Simple User and Account Management System

# 1. Project Overview

The Simple User and Account Management System is designed to simulate core functionalities of a banking system where users can create and manage accounts across multiple banks, perform transactions, and have role-based permissions.

# 2. Project Scope

Users: Normal users and Bank users.  
Accounts: Savings, Current, and Term Deposit accounts.  
Banks: Each bank can have multiple branches.  
Access Control: Role-based permissions.  
Operations: Deposit, Withdraw, Close Account, Check Balance.  
API Exposure: REST APIs to manage Users, Banks, Accounts, and Authentication.

# 3. System Architecture

Architecture Type: Layered (3-tier Architecture)

1. Presentation Layer (API Controllers)  
2. Business Layer (Services)  
3. Data Layer (Entity Framework Core + SQL Server)

# 4. Entities and Relationships

User, Bank, Branch, Account (Base), SavingsAccount, CurrentAccount, TermDepositAccount, Role, and Permission.

# 5. Database Design

Database: BankingDB  
Includes tables for Users, Banks, Branches, Accounts (with inheritance), Roles, Permissions, and mapping tables.

# 6. Technology Stack

Backend: .NET 8 / ASP.NET Core Web API  
ORM: Entity Framework Core  
Database: SQL Server  
Authentication: JWT  
Language: C#

# 7. Development Steps

1. Create project using 'dotnet new webapi'.  
2. Add EF Core packages.  
3. Define models and DbContext.  
4. Create migrations and update database.  
5. Seed initial data.  
6. Implement REST controllers.  
7. Add JWT authentication.  
8. Test via Swagger.

# 8. REST API Design

Endpoints:  
/user - Create, Update, Delete, Read  
/accounts - CRUD Operations  
/banks/accounts/user/{id} - Get user accounts  
/login - Authenticate and assign roles

# 9. Role & Permission Mapping

Admin: Create, Update, Delete, Read  
User: Read, Limited Create  
Manager: Create, Update, Read

# 10. Testing Process

Unit Testing, Integration Testing, Manual Testing via Swagger, Input Validation.

# 11. Deliverables

1. Source Code  
2. SQL Database  
3. REST API Documentation  
4. ER Diagram  
5. Project Report

# 12. Future Enhancements

Add transaction logs, KYC verification, notifications, and an admin dashboard.

# 13. Conclusion

This project demonstrates entity relationships using EF Core, REST API design, and secure role-based access control for a banking-style system.