Iurii Teslia

Home assignment Part1

ExBanking test cases

ExBanking test cases

Description / Assumptions:

Since I have been asked to make assumptions on details of implementation methods for ExBanking application, I would like to provide simple descriptions of these methods for testing:

Operation_ name	*Request*	*Response*
create_user	*Mandatory fields* - name, surname,password, is_us_tax_resident *Optional fields* - currency, country, is_2fa	user_id
deposit	*Mandatory fields* - amount, currency *Optional fields* - description, source_transaction	{user_id: [new_balance]}
withdraw	*Mandatory fields* - amount, currency *Optional fields* - description, source_transaction	{user_id: [new_balance]}
get_balance	Param - user_id	
send	*Mandatory fields* - amount, currency, from_user_id, to_user_id *Optional fields* - description	{from_user_id: [new_balance], to_user_id: [new_balance]}

Idea of providing test cases is to show the way of thinking and understanding of the system under test. Hence I would like to provide more test case ideas/checklists rather than a full test case description with Steps to reproduce, Expected results, ID, Date of design, test data, Preconditions, etc.
Please let me know if you need a full description of each / particular test cases.

Test case ideas for ExBanking system:

Functional test cases All test cases should check: - status code - error message in case of negative test - check the correctness of saving data (better to fetch data via API than directly in DB if DB structure is complicated) Optional could be added check for response time for every request.	Create new user with valid data
	Create new user with invalid data,e.g. Without mandatory fields
	Create new user with existing data
	Create new user with invalid data in the fields, e.g. not supported currencies
	Check DB after the creation of new user, e.g. other column are filled correctly (like triggering of the verification process, saving the password in hash format or in the Vault, etc)
	Deposit new amount for user in user’s currency
	Deposit new amount not for user in user’s currency
	Deposit new amount for user with negative amount
	Deposit new amount for user with not supported source_transaction
	Withdraw amount for user in user’s currency. Check user’s balance (here and furthemore)
	Withdraw amount not for user in user’s currency
	Withdraw amount more than on user’s balance
	Withdraw amount for user with negative amount
	Withdraw new amount for user with not supported source_transaction
	Get user balance right after all steps of registration completed
	Get user balance after deposit / withdraw / send while success operations
	Get user balance after deposit / withdraw / send while insuccess operations
	Send available amount of money from user_1 to user_2 and check their balances
	Send exceed amount of money from user_1 to user_2
	Send money from user_1 to user_2 while they have different currencies and check convertion

Non-Functional test cases	Performance testing for big amount of users / services using API simultaneously,like increase the number of transactions and / or data volume
	Time for recovery in case application is crashed, like timing for k8s restart containers, DB has deadlock etc.
	Data consistency testing, while app has a couple of kafka topics for example or in case of failure
	Localization testing in case of app support only some number of currencies/locales