Report for node.js with mongodb

**Connection with mongodb database**

Mongoodb.js

Create an page with name Mongoodb.js

Module require for connecting mongodb

const mongoose=require('mongoose');

connection string

mongoose.connect('mongodb+srv://safvan:safvan13058@safvan.v5qoz.mongodb.net/safvan?retryWrites=true&w=majority&appName=safvan')

Erorr and susscess message

.then(()=>console.log('Connected to MongoDB'))

.catch((err) => console.error('could not connect to MongoDB',err));

Full code

const mongoose=require('mongoose');

mongoose.connect('mongodb+srv://safvan:safvan13058@safvan.v5qoz.mongodb.net/safvan?retryWrites=true&w=majority&appName=safvan')

.then(()=>console.log('Connected to MongoDB'))

.catch((err) => console.error('could not connect to MongoDB',err));

**Creating models in Mongodb**

Models.js

Creating a page for models with name Models.js

Module require for connecting mongodb

const mongoose=require('mongoose');

Creating a model for Signin

const signin =new mongoose.Schema({

    Name:{type:String,required:true},

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

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

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

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

});

const Signin=mongoose.model('Singin',signin);

Creating a model for Doctors

const Doctors=new mongoose.Schema({

    Name:{type:String,required:true},

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

    Email:{type:String},

    Catagory:{type:String},

    Qualification:{type:String},

    Address:{type:String},

    Dateofbirth:{type:String},

    Username:{type:String},

    Password:{type:String},

});

const Doctor=mongoose.model('Doctor',Doctors);

Creating a model for Booking

const booking=new mongoose.Schema({

    Name:{type:String,required:true},

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

    Email:{type:String},

    Doctor:{type:String},

    Age:{type:Number},

    Gender:{type:String},

    Token:{type:Number},

});

const bookings=mongoose.model('Booking',booking);

export the page for Routing

module.exports={Signin,bookings,Doctor};

Full code

const mongoose=require('mongoose');

const signin =new mongoose.Schema({

    Name:{type:String,required:true},

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

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

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

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

});

const Signin=mongoose.model('Singin',signin);

const booking=new mongoose.Schema({

    Name:{type:String,required:true},

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

    Email:{type:String},

    Doctor:{type:String},

    Age:{type:Number},

    Gender:{type:String},

    Token:{type:Number},

});

const bookings=mongoose.model('Booking',booking);

const Doctors=new mongoose.Schema({

    Name:{type:String,required:true},

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

    Email:{type:String},

    Catagory:{type:String},

    Qualification:{type:String},

    Address:{type:String},

    Dateofbirth:{type:String},

    Username:{type:String},

    Password:{type:String},

});

const Doctor=mongoose.model('Doctor',Doctors);

module.exports={Signin,bookings,Doctor};

**index.js**

Modules require in index page

const express=require('express')

const app=express()

const  database=require('./mongoodb');

const nodemailer = require('nodemailer');

const session = require('express-session');

app.use(express.json());

connecting models into index page

const {Signin,bookings}=require('./models/models.js')

Routing other pages

// ------------------admin page-----------------------

const admin=require('./admin.js');

app.use('/admin',admin);

// ------------doc page----------------------

const doctor=require('./doctor.js');

app.use('/doctor',doctor);

// ------------------

const reception=require('./reception.js');

app.use('/reception',reception);

creating session

app.use(session({

    secret:"e3b62cf12a2f7bbe1b674619f00bc96faaed93c4aa9e58f30f11b6b12e1dff6b" ,          // Secret key used to sign the session ID cookie

    resave: false,                     // Do not save session if unmodified

    saveUninitialized: false,          // Do not create a session until something is stored

    // cookie: { maxAge: 120000 }          // Session expiration time (in milliseconds, e.g., 60000 = 1 min)

}));

Creating Signin Api

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

    const {  Name,Phone,Username,Password,Email} = req.body;

    if (!Name || !Phone || !Username || !Password| !Email) {

       res.send( "All fields are required");

    }

    const  Singin= await Signin.find({ Username:Username });

    if (Singin.length === 0) {

        console.log(Email)

        const OTP = Math.floor(100000 + Math.random() \* 900000);

        console.log(OTP);

        var transport = nodemailer.createTransport({

            service: 'gmail',

            auth:{

               user: 'make13058@gmail.com',

               pass: 'keaj awno ztep vcmu'

            }

         });

         message = {

            from: "make13058@gmail.com",

            to: Email,

            subject: "OTP of your Registration",

            text: `Dear ${Name},Your OTP is ${OTP}`

         }

         transport.sendMail(message, function(err, info) {

            if (err) {

               res.json({message:"Not able to send OTP"});

             }

             else {

                req.session.user = {name:Name, phone:Phone,Username:Username,password:Password,email:Email,otp:OTP};

                console.log("sending otp else")

             }});

            req.session.user = { name: Name, phone:Phone,Username:Username,password:Password,email:Email,otp:OTP};

            console.log("send otp")

            res.json({message:"Check your Email for OTP"});

         }

    else{

        return res.status(404).json({ message: 'Username is already exist' });

    }

});

Creating otp validation for signin Api

app.post('/otp',async(req,res)=>

    {

        const {otp} = req.body;

        // const otps=req.session.user.OTP

        const Name=req.session.user.name

        const Phone=req.session.user.phone

        const Username=req.session.user.Username

        const Password=req.session.user.password

        const Email=req.session.user.email

        const Otp=req.session.user.otp

        console.log(Otp)

        if (otp==Otp){

            const newStudent = new Signin({

                Name,Email,Phone,Username,Password

            });

            await newStudent.save();

            res.send({ message: "Student registered successfully", student: newStudent });

        }

});

Creating login Api

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

    if(req.session.login){

        res.send("you are already login")

    }

    else{

    const {user,pass}=req.body

    const  login= await Signin.find({ Username:user ,Password:pass});

       if(login.length===0){

        res.send("Username or Password is wrong,Try again");

       }

       else{

       if(login.length!== 0){

           console.log('login ');

           console.log(login)

           req.session.login={user:user,pass:pass}

           res.send("Login successfully")

      }}}});

Creating logout Api

app.get('/logout', (req, res) => {

    req.session.destroy(err => {

      if (err) {

        return res.status(500).send('Failed to log out');

      }

      res.send('Logged out successfully');

    });

  });

Creating Booking Api

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

    const { name, phone, Doctor,Age,gender,email } = req.body;

    const  booking= await bookings.find({Phone:phone});

      if (booking.length!==0){

        res.send('Your already Appointed')

        console.log(booking.length!==0)

        console.log('Your already Appointed')

      }

      else{

    // console.log(results)

       const lastBooking = await bookings.findOne().sort({ Token: -1 }); // Get the latest booking by Token number

       let Token = 100; // Default token if no bookings exist

       if (lastBooking && lastBooking.Token) {

         Token = lastBooking.Token + 1; // Increment token based on the last booking

         }

                const newbooking = new bookings({

                    Name:name,Email:email,Phone:phone,Age:Age,Doctor:Doctor,Gender:gender,Token

                });

                await newbooking.save();

                res.send(`Your booking is confirmed. Your token number is: ${Token}`);

            }

        });

Creating Delete booking Api

app.delete ('/deletebooking/:id', async (req,res)=>{

            console.log(req.params)

            const {id}=req.params;

             console.log(id)

          const dltbook=await bookings.findByIdAndDelete(id)

                if(!dltbook){

                  res.send("booking not able to delete")

                }

                else{

                  res.send(`delete successfully${dltbook}`)

                }

            });

Full code

const express=require('express')

const app=express()

const  database=require('./mongoodb');

const nodemailer = require('nodemailer');

const session = require('express-session');

app.use(express.json());

const {Signin,bookings}=require('./models/models.js')

// ------------------admin page-----------------------

const admin=require('./admin.js');

app.use('/admin',admin);

// ------------doc page----------------------

const doctor=require('./doctor.js');

app.use('/doctor',doctor);

// ------------------

const reception=require('./reception.js');

app.use('/reception',reception);

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

    res.send(`<h1>Hello</h1>`);

});

app.use(session({

    secret:"e3b62cf12a2f7bbe1b674619f00bc96faaed93c4aa9e58f30f11b6b12e1dff6b" ,          // Secret key used to sign the session ID cookie

    resave: false,                     // Do not save session if unmodified

    saveUninitialized: false,          // Do not create a session until something is stored

    // cookie: { maxAge: 120000 }          // Session expiration time (in milliseconds, e.g., 60000 = 1 min)

}));

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

    const {  Name,Phone,Username,Password,Email} = req.body;

    if (!Name || !Phone || !Username || !Password| !Email) {

       res.send( "All fields are required");

    }

    const  Singin= await Signin.find({ Username:Username });

    if (Singin.length === 0) {

        console.log(Email)

        const OTP = Math.floor(100000 + Math.random() \* 900000);

        console.log(OTP);

        var transport = nodemailer.createTransport({

            service: 'gmail',

            auth:{

               user: 'make13058@gmail.com',

               pass: 'keaj awno ztep vcmu'

            }

         });

         message = {

            from: "make13058@gmail.com",

            to: Email,

            subject: "OTP of your Registration",

            text: `Dear ${Name},Your OTP is ${OTP}`

         }

         transport.sendMail(message, function(err, info) {

            if (err) {

               res.json({message:"Not able to send OTP"});

             }

             else {

                req.session.user = { name: Name, phone:Phone,Username:Username,password:Password,email:Email,otp:OTP};

                console.log("sending otp else")

             }});

            req.session.user = { name: Name, phone:Phone,Username:Username,password:Password,email:Email,otp:OTP};

            console.log("send otp")

            res.json({message:"Check your Email for OTP"});

         }

    else{

        return res.status(404).json({ message: 'Username is already exist' });

    }

    // res.send({ message: "Student registered successfully"});

});

// for validation

app.post('/otp',async(req,res)=>

    {

        const {otp} = req.body;

        // const otps=req.session.user.OTP

        const Name=req.session.user.name

        const Phone=req.session.user.phone

        const Username=req.session.user.Username

        const Password=req.session.user.password

        const Email=req.session.user.email

        const Otp=req.session.user.otp

        console.log(Otp)

        if (otp==Otp){

            const newStudent = new Signin({

                Name,Email,Phone,Username,Password

            });

            await newStudent.save();

            res.send({ message: "Student registered successfully", student: newStudent });

        }

});

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

    if(req.session.login){

        res.send("you are already login")

    }

    else{

    const {user,pass}=req.body

    const  login= await Signin.find({ Username:user ,Password:pass});

       if(login.length===0){

        res.send("Username or Password is wrong,Try again");

       }

       else{

       if(login.length!== 0){

           console.log('login ');

           console.log(login)

           req.session.login={user:user,pass:pass}

           res.send("Login successfully")

      }

}

}});

app.get('/logout', (req, res) => {

    req.session.destroy(err => {

      if (err) {

        return res.status(500).send('Failed to log out');

      }

      res.send('Logged out successfully');

    });

  });

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

    const { name, phone, Doctor,Age,gender,email } = req.body;

    const  booking= await bookings.find({Phone:phone});

      if (booking.length!==0){

        res.send('Your already Appointed')

        console.log(booking.length!==0)

        console.log('Your already Appointed')

      }

      else{

    // console.log(results)

       const lastBooking = await bookings.findOne().sort({ Token: -1 }); // Get the latest booking by Token number

       let Token = 100; // Default token if no bookings exist

       if (lastBooking && lastBooking.Token) {

         Token = lastBooking.Token + 1; // Increment token based on the last booking

         }

                // if () {

                //     console.error("Booking failed:", err);

                //     return res.status(500).send('Booking failed');

                // }

                const newbooking = new bookings({

                    Name:name,Email:email,Phone:phone,Age:Age,Doctor:Doctor,Gender:gender,Token

                });

                await newbooking.save();

                res.send(`Your booking is confirmed. Your token number is: ${Token}`);

            }

        });

        app.delete ('/deletebooking/:id', async (req,res)=>{

            console.log(req.params)

            const {id}=req.params;

             console.log(id)

          const dltbook=await bookings.findByIdAndDelete(id)

                if(!dltbook){

                  res.send("booking not able to delete")

                }

                else{

                  res.send(`delete successfully${dltbook}`)

                }

            });

app.listen(3001)

**doctor.j**s

Modules require for doctor page

const express = require('express');

const doctor = express.Router();

const fs = require('fs');

const puppeteer = require('puppeteer');

const path = require('path');

const {Signin,bookings}=require('./models/models.js')

Creating booking Api

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

    if (!req.session.login) {

        return res.status(401).send('Session not available');

    }

    const { user } = req.session.login;

    try {

        const results = await bookings.find({ Doctor: user });

        if (results.length === 0) {

            return res.status(404).send('No bookings found for this doctor.');

        }

        res.json(results);

    } catch (err) {

        console.error('Error fetching bookings:', err);

        res.status(500).send('Error fetching bookings.');

    }

});

Creating Delete Booking Api

doctor.delete('/deletebooking/:id', async (req, res) => {

    const { id } = req.params;

    try {

        const deletedBooking = await bookings.findByIdAndDelete(id);

        if (!deletedBooking) {

            return res.status(404).send('Booking not found or already deleted.');

        }

        res.send('Booking deleted successfully.');

    } catch (err) {

        console.error('Error deleting booking:', err);

        res.status(500).send('Error deleting booking.');

    }

});

Creating prescription letter Api

doctor.get('/peric2/:id', async (req, res) => {

    const { id } = req.params;

    try {

        const booking = await bookings.findById(id);

        if (!booking) {

            return res.status(404).send('No booking found.');

        }

        const { \_id: ids, Name: name, Age, Place: place, Doctor: doctorName, Gender: gender } = booking;

        const pdfPath = path.join(\_\_dirname, 'report', `prescription${ids}.pdf`);

        async function generatePrescription() {

            let html = fs.readFileSync(path.join(\_\_dirname, 'template.html'), 'utf8');

            const absoluteLogoPath = path.join(\_\_dirname, 'logo', 'logo.jpg');

            const encodedLogoPath = `file://${absoluteLogoPath.replace(/\\/g, '/')}`;

            const date = new Date().toLocaleDateString();

            html = html

                .replace('{{patientName}}', name)

                .replace('{{Age}}', Age)

                .replace('{{date}}', date)

                .replace('{{place}}', place)

                .replace('{{gender}}', gender)

                .replace('{{doctor}}', doctorName)

                .replace('{{doctors}}', doctorName)

                .replace('{{logo}}', encodedLogoPath);

            const browser = await puppeteer.launch();

            const page = await browser.newPage();

            await page.setContent(html);

            await page.pdf({ path: pdfPath, format: 'A4' });

            await browser.close();

            console.log('Prescription PDF generated:', pdfPath);

        }

        await generatePrescription();

        res.json({

            message: 'Prescription generated successfully',

            downloadUrl: `/doctor/prescriptions/prescription${ids}.pdf`,

            printUrl: `/doctor/prescriptions/prescription${ids}.pdf`,

        });

    } catch (error) {

        console.error('Error generating prescription:', error);

        res.status(500).send('Error generating the prescription.');

    }

});

Retriving prescriptions

doctor.use('/prescriptions', express.static(path.join(\_\_dirname, 'report')));

Full code

const express = require('express');

const doctor = express.Router();

const fs = require('fs');

const puppeteer = require('puppeteer');

const path = require('path');

const {Signin,bookings}=require('./models/models.js')

// -------------------------------

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

    if (!req.session.login) {

        return res.status(401).send('Session not available');

    }

    const { user } = req.session.login;

    try {

        const results = await bookings.find({ Doctor: user });

        if (results.length === 0) {

            return res.status(404).send('No bookings found for this doctor.');

        }

        res.json(results);

    } catch (err) {

        console.error('Error fetching bookings:', err);

        res.status(500).send('Error fetching bookings.');

    }

});

// -------------------------------------------------

doctor.delete('/deletebooking/:id', async (req, res) => {

    const { id } = req.params;

    try {

        const deletedBooking = await bookings.findByIdAndDelete(id);

        if (!deletedBooking) {

            return res.status(404).send('Booking not found or already deleted.');

        }

        res.send('Booking deleted successfully.');

    } catch (err) {

        console.error('Error deleting booking:', err);

        res.status(500).send('Error deleting booking.');

    }

});

// -------------------------------------------------

doctor.get('/peric2/:id', async (req, res) => {

    const { id } = req.params;

    try {

        const booking = await bookings.findById(id);

        if (!booking) {

            return res.status(404).send('No booking found.');

        }

        const { \_id: ids, Name: name, Age, Place: place, Doctor: doctorName, Gender: gender } = booking;

        const pdfPath = path.join(\_\_dirname, 'report', `prescription${ids}.pdf`);

        async function generatePrescription() {

            let html = fs.readFileSync(path.join(\_\_dirname, 'template.html'), 'utf8');

            const absoluteLogoPath = path.join(\_\_dirname, 'logo', 'logo.jpg');

            const encodedLogoPath = `file://${absoluteLogoPath.replace(/\\/g, '/')}`;

            const date = new Date().toLocaleDateString();

            html = html

                .replace('{{patientName}}', name)

                .replace('{{Age}}', Age)

                .replace('{{date}}', date)

                .replace('{{place}}', place)

                .replace('{{gender}}', gender)

                .replace('{{doctor}}', doctorName)

                .replace('{{doctors}}', doctorName)

                .replace('{{logo}}', encodedLogoPath);

            const browser = await puppeteer.launch();

            const page = await browser.newPage();

            await page.setContent(html);

            await page.pdf({ path: pdfPath, format: 'A4' });

            await browser.close();

            console.log('Prescription PDF generated:', pdfPath);

        }

        await generatePrescription();

        res.json({

            message: 'Prescription generated successfully',

            downloadUrl: `/doctor/prescriptions/prescription${ids}.pdf`,

            printUrl: `/doctor/prescriptions/prescription${ids}.pdf`,

        });

    } catch (error) {

        console.error('Error generating prescription:', error);

        res.status(500).send('Error generating the prescription.');

    }

});

// ------------------------------------------------------

doctor.use('/prescriptions', express.static(path.join(\_\_dirname, 'report')));

module.exports = doctor;

**Reception.js**

Modules require for reception page

const express = require('express');

const reception = express.Router();

const mongoose = require('mongoose');

const fs = require('fs');

const puppeteer = require('puppeteer');

const path = require('path');

const {Signin,bookings,Doctor}=require('./models/models.js')

Creating Booking Api

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

    const { name, phone, Doctor: doctor, Age, Place, gender } = req.body;

    try {

        // Check if a booking already exists for the phone number

        const existingBooking = await Booking.findOne({ Phone: phone });

        if (existingBooking) {

            return res.status(400).send('You are already appointed.');

        }

        // Get the maximum token value from the bookings collection

        const lastBooking = await Booking.findOne().sort({ Token: -1 });

        const Token = lastBooking ? lastBooking.Token + 1 : 100; // Default token starts at 100

        // Create a new booking

        const newBooking = new Booking({ Name: name, Phone: phone, DoctorName: doctor, Age, Place, gender, Token });

        await newBooking.save();

        res.send(`Your booking is confirmed. Your token number is: ${Token}`);

    } catch (error) {

        console.error('Booking failed:', error);

        res.status(500).send('Booking failed.');

    }

});

Creating Delete booking Api

reception.delete('/deletebooking/:id', async (req, res) => {

    const { id } = req.params;

    try {

        const deletedBooking = await Booking.findByIdAndDelete(id);

        if (!deletedBooking) {

            return res.status(404).send('Booking not found.');

        }

        res.send('Booking deleted successfully.');

    } catch (error) {

        console.error('Error deleting booking:', error);

        res.status(500).send('Error deleting booking.');

    }

});

Creating Display Doctor Api

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

    try {

        const doctors = await Doctor.find();

        res.json(doctors);

    } catch (error) {

        console.error('Error fetching doctors:', error);

        res.status(500).send('Error displaying doctors.');

    }

});

Creating prescription Api

reception.get('/peric2/:id', async (req, res) => {

    const { id } = req.params;

    try {

        const booking = await Booking.findById(id);

        if (!booking) {

            return res.status(404).send('No booking found.');

        }

        const { \_id, Name: name, Age, Place: place, DoctorName: doctorName, gender } = booking;

        const pdfPath = path.join(\_\_dirname, 'report', `prescription${\_id}.pdf`);

        async function generatePrescription() {

            let html = fs.readFileSync(path.join(\_\_dirname, 'template.html'), 'utf8');

            const absoluteLogoPath = path.join(\_\_dirname, 'logo', 'logo.jpg');

            const encodedLogoPath = `file://${absoluteLogoPath.replace(/\\/g, '/')}`;

            const date = new Date().toLocaleDateString();

            html = html

                .replace('{{patientName}}', name)

                .replace('{{Age}}', Age)

                .replace('{{date}}', date)

                .replace('{{place}}', place)

                .replace('{{gender}}', gender)

                .replace('{{doctor}}', doctorName)

                .replace('{{logo}}', encodedLogoPath);

            const browser = await puppeteer.launch();

            const page = await browser.newPage();

            await page.setContent(html);

            await page.pdf({ path: pdfPath, format: 'A4' });

            await browser.close();

            console.log('Prescription PDF generated:', pdfPath);

        }

        await generatePrescription();

        res.json({

            message: 'Prescription generated successfully',

            downloadUrl: `/reception/prescriptions/prescription${\_id}.pdf`,

            printUrl: `/reception/prescriptions/prescription${\_id}.pdf`,

        });

    } catch (error) {

        console.error('Error generating prescription:', error);

        res.status(500).send('Error generating the prescription.');

    }

});

Retriving Prescription Api

reception.use('/prescriptions', express.static(path.join(\_\_dirname, 'report')));

Full code

const express = require('express');

const reception = express.Router();

const mongoose = require('mongoose');

const fs = require('fs');

const puppeteer = require('puppeteer');

const path = require('path');

const {Signin,bookings,Doctor}=require('./models/models.js')

reception.use(express.json());

reception.use(express.urlencoded({ extended: true }));

// ---------------------------------------------

// Route: Create Booking

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

    const { name, phone, Doctor: doctor, Age, Place, gender } = req.body;

    try {

        // Check if a booking already exists for the phone number

        const existingBooking = await Booking.findOne({ Phone: phone });

        if (existingBooking) {

            return res.status(400).send('You are already appointed.');

        }

        // Get the maximum token value from the bookings collection

        const lastBooking = await Booking.findOne().sort({ Token: -1 });

        const Token = lastBooking ? lastBooking.Token + 1 : 100; // Default token starts at 100

        // Create a new booking

        const newBooking = new Booking({ Name: name, Phone: phone, DoctorName: doctor, Age, Place, gender, Token });

        await newBooking.save();

        res.send(`Your booking is confirmed. Your token number is: ${Token}`);

    } catch (error) {

        console.error('Booking failed:', error);

        res.status(500).send('Booking failed.');

    }

});

// ---------------------------------------------

// Route: Delete Booking by ID

reception.delete('/deletebooking/:id', async (req, res) => {

    const { id } = req.params;

    try {

        const deletedBooking = await Booking.findByIdAndDelete(id);

        if (!deletedBooking) {

            return res.status(404).send('Booking not found.');

        }

        res.send('Booking deleted successfully.');

    } catch (error) {

        console.error('Error deleting booking:', error);

        res.status(500).send('Error deleting booking.');

    }

});

// ---------------------------------------------

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

    try {

        const doctors = await Doctor.find();

        res.json(doctors);

    } catch (error) {

        console.error('Error fetching doctors:', error);

        res.status(500).send('Error displaying doctors.');

    }

});

// ------------------------------z---------------

// ---------------------------------------------

reception.get('/peric2/:id', async (req, res) => {

    const { id } = req.params;

    try {

        const booking = await Booking.findById(id);

        if (!booking) {

            return res.status(404).send('No booking found.');

        }

        const { \_id, Name: name, Age, Place: place, DoctorName: doctorName, gender } = booking;

        const pdfPath = path.join(\_\_dirname, 'report', `prescription${\_id}.pdf`);

        async function generatePrescription() {

            let html = fs.readFileSync(path.join(\_\_dirname, 'template.html'), 'utf8');

            const absoluteLogoPath = path.join(\_\_dirname, 'logo', 'logo.jpg');

            const encodedLogoPath = `file://${absoluteLogoPath.replace(/\\/g, '/')}`;

            const date = new Date().toLocaleDateString();

            html = html

                .replace('{{patientName}}', name)

                .replace('{{Age}}', Age)

                .replace('{{date}}', date)

                .replace('{{place}}', place)

                .replace('{{gender}}', gender)

                .replace('{{doctor}}', doctorName)

                .replace('{{logo}}', encodedLogoPath);

            const browser = await puppeteer.launch();

            const page = await browser.newPage();

            await page.setContent(html);

            await page.pdf({ path: pdfPath, format: 'A4' });

            await browser.close();

            console.log('Prescription PDF generated:', pdfPath);

        }

        await generatePrescription();

        res.json({

            message: 'Prescription generated successfully',

            downloadUrl: `/reception/prescriptions/prescription${\_id}.pdf`,

            printUrl: `/reception/prescriptions/prescription${\_id}.pdf`,

        });

    } catch (error) {

        console.error('Error generating prescription:', error);

        res.status(500).send('Error generating the prescription.');

    }

});

reception.use('/prescriptions', express.static(path.join(\_\_dirname, 'report')));

// ---------------------------------------------

module.exports = reception;

**admin.js**

Module require for admin page

const express = require('express');

const admin = express.Router();

const {Signin,bookings,Doctor}=require('./models/models.js')

Creating Doctor registration Api

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

    const { name, Phone, Email, Catagory, Qualification, Address, Dateofbirth, Username, password } = req.body;

    console.log(name, Phone, Email, Catagory, Qualification, Address, Dateofbirth);

    try {

        const existingUser = await Signin.findOne({ Username });

        if (existingUser) {

            return res.status(400).send('The username is already registered.');

        }

        const newSignin = new Signin({ Name: name, Phone, Email, Password: password, Username });

        await newSignin.save()

        const newDoctor = new Doctor({

            Name: name,

            Phone,

            Email,

            Catagory,

            Qualification,

            Address,

            Dateofbirth,

            Username,

            Password: password,

        });

        const result = await newDoctor.save();

        res.send(`Doctor registered successfully: ${result}`);

    } catch (err) {

        console.error('Error during registration:', err);

        res.status(500).send('Error during registration.');

    }

});

Creating delete doctor Api

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

    try {

        const bookings = await Bookings.find();

        if (bookings.length === 0) {

            return res.send('No bookings found.');

        }

        res.json(bookings);

    } catch (err) {

        console.error('Error fetching bookings:', err);

        res.status(500).send('Error fetching bookings.');

    }

});

Creating display booking Api

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

    try {

        const bookings = await Bookings.find();

        if (bookings.length === 0) {

            return res.send('No bookings found.');

        }

        res.json(bookings);

    } catch (err) {

        console.error('Error fetching bookings:', err);

        res.status(500).send('Error fetching bookings.');

    }

});

Creating update details Api

admin.put('/updatedata/:id', async (req, res) => {

    const { id } = req.params;

    const { name, Phone, Email, Catagory, Qualification, Address, Dateofbirth } = req.body;

    try {

        const updatedDoctor = await Doctor.findByIdAndUpdate(

            id,

            { Name: name, Phone, Email, Catagory, Qualification, Address, Dateofbirth },

            { new: true } // Return the updated document

        );

        if (!updatedDoctor) {

            return res.status(404).send('Doctor not found.');

        }

        res.send('Doctor updated successfully.');

    } catch (err) {

        console.error('Error updating doctor:', err);

        res.status(500).send('Error updating doctor.');

    }

});

Creating delete Doctor Api

admin.delete('/delete/:id', async (req, res) => {

    const { id } = req.params;

    try {

        const deletedDoctor = await Doctor.findByIdAndDelete(id);

        if (!deletedDoctor) {

            return res.status(404).send('No doctor found.');

        }

        res.send('Doctor deleted successfully.');

    } catch (err) {

        console.error('Error deleting doctor:', err);

        res.status(500).send('Error deleting doctor.');

    }

});

Full code

const express = require('express');

const admin = express.Router();

const {Signin,bookings,Doctor}=require('./models/models.js')

admin.use(express.json());

admin.use(express.urlencoded({ extended: true }));

// -----------------------------------------------

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

    const { name, Phone, Email, Catagory, Qualification, Address, Dateofbirth, Username, password } = req.body;

    console.log(name, Phone, Email, Catagory, Qualification, Address, Dateofbirth);

    try {

        const existingUser = await Signin.findOne({ Username });

        if (existingUser) {

            return res.status(400).send('The username is already registered.');

        }

        const newSignin = new Signin({ Name: name, Phone, Email, Password: password, Username });

        await newSignin.save();

        const newDoctor = new Doctor({

            Name: name,

            Phone,

            Email,

            Catagory,

            Qualification,

            Address,

            Dateofbirth,

            Username,

            Password: password,

        });

        const result = await newDoctor.save();

        res.send(`Doctor registered successfully: ${result}`);

    } catch (err) {

        console.error('Error during registration:', err);

        res.status(500).send('Error during registration.');

    }

});

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

    try {

        const doctors = await Doctor.find();

        if (doctors.length === 0) {

            return res.send('No doctors found.');

        }

        res.json(doctors);

    } catch (err) {

        console.error('Error fetching doctors:', err);

        res.status(500).send('Error displaying doctors.');

    }

});

// -----------------------------------------------

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

    try {

        const bookings = await Bookings.find();

        if (bookings.length === 0) {

            return res.send('No bookings found.');

        }

        res.json(bookings);

    } catch (err) {

        console.error('Error fetching bookings:', err);

        res.status(500).send('Error fetching bookings.');

    }

});

// -----------------------------------------------

admin.put('/updatedata/:id', async (req, res) => {

    const { id } = req.params;

    const { name, Phone, Email, Catagory, Qualification, Address, Dateofbirth } = req.body;

    try {

        const updatedDoctor = await Doctor.findByIdAndUpdate(

            id,

            { Name: name, Phone, Email, Catagory, Qualification, Address, Dateofbirth },

            { new: true } // Return the updated document

        );

        if (!updatedDoctor) {

            return res.status(404).send('Doctor not found.');

        }

        res.send('Doctor updated successfully.');

    } catch (err) {

        console.error('Error updating doctor:', err);

        res.status(500).send('Error updating doctor.');

    }

});

// -----------------------------------------------

admin.delete('/delete/:id', async (req, res) => {

    const { id } = req.params;

    try {

        const deletedDoctor = await Doctor.findByIdAndDelete(id);

        if (!deletedDoctor) {

            return res.status(404).send('No doctor found.');

        }

        res.send('Doctor deleted successfully.');

    } catch (err) {

        console.error('Error deleting doctor:', err);

        res.status(500).send('Error deleting doctor.');

    }

});

// -------------------------------------------

module.exports = admin;

**template.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Doctor's Prescription</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 20px;

            line-height: 1.6;

        }

        .header {

            display: flex;

            justify-content: space-between;

            align-items: center;

            border-bottom: 2px solid #000;

            padding-bottom: 10px;

            margin-bottom: 20px;

        }

        .header .doctor-info {

            font-size: 14px;

        }

        .header .clinic-info {

            text-align: right;

            font-size: 14px;

        }

        .header .logo {

            text-align: center;

        }

        .content {

            margin-top: 20px;

        }

        .content .section {

            margin-bottom: 20px;

        }

        .content .section h3 {

            margin-bottom: 10px;

            text-decoration: underline;

            font-size: 16px;

        }

        .prescription-table {

            width: 100%;

            border-collapse: collapse;

        }

        .prescription-table th, .prescription-table td {

            border: 1px solid #000;

            padding: 8px;

            text-align: left;

            font-size: 12px;

        }

        .prescription-table th {

            background-color: #f2f2f2;

        }

        .advice {

            font-size: 12px;

            margin-top: 20px;

        }

        .footer {

            margin-top: 50px;

            text-align: right;

            font-size: 12px;

        }

    </style>

</head>

<body>

    <!-- Header Section -->

    <div class="header">

        <div class="doctor-info">

            <strong>Dr. {{doctor}}</strong><br>

            MBBS <br>

        </div>

        <div class="logo">

            <img src="{{logo}}" alt="" style="height: 50px;">

        </div>

        <div class="clinic-info">

            <strong>Landmark Polyclinic</strong><br>

            12/4, Paud Road, Kothrud, Pune - 411023<br>

            Ph: 02025226425, Timing: 09:00 AM - 02:00 PM,<br>

            05:00 PM - 09:00 PM | Closed: Monday

        </div>

    </div>

    <!-- Patient and Diagnosis Details -->

    <div class="content">

        <div class="section">

            <strong>Name:</strong> {{patientName}}<br>

            <strong>Address:</strong> {{place}}<br>

            <strong>Gender:</strong> {{gender}}<br>

            <strong>Age:</strong> {{Age}}<br>

            <strong>Date:</strong> {{date}}<br>

            <!-- <strong>Referred By:</strong> Dr. Avidip<br> -->

        </div>

        <!-- Footer -->

        <div class="footer">

            <!-- <strong>Next Visit:</strong> 13-08-2019<br> -->

            <p>\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_</p>

            <p>Dr.{{doctors}}</p>

            <p>MBBS</p>

        </div>

    </div>

</body>

</html>