IEEE TEST PLAN TEMPLATE

Test Plan Identifier

LIS-TESTPLAN-1

REFERENCES

Software Requirement Specification

Use Case Diagram

Class Diagram

INTRODUCTION

This is the Unit-Test plan for Library Information System Version 1.0

This plan will aim at providing details for the testing of the different methods used in the functioning of LMS both for a user as well as a software developer

The explanation for developers will be more details and for users will include a brief outline

TEST ITEMS (FUNCTIONS)

1. The Constructors for all the Classes

1.1 Book

1.2 UnderGraduateStudent

1.3 PostGraduateStudent

1.4 ResearchScholar

1.5 FacultyMember

1.6 BookHandler

1.7 LibraryClerk

1.8 Librarian

1.9 ActiveReservation

2. The Singleton Nature of the Singleton Classes

3. Member Functions of all Classes

4. Functions outside the classes

5. Utility Functions

FEATURES TO BE TESTED

1. Issue Of a Book

2. Return Of a Book

3. Reservation Of a Book

4. Removing Expired Reservations

5. Update Pending Reservations

6. Penalty Calculation

7. Add a new Member

8. Remove a Member

9. Login

10. Check Issue statistics of Books

11. Add New Book

12. Remove Old/Damaged Books

FEATURES NOT TO BE TESTED

1. Graphic User Interface will not be tested manually.

ITEM PASS/FAIL CRITERIA

We will provide Golden outputs for the appropriate tests.

We will provide appropriate Exception classes for the exceptions

Whenever an expected parameter is not passed, a TypeError is raised.

Match with Golden Output/Exception class will be a PASS, otherwise would be a FAIL

Efficacy would be judged by % of tests passes

SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

Stop tests when some required package compatibility fails

TEST DELIVERABLES

Test Plan Document

Test Suite Document

Unit Testing

MemberLogin()

General Input

- Member ID
- Password
- MEMBERS Table

General Output

- Constructed Object of the Member

Scenarios

- Member Logins in successfully
- Member ID not in database
- Password does not match with Member ID

EmployeeLogin()

General Input

- Employee ID
- Password
- EMPLOYEES table

General Output

- Constructed Object of the User

Scenarios

- Employee Logins in successfully
- Employee ID not in database
- Password does not match with Employee ID

Library Member

- **Test Getter Functions**

  #### General Input

  LibraryMember

  #### General Output

  Specific to the Function (Returns value of field we want to get)

  #### Scenarios

  - Getting the Member ID of the Member
  - Getting the Name of the Member
  - Getting the Number of Books Issued by the Member
- **Test CheckAvailabilityOfBook()**

  #### General Input

  - LibraryMember
  - ISBN of the books
  - Database entry in the RESERVATIONS table for the corresponding ISBN.

  #### General Output

  - Book UIDs when available(depending on reservation status of the Library Member).

    OR
  - Status of Reservation if user has a reservation.

    Scenarios

    - The user has an Active Reservation on this ISBN.
    - The user has a Pending Reservation on this ISBN.
    - The user has no reservation on this ISBN and some UIDs are available. (May have reservations on other ISBN)
    - The user has no reservation on any ISBN and no UIDs are available.
    - The user has a reservation on a different ISBN and no UIDs are avalaible.

- **Test IssueBook()**

    General Input

    - LibraryMember
    - ISBN of book to be issued.
    - Database entry in RESERVATIONS table for the book.
    - Database entry in MEMBERS table for the member.

General Output

- Database record in MEMBERS Table updated with the new Book added to the Issued list.
- Number of Issued books is increased.
- BOOKS and RESERVATIONS Table is updated.

Scenarios

- User tries to claim a book they have already issued
- User has exhausted their permitted number of issues
- User claims a reserved book.
- User issues an available book.
- **Test ReserveBook()**

    General Input

    - LibraryMember
    - ISBN of book to be reserved.
    - Database entry in RESERVATIONS table for the book.
    - Database entry in MEMBERS table for the member.

    General Output

    - Database record in MEMBERS Table is updated with included the new reservation.
    - RESERVATIONS Table is updated.

    Scenarios

    - The book is available e*
    - The book is unavailable and user has made no reservation for any book.
    - The book is unavailable and user has pending/active reservations for some other case.
    - The book is unavailable and user has an active reservation for this book.
    - The book is unavailable and user has a pending reservation for this book.

- **Test CheckForReminder()**

General Input

  - LibraryMember object
  - ISBN
  - MEMBERS table
  - Todays Date

General Output

  - MEMBERs table is updtades
  - Check whether memeber is reminded

Scenarios

  - The user has a reservation on a different ISBN and no UIDs are avalaible.
  - The librarian has called the SendReminder function and the Member has no overdue book/s.
  - The librarian has not called the SendReminder function.

- **Test SearchBook()**

  General Input

  - LibraryMember object
  - Search String
  - BOOKS Table

General Output

- List of ISBN and Names of matching books
- Message if no books present

Scenarios

- No book in the system matches with the search string
- Some subset of books in the system matches with search string
- Searching by Name
- Searching by Author
- **Test UpdateFromDatabase()**

  General Input

  - LibraryMember
  - RESERVATIONS Table

  General Output

  - Database records in MEMBERS updated
  - RESERVATIONS Table is updated with the expired reservation entry deleted
  - LibraryMember gets update

  General Output

  - Member has an expired active reservation
  - Member has pending reservation which becomes active
  - Member has no reservation

UnderGraduateStudent

- **Test Constructor**

  General Input

  Scenario 1:

  - Member ID
  - Name of the Member

  Scenario 2:

  - Member ID
  - Name of the Member
  - Database Entries in the MEMBERS table corresponding to the Member ID
  - Number of Books Issued (calculable)

  General Output

  - Correctly constructed UnderGraduateStudent Object

  Scenarios

  - Librarian wants to add a new Member
  - Existing Member wants to Login,Library Clerk wants to process Return
  - Invalid Member wants to Login, Library Clerk wants to process Return for Invalid Member
- **Test CanIssue()**

  General Input

  - UnderGraduateStudent

  General Output

  - Returns whether the user can issue another book or not

  Scenarios

  - The user has exhausted his limit of book.
  - The user has not exhausted his limit of book.

PostGraduateStudent

- **Test Constructor**

  General Input

  Scenario 1:

  - Member ID
  - Name of the Member

  Scenario 2:

  - Member ID
  - Name of the Member
  - Database Entries in the MEMBERS table corresponding to the Member ID
  - Number of Books Issued (calculable)

  General Output

  - Correctly constructed PostGraduateStudent Object

  Scenarios

  - Librarian wants to add a new Member
  - Existing Member wants to Login,Library Clerk wants to process Return
  - Invalid Member wants to Login, Library Clerk wants to process Return for Invalid Member

- **Test CanIssue()**

  General Input

  - PostGraduateStudent

  General Output

  - Returns whether the user can issue another book or not

  Scenarios

  - The user has exhausted his limit of book.
  - The user has not exhausted his limit of book.

ResearchScholar

- **Test Constructor**

  General Input

  Scenario 1:

  - Member ID
  - Name of the Member

  Scenario 2:

  - Member ID
  - Name of the Member
  - Database Entries in the MEMBERS table corresponding to the Member ID
  - Number of Books Issued (calculable)

  General Output

  - Correctly constructed ResearchScholar Object

  Scenarios

  - Librarian wants to add a new Member
  - Existing Member wants to Login, Library Clerk wants to process return
  - Invalid Member wants to Login, Library Clerk wants to process Return for Invalid Member

- **Test CanIssue()**

  General Input

  - ResearchScholar

General Output

- Returns whether the user can issue another book or not

Scenarios

- The user has exhausted his limit of book.
- The user has not exhausted his limit of book.

FacultyMember

- **Test Constructor**

  General Input

  Scenario 1:

  - Member ID
  - Name of the Member

  Scenario 2:

  - Member ID
  - Name of the Member
  - Database Entries in the MEMBERS table corresponding to the Member ID
  - Number of Books Issued (calculable)

  General Output

  - Correctly constructed FacultyMember Object

  Scenarios

  - Librarian wants to add a new Member
  - Existing Member wants to Login, Library Clerk wants to process Return
  - Invalid Member wants to Login, Library Clerk wants to process Return for Invalid Member
- **Test CanIssue()**

  General Input

  - FacultyMember

  General Output

  - Returns whether the user can issue another book or not

  Scenarios

  - The user has exhausted his limit of book.
  - The user has not exhausted his limit of book.

Library Clerk

- **Test Constructor**

General Input

- EmployeeID
- Database entry in EMPLOYEES table with the corresponding EmployeeID

General Output

- Fully Constructed Library Clerk

Scenarios

- When the library clerk logs in.
- Employee wants to login but EmployeeID is not of a library clerk

- **Test AddBook()**

  General Input

  - ISBN
  - Name
  - Author
  - Rack number
  - Today's Date

  General Output

  - BOOKS and RESERVATIONS tables are updated.

  Scenarios

  - The book with same ISBN already exists and pending reservations exist
  - The book with same ISBN already exists and pending reservations do not exist
  - The book with same ISBN doesn't already exist.

- **Test DeleteBook()**

  General Input

  - BOOKS TABLE

  General Output

  - BOOKS and RESERVATIONS tables are updated.

  Scenarios

  - Books are marked as disposed
  - No books marked as disposed

- **Test ReturnBook()**

  General Input

  - A Book object
  - LibraryMember object
  - RESERVATIONS and MEMBERS tables

  General Output

  - MEMBERS and RESERVATIONS tables are updated.

Scenarios

- Member tries to return a book they havent issued
- Member tries to return a book which is not present in the library
- The book has pending reservation which moves to active.
- The book doesn't have pending reservation.

- **Test CollectPenalty()**

  General Input

  - A Book object
  - LibraryMember Object
  - Today's Date

  General Output

  - Penalty collected by formula

  Scenarios

  - The return date is beyond due date.
  - The return date is within the due date.

Librarian

- **Test Constructor**

  General Input

  - EmployeeID
  - Database entry in EMPLOYEES table corresponding to the EmployeeID

  General Output

  - Fully Constructed Librarian

  Scenarios

  - The EmployeeID is of the Librarian, i.e., LIB0001 (fixed ID of Librarian)
  - The EmployeeID is not the Librarian but is a Library Clerk.
  - The EmployeeID is not the Librarian but is a Library Member
- **Test Super Class Functionalities**

  General Input

  - Specific to each function

  General Output

  - Specific to each function

  Scenarios

  - Same scenarios as each Functions of the Super Class
- **Test AddMember()**

General Input

    - Library Member
    - MEMBERS table
    - Password

  General Output

    - MEMBERS table updated

  Scenarios

    - The librarian tries to add a person who is already a member
    - The librarian wants to add a new member.
    - Name is missing in entry field
    - MemberID is missing in entry field
    - Type is missing in entry field
    - Password is missing in entry field

- **Test DeleteMember()**

  General Input

    - Library Member
    - MEMBERS table.

  General Output

    - MEMBERS table updated

  Scenarios

    - Try to delete a person who is not a member
    - Delete an existing member
    - Delete a member with overdue/unreturned books

- **Test SendReminder()**

  General Input

    - MEMBER TABLE

  General Output

    - MEMBER TABLE updated

  Scenarios

    - Send Reminder to members

- **Test CheckBookIssueStatistics()**

  General Input

    - BOOKS table

  General Output

    - List og Books which have not been issued in the last 5 years

Scenarios

- All books have been issued in the last 5 years.
- Books have not been issued in the last 5 years.

- **Test DisposeBook()**

  ### General Input

  - UID
  - BOOKS table

  ### General Output

  - Database entry in BOOKS table has been marked as disposed.

  ### Scenarios

  - UID does not exist
  - The UID has not been issued in last 5 years
  - The UID has been issued in the last 5 years

Book Handler

- **Test Create Function**

  ### General Input

  - None

  ### General Output

  - Fully constructed Singleton BookHandler Object (Constructed only the first time, same instance is returned every time)

  ### Scenarios

  - No specific scenarios, only called to create a reference to Singleton BookHandler Object whenever required.

- **Test OpenBook()**

  ### General Input

  - A Book Object

    OR
  - ISBN

  ### General Output

  - The BookHandler's data members are populated.

  ### Scenarios

  - Called with the ISBN when UID is irrelevant for the function calling OpenBook()
  - Called with the Book Object when UID is relevent for the function calling OpenBook()

- **Test Singleton Nature of the object**

  General Input

  - None

  General Output

  - None

  Scenarios

  - Call Create() twice and compare address of the objects returned by them
- **Test UpdateBook()**

  General Input

  - None

  General Output

  - Data members are updated
  - Database entry, corresponding to the Members whose active reservation expired, in MEMBERS table is updated

  Scenarios

  - Pending reservations are there, Some active reservations are expired.
  - Pending reservation are there, No active reservations are expired.
  - No pending reservations are there, Some active reservations are expired.
  - No pending reservation are there, No active reservations are expired.
- **Test IssueSelected()**

  General Input

  - MemberID

General Output

- MEMBERS, BOOKS and RESERVATIONS table are updated.

Scenarios

- Member is claiming a book reserved to them.
- Member is issuing an available book
- **Test ReturnSelected()**

  General Input

  - MemberID

  General Output

  - MEMBERS, BOOKS and RESERVATIONS table are updated.

Scenarios

  - The book has pending reservation which moves to active.
  - The book doesn't have pending reservation.
- **Test ReserveSelected()**

  #### General Input

- MemberID

General Output

- MEMBERS, BOOKS and RESERVATIONS table are updated.

Scenarios

- The member doesn't have pending/active reservation for this/another book

Book

- **Test Constructor**

  #### General Input

  - Book basic information

  OR

- Database entry of the book in BOOKS table.

General Output

- MEMBERS, BOOKS and RESERVATIONS table are updated.

  #### Scenarios

  - Book is created for adding
  - Book is created for using with BookHandler.

Active Reservation

- **Test Constructor**

  #### General Input

  - Member ID
  - Date reservation became active.

General Output

- Object Created.

Scenarios

- Active reservation is made at any time in the run

GUI Testing

Check Basic GUI elements

For the following Tkinter GUI elements, we describe the basic properties that must be tested for appropriate/error-free behaviour wherever they appear in our GUI

1. Buttons

Check if all buttons are clickable and active

2. RadioButtons

Check if exactly one is selected

3. CheckBoxes

Check if atleast one is selected

4. TextBoxes

Check for text entry is not empty

5. DropDown Menus

Check if exactly one option is selected

6. ListBoxes/ComboBoxes

Check if atleast one option is selected

Check Common GUI features

1. 'Back' Buttons

Check if provided in every page to Go Back to the previous page

2. Submit/OK Buttons

Check if all 'Required' text entries are filled before execution

3. 'LogIn' and 'LogOut' Buttons

Check if these buttons safely execute login and logout for Members and Employees

Other Specific Features

1. Dynamic Search List Boxes

These are searchable ListBoxes, with a search TextBox linked. The string being entered into the TextBox is used for substring search in the ListBox in real-time. The matching results are filtered and displayed in real-time.

2. Message Box displayed for Exceptional situations

We display a message box, whenever an exceptional situation is met. Example: A required text entry is left empty.

The above features are to be tested as well