|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Name | Test Intention | What went wrong | The error | How error was fixed |
| **addcredit 03** | Test to confirm if the username that the admin wants to add credit to actually exists within the system | Existing user check was not implemented |  | Implemented read file method to look for user (findUser(token)) |
| **addcredit 04** | Test to confirm whether the amount of credit added is at most equal to $1000.00 in a given session. | Credit amount more than $1000.00 was accepted | Entering string crashed the program | Create a method called “constraintAmount()” which takes in the amount, maximum and minimum and returns true if constraint met else false. This decides if the transaction should proceed further. |
| **addcredit 05** | Test to confirm if the amount of credit to be added onto an account is greater than $0.00 | 0$ and negative credit amounts were accepted. | Entering string crashed the program |  |
| **Refund 01** | Test to confirm money being refunded is more than 0 credit amount. | 0$, negative, and credit amounts over 1000000 were accepted. | Entering string crashed program | Used constraintAmount() function to prevent credit entries below 1 and over 1000000.  Added condition to allow only digit entries. |
| **Refund 03** | Check if buyer is current user/ exists | Refunded money to a non-existing buyer |  | Implemented read file method to look for user (findUser(token)) |
| **Refund 04** | Check if seller is current user/ exists | Deducted money to from a non-existing seller |  | Implemented read file method to look for user (findUser(token)) |
| **create 02** | Test to confirm whether username of newly created user is unique. | Created duplicate users |  | Used (findUser()) function to check for already existing user. |
| **create 03** | Test to check whether the character count of the newly created user is within 15 characters. | Accepted usernames exceeding 15 character count |  | Created a method called “constraintStrLength()” which takes in the name, maximum and minimum and returns true if constraint met else false. This decides if the transaction should proceed further. |
| **create 04** | Test to check if the character count of a newly created user is greater than zero. | Accepted empty username fields |  | Used “constraintStrLength()” method to check if name length is greater than 0 characters |
| **create 05** | Test to check if a newly created user does have a User Type which allows certain privileges like buying or selling. | Any string was accepted. |  | Included a condition which checks for user type in the 4 types allowed. |
| **Delete 02** | Test to check if any outstanding tickets for purchase or sale are cancelled before deletion of the account. | Did not cancel outstanding ticket sales, tickets remained in event’s file |  | Used function findUser() in events file to make sure the individual’s sale offers are deleted! |
| **Delete 03** | Test to confirm that the username of the account to be deleted must be the name of an existing user | Attempted deletion of non-existing users. |  | Used function findUser() to search for an existing user. |
| **Delete 06** | Test to confirm if the user that is deleting an account does not have the same username (basically no self-deletion). | Allowed admins to delete themselves |  | Included a condition checker, which prevents admin from trying to delete himself! |
| **Buy 02** | Test if maximum of 4 tickets are purchased and no more! | Accepted negative, 0 and more than 4 tickets. | Used ‘or’ operator instead of ‘and’ when checking both max and min(1) | Used constraintAmount() function to meet maximum requirement of 4 tickets. |
| **Buy 04** | Test if confirmation page is displayed in the form of Yes or No | The confirmation feature was not implemented |  | Added feature to ask for confirmation from buyer before proceeding and any other input would close transaction with error message |
| **Buy 06** | Check if seller and event exists in event’s file. | Did not check the existence of event or seller before purchasing! |  | Used findUser() to check for existing offer in event’s file |
| **Sell 02** | Test to verify the maximum length of an event title does not exceed 25 characters | Accepted empty string and strings with more than 25 characters |  | Used constraintStrLength() to check for restrictions. |
| **Sell 04** | Test to verify the maximum price for a ticket sale does not exceed 999.99$ | Accepted values more than 999.99$ | Entries other than digits crashed program | Used constraintAmount() to check for restrictions.  Added condition to allow only digit entries. |
| **Sell 06** | Test to verify the maximum number of tickets for sale does not exceed 100 tickets | Accepted negative, 0 and more than 100 tickets as input | String entries crashed program | Used constraintAmount() to check for restrictions.  Added condition to allow only digit entries. |
| **Sell 08** | Ascertain no further transactions are accepted on a new ticket for sale until the next session. | Kept accepting more transactions. |  | Implemented way to make the session end automatically after a sell transaction and write to Daily Transaction file. |
| **Login 00** | Test to confirm a transaction other than login is not accepted before a login | Login transaction was not implemented resulting in username first being entered |  | Implemented loop that would only accept “login” or “exit” command else would ask to re-enter command. |
| **Login 01** | Test to see if login functions properly | Running the software begins with an unexpected entry of the user name when it should not. | When entering input with spaces, it registers them as 2 different inputs. So we added getline(cin)  This resulted in more errors. | We used cin.clear() and cin.sync() to fix these double enteries before each getline(cin) statements. |
| **Login 02** | Test to see if constraints are met (15+ character & empty usernames) | Unnecessary access to user data file for obvious constraint restrictions |  | Added additionally conditions to check for username basic constraints before accessing user file for efficiency. |