MIDNAPORE COLLEGE (AUTONOMOUS)



**Project Report**

**On**

**E-BookLibrary**

**Management**

**System**

Department of Computer Science and Application

**Submitted By:**

**Subham Sarkar**

**Roll No: 2636, UG, BCA**

**ID: 2018-1618**

**SUBJECT: BCA , PAPER CODE: 605**

**UG, Bachelor in Computer Application**

**Under The Guidance of:**

**Mr. Santi Pada Dua**

**Assistant Professor of Computer Science & Application Department.**

**CERTIFICATE**

Certified that this is a bona fide record of the project work titled

**E-BookLIBRARY MANAGEMENT SYSTEM**

**Done By:**

**Subham Sarkar**

of VI semester BCA,UG in the year of 2021

partial fulfillment of the requirements for the award of Degree of

Bachelor in Computer Application

**Head of The Department : Project Guided By:**

**Mr. Krishna Gopal Dhal Mr. Santi Pada Dua**

**ACKNOWLEDGEMENT**

I would like to express my special thanks of gratitude to our Teacher/Project Guide “Mr. Santi Pada Dua ”, the HOD of BCA “Sir Krishna Gopal Dhal” as well as the honorable principal of Midnapore College “Dr. Gopal Chandra Bera” who gave me the golden opportunity to do this wonderful project, which also helped me in doing a lot of Research and I came to know about so many new things, I am really thankful to them.

Secondly, I would also like to thank my family and friends who helped me a lot in finalizing this project within the limited time frame. I am over helped in all humbleness and gratefulness to acknowledge my depth to all those who have helped me to put these ideas, well above the level of simplicity and into something concrete.

Thanking you!

Subham Sarkar

2636, Semester-VI,

UG, BCA

MIDNAPORE COLLEGE(AUTONOMOUS)

Dept- Computer Science & Application

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| ABSTRACT |  |
| **1. INTRODUCTION** |  |
| 1.1 PROJECT AIMS AND OBJECTIVES |
| 1.2 BACKGROUND OF PROJECT |
| 1.3 SOFTWARE REQUIRMENT  1.4 HARDWARE REQUIRMENT |
| **2. SYSTEM ANALYSIS** |
| 2.1 SOFTWARE REQUIREMENT SPECIFICATION  2.2 SOFTWARE TOOL USED |
| **3. SYSTEM DESIGN** |
| 3.1 TABLE DESIGN ALONG WITH CODE |
| 3.2 DATA FLOW DIAGRAM**’**S |
| **4. SYSTEM IMPLEMENTATION** |
| 4.1 SCREEN SHOTS WITH API’S CODE |
| **5. SYSTEM TESTING** |
| 5.1 UNIT TESTING |
| 5.2 INTEGRATION TESTING |
| **6. CONCLUSION & FUTURE SCOPE** |
| **7. REFERENCES** |

Abstract:

Online Library Management System is a system which maintains the information about the books present in the library, their authors, the members of library to whom books are issued actually members refers to users who register on the website, library staff and all. This is very difficult to organize manually. Maintenance of all this information manually is a very complex task. Owing to the advancement of technology, organization of an Online Library becomes much simple. The Online Library Management has been designed to computerize and automate the operations performed over the information about the members, various category book download, book issues and returns and all other operations. This computerization of library helps in many instances of its maintenances. It reduces the workload of management as most of the manual work done is reduced.

**CHAPTER 1**

**INTRODUCTION**

This chapter gives an overview about the aim , objectives ,background and operation environment of the system.

**1.1 PROJECT AIMS AND OBJECTIVES**

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter. The aims and objectives are as follows:

* + Online book reading by clicking preview button of the book card.****
  + A search bar presents to search availability of books.****
  + Facility to download required book.****
  + Facility to view particular book on particular book section for better experience.
  + User registration & login page for security purpose.
  + Contact feature with google map.
  + Download pdf format of particular book feature.
  + An admin login page for uploading books along with containing book-data.

****

**1.2 BACKGROUND OF PROJECT**

E-Booklibrary Management System is an application which refers to library systems which are generally small or medium in size. It is used by librarian or the admin of the website to manage the library using a computerized system where he/she can add new books along with book data, book cover image and Page sources from where a user can get proper information about the subject.

Books and users maintenance modules are also included in this system which would keep track of the users 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 admin to manage the library with more convenience and in a more efficient way as compared to library systems which are not computerized.

|  |  |
| --- | --- |
| OPERATING SYATEM : | Windows 10 Pro |
| IDE : | intellij idea |
| DATABASE : | MySql |
| BROWSER : | Google Crome |
| Rest Client : | Postman Client |

**1.3 SOFTWARE REQUIRMENT**

|  |  |
| --- | --- |
| PROCESSOR : | Intel i3 6th gen or above for better performance |
| DISK SPACE: | Minimum 50GB of blank space required for storing the project file |

**1.4 HARDWARE REQUIRMENT**

|  |  |
| --- | --- |
| RAM: | Minimum 8GB of ram preferred. |

**CHAPTER 2**

**SYSTEM ANALYSIS**

In this chapter, we will discuss and analyze about the developing process of Library Management System including software requirement specification (SRS) . The functional and non functional requirements are included in SRS part to provide complete description and overview of system requirement before the developing process is carried out.

**2.1** **SOFTWARE REQUIREMENT SPECIFICATION**

**2.1.1 GENERAL DESCRIPTION**

PRODUCT DESCRIPTION:

Library Management System is a computerized system which helps admin(librarian) to manage the library daily activity in electronic format. It reduces the risk of paper work such as file lost, file damaged and time consuming.

PROBLEM STATEMENT:

The problem occurred before having computerized system includes:

**File Cost :**

When computerized system is not implemented file is always lost because of human environment. Some times due to some human error there may be a loss of records.

**File Damaged :**

When computerized system is not invented , file always lost or damaged due to some accident like spilling of water by the authority members on files accidentally. Beside some natural disaster like floods or fires may also damaged the files.

**Difficult to search record :**

When there is no computerized environment there is always a difficulty in searching of records are large in number.

**Space Consuming :**

After the number of records become large the space for physical storage of file and records also increases if no computerized system is implemented.

**Cost Consuming :**

To keep track and manage all the data lot of workers are required, for that a lot of money is waste.

**2.1.2 SYSTEM OBJECTIVES**

**Improvement in control and performance:**

The system is developed to cope up with the current issue and problems of library. The system can add users. Validate users based upon admin and normal users and also bug free.

**Save cost:**

After computerized system implemented less human force will be required to maintain the library thus reducing the overall cost.

**Save time:**

Librarian(Admin) is able to search record and also able to upload various several books using few clicks of mouse and few search keyword thus saving his/her valuable time.

**Lecture note:**

Teachers have a facility to upload lecture notes and various book’s in a pdf file format.

**2.1.3 SYSTEM REQUIREMENTS**

2.1.3.1 NON FUNCTIONAL REQUIREMENTS

* Product Requirment

EFFICIENCY REQUIRMENT :

When online library management system will be implemented librarian(admins) and users will easily access library as searching and book transction will be very fast.

RELIABILITY REQUIRMENT :

The system should accurately performs users registration, user login, user validation, book download feature, book transaction and search feature.

USABILITY REQUIRMENT :

The system is design for a user friendly environment so that normal users and librarian(admins) can perform the various task easily and in an effective way.

ORGANIZATION REQUIRMENT :

**IMPLEMENTATION REQUIRMENT :** In implementing whole system it required two phases. One is the frontend part and another is the backend part. The whole system’s backend part is written in JAVA Spring Boot FrameWork on intellij idea ide software, where the frontend part is based on HTML,CSS,JAVASCRIPT & BOOTSTRAP based on spring boot thymeleaf concept, the front end part also written in intellij idea ide.

2.1.3.2 FUNCTIONAL REQUIREMENTS

1. NORMAL USER

* 1. **REGISTER NEW USER:** At first users register to Your Library website as a normal users. To register in the website users must specify the email id and user name . The password with which the users create their account must have a special character, uppercase and lowercase letters.

**1.2 USER LOGIN**: This feature used by the normal user to login into system. They are required to enter username or the email id and the password which was used in registration. The username and password will be verified and if invalid a pop up message appears and the user is not allowed to enter the system.

**1.3 USER FEATURES :** After log in normal users can visit the website and can view the content of the website. Also able to search their required book from the search bar. The normal user can view the all books section where they find lots of subjects related books. They provide the feature of several book section, from where the find several subject related books and they able to read the book content by clicking preview button option and able to download that book on clicking download book button.

**1.4 USER LOGOUT :** Normal users able to log out from the website. After log out the redirected to the home page of the website.

2. ADMIN USER

**2.1 ADMIN LOGIN**: This feature used by the admin user to login into system. They are required to enter username or the email id and the password which was reside in the the librarymanagement sql file in the source code and this file is imported in the librarymanagement schema of the mysql database. The username and password will be verified and if invalid a pop up message appears and the admin is not allowed to enter the system.

**2.2 ADMIN FEATURES**: After login in the system the admin is able to upload several books and their data such as book name, author name, book content, book pdf fie and source link from several book upload section to several books gallery. Admin can preview the book, delete a particular book from database by clicking delete button, download that book and also able to edit that book. Admin also have the access to whole website.

3. SEARCH FUNTIONILITY

Website provides search functionality to both normal users and admin users. By this feature an both of them able to search book by author name or book name.

**2.1.4 SOFTWARE AND HARDWARE REQUIREMENTS**

This section describes the software and hardware requirements of the system.

2.1.4.1 SOFTWARE REQUIREMENTS

* Operating system- Windows 10 pro is used as the operating system as it is stable and supports more features and is more user friendly****
* Ide - intellij idea is used as a platform to write the whole code. Intellij idea provides lots of advance features and best stability, easy debug, and easy to review the code feature.

****

* Database MYSQL is used as database as it easy to maintain and retrieve records by simple queries which are in English language which are easy to understand and easy to write.****
* Development tools and Programming language- Java Spring boot technology is used to give structure of backend part of the project and spring boot thymeleaf HTML is used to develop webpages as a frotend part of the project , use css for styling the website and also use java script for adding behaviour to our website.

2.1.4.2 HARDWARE REQUIREMENTS

* Intel core i3 6th generation is used as a processor because it is faster than other dual core processors and provide reliable and stable performance.by using this processor we can developing our project without any worries.
* Ram 8 gb is used as it will provide fast reading and writing capabilities and will in turn support in processing.

****

**2.2** **SOFTWARE TOOLS USED**

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

2.2.1 Front end: The front end is designed using of html , css, Java script & Bootstrap.

HTML- **HTML**or**Hyper Text Markup Language**is the main markuplanguage for creating web pages and other information that can be displayed in a web browser.HTML is written in the form of HTML elements consisting of *tags* enclosed in angle brackets (like <html>), within the web page content. HTML tags most commonly come in pairs like <h1> and </h1>, although some tags represent *empty elements* and so are unpaired, for example <img>. The first tag in a pair is the *start tag*, and the second tag is the *end tag* (they are also called *opening tags* and *closing tags*). In between these tags web designers can add text, further tags,comments and other types of text-based content. The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page.HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

 CSS- **Cascading Style Sheets**(**CSS**) is a style sheet language used fordescribing the look and formatting of a document written in a markup language. While most often used to style web pages and interfaces written in HTML and XHTML, the language can be applied to any kind

of XML document, including plain XML, SVG and XUL. CSS is a cornerstone specification of the web and almost all web pages use CSS style sheets to describe their presentation.CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification.

of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content (such as by allowing for table less web design).CSS can also allow the same markup page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices. It can also be used to allow the web page to display differently depending on the screen size or device on which it is being viewed. While the author of a document typically links that document to a CSS file, readers can use a different style sheet, perhaps one on their own computer, to override the one the author has specified. However if the author or the reader did not link the document to a specific style sheet the default style of the browser will be applied.CSS specifies a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called *cascade*, priorities or *weights* are calculated and assigned to rules, so that the results are predictable.

JAVA SCRIPT- **JavaScript**(**JS**) is a dynamic computer programminglanguage. It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. It is also being used in server-side programming, game development and the creation of desktop and mobile applications. JavaScript is a prototype-based scripting language with dynamic typing and has first-class functions. Its syntax was influenced by C. JavaScript copies many names and naming conventions from Java, but the two languages are otherwise unrelated and have very different semantics. The key design principles within JavaScript are taken from

the Self and Scheme programming languages. It is a multi-paradigm language, supporting object-oriented, imperative,

and functional programming styles. The application of JavaScript to use outside of web pages**—**for example, in PDF documents, site-specific browsers, and desktop widgets**—**is also significant. Newer and faster JavaScript VMs and platforms built upon them (notably Node.js) have also increased the popularity of JavaScript for server-side web applications. On the client side, JavaScript was traditionally implemented as an interpreted language but just-in-time compilation is now performed by recent (post-2012) browsers.

* BOOTSTRAP:- **Bootstrap** is a potent front-end framework used to create modern websites and web apps. It's open-source and free to use, yet features numerous HTML and CSS templates for UI interface elements such as buttons and forms. **Bootstrap** also supports JavaScript extensions. **Bootstrap** is a framework to help you design websites faster and easier. It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels, etc. It also gives you support for JavaScript plugins.

2.2.2 Back end: The back end part is designed using of spring boot technology which was a framework of java and used mysql database.

* SPRING BOOT - Spring Boot is an open source Java-based framework used to create a micro Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications. Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can **just run**. You can get started with minimum configurations without the need for an entire Spring configuration setup.

MYSQL- **MySQL**("My S-Q-L", officially, but also called "My Sequel") is (as of July 2013) the world's second most widely used open-source relational database management system (RDBMS). It is named after co-founder Michael Widenius daughter, My. The SQL phrase stands for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety

of proprietary agreements. MySQL was owned and sponsored by a single for -profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other 'AMP' stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL. For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases

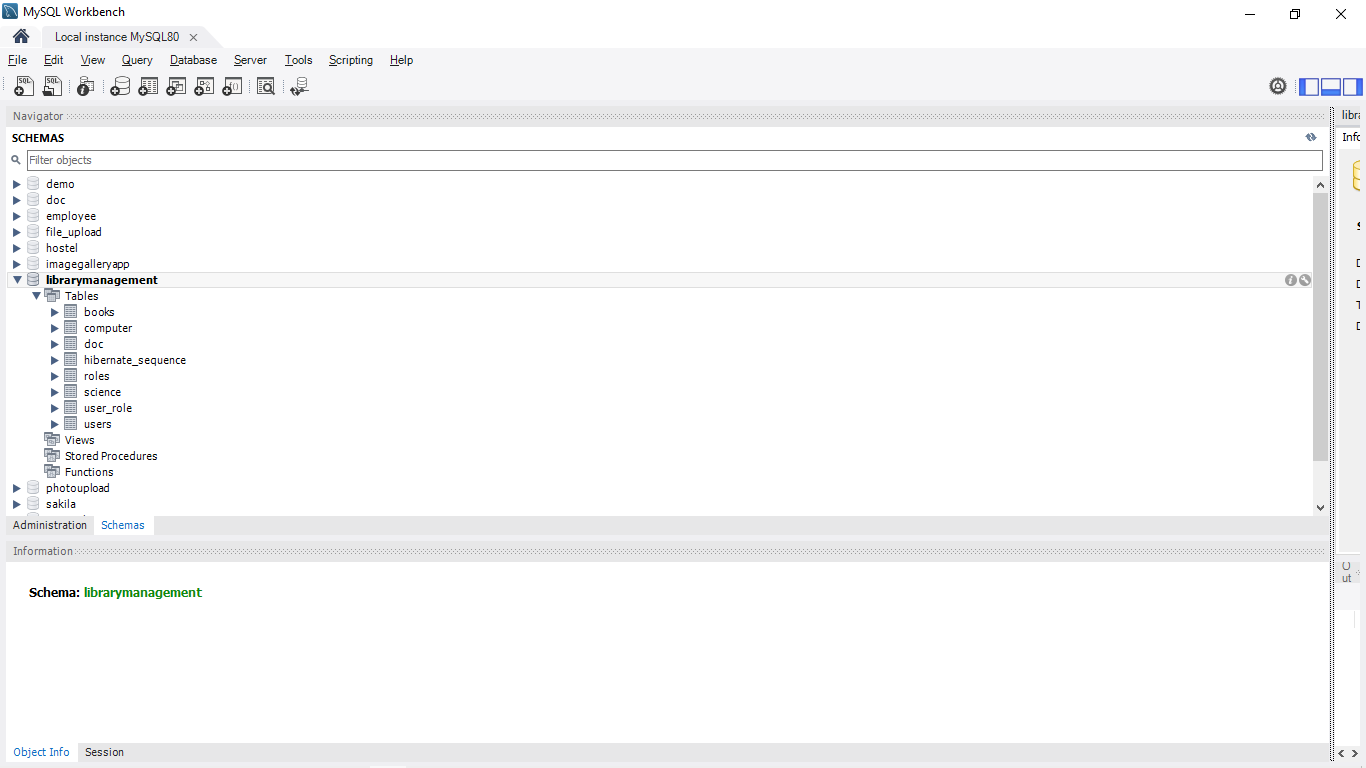
**CHAPTER 3**

**SYSTEM DESIGN ALONG WITH CODE**

3.1 **TABLE DESIGN**

VARIOUS TABELS TO MAINTAIN INFORMATION

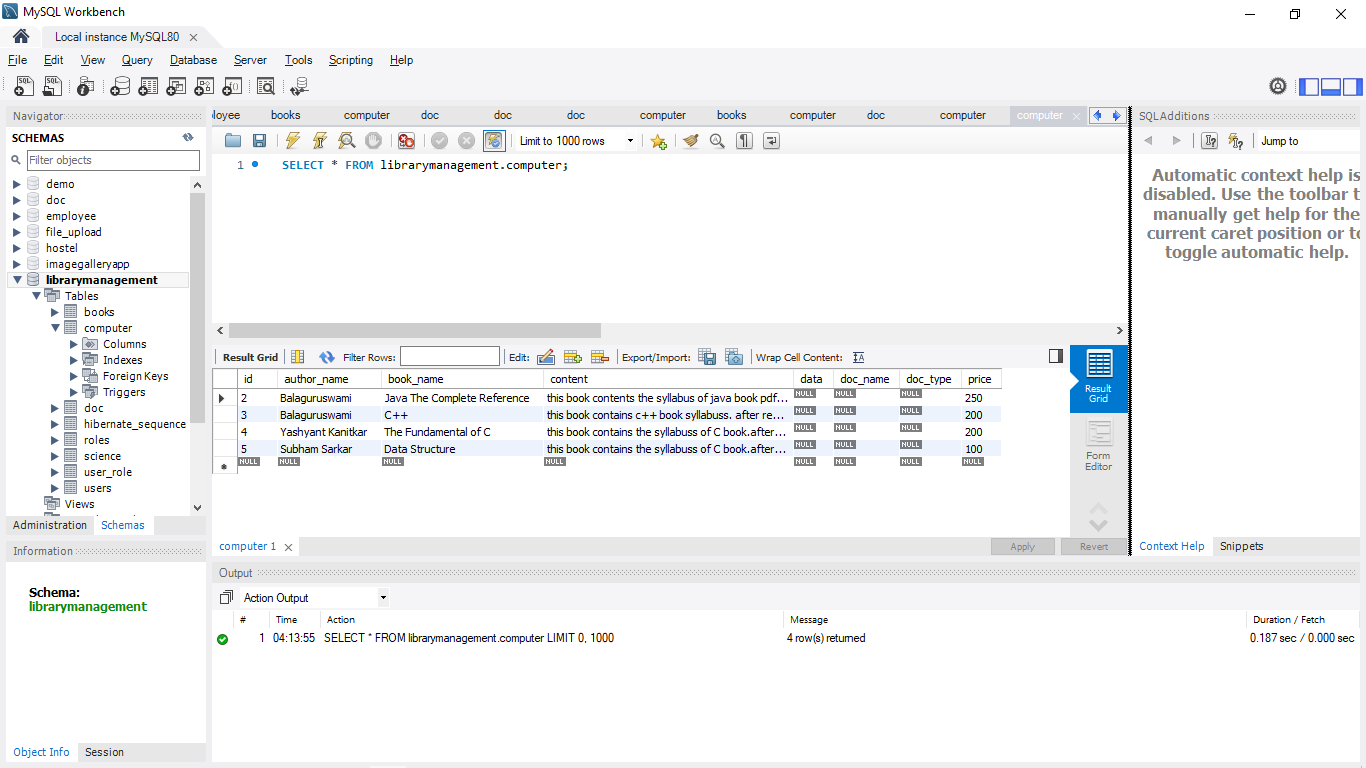
* **Librarymanagement Schema from MySql Database**

****

**Source code for creation and setting up the database :**

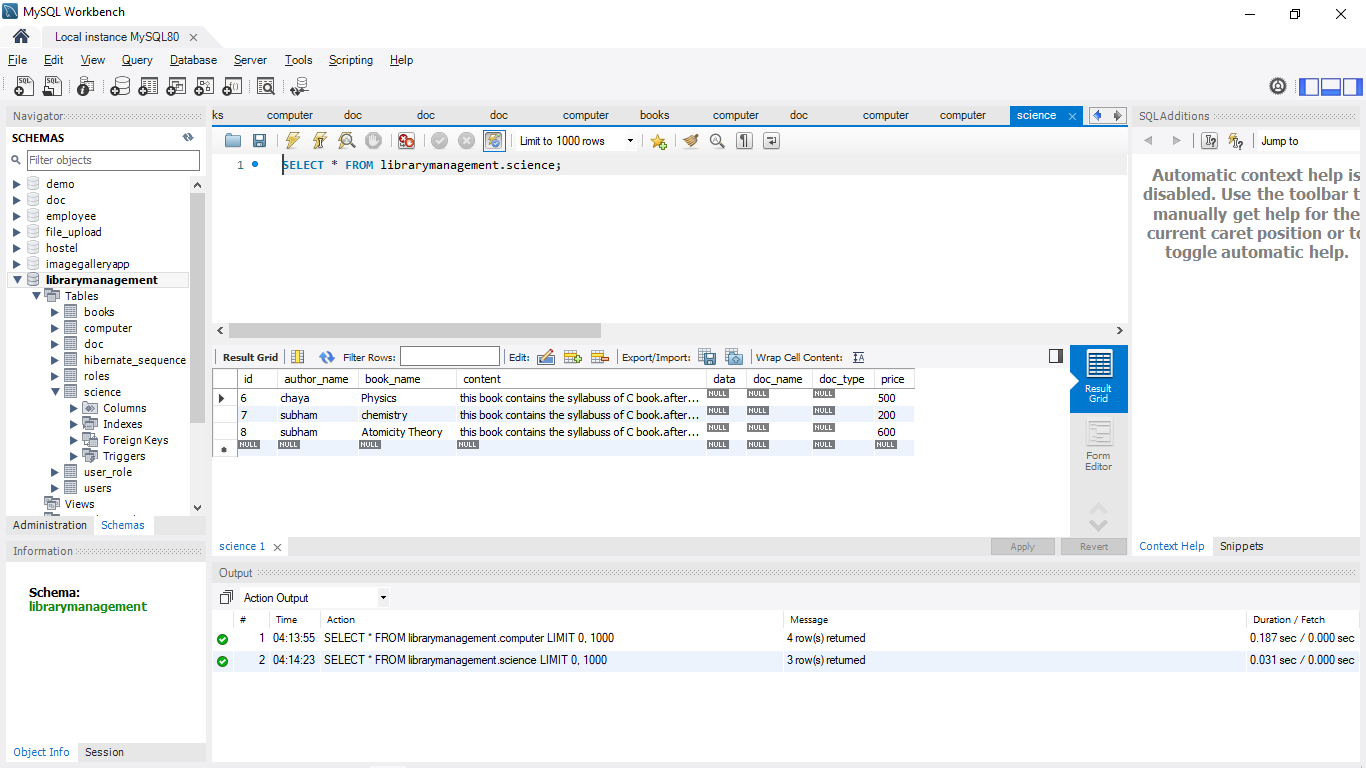
spring.datasource.url=jdbc:mysql://localhost:3306/librarymanagement?useLegacyDatetimeCode=false&serverTimezone=Asia/Jakarta&useSSL=false  
spring.datasource.username=root  
spring.datasource.password=subham99  
spring.jpa.database=MYSQL  
spring.datasource.driver-class-name=com.mysql.jdbc.Driver  
  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect  
server.port=8282  
  
spring.servlet.multipart.enabled=true  
spring.servlet.multipart.file-size-threshold=2KB  
spring.servlet.multipart.max-file-size=200MB  
spring.servlet.multipart.max-request-size=200MB

above code describes setup and connectivity with database in application.properties file. On here I also describes the max file size and max request size which was the size of pdf files ,which was uploaded into databsa.

**Several Books Table from Database(Computer Book Table)**

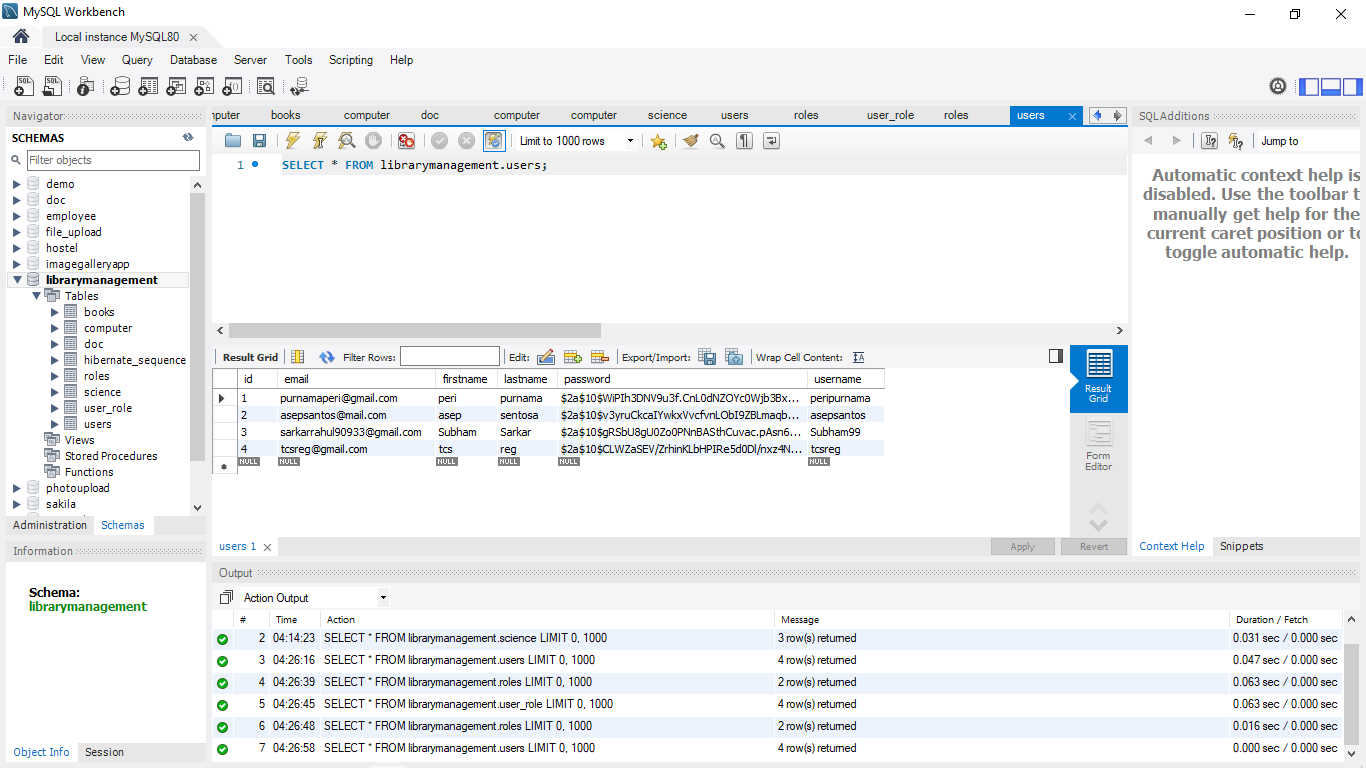
**Source code for creation the computer book table :**

package com.example.subham.model;  
  
import javax.persistence.\*;  
import javax.validation.constraints.NotNull;  
import java.util.Arrays;  
  
@Entity  
@Table(name = "computer")  
public class Computer {  
 @Id  
 @GeneratedValue  
 private int id;  
  
 @Override  
 public String toString() {  
 return "Books{" +  
 "id=" + id +  
 ", bookName='" + bookName + '\'' +  
 ", authorName='" + authorName + '\'' +  
 ", docName='" + docName + '\'' +  
 ", docType='" + docType + '\'' +  
 ", price='" + price + '\'' +  
 ", content='" + content + '\'' +  
 ", data=" + Arrays.*toString*(data) +  
 '}';  
 }  
  
 @NotNull  
  
 private String bookName;  
 @NotNull  
  
 private String authorName;  
  
 private String docName;  
  
  
 public Computer(int id, String bookName, String authorName, String docName, String docType, String price, String content, byte[] data) {  
 this.id = id;  
 this.bookName = bookName;  
 this.authorName = authorName;  
 this.docName = docName;  
 this.docType = docType;  
 this.price = price;  
 this.content = content;  
 this.data = data;  
 }  
  
  
 private String docType;  
 @NotNull  
 private String price;  
  
 public String getPrice() {  
 return price;  
 }  
  
 public void setPrice(String price) {  
 this.price = price;  
 }  
  
 public String getContent() {  
 return content;  
 }  
  
 public void setContent(String content) {  
 this.content = content;  
 }  
  
 @Lob  
 private String content;  
 @Lob  
 private byte[] data;  
  
  
  
  
  
 public Computer() {  
  
 }  
  
 public Computer(String docname, String contentType, byte[] bytes) {  
  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getBookName() {  
 return bookName;  
 }  
  
 public void setBookName(String bookName) {  
 this.bookName = bookName;  
 }  
  
 public String getAuthorName() {  
 return authorName;  
 }  
  
 public void setAuthorName(String authorName) {  
 this.authorName = authorName;  
 }  
  
 public String getDocName() {  
 return docName;  
 }  
  
 public void setDocName(String docName) {  
 this.docName = docName;  
 }  
  
 public String getDocType() {  
 return docType;  
 }  
  
 public void setDocType(String docType) {  
 this.docType = docType;  
 }  
  
 public byte[] getData() {  
 return data;  
 }  
  
 public void setData(byte[] data) {  
 this.data = data;  
 }  
  
 public void setAuthorName() {  
 }  
  
 public void setBookName() {  
 }  
  
 public void setContent() {  
 }  
  
 public void setData() {  
 }  
  
 public void setDocName() {  
 }  
  
 public void setPrice() {  
 }  
}

**Several Books Table from Database (Science Book Table)**

**Source code for creation the Science book table :**

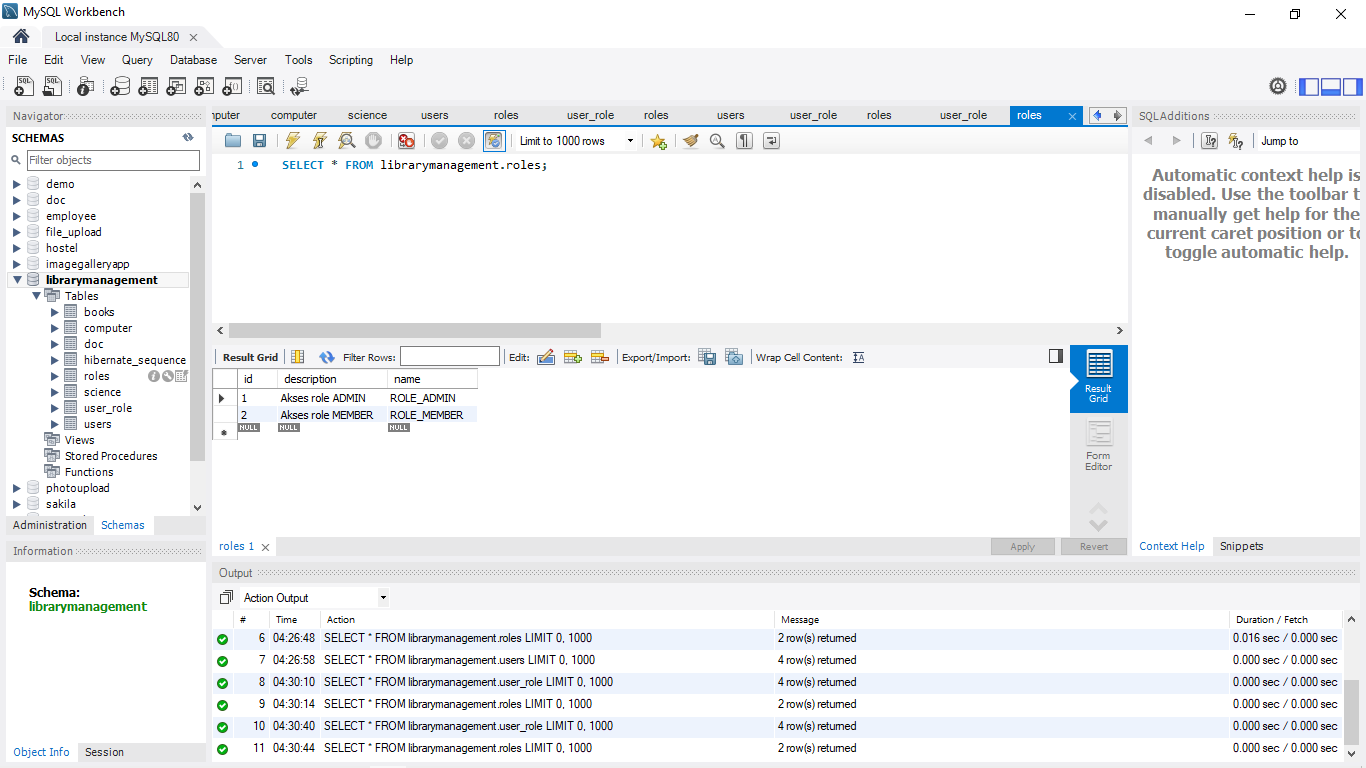
package com.example.subham.model;  
  
import javax.persistence.\*;  
import javax.validation.constraints.NotNull;  
import java.util.Arrays;  
  
@Entity  
@Table(name = "science")  
public class Science {  
 @Id  
 @GeneratedValue  
 private int id;  
  
 @Override  
 public String toString() {  
 return "Books{" +  
 "id=" + id +  
 ", bookName='" + bookName + '\'' +  
 ", authorName='" + authorName + '\'' +  
 ", docName='" + docName + '\'' +  
 ", docType='" + docType + '\'' +  
 ", price='" + price + '\'' +  
 ", content='" + content + '\'' +  
 ", data=" + Arrays.*toString*(data) +  
 '}';  
 }  
  
 @NotNull  
  
 private String bookName;  
 @NotNull  
  
 private String authorName;  
  
 private String docName;  
  
  
 public Science(int id, String bookName, String authorName, String docName, String docType, String price, String content, byte[] data) {  
 this.id = id;  
 this.bookName = bookName;  
 this.authorName = authorName;  
 this.docName = docName;  
 this.docType = docType;  
 this.price = price;  
 this.content = content;  
 this.data = data;  
 }  
  
  
 private String docType;  
 @NotNull  
 private String price;  
  
 public String getPrice() {  
 return price;  
 }  
  
 public void setPrice(String price) {  
 this.price = price;  
 }  
  
 public String getContent() {  
 return content;  
 }  
  
 public void setContent(String content) {  
 this.content = content;  
 }  
  
 @Lob  
 private String content;  
 @Lob  
 private byte[] data;  
  
  
  
  
  
 public Science() {  
  
 }  
  
 public Science(String docname, String contentType, byte[] bytes) {  
  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getBookName() {  
 return bookName;  
 }  
  
 public void setBookName(String bookName) {  
 this.bookName = bookName;  
 }  
  
 public String getAuthorName() {  
 return authorName;  
 }  
  
 public void setAuthorName(String authorName) {  
 this.authorName = authorName;  
 }  
  
 public String getDocName() {  
 return docName;  
 }  
  
 public void setDocName(String docName) {  
 this.docName = docName;  
 }  
  
 public String getDocType() {  
 return docType;  
 }  
  
 public void setDocType(String docType) {  
 this.docType = docType;  
 }  
  
 public byte[] getData() {  
 return data;  
 }  
  
 public void setData(byte[] data) {  
 this.data = data;  
 }  
  
 public void setAuthorName() {  
 }  
  
 public void setBookName() {  
 }  
  
 public void setContent() {  
 }  
  
 public void setData() {  
 }  
  
 public void setDocName() {  
 }  
  
 public void setPrice() {  
 }  
}

**Users Table from Database :**

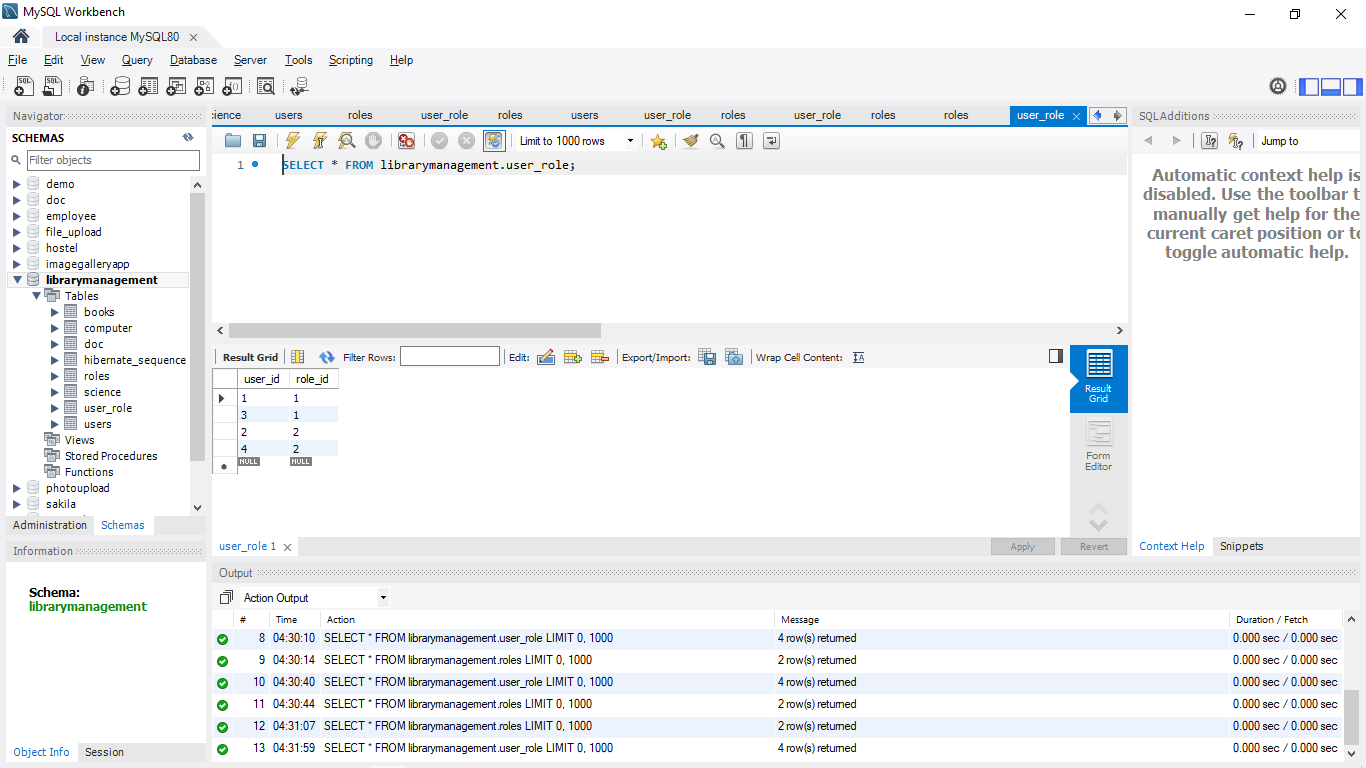
|  |
| --- |
| **** |
|
|  |

**Source code for creation the user table :**

package com.example.subham.model;  
  
import javax.persistence.\*;  
import java.util.Set;  
  
@Entity  
@Table(name = "USERS")  
public class User {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @Column(name = "FIRSTNAME", length = 75)  
 private String firstname;  
  
 @Column(name = "LASTNAME", length = 75)  
 private String lastname;  
  
 @Column(name = "USERNAME", length = 100, unique = true)  
 private String username;  
  
 @Column(name = "EMAIL", length = 100, unique = true)  
 private String email;  
  
 @Column(name = "PASSWORD")  
 private String password;  
  
 @ManyToMany(fetch = FetchType.*LAZY*)  
 @JoinTable(name = "user\_role", joinColumns = {  
 @JoinColumn(name = "user\_id", referencedColumnName = "id") }, inverseJoinColumns = @JoinColumn(name = "role\_id", referencedColumnName = "id"))  
 private Set<Role> roles;  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getFirstname() {  
 return firstname;  
 }  
  
 public void setFirstname(String firstname) {  
 this.firstname = firstname;  
 }  
  
 public String getLastname() {  
 return lastname;  
 }  
  
 public void setLastname(String lastname) {  
 this.lastname = lastname;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public Set<Role> getRoles() {  
 return roles;  
 }  
  
 public void setRoles(Set<Role> roles) {  
 this.roles = roles;  
 }  
  
}

**Role Table from Database:**

**Here role table defines the two phase roles of a normal user one is admin role and one is normal use rolw**

**User\_Role Table from Database:** with the help of user role table we change a normal user into admin user by identifying their ID.

**Source code for creation the user\_role table :**

package com.example.subham.model;  
  
import javax.persistence.\*;  
import java.util.HashSet;  
import java.util.Set;  
  
@Entity  
@Table(name = "ROLES")  
public class Role {  
  
 @Id  
 @GeneratedValue(strategy=GenerationType.*IDENTITY*)  
 private Long id;  
   
 @Column(name="name", unique=true, length=100)  
 private String name;  
   
 @Lob  
 @Column(name="description")  
 private String description;  
   
 @ManyToMany(mappedBy="roles")  
 private Set<User> users = new HashSet<>();  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
   
}

with the help of above code we construct the entities for the user role table into database.

**3.2** **DATA FLOW DIAGRAMS**

DATA FLOW DIAGRAM FOR ADMIN USER LOGIN

USERNAME

PASSWORD

ADMIN

USER

If false If true

After performing all tasks by

Admin, admin can log out and

return To main page that is home

page.

LOG OUT

ADMIN PAGE

USE CASE DIAGRAM FOR ADMIN USER

VIEW ALL AND SEVERAL BOOKS

SEARCH FUNCTIONALITY

EDIT BOOK DATA

DELETE PARTICULAR BOOK

UPLOAD BOOK PDF

UPLOAD SEVERAL BOOK WITH DATA

LOGIN

ADMIN USER

DATA FLOW DIAGRAM FOR NORMAL USER

If already registered

REGISTRATION

After performing several

Tasks by user. User can log

Out from the page and

Redirected to home page

NORMAL

USER PAGE

LOG OUT

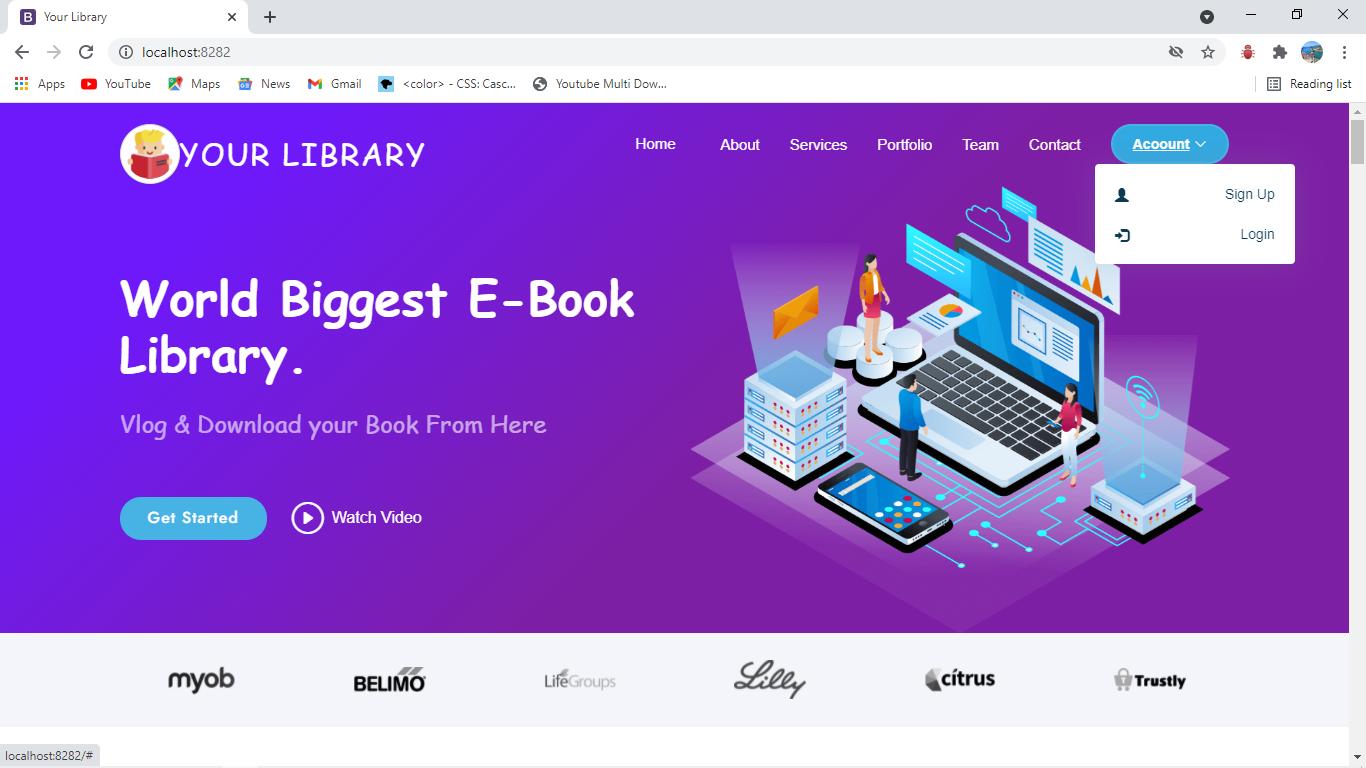
USE CASE DIAGRAM FOR USER

**CHAPTER 4**

**SYSTEM IMPLEMENTATION**

SCREENSHOTS WITH REST API’S CODE

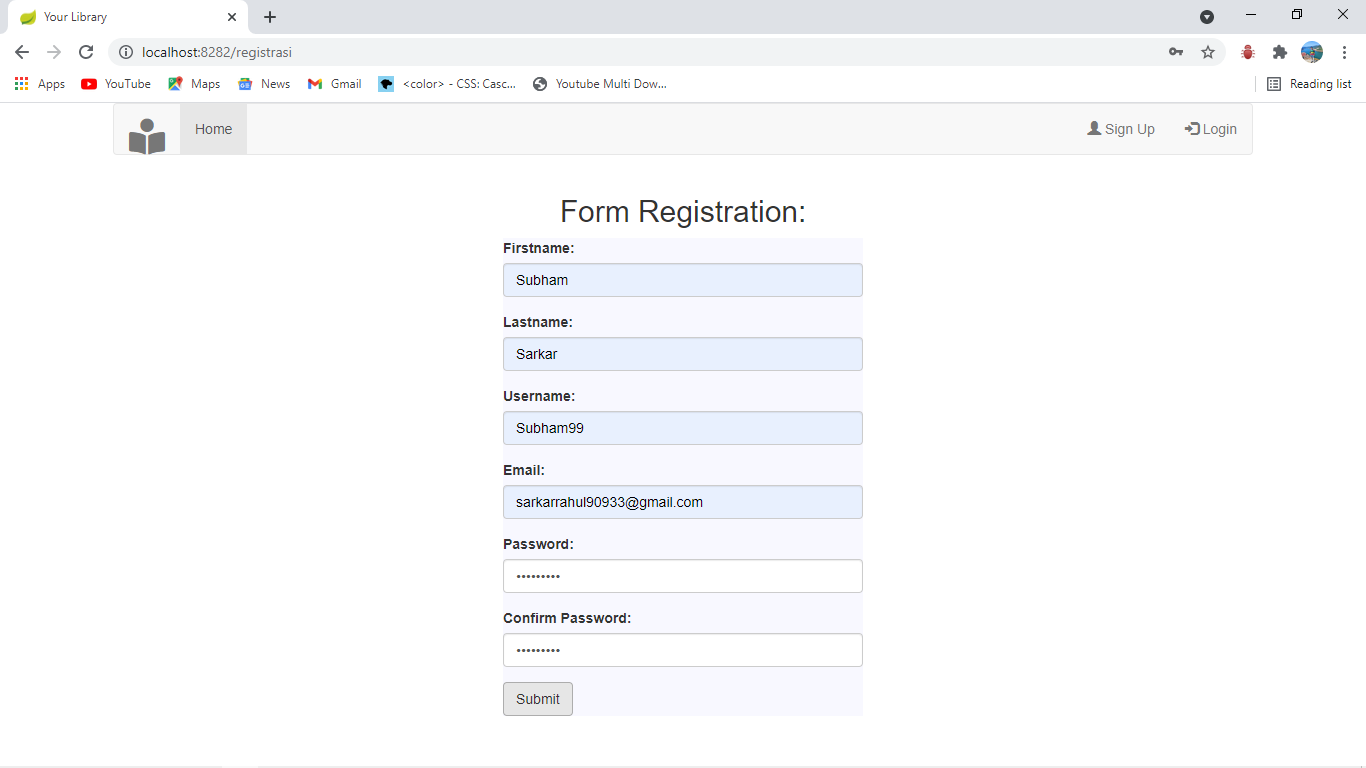
**4.1 SCREEN SHOTS OF HOME PAGE**



@Controller  
@RequestMapping("/")  
public class HomeController {  
  
 @GetMapping  
 public String home(Model model) {  
 return "home";  
 }  
   
 @GetMapping("login")  
 public String login(Model model, HttpServletRequest request) {  
 return "login";  
 }  
   
}

With the help of GetMapping method we can get the home page, when we fire the server port. Here “/” helps to fire the root page of the website which was home page.

**4.2 SCREEN SHOTS OF REGISTRATION PAGE**

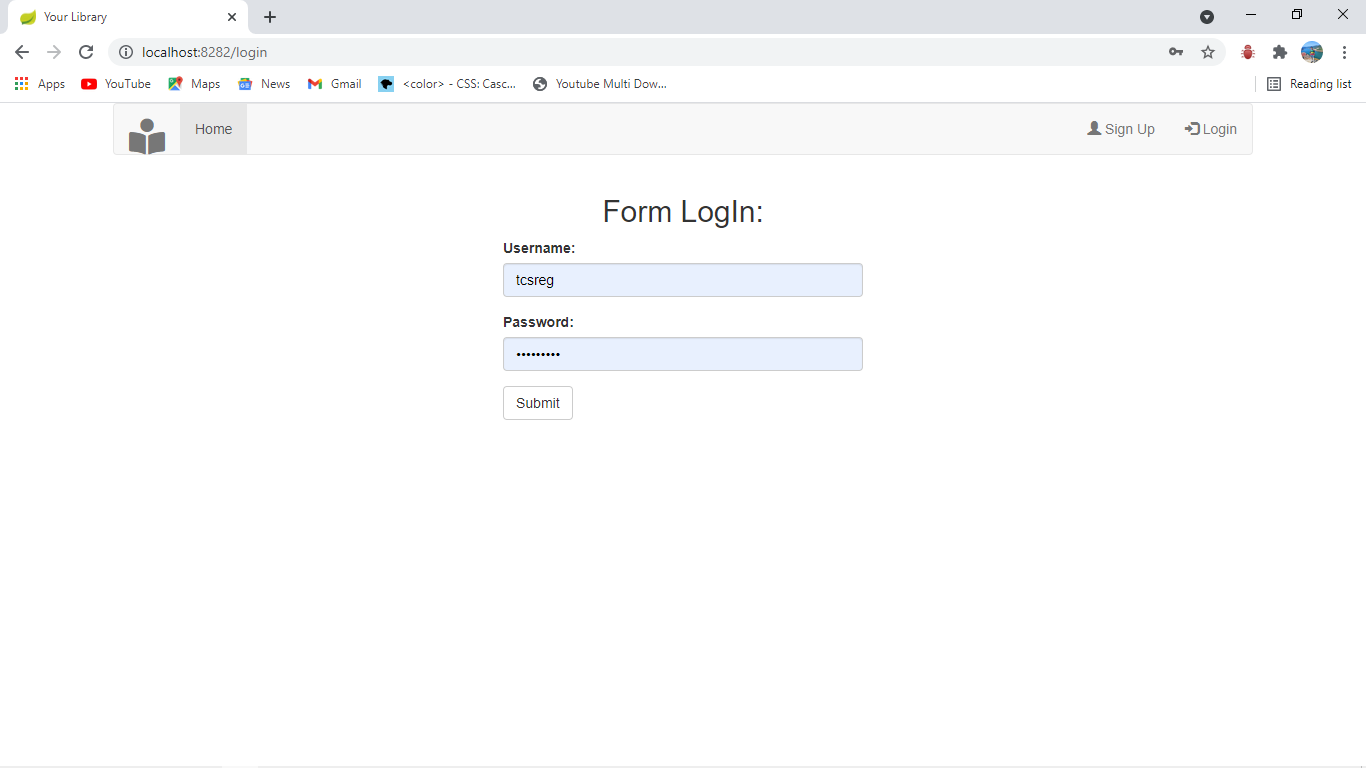
****

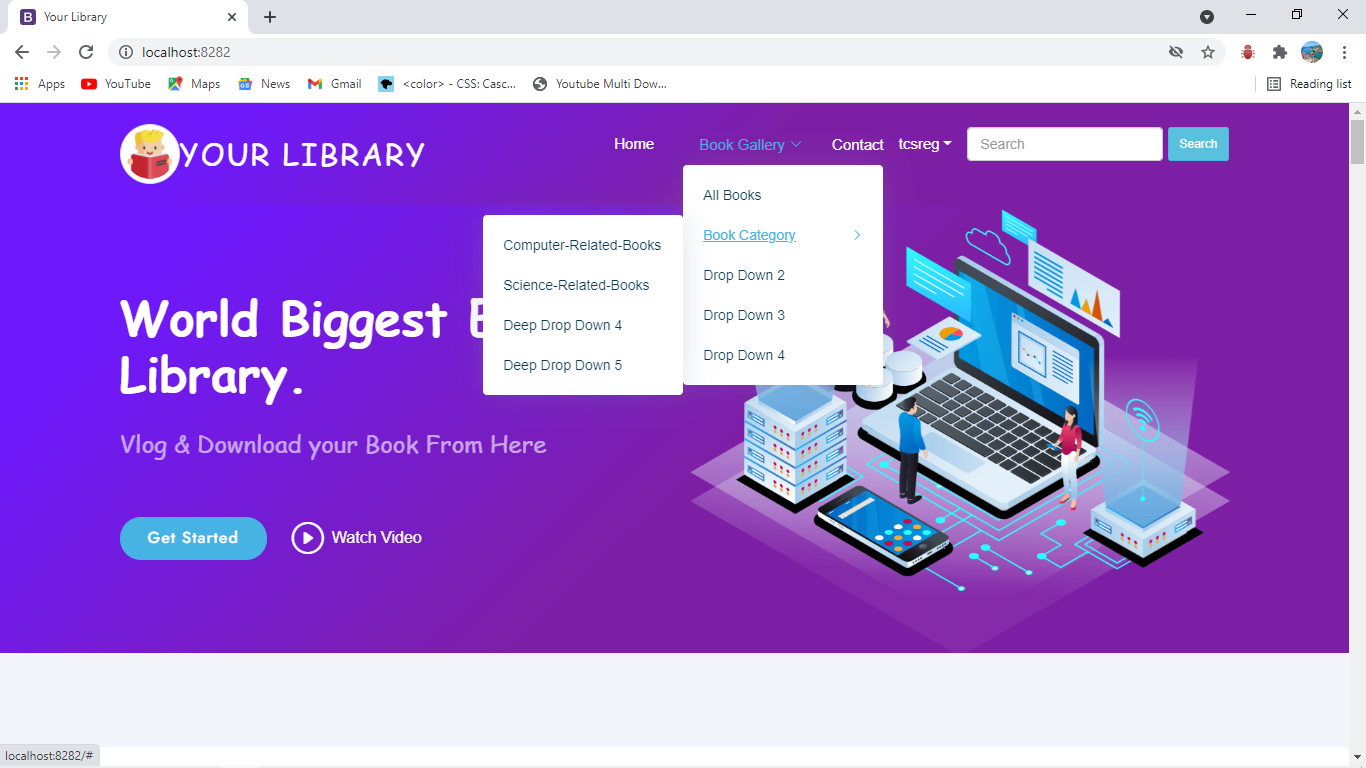
package com.example.subham.dto;  
  
  
import com.example.subham.validator.PasswordMatches;  
import com.example.subham.validator.ValidEmail;  
import com.example.subham.validator.ValidPassword;  
import com.example.subham.validator.ValidUsername;  
  
  
  
import javax.validation.constraints.NotEmpty;  
import javax.validation.constraints.NotNull;  
  
@PasswordMatches  
@ValidPassword  
public class UserDto {  
  
 @NotNull  
 @NotEmpty  
 private String firstname;  
  
 @NotNull  
 @NotEmpty  
 private String lastname;  
  
 @ValidUsername  
 @NotNull  
 @NotEmpty  
 private String username;  
  
 @ValidEmail  
 @NotNull  
 @NotEmpty  
 private String email;  
  
 @NotNull  
 @NotEmpty  
 private String password;  
   
 @NotNull  
 @NotEmpty  
 private String matchingPassword;  
  
 public String getFirstname() {  
 return firstname;  
 }  
  
 public void setFirstname(String firstname) {  
 this.firstname = firstname;  
 }  
  
 public String getLastname() {  
 return lastname;  
 }  
  
 public void setLastname(String lastname) {  
 this.lastname = lastname;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public String getMatchingPassword() {  
 return matchingPassword;  
 }  
  
 public void setMatchingPassword(String matchingPassword) {  
 this.matchingPassword = matchingPassword;  
 }  
  
}

in UserDto class I define all the entities which is required to register a normal user and an admin . following code is describes how I handle user registration,user login page on server.

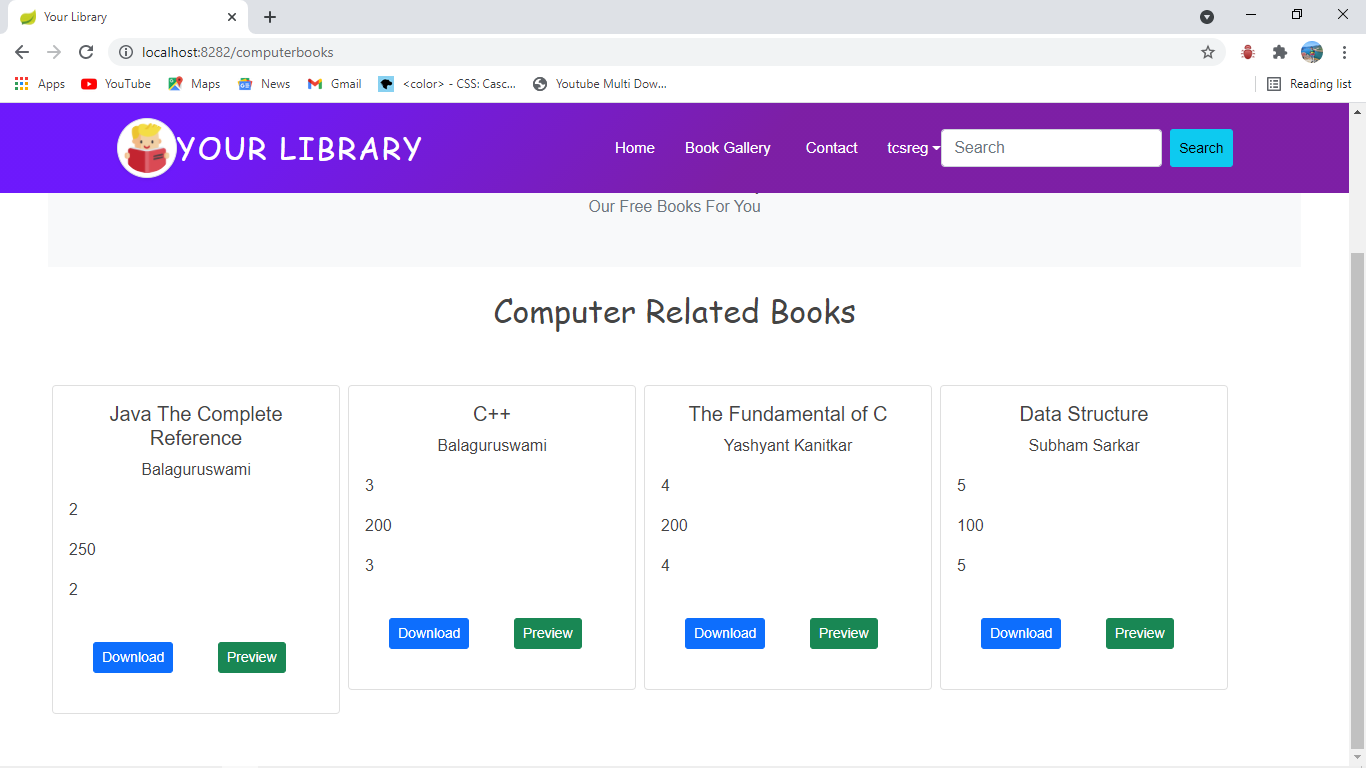
package com.example.subham.controller;  
  
  
import com.example.subham.dto.UserDto;  
import com.example.subham.model.Role;  
import com.example.subham.model.User;  
import com.example.subham.repository.RoleRepository;  
import com.example.subham.repository.UserRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.validation.BindingResult;  
import org.springframework.validation.FieldError;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.servlet.ModelAndView;  
  
import javax.validation.Valid;  
import java.util.HashSet;  
import java.util.Set;  
  
@Controller  
public class UserController {  
  
 @Autowired  
 private UserRepository usersRepository;  
   
 @Autowired  
 private RoleRepository roleRepository;  
   
 @Autowired  
 private BCryptPasswordEncoder encoder;  
   
 @GetMapping("/registrasi")  
 public String registrationForm(UserDto userDto, Model model) {  
 return "registration";  
 }  
   
 @PostMapping("/registrasi")  
 public ModelAndView registerUser(@Valid UserDto userDto, BindingResult result) {  
 ModelAndView view = new ModelAndView("registration");  
 if (result.hasErrors()) {  
 return view;  
 }  
   
 String email = userDto.getEmail();  
 String firstname = userDto.getFirstname();  
 String lastname = userDto.getLastname();  
 String username = userDto.getUsername();  
 String password = userDto.getPassword();  
   
 if (usersRepository.findByUsername(username).isPresent()) {  
 result.addError(new FieldError("userDto", "username", "Username has been used."));  
 return view;  
 }  
   
 if (usersRepository.findByEmail(email).isPresent()) {  
 result.addError(new FieldError("userDto", "email", "Email has been used."));  
 return view;  
 }  
 Set<Role> roles = new HashSet<>();  
 roles.add(roleRepository.findById(2l).get());  
 User user = new User();  
 user.setEmail(email);  
 user.setFirstname(firstname);  
 user.setLastname(lastname);  
 user.setUsername(username);  
 user.setPassword(encoder.encode(password));  
 user.setRoles(roles);  
 usersRepository.save(user);  
   
 return new ModelAndView("success");  
 }  
   
 @GetMapping("/users")  
 public String showUsers(Model model) {  
 model.addAttribute("users", usersRepository.findAll());  
 return "users";  
 }  
}

the above code describes the handling procedure of registration and log in page. With the help of GetMapping and PostMapping we esily post the required data to our database for matching purpose and with the help of GetMapping api we can retrieve that data on server.

**4.3 SCREEN SHOTS OF NORMAL USER PAGE**

****

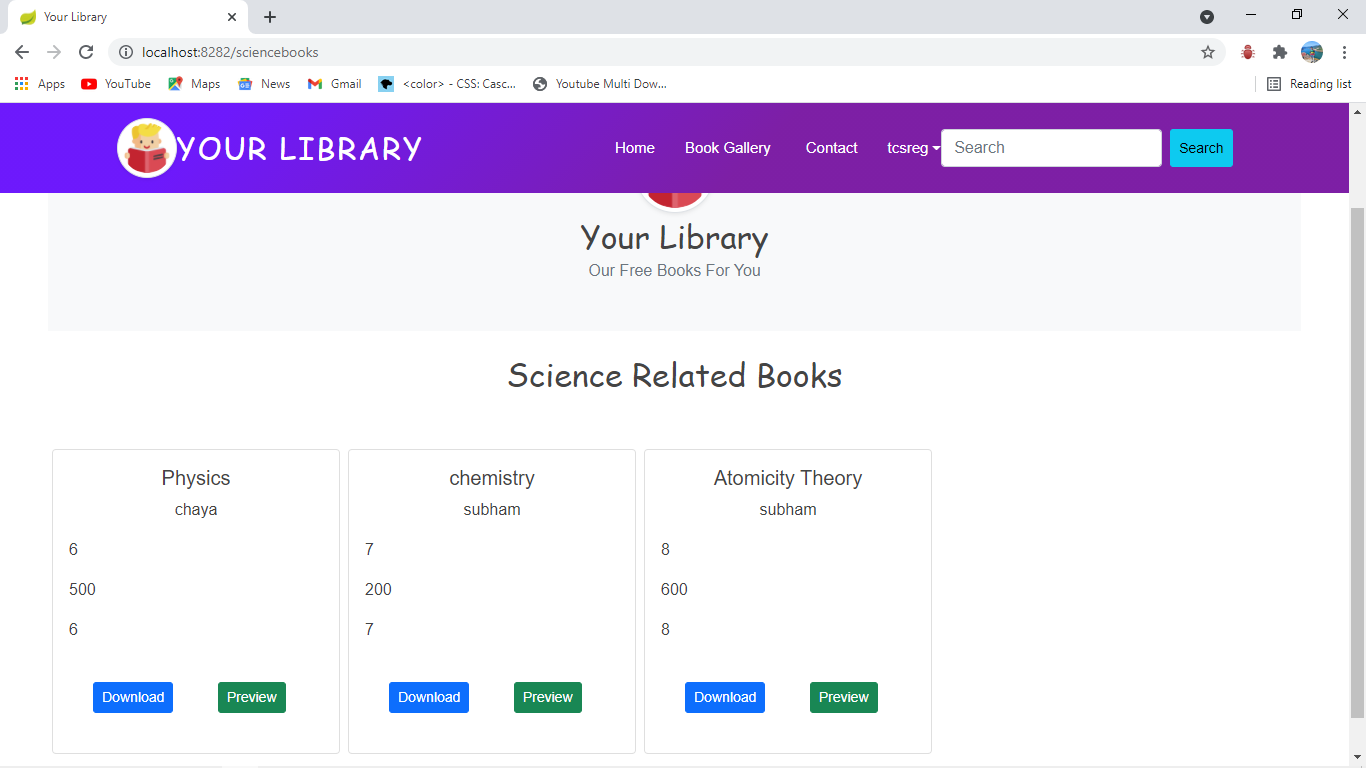
**4.4 SCREEN SHOTS OF COMPUTER-RELATED-BOOKS PAGE**

****

package com.example.subham.controller;  
  
import com.example.subham.model.Books;  
import com.example.subham.model.Computer;  
import com.example.subham.model.Doc;  
import com.example.subham.repository.BooksRepo;  
import com.example.subham.repository.ComputerRepo;  
import com.example.subham.repository.DocRepository;  
import com.example.subham.service.DocStorageService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PostMapping;  
  
import java.util.ArrayList;  
import java.util.List;  
  
@Controller  
public class ComputerBookController {  
 @Autowired  
 BooksRepo booksRepo;  
 @Autowired  
 DocStorageService docStorageService;  
 @Autowired  
 DocRepository docRepository;  
 @Autowired  
 ComputerRepo computerRepo;  
  
 @GetMapping("computerupload")  
 public String computerupload(Model model){  
 ArrayList<Computer> oa = (ArrayList<Computer>) computerRepo.findAll();  
 model.addAttribute("computer",oa);  
 List<Doc> docs2 = docStorageService.getFiles();  
 model.addAttribute("docs", docs2); //attribute name should be same  
  
  
 return "computerupload";  
 }  
  
 @GetMapping("/computerbooks")  
 public String getComputerBookController(Model model)  
 {  
 ArrayList<Computer> oa=(ArrayList<Computer>)computerRepo.findAll();  
 model.addAttribute("computer",oa);  
  
 return "computerbooks";  
 }  
 @PostMapping("/computerbooks")  
 public String SaveBooks(Computer oa){  
  
 oa.getAuthorName(); oa.setAuthorName();oa.getBookName();oa.setBookName();oa.getContent();oa.setContent();oa.getData();oa.setData();oa.getPrice();oa.setPrice();  
 computerRepo.save(oa);  
  
 System.*out*.println(oa);  
 return "doc";  
 }  
  
}

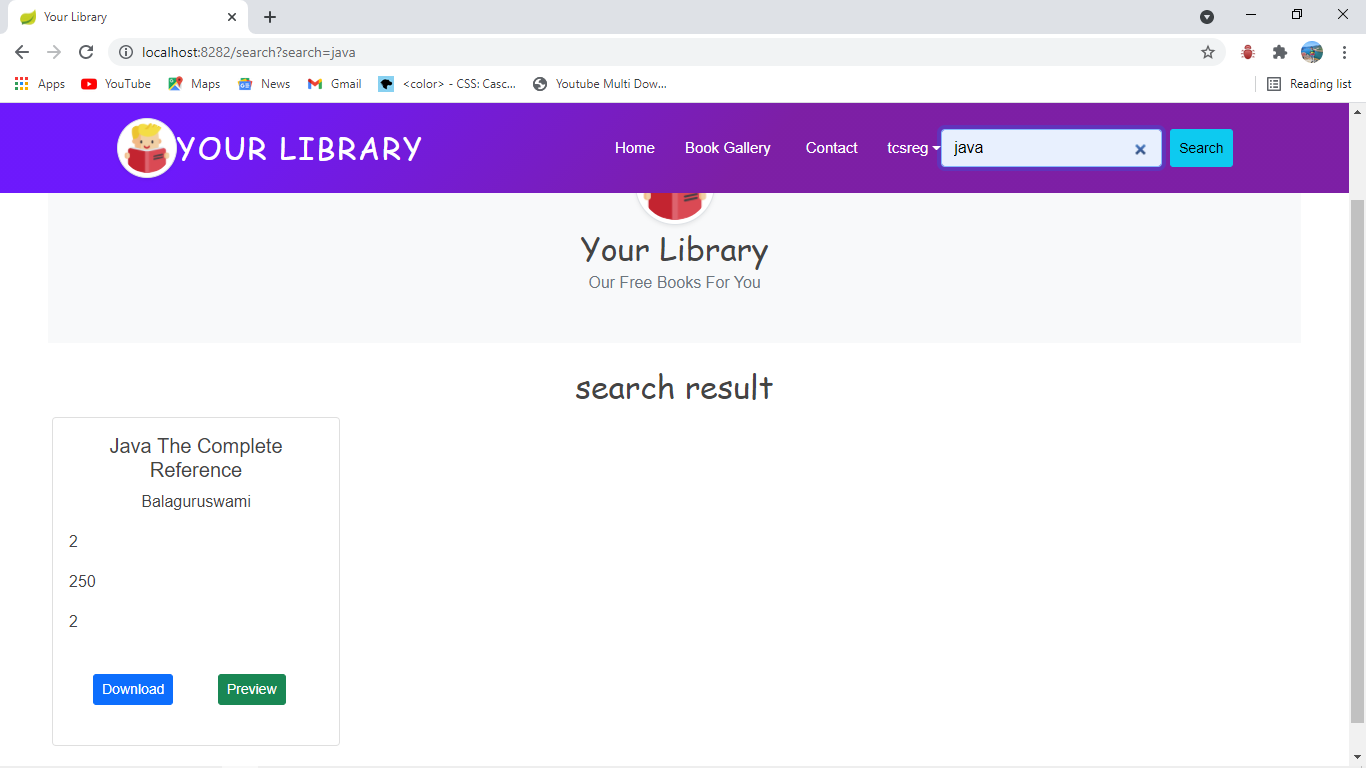
above code is for computer-related-books section.by this code we can get computer books section and view the content of this section, this procedure done by get api’s.

**4.5 SCREEN SHOTS OF SCIENCE-RELATED-BOOKS PAGE**

****

package com.example.subham.controller;  
  
import com.example.subham.model.Computer;  
import com.example.subham.model.Doc;  
import com.example.subham.model.Science;  
import com.example.subham.repository.BooksRepo;  
import com.example.subham.repository.ComputerRepo;  
import com.example.subham.repository.DocRepository;  
import com.example.subham.repository.ScienceRepo;  
import com.example.subham.service.DocStorageService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PostMapping;  
  
import java.util.ArrayList;  
import java.util.List;  
  
@Controller  
public class ScienceBookController {  
 @Autowired  
 BooksRepo booksRepo;  
 @Autowired  
 DocStorageService docStorageService;  
 @Autowired  
 DocRepository docRepository;  
 @Autowired  
 ComputerRepo computerRepo;  
 @Autowired  
 ScienceRepo scienceRepo;  
  
  
 @GetMapping("scienceupload")  
 public String scienceupload(Model model){  
 ArrayList<Science> oc = (ArrayList<Science>) scienceRepo.findAll();  
 model.addAttribute("science",oc);  
 List<Doc> docs2 = docStorageService.getFiles();  
 model.addAttribute("docs", docs2); //attribute name should be same  
  
  
 return "scienceupload";  
 }  
  
 @GetMapping("/sciencebooks")  
 public String getScienceBookController(Model model)  
 {  
 ArrayList<Science> oc=(ArrayList<Science>)scienceRepo.findAll();  
 model.addAttribute("science",oc);  
  
 return "sciencebooks";  
 }  
 @PostMapping("/sciencebooks")  
 public String SaveBooks(Science oc){  
  
 oc.getAuthorName(); oc.setAuthorName();oc.getBookName();oc.setBookName();oc.getContent();oc.setContent();oc.getData();oc.setData();oc.getPrice();oc.setPrice();  
 scienceRepo.save(oc);  
  
 System.*out*.println(oc);  
 return "doc";  
 }  
}

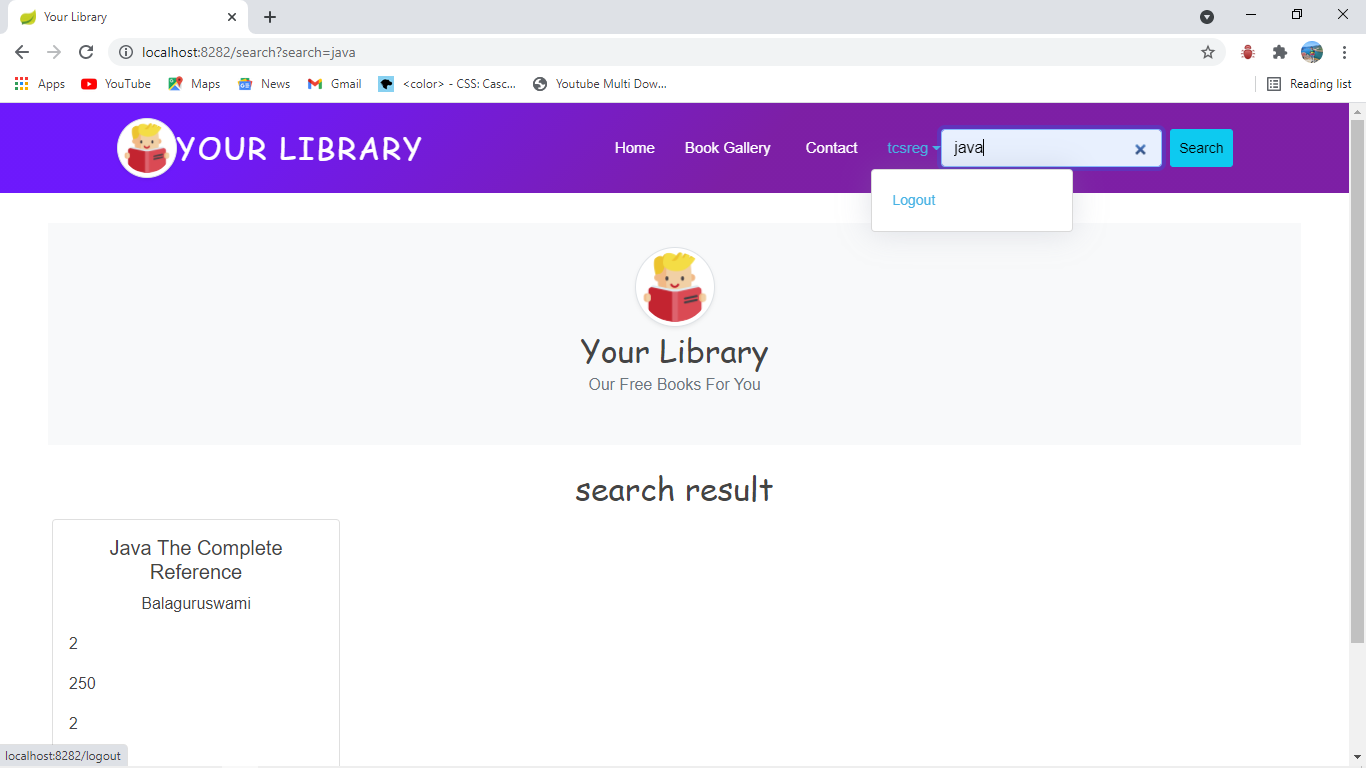
above code is for science-related-books section.by this code we can get computer books section and view the content of this section, this procedure done by get api’s.

**4.5 SCREEN SHOTS OF SEARCH PAGE**

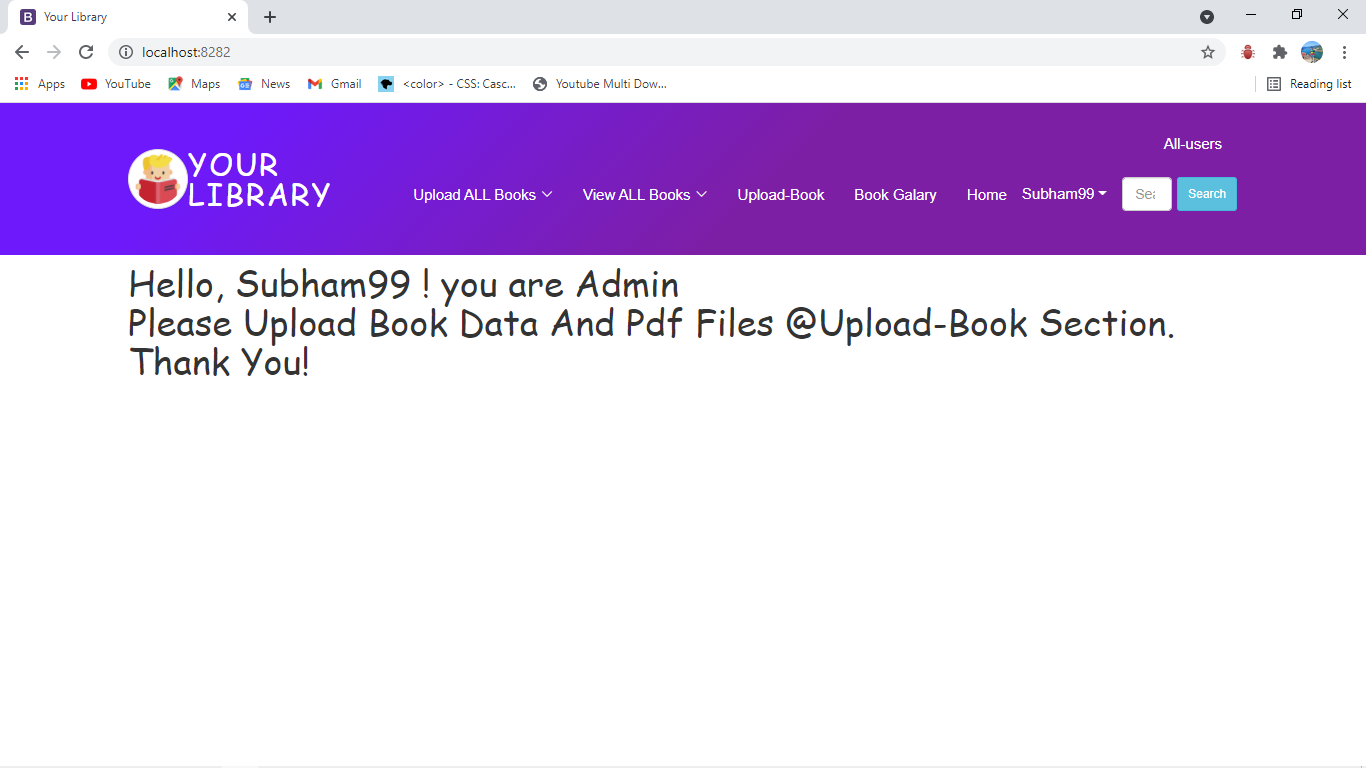
@GetMapping("/search")  
public String Search(HttpServletRequest request, Model m)  
{  
 String name= request.getParameter("search");  
 List<Books> ob=booksRepo.findAllByBookNameContaining(name);  
 m.addAttribute("data",ob);  
 List<Computer> oa=computerRepo.findAllByBookNameContaining(name);  
 m.addAttribute("computer",oa);  
 List<Science> oc=scienceRepo.findAllByBookNameContaining(name);  
 m.addAttribute("science",oc);  
 List<Doc> docs2 = docStorageService.getFiles();  
 m.addAttribute("docs", docs2);  
  
  
 System.*out*.println(ob);  
  
 return "search";  
  
}

above code is for search section for both user and admin.by this code we can get search result and view can the content of this section, this procedure done by get api’s.

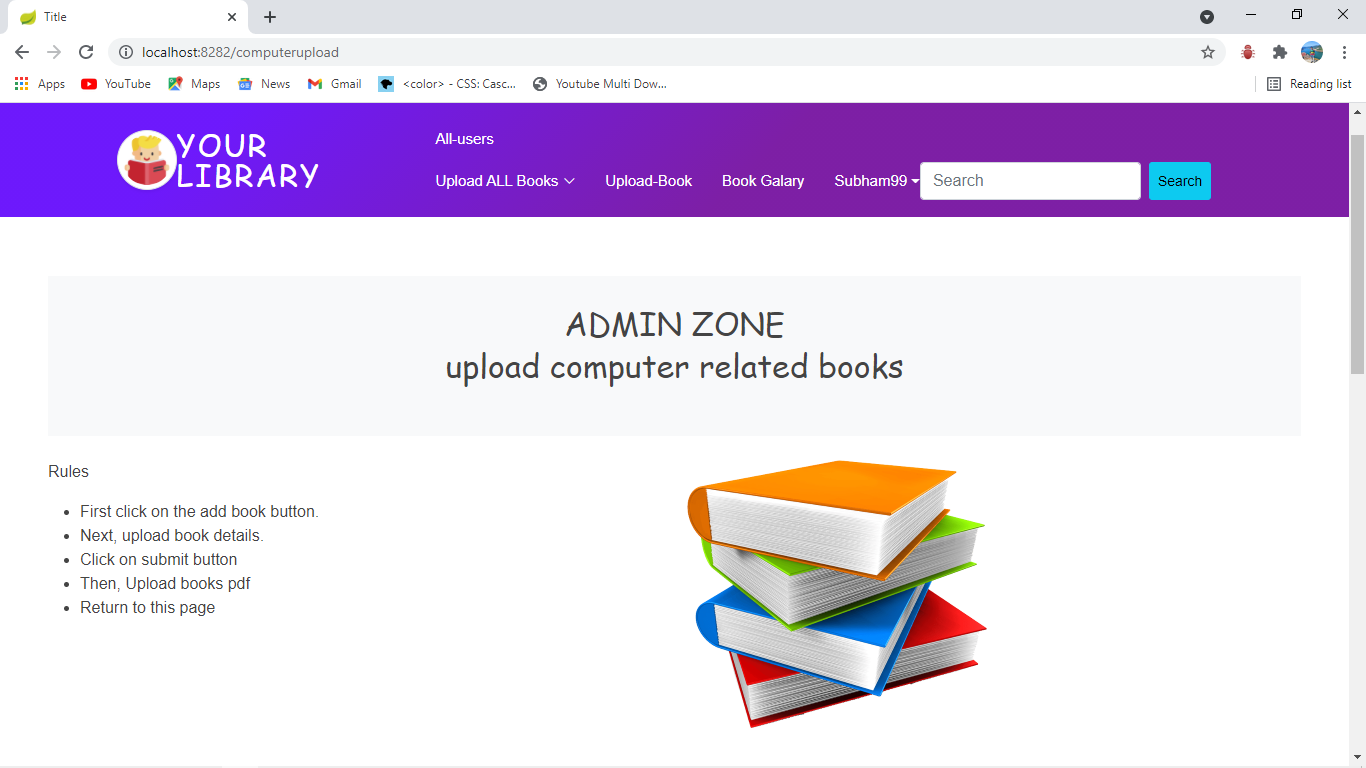
**4.6 SCREEN SHOTS OF USER LOGOUT**



**4.7 SCREEN SHOTS OF ADMIN PAGE**

****

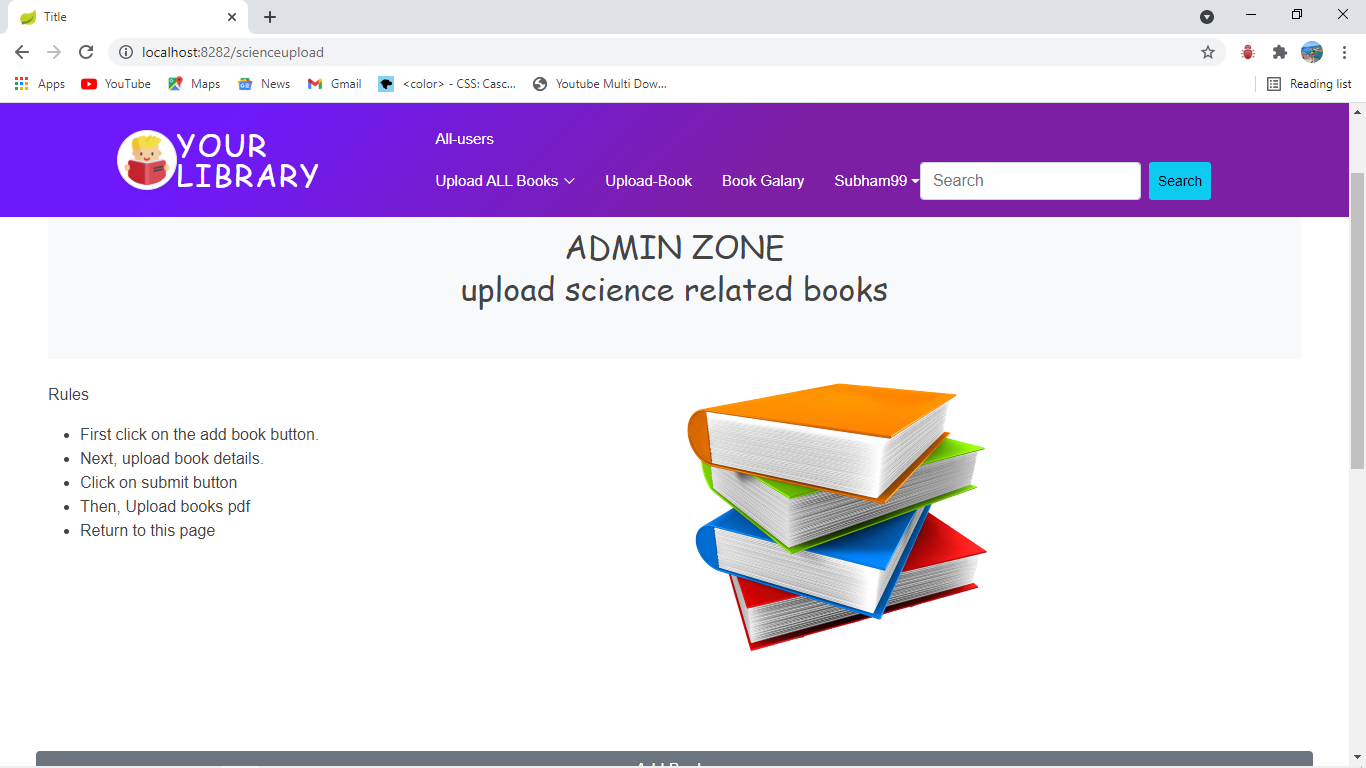
**4.7 SCREEN SHOTS OF UPLOAD COMPUTER BOOK PAGE BY ADMIN**

****

@PostMapping("/computerbooks")  
public String SaveBooks(Computer oa){  
  
 oa.getAuthorName(); oa.setAuthorName();oa.getBookName();oa.setBookName();oa.getContent();oa.setContent();oa.getData();oa.setData();oa.getPrice();oa.setPrice();  
 computerRepo.save(oa);  
  
 System.*out*.println(oa);  
 return "doc";  
}

Using post mapping we post a new computer related book on compueter book section.

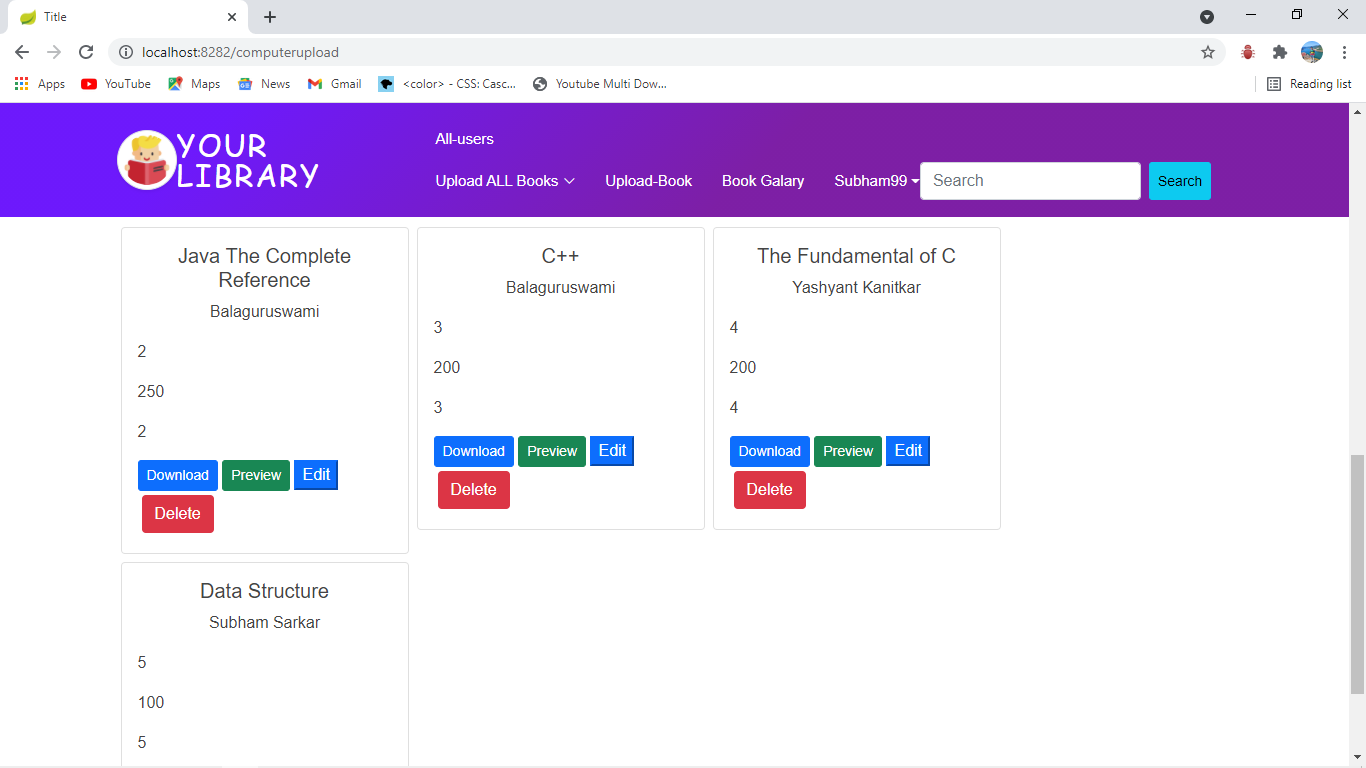
**4.7 SCREEN SHOTS OF UPLOAD SCIENCE BOOK PAGE BY ADMIN**

****

@PostMapping("/sciencebooks")  
public String SaveBooks(Science oc){  
  
 oc.getAuthorName(); oc.setAuthorName();oc.getBookName();oc.setBookName();oc.getContent();oc.setContent();oc.getData();oc.setData();oc.getPrice();oc.setPrice();  
 scienceRepo.save(oc);  
  
 System.*out*.println(oc);  
 return "doc";  
}

Using post mapping we post a new sciencerelated book on science book section.

**4.8 SCREEN SHOTS OF DELETING, EDITING SEVERAL BOOKS BY ADMIN**

****

@GetMapping("/delete/{id}")  
public String DeleteBooks(@PathVariable("id") int id){  
 booksRepo.deleteById(id);  
 computerRepo.deleteById(id);  
 scienceRepo.deleteById(id);  
 docRepository.deleteById(id);  
 return "redirect:/admin";  
}

@GetMapping("/EditBooks")  
public String SaveBooks( Books ob){  
 System.*out*.println("hi");  
 System.*out*.println(ob.toString());  
 return "hi";  
}

**CHAPTER 5**

**SYSTEM TESTING**

The aim of the system testing process was to determine all defects in our project .The

program was subjected to a set of test inputs and various observations were made and based

on these observations it will be decided whether the program behaves as expected or not. Our

Project went through two levels of testing

1.Unit testing

2.integration testing

**UNIT TESTING**

1. Test For the admin module

**** Testing admin login form-This form is used for log in of administrator of the system. In this we enter the username and password if both are correct administration page will open otherwise if any of data is wrong it will get redirected back to the login page and again ask for username and password****

****

* Book upload- Admin can enter details of book and can add the details to the several book tables also he can view the books requests .
* Testing delete button and edit button by which admin can delete or edit a book from database

2. Test for Normal User login module

**** Test for normal user login Form-This form is used for log in of normal users. In this we enter username and password if all these are correct users page will open other wise if any of data is wrong it will get redirected back to the login page and again ask for username and password.

****

**** Test for account creation- This form is used for new account creation when users does not fill the form completely it asks again to fill the whole form when he fill the form fully it gets redirected to page which show registration successful message.

**INTEGRATION TESTING**

In this type of testing we test various integration of the project module by providing the input

.The primary objective is to test the module interfaces in order to ensure that no errors are occurring when one module invokes the other module.

****

**CHAPTER 6**

**CONCLUSION & FUTURE SCOPE**

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

It makes entire process online where student can search books, and download book pdf. It also has a facility for users login where student can login and register themsels. It has a facility of admin login where admins can add lectures notes.

There is a future scope of this facility that many more features such as online lectures video tutorials can be added by admins as well as online assignments submission facility , thus making it more interactive more user friendly and project which fulfills each users need in the best way possible.

**CHAPTER 7**

**REFERENCES**

* <http://www.w3schools.com/html/html_intro.asp>
* <http://spring.io/guides/gs/spring-boot/>
* [http://spring.io/guides/tutorials /spring-boot-kotlin/](http://spring.io/guides/tutorials%20/spring-boot-kotlin/)
* <http://www.springboottutorial.com/>
* <https://www.youtube.com/watch?v=ZJ7afDSrb3s&list=PLGRDMO4rOGcN21sn_D3F1WOa8O38c_10R>
* https://www.youtube.com/watch?v=729Pd-ZQ4uA&list=PL0zysOflRCelmjxj-g4jLr3WKraSU\_e8q

