# Test Plan Document

**For**

# LIBRARY INFORMATION SYSTEM


## Authors:

**Paramananda Bhaskar (21CS30035)**
**Galipelli Sai Mallikarjun (21CS30019)**
**Datta Ksheeraj (21CS30037)**

## Instructors:

**Prof. Sourangshu Bhattacharya**
**Prof. Abir Das**
**Animesh Mukherjee**

**Indian Institute Of Technology Kharagpur**

**3rd April 2023**

# 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 version 1.0.
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:

1) **def index (request):** This function renders Homepage
2) **def reg(request) :** This function creates an User object along with the inherited Student or Faculty class object depending upon the registration credentials.
3) **def login_karo (request) :** This function authenticates the details provided in the login form by the user. Depending upon the credentials, the user is recognised properly as a Student or a Faculty or a Clerk. Upon recognition, Students and Faculties are redirected to their corresponding profile pages and Clerks are redirected to a page from where they can carry on their authorized functionalities.
4) **def send_otp (request):** sends otp upon request through email in case of "Forgot Institute Id or Password".
5) **def enter_otp (request):** authenticates otp entered by user
6) **def forgot_password(request):** shows the Institute Id and gives option to change to new password
7) **def profile(request):** renders profile page for Students and Faculties where they can see their personal info, issued books, reserved books; also can issue , reserve or return books.
8) **def afterlogin(request):** renders after-login page for clerks having options for adding new books, view books, edit books and delete books.
9) **def edit_profile(request):** Students and Faculties can change their editable personal info.

10) **def add_book(request):** This function allows the Clerks to add a new book to the database.
11) **def view_books(request):** This function allows the Clerks to view the book database.
12) **def delete_book(request, myid):** This function allows the Clerks to delete a book having id =myid from the database.
13) **def edit_book(request, myid):** This function allows the Clerks to edit a book's details having id =myid from the database.
14) **def change_password(request):** This function allows the users to change their login password.
15) **def Logout(request):** Logs out users to the Homepage
16) **def issue_book(request, myid):** Students and Faculties can issue books having id = myid using this function.
17) **def reserve_book(request, myid):** Students and Faculties can reserve books(if available) having id = myid using this function.
18) **def return_book(request, myid):** Students and Faculties can return books having id = myid using this function,

# FEATURES TO BE TESTED :

### 1. ADMIN-RELATED FEATURES:

a. *Library Clerk*:
   1. Add new books
   2. Delete old books from database

b. *Librarian*:
   1. Add new user
   2. Add Library clerks
   3. Delete existing members
   4. Add new books
   5. Delete old books from database

### 2. MEMBER-RELATED FEATURES:

1. Issue Books
2. Returning Books
3. Reserve Books
4. Edit his/her profile
5. Changing his/her password

### 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**

MemberLogin()

1.General Input

1. Institute_Id
2. Password

Test SearchBook()

1.General Input

1.Search String (Can be title, author, ISBN, rack number, number of copies)

2. General Output

1. List of ISBN and names of matching books

## 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:

## user_login()

- **General Input**
  Institute ID
  Password

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

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

## Stafflogin()

- **General Input**
  Staff Institute ID
  Password

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

- **Scenarios**
  Staff logs in successfully
  Staff Institute ID does not match the database

Password does not match with staff ID
User tries to enter User 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 User

- ## Test OpenProfile()

  General Input:

  Library User(Faculty, Students)

  General Output:

  User Profile

  Scenarios:

  Getting the Name of the Member

  Getting the ID of the Member

  Getting the Institute ID of the Member

  Getting the list of currently issued books by the user

  Getting the information of currently reserved book by the user

- ## Test CheckAvailabilityofBook()

  **General Input**:

  Book object previously retrieved from the database.

  **General Output**:

  Output the total number of copies, number of copies issued and whether it is available to issue or not.

**Scenarios:**

The book number of total book copies greater than copies issued the it is available for issuing.

The book is currently unavailable to issue

- Test IssueBook()

It is a function of each of the 4 derived classes of the User class.

**General Input:**

User object.

Book object for the book to be issued

**General output:**

Returns whether the Member can issue another book or not depending on the user book limit.

Return whether the book is available to issue or not.

**Scenarios:**

The user has exhausted his limit of books.

The book the user wants to issue is not available.

The book the user wants to issue can be reserved if no one reserved it earlier.

- Test ReserveBook()

**General Input:**

Library User object.

Book object that the user wants to reserve.

**General output:**

Returns whether the Member can reserve another book or not as a user can reserve only one book .

Return whether the book is available to reserve or not.

**Scenarios:**

The user has exhausted his limit of books to reserve.

The book the user wants to issue is not available to reserve.

The book the user wants to reserve is already reserved by the user.

- Test ReturnBook()

**General Input:**

Library User object.

Book object that the user wants issued already.

**General output:**

Issued book objects get deleted.

Updates the 7 days availability if anyone has reserved

**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 the user.

- Test CheckforReminder()

**General Input:**

Library User object

**General output:**

Shows all the notifications and reminders to the member (if any).

**Scenarios:**

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

# Library Clerk

- Test AddBook()

**General input:**

ISBN, Title, Author
Rack number
Category
Number of copies

**General Output:**

The book gets added to the database if the same ISBN book doesn't already and the Books section of database is updated

If the same ISBN book exists, the number of copies will increase.

**Scenarios:**

The book with the same ISBN already exists -> Increase the number of copies.

The book with same ISBN doesn't exist in the database then the book is Successfully added.

- Test DeleteBook()

  **General Input:**

  Book object

  **General Output:**

  Book object is deleted from the database.

- Test EditBook()

  **General Input:**

  Book object

  ISBN, Title, Author, rack_number etc.

  **General Output:**

  Book object edited with the details in the database.

- Test CollectPenalty()

  **General Input:**

  A book object

  Library user object

  Date

  **General Output:**

  A penalty is calculated by the appropriate formula will be displayed in the penalty slip for the member which is to be paid to the library.

# Librarian

- **Test Constructor** ( command: **python manage.py createsuperuser**)
- **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 messages.
    Incomplete details provided. -> Show appropriate message.

- **Test DeleteMember()**
  - **General Input**

    Library Member

- **General Output**

  Member's information and account is deleted from the database

- **Test SendReminder()**
  - **General Input**

    Member object
    Subject of reminder

  - **General Output**

    Notification/Reminder added to the member's reminder inbox.

  - **Scenarios**

    Librarian sends a 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**

    Category 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 Getter Functions

**General Input:**

Book class object

**General Output:**

Output is function-dependent.

**Scenarios:**

Getting 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 the 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 organized 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 unauthorized web pages/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