**LIBRARY MANAGEMENT SYSTEM**

SOFTWARE ENGINEERING PROJECT REPORT

**BSC (H) COMPUTER SCIENCE**

**FROM**

**MATA SUNDRI COLLEGE (W)**

****

**(**UNIVERSITY OF DELHI)

**SUBMITTED BY: -**

* **MAHIMA SHARMA (17044570005)**
* **AASHI RATHI (17044570035)**

**ACKNOWLEDGEMENT**

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. We would like to thank all of them.

We are sincerely thankful to our project guide Ms. Ashema Hasti, Assistant professor in Mata sundry College, Delhi University for her guidance and constant supervisions as well as providing necessary information regarding the project. We feel honoured to work under her guidance. This project would not have been possible without her guidance.

Thank you

Mahima Sharma(17044570005)

Aashi Rathi(17044570035)

**CERTIFICATE**

This is to certify that the project titled “Library Management System” is the bona fide work carried out by Mahima Sharma and Aashi Rathi students of B.Sc. (H) Computer Science of Mata Sundri College For Women, Mata Sundri lane, Delhi affiliated to Delhi University, in partial fulfillment of the requirements of Bachelor of Science (H) Computer Science, IV semester under the supervision of Ms. Ashema Hasti, Assistant professor in Mata Sundri college, Delhi University.

Ms. Ashema Hasti

(Mentor)

**ABSTSRACT**

The ‘Library Management system’ undertaken as a project is based on relevant technologies, which is an attempt to automate the existing library. The project enables its user to perform all the operations regarding a library. The project enables the user to make entry of a new book, deleting the record of a book from the library, issuing a book to member, making entry of a new member, deleting the record of a member from the library etc. The process model we have used for our project is Linear Sequential because the requirements are well stated and understood before in hand. In analysis phase we analysed the requirements of what the project will do. We collected the requirements needed to develop the project. Then in the design phase we designed our project according to user satisfaction. We created database to store the details of members, books in tables. We designed DFD diagrams based upon the operations that was carrying in the project. Then cost and effort estimations are calculated and testing and coding processes have been carried out.

Hence in the existing system for LIBRARY MANAGEMENT SYSTEM, the performance evaluation system and the maintenance are done manually. The proposed system will maintain all the information in a standard database and will be able to generate reports when necessary.

**Table of Contents**

1. Problem Statement..................................................................................................

2. Process Model

3. Project Scheduling

4. Timeline Chart

5. Context level diagram

6. DFD Level 1

7. DFD Level 2

8. Data Dictionary

9. Software Requirement Specifications

1. Introduction......................................................................................................................

1.1 Purpose...............................................................................................................

1.2 Scope................................................................................................................

1.3 Definitions................................................................................................

1.4 Overview............................................................................................

2. Project Description

2.1 Product Perspective

2.1.1 System Interfaces

2.1.2 System Specifications

2.1.2.1 H/W Requirement

2.1.2.2 S/W Requirement

2.1.3 Communication Interfaces

2.1.4 Memory Constraints

2.1.5 Operations

2.1.6 Site Adaption Requirements

2.2 Product functions

2.3 User characteristics

2.4 Constraints

2.5 Assumptions and dependencies

2.6 Apportioning of Requirements

3. Specific Requirements

3.1 External interfaces

3.2 Functions

3.3 Performance requirements

3.4 Logical database requirements

3.5 Design constraints

3.5.1 Standard Compliance

3.6 Software system attributes

3.6.1 Reliability

3.6.2 Availability

3.6.2 Security

3.6.3 Maintainability

3.6.4 Portability

10. Software Metrics

11. COCOMO Model

12. Risk Analysis

13. ER Diagram

14. Data Design

15. Component Level Design

16. Use Case Diagram

17. Use Case Description

18. Testing

**1.** **PROBLEM STATEMENT**

A college library management is a project that manages and stores books information electronically according to student’s needs. The system helps both students and library manager to keep a constant track of all the books available in the library. It allows both the admin and the student to search for the desired book. It becomes necessary for colleges to keep a continuous check on the books issued and returned and even calculate fine. This task if carried out manually will be tedious and includes chances of mistakes. These errors are avoided by allowing the system to keep track of information such as issue date, last date to return the book and even fine information and thus there is no need to keep manual track of this information which thereby avoids chances of mistakes.

Thus, this system reduces manual work to a great extent allows smooth flow of library activities by removing chances of errors in the details.

**2. Process Model**

A process model for software engineering is chosen based upon: -

* Nature of the Project.
* Methods and Tools to be used.
* Control and desired deliverable.

The process model, we have chosen to develop this software is a Linear Sequential Model (Waterfall Model). Linear Sequential Model suggests a systematic, sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing and support.

Linear Sequential Model approach has the following phases: -

Software requirements analysis: -

In this, software engineer understands the nature of a program to be built, he must understand the information domain for the software as well as required function, behavior, performance and interface. Requirements for both the system and the software are documented and reviewed with the customer.

Design: -

It has four distinct attributes of a program: data structure, software architecture, interface representations and procedural detail. It is documented and becomes part of the software.

Code generation: -

Design must be translated into a machine-readable form which is done by code generation.

Testing: -

It focuses on the logical internals of the software, ensuring that all the statements have been tested, and on the functional externals; that is conducting test to uncover errors and ensure that defined input will produce actual results.

LINEAR SEQUENTIAL MODEL has been used because of the following reasons: -

It is relatively simple and easier to understand approach as compared to other models. The requirements are well stated and understood before in hand. In this model we have to complete one stage before proceeding to next. So, we have clearly defined stages and well understood milestones. The advancement in program does not need to be checked upon by the customer during the process. So, this model does not create problem. The requirements are fixed and work can proceed to completion in a linear manner. The model provides a structural approach.

**3. PROJECT SCHEDULING**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **WORK TASKS** | **PLANNED START** | **ACTUAL START** | **PLANNED COMLETE** | **ACTUAL COMPLETE** | **ASSIGNED PERSO**N | **EFFORT ALLOCATED** |
| Problem Statement | Jan,w1 | Jan,w1 | Jan,w2 | Jan,w2 | Mahima Sharma, Aashi Rathi | 2 p-w |
| Software process model | Jan,w2 | Jan,w3 | Jan,w3 | Jan,w3 | Aashi Rathi,  Mahima Sharma | 2 p-w |
| Software Requirement Specifications | Jan,w2 | Jan,w2 | Jan, w4 | Jan,w4 | Aashi Rathi,  Mahima Sharma | 2 p-w |
| DFD level0 | Jan,w4 | Jan,w4 | Jan,w4 | Jan,w4 | Aashi Rathi,  Mahima Sharma | 2 p-w |
| DFD level1 | Jan,w3 | Jan,w3 | Feb,w1 | Feb,w1 | Mahima Sharma, Aashi Rathi | 2 p-w |
| DFD level2 | Feb,w2 | Feb,w2 | Feb,w2 | Feb,w2 | Mahima Sharma, Aashi Rathi | 2 p-w |
| Software Metrics | Feb,w3 | Feb,w3 | Feb,w4 | Feb,w4 | Aashi Rathi,  Mahima Sharma | 2 p-w |
| Effort estimation using COCOMO Model | Feb, w4 | March,w1 | March,w1 | March,w2 | Mahima Sharma, Aashi Rathi | 2 p-w |
| Risk Analysis | March, w  2 | March,w3 | March,w4 | April,w1 | Mahima Sharma, Aashi Rathi | 2 p-w |
| ER Diagram | March, w  4 | March,w4 | March,w4 | March,w4 | Aashi Rathi,  Mahima Sharma | 2 p-w |
| Data Design | April,w1 | April,w1 | April,w1 | April,w1 | Mahima Sharma, Aashi Rathi | 2 p-w |
| Use Case Diagram | April,w2 | April,w2 | April,w2 | April,w2 | Mahima Sharma, Aashi Rathi | 2 p-w |
| Use Case Description | April,w2 | April,w2 | April,w2 | April,w2 | Aashi Rathi,  Mahima Sharma | 2 p-w |
| Testing | April,w2 | April,w3 | April,w3 | April,w3 | Mahima Sharma, Aashi Rathi | 2 p-w |

**4. TIMELINE CHART**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **WORK TASKS** | January | | | | February | | | | March | | | | April | | | | |
| W1 | W2 | W3 | W4 | W1 | W2 | W3 | W4 | W1 | W2 | W3 | W4 | W1 | W2 | W3 | W4 |
| Problem Statement |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Software process model |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Software Requirement Specifications |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Context level diagram |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| DFD level1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| DFD level2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Software Metrics |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| COCOMO Model |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Risk Analysis |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ER diagram |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Data Design |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Use Case Diagram |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Use Case Description |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Testing |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**3. SOFTWARE REQUIREMENT SPECIFICATIONS**

**1. INTRODUCTION**

Library Management System is a comprehensive library management solution that is suitable for both large and small libraries. Its flexible design enables Library Management System to be installed in a range of Library organizations, ranging from public libraries, through to academic, joint use and special libraries. This Library Management System Software is capable of handling Books with equal ease and efficiency. This is a Windows-based Library Management System, utilizing the latest advancements in the Information Technology to provide and improve Library Services.

**1.1 Purpose**

The purpose of this project is to develop an application that will automate the whole procedure of a library. The software that would be developed should have facilities like Add / Delete Members, Add / Delete Books, Issue & Return. The application should be secured, as well as with limited access. The main requirement of the project will be the ease of use, besides being the most efficient and effective tool for the purpose. The application should be user friendly. It should be robust and scalable. An automated solution would be very beneficial to the organization, as it would bring structure to the whole process so that it can be traced for any kind of query. Also, an automated solution will lead to optimal utilization of the available resources, reducing duplication of effort, increasing efficiency and minimizing time-delays. Following are the main purpose of computerization:

* To provide services to all the employees for issue, return & search etc. at one place.
* To improve co-ordination in staff.
* To reduce paper filling work.
* To reduce risk of fraud.
* To reduce chances of information leaking.

**1.2 Scope**

For Members: -

* Facility for search of Books based on Access Number, Title, Author, Subject, Keyword.
* Facility for ISSUE / RETURN Books.
* Facility for RENEWAL of Books.

For Library Staff: -

* Automatic installation.
* Simple and intuitive GUI for performing all functions.
* Short-cut keys and point-and-click operation.
* Security features like access control using passwords and login-i.d.
* Automatic calculation of late-fee.
* Facility to ADD / DELETE Members, Library Staff & Books and Maintain an easy record of all these.

**1.3 Acronyms used**

LMS - Library Management System

UI - User Interface

DBMS - Database Management System

**1.4 Overview**

The rest of the document deals with all the main features of this software. It not only describes various functions but also gives details about how these functions are related to each other. Apart from the data flow diagrams, the document also contains cost estimates for developing this system. Various risks associated with the system have also been mentioned along with the ways to mitigate them.

The timeline chart describing how the entire project was scheduled has been attached. At the end a pseudo code for the customer management module” has been provided. A flow graph has been generated corresponding to this module and test cases that were used to test the system have also been mentioned.

**2. PROJECT DESCRIPTION**

**2.1 Product Perspective**

The manual library management system includes following drawbacks:

* The existing system involves a lot of paper work and manual calculation. This has led to inconsistency and inaccuracy in the maintenance of data.
* The data, which is stored on the paper only, may be lost, stolen or destroyed due to natural calamity like fire and water.
* The existing system is sluggish and time-consuming causing inconvenience to library staff.
* Since there are many books related to different scopes thus it would be very difficult to find a specific book, or edit the data of some book.

Hence the library management system is proposed with the following Product perspective:

* The computerization of the library system will reduce a lot of paperwork and hence the load on the library staff.
* It would be easy to search a specific book.
* The machine performs all calculations for fines and all. Hence chances of error are nil.
* The system provides for user-ID validation, hence unauthorized access is prevented.

**2.1.1 System Interfaces**

Software will work on Windows OS. The database used will be an open source database like MySQL and the system will work on JVM.

**2.1.2 System Specification**

**Software requirements**

Operating System: we have chosen windows operating system for its best support and user friendliness.

Database: to save the records of the applicants and their details, SQL database is used.

**Hardware requirements**

LMSuses standard java classes and databases. The database should have backup capabilities.

**2.1.3 Communication interfaces**

Noadditional specific communication interfaces are needed during the operation of LMS.

**2.1.4 Memory constraints**

The system is expected to have a memory of 256 MB and disk space of 500 MB. But it is recommended that the system has memory capacity of 1 GB and disk space of 1GB.

**2.1.5 Operations**

The basic operations of the ‘Library Management System’ are described as follows:

* The staff member first has to register him/herself before using the system.
* The staff member can then login into the system with his/her username and password.
* The staff member can then add, delete or update the book records within the system.
* The staff member can issue books when requesting by the members and then update the book record too.
* The system will generate the fine slip according to the returned date when book is returned by the member.
* The staff member will update the book issue and return records into the book management system.
* The member can search for a specific book by entering book information.
* The staff member can update member records within the system.

**2.1.6 Site Adaption Requirements**

The system will require an application server for the runtime components and a database for storage. The system will run on select popular application servers and use select popular database for data storage.

**2.2 Product Functions**

There are two different users who will be using this product:

* Librarian who will be acting as the administrator.
* Member who can also use product for search operations.

The features that are available to the Librarian are:

* A librarian can issue a book to the member.
* Can view different categories of books available in library.
* Can view the List of books available in each category
* Can take the book returned from students
* Add books and their information of the books to the database
* Edit the information of the existing books.
* Can check the report of the issued Books.
* Can access all the accounts of the students.

The features available to the Member are:

* Can view the different categories of books available in the library
* Can view the List of books available in each category
* Can search for a particular book

**2.3 User characteristics**

User1: Staff- Staff will add, delete or update book records within the system. He/she will issue the books as per requested by the member and will calculate the fine according to returned date of the book. Staff needs to have complete understanding of functionalities and internal processing of the system.

User2: Member- Member will request for book issue and then will return the book. They can search for a specific book. The user does not need to have complete understanding of complex functionalities and internal processing of the system.

**2.4 General Constraints**

Any update regarding the book from the library is to be recorded to have update and correct values and any fine on a member should be notified as soon as possible and should be calculated correctly.

**2.5 Assumptions and Dependencies**

The assumptions are:

* The coding should be error free.
* The system should be user-friendly so that it is easy to use for the users.
* The information of all users, books and libraries must be stored in a database.
* The system should have more storage capacity and provide fast access to the database.

The dependencies are:

* On the basis of listing requirements and specifications the project will be developed.
* The end users should have proper understanding of the product.
* The information of all users must be stored in a database that is accessible by the LMS.
* Any update regarding the book from the library is to be recorded to the database and the data entered should be correct .

**2.6 Apportioning of Requirements**

This is an academic project and hence all the requirements will be completed before the end of the semester. There will be no delay in the delivery of any of the requirements.

**3. SPECIFIC REQUIREMENTS**

**3.1 External Interfaces**

**User Interface**

Various GUI elements like forms, images and standard buttons will be included in the User Interface.

**3.2 Functional Requirements**

* Login:

Description: Staff member will login to the system. The user must be registered in the system before login.

Input: Enter the username and password.

Output: Staff will be able to use the features of software.

Processing: Username and password will be checked by the system. If they are incorrect a message will be displayed.

* Add/Remove books:

Description: The staff can add or remove book by entering details.

Input: Enter the book detail you want to remove or add within the stock.

Output: Confirmation of addition or deletion and update list of books available.

Processing: The details of books must be right in order to add them else there will be problems in future.

* Search:

Description: The users can search a book by entering book details such as author’s name, book name etc.

Input: Enter the details you know about the book.

Output: The list of available books is displayed.

Processing: A message is displayed if the book related to the entered details is not available.

* Issues book:

Description: The staff member checks the availability of book which the member want to get issued.

Input: Enter book code.

Output: Confirmation for book issue or apology for failure in issue.

Processing: If selected book is available then the book will be issued and the record is updated else error will be displayed.

* Return book:

Description: The member wants to return the book.

Input: return the book to the library.

Output: The record will be updated.

Processing: If book is not returned on the time then fine is calculated .

* Fine:

Description: If book is not returned on the time by member then fine is charged on per day basis.

Input: check for the fines.

Output: Details about fines on the book issued by the staff.

Processing: The fine will be calculated, if it crossed the date of return.

**3.3 Performance requirements**

There are no particular extra performance requirements at this point of time.

**3.4 Logical Database Requirements**

Proposed database is intended to store, retrieve, update and manipulate information related to college which include:

* Books availability
* Staff information
* Member details
* Calculation of fines

**3.5 Design Constraints**

Software Constraints: The application shall meet the general standards of web applications.

Hardware Constraints: There is no hardware constraints identified at this point.

Acceptance Constraints:

To validate the system, the developers must complete the following:

1. Demo the working system and any features upon request.

2. Prove that all the significant functional requirements are met.

3. Provide sufficient test cases to show that the system is complete and correct.

The system must be designed to allow web usability. That is, the system must be designed in such a way that will be easy to use and visible on most of the browsers.

**3.5.1 Standard Compliance**

Report Format: All the reports produced for this project are in compliance with the standard templates provided in the class by the advisor.

Naming Conventions: All the documents will be named using the standard naming conventions.

**3.6 Software System Attributes**

**Reliability:**

The application would efficiently store all the information related to the various processes in the system and output the relevant information.

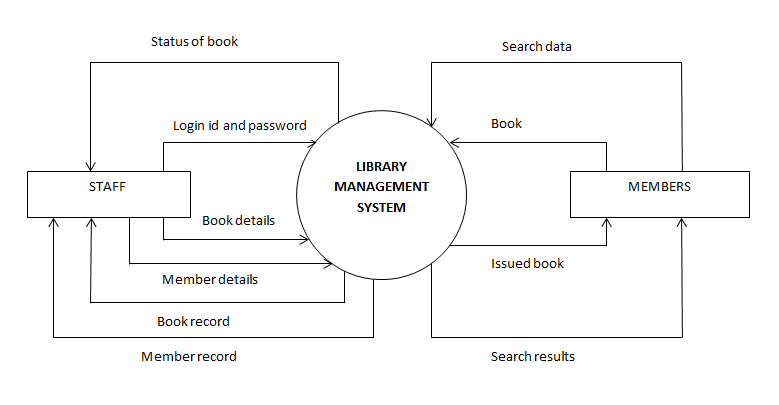
**Availability:**

The application would be available to all the employees of the organizations with an authorized access to the workstations and those who are subject to the authorization permissions.

**Security:** The system would have adequate security checking through the authentication of the users. The reports would only be available to the employees of the library as per their specific requirements.

**Maintainability:** The software should not require any additional maintenance. If any errors occur, the user should be able to login again with his credentials. The system shall be flexible enough to add new modules and upgrade the existing modules.

**4. CONTEXT LEVEL DIAGRAM**

****

**5. DFD LEVEL 1**

Fetch

Book record

Fetch

Book details

Invalid user message

Username and password

Update

STAFF

Book details

Book

Fetch

Fetch

Book code

Member record

MEMBER

Update

Book

Fetch

Fetch

Update

Borrower’s record

Returned book data

Fetch

Return slip

Update

Book details

Return record

Fetch

Search results

Fetch

Update

Staff record

Update

Update

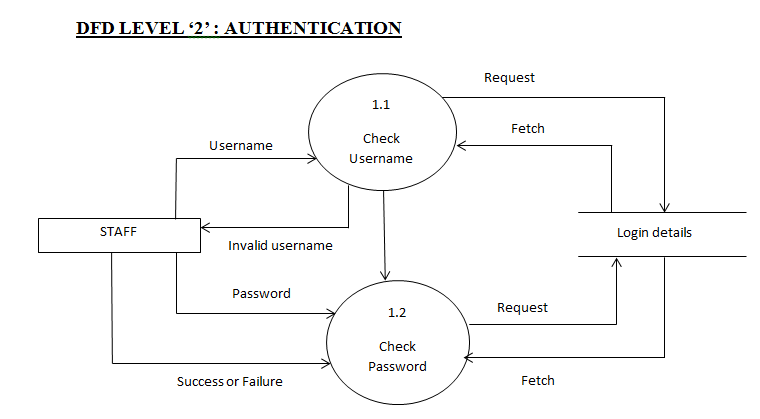
Fetch

Member details

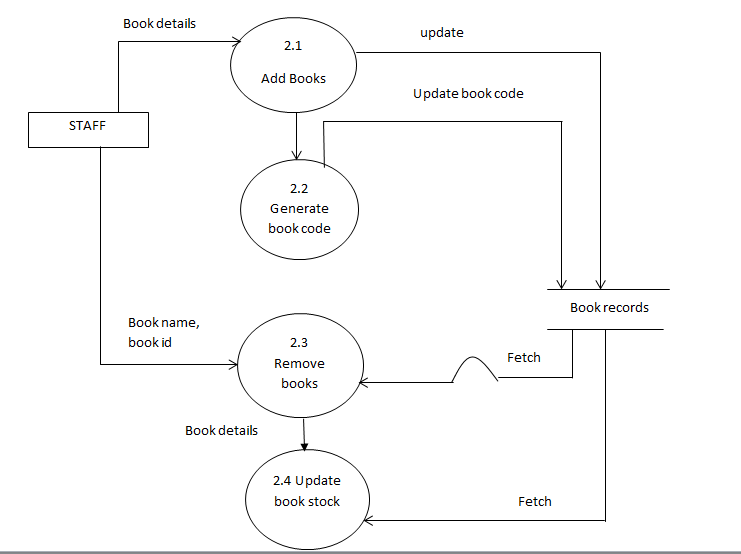
Member record

Fetch

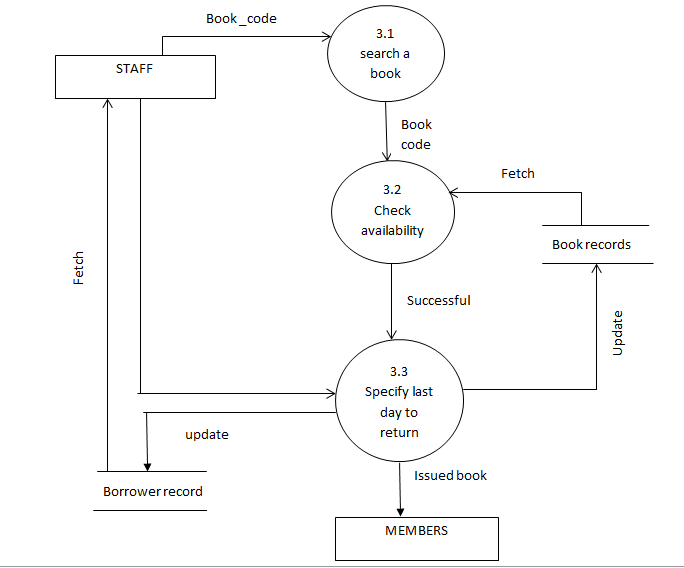
**6.DFD LEVEL-2**

****

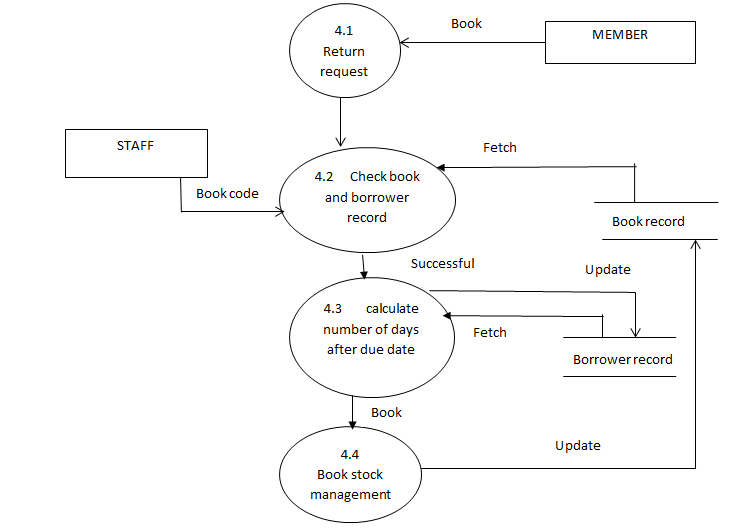
**DFD LEVEL ‘2’ : BOOK STOCK MANAGEMENT**

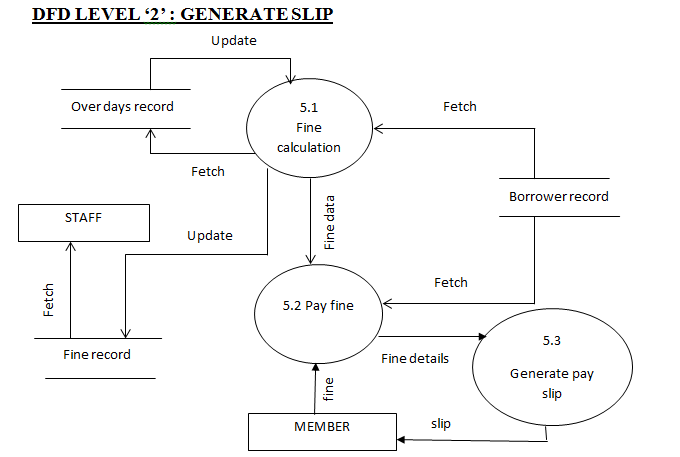
****

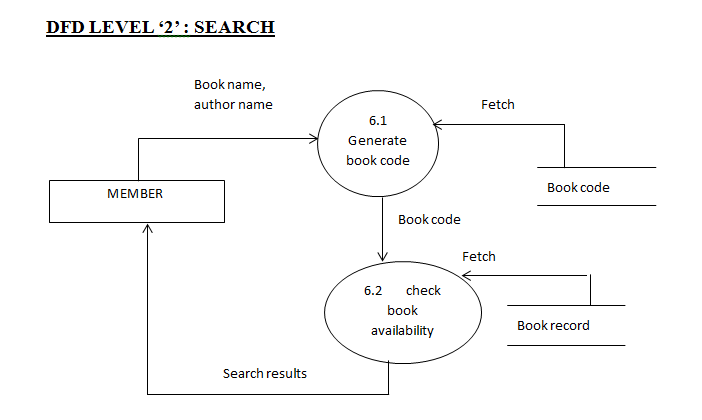
**DFD LEVEL ‘2’ : ISSUE A BOOK**

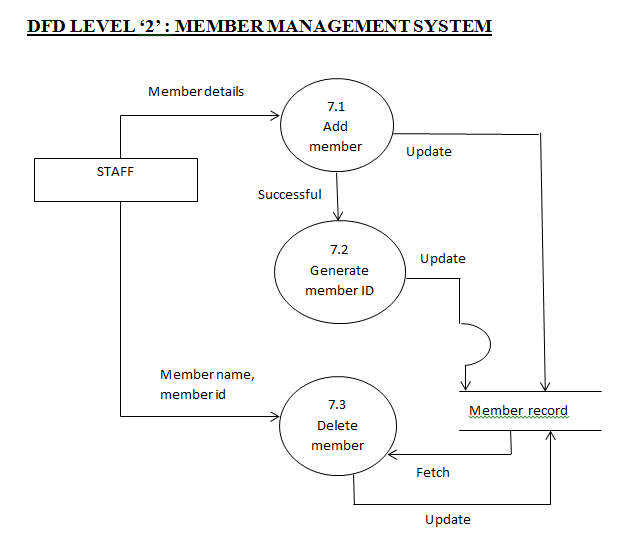
****

**DFD LEVEL ‘2’ : RETURN A BOOK**

****

****

****

****

**7. DATA DICTIONARY**

Legal\_character = [a-z|A-Z]

Legal digit = [0-9]

Special\_character = [@|$|#]

Book\_Details = book \_code + book\_name + price +author\_name + publisher + edition + copies

Book\_code = (legal character +legal digit) \*

Book\_name = (legal character) \*

Price = (legal digit) \*

Author\_name = (legal character) \*

Publisher = (legal character) \*

Edition = (legal digit) \*

Copies = (legal digit) \*

Member\_details = name+member\_id

Name=first\_name + (middle name) + last name

first\_name = (legal character) \*

member\_id= (legal digits) \*

Staff\_details = name + staff\_id

staff\_id = (legal digit) \*

login=username +password

username = (legal character + legal digit) \*

password = (legal character + legal digit) \*

**SCREENS**

1. LOGIN



No. of External Inputs: 3

No. of External Outputs: 1

No. of External Inquiries: 0

No. of Internal Logical File: 1

No. of External Interface files: 0

1. Insert book



No. of External Inputs: 10

No. of External Outputs: 1

No. of External Inquiries: 0

No. of Internal Logical File: 1

No. of External Interface files: 0

1. Remove member



No. of External Inputs: 2

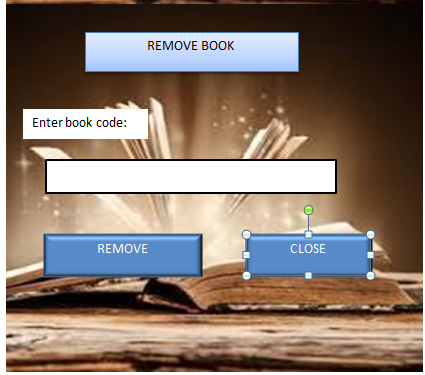
No. of External Outputs: 1

No. of External Inquiries: 0

No. of Internal Logical File: 1

No. of External Interface files: 0

1. Remove book



No. of External Inputs: 2

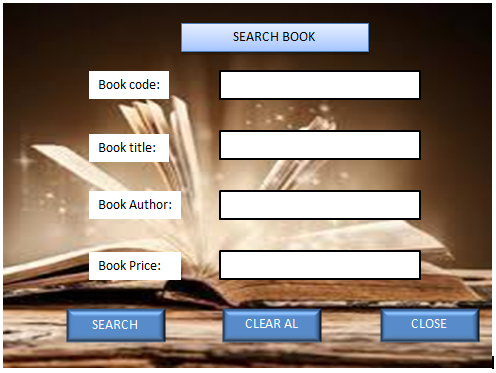
No. of External Outputs: 1

No. of External Inquiries: 0

No. of Internal Logical File: 1

No. of External Interface files: 0

1. Search books



No. of External Inputs: 5

No. of External Outputs: 2

No. of External Inquiries: 1

No. of Internal Logical File: 1

No. of External Interface files: 0

1. Return book



No. of External Inputs: 2

No. of External Outputs: 1

No. of External Inquiries: 0

No. of Internal Logical File: 1

No. of External Interface files: 0

1. Create new user



No. of External Inputs: 6

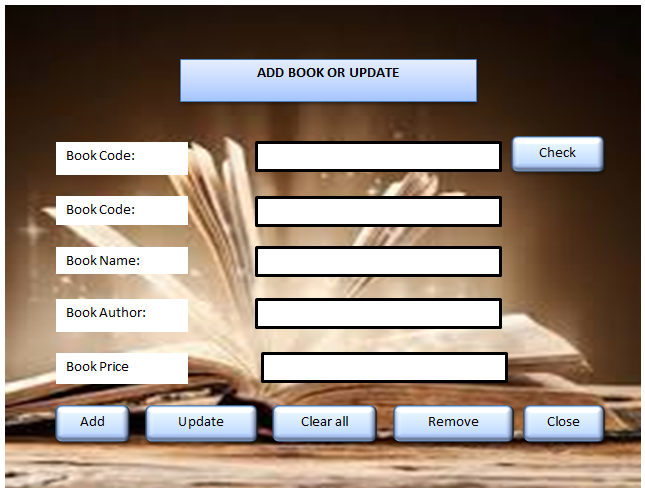
No. of External Outputs: 1

No. of External Inquiries: 0

No. of Internal Logical File: 1

No. of External Interface files: 0

1. Add book or update



No. of External Inputs: 6

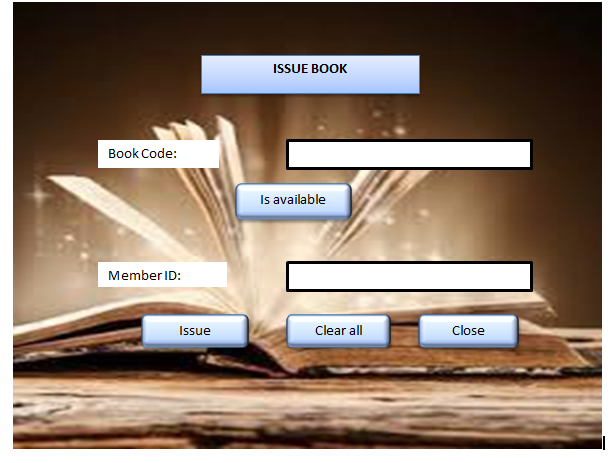
No. of External Outputs: 1

No. of External Inquiries: 1

No. of Internal Logical File: 1

No. of External Interface files: 0

9.Issue book



No. of External Inputs: 3

No. of External Outputs: 1

No. of External Inquiries: 1

No. of Internal Logical File: 1

No. of External Interface files: 0

**8.SOFTWARE METRICS**

This metrics describe the project characteristics and execution. Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity.

**FUNCTION BASED METRICS:** The function point metric can be used effectively as a means for measuring the functionality delivered by the system. Using historical data FP metric can be used to: 1. Estimate the cost or effort required to design, code and test the software.

2. Predict the number of errors that will be encountered during testing.

3.Forecast the number of components and/or the number of projected source lines in the implemented system.

Function points are computed by the following relationship:

**FP= count total \* [0.65 + 0.05\*(∑fi)**

Where count total is the sum of all FP entries obtained from the following table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INFORMATION DOMAIN VALUE** | **COUNT** | **Weighting factor**  **Simple Average Complex** | | | |
| External inputs  (EIs) | 39 | 3 | 4 | 6 | 117 |
| External outputs  (Eos) | 10 | 4 | 5 | 7 | 40 |
| External inquiries  (EQs) | 3 | 3 | 4 | 6 | 9 |
| Internal logical files (ILFs) | 9 | 7 | 10 | 15 | 63 |
| External interface files (EIFs) | 0 | 5 | 7 | 10 | 0 |
| **COUNT TOTAL** |  |  |  |  | 229 |

**Assuming all are of simple complexity.**

**So, count total= UFP=229**

In the table, Information domain values are defined in the following manner:

1. **Number of external inputs (EIs):** Each external input originates for a user or is transmitted from another application and provide distinct application oriented data or control information. Inputs are often used to update internal logical files (ILFs). Inputs should be distinguished from inquiries, which are counted separately.
2. **Number of external outputs (EOs):** Each external output is derived data within the application that provides information to the user. External output refers to reports, screens, error messages etc.
3. **Number of external inquiries (EQs):** An external inquiry is defined as an online input that results in the generation of some imediate software response in the form of an online output.
4. **Number of internal logical files (ILFs):** Each ILF is a logical grouping of data that resides within the application’s boundary and is maintained via external input.
5. **Number of external interface files (EIFs):** EachEIF is a logical grouping of data that resides external to the application but provides information that maybe of used to the application.

The fi (i= 1 to 14) are value adjustment factors (VAF) based on responses to the following questions:

|  |  |  |
| --- | --- | --- |
| 1. | Does the system require reliable backup and recovery? | 4 |
| 2. | Are specialized data communications required to transfer information to or from the application? | 1 |
| 3. | Are there distributed processing functions? | 3 |
| 4. | Is performance critical? | 2 |
| 5. | Will the system run in an existing, heavily utilized operational environment? | 3 |
| 6. | Does the system require online data entry? | 5 |
| 7. | Does the online data entry require the input transaction to be built over multiple screens or operations? | 5 |
| 8. | Are the ILFs updated online? | 5 |
| 9. | Are the inputs, outputs, files, or inquiries complex? | 1 |
| 10. | Is the internal processing complex? | 1 |
| 11. | Is the code designed to be reusable? | 2 |
| 12. | Are conversion and installation included in the design? | 0 |
| 13. | Is the system designed for multiple installations in different organizations? | 4 |
| 14. | Is the application designed to facilitate change and ease of use by the user? | 2 |

Σfi=38

**FP= count total \* [0.65 + 0.05\*(∑fi)**

FP= 229\*(0.65+ (0.01\*38))

FP=235.87

**9. COCOMO MODEL**

COnstructive COst MOdel (COCOMO II) is a more comprehensive estimation model. COCOMO II is actually a hierarchy of estimation models that address the following areas:

* Application composition model - Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.
* Early design stage model - Used once requirements have been stabilized and basic software architecture has been established.
* Post-architecture-stage model - Used during the construction of the software.

The COCOMO II model require sizing information. Three different sizing options are available as part of model hierarchy:

1. Object points
2. Function points
3. Lines of source code

Like function points the object point is an indirect software measure that is computed using counts of the number of screens, reports and components likely to be required to build the application. Each object instance is classified into one of the three complexity levels (simple, medium or difficult) as shown in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Object type** | Weighing factotr | | |
| **Simple** | **Medium** | **Difficult** |
| Screen | 1 | 2 | 3 |
| Report | 2 | 5 | 8 |
| 3GL Component |  |  | 10 |

The object count is determined by multiplying the total number of object instances by weighing factor in the figure and summing to obtain a total object point count. When component based development or general software reuse is to be applied, the percent of reuse is estimated (%reuse) and the object point count is adjusted.

**NOP = (object points) \* [(100 - % reuse)/100]**

where NOP is defined as new object points.

To derive an estimate of effort based on the computed NOP value, a “productivity rate” must be derived.

**PROD = NOP / person-month**

Productivity rate for different level of developer experience and development environment maturity:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Developer’s experience/capability** | Very low | Low | Nominal | High | Very high |
| **Environment maturity/capability** | Very low | Low | Nominal | High | Very high |
| **PROD** | 4 | 7 | 13 | 25 | 50 |

Once the productivity rate has been determined, an estimate of project effort is computed using –

**Estimated effort = NOP / PROD**

**COCOMO estimation for our project:**

1. Number of screens =9
2. Number of reports =4

( book stock report, fine report, issued books report, returned books report)

1. Number of 3GL components used =0
2. Person-month=NOP/ PROD

Person-month =4.25

**5.)Object Point Count=( Screen\*weighing factor + reports\*weighing factor + 3GL Componenets\*weighing factor)**

**Object Point Count =17**

**6.)NOP** = OPC x [1- (%reuse) / 100]

= Taking reusability to be 0

= NOP = OPC = 17

**7.) Prod =4**

**8.)** **Estimated Effort =**NOP/PROD

**Estimated Effort =4.25**

**10. RISK ANALYSIS**

Risk always involves two characteristics -:

* Uncertainty - The risk may or may not happen; that is, there are no 100 percent probable risks.
* Loss - If the risk becomes a reality, unwanted consequences or losses will occur.

When risks are analysed, it is important to quantify the level of uncertainty and the degree of loss associated with each risk.

Different categories of risks are considered –:

• **Project risks** –The risks that threaten the project plan. That is, if project risks become real, it is likely that the project schedule will slip and that costs will increase.

• **Technical risks** – The risks that threaten the quality and timeliness of the software to be produced. If a technical risk becomes a reality, implementation may become difficult or impossible.

• **Business risks** – The risks that threaten the viability of the software to be built and often jeopardize the project or the product.

Another general categorization of risks has been proposed by Charette –

• **Known Risks** - Those risks that can be uncovered after careful evaluation of the project plan, the business and technical environment in which the project is being developed, and other reliable information sources.

• **Predictable Risks** - Risks that are extrapolated from past project experience.

• **Unpredictable Risks** – Risks that are the joker in the deck. They can and do occur, but they are extremely difficult to identify in advance.

The risk components are defined in the following manner –

• **Performance Risk** – The degree of uncertainty that the product will meet its requirements and be fit for the intended use.

• **Support Risk** – The degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.

• **Schedule Risk** – The degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.

* **Cost Risk** – The degree of uncertainty that the product budget will be maintained.

**RISK IDENTIFICATION**

Checklist for risk identification –

• Product size - Risks associated with the overall size of the software to be built or modified.

• Business impact - Risks associated with constraints imposed by management or the marketplace.

• Stakeholder characteristics - Risks associated with the sophistication of the stakeholders and the developer’s ability to communicate with stakeholders in a timely manner.

• Process definition - Risks associated with the degree to which the software process has been defined and is followed by the development organization.

• Development environment - Risks associated with the availability and quality of the tools to be used to build the product.

• Technology to be built - Risks associated with the complexity of the system to be built and the “newness” of the technology that is packaged by the system.

• Staff size and experience - Risks associated with the overall technical and project experience of the software engineers who will do the work.

**ASSESSING OVERALL PROJECT RISKS**

1. Have top software and customer managers formally committed to support the project?

YES

2. Are end users enthusiastically committed to the project and the system product to be built?

YES

3. Are requirements fully understood by the software engineering team and its customers? YES

4. Have customers been involved fully in the definition of requirements?

YES

5. Do end users have realistic expectations?

YES

6. Is the project scope stable?

YES

7. Does the software engineering team have the right mix of skills?

YES

8. Are project requirements stable?

YES

9. Does the project team have experience with the technology to be implemented?

YES

10. Is the number of people on the project team adequate to do the job?

YES

11. Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?

YES

**RISK TABLE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Risks identified** | **Risk type** | **Probability** | **Impact** |
| Maintenance Problem | Technical Risk | 0.6 | Negligible |
| High work load | Project Risk | 0.3 | Critical |
| Design and implementation | Technical Risk | 0.95 | Marginal |
| Hardware failure | Technical Risk | 0.4 | Critical |
| Inexperienced Instructor | Project Risk | 0.8 | Critical |

**RMM Plan**

The goal of the risk mitigation, monitoring and management plan is to identify as many potential risks as possible. The project will then be analysed to determine any project-specific risks.

* When all risks have been identified, they will then be evaluated to determine their probability of occurrence. Plans will then be made to avoid each risk, to track each risk to determine if it is more or less likely to occur, and to plan for those risks should they occur.
* It is the organization’s responsibility to perform risk mitigation, monitoring, and management in order to produce a quality product.
* The quicker the risks can be identified and avoided, the smaller the chances of having to face that particular risk’s consequence. The fewer consequences suffered as a result of good RMMM plan, the better the product, and the smoother the development process.

**RISK CONTROL**

|  |  |
| --- | --- |
| **Risk identified** | **Risk control Plan** |
| Maintenance Problem | Proper use of the resources so as to avoid any hardware failures and taking timely backups. |
| High work load | Divide the work into smaller tasks. |
| Design and implementation | Keep it simple for implementation. |
| Hardware failure | Invest in good quality hardware components. |
| Inexperienced Instructor | Alert client of potentially difficulties and the possibilities of the delays, investigate buying-in components |

**11. USE CASE DIAGRAM**

Library Staff

Member

**12. USECASE DESCRIPTION**

**LOGIN**

**1. Brief Description:**

This use case describes how an actor login into the ‘Library Management System’.

**2. Actors:**

Library Staff

**3. Flow of events:**

**a.** Basic Flow:

This use case starts when the actor wishes to log into the ‘Library Management System’.

The system requests the actor to enter the username and password.

The actor enters his/her username and password.

The system validates the username and password and then the actor is logged into the system.

**b.** Alternative flows:

If in the basic flow, the actor enters an invalid name or password, the system displays an error message. The actor can use to either return to the beginning of the basic flow or cancel the login at the point where use case ends.

**4. Special requirements:**

None

**5. Pre-conditions:**

The actor must have an account created in the system prior to executing the use cases.

**6. Post conditions:**

If the use case was successful, the actor is logged in to the system. If not, the system state is unchanged.

**7. Extension points:**

None.

**ADD/DELETE MEMBER**

**1. Brief Description:**

This use case describes how an actor adds or deletes the user’s record in the system.

**2. Actors:**

Library Staff

**3. Flow of events:**

**a.** Basic Flow:

This use case starts when the actor has to add or delete users/members within the ‘Library Management System’.

The actor fills the details of the member.

The member is then added to the system now the member got registered for further uses.

The actor can delete a member by filling the appropriate details.

**b.** Alternative flows:

If in the basic flow the actor fills wrong details while deleting a member then the system displays an error message. The actor can either return to the beginning of the basic flow or can cancel the deletion of the member from the system.

**4. Special requirements:**

None.

**5. Pre-conditions:**

The member must added into the system before the actor deletes him/her.

**6. Post conditions:**

Record for a member has been added.

**7. Extension points:** None.

**ISSUE BOOK**

**1. Brief Description:**

This use case describes how the staff issues book when requested by the member.

**2. Actors:**

Library Staff

Members

**3. Flow of events:**

**a.** Basic Flow:

If a member wants to borrow a book it is important that the staff should login to the system.

If login is successful the staff should enter the member id to be searched.

If the member search is successful the staff should enter the book id.

If the book is available then it can be borrowed.

**b.** Alternative flows:

If the login fails then the staff should re- register themselves.

If the member search is unsuccessful then the staff should re-register the student.

If the book search is unsuccessful and book data is not found then the staff must enter the book in requisition report.

**4. Special requirements:**

None.

**5. Pre-conditions:**

To borrow any books it is important that the member is registered and the book to be borrowed is available.

**6. Post conditions:**

Book is reserved for the member.

**7. Extension points:**

None.

**RETURN BOOK**

**1. Brief Description:**

This use case describes how the return book procedure carried out when requested by the member.

**2. Actors:**

Library Staff

Members

**3. Flow of events:**

**a.** Basic Flow:

Member gives the book to be returned to the staff member.

Staff member checks if the book is returned on time.

Staff member update the book records.

**b.** Alternative flows:

In the basic flow the staff member checks if the book is returned on time if it is not on the time then he/she generates slip of calculated fine.

The member submits the fine.

**4. Special requirements:**

None.

**5. Pre-conditions:**

Book/s must have been issued to the member.

**6. Post conditions:**

Report on fine is updated.

**7. Extension points:**

None.

**MAINTAIN MEMBER RECORD**

**1. Brief Description:**

This use case describes how the actor maintains the record of members which includes edit or view the member’s data.

**2. Actors:**

Library Staff

**3. Flow of events:**

**a.** Basic Flow:

Staff member login to the system and selects the menu option to change the data of specific member.

Enter the name of categories that he/she want to change.

The system save the change in membership record and update previous record.

To view the record of a member the actor selects from the menu option.

LMS presents the record of members.

**b.** Alternative Flows

If the password is incorrect then a message is printed on the screen and staff member is returned to the beginning.

If the name of changed to be category is not among the existing categories a message is printed on the screen and the actor is returned to the menu screen.

**4. Special requirements:**

None.

**5. Pre-conditions:**

Password or username must have registered.

**6. Post conditions:**

Member record is updated in database.

**7. Extension points:**

None.

**SEARCH A BOOK**

**1. Brief Description:**

This use case describes how the actor can search for a particular book.

**2. Actors:**

Member

Library staff

**3. Flow of events:**

**a.** Basic Flow:

Member or staff enters the book name or ISBN or author name and presses search

If the search is successful then that book is displayed on the screen.

**4. Special requirements:**

None.

**5. Pre-conditions:**

The book to be searched should be registered into the database of the ‘Library Management System’.

**6. Post conditions:**

List of books according to search data is appeared on the screen.

**7. Extension points:**

None.

**ADD/DELETE BOOK**

**1. Brief Description:**

This use case describes how the actor adds or removes the books.

**2. Actors:**

Library staff

**3. Flow of events:**

**a.** Basic flow:

The staff  login to the system.

If login is successful then to add a book the staff must search for the book.

If the book is not found then it is checked in the requisition list.

If the book is not currently available and is part of the requisition list it can be added to the book database.

To remove a book it is again searched in the library system.

If it is found it should be checked in borrower record.

If it is not in the borrowed list it can be removed.

**4. Special requirements:**

None.

**5. Pre-conditions:**

To add any book that should be part of requisition list and to delete the book must be part of library.

**6. Post conditions:**

List of books is updated according to added or removed books.

**7. Extension points:**

None.

**MAINTAIN BOOK RECORD**

**1. Brief Description:**

This use case describes how the actor maintains the book record in the database of the system which includes added, issued or returned books record

**2. Actors:**

Library staff

**3. Flow of events:**

**a.** Basic flow:

1. The staff selects the menu option to add record of different books. He/ She enters the name , Author’s name, edition etc. or he/she can use a barcode.

2. To enter the record of issued book the staff member goes to menu option.

LMS presents the menu for maintaining the issued books record which contains two options a. Add issued books record b. Edit issued books record.

LMS waits for user input.

3. Staff selects the menu option to enter in the returned books record.

LMS presents the menu for maintaining the issued books record which contains two options a. Add returned books record b. Edit returned books record.

LMS waits for user input.

**b.** Alternative Flows

If the password is incorrect the actor goes back to the beginning.

**5. Pre-conditions:**

Password or username must have registered.

**6. Post conditions:**

Books records are updated in the database

**7. Extension points:**

None.

**13.ER DIAGRAM**

**TEACHERS**

Add.

Sex

F.name

M.name

Yr. of join

D.O.B

Name

Yr. of join

Course

**STUDENTS**

**MEMBERS**

Returns

Add

Search

del

Issue

**BOOKS**

**STAFF**

Remove

Add

**14. Data Design**













**15.TESTING**

**White box testing:**

White box testing, sometimes called glass box testing, is a test case design philosophy that uses the control structure described as a part of component level design to derive test cases.

Using white box testing technique we can derive test cases that:

1. Guarantee that all the independent paths within a module have been exercised at least once.

2. Exercise all logical decisions on their true and false sides.

3. Execute all loops at their boundaries and within their operational bounds and

4. Exercise internal data structures to ensure their validity.

**Basis – Path Testing**

It is a white-box testing technique which enables the test case designer to derive a logical complexity measure as a guide for defining a basis set of execution paths. Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least once during testing.

We are doing white box testing for Issue\_book screen:

**Module**

1)Issue\_Book/

2)Enter choice

(a)select book according to book name

(b)select book according to author name

3)If (Choice=’a’)

4)then Enter the book name

5)If book name found and available

6)then issue the book

7)Else not issue the book

8)End if

9)Else

10)Enter the author name

11)If author’s name found and available

12)then issue the book

13)Else not issue the book

14)End if

15)End if

16)End Issue\_Book

**FLOW GRAPH**

**Cyclomatic Complexity:**

1. Number of regions=4
2. V(G)=Number of edges-Number of nodes+2

18-16+2=4

1. V(G)=Number of predicate nodes+1

=3+1=4

**Independent Paths**:

1🡪2🡪3🡪4🡪5🡪6🡪8🡪15🡪16

1🡪2🡪9🡪10🡪12🡪14🡪15🡪16

1🡪2🡪3🡪4🡪7🡪8🡪15🡪16

1🡪2🡪9🡪10🡪11🡪13🡪14🡪15🡪16

**BIBLIOGRAPHY**

1. R.S. Pressman, software engineering: A Practitioner’s Approach (7th Edition), McGraw-Hill, 2009.
2. P. Jalote, An Integrated Approach to Software Engineering(2nd Edition ), Narosa Publishing House, 2003.