**ONLINE LIBRARY MANAGEMENT SYSTEM**

A report submitted in partial fulfillment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY**

in

**ELECTRONICS AND COMMUNICATION ENGINEERING**

By

**SURAJ SINGH NEGI (IIITU17253)**

**RAVINA BHARTI (IIITU17247)**



**SCHOOL OF ELECTRONICS**

**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY UNA**

**HIMACHAL PRADESH**

**BONAFIDE CERTIFICATE**

This is to certify that the project titled ONLINE LIBRARY MANAGEMENT SYSTEM is a bonafide record of the work done by

SURAJ SINGH NEGI (IIITU17253)

RAVINA BHARTI (IIITU17247)

in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in ELECTRONICS AND COMMUNICATION ENGINEERING of the INDIAN INSTITUTE OF INFORMATION TECHNOLOGY UNA, HIMACHAL PRADESH, during the year 2020 - 2021.

Under the guidance of

MR. SATISH KUMAR

Project viva-voce held on: 17/5/2021

**ORIGINALITY / NO PLAGIARISM DECLARATION**

We certify that this project report is our original report and no part of it is copied from any published reports, papers, books, articles, etc. We certify that all the contents in this report are based on our personal findings and research and we have cited all the relevant sources which have been required in the preparation of this project report, whether they be books, articles, reports, lecture notes, and any other kind of document. We also certify that this report has not previously been submitted partially or as whole for the award of degree in any other university in India and/or abroad.

We hereby declare that, we are fully aware of what constitutes plagiarism and understand that if it is found at a later stage to contain any instance of plagiarism, our degrees may be canceled.

**SURAJ SINGH NEGI (IIITU17253)**

**RAVINA BHARTI (IIITU17247)**

**ABSTRACT**

Our project Online library management system is a web application for assisting librarian in managing book library in an organization. This enables the admin to enter the record of new books and retrieve the details of books available in the library. We can issue the books to the students and maintain their records and can also check how many books are issued and stock available in the library.. It allows students to apply various sorts of filters to find the required book using filters such as category, author’s name.

In our project, we have used HTML, CSS and Bootstrap for front-end development. We have our server side built using NodeJS to provide the run time environment. It is one of the most popular and fast growing open source web platforms currently available. MongoDB is used for the database to store the details and data of the users. MongoDB is a general purpose, document-based, distributed database built for modern application developers and for the cloud era.

***Keywords*:** NodeJS [1], ExpressJS [2], MongoDB [3]

**ACKNOWLEDGEMENT**

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

We would like to express our sincere gratitude and heartfelt thanks to MR. SATISH KUMARfor his unflinching support and guidance, valuable suggestions and expert advice. His words of wisdom and expertise in subject matter were of immense help throughout the duration of this project.

We also take the opportunity to thank ourDirectorand all the faculty of School of Electronics, IIIT Una for helping us by providing the necessary knowledge base and resources.

We would also like to thank our parents and friends for their constant support.

**SURAJ SINGH NEGI (IIITU17253)**

**RAVINA BHARTI(IIITU17247)**

**TABLE OF CONTENTS**

**Title Page No.**

**ABSTRACT 4**

**ACKNOWLEDGEMENT 5**

**TABLE OF CONTENTS 6**

**LIST OF ACRONYMS 8**

**LIST OF TABLES 9**

**LIST OF FIGURES 10**

**LIST OF APPENDICES 11**

**1 Introduction 12**

1.1 What is online library management system? 12

1.2 How to issue a book? 12

1.3 How to add/delete or update books? 12

**2 Review of Literature 13**

2.1 Previous Works 13

**3 Motivation 14**

**4 Proposed work 15**

4.1 Applications and Technologies used 15

4.1.1 NodeJS 15

4.1.2 ExpressJS 15

4.1.3 MongoDB 16

4.2 How we started? 17

4.2.1 Main functionalities 18

4.2.2 Admin 22

**5 Conclusions, Future works and Learning Outcomes 24**

**References 25**

**Appendices 26**

**LIST OF ACRONYMS**

**DBMS** Database Management System

**env** Environment Variables

**config** Configuration

**npm** Node Package Manager

**json** Javascript Object Notation

**api** application program interface

**LIST OF TABLES**

**LIST OF FIGURES**

**4.1** Index page of deployed website 7

**4.2** Signup page 8

**4.3** Our books page 9

**4.4** Profile page 10

**4.5** Admin page 11

**4.6** Add book Form 12

**LIST OF APPENDICES**

**A.1** Starting a Web Server 27

**A.2** Defining schema for book 27

**A.3** Defining schema for student 29

**A.4** Connection of database 29

**A.5** APIs 30

**A.5.1** API for signup30

**A.5.2** API to view books33

**A.5.3** API to show profile35

**Chapter 1**

**Introduction**

* 1. **What is Online library management system?**

Online library management system is a web application designed to help librarian in hassle free management of book library in an organisation by using various features such as add/delete/update books and view student records. It also allows students to search for availability of books, checks books issued against their name and due date of the return of these books from the comforts of their room. Students can also apply filters while searching for books such as category, author’s name to make searching convenient.

* 1. **How to issue a book?**

Online library management system provides an option to login with your username and password, given that you have already registered on the website. It provides a signup page where you have to fill in some details to have your account set up. Once registered on the website you are eligible to login. Once logged in, users can find required books by applying proper filters. Thereafter he can issue the book.

* 1. **How to add/delete or update books?**

To add/delete or update books from the database admin/librarian will have to login with his admin username and password. Post login the admin can add/delete or update books from the database.

**Chapter 2**

**Review of Literature**

**2.1 Previous Works**

There are various other web applications which provide library management facilities over the internet. In our work, we have included the latest web technologies and compiled into our project.

Early mechanization came in 1936, when the University of Texas began using a punch card system to manage library circulation. In the 1960s due to the growth of computer technologies library automation was born. During the 1970s-1980s integrated library management systems appeared which included hardwares and softwares. In the 2000s due to growth of the internet, online web based portals came into existence to empower users. It allowed users to reserve or renew books and authenticate themselves to library subscribed online databases.

Our project also serves as a database management system for the institution maintaining the library. DBMS makes it possible for users to create, edit and update data in database files. Once created, the DBMS makes it possible to store and retrieve data from those database files.

**Chapter 3**

**Motivation**

Inspiration for the project came from our own campus library experiences. Our institution uses a manual system for issuing books. Students have to physically visit the library to search for books and to view their records. Such manual system is ineffective due to the following bottleneck and drawback associated with it-

* Students had to visit the library to search for books..
* Students had to be physically present to check their profile.
* Tiresome process to check for availability of books.
* Central library system present in the campus is restricted only to library premises.
* Manual entries can sometimes lead to discrepancies.

Therefore it calls for the need of an automated online car booking and database management system. It provides a convenient method for users to book a car. It also allows the institution to effectively manage their library.

**Chapter 4**

**Proposed Work**

**4.1 Applications and Technologies Used**

**4.1.1 NodeJS**

Node**.**js [1] is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux. Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server- and client-side scripts.

**4.1.2 ExpressJS**

Express.js [2] is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It is an open source framework developed and maintained by the Node.js foundation.

Some of the core features of Express framework:

* It can be used to design single-page and multi-page and hybrid web applications.
* It allows to setup middleware to respond to HTTP Requests.
* It allows to dynamically render HTML Pages based on passing arguments to templates.

Various dependencies used:

* dotenv
* ejs (Embedded JavaScript Templates)
* cookie-parser
* nodemailer
* mongoose
* bcrypt
* passport
* body-parser, and so on.

**4.1.3 MongoDB**

MongoDB, the most popular NoSQL database, is an open-source document-oriented database. The term ‘NoSQL’ means ‘non-relational’. It means that MongoDB isn’t based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data. This format of storage is called BSON ( similar to JSON format) .

MongoDB is available under General Public license for free, and it is also available under Commercial license from the manufacturer.

**4.2 How we started?**

Firstly the project started by building its backend server, creating various APIs to receive request by the frontend part of the website. The backend server is built using the most common and widely used framework ExpressJS which runs on the environment served by NodeJS. Initially the server runs on the localhost during the development phase of the project. We used a port of 5000 to run and develop the website locally.

That is, the link to the development site was:

[http://localhost:5000/](http://localhost:3000/)

Currently we have deployed the project on the platform Heroku which provides free deploying service for any projects based on NodeJS, Flask, Django, etc.

The link to which the project has been deployed is

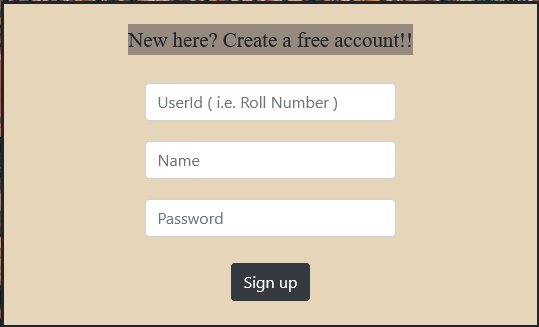
https://online-library-iiituna.herokuapp.com/

**4.2.1 Main Functionalities**

****

**Figure 4.1 Home Page**

The above image shows the index/home page of our web application. It has options such as to create a new account or login using your username and password. It also provides an option for admin to login using admin credentials.

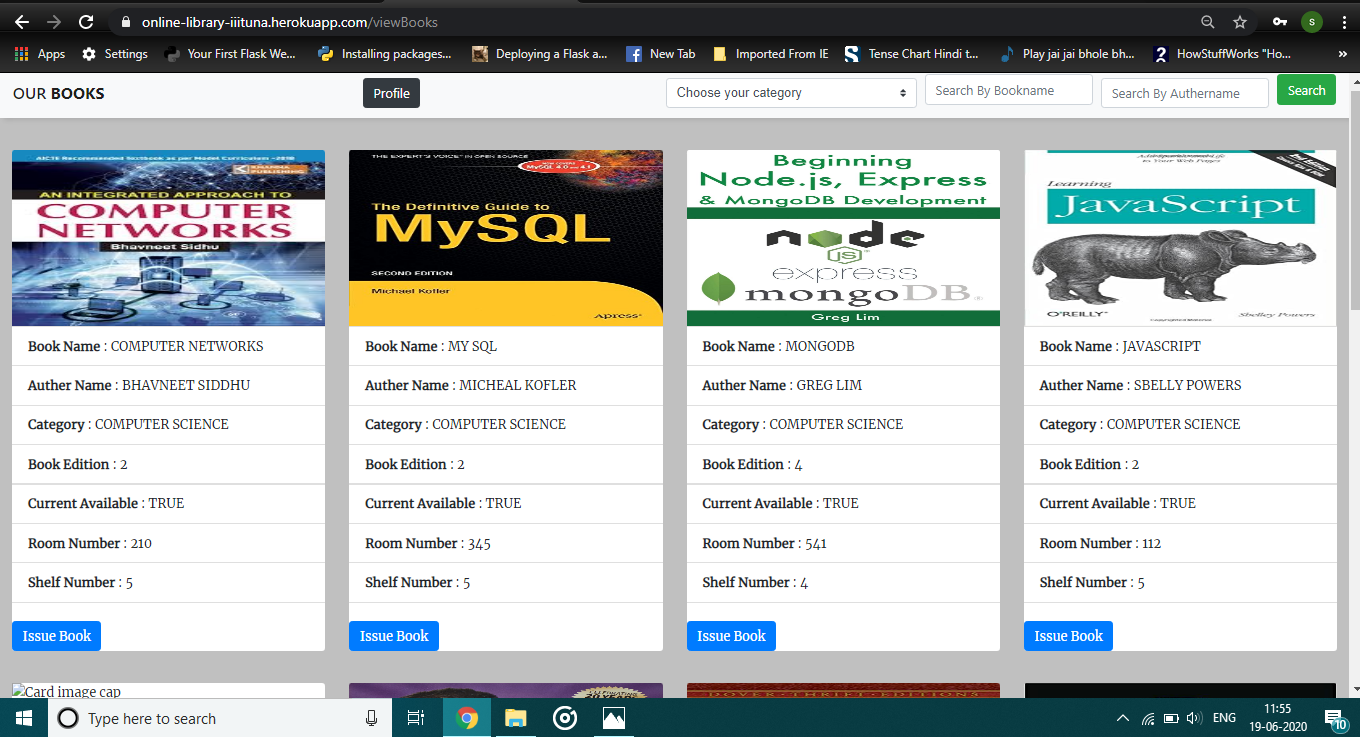


**Figure 4.2 Sign Up Page**

The users could login to the website or to signup using your details. The details include choosing your User Id which is student’s roll number, name and password. Once signed up you can login using the User Id and password created using signup function. User Ids used for any two users are different so every user has a unique username to avoid mistakes. This will avoid any cases of duplicate reservations made on the website.

User will have to use the same login credentials each time he wants to login.

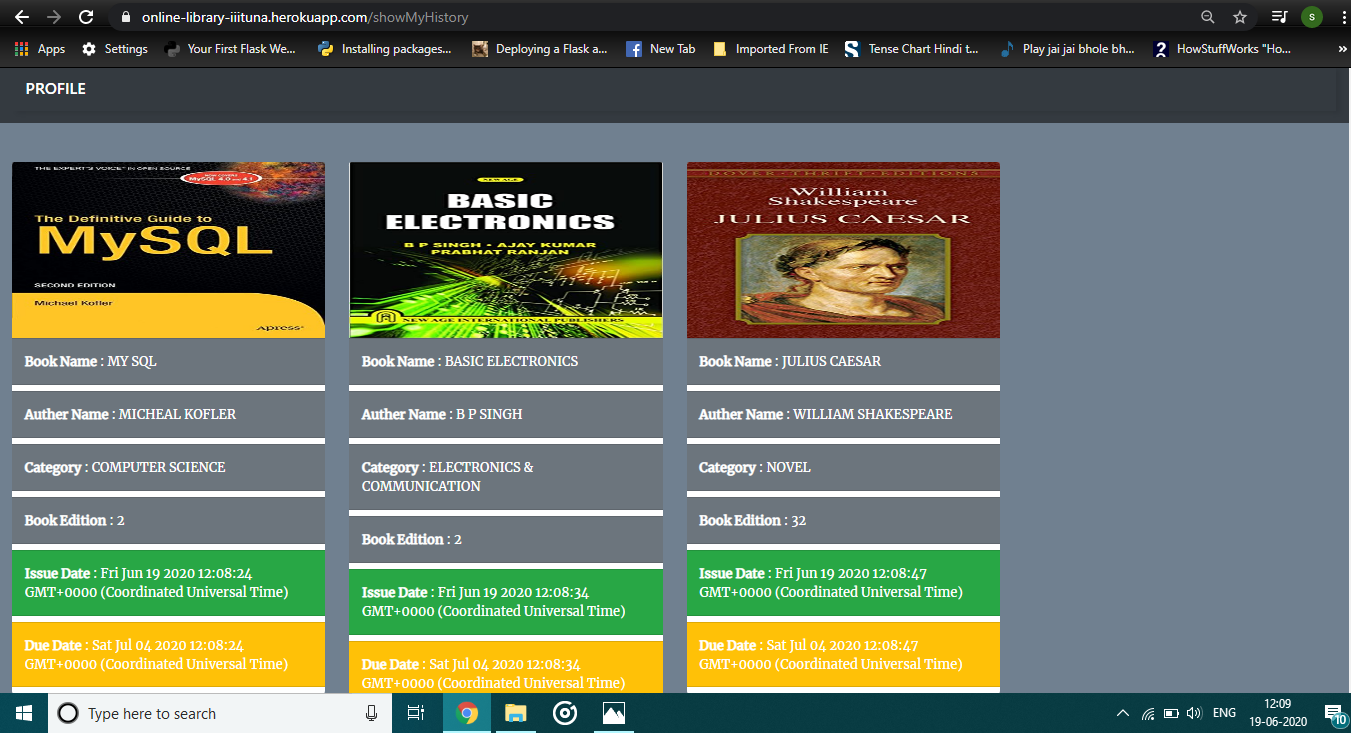
Once you are logged in using your credentials, it will show you all the available books in the library. Then users can apply various filters to find the required book.



**Figure 4.3 Our books page**

The available filters in this web application include categories of the book you are looking for such as computer science, novels, magazines etc. Another filter allows you to search books directly by entering the particular book name. The user can also filter books on the basis of author name to get books written by a particular author. These filters make finding a required book very easy, fast and less cumbersome as compared to manually searching for books in a library.

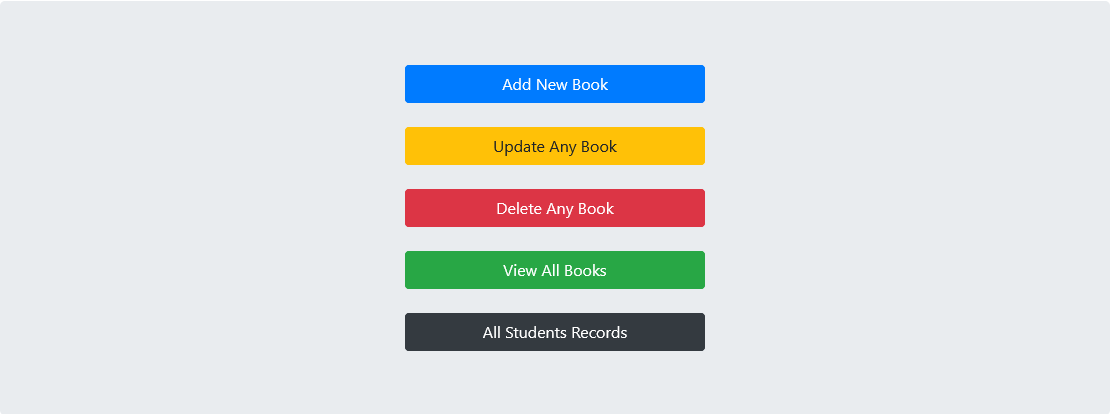
Once you have found the required book then you can issue a book by clicking on issue book button.



**Figure 4.4 Profile page**

The profile tab can be used by the user to view the list of books issued against his name. It also shows the due date against each of the issued books so as to remind the student to return the book in time to avoid late fines.

**4.2.2 Admin**

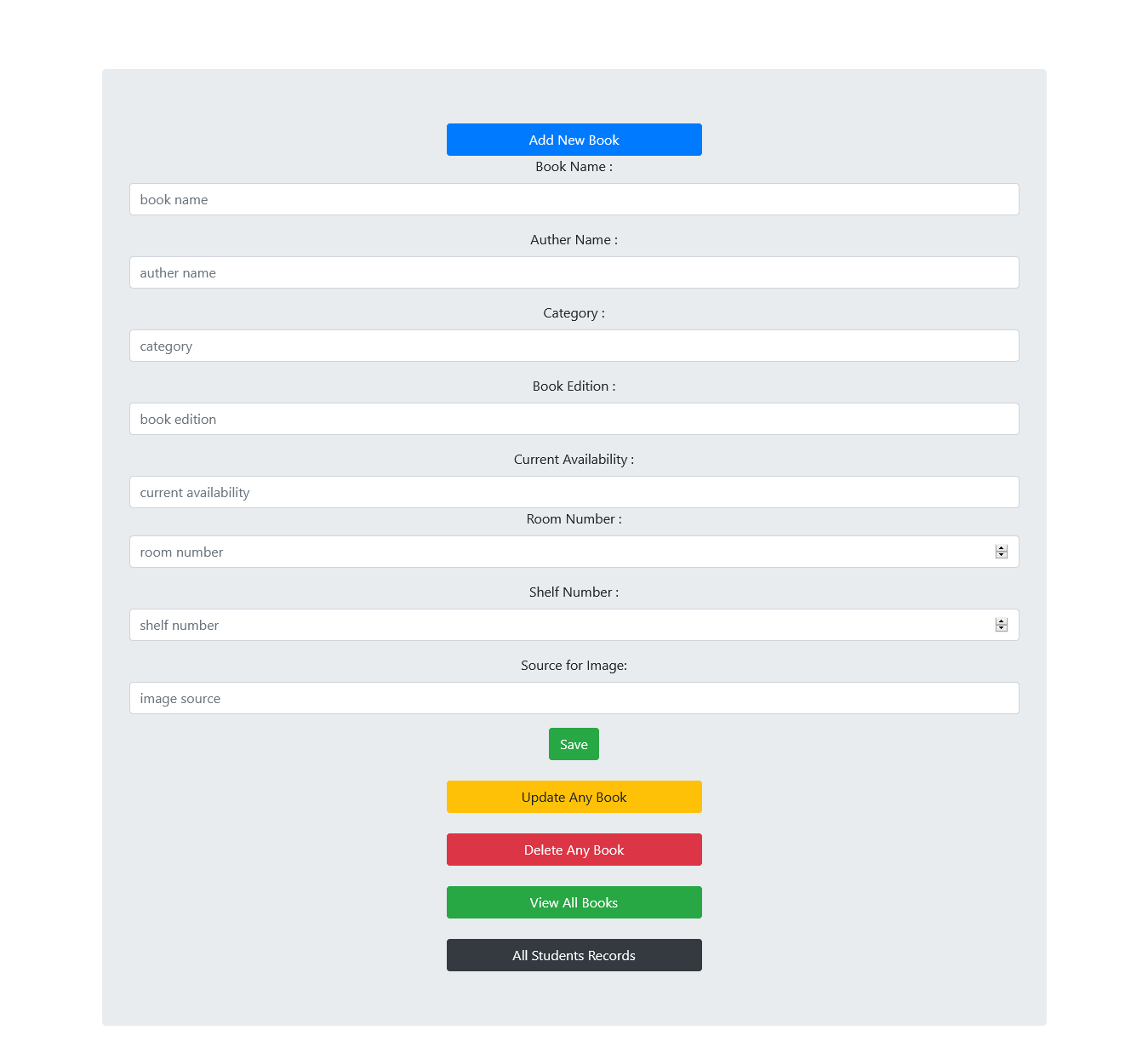
****

**Figure 4.5 Admin Edits page**

Another feature of this application allows admin to add, update, delete or view books from the database. For this admin will have to login using his admin username and password. Then the admin can add, update or delete books by selecting corresponding buttons.

To delete a book, the admin will have to provide a unique book id and the corresponding book will be deleted from the database. In updating any book information, the admin will have to provide a unique book id and provide new information about the book.

When the admin wants to add a book then he can do so by clicking the Add New Book button then the web application will provide the admin with a form which will ask information about the new book. Post providing information web application will attach a unique book id and add it to the database.



**Figure 4.6 Add book form**

**Chapter 5**

**Conclusions, Future Work, and Learning Outcomes**

Node.js is primarily a JavaScript runtime that is powered by V8, developed by Google for use in Chrome. V8 has the ability to compile and execute JavaScript at lightning fast speed, mainly because it compiles JavaScript into a native machine code. Node.js is not perfect, but it is a great tool for building server-side web applications in JavaScript. NodeJS is always evolving and expanding its use. It is relatively simple to get into while good enough to handle today’s web development tasks. Another key benefit of Node.js is language re-use. Other web technologies like Spring or ASP.NET requires developers to have knowledge in another language to write code on the server-side, be it VB.NET, Java, or C#.

During this project we have developed a basic understanding of NodeJs and other web development tools and technologies. There’s a lot more to learn about, including session management, token authentication, API design, etc. Future work in our project may include adding a reminder system so as to remind students to return their books in time and providing payment interface to online submit the late fees. These are the works to be added in the future.

We learned a lot during the course of the project and we still have a lot to learn in this everyday changing field. There are various resources over the internet to learn new things and implement them, even if we did our studies over the internet. Learning process increased our knowledge in various fields of web development and increased our interest in the ever growing field of web development.

**References**

|  |  |
| --- | --- |
| [1] | R. Dahl, “NodeJS,” [Online]. Available: https://nodejs.org/en/. |
| [2] | S. TJ Holowaychuk, “Fast, unopinionated, minimalist web framework for Node.js,” [Online]. Available: https://expressjs.com/. |
| [3] | “The database for modern applications,” MongoDB Inc., [Online]. Available: https://www.mongodb.com/. |
| [4] | “Essential JavaScript development tools,” [Online]. Available: <https://www.npmjs.com> |
|  |  |

**Appendices**

**Appendix A**

**Code Attachments**

The following is the partial / subset of the code. Code of some module(s) have been wilfully suppressed.

**A.1 Starting a Web Server**

const express = require('express');

const app = express();

app.set('port', process.env.PORT || 5000);

app.listen(app.get('port'), function (err) {

if (err)

console.log(err);

console.log('Running on http://localhost:%s', app.get('port'));

});

**A.2 Defining schema for book**

// DEFINE SCHEMA FOR BOOK

var bookSchema = new mongoose.Schema({

category: {

type: String,

required: [true, "Please! provide \*\*Category\*\* of this book to add into database ?"]

},

edition: {

type: Number,

required: [true, "Please! provide \*\*Edition of Book\*\* to add this book into database ?"]

},

bookName: {

type: String,

required: [true, "Please! provide \*\*BookName\*\* to add this book into database ?"]

},

autherName: {

type: String,

required: [true, "Please! provide \*\*AutherName\*\* to add this book into database ?"]

},

currentAvailable: {

type: String,

required: [true, "Please! provide \*\*Current Availability\*\* to add this book into database ?"]

},

bookImage: String,

dueDate: Date,

issueDate: Date,

roomNumber: Number,

shelfNumber: Number

});

**A.3 Defining schema for student**

// DEFINE STUDENT SCHEMA

var studentSchema = new mongoose.Schema({

studentName: String,

username: String,

password: String,

data: Array

})

**A.4 Connection for database**

// URL OF MONGODB ATLAS

const url = 'mongodb+srv://dev\_chauhan\_10:mypassword123@myproject-be1gc.mongodb.net/test?retryWrites=true&w=majority';

mongoose.Promise = global.Promise;

// CONNECT MONGOOSE TO MONGODB ATLAS

mongoose.connect(url, {

useNewUrlParser: true

});

var db = mongoose.connection;

db.on('error', function (err) {

throw err;

});

**A.5 APIs**

**A.5.1 API for Signup**

// SIGNUP API FOR USER TO REGISTER ON WEBSITE

app.post('/signup', function (req, res) {

if (req.body.username) {

studentModel.findOne({

username: req.body.username

})

.exec()

.then((doc) => {

if (doc) {

res.send("Sorry!!..This \*\*userId\*\* is already taken..Try some other");

} else {

if (req.body.password) {

bcrypt.hash(req.body.password, saltrounds, function (err, hash) {

if (err) {

res.send("Some error occurred!!..Student not registered");

} else {

var student = new studentModel({

studentName: req.body.studentName,

username: req.body.username,

password: hash

});

student.save(function (err) {

if (err) {

res.send("Some error occurred!!..Student not saved");

} else {

res.redirect(307,'/viewBooks');

}

});

}

});

} else {

res.send("Warning!!..You must provide \*\*password\*\* to register");

}

}

})

.catch((err) => {

res.send("Some error occurred !!.....Please try again after sometime");

});

} else {

res.send("Warning!!..UserId cannot be empty");

}

});

**A.5.2 API for viewing books**

app.post('/viewBooks', async function (req, res) {

var allBooks = await new Promise(function (resolve, reject) {

var temp = bookModel.find({});

resolve(temp);

});

if (req.body.bookName) {

allBooks = allBooks.filter(function (book) {

if (req.body.bookName.toUpperCase() == book.bookName.toUpperCase()) {

return book;

}

});

}

if (req.body.autherName) {

allBooks = allBooks.filter(function (book) {

if (req.body.autherName.toUpperCase() == book.autherName.toUpperCase()) {

return book;

}

});

}

if (req.body.category) {

allBooks = allBooks.filter(function (book) {

if (req.body.category.toUpperCase() == book.category.toUpperCase()) {

return book;

}

});

}

allFilteredBooks = allBooks.map(function (book) {

var filteredBook = {

"bookId": book.\_id,

"bookName": book.bookName,

"autherName": book.autherName,

"category": book.category,

"currentAvailable": book.currentAvailable,

"edition": book.edition,

"bookImage": book.bookImage,

"roomNumber": book.roomNumber,

"shelfNumber": book.shelfNumber

};

return filteredBook;

});

if (allFilteredBooks == null)

res.send("No such Book exists for these filters..!!");

else

res.render("allBooks", { allFilteredBooks: allFilteredBooks });

});

**A.5.3 API to show profile of particular student**

// API TO SHOW PROFILE OF PARTICULAR STUDENT

app.post('/showMyHistory', async function (req, res) {

if( !req.user ){

res.send("You are not Logged in");

}

else{

var myUser = await studentModel.findOne({ username: req.user })

.exec()

.then((doc) => {

return doc;

})

.catch((err) => {

res.send("Some error occurred !!.....Please check your \*\*User Id\*\* again");

});

var userHistory = myUser.data.map( function (book) {

var tempdata = {

"bookName": book.bookName,

"autherName": book.autherName,

"category": book.category,

"currentAvailable": book.currentAvailable,

"edition": book.edition,

"bookImage": book.bookImage,

"issueDate": book.issueDate,

"dueDate": book.dueDate

};

return tempdata;

});

if (userHistory == null)

res.send("No history here..!!");

else

res.render('profile', { userHistory: userHistory });

}});