***Create a table named Customer\_review in mssql server and perform crud operation through apis with entityframework, microsoft design and tools***

Set Up Entity Framework in Your Project

dotnet add package Microsoft.EntityFrameworkCore dotnet add package Microsoft.EntityFrameworkCore.SqlServer dotnet add package Microsoft.EntityFrameworkCore.Tools

5. Configure the PortfolioContext in Startup.cs or Program.cs

In Startup.cs or Program.cs, register the database context:

csharp

Copy code

services.AddDbContext<PortfolioContext>(options =>

options.UseSqlServer(Configuration.GetConnectionString("PortfolioDb")));

6. Create and Apply Migrations

Run the following commands to create and apply the initial migration:

bash

Copy code

dotnet ef migrations add InitialCreate

dotnet ef database update

7. Create CRUD Endpoints in a Controller

Create a new controller, CustomerReviewController, and add CRUD actions:

csharp

Copy code

[Route("api/[controller]")]

[ApiController]

public class CustomerReviewController : ControllerBase

{

private readonly PortfolioContext \_context;

public CustomerReviewController(PortfolioContext context)

{

\_context = context;

}

[HttpGet]

public async Task<ActionResult<IEnumerable<CustomerReview>>> GetReviews()

{

return await \_context.CustomerReviews.ToListAsync();

}

[HttpGet("{id}")]

public async Task<ActionResult<CustomerReview>> GetReview(int id)

{

var review = await \_context.CustomerReviews.FindAsync(id);

if (review == null) return NotFound();

return review;

}

[HttpPost]

public async Task<ActionResult<CustomerReview>> CreateReview(CustomerReview review)

{

\_context.CustomerReviews.Add(review);

await \_context.SaveChangesAsync();

return CreatedAtAction(nameof(GetReview), new { id = review.Id }, review);

}

[HttpPut("{id}")]

public async Task<IActionResult> UpdateReview(int id, CustomerReview review)

{

if (id != review.Id) return BadRequest();

\_context.Entry(review).State = EntityState.Modified;

await \_context.SaveChangesAsync();

return NoContent();

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteReview(int id)

{

var review = await \_context.CustomerReviews.FindAsync(id);

if (review == null) return NotFound();

\_context.CustomerReviews.Remove(review);

await \_context.SaveChangesAsync();

return NoContent();

}

}

8. Test Your API Endpoints

Use tools like Postman or Swagger to test the CRUD operations.