ISC 559 – IS Application Design and Implementation
Assignment 3: Creating C# Classes and Setting Up Entity Framework (10 points)

In this assignment, we're going to create an MVC Core project that incorporates a database using the Code First approach for Entity Framework.

Phil is a concert promoter in Mobile. His job is to find performers that would like to come to Mobile and match them up with local venues. He also works with local production companies, who assist in setting up audio/video equipment at the venue of choice.

For this assignment, I'd like you to create a new MVC Core project. Make sure to follow the steps included in the "Setting up MVC Applications" handout including:

- program.cs file
- cs.proj file
- appsetting.json

Create a Models folder, and create 4 new C# classes in the folder: PerformingAct, Venue, ProductionCompany, and Event. Event will have a many-to-one relationship with each of the other classes (one ProductionCompany can have many Events, one PerformingAct can have many Events, and one Venue can have many Events; an Event can be associated with just one ProductionCompany, one PerformingAct, and one Venue).

PerformingAct should include the following properties: PerformingActId, Name, NumberOfPerformers (in case the act is a band or some other group), Manager, PerformerType (comedian, musician, etc.) and AverageAttendance (this is the average number of people who have attended this performers show in the past).

Venue should include the following properties: VenueId, Name, Address, MaxCapacity.

ProductionCompany should include the following properties: ProductionCompanyId, Name, Address, StaffSize.

The Event class should contain the following properties: EventId, EventDateTime, CurrentAttendance, and TicketPrice.

For all of your classes' Id properties, make sure you use int as the data type. Use the data types you think are most appropriate for other properties.

As you create your classes, make sure you also create the appropriate navigation properties so that when you migrate your models to Entity Framework, EF can understand the nature of the relationships. For example, An Event is associated with one Venue, so Event would need the following properties to allow EF to automatically map the relationship:

```
public int? VenueId { get; set; }
public virtual Venue Venue { get; set; }
```

For the other side of the relationship, a Venue can have many Events, so that gets represented as the following property in the Venue class:

```
public virtual ICollection<Event> Events { get; set; }
```

Follow the same pattern for mapping the rest of your classes' relationships.

Add an appsettings.json file to your project, and modify the connection string entry so that your database will be called EventPlanner.

For the database that we want to create based on our models, Event will have a many-to-one relationship with each of the other classes (one ProductionCompany can have many Events, one PerformingAct can have many Events, and one Venue can have many Events; an Event can be associated with just one ProductionCompany, one PerformingAct, and one Venue). In your context class, you will need to have a DbSet property for each of your classes. Make sure to name your properties appropriately for the names of your tables (i.e. the Venue class should be mapped to a table that will be named Venues).

With Entity Framework (using the Code First version as seen in the in-class lecture), create a database context class called EventPlannerContext for mapping your C# classes to database tables. You may create a separate folder for your context class, or you can place the context class in your Models folder with the rest of you classes. I'll leave this up to you.

Open the Package Manager Console and run the Add-Migration Initial and Update-Database commands to finish your Entity Framework setup.

And that's it! If you have any questions, please let me know.

For your submission, create a .zip file of your project and attach it.