Wallet documentation

This is a simple wallet API that can be used as a base template for building a digital wallet for any application. Currently the wallet only supports the following:

* Wallet user creation
* Creating debit/credit transaction(s) to a user’s account
* Retrieving a user account’s transaction history
* Retrieving a user account current balance

# Getting started

Before you start, you’ll need the following installed on your machine:

* Visual studio 2022
* Postman
* Docker
* PgAdmin4
* Browser of your choice

# Running it locally

To run it locally once the prerequisite software has been installed firstly download/clone the code from the following repo <https://github.com/Themba-Nzimande/wallet>. This repo contains everything you need to get started.

**Run the monolith version of the application locally**

1. Make sure Docker desktop is installed and running. (Use the docker info command in cmd [docker info](https://docs.docker.com/config/daemon/troubleshoot/%23:~:text=The%20operating%2Dsystem%20independent%20way,service%20status%20using%20Windows%20utilities.))
2. Next you’ll need to create a container of the Postgres DB that the application will use. To do this use a cmd/terminal and navigate to “{your directory}/ Docker/db” and run ‘docker-compose up -d’. This will create a Postgres DB container with sample data.

A picture containing text, screenshot, display, software

Description automatically generated

1. Once your Postgres DB container is running you can open up the monolith version of the solution by double clicking the ‘Monolith.sln’ in the directory you cloned/downloaded in Step 1. This will open up the solution in Visual Studio 2022.
2. Then press Crtl+Shift+b to build the solution. Once the build is complete without errors you can run the solution by pressing the F5 key.
3. Once the application is running a swagger page will open up on your browser similar to the screenshot below. If you are familiar with swagger you can go ahead and test the API endpoints either using swagger or the postman collection (link)

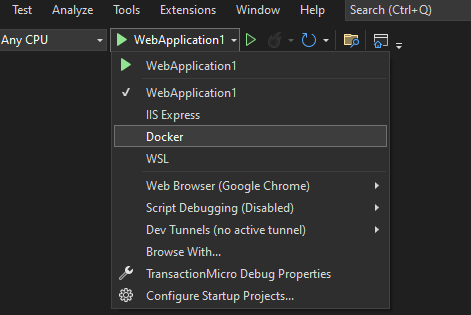
**Run the microservices version of the application locally**

1. Make sure Docker desktop is installed and running. (Use the docker info command in cmd [docker info](https://docs.docker.com/config/daemon/troubleshoot/%23:~:text=The%20operating%2Dsystem%20independent%20way,service%20status%20using%20Windows%20utilities.))
2. Next you’ll need to create a container of the Postgres DB that the application will use. To do this use a cmd/terminal and navigate to “{your directory}/ Docker/db” and run ‘docker-compose up -d’. This will create a Postgres DB container with sample data.

A picture containing text, screenshot, display, software

Description automatically generated

1. Once your Postgres DB container is running you can move onto running the microservices.
2. Navigate to the ‘micros’ directory and open the solution within each folder by double clicking the ‘.sln’ file in the folder directory you cloned/downloaded in Step 1. This will open up the solution in Visual Studio 2022.
3. Within each of the three solutions click the run button’s dropdown toggle and select docker to run the container version



1. When a container is running a swagger page will open up on your browser similar to the screenshot below. If you are familiar with swagger you can go ahead and test the API endpoints either using swagger or the postman collection (link)

A screenshot of a computer

Description automatically generated with medium confidence

# API endpoints

Below is a brief run down of the API endpoints available and what they do with status codes.

Api/Login

* Login:
  + Purpose: Used for user login
  + Returns:
    - 200 Ok with jwt string token for successful login
    - 403 Unauthorized for unsuccessful login with invalid login response message
* CreateAccount:
  + Purpose: Used for new user account creation
  + Returns:
    - 200 Ok with jwt string token for to skip process of logging in with newly created user account

Api/Transaction

* TransactionHistory:
  + Purpose: Get all the transactions for an account
  + Returns:
    - 200 Ok list of transaction objects with the following properties:
      * Transcation\_entry\_type – states whether transaction was a debit or credit type
      * Amount – amount for transaction
      * Transaction\_timestamp – timestamp of transaction
    - 403 Unauthorized when request is attempted with invalid jwt token
* Transact:
  + Purpose: Used transact on a user account
  + Returns:
    - 200 Ok with string on whether the transaction was successful or not.

Api/Account

* GetBalance:
  + Purpose: Get balance of a user account
  + Returns:
    - 200 Ok returns the account’s current balance
    - 403 Unauthorized when request is attempted with invalid jwt token