**St Joseph’s University**

**BENGALURU - 560027**



**ESTD – 1882**

**DATABASE APPLICATIONS DEVELOPMENT LAB**

**MEDIKARE**

**DOCTOR APPOINTMENT SYSTEM**

**Submitted By:**

Mohammed Naseef Bin Mashood

Reg No: 232CSC10

**St Joseph’s University**

**Bengaluru**



**Laboratory Certification**

This is to certify that MOHAMMED NASEEF BIN MASHOOD (Reg No : 232CSC10) has successfully completed the project titled **“MEDIKARE : DOCTOR APPOINTEMENT SYSTEM”** at St Joseph’s University under the supervision and guidance in the fulfilment of requirements of THIRD semester, Masters of Science (Computer Science), Bengaluru, India.

**Teachers-In-Charge:**

Dr. B Nithya

Dr. B.G. Prashanthi

Dr. Mrinmoyee Bhattacharya

Dr. Shashikala D

Dr. B Nithya

**PG Co-ordinator**

Department of Computer Science

**ACKNOWLEDGEMENT**

We recognize that the magnitude of this project necessitated the cooperation, guidance, and support of many individuals, and we are immensely grateful for their assistance throughout the project's completion.

We extend our sincere appreciation to our Principal, Fr. Dr. Victor Lobo, S.J., for providing us with the necessary facilities and resources to carry out this project.

We are indebted to Dr. B.G Prashanthi, Department of Computer Science at St. Joseph's University, for granting us permission to undertake this project.

Our heartfelt thanks go to our project guide, Prof. Mrinmoyee Bhattacharya, Dr. Shashikala D, Dr. Nithya B from the Department of Computer Science at St. Joseph's University, for their invaluable guidance and continuous encouragement, which played a pivotal role in shaping this project.

We also want to express our gratitude to all the teaching and non-teaching staff members of the Computer Science department at St. Joseph's University for their unwavering support and encouragement, which greatly contributed to the successful completion of our project.

Last but not least, we are deeply appreciative of everyone who, whether directly or indirectly, contributed to the success of this project. Your support was instrumental in making this endeavour a reality

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **CONTENTS** | **PAGE NO** |
| 1 | ABSTRACT | 5 |
| 2 | INTRODUCTION | 6 |
| 3 | CHAPTER 1: |  |
|  | * Problem definition | 7 |
|  | * Objectives | 7 |
| 4 | CHAPTER 2: |  |
|  | * Requirements | 8 |
| 5 | CHAPTER 3: |  |
|  | * Modules used | 9 |
| 6 | CHAPTER 4: |  |
|  | * Coding and implementation | 10 - 98 |

**ABSTRACT**

The healthcare industry is experiencing a digital transformation, with a growing reliance on technology to enhance patient care and streamline services. As telemedicine and online healthcare become more prevalent, there is an increasing need for efficient systems that allow patients to easily access medical appointments while enabling doctors to manage their schedules effectively.

The Doctor Appointment Web Application **MEDIKARE** addresses these needs by offering a user-friendly platform for scheduling appointments. Patients can seamlessly book, reschedule, or cancel appointments in real-time, while doctors benefit from tools to optimize their availability and interactions with patients.

Developed using the **MERN Stack** (MongoDB, Express, React, Node.js), the application utilizes this technology’s scalability and performance. This stack enables high-performance user interfaces and robust backend functionality, allowing for real-time updates and secure data handling. Overall, this solution enhances patient-provider interactions and highlights the crucial role of technology in modern healthcare.

**INTRODUCTION**

Due to the increasing demand for more efficient service delivery and better patient care, the healthcare sector is changing quickly. Systems that let patients easily make appointments and help medical professionals run their offices efficiently are in high demand given the increasing popularity of telemedicine and online healthcare solutions. This need is met by the Doctor Appointment Web Application **MEDIKARE**, which provides a user-friendly interface for users to quickly plan, reschedule, or cancel appointments in real time. With the help of this software, patients and physicians may communicate better and organise their schedules more efficiently.

For this project, the **MERN Stack** (MongoDB, Express, React, Node.js) is crucial, as it integrates the most effective features of modern web development technologies. **MongoDB** offers a flexible and scalable database capable of handling various data structures, while **Express** simplifies server-side operations, making it easier to develop robust APIs. **React** delivers an interactive and dynamic user interface, ensuring an outstanding user experience, and **Node.js** allows for fast and efficient data processing, enabling the application to handle multiple requests effectively.

By leveraging the **MERN Stack**, the Doctor Appointment Web Application provides a secure, adaptable, and scalable solution that meets the evolving demands of the healthcare industry, ultimately enhancing patient care and accessibility.

**CHAPTER 1**

**PROBLEM DEFINITION**

The healthcare industry struggles with inefficient appointment management, leading to frustrated patients and increased administrative workloads for providers. Traditional booking methods often result in communication errors and missed appointments. As telemedicine rises, there is growing demand for a digital solution that streamlines scheduling and enhances patient-provider interaction. **MEDIKARE** aims to address these challenges by offering a user-friendly platform for patients to manage appointments efficiently, while enabling healthcare providers to optimize their schedules and improve patient care.

**OBJECTIVES**

With the **MEDIKARE** project, we aim to develop an intuitive platform that transforms how patients schedule, manage, and track their medical appointments. Our goal is to create a seamless system where patients can easily book, reschedule, or cancel appointments, ensuring convenience and accessibility. By the end of the project, we aim to enhance communication between patients and healthcare providers, facilitating better interaction and improving overall healthcare experiences through real-time updates and secure access to important medical information.

**CHAPTER 2**

**SYSTEM REQUIREMENTS:**

* Operating System: 64-bit Windows 10 or higher.
* Coding Platform: Visual Studio Code.
* Server: MongoDB.
* Web Browser: Latest versions of Chrome, Firefox, Brave.
* Processor: Dual-core processor or better (e.g., Intel i3/i5 or equivalent).
* RAM: Minimum of 4 GB (8 GB recommended for smoother multitasking).
* Network Speed: 2Mbps or higher.

**CHAPTER 3**

**MODULES USED IN THIS PROJECT:**

* Home
* Doctors Listing
* Services
* Contact Us
* FAQ’s
* Login page
* Registration Page (Doctor and Patient)
* Doctor Dashboard
* Appointments View and Verification
* Doctor Details Updation
* Patient Dashboard
* Booking Appointments
* Patient Details Updation
* Admin Dashboard
* Doctor Management

**CHAPTER 4**

**CODING AND IMPLEMENTATION:**

**DoctorSchema.js**

import mongoose from "mongoose";

const DoctorSchema = new mongoose.Schema(

  {

    email: { type: String, required: true, unique: true },

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

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

    phone: { type: Number },

    photo: { type: String },

    ticketPrice: { type: Number },

    role: {

      type: String,

    },

    specialization: { type: String },

    qualifications: {

      type: Array,

    },

    experiences: {

      type: Array,

    },

    bio: { type: String, maxLength: 50 },

    about: { type: String },

    timeSlots: { type: Array },

    reviews: [{ type: mongoose.Types.ObjectId, ref: "Review" }],

    averageRating: {

      type: Number,

      default: 0,

    },

    totalRating: {

      type: Number,

      default: 0,

    },

    isApproved: {

      type: String,

      enum: ["pending", "approved", "cancelled"],

      default: "pending",

    },

    deletedAt: {

      type: Date,

      default: null,

    },

    appointments: [{ type: mongoose.Types.ObjectId, ref: "Appointment" }],

  },

  { timestamps: true }

);

export default mongoose.model("Doctor", DoctorSchema);

**FaqSchema.js**

import mongoose from "mongoose";

const FaqSchema = new mongoose.Schema({

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

  content: { type: String, required: true, maxLength: 250 },

});

export default mongoose.model("Faq", FaqSchema);

**MessageSchema.js**

import mongoose from "mongoose";

const MessageSchema = new mongoose.Schema({

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

  subject: { type: String, required: true, maxLength: 150 },

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

});

export default mongoose.model("Messages", MessageSchema);

**ReviewSchema.js**

import mongoose from "mongoose";

import Doctor from "./DoctorSchema.js";

const reviewSchema = new mongoose.Schema(

  {

    doctor: {

      type: mongoose.Types.ObjectId,

      ref: "Doctor",

    },

    user: {

      type: mongoose.Types.ObjectId,

      ref: "User",

    },

    reviewText: {

      type: String,

      required: true,

    },

    rating: {

      type: Number,

      required: true,

      min: 0,

      max: 5,

      default: 0,

    },

  },

  { timestamps: true }

);

reviewSchema.pre(/^find/, function (next) {

  this.populate({

    path: "user",

    select: "name photo",

  });

  next();

});

reviewSchema.statics.calcAvgRatings = async function (doctorId) {

  // 'this' points to current review

  const stats = await this.aggregate([

    {

      $match: { doctor: doctorId },

    },

    {

      $group: {

        \_id: "$doctor",

        numOfRating: { $sum: 1 },

        avgRating: { $avg: "$rating" },

      },

    },

  ]);

  await Doctor.findByIdAndUpdate(doctorId, {

    totalRating: stats[0].numOfRating,

    averageRating: stats[0].avgRating,

  });

};

reviewSchema.post("save", function () {

  this.constructor.calcAvgRatings(this.doctor);

});

export default mongoose.model("Review", reviewSchema);

**ServiceSchema.js**

import mongoose from "mongoose";

const ServicesSchema = new mongoose.Schema({

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

  description: { type: String, required: true, maxLength: 150 },

});

export default mongoose.model("Services", ServicesSchema);

**UsersSchemajs**

import mongoose from "mongoose";

const UserSchema = new mongoose.Schema({

  email: { type: String, required: true, unique: true },

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

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

  phone: { type: Number },

  photo: { type: String },

  role: {

    type: String,

    enum: ["patient", "admin"],

    default: "patient", },

  gender: { type: String, enum: ["male", "female", "other"] },

  bloodType: { type: String },

  appointments: [{ type: mongoose.Types.ObjectId, ref: "Appointment" }],

});

export default mongoose.model("User", UserSchema);

**Front-End**

**Home.jsx**

import heroImg01 from "../assets/images/hero-img01.png";

import heroImg02 from "../assets/images/hero-img02.png";

import heroImg03 from "../assets/images/hero-img03.png";

import icon1 from "../assets/images/icon01.png";

import icon2 from "../assets/images/icon02.png";

import icon3 from "../assets/images/icon03.png";

import featureImg from "../assets/images/feature-img.png";

import videoIcon from "../assets/images/video-icon.png";

import avatarIcon from "../assets/images/avatar-icon.png";

import About from "../components/About/About";

import { Link } from "react-router-dom";

import { BsArrowRight } from "react-icons/bs";

import ServiceList from "../components/Services/ServiceList";

import DoctorList from "../components/Doctors/DoctorList";

import faqImg from "../assets/images/faq-img.png";

import FaqList from "../components/Faq/FaqList";

import Testimonial from "../components/Testimonial/Testimonial";

const Home = () => {

  return (

    <>

      {/\* HERO SECTION \*/}

      <section className="hero\_section pt-[50px] md:pt-[80px] xl:pt-[10px] 2xl:h-[800px]">

        <div className="container">

          <div className="flex flex-col lg:flex-row gap-[90px] items-center justify-between">

            {/\* Hero Content \*/}

            <div>

              <div className="lg:w-[570px]">

                <h1 className="text-[36px] leading-[46px] text-headingColor font-[800] md:text-[60px] md:leading-[70px]">

                  Healthcare, Streamlined for you

                </h1>

                <p className="text\_para pt-5">

                  MediKare allows you to book appointments in real time in just

                  a few taps, whether it is scheduling routine check-ups or

                  seeking specialist care.

                  <br />

                  Medikare is the smarter, faster way to manage your health, all

                  in one place

                </p>

                <Link to="/doctors">

                  <button className="btn">Request an Appointment</button>

                </Link>

                {/\* Hero Counter \*/}

                <div className="mt-[30px] lg:mt-[70px] flex flex-col md:flex-row lg:items-center gap-5 lg:gap-[30px]">

                  <div>

                    <h2 className="text-[36px] leading-[56px] lg:text-[44px] lg:leading-[54px] font-[700] text-headingColor">

                      10+

                    </h2>

                    <span className="w-[100px] h-2 bg-lighterColor rounded-full block mt-[-8px]"></span>

                    <p className="text\_\_para mt-[8px]">Years of Experience</p>

                  </div>

                  <div>

                    <h2 className="text-[36px] leading-[56px] lg:text-[44px] lg:leading-[54px] font-[700] text-headingColor">

                      15+

                    </h2>

                    <span className="w-[100px] h-2 bg-lighterColor rounded-full block mt-[-8px]"></span>

                    <p className="text\_\_para mt-[8px]">Clinic Location</p>

                  </div>

                  <div>

                    <h2 className="text-[36px] leading-[56px] lg:text-[44px] lg:leading-[54px] font-[700] text-headingColor">

                      100%

                    </h2>

                    <span className="w-[100px] h-2 bg-lighterColor rounded-full block mt-[-8px]"></span>

                    <p className="text\_\_para mt-[8px]">Patient Satisfaction</p>

                  </div>

                </div>

              </div>

            </div>

            {/\* Hero Content 2 \*/}

            <div className="flex gap-[30px] justify-end">

              <div>

                <img className="w-full" src={heroImg01} alt="" />

              </div>

            </div>

          </div>

        </div>

      </section>

      {/\* Hero Section end \*/}

      <section>

        <div className="xl:w-[470px] mx-auto">

          <h2 className="heading text-center text-[50px]">

            Providing the best medical services

          </h2>

          <p className="text\_\_para text-center">

            World-class care for everyone. Our health System offers unmatched,

            expert health care.

          </p>

        </div>

        <div className="grid grid-cols md:grid-cols-2 lg:grid-cols-3 gap-5 lg:gap-[30px] mt-[30px] lg:mt-[55px] text-center">

          <div className="py-[30px] px-5">

            <div className="flex items-center justify-center">

              <img src={icon2} className="xl:px-20" />

            </div>

            <div className="mt-[30px]">

              <h2 className="text-[26px] leading-9 text-headingColor font-[7]">

                Get Best Treatments

              </h2>

              <p className="text-[16px] leading-7 text-textColor font-[400] mt-4 ">

                Medikare has just the best doctors for any treatments you need.

                Patients leave healthy with the care from our experts ranging

                from dermatology to surgery

              </p>

            </div>

          </div>

          <div className="py-[30px] px-5">

            <div className="flex items-center justify-center">

              <img src={icon1} className="xl:px-20" />

            </div>

            <div className="mt-[30px]">

              <h2 className="text-[26px] leading-9 text-headingColor font-[7]">

                Find your Doctor

              </h2>

              <p className="text-[16px] leading-7 text-textColor font-[400] mt-4 ">

                World-class care for everyone. Our health system offers

                unmatched, expert health care. From the lab to the clinic

              </p>

            </div>

          </div>

          <div className="py-[30px] px-5">

            <div className="flex items-center justify-center">

              <img src={icon3} className="xl:px-20" />

            </div>

            <div className="mt-[30px]">

              <h2 className="text-[26px] leading-9 text-headingColor font-[7]">

                Book Appointment

              </h2>

              <p className="text-[16px] leading-7 text-textColor font-[400] mt-4 ">

                Booking appointments with Medikare is just a few taps, find the

                best doctor for you with our system, it will never be easier

                than this.

              </p>

            </div>

          </div>

        </div>

      </section>

      {/\* About Section \*/}

      <About />

      {/\* About Section End \*/}

      {/\* Services Section \*/}

      <section>

        <div className="container">

          <div className="xl:w-[470px] mx-auto">

            <h2 className="heading text-center text-[50px]">

              Our medical services

            </h2>

            <p className="text\_\_para text-center">

              World Class care for all, these are the services we provide

              especially

            </p>

          </div>

          <ServiceList />

        </div>

      </section>

      {/\* Services End \*/}

      {/\* Features Start \*/}

      <section>

        <div className="container">

          <div className="flex items-center justify-between flex-col lg:flex-row">

            <div className="xl:w-[670px]">

              <h2 className="heading text-[50px]">

                Get virtual treatment <br /> anytime.

              </h2>

              <ul className="pl-4">

                <li className="text\_\_para">

                  1. Schedule the appointment directly.

                </li>

                <li className="text\_\_para">

                  2. Search for your physician here, and contact their office.

                </li>

                <li className="text\_\_para">

                  3. View our physicians who are accepting new patients, use the

                  online scheduling tool to select an apponintment time.

                </li>

              </ul>

            </div>

            <div className="relative z-10 xl:w-[770px] flex justify-end mt-[50px] lg:mt-0">

              <img src={featureImg} alt="" className="w-3/4" />

            </div>

          </div>

        </div>

      </section>

      {/\* Features End \*/}

      {/\* Doctors Section \*/}

      <section>

        <div className="container">

          <div className="xl:w-[470px] mx-auto">

            <h2 className="heading text-center text-[50px]">

              Our Great Doctors

            </h2>

            <p className="text\_\_para text-center">

              World-class care for everyone. Our health systems offers

              unmatched, expert health care.

            </p>

          </div>

          <DoctorList />

        </div>

      </section>

      {/\* Doctors End \*/}

      {/\* Faq Section Start\*/}

      <section>

        <div className="container mt-5">

          <div className="flex justify-between gap-[50px] xl:gap-8">

            <div className="w-1/2 hidden relative mx-auto md:block">

              <img src={faqImg} alt="" className="sticky top-28" />

            </div>

            <div className="w-full md:w-1/2">

              <h2 className="heading text-[50px]">

                Frequently asked questions by our beloved patients

              </h2>

              <FaqList />

            </div>

          </div>

        </div>

      </section>

      {/\* Faq Section End \*/}

      {/\* Testimonials \*/}

      <section>

        <div className="container">

          <div className="xl:w-[470px] mx-auto">

            <h2 className="heading text-center text-[50px]">

              What our patients say

            </h2>

          </div>

          <Testimonial />

        </div>

      </section>

      {/\* Testimonials end \*/}

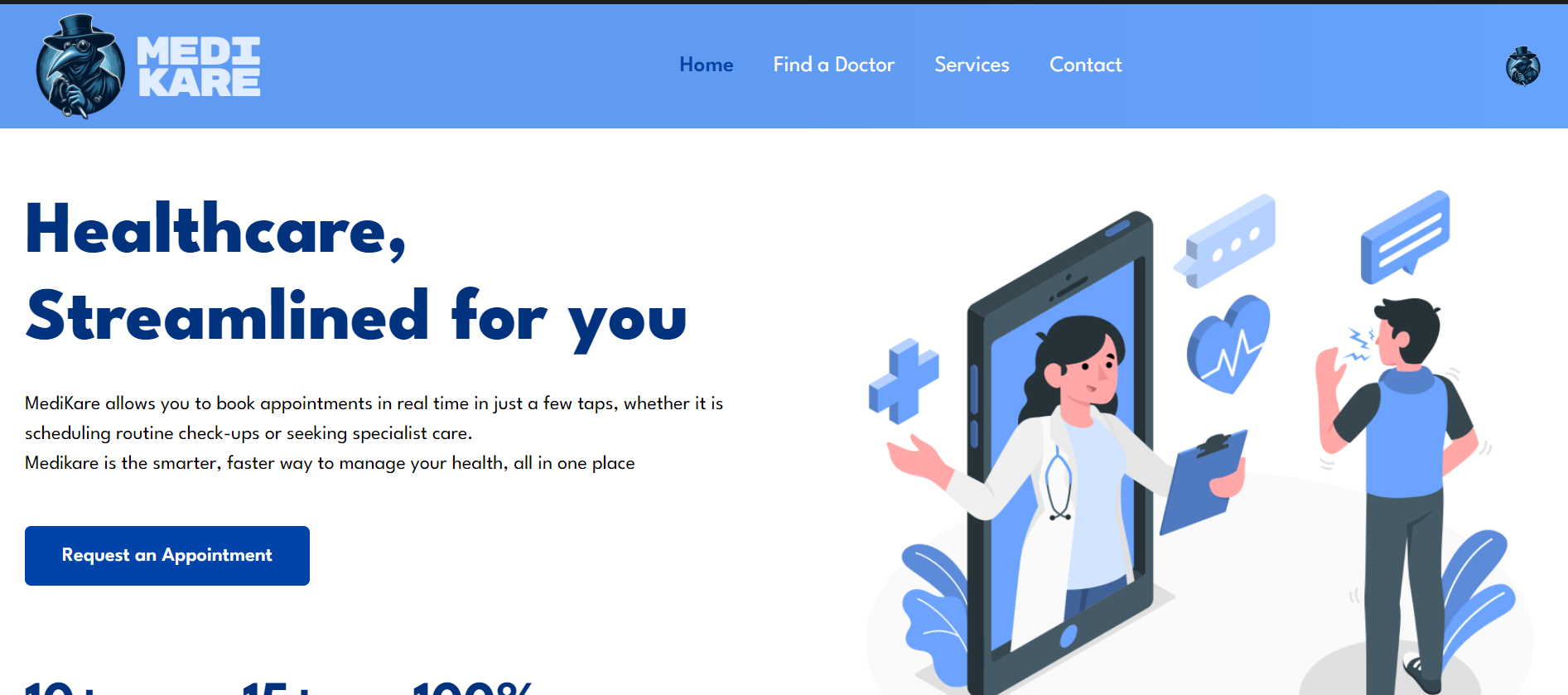
    </>

  );

};

export default Home;

**Output**

****

**Doctors.jsx**

import { doctors } from "../../assets/data/doctors";

import DoctorCard from "../../components/Doctors/DoctorCard";

import { BASE\_URL } from "../../config";

import useFetchData from "../../hooks/useFetchData";

import Loader from "../../components/Loader/Loading";

import Error from "../../components/Error/Error";

import { useEffect, useState } from "react";

const Doctors = () => {

  const [query, setQuery] = useState("");

  const [debounceQuery, setDebounceQuery] = useState("");

  const handleSearch = () => {

    setQuery(query.trim());

    console.log("handle Search");

  };

  useEffect(() => {

    const timeout = setTimeout(() => {

      setDebounceQuery(query);

    }, 700);

    return () => clearTimeout(timeout);

  }, [query]);

  const {

    data: doctors,

    loading,

    error,

  } = useFetchData(`${BASE\_URL}/doctors?query=${debounceQuery}`);

  return (

    <>

      <section className="pb-4">

        <div className="container text-center">

          <h2 className="heading">Find a Doctor</h2>

          <div className="max-w-[570px] mt-[30px] mx-auto bg-[#0066ff2c] rounded-md flex items-center justify-between">

            <input

              type="search"

              className="py-4 pl-4 pr-2 bg-transparent w-full focus:outline-none cursor-pointer placeholder:text-textColor"

              placeholder="Search Doctor"

              value={query}

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

            />

            <button

              className="btn mt-0 me-[5px] rounded-r-md"

              onClick={handleSearch}

            >

              Search

            </button>

          </div>

        </div>

      </section>

      <section>

        <div className="container">

          {loading && <Loader />}

          {error && <Error />}

          {!loading && !error && (

            <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-5">

              {doctors.map((doctor) => (

                <DoctorCard key={doctor.id} doctor={doctor} />

              ))}

            </div>

          )}

        </div>

      </section>

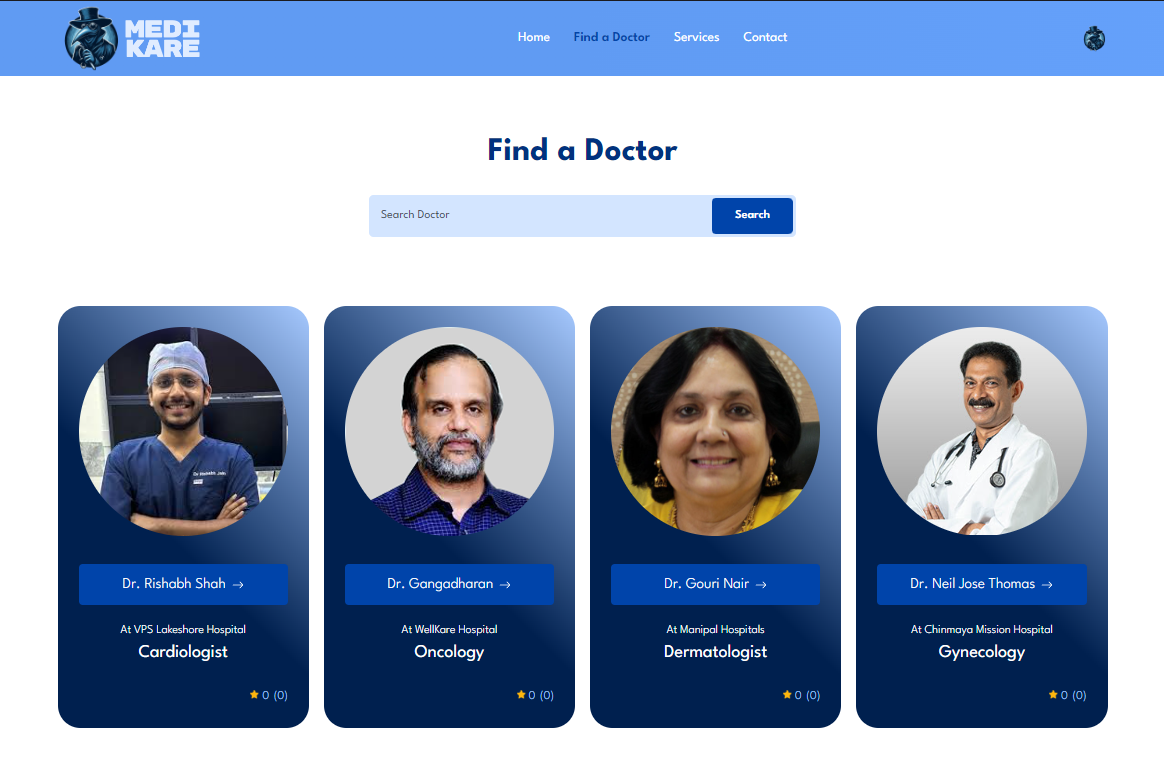
    </>

  );

};

export default Doctors;

**Output:**



**Services.jsx**

import { services } from "../assets/data/services";

import ServiceCard from "../components/Services/ServiceCard";

import ServiceList from "../components/Services/ServiceList";

const Services = () => {

  return (

    <section>

      <div className="container text-center">

        <h2 className="heading">Our Medical Services</h2>

        <p className="mb-8 lg:mb-16 font-light text-center text\_\_para">

          World-class care for everyone. Our health systems offers unmatched,

          expert health care.

        </p>

        <ServiceList />

      </div>

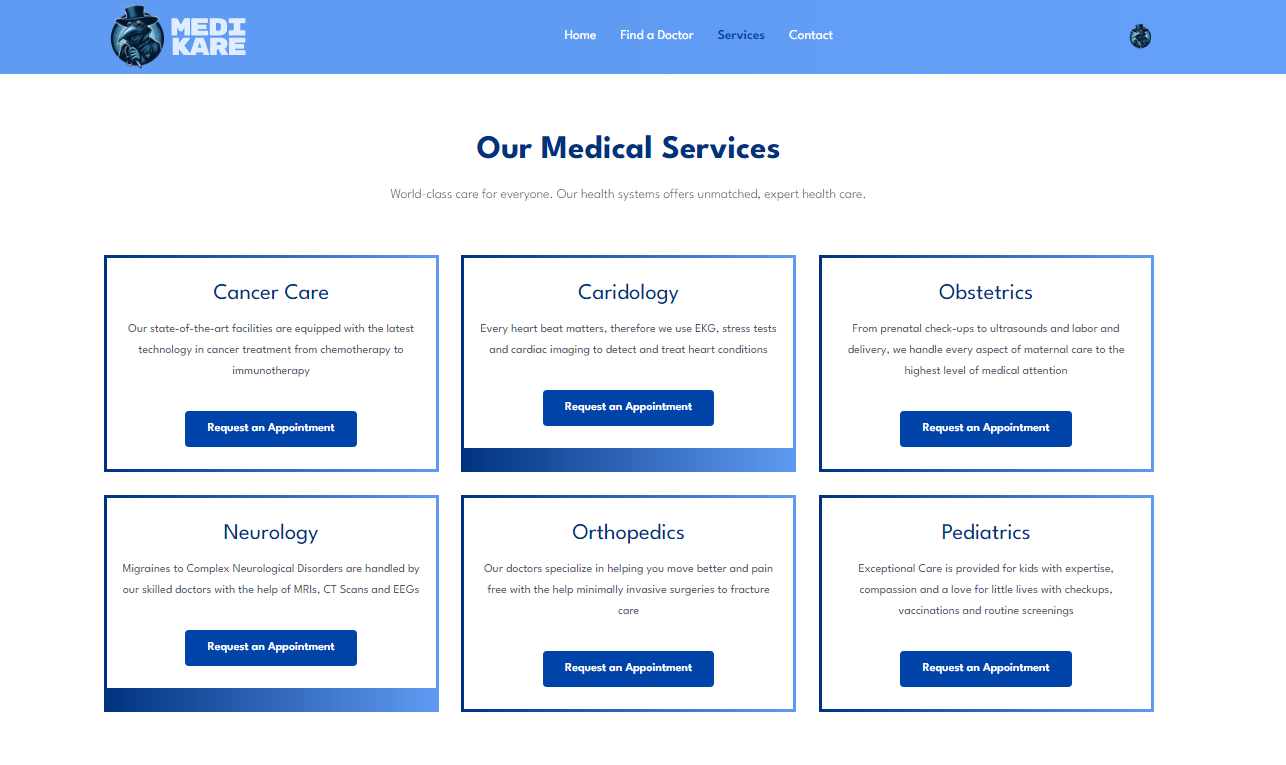
    </section>

  );

};

export default Services;

**Output:**



**Contact.jsx**

import { useState } from "react";

import { useNavigate, useParams } from "react-router-dom";

import { BASE\_URL, token } from "../config";

import { toast } from "react-toastify";

import SyncLoader from "react-spinners/SyncLoader";

const Contact = () => {

  const [contact, setContact] = useState({

    email: "",

    subject: "",

    message: "",

  });

  const [loading, setLoading] = useState(false);

  const navigate = useNavigate();

  const handleInputChange = (e) => {

    setContact({ ...contact, [e.target.name]: e.target.value });

  };

  const submitHandler = async (event) => {

    event.preventDefault();

    setLoading(true);

    try {

      const res = await fetch(`${BASE\_URL}/message`, {

        method: "post",

        headers: {

          "Content-Type": "application/json",

        },

        body: JSON.stringify(contact),

      });

      const result = await res.json();

      if (!res.ok) {

        throw new Error(result.message);

      }

      setLoading(false);

      toast.success(result.message);

      navigate(0);

    } catch (err) {

      toast.error(err.message);

      setLoading(false);

    }

  };

  return (

    <section>

      <div className="px-4 sm:px-20 lg:px-4 lg:w-[1024px] mx-auto max-w-wscreen-md">

        <h2 className="heading text-center">Contact Us</h2>

        <p className="mb-8 lg:mb-16 font-light text-center text\_\_para">

          Got a technical issue? Want to send feedback about a beta feature? Let

          us know.

        </p>

        <form action="#" className="space-y-8" onSubmit={submitHandler}>

          <div>

            <label htmlFor="email" className="form\_\_label">

              Your Email\*

            </label>

            <input

              type="email"

              id="email"

              name="email"

              value={contact.email}

              onChange={handleInputChange}

              placeholder="example@gmail.com"

              required

              className="form\_\_input mt01"

            />

          </div>

          <div>

            <label htmlFor="subject" className="form\_\_label">

              Subject\*

            </label>

            <input

              type="text"

              id="subject"

              name="subject"

              required

              value={contact.subject}

              onChange={handleInputChange}

              placeholder="Let us know how we can help you"

              className="form\_\_input mt-1"

            />

          </div>

          <div className="sm:col-span-2">

            <label htmlFor="message" className="form\_\_label">

              Your Message\*

            </label>

            <textarea

              rows="6"

              type="text"

              id="message"

              name="message"

              required

              value={contact.message}

              onChange={handleInputChange}

              placeholder="Leave a comment..."

              className="form\_\_input mt-1"

            />

          </div>

          <button type="submit" className="btn rounded sm:w-fit">

            {loading ? <SyncLoader size={10} color="#ffffff" /> : "Submit"}

          </button>

        </form>

      </div>

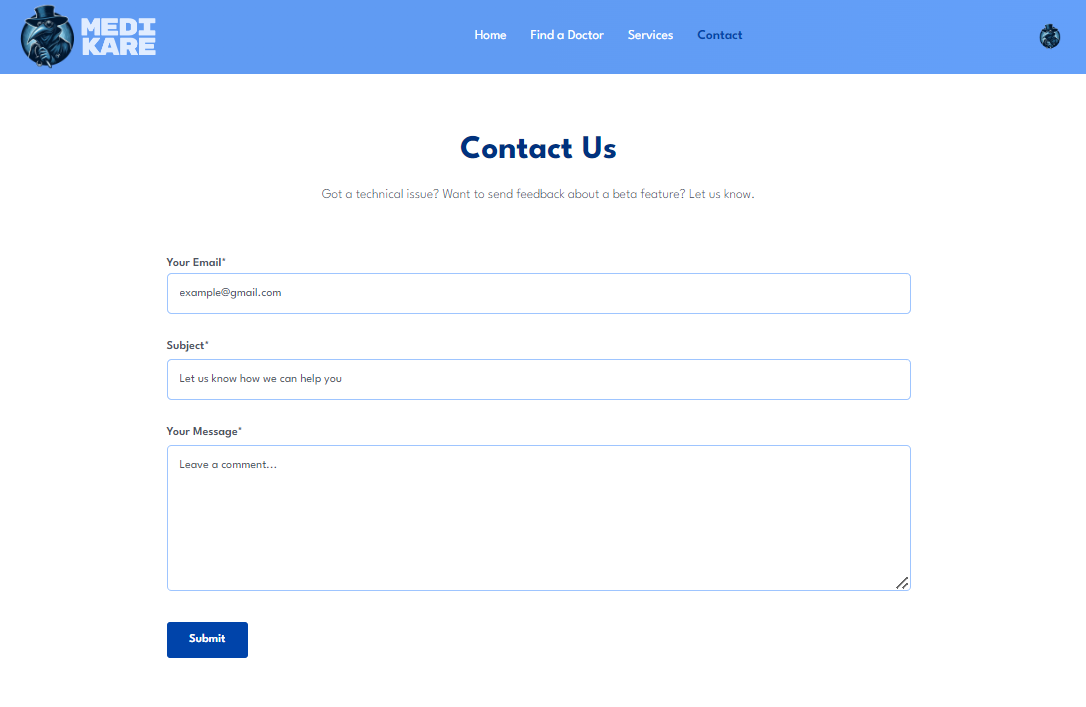
    </section>

  );

};

export default Contact;

**Output:**



**Login.jsx**

import { useState, useContext } from "react";

import { Link, useNavigate } from "react-router-dom";

import { BASE\_URL } from "../config";

import { toast } from "react-toastify";

import { authContext } from "../context/AuthContext.jsx";

import SyncLoader from "react-spinners/SyncLoader.js";

const Login = () => {

  const [loading, setLoading] = useState(false);

  const [formData, setFormData] = useState({

    email: "",

    password: "",

  });

  const navigate = useNavigate();

  const { dispatch } = useContext(authContext);

  const handleInputChange = (e) => {

    setFormData({ ...formData, [e.target.name]: e.target.value });

  };

  const submitHandler = async (event) => {

    event.preventDefault();

    setLoading(true);

    try {

      const res = await fetch(`${BASE\_URL}/auth/login`, {

        method: "post",

        headers: {

          "Content-Type": "application/json",

        },

        body: JSON.stringify(formData),

      });

      const result = await res.json();

      if (!res.ok) {

        throw new Error(result.message);

      }

      dispatch({

        type: "LOGIN\_SUCCESS",

        payload: {

          user: result.data,

          token: result.token,

          role: result.role,

        },

      });

      setLoading(false);

      toast.success(result.message);

      result.role == "admin" ? navigate("/admin") : navigate("/home");

    } catch (err) {

      toast.error(err.message);

      setLoading(false);

    }

  };

  return (

    <section className="px-5 lg:px-0">

      <div className="w-full max-w-[570px] mx-auto rounded-lg shadow-md md:p-10">

        <h3 className="text-headingColor text-[40px] leading-9 font-bold mb-10">

          <span className="text-primaryColor">Welcome</span> Back!

        </h3>

        <form className="py-4 md:py-0" onSubmit={submitHandler}>

          <div className="mb-5">

            <input

              type="email"

              placeholder="Enter Your Email"

              name="email"

              value={formData.email}

              onChange={handleInputChange}

              className="w-full py-3 border-b border-solid border-[#0066ff61 focus:outline-none focus:border-b-primaryColor text-[16px] leading-7 text-headingColor placeholder:text-textColor cursor-pointer"

              required

            />

          </div>

          <div className="mb-5">

            <input

              type="password"

              placeholder="Password"

              name="password"

              value={formData.password}

              onChange={handleInputChange}

              className="w-full py-3 border-b border-solid border-[#0066ff61 focus:outline-none focus:border-b-primaryColor text-[16px] leading-7 text-headingColor placeholder:text-textColor cursor-pointer"

              required

            />

          </div>

          <div className="mt-7">

            <button

              type="submit"

              className="w-full bg-primaryColor text-white text-[18px] leading-[30px] rounded-lg px-4 py-3"

            >

              {loading ? <SyncLoader size={10} color="#ffffff" /> : "Login"}

            </button>

          </div>

          <p className="mt-5 text-textColor text-center">

            Don&apos;t have an account?

            <Link to="/register" className="text-primaryColor font-medium ml-1">

              Register

            </Link>

          </p>

        </form>

      </div>

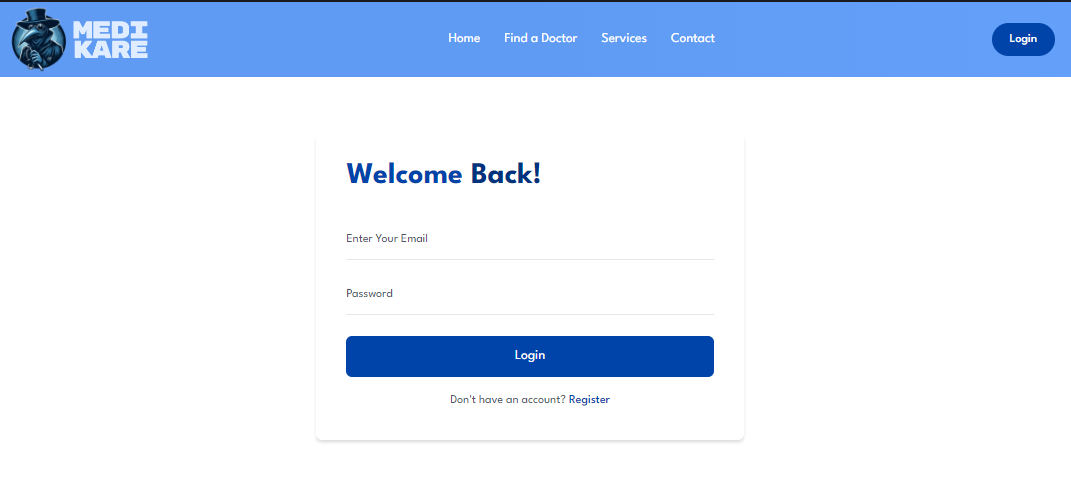
    </section>

  );

};

export default Login;

**Output:**



**PatientSignup.jsx**

import { useState } from "react";

import patientSignupImg from "../assets/images/patient-signup.png";

import { Link, useNavigate } from "react-router-dom";

import uploadImageToCloudinary from "../utils/uploadCloudinary";

import { BASE\_URL } from "../config";

import { toast } from "react-toastify";

import SyncLoader from "react-spinners/SyncLoader";

const PatientSignup = () => {

  const [selectedFile, setSelectedFile] = useState(null);

  const [previewURL, setPreviewURL] = useState("");

  const [loading, setLoading] = useState(false);

  const [formData, setFormData] = useState({

    name: "",

    email: "",

    password: "",

    photo: selectedFile,

    gender: "",

    role: "patient",

  });

  const navigate = useNavigate();

  const handleInputChange = (e) => {

    setFormData({ ...formData, [e.target.name]: e.target.value });

  };

  const handleFileInputChange = async (event) => {

    const file = event.target.files[0];

    const data = await uploadImageToCloudinary(file);

    setPreviewURL(data.url);

    setSelectedFile(data.url);

    setFormData({ ...formData, photo: data.url });

  };

  const submitHandler = async (event) => {

    event.preventDefault();

    setLoading(true);

    try {

      const res = await fetch(`${BASE\_URL}/auth/register`, {

        method: "post",

        headers: {

          "Content-Type": "application/json",

        },

        body: JSON.stringify(formData),

      });

      const { message } = await res.json();

      if (!res.ok) {

        throw new Error(message);

      }

      setLoading(false);

      toast.success(message);

      navigate("/login");

    } catch (err) {

      toast.error(err.message);

      setLoading(false);

    }

  };

  return (

    <section className="px-5 xl:px-0">

      <div className="max-w-[1170px] mx-auto">

        <div className="grid grid-cols-1 lg:grid-cols-2">

          {/\* IMAGE BOX \*/}

          <div className="hidden lg:block w-fit mx-auto bg-darkerColor rounded-2xl">

            <h2 className="text-center py-7 text-lighterColor uppercase font-extrabold text-2xl">

              Patient Sign-Up

            </h2>

            <figure className="rounded-b-2xl">

              <img

                src={patientSignupImg}

                alt=""

                className="w-full rounded-b-2xl"

              />

            </figure>

          </div>

          {/\* SIGN UP FORM \*/}

          <div className="rounded-l-lg lg:pl-16 py-10">

            <h3 className="hidden lg:block text-headingColor text-[50px] leading-9 font-bold mb-10">

              Create an <span className="text-primaryColor">account</span>

            </h3>

            <h3 className="block lg:hidden text-headingColor text-[50px] leading-9 font-bold mb-10">

              Create a{" "}

              <span className="text-primaryColor">Patient account</span>

            </h3>

            <form onSubmit={submitHandler}>

              <div className="mb-5">

                <input

                  type="text"

                  placeholder="Full Name"

                  name="name"

                  value={formData.name}

                  onChange={handleInputChange}

                  className="w-full pr-4 py-3 border-b border-solid border-[#0066ff61 focus:outline-none focus:border-b-primaryColor text-[16px] leading-7 text-headingColor placeholder:text-textColor cursor-pointer"

                  required

                />

              </div>

              <div className="mb-5">

                <input

                  type="email"

                  placeholder="Enter your Email"

                  name="email"

                  value={formData.email}

                  onChange={handleInputChange}

                  className="w-full pr-4 py-3 border-b border-solid border-[#0066ff61 focus:outline-none focus:border-b-primaryColor text-[16px] leading-7 text-headingColor placeholder:text-textColor cursor-pointer"

                  required

                />

              </div>

              <div className="mb-5">

                <input

                  type="password"

                  placeholder="Password"

                  name="password"

                  value={formData.password}

                  onChange={handleInputChange}

                  className="w-full pr-4 py-3 border-b border-solid border-[#0066ff61 focus:outline-none focus:border-b-primaryColor text-[16px] leading-7 text-headingColor placeholder:text-textColor cursor-pointer"

                  required

                />

              </div>

              <div className="mb-5 flex items-center justify-between">

                <label className="text-headingColor font-bold text-[16px] leading-7">

                  Gender:

                  <select

                    name="gender"

                    value={formData.gender}

                    onChange={handleInputChange}

                    className="text-textColor font-semibold text-[15px] leading-7 px-4 py-3 focus:outline-none"

                    id=""

                  >

                    <option value="">Select</option>

                    <option value="male">Male</option>

                    <option value="female">Female</option>

                    <option value="other">Other</option>

                  </select>

                </label>

              </div>

              <div className="mb-5 flex items-center gap-3">

                {selectedFile && (

                  <figure className="w-[60px] h-[60px] rounded-full border-2 border-solid border-primaryColor flex items-center justify-center overflow-hidden">

                    <img

                      src={previewURL}

                      alt=""

                      className="w-full rounded-full"

                    />

                  </figure>

                )}

                <div className="relative w-[160px] h-[50px]">

                  <input

                    type="file"

                    name="photo"

                    id="customFile"

                    onChange={handleFileInputChange}

                    accept=".jpg,.png"

                    className="absolute top-0 left-0 w-full h-full opacity-0 cursor-pointer"

                  />

                  <label

                    htmlFor="customFile"

                    className="absolute top-0 left-0 w-full h-full flex items-center px-[0.75rem] py-[0.375rem] text-[15px] leading-6 overflow-hidden bg-[#0066ff46] text-headingColor font-semibold rounded-lg truncate cursor-pointer"

                  >

                    Upload Photo

                  </label>

                </div>

              </div>

              <div className="mt-7">

                <button

                  disabled={loading}

                  type="submit"

                  className="w-full bg-primaryColor text-white text-[18px] leading-[30px] rounded-lg px-4 py-3"

                >

                  {loading ? (

                    <SyncLoader size={10} color="#ffffff" />

                  ) : (

                    "Sign Up"

                  )}

                </button>

              </div>

              <p className="mt-5 text-textColor text-center">

                Already have an account?

                <Link

                  to="/login"

                  className="text-primaryColor font-medium ml-1"

                >

                  Login

                </Link>

              </p>

            </form>

          </div>

        </div>

      </div>

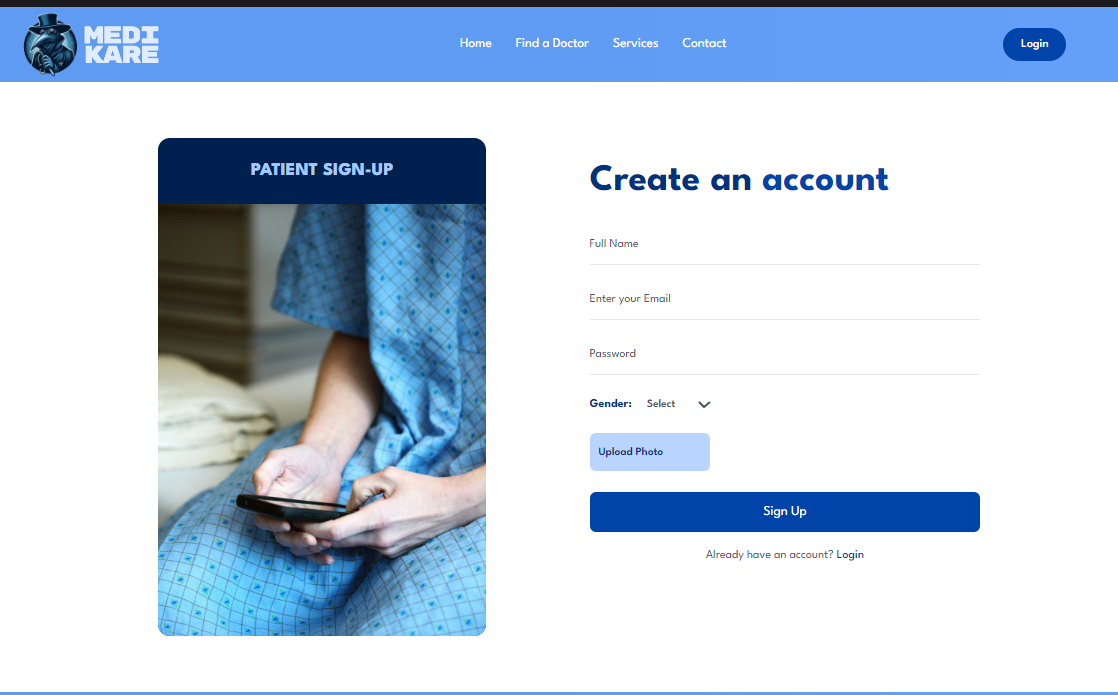
    </section>

  );

};

export default PatientSignup;

**Output:**



**MyAccount.jsx**

import { useContext, useState } from "react";

import { authContext } from "../../context/AuthContext";

// import userImg from "../../assets/images/doctor-img01.png";

import MyBookings from "./MyBookings";

import Profile from "./Profile";

import useGetProfile from "../../hooks/useFetchData";

import { BASE\_URL } from "../../config";

import Loading from "../../components/Loader/Loading";

import Error from "../../components/Error/Error";

const MyAccount = () => {

  const { dispatch } = useContext(authContext);

  const [tab, setTab] = useState("bookings");

  const {

    data: userData,

    loading,

    error,

  } = useGetProfile(`${BASE\_URL}/users/profile/me`);

  const handleLogout = () => {

    dispatch({ type: "LOGOUT" });

  };

  return (

    <section>

      <div className="max-w-[1170px] px-5 mx-auto">

        {loading && <Loading />}

        {error && <Error errMessage={error} />}

        {!loading && !error && (

          <div className="grid md:grid-cols-3 gap-10">

            <div className="pb-[50px] px-[30px] rounded-md">

              <div className="flex items-center justify-center">

                <figure className="w-[100px] h-[100px] rounded-full border-2 border-solid border-primaryColor">

                  <img

                    src={userData.photo}

                    alt=""

                    className="w-full h-full rounded-full"

                  />

                </figure>

              </div>

              <div className="text-center mt-4">

                <h3 className="text-[18px] leading-[30px] text-headingColor font-bold">

                  {userData.name}

                </h3>

                <p className="text-textColor text-[15px] leading-6 font-medium">

                  {userData.email}

                </p>

                <p className="text-textColor text-[15px] leading-6 font-medium">

                  Blood Type:{!userData.bloodType && " Not added"}

                  <span className="ml-2 text-headingColor text-[22px] leading-8">

                    {userData.bloodType}

                  </span>

                </p>

              </div>

              <div className="mt-[50px] md:mt-[100px]">

                <button

                  onClick={handleLogout}

                  className="w-full bg-[#181A1E] text-white p-3 text-[16px] leading-7 rounded-md"

                >

                  Log Out

                </button>

                <button className="w-full bg-red-600 text-white mt-4 p-3 text-[16px] leading-7 rounded-md">

                  Delete Account

                </button>

              </div>

            </div>

            <div className="md:col-span-2 md:px=[30px]">

              <div>

                <button

                  onClick={() => setTab("bookings")}

                  className={`${

                    tab === "bookings" &&

                    "bg-primaryColor text-white font-normal"

                  } p-2 mr-5 px-5 rounded-md text-headingColor font-semibold text-[16px] leading-7 border border-solid border-primaryColor`}

                >

                  My Bookings

                </button>

                <button

                  onClick={() => setTab("settings")}

                  className={`${

                    tab === "settings" &&

                    "bg-primaryColor text-white font-normal"

                  } p-2 px-5 rounded-md text-headingColor font-semibold text-[16px] leading-7 border border-solid border-primaryColor`}

                >

                  Profile Settings

                </button>

              </div>

              {tab == "bookings" && <MyBookings />}

              {tab == "settings" && <Profile user={userData} />}

            </div>

          </div>

        )}

      </div>

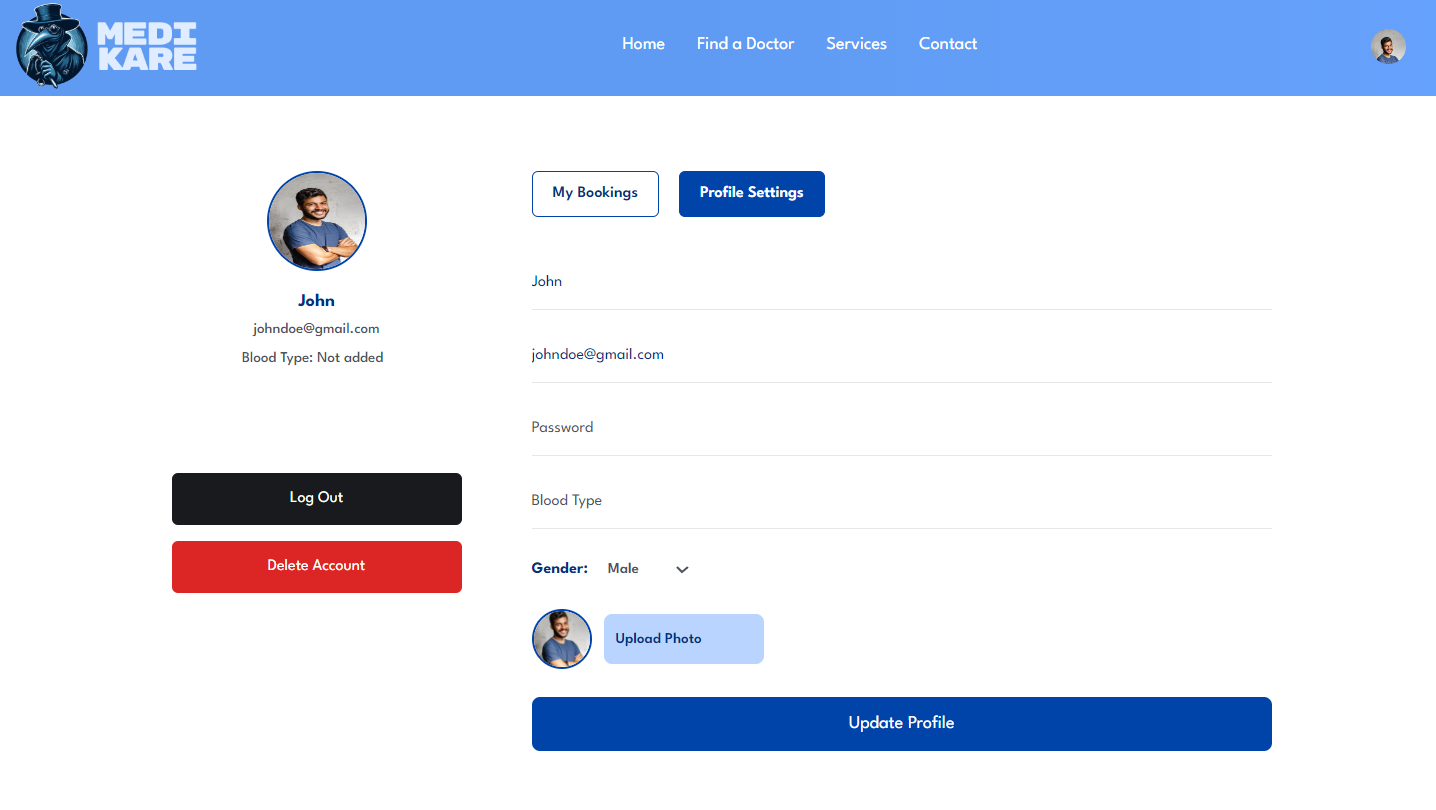
    </section>

  );

};

export default MyAccount;

**Output:**

****

**DoctorSignup.jsx**

import { useState } from "react";

import doctorSignupImg from "../assets/images/doctor-signup.png";

import { Link, useNavigate } from "react-router-dom";

import uploadImageToCloudinary from "../utils/uploadCloudinary";

import { BASE\_URL } from "../config";

import { toast } from "react-toastify";

import SyncLoader from "react-spinners/SyncLoader";

const DoctorSignup = () => {

  const [selectedFile, setSelectedFile] = useState(null);

  const [previewURL, setPreviewURL] = useState("");

  const [loading, setLoading] = useState(false);

  const [formData, setFormData] = useState({

    name: "",

    email: "",

    password: "",

    photo: selectedFile,

    gender: "",

    role: "doctor",

  });

  const navigate = useNavigate();

  const handleInputChange = (e) => {

    setFormData({ ...formData, [e.target.name]: e.target.value });

  };

  const handleFileInputChange = async (event) => {

    const file = event.target.files[0];

    const data = await uploadImageToCloudinary(file);

    setPreviewURL(data.url);

    setSelectedFile(data.url);

    setFormData({ ...formData, photo: data.url });

  };

  const submitHandler = async (event) => {

    event.preventDefault();

    setLoading(true);

    try {

      const res = await fetch(`${BASE\_URL}/auth/register`, {

        method: "post",

        headers: {

          "Content-Type": "application/json",

        },

        body: JSON.stringify(formData),

      });

      const { message } = await res.json();

      if (!res.ok) {

        throw new Error(message);

      }

      setLoading(false);

      toast.success(message);

      navigate("/login");

    } catch (err) {

      toast.error(err.message);

      setLoading(false);

    }

  };

  return (

    <section className="px-5 xl:px-0">

      <div className="max-w-[1170px] mx-auto">

        <div className="grid grid-cols-1 lg:grid-cols-2">

          {/\* IMAGE BOX \*/}

          <div className="hidden lg:block w-fit mx-auto bg-darkerColor rounded-2xl">

            <h2 className="text-center py-7 text-lighterColor uppercase font-extrabold text-2xl">

              Doctor Sign-Up

            </h2>

            <figure className="rounded-b-2xl">

              <img

                src={doctorSignupImg}

                alt=""

                className="w-full rounded-b-2xl"

              />

            </figure>

          </div>

          {/\* SIGN UP FORM \*/}

          <div className="rounded-l-lg lg:pl-16 py-10">

            <h3 className="hidden lg:block text-headingColor text-[50px] leading-9 font-bold mb-10">

              Create an <span className="text-primaryColor">account</span>

            </h3>

            <h3 className="block lg:hidden text-headingColor text-[50px] leading-9 font-bold mb-10">

              Create a <span className="text-primaryColor">Doctor account</span>

            </h3>

            <form onSubmit={submitHandler}>

              <div className="mb-5">

                <input

                  type="text"

                  placeholder="Full Name"

                  name="name"

                  value={formData.name}

                  onChange={handleInputChange}

                  className="w-full pr-4 py-3 border-b border-solid border-[#0066ff61 focus:outline-none focus:border-b-primaryColor text-[16px] leading-7 text-headingColor placeholder:text-textColor cursor-pointer"

                  required

                />

              </div>

              <div className="mb-5">

                <input

                  type="email"

                  placeholder="Enter your Email"

                  name="email"

                  value={formData.email}

                  onChange={handleInputChange}

                  className="w-full pr-4 py-3 border-b border-solid border-[#0066ff61 focus:outline-none focus:border-b-primaryColor text-[16px] leading-7 text-headingColor placeholder:text-textColor cursor-pointer"

                  required

                />

              </div>

              <div className="mb-5">

                <input

                  type="password"

                  placeholder="Password"

                  name="password"

                  value={formData.password}

                  onChange={handleInputChange}

                  className="w-full pr-4 py-3 border-b border-solid border-[#0066ff61 focus:outline-none focus:border-b-primaryColor text-[16px] leading-7 text-headingColor placeholder:text-textColor cursor-pointer"

                  required

                />

              </div>

              <div className="mb-5 flex items-center justify-between">

                <label className="text-headingColor font-bold text-[16px] leading-7">

                  Gender:

                  <select

                    name="gender"

                    value={formData.gender}

                    onChange={handleInputChange}

                    className="text-textColor font-semibold text-[15px] leading-7 px-4 py-3 focus:outline-none"

                    id=""

                  >

                    <option value="">Select</option>

                    <option value="male">Male</option>

                    <option value="female">Female</option>

                    <option value="other">Other</option>

                  </select>

                </label>

              </div>

              <div className="mb-5 flex items-center gap-3">

                {selectedFile && (

                  <figure className="w-[60px] h-[60px] rounded-full border-2 border-solid border-primaryColor flex items-center justify-center overflow-hidden">

                    <img

                      src={previewURL}

                      alt=""

                      className="w-full rounded-full"

                    />

                  </figure>

                )}

                <div className="relative w-[160px] h-[50px]">

                  <input

                    type="file"

                    name="photo"

                    id="customFile"

                    onChange={handleFileInputChange}

                    accept=".jpg,.png"

                    className="absolute top-0 left-0 w-full h-full opacity-0 cursor-pointer"

                  />

                  <label

                    htmlFor="customFile"

                    className="absolute top-0 left-0 w-full h-full flex items-center px-[0.75rem] py-[0.375rem] text-[15px] leading-6 overflow-hidden bg-[#0066ff46] text-headingColor font-semibold rounded-lg truncate cursor-pointer"

                  >

                    Upload Photo

                  </label>

                </div>

              </div>

              <div className="mt-7">

                <button

                  disabled={loading}

                  type="submit"

                  className="w-full bg-primaryColor text-white text-[18px] leading-[30px] rounded-lg px-4 py-3"

                >

                  {loading ? (

                    <SyncLoader size={10} color="#ffffff" />

                  ) : (

                    "Sign Up"

                  )}

                </button>

              </div>

              <p className="mt-5 text-textColor text-center">

                Already have an account?

                <Link

                  to="/login"

                  className="text-primaryColor font-medium ml-1"

                >

                  Login

                </Link>

              </p>

            </form>

          </div>

        </div>

      </div>

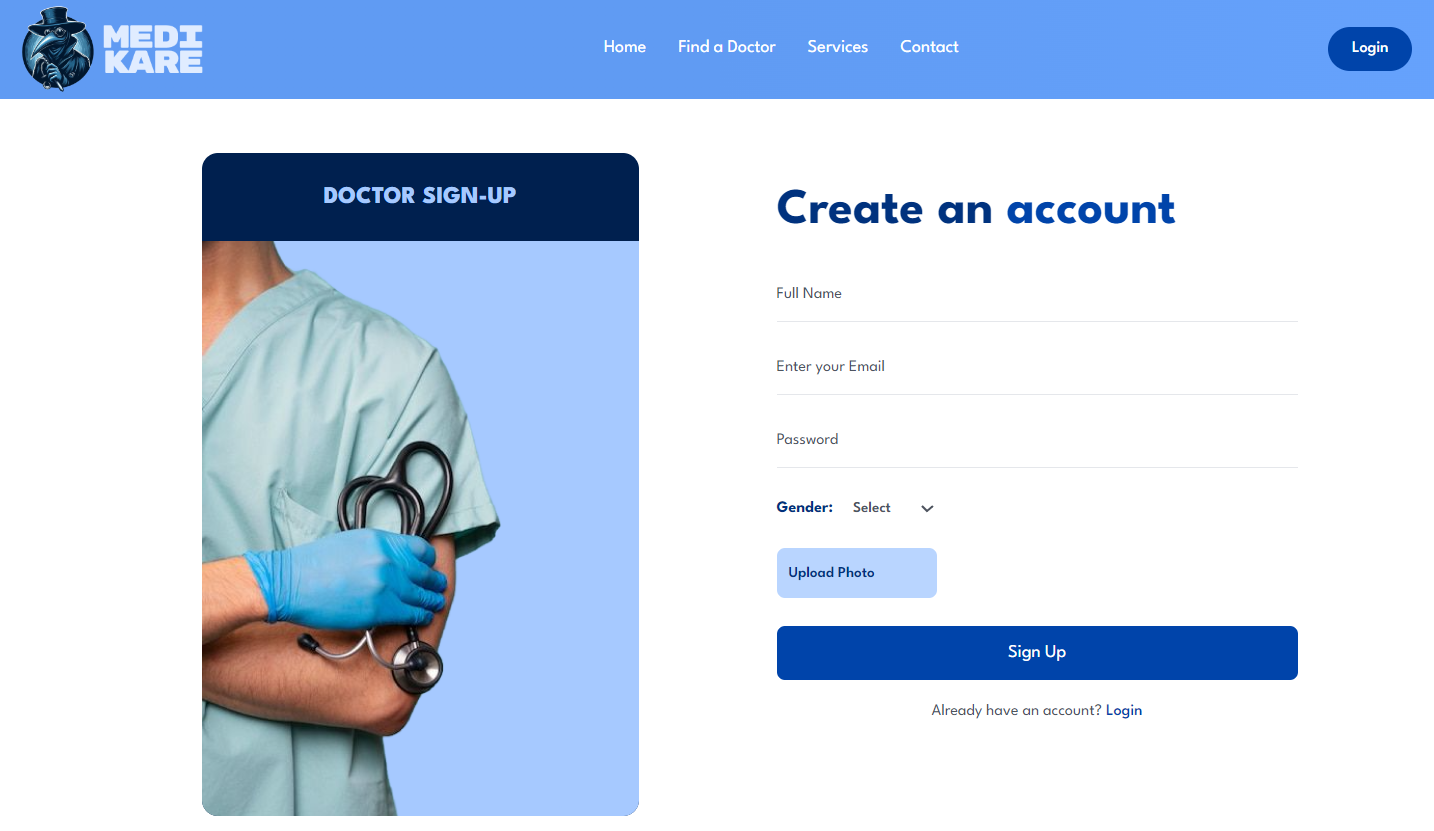
    </section>

  );

};

export default DoctorSignup;

**Output:**



**Src/frontend/Dashboard/doctor-account/Dashboard.jsx**

import Loader from "../../components/Loader/Loading";

import Error from "../../components/Error/Error";

import useGetProfile from "../../hooks/useFetchData";

import { BASE\_URL } from "../../config";

import Tabs from "./Tabs";

import { useState } from "react";

import Profile from "./Profile";

import Appointments from "./Appointments";

import starIcon from "../../assets/images/Star.png";

import DoctorAbout from "../../pages/Doctors/DoctorAbout";

const Dashboard = () => {

  const { data, loading, error } = useGetProfile(

    `${BASE\_URL}/doctors/profile/me`

  );

  const [tab, setTab] = useState("overview");

  return (

    <section>

      <div className="max-w-[1170px] px-5 mx-auto">

        {loading && !error && <Loader />}

        {error && !loading && <Error errMessage={error} />}

        {!loading && !error && (

          <div className="grid lg:grid-cols-3 gap-[30px] lg:gap-[50px]">

            <Tabs tab={tab} setTab={setTab} data={data} />

            <div className="lg:col-span-2">

              {data.isApproved === "pending" && (

                <div className="flex p-4 mb-4 text-yellow-800 bg-yellow-50 rounded-lg">

                  <svg

                    xmlns="http://www.w3.org/2000/svg"

                    x="0px"

                    y="0px"

                    width="100"

                    height="100"

                    viewBox="0 0 50 50"

                    className="flex-shrink-0 w-5 h-5"

                    fill="#854d0e"

                  >

                    <path d="M25,2C12.297,2,2,12.297,2,25s10.297,23,23,23s23-10.297,23-23S37.703,2,25,2z M25,11c1.657,0,3,1.343,3,3s-1.343,3-3,3 s-3-1.343-3-3S23.343,11,25,11z M29,38h-2h-4h-2v-2h2V23h-2v-2h2h4v2v13h2V38z"></path>

                  </svg>

                  <span className="sr-only">Info</span>

                  <div className="ml-3 text-sm font-medium">

                    To get approval please complete your profile. It will be

                    reviewed manually and approved within 3 days.

                  </div>

                </div>

              )}

              <div className="mt-8">

                {tab == "overview" && (

                  <div>

                    <div className="flex items-center gap-4 mb-10">

                      <figure className="max-w-[200px] max-h-[200px]">

                        {data?.photo ? (

                          <img src={data?.photo} alt="" className="w-full" />

                        ) : (

                          <h2 className="text-textColor">No Image added</h2>

                        )}

                      </figure>

                      <div>

                        <span className="bg-[#CCF0F3] text-irisBlueColor py-1 px-4 lg:py-2 lg:px-6 rounded text-[12px] leading-4 lg:text-[16px] lg:leading-6 font-semibold">

                          {data.specialization

                            ? data.specialization

                            : "No Specialization added"}

                        </span>

                        <h3 className="text-[22px] leading-9 font-bold text-headingColor mt-3">

                          {data.name}

                        </h3>

                        <div className="flex items-center gap-[6px]">

                          <span className="flex items-center gap-[6px] text-textColor text-[14px] leading-5 lg:text-[16px] lg:leading-6 font-semibold ">

                            <img src={starIcon} alt="" />

                            {data.averageRating}

                          </span>

                          <span className="text-textColor text-[14px] leading-5 lg:text-[16px] lg:leading-6 font-semibold">

                            ({data.totalRating})

                          </span>

                        </div>

                        {data?.bio ? (

                          <p className="text\_\_para font-[15px] lg:max-w-[390px] leading-6">

                            {data?.bio}

                          </p>

                        ) : (

                          <h2 className="text-textColor">No Bio added</h2>

                        )}

                      </div>

                    </div>

                    <DoctorAbout

                      name={data.name}

                      about={data.about}

                      qualifications={data.qualifications}

                      experiences={data.experiences}

                    />

                  </div>

                )}

                {tab == "appointments" && (

                  <Appointments appointments={data.appointments} />

                )}

                {tab == "settings" && <Profile doctorData={data} />}

              </div>

            </div>

          </div>

        )}

      </div>

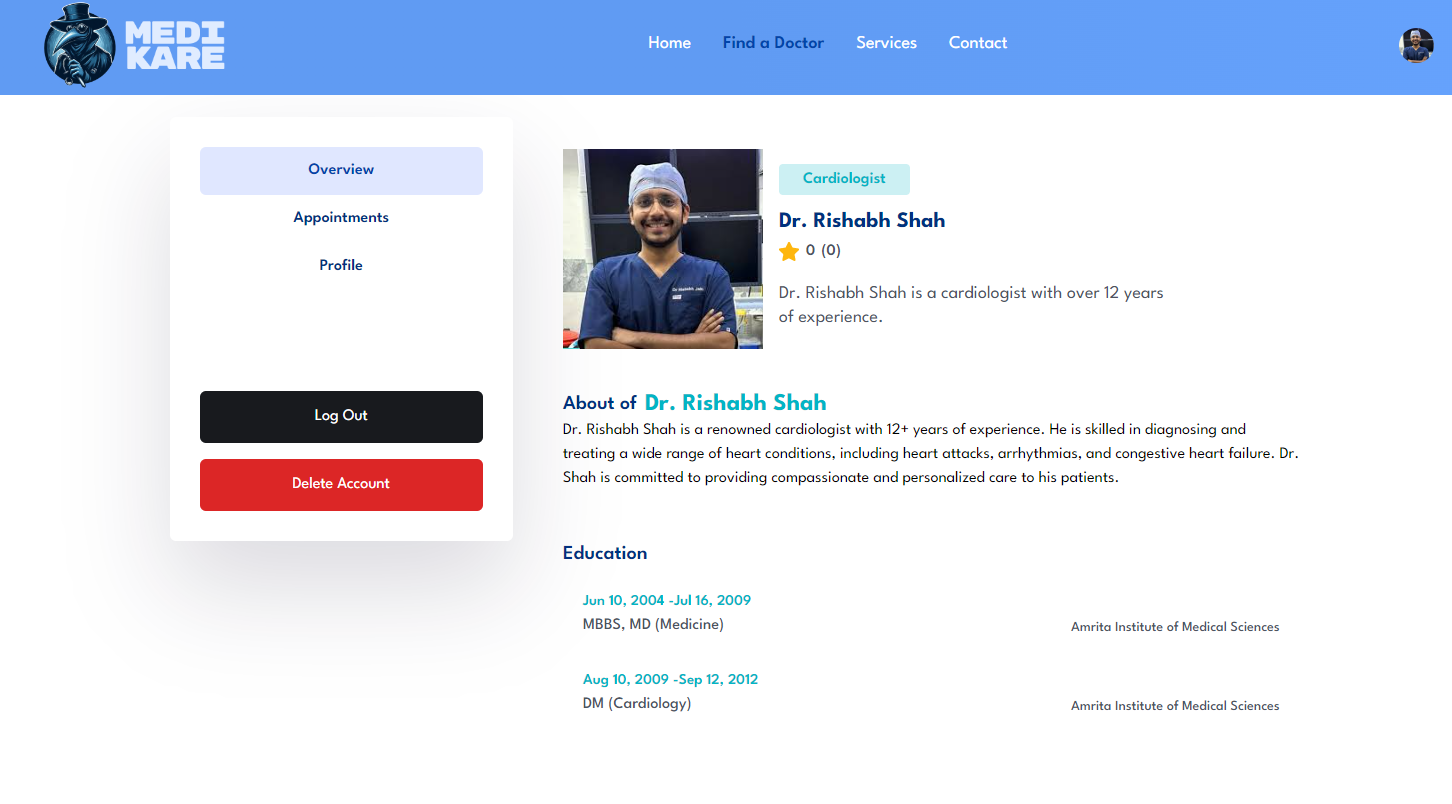
    </section>

  );

};

export default Dashboard;

**Output:**



**AdminDashboard.jsx**

import { useState } from "react";

// import userImg from "../../assets/images/doctor-img01.png";

import useGetProfile from "../../hooks/useFetchData";

import { BASE\_URL } from "../../config";

import Loading from "../../components/Loader/Loading";

import Error from "../../components/Error/Error";

import PatientsData from "./PatientsData";

import ServicesData from "./Services/ServicesData";

import DoctorsData from "./Doctors/DoctorsData";

import FAQData from "./Faq/FAQData";

import { useContext } from "react";

import { BiMenu } from "react-icons/bi";

import { authContext } from "../../context/AuthContext";

import { useNavigate } from "react-router-dom";

import MessageData from "./Messages/MessageData";

const AdminDashboard = () => {

  const [tab, setTab] = useState("doctors");

  const { dispatch } = useContext(authContext);

  const navigate = useNavigate();

  const {

    data: userData,

    loading,

    error,

  } = useGetProfile(`${BASE\_URL}/users/profile/me`);

  const handleLogout = () => {

    dispatch({ type: "LOGOUT" });

    navigate("/");

  };

  return (

    <section>

      <div className="max-w-[1400px] px-5 mx-auto">

        {loading && <Loading />}

        {error && <Error errMessage={error} />}

        {!loading && !error && (

          <div className="grid lg:grid-cols-3 gap-10">

            <div className="pb-[50px] px-[30px] rounded-md">

              <div>

                <span className="lg:hidden">

                  <BiMenu className="w-6 h-6 cursor-pointer" />

                  <div className="text-center">

                    <h3 className="text-[18px] leading-[30px] text-headingColor font-bold">

                      {userData.name}

                    </h3>

                    <p className="text-textColor text-[15px] leading-6 font-medium">

                      {userData.email}

                    </p>

                  </div>

                </span>

                <div className="hidden lg:flex flex-col p-[30px] bg-white shadow-panelShadow items-center h-max rounded-md">

                  <div className="text-center my-4">

                    <h3 className="text-[18px] leading-[30px] text-headingColor font-bold">

                      {userData.name}

                    </h3>

                    <p className="text-textColor text-[15px] leading-6 font-medium">

                      {userData.email}

                    </p>

                  </div>

                  <button

                    onClick={() => setTab("doctors")}

                    className={`${

                      tab === "doctors"

                        ? "bg-indigo-100 text-primaryColor"

                        : "bg-transparent text-headingColor"

                    } w-full btn mt-0 rounded-md`}

                  >

                    Doctors

                  </button>

                  <button

                    onClick={() => setTab("patients")}

                    className={`${

                      tab === "patients"

                        ? "bg-indigo-100 text-primaryColor"

                        : "bg-transparent text-headingColor"

                    } w-full btn mt-0 rounded-md`}

                  >

                    Patients

                  </button>

                  <button

                    onClick={() => setTab("services")}

                    className={`${

                      tab === "services"

                        ? "bg-indigo-100 text-primaryColor"

                        : "bg-transparent text-headingColor"

                    } w-full btn mt-0 rounded-md`}

                  >

                    Services

                  </button>

                  <button

                    onClick={() => setTab("faq")}

                    className={`${

                      tab === "faq"

                        ? "bg-indigo-100 text-primaryColor"

                        : "bg-transparent text-headingColor"

                    } w-full btn mt-0 rounded-md`}

                  >

                    FAQ

                  </button>

                  <button

                    onClick={() => setTab("messages")}

                    className={`${

                      tab === "messages"

                        ? "bg-indigo-100 text-primaryColor"

                        : "bg-transparent text-headingColor"

                    } w-full btn mt-0 rounded-md`}

                  >

                    Messages

                  </button>

                  <div className="mt-[100px] w-full">

                    <button

                      onClick={handleLogout}

                      className="w-full bg-[#181A1E] text-white p-3 text-[16px] leading-7 rounded-md"

                    >

                      Log Out

                    </button>

                  </div>

                </div>

              </div>

              {/\* <Tabs tab={tab} setTab={setTab} /> \*/}

            </div>

            <div className="lg:col-span-2 md:px=[30px]">

              {tab == "doctors" && <DoctorsData />}

              {tab == "patients" && <PatientsData />}

              {tab == "services" && <ServicesData />}

              {tab == "faq" && <FAQData />}

              {tab == "messages" && <MessageData />}

            </div>

          </div>

        )}

      </div>

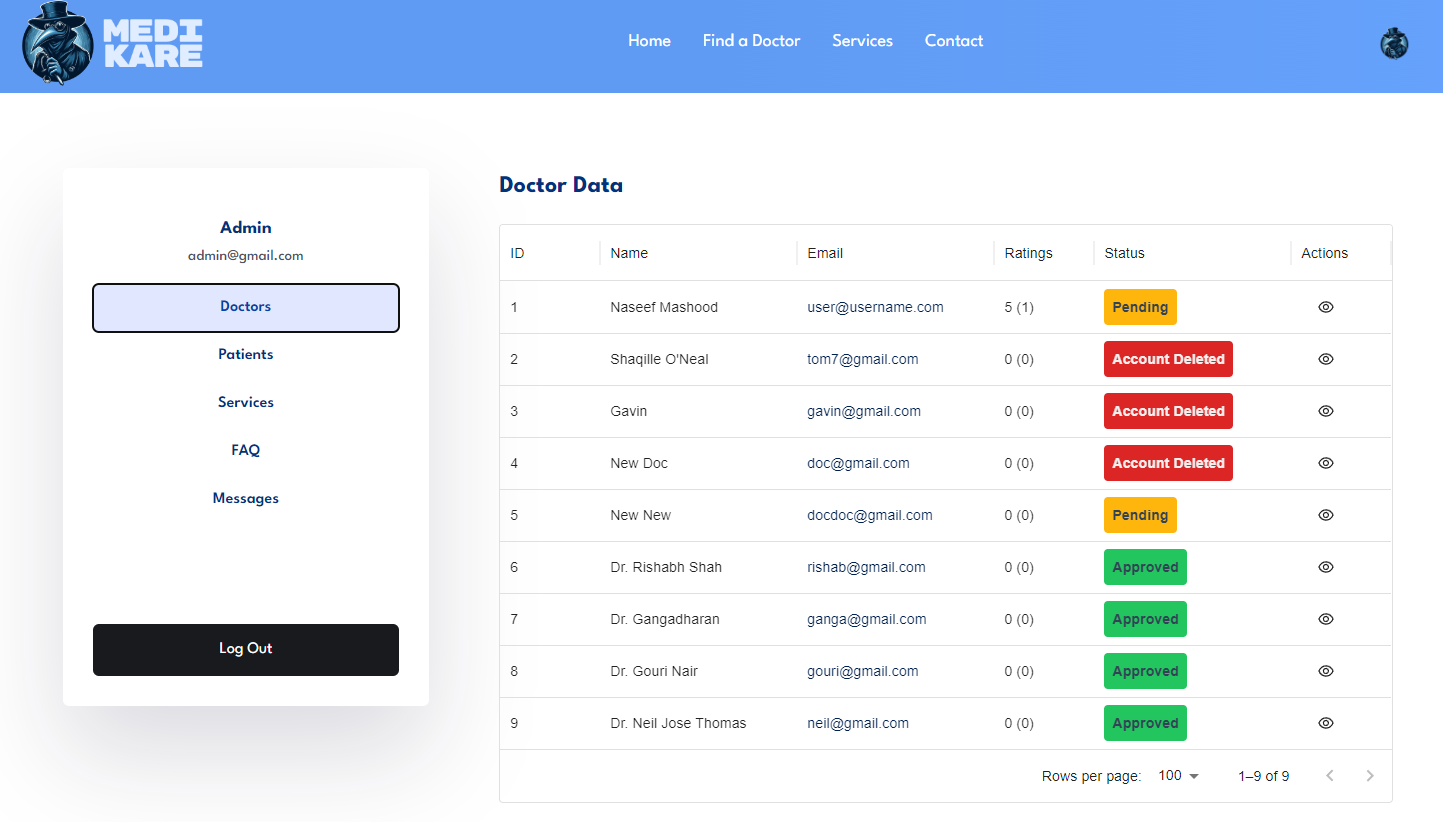
    </section>

  );

};

export default AdminDashboard;

**Output:**



**Header.jsx**

import { useRef, useContext } from "react";

import logo from "../../assets/images/logo-1.png";

import mobileLogo from "../../assets/images/mobile-logo.png";

import logoText from "../../assets/images/medikare.png";

import profile from "../../assets/images/admin-profile.png";

import { NavLink, Link } from "react-router-dom";

import { BiMenu } from "react-icons/bi";

import { authContext } from "../../context/AuthContext";

const navLinks = [

  {

    path: "/home",

    display: "Home",

  },

  {

    path: "/doctors",

    display: "Find a Doctor",

  },

  {

    path: "/services",

    display: "Services",

  },

  {

    path: "/contact",

    display: "Contact",

  },

];

const Header = () => {

  const headerRef = useRef(null);

  const menuRef = useRef(null);

  const { user, role, token } = useContext(authContext);

  const toggleMenu = () => menuRef.current.classList.toggle("show\_\_menu");

  return (

    <header className="header flex items-center sticky\_\_header" ref={headerRef}>

      <div className="container">

        <div className="flex items-center justify-between">

          {/\* LOGO \*/}

          <NavLink to="/home">

            <div className="flex items-center">

              <img src={logo} alt="" className="w-[90px] hidden sm:block" />

              <img

                src={logoText}

                alt=""

                className="w-[100px] hidden sm:block"

              />

              <img

                src={mobileLogo}

                alt=""

                className="w-[90px] block sm:hidden"

              />

            </div>

          </NavLink>

          {/\* MENU \*/}

          <div className="navigation" ref={menuRef} onClick={toggleMenu}>

            <ul className="menu flex items-center gap-[2rem] bg-gradient-to-t from-[#609BF3] from-50% to-[#B7D4FF] md:bg-none">

              {/\* LOGO \*/}

              <NavLink to="/home">

                <div className="flex items-center">

                  <img src={logo} alt="" className="w-[100px] md:hidden" />

                  <img src={logoText} alt="" className="w-[100px] md:hidden" />

                </div>

              </NavLink>

              {navLinks.map((link, index) => (

                <li key={index}>

                  <NavLink

                    to={link.path}

                    className={(navClass) =>

                      navClass.isActive

                        ? "text-primaryColor text-[18px] leading-7 font-[600]"

                        : "text-whiteColor text-[18px] leading-7 hover:text-primaryColor"

                    }

                  >

                    {link.display}

                  </NavLink>

                </li>

              ))}

            </ul>

          </div>

          {/\* NAV RIGHT \*/}

          <div className="flex items-center gap-4">

            {token && user ? (

              <div>

                <Link

                  to={`${

                    role === "doctor"

                      ? "/doctors/profile/me"

                      : role === "admin"

                      ? "/admin"

                      : "/users/profile/me"

                  }`}

                >

                  <figure className="w-[35px] h-[35px] overflow-hidden rounded-full cursor-pointer">

                    <img

                      src={

                        role === "admin"

                          ? logo

                          : user?.photo

                          ? user?.photo

                          : profile

                      }

                      className="w-full rounded-full"

                      alt=""

                    />

                  </figure>

                </Link>

              </div>

            ) : (

              <Link to="/login">

                <button className="bg-primaryColor py-2 px-6 text-white text-[16px] font-[500] h-[44px] flex items-center justify-center rounded-[50px]">

                  Login

                </button>

              </Link>

            )}

            <span className="md:hidden" onClick={toggleMenu}>

              <BiMenu className="w-6 h-6 cursor-pointer" />

            </span>

          </div>

        </div>

      </div>

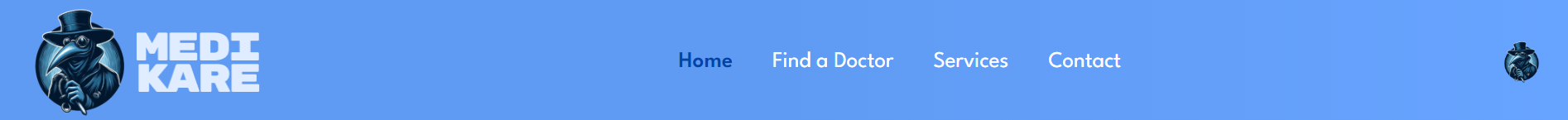
    </header>

  );

};

export default Header;

**Output:**



**Footer.jsx**

import { Link } from "react-router-dom";

import logo from "../../assets/images/logo-1.png";

import logotext from "../../assets/images/medikare.png";

import { RiLinkedinFill } from "react-icons/ri";

import {

  AiFillYoutube,

  AiFillGithub,

  AiOutlineInstagram,

} from "react-icons/ai";

const socialLinks = [

  {

    path: "https://youtu.be/ZQdQIIF1DOY?si=nOdp14P-S1J2GDTq",

    icon: <AiFillYoutube className="group-hover: text-black w-4 h-5" />,

  },

  {

    path: "https://github.com/codingwithmuhib",

    icon: <AiFillGithub className="group-hover: text-black w-4 h-5" />,

  },

  {

    path: "https://www.instagram.com/muhib160.official/",

    icon: <AiOutlineInstagram className="group-hover: text-black w-4 h-5" />,

  },

  {

    path: "https://www.linkedin.com/in/codingwithmuhib/",

    icon: <RiLinkedinFill className="group-hover: text-black w-4 h-5" />,

  },

];

const quickLinks01 = [

  {

    path: "/home",

    display: "Home",

  },

  {

    path: "/",

    display: "About Us",

  },

  {

    path: "/services",

    display: "Services",

  },

  {

    path: "/",

    display: "Blog",

  },

  {

    path: "/contact",

    display: "Contact Us",

  },

];

const quickLinks02 = [

  {

    path: "/find-a-doctor",

    display: "Find a Doctor",

  },

  {

    path: "/",

    display: "Request an Appointment",

  },

  {

    path: "/",

    display: "Find a Location",

  },

  {

    path: "/",

    display: "Get an Opinion",

  },

  {

    path: "/",

    display: "Donate",

  },

];

const Footer = () => {

  const year = new Date().getFullYear();

  return (

    <footer className="footer pb-16 pt-10">

      <div className="container">

        <div className="flex justify-evenly items-center text-center flex-col md:flex-row md:text-left flex-wrap gap-[30px]">

          <div>

            <div className="flex flex-col items-center md:block">

              <img src={logo} alt="" width="100px" />

              <img src={logotext} alt="" width="100px" />

            </div>

            <div className="flex justify-center items-center gap-3 mt-4 md:justify-normal">

              {socialLinks.map((link, index) => (

                <Link

                  to={link.path}

                  key={index}

                  className="w-9 h-9  rounded-full flex items-center justify-center group hover:bg-iconColor hover:border-none"

                >

                  {link.icon}

                </Link>

              ))}

            </div>

            <p className="text-[14px] leading-5 font-[400] text-black mt-4">

              Copyright © {year} <br></br>developed by 3Bloke Technologies all rights

              reserved

            </p>

          </div>

          <div>

            <h2 className="text-[20px] leading-[30px] font-[700] mb-6 text-headingColor">

              Quick Links

            </h2>

            <ul>

              {quickLinks01.map((item, index) => (

                <li key={index} className="mb-4">

                  <Link

                    to={item.path}

                    className="text-[16px] leading-7 font-[400] text-whiteColor hover:text-primaryColor"

                  >

                    {item.display}

                  </Link>

                </li>

              ))}

            </ul>

          </div>

          <div>

            <h2 className="text-[20px] leading-[30px] font-[700] mb-6 text-headingColor">

              Want to:

            </h2>

            <ul>

              {quickLinks02.map((item, index) => (

                <li key={index} className="mb-4">

                  <Link

                    to={item.path}

                    className="text-[16px] leading-7 font-[400] text-whiteColor hover:text-primaryColor” >

                    {item.display}

                  </Link>

                </li>

              ))}

            </ul>

          </div>

        </div>

      </div>

    </footer>

  );

};

export default Footer;

**Output:**



**Back-End**

**Index.js**

import express from "express";

import cookieParser from "cookie-parser";

import cors from "cors";

import mongoose from "mongoose";

import dotenv from "dotenv";

import authRoute from "./Routes/auth.js";

import userRoute from "./Routes/user.js";

import doctorRoute from "./Routes/doctor.js";

import reviewRoute from "./Routes/review.js";

import faqRoute from "./Routes/faq.js";

import serviceRoute from "./Routes/services.js";

import bookingRoute from "./Routes/booking.js";

import messageRoute from "./Routes/messages.js";

dotenv.config();

const app = express();

const port = process.env.PORT || 8000;

const corsOptions = {

  origin: true,

};

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

  res.send("Api is working");

});

// DataBase Connection

mongoose.set("strictQuery", false);

const connectDB = async () => {

  try {

    await mongoose.connect(process.env.MONGO\_URL, {

      useNewUrlParser: true,

      useUnifiedTopology: true,

    });

    console.log("MongoDB database is connected");

  } catch (err) {

    console.log("MongoDB database connection failed");

  }

};

// Middleware

app.use(express.json());

app.use(cookieParser());

app.use(cors(corsOptions));

app.use("/api/v1/auth", authRoute);

app.use("/api/v1/users", userRoute);

app.use("/api/v1/doctors", doctorRoute);

app.use("/api/v1/reviews", reviewRoute);

app.use("/api/v1/services", serviceRoute);

app.use("/api/v1/faq", faqRoute);

app.use("/api/v1/bookings", bookingRoute);

app.use("/api/v1/message", messageRoute);

app.listen(port, () => {

  connectDB();

  console.log("Server is running on port " + port);

});

**authController.js**

import User from "../models/UserSchema.js";

import Doctor from "../models/DoctorSchema.js";

import jwt from "jsonwebtoken";

import bcrypt from "bcryptjs";

const generateToken = (user) => {

  return jwt.sign(

    { id: user.\_id, role: user.role },

    process.env.JWT\_SECRET\_KEY,

    {

      expiresIn: "15d",

    }

  );

};

export const register = async (req, res) => {

  const { name, email, password, role, gender, photo } = req.body;

  try {

    let user = null;

    if (role === "patient") {

      user = await User.findOne({ email });

    } else if (role === "doctor") {

      user = await Doctor.findOne({ email });

    }

    // Check if existing user

    if (user) {

      return res.status(400).json({ message: "User already exists." });

    }

    // Hash password

    const salt = await bcrypt.genSalt(10);

    const hashPassword = await bcrypt.hash(password, salt);

    if (role === "patient") {

      user = new User({

        name,

        email,

        password: hashPassword,

        role,

        gender,

        photo,

      });

    }

    if (role === "doctor") {

      user = new Doctor({

        name,

        email,

        password: hashPassword,

        role,

        gender,

        photo,

      });

    }

    await user.save();

    res

      .status(200)

      .json({ success: true, message: "User created successfully" });

  } catch (err) {

    res

      .status(500)

      .json({ success: false, message: "Internal server error, try again" });

  }

};

export const login = async (req, res) => {

  const { email } = req.body;

  try {

    let user = null;

    const patient = await User.findOne({ email });

    const doctor = await Doctor.findOne({ email, deletedAt: null });

    if (patient) {

      user = patient;

    }

    if (doctor) {

      user = doctor;

    }

    // Check if user exists

    if (!user) {

      return res.status(404).json({ message: "User not found" });    }

    // Compare password

    const isPasswordMatch = await bcrypt.compare(

      req.body.password,

      user.password

    );

    if (!isPasswordMatch) {

      return res

        .status(400)

        .json({ status: false, message: "Invalid credentials" });

    }

    // get token

    const token = generateToken(user);

    const { password, role, appointments, ...rest } = user.\_doc;

    res.status(200).json({

      status: true,

      message: "Logged in Successfully",

      token,

      data: { ...rest },

      role,

    });

  } catch (err) {

    res

      .status(500)

      .json({ status: false, message: "Login failed. Try again." });

  }

};

**doctorController.js**

import Booking from "../models/BookingSchema.js";

import Doctor from "../models/DoctorSchema.js";

export const updateDoctor = async (req, res) => {

  const id = req.params.id;

  try {

    const updatedDoctor = await Doctor.findByIdAndUpdate(

      id,

      { $set: req.body },

      { new: true }

    ).select("-password");

    res.status(200).json({

      success: true,

      message: "Doctor updated successfully",

      data: updatedDoctor,

    });

  } catch (err) {

    res

      .status(500)

      .json({ success: false, message: "Failed to update doctor" });

  }

};

export const deleteDoctor = async (req, res) => {

  const id = req.params.id;

  try {

    const deletedDoctor = await Doctor.findByIdAndUpdate(

      id,

      { $set: { deletedAt: new Date() } },

      { new: true }

    ).select("-password");

    res.status(200).json({

      success: true,

      message: "Doctor deleted successfully",

      data: deleteDoctor,

    });

  } catch (err) {

    res

      .status(500)

      .json({ success: false, message: "Failed to delete doctor" });

  }

};

export const getSingleDoctor = async (req, res) => {

  const id = req.params.id;

  try {

    const doctor = await Doctor.findById(id)

      .populate("reviews")

      .select("-password");

    res.status(200).json({

      success: true,

      message: "Doctor found",

      data: doctor,

    });

  } catch (err) {

    res.status(404).json({ success: false, message: "No doctor found" });

  }

};

export const getAllDoctors = async (req, res) => {

  try {

    const { query } = req.query;

    let doctors;

    if (query) {

      doctors = await Doctor.find({

        isApproved: "approved",

        deletedAt: null,

        $or: [

          { name: { $regex: query, $options: "i" } },

          { specialization: { $regex: query, $options: "i" } },

        ],

      })

        .select("-password")

        .sort({ averageRating: -1 });

    } else {

      doctors = await Doctor.find({ isApproved: "approved", deletedAt: null })

        .select("-password")

        .sort({ averageRating: -1 });

    }

    res.status(200).json({

      success: true,

      message: "Doctors found",

      data: doctors,

    });

  } catch (err) {

    res.status(404).json({ success: false, message: "Not found" });

  }

};

export const getDoctorProfile = async (req, res) => {

  const doctorId = req.userId;

  try {

    const doctor = await Doctor.findById(doctorId);

    if (!doctor) {

      return res

        .status(404)

        .json({ success: false, message: `doctor not found` });

    }

    const { password, ...rest } = doctor.\_doc;

    const appointments = await Booking.find({ doctor: doctorId });

    res.status(200).json({

      success: true,

      message: "Profile info successfully retrieved",

      data: { ...rest, appointments },

    });

  } catch (err) {

    res.status(500).json({ success: false, message: "Something went wrong" });

  }

};

export const getDoctorsList = async (req, res) => {

  try {

    const { query } = req.query;

    let doctors;

    doctors = await Doctor.find().select("-password");

    res.status(200).json({

      success: true,

      message: "Doctors found",

      data: doctors,

    });

  } catch (err) {

    res.status(404).json({ success: false, message: "Not found" });

  }

};

export const approveDoctor = async (req, res) => {

  const id = req.params.id;

  try {

    const approvedDoctor = await Doctor.findByIdAndUpdate(

      id,

      { isApproved: "approved" },

      { new: true }

    ).select("-password");

    res.status(200).json({

      success: true,

      message: "Doctor approved successfully",

      data: approvedDoctor,

    });

  } catch (err) {

    res

      .status(500)

      .json({ success: false, message: "Failed to approve doctor" });

  }

};

**faqController.js**

import Faq from "../models/FaqSchema.js";

export const addFaq = async (req, res) => {

  const { question, content } = req.body;

  try {

    const faq = new Faq({

      question,

      content,

    });

    await faq.save();

    res.status(200).json({ success: true, message: "FAQ added successfully" });

  } catch (err) {

    res

      .status(500)

      .json({ success: false, message: "Internal server error, try again" });

  }

};

export const deleteFaq = async (req, res) => {

  const id = req.params.id;

  try {

    const faq = await Faq.findByIdAndDelete(id);

    if (!faq) {

      return res.status(404).json({

        success: false,

        message: "FAQ not found",

      });

    }

    res.status(200).json({

      success: true,

      message: "FAQ deleted successfully",

    });

  } catch (err) {

    res.status(500).json({ success: false, message: "Failed to delete FAQ" });

  }

};

export const getFaqs = async (req, res) => {

  try {

    const faq = await Faq.find();

    res.status(200).json({

      success: true,

      message: "FAQ found",

      data: faq,

    });

  } catch (error) {

    res.status(404).json({ success: false, message: "Not found" });

  }

};

**messageController.js**

import Messages from "../models/MessageSchema.js";

export const createMessage = async (req, res) => {

  const newMessage = new Messages(req.body);

  try {

    const savedMessage = await newMessage.save();

    res.status(200).json({

      success: true,

      message: "Message submitted",

      data: savedMessage,

    });

  } catch (err) {

    res.status(500).json({ success: false, message: err.message });

  }

};

export const getAllMessages = async (req, res) => {

  try {

    const messages = await Messages.find({});

    res

      .status(200)

      .json({ success: true, message: "Successful", data: messages });

  } catch (err) {

    res.status(404).json({ success: false, message: "Not found" });

  }

};

export const getSingleMessage = async (req, res) => {

  const id = req.params.id;

  try {

    const messages = await Messages.findById(id);

    res

      .status(200)

      .json({ success: true, message: "Successful", data: messages });

  } catch (err) {

    res.status(404).json({ success: false, message: "Not found" });

  }

};

**reviewController.js**

import Review from "../models/ReviewSchema.js";

import Doctor from "../models/DoctorSchema.js";

export const getAllReviews = async (req, res) => {

  try {

    const reviews = await Review.find({});

    res

      .status(200)

      .json({ success: true, message: "Successful", data: reviews });

  } catch (err) {

    res.status(404).json({ success: false, message: "Not found" });

  }

};

export const createReview = async (req, res) => {

  if (!req.body.doctor) req.body.doctor = req.params.doctorId;

  if (!req.body.user) req.body.user = req.userId;

  const newReview = new Review(req.body);

  try {

    const savedReview = await newReview.save();

    await Doctor.findByIdAndUpdate(req.body.doctor, {

      $push: { reviews: savedReview.\_id },

    });

    res

      .status(200)

      .json({ success: true, message: "Review submitted", data: savedReview });

  } catch (err) {

    res.status(500).json({ success: false, message: err.message });

  }

};

**servicesController.js**

import Services from "../models/ServicesSchema.js";

export const addService = async (req, res) => {

  const { title, description } = req.body;

  try {

    const service = new Services({

      title,

      description,

    });

    await service.save();

    res

      .status(200)

      .json({ success: true, message: "Service added successfully" });

  } catch (err) {

    res

      .status(500)

      .json({ success: false, message: "Internal server error, try again" });

  }

};

export const deleteService = async (req, res) => {

  const id = req.params.id;

  try {

    const service = await Services.findByIdAndDelete(id);

    if (!service) {

      return res.status(404).json({

        success: false,

        message: "Service not found",

      });

    }

    res.status(200).json({

      success: true,

      message: "Service deleted successfully",

    });

  } catch (err) {

    res

      .status(500)

      .json({ success: false, message: "Failed to delete service" });

  }

};

export const getServices = async (req, res) => {

  try {

    const services = await Services.find();

    res.status(200).json({

      success: true,

      message: "Services found",

      data: services,

    });

  } catch (error) {

    res.status(404).json({ success: false, message: "Not found" });

  }

};

**userController.js**

import User from "../models/UserSchema.js";

import Booking from "../models/BookingSchema.js";

import Doctor from "../models/DoctorSchema.js";

export const updateUser = async (req, res) => {

  const id = req.params.id;

  try { const updatedUser = await User.findByIdAndUpdate(

      id,

      { $set: req.body },

      { new: true }

    ).select("-password");

    res.status(200).json({

      success: true,

      message: "User updated successfully",

      data: updatedUser,

    });

  } catch (err) {

    res.status(500).json({ success: false, message: "Failed to update user" });

  }

};

export const deleteUser = async (req, res) => {

  const id = req.params.id;

  try {

    await User.findByIdAndDelete(id);

    res.status(200).json({

      success: true,

      message: "User deleted successfully",

    });

  } catch (err) {

    res.status(500).json({ success: false, message: "Failed to delete user" });

  }

};

export const getSingleUser = async (req, res) => {

  const id = req.params.id;

  try {

    const user = await User.findById(id).select("-password");

    res.status(200).json({

      success: true,

      message: "User found",

      data: user,

    });

  } catch (err) {

    res.status(404).json({ success: false, message: "No user found" });

  }

};

export const getAllUsers = async (req, res) => {

  try {

    const users = await User.find({ role: "patient" }).select("-password");

    res.status(200).json({

      success: true,

      message: "Users found",

      data: users,

    });

  } catch (err) {

    res.status(404).json({ success: false, message: "Not found" });

  }

};

export const getUserProfile = async (req, res) => {

  const userId = req.userId;

  try {

    const user = await User.findById(userId);

    if (!user) {

      return res

        .status(404)

        .json({ success: false, message: "User not found" });

    }

    const { password, ...rest } = user.\_doc;

    res.status(200).json({

      success: true,

      message: "Profile info successfully retrieved",

      data: { ...rest },

    });

  } catch (err) {

    res.status(500).json({ success: false, message: "Something went wrong" });

  }

};

export const getMyAppointments = async (req, res) => {

  try {

    // Retrieve appointments for specific user

    const bookings = await Booking.find({ user: req.userId });

    // Extract doctor IDs from appointments

    const doctorIds = bookings.map((e) => e.doctor.toString());

    // Retrieve doctors from IDs

    const doctors = await Doctor.find({ \_id: { $in: doctorIds } }).select(

      "-password"

    );

    console.log(doctorIds);

    res.status(200).json({

      success: true,

      message: "Appointments retrieved successfully",

      data: doctors,

    });

  } catch (err) {

    res.status(500).json({ success: false, message: "Something went wrong" });

  }

};