Bank Account API - Project Plan & Developer Guide

Main Goal

Build a simple, file-based banking API using ASP.NET Core Web API.
The goal is to let users:
- Open accounts
- Deposit and withdraw funds
- Update and soft-delete accounts
- View account info
All stored in a JSON file for persistence.

Architecture

1. BankAccount.cs - Represents a single account.
2. Bank.cs - Handles logic for managing accounts and file saving.
3. AccountsController.cs - Web API controller exposing HTTP routes.
4. accounts.json - Stores persistent data.

Models and Methods

BankAccount.cs
- Properties:
  int AccountID
  string Owner
  decimal Balance
  bool IsDeleted
- Methods:
  void Deposit(decimal amount)
  void Withdraw(decimal amount)
  void UpdateOwner(string newOwner)
  void MarkAsDeleted()
  bool IsActive()
  override string ToString()
  static void SetLastAccountId(int maxId)

Bank.cs
- Methods:
  List<BankAccount> GetAllAccounts()
  BankAccount? GetAccountById(int id)
  BankAccount CreateAccount(string owner, decimal initialBalance)
  bool Deposit(int id, decimal amount)
  bool Withdraw(int id, decimal amount)
  bool DeleteAccount(int id)
  bool UpdateAccount(int id, string newOwner)

Bank Account API - Project Plan & Developer Guide

List<BankAccount> GetAccountsByOwner(string owner)
void SaveAccounts()
List<BankAccount> LoadAccounts()

API Endpoints (AccountsController.cs)

POST /api/accounts
- Create new account with owner name and optional initial balance

GET /api/accounts
- Get all non-deleted accounts

GET /api/accounts/{id}
- Get account by ID

GET /api/accounts/by-owner/{owner}
- Get accounts by owner name

PUT /api/accounts/{id}
- Update owner name

POST /api/accounts/{id}/deposit
- Deposit funds into account

POST /api/accounts/{id}/withdraw
- Withdraw funds from account

DELETE /api/accounts/{id}
- Soft-delete an account

Project Structure

```
BankAPI/
 Controllers/
   AccountsController.cs
 Models/
   Bank.cs
   BankAccount.cs
 accounts.json
 Program.cs
 BankAPI.csproj
```

Bank Account API - Project Plan & Developer Guide

Development Steps

1. Create the API project: dotnet new webapi -n BankAPI
2. Create Models/ and Controllers/ folders
3. Implement BankAccount.cs with methods for deposit/withdraw/etc.
4. Implement Bank.cs for managing multiple accounts
5. Implement AccountsController.cs to expose all endpoints
6. Use Swagger to test: http://localhost:5154/swagger
7. Inspect accounts.json for saved accounts

Learning Objectives

- Understand MVC-style API design
- Manage object lifecycles and persistence
- Handle edge cases: negative balance, deleted accounts
- Improve testability by separating logic and interface