BOOK A DOCTOR USING MERN

|  |  |
| --- | --- |
| K .ABINAYA | au410121243001 |
| R.ABINAYA | au410121243002 |
| S.ANBARASI | au410121243004 |
| K.BHUVANASURUTHI | au410121243005 |

**Abstract:**

The **Doctor Appointment Booking System** is a web application developed using the **MERN stack** (MongoDB, Express, React, Node.js) that allows patients to book appointments with doctors and view existing appointments. The system features a simple and intuitive interface where patients can fill out a form with their personal details, select a doctor, and choose a time for their appointment.

The **backend** is powered by **Node.js** and **Express.js**, providing RESTful API endpoints to handle appointment creation and retrieval, with data stored in a **MongoDB** database. The **frontend** is built using **React**, offering a dynamic and responsive user experience. Axios is used for communication between the client and the server, ensuring smooth data handling.

This system helps streamline the appointment booking process for both patients and healthcare providers, making it easy to manage appointments in real-time and providing a scalable solution for future enhancements.

**Technologies Used**:

* **Frontend**: React.js
* **Backend**: Node.js, Express.js
* **Database**: MongoDB
* **Additional Tools**: Axios (for HTTP requests), CORS (for cross-origin resource sharing), Nodemon (for development

**Objective:**

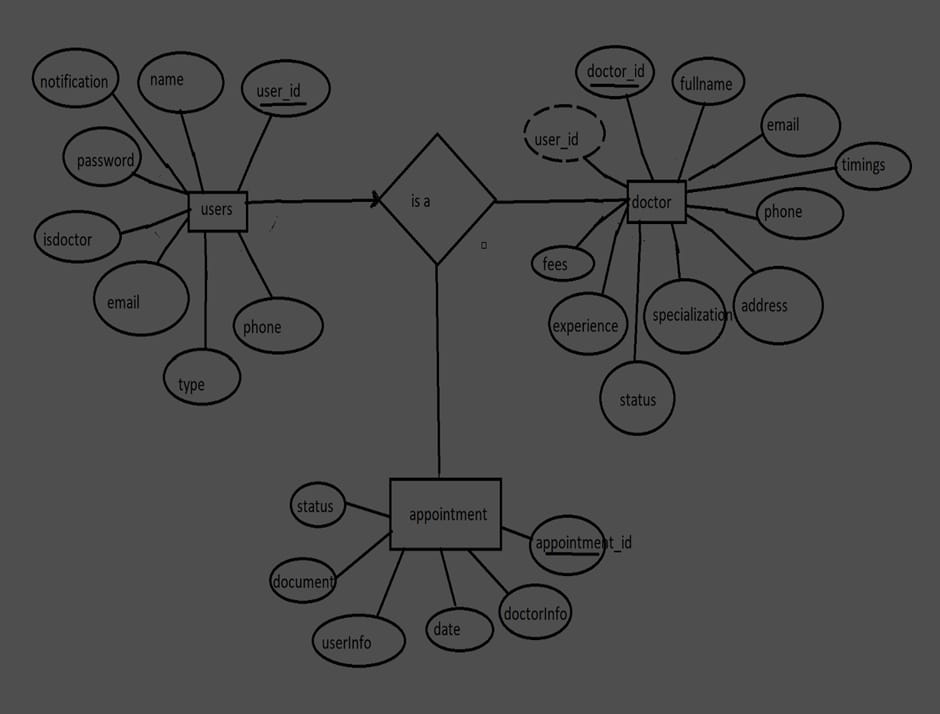
The objective of this project is to develop a **Doctor Appointment Booking System** using the **MERN** stack (MongoDB, Express, React, Node.js). This system allows patients to book appointments with doctors, view a list of existing appointments, and manage them efficiently in a user-friendly interface. By implementing this system, the project aims to simplify the booking process for both patients and healthcare providers while maintaining data integrity and accessibility.

**Key Features:**

1. **Appointment Booking**: Patients can book an appointment by providing their name, the doctor's name, the appointment date, and time.
2. **Appointment Display**: A real-time list of all booked appointments is displayed, allowing patients to view upcoming appointments.
3. **Backend API**: A RESTful API (built with Express and Node.js) handles appointment creation, retrieval, and storage in a MongoDB database.
4. **Frontend Interface**: A React-based frontend allows users to interact with the system easily. It features forms for booking appointments and displaying lists of appointments.
5. **Database Management**: MongoDB serves as the database, storing all appointments securely. The database is accessed via the Express API to manage and retrieve appointment data.

**Methodology:**

* **Frontend Development**: The React.js framework is used to build a dynamic and responsive user interface. React's component-based architecture facilitates the creation of reusable UI elements such as forms and lists.
* **Backend Development**: The backend is developed using Node.js and Express.js, with a RESTful API that exposes endpoints to manage appointment data (e.g., POST for creating appointments, GET for retrieving them).
* **Database**: MongoDB, a NoSQL database, is used for storing appointment data in a flexible, scalable format. The data includes patient name, doctor name, appointment date, and time.
* **HTTP Requests**: Axios is used to make HTTP requests from the React frontend to the Express backend, ensuring smooth communication between the client and server.

**SOURCE CODE**

**Initialize a Node.js project :**

**mkdir doctor-appointment-system**

**cd doctor-appointment-system**

**npm init -y**

**Install the necessary dependencies:**

npm install express mongoose cors body-parser

npm install --save-dev nodemon

**Create the folder structure:**

mkdir server

cd server

mkdir models routes

**Create server.js file:**

// server/server.js

const express = require('express');

const cors = require('cors');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const appointmentRoutes = require('./routes/appointmentRoutes');

const app = express();

// Middleware

app.use(cors());

app.use(bodyParser.json());

// Database connection (MongoDB)

mongoose.connect('mongodb://localhost:27017/doctorAppointments', {

useNewUrlParser: true,

useUnifiedTopology: true,

}).then(() => console.log('MongoDB connected'))

.catch(err => console.log(err));

// Routes

app.use('/api/appointments', appointmentRoutes);

const port = 5000;

app.listen(port, () => {

console.log(`Server is running on http://localhost:${port}`);

});

Create Appointment model (models/Appointment.js):

// server/models/Appointment.js

const mongoose = require('mongoose');

const appointmentSchema = new mongoose.Schema({

patientName: { type: String, required: true },

doctorName: { type: String, required: true },

date: { type: Date, required: true },

time: { type: String, required: true },

});

const Appointment = mongoose.model('Appointment', appointmentSchema);

module.exports = Appointment;

Create routes for handling appointments (routes/appointmentRoutes.js):

// server/routes/appointmentRoutes.js

const express = require('express');

const Appointment = require('../models/Appointment');

const router = express.Router();

// Get all appointments

router.get('/', async (req, res) => {

try {

const appointments = await Appointment.find();

res.json(appointments);

} catch (err) {

res.status(400).json({ message: err.message });

}

});

// Create a new appointment

router.post('/', async (req, res) => {

const { patientName, doctorName, date, time } = req.body;

const appointment = new Appointment({

patientName,

doctorName,

date,

time

});

try {

await appointment.save();

res.status(201).json(appointment);

} catch (err) {

res.status(400).json({ message: err.message });

}

});

module.exports = router;

. Frontend: Set up React

1. Create a React app:

npx create-react-app client

cd client

npm install axios

Create a component to display appointments and book an appointment:

* **AppointmentForm.js:**

// client/src/components/AppointmentForm.js

import React, { useState } from 'react';

import axios from 'axios';

function AppointmentForm() {

const [patientName, setPatientName] = useState('');

const [doctorName, setDoctorName] = useState('');

const [date, setDate] = useState('');

const [time, setTime] = useState('');

const handleSubmit = async (e) => {

e.preventDefault();

const newAppointment = { patientName, doctorName, date, time };

try {

await axios.post('http://localhost:5000/api/appointments', newAppointment);

alert('Appointment booked successfully!');

} catch (err) {

console.error(err);

alert('Failed to book appointment.');

}

};

return (

<form onSubmit={handleSubmit}>

<input

type="text"

placeholder="Patient Name"

value={patientName}

onChange={(e) => setPatientName(e.target.value)}

required

/>

<input

type="text"

placeholder="Doctor Name"

value={doctorName}

onChange={(e) => setDoctorName(e.target.value)}

required

/>

<input

type="date"

value={date}

onChange={(e) => setDate(e.target.value)}

required

/>

<input

type="time"

value={time}

onChange={(e) => setTime(e.target.value)}

required

/>

<button type="submit">Book Appointment</button>

</form>

);

}

export default AppointmentForm;

// client/src/components/AppointmentList.js

import React, { useEffect, useState } from 'react';

import axios from 'axios';

function AppointmentList() {

const [appointments, setAppointments] = useState([]);

useEffect(() => {

async function fetchAppointments() {

try {

const response = await axios.get('http://localhost:5000/api/appointments');

setAppointments(response.data);

} catch (err) {

console.error(err);

}

}

fetchAppointments();

}, []);

return (

<div>

<h2>All Appointments</h2>

<ul>

{appointments.map((appointment) => (

<li key={appointment.\_id}>

{appointment.patientName} - {appointment.doctorName} on {new Date(appointment.date).toLocaleDateString()} at {appointment.time}

</li>

))}

</ul>

</div>

);

}

export default AppointmentList;

App.js:

// client/src/App.js

import React from 'react';

import AppointmentForm from './components/AppointmentForm';

import AppointmentList from './components/AppointmentList';

function App() {

return (

<div className="App">

<h1>Doctor Appointment System</h1>

<AppointmentForm />

<AppointmentList />

</div>

);

}

export default App;

3. Run the Application

1. Start the backend (from the server directory):

Start the frontend (from the client directory):

npm start

**The React app should open at http://localhost:3000, where you can book new appointments and view the list of appointments.**

**4. Testing API with Postman**

**To ensure the backend is working correctly, use Postman to:**

1. **GET http://localhost:5000/api/appointments to fetch all appointments.**
2. **POST http://localhost:5000/api/appointments with a JSON body to create a new appointment:**

**{**

**"patientName": "John Doe",**

**"doctorName": "Dr. Smith",**

**"date": "2024-11-15",**

**"time": "10:00"**

**}**

**SAMPLE OUTPUT:**

**--------------------------------------------------**

**| Doctor Appointment System |**

**--------------------------------------------------**

**[ Patient Name: John Doe ]**

**[ Doctor Name: Dr. Smith ]**

**[ Date: 2024-11-15 ] [ Time: 10:00 ]**

**[ Book Appointment Button ]**

**--------------------------------------------------**

**| All Appointments |**

**--------------------------------------------------**

**1. John Doe - Dr. Smith on 11/15/2024 at 10:00**

**Conclusion**

**Now you have a basic Doctor Appointment System using the MERN stack, where users can book appointments and view all the existing appointments. This is just a starting point, and you can add more features like user authentication, appointment cancellation, validation, or even an admin panel to manage appointments.**