Book It – Acropolis Hall Booking System

-By Siddharth Jain IIMCA - 5

Overview

I am pleased to present to you a report on the deployment of our Booking Web Application, developed using the MERN Stack (MongoDB, Express.js, React.js, and Node.js). This report provides essential information regarding the technology used, server storage requirements, and database details to assist you in making informed decisions for the deployment process.

Technology Used

The Booking Web Application has been built using the MERN Stack, which is a popular and powerful combination of technologies for developing robust and scalable web applications. Here's a breakdown of each component:

- **MongoDB:** As the database system, MongoDB is a NoSQL database that provides a flexible and scalable solution for managing data in our application. It stores structured data in a JSON-like format, making it suitable for handling various types of booking-related information.
- **Express.js:** Express.js is a minimal and flexible web application framework for Node.js. It provides a lightweight and efficient structure for developing the server-side of our web application, handling HTTP requests and routing.
- **React.js:** React.js is a JavaScript library for building user interfaces. It allows for the creation of interactive and dynamic components, resulting in a smooth and responsive user experience. React.js is particularly well-suited for single-page applications, such as our booking app.

- **Node.js:** Node.js is a server-side JavaScript runtime environment that allows us to execute JavaScript code outside of a web browser. It provides the necessary tools and libraries to build scalable and high-performance web applications.

Server Storage Requirements

- **Database Storage:** MongoDB stores data in a flexible and schema-less format. The storage requirements will depend on the number of bookings, users, and associated data, such as customer information, booking details, and any additional custom fields. It is recommended to allocate sufficient storage to accommodate growth and ensure optimal performance.
- **Application Code and Assets:** The application's code, including JavaScript files, CSS stylesheets, and other assets, will require storage space on the server. The total size of the codebase, including the node_modules directory, is approximately 1 GB.

Process for Deployment

1. Install Node.js and npm:

- Install Node.js on the server, preferably using a version manager like nvm (Node Version Manager) to manage different Node.js versions.
- Verify the installation by running node -v and npm -v commands, which should display the installed versions.

2. Clone the Repository:

- Clone the Git repository of the Booking Web Application onto the server.
- Use the git clone command followed by the repository URL to create a local copy of the application code.

3. Install Client Dependencies:

- Navigate to the `/client` directory using `cd /client`.
- Run `npm install` to install all the project dependencies specified in the `package.json` file.
- This will download and install the required libraries and packages for the application to run.

4. Install Client Dependencies:

- Navigate to the `/server` directory using `cd /server`.
- Run `npm install` to install all the project dependencies specified in the `package.json` file.
- This will download and install the required libraries and packages for the application to run.

5. Configure Environment Variables:

For Client

- Navigate to the `/client` directory using `cd /client`.
- Create a new file named `.env` using a text editor or the command line.
- Open the `.env` file in a text editor.
- Add the following lines to the `.env` file, specifying the environment variable values as needed:

```
#Client .env

REACT_APP_ADMIN_SIGN_UP = true #after admin is signed up
then make it false

REACT_APP_SERVER_URL = #add server url for api request
```

```
REACT_APP_HOD_FEATURE= #To Enable HOD Feature true for On
and false for
```

- Save the .env file

For Server

- Navigate to the /server directory using cd /server.
- Create a new file named .env using a text editor or the command line.
- Open the .env file in a text editor.
- Add the following lines to the .env file, specifying the environment variable values as needed:

```
#Server .env

DATABASE = #add mongodb cluster url here

PORT=9002 # port no for server

SECRET_KEY= #secret key for jwt hashing

ADMIN_KEY= #unique keyword for admin signup

SENDER_EMAIL= #email that is to be used for sending email
verification and reset password link

SENDER_PASSWORD= #password of above email that is to be
used for sending email verification and reset password link


# https://myaccount.google.com/lesssecureapps go to this
and enable access to less secure app link otherwise
verification mail will not be send from above email
```

```
ADMIN_EMAIL= #email of admin so that whenever new user is
registered a mail is send to admin

CLIENT_URL= # url for client to include in mail template

REACT_APP_HOD_FEATURE= #To Enable HOD Feature true for On
and false for
```

- Save the .env file.

6. Start the Application:

   For Client

- Navigate to the /client directory using cd /client.
- Start the server by running npm start command.

For Server

- Navigate to the /server directory using cd /server.
- Start the server by running node app.js command.

Testing / Demo

The project is currently hosted on Render.com, and it is accessible through the following link: https://bookit-client-lezg.onrender.com/

Currently for database management, MongoDB Atlas is utilized as the database solution.

Test Emails

Email : adminofficer@acropolis.in

Password : 123456789

Email : neelamrajput@acropolis.in

Password : 123456789

Email : ajayajmera@acropolis.in

Password : 123456789

Email : faculty@acropolis.in

Password : 123456789