EF Core Assessment: Single Entity Focus on StudentEnrollment

Objective:
This assessment is designed to test the student's understanding of Entity Framework Core (EF Core) with a focus on the StudentEnrollment entity. The student will be assessed on their ability to install and set up EF Core, create and map a model to a database table using the Code-First approach, perform data migrations, and execute CRUD operations. Additionally, the student will demonstrate their understanding of LINQ to Entities, data annotations, and validations.

Assessment Overview:

- **Total Time:** 2 hours
- **Total Marks:** 100

Part 1: EF Core Installation and Setup (10 Marks)

Task:

1. Create a new .NET Core Console Application project.
2. Install the required EF Core NuGet packages (Microsoft.EntityFrameworkCore, Microsoft.EntityFrameworkCore.SqlServer, and Microsoft.EntityFrameworkCore.Tools).

Instructions:

- Ensure that the packages are installed correctly by checking the csproj file and verifying the versions.

Deliverable:

- Submit a screenshot of the installed packages in the project.

Part 2: Code-First Approach and Entity Setup (20 Marks)

Task:

1. Create a StudentEnrollment entity with the following properties:
    - EnrollmentId (Primary Key)
    - StudentId (Foreign Key)
    - CourseId (Foreign Key)
    - EnrollmentDate (DateTime, required)
    - Grade (nullable decimal, optional)
2. Use data annotations to enforce the following validations:
    - EnrollmentDate is a required field.
    - Grade can be null but should have a valid decimal value if provided.

Instructions:

- Define the StudentEnrollment entity in a new class file.
- Apply data annotations to enforce the validations.

Deliverable:

- Submit the StudentEnrollment class code.

Part 3: Mapping and Database Context Setup (20 Marks)

Task:

1. Create a SchoolContext class that inherits from DbContext.
2. Configure the DbSet<StudentEnrollment> property within the SchoolContext.
3. Override the OnModelCreating method to configure the StudentEnrollment entity using Fluent API to ensure the following:
    - The EnrollmentId is the primary key.
    - The StudentId and CourseId are foreign keys.
    - The EnrollmentDate column is required.

Instructions:

- Ensure that the Fluent API configurations do not contradict the data annotations.

Deliverable:

- Submit the SchoolContext class code.

Part 4: Data Migrations (15 Marks)

Task:

1. Add an initial migration to create the database schema based on the StudentEnrollment entity.
2. Apply the migration to create the database.

Instructions:

- Use the EF Core CLI tools or Package Manager Console to create and apply migrations.
- Verify the database schema using SQL Server Management Studio (SSMS) or any other database management tool.

Deliverable:

- Submit the migration file and a screenshot of the database schema in SSMS.

Part 5: CRUD Operations (20 Marks)

Task:

1. Implement the following CRUD operations:

- **Create:** Insert a new StudentEnrollment record into the database.
- **Read:** Retrieve all StudentEnrollment records and display them.
- **Update:** Update the Grade of an existing StudentEnrollment record.
- **Delete:** Remove a StudentEnrollment record from the database.

Instructions:

- Implement these operations in the Main method of your console application.
- Ensure that the application correctly interacts with the database using the SchoolContext class.

Deliverable:

- Submit the code for the CRUD operations.

Part 6: LINQ to Entities (10 Marks)

Task:

1. Write a LINQ query to fetch all StudentEnrollment records where the Grade is above 80%.
2. Sort the results by EnrollmentDate in descending order.

Instructions:

- Ensure that the LINQ query is optimized and returns the correct results.

Deliverable:

- Submit the LINQ query code and the output results.

Part 7: Data Annotations and Validations (5 Marks)

Task:

1. Test the StudentEnrollment entity validations by attempting to add an Enrollment record with a missing EnrollmentDate or an invalid Grade.
2. Capture and display the validation errors in the console.

Instructions:

- Ensure that the validation errors are correctly captured and displayed.

Deliverable:

- Submit the code that demonstrates the validation testing and the console output showing the errors.

Part 8: Reflection and Understanding (10 Marks)

Task:

1. Write a brief reflection (200-300 words) on your experience with EF Core during this assessment. Focus on challenges faced, what you learned, and how you would approach similar tasks in the future.

Instructions:

- Provide a well-thought-out reflection that demonstrates your understanding and learning process.

Deliverable:

- Submit your written reflection.

Submission:

- Ensure that all parts of the assessment are completed and submitted in a single compressed file (zip) with a clear structure.
- Include a README file that briefly explains how to run your project.

Grading Criteria:

- **Correctness:** Does the solution meet the requirements?
- **Code Quality:** Is the code well-organized, with proper naming conventions and comments?
- **Understanding:** Does the reflection show a clear understanding of EF Core concepts?

Good luck!