for your reference in the code template. Do not make any changes here.

Objective: This application aims to educate learners about the concept of lazy loading in the Entity Framework Core. Lazy Loading, the default behavior in Entity Framework Core, defers the loading of related entities until they are accessed for the first time.

Scenario:

Sam has been assigned the new task of creating an advanced application for retrieving car and make details from the company's database.

Help them create a **C#** application to retrieve details from the **SQLSERVER** database and use the **entity framework core** to connect to the database, which provides the following functionalities.

Functionalities:

- Retrieving car and make details from the database using **Lazy Loading.**

Requirements:

1. In the class **Car,** implement the below given public properties.

Data Type	Property Name
int	Id

string	Model
int	Year
int	MakeId
Make	Make

Note : To enable Lazy Loading, use the **virtual** keyword for **Make** Property.

Add the below data annotation attributes :

- **Key** attribute - which is used to make the **Id** property a primary key in the **Car** class.
- **DatabaseGeneratedOption.None** attribute - which is used to make the **Id** property a non-identity primary key in the **Car** class.

2. In the class **Make,** implement the below given public properties.

Data Type	**Property Name**
int	Id
string	Name

Add the below data annotation attributes :

- **Key** attribute - which is used to make the **Id** property a primary key in the **Make** class.
- **DatabaseGeneratedOption.None** attribute - which is used to make the **Id** property a non-identity primary key in the **Make** class.

3. In the class **Car Context,** implement the below given property, method and also inherit the class **DbContext**.

Data Type	Property Name
DbSet<Car>	Cars
DbSet<Make>	Makes

Method	Description
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)	This method is used to configure the database connection for the **DbContext**. **Note** : In this method, call the **UseLazyLoadingProxies()** method to enable lazy loading.

appsettings.json:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial Catalog=StudentDB;Integrated Security=True;"
  }
}
```

This file is already provided for you. Use the connection string in your context class **OnConfiguring** method to configure the database.

The below-given reference code is the part of the CarContext.cs

4. In the class **CarRepository,** implement the below given public method.

Method	Description
public static IEnumerable<Car> GetAllCarsWithMake(CarContext context)	This method is used to get the cars with make details from the database.

4. In the class **Program - Main()** method,

-- Call the GetAllCarsWithMake method and display the result as per the sample output.

5. The "**GetMyExpression**" method is for testing your LINQ QUERY EXPRESSION. So fill your query expression in the space holder provided. ONLY THE QUERY EXPRESSION Nothing more needs to be implemented in this method.

The below sample data is already available in the database table **Makes**.

Id	Name

101	Toyota
102	Ford
103	Honda
104	BMW

The below sample data is already available in the database table **Cars**.

Id	**Model**	**Year**	**MakeId**
1	Camry	1999	101
2	Mustang	2002	102
3	Sienna	2000	101
4	Civic	1998	103
5	X3	2002	104

Note:

- Keep the properties, methods and classes as **public.**
- Please read the method rules **clearly**.
- Do not use **Environment.Exit()** to terminate the program.
- Do not change the given code template.

Sample Output :

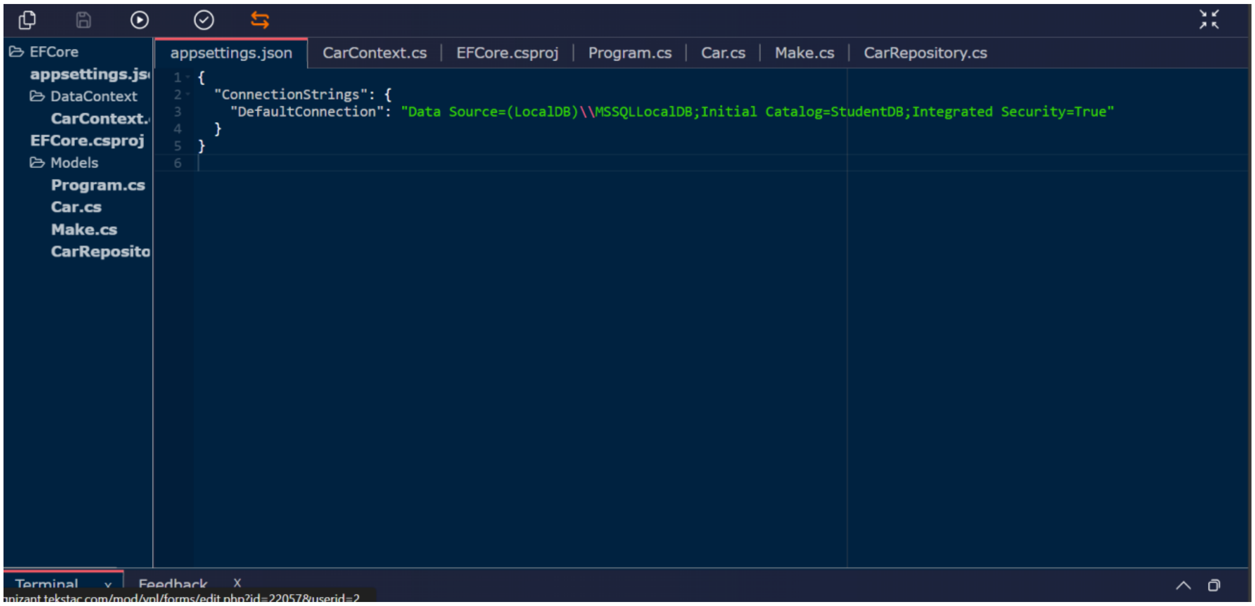
```
Car Id: 1,     Make: Toyota,     Model: Camry,      Year: 1999
```

Car Id: 2, Make: Ford, Model: Mustang, Year: 2002

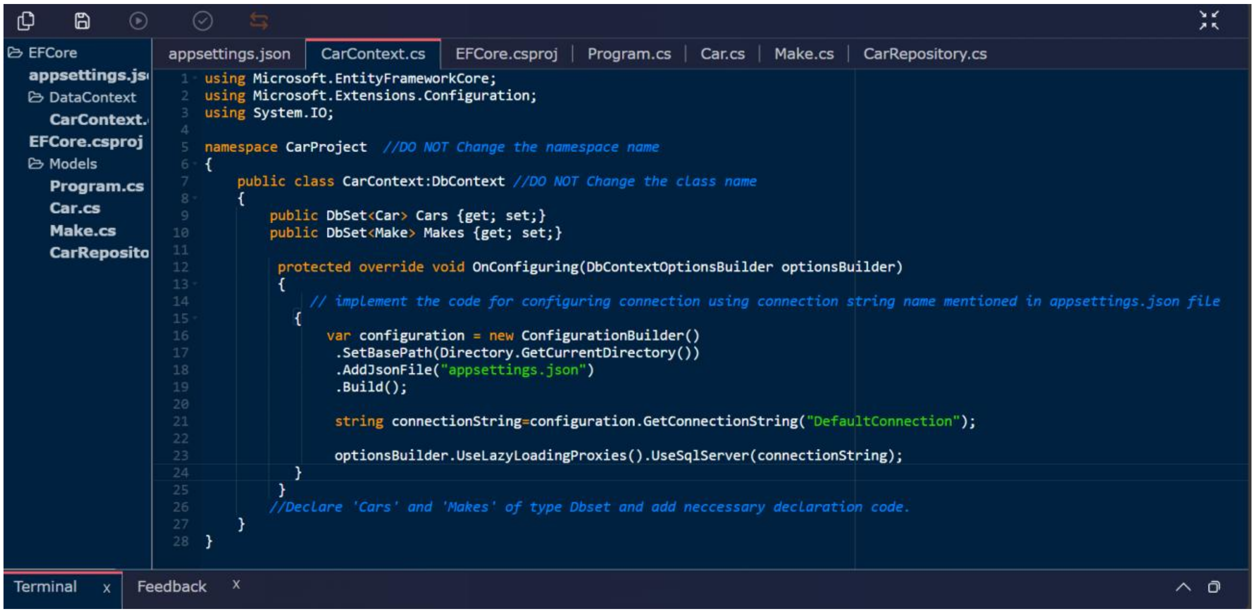
Car Id: 3, Make: Toyota, Model: Sienna, Year: 2000

Car Id: 4, Make: Honda, Model: Civic, Year: 1998

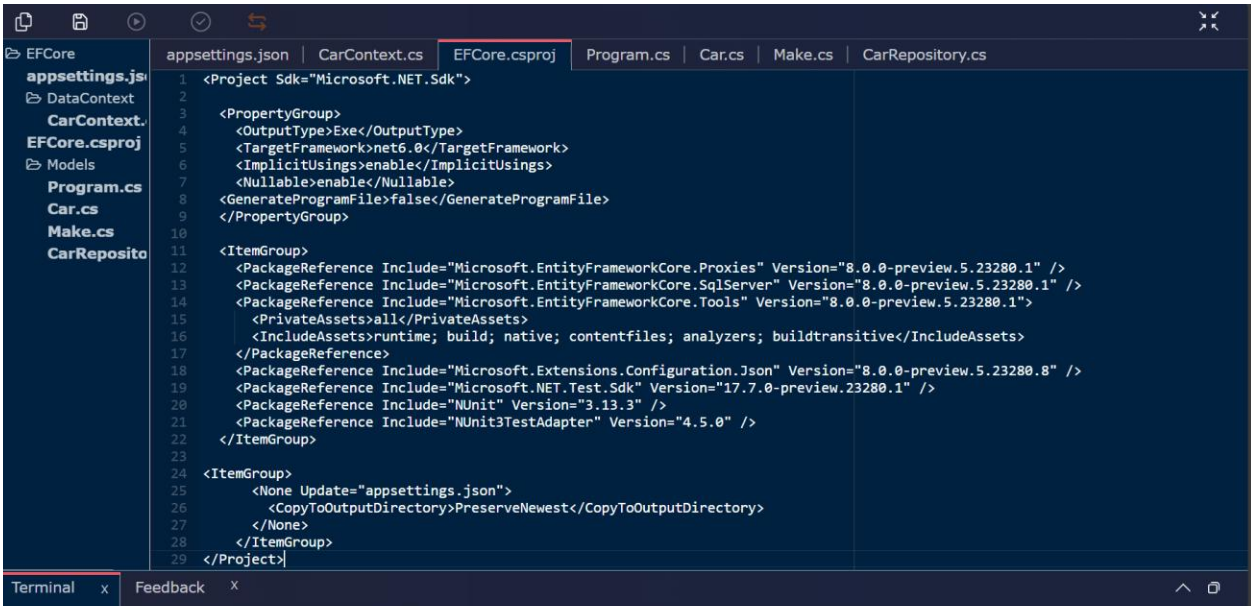
Car Id: 5, Make: BMW, Model: X3, Year: 2002



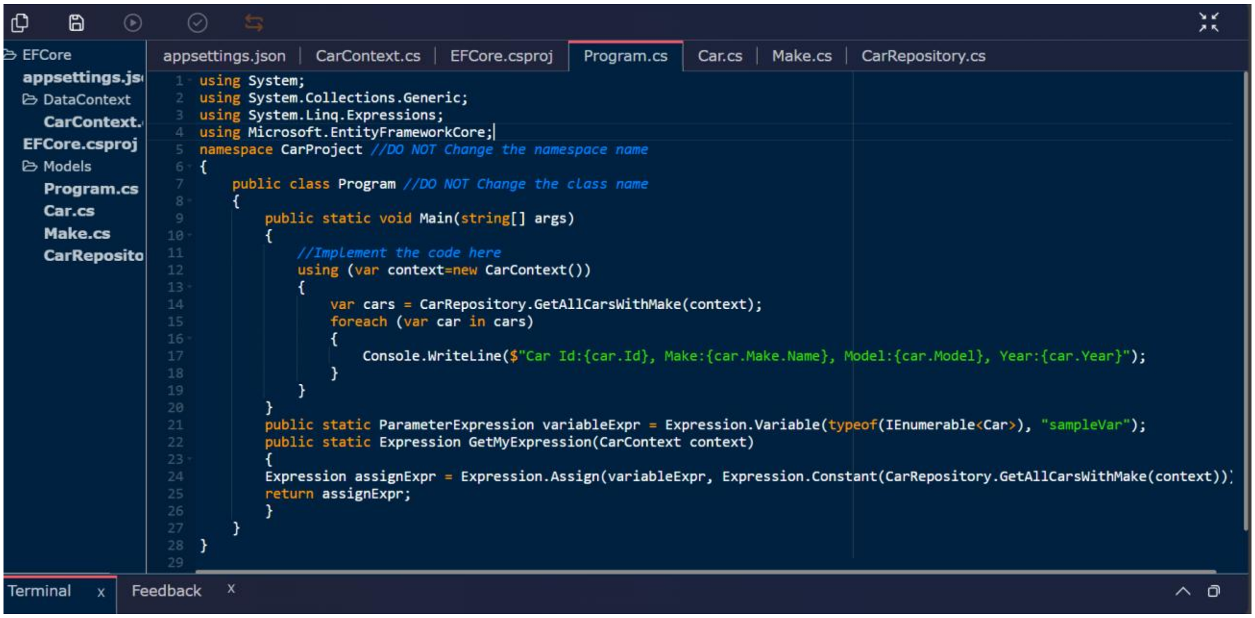
```
{
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial Catalog=StudentDB;Integrated Security=True"
  }
}
```



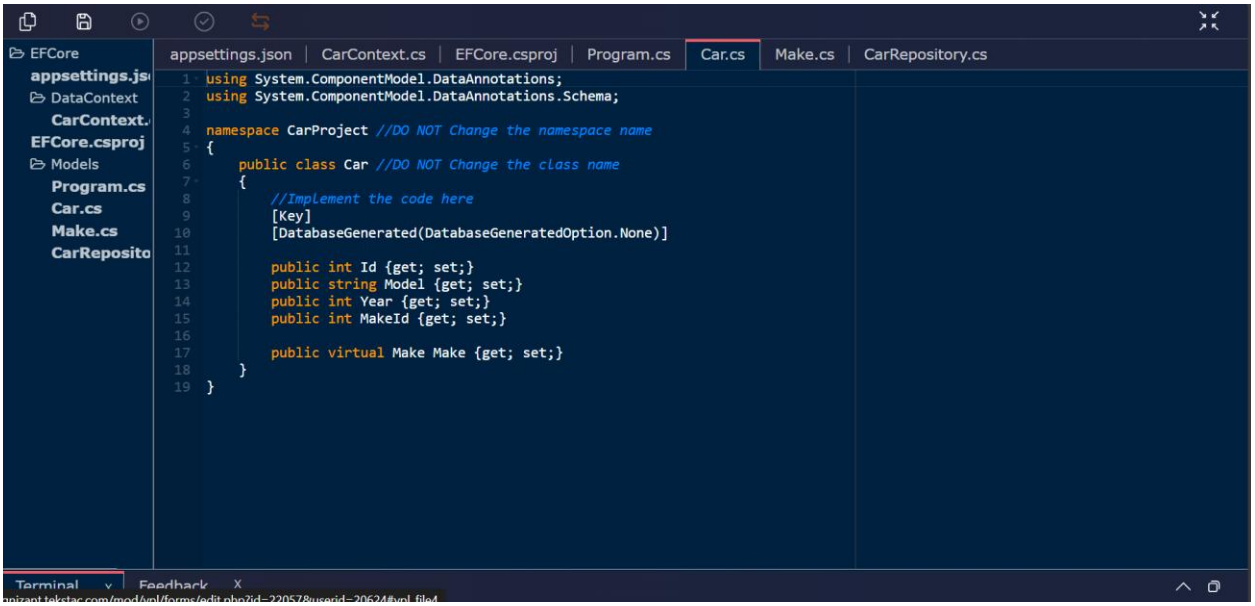
```
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using System.IO;

namespace CarProject  //DO NOT Change the namespace name
{
    public class CarContext:DbContext //DO NOT Change the class name
    {
        public DbSet<Car> Cars {get; set;}
        public DbSet<Make> Makes {get; set;}

         protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
         {
             // implement the code for configuring connection using connection string name mentioned in appsettings.json file
           {
               var configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("appsettings.json")
                .Build();

                string connectionString=configuration.GetConnectionString("DefaultConnection");

                optionsBuilder.UseLazyLoadingProxies().UseSqlServer(connectionString);
           }
         }
        //Declare 'Cars' and 'Makes' of type Dbset and add neccessary declaration code.
    }
}
```



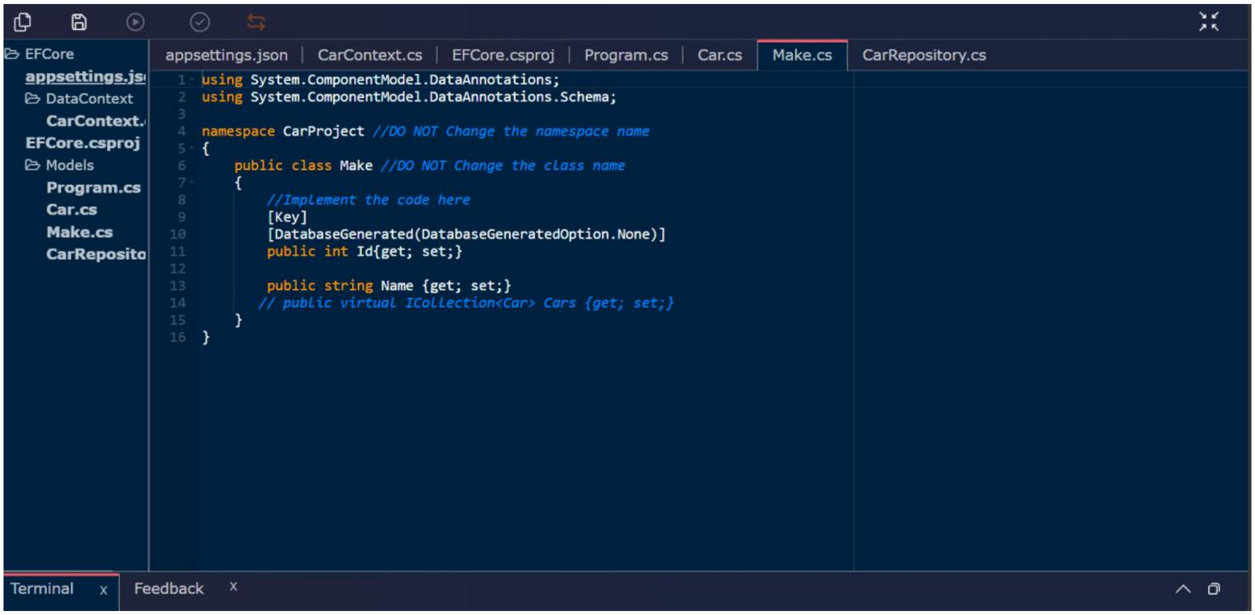
```xml
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net6.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  <GenerateProgramFile>false</GenerateProgramFile>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore.Proxies" Version="8.0.0-preview.5.23280.1" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.0-preview.5.23280.1" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="8.0.0-preview.5.23280.1">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.Extensions.Configuration.Json" Version="8.0.0-preview.5.23280.8" />
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.7.0-preview.23280.1" />
    <PackageReference Include="NUnit" Version="3.13.3" />
    <PackageReference Include="NUnit3TestAdapter" Version="4.5.0" />
  </ItemGroup>

<ItemGroup>
      <None Update="appsettings.json">
        <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
      </None>
    </ItemGroup>
</Project>
```



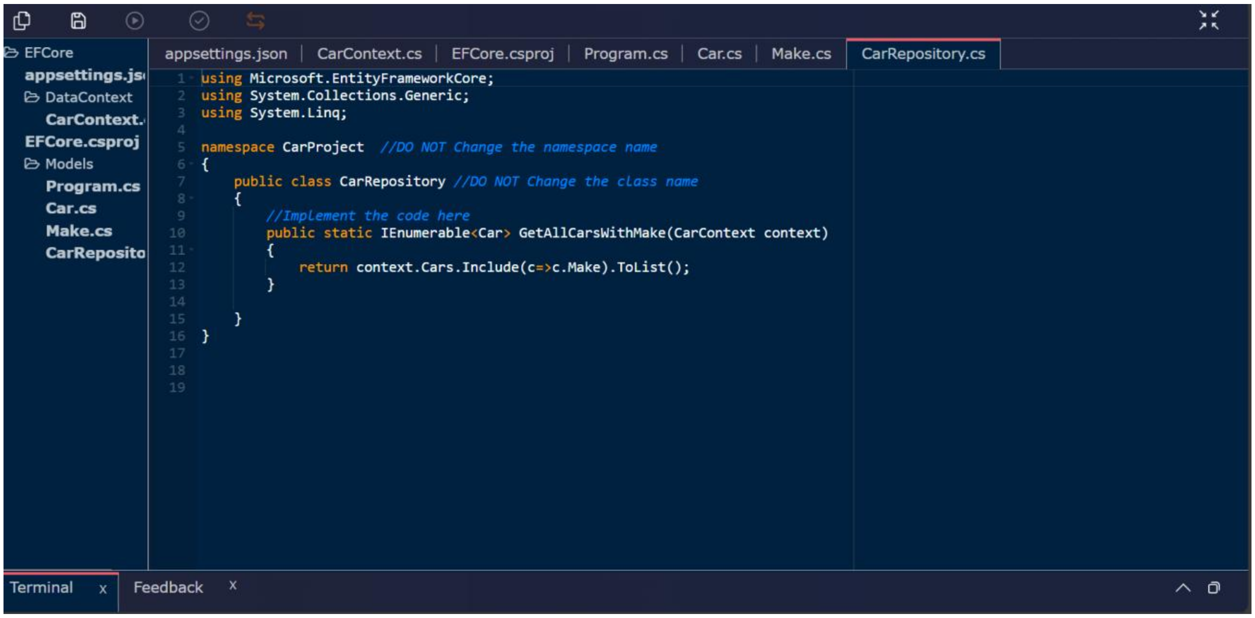
```csharp
using System;
using System.Collections.Generic;
using System.Linq.Expressions;
using Microsoft.EntityFrameworkCore;
namespace CarProject //DO NOT Change the namespace name
{
    public class Program //DO NOT Change the class name
    {
        public static void Main(string[] args)
        {
            //Implement the code here
            using (var context=new CarContext())
            {
                var cars = CarRepository.GetAllCarsWithMake(context);
                foreach (var car in cars)
                {
                    Console.WriteLine($"Car Id:{car.Id}, Make:{car.Make.Name}, Model:{car.Model}, Year:{car.Year}");
                }
            }
        }
        public static ParameterExpression variableExpr = Expression.Variable(typeof(IEnumerable<Car>), "sampleVar");
        public static Expression GetMyExpression(CarContext context)
        {
        Expression assignExpr = Expression.Assign(variableExpr, Expression.Constant(CarRepository.GetAllCarsWithMake(context))
        return assignExpr;
        }
    }
}
```



```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace CarProject //DO NOT Change the namespace name
{
    public class Car //DO NOT Change the class name
    {
        //Implement the code here
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.None)]

        public int Id {get; set;}
        public string Model {get; set;}
        public int Year {get; set;}
        public int MakeId {get; set;}

        public virtual Make Make {get; set;}
    }
}
```



```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace CarProject //DO NOT Change the namespace name
{
    public class Make //DO NOT Change the class name
    {
        //Implement the code here
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int Id{get; set;}

        public string Name {get; set;}
       // public virtual ICollection<Car> Cars {get; set;}
    }
}
```



```
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;
using System.Linq;

namespace CarProject  //DO NOT Change the namespace name
{
    public class CarRepository //DO NOT Change the class name
    {
        //Implement the code here
        public static IEnumerable<Car> GetAllCarsWithMake(CarContext context)
        {
            return context.Cars.Include(c=>c.Make).ToList();
        }

    }
}
```