**LAB # 10**

**Task # 01: Create a microservices for banking system**

**Solution:**

**Bank\_Transaction Model:**

public class Bank\_Transaction

{

[Key]

public int TransactionId { get; set; }

public string AccountNumber { get; set; }

public double Amount { get; set; }

public DateTime TransactionDate { get; set; }

public string TransactionType { get; set; }

}

**BankRepository.cs:**

namespace BankingMicroService.Repository

{

public class BankRepository : IBankRepository

{

private readonly TransactionContext \_context;

public BankRepository(TransactionContext context)

{

\_context = context;

}

public void AddTransaction(Bank\_Transaction transaction)

{

\_context.Bank\_Transactions.Add(transaction);

\_context.SaveChanges();

}

public IEnumerable<Bank\_Transaction> GetAllTransactions()

{

return \_context.Bank\_Transactions.ToList();

}

public Bank\_Transaction GetTransactionById(int transactionId)

{

return \_context.Bank\_Transactions.FirstOrDefault(t => t.TransactionId == transactionId);

}

}

}

**IBankRepository.cs:**

public interface IBankRepository

{

void AddTransaction(Bank\_Transaction transaction);

Bank\_Transaction GetTransactionById(int transactionId);

IEnumerable<Bank\_Transaction> GetAllTransactions();

}

**TransactionContext.cs:**

namespace BankingMicroService.DBContext

{

public class TransactionContext : DbContext

{

public TransactionContext(DbContextOptions<TransactionContext> options) : base(options)

{

}

public DbSet<Bank\_Transaction> Bank\_Transactions { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Bank\_Transaction>().HasData(new Bank\_Transaction

{

TransactionId = 1,

AccountNumber = "123456789",

Amount = 100.00,

TransactionDate = DateTime.Now,

TransactionType = "Deposit"

},

new Bank\_Transaction

{

TransactionId = 2,

AccountNumber = "987654321",

Amount = 50.00,

TransactionDate = DateTime.Now,

TransactionType = "Withdrawal"

}

);

}

}

}

**BankController:**

namespace BankingMicroService.Controllers

{

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

[ApiController]

public class BankController : ControllerBase

{

private readonly IBankRepository \_bankRepository;

public BankController(IBankRepository bankRepository)

{

\_bankRepository = bankRepository;

}

[HttpPost]

public IActionResult AddTransaction(Bank\_Transaction transaction)

{

\_bankRepository.AddTransaction(transaction);

return Ok();

}

[HttpGet]

public IActionResult GetAllTransactions()

{

var transactions = \_bankRepository.GetAllTransactions();

return Ok(transactions);

}

[HttpGet("{transactionId}")]

public IActionResult GetTransactionById(int transactionId)

{

var transaction = \_bankRepository.GetTransactionById(transactionId);

if (transaction == null)

{

return NotFound();

}

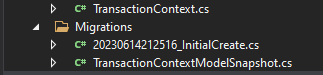
return Ok(transaction);

}

}

}

**Migration Output:**



**Connection String:**

"ConnectionStrings": {

"BankDB": "Data Source=DESKTOP-27TFO05;Initial Catalog=BankDB;Integrated Security=True"

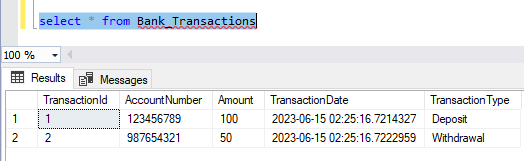
}

**Added Services:**

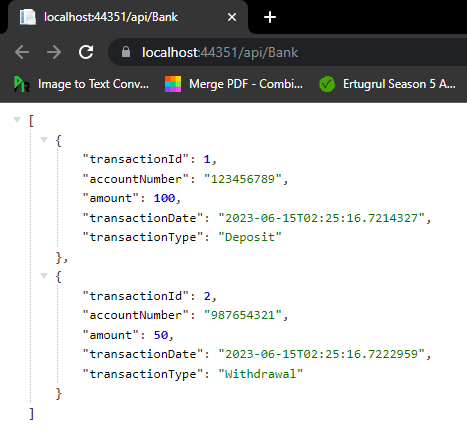
services.AddDbContext<TransactionContext>(o => o.UseSqlServer(Configuration.GetConnectionString("BankDB")));

services.AddTransient<IBankRepository, BankRepository>();

**DB After Creation Of Microservices:**



**Running Microservice:**



**Task # 02: Implement example**

**Solution:**

**Product Model:**

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public decimal Price { get; set; }

public int CategoryId { get; set; }

}

**Category Model:**

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

public string Description { get; set; }

}

**IProduct Repostiory:**

public interface IProductRepository

{

IEnumerable<Product> GetProducts();

Product GetProductByID(int ProductID);

void InsertProduct(Product product);

void DeleteProduct(int ProductID);

void UpdateProduct(Product product);

void Save();

}

**Product Repository:**

namespace ProductMicroService.Repository

{

public class ProductRepository : IProductRepository

{

private readonly ProductContext \_dbContext;

public ProductRepository(ProductContext dbContext)

{

\_dbContext = dbContext;

}

public void DeleteProduct(int ProductID)

{

var product = \_dbContext.Products.Find(ProductID);

\_dbContext.Products.Remove(product);

Save();

}

public Product GetProductByID(int ProductID)

{

return \_dbContext.Products.Find(ProductID);

}

public IEnumerable<Product> GetProducts()

{

return \_dbContext.Products.ToList();

}

public void InsertProduct(Product product)

{

\_dbContext.Add(product);

Save();

}

public void Save()

{

\_dbContext.SaveChanges();

}

public void UpdateProduct(Product product)

{

\_dbContext.Entry(product).State = EntityState.Modified;

Save();

}

}

}

**Product Controller:**

namespace ProductMicroService.Controllers

{

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

[ApiController]

public class ProductController : ControllerBase

{

private readonly IProductRepository \_productRepository;

public ProductController(IProductRepository productRepository)

{

\_productRepository = productRepository;

}

[HttpGet]

public IActionResult Get()

{

var products = \_productRepository.GetProducts(); return new OkObjectResult(products);

}

[HttpGet("{id}", Name = "Get")]

public IActionResult Get(int id)

{

var product = \_productRepository.GetProductByID(id); return new OkObjectResult(product);

}

[HttpPost]

public IActionResult Post([FromBody] Product product)

{

using (var scope = new TransactionScope())

{

\_productRepository.InsertProduct(product); scope.Complete();

return CreatedAtAction(nameof(Get), new { id = product.Id }, product);

}

}

[HttpPut]

public IActionResult Put([FromBody] Product product)

{

if (product != null)

{

using (var scope = new TransactionScope())

{

\_productRepository.UpdateProduct(product); scope.Complete();

return new OkResult();

}

}

return new NoContentResult();

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

\_productRepository.DeleteProduct(id); return new OkResult();

}

}

}

**ProductContext Class:**

namespace ProductMicroService.DBContexts

{

public class ProductContext : DbContext

{

public ProductContext(DbContextOptions<ProductContext> options) : base(options)

{

}

public DbSet<Product> Products { get; set; }

public DbSet<Category> Categories { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Category>().HasData(new Category

{

Id = 1,

Name = "Electronics",

Description = "Electronic Items",

},

new Category

{

Id = 2,

Name = "Clothes",

Description = "Dresses",

},

new Category

{

Id = 3,

Name = "Grocery",

Description = "Grocery Items",

}

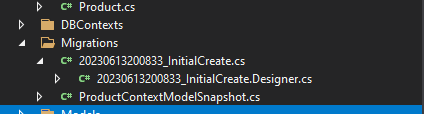
);

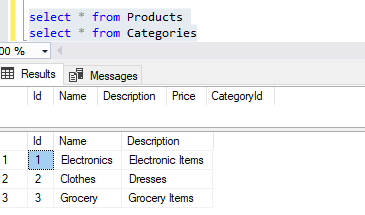
}

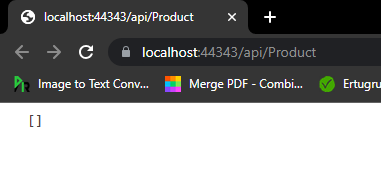
}

}

**Migration Output:**



**DB After Creation:**  

**Running Microservice:**  

It is showing empty because no value is added in Product.

Now, adding value by using postman,

