CMPG 323 Project 1

S. Strydom

orcid.org/0000-0000-0000-000X

Project documentation submitted for the degree *Bachelor Scientiae* in Information Technology at the North-West University

Lecturer: Dr. JT Janse van Rensburg
Mr. Zander Boonzaaier
Supervisor: Me. Chantal de Kock

Graduation: 2022

Student number: 24306223

TABLE OF CONTENTS

LIST OF FIGURES (HEADING 0)

CHAPTER 1 INTRODUCTION

1.1 Discovery account system

The project scope requires the coding of the Account system for the Discovery mobile application. This system manages Discovery Miles, the Active Rewards currency. Making use of the given technology stack, three backend services were coded: adding of Miles to Miles account, viewing of Miles in Miles account and subtraction of Miles from Miles account.

The project proved a challenge due to the fact that I was unfamiliar with the majority of technologies in the technology stack, necessitating effort to research and implement these technologies. This can also be regarded as a highlight of the project since I am now much better acquainted with these technologies specifically IntelliJ IDEA, GIT, Maven, Springboot and JPA. A challenge that hindered the development of the application was the failure of the Maven build due to its inability to find the main class. This problem was solved by restructuring of the project and reconfiguration of the run settings. This solution was only discovered after about 80% of the application was recoded.

The aim of the system is to provide backend services to view the current balance of Discovery Miles (or other currencies) of a member, to add Miles (or other currencies) to their current balance and to subtract Miles (or other currencies) from their current balance. The system is divided into five modules that is each handling different aspect of the application. The system Domain contains the Dto and entities (DB tables) code. The system Logic contains the logical flow code. The system Repository contains the SQL code. The system Translator contains the code that enables the Dtos and repositories to work together. The system Web module contains the REST API code.

CHAPTER 2 ENTITY RELATIONSHIP DIAGRAM

2.1 The Database and the ERD

An Entity Relationship Diagram (ERD) is used to visually represent the composition and structure of a database. As defined by the scope of the project, a database is required to store the data of the backend application we have to wright. The database needed to store the amount of Miles that a certain member has therefore the database must contain a MEMBER_ID and AMOUNT column. The date of transactions should also be recorded therefore a TX_DATE (transaction date) column is added to the database. The next criteria is to be able to use different currencies, thus a separate table to record the currencies. The currency table is named ACCOUNT_TYPE and has the following fields: ACCOUNT_TYPE_ID, MNEMONIC, ACCOUNT_TYPE and CREATION_DATE. We link the ACCOUNT_TYPE table to the ACCOUNT_TX table by ACCOUNT_TYPE_ID with a one to many relationship. Figure 2-1 expresses the database as an ERD.

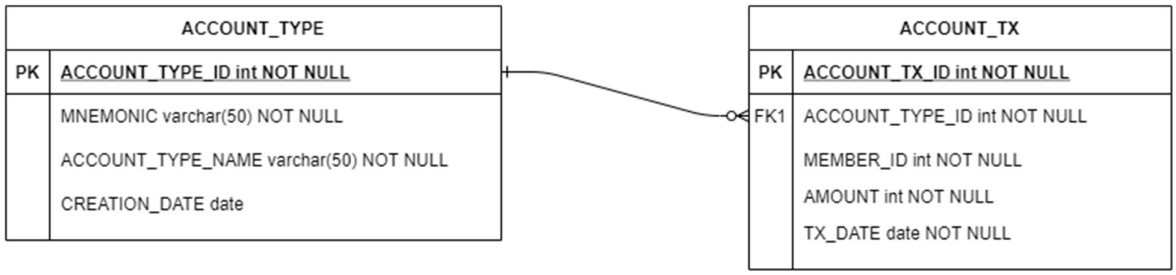

Figure 2-1: Entity Relationship Diagram.

CHAPTER 3 USE CASE DIAGRAM

3.1 Explaining the uses of the three backend services

It is planned that the three backend services (view Miles, add Miles, subtract, Miles) to be implemented by a front end. Thus the user or client will interact with a frontend(GUI) that will in turn call the services. The front end can be either a mobile application or a webpage in an Internet browser. The frontend will then call one of the backend services to either view the amount of Miles (or other currencies) they have available. Or they can add Miles (or other currencies) to their current balance or they can subtract Miles (or other currencies) from their current balance. The services will communicate with tod database and retrieve the requested data for the client.

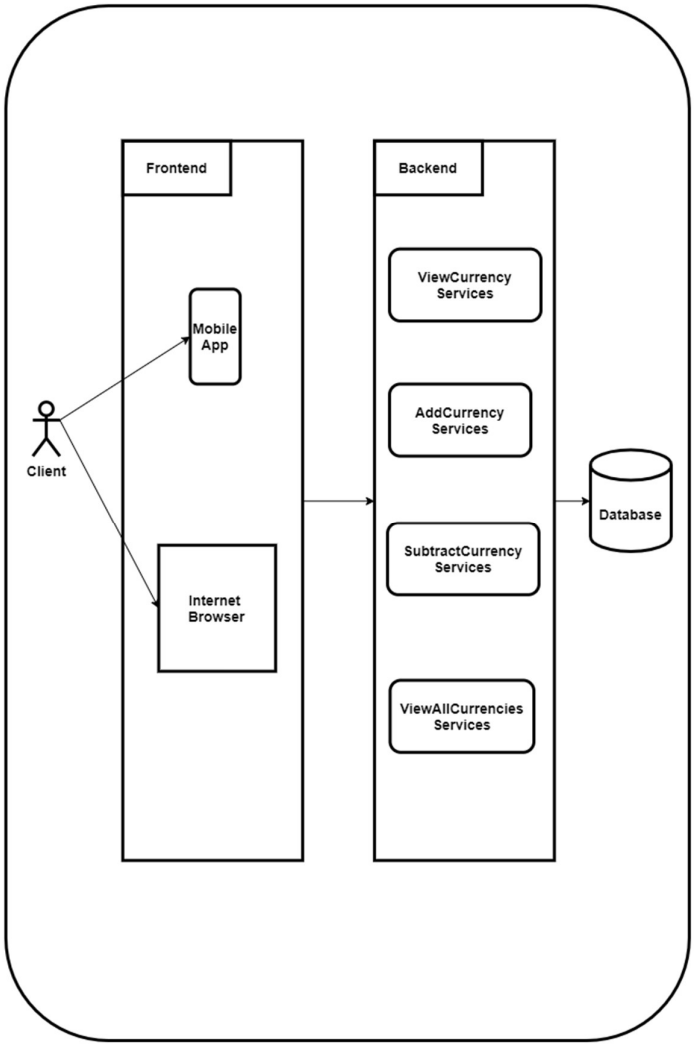

Figure 3-1: Use Case Diagram.

CHAPTER 4 FLOW DIAGRAMS

4.1 The logic flow

The logic flow of the three services are similar and is composed from seven interfaces and classes. The service Repository interface contains the SQL queries that is used on the database. The Service Dto class is a java object that represents the records of the table in the database where the methods represent the columns in the table. The service Translator interface brings together the Dto and the repository and allows them to interact. The service TranslatorImpl class is the implementation of the service translator interface and does some exception handling. The service Flow interface is the logical flow manager between the controller and the translator and helps with separation service calls. The service FlowImpl class is the implementation of the Flow interface. Lastly there is the service Controller class which interacts with the frontend, it also has some error handling and is responsible to initiate the service that is requested from the frontend and return the information recovered from the service to the frontend.

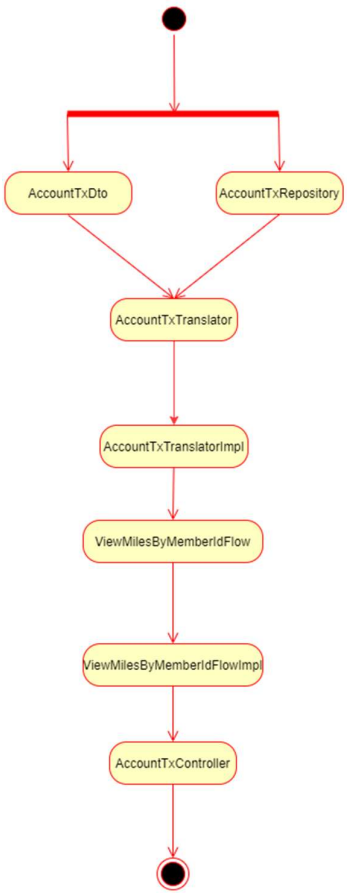

Figure 4-1: Logic flow for the ViewMilesByMemberId service

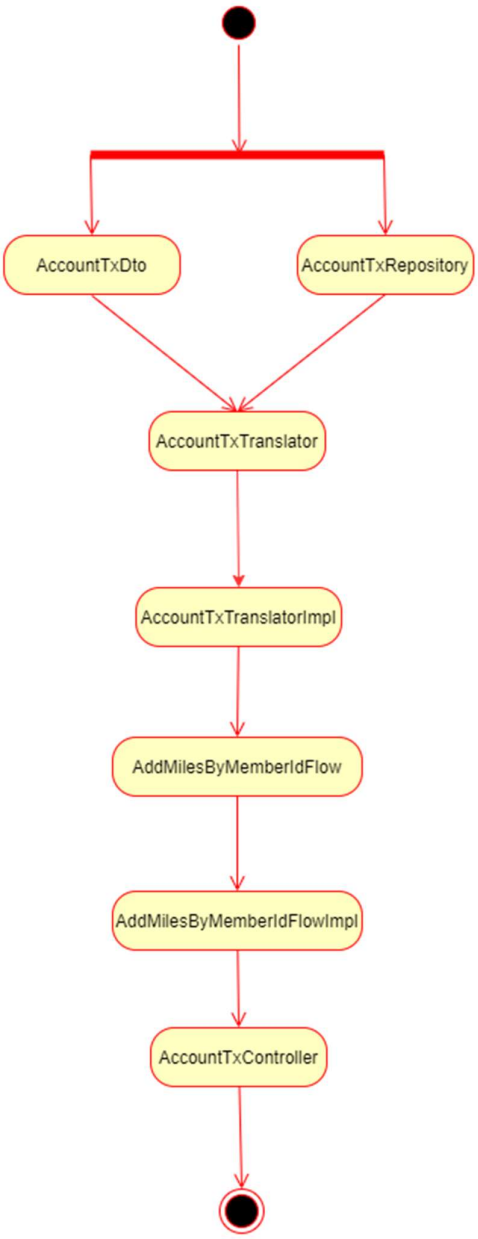

Figure 4-2: Logic flow for the AddMilesByMemberId service

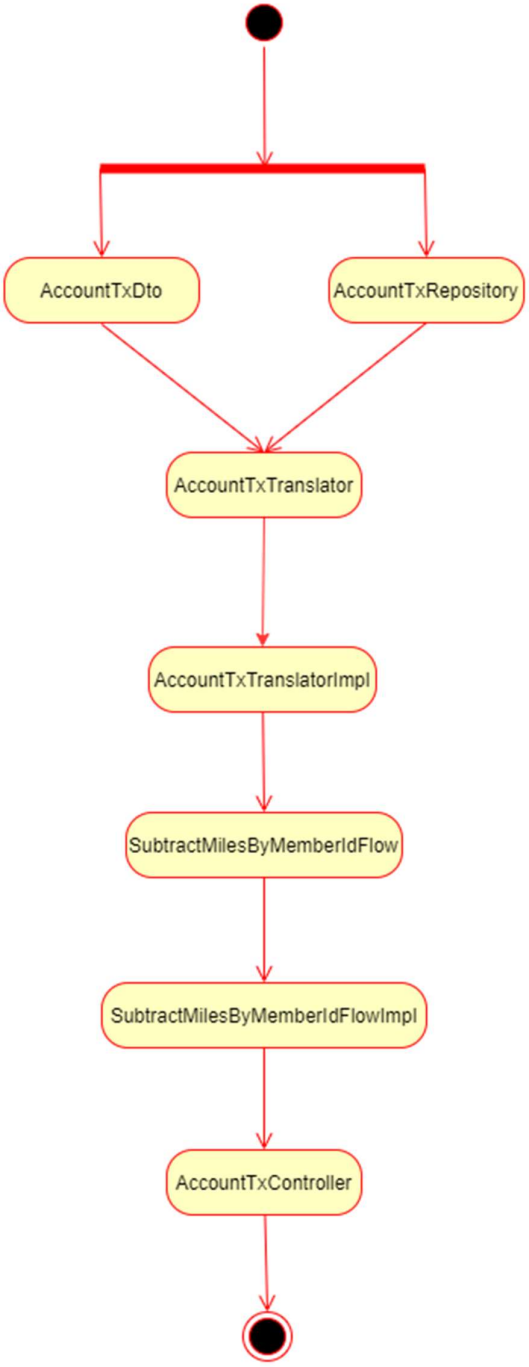

Figure 4-3: Logic flow for the SubtractMilesByMemberId service

CHAPTER 5 USER HOW-TO GUIDE

5.1 How to interact with the application

The frontend for the application does not yet exist but the backend services can easily be interacted with vai the integrated Swagger dependencies. Swagger supplies a UI that can be used to interact with the backend code and test the services.

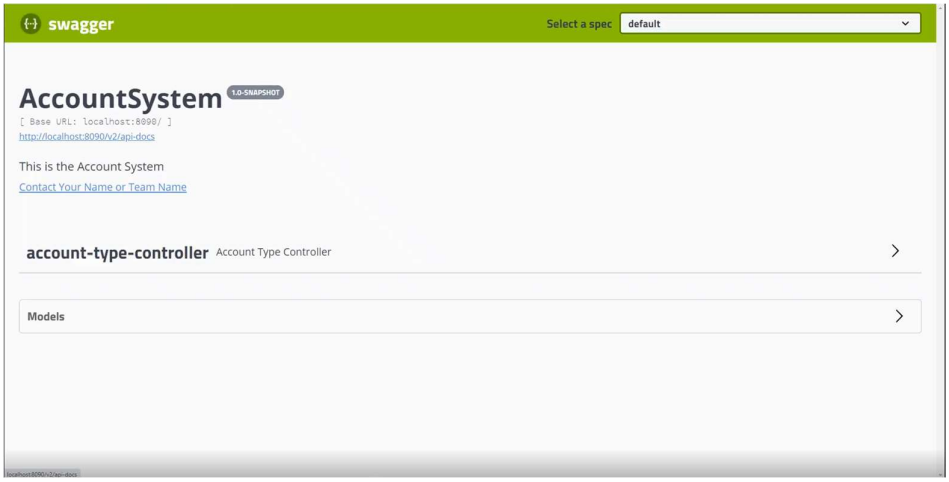

Figure 5-1: Swagger main page

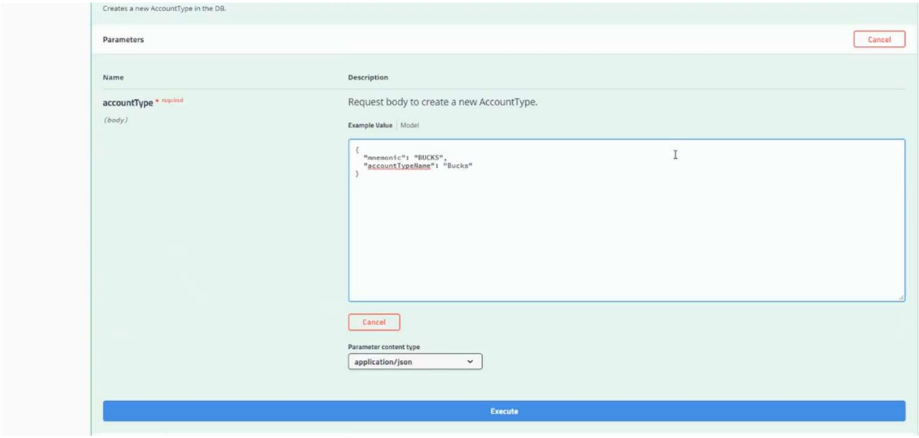

Figure 5-2: Swagger service test

CHAPTER 6 CODE COVERAGE REPORT

6.1 The percentage of code that is tested by the unit tests

Code coverage is simply the percentage of code that is tested and passes in the unit testing. The code coverage for this application is done by jacoco dependencies and examples of generated reports are shown in figure 6-1 and 6-2.

```
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ AccountSystemWeb ---
[INFO] Installing C:\DO\nwu\AccountSystem\AccountSystemWeb\target\AccountSystemWeb-1.0-SNAPSHOT.war to C:\tools\repository\za\ac\nwu\ac\AccountSystemWeb\1.0-SNAPSHOT\AccountSystemWeb-1.0
[INFO] Installing C:\DO\nwu\AccountSystem\AccountSystemWeb\pom.xml to C:\tools\repository\za\ac\nwu\ac\AccountSystemWeb\1.0-SNAPSHOT\AccountSystemWeb-1.0-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] AccountSystem ..................................... SUCCESS [  1.409 s]
[INFO] AccountSystemDomain ............................... SUCCESS [  5.607 s]
[INFO] AccountSystemRepository ........................... SUCCESS [ 15.833 s]
[INFO] AccountSystemTranslator ........................... SUCCESS [  0.735 s]
[INFO] AccountSystemLogic ................................ SUCCESS [  4.499 s]
[INFO] AccountSystemWebSpringBoot ........................ SUCCESS [ 14.590 s]
[INFO] AccountSystemWeb .................................. SUCCESS [  1.776 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 46.056 s
[INFO] Finished at: 2021-09-13T22:48:32+02:00
[INFO] Final Memory: 57M/688M
[INFO] ------------------------------------------------------------------------

Process finished with exit code 0
```

Figure 6-1: Example of the IDE report

AccountSystemLogic > za.ac.nwu.ac.logic.flow.impl

za.ac.nwu.ac.logic.flow.impl

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
CreateAccountTransactionFlowImpl		0%		0%	3	3	16	16	2	2	1	1
FetchAccountTransactionFlowImpl		0%		0%	5	5	10	10	3	3	1	1
ModifyAccountTypeFlowImpl		0%		0%	4	4	12	12	3	3	1	1
FetchAccountTypeFlowImpl		36%		n/a	3	5	3	7	3	5	0	1
CreateAccountTypeFlowImpl		61%		50%	1	3	2	7	0	2	0	1
Modulo		100%		100%	0	3	0	4	0	2	0	1
Total	163 of 199	18%	9 of 12	25%	16	23	43	56	11	17	3	6

Figure 6-2: Example of the HTML report

Due to time constraints and incomplete unit tests, the code coverage for this application is not available yet.