Documentation process for Fundamentals of Software testing project

- specify core elements of system

create demo connections of system

create simple inerataction

first test – offline verification techinques – luhn check

this is does since these elements can be verified without integration tests

(overall aporach unit tests → integration tests → system tests)

Assumption – date is inputted as follows – mm/yy

(write an assumption for all of the inputs that are received mocked and also set up)

assumption – there no limitations for address, name and ccv

assumption – the possible states of a transaction consist of

* capture
* void
* invalid
* refund?
* authorise

Valid Operations Step:

1. Verify Luhn
2. verification checks for other details
3. Authorisation request to bank
4. Record transaction
5. Capture

modified verifyLuhm from boolean to int

Payment process is the next one to be developed – the first tests for these are the expections as they are the simpler option to handle

Testing approach – start from edge cases then go into generic tests – this ensures better code coverage and that the system is tested fully

it is assumed that that the system will have some form of system to distinguish between a refund, capture and void operation. For my approach I am used an input argument.

Also the offline verification and bank authentication will always take place before each operaion

- working on transaction database as this is now required to store data and retreive information

Please note that the trasaction number received from the mock bank system is not the same as the transaction ID of the

when a transaction fails to carry out – no values are recorded in the transaction database

mention the assumed process of operation for the system

ie – offline verification and

it is assumed that the authentication step is not a transaction – thus it will not be included in the transaction database

it is assumed that for a refund to be called – a capture DOES NOT called needs to be carried beforehand as both functions are indepenent tasks

a transaction was only recorded when a successful result of obtained

for authorisation → it is assumed that a positive number is >0

since we do not have a valid transaction ID at the offline verification step – these checks cannot be included in the transaction database

Since this was not specified – it is assumed that whenever an error occurs, the assigned record is changed to a state of invalid

Items tested for Luhn Operation

* valid card number – american express
* valid card number – mastercard
* valid card number - visa
* blank card number
* invalid card number – all digits
* invalid card number – contains letters
* invalid card number – contains special characters
* invalid card number – does not contain digits

These unit tests prove that the Luhn operation is specifically designed for a Digit only Card type and that it does not consider any other character to be valid.

Verify Offline tests

* Checks for a valid Prefix number and Card Type for American Express
* Checks for a valid Prefix number and Card Type for Mastercard
* Checks for a valid Prefix number and Card Type for VISA
* Valid Prefix number and Invalid Card Type – American Express
* Invalid Prefix number and Valid Card Type – American Express
* Invalid card length number and valid card type
* Valid Prefix number and Invalid Card Type – Mastercard
* Invalid Prefix number and Valid Card Type – Mastercard
* Invalid card length number and valid card type
* Valid Prefix number and Invalid Card Type – VISA
* Invalid Prefix number and Valid Card Type – VISA
* Invalid card length number and valid card type
* In general – Invalid card type
* Blank Card Number
* Valid Date
* Expired Date
* Invalid date – special characters
* Invalid date - letters
* Valid Address
* Missing Address
* Valid Name
* Missing Name
* Valid CVV – American Express
* Valid CVV – Mastercard
* Valid CVV - Visa
* Missing CVV
* Invalid CVV for American Express
* Invalid CVV for Master card
* Invalid CVV for VISA
* Invalid CVV – contains Letters
* Invalid CVV – contains special characters
* Invalid CVV – No digits

Offline verification Tests

* Valid offline verification for American Express card
* Valid offline verification for Mastercard
* Valid offline verification for VISA
* Invalid offline verification for an Invalid card type
* Invallid offline verification for a valid card Type and valid Card number but invalid combination i.e. - cardType – American Express and card number for a visa card – 4111111111111111
* Expired card
* Missing Name
* Missing Address
* Missing CVV

this does not need further tests as the core elements of the system in use have already been verified and repeating similar tests would not yield any additional vital information for this scenario.

Authorisation Tests

* Valid authorisation transaction
* Invalid credit card details
* Customer does not have enough funds
* Unknown error
* Bank returns valid request even though card is expired

The last test is added to prove that the system always carries out the Offline verification aspect before each action to verify the card that is being used

same reason – offline verification is not tests as this was already covered in prior tests

Capture Tests

* Valid capture
* Transaction does not exist
* Already captured
* Is void
* Unknown error
* Bank returns valid request even though card is expired
* bank allows invalid transaction to capture – no record in transaction database
* Bank allows invalid transaction to capture – already recorded in DB to be captured
* Bank allows – but transaction is voided

Refund Tests

* Valid refund
* Transaction does not exist
* Transaction exists but is not captured
* Transaction exists but was already refunded
* Refund is greater than amount
* Unknown error occurred
* Bank returns valid but card is expired
* Bank returns valid but transaction does not exist – no record in database
* Bank returns valid but transaction not captured
* Bank returns valid but transaction already refunded
* Bank returns valid but transaction greater than value recorded in DB

The system is designed in such a way as to compare the results both from the Transaction Database and the Bank System results. If either one of these do not agree with one another the resultant recorded is set to an Invalid State. Only when both match does the procedure continue on.

Payment Process tests

* Multiple valid tests – different Accounts
* Multiple error transactions – different Accounts
* Error and valid transactions – different Accounts
* Multiple valid tests – same Account – Different transactions
* Multiple valid tests – same Account – Same Transactions
* Error and valid transactions – Same Account

Refactored transaction Database to record all actions – ie for each step a new record is kept

this way a history is kept of all of the transactions that have occurred

Mention the testing axioms -

testing cannot show that bugs do not exist

Exhaustive testing is impossible

Software Testing is a Risk-Based Exercise. Testing is done differently in different contexts, i.e. safety-critical software is tested differently from an e-commerce site

for this scenario tests are made to prove the functional behaviour of each unit of this system

Beware of the pesticide paradox

“If you always execute the same tests against a system, eventually those tests will cease to find bugs.”