1. Database schema and SQL scripts to create the tables

A screenshot of a computer

Description automatically generated with medium confidence

I have create database with code-first method. Code first is the approach where you write the classes that represent your data model in code, and then use EF to generate the database schema from them. This is useful if you want to have full control over your code and avoid editing the database manually.

The following steps were followed to implement the code-first approach:

1.Model Classes: Model classes were defined to represent the entities in the database. Each model class corresponds to a table in the database and includes properties that map to the table columns.

A screen shot of a computer program

Description automatically generated with medium confidence

2.DbContext: A DbContext class was created to provide a database context for the application. It inherits from the DbContext class provided by ASP.NET and includes DbSet properties for each model class representing a database table.

A picture containing text, screenshot, software, multimedia software

Description automatically generated

3.Database Configuration: In the DbContext class, the database connection string was configured using the OnConfiguring method, specifying the appropriate provider and connection details.

A picture containing screenshot, black

Description automatically generated

4.Database Creation: The database was created by calling the EnsureCreated method of the DbContext class. This method checks if the database exists and creates it if it doesn't.

A screenshot of a computer program

Description automatically generated with low confidence

5.Migrations: Migrations were enabled to manage future changes to the database schema. Migration files were created using the Package Manager Console commands Add-Migration [MigrationName] and applied using the Update-Database command.

A picture containing text, software, multimedia software, graphics software

Description automatically generated

3.API documentation with sample requests and responses

**Get Items:**

A screenshot of a computer

Description automatically generated with medium confidence

**Get Item by ID:**

A screenshot of a computer

Description automatically generated with medium confidence

Post Item:

A screenshot of a computer

Description automatically generated with medium confidence

Put Item:

A screenshot of a computer

Description automatically generated with medium confidence

Delete Item:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

TodoList project aimed to create a web application that allows users to manage their tasks and organize their daily activities effectively.

Approach:

1.Database Design: The database schema was designed to store the necessary information, such as tasks, task categories, due dates, and user details. Entity Framework Core was utilized as an ORM (Object-Relational Mapping) tool to facilitate database operations.

2.Backend Development: Controllers and models were implemented using C# within the ASP.NET Controllers handled user requests, retrieved data from the database, and orchestrated the flow of information from models. Models represented the business logic and interacted with the database using Entity Framework Core.