BASALT WALLET

M Mahlangu

Introduction

The purpose of this design document is to outline the architecture and functionalities of an application that allows users to create accounts, login, view transaction history, check their account balance, credit their accounts, and debit their accounts. The application will be developed using C# Web API Core .NET 6 on the backend, Angular on the frontend, and will follow a microservices architecture. It will use JWT Token for authentication, Postman Collections for testing APIs, MySQL for data storage, Git for version control, and CoinBase Commerce for crediting user accounts.

System Overview

The application will consist of the following components.

Frontend: Angular application responsible for user interfaces and interactions.

Backend: C# Web API Core .NET 6 microservices providing RESTful APIs for various functionalities.

Database: MySQL database to store user account information and transaction history.

CoinBase Commerce: External service used to handle the credit of user accounts.

JWT Token: Used for secure authentication and authorization.

3. Microservices Architecture

The application will be designed using a microservices architecture to decouple different functionalities into separate services. Each service will have its own dedicated purpose and will communicate through APIs.

Microservices:

User Service: Responsible for handling user account creation, login, and account-related operations.

Transaction Service: Manages transaction history and related operations.

Account Service: Handles account balance, credit, and debit operations.

Authentication Service: Responsible for user authentication using JWT Token.

Technology Stack

Backend: C# Web API Core .NET 6, JWT Token, MySQL Database, CoinBase Commerce API.

Frontend: Angular.

Tools: Postman Collections, Git.

API Endpoints and Functionality

Authentication Service API:

GET /api/Auth/CreateAccount: Create Account

POST /api/Auth/Login: Login

Account Service API:

GET /api/Account/{AccountId}/Balance: Get account balance for a specific user.

GET /api/Account/{ AccountId }/TransactionHistory: Get transaction history for a specific user.

POST /api/Account/{ AccountId }/Credit: Credit user's account using CoinBase Commerce.

POST /api/Account/{ AccountId }/Debit: Debit user's account.

Data Models

Account Model:

- AccountId (Guid)
- AccountHolderName (string)
- AccountNumber (string)
- Password (hashed string)

Transaction Model:

- TransactionId (Guid)
- AccountId (Guid)
- Amount (decimal)
- Date (DateTime)
- Balance (decimal)

Authentication and Authorization

The application will use JWT Token for user authentication and authorization. When a user logs in successfully, they will receive a JWT Token, which they need to include in the headers of subsequent API requests to access secured endpoints.

Database Design

The MySQL database will have the following tables:

Accounts

Transactions

External Service- CoinBase Commerce

CoinBase Commerce will be used to handle the credit of user accounts. It will be integrated into the Account Service API to facilitate account crediting operations.

Postman Collections

A Postman collection will be created to test the functionality of each API endpoint. It will include sample requests and expected responses for each API.

Version Control- Git

The source code will be version-controlled using Git. A centralized repository will be set up, and developers will work on feature branches. The code will go through code reviews, and once approved, it will be merged into the main branch.

Conclusion

This design document outlines the architecture and functionalities of the C# Web API Core .NET 6 and Angular app using a microservices architecture, JWT Token, MySQL, CoinBase Commerce, Postman Collections, and Git. It provides a solid foundation for the development and successful implementation of the application.