ExBanking Test Plan

Assumptions and decisions made to solve the challenge

- ExBanking uses a REST API (a mock is used for testing purposes as required per the Assignment Document).
- For the sake of simplicity, it is assumed that ExBanking only offers one financial product: a savings account. And with the registration such an account is opened with a deposit of $15 (hence initial balance). (Please refer to the Assignment description where there is no mention of a separate “Create Account” operation)
- No security, role or authorization tests are needed.
- No preferences of testing frameworks, development processes were indicated.
- The Automated tests implemented are listed below under the **Specification** column.

API Overview

Operation	Endpoint	HTTP verb	Request Body	Response Body

create_user	{{BASE_URL}}/users	POST	{
			"username": "quentin10",
			"firstname": "Quentin",
			"lastname": "Tarantino",
			"CI":"ZOHIBAEGXO1"
			}
				{
				"user_id":"e4cf875f-f66d-4b0b-97c7-ba97808cc155",
				"username": "quentin10",
				"firstname": "Quentin",
				"lastname": "Tarantino",
				"CI":"ZOHIBAEGXO1"
				"status": "active",
				"accountNumber": "888004928"
				"timestamp":"06/07/2023T11:15"
				}

deposit	{{BASE_URL}}/deposits	POST	{
			"account": "888004928",
			"amount": 120.00
			}
				{
				"tx_id":"3158613218",
				"status": "Successful",
				"account": "888004928",
				"amount": 120.00,
				“balance”: 135.00,
				"timestamp":"06/07/2023T11:15"
				}

withdraw	{{BASE_URL}}/withdrawals	POST	{
			"account": "28468326",
			"amount": 17.00
			}
				{
				"tx_id":"3158613233",
				"status": "Successful",
				"account": "28468326",
				"withdraw": 30.00,
				“balance”:90.00,
				"timestamp":"06/07/2023T13:15"
				}

get_balance	{{BASE_URL}}/balance/{{account}}	GET		{
				"account": "888004928",
				“balance”:90.00
				}

send	{{BASE_URL}}/transfer	POST	{
			"account": "888004928",
			"destination": "610838344",
			"sum": 45.00,
			"concept": "Thanks!”
			}
				{
				"tx_id":"3158613785",
				"status": "Successful",
				"account": "888004928",
				"destination": "610838344",
				"sum": 45.00,
				“balance”:45.00,
				"concept": "Thanks!”
				"timestamp":"06/07/2023T13:15"
				}

Functional testing

Code	Test Scenarios	Description	Specification
EB01	New User: Successful creation of a new user	As a developer I should be able to create a user.	**users.tests**
EB02	New User: Cannot create an already existing user	As a developer I should not be able to create a user that already exists.	
EB03	Get all registered users	As a developer I should not be able to get all users.	
EB04	Get a user by CI	As a developer I should be able to get a user.	
EB05	Get an error message when trying to fetch a non existing user	When trying to fetch a non existing user, the API should return a message stating that such user does not exist.	
EB06	Make a Deposit: Successful transaction	As a developer I should be able to make a deposit to a valid sum to an active account.	**deposits.tests**
EB07	Make a Deposit: Transaction not completed	When trying to make a deposit to a non active account, the service should return a message stating that the transaction was not completed.	
EB08	Withdraw money: Success	As a developer I should be able to withdraw money from an account.	**withdraw.tests**
EB09	Withdraw money: Insufficient funds	When attempting to withdraw a sum greater than the account's balance, the service should return a message stating that the transaction was not completed due to lack of funds.	
EB10	Withdraw money: Inactive account	When attempting to withdraw money from an inactive account, the service should return a message stating that the transaction cannot be completed.	
EB11	Get Balance: Success	As a developer I should be able to get the balance of an account	**balance.tests**

EB12	Get Balance of an inactive account	When attempting to get the balance of an inactive account, the service should return a message stating that it is not possible.	
EB13	Send: transfer a valid sum of money	As a developer I should be able to send money to another account	
EB14	Send: prevent from transfering money to an inactive account	When attempting to send money to an inactive account, the service should return a message stating that the transaction cannot be completed.	
EB15	Send: Insufficient balance	When attempting to send a sum greater than the account's balance, the service should return a message stating that the transaction cannot be completed.	

Non-functional testing

Code	**Test Scenarios**	**Description**	**Specification**
EB16	Concurrency	The service must support n concurrent users	**artillery.yml**
EB17	Response	The service must return a response within 500ms	
EB18	Deployment	The service must be containerized to enable CI/CD within the organisation.	
EB19	Privacy	The service must not share users data with third parties.	
EB20	Localization	The service configuration should allow the user to change language from a preconfigured list	