**Final Project Report: Library Management System**

**Summary:**LibraryApp, a library management system, is a Spring MVC web application that allows the communication between the library admin and library user. Hibernate is used to map the POJOs to the MySQL database.

The system is fully menu driven and easy to use. It provides quick and accurate information about all the books present in the library, the members who have lent the book, issued and return status of book and the availability of the book.

**Functionality:**

Admin Role:

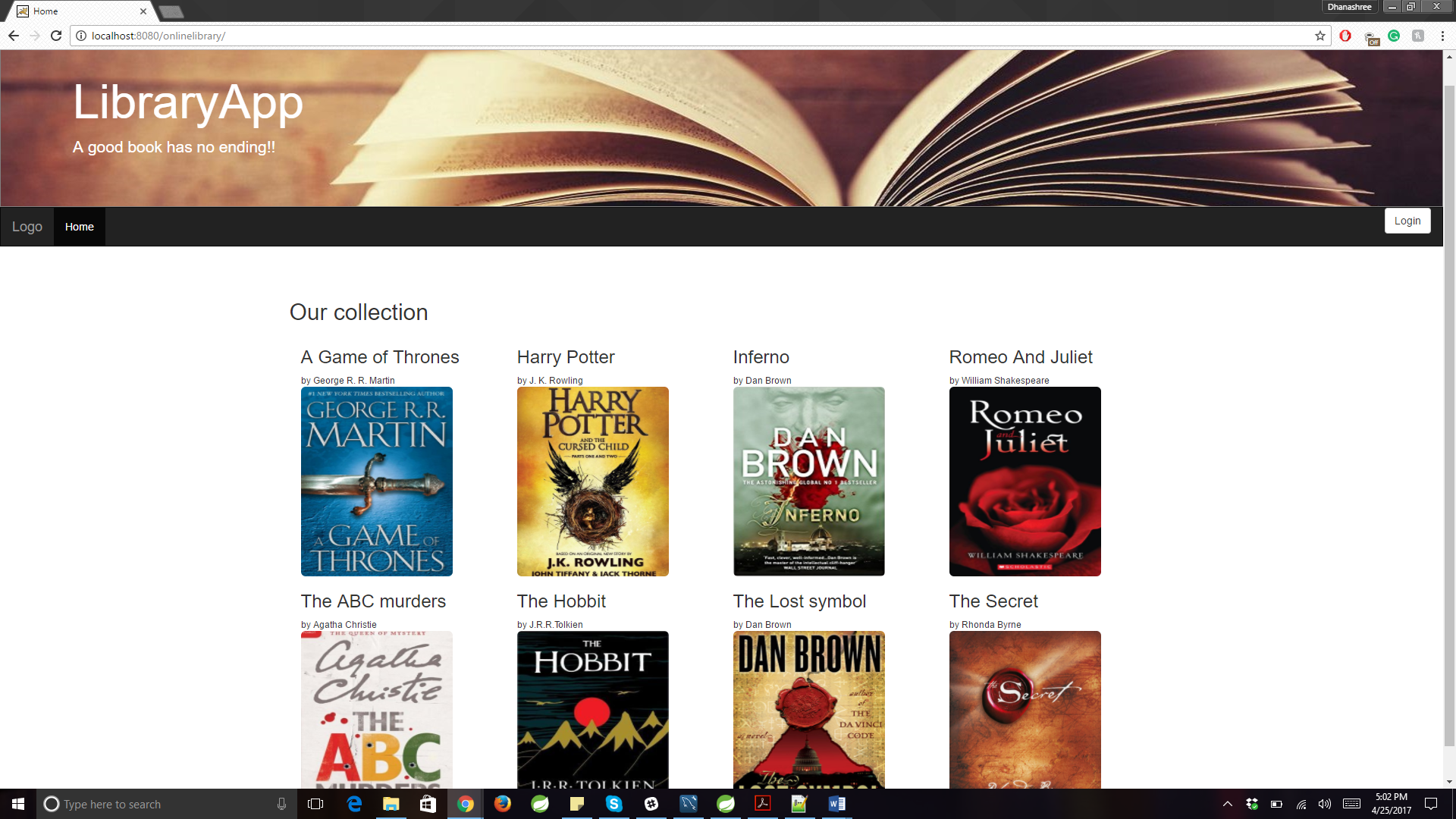
1. Add new books to the library
2. Add new members to the library
3. Approve Book issue requests
4. Handle return requests

Library Member Role:

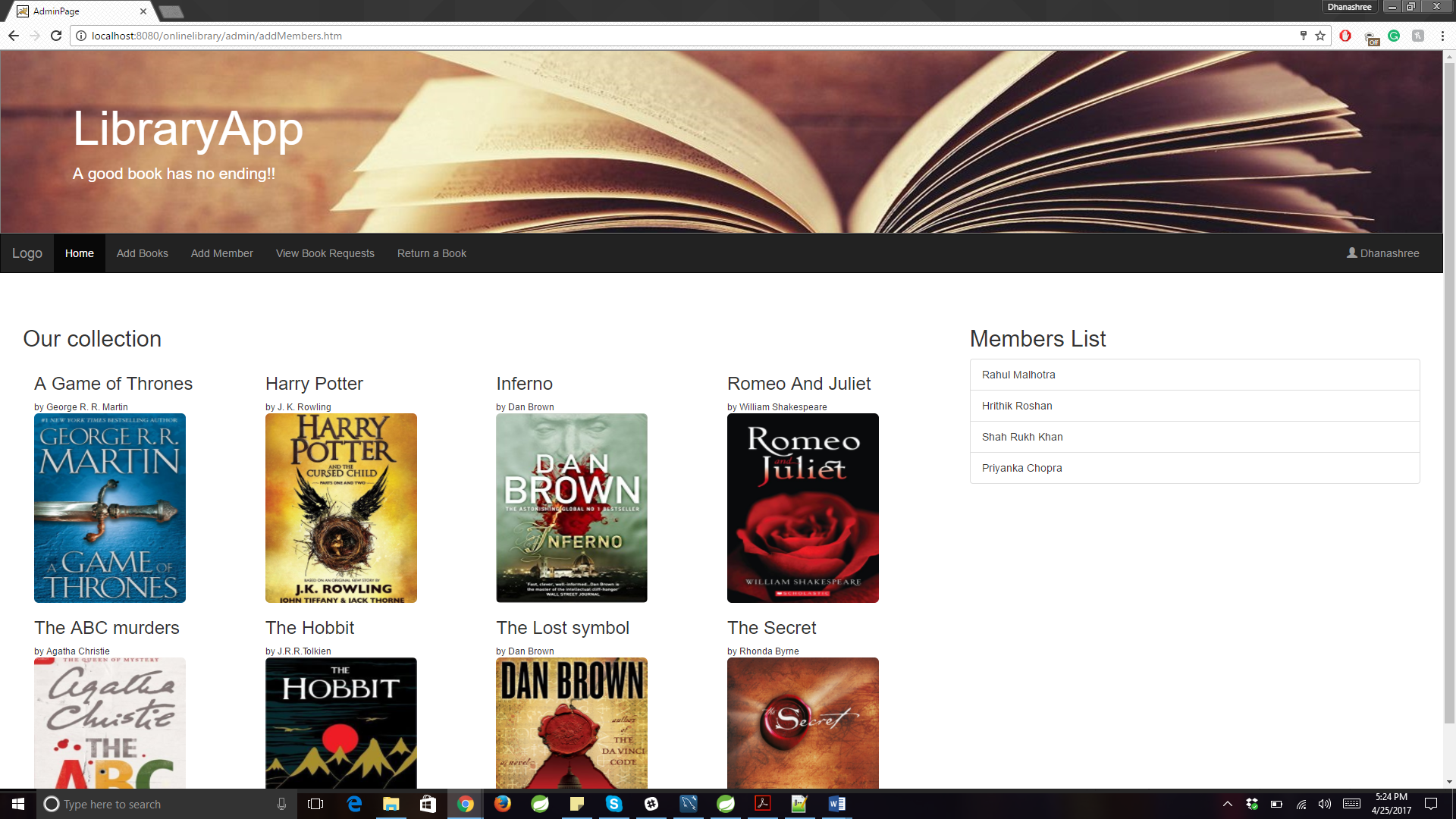
1. Raise a book issue request
2. View the available books
3. Check the status of the book issue requests
4. View all the books issued under his membership

**Screenshots:**

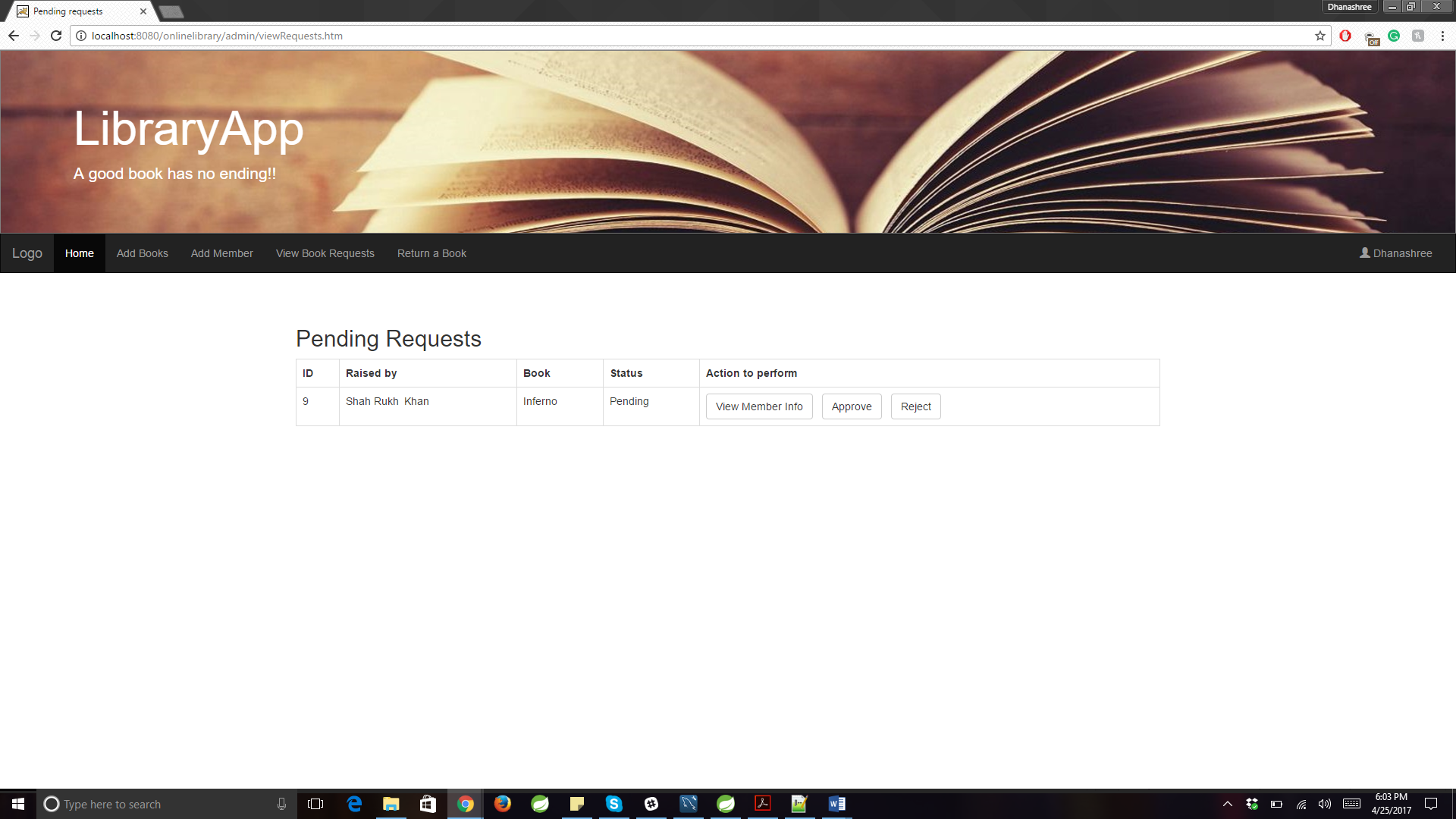
Main Screen:



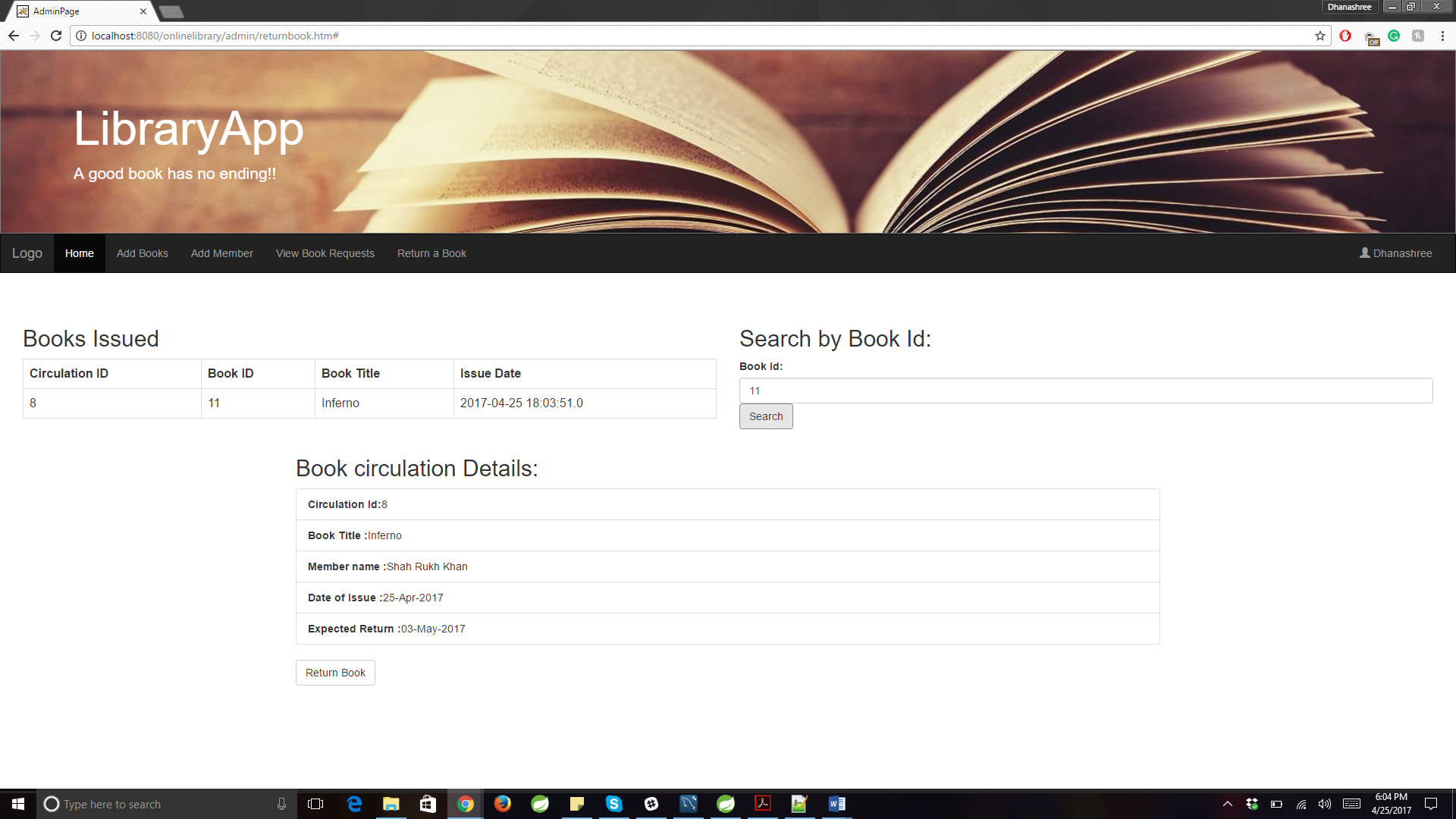
AdminHome:



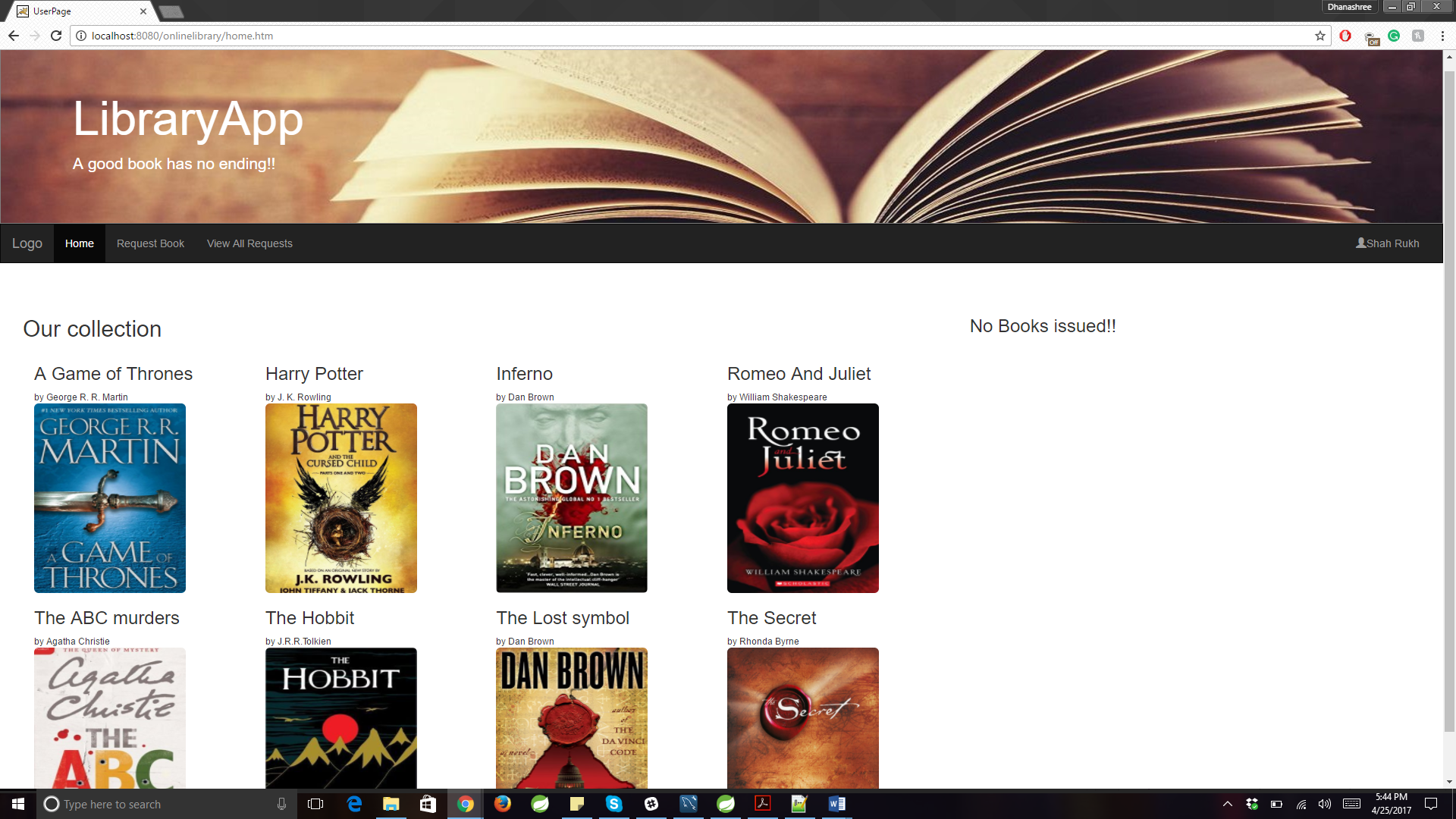
**Admin-approve book issue request:**



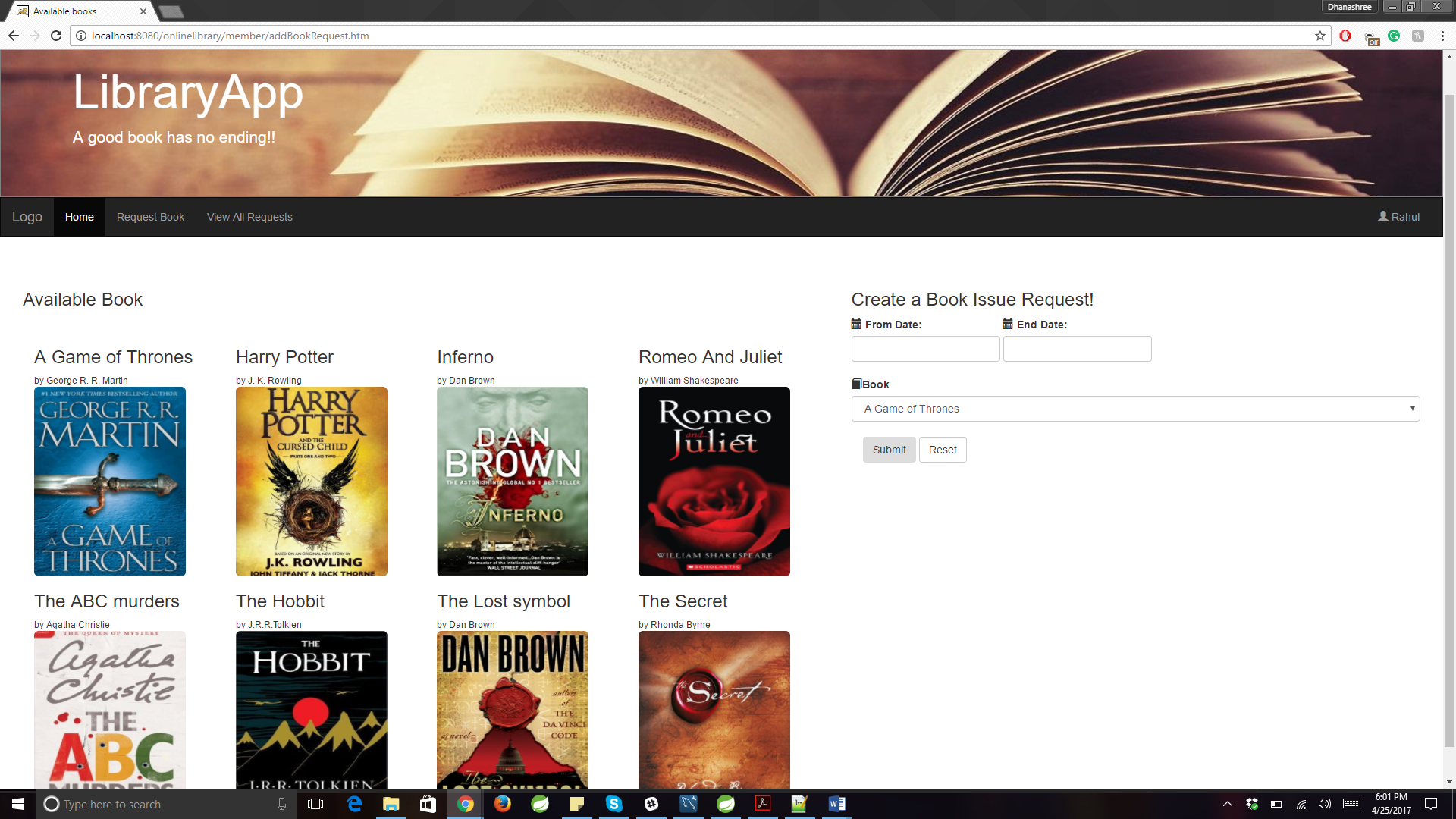
**Admin-Return a book:**



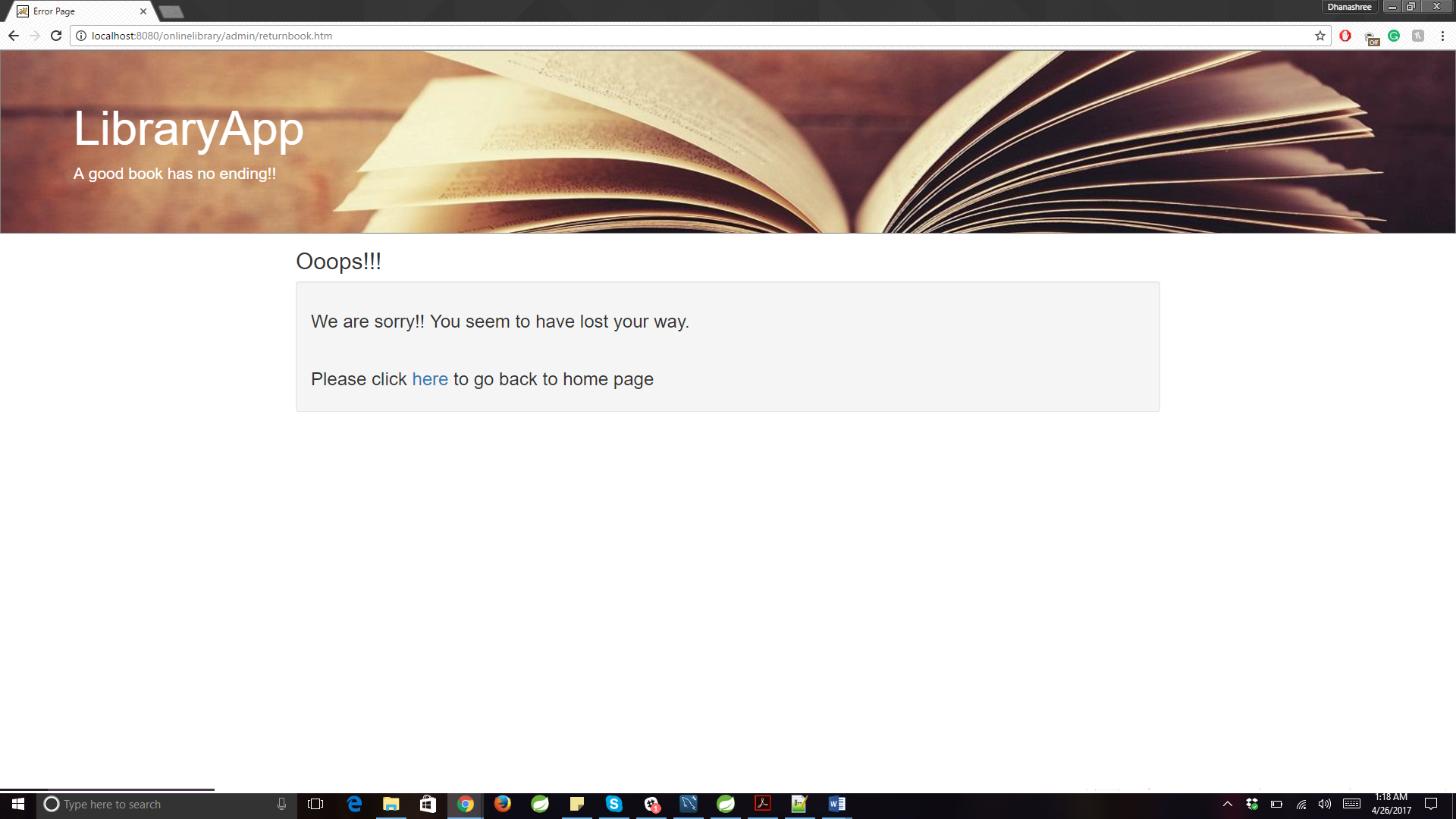
**UserHome:**



**User-Add book issue request:**



**Unauthorized Access:**



**MainController.java**

package com.my.onlinelibrary.controller;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.servlet.ModelAndView;

import com.my.onlinelibrary.dao.BookDAO;

import com.my.onlinelibrary.dao.CirculationDAO;

import com.my.onlinelibrary.dao.UserDAO;

import com.my.onlinelibrary.exception.BookException;

import com.my.onlinelibrary.exception.UserException;

import com.my.onlinelibrary.pojo.Book;

import com.my.onlinelibrary.pojo.BookCirculation;

import com.my.onlinelibrary.pojo.LibraryMember;

import com.my.onlinelibrary.pojo.LibraryUsers;

@Controller

public class MainController {

@Autowired

@Qualifier("bookDAO")

BookDAO bookDAO;

@Autowired

@Qualifier("userDAO")

UserDAO userDAO;

@Autowired

@Qualifier("circulationDAO")

CirculationDAO bookCirculationDAO;

@RequestMapping(value = "/", method = RequestMethod.GET)

public ModelAndView home(HttpServletRequest request) {

request.getSession().invalidate();

ModelAndView model = new ModelAndView("home");

List<Book> books;

try {

books = bookDAO.listAllDistinctBooks();

model.addObject("books", books);

} catch (BookException e) {

e.printStackTrace();

}

return model;

}

@RequestMapping(value="/login", method = RequestMethod.POST)

public ModelAndView checkLogin(HttpServletRequest request){

String username = request.getParameter("username");

String password = request.getParameter("password");

HttpSession session = (HttpSession) request.getSession();

ModelAndView result = new ModelAndView();

try {

List<Book> books = bookDAO.getAvailableBooks();

session.setAttribute("books", books);

} catch (BookException e1) {

e1.printStackTrace();

}

try {

System.out.print("loginUser");

LibraryUsers user = userDAO.checkCredentials(username, password);

if(user == null){

System.out.println("UserName/Password does not exist");

result.addObject("errorMessage", "UserName/Password does not exist");

result.setViewName("home");

}

else if(user.getAuthority().getRole().equalsIgnoreCase("ROLE\_ADMIN")){

session.setAttribute("user", user);

List<LibraryMember> members = userDAO.getAllLibraryMembers();

session.setAttribute("members", members);

result.setViewName("admin-home");

}else if(user.getAuthority().getRole().equalsIgnoreCase("ROLE\_MEMBER")){

session.setAttribute("user", user);

LibraryMember mem = userDAO.getLibraryMember(user);

List<BookCirculation> bookCirculationList = bookCirculationDAO.getBooksInCirculation(mem);

session.setAttribute("bookcirculationlist", bookCirculationList);

result.setViewName("user-home");

}

return result;

} catch (UserException e) {

// TODO Auto-generated catch block

e.printStackTrace();

result.addObject("someerror", "Some error occured");

result.setViewName("error-page");

return result;

}

}

@RequestMapping(value={"/home.htm","/login"}, method = RequestMethod.GET)

public String returnHome(HttpServletRequest request, Model model){

HttpSession session = request.getSession();

LibraryUsers u = (LibraryUsers) session.getAttribute("user");

if(u == null){

return "unauthorizedaccess";

}

if(u.getAuthority().getRole().equalsIgnoreCase("ROLE\_ADMIN")){

return "admin-home";

}else{

return "user-home";

}

}

}

**BookController.java**

package com.my.onlinelibrary.controller;

import java.io.File;

import java.io.IOException;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import javax.servlet.ServletContext;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.stereotype.Controller;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.WebDataBinder;

import org.springframework.web.bind.annotation.InitBinder;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.multipart.commons.CommonsMultipartFile;

import org.springframework.web.servlet.ModelAndView;

import com.my.onlinelibrary.dao.BookDAO;

import com.my.onlinelibrary.dao.CirculationDAO;

import com.my.onlinelibrary.dao.UserDAO;

import com.my.onlinelibrary.exception.BookException;

import com.my.onlinelibrary.json.JsonUtils;

import com.my.onlinelibrary.pojo.Book;

import com.my.onlinelibrary.pojo.BookCirculation;

import com.my.onlinelibrary.pojo.BookReservation;

import com.my.onlinelibrary.pojo.LibraryMember;

import com.my.onlinelibrary.pojo.LibraryUsers;

import com.my.onlinelibrary.validator.BookReservationValidator;

import com.my.onlinelibrary.validator.BookValidator;

@Controller

public class BookController {

@Autowired

@Qualifier("bookDAO")

BookDAO bookDAO;

@Autowired

@Qualifier("userDAO")

UserDAO userDAO;

@Autowired

@Qualifier("circulationDAO")

CirculationDAO bookCirculationDAO;

@Autowired

@Qualifier("bookValidator")

BookValidator validator;

@Autowired

@Qualifier("bookreservationValidator")

BookReservationValidator brvalidator;

@InitBinder("book")

private void initBookBinder(WebDataBinder binder) {

binder.setValidator(validator);

}

@InitBinder("reserveBook")

private void initReservationBinder(WebDataBinder binder) {

binder.setValidator(brvalidator);

}

@Autowired

ServletContext servletContext;

private static String UPLOAD\_LOCATION = "C:/images/books";

// display Add new books page to admin

@RequestMapping(value = "/admin/addBooks.htm", method = RequestMethod.GET)

protected ModelAndView addBooksForm() throws Exception {

return new ModelAndView("add-books", "book", new Book());

}

// add new books to the system

@RequestMapping(value = "/admin/addBooks.htm", method = RequestMethod.POST)

public String handleUpload(HttpServletRequest request, @ModelAttribute("book") Book book, BindingResult result) {

validator.validate(book, result);

if (result.hasErrors()) {

return "add-books";

}

HttpSession session = request.getSession();

try {

int countOfBooks = Integer.parseInt(request.getParameter("totalbooks"));

CommonsMultipartFile photoInMemory = book.getPhoto();

String fileName = photoInMemory.getOriginalFilename();

File localFile = new File(UPLOAD\_LOCATION + File.separator + fileName);

photoInMemory.transferTo(localFile);

book.setFilename(File.separator + "images" + File.separator + fileName);

System.out.println(book.getFilename());

System.out.println("File is stored at" + localFile.getPath());

System.out.print("added new Book");

List<Book> books = new ArrayList<Book>();

for (int i = 0; i < countOfBooks; i++) {

System.out.println(i + " " + countOfBooks);

books.add(book);

}

System.out.println(books.size());

bookDAO.addBooks(books);

List<Book> updatedbooks = bookDAO.listAllDistinctBooks();

session.setAttribute("books", updatedbooks);

} catch (IllegalStateException e) {

System.out.println("\*\*\* IllegalStateException: " + e.getMessage());

} catch (IOException e) {

System.out.println("\*\*\* IOException: " + e.getMessage());

} catch (BookException e) {

e.printStackTrace();

}

return "admin-home";

}

@RequestMapping(value = "/admin/returnbook.htm", method = RequestMethod.GET)

protected ModelAndView returnBookform() throws Exception {

List<BookCirculation> result = bookCirculationDAO.getAllBooksInCirculation();

return new ModelAndView("return-book", "bookcirculationlist", result);

}

@RequestMapping(value = "/admin/getBookCirculation.htm", method = RequestMethod.POST, produces = "application/json")

@ResponseBody

protected String getBookbyID(HttpServletRequest request, @RequestParam String bookid) throws Exception {

String result;

System.out.println("Inside getBookid" + bookid);

long bookId = Long.parseLong(bookid);

BookCirculation bc = bookCirculationDAO.getBookCirculationByBookID(bookId);

if (bc == null) {

result = "{" + JsonUtils.toJsonField("errormessage", "No circulation request found") + "}";

return result;

}

request.getSession().setAttribute("bookCirculation", bc);

DateFormat dateFormat = new SimpleDateFormat("dd-MMM-yyyy");

result = "{" + JsonUtils.toJsonField("circulationId", Long.toString(bc.getCirculationId()))

+ (", " + JsonUtils.toJsonField("booktitle", bc.getBook().getTitle()))

+ (", " + JsonUtils.toJsonField("libraryuserfirstname",

bc.getLibraryMember().getLibraryUser().getFirstName()))

+ (", " + JsonUtils.toJsonField("libraryuserlastname",

bc.getLibraryMember().getLibraryUser().getLastName()))

+ (", " + JsonUtils.toJsonField("issuedate", dateFormat.format(bc.getIssueDate())))

+ (", " + JsonUtils.toJsonField("expecteddate", dateFormat.format(bc.getExpectedDate()))) + "}";

return result;

}

@RequestMapping(value = "/admin/confirmbookreturn.htm", method = RequestMethod.GET)

public ModelAndView confirmBookReturn(HttpServletRequest request) throws Exception {

System.out.println("Inside Return book");

HttpSession session = request.getSession();

BookCirculation bc = (BookCirculation) session.getAttribute("bookCirculation");

BookCirculation updatedbc = bookCirculationDAO.getBookCirculationByBookID(bc.getBook().getBookId());

Book b = updatedbc.getBook();

b.setAvailable(true);

bookDAO.updateBookAvailability(b);

if (updatedbc.getExpectedDate().before(new Date())) {

LibraryMember mem = updatedbc.getLibraryMember();

mem.setLatesubmission(mem.getLatesubmission() + 1);

userDAO.updateLibraryMember(mem);

}

updatedbc.setReturnDate(new Date());

bookCirculationDAO.updateBookCirculation(updatedbc);

return new ModelAndView("return-book", "successmessage", "Book Returned successfully");

}

@RequestMapping(value = "/member/addBookRequest.htm", method = RequestMethod.GET)

public ModelAndView getAvailableBooks(HttpServletRequest request) throws Exception {

List<Book> availableBooks = new ArrayList<Book>();

try {

availableBooks = bookDAO.getAvailableBooks();

} catch (BookException e) {

e.printStackTrace();

}

ModelAndView model = new ModelAndView();

model.addObject("availableBooks", availableBooks);

BookReservation br = new BookReservation();

LibraryUsers lb = (LibraryUsers) request.getSession().getAttribute("user");

br.setLibrarymember(userDAO.getLibraryMember(lb));

model.addObject("reserveBook", br);

model.setViewName("books-available");

return model;

}

@RequestMapping(value = "/member/addBookRequest.htm", method = RequestMethod.POST)

public ModelAndView registerBookRequest(HttpServletRequest request,

@ModelAttribute("reserveBook") BookReservation reserveBook, BindingResult result) throws Exception {

HttpSession session = request.getSession();

brvalidator.validate(reserveBook, result);

if (result.hasErrors()) {

return new ModelAndView("books-available", "reserveBook", reserveBook);

}

ModelAndView model = new ModelAndView();

LibraryUsers u = (LibraryUsers) session.getAttribute("user");

LibraryMember mem = userDAO.getLibraryMember(u);

reserveBook.setLibrarymember(mem);

Long bookId = Long.parseLong(request.getParameter("bookid"));

Book b = bookDAO.getBookRequested(bookId);

b.setAvailable(false); // setting the book availability as false

bookDAO.updateBookAvailability(b);

reserveBook.setBook(b);

BookReservation r = bookDAO.createReservation(reserveBook);

model.addObject("requestedBook", r.getBook());

model.addObject("requestNumber", r);

List<BookCirculation> bookCirculationList = bookCirculationDAO.getBooksInCirculation(mem);

session.setAttribute("bookcirculationlist", bookCirculationList);

model.setViewName("user-home");

return model;

}

}

**MemberController.java**

package com.my.onlinelibrary.controller;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.stereotype.Controller;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.WebDataBinder;

import org.springframework.web.bind.annotation.InitBinder;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.servlet.ModelAndView;

import com.my.onlinelibrary.dao.UserDAO;

import com.my.onlinelibrary.exception.UserException;

import com.my.onlinelibrary.pojo.LibraryMember;

import com.my.onlinelibrary.validator.UserValidator;

@Controller

public class MemberController {

@Autowired

@Qualifier("userDAO")

UserDAO userDAO;

@Autowired

@Qualifier("userValidator")

UserValidator userValidator;

@InitBinder

private void initBinder(WebDataBinder binder) {

binder.setValidator(userValidator);

}

@RequestMapping(value = "/admin/addMembers.htm", method = RequestMethod.GET)

protected ModelAndView addMemberForm() throws Exception{

ModelAndView model = new ModelAndView();

model.addObject("member", new LibraryMember());

model.setViewName("add-member");

return model;

}

@RequestMapping(value = "/admin/addMembers.htm", method = RequestMethod.POST)

protected ModelAndView addMember(HttpServletRequest request, @ModelAttribute("member") LibraryMember member, BindingResult result){

HttpSession session = request.getSession();

userValidator.validate(member, result);

if (result.hasErrors()) {

return new ModelAndView("add-member", "member", member);

}

try {

LibraryMember createdMember = userDAO.createLibraryMemeber(member);

List<LibraryMember> members = userDAO.getAllLibraryMembers();

session.setAttribute("members", members);

} catch (UserException e) {

e.printStackTrace();

}

return new ModelAndView("admin-home");

}

}

RequestController.java

package com.my.onlinelibrary.controller;

import java.util.Date;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.http.MediaType;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.servlet.ModelAndView;

import com.my.onlinelibrary.dao.BookDAO;

import com.my.onlinelibrary.dao.CirculationDAO;

import com.my.onlinelibrary.dao.RequestDAO;

import com.my.onlinelibrary.dao.UserDAO;

import com.my.onlinelibrary.json.JsonUtils;

import com.my.onlinelibrary.pojo.Book;

import com.my.onlinelibrary.pojo.BookCirculation;

import com.my.onlinelibrary.pojo.BookReservation;

import com.my.onlinelibrary.pojo.LibraryMember;

import com.my.onlinelibrary.pojo.LibraryUsers;

@Controller

public class RequestController {

@Autowired

@Qualifier("bookrequestDAO")

RequestDAO bookrequestDAO;

@Autowired

@Qualifier("userDAO")

UserDAO userDAO;

@Autowired

@Qualifier("bookDAO")

BookDAO bookDAO;

@Autowired

@Qualifier("circulationDAO")

CirculationDAO circulationDAO;

@RequestMapping(value = "/admin/viewRequests.htm", method = RequestMethod.GET)

protected ModelAndView displayOpenRequests() throws Exception {

ModelAndView model = new ModelAndView();

List<BookReservation> bookRequests = bookrequestDAO.getPendingRequest("Pending");

model.addObject("bookRequests", bookRequests);

model.setViewName("pending-requests");

return model;

}

@RequestMapping(value = "/admin/viewmemberinfo.htm", method = RequestMethod.GET, produces = "application/json")

@ResponseBody

public String getUserInfo(@RequestParam String memberID, HttpServletRequest request) throws Exception {

String result;

System.out.println(memberID);

Long id = Long.parseLong(memberID);

LibraryMember member = userDAO.getMember(id);

System.out.println("Member ID: " + member.getMemberID());

result = "{" + JsonUtils.toJsonField("memberId", Long.toString(member.getMemberID()))

+ (", " + JsonUtils.toJsonField("memberfirstname", member.getLibraryUser().getFirstName()))

+ (", " + JsonUtils.toJsonField("memberlastname", member.getLibraryUser().getLastName()))

+ (", " + JsonUtils.toJsonField("latesubmission", Integer.toString(member.getLatesubmission())))

+ (", " + JsonUtils.toJsonField("emailId", member.getLibraryUser().getEmailId())) + "}";

return result;

}

@RequestMapping(value = "/admin/approverequest.htm", method = RequestMethod.POST, produces = "application/json")

@ResponseBody

public Long approveRequest(@RequestParam String requestID, HttpServletRequest request) throws Exception {

System.out.println(requestID);

Long id = Long.parseLong(requestID);

BookReservation br = bookrequestDAO.getRequest(id);

System.out.println("Request ID: " + br.getReservationID());

bookrequestDAO.updateStatus(br, "Approved");

BookCirculation bc = new BookCirculation(br.getLibrarymember(), br.getBook(), br.getRequestDate(), br.getTillDate());

bc = circulationDAO.addBookCirculation(bc);

return bc.getCirculationId();

}

@RequestMapping(value = "/admin/rejectrequest.htm", method = RequestMethod.POST, produces = "application/json")

@ResponseBody

public int deleteRequest(@RequestParam String requestID, HttpServletRequest request) throws Exception {

System.out.println(requestID);

Long id = Long.parseLong(requestID);

BookReservation br = bookrequestDAO.getRequest(id);

System.out.println("Request ID: " + br.getReservationID());

Book b = br.getBook();

b.setAvailable(true);

bookDAO.updateBookAvailability(b);

bookrequestDAO.updateStatus(br, "Declined");

br.getBook().setAvailable(true);

int data = 1;

return data;

}

@RequestMapping(value = "/member/viewraisedrequest.htm", method = RequestMethod.GET)

public ModelAndView getUserRequests(HttpServletRequest request) throws Exception {

HttpSession session = request.getSession();

LibraryUsers u = (LibraryUsers) session.getAttribute("user");

LibraryMember mem = userDAO.getLibraryMember(u);

List<BookReservation> result = bookrequestDAO.getUserRequests(mem);

return new ModelAndView("book-requested", "bookreservationlist", result);

}

}