Software Requirements Specifications

for

LIBRARY INFORMATION SYSTEM

Version 1.O approved

Authors :
Nakul Aggarwal (19CS10044)
Rupinder Goyal (19CS10050)
Ashwamegh Rathore (19CS30009)

Instructors :
Prof. Sourangshu Bhattacharya
Prof. Abir Das
Owais Iqbal

Indian Institute Of Technology Kharagpur
20 March 2021

TABLE OF CONTENTS

LIBRARY INFORMATION SYSTEM

1. INTRODUCTION

With the increase in the number of readers and also in the number of books, better management systems of libraries are required. This *Library Information System* focuses on improving the management of a Library, particularly in a university. “What if you could check whether a book is available in the Library through a simple interface?”. The integrated *Library Information System* provides you the ease of issuing, returning, or reserving a book from a Library within your campus through a simple application. The *Library Information System* is developed on the *Windows* platform which basically focuses on the most fundamental tasks concerning a *Library*, alongside several other features that come in very handy in managing hundreds of thousands of books in a typical *Library*.

1.1. PURPOSE

The purpose of the project is to maintain the details of books and members of a Library in a structured way. The principal motive is to maintain an easy circulation system between the individuals who use the Library and the management staff that comprises of the clerks and the Librarian. The fundamental tasks that form the foundation of this application are issuing, returning and reserving a book; analyzing popularity of books and maintaining histories of all activities that are associated with some of these.

This document describes the hardware and software interface requirements alongwith *UML* diagrams. The document gives the detailed description of the both functional and non-functional requirements proposed by the client.

1.2. DOCUMENT CONVENTIONS

- Entire document is *justified*.
- Convention for *Main Title*
  - Font face: *Arial*
  - Font style: *Bold / Italics*
  - Font Size: 18
- Convention for *Sub Title*
  - Font face: *Arial*
  - Font style: *Bold / Italics*
  - Font Size: 14

● Convention for *Body*

Font face: *Arial*

Font Size: 12

Important keywords inside a *body* may be italicized.

Indentation of text is done wherever required.

1.3. INTENDED AUDIENCE AND READING SUGGESTIONS

The intended readers of this document are the developers of this application, testers, library owners, managers, coordinators and administrators. Any suggested changes on the requirements listed on this document should be included on a fresh page succeeding the last page of this document so that it can be a reference to the development and validating teams.

The readers are suggested to refer to the section *1.5. REFERENCES* to understand the meaning of certain abbreviations and acronyms and to learn about the standard *SRS Document* format. The readers are also suggested to refer to *APPENDIX A : GLOSSARY* to know the meaning and context of certain keywords used in the document.

1.4. PRODUCT SCOPE

LIS product is basically updating the manual library system into a *GUI-based* application so that the users can know the details of their accounts, availability of books and perform various tasks associated with issuing and returning books. The project is specifically designed for the use of *Library administration* and *Library users*. The product will work as a complete user interface for library management processes and library usage for ordinary users. *LIS* can be used by any existing or new library to manage its books and book borrowing, insertion and monitoring . *LIS* can work as a powerful library management system for big libraries, and can provide a free, easy-to-use system for rising libraries.

1.5. REFERENCES

● *Acronyms / Abbreviations*

- API : Application Programming Interface
- CPU : Central Processing Unit
- DB : Database
- DBMS : Database Management System
- GB : GigaByte
- GUI : Graphical User Interface

- ID : Identity
- IDE : Integrated Development Environment
- IEEE : Institute of Electrical and Electronics Engineers
- ISBN : International Standard Book Number
- LIS : Library Information System
- LMCN : Library Membership Code Number
- MS : Microsoft
- OS : Operating System
- PC : Personal Computer
- PS/2 : Personal System/2
- R : Registered Trademark
- RAM : Random Access Memory
- RDBMS : Relational Database Management System
- SQL : Structured Query Language
- SRS : Software Requirement Specification
- TM : Trademark
- UML : Unified Modeling Language

- *Texts*
  - *IEEE 830-1998 Standard* for writing SRS documents.
  - *I. Sommerville, Software Engineering, 8th ed. England: Addison-Wesley 2007*

- *Links*

  The *UML Diagrams* were designed on ***Visual Paradigm Online***

2. OVERALL DESCRIPTION

2.1. PRODUCT PERSPECTIVE

The *Library Information System* will take care of the details and availability of every book in the Library. The users can be *students*, *faculty members* or the management staff members that include *clerks* and the *Librarian*. The system will provide a search functionality to facilitate the search of resources. This search will be based on various categories viz. book's title, book's author or the *ISBN* identity code. Further the Library staff personnel can add the resources and register new or deregister the existing members from the system.

Users should have a password to access their respective accounts. The functionalities and interfaces available to every user depends on whether she is a

staff personnel or a member of the library. Books can be *issued, reserved* or *returned* and the fine can be paid only through a *clerk's* account. Besides, the *clerk* can have additional liberties to *add* or *remove* books from the catalogue.
The *Librarian* should be able to view the current state of the *Library*, in terms of the details of all the *members*, various books in the catalogue, books that are currently issued out, books that have not been issued in three or five years etc. She should also be able to view the old records related to the books that were issued in the past.

2.2. PRODUCT FUNCTIONS

- *REGISTRATION / DEREGISTRATION*
    - ○ Only the *Librarian* has the privilege of registering a new member as a user of *LIS* or de-registering an existing user.
    - ○ A member can be a *faculty* member or a *non-faculty* member. A *non-faculty* member can be an *undergraduate student*, a *postgraduate student*, or a *research scholar.*
    - ○ Once a member's account is created by the *Librarian*, the member can sign into the *LIS* with the *unique Library Membership Code Number* and a default password. The password can be *reset* by the member anytime by logging in.

- *QUERY SECTION*
    - ○ No account needed to search through query *section* of the *LIS*.
    - ○ Queries include search by the book's *title, author* or *ISBN code.*

- *ISSUE / RETURN / RESERVE A BOOK*
    - ○ A book can only be issued/reserved/returned through a *clerk*'s account.
    - ○ The *LIS* should update the concerned tables in all the databases to reflect the activity of issuing/returning/reserving performed.
    - ○ A *notification* should automatically be sent to the *member*'s account (with all the details) confirming the activity performed.
    - ○ The member must be notified about the *penalty* in case the book is returned after the due date.

- *ANALYSIS ON POPULARITY OF BOOKS*
    - ○ The *Librarian* can log into her account and get a list of books in the catalogue that are not *issued* in the last one, three or five years.

- *ADD / REMOVE BOOKS FROM THE CATALOGUE*
    - A clerk has the privilege to update the *catalogue* when a book is disposed off.
    - Similarly, she must be able to add books to the *catalogue* when a new book is received by the *Library*.

2.3. USER CLASSES AND CHARACTERISTICS

The system *LIS* provides different types of services, privileges and functionalities based on different types of users (*members/Librarian/clerks*). The *Librarian* will be acting as a controller and will have most of the privileges of an administrator. A *clerk* conducts all the general office tasks related to managing books, fines and issuing. A *member* has the least control over the system in terms of access to the databases and transparency. The member can be either a student or faculty-staff of the university who will be accessing the Library through this portal.

The features that are available to the *Librarian* are -
- *Register* a new member of the Library as a user of *LIS*
- *De-register* an existing member of the Library as a user of *LIS*
- *Perform analytics* on the popularity/usage of books in the Library
- Check the details of all the *existing books* in the *catalogue*
- Check the details of all *currently issued* books
- Check the details of all books *issued in the past*
- Check the books that are currently issued out and are *over due*
- *Get details* of all the members of the Library
- *Ask queries* through search by book's title, author or ISBN code
- *Reset* their account's *password*

The features that are available to the *Clerk* are -
- *Issue* an available book for a member and update it in the system
- *Reserve* an issued book for a member and update it in the system
- Collect the book *returned* by a member and update it in the system
- Collect *fine paid* by a member as a penalty of returning a book late
- *Add new books* to the Library's *catalogue* and update it in the system
- *Remove existing books* from the Library's *catalogue*
- *Ask queries* through search by book's title, author or ISBN code
- *Reset* their account's *password*

The features that are available to a *Member* are -
- *Ask queries* through search by book's title, author or ISBN code

- Check the details of the books *currently issued* by them
- Check the details of the books *issued* by them in the *past*
- *Receive automatic notifications/reminders* through *messages*
- *Reset* their account's *password*

2.4. OPERATING ENVIRONMENT

The product would be operating in *Microsoft Windows 10* environment. The hardware configurations of the system on which this application would be used would include

Hard Disk: 474 GB
System Type: x64-based PC
Processor: Intel(R) Core(TM) i5-8265U CPU@1.60GHz 1.80 GHz
Installed RAM: 8.00 GB (7.85 GB usable)
Keyboard: Standard PS/2 Keyboard

It should preferably be a *Python* application with a *graphical user interface* and *3.7.8 version* should be used. At the back end, it must preferably use *MySQL* as the *open-source RDBMS.*

The basic input devices that can be needed are *keyboard*, *trackpad (or mouse)* and output devices such as a *monitor*.

2.5. DESIGN AND IMPLEMENTATION CONSTRAINTS

- *IMPLEMENTATION CONSTRAINTS*
  - *Relational Databases* must be used. Avoid *object databases*.
  - *MySQL* must be highly preferred as the *RDBMS* at the back-end. Other modern *SQL* based RDBMSs like *PostgreSQL, MS Access, IBM DB2* can be used in extreme cases.
  - The application must have a *graphical user interface*.
  - Interaction between the interfaces that require external input and the users must be possible with minimal input from the user's side.

- *SECURITY / PRIVACY CONSTRAINTS*
  - Passwords must not be stored as plain texts in the database.
  - Passwords must be *hashed* before storing in the database.
  - *Members* must have full privilege to change their passwords that are unknown to the administration.
  - Neither a *clerk* nor the *Librarian* can change / control the password of a *member* through the *LIS*, except when she is registered and is allotted a *default password*.

○ Messages sent to the members' accounts must not be stored as plain text in the database.
○ *Reversible cryptographic encryption* must be used to store the messages in the database.

- *LIMIT AND DUE-DATE CONSTRAINTS*
  ○ The maximum number of books an *under-graduate* member can have issued currently is 2.
  ○ The maximum number of books a *post-graduate* member can have issued currently is 4.
  ○ The maximum number of books a *research scholar* member can have issued currently is 6.
  ○ The maximum number of books an *faculty member* can have issued currently is 10.
  ○ An *under-graduate* member and a *post-graduate* member can have a book issued for at most 1 month without any penalty.
  ○ A *research scholar* member can have a book issued for at most 3 months without any penalty.
  ○ A *faculty member* can have a book issued for at most 6 months without any penalty.

- *ACCESS CONSTRAINTS*
  ○ *Members* must not be able to access any underlying databases.
  ○ *Clerk* should have *write* access to only the selected tables concerning the *Library's catalogue* in the database; just to add and remove books.
  ○ *Librarian* should have *write* access only to the *Accounts* database. The access must be as limited as possible so that only *insertion* and *deletion* of accounts is allowed.

- *COMMUNICATION CONSTRAINTS*
  ○ All the activity-related messages must be sent automatically by the *LIS,* without the administrators to explicitly write a message.
  ○ This automatic messaging system should be triggered when
    - a book is issued successfully
    - a book is reserved successfully
    - a book is returned successfully
    - fine is paid successfully
    - reservation of a book has expired
    - the reserved book is returned by the former issuer

○ *Members* should not be able to send messages to a *clerk*, to the *Librarian*, or to other *members*.
○ The *administrators* also might not need to send explicit/personal messages to any user.

- *DESIGN CONSTRAINTS*
  ○ All icons and modules must fit and be properly spaced and arranged on the screen without causing any clutter.
  ○ The role of each and every module/icon must be specified clearly by its name.
  ○ *Back-navigation* must be allowed.
  ○ *Colour scheme* and *design* must be simple and minimal.

2.6. ASSUMPTIONS AND DEPENDENCIES

- *ASSUMPTIONS*
  ○ The application is able to *handle all use cases*, through errors and exceptions if needed in some scenarios.
  ○ All the users *know their login IDs and their passwords*.
  ○ The system *deletes/frees records* as and when needed to prevent an exponential rise in the memory used by the database(s).
  ○ The interface between the product and the database is *efficient and quick*.
  ○ The application has the ability to *run for at least 20 hours a day* without crashing.
  ○ The application *selectively* provides functionalities to its users depending on their designation.

- *DEPENDENCIES*
  The product needs *MySQL* server to store the databases. The success of this system depends on
    ○ end users (*members* and *administrators*) are able to use the application without any external guide.
    ○ application interface is friendly and easy-to-use.
    ○ users are able to learn more about the product through the error pop-ups and warning messages.
    ○ the application is very responsive and does not hang/glitch
    ○ the users need to enter minimal details to run interfaces.

3. EXTERNAL INTERFACE REQUIREMENT

3.1. USER INTERFACES

The application should have a *graphical user interface* through which the users can perform the required tasks. The following interfaces should be included in the implementation of the *LIS.*

● *LOGIN INTERFACE*

On the *homepage* of the application there should be three modules through which a *Librarian, clerk* and a *member* can respectively sign in. The *log-in page* should ask for *LMCN* from the members (and *Library ID Code* from the administrators) alongwith their respective passwords to sign-in successfully.

● *SEARCH INTERFACE*

The primary search functionality to be implemented is *Search Through Library's Catalogue*. This page should be navigable from the homepage of the application. One need not be signed-in, or even have an account for that matter, to use this functionality.

The search criteria should include search by book's *title, author* and *ISBN code number.* Hybrid/mixing of two search criterion need not be implemented, i.e, for a particular search query, exactly one search criterion out of the three should be used.

● *LIBRARIAN'S CONTROL PANEL*

When the *Librarian* logs into her account, she should have certain modules displayed on the *first page*. Modules for each of the following interfaces must be available.

- *Register a new member* -- The *Institute ID Number*, *Name* and *Date-Of-Birth* of a person should be enough information to register her as a member. Besides, the type of member (*faculty, under-graduate, post-graduate, research scholar*) can also be explicitly taken as input.
- *De-register an existing member* -- The *LMCN Code Number* should be enough information to identify the member and de-register her.
- *See popularity of certain books* -- The *Librarian* should be able to choose any number as input (or rather choose from a

predefined list of numbers) and then see the list of books that have not been issued in those many number of years.

- *See all currently issued books* -- This module must not require any input to function. It displays all the books that are currently issued out, alongwith the limited details of the issuer.
- *See all books issued in the past* -- This module must not require any input to function. It displays all the books that were issued in the past and are now returned, alongwith the limited details of the issuer and some details related to the date of return and fine.
- *See all current members of the Library* -- This module must not require any input to function. It displays the details (limited) of all the members of the Library.
- *Reset password* -- The user must specify her current password (once) and the new desired password (twice) in order to change password.

● *CLERK'S CONTROL PANEL*

When a *clerk* logs into her account, she should have certain modules displayed on the *first page*. Modules for each of the following interfaces must be available.

- *Issue a book to a member* -- The *LMCN* of the issuer and *ISBN* of the book should be enough information to issue her a book. Inside this module, there should be a sub-module that has an interface to issue a *reserved book*. To issue such a book, the member who *reserved* the book must provide the *reservation key* (that was sent to her when the reservation was confirmed), optionally in addition to the details to be provided in the base *issue page*.
- *Receive a returned book* -- The *Issue ID Number* should be enough information to identify the details of both the book issued and the issuer from the database. The module to *Pay Fine* must be navigable from inside this module. If the returned book has a penalty, a dialogue box should ask to navigate to the Pay *Fine* module in order to pay the penalty. This dialogue box can be closed if the issuer at the moment does not have any money (*fine pending* status should be reflected in the database in this case).

- *Reserve an issued book* -- The *LMCN* of the member and the *ISBN* of the book are enough information to search through the system for the *Issue ID* of the record that has this desired book and is not already *reserved*.
- *Pay fine* -- The *clerk* should specify the *Issue ID* corresponding to the issued book for which the *issuer* has come to pay the fine.
- *Add a new book to the catalogue* -- The clerk must specify the details such as the *ISBN*, *title*, *author* and *language* of the book, *number of copies* of the book and the *rack number/location* of the book in the library, in order to add the book to the catalogue.
- *Remove existing books* -- The *ISBN code number* is enough information to identify a book. This must be specified in order to remove the desired book from the catalogue.
- *Reset password* -- The user must specify her current password (once) and the new desired password (twice) in order to change password.

● *MEMBER'S CONTROL PANEL*

When a *member* logs into her account, she should have certain modules displayed on the *first page*. Modules for each of the following interfaces must be available.

- *Show messages* -- Perhaps the most important module for a *member*, this must show a list of all the *notifications* that are sent by the *LIS*. Besides, this must also show all the *reminders*, related to the
    - ○ approaching due date of returning an issued book
    - ○ pending return of issued book beyond its due date
    - ○ reserved book waiting to be formally issued
    - ○ pending payment of a fine

  Reminders may be highlighted differently to distinguish from the *notifications*.
- *Show currently issued books* -- This module must not require any input to function. It shows all the books that are currently issued by the user at that moment.
- *Show books issued in the past* -- This module must not require any input to function. It shows all the books that were issued by the user in the past.

- *Reset password* -- The user must specify her current password (once) and the new desired password (twice) in order to change password.

● *STANDARD BUTTONS / OPTIONS*

- There must be a button to *log out* of the account on the *first page* that appears after logging in. Only one *log out* button should be enough.
- On every page/screen (except on the *homepage* and the *first page* that appears after logging in) there must be a *Go Back* button to revisit or go back to the previous page.
- The user should be able to *close* the application arbitrarily at any point of time irrespective of on which page she is and if she is logged in or not. A *close* button on the top bar of the application should solve the purpose.

● *ERROR MESSAGE DISPLAY STANDARDS*

Error messages must pop up in case the task the user wanted to perform could not be accomplished. This might be due to

- *incorrect format* of any of the input details
- correct format of input details but *lack of compatibility or relevance*
- *incorrect values* of any of the input details
- *certain non-optional input fields were not filled*
- *restricted action* requested by the user to be performed

Following are some obvious examples where the error messages must pop up to intimate the user of the issue with the task.

- Login ID does not exist (*login error*)
- Incorrect password entered (*login error*, *change password error*)
- Two new passwords do not match (*change password error*)
- No book with the entered *ISBN* exists (*issue error, reserve error, remove book error*)
- No member with the entered *LMCN* exists (*issue error, reserve error, de-register error*)
- Reservation key does not match (*issue reserved book error*)
- *Issue ID* does not exist (*return error, pay fine error*)
- Entered *ISBN* already exists (*add new book error*)
- Entered *Institute ID* already exists (*register error*)
- Reserved book has not been yet returned (*issue reserved book error*)

Similarly other such important error messages should pop up in suitable scenarios.

Following are some specific actions (*restricted actions*) that should not be performed as per the norms of the Library.

- Member has not cleared all her fines (*de-register error*)
- Member has not returned all the issued books (*de-register error*)
- Some copy(ies) of this book are issued out (*remove book error*)
- Some copy(ies) of this book are reserved (*remove book error)*
- The *under-graduate member* has already met the limit of 3 books. (*issue book error*)
- The *post-graduate member* has already met the limit of 4 books. (*issue book error*)
- The *research scholar member* has already met the limit of 6 books. (*issue book error*)
- The *faculty member* has already met the limit of 10 books. (*issue book error*)

● *CONFIRMATION DIALOG BOXES*

Tasks like *removing a book* and *de-registering a member user* are very critical and almost irreversible. Though *removing a book* from the *catalogue* accidently would not have much impact because this command is executed by the *LIS* only if any copy of that book is neither issued out nor reserved. Therefore it can be easily added back to the catalogue by the *Librarian.*

But if a member is accidentally deregistered, the database will lose all history of her past issued books (and possibly other integral information) that cannot be restored on re-registering that member. To prevent these accidents, *warning dialogue boxes* must pop up whenever these tasks are to be performed. The boxes must briefly specify the consequences of the action.

3.2. HARDWARE INTERFACES

The primary (and possibly the only) hardware interface between the software product and the user includes a *keyboard* and a *trackpad/mouse*. All pages/modules that require some user input to function should be able to accept input through a *Standard PS/2 Keyboard*. The mouse/trackpad should be functional. Optionally, the application might render a sharp *error sound* in case of errors and a *soft sound* when an information/warning dialog box pops up.

The application should *not* showcase any more extensive, heavy hardware dependency or tethering, such as with *printers* or *scanners*.

3.3. SOFTWARE INTERFACES

- *DATABASES*

There should mainly be two databases that support the back-end.

- *DB : Library*
  - *TABLE : Catalogue*
  - *TABLE : BooksCurrentlyIssued*
  - *TABLE : ReservedBooks*
  - *TABLE : BooksIssuedInPast*
- *DB : Accounts*
  - *TABLE : Administrators*
  - *TABLE : FacultyMembers*
  - *TABLE : NonFacultyMembers*
  - *TABLE : Messages*

Functionality of almost all the *user interfaces* is tethered to one or both the databases. These connections between the product and the databases are in almost all cases a *write* access, with varying degrees of interaction and freedom. The touch points between the databases and the product are as follows.

User Interface	Database.Table	Interface Type
Login	*Accounts.Administrators* *Accounts.FacultyMembers* *Accounts.NonFacultyMembers*	*Read-only* *Read-only* *Read-only*
Query Section	*Library.Catalogue*	*Read-only*
Register	*Accounts.FacultyMembers* *Accounts.NonFacultyMembers*	*Write (Insertion)* *Write (Insertion)*
De-register	*Accounts.FacultyMembers* *Accounts.NonFacultyMembers* *Accounts.Messages* *Library.BooksCurrentlyIssued* *Library.ReservedBooks* *Library.BooksIssuedInPast*	*Write (Deletion)* *Write (Deletion)* *Write (Deletion)* *Read-only* *Write (Deletion)* *Write (Deletion)*
Popularity Analysis	*Library.Catalogue*	*Read-only*

See Currently Issued Books	*Library.BooksCurrentlyIssued*	*Read-only*
See Books Issued In Past	*Library.BooksIssuedInPast*	*Read-only*
See All Members	*Accounts.FacultyMembers* *Accounts.NonFacultyMembers*	*Read-only* *Read-only*
Reset Password	*Accounts.Administrators* *Accounts.FacultyMembers* *Accounts.NonFacultyMembers*	*Write (Update)* *Write (Update)* *Write (Update)*
Issue Book	*Accounts.FacultyMembers* *Accounts.NonFacultyMembers* *Accounts.Messages* *Library.Catalogue* *Library.BooksCurrentlyIssued* *Library.ReservedBooks*	*Write (Update)* *Write (Update)* *Write (Insertion)* *Write (Update)* *Write (Insertion)* *Write (Deletion)*
Reserve Book	*Library.BooksCurrentlyIssued* *Library.ReservedBooks* *Accounts.Messages*	*Write (Update)* *Write (Insertion)* *Write (Insertion)*
Return Book	*Library.BooksCurrentlyIssued* *Library.BooksIssuedInPast* *Library.ReservedBooks* *Library.Catalogue* *Accounts.Messages*	*Write (Deletion)* *Write (Insertion)* *Write (Update)* *Write (Update)* *Write (Insertion)*
Pay Fine	*Library.BooksIssuedInPast* *Accounts.Messages*	*Write (Update)* *Write (Insertion)*
Add New Book	*Library.Catalogue*	*Write (Insertion)*
Remove Book	*Library.Catalogue*	*Write (Deletion)*
Show Messages	*Accounts.Messages*	*Read-only*

● *OPERATING SYSTEM*

Extensions in the programming languages, specifically meant for *Microsoft Windows*, that might provide access to much of the *WIN32 APIs* should be strictly avoided. Any other *operating system specific* tools, libraries and packages should be avoided.

4. SYSTEM FEATURES

4.1. LOGIN SYSTEM / ACCOUNTS

4.1.1. Description And Priority

The *LIS* should have a *login system* through which the administrators and the members can sign-in and use the various available functionalities. Most of the modules are accessible only when the user is logged-in. A login-id, which the *LMCN* for the members and some *Library Identity* for the administrators, along with a password should be enough information required to log into an account.

Accounts for new *members* can be registered only by the *Librarian*. Similarly, only the Librarian can deregister an existing member from the *LIS*. All users have the liberty to change their passwords (once they login with the password provided by the administration) and log-out of their accounts.

Overall Priority Quotient : HIGH

4.1.2. Stimulus / Response Sequences

- The user opens the *log-in form* by clicking on one of the buttons on the home screen.
- The form asks for the user's login-id and password.
- In case the user is a *member*, she must enter her *LMCN* as the log-in id. Otherwise if she is an *administrator* (*clerk* or *Librarian*), she should enter her *Library ID* as the log-in id.
- The user should enter the correct password.
- Provided the details entered by the user are correct, she should be able to log into her account by clicking on a button and then see the first page where all the modules are provided.
- The user can log out of the account by clicking a button on the first page.
- The user can change the password of her account by visiting a module specifically meant for this functionality.

4.1.3. Functional Requirements

- *Requirement ID : R01.F01.01*
- *Title :* Erroneous Inputs / Validate Inputs
- *Description :* The software should check the correctness/validity of the input details by matching with the concerned records in the concerned tables of the database(s).

If the entered log-in ID does not exist, an error message explicitly mentioning this issue should pop up while staying on the same page. Otherwise, if the login-ID exists but the corresponding password does not match, then also an error message explicitly mentioning this issue should pop up while staying on the same page.

- *Priority : 9*

- *Requirement ID : R01.F01.02*
- *Title :* Incomplete Inputs
- *Description :* The software should ensure before searching the database(s) that both the input fields are filled. If any field is left vacant an error message explicitly mentioning this issue should pop up while staying on the same page.
- *Priority : 8*

- *Requirement ID : R01.F01.03*
- *Title :* Valid Input Details For Registration
- *Description :* The software should minimize the scope of mistakes in entering information of the new *to-be-registered* member. This includes not allowing an option of entering the date of birth in the future. Other possible wrong inputs should also be handled exhaustively. These include
    - ○ the age of a *non-faculty member* cannot certainly be less than *16.*
    - ○ the age of a *faculty member* cannot certainly be less than *25.*
    - ○ the institute ID or roll number cannot certainly be already associated with a member.
- *Priority : 9*

- *Requirement ID : R01.F01.04*
- *Title :* Register With Ease
- *Description :* The *Librarian* must not need to enter a whole bunch of information (like gender, contact number, address, blood group etc) for registering a member. The *institute ID* is enough proof for uniqueness and verification.
  When a new account is created, the *LIS* should automatically allocate the member a unique *LMCN* which serves as the *login identity* for the members. Besides, it

should generate a password that can enable the member to login for the first time.

- *Priority : 9*

- *Requirement ID : R01.F01.05*
- *Title :* De-Registration Of A Member
- *Description :* The Librarian should be asked to reconsider whenever she attempts to de-register a member, by explicitly *confirming the de-registration* of the member through *confirmation dialog box.* According to the *Library's Policy,* a member can not be de-registered if she has any issued books or any pending fines. When a member is successfully de-registered form the sysyem, the *LIS* should delete all the records in the database associated with that member to free memory. This includes entire *history of issued books*, *all notifications*, *all reservations for the issued books* and *all un-issued reserved books in the waiting*.
- *Priority : 8*

- *Requirement ID : R01.F01.06*
- *Title :* Miscellaneous Features
- *Description :* The user should be able to *log out* from and then *login again* into her account.
  Every user must have the privilege to change her password.
  On the home screen, instead of a single login button, it can be *trifurcated* into three buttons -- *Log In As Librarian, Log In As Clerk, Log In As Member.*
- *Priority : 8*

4.2. CATALOGUE QUERY SECTION

4.2.1. Description And Priority

The *LIS* has a query section that is able to search the catalogue based on certain *search criterion* selected by the user. This functionality is *universal*, that is, one need not be logged in, or even have an account for that matter on *LIS*, to be able to use this module. This is present on the home-page, clearly visible.

Overall Priority Quotient : HIGH

4.2.2. Stimulus / Response Sequences

- Any user opens this section by clicking on a button or icon on the home page.

- The user should select *exactly one* of (atleast) three available criteria -- *ISBN, title, author* -- from a drop-menu or any other widget.
- Upon selecting the criterion, the user specifies the desired value or state of the book with respect to that criterion.
- The user performs search by clicking on a button, when details all the relevant books are displayed in a nice and neat format.

4.2.3. Functional Requirements

- *Requirement ID : R01.F02.01*
- *Title :* Limited Transparency
- *Description :* Not all the details concerning a book might be of any interest to the user -- for example *date the stock arrived* and *date of last issue*. These details are not *universally* significant and are more important for the administrative tasks. So this information need not be displayed for any book, to both preserve transparency and avoid clutter. But the *ISBN*, *title, author, language, rack number, number of copies available* must obviously be displayed.
- *Priority : 5*

- *Requirement ID : R01.F02.02*
- *Title :* Lenient Search
- *Description :* The *search* should be mild and tolerant to certain *incorrect* or *partially correct* query inputs. In other words, in case of *search by title*, the *LIS* must not just look for exact matches from the catalogue but must perform a *case-insensitive* matching, along with *substring matching*. Use of *regular expressions* is encouraged. Similar should be the case with *search by author* but for *search by ISBN* however, this functionality is redundant because its value is unique for each book.
- *Priority : 8*

4.3. CLERICAL JOBS

4.3.1. Description And Priority

There are some activities or tasks the interfaces for which are only available to the *clerks* of the Library. One must be *logged-in as a*

clerk to be able to use these interfaces. These interfaces realize the clerical tasks related to -- *issuing a book, returning an issued book, reserving an issued book, paying fine for returning book beyond its due date, adding books to the catalogue, removing books from the catalogue*.

Overall Priority Quotient : HIGH

4.3.2. Stimulus / Response Sequences

- A *clerk* logs into her account by providing the current *login ID* and *password* on the *Log In As Clerk* page.
- The clerk is able to see several modules on the very first page after logging in, with the function of each module clearly mentioned.
- The clerk clicks on any module/icon and is transferred to a page that receives input to perform that task.
- The clerk enters necessary information into the entry boxes and triggers the execution of the task by clicking on a button.
- An *error message* pops up in case the values entered by the clerk are invalid, incorrectly formatted or of incorrect data type.
- A *warning message / confirmation dialog box* appears in case the clerk attempts to *delete a book form the record*.
- The clerk can go back to the first page by clicking on a *go back* button.

4.3.3. Functional Requirements

- *Requirement ID : R01.F03.01*
- *Title :* Reservation Key
- *Description :* It is observed that certain books are so popular that members want to reserve an already issued book rather than issuing another book at its place. The competition in issuing a book as soon as it is returned by the former issuer can be reduced by enabling reservations on issued out books. To enable a fair and malpractice-free system of issuing out the returned book only to the member who reserved the book is to use a *reservation key*, which is sent by the *LIS* to only that member's account who reservesd the book.

  The member must provide the *reservation key* at the clerk's desk to issue a reserved book, in order to confirm the

identity of the reserver. The clerk matches the key with the one that is recorded in the database and issues the book to the *reserver* if the match succeeds.
(It is totally the responsibility of the member to not leak the reservation key to others.)

- *Priority : 8*

- *Requirement ID : R01.F03.02*
- *Title :* Minimal Input
- *Description :* A *clerk* has many tasks at her disposal and it can be very tedious to enter a whole bunch of values to perform som frequent tasks like *issuing, reserving, returning* a book. *Maximize the use of IDs*. Associate every instance of issue and reservation with a suitable *Issue ID* and *Reservation ID* and use these IDs to uniquely identify the records in the database. Maximal use of IDs can reduce the net time spent in entering values drastically. So the respective interfaces should ask for
    - ○ only *ISBN* of a book to remove a book from the catalogue.
    - ○ only *ISBN* of the desired book and *LMCN* of the issuer to issue a book.
    - ○ only *ISBN* of the desired book and *LMCN* of the issuer to reserve an issued book.
    - ○ only *Issue ID* to return a book.
    - ○ only *Issue ID* to pay fine corresponding to the event of late returning of that book.
    - ○ only *Reservation ID* and *Reservation Key* to issue a reserved book.

  For *adding a new book,* however, the clerk must specify all the necessary information that includes -- *ISBN, title, author(s), language, rack number and total number of copies*.
- *Priority : 8*

- *Requirement ID : R01.F03.03*
- *Title :* Allowing Pending Fines
- *Description :* When a former issuer returns a book after the due date, the *LIS* computes the penalty and displays it on the screen (and also sends a notification to the member’s

account regarding the same). In this scenario, the *clerk* will be given an option to navigate to the *Pay Fine Section* in order to clear the fine. Perhaps, the member might not necessarily have cash at the moment. In this case, the clerk can reject the option to navigate to the *Pay Fine Section*; the *LIS* would then record the fine as *pending* in the database. The *Pay Fine Section* can be visited separately any time by the clerk to clear pending fines.

- *Priority : 6*

- *Requirement ID : R01.F03.04*
- *Title :* Constraints Due to *Library's Policy*
- *Description :* There are certain norms that the *Library* must follow while operating through the *LIS*. Appropriate error messages should pop up when this policy is getting violated.
    - ○ A clerk can not remove a book from the catalogue if its at least one copy is issued out.
    - ○ A clerk can not remove a book from the catalogue if there is a reserved book which has not been yet formally issued.

      In other words, a book can be removed if an only if all its copies are available on the rack.
    - ○ Reservation of a book gets expired after 7 days of waiting since it is returned by the former issuer.
    - ○ An under-graduate member cannot issue a book if she already has 2 books issued.
    - ○ A post-graduate member cannot issue a book if she already has 4 books issued.
    - ○ A research scholar member cannot issue a book if she already has 6 books issued.
    - ○ A faculty member cannot issue a book if she already has 10 books issued.
    - ○ A clerk can not *re-insert / override* a book in the catalogue by trying to add another book with the same *ISBN*.
- *Priority : 9*

- *Requirement ID : R01.F03.05*
- *Title :* Error Messages And Confirmation Boxes

- *Description :* Besides the errors arising due to attempted violation of the *Library's Policy*, incorrectly formatted, invalid and non-existing (in the database) input values should also throw an error message. In case of removing a book from the catalogue, the clerk should be asked to reconsider before explicitly *confirming* the action through *confirmation dialog box*.
- *Priority : 7*

4.4. REMINDERS / NOTIFICATIONS

4.4.1. Description And Priority

The member should have an inbox in their accounts where they receive all the *notifications*. These notifications are sent by the *LIS* when certain activity is detected by the *LIS* like issuing a book, returning a book, reserving a book and paying fine. These notifications contain the details of each activity that can range from the characteristics of the book involved in the activity to other attributes associated with the activity itself (like *Issue ID*, *Reservation Key* etc).

Besides this, in the inbox the member also receives *reminders*. These reminders are not triggered by any activity per se but are displayed if the *LIS* finds that the member has some pending duty -- like returning an issued book, collecting a reserved book that is already returned by the former issuer and pending payment of fines.

Overall Priority Quotient : MEDIUM

4.4.2. Stimulus / Response Sequences

- Any member has to be logged-into her account to be able to acces her inbox. The member can open the inbox by clicking a button on the first page after login.
- There is a list of notifications, with reminders on top of them, sorted by the date and time of when the notification was sent.
- Any message can be clicked to view it on a full display.
- The member when clicks the *Go Back* button reaches the first page from where she started.

The various *stimuli* that trigger *sending a notification* to the respective member are -

- The member *issues a book.*
- The member *returns an issued book.*
- The member *reserves an already issued book.*
- The member *clears her fine/penalty of late return.*
- The reserved book is *returned by the former issuer.*
- The *reservation got expired.*

4.4.3. Functional Requirements

- *Requirement ID : R01.F04.01*
- *Title :* Privacy / Message Encryption
- *Description :* All the notifications that are explicitly sent to any member need to be remembered in the database, along with the date and time the notification was sent. (This is not the case with the reminders because they are displayed based on the *current state of the member*. A member need not be reminded that he had a pending fine 3 months ago.)
  The notification messages in the database must not be stored as plain texts but rather as *encrypted messages* using some constant key that is remembered by the *LIS*. These messages are decrypted when they have to be displayed on the screen.
- *Priority : 8*

4.5. OTHER AUTOMATIC FEATURES

4.5.1. Description And Priority

Certain tasks in the *LIS* are automatic in the sense that they are not directly performed by a user but are triggered by some other actions that the user performs. For example, sending notifications; these are not explicitly written by the *clerk* to the member everytime a book is issued/returned/reserved but are *automatically* sent by the *LIS*.

But there are other tasks that are automatic but are not triggered by any activity that any user performs. These tasks are performed *periodically* and are triggered after every certain amount of time. There are certain tasks that the *LIS* should perform periodically to enable maximum knowledge about the current state of the Library. These include -

- ○ periodically checking if a reservation is old enough to be expired.

○ periodically checking if the due date of returning an issued book is crossed.

These tasks are not qualified by any administrator or member and hence must be checked for relevance after every certain period of time. Everytime the software is *run*, the very first "*preprocessing*" task it does is check for the relevance of the above two scenarios. But if the software continues to run for say 24 more hours, the above scenarios might become relevant for the records that were previously irrelevant.

Therefore these tasks should be *scheduled to run on a separate thread* that carries out the action, every say 12 hours, starting when the application is launched.

Overall Priority Quotient : HIGH

4.5.2. Stimulus / Response Sequences

Stimulus : The *LIS* clock completes one cycle (of certain period).

Response :

- Check all the reservations in the database for which the former issuer has already returned the book. If the book was returned before the limiting number of days, *delete the reservation*, *send a notification* to the member regarding the same and make the book *available for issue*.
- Check all the issued books in the database for which the due date of return is the current date on the calendar. In this case, these books become *over-due*.
- Forget the current cycle. Start a new cycle with same period.

4.5.3. Functional Requirements

- *Requirement ID : R01.F05.01*
- *Title :* Periodicity
- *Description :* The periodicity, that is the period or duration after which these tasks have to performed has to be chosen optimally. The system need not perform these tasks after every minute or even hour. Not only will this hamper the responsiveness of the application but also it is not required. The relevance of these scenarios with respect to a particular record can change in at most *24 hours*. Depending on when

the application was launched, these scenarios might possibly need to be checked even after just 5 minutes (if the application was launched near the *midnight*). Hitting a middle ground, a period of *12 hours duration* is fine.

- *Priority : 9*

5. OTHER NON-FUNCTIONAL REQUIREMENTS

5.1. ERROR HANDLING REQUIREMENTS

- *LIS* shall handle expected and non-expected errors in ways that prevent loss in information and long downtime periods.
- The databases shall not be polluted with erroneous inputs that are of incorrect formats or data types.

5.2. PERFORMANCE REQUIREMENTS

- The system should be able to accommodate decently high numbe of accounts and books without any fault.
- Responses to view information shall take no longer than 10 seconds to appear on the screen.

5.3. SECURITY REQUIREMENTS

- The privacy of the users must be ensured.
- All the databases must be secured.
- Non-adminstrative users can just read information but they cannot edit or modify anything except their personal information.
- Every user shall have certain access constraints. No user shall have a completely transparent interface with the underlying databases.

5.4. SOFTWARE QUALITY ATTRIBUTES

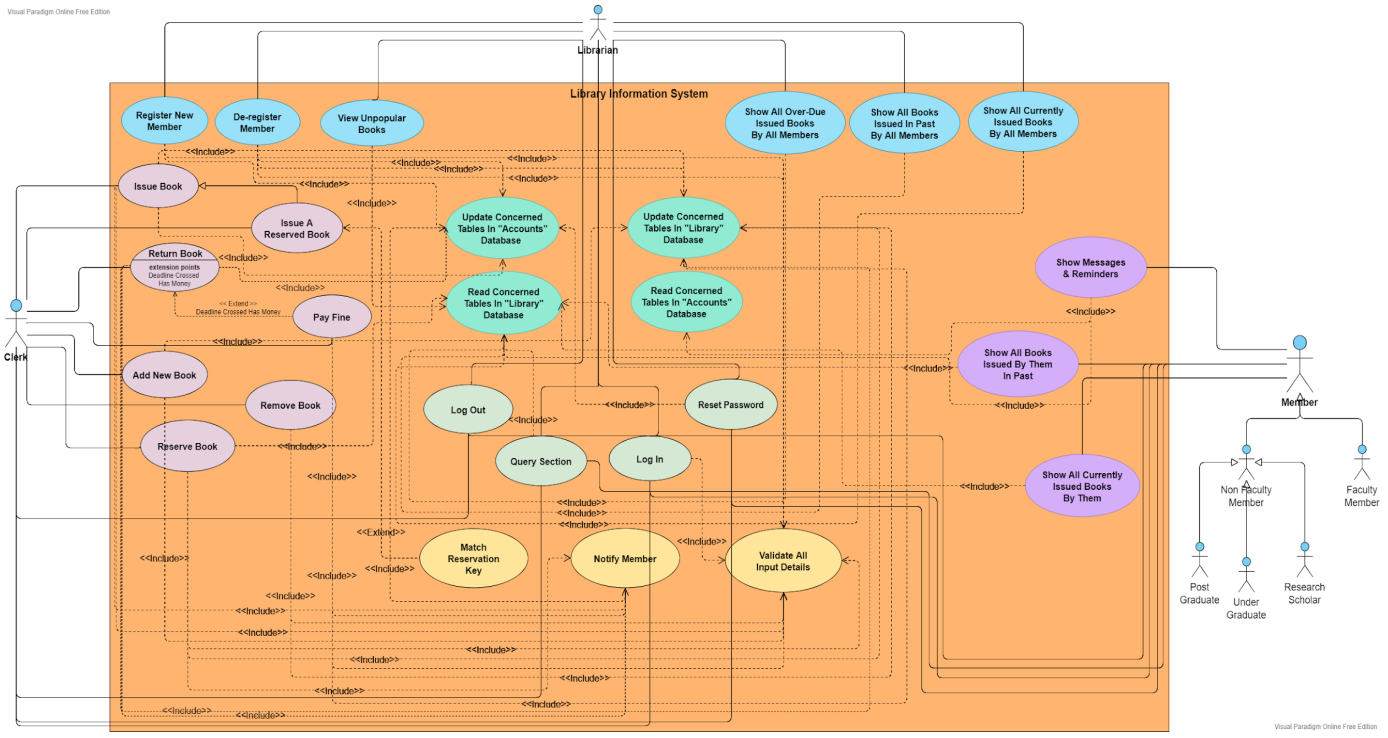
- The product must be easy to understand and hence use.
- The product shall be reliable in case of handling human errors.
- Correctness and robustness of this version are to be preferred over ease of reusing and extending this version into newer versions.

APPENDIX A : GLOSSARY

- *Administrator*: A person who is responsible for the upkeep, configuration, and reliable operation of computer systems; especially multi-user computers, such as servers.
- *Class Diagram*: A type of static structure diagram in UML that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.
- *Database*: An organized collection of data, generally stored and accessed electronically from a computer system.
- *IEEE*: A professional association for electronic engineering and electrical engineering with its corporate office in New York City and its operations centre in Piscataway, New Jersey.
- *ISBN*: A numeric commercial book identifier that is intended to be unique. Publishers purchase ISBNs from an affiliate of the International ISBN Agency. An ISBN is assigned to each separate edition and variation of a publication.
- *Operating System*: A system software that manages computer hardware, software resources, and provides common services for computer programs.
- *Random Access Memory*: A form of computer memory that can be read and changed in any order, typically used to store working data and machine code.
- *Regular Expression*: A sequence of characters that specifies a search pattern.
- *SQL*: A domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system.
- *Use Case Diagram*: A representation of a user's interaction in UML with the system that shows the relationship between the user and the different use cases in which the user is involved.

APPENDIX B : ANALYSIS MODELS

B.2. USE CASE DIAGRAM



B.1. CLASS DIAGRAM