Test Plan Document

for

LIBRARY INFORMATION SYSTEM

Authors:

Pranav Nyati (20CS30037)
Shreyas Jena (20CS30049)
Utsav Vinay Mehta (20CS10069)

Instructors:

Prof. Sourangshu Bhattacharya
Prof. Abir Das
Owais Iqbal

Indian Institute Of Technology Kharagpur

23rd March 2022

TEST PLAN IDENTIFIER :

LIS_UNIT_TEST1

REFERENCES :

- Software Requirement Specification
- Use Case Diagram
- Class Diagram

INTRODUCTION:

This is the Master Test Plan for the Library Information System version1.0. This plan will address exhaustive and robust testing of all the items, elements, and features that are associated with the LIS, directly or indirectly. The project deploys unit testing, and the details of each test are addressed in the appropriate section.

TEST ITEMS (FUNCTIONS):

This part includes testing of all such classes and their methods, objects and other utility functions which control the internal working of the software application (implemented using an Object-Oriented Programming paradigm) which are not visible to the various users of the software as they are not a part of the user-interface. Thus, this includes testing of constructors of various classes, methods of the respective classes, functions outside the classes, other utility and friend functions, etc.

The LIS software has the following Test Items:

(A) Functions related to the Classes representing entities such as Library Members, Employees, Books

- Class ***Librarian*** :
    - Construction
    - getEmployeeID()
    - getEmployeeName()

- Class ***Library_Clerk*** :
    - Construction
    - getEmployeeID()
    - getEmployeeName()

- Class ***Lib_Member*** :
    - Construction
    - getMemberID()
    - getMemberName()

- Class ***UG_Member (derived class of Lib_Member)*** :
    - Construction
    - getMemberID()
    - getMemberName()
    - getMaxBooksAllowed()
    - getMaxAvailTime()
    - canIssue()

- The getter functions of ***PG_Member, RS_Member, Faculty_Member*** are exactly similar to that of ***UG_Member***, and hence will be tested in the same manner.

- Class ***Book***:
    - Constructor
    - getBookName()
    - getBookAuthor()
    - getBookTitle()
    - IssueStatus() : stores NULL if currently not issued else stores IssueDate
    - getDueDate()
    - getRackNumber()

(B) Internal functionalities related to the Accounts and Login processes :

- Verifying the user login credentials (user types -> admin or member) and user password

FEATURES TO BE TESTED :

1. **ADMIN-RELATED FEATURES:**
    a. *Library Clerk:*
        i. Add new books
        ii. Approve issue of books
        iii. Approve return of books
        iv. Delete books (suggested by Librarian)
        v. Issue penalty slip for overdue books

    b. *Librarian:*
        i. Add new members
        ii. Delete existing members
        iii. Check book issue statistics

iv. Send reminders for overdue books
v. Send notification for reserved books

2. **MEMBER-RELATED FEATURES**

    a. Issuing books
    b. Returning books
    c. Reserving books

3. **Searching for books (search by Name / ISBN / Author)**
   Common feature of the Librarian, Clerks and the Members

4. **Sign up/Login/Logout related Functionalities**

FEATURES NOT TO BE TESTED:

The Graphic User Interface of the Library Information System will not be tested manually.

ITEM PASS/FAIL CRITERIA:

- We will provide Golden outputs for the appropriate tests for each function to be tested, as well as appropriate Exception classes for the exceptions.
- Whenever an expected parameter is not passed to a functionality by the user, a TypeError is raised.
- An exact Match with golden output / Exception class will be considered to PASS the test case, otherwise, it would be a FAIL.
- Efficacy would be judged by the % of tests passed.

SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS:

Stop further testing when some required package compatibility fails.
The testing will also be suspended in case the percentage of tests failed up to the current number of tests executed crosses a certain threshold.

TEST DELIVERABLES:

1. *Test Plan Document*
2. *Test Suite Document*

UNIT TESTS:

MemberLogin()

- **General Input**
  - Member ID
  - Password

- **General Output**
  - Retrieved member object from the database.

- **Scenarios**
  - Member logs in successfully.
  - Member ID doesn't match with any entry in the database.
  - The Member ID matches in the dataset but the password does not match with Member ID.
  - User tries to enter Staff ID in Member Login

StaffLogin()

- **General Input**
  - Staff ID
  - Password

- **General Output**
  - Retrieved staff object from the database.

- **Scenarios**

  - Staff Logins in successfully
  - Staff ID not in the database
  - Password does not match with Staff ID
  - User tries to enter Member ID in Staff Login

Test SearchBook()

- **General Input**
  - Search String (Can be title, author, ISBN)

- **General Output**

  - List of ISBN and names of matching books
  - Message if no books match the search

- **Scenarios**

  - No book in the system matches with the search string
  - Some subset of books in the system matches with the search string

Library Member

- **Test Constructor**
- **Test Getter Functions**
    - **General Input**
        - LibraryMember credentials
    - **General Output**
        - Output is function-dependent.
    - **Scenarios**
        - Getting the Name of the Member
        - Getting the Number of Books Issued by the Member
        - Getting list of currently issued books by the Member
        - Getting the information of currently reserved book by the Member
- **Test CheckAvailabilityOfBook()**
    - **General Input**
        - Book object previously retrieved from the database
    - **General Output**
        - Outputs the current availability status of the book based on whether it is issued currently to someone or not issued.
    - **Scenarios**
        - The book is not currently issued by any member and is available for issuing

        - The book is currently issued by another member, but can be reserved by the member for issuing it once it is returned
        - The book has been returned by the previous user but the current day is within the 7-day period of the another member to reserve that book,hence it cannot be issued though available. Hence, the member can only opt to reserve it.

- **Test CanIssue()**

  It is a function of each of the 4 derived classes of the Member class (namely the UG/PG/RS/Faculty), but we have written it here to follow the testing sequence.

    - **General Input**
        - Member object

    - **General Output**
        - Returns whether the Member can issue another book or not depending on her issue limit

    - **Scenarios**
        - The Member has exhausted his limit of book.
        - The Member has not exhausted his limit of book.

- **Test IssueBook()**

    - **General Input**
        - LibraryMember object.
        - Book Object for the Book to be Issued

    - **General Output**
        - Issue Request is sent to Library clerks for approval of issue.

        - The database will be updated only after the Clerk approves the issue request through his account

    - **Scenarios**
        - User tries to issue a book she has already issued.
        - User issues an available book.

- **Test ReserveBook()**

    - **General Input**
        - LibraryMember object
        - Book object for the book to be reserved.

    - **General Output**
        - Database record in Members section is updated with the new reservation included.

    - **Scenarios**

        - The book is unavailable and member has pending / active reservations for some other book -> cannot reserve this book
        - The book is unavailable and member has an active/pending reservation for this book -> redundant case as already reserved by this user
        - The book is unavailable and the user has no prior reservation for this book -> book is reserved for the user

- **Test ReturnBook()**

    - **General Input**
        - A Book object
        - LibraryMember object

- **General Output**
  - Return Request for the book is sent to the Library Clerks for approval of the return.
  - The database will be updated only after the Clerk approves the return through his account

- **Scenarios**
  - The book was returned within the due date, hence no penalty was incurred on the member.
  - Book was overdue and hence a penalty was incurred on the member, with a penalty slip issued to him/her.

- **Test CheckForReminder()**

  - **General Input**
    - LibraryMember object.

  - **General Output**
    - Shows all the notifications and reminders to the member (if any).

  - **Scenarios**

    - The librarian calls the SendReminder function for that member (for reminding of overdue books / issuing a reserved book if member has active reservation), and the Library Member wants to check for such notifications.

Library Clerk

- **Test Constructor**
- **Test AddBook()**
    - **General Input**
        - ISBN, Title, Author
        - Rack number
        - Today's Date
    - **General Output**
        - The book gets added to the database if the same ISBN book doesn’t already exist, and the Books section of database is updated
        - If the same ISBN book exists, its number of copies get increased.
    - **Scenarios**
        - The book with the same ISBN already exists -> create a new copy of the same book and add to the database
        - The book with the same ISBN doesn't exist in the database -> the book is successfully added
- **Test DeleteBook()**
    - **General Input**
        - Book object
    - **General Output**
        - Book object is deleted from the database

- **Test ApproveIssue()**
    - **General Input**
        - Member ID
        - Book object
    - **General Output**
        - The library clerk will approve the issue request of the member for the book if all other constraint are satisfied. The approval is reflected in the member's account after which he can collect the book from the Library.
        - Database record in Members section updated with the new Book added to the Issued list of the given member after the Library clerk approves the issue request.
        - The number of Issued books for that member is increased.
        - Books and Reservations section in the database is updated.

- **Test ApproveReturn()**
    - **General Input**
        - Member ID
        - Book object
    - **General Output**
        - The library clerk will approve the return request of the member from his account after checking the book if it is not tampered. If the book was overdue, he would click on CollectPenalty function, to generate the penalty slip for the member.

        - The database will be updated for the Member as well in the Books section after the Clerk approves the return through his account

- **Test CollectPenalty()**
    - **General Input**
        - A Book object
        - LibraryMember Object
        - Today's Date
    - **General Output**
        - A penalty calculated by the appropriate formula will be displayed in the Penalty Slip for the member which is to be paid to the clerk

Librarian

- **Test Constructor**
- **Test AddMember()**
    - **General Input**
        - Library Member details to create a new member account
    - **General Output**
        - New member is added to the database if all information is provided correctly.
    - **Scenarios**

        - The librarian tries to add a person who is already a member. -> abort the function call and display appropriate message.
        - Incomplete details provided. -> Show appropriate message.

- **Test DeleteMember()**
    - **General Input**
        - Library Member
    - **General Output**
        - Member's information and account is deleted from the database
    - **Scenarios**
        - Try to delete a person who is not a member -> show error message.
        - Delete a member with overdue/unreturned books -> show appropriate warnings and leave the decision to the discretion of the Librarian.

- **Test SendReminder()**
    - **General Input**
        - Member object
        - Subject of reminder
    - **General Output**
        - Notification/Reminder added to the member's reminder inbox.
    - **Scenarios**
        - Librarian sends reminder to a member for an overdue book.

        - Librarian sends reminder to a member that she can issue a book which was reserved by her as the previous issuer of the book has returned it

- **Test CheckBookIssueStatistics()**
    - **General Input**
        - Books section
    - **General Output**
        - List of Books which have not been issued in the last 5 years
    - **Scenarios**
        - All books have been issued in the last 5 years -> shows appropriate message that no such book found
        - Certain books have not been issued in the last 5 years.

Book

- **Test Constructor**
- **Test Getter Functions**
    - **General Input**
        - Book class object
    - **General Output**
        - Output is function-dependent.

- **Scenarios**
    - Getting the ISBN of the Book
    - Getting the Title of the Book
    - Getting the Author of the Book
    - Get the Rack Number of the Book
    - Getting the Issue Status, Issue Date or Due Date of the Book]
    - Getting the number of copies of the book

Basic Guidelines to be followed for the GUI-based web interface

Check Basic GUI elements of the web-interface that will be used by the Staff and the Members for the various functions related to Library.

The following guidelines are to be followed :

1. Aesthetics:

- The web-interface should have proper spacing between the different elements and buttons.
- Line spacing, padding and margins should be uniform and sufficient.
- Font size must be big enough to be easily read. Font family / style should be simple and comprehensive.
- There should be enough space for the search results of Books. The search results should be scrollable if they are large in number.

2. Proper Structure:

- The web-interface should be organised into proper sections and sub-sections for the different functionalities of the different types of users.
- A web-page when refreshed should keep the user logged-in to his account and display the same page as was being displayed before refreshing the webpage.
- A user should not be able to access unauthorised webpages/sections of the interface by making modifications in the URL displayed on Web Browser.

3. Buttons

Check if all buttons are clickable and active and have appropriate labels on them.

4. TextBoxes

Text boxes used on the web-interface to take text input from the user should be wide enough to accept the entire input.

5. Back Navigability

There should be a Go Back button on every web-page of the interface that is active and clickable and navigates back to the previous page. A similar button should be present for returning back to the Home page.

6. Message Boxes and Prompts displayed for Exceptional situations

Errors or warning should be displayed as a prompt for invalid inputs or opting for unavailable functionalities.