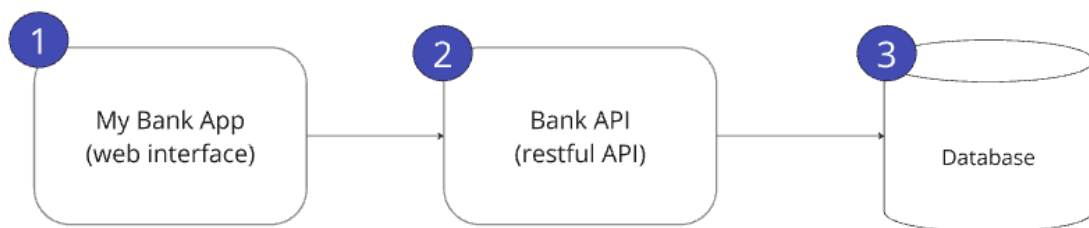


# Introduction

This document provides information about a banking application designed to enable users to log in, view their account dashboard, and transfer funds between accounts. The application offers a user-friendly interface for managing bank accounts and transactions, ensuring a seamless and efficient banking experience.

## Application Components



### 1. My Bank App

App is developed using React library. It utilizes component based development, state management using Redux store and role based access control.

#### a. Key Features

##### i. Login

The screenshot shows the 'My Bank' login page. It features a central 'Login' form with fields for 'Username' and 'Password', and a blue 'Login' button. The page has a light gray header with 'My Bank' and a footer with '© 2023 My Bank'.

#### Username/Password Error

This screenshot shows the login page with an error message: 'Username and password are required.' displayed in a red box above the 'Username' and 'Password' input fields. The 'Login' button remains visible at the bottom.

Login unsuccessful

My Bank

Log Out

Login

Login unsuccessful. Please check your username and password.

Username

account-holder1

Password

\*\*\*\*\*

Login

ii. Account Dashboard

My Bank

Log Out

Account Dashboard

Transfer Between Accounts

Account #1 - Balance: \$9,161.00

View Transactions

Account #2 - Balance: \$5,839.00

View Transactions

© 2025 My Bank

View Transactions

My Bank

Log Out

Account Dashboard

Transfer Between Accounts

Account #1 - Balance: \$9,458.00

View Transactions

Account #2 - Balance: \$5,542.00

View Transactions

Transactions for Account #1

Credit - \$1,000.00 on 2025-01-30

Debit - \$200.00 on 2025-01-30

Debit - \$250.00 on 2025-01-31

Debit - \$380.00 on 2025-01-31

Credit - \$297.00 on 2025-02-01

iii. Transfer Between Accounts

My Bank

Log Out

Transfer Between Accounts

From Account

Select Account

To Account

Select Account

Amount

Transfer

Back to Dashboard

© 2025 My Bank

From Account error

My Bank Log Out

### Transfer Between Accounts

Please select a From Account.

From Account  
Select Account

To Account  
Select Account

Amount

Transfer Back to Dashboard

#### Same Account error

My Bank Log Out

### Transfer Between Accounts

From Account and To Account cannot be the same.

From Account  
Account #1

To Account  
Account #1

Amount

Transfer Back to Dashboard

#### Amount error

My Bank Log Out

### Transfer Between Accounts

Please enter a valid amount greater than 0.

From Account  
Account #2

To Account  
Account #1

Amount

Transfer Back to Dashboard

#### Transfer successful

My Bank Log Out

### Transfer Between Accounts

Transfer successful.

From Account  
Select Account

To Account  
Select Account

Amount

Transfer Back to Dashboard

#### b. Architecture

The app is built using component-based architecture, the main components are:

- i. Header Component  
Displays the app's navigation links and the "Log Out" button. The "Log Out" button is conditionally shown based on the user's authentication status.
- ii. Footer Component  
Displays a simple footer with a centered copyright message.
- iii. Login Component  
Provides the login form for user authentication. Displays error messages if the username or password is missing or incorrect.
- iv. AccountDashboard Component

Displays the user's bank accounts and their balances. Allows users to view detailed transactions for each account.

v. Transfer Component

Provides the form for transferring funds between accounts. Validates the transfer details and displays error messages if necessary.

2. Bank API

API is developed using .Net Core 8 Web API. Database used is SQL Server.

a. API Endpoints

i. api/auth/register - POST

Allow new users to register.

ii. api/auth/login - POST

Allow users to log in.

iii. api/account/user/{userId} - GET

Retrieves the account details for a specific user.

iv. api/account/transfer - POST

Allows transfer between accounts.

b. Architecture

The code is separated into various projects to enable separation of concerns and ease the maintainability. Following .Net Core projects are created:

i. BankDash.API

Serves as the main entry point for the application's API. It defines the endpoints, routes, and controllers that handle HTTP requests and responses. It acts as the interface between the front-end.

ii. BankDash.Common

Common utilities, helper functions, and shared resources that can be used across different projects within the solution.

iii. BankDash.DB

Handles database-related operations, including the setup and configuration of the Entity Framework (EF) context, database migrations, and repository pattern implementation. It interacts with the database to perform CRUD operations.

iv. BankDash.Model

Defines the data models and DTOs (Data Transfer Objects) used throughout the application. These models represent the structure of data and are used for data exchange between different layers of the application.

v. BankDash.Service

Contains the business logic and services of the application. It implements the core functionality of the application, including various operations, rules, and validations.

vi. BankDash.UnitTests

Contains unit tests for the application. It ensures that the individual units of the application (such as methods) work correctly.

### 3. Database

