Capstone Project: Building a Banking Application with ASP.NET Core Web API and Angular Using Clean Architecture

1. Overview

The banking system should allow users to create accounts, deposit and withdraw funds, transfer money, view transaction history, and manage their profiles. Admins can manage users, transactions, and security settings.

2. Technology Stack

- Backend: ASP.NET Core Web API

- Frontend: Angular

- Database: SQL Server

- ORM: Entity Framework Core

- Authentication: JWT Authentication

- Exception Handling: Global exception handling middleware

- Architecture: Clean Architecture

3. Clean Architecture Implementation

a) Domain Layer (Core Business Logic)

Defines entities such as User, Account, Transaction, etc. Contains business rules and validation logic.

Models:

- User (Id, Name, Email, Password, Role)

- Account (Id, UserId, AccountNumber, Balance, AccountType, CreatedDate)

- Transaction (Id, AccountId, Type, Amount, Date, Description)

b) Application Layer (Use Cases and Services)

Implements interfaces for business logic. Contains services such as AccountService, TransactionService, and UserService.

Interfaces:

- IAccountRepository

- ITransactionRepository

- IUserRepository

c) Infrastructure Layer (Database and External Services)

Implements repositories for data access using Entity Framework Core.

d) Presentation Layer (API Controllers)

Exposes endpoints for the Angular frontend. Uses MediatR for CQRS (Command Query Responsibility Segregation).

Controllers:

- AccountsController

- TransactionsController

- UsersController

4. Backend Implementation (ASP.NET Core Web API)

- Authentication & Authorization: Implement using JWT tokens.

- Controllers & Endpoints: RESTful API with endpoints for account management, transactions, and user authentication.

- Dependency Injection: Ensure loose coupling.

5. Exception Handling in Web API

Custom Exceptions: Defines exceptions such as:

- AccountNotFoundException

- InsufficientBalanceException

6. Frontend Implementation (Angular)

- Angular Modules: Organized into feature-based modules such as AccountModule, TransactionModule, UserModule.

- Services: HTTP services for API communication.

- Error Handling: Implement interceptors to handle HTTP errors globally.

Angular Services:

- AccountService

- TransactionService

- AuthService

7. Database Design

Tables:

- Users

- Accounts

- Transactions

Relationships:

- One-to-many between Users and Accounts

- One-to-many between Accounts and Transactions