A Project Report

On

BOOKSHELF

Submitted in partial fulfillment of the requirements for the

award of the degree of

(DEGREE)

in

Session: 2020-2022

By

RUPESH GAUR (20MCA1316)

CHINU RATHI (20MCA1351)

Supervised By

Amandeep Kaur

DESIGNATION

University Institute of Computing

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI, PUNJAB,140413

Project Coordinator **Head-UIC**

2021-2022

DECLARATION

We, Rupesh Gaur(20MCA1316) and Chinu Rathi (20MCA1351)
20MCA 2(A) student of Chandigarh University , hereby declare that the Project Report entitled **"BookShelf"** is an original work and data provided in the study is authentic to the best of our knowledge under the supervision of **Ms. Amandeep Kaur**.

Place:

Date:

Signature of Supervisor:

Sign of Student 1	**Sign of Student 2**
Rupesh Gaur	Chinu Rathi
20MCA1316	20MCA1351

ACKNOWLEDGEMENT

It is our pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced our thinking, behaviour and acts during the course of study.

We express our sincere gratitude to all for providing us an opportunity to undergo Project as the part of the curriculum.

We are thankful to "**Ms. Amandeep Kaur**" for her support, cooperation, and motivation provided to us during the training for constant inspiration, presence and blessings.

We also extend our sincere appreciation to **"Ms. Amandeep Kaur"** who provided her valuable suggestions and precious time in accomplishing our Integrated project report.

Lastly, we would like to thank the almighty and our parents for their moral support and friends with whom we shared our day-to-day experience and received lots of suggestions that improve our quality of work.

Table of contents

Abstract

Bookshelf is an inspiring, easy-to-use, intuitive website built to help enhance your reading, studying, and learning experience.

Bookshelf offers industry leading, easy-to-use books to help you learn more efficiently. With Bookshelf, you can read, highlight, and annotate just as you would on paper. Even subscribe to your want to read and read book highlights and notes to view in your book.

Bookshelf's features go beyond traditional print textbooks.

See, all of your E-Textbooks at a glance and access them instantly, anywhere at anytime from your Bookshelf - no backpack required.

You'll find multiple ways to move between pages and sections including linked Table of Contents and Search, make navigating E-Textbooks a snap

Introduction

Problem statement:

This project aims to developed for the purpose of:

Here we are try to develop such type of application which provides to manage any type of book from the bookshelf . That means a shelf which has the type of system which provides the facility to the customers of managing the books and to differentiate the book from the shelf without any complexity.

Purpose of Project and Overview of Project Report:

Our project is based on managing the books in 3 different categories.

In this application we have used many different functionality 1st you can manage your books 2nd you can search for books also you can delete the books from the given categories and you also have a log in page. It really helps you to save your time and to find the books and easy to remember on which book you are up to. According to the above facts, managing and maintaining a book shelf could also be controlled by efficient software. This project focuses attention on designing efficient and reliable software which controls the management of a book.

Technologies used:

The website has been developed in **CSS**, JavaScript, React js, API, NPM.

HTML

HTML is a mark-up language which is in reality the is a backbone of any site. We can't develop a site without the basic knowledge of HTML. Whereas only using HTML the website will not look attractive to the user. To add more effects and features to this website **CSS** and **ReactJS** is used.

CSS

CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once.External style sheets are stored in CSS files.

REACT JS

It is a JavaScript library for building user interfaces.

React makes it painless to create interactive UIs.

Design simple views for each state in your application,

and React will efficiently update and render just the

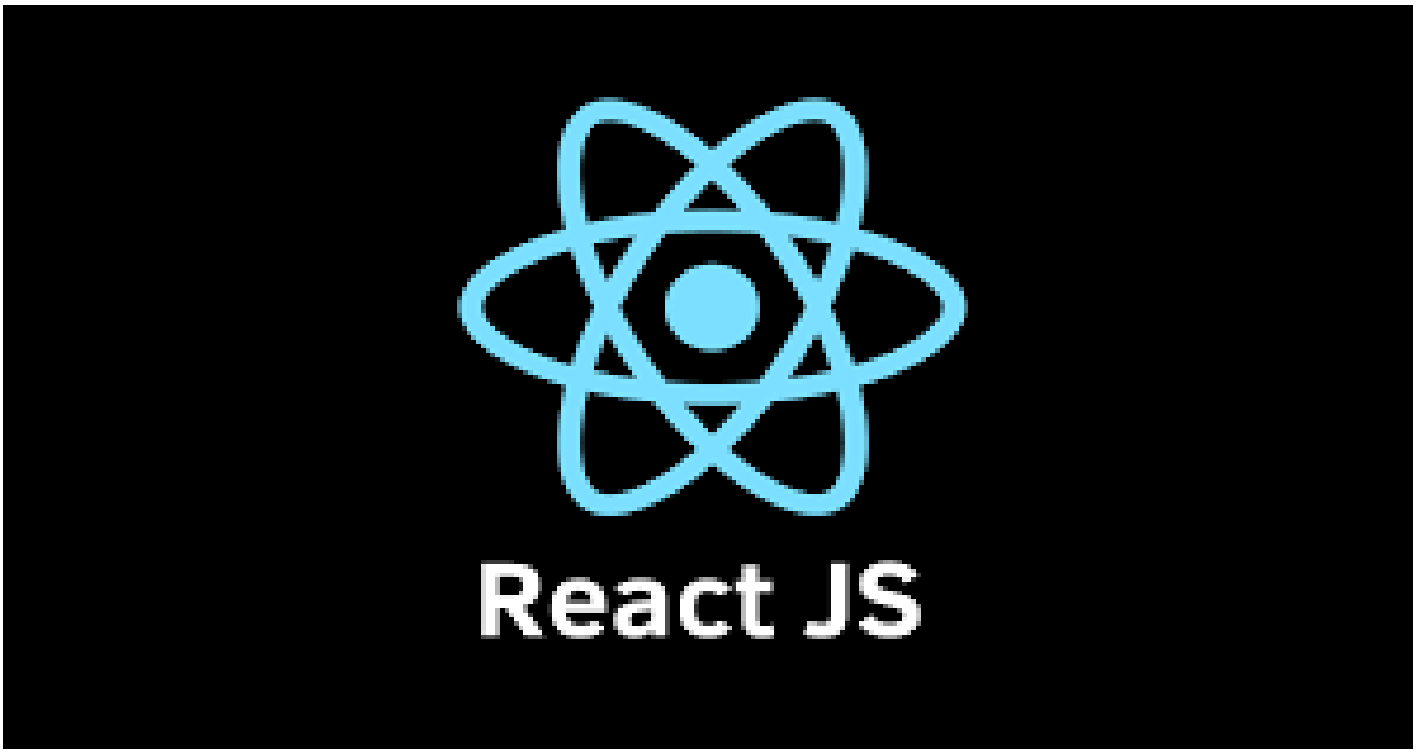
right components when your data changes. Declarative

views make your code more predictable and

easier to debug.It can also render on the

server using Node, and it can power native

apps using React Native.



API

An application programming interface (**API**) is a computing interface which defines interactions between multiple software intermediaries.

It defines the kinds of calls or requests

>can be made,

>how to make them,

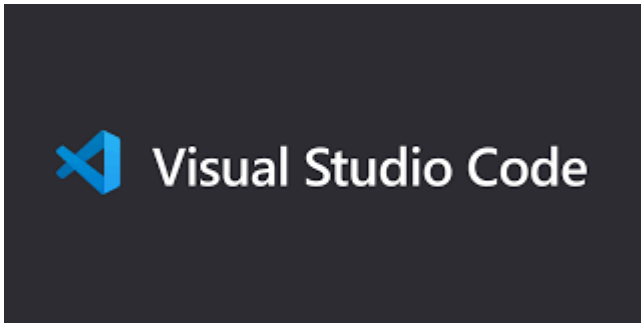
>the data formats that should be used,

>the conventions to follow.

VISUAL STUDIO CODE

Visual Studio Code is a free source-code editor made by Microsoft for **Windows**, **Linux** and **MacOS**.

Features include support for debugging, syntax highlighting,intelligent code completion, snippets, code refactoring, and embedded Git.



NPM

Npm (**Node Package Manager**) is package manager for the JavaScript programming language.

It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry. The registry is accessed via the client, and the available packages can be browsed and searched via the npm website. For running the application you have to open the terminal and run npm install(to install the dependencies) and then npm start(to start the application on localhost)

Advantage of Book Shelf App: -

- Easy to manage the books.
- Save Time.
- Place the book according to the situation.

Software and Hardware Requirement Specification

1. Methods

The way we reach to get this solution of the problem or attacked the problem contains sequence of steps: -

a. First, we discuss about this problem statement with our team members about the project that aims to develop an website that enables users to place the books according to situation. This application can be efficiently used as a medium. The functionalities provided by this applications include the search page to find the book easily and place the book in different three categories -:

   1) Want to read

   2)Currently read

   3)Read

b. After that we all are with some major points that are raised in this problem and thinkover it.

c. Many solutions of this problem are come out of our mind that are

   a. A website can also be made containing all the major aspects of the problem.

   b. A guide book also be made .

   c. An application can also be made.

   And so many options we have to get out of this problem.

2. Why we take website?

A website consists of browser- based HTML pages that are linked together and accessed over the internet whereas Apps are actual applications that are downloaded and installed on our mobile device. Users visit device-specific browsers such as google chrome in order to browse and use link for a given system. Rather than searching for an app and downloading it , customers can easily browse this website anywhere anytime.

1.2 Programming/Working Environment

Building a website comes down to major skills/languages: Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and React JS, it forms a triad of cornerstone technologies for the World Wide Web.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

Once you learn HTML, CSS is rulesets for telling the browser how to display the HTML formatted content. It is also not a programming language in the same way HTML is, although it can be alot more powerful.

Now for a total beginner, recommend youtube tutorials to get the basic idea of syntax behind javascript concepts,such as for loops and if statements.

Another language is React. React is a JavaScript library that aims to simplify development of visual interfaces.React is used to build single-page web applications, among with many other libraries and frameworks that were available before React came into life.

Learning to code is difficult enough on its own but not impossible. Once you know how to use these three languages, its pretty easy to make an effective website using these three languages. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry. The registry is accessed via the client, and the available packages can be browsed and searched via the npm website. For running the application you have to open the terminal and run npm install(to install the dependencies) and then npm start(to start the application on localhost)

4. Requirements to run this website

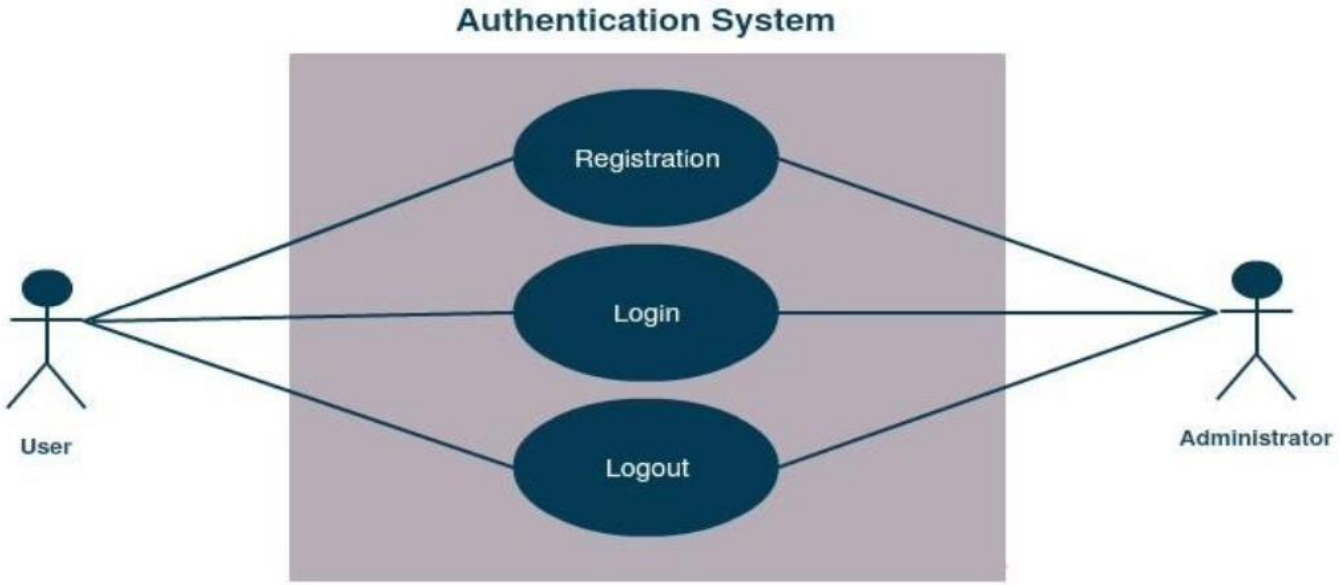
Each web browser renders HTML and CSS in a different way, so you'll also need to make sure that your device must contain a browser. The five most popular web browsers, in order from most to least popular, are:

- Google Chrome
- Windows Internet Explorer
- Mozilla Firefox
- Apple Safari
- Opera

You also need to have:

- Node.js

Authentication System



Overall Description

1. Product Perspective: -

- Easy to Search the book
- Manages the book
- Login
- Available on all devices
- Great user interface

2. Product Function: -

- Registration
- User signup/login
- Storage of book

3. Operating Environment: -

'BookShelf' is a website that will operate in the entire famous browser like Mozilla Firefox, Google Chrome, Internet explorer.

This software package is supposed to work in the following atmosphere:

1. OS - Windows 7, 8, XP, 10

2. Visual Studio

3. HTML, CSS, React JS.

4. Node.js, on the server.

4. Design and Implementation constraints: -

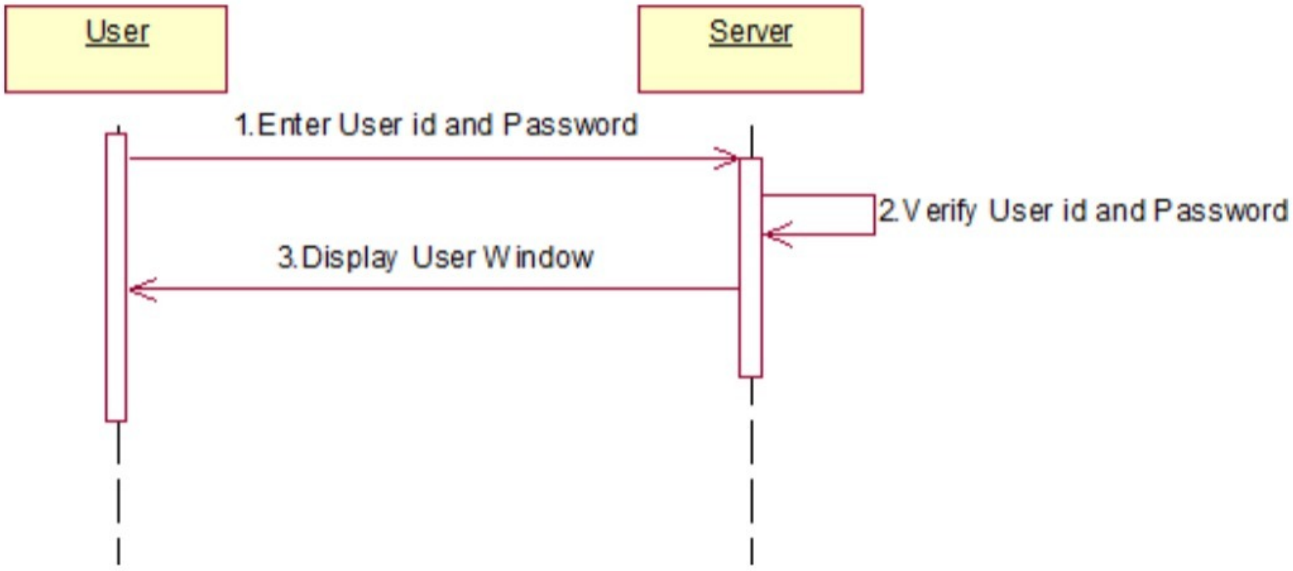
1. Login email should be valid means '@' is required and '.com' is required to be a valid email.
2. Password Length should be of at least 6 characters.

3. Email and password must not be the same

Functional Requirements

1. **User Registration**

   User must be able to register for the application through a valid username. If user skips this step, application should close. The users username will be the unique identifier of his/her account on Bookshelf Application.



2. **Adding New Books**

   The application should detect all books from the users database.

3. **Search Option**

   You can search any book which are available on website.

4. **Global**

   User should be able to enter the site from any country. You can easily access the websites.

External Interface Requirements

1. User Interface:

- Login Page
- Registration Page
- Main Page
- Search Page

2. Hardware Interface:

It must be pc computer or laptop to link with website.

3. Software Interface:

- Browser to load and view web pages.
- Operating System (Windows 8/7/XP)
- Npm should be installed.

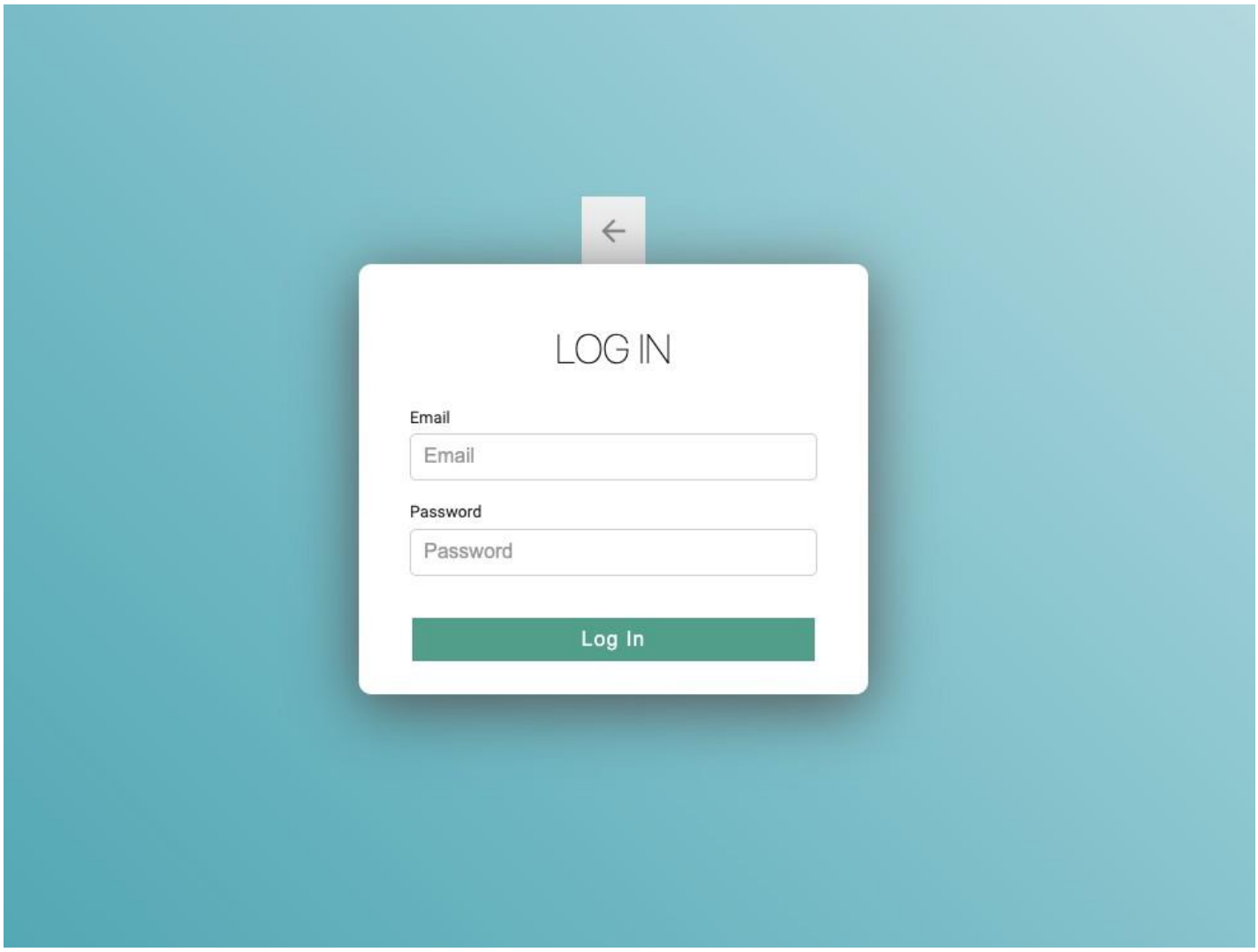
Program's Structure Analyzing and GUI Constructing (Project Snapshots)

- **Main Page**

  Open the Bookshelf Application and you will enter the on the following main page.

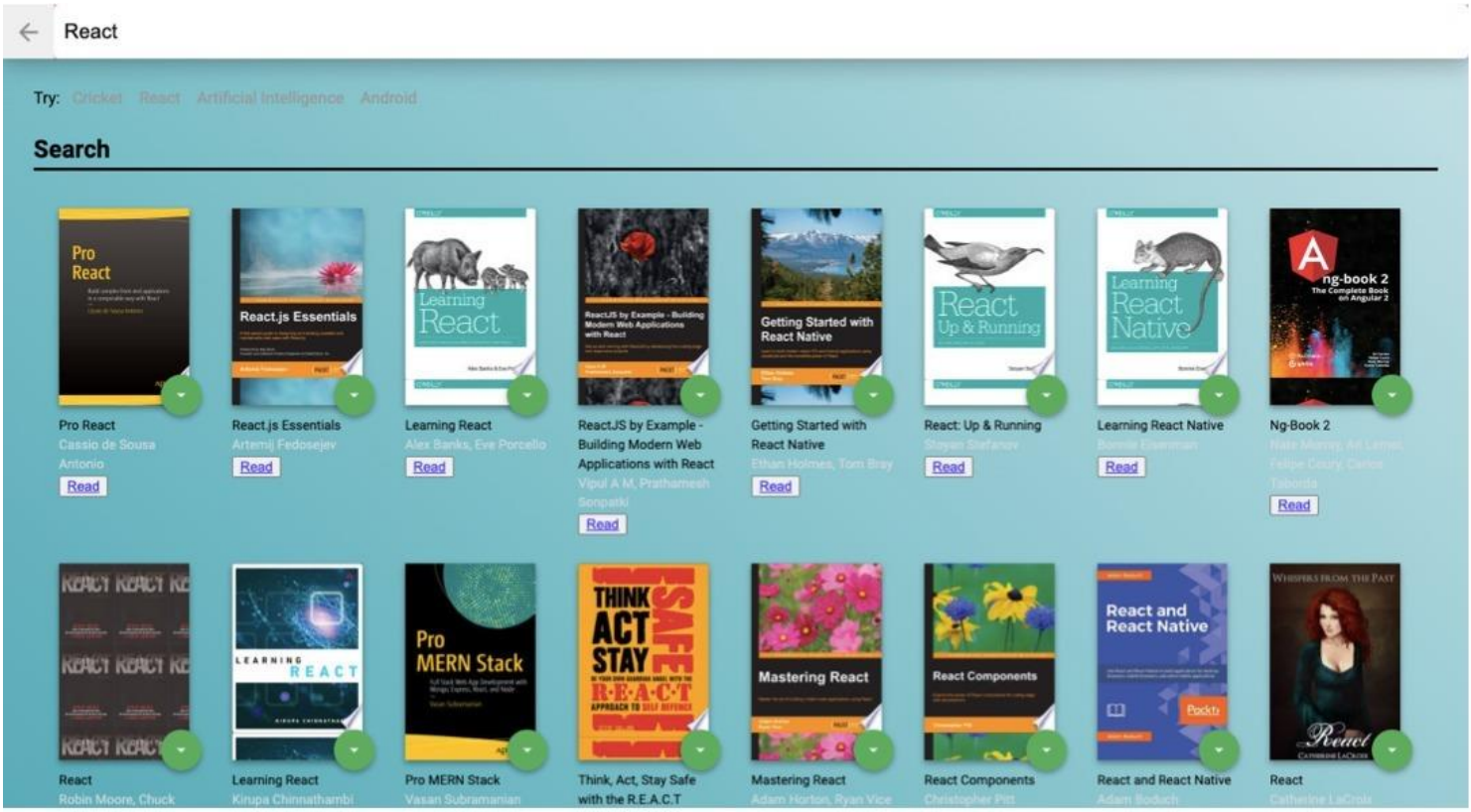
- **Login**

  You will have to login with a valid username and a password. The user interface of this page is really different as compare to other.

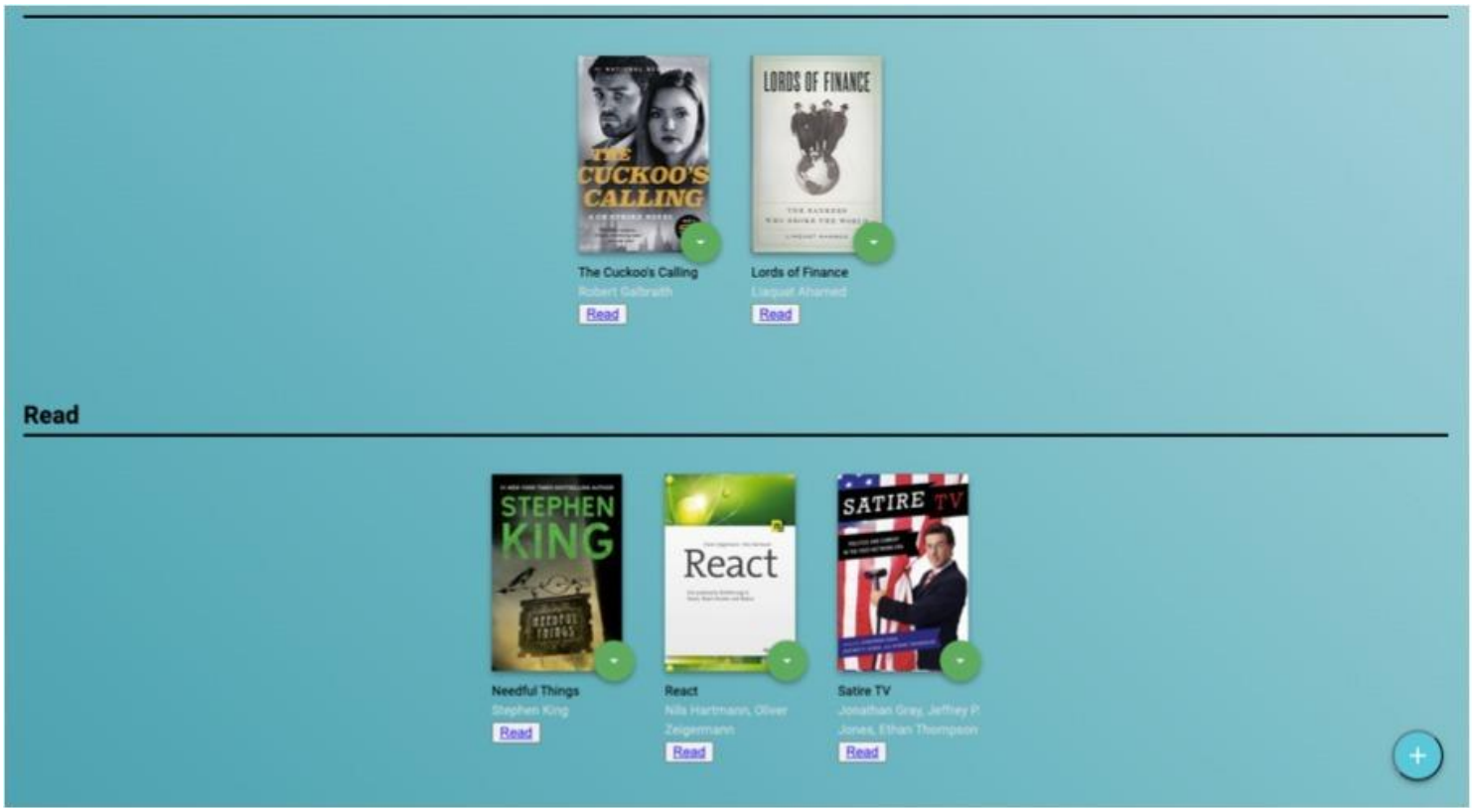


- **Search Option**

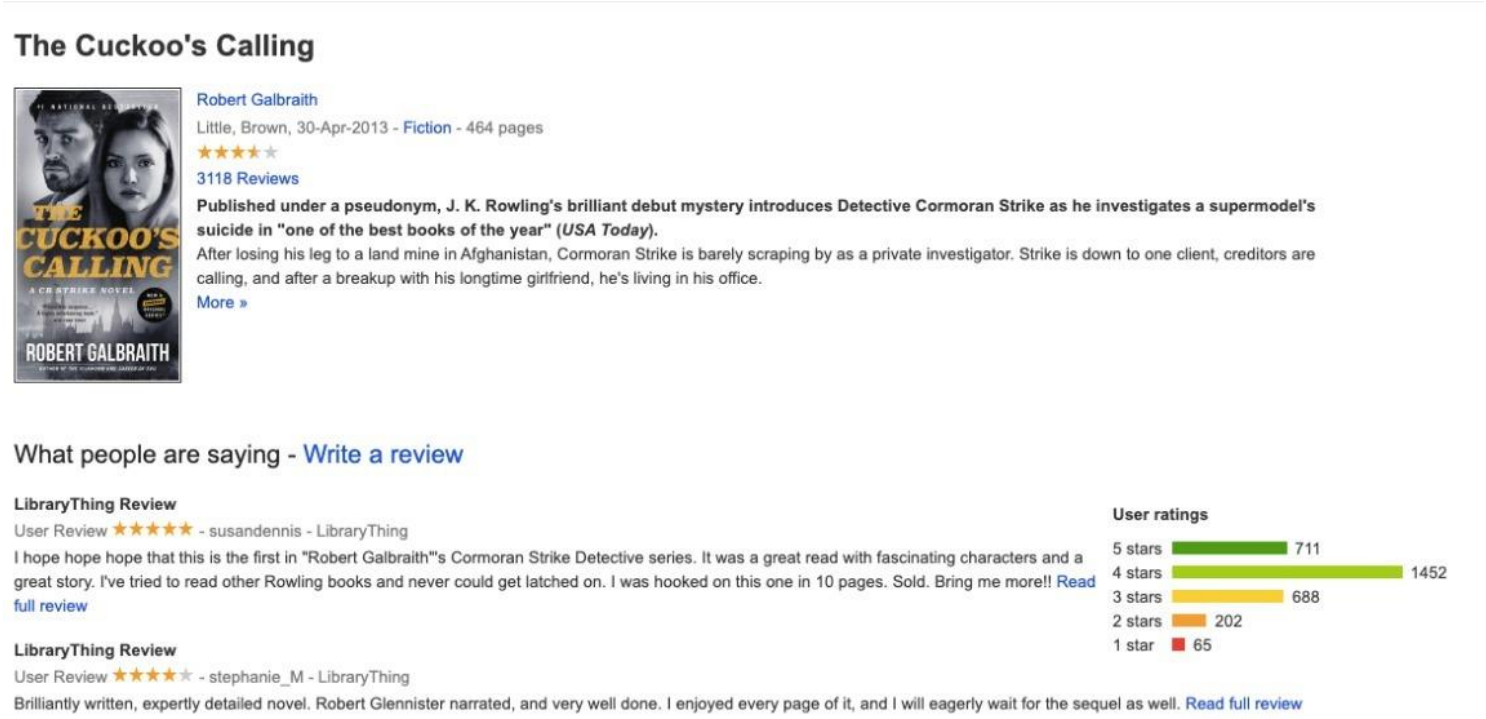
Here you can search any books which you want to read. All your favourite books are here.



- **Books Page**


THE CUCKOO'S CALLING
LORDS OF FINANCE
The Cuckoo's Calling
Robert Galbraith
Read
Lords of Finance
Liaquat Ahamed
Read
Read
STEPHEN KING
React
SATIRE TV
Needful Things
Stephen King
Read
React
Nils Hartmann, Oliver Zeigermann
Read
Satire TV
Jonathan Gray, Jeffrey P Jones, Ethan Thompson
Read

- **Reading books**



CODE SCREENSHOTS-

1) Apps.js

```
JS App.js
src > JS App.js > ...
1  import React from 'react'
2  import * as BooksAPI from './BooksAPI'
3  import './App.css'
4  import { Route } from 'react-router-dom';
5  import Login from './Login';
6  import Search from './Search';
7  import MainPage from './MainPage';
8
9  class BooksApp extends React.Component {
10   state = {
11     books: [],
12     loading: false
13   }
14
15   componentDidMount = () => {
16     this.fetchAllBooks();
17   }
18
19   fetchAllBooks = () => {
20     BooksAPI.getAll()
21       .then(books => this.setState({ books }))
22       .then(() => this.setState({ loading: false }))
23   }
24
25   // move book to a different shelf
26   onShelfChange = async (book, shelfName) => {
27     this.setState({ loading: true });
28     await BooksAPI.update(book, shelfName)
29
30     this.fetchAllBooks()
31   }
32
```

```
33   render() {
34     return (
35       <div className={`app ${this.state.loading && 'dimmed'}`}>
36         <Route path="/search">
37           <Search
38             userBooks={this.state.books}
39             onShelfChange={this.onShelfChange}
40           />
41         </Route>
42         <Route exact path="/">
43           <MainPage books={this.state.books} onShelfChange={this.onShelfChange} />
44         </Route>
45         <Route path="/login"><Login books={this.state.books} onShelfChange={this.onShelfChange}/></Route>
46
47       </div>
48     )
49   }
50 }
51
52 export default BooksApp;
53
```

2) BookItem.js

```
BookItem.js
src > BookItem.js > ...
import React from 'react';
import PropTypes from 'prop-types';


class BookItem extends React.Component {

    onChangeSelect = (newShelf) => {
        this.props.onShelfChange(this.props.book, newShelf);
    }

    render() {
        const { book } = this.props;
        const hasImage = book.imageLinks ? book.imageLinks.smallThumbnail : '';

        return (
            <li>
                <div className="book">
                    <div className="book-top">
                        <div className="book-cover" style={{ backgroundImage: `url(${hasImage})` }}></div>
                        <div className="book-shelf-changer">
                            <select value={book.shelf || 'none'} onChange={(e) => this.onChangeSelect(e.target.value)}>
                                <option value="move" disabled>Move to...</option>
                                <option value="currentlyReading">Currently Reading</option>
                                <option value="wantToRead">Want to Read</option>
                                <option value="read">Read</option>
                                <option value="none">None</option>
                            </select>
                        </div>
                    </div>
                    <div className="book-title">{book.title}</div>
                    <div className="book-authors">{book.authors && book.authors.join(', ')}</div>
                    <button className="button"><a href={book.previewLink}>Read</a></button>
                </div>
            </li>
        )
    }
}

BookItem.propTypes = {
    book: PropTypes.object.isRequired,
    onShelfChange: PropTypes.func.isRequired
}

export default BookItem;
```

3) BookList.js

```
BookList.js
src > BookList.js > ...
import React from 'react'
import PropTypes from 'prop-types';
import BookItem from './BookItem';

function BookList({ name, books, onShelfChange }) {
    return (
        <div className="bookshelf">
            <h2 className="bookshelf-title">{name}</h2>
            <div className="bookshelf-books">
                <ol className="books-grid">
                    {books.length === 0 ? <div>There are no books to display.</div>
                        : books.map(book => (
                            <BookItem book={book} onShelfChange={onShelfChange} key={book.id} />
                        ))}
                </ol>
            </div>
        </div>
    )
}

BookList.propTypes = {
    name: PropTypes.string.isRequired,
    books: PropTypes.arrayOf(PropTypes.object),
    onShelfChange: PropTypes.func.isRequired
}

export default BookList;
```

4) BookAPI.js

BooksAPI.js

src > BooksAPI.js > ...

```javascript

const api = "https://reactnd-books-api.udacity.com"


// Generate a unique token for storing your bookshelf data on the backend server.
let token = localStorage.token
if (!token)
  token = localStorage.token = Math.random().toString(36).substr(-8)

const headers = {
  'Accept': 'application/json',
  'Authorization': token
}

export const get = (bookId) =>
  fetch(`${api}/books/${bookId}`, { headers })
    .then(res => res.json())
    .then(data => data.book)

export const getAll = () =>
  fetch(`${api}/books`, { headers })
    .then(res => res.json())
    .then(data => data.books)

export const update = (book, shelf) =>
  fetch(`${api}/books/${book.id}`, {
    method: 'PUT',
    headers: {
      ...headers,
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({ shelf })
  }).then(res => res.json())

export const search = (query) =>
  fetch(`${api}/search`, {
    method: 'POST',
    headers: {
      ...headers,
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({ query })
  }).then(res => res.json())
    .then(data => data.books)

```

5) index.js

index.js

src > index.js

```javascript
import React from 'react'
import ReactDOM from 'react-dom'
import App from './App'
import './index.css'
import { BrowserRouter } from 'react-router-dom'

ReactDOM.render(<BrowserRouter><App /></BrowserRouter>, document.getElementById('root'))

```

6) Login.js

```
JS Login.js
src > JS Login.js > ...
import React, { Component } from "react";
import "./App.css";
import { Link, useHistory, Redirect } from "react-router-dom";



const emailRegex = RegExp(
  /^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/
);

const formValid = ({ formErrors, ...rest }) => {
  let valid = true;

  // validate form errors being empty
  Object.values(formErrors).forEach(val => {
    val.length > 0 && (valid = false);
  });

  // validate the form was filled out
  Object.values(rest).forEach(val => {
    val === null && (valid = false);
  });

  return valid;
};

class App extends Component {

  constructor(props) {
    super(props);

    this.state = {

      email: null,
      password: null,
      login: false,
      formErrors: {

        email: "",
        password: "",
        username: ""
      },
      renderHome : false
    };
  }
```

```
handleSubmit = e => {
  e.preventDefault();

  if (formValid(this.state)) {
    console.log(`
      --SUBMITTING--

      Email: ${this.state.email}
      Password: ${this.state.password}
    `
    );
    this.setState({ login: true });
    sessionStorage.setItem("login", "true");
    sessionStorage.setItem("email", this.state.email)
    this.setState({
      renderHome: true
    });

    // let history = useHistory();
    // history.goBack();
    //return this.setState({ error: "" });
  } else {
    console.error("FORM INVALID - DISPLAY ERROR MESSAGE");
  }

};

routeChange=()=> {
  let history = useHistory();
  console.log('sdsdsdsdss');
  history.goBack();
}

handleChange = e => {
  e.preventDefault();
  const { name, value } = e.target;
  let formErrors = { ...this.state.formErrors };
```

```
  switch (name) {
    case "firstName":
      formErrors.firstName =
        value.length < 3 ? "minimum 3 characaters required" : "";
      break;
    case "lastName":
      formErrors.lastName =
        value.length < 3 ? "minimum 3 characaters required" : "";
      break;
    case "email":
      formErrors.email = emailRegex.test(value)
        ? ""
        : "invalid email address";
      break;
    case "password":
      formErrors.password =
        value.length < 6 ? "minimum 6 characaters required" : "";
      break;
    default:
      break;
  }

  this.setState({ formErrors, [name]: value }, () => console.log(this.state));
};
```

```
  render() {
    const { formErrors } = this.state;

    const renderToHomePage = this.state.renderHome;
    if (renderToHomePage === true) {
      return <Redirect to="/" />
    }

    return (

      <div className="wrapper">
        <Link to="/"><button className="close-search">Close</button></Link>
        <div className="form-wrapper">
          <h1>LOG IN</h1>
          <form onSubmit={this.handleSubmit} noValidate>
            <div className="email">
              <label htmlFor="email">Email</label>
              <input
                className={formErrors.email.length > 0 ? "error" : null}
                placeholder="Email"
                type="email"
                name="email"
                noValidate
                onChange={this.handleChange}
              />
              {formErrors.email.length > 0 && (
                <span className="errorMessage">{formErrors.email}</span>
              )}
            </div>
            <div className="password">
              <label htmlFor="password">Password</label>
              <input
                className={formErrors.password.length > 0 ? "error" : null}
                placeholder="Password"
                type="password"
                name="password"
                noValidate
                onChange={this.handleChange}
              />
              {formErrors.password.length > 0 && (
                <span className="errorMessage">{formErrors.password}</span>
              )}
            </div>
```

```
            <div className="createAccount">
              <button type="submit">Log In</button>
            </div>

          </form>
        </div>
      </div>
    );
  }
}

export default App;
```

7) Loginbtn.js

```
JS Loginbtn.js ×
src > JS Loginbtn.js > ...
1  import React from 'react';
2  import { Link } from 'react-router-dom';
3
4  function LoginBtn() {
5      return (
6          <div className="loginbtn">
7              <Link to="/login">
8                  <button className="btn">Login</button>
9              </Link>
10         </div>
11     )
12
13 }
14
15 export default LoginBtn;
```

8) MainPage.js

MainPage.js

src > MainPage.js > ...

```javascript
import React from 'react';
import PropTypes from 'prop-types';
import OpenSearchBtn from './OpenSearchBtn';
import BookList from './BookList';
import LoginBtn from './Loginbtn';



class MainPage extends React.Component {
    // find all books by their shelf
    filterBooks = (shelfName) => {
        return this.props.books.filter(book => book.shelf === shelfName)
    }
    changeLoginStatus = () => {
        sessionStorage.setItem("login", "false");
        this.setState({
          username: "",
          password: "",
        });
      };

    render() {
        return (
            <div className="list-books">
                <div className="list-books-title">
                <div className="foo">
                    <span className="letter" data-letter="B">B</span>
                    <span className="letter" data-letter="O">O</span>
                    <span className="letter" data-letter="O">O</span>
                    <span className="letter" data-letter="K">K</span>
                    <span className="letter" data-letter="S">S</span>
                    <span className="letter" data-letter="H">H</span>
                    <span className="letter" data-letter="E">E</span>
                    <span className="letter" data-letter="L">L</span>
                    <span className="letter" data-letter="F">F</span>
                </div>
                </div>
                <div className="login">{sessionStorage.getItem("login")=="true" ? (
            <div> <div>{sessionStorage.getItem("email")}</div>
            <button className="btn" onClick={this.changeLoginStatus}>Sign Out</button>
            </div>

        ) : (
                <div className="login"><LoginBtn /></div>
        )}</div>
```

```
                <div>
                    <div className="list-books-content">
                        <div>
                            <BookList
                                name="Currently Reading"
                                books={this.filterBooks('currentlyReading')}
                                onShelfChange={this.props.onShelfChange}
                            />
                            <BookList name="Want to Read"
                                books={this.filterBooks('wantToRead')}
                                onShelfChange={this.props.onShelfChange} />
                            <BookList
                                name="Read"
                                books={this.filterBooks('read')}
                                onShelfChange={this.props.onShelfChange}
                            />
                        </div>
                    </div>
                    <OpenSearchBtn />

                </div>
            </div>
        )
    }
}

MainPage.propTypes = {
    books: PropTypes.arrayOf(PropTypes.object),
    onShelfChange: PropTypes.func.isRequired
}

export default MainPage;
```

9) OpenSearchBtn.js

JS OpenSearchBtn.js ×

src > JS OpenSearchBtn.js > ...

```
import React from 'react';
import { Link } from 'react-router-dom';

function OpenSearchBtn() {
    return (
        <div className="open-search">
            <Link to="/search">
                <button>Add a book</button>
            </Link>
        </div>
    )

}

export default OpenSearchBtn;
```

```
        )
    }
}

Search.propTypes = {
    userBooks: PropTypes.arrayOf(PropTypes.object),
    onShelfChange: PropTypes.func.isRequired
}

export default Search;
```

10) SearchBar.js

SearchBar.js

src > SearchBar.js > ...

```
import React from 'react';
import PropTypes from 'prop-types';
import { Link } from 'react-router-dom';

function SearchBar({ query, handleChange }) {
    return (
        <div className="search-books-bar">
            <Link to="/">
                <button className="close-search">Close</button>
            </Link>
            <div className='search-books-input-wrapper'>
                {/*
                    NOTES: The search from BooksAPI is limited to a particular set of search terms.
                        */}
                <input
                    type="text"
                    placeholder="Search by title or author"
                    value={query}
                    onChange={(e) => handleChange(e.target.value)}
                />
            </div>
        </div>
    )
}

SearchBar.propTypes = {
    query: PropTypes.string.isRequired,
    handleChange: PropTypes.func.isRequired
}

export default SearchBar;
```

Objective of project

The main objective of the project is to allow to each and every person to get connected to new books. Some other objectives are listed below:

- Providing a private platform to users.
- To allow each and every person to read their favourite book.
- Search their favourite book.
- Manage their books.

Conclusion

Bookshelf is basically a website to manage books. Users can read their books to enhance their knowledge and can learn while sitting at home. This website is developed by using React , JavaScript, HTML, CSS.

The front-end of this website is developed by using HTML, CSS, React. We have used Session storage to store the information of user at backend. This is an easy to deploy website.
This has an interactive User Interface which will attract a greater number of clients. This website can be used by students who can't afford expensive tuitions. We can deploy it by following some basic commands and also, it's easy to use for clients too. Hence, we can deploy it on internet too.

Future Scope

There is always a scope for improvements in any apps. Right now, we are just dealing with book management. There are several android apps which serve similar purpose as this project, but these apps were rather difficult to use and provide confusing interfaces. A positive first impression is essential in human relationship as well as in human computer interaction. This project hopes to develop a books web app with high quality user interface. In future we may be extended to include features such as:

1. Adding books
2. Issue books
3. Write books
4. Publish books

Bibliography/References

1. https://www.udemy.com
2. https://reactjs.org/docs/getting-started.html
3. www.youtube.com
4. www.wikipedia.com
5. http://www.geeksforgeeks.org
6. https://www.w3schools.com/