Documentation

LIBRARY MANAGEMENT SYSTEM

Submitted By:

MANAS KUMAR GHOSH

(IS/A1/A5700)

INTRODUCTION

“Library Management System” is an online web based system that application which refers to library systems which are generally small or medium in size. It is used by librarian to manage the library using a computerized system where he/she can add new books, videos and Page sources.

Books and student maintenance modules are also included in this system which would keep track of the students using the library and also a detailed description about the books a library contains.

With this computerized system there will be no loss of book record or member record which generally happens when a non-computerized system is used.

All these modules are able to help librarian to manage the library with more convenience and in a more efficient way as compared to library systems which are not computerized.
We will build ***Library Management System Project in Java and MySQL*** with source code. This project is great for those at an intermediate level in Java who want to advance their coding skills. In this project, the users will be able to perform the following functionalities ***Login***, ***View Categories***, ***Book details***, ***Author details***, ***Book issue, and Book return***. Let’s get started!

PROJECT AIMS AND OBJECTIVES

The project aims and objectives that will be achieved after completion of this project

The aims and objectives are as follows:

- Online book reading.
- A search column to search availability of books.
- Facility to download required book.
- Video tutorial for students.
- An Admin login page where admin can add books, videos or page sources
- Open link for Learning Website

TOOLS/PLATFORM, HARDWARE AND SOFTWARE REQUIREMENT SPECIFICATION

TOOLS/PLATFORM

Library Management System Project is developed using **JAVA** and **MYSQL** for storing data.

HARDWARE REQUIREMENT AND SPECIFICATION

The hardware's which was required for developing the application is as under:

Altogether a Computer with following components**:**

- Intel Dual Core 2.4 GHz equivalent or above Processor
- 4 GB Memory (RAM)
- Hard Disk 500 GB
- Color Monitor
- Keyboard
- Mouse
- Internet Connection

SOFTWARE TOOLS USED

The whole Project is divided in two parts the front end and the back end

Front end

The front end is designed using of HTML,CSS, Javascript.

Back end

The back end is designed using of JAVA and MYSQL

TABLE DESIGN

VARIOUS TABELS TO MAINTAIN INFORMATION

➢ **Library Table from Database**

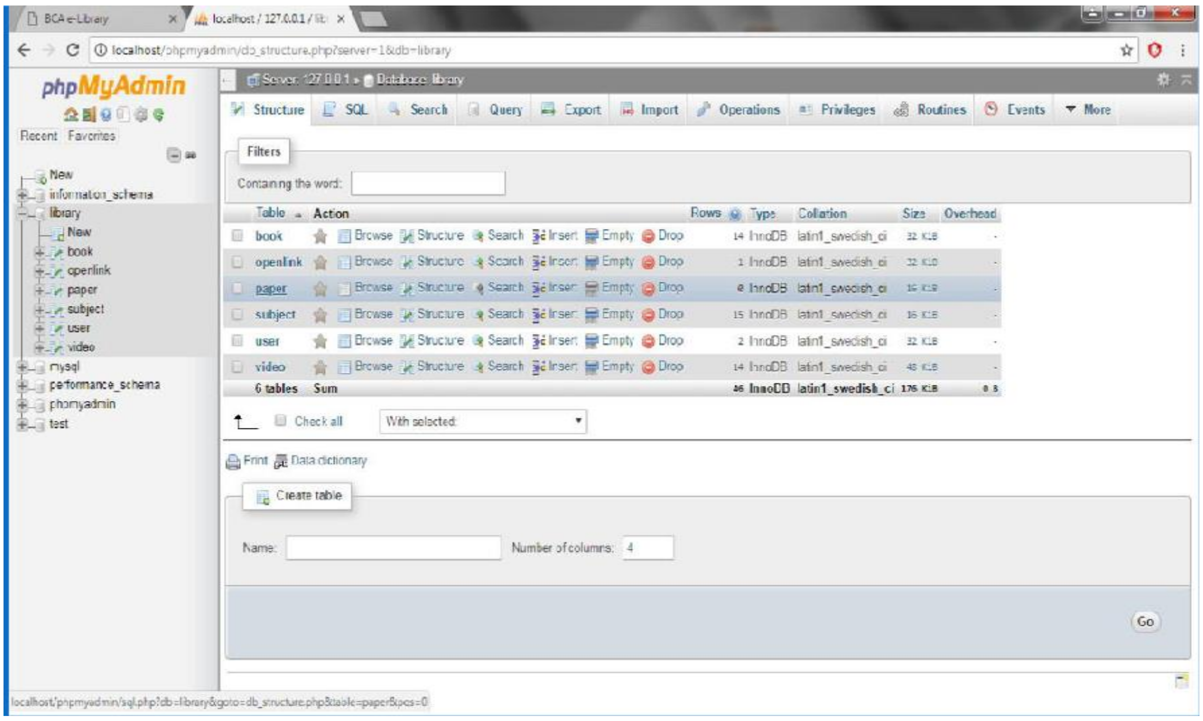
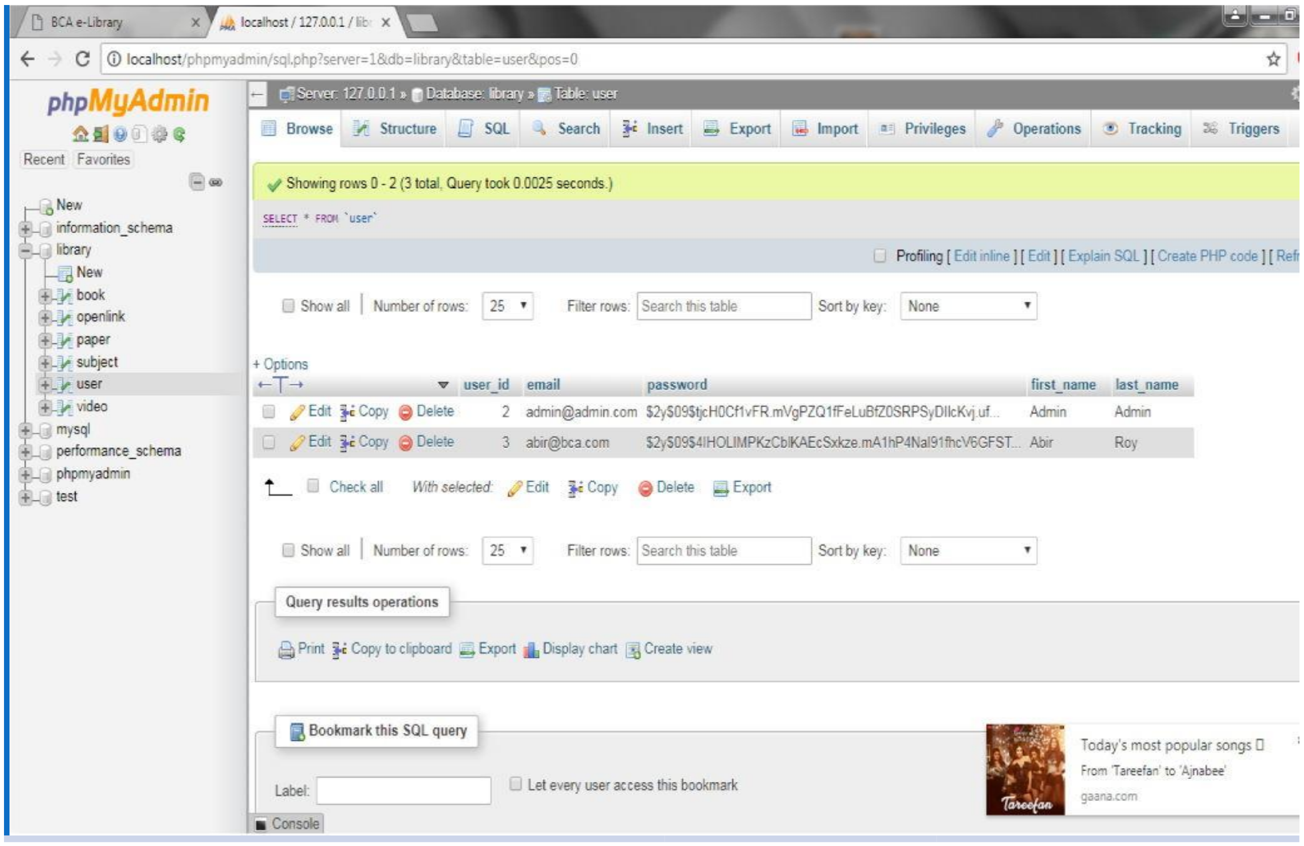


Table	Action	Rows	Type	Collation	Size	Overhead
book	Browse Structure Search Insert Empty Drop	14	InnoDB	latin1_swedish_ci	32 KiB	-
openlink	Browse Structure Search Insert Empty Drop	1	InnoDB	latin1_swedish_ci	32 KiB	-
paper	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	16 KiB	-
subject	Browse Structure Search Insert Empty Drop	15	InnoDB	latin1_swedish_ci	16 KiB	-
user	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	32 KiB	-
video	Browse Structure Search Insert Empty Drop	14	InnoDB	latin1_swedish_ci	48 KiB	-
6 tables	Sum	46	InnoDB	latin1_swedish_ci	176 KiB	0 B

➢ Admin Table from Database



localhost/phpmyadmin/sql.php?server=1&db=library&table=user&pos=0

Server: 127.0.0.1 » Database: library » Table: user

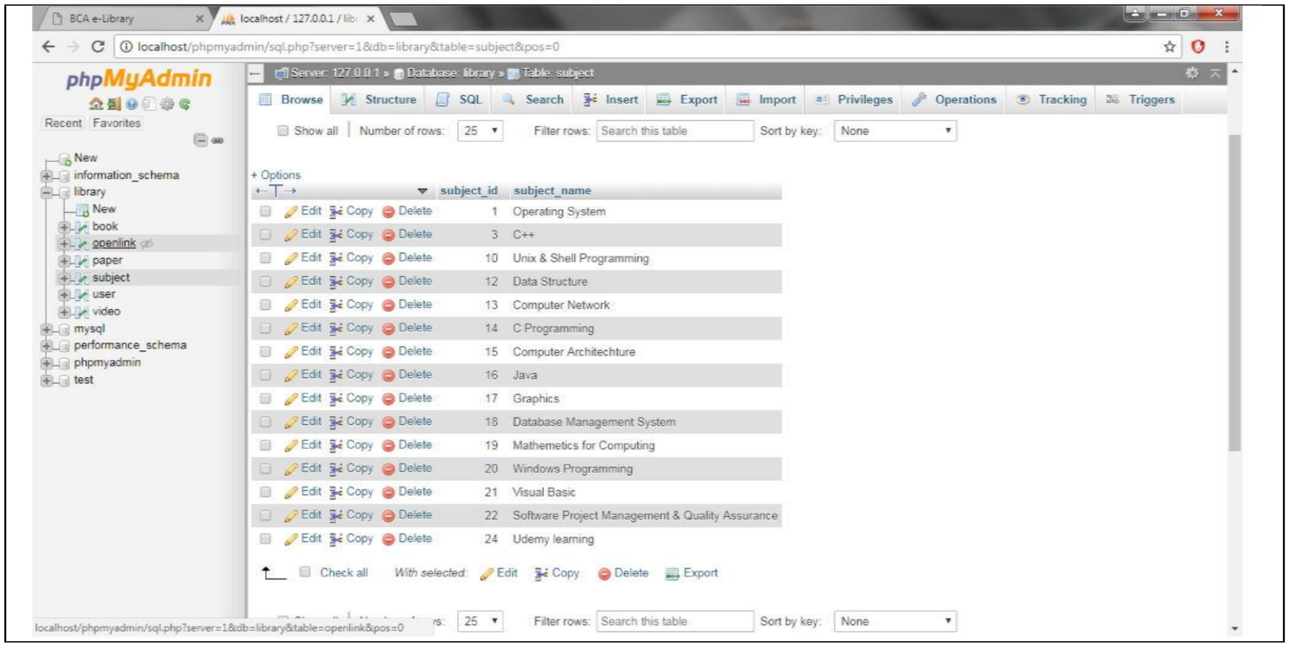
✓ Showing rows 0 - 2 (3 total, Query took 0.0025 seconds.)

```
SELECT * FROM `user`
```

user_id	email	password	first_name	last_name
2	admin@admin.com	$2y$09$tjcH0Cf1vFR.mVgPZQ1fFeLuBfZ0SRPSyDIlcKvj.uf...	Admin	Admin
3	abir@bca.com	$2y$09$4IHOLIMPKzCblKAEcSxkze.mA1hP4Nal91fhcV6GFST...	Abir	Roy

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

➢ Subjects Table from Database



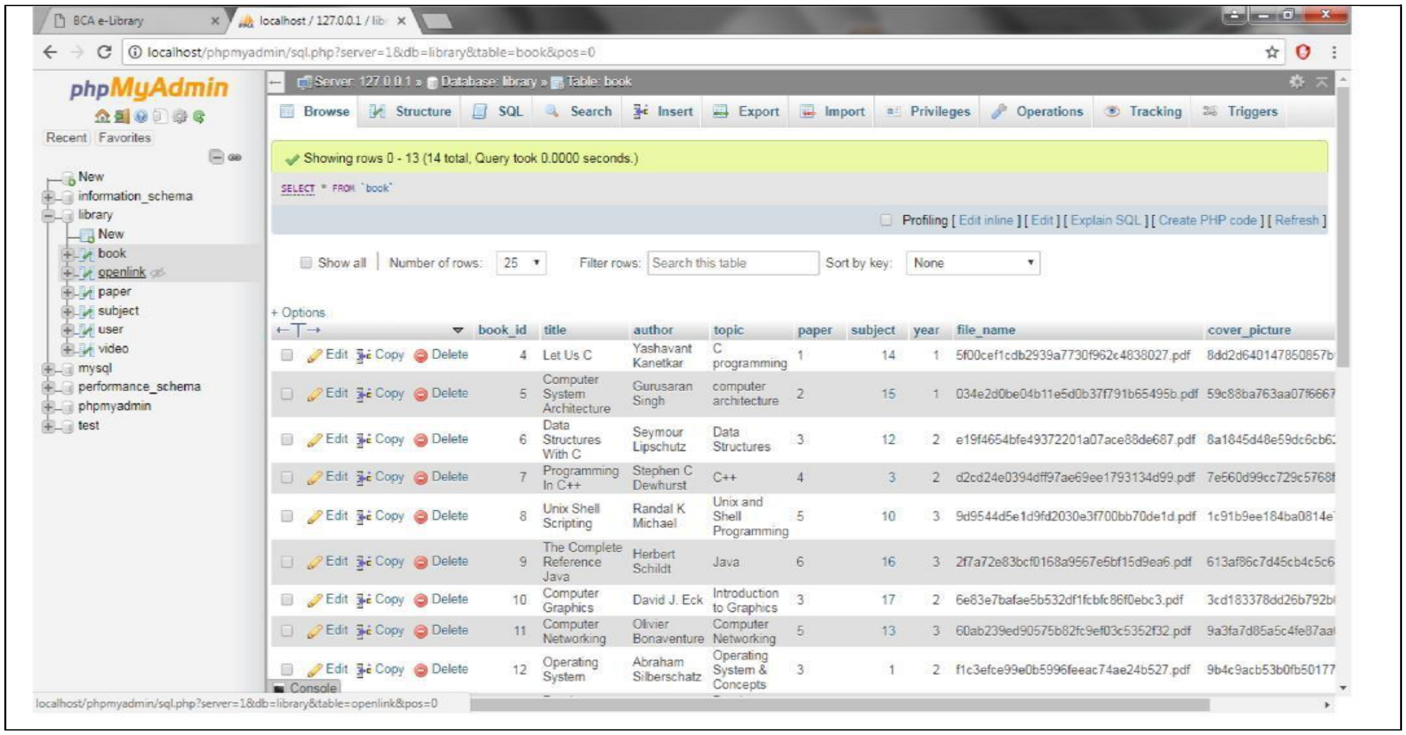
localhost/phpmyadmin/sql.php?server=1&db=library&table=subject&pos=0

Server: 127.0.0.1 » Database: library » Table: subject

subject_id	subject_name
1	Operating System
3	C++
10	Unix & Shell Programming
12	Data Structure
13	Computer Network
14	C Programming
15	Computer Architechture
16	Java
17	Graphics
18	Database Management System
19	Mathemetics for Computing
20	Windows Programming
21	Visual Basic
22	Software Project Management & Quality Assurance
24	Udemy learning

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

➢ Books Table from Database Books Table from Database



localhost/phpmyadmin/sql.php?server=1&db=library&table=book&pos=0

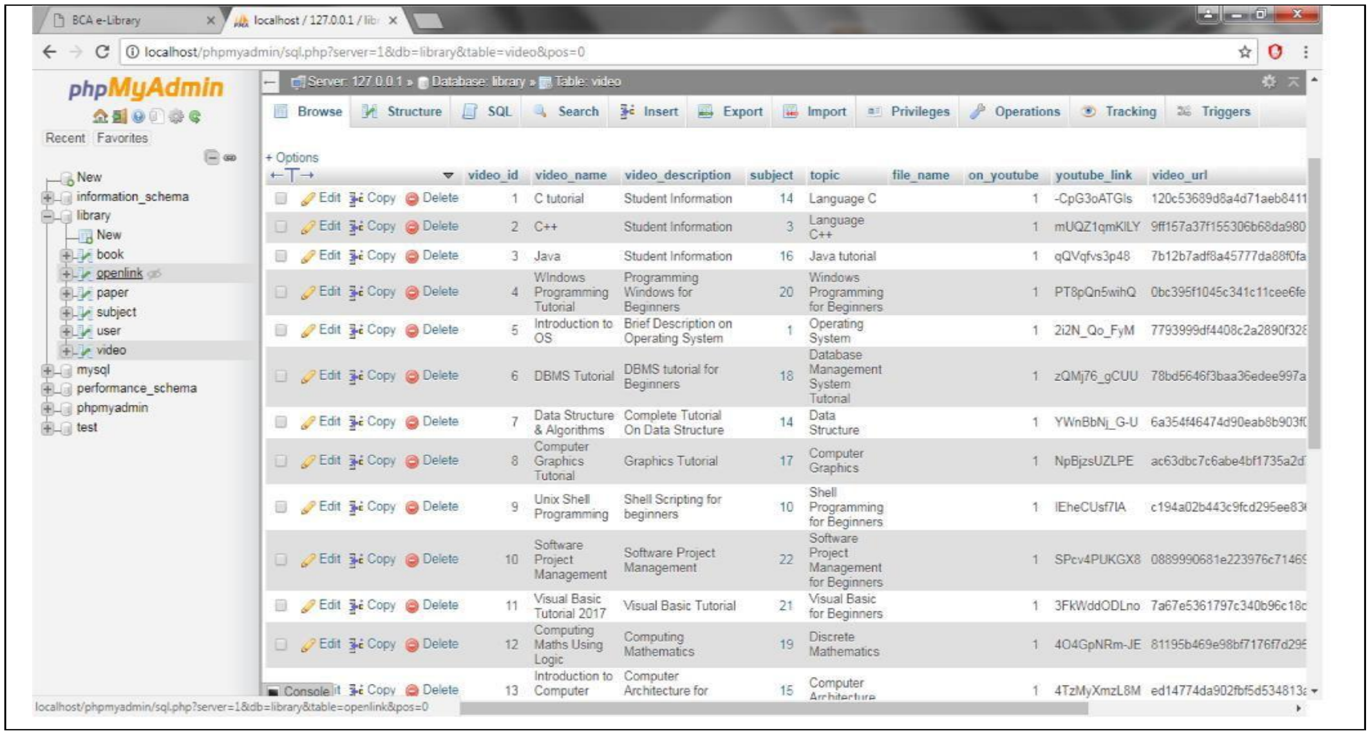
Server: 127.0.0.1 » Database: library » Table: book

Showing rows 0 - 13 (14 total, Query took 0.0000 seconds.)

```
SELECT * FROM `book`
```

book_id	title	author	topic	paper	subject	year	file_name	cover_picture
4	Let Us C	Yashavant Kanetkar	C programming	1	14	1	5f00cef1cdb2939a7730f962c4838027.pdf	8dd2d640147850857b
5	Computer System Architecture	Gurusaran Singh	computer architecture	2	15	1	034e2d0be04b11e5d0b37f791b65495b.pdf	59c88ba763aa07f6667
6	Data Structures With C	Seymour Lipschutz	Data Structures	3	12	2	e19f4654bfe49372201a07ace88de687.pdf	8a1845d48e59dc6cb6
7	Programming In C++	Stephen C Dewhurst	C++	4	3	2	d2cd24e0394dff97ae69ee1793134d99.pdf	7e560d99cc729c5768
8	Unix Shell Scripting	Randal K Michael	Unix and Shell Programming	5	10	3	9d9544d5e1d9fd2030e3f700bb70de1d.pdf	1c91b9ee184ba0814e
9	The Complete Reference Java	Herbert Schildt	Java	6	16	3	2f7a72e83bcf0168a9567e5bf15d9ea6.pdf	613af86c7d45cb4c5c6
10	Computer Graphics	David J. Eck	Introduction to Graphics	3	17	2	6e83e7bafae5b532df1fcbfc86f0ebc3.pdf	3cd183378dd26b792b
11	Computer Networking	Olivier Bonaventure	Computer Networking	5	13	3	60ab239ed90575b82fc9ef03c5352f32.pdf	9a3fa7d85a5c4fe87aa
12	Operating System	Abraham Silberschatz	Operating System & Concepts	3	1	2	f1c3efce99e0b5996feeac74ae24b527.pdf	9b4c9acb53b0fb50177

➢ Videos Table from Database

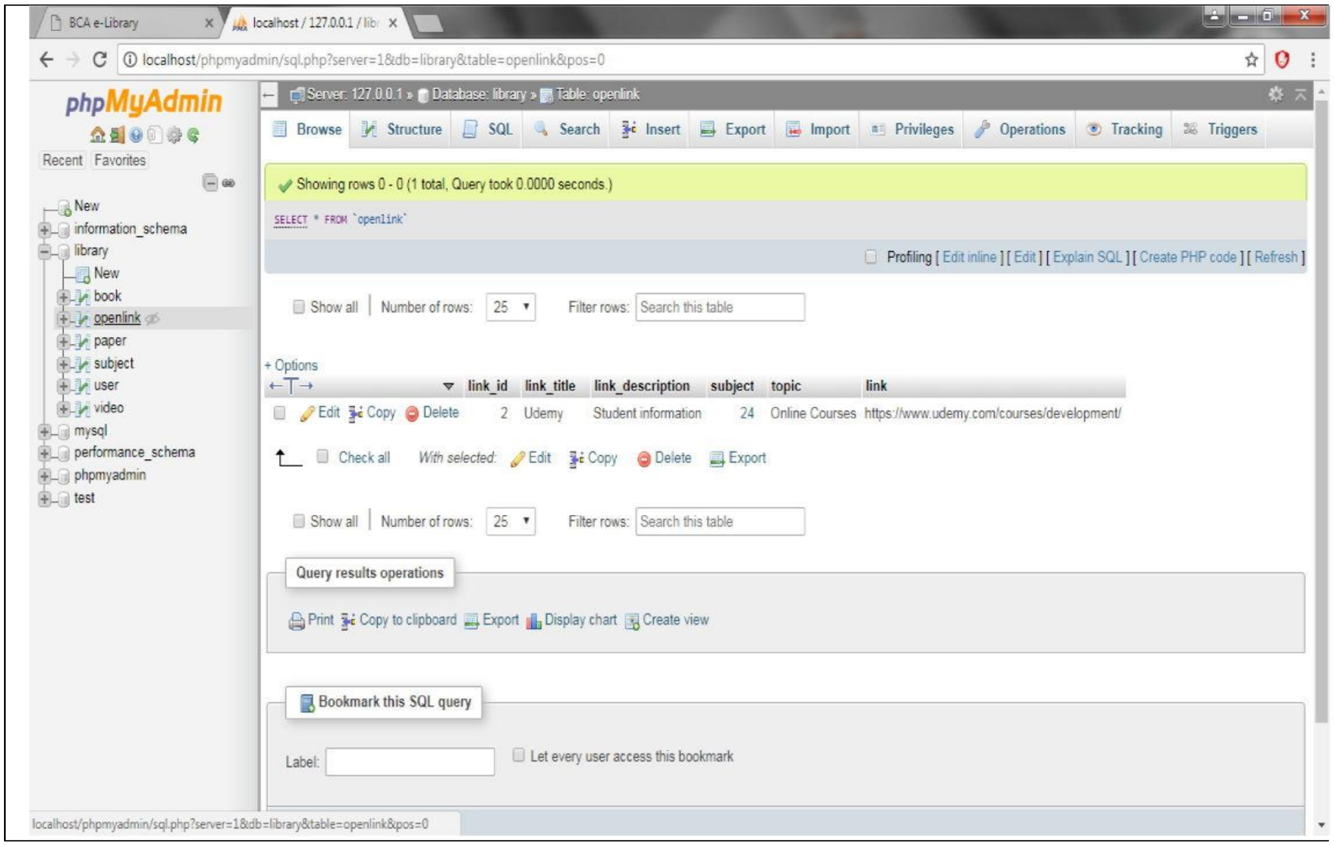


localhost/phpmyadmin/sql.php?server=1&db=library&table=video&pos=0

Server: 127.0.0.1 » Database: library » Table: video

video_id	video_name	video_description	subject	topic	file_name	on_youtube	youtube_link	video_url
1	C tutorial	Student Information	14	Language C		1	-CpG3oATGIs	120c53689d8a4d71aeb8411
2	C++	Student Information	3	Language C++		1	mUQZ1qmKILY	9ff157a37f155306b68da980
3	Java	Student Information	16	Java tutorial		1	qQVqfvs3p48	7b12b7adf8a45777da88f0fa
4	Windows Programming Tutorial	Programming Windows for Beginners	20	Windows Programming for Beginners		1	PT8pQn5wihQ	0bc395f1045c341c11cee6fe
5	Introduction to OS	Brief Description on Operating System	1	Operating System		1	2i2N_Qo_FyM	7793999df4408c2a2890f32
6	DBMS Tutorial	DBMS tutorial for Beginners	18	Database Management System Tutorial		1	zQMj76_gCUU	78bd5646f3baa36edee997a
7	Data Structure & Algorithms	Complete Tutorial On Data Structure	14	Data Structure		1	YWnBbNj_G-U	6a354f46474d90eab8b903f
8	Computer Graphics Tutorial	Graphics Tutorial	17	Computer Graphics		1	NpBjzsUZLPE	ac63dbc7c6abe4bf1735a2d
9	Unix Shell Programming	Shell Scripting for beginners	10	Shell Programming for Beginners		1	lEheCUsf7lA	c194a02b443c9fcd295ee83
10	Software Project Management	Software Project Management	22	Software Project Management for Beginners		1	SPcv4PUKGX8	0889990681e223976c7146
11	Visual Basic Tutorial 2017	Visual Basic Tutorial	21	Visual Basic for Beginners		1	3FkWddODLno	7a67e5361797c340b96c18c
12	Computing Maths Using Logic	Computing Mathematics	19	Discrete Mathematics		1	4O4GpNRm-JE	81195b469e98bf7176f7d29
13	Introduction to Computer	Computer Architecture for	15	Computer Architecture		1	4TzMyXmzL8M	ed14774da902fbf5d534813

➢ Open link Table from Database



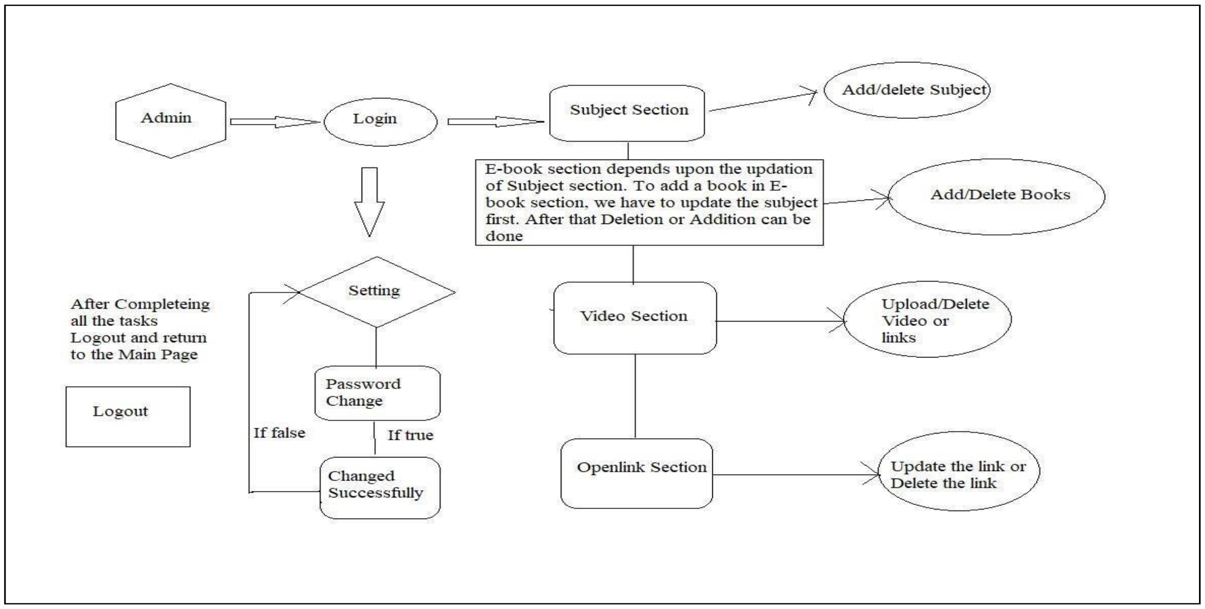
Showing rows 0 - 0 (1 total, Query took 0.0000 seconds.)

```
SELECT * FROM `openlink`
```

link_id	link_title	link_description	subject	topic	link
2	Udemy	Student information	24	Online Courses	https://www.udemy.com/courses/development/

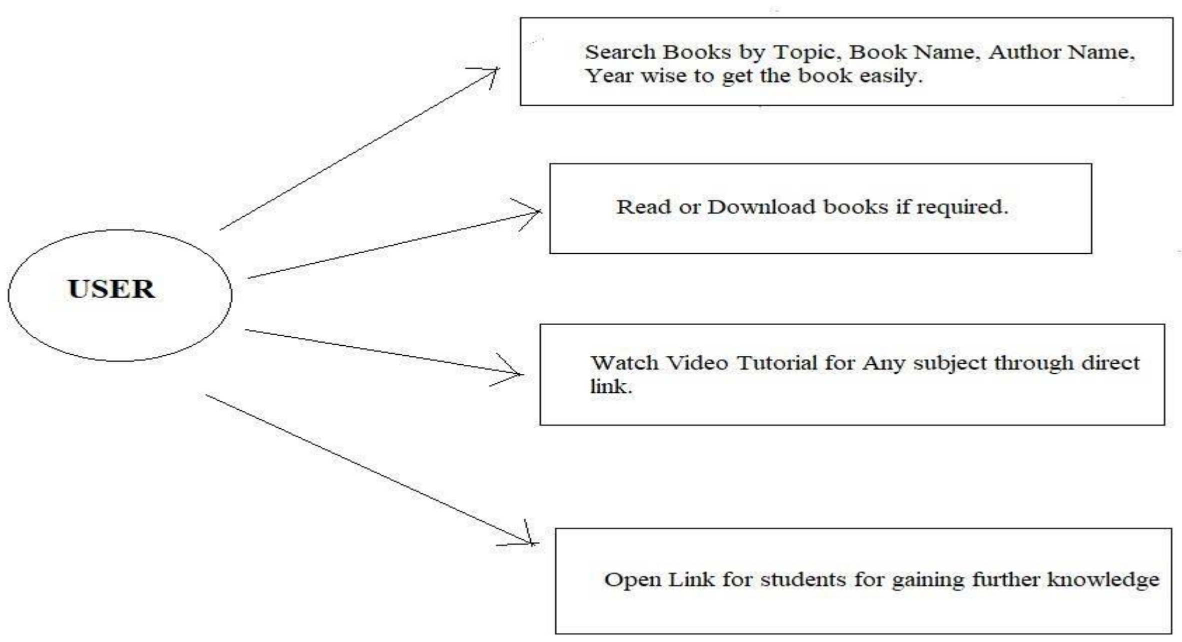
DATA FLOW DIAGRAMS

DATA FLOW DIAGRAM FOR ADMIN LOGIN


Admin
Login
Subject Section
Add/delete Subject
E-book section depends upon the updation of Subject section. To add a book in E-book section, we have to update the subject first. After that Deletion or Addition can be done
Add/Delete Books
Setting
After Completeing all the tasks Logout and return to the Main Page
Logout
Password Change
If false
If true
Changed Successfully
Video Section
Upload/Delete Video or links
Openlink Section
Update the link or Delete the link

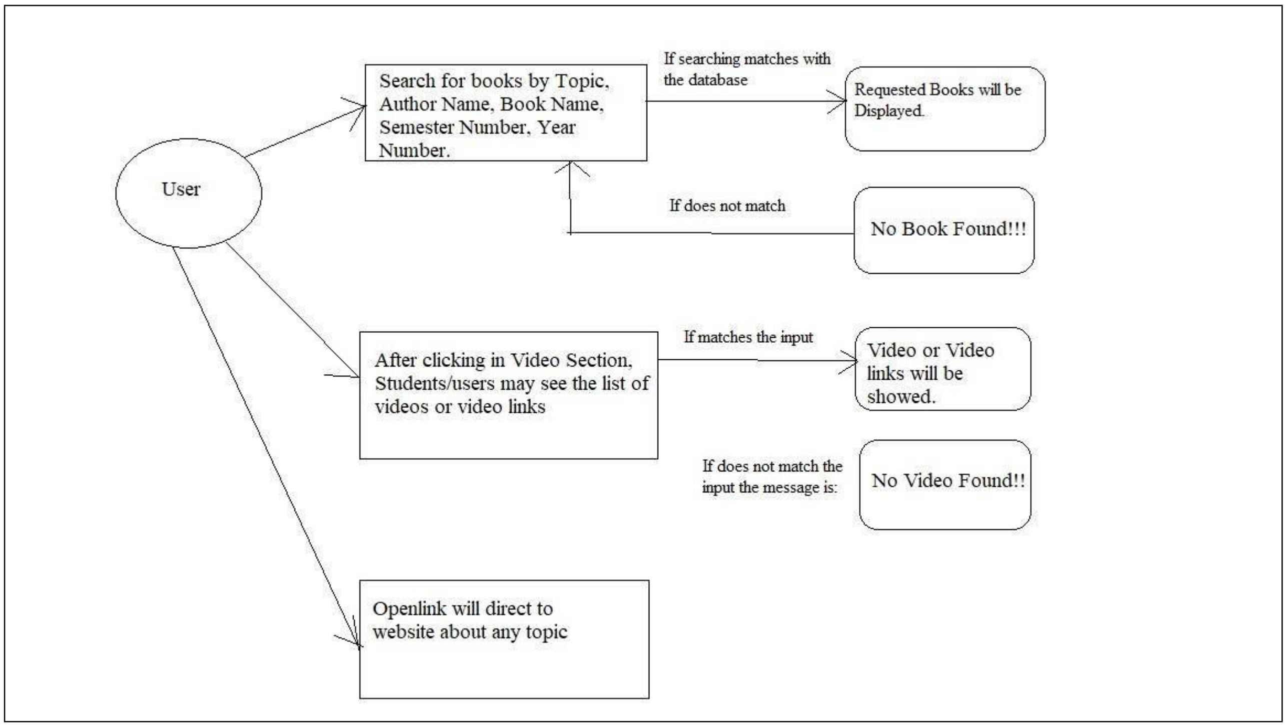
After entering to the home page of the website Admin can choose the Admin Login option where they are asked to enter username & password and if he/she is a valid user then a teacher login page will be displayed.

USE CAESE DIAGRAM FOR USER

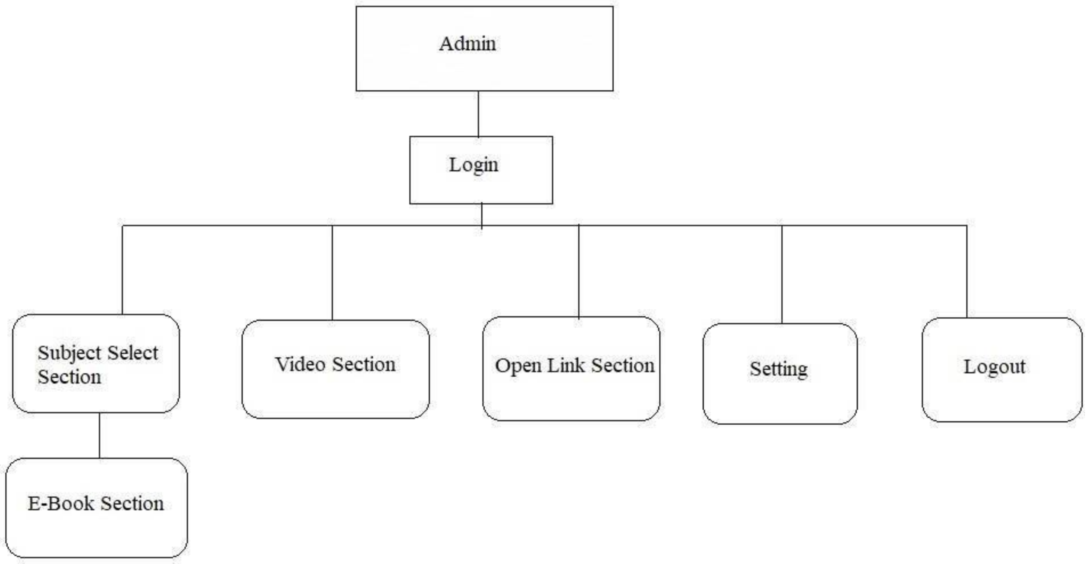

USER
Search Books by Topic, Book Name, Author Name, Year wise to get the book easily.
Read or Download books if required.
Watch Video Tutorial for Any subject through direct link.
Open Link for students for gaining further knowledge

After entering to the home page of the website , student can choose the USER LOGIN option where they are asked to enter username & password , and if he/she is a valid user then a student login page will be displayed.

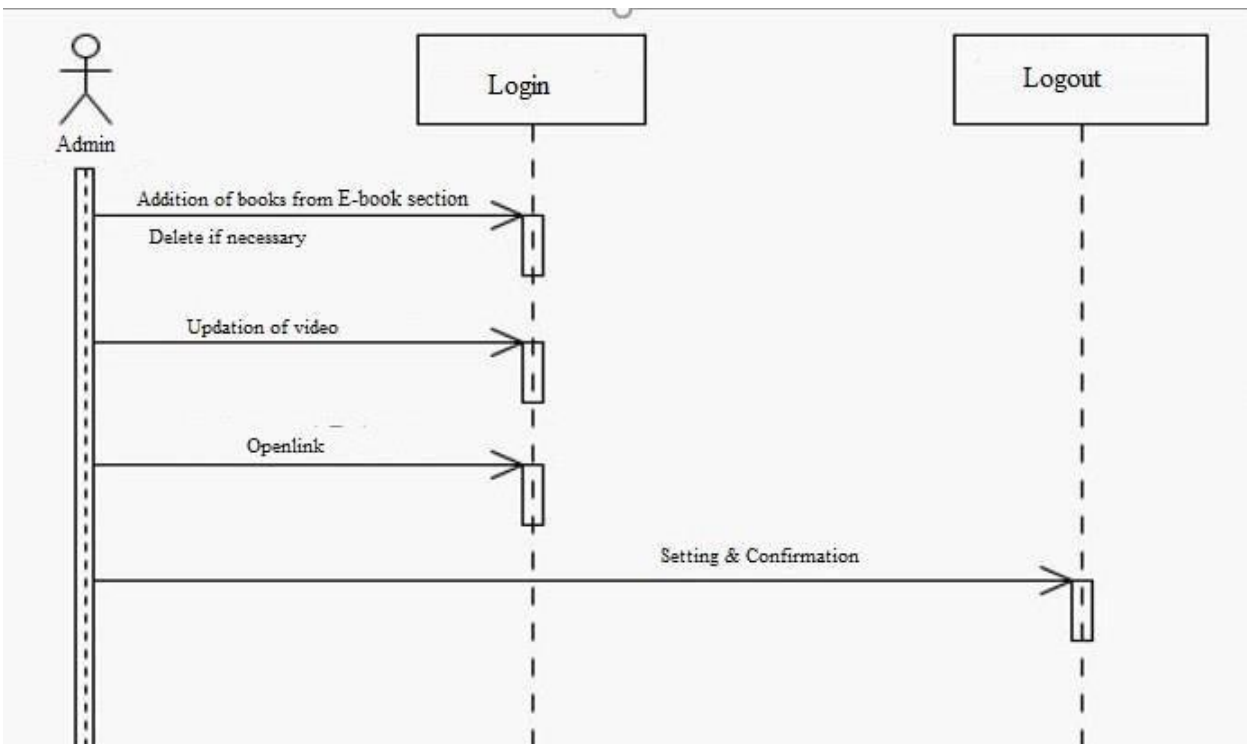
DATA FLOW DIAGRAM FOR USER


User
Search for books by Topic, Author Name, Book Name, Semester Number, Year Number.
If searching matches with the database
Requested Books will be Displayed.
If does not match
No Book Found!!!
After clicking in Video Section, Students/users may see the list of videos or video links
If matches the input
Video or Video links will be showed.
If does not match the input the message is:
No Video Found!!
Openlink will direct to website about any topic

USER CASE DIAGRAM FOR ADMIN

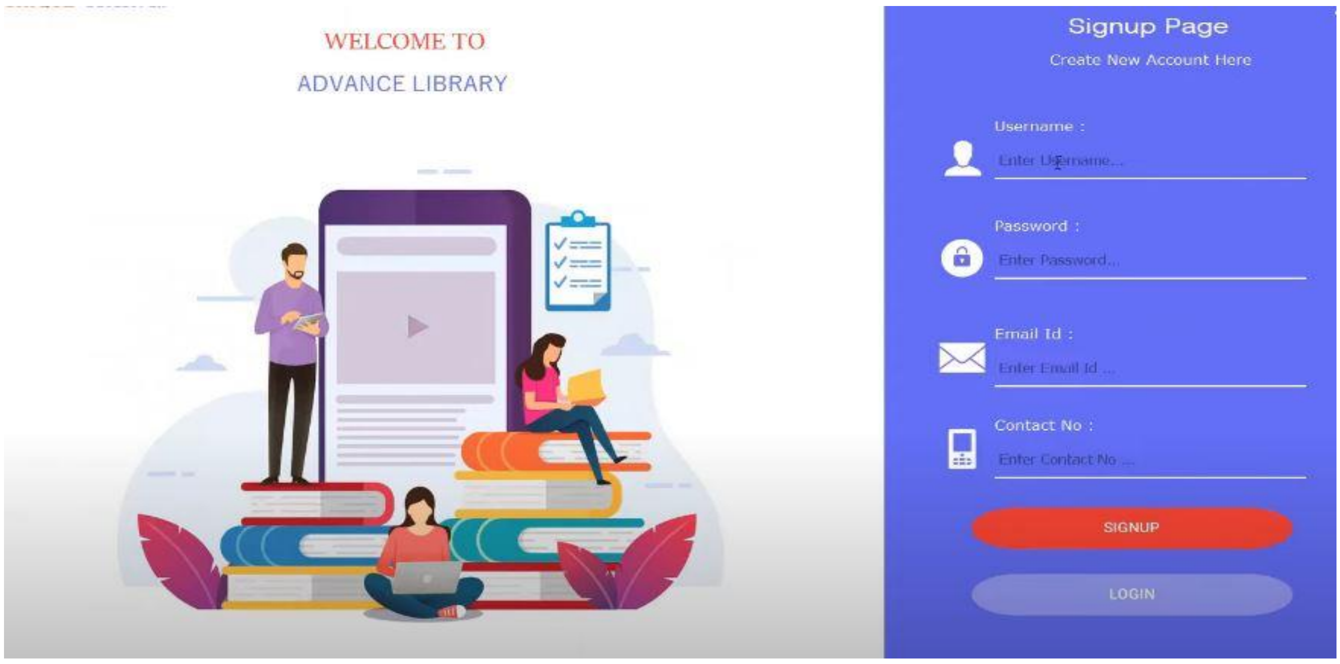


SEQUENCE DIAGRAM

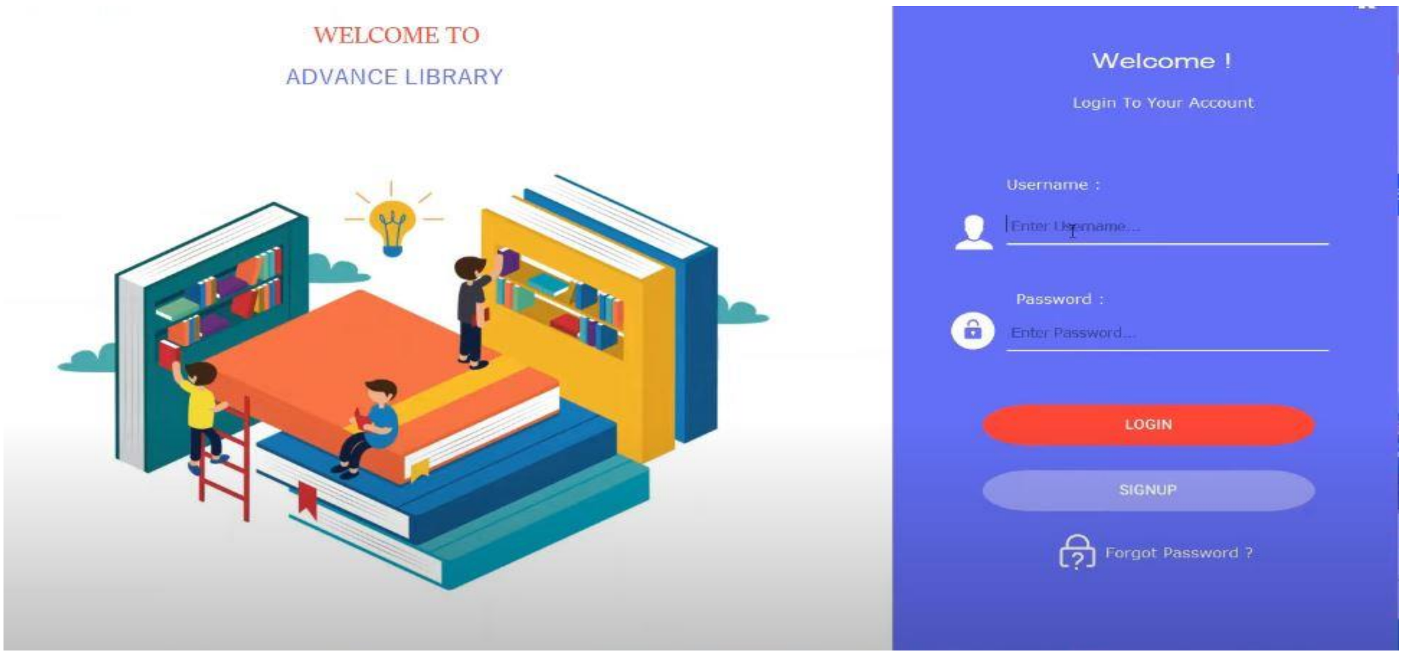


SYSTEM IMPLEMENTATION

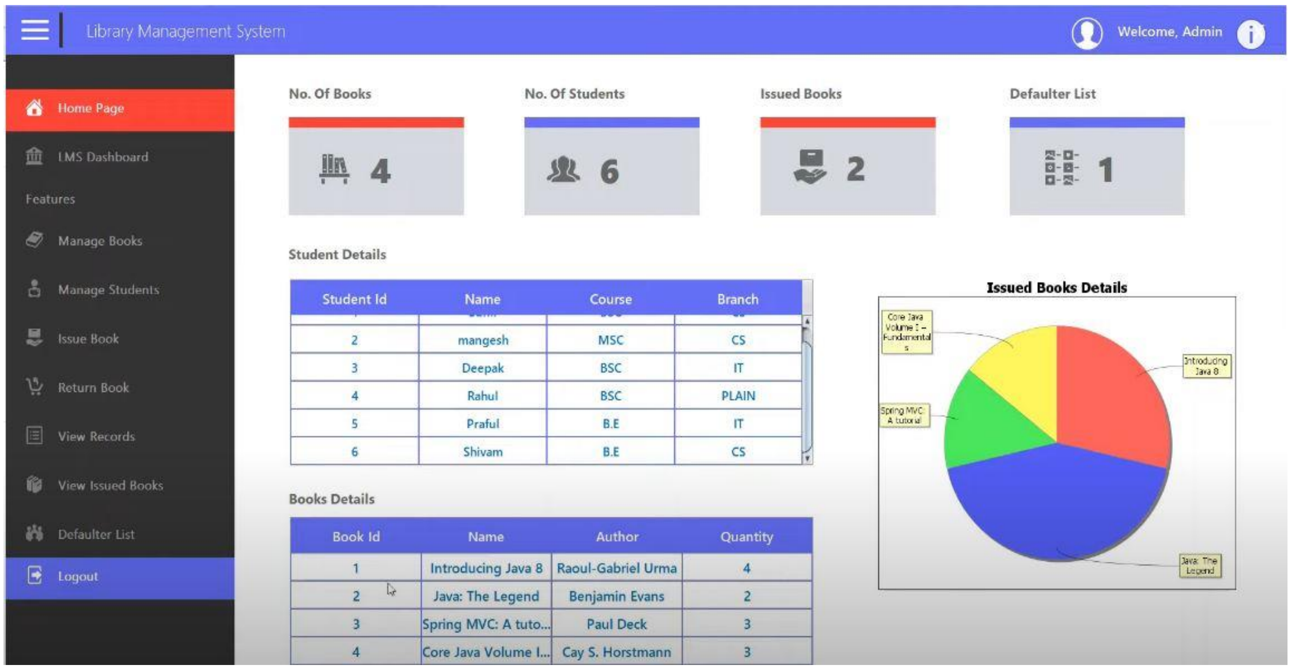
SCREEN SHOT OF SIGN PAGE



SCREEN SHOT OF LOGIN PAGE



SCREEN SHOT OF ANDIM PANEL



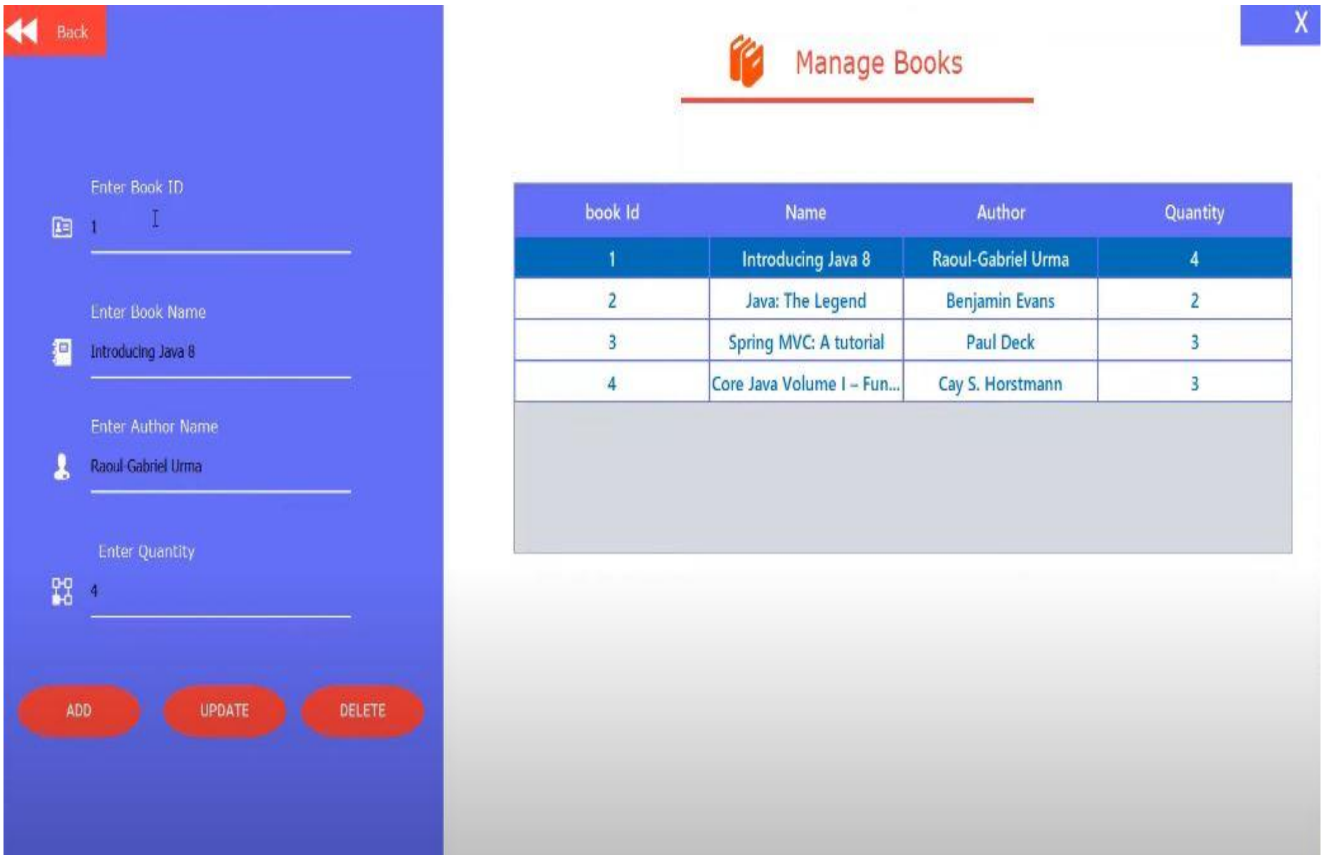
Student Details

Student Id	Name	Course	Branch
2	mangesh	MSC	CS
3	Deepak	BSC	IT
4	Rahul	BSC	PLAIN
5	Praful	B.E	IT
6	Shivam	B.E	CS

Books Details

Book Id	Name	Author	Quantity
1	Introducing Java 8	Raoul-Gabriel Urma	4
2	Java: The Legend	Benjamin Evans	2
3	Spring MVC: A tuto...	Paul Deck	3
4	Core Java Volume I...	Cay S. Horstmann	3

SCREEN SHOT OF MANAGE BOOK



Enter Book ID
1

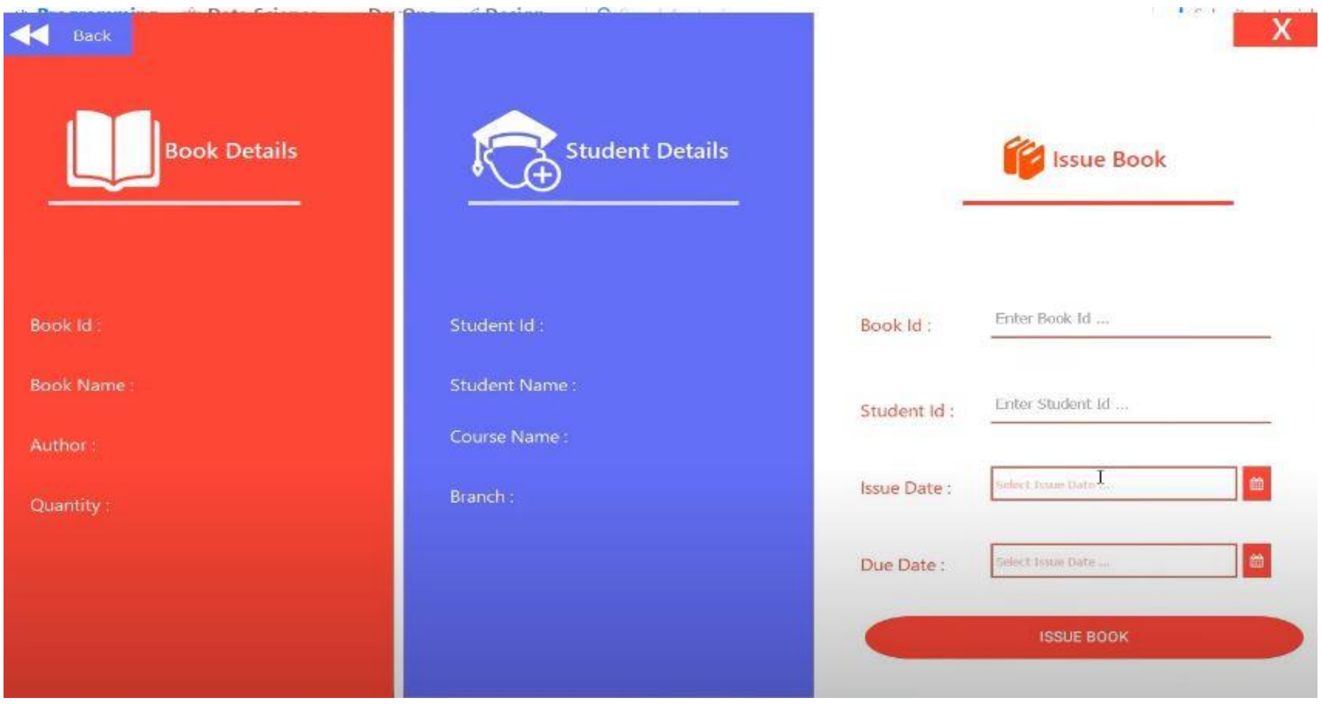
Enter Book Name
Introducing Java 8

Enter Author Name
Raoul-Gabriel Urma

Enter Quantity
4

book Id	Name	Author	Quantity
1	Introducing Java 8	Raoul-Gabriel Urma	4
2	Java: The Legend	Benjamin Evans	2
3	Spring MVC: A tutorial	Paul Deck	3
4	Core Java Volume I – Fun...	Cay S. Horstmann	3

SCREEN SHOT OF ISSUE BOOK PAGE

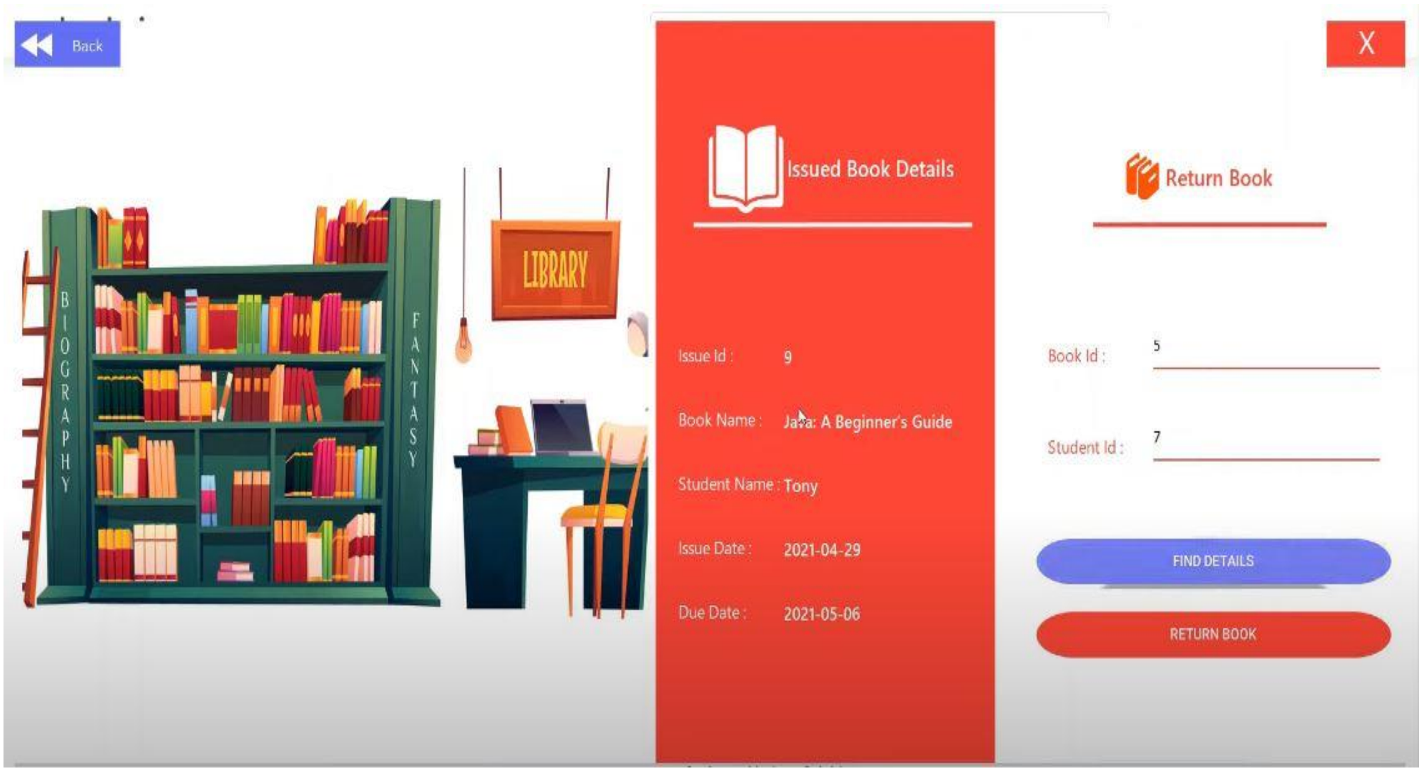


Book Id :
Book Name :
Author :
Quantity :

Student Id :
Student Name :
Course Name :
Branch :

Book Id : Enter Book Id ...
Student Id : Enter Student Id ...
Issue Date : Select Issue Date ...
Due Date : Select Issue Date ...

SCREEN SHOT OF RETURN BOOK



Issue Id : 9

Book Name : Java: A Beginner's Guide

Student Name : Tony

Issue Date : 2021-04-29

Due Date : 2021-05-06

Book Id : 5

Student Id : 7

SYSTEM TESTING

- ☐ **SOFTWARE TESTING STRATEGY**
- ☐ **SOFTWARE TESTING TECHNIQUES**
- ☐ **SPECIAL SYSTEMS TESTS**
- ☐ **MAINTENANCE**

Software Testing Strategy:

A strategy for software testing may be viewed in the context of the spiral. Unit testing begins at the vortex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progresses by moving outward along the spiral to integration testing, where the focus is on design and the construction of the software architecture. Taking another turn outward on the spiral testing, we encounter validation testing, where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally, we arrive at system testing, where the software and other system elements are tested as a whole.

Considering the process from a procedural point of view, testing within the context of software engineering is actually a series of four steps that are implemented sequentially. Initially, tests focus on each component individually, ensuring that it functions properly as a unit. That is why it is called unit testing. Unit testing makes heavy use of white-box testing techniques, exercising specific paths in a module's control structure to ensure complete coverage and maximum error detection. Integration testing addresses the issues associated with the dual problems of verification and program construction. Black-box test case design techniques are the most prevalent during integration, although a limited amount of white-box testing may be used to ensure coverage of major control paths. Validation criteria must be tested. Validation testing provides final assurance that software meets all functional, behavioral, and performance requirements. Black-box testing techniques are used exclusively during validation.

Following are the systematic strategy for software testing:

1. Specify product requirements in a quantifiable manner long before testing commences.
2. State testing objectives explicitly.
3. Understand the users of the software and develop a profile for each user category.
4. Develop a testing plan that emphasizes “rapid cycle testing”.
5. Build “robust” software that is designed to test itself.
6. Use effective formal technical reviews as a filter prior to test itself.
7. Conduct formal technical reviews to assess the test strategy and test cases themselves.
8. Develop a continuous improvement approach for the testing process.

Software Testing Techniques:

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, deign, and code generation. Once source code has been generated, software must be tested to uncover (and correct) as many errors as possible before delivery to the customer. The goal is to design a series of test cases that have a high likelihood of finding errors-but how?. That’s where software testing techniques enter the picture. These techniques provide systematic guidance for designing tests that

1. Exercise the internal logic of software components.
2. Exercise the input and output domains of the program to uncover errors in program function, behavior and performance.

Software is tested from two different perspectives: (A) internal program logic is exercised using “white box” test case design techniques. Software requirements are exercised using “black box” test case design techniques. In both cases, the intent is to find the maximum number of errors with the minimum amount of effort and time. A set of test cases designed to exercise both internal logic and external requirements is designed and documented, expected results are defined, and actual results are recorded.

1.) White Box Testing –

This is also called “glass-box testing”, is a test case design method that uses the control structure of the procedural design to derive test cases. Using whit-box testing methods, we can derive test cases that

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

b) Exercise all logical decisions on their true and false sides.

c) Execute all loops at their boundaries and within their operational bonds.

d) Exercise internal data structures to ensure their validity.

2) Black-Box Testing:

This testing is also called as behavioral testing, focuses on the functional requirements of the software. That is, black-box testing enables the developer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black-box testing is not an alternative to white-box techniques. Black-box testing attempts to find errors in the following categories:

1) Incorrect or missing functions,

2) Interface errors,

3) Errors in data structures or external data base access,

4) Behavior or performance errors, and

5) Initialization and termination errors.

Unlike white-box testing, which is performed early in the testing process, black box testing tends to be applied during later stages of testing. Because black-box testing purposely disregards control structure, attention is focused on the information domain

The first step in black-box testing is to understand the objects that are modeled in software and the relationship that connect these objects. Once this has been accomplished, the next step is to define a series of tests that verify “all objects have the expected relationship to one another.

Unit Testing:

In unit testing the analyst tests the programs making up a system. For this reason, unit testing is sometimes called program testing. Unit testing gives stress on the modules independently of one another, to find errors. This helps the tester in detecting errors in coding and logic that are contained within that module alone. The errors resulting from the interaction between modules are initially avoided. For example, a hotel information system consists of modules to handle reservations; guest checking and checkout; restaurant, room service and miscellaneous charges; convention activities; and accounts receivable billing. For each, it provides the ability to enter, modify or retrieve data and respond to different types of inquiries or print reports. The test cases needed for unit testing should exercise each condition and option.

Unit testing can be performed from the bottom up, starting with smallest and lowest-level modules and proceeding one at a time. For each module in bottom-up testing a short program is used to execute the module and provides the needed data, so that the module is asked to perform the way it will when embedded within the larger system.

Integration Testing:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design.

A) **Top– down Integration**:

This is an incremental approach to construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main control; module (main program). Modules subordinate to the main control module are incorporated into the structure in either a depth-first or breadth-first manner.

B) **Bottom-up Integration**:

Bottom-up integration testing, as its name implies, begins construction and testing with atomic modules (i.e. components at the lowest levels in the program structure). Because components are integrated from the bottom-up, processing required for components subordinate to a given level is always available and the need for stubs is eliminated.

C) **Regression Testing**:

Each time a new module is added as part of integration testing, the software changes. New data paths are established, new I/O may occur, and new control logic is invoked. These changes may cause problems with functions that previously worked flawlessly. In context of an integration test strategy, regression testing is the re-execution of some of tests that have already been conducted to ensure that changes have not propagated side effects. Regression testing is the activity that helps to ensure that changes (due to testing or for other reasons) do not introduce unintended behavior or additional errors.

Validation Testing:

At the culmination of integration testing, software is completely assembled as a package, interfacings have been uncovered and corrected, and a final series of software tests-validation testing-may begin. Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by the customer.

A) **Alpha Testing**:

It is virtually for a software developer to foresee how the customer will really use a program. Instructions for use may be misinterpreted; strange combinations of data may be regularly used; output that seemed clear to the tester may be unintelligible to a user in the field.

B) **Beta Testing**:

The beta test is conducted at one more customer sited by the end-user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a “live” application of the software in an environment that cannot be controlled by the developer.

System Testing:

The important and essential part of the system development phase, after designing and developing the software is system testing. We cannot say that every program or system design is perfect and because of lack of communication between the user and the designer, some error is there in the software development. The number and nature of errors in a newly designed system depend on some usual factors like communication between the user and the designer; the programmer's ability to generate a code that reflects exactly the systems specifications and the time frame for the design.

Theoretically, a newly designed system should have all the parts or sub-systems are in working order, but in reality, each sub-system works independently. This is the time to gather all the subsystem into one pool and test the whole system to determine whether it meets the user requirements. This is the last change to detect and correct errors before the system is installed for user acceptance testing. The purpose of system testing is to consider all the likely variations to which it will be subjected and then push the system to its limits.

Testing is an important function to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully activated. Another reason for system testing is its utility as a user-oriented vehicle before implementation.

System testing consists of the following five steps:

A) **Recovery Testing**:

Recovery testing is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed. If recovery is automatic, re initialization, check pointing mechanism, data recovery, and restart are evaluated for correctness.

B) **Security Testing**:

Security testing attempts to verify that protection mechanism built into a system will, in fact, protect it from improper penetration. During security testing, the tester plays the role of the individual who desires to penetrate the system.

C) **Stress Testing**:

Stress tests are designed to confront programs with abnormal situations. Stress testing executes a system in manner 0that demand resources in the abnormal quantity, frequency, or volume. E.g. special tests may be designed that generate ten interrupts per second, when one or two is the average rate.

D) **Performance Testing**:

This is designed to test the run-time performance of software within the context of an integrated system. Performance testing occurs throughout all steps in the testing process. Even at unit level, the performance of an individual module may be assessed as white-box tests are conducted.

MAINTENANCE

An application has served the business needs of an organization for 10 to 15 years. During that time it has been corrected, adapted, and enhanced many times. The application becomes unstable. It still works, but every time a change is attempted. Much of the software we depend on today is on average 10 to 15 years old. Even when these programs were created using the best design in coding techniques known at the time and (most were not), they were created when program size and storage space were principle concern. They were then migrated to new platform adjusted for changes in machine and operating system technologies and enhance to meet new user needs all without enough regard to overall architecture. The maintenance of existing software can account for over 60% of all effort expended by a development organization and the percentage continues to rise as more software is produced. There are four different maintenance activities-

A) Corrective Maintenance –

Even with the best quality assurance activities, it is likely that the customer will uncover defects in the software. Corrective maintenance changes the software correct defects.

B) Adaptive Maintenance –

Over time, the original environment (e.g. CPU, operating system, business rules, external product characteristics) for which the software was developed is likely to change. Adaptive maintenance results in modification to the software to accommodate the changes to its external environments.

C) Perfective Maintenance –

As software is used, the customer/user will recognize additional functions that will provide benefit. Perfective maintenance extends the software beyond its original functional requirements.

D) Preventive Maintenance –

Computer software deteriorates due to change and because of this preventive maintenance often called software reengineering must be conducted to enable the software to serve the needs of its end users. In sense preventive maintenance make changes to computer program so that they can be more easily corrected, adapted and enhanced.

CONCLUSION & FUTURE SCOPE

This website provides a computerized version of library management system which will benefit the students as well as the staff of the library.

It makes entire process online where student can search books, staff can generate reports and do book transactions. It also has a facility for student login where student can login and can see status of books issued as well request for book or give some suggestions. It has a facility of teacher's login where teachers can add lectures notes and also give necessary suggestion to library and also add info about workshops or events happening in our college or nearby college in the online notice board.

There is a future scope of this facility that many more features such as online lectures video tutorials can be added by teachers as well as online assignments submission facility , a feature Of group chat where students can discuss various issues of engineering can be added to this project thus making it more interactive more user friendly and project which fulfills each users need in the best way possible.