American International University-Bangladesh (AIUB)

Department of Computer Science

Faculty of Science &Technology (FST)

Summer 21 22

Section: E

Software Quality Assurance and Testing

Library Management System

A Report submitted By

SN	Student Name	Student ID
1	Islam, Md. Ahasanul	19-40396-1
2	Sheikh Muhtasim Nasif	20-42119-1
3	Farhan Israk Mahmud	18-38057-2
4	Asraful Islam	19-40166-1

Under the supervision of
Abhijit Bhowmik
Associate Professor
Department of Computer Science
Faculty of Science & Technology
American International University-Bangladesh

Software Test Plan

for

Library Management System

Version 1.0 approved

Prepared by

Islam, Md. Ahasanul
Sheikh Muhtasim Nasif
Farhan Israk Mahmud
Asraful Islam

American International University-Bangladesh
14 August, 2022

Checked By Industry Personnel

Name:

Designation:

Company:

Sign:

Date:

Table of Contents

Revision History

Revision	Date	Updated by	Update Comments
0.1	2022.07.10	Islam, Md. Ahasanul	First Draft
0.2	2022.07.11	Sheikh Muhtasim Nasif	Second Draft
0.3	2022.07.12	Farhan Israk Mahmud	Third Draft
0.4	2022.07.13	Asraful Islam	Forth Draft
0.5	2022.07.14	Islam, Md. Ahasanul	Fifth Draft
0.6	2022.07.16	Sheikh Muhtasim Nasif	Sixth Draft
0.7	2022.07.17	Farhan Israk Mahmud	Seventh Draft
0.8	2022.07.18	Islam, Md. Ahasanul	Eighth Draft
0.9	2022.07.20	Asraful Islam	Ninth Draft

1. TEST PLAN IDENTIFIER:RS-MTP01.3

2. REFERENCES

- Software Quality and Testing Course PowerPoint Slides
- https://www.blazemeter.com/blog/how-use-postman-test-apis
- https://sourceforge.net/projects/projectlibre/
- https://www.postman.com/automated-testing/
- http://www.onestoptesting.com/test-plan/features-not-to-be-tested.asp
- SQAT - CH.02 - Quality Attributes [4.2: System quality Attributes]
- https://www.softwaretestinghelp.com/
- https://www.softwaretestinghelp.com/how-to-write-effective-test-cases-test-cases-procedures-and-definitions/
- https://www.w3schools.com/

3. INTRODUCTION

3.1 Background to the Problem

In our university Library we can see that we do not have any modern system where we can see booklist, quantity of books. Moreover, if any student wants to discover desired books, he has to do it manually. It will be beneficial for students to use a self-centered system while discovering his/her desired books.

3.2 Solution to the Problem

We want to build a system where a student can see the booklist and the quantity of books in the library by using the system and a student can borrow or return the book by using the self-centered system. If a student delay to return the borrow book a punishment amount automatically will be added for the student after the deadline.

4. REQUEIREMNT SPECIFICATION

4.1 System Features

1.Admin Feature

Priority Level: High

Precondition: admin must have valid user name and password

1.1 Login to the system

1.2 Logout from the system

1.3 Can change the password

1.4 Can add a Librarian

1.5 Can see the Librarian details

1.6 Can delete the Librarian

1.7 Can update the Librarian details

1.8 Can delete the Librarian

Priority Level: High

Precondition: Librarian must have valid user name and password

2.Librarian Feature

2.1 Login to the website

2.2 Logout from the system

2.3 Can change the password

2.4 Can see all booklist and add or delete any book

2.5 can update any book

2.6 can view all users list

2.7 Can delete or update any user

2.8 Can see the Requested booklist

Priority Level: High

Precondition: Users must have valid user name and password

3.Users Feature

3.1 Register to the system to make their own account

3.2 Login to the system

3.3 Logout from the system

3.4 Can change the password

3.5 can requested for a book

3.6 can borrow a book

3.7 Can see the borrowed booklist

3.8 Can update own information

4.2 System Quality Attributes

1. **Usability:** This can be estimated with regards to usability. The system ought to be easy to understand. An everyday customer of our very own ought to have the alternative to post about his/her gadgets on a regular of 5 minutes and a restrict of 7 minutes. Our framework highlights are not difficult to advance as route is particularly basic. It's Easy for a new or uncommon user to determine out how to utilize the framework.

2. **RELIABILITY AND CORRECTNESS:** No software is free from bug. Our software offers the actual output what user wants; its correctness has been ensured. Our software does not crash randomly as it has been tested and no false output is generated so It's more reliable to use for our users.

3. **Modularity**: Each system should be developed according to our modules. Our system consists of many modules and integrated to make it a complete system. So, we can easily identify the bug in any module and then we need to fix only that particular module we don't have to worry about other modules so it makes the work easier for our tester. In addition to this we can add new features to our system as it is built in modules.

4. **Maintainability:** This means how without problems the maintenance team can perform their work. The major task of our maintenance team is to fix bugs, add something new or exchange some features. One of our maintenance programmers will be able to exchange any function with 24-man hours or much less of development effort.

5. **Efficiency:** Main system quality features. Any given task in the system is measured in terms of time required to complete it. The efficiency of the application will be large.

6. **TESTABILITY:** It means how easily the testing team can perform their work. Testability is highly dependent on modularity and since our system is built module wise the tester will go easy; they don't have to test every module to fix bugs.

7. **FLEXIBILITY:** The effort required to change an operational program. A maintenance programmer with at least one year of experience supporting this product should be able to provide a new copy of output for the product, including code changes and testing, with no more than 2 hours of labor.

4.3 System Interface

1. Homepage interface:

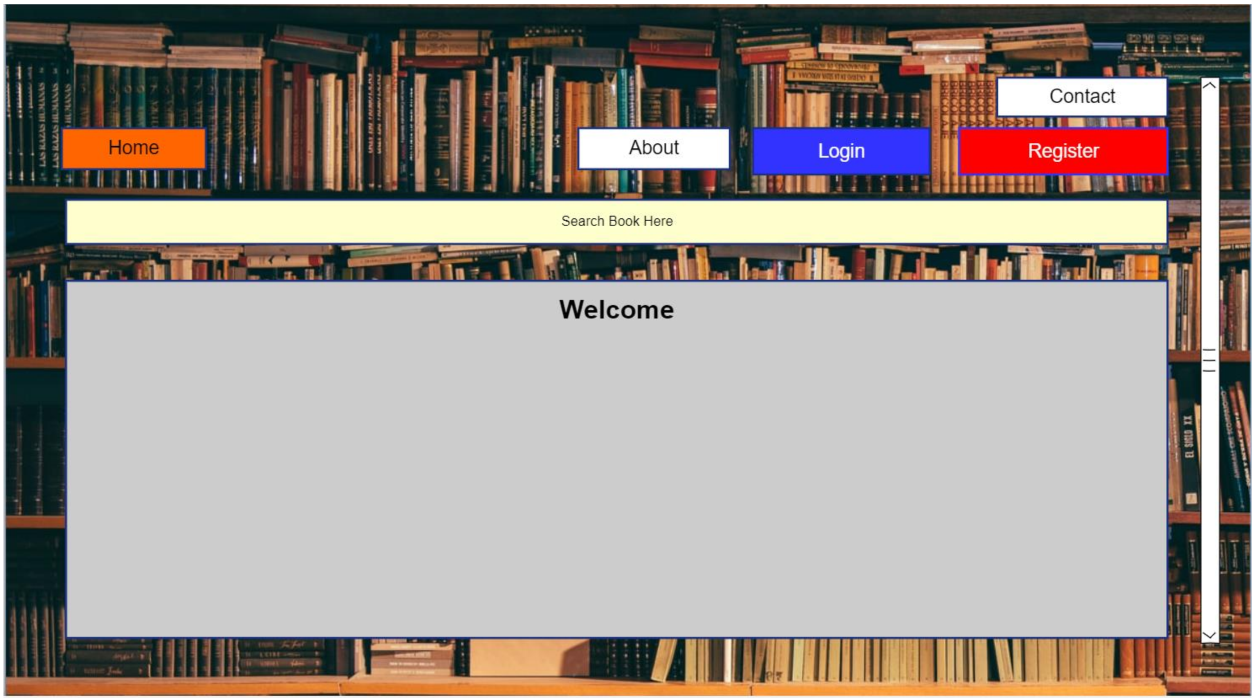


Fig.01: This is an interface for our system home page. This will be appeared whenever one open our website.

2. Login interface:

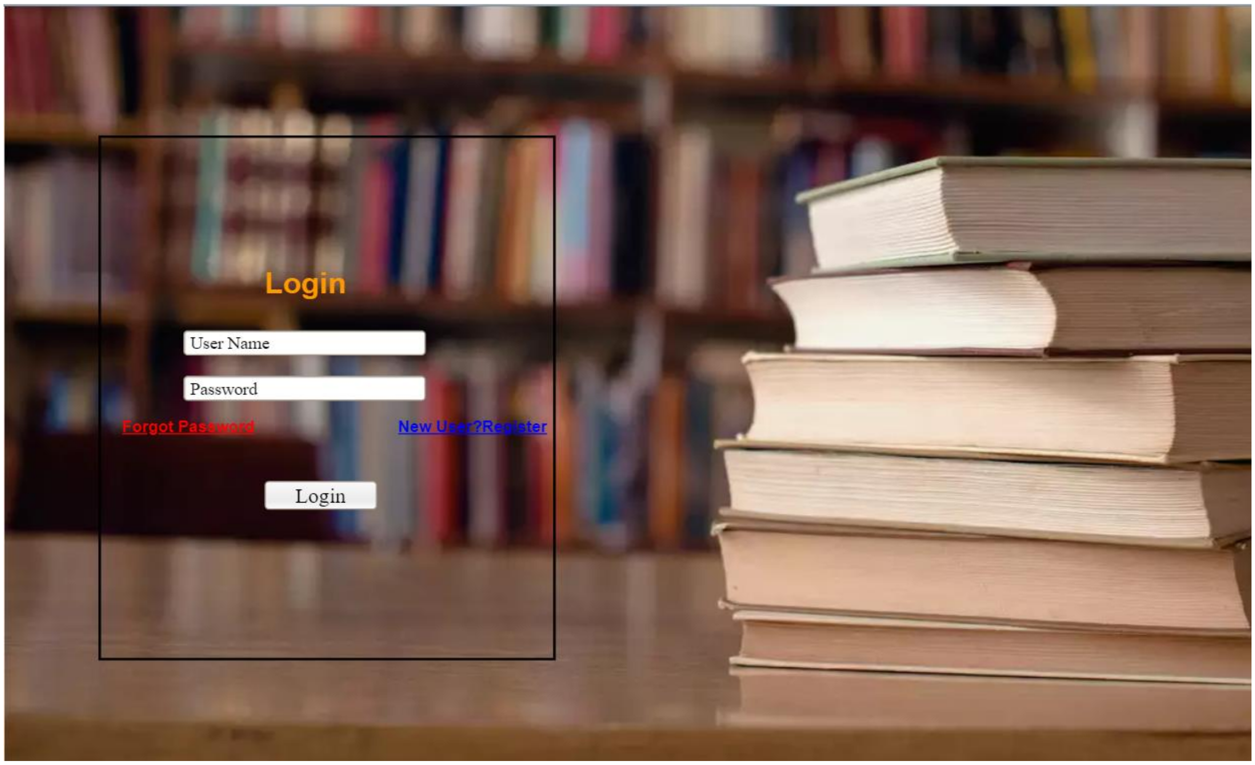


Fig.02: This is an interface for our system Login. This will be appeared whenever one go to the login in our system

3. Registration interface:

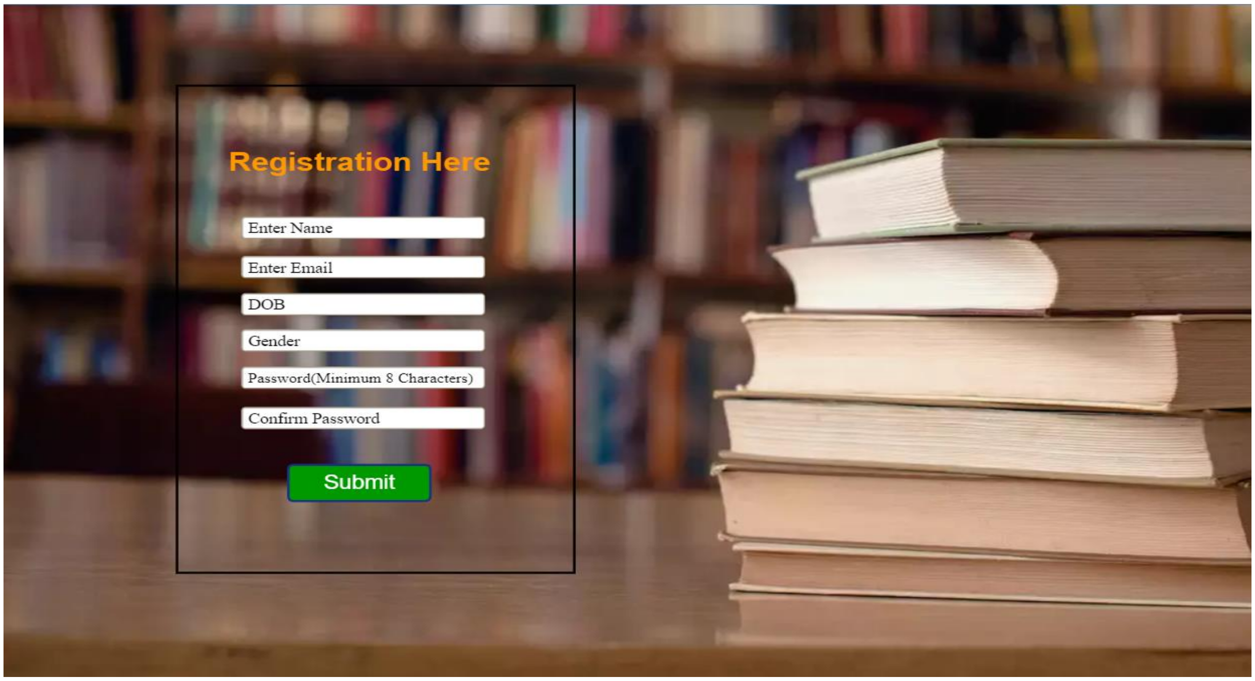


Fig.03: This is an interface for our system Registration. This will be appeared whenever one go to for registration

4. Password Change Interface:

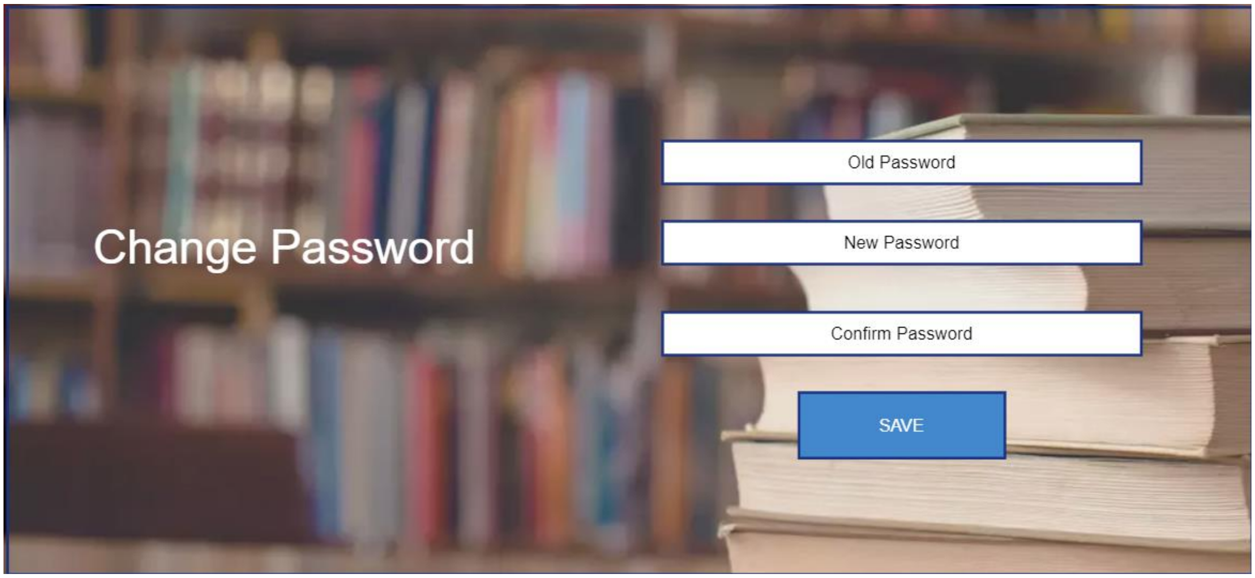
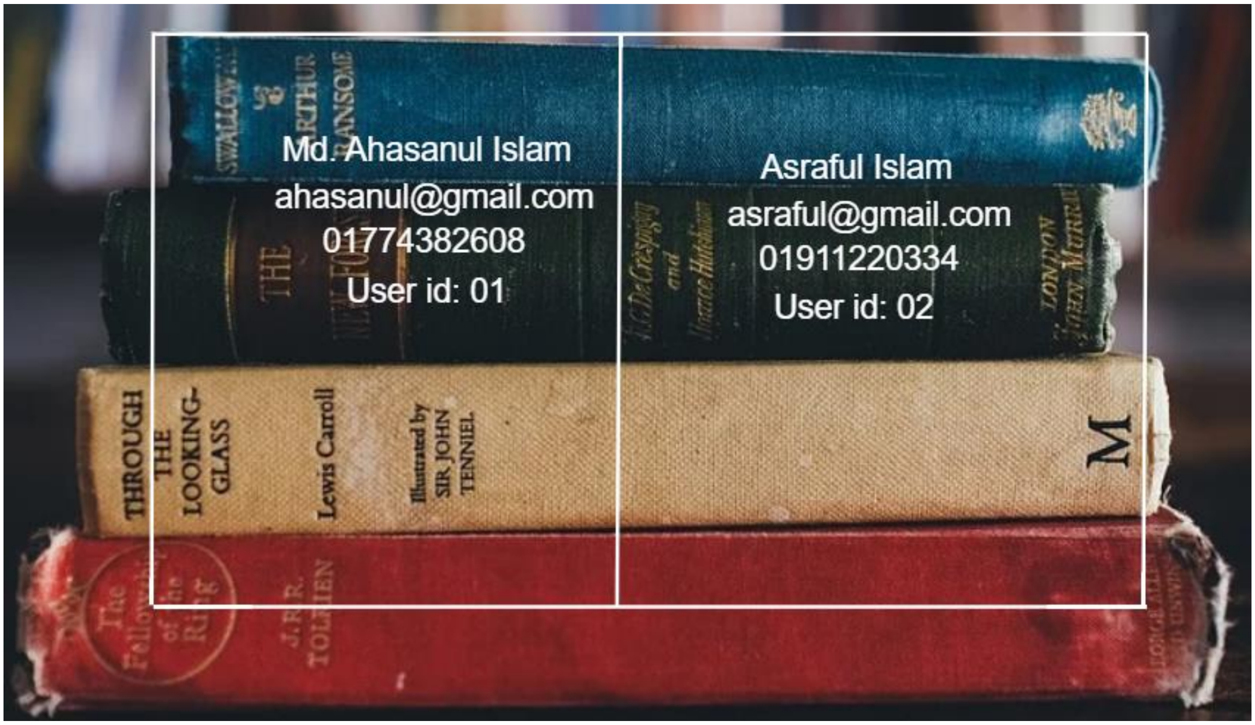


Fig.04: This is an interface for our system Change Password. When any user wants to change password, this interface will be appeared

5. View Users interface:



Md. Ahasanul Islam
ahasanul@gmail.com
01774382608
User id: 01

Asraful Islam
asraful@gmail.com
01911220334
User id: 02

THROUGH THE LOOKING-GLASS
Lewis Carroll
Illustrated by SIR JOHN TENNIEL
M
The Fellowship of the Ring
J.R.R. TOLKIEN

Fig.05: This is an interface for our system View user. This will be appeared whenever Librarian want to see the user details.

6.Delete User interface:

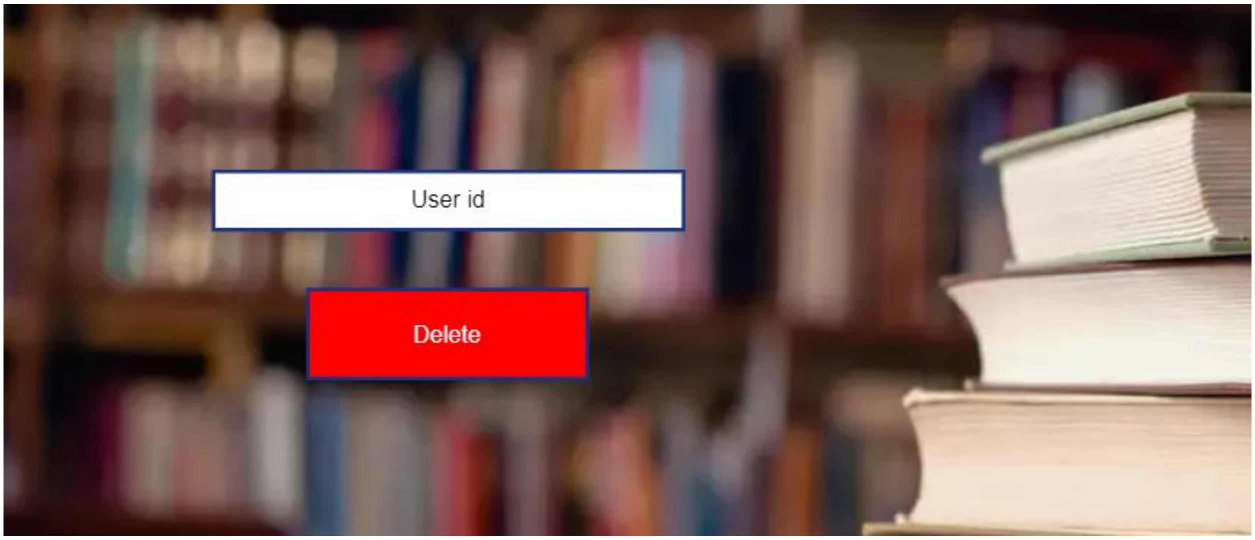


Fig.06: This is an interface for our system delete user. This will be appeared whenever a Librarian wants to delete a user.

7. View Booklist interface:

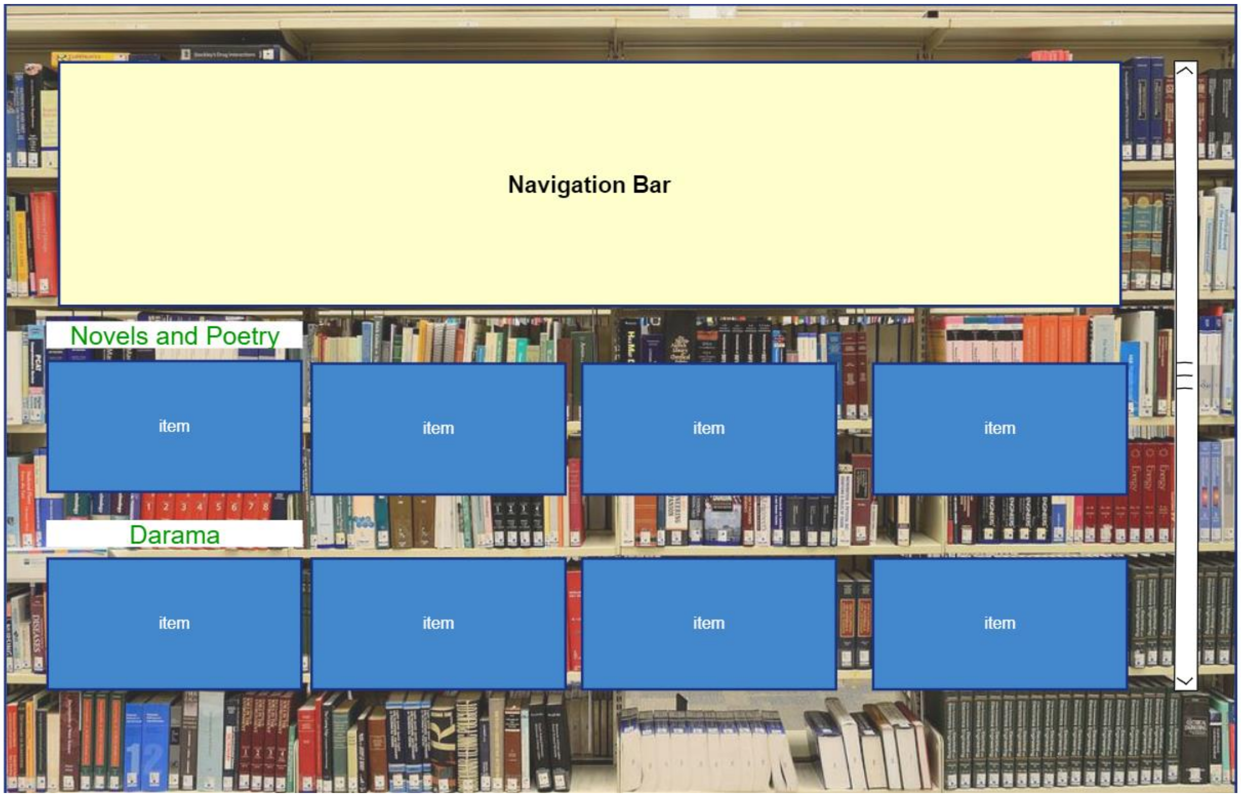


Fig.07: This is an interface for our system View books details.

8. Delete Book interface:

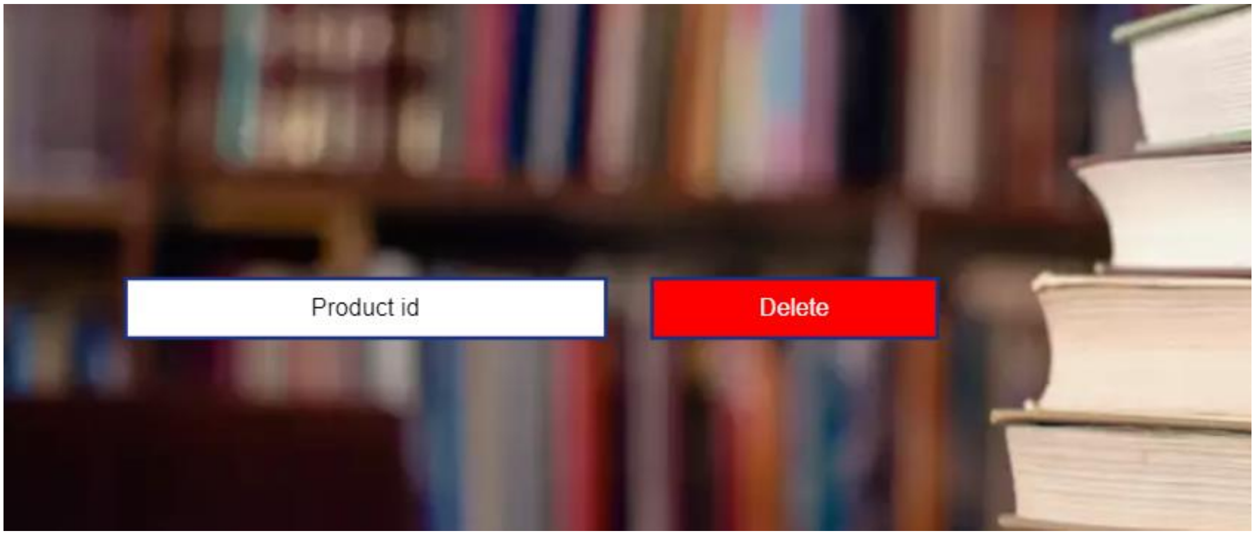


Fig.08: This is an interface for our system View books details.

4.4 Project Requirements

Effort Estimation:

Our project is to develop an application named “Library Management system”.

Development Time = 3 Months

Required number of peoples = 4

Budget Estimation:

Duration in weeks = 3*4 = 12 weeks

Office days = 5 days

Working hours = 8 Hours

So, Per week working hours is = (5*8) hours = 40 hours

So Total Working hours is = (40*8) hours

= 320 hours.

Developer salary is = 800 Taka

Total developers Salary = (800*320) Taka

= 25,6000 Taka

Expanse	Amount	Total Amount
Salary for 4 developers		256,000 Taka
3 months office rent	3*8000	24000 Taka
Electricity and other costs		10000 Taka
1 months Maintenance cost	4*10*1200	48000 Taka
Travel Cost	1*5000	5000 Taka
Total Cost		343000Taka
15% of total cost(profit)		51,450 Taka
Now total budget is		394450 Taka

5. FEATURES NOT TO BE TESTED

The following is a list of the areas that will not be specifically addressed. All testing in these areas will be indirect as a result of other testing efforts. For example:

Users can contact Librarian: In our web application there is a feature where a user can contact with librarian. This feature is not to be tested in this release of the software.

Automatic fine added: In this system where a user can borrow a book from the library. There is a fixed time to returning the borrowing book so if a user returns the borrowing book after fixed time, he/she will be fined a significant amount for the delay. This feature is not to be tested in this release of the software.

Librarian Can contact system admin: This feature was also not been tested because of the same reason of user can contact librarian feature, due to low risk.

User Can add own location information: We have a feature in our system where a user can add own location but it is not mandatory at the first place and due to a tight schedule to follow so we opt out this feature to be tested of this version of the software.

6. TESTING APPROACH

- **Unit Testing:** First phase we will do unit testing. In this test we will test individual software units or components. For example, each class or method. The aim is to ensure that every unit of software code works as intended. This testing is done by developers during the development section of an application. In this step, we will apply the "White Box Testing" technique.

- **Integration Testing:** Then in the second phase we will do the integration. In this test we will ensure that all software modules are logically integrated and tested as a group. Our project consists of several software modules written by four programmers. The purpose of this level of testing is to find errors in the way different software modules interact when integrated in this step, we will follow the "bottom-up integration" approach.

- **System Testing:** Then we will do system testing. Through system testing we will check full-featured, fully integrated systems. Then we will verify if it meets all the requirements. Black-box testing falls under this condition. So, at this level, we will follow the "Black Box Testing" technique.

- **Acceptance Testing:** The last phase of our testing is acceptance testing. We will perform this check to check the acceptability of our products. This test will be completed to check if any errors have been missed throughout the functional testing phase. At this level, we will follow the "Black Box Testing" technique. After that, we can run unit tests again.

6.1 Testing Levels

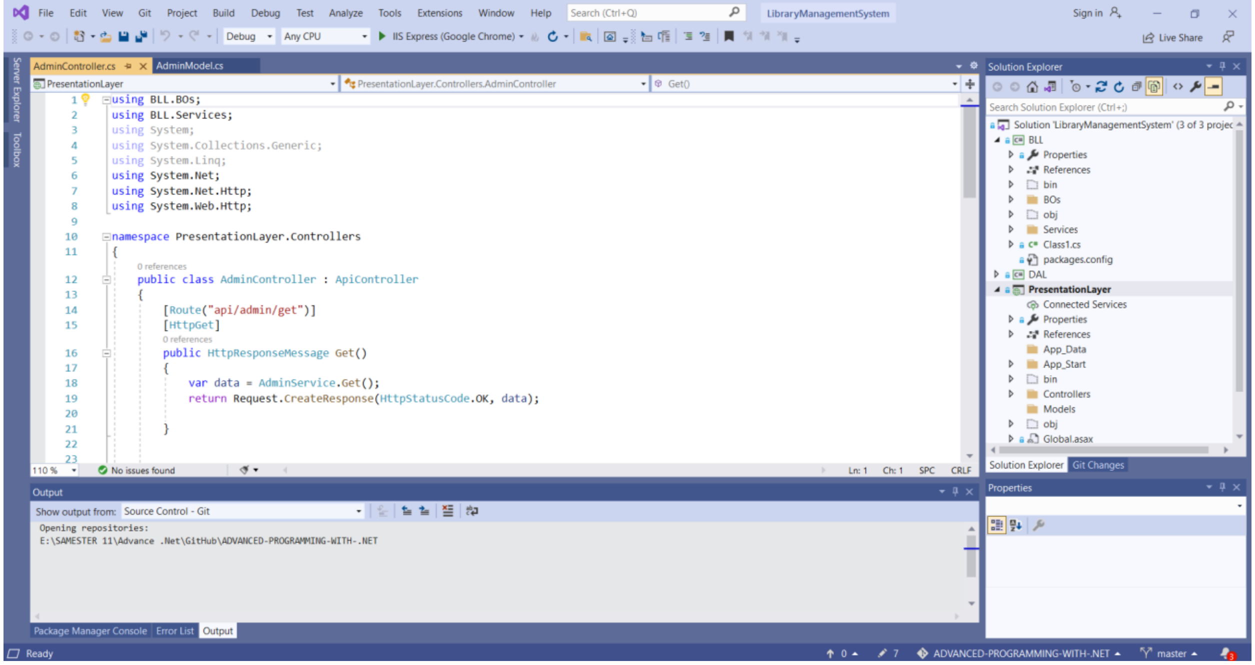
SYSTEM/INTEGRATION After completing the test of all the smallest part, the system must be connected together. The 100% tested all parts of the units are converted into a module where another test is done. It’s called the integration testing. After adding all modules together, the regression testing is done to the units but in here, the main purpose is to test the interface of the system that all the connections are working perfectly or not. Also, it can be called data flow test.

- UNIT TESTING Unit testing is the most common part of a testing where a small part of the project is going to be tested by the developers. After developing a small part of the project, this small part or portion is tested to get understand the system is working properly or not. In here test personal should find out the correctness of the inaccessible code.

- The customer will do ACCEPTANCE testing with the help of the test manager and development team leader. After the System/Integration test is completed, the acceptance test will run in parallel with the existing manual ZIP/FAX process for one month.

6.2 Test Tools

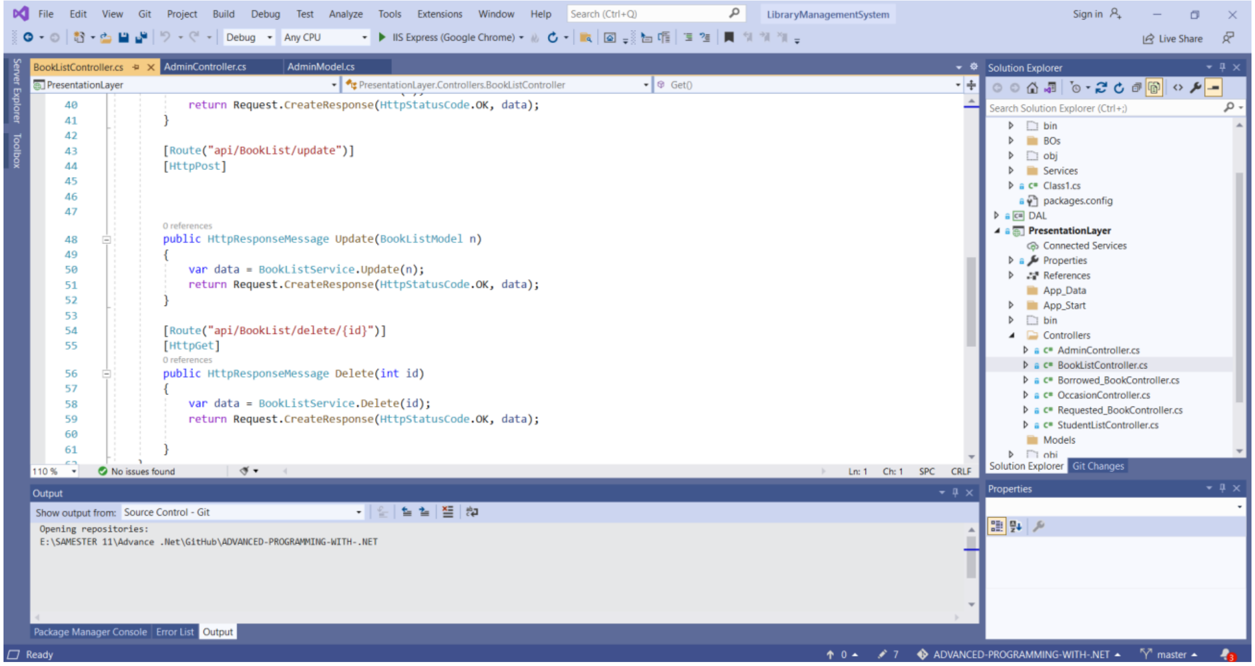
1) Post Man



```
using BLL.BOs;
using BLL.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;

namespace PresentationLayer.Controllers
{
    0 references
    public class AdminController : ApiController
    {
        [Route("api/admin/get")]
        [HttpGet]
        0 references
        public HttpResponseMessage Get()
        {
            var data = AdminService.Get();
            return Request.CreateResponse(HttpStatusCode.OK, data);

        }

```

Show output from: Source Control - Git

```
Opening repositories:
E:\SAMESTER 11\Advance .Net\GitHub\ADVANCED-PROGRAMMING-WITH-.NET
```



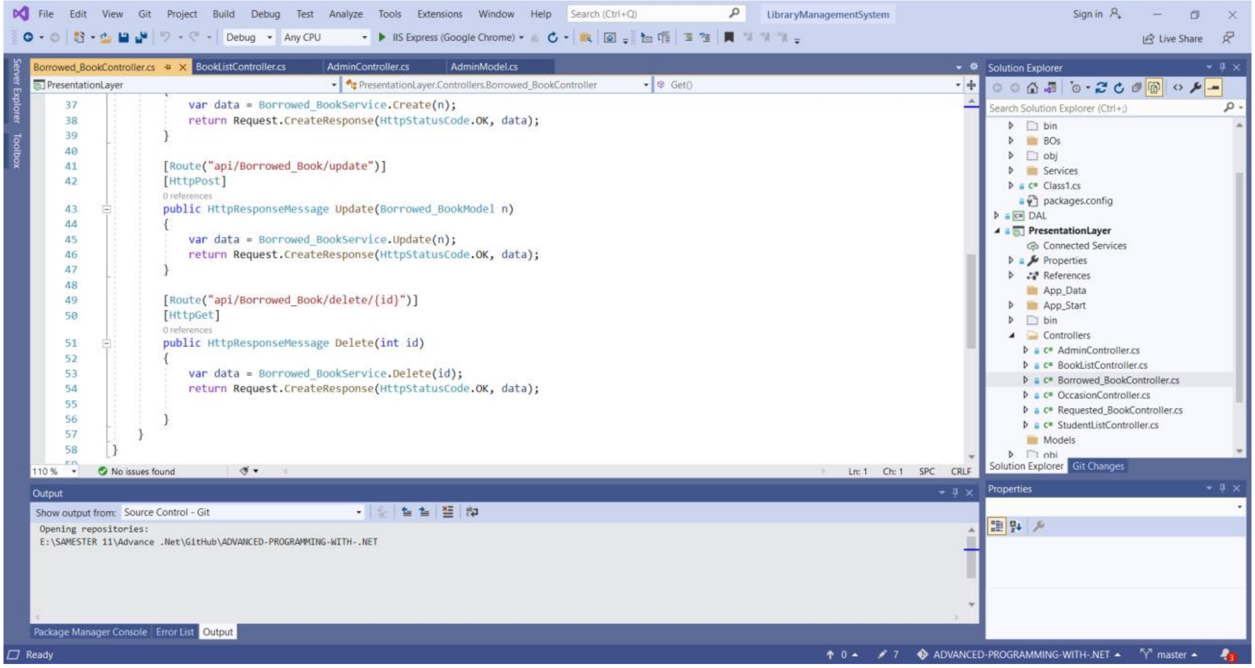
```
            return Request.CreateResponse(HttpStatusCode.OK, data);
        }

        [Route("api/BookList/update")]
        [HttpPost]



        0 references
        public HttpResponseMessage Update(BookListModel n)
        {
            var data = BookListService.Update(n);
            return Request.CreateResponse(HttpStatusCode.OK, data);
        }

        [Route("api/BookList/delete/{id}")]
        [HttpGet]
        0 references
        public HttpResponseMessage Delete(int id)
        {
            var data = BookListService.Delete(id);
            return Request.CreateResponse(HttpStatusCode.OK, data);

        }
```

110 % No issues found Ln: 1 Ch: 1 SPC CRLF

Output

Show output from: Source Control - Git

```
Opening repositories:
E:\SAMESTER 11\Advance .Net\GitHub\ADVANCED-PROGRAMMING-WITH-.NET
```



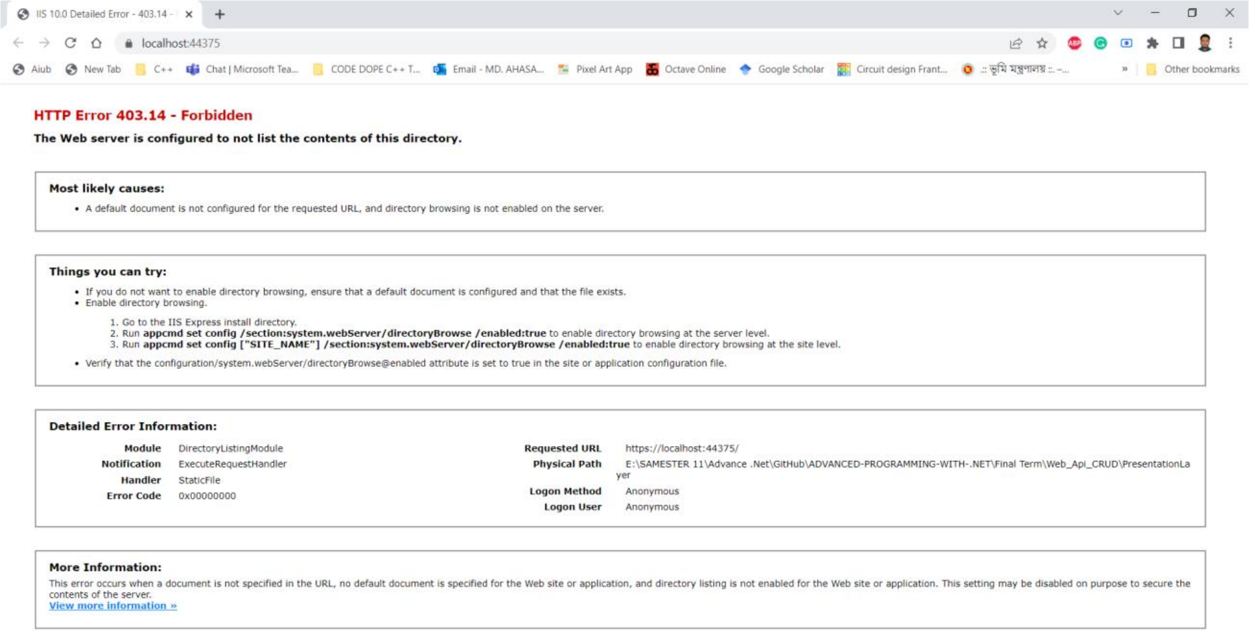
```
            var data = Borrowed_BookService.Create(n);
            return Request.CreateResponse(HttpStatusCode.OK, data);
        }

        [Route("api/Borrowed_Book/update")]
        [HttpPost]
        0 references
        public HttpResponseMessage Update(Borrowed_BookModel n)
        {
            var data = Borrowed_BookService.Update(n);
            return Request.CreateResponse(HttpStatusCode.OK, data);
        }

        [Route("api/Borrowed_Book/delete/{id}")]
        [HttpGet]
        0 references
        public HttpResponseMessage Delete(int id)
        {
            var data = Borrowed_BookService.Delete(id);
            return Request.CreateResponse(HttpStatusCode.OK, data);

        }
    }
}
```

110 % No issues found Ln: 1 Ch: 1 SPC CRLF

Output

Show output from: Source Control - Git

```
Opening repositories:
E:\SAMESTER 11\Advance .Net\GitHub\ADVANCED-PROGRAMMING-WITH-.NET
```



HTTP Error 403.14 - Forbidden

The Web server is configured to not list the contents of this directory.

Most likely causes:

- A default document is not configured for the requested URL, and directory browsing is not enabled on the server.

Things you can try:

- If you do not want to enable directory browsing, ensure that a default document is configured and that the file exists.
- Enable directory browsing.
  1. Go to the IIS Express install directory.
  2. Run **appcmd set config /section:system.webServer/directoryBrowse /enabled:true** to enable directory browsing at the server level.
  3. Run **appcmd set config ["SITE_NAME"] /section:system.webServer/directoryBrowse /enabled:true** to enable directory browsing at the site level.
- Verify that the configuration/system.webServer/directoryBrowse@enabled attribute is set to true in the site or application configuration file.

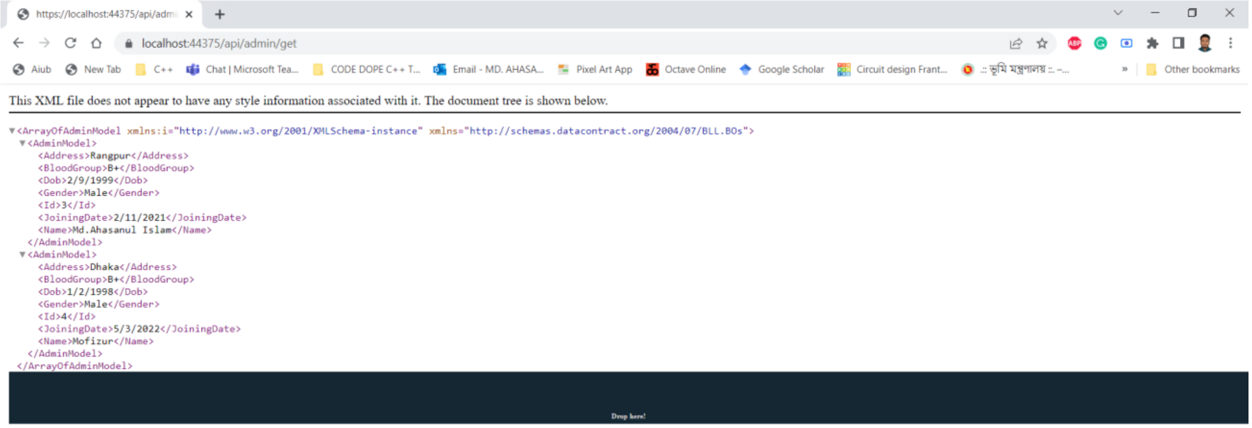
Detailed Error Information:

Module	DirectoryListingModule	**Requested URL**	https://localhost:44375/
Notification	ExecuteRequestHandler	**Physical Path**	E:\SAMESTER 11\Advance .Net\GitHub\ADVANCED-PROGRAMMING-WITH-.NET\Final Term\Web_Api_CRUD\PresentationLayer
Handler	StaticFile	**Logon Method**	Anonymous
Error Code	0x00000000	**Logon User**	Anonymous

More Information:

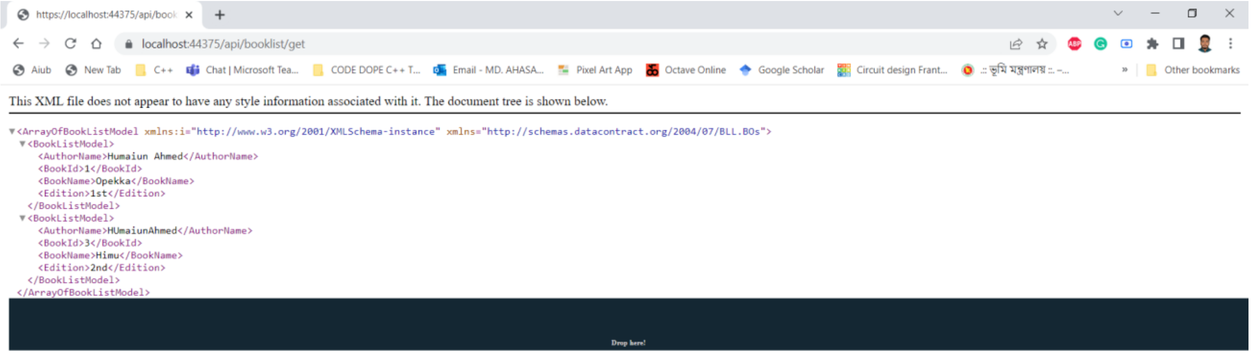
This error occurs when a document is not specified in the URL, no default document is specified for the Web site or application, and directory listing is not enabled for the Web site or application. This setting may be disabled on purpose to secure the contents of the server.

View more information »



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<ArrayOfAdminModel xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/BLL.BOs">
  ▼<AdminModel>
     <Address>Rangpur</Address>
     <BloodGroup>B+</BloodGroup>
     <Dob>2/9/1999</Dob>
     <Gender>Male</Gender>
     <Id>3</Id>
     <JoiningDate>2/11/2021</JoiningDate>
     <Name>Md.Ahasanul Islam</Name>
   </AdminModel>
  ▼<AdminModel>
     <Address>Dhaka</Address>
     <BloodGroup>B+</BloodGroup>
     <Dob>1/2/1998</Dob>
     <Gender>Male</Gender>
     <Id>4</Id>
     <JoiningDate>5/3/2022</JoiningDate>
     <Name>Mofizur</Name>
   </AdminModel>
 </ArrayOfAdminModel>
```

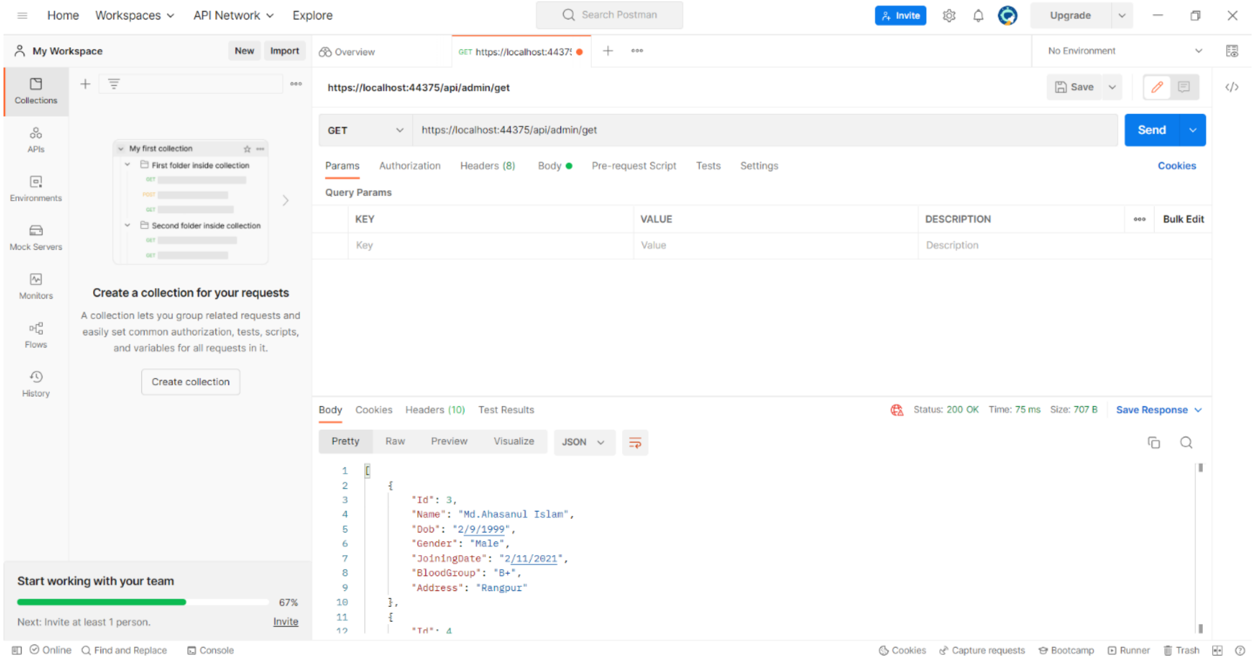


localhost:44375/api/booklist/get

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<ArrayOfBookListModel xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/BLL.BOs">
  ▼<BookListModel>
     <AuthorName>Humaiun Ahmed</AuthorName>
     <BookId>1</BookId>
     <BookName>Opekka</BookName>
     <Edition>1st</Edition>
   </BookListModel>
  ▼<BookListModel>
     <AuthorName>HUmaiunAhmed</AuthorName>
     <BookId>3</BookId>
     <BookName>Himu</BookName>
     <Edition>2nd</Edition>
   </BookListModel>
 </ArrayOfBookListModel>
```

Show librarian details:

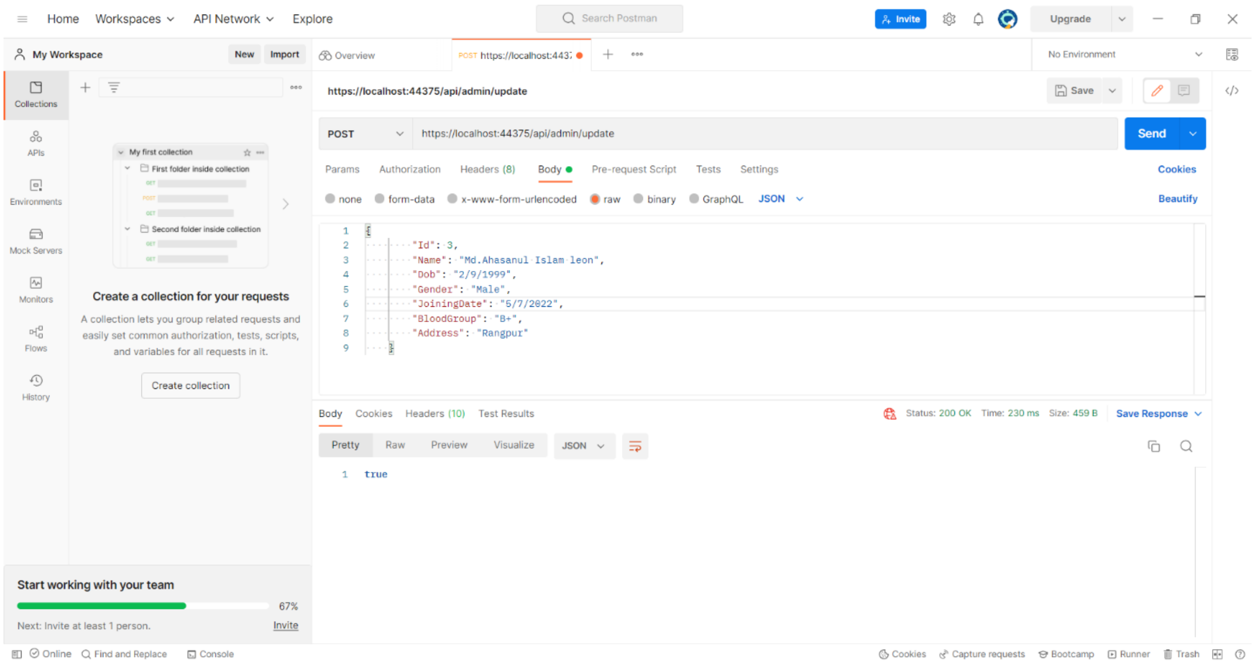


GET https://localhost:44375/api/admin/get

Status: 200 OK Time: 75 ms Size: 707 B

```
[
    {
        "Id": 3,
        "Name": "Md.Ahasanul Islam",
        "Dob": "2/9/1999",
        "Gender": "Male",
        "JoiningDate": "2/11/2021",
        "BloodGroup": "B+",
        "Address": "Rangpur"
    },
    {
        "Id": 4
```

Update librarian list:



https://localhost:44375/api/admin/update

POST https://localhost:44375/api/admin/update

Params | Authorization | Headers (8) | Body ● | Pre-request Script | Tests | Settings

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

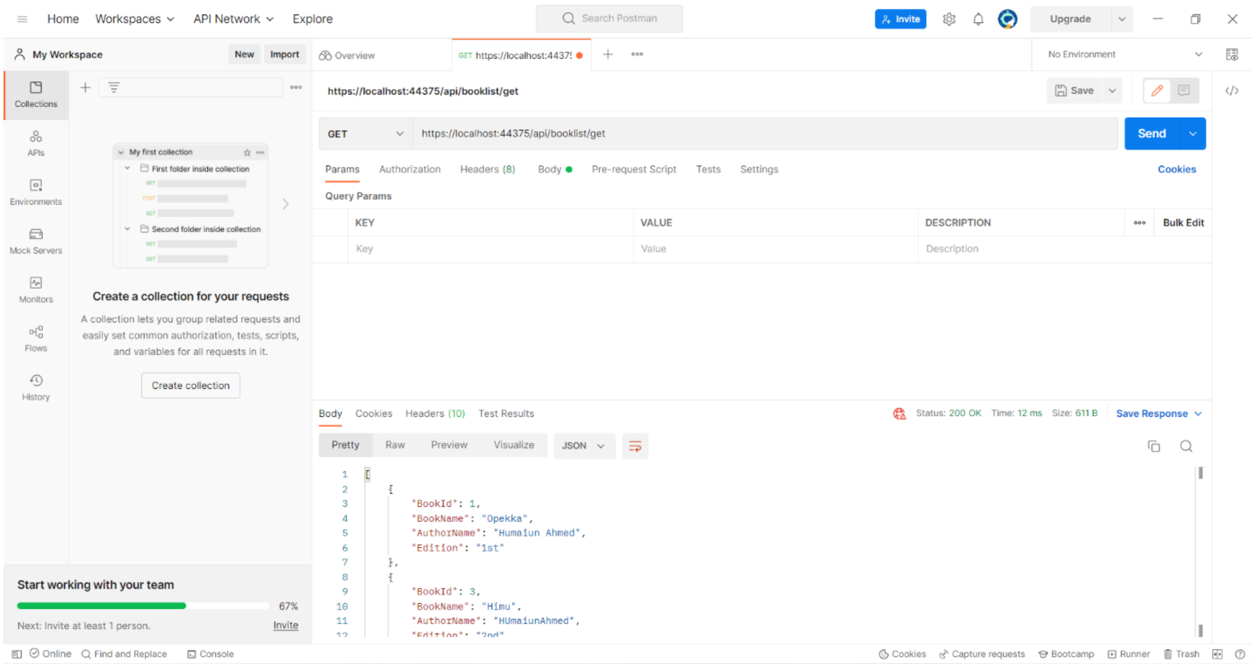
```
{
    "Id": 3,
    "Name": "Md.Ahasanul Islam leon",
    "Dob": "2/9/1999",
    "Gender": "Male",
    "JoiningDate": "5/7/2022",
    "BloodGroup": "B+",
    "Address": "Rangpur"
}
```

Body | Cookies | Headers (10) | Test Results — Status: 200 OK Time: 230 ms Size: 459 B — Save Response

Pretty | Raw | Preview | Visualize | JSON

```
true
```

Show all booklist:


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
No Environment
Save
Send
Cookies
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
My first collection
First folder inside collection
Second folder inside collection
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create collection
Start working with your team
67%
Next: Invite at least 1 person.
Invite
Online
Find and Replace
Console
Cookies
Capture requests
Bootcamp
Runner
Trash

https://localhost:44375/api/booklist/get

GET https://localhost:44375/api/booklist/get

Params | Authorization | Headers (8) | Body ● | Pre-request Script | Tests | Settings

Query Params

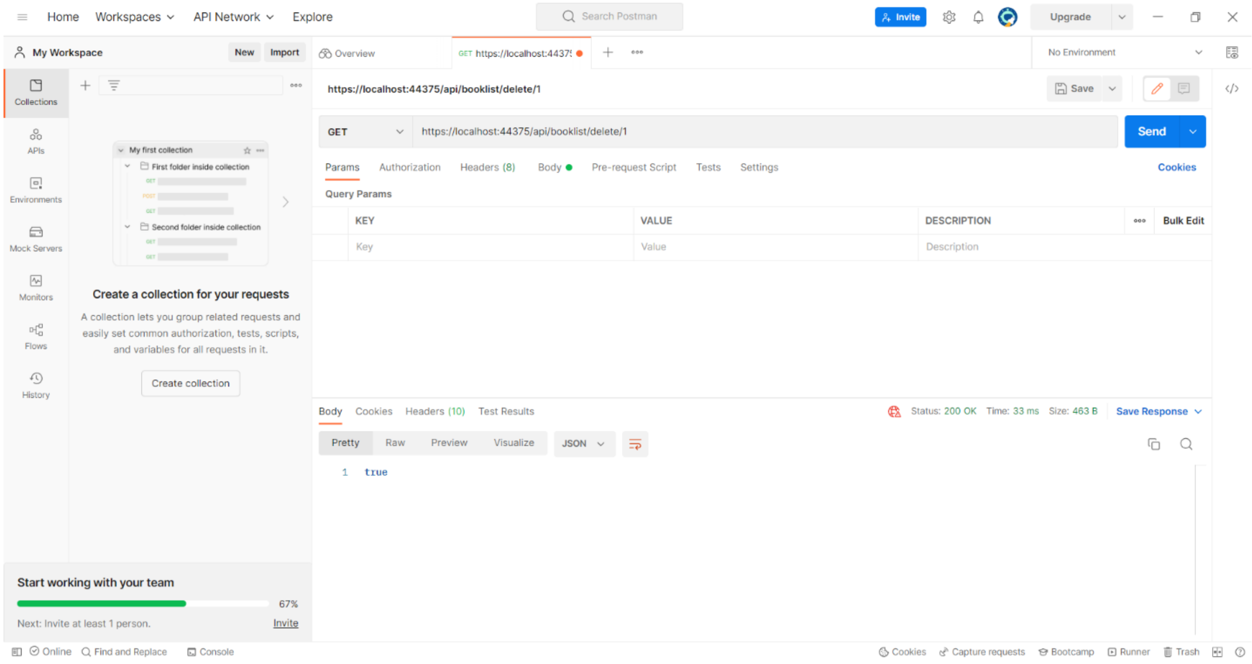
KEY	VALUE	DESCRIPTION
Key	Value	Description

Body | Cookies | Headers (10) | Test Results — Status: 200 OK Time: 12 ms Size: 611 B — Save Response

Pretty | Raw | Preview | Visualize | JSON

```
[
    {
        "BookId": 1,
        "BookName": "Opekka",
        "AuthorName": "Humaiun Ahmed",
        "Edition": "1st"
    },
    {
        "BookId": 3,
        "BookName": "Himu",
        "AuthorName": "HUmaiunAhmed",
        "Edition": "2nd"
```

Delete a book:



GET https://localhost:44375/api/booklist/delete/1

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 33 ms Size: 463 B

```
1  true
```

6.3 Meetings

The testing team will meet every week and judge the system quality and preparing for the next step. The team leader will meet the developer and testers timely and check the progress of the system. Then the project manager will check the overall process of the system time to time. These two meetings are arranged on different weeks. Emergency meeting are called as needed for emergency situation.

7. TEST CASES

7.1 Log in

This Test case will Test the website login page by verifying valid username and password

Project Name: Library Management System	**Test Designed by:** Farhan Israk Mahmud
Test Case ID: Login_01	**Test Designed date:** 10-06-2022
Test Priority (Low, Medium, High): High	**Test Executed by**: Farhan Israk Mahmud
Module Name: Login	**Test Execution date:** 13-06-2022
Test Title: Verifying login with valid username, password	
Description: Test the website login page	
Precondition (If any): Users must type valid username & password	

Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Go to the website. 2. Click the "Login" button. 3. Enter username 4. Enter password. 5. Click "Login" button	Username: Farhan_45 Password: Prime4455	Users should Login to the website	As expected,	Pass

7.2 Registration: This Test case will check the registration with valid username, email, password and date of birth, gender, Password and Confirm Password.

<table>
<tr><td colspan="3"><b>Project Name:</b> Library Management System</td><td colspan="2"><b>Test Designed by:</b> Farhan Israk Mahmud</td></tr>
<tr><td colspan="3"><b>Test Case ID:</b> Registration_02</td><td colspan="2"><b>Test Designed date:</b> 10-06-2022</td></tr>
<tr><td colspan="3"><b>Test Priority (Low, Medium, High):</b> High</td><td colspan="2"><b>Test Executed by</b>: Farhan Israk Mahmud</td></tr>
<tr><td colspan="3"><b>Module Name:</b> Registration</td><td colspan="2"><b>Test Execution date:</b> 13-06-2022</td></tr>
<tr><td colspan="3"><b>Test Title:</b> Registration with valid username, email, DOB, Gender and password.</td><td colspan="2"></td></tr>
<tr><td colspan="3"><b>Description:</b> Test the website registration.</td><td colspan="2"></td></tr>
<tr><td colspan="5"><b>Precondition (If any):</b> User must fill up all the input field.</td></tr>
<tr><td><b>Test Steps</b></td><td><b>Test Data</b></td><td><b>Expected Results</b></td><td><b>Actual Results</b></td><td><b>Status (Pass/Fail)</b></td></tr>
<tr><td>1. Go to the website.
2. Click the “registration” button
3. Enter all valid information.
4. Click “Submit” button</td><td>Username: Farhan_45
Gmail: shahidinfo.44@gmail.com
Date of birth: 19/11/1998
Gender: male
Password: Prime4455</td><td>Users should registration to the website</td><td>As expected,</td><td>Pass</td></tr>
<tr><td colspan="5"><b>Post Condition:</b> User is validated with database and successfully register an account. The account details are stored in the database</td></tr>
</table>

7.3 Log out:

This Test case will Test that the logout option is working properly or not

<table>
<tr><td colspan="3"><b>Project Name:</b> Library Management System</td><td colspan="2"><b>Test Designed by:</b> Farhan Israk Mahmud</td></tr>
<tr><td colspan="3"><b>Test Case ID:</b> Logout_03</td><td colspan="2"><b>Test Designed date:</b> 11-06-2022</td></tr>
<tr><td colspan="3"><b>Test Priority (Low, Medium, High):</b> High</td><td colspan="2"><b>Test Executed by</b>: Farhan Israk Mahmud</td></tr>
<tr><td colspan="3"><b>Module Name:</b> logout</td><td colspan="2"><b>Test Execution date:</b> 14-06-2022</td></tr>
<tr><td colspan="3"><b>Test Title:</b> Verifying logout option</td><td colspan="2"></td></tr>
<tr><td colspan="3"><b>Description:</b> Test the website logout option</td><td colspan="2"></td></tr>
<tr><td colspan="5"><b>Precondition (If any):</b> 1. Need account on this website
2. Need to be logged in</td></tr>
<tr><td><b>Test Steps</b></td><td><b>Test Data</b></td><td><b>Expected Results</b></td><td><b>Actual Results</b></td><td><b>Status (Pass/Fail)</b></td></tr>
<tr><td>1. go to the site
2. log in to the site
3. Click the “Logout” button</td><td></td><td>Successful logout</td><td>As expected,</td><td>Pass</td></tr>
<tr><td colspan="5"><b>Post Condition:</b> User goes back to the home page</td></tr>
</table>

7.4 Change Password:

This Test case will check the change password feature is working accurately or not

Project Name: Library Management System	**Test Designed by:** Farhan Israk Mahmud
Test Case ID: changePassword_04	**Test Designed date:** 11-06-2022
Test Priority (Low, Medium, High): High	**Test Executed by**: Farhan Israk Mahmud
Module Name: Update Password	**Test Execution date:** 14-06-2022
Test Title: Can change password from the profile.	
Description: Test password is update or not.	
Precondition (If any): User Must have to login into profile.	

Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. go to the site 2. log in to the site 3. Input old password. 4. Input new password 5. Confirm new password 4. click "save" to change password	New Password=optimus#11	Successful update password	As expected,	Pass

Post Condition: The updated password is stored in the database

7.5 Show books:

This Test case will check that the feature is showing Books with their corresponding details or not

Project Name: Library Management System			Test Designed by: Islam, Md. Ahasanul	
Test Case ID: bookstat01			Test Designed date: 12-06-2022	
Test Priority (Low, Medium, High): High			Test Executed by: Islam, Md. Ahasanul	
Module Name: show Books			Test Execution date: 15-06-2022	
Test Title: show all the books with their corresponding details				
Description: Test all the books are visible or not				
Precondition (If any):				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. go to the site 2. log in to the site 3. go to the books page		Can see all the books with details	As expected,	Pass
Post Condition:				

7.6 Book Status:

This feature shows the status of the book in borrow or not

Project Name: Library Management System			Test Designed by: Islam, Md. Ahsanul	
Test Case ID: bookstat01			Test Designed date: 12-06-2022	
Test Priority (Low, Medium, High): High			Test Executed by: Islam, Md. Ahasanul	
Module Name: Status			Test Execution date: 15-06-2022	
Test Title: Show the book is under borrow or not				
Description: Book Status				
Precondition (If any): The book name should be available into the book database				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Go to check book status 2. In search box, give a book name 3. Press check status	1.Book Name: Teach yourself C++ 2.Current Status: Borrowed	Show a message that book is under borrow	As expected,	Pass
Post Condition:				

7.7 Add or remove a book:

This test is executed to add or remove a book from the central database via admin

Project Name: Library Management System			Test Designed by: Sheikh Muhtasim Nasif	
Test Case ID: add_remove_book			Test Designed date: 013-06-2022	
Test Priority (Low, Medium, High): High			Test Executed by: Sheikh Muhtasim Nasif	
Module Name: Add Remove			Test Execution date: 16-06-2022	
Test Title: Add or remove a book				
Description: Book Status				
Precondition (If any): Add and remove the book into central database				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1.Go to login 2.Give Id and password as a valid admin 3.Go to Add Book/ Remove book in central database 4.Type the book name and quantity to be added or removed 5.Press add or remove	1.Book Name: Data 2.Communication and Network 3.Book Amount: 25 4.Press Add	The central database will be updated and show the add or removed book from the central database page	As expected,	Pass
Post Condition: The login session is going to be stored into the central login database and the book adding time will be written at the book add column with time and date.				

7.8 Library Card issue:

It will generate a library card for the users

Project Name: Library Management System			Test Designed by: Sheikh Muhtasim Nasif	
Test Case ID: add_remove_book			Test Designed date: 13-06-2022	
Test Priority (Low, Medium, High): Low			Test Executed by: Sheikh Muhtasim Nasif	
Module Name: Card			Test Execution date: 16-06-2022	
Test Title: Card Generate				
Description: An existing user will be applied for the card				
Precondition (If any): The librarian must be a valid and username must be inside the user database				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Go to Login 2. Login as librarian using valid name and Id 3. Press the card generate 4. Give the username into card generate form 5. Press Go	1. Username: Mr Jack 2. Status in database: Valid	Card is issued and a date or card delivery inside the dashboard	As expected,	Pass
Post Condition:				

7.9 Feedback

User can give feedback or complain about the books or library stuff

Project Name: Library Management System			Test Designed by: Sheikh Muhtasim Nasif	
Test Case ID: Feedback_01			Test Designed date: 14-06-2022	
Test Priority (Low, Medium, High): Low			Test Executed by: Sheikh Muhtasim Nasif	
Module Name: Feedback			Test Execution date: 17-06-2022	
Test Title: Feedback				
Description: The user will give a feedback into the box				
Precondition (If any): User must be validated				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Log in as user 2. Go to feedback page 3.Write feedback	User Feedback: The books pages were torn apart	A complain will be generate into complain database which is accessible by the admin only	As expected,	Pass
Post Condition:				

7.10 Remove a user:

This test case will check can we remove an existing user or not

Project Name: Library Management System			Test Designed by: Sheikh Muhtasim Nasif	
Test Case ID: Remove user			Test Designed date: 15-06-2022	
Test Priority (Low, Medium, High): High			Test Executed by: Sheikh Muhtasim Nasif	
Module Name: Remove			Test Execution date: 18-06-2022	
Test Title: Remove a user				
Description: The admin can remove a user				
Precondition (If any): The admin must have a valid account and the user to be removed must be on the user database				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Log in as admin 2. Go to user information page 3. Search the username and ID to be removed 4.Press remove	1. Username: Mr X 2. ID; 125478	The user will be removed from the user database	As expected,	Pass
Post Condition:				

8. ITEM PASS/FAIL CRITERIA

Here we have implemented a total of 10 test cases. At first, when applied the test case to system 70% of the test cases were passed successfully and 30% were fail. The test cases were failing due to some query related issues on the database. When the test case was applied after solving query related problem, all the test cases are successfully passed.

9. TEST DELIVERABLES

- Test plan
- Test results documents
- Test summary
- Errors
- Bug report

10. STAFFING AND TRAINING NEEDS

This section shows how to staff the test jobs and prepare them for the work. Staffing is set for the duration of the project. It's realistic to assume that the vast majority of the staff will agree to do some testing. The following occupations are recognized:

Project Manager: Responsible for maturing the complete execution of the Web website. This includes creating requirements, managing the seller relationship, and overseeing the testing cycle.

Test Manager: Responsible for fostering the expert test strategy, examining the test deliverable, dealing with test cycles, collecting measurements and reporting progress to the Project Manager, and recommending when testing should be completed.

Test Engineer: Planning tests, creating test methods, creating test information, running tests, preparing occurrence reports, examining episodes, writing mechanized test strategies, and detailing measurements to the test administrator are all responsibilities of this position.

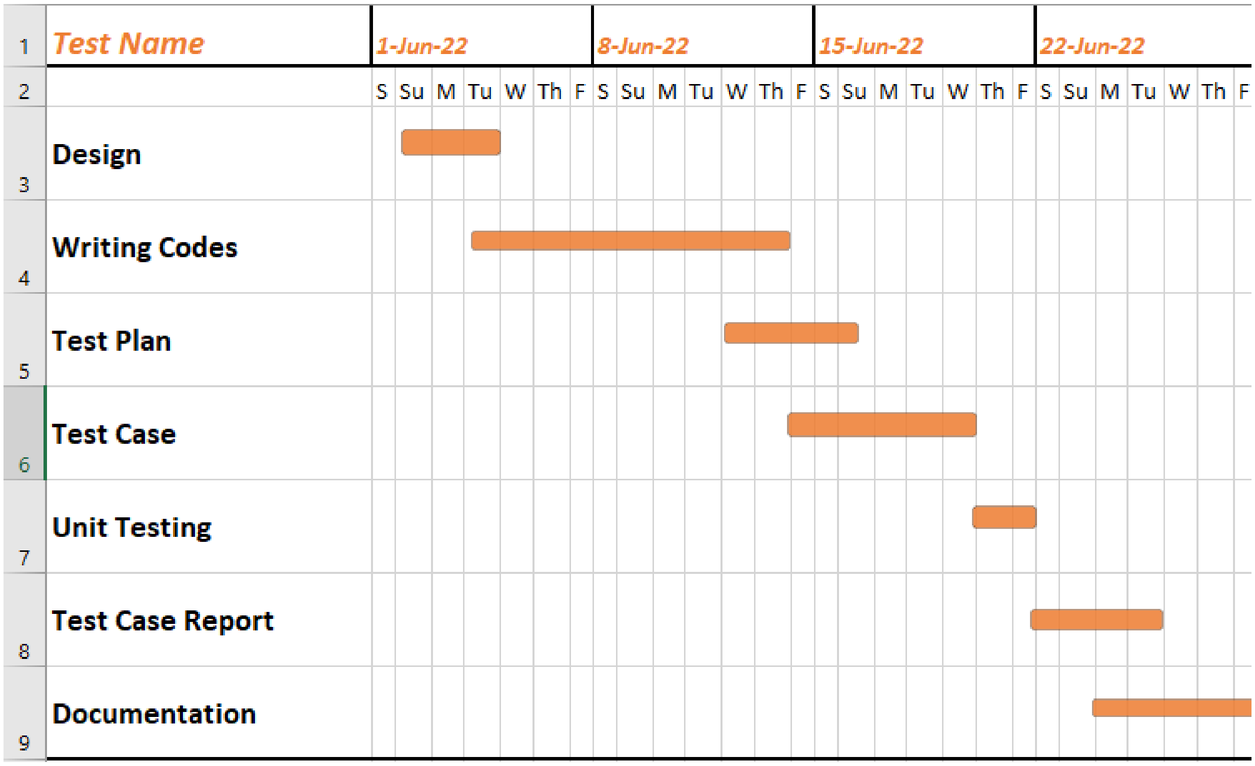
The test manager and test specialists should be familiar with the website development life cycle. Because this project is being developed in a traditional manner, this is a nonexclusive depiction of Staffing and Training requirements. As a result, the names of conscious people for each project aren't given.

11. RESPONSIBILITIES

Name	Role	Responsibilities
FARHAN ISRAK MAHMUD	Project Manager	1. Completion of the Project. 2. Execute the test cases and report defects. 3. Control whole project.
ASRAFUL ISLAM	Quality Analyst	1. Creation of test plans, test forms and test information. 2. Carry out testing as per the characterized methods. 3.Prepare all reports related to program testing carried out.
SHEIKH MUHTASIM NASIF	Programmer	1. Researching, designing, implementing, test cases and managing software programs. 2. Writing and implementing efficient code. 3. Deploying software tools, processes, and metrics.
ISLAM, MD. AHASANUL	Test Engineer	1. Characterizing the testing activities All obligations of test planning 2. To check in the event that the group has all the fundamental assets to execute the testing exercises. 3. Prepare the report of testing activities. 4. Overhauling extend directors frequently around the advance of testing exercises. 5. Develop test cases and prioritize testing activities. 6. Execute all the test cases and report defects. 7. make documentation

12. TESTING SCHEDULE

Task Name	Duration	Responsible
Design	3 days	Project Manager
Writing code	10 days	programmer
Test plan	2 days	Quality Analyst & Test Engineer
Test case	4 days	Quality Analyst
Unit testing	2 days	Project Manager & Test Engineer
Test case report	4 days	Test Engineer
Documentation	5 days	Test Engineer



13. PLANNING RISKS AND CONTINGENCIES

Risk	Probability	Impact	Mitigation
Error in function	Medium	Medium	Test the web app frequently and maintain daily backup.
Give invalid input	High	high	Tell user to use right data type in each input field
Loss of encrypted data(password)	Medium	high	Maintain security check and backup

14. APROVALS

Project Sponsor	Batighor Library
Development Management	Farhan Israk Mahmud
EDI Project Manager	Islam, Md. Ahasanul
RS Test Manager	Asraful Islam
RS Development Team Manager	Islam, Md. Ahasanul
Reassigned Sales	Farhan Israk Mahmud
Order Entry EDI Team Manager	Asraful Islam