Software Quality

Assignment 1

Joseph Heron 100425488
Khalil Fazal 100425046
Carly Marshall 100426654
Ryan Crawford 100425694

Test Cases

Transaction	Test Name	Intention
Login	Login 1	Test that we can log in as admin
	Login 2	Test that we can log in as full standard
	Login 3	Test that we can log in as buy standard
	Login 4	Test that we can log in as sell standard
	Login 5	No user should be allowed to login while a user is still logged in
	Login 6	“END ” should not do anything when anyone is logged in
	Login 7	Username must exist
	Login 8	Enter invalid characters
Logout	Logout 1	Transactions should be disallowed after logout
	Logout 2	Cannot logout before a login has been made
	Logout 3	“END ” should terminate the program if no users are logged in
Create	Create 1	Username must be a maximum of 15 characters
	Create 2	New usernames must be unique
	Create 3	New user types can be either: AA, FS, BS or SS
	Create 4	“END “ cannot be used as a username
	Create 5	Test that only admins can create accounts
	Create 6	Admin user can be created
	Create 7	Full standard user can be

		created
	Create 8	Buy standard user can be created
	Create 9	Sell standard user can be created
	Create 10	User can be created with initial account greater than 0.00
	Create 11	User cannot be created with initial account less than 0.00
	Create 12	Use must enter in a integer for the starting balance of a new user
	Create 13	Username entered is valid characters
Delete	Delete 1	Test that a user can be deleted
	Delete 2	Cannot delete the current logged in user
	Delete 3	Only allow transaction if an admin is logged in
	Delete 4	User cannot be deleted if the username does not exist
	Delete 5	Invalidate transactions of deleted users
	Delete 6	Username must be valid characters
Sell	Sell 1	Event name must be comprised of only the uppercase and lowercase alphabet
	Sell 2	Event name must be left justified and filled with spaces
	Sell 3	Account must have sell privilege
	Sell 4	Event names must be unique
	Sell 5	Different users can sell tickets to the same event

	Sell 6	Prices cannot be over $999.99 for each ticket
	Sell 7	Event title cannot be more than 25 characters
	Sell 8	Maximum number of tickets on sale is 100 per event
	Sell 9	If tickets have just been put up for sale, they can't be bought until the next session
	Sell 10	Create an event for tickets to be sold at as sell standard
	Sell 11	Create an event for tickets to be sold at as full standard
	Sell 12	Create an event for tickets to be sold at as admin
	Sell 13	Number of tickets sold must be greater than 0
	Sell 14	Price of a ticket must be greater or equal to 0
	Sell 15	Price of ticket must be a valid integer
	Sell 16	Number of tickets must be a valid integer
	Sell 17	Event name must be valid characters
Buy	Buy 1	Test that a purchase can be made with buy_standard
	Buy 2	Test that a purchase can be made with full_standard
	Buy 3	Test that a purchase can be made with admin
	Buy 4	Account must have buy privilege
	Buy 5	Buyer must have sufficient funds to make the purchase

	Buy 6	Reject any requests for purchases of more than 4 tickets
	Buy 7	User cannot buy more tickets than are left for sale
	Buy 8	Test to see if users with the admin permission can buy more than 4 tickets
	Buy 9	Requested event title must exist
	Buy 10	Seller's username of requested event must exist
	Buy 11	Must enter a valid integer for number of tickets to purchase
	Buy 12	Event name must contain only valid characters
	Buy 13	Sell name must contain only valid characters
	Buy 14	Seller must not exceed max funds as a result of the transaction
	Buy 15	Sold out tickets are removed from available tickets file
	Buy 16	Seller of tickets can 'buy tickets' back
Refund	Refund 1	Only allow transaction if an admin is logged in
	Refund 2	Check that specified amount was transferred from seller to buyer
	Refund 3	Sell must have sufficient funds to refund buyer
	Refund 4	Buyer must exist
	Refund 5	Seller must exist
	Refund 6	Refund cannot put credit balance over maximum amount

	Refund 7	Refunded money is not available until next transaction
	Refund 8	Funds transferred cannot be less than 1
	Refund 9	Funds transferred must be a valid integer
	Refund 10	Sell username must be valid characters
	Refund 11	Buy username must be valid characters
Addcredit	Addcredit 1	Add an acceptable amount of money to a user
	Addcredit 2	Add acceptable amount as an admin to self
	Addcredit 3	Add acceptable amount as an admin to another user
	Addcredit 4	Only allow a maximum of $1000.00 added to an account per session
	Addcredit 5	Username must exist
	Addcredit 6	Reject any requests that put users credits over $999,999
	Addcredit 7	Reject any request for negative credit
	Addcredit 8	Credit added must be a valid integer
	Addcredit 9	Username must contain only valid characters
	Addcredit 10	Sell standard cannot add credit to their account
General	General 1	Test to see that it doesn't crash during minimum transactions
	General 2	Be able to handle invalid input (al test cases)

Requirements Problems (Assumptions)

As we were creating our test cases, we encountered some problems not specified in the requirements for which we assumed the solutions. Firstly, we assumed that all monetary values would be represented in the form $xx.xx and would be in US dollars. We assumed that when a new account is created, a starting balance can be specified for that account and the credit balance can never go below $00.00. We also assumed that when a user logs in, they will not need to specify their account type, we will read it from the global user accounts file given the username.

When a user sells tickets for a new event, they must sell at least 1 ticket for that event. Also, a user cannot buy more than the number of tickets available for sale. For example, if there are 2 tickets and the user requests 3, the system will give them 0 tickets and return an error. During a refund transaction, we required a minimum refund amount of $1.00. After a refund, the tickets refunded are not added to the number of tickets available for the event. Finally, all input files and daily transaction files for test cases will end with the keyword 'END_' in order to keep each test isolated.

If a seller wants to remove tickets they have for sale, they can do so by buying back their own tickets. If a seller is buying their own tickets, they may buy as many as are available and at no cost. Once the event is sold out, it is removed from the available tickets file. Sellers cannot add credit to their own accounts and a buy transaction should not be processed if it puts the seller over the maximum credit balance.

Format of Deliverables

Our project will be laid out such that there will be a folder containing folders for each test category. Within each folder, there will be another folder for each different test case. For example, in the 'tests' folder there will be multiple folders named after sets of tests that are contained within the folder; the first test folder will be called 'login'. The name will be based on the test name. Each of these folders will then contain sub-folders for each of the tests related to the topic, for example: 'login1'. Within each of these folders, there will be relevant files such as the input file 'login1.inp', the transaction file output 'login1.etf', and the expected output file 'login1.bto'. Also contained within each of the different tests folders, will be the test specific output files for the daily transactions file, the user accounts file, and the available ticket file. Within the root folder there will be another folder named 'global'; this folder will contain the input used globally for user accounts file and available tickets file.

Example ./FirstVu/tests/login/login1/login.inp
or
Example ./FirstVu/tests/addcredit/addcredit2/addcredit2.bto

Testing Plan

Our plan is to test systematically through each sub-test-folder and match each input with

output files, and also use the global input files. The test results will be displayed to the users through the terminal; they will also be recorded to an output file called ‘result.log’ which will be found in a subfolder called ‘log’ in the ‘tests’ folder. The tests will be executed such that all other commands that are a part of the test (but not the command being tested) have been tested prior to the test to be working. This will cause our tests to take a progression through isolated tests of specific elements that are verified and then used in later tests.

In order to ensure that the necessary dependencies are tested prior to a test case the tests will be executed in the following order:

- Login 1 - Login 4,
- Logout 1 - Logout 3,
- Create 7 - Create 9,
- Delete 1,
- Delete 3,
- Sell 5,
- Sell 10 - Sell 12,
- Buy 1 - Buy 3,
- Refund 3,
- Addcredit 1 - Addcredit 2

The rest of the test cases are able to be completed in order defined in the above table.