|  |  |
| --- | --- |
| **Test Conducted: (v0.1)** | **Test Results:** |
| Tested that attributes would be assigned to an item when it is created | They were |
| Tested that new item would be added to the item.items list | It was |
| Tested that the class issue function would correctly issue the item | It was |
| Tested that issue\_item function would issue an item with expected inputs | It did |
| Tested the boundary input/invaid inputs in the issue\_item function and that they would not issue the item | For the boundary limits of the it worked – |
| Tested the invalid input in the issue\_item function so that it would not execute if not a valid length of days | It didn’t work so I added a return statement because the function wasn’t ending if it couldn’t turn issue\_days into an integer test6.png |
| Tested the invalid input in the issue\_item function so that it would not execute if not a valid length of days | It worked and stopped the function from continuing test7.png |
| Tested item.days\_til\_due function to make sure that it was returning the correct number of days | It was – test8.png |
| Tested that item.get\_info function on both issued and returned items to make sure it returned the correct information | It did test10.png – test 11.png |
| Tested that the item.remove\_item will remove an item from the item.items and item.issued\_items lists and will cause it to be garbagecollected – I did this by addingthe \_\_del\_\_ function to the item | It did (Although because we referenced the item again it wasn’t garbage collected until after the second print(item.items) statement  Test12.png + test13.png |
| Tested that returning an item would change its info and move it from the issued items list | It did |

|  |  |
| --- | --- |
| **Test Conducted: (v0.2)** | **Test Results:** |
| Tested that the grid layout of the main GUI window would be even and split into two primary section | It was |
| Tested that the “issue item” button would create the issuing window | It did – test16.png |
| Tested that the item management button would create the item management window | It did test17.png |
| Tested that the “add item” button would create the “add item” menu | It did – test18.png |
| Tested that the “add item” button in the “add item” menu would run the add\_item function | It did – test 19.png + test20.png |
| Tested that the item management window is arranged with the two bottom buttons outside the “Item Catalog” frame and both evenly sized | They weren’t initially – so I fixed this |
| Tested that the item management window is arranged with the two bottom buttons outside the “Item Catalog” frame and both evenly sized | They are now – test21.png |

|  |  |
| --- | --- |
| **Test Conducted: (v0.3)** | **Test Results:** |
| Tested that clicking on an item in either the returned items or items on loan section would show me the item details | It did – test22.png |
| Tested that issuing an item would switch it to the “items on loan” listbox and update the details | It did – test23.png + test24.png |
| Tested that pressing the “return item” button would return it | It did – test25.png + test26.png |
| Tested the invalid and boundary inputs on the issue\_item function in the gui | They worked – the function would not complete.. **However the window would still close even if they were invalid – so I fixed this** -- test27.png + test28.png |
| Tested that the window would no longer close even if the issue inputs were invalid | It no longer did – test29.png |
| Tested that pressing “view details” in the “item management” section would show the item details | It didn’t because I forgot to change a section of code in one place when I changed it in another **so I fixed this**– test31.png |
| Tested that pressing “view details” in the “item management” section would show the item details | It worked – test30.png (these images are out of order because I used ‘ctrl+z’ to undo the changes I made to fix it in order to test the first one |
| Tested that adding an through the “add item” menu worked | It did – test32.png + test33.png |
| Tested the boundary inputs for the item name when adding an item | They prevented the item from being added – test34.png + test35.png |
| Tested the boundary inputs for the item description when adding an item | They prevented the item from being added – test36.png + test37.png |
| Tested that the “remove item” button worked properly and removed the items from the GUI as well as lead them to be garbage collected | They did – test38.png + test39.png |
| Tested the boundary input for the amount of days to make sure that it was above 0. There is no limit to this because this Bean’s Phat Library that will operate for all eternity cause it will. | It worked and stopped the function from issuing the item  Test40.png |
| Tested that the dates shown in the date\_due is correct | It is – test41.png |

|  |  |
| --- | --- |
| **Test Conducted: (v0.4)** | **Test Results:** |
| Tested that all the places where there were print statements are now replaced with the error\_notif window | They are e.g. test42.png |
| Tested that error sound would be played asynchronously and not delay any functions | It didn’t |
| Tested that items would correctly save/append to the file. | They do – test43.png, test44.png, test45.png |
| Tested that items would correctly load from the file | --- this is a new window created after the last -- test46.png |
| Tested that items would be correctly removed from the file | They are test47.png, + test48.png + test49.png + test50.png |
|  |  |
| Tested that all the windows would now load in the correct places | They did – test52.png, test53.png, test54.png, test55.png |

|  |  |
| --- | --- |
| **Test Conducted: (v0.5)** | **Test Results:** |
| Re-tested all previous tests with ‘item\_controller’ class | All outputs were the same |
| Tested all previous tests with modified ‘item’ class | All outputs were the same |
| Tested all previous GUI tests with ‘main\_window’ class | All outputs were the same |
| Tested that users couldn’t issue items for negative numbers of days | They could – so I fixed it |
| Tested that users couldn’t issue items for negative numbers of days again | They couldn’t anymore as it now raises a ValueError |
| Tested fix for minor error when exiting program after closing “item details” window | It no longer gave the error (it didn’t affect anything anyway – but was annoying during testing) |
| **Resub Part 2** | **Test Results: (Retesting every conditional expression)** |
| Tested that ‘winsound’ would make a sound on windows | It did |
| Tested that the program would still run on other OSes (because of the try/except statement that removes winsound | Tested on linux and it did. Second variation of function did not run (this function was just because I thought it was cool / OS-themed, and didn’t know the error commands for LINUX/OSX) |
| Tested that class mainwindow was created by printing | It was |
| Tested that item controller attribute was set to MainWindow class by printing | It was |
| Tested that MainWindow UI was created upon object creation | It was |
| Tested that all labels and buttons on MainWindow had the right text | They did (Bean’s Library System, Administrator, Returned Items, Item’s on loan, etc) |
| Tested that clicking on an item in the MainWindow listbox showed the item details in a separate window | It did – if window was not open it was created |
| Tested that the item details window would be deleted if MainWindow was destroyed | It was (other ‘daughter’ windows would have to be closed manually, because they are not expected to be open when the main window is closed) |
| Tested that pressing the issue item button would open the print to issue an item | It did |
| Tested that if no item was selected it would select the last item in the list to issue | It did |
| Checked that issuing an item would issue it | It did |
| Tested that you could not issue a number for negative days and a ValueError would be raised if it was. | You couldn’t. And it was. |
| Tested that you could not issue an item for >999 days, and a ValueError would be raised if above | You couldn’t. And it was. |
| Tested that you could not enter a name <4 chars | You couldn’t |
| Tested that you couldn’t enter a name >25 chars | You couldn’t |
| Tested that items would be added to the management GUI from the database when the MainWindow is created | They were |
| Tested that the management GUI would update when you add a new item and on window creation | It was |
| Tested that the ‘item management’ window would open when you press the button | It would |
| Tested that the ‘item management’ window had the right text | It did |
| Tested that you could view item details from the item management window | You could |
| Tested that you could remove an item | You could |
| Tested that you could add an item | You could |
| Tested that the item management and main window would update when you added or removed an item | It did |
| Tested that the add item menu had the right text | It did |
| Tested that the ‘view item details’ menu would not reopen if it was already open | It didn’t – the content just updated |
| Tested that the ‘view item details’ window had the right text | It did |
| Tested that the listbox event would not fire when clicking on another listbox (two main GUI window listboxes causing duplicate details) | It did not |
| Tested that the right item into would be selected from the list of items when you select an item to view detailis | It was |
| Tested that the main window could be destroyed | It couldn’t – I accidentally deleted a line |
| Tested that the main window could be destroyed | It could after fixing ^^ |
| Tested that the error notification window would pop up if you did something invalid/out of bounds | It did |
| Tested that the error notification window would make a sound if on Windows 10 | It did |
| Tested that you could close the error notification window | You could |
| Tested that the issued and non-issued items would update correctly when an item is issued / on MainWindow initialisation | They were |
| Tested that a UID would be added to an item if it did not have one upon initialisation | It was, but not changed if it already had one |
| Tested that title,description,issue status,issuer name,issue date, due date were set to an item on creation | They were |
| Tested that two objects could not have the same ID for <100000 items | They couldn’t (unless > 100000 items, but that is beyond the scope of this simple code). ID generation isn’t very efficient here but that’s because this is designed for <100 items so it’s beyond the scope of the program. |
| Tested that an item could be issued with issue\_item | It could be |
| Tested that an item could be returned with return\_items | It could be |
| Tested that an item could be removed with remove\_item | It could be |
| Tested that an item would be added to the list ItemController.issuedtems with load\_item | It could be |
| Tested that a new item would be created upon opening if it was not issued and would then be added to the ItemController.Items list | It was |
| Tested that using get\_info() would return the info (as a string) of the item in a formatted arrangement | It did |
| Tested that the returned info string would not show ‘issue’ information if it was not issued, and would show ‘issue’ information if issued | It did |
| Tested that all unissued items would be added to the ‘ItemController.items’ list | They were (On creation of MainWindow, and on creation of item) |
| Tested that all issued item wold be added to the ‘ItemController.issued\_items’ list | Same as above |
| Tested that all items names would be added to the ‘ItemController.item\_names’ list | Same as above (to prevent items with the same names) – not the most memory efficient way to do this but it’s for a small-scale program so is irrelevant. |
| Tested that the ItemController class could be created | It could |
| Tested that a list of item UID’s could be fetched from the SQL class | They could and it saved to a list |
| Tested that when items were initialised it would recognise if it was issued or not (and then defer to a previous test about how they are handled) | It did |
| Tested that all of the current data in the SQL database would be loaded upon program start (in format of list of item information) | It was |
| Tested that if a invalid name was edited for issuing it would actually direct the issue to the error\_notif function (conditions are for all previous tests regarding issuing) | It was |
| Tested that the library db was saved to ‘LibraryDB.db’ | It was |
| Tested that a connection with a SQLite database could be established, and that it would create one if one was not present | It did |
| Tested that the default tables would be made when the database is created | It was |
| Tested that an error would be printed if there was an error connecting/creating a databse (e.g. no database and read only) | It was printed |
| Tested that the db connection would be closed after each query was completed (so the DB could be connected to while the program is open, and that the program could run multiple instances) | It was |
| Tested that user data could be inserted into the data table | It was |
| Tested that rows could be deleted from the database based on a UID reference | They could |
| Tested that user data could be updated to change the issue status in the database | It could be. |
| Tested that all user data could be fetched from the database using Sql.fetch() | It could be |
| Tested that you could fetch a list of all ‘unqiue ID’s’ in the database using Sql.get\_uids) | It could be |
| Tested that an error would be printed if sql.py was attempted to be run in namespace ‘\_\_main\_\_’ | It was |
| Tested that my programs followed standard python conventions by code reviewing and using the tool pyLint to validate the conventions of the program | It failed because of class ‘item’, so I changed it to ‘Item’ |
| Tested that my programs followed standard python conventions by code reviewing and using the tool pyLint to validate the conventions of the program | It did |

I used pyLint to test the conventions of my program and it only found ‘errors’ due to mixed space/tab indenting – which I then fixed.

What I did during Resub:

* Support cross-platform execution
* Fixed error with window closing
* Moved to proper ‘object-oriented’ program structure
* Redid all previous tests after that
* Fixed boundary tests
* Optimised memory performance (a bit, still not great)
* Created custom SQLite object package
* Added SQLite data saving rather than to a text file