ICT ACADEMY OF KERALA

Summer Internship Report

Full Stack Application Development with ReactJS

Name of Team Member

Arya Pradeep,AmayaRaj k.V,Fasila Kasim,Leona Sibu,Nimisha N.S,Sabiya Abraham

GPTC PERUMBAVOOR

08.05.2023

EXECUTIVE SUMMARY

The Bookstore Library App is a comprehensive web-based application that offers users the ability to browse, rent, and manage books from a bookstore library. It includes features like user registration and login, book listings with availability status, rental requests, profile management, and an admin dashboard for user and book administration.

Users can view, like, comment, and rent books from the library. The app also includes an admin module with separate login details, enabling the admin to control user access, manage book listings, and track the rental status of books. The app features a signup page for users, a home page displaying random book details, and a book list page for browsing and renting books. The admin has a dedicated page to view and edit user accounts, an admin dashboard to manage books and users, and the ability to add new books. The app includes essential features such as book availability tracking and rental expiry notifications.

INTRODUCTION

The Bookstore Library App is a comprehensive web-based application designed to streamline the process of book browsing, renting, and user management for a bookstore library. With a user-friendly interface and robust functionality, the app aims to provide an efficient and enjoyable experience for both users and administrators.

The app offers a range of features for users, including a secure login module that allows them to create an account or log in using their credentials. Once logged in, users have access to various functionalities, such as browsing and viewing book lists, liking books of interest, and submitting rental requests. They can also manage their personal profiles, update their information.

On the other hand, the app provides an administrative module specifically tailored for the bookstore library's administrators. The admin module includes features like user management, where administrators can control user access, delete or block user accounts, and view and edit user profiles. Additionally, the admin has complete control over book management, enabling them to list and manage books, update rental availability status, and add new books to the inventory.

To enhance the user experience and improve efficiency, the app incorporates additional features. It tracks the availability status of each book, indicating whether it is available for rental or already checked out. This helps users make informed decisions while browsing the book list.

With a user-centric design and comprehensive functionality, the Bookstore Library App aims to create a seamless and convenient platform for users to explore and rent books while providing efficient user and inventory management tools for administrators.

OBJECTIVES

The objective of studying MERN stack development is to learn how to use JavaScript to build full-stack web applications. The MERN stack is a popular choice for web development because it is a JavaScript-based stack that consists of four technologies:

- MongoDB: A NoSQL database that is document-oriented and schema-free.
- Express: A web application framework that is built on top of Node.js.
- React: A JavaScript library that is used to create user interfaces.
- Node.js: A runtime environment that allows JavaScript to be run on the server-side.

SCOPE AND DELIVERABLES

The scope of the MERN Stack Bookstore Library App Project is to create a web application that allows users to browse, search, and borrow books. The application will be built using the MERN stack (MongoDB, Express, React, and Node.js).

The following are the key deliverables of the project:

- A user-friendly interface that allows users to browse and search for books.
- The ability for users to borrow books.
- A notification system that alerts users when their books are due.
- A reporting system that tracks book usage.

The project will be delivered on time and within budget. The application will be of high quality and will meet the needs of the users.

METHODOLOGY

1.Understanding the MERN Stack:

- Gain a deep understanding of **MongoDB**, **Express.js, React**, and **Node.js**.
- Learn how to integrate these technologies to build full-stack web applications.

2.Front-End Development :

- Used **MaterialUI** for framework and **react**(JavaScript library for building user interfaces).
- Uses **react-router-dom** for routing purpose.
- Used **css** for styling.
- **JavaScript** for client-side validation.
- And **axios** for making http request from web browsers and Node.js.

3.Back-End Development:

- Build server-side applications using **Node.js** and **Express.js**.
- Handle HTTP requests, routing, **middleware**, and interact with databases.
- For login purpose we used **jsonwebtoken**.
- Used **bcrypt** password for password security.
- Uses **mongoose** for database connectivity.
- And **cors** for allowing web browsers to make cross-origin HTTP requests securely.
- For backend code testing we used **postman** application and **web browsers**.

4.Database:

- Learn the basics of **MongoDB** and integrate it with Node.js.

- Understand data modeling, CRUD operations, indexing, and querying techniques.

5.Full-Stack Development:

- Integrate front-end and back-end components to build full-stack web applications.
- Implement RESTful APIs, data flow management, and authentication/authorization mechanisms.

PROJECT ACTIVITIES

During my internship, we undertook various activities related to the development of the Bookstore Library App using the MERN stack. Here is a detailed breakdown of the tasks we performed, the tools and technologies we used, and the challenges we encountered:

Requirement Gathering and Analysis:

- Conducted meetings with stakeholders to gather project requirements.
- Analyzed the scope and objectives of the app to determine the key functionalities and features.

Technology Selection:

- Researched and evaluated different technologies for building the app.
- Selected the MERN stack (MongoDB, Express.js, React, and Node.js) as the primary technology stack.

Design and Planning:

- Created wireframes and mockups to visualize the app's user interface.
- Designed the database schema for storing book details, user information, and rental requests.

Front-end Development:

- Developed the user interface using React.js.
- Implemented components for the user login/signup pages, book listing page, user profile page, admin login, book details and user details for admin,book adding pages, etc..
- Utilized MaterialUI, CSS, and JavaScript to create responsive and visually appealing UI elements.

- Used React Router for client-side routing and navigation.
- Also used axios request for making http request from web browsers and NodeJS

Back-end Development:

- Built the server-side using Node.js and Express.js.
- Implemented RESTful APIs to handle user authentication, book management,and admin authentication.
- Integrated MongoDB as the database to store and retrieve data.
- Used Mongoose, an Object Data Modeling (ODM) library, for interacting with MongoDB.

Database Integration:

- Created database models and schemas for books, users and admin..
- Implemented CRUD operations (Create, Read, Update, Delete) for managing data.
- Handled database queries and data validation using Mongoose.

User Authentication and Authorization:

- Implemented user registration and login functionality.
- Used bcrypt for password hashing and validation.
- Implemented JSON Web Tokens (JWT) for secure user authentication and authorization.

Admin Authentication:

- Implemented admin login functionality.
- Implemented JSON Web Tokens (JWT) for secure admin authentication.

Testing and Debugging:

- Conducted unit tests and integration tests to ensure the functionality and reliability of the app.
- Debugged and fixed issues identified during testing phases using postman and web browsers.

Continuous Integration:

- Set up continuous integration using tools like Git, GitHub.

Challenges and Obstacles:

- Learning Curve: As an intern, we faced a learning curve in understanding and implementing the MERN stack technologies. It required dedicated self-study, group discussion and hands-on practice to grasp the concepts and best practices.
- Integration Complexity: Integrating different components of the MERN stack, such as connecting React with Node.js and Express.js, and establishing communication with MongoDB, presented challenges in terms of configuration and debugging.
- Authentication and Security: Implementing secure user authentication and authorization mechanisms, including JWT and password hashing, required careful consideration of security vulnerabilities and best practices.
- Performance Optimization: Optimizing the app's performance, especially when dealing with large datasets and complex queries, required implementing indexing, pagination, and caching techniques.

Overall, through these activities, we gained practical experience in MERN stack development, problem-solving, and working with modern web development tools and technologies.

RESULTS & FINDINGS

Results:

- Fully Functional Bookstore Library App: The main outcome of the internship project was the successful development of a fully functional Bookstore Library App using the MERN stack. The app incorporated the specified features, such as user and admin authentication, book listing, rental status, and admin management.
- User Authentication and Registration: Users were able to create accounts, log in securely, and manage their personal profiles. The authentication system ensured the security of user data and protected sensitive information.
- Book Listing and Availability: The app displayed a collection of books with relevant details, including title, author, genre, and availability status. Users could browse the book list, and check the availability of books.
- Admin Dashboard and Book Management: The admin had access to an admin dashboard where

they could manage user accounts, list and manage books, update rental status, and track the availability of books.

Findings:

1. Seamless Integration of MERN Stack: The project demonstrated the successful integration of the MERN stack components (MongoDB, Express.js, React, and Node.js) to build a full-stack web application. The technologies worked together harmoniously to provide a robust and efficient system.

2. Effective User Experience: The use of React.js for the front-end development resulted in a smooth and interactive user interface. The component-based architecture of React allowed for the development of reusable UI elements, enhancing the user experience.

3. Efficient Data Management: The integration of MongoDB with Node.js and Express.js provided a flexible and scalable database solution. The NoSQL nature of MongoDB allowed for efficient storage and retrieval of book details, user information, and rental requests.

4. Secure User Authentication: The implementation of user authentication and authorization using JWT ensured secure access to the app and protected user data. Password hashing with bcrypt enhanced the security of user passwords.

5. Continuous Integration: The use of continuous integration ensured a streamlined development.

Overall, the internship project achieved its objectives of developing a Bookstore Library App using the MERN stack. It provided valuable insights into full-stack development, integration of different technologies, and implementing key features of a web application. The results highlighted the effectiveness and efficiency of the MERN stack for building robust and scalable applications.

CONCLUSION

In conclusion, the internship project to develop the Bookstore Library & Stock Keeping App was a success. The project aimed to create a functional web application using the MERN stack that allows users to login, register, view and rent books, and manage their profiles. Additionally, an admin interface was implemented to manage users, books, and rental statuses.

Throughout the project, all the defined requirements were met, and the application was successfully developed with the desired functionalities. Users were able to easily navigate through the app, view book lists, check availability, and send rental requests. The admin had full control over user management, book management, and rental status updates.

The project provided valuable insights and learnings in several areas. It enhanced knowledge and proficiency in the MERN stack, including ReactJS, Node.js, Express.js, and MongoDB. It also reinforced understanding of user authentication, CRUD operations, and data management. The project allowed for practical experience in implementing key features and functionalities required in a real-world application.

The outcomes of the project demonstrated the ability to design and develop a fully functional Bookstore Library & Stock Keeping App. The application met the objectives of providing an intuitive user interface, seamless navigation, and effective user and admin management. The successful completion of the project validated the skills and knowledge acquired during the internship.

Overall, the internship project was a valuable experience that provided practical exposure to MERN stack development and web application design. It enhanced problem-solving skills, teamwork, and project management capabilities. The project's outcomes and accomplishments serve as a testament to the ability to deliver high-quality software solutions and readiness for future development projects.

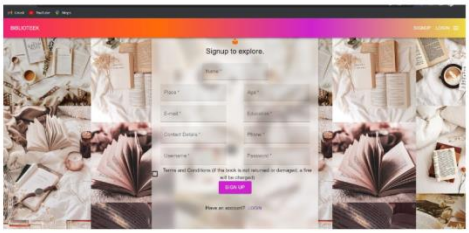
APPENDIX

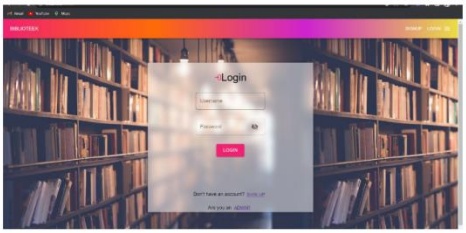
```
//importing
const express = require('express');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const cors = require('cors');
const mongoose = require('mongoose');
const userModel = require('./userModel');
const bookModel = require('./bookModel');
const adminModel = require('./adminModel');

// App Initialization
const JWT_SECRET = "AryaPradeep212";
const port = 9453;
const app = express();

// Middleware
app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.use(cors());

//MongoDB Connection
mongoose.connect(
'mongodb+srv://aryapradeep212:Arya@cluster0.zbd2ev1.mongodb.net/?retryWrites=true&w=majority'
, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
})
  .then(() => {
    console.log('Connected to MongoDB');
  })
  .catch((error) => {
    console.error('Error connecting to MongoDB', error);
  });

  // user registration
app.post('/register', async (req, res) => {
  const { firstName, username, place, age, email,
education, contactDetails, phone, password } = req.body;
  const hashedPassword = await bcrypt.hash(password, 10
);
  try {
    const oldUser = await userModel.findOne({ username
 });
    if (oldUser) {
      return res.send({ error: "User Exists" });
    }
    await userModel.create({
      firstName,
      username,
      place,
      age,
      email,
      education,
      contactDetails,
      phone,
      password: hashedPassword,
    });
    return res.send({ status: 'ok' });
  } catch (error) {
    console.log(error);
    res.send({ status: 'error' });
  }
});

//book view
app.get('/viewbook', async (req, res) => {
  try {
    const data = await bookModel.find();
    res.json(data);
  } catch (error) {
    console.log(error);
    res.send({ status: 'error' });
  }
});

//user view
app.get('/viewuser', async (req, res) => {
  try {
    const data = await userModel.find();
    res.json(data);
  } catch (error) {
    console.log(error);
    res.send({ status: 'error' });
  }
});
```

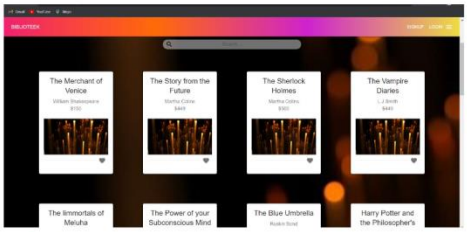
```
//book details fetch
app.get("/singlebook/:id", async (req, res) => {
  try {
    const { id } = req.params;
    const data = await bookModel.findOne({ _id: id });
    res.send(data);
  } catch (error) {
    console.error(error);
    res.send("Unable to fetch data");
  }
});

//user details fetch
app.get("/singleuser/:id", async (req, res) => {
  try {
    const { id } = req.params;
    const data = await userModel.findOne({ _id: id });
    res.send(data);
  } catch (error) {
    console.error(error);
    res.send("Unable to fetch data");
  }
});

//book details post
app.post('/createbook', async (req, res) => {
  console.log(req.body)
  try {
    console.log(req.body);
    const result = new bookModel(req.body);
    await result.save();
    res.send("Data Added");
  } catch (error) {
    console.error(error);
    res.status(500).send(
"An error occurred while saving the data.");
  }
});

//admin add
app.post('/createadmin', async (req, res) => {
  try {
    console.log(req.body);
    const result = new adminModel(req.body);
    await result.save();
    res.send("Data Added");
  } catch (error) {
    console.error(error);
    res.status(500).send(
"An error occurred while saving the data.");
  }
});

//delete book
app.delete('/delete/:id',async(req,res)=>{
  var id=req.params.id
  await bookModel.findByIdAndDelete(id)
  res.send("Deletnd")
})

//delete user
app.delete('/deleteuser/:id',async(req,res)=>{
  var id=req.params.id
  await userModel.findByIdAndDelete(id)
  res.send("Deletnd")
})

//update book
app.put('/update/:id',async(req,res)=>{
  let id=req.params.id
  var result=await bookModel.findByIdAndUpdate(id,req.
body)
  res.send("Updated")
})

//update book
app.put('/update/:id',async(req,res)=>{
  let id=req.params.id
  var result=await userModel.findByIdAndUpdate(id,req.
body)
  res.send("Updated")
})

app.put('/blockuser/:id', async (req, res) => {
  var id = req.params.id;
// Fix: Corrected variable assignment
  var user = await userModel.findById(id);
  if (!user) {
    return res.status(404).send("User not found");
  }
  user.blocked = true;
  await user.save();
  res.send("Blocked");
});

//Rent status
app.post("/rent", async (req, res) => {
  const { bookId} = req.body;
  await bookModel.findOneAndUpdate({ _id:bookId}, {
status:"Not Available"});
  res.send("Borrowed");
})

// Port Checking
app.listen(port, () => {
  console.log('App listening on port 9453');
});
```

```
//book details fetch
app.get("/singlebook/:id", async (req, res) => {
  try {
    const { id } = req.params;
    const data = await bookModel.findOne({ _id: id });
    res.send(data);
  } catch (error) {
    console.error(error);
    res.send("Unable to fetch data");
  }
});

// user login
app.post("/login", async (req, res) => {
  const { username, password } = req.body;
  try {
    const user = await userModel.findOne({ username });
    if (!user) {
      return res.send({ error: "User not found" });
    }
    if(user.blocked){
      res.send({status:"User Blocked"})
      return;
    }
    if (await bcrypt.compare(password, user.password)) {
      const token = jwt.sign({id: user._id}, JWT_SECRET
,{
        expiresIn:"1W"
      });
      if (res.status(201)) {
        return res.json({ status: "ok", data: token,
user: user });
      } else {
        return res.json({ error: "error" });
      }
    }
    res.json({ status: "error", error:
"Invalid Password" });
  } catch (error) {
    console.log(error);
    res.send({ status: 'error' });
  }
});


//admin login
app.post("/adminlogin", async (req, res) => {
  const { username, password } = req.body;
  try {
    const user = await adminModel.findOne({ username });
    if (!user) {
      return res.status(404).json({ error:
"User not found" });
    }
    if (password !== user.password) {
      return res.status(401).json({ error:
"Invalid Password" });
    }
    // Generate and send a JWT token
    const token = jwt.sign({ id: user._id }, JWT_SECRET
, {
      expiresIn: "1W"
    });
    res.status(200).json({ status: "ok", data: token });
  } catch (error) {
    console.log(error);
    res.status(500).json({ error:
"Internal Server Error" });
  }
});

//routing
app.post("/homepage",async(req,res)=>{
  const{token}=req.body
  try{
    const user=jwt.verify(token,JWT_SECRET,(err,res)=>{
      console.log(err),"error";
      console.log(res,"result");
    })
    const Username=user.username
    userModel.findOne({username:Username})
    .then((data)=>{
      res.send({status:"ok",data:data})
    })
    .catch((error)=>{
      res.send({status:"error",data:error})
    })
  }catch(error){
  }
})
```

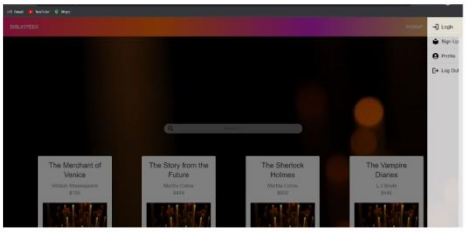

'Nothing Is More Expensive Than A Library'
-Walter Savage Landor

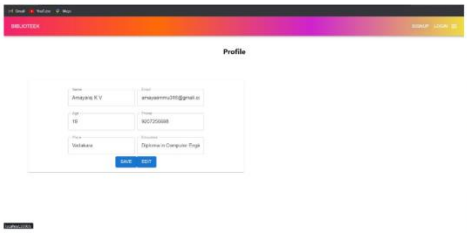

Signup to explore.

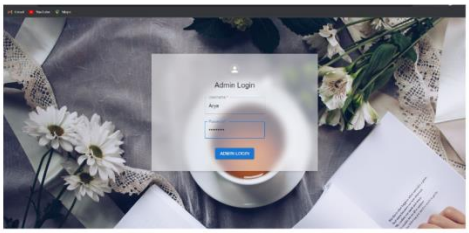

Login

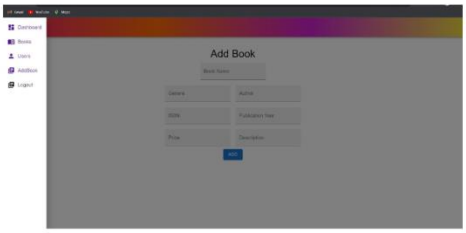

The Merchant of Venice
The Story from the Future
The Sherlock Holmes
The Vampire Diaries
The Immortals of Meluha
The Power of your Subconscious Mind
The Blue Umbrella
Harry Potter and the Philosopher's

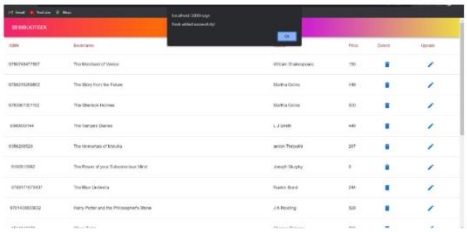

Title: The Vampire Diaries
Author: L J Smith

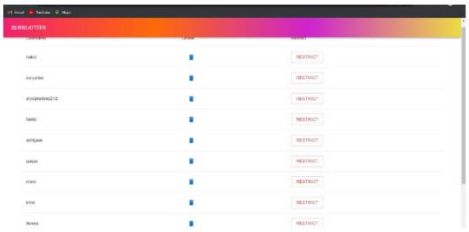

The Merchant of Venice
The Story from the Future
The Sherlock Holmes
The Vampire Diaries

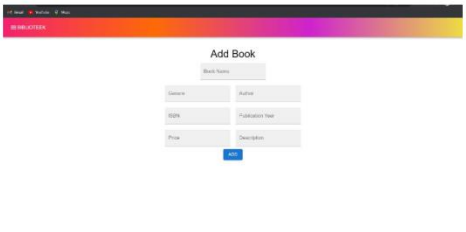

Profile


Admin Login


Add Book






Add Book