PROJECT REPORT

My Favourite Books - '*A RESTful Web App*'

Submitted by

Alfiya Fazil

ADIT/TVM/19/001

ADIT (2019-2021)

National Skill Training Institute For Women, Trivandrum

ABSTRACT

The project titled ‘My Favourite Books - *A RESTful Web App’* is a simple RESTful Web Application built with Node.js, Express, and MongoDB. This is an easy, fast and single-page web application that allows users to keep track of their best reads. The Node.js application uses the Express framework to read from and write to a MongoDB database and it entirely eliminates the need for page refreshing and visiting separate URIs.

CONTENTS

ABSTRACT

1. INTRODUCTION

1.1. OBJECTIVE

1.2. PROJECT DESCRIPTION

1.3. SCOPE OF WORK

2. SYSTEM ANALYSIS

2.1. PROPOSED SYSTEM

3. SOFTWARE DEVELOPMENT ENVIRONMENT

4. SYSTEM DESIGN

4.1. DATA MODEL DIAGRAM

4.2. USE CASE DIAGRAM

4.3. FLOW CHART

5. SYSTEM REQUIREMENTS

5.1. SOFTWARE SPECIFICATION

5.2. HARDWARE SPECIFICATION

6. APPENDICES

6.1. SOURCE CODE

6.2. SCREENSHOTS

7. CONCLUSION

8. REFERENCE

1. INTRODUCTION

1.1 OBJECTIVE

The objective of this project is to create a single-page web application that uses Node.js as a web server and MongoDB as a database to organise the details of books that the users have previously read. This application simply allows readers to add, retrieve, and delete book information. In the long run, the application will be a useful tool for keeping track of a user's favourite books.

1.2 PROJECT DESCRIPTION

The purpose of this project is to make information on user's favourite books easily available for them. The users of this web application will be readers who can add, view and remove details on their best reads. The readers can add details such as book name, author, genre and year which will help them remember the books they have read in the past.

The application uses a web server running with Express framework, and uses app.get and app.post to communicate with both the server and the database. This simple RESTful web application which uses AJAX for all data operations and plain JavaScript for updating DOM works without a single page refresh. It stores and retrieves JSON data in a MongoDB collection using HTTP POST and HTTP GET and removes data using HTTP DELETE.

1.3 SCOPE OF WORK

Since forgetting is an all-too-common occurrence in everyday life, people tend to rely on a variety of techniques to assist them recall crucial information. In such an instance, a web application can be really handy. ‘My Favorite Books’ is a single-page web application that leverages the REST API to keep track of the users' favourite readings. This application allows users to add, view, and delete information about their favourite books. With the support of AJAX and DOM, the web application functions efficiently without requiring a single page refresh.

In future ‘My favourite books’ can be used or produced in a range of capabilities by adding more features to the application. It is not mandatory that the application be used to store book information; it can also be used to store information about movies, events, places, people, and so on. The single-page application can be scaled into a multi-page web application with additional functionalities such as editing book information and file uploads. It can even be turned into a personal memory box.

2. SYSTEM ANALYSIS

2.1 PROPOSED SYSTEM

The system proposed here is a single-page web application that assists readers in keeping track of their favourite books. Readers can add information such as the title of the book, author, genre, and year to help them recall what they've read in the past. The application communicates with the server and the database via app.get and app.post, which are both implemented in the Express framework. This is a simple RESTful web application that uses AJAX for data manipulation and plain JavaScript for DOM updates and runs without requiring a single page refresh. It uses HTTP POST and HTTP GET to store and retrieve JSON data in a MongoDB collection, and HTTP DELETE to delete data.

3. SOFTWARE DEVELOPMENT ENVIRONMENT

"My favourite books" is a fully functional Node.js RESTful web application that reads and writes to a MongoDB database and runs on the Express framework. Since Node.js is a lightweight, scalable and open-source language platform, it improves the efficiency of the development process by bridging the gap between frontend and backend applications. The Express framework, on the other hand, is built on top of the node.js framework and offers a robust set of features for building single-page, multi-page, and hybrid web applications. The goal of using MongoDB as the database is to create a data store that is high-performing, highly available, and scales automatically.

In Node.js, package is a folder tree that is specified by a package.json file. The package consists of the folder containing the package.json file and all subfolders until the next folder containing another package.json file, or a folder named node_modules. The main packages used in this project other than Express and Mongodb are Monk, Morgan and Jade. While monk is a tiny layer that provides simple yet substantial usability improvements for MongoDB usage within Node.JS, morgan is an HTTP request logger middleware for node.js. And, because Jade is an elegant templating engine that is primarily used for NodeJS server-side templating, it gives us a powerful new approach to write markup, with a number of advantages over plain HTML.

4. SYSTEM DESIGN

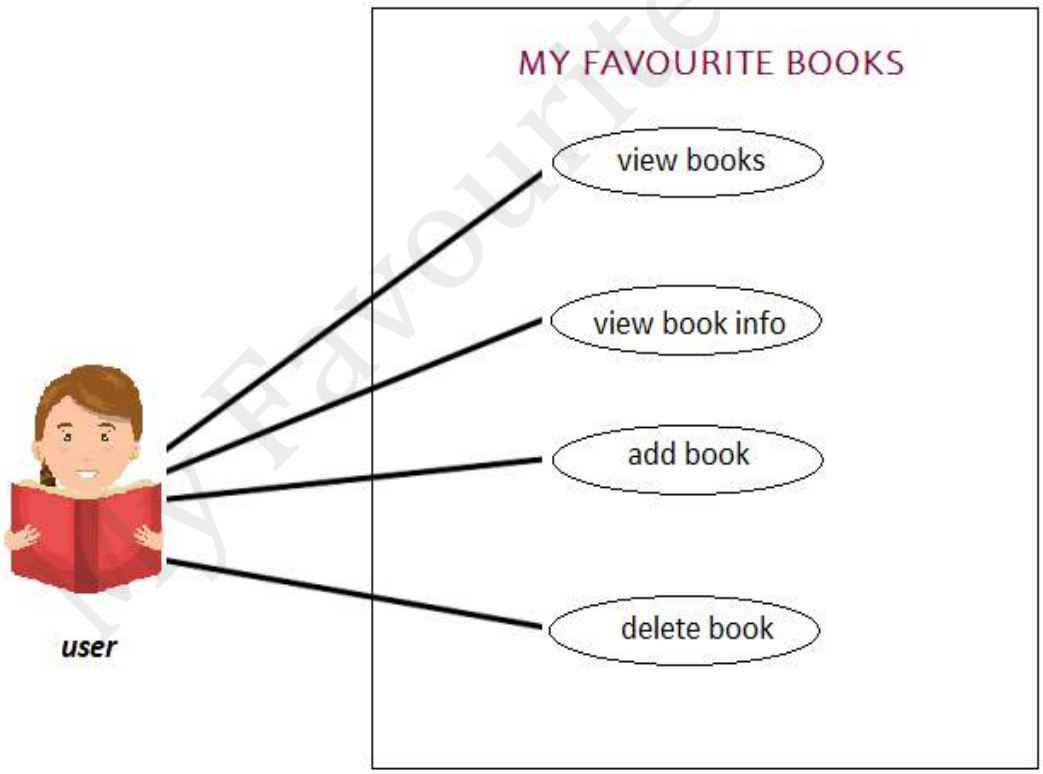
4.1 DATA MODEL DIAGRAM

NoSQL, unlike SQL which has ER and class diagrams, has neither names nor constraints for data modeling diagram(s). The obvious reason is NoSQL's lack of hard and fast relationship rules, which aims to get a developer started with minimum requirements. The diagram below represents the MongoDB databases, collections and documents used in the web application, where *my_favourite_books* is the database, *booklist* is the collection and the example that follows is a document. .


```
{
  "_id" : ObjectId("60c0aec73ac4b114848a7283"),
  "bookname" : "Harley Merlin Series",
  "author" : "Bella Forest",
  "genre" : "Fantasy",
  "year" : "2018"
}
```

4.2 USE CASE DIAGRAM

A use case diagram is a visual representation of the processes that take place within the application. It summarizes the details of the application's users and their interactions with the application. The users can view their favourite books, add books and also delete the books from the list.



4.3 FLOW CHART

A flowchart is a picture of the separate steps of a process in sequential order. It is a generic tool that can be adapted for a wide variety of purposes, and can be used to describe various processes, such as a manufacturing process, an administrative or service process, or a project plan.

5. SYSTEM REQUIREMENTS

5.1 SOFTWARE SPECIFICATION

Operating System	:	Windows 7 or above
Front End	:	HTML, CSS, JavaScript, AJAX
Back End	:	Node.js, MongoDB
Node.js Packages	:	Express, Mongodb, Monk, Morgan, Jade
Code Editor	:	Visual Studio Code
Browser	:	Google Chrome

5.2 HARDWARE SPECIFICATION

RAM	:	1 GB or above
Processor	:	1 GHz or more
Hard Drive	:	32 GB or above
Network Connectivity	:	LAN or Wi-Fi

6. APPENDICES

6.1 SOURCE CODE

1. package.json

```
{
  "name": "my_favourite_books",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "cookie-parser": "~1.4.3",
    "debug": "~2.6.9",
    "express": "~4.16.0",
    "http-errors": "~1.6.2",
    "jade": "~1.11.0",
    "mongodb": "^3.0.5",
    "monk": "^6.0.5",
    "morgan": "~1.9.0"
  }
}
```

2. app.js

```
var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');

// Database
var mongo = require('mongodb');
var monk = require('monk');
var db = monk('localhost:27017/my_favourite_books');
```

```
var indexRouter = require('./routes/index');
var booksRouter = require('./routes/books');

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

// Make our db accessible to our router
app.use(function(req,res,next){
    req.db = db;
    next();
});

app.use('/', indexRouter);
app.use('/books', booksRouter);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404));
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});
```

```
module.exports = app;
```

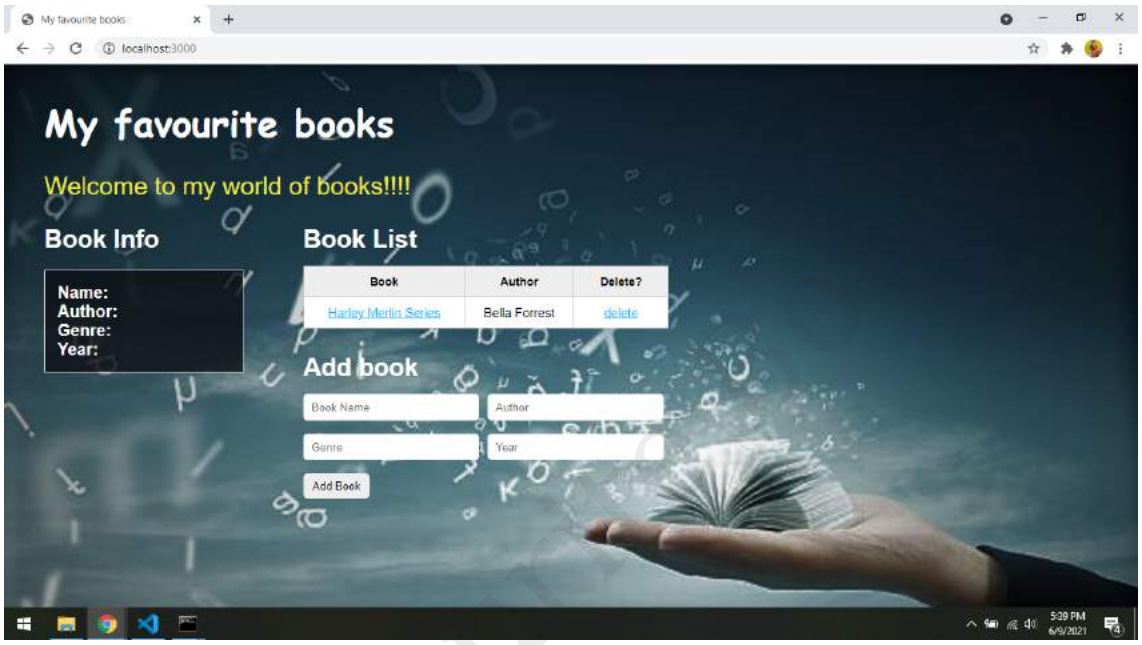
3. index.jade

```
extends layout

block content
  h1= title
  p Welcome to my world of books!!!!

  // Wrapper
  #wrapper

    // book INFO
    #bookInfo
      h2 Book Info
      p
        strong Name:
        |  <span id='bookInfoName'></span>
        br
        strong Author:
        |  <span id='bookInfoAuthor'></span>
        br
        strong Genre:
        |  <span id='bookInfoGenre'></span>
        br
        strong Year:
        |  <span id='bookInfoYear'></span>
    // /book INFO

    // book LIST
    h2 Book List
    #bookList
      table
        thead
          th Book
          th Author
          th Delete?
```

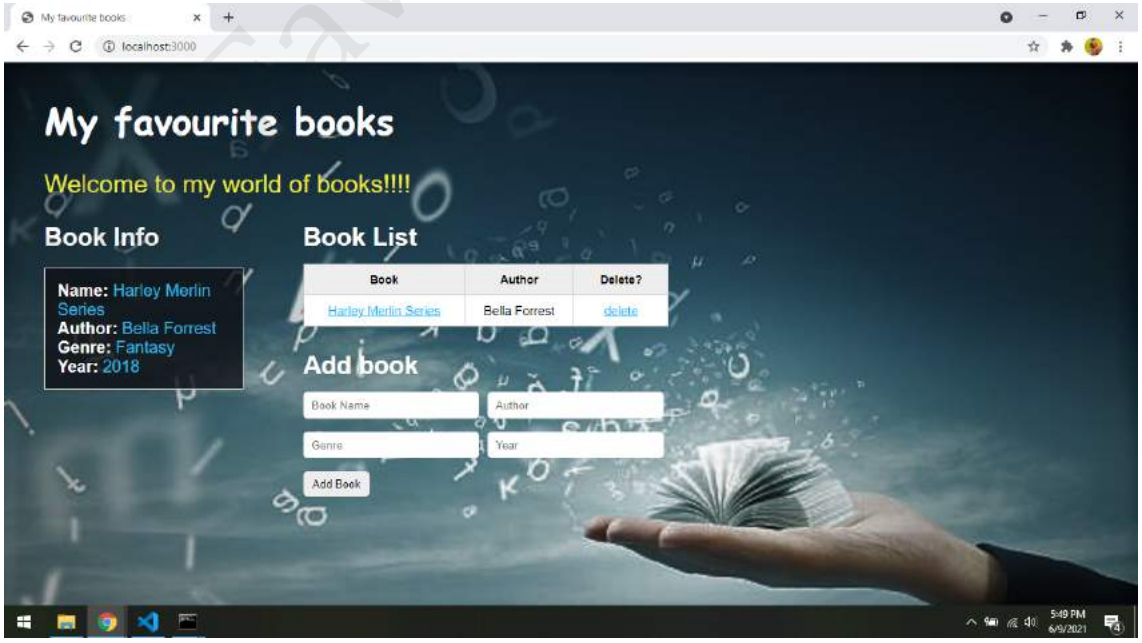
```
        tbody
    // /book LIST

    // ADD book
    h2 Add book
    #addbook
      fieldset
        input#inputbookname(type='text', placeholder='Book Name')
        input#inputbookauthor(type='text', placeholder='Author')
        br
        br
        input#inputbookgenre(type='text', placeholder='Genre')
        input#inputbookyear(type='text', placeholder='Year')
        br
        br
        button#btnAddbook Add Book
    // /ADD book

// /WRAPPER
```

6.2 SCREENSHOTS

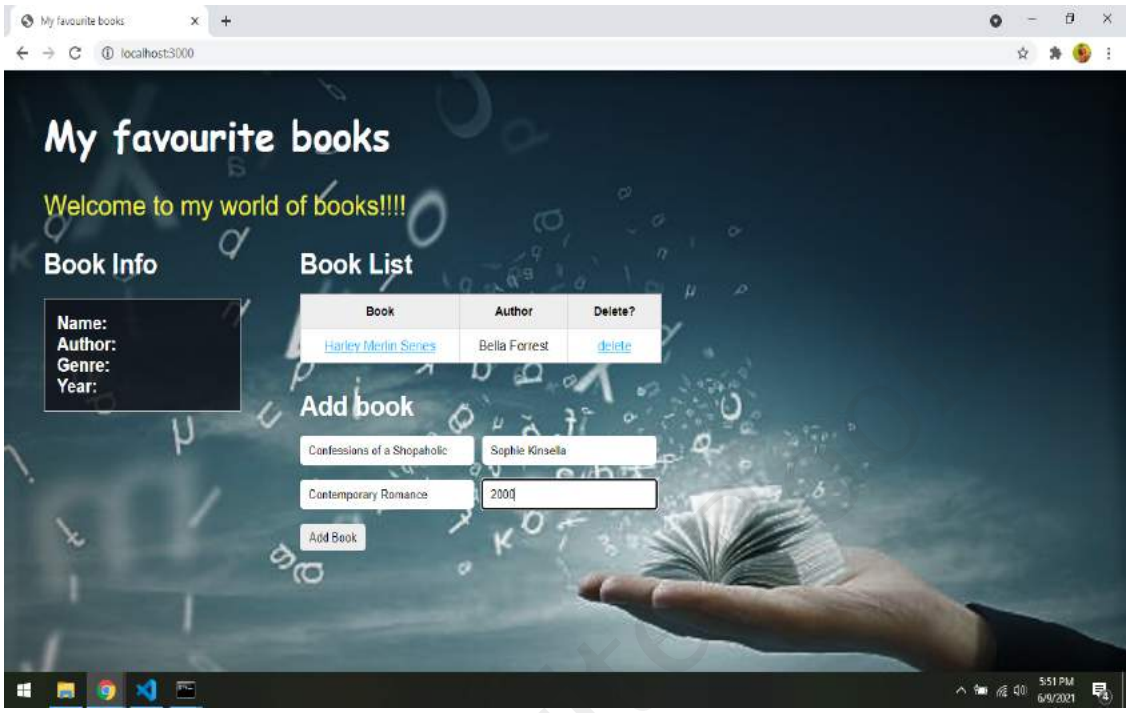
1. **Book List**



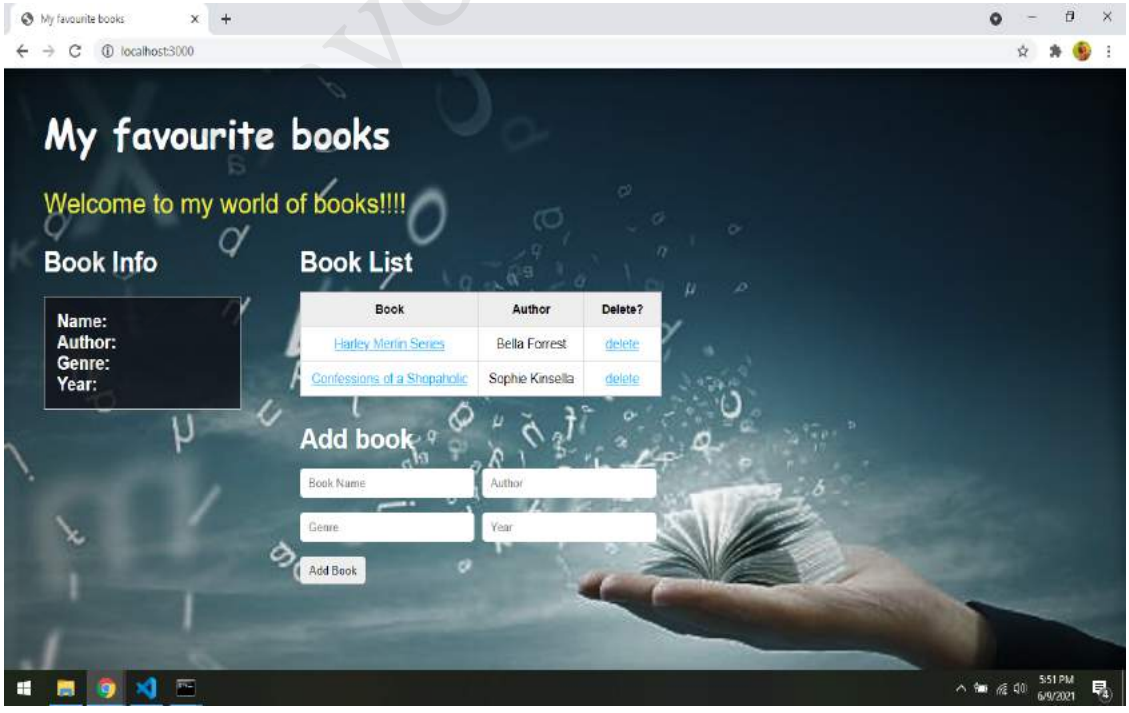
2. **Book Info**



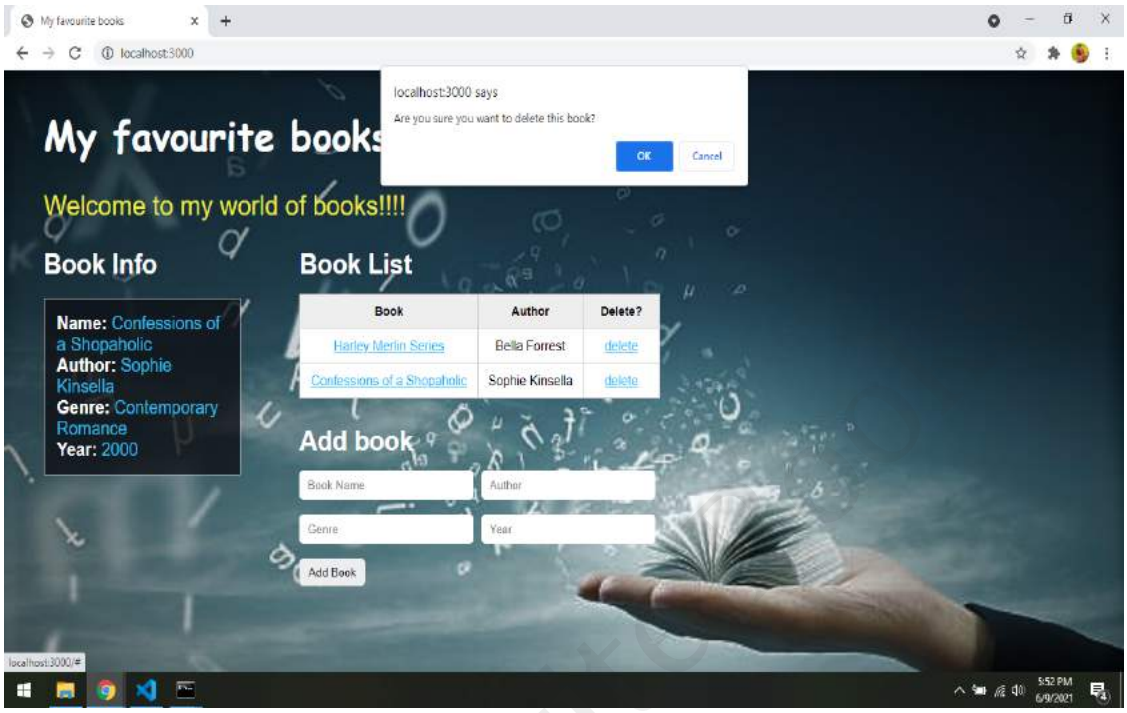
3. Add Book



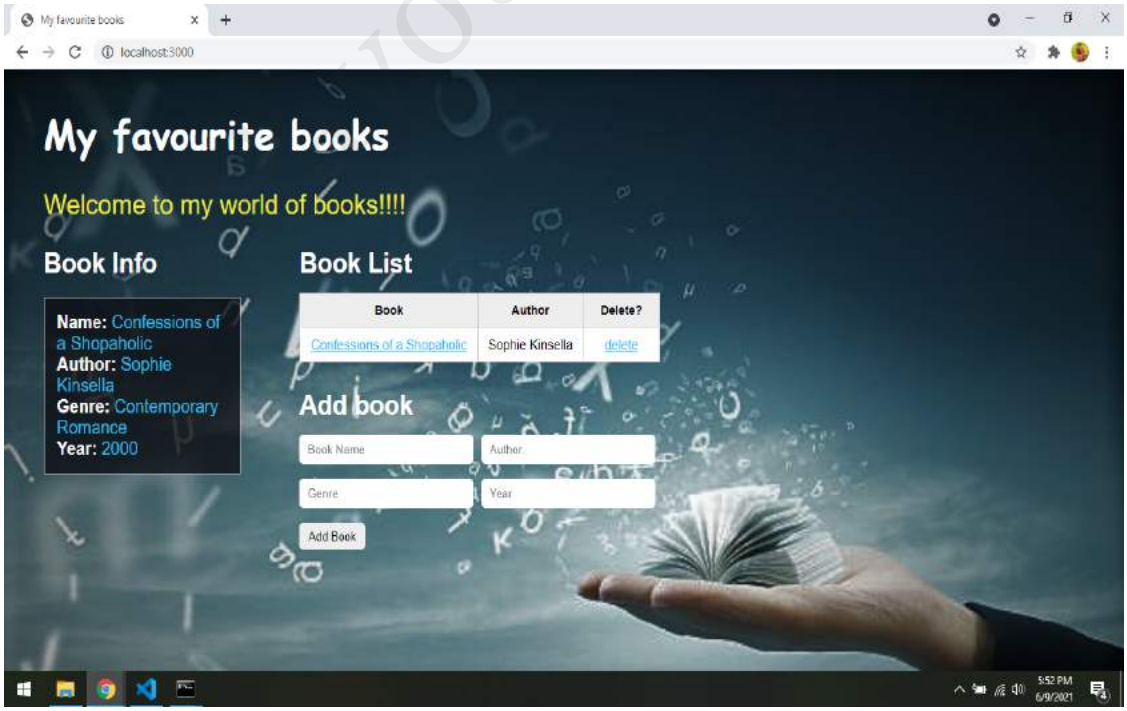
4. Book Added



5. Delete Book



6. Book Deleted



7. CONCLUSION

The goal of this project was to create a web application that would allow users to keep track of their favourite readings. Following the development and execution of the code, it may be stated that the web application's overall behaviour is satisfactory and meets the needs of the users. This project can be improved in the future because it is scalable and can accommodate numerous advanced features.

8. REFERENCE

1. https://closebrace.com/
2. https://www.geeksforgeeks.org/