**Test Plan**

## CSCI 3060 Winter 2024 Phase I

Public Github Repository: <https://github.com/Linderwolf/SQA>

**Team Members**

|  |  |  |
| --- | --- | --- |
| Name | Email | Github Account |
| Peter Wolfgang Linder | Peter.linder @ontariotechu.net | Linderwolf |
| Russell Ngo | Russell.ngo @ontariotechu.net | RusselllN |
| Ajaane Kanagasabai | Ajaane.kanagasabai @ontariotechu.net | ajaane |
| Matthew Witvoet | Matthew.witvoet @ontariotechu.net | mwitiwm |
| Stacey Koornneef (TA) |  |  |

**Scope**

*(Test Scenarios/Test objectives that will be validated)*

The initial scope for phase 1 is limited to the testing plan for a console-based front-end which includes the following features:

Account Types

Admin (privileged permissions)

full-standard (buy and sell)

buy-standard (buy only)

sell-standard (sell only)

Transactions

*Login:* Required to start a front-end session. Only by entering a valid username in the system may a session start and the user proceed. All transactions the user performs will be logged.

*Logout*: Ends a front-end session. Writes out a list of transactions to a Daily Transactions File.

*Create:* Add a user with the ability to buy and sell games (A privileged transaction)

*Delete*: Remove a user (A privileged transaction)

*Sell*: List a game up for sale

*Buy*: Purchase a game available for sale

*Refund*: Issue a credit to a buyer’s account from a seller’s account (A privileged transaction)

*Add Credit*: Add credit into the system for the purchase of accounts

#### Presently Out of Scope

*(Enhanced clarity on what we are not going to cover)*

To be implemented in future phases as requirements are updated:

Exit / Break out of Transaction without completion

Lacking this is likely to cause inconvenience for users, but it also impedes the automation of test cases. A test case that fails may stall and cause subsequent tests to thus be unable to complete. (e.g. A login without a logout does not write to the Daily Transactions File)

List of Games Transaction

This is stated by the client to be a deliverable for the next phase.

The Backend

Daily Transaction, Current User Accounts, and Available Games Files (See Assumptions)

Other Features

There are features we would highly recommend in the future, but are not customer requirements at this time, such as User Authentication.

**Assumptions**

*(All the conditions that need to hold true for us to be able to proceed successfully)*

The system must follow-through with any transaction. There is no exiting out of a process tree. e.g.  You cannot go to a sell transaction in the midst of a buy transaction; the sell transaction must be completed first.

There are no users interacting with the system maliciously. There is no form of login authentication or requesting of a password when logging into the system.

Logout assumes a successful login functionality

Sell assumes successful buy functionality

Refund assumes successful buy functionality

Daily Transaction, Current User Accounts, and Available Games Files were stated by the client to be backend and outside the scope of this phase. Current User Accounts and Available Games, however are necessary for the functioning of the front-end. We will proceed with sample data to fill these so that the front-end may function with account data, and operate under the assumption that reading and writing to them must be implemented earlier than was stated, until this can be clarified.

**Schedule**

* Phase #1: Front End Requirements

List of Test Cases

Test Cases

Test Plan

To be delivered:  Friday, February 2, 2024

* Phase #2: Front End Rapid Prototype

Sample Log File data/

Login & Logout functionality

Writing to Daily Transaction File

Buy & Sell functionality

Writing to Game Collection File

Writing to Available Games File

Add Credit functionality

Create & Delete (Privileged Transactions)

Writing to Current User Accounts File

Refund (Privileged Transactions)

To be delivered:  Friday, February 16, 2024

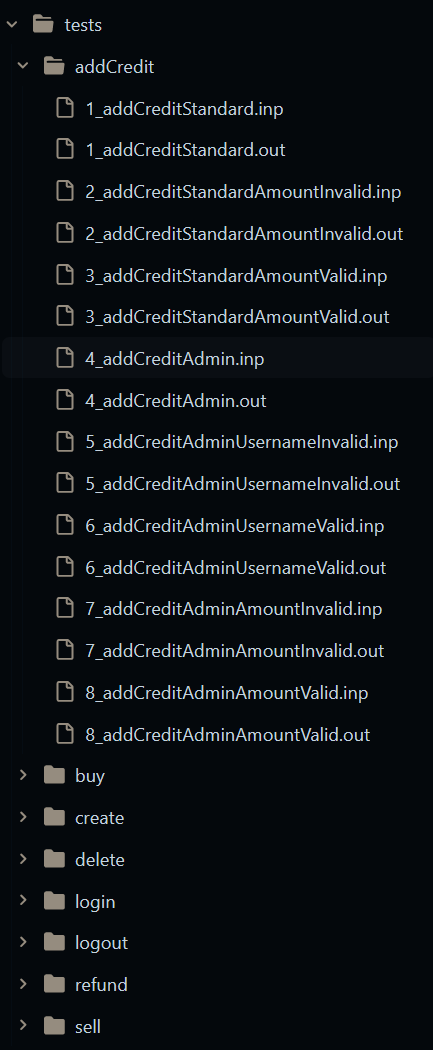
* Phase #3: Front End Requirements Testing

Update Tests to better fit prototype and new requirements

Batch Scripts to Automate Testing

To be delivered:  Friday, March 1, 2024

**Project Directory & Tree Structure**

 Each set of test-cases are placed into a separate folder based on their transaction code/functionality. The tree structure is listed in the image right. Each test case file follows a specific naming convention enabling us to run tests automatically in numerical sequence for expected inputs and outputs. An underscore is included as a delimiting character. The convention is camelCase as follows:

##\_transactionFunctionalityBeingTestedForValid**or**Invalid”.in**or**.out

“Valid” tests are those for which it is expected that the specified inputs will cause the transaction to be completed.

“Invalid” tests are those for which it is expected that the specified inputs will cause the transaction to *not* be completed, provide an error message and re-prompt for input.

**Running Tests**

We’ll be writing a .bat file and shell scripts to automate testing. The bat file will automate the execution of the front end of the Digital Games Distribution system, and scripts shall loop through our test case directory to input the .inp files, then compare the output with what is expected in the .out files.

Successful tests will be those whose output matches the expected. Unsuccessful tests are any tests whose output does not match the expected. Each test will be run sequentially based on the folder structure stated under Project Directory above. A log file will be generated which shall state the test being run, its success or failure, which lines did not match the expected, a final count of tests run, successes, and failures, # of test successes per transaction folder, and # of test failures per transaction folder.

**Environment & Tools**

* Trello will be used to track issues.
* A text editor is the only necessity for scripting.
* Discord is the main communication platform for the team. It will be used to hold meetings, discuss progress, report bugs, and share screens for pair programming.
* The front-end program will be written in C++, with programmer-specific environments.  
  Likely dev environments of Visual Studio, or CMake, clang and either CLion or VSCode.
* If developer environments drastically differ in platform, Docker will be used for containerization for simple collaboration.

**Exit Criteria**

Testing will be completed (temporarily) in Phase 3, when the requirements are satisfied by all test cases, all automated test cases pass, and the customer is satisfied with the product’s front-end experience.

There will be additional unit testing for the back-end throughout the remaining phases, as well as integration testing in phase 6.