Landing Page:

```
import React from "react";
import "../styling/landingPage.css";
import TestimonialsCarousel from "./carousel";
import Login from "./login";
import { useNavigate, useLocation } from "react-router-dom";
import { useState } from "react";
import appointment from "../assets/appointment.png";
import orderMed from "../assets/order medicines.png";
import docVisit from "../assets/docVisit.png";

const MedicalLandingPage = () => {
  const location = useLocation();
  const { userData } = location.state || {};
  const currentHour = new Date().getHours();
  let greeting;

  if (currentHour < 12) {
    greeting = "Good Morning";
  } else if (currentHour < 18) {
    greeting = "Good Afternoon";
  } else {
    greeting = "Good Evening";
  }

  return (
    <div style={{ display: "flex", flexDirection: "column" }}>
      <Header userData={userData} />
      <MainContent userData={userData} greeting={greeting} />
      <Footer />
    </div>
  );
};

const Header = ({ userData }) => {
  const [activeLink, setActiveLink] = useState("services");
  const navigate = useNavigate();

  const handleLinkClick = (link) => {
```

```
  setActiveLink(link);
};

const handleNavigation = () => {
  navigate("/login");
};

return (
  <header
    className="header"
    style={{
      display: "flex",
      justifyContent: "space-between",
      alignItems: "center",
    }}
  >
    <div>
      <h1>eMed</h1>
    </div>
    <div>
      <nav>
        <ul className="nav-links">
          <li>
            <a
              // href="#services"
              className={activeLink === "services" ? "active" : ""}
              onClick={() => handleLinkClick("services")}
            >
              Services
            </a>
          </li>
          <li>
            <a
              // href="#testimonials"
              className={activeLink === "testimonials" ? "active" : ""}
              onClick={() => handleLinkClick("testimonials")}
            >
              Testimonials
            </a>
          </li>
```

```
            <li>
              <a
                // href="#appointment"
                className={activeLink === "appointment" ? "active" : ""}
                onClick={() => handleLinkClick("appointment")}
              >
                Make Appointment
              </a>
            </li>
            <li>
              <a
                // href="#buy-medicine"
                className={activeLink === "buy-medicine" ? "active" : ""}
                onClick={() => handleLinkClick("buy-medicine")}
              >
                Buy Medicine
              </a>
            </li>
          </ul>
        </nav>
      </div>
      <div>
        <button onClick={handleNavigation} className="loginBut">
          {userData?.name ? "Logout" : "Login/Register"}
        </button>
      </div>
    </header>
  );
};

const MainContent = ({ userData, greeting }) => {
  const navigate = useNavigate();

  const handleBuyNowButton = () => {
    navigate("/medicines", { state: { userData } });
  };

  const handleBookNow = () => {
    navigate("/docAppointment", { state: { userData } });
  };
  return (
    <main>
      <section className="hero" style={{ display: "flex", gap: "20px" }}>
        <div
          style={{
            display: "flex",
            flexDirection: "column",
            justifyContent: "space-around",
            gap: "50px",
            marginLeft: "100px",
          }}
        >
          <div>
            <h2>
              {greeting}, {userData?.name}
            </h2>
          </div>
          <div style={{ flex: 1 }}>
            <h2>Your Health, Our Priority</h2>
            <p>Connecting you to the best healthcare services online.</p>
          </div>
        </div>
        <div style={{ flex: 1 }}>
          <img src={docVisit} alt="Doctor Visit" />
        </div>
      </section>
      <section id="services" className="services">
        <h1
          style={{ display: "flex", justifyContent: "center", fontSize:
"2em" }}
        >
          Services
        </h1>
        <div className="service-list">
          <div className="service-item">
            <h3>Online Consultations</h3>
            <p>Consult with top doctors from the comfort of your home.</p>
          </div>
          <div className="service-item">
```

```
            <h3>Prescription Medicines</h3>
            <p>Order your prescribed medicines with easy home
delivery.</p>
          </div>
          <div className="service-item">
            <h3>Health Tracking</h3>
            <p>Track your health and medication adherence
effortlessly.</p>
          </div>
        </div>
      </section>
      <section id="testimonials">
        <TestimonialsCarousel userData={userData} />
      </section>

      <section id="appointment" className="appointment">
        <h1
          style={{ display: "flex", justifyContent: "center", fontSize:
"2em" }}
        >
          Make an Appointment
        </h1>
        <div
          style={{
            height: "400px",
            display: "flex",
            flexDirection: "row",
            justifyContent: "space-around",
            alignItems: "center",
          }}
        >
          <div>
            <button className="appointment-button"
onClick={handleBookNow}>
              Book Now
            </button>
          </div>
          <div>
            <img src={appointment} alt="Appointment" />
          </div>
        </div>
      </section>

      <section id="buy-medicine" className="appointment">
        <h1
          style={{ display: "flex", justifyContent: "center", fontSize:
"2em" }}
        >
          Order Medicines
        </h1>
        <div
          style={{
            height: "400px",
            display: "flex",
            flexDirection: "row",
            justifyContent: "space-around",
            alignItems: "center",
          }}
        >
          <div>
            <img
              style={{ height: "300px" }}
              src={orderMed}
              alt="Order Medicines"
            />
          </div>
          <div>
            <button className="appointment-button"
onClick={handleBuyNowButton}>
              Order Medicines
            </button>
          </div>
        </div>
      </section>
    </main>
  );
};

const Footer = () => {
  return (
    <footer className="footer">
      <p>Contact Us: eMed@gmail.com | Phone: xxx-xxx-xxxx</p>
    </footer>
  );
};

export default MedicalLandingPage;
```

Landing page for users contains the services provided by us, testimonials by users, booking appointments, option for ordering medicines. It also has some contact information.

Login/Signup:

```
import React, { useState } from "react";
import "../styling/login.css";
import axios from "axios";
import { useNavigate } from "react-router-dom";
import MessageScreen from "./messageScreen";

const Login = () => {
  const navigate = useNavigate();

  const [loginEmail, setLoginEmail] = useState("");
  const [loginPassword, setLoginPassword] = useState("");

  // Signup Form State
  const [signupDetails, setSignupDetails] = useState({
    name: "",
    email: "",
    password: "",
    phone: "",
    birthDate: "",
    bloodGroup: "",
    street: "",
    city: "",
    state: "",
    zipCode: "",
    emergencyName: "",
    emergencyPhone: "",
  });

  const [isLoading, setIsLoading] = useState(false);
  const [error, setError] = useState(null);
  const [user, setLoggedUser] = useState(null);
  const [successMessage, setSuccessMessage] = useState("");

  const handleSignupChange = (e) => {
    const { name, value } = e.target;
    setSignupDetails((prevDetails) => ({ ...prevDetails, [name]: value
}));
  };

  const displayMessageAndNavigate = (message, userData) => {
    setSuccessMessage(message);
    setTimeout(() => {
      setSuccessMessage("");
      navigate("/landing", { state: { userData } });
    }, 3000);
  };

  const handleLogin = async (e) => {
    e.preventDefault();
    setIsLoading(true);
    setError(null);

    try {
      const data = await loginApi({
        email: loginEmail,
        password: loginPassword,
      });
      console.log("data", data);

      setLoggedUser(data.user);
      const userData = data.user.user;
      displayMessageAndNavigate("Login Successful!", data.user.user);

      // Set user in state

      // Navigate to landing page
    } catch (err) {
      setError(err.message);
    } finally {
      setIsLoading(false);
    }
  };

  // Handle signup form submission
  const handleSignup = async (e) => {
    e.preventDefault();
    setIsLoading(true);
    setError(null);

    try {
      const data = await signupApi(signupDetails);
      setLoggedUser({ name: signupDetails.name });
      console.log(data, "signupData");
      const userData = signupDetails;

      if (data) {
        navigate("/landing", { state: { userData } });
      }
    } catch (err) {
      setError(err.message);
    } finally {
      setIsLoading(false);
    }
  };

  const loginApi = async (loginData) => {
    try {
      const response = await axios.post(
        "http://localhost:5000/api/login",
        loginData
      );
      console.log(response);

      return response.data;
    } catch (error) {
      throw new Error(error.response?.data?.error || "Login failed.");
    }
  };

  const signupApi = async (signupData) => {
    console.log(signupData);
    try {
      const response = await axios.post(
        "http://localhost:5000/api/signup",
        signupData
      );
      return response.data;
    } catch (error) {
      throw new Error(error.response?.data?.error || "Signup failed.");
    }
  };

  if (successMessage) {
    return <MessageScreen message={successMessage} />;
  }
  return (
    <>
      <div className="wrapper ">
        <div className="card-switch">
          <label className="switch">
            <input type="checkbox" className="toggle" />
            <span className="slider"></span>
            <span className="card-side"></span>
            <div className="flip-card__inner">
              <div className="flip-card__front">
                <div className="title">Log in</div>
                <form className="flip-card__form" onSubmit={handleLogin}>
                  <input
                    className="flip-card__input"
                    name="email"
                    placeholder="Email"
                    type="email"
                    value={loginEmail}
                    onChange={(e) => setLoginEmail(e.target.value)}
                  />
                  <input
```

```
                    className="flip-card__input"
                    name="password"
                    placeholder="Password"
                    type="password"
                    value={loginPassword}
                    onChange={(e) => setLoginPassword(e.target.value)}
                  />
                  <button className="flip-card__btn"
onClick={handleLogin}>
                    Let’s go!
                  </button>
                </form>
              </div>

              <div className="flip-card__back">
                <div className="title">Sign up</div>
                <form className="flip-card__form" onSubmit={handleSignup}>
                  <input
                    className="flip-card__input"
                    name="name"
                    placeholder="Name"
                    type="text"
                    value={signupDetails.name}
                    onChange={handleSignupChange}
                  />
                  <div
                    style={{
                      display: "flex",
                      flexDirection: "row",
                      gap: "30px",
                    }}
                  >
                    {" "}
                    <input
                      className="flip-card__input"
                      name="email"
                      placeholder="Email"
                      type="email"
                      value={signupDetails.email}
                      onChange={handleSignupChange}
```

```
  />
  <input
    className="flip-card__input"
    name="password"
    placeholder="Password"
    type="password"
    value={signupDetails.password}
    onChange={handleSignupChange}
  />
</div>
<div
  style={{
    display: "flex",
    flexDirection: "row",
    gap: "30px",
  }}
>
  {" "}
  <input
    className="flip-card__input"
    name="phone"
    placeholder="Phone Number"
    type="text"
    value={signupDetails.phone}
    onChange={handleSignupChange}
  />
  <input
    className="flip-card__input"
    name="birthDate"
    placeholder="Date of Birth"
    type="date"
    value={signupDetails.birthDate}
    onChange={handleSignupChange}
  />
</div>
<select
  className="flip-card__input"
  name="bloodGroup"
  value={signupDetails.bloodGroup}
  style={{ width: "270px", height: "50px" }}
  onChange={handleSignupChange}
>
  <option value="" disabled>
    Select Blood Group
  </option>
  <option>A+</option>
  <option>A-</option>
  <option>B+</option>
  <option>B-</option>
  <option>AB+</option>
  <option>AB-</option>
  <option>O+</option>
  <option>O-</option>
</select>
<div
  style={{
    display: "flex",
    flexDirection: "row",
    gap: "30px",
  }}
>
  {" "}
  <input
    className="flip-card__input"
    name="street"
    placeholder="Street"
    type="text"
    value={signupDetails.street}
    onChange={handleSignupChange}
  />
  <input
    className="flip-card__input"
    name="city"
    placeholder="City"
    type="text"
    value={signupDetails.city}
    onChange={handleSignupChange}
  />
</div>
<div
```

```
                    style={{
                      display: "flex",
                      flexDirection: "row",
                      gap: "30px",
                    }}
                  >
                    <input
                      className="flip-card__input"
                      name="state"
                      placeholder="State"
                      type="text"
                      value={signupDetails.state}
                      onChange={handleSignupChange}
                    />
                    <input
                      className="flip-card__input"
                      name="zipCode"
                      placeholder="Zip Code"
                      type="text"
                      value={signupDetails.zipCode}
                      onChange={handleSignupChange}
                    />
                  </div>
                  <div
                    style={{
                      display: "flex",
                      flexDirection: "row",
                      gap: "30px",
                    }}
                  >
                    <input
                      className="flip-card__input"
                      name="emergencyName"
                      placeholder="Emergency Contact Name"
                      type="text"
                      value={signupDetails.emergencyName}
                      onChange={handleSignupChange}
                    />
                    <input
                      className="flip-card__input"
                      name="emergencyPhone"
                      placeholder="Emergency Contact Phone"
                      type="text"
                      value={signupDetails.emergencyPhone}
                      onChange={handleSignupChange}
                    />
                  </div>
                  <button className="flip-card__btn"
onClick={handleSignup}>
                    Confirm!
                  </button>
                </form>
              </div>
            </div>
          </label>
        </div>

        {/* {isLoading && <p>Loading...</p>}
        {error && <p>Error: {error}</p>} */}
      </div>
    </>
  );
};

export default Login;
```

Login page has the signup and login features where user provide their details like email, address phone number, blood group and other emergency details for registration and it also has a page for login which redirects to landing page with the user details on successful login or signup.

Doctor Appointment:

```
import React, { useState, useEffect } from "react";
import { useLocation, useNavigate } from "react-router-dom";
import "../styling/appointment.css";
import {
  TextField,
  Button,
  MenuItem,
  Typography,
  Container,
  Grid,
  InputLabel,
  Select,
  FormControl,
} from "@mui/material";

const DoctorAppointment = () => {
  const location = useLocation();
  const { userData } = location.state || {};
  console.log(userData);
  const [formData, setFormData] = useState({
    name: userData.name,
    email: userData.email,
    date: "",
    time: "",
    doctor: "",
  });
  const [isSubmitted, setIsSubmitted] = useState(false);
  const navigate = useNavigate();

  const doctorsList = [
    "Dr. John Doe - Cardiologist",
    "Dr. Sarah Smith - Dermatologist",
    "Dr. Emma Taylor - Neurologist",
  ];

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData((prevData) => ({ ...prevData, [name]: value }));
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("Appointment booked:", formData);
    setIsSubmitted(true);
  };
  useEffect(() => {
    if (isSubmitted) {
      const timer = setTimeout(() => {
        navigate("/landing", { state: { userData } });
      }, 5000);

      return () => clearTimeout(timer);
    }
  }, [isSubmitted, navigate]);

  return (
    <Container className="appointment-container">
      {/* <Typography className="title">Book a Doctor
Appointment</Typography> */}
      {isSubmitted ? (
        <Typography className="success-message">
          Appointment successfully booked! We’ll see you soon.
        </Typography>
      ) : (
        <div>
          <h1> Book your Appointment</h1>
          <form onSubmit={handleSubmit} noValidate>
            <Grid container spacing={2}>
              <Grid item xs={12}>
                <TextField
                  fullWidth
                  required
                  label="Your Name"
                  name="name"
                  value={formData.name}
                  onChange={handleChange}
                />
              </Grid>
              <Grid item xs={12}>
                <TextField
                  fullWidth
                  required
                  label="Email Address"
                  name="email"
                  type="email"
                  value={formData.email}
                  onChange={handleChange}
                />
              </Grid>
              <Grid item xs={6}>
                <TextField
                  fullWidth
                  required
                  name="date"
                  label="Appointment Date"
                  type="date"
                  InputLabelProps={{ shrink: true }}
                  value={formData.date}
                  onChange={handleChange}
                />
              </Grid>
              <Grid item xs={6}>
                <TextField
                  fullWidth
                  required
                  name="time"
                  label="Appointment Time"
                  type="time"
                  InputLabelProps={{ shrink: true }}
                  value={formData.time}
                  onChange={handleChange}
                />
              </Grid>
              <Grid item xs={12}>
                <FormControl fullWidth required>
                  <InputLabel>Select Doctor</InputLabel>
                  <Select
                    name="doctor"
                    value={formData.doctor}
                    onChange={handleChange}
                  >
                    {doctorsList.map((doctor, index) => (
                      <MenuItem key={index} value={doctor}>
                        {doctor}
                      </MenuItem>
                    ))}
                  </Select>
                </FormControl>
              </Grid>
            </Grid>
            <Button
              type="submit"
              fullWidth
              variant="contained"
              className="book-button"
            >
              Book Appointment
            </Button>
          </form>
        </div>
      )}
    </Container>
  );
};

export default DoctorAppointment;
```

After clicking on the book appointment button in the landing page it redirects to the appointment booking page with the user details present

Here users can select the doctors details along with data and time for the appointment and can confirm booking.

Medicines:

```
import React, { useEffect, useState } from "react";
import { useNavigate, useLocation } from "react-router-dom";
import { Pagination } from "@mui/material";
import "../styling/medicines.css";
import output100 from "../assets/output100.json";

const MedicinePage = () => {
  const [medicines, setMedicines] = useState([]);
  const [filteredMedicines, setFilteredMedicines] = useState([]);
  const [cart, setCart] = useState(new Map());
  const [currentPage, setCurrentPage] = useState(1);
  const [searchTerm, setSearchTerm] = useState("");
  const itemsPerPage = 10;
  const navigate = useNavigate();
  const location = useLocation();
  const { cartSaved } = location.state || {};
  const { userData } = location.state || {};

  console.log("user data in medicines", userData);
  useEffect(() => {
    setMedicines(output100);
    setFilteredMedicines(output100);

    const savedCart = localStorage.getItem("cart");
    if (savedCart) setCart(new Map(JSON.parse(savedCart)));

    if (cartSaved) {
      const savedCartMap = new Map(cartSaved.map((item) => [item.id,
item]));
      setCart(savedCartMap);
    }
  }, [cartSaved]);

  useEffect(() => {
    localStorage.setItem("cart",
JSON.stringify(Array.from(cart.entries())));
  }, [cart]);

  const handleAddToCart = (medicine) => {
    setCart((prevCart) => {
      const newCart = new Map(prevCart);
      const currentQty = newCart.get(medicine.id)?.quantity || 0;
      newCart.set(medicine.id, { ...medicine, quantity: currentQty + 1 });
      return newCart;
    });
  };

  const handleRemoveFromCart = (medicine) => {
    setCart((prevCart) => {
      const newCart = new Map(prevCart);
      const currentQty = newCart.get(medicine.id)?.quantity || 0;

      if (currentQty > 1) {
        newCart.set(medicine.id, { ...medicine, quantity: currentQty - 1
});
      } else {
        newCart.delete(medicine.id);
      }
      return newCart;
    });
  };

  const handlePageChange = (event, value) => setCurrentPage(value);

  const handleCheckout = () => {
    const cartArray = Array.from(cart.values());
    navigate("/cart", { state: { cart: cartArray, userData } });
  };

  const handleSearch = (e) => {
    const value = e.target.value.toLowerCase();
    setSearchTerm(value);
    const filtered = medicines.filter((medicine) =>
      medicine.name.toLowerCase().includes(value)
    );
    setFilteredMedicines(filtered);
    setCurrentPage(1);
  };

  const indexOfLastItem = currentPage * itemsPerPage;
  const indexOfFirstItem = indexOfLastItem - itemsPerPage;
  const currentMedicines = filteredMedicines.slice(
    indexOfFirstItem,
    indexOfLastItem
  );

  return (
    <div className="medicine-container">
      <h1>Available Medicines</h1>
      <input
        type="text"
        placeholder="Search for a medicine..."
        value={searchTerm}
        onChange={handleSearch}
        // style={{ marginBottom: "20px", padding: "10px", width: "100%"
}}
        className="search"
      />
      <div className="medicine-list">
        {currentMedicines.map((medicine) => {
          const cartItem = cart.get(medicine.id) || {};
          const quantity = cartItem.quantity || 0;

          return (
            <div key={medicine.id} className="medicine-item">
              <h2>{medicine.name}</h2>
              <p>Price: Rs {medicine.price}</p>
              {quantity > 0 ? (
                <div className="quantity-controls">
                  <button onClick={() => handleRemoveFromCart(medicine)}>
                    -
                  </button>
                  <span>{quantity}</span>
                  <button onClick={() =>
handleAddToCart(medicine)}>+</button>
                </div>
              ) : (
                <button onClick={() => handleAddToCart(medicine)}>
                  Add to Cart
                </button>
              )}
            </div>
          );
        })}
      </div>

      <Pagination
        count={Math.ceil(filteredMedicines.length / itemsPerPage)}
        page={currentPage}
        onChange={handlePageChange}
        color="primary"
        style={{ marginTop: "20px", display: "flex", justifyContent:
"center" }}
        sx={{
          "& .MuiPaginationItem-root": {
            color: "black",
            backgroundColor: "#f0f0f0",
            margin: "0 5px",
          },
          "& .Mui-selected": {
            backgroundColor: "#1976d2",
            color: "white",
          },
          "& .MuiPaginationItem-ellipsis": {
            color: "#888",
          },
        }}
      />

      <button
        onClick={handleCheckout}
        style={{ marginTop: "20px", padding: "10px 20px", cursor:
"pointer" }}
      >
        Go to Cart (
        {Array.from(cart.values()).reduce((a, b) => a + b.quantity, 0)})
      </button>
    </div>
  );
};

export default MedicinePage;
```

This page can be access by clicking on order now button on landing page

This page has a list of medicines. From here users can select the required medicines and can add them to the cart.

Cart:

```
import React, { useState, useEffect } from "react";
import { useLocation, useNavigate } from "react-router-dom";
import "../styling/cart.css";

const CartPage = () => {
  const { state } = useLocation();
  const navigate = useNavigate();
  const [cart, setCart] = useState(state?.cart || []);
  const [deliveryMessage, setDeliveryMessage] = useState("");
  const userData = state?.userData || {};

  const handleBackToMedicine = () => {
    const cartSaved = cart;
    navigate("/medicines", { state: { cartSaved, userData } });
  };

  const handleCheckout = () => {
    const address = userData.address || "your address";
    setDeliveryMessage(
      `Your medicines will be delivered to ${address.street},
${address.city}, ${address.state} shortly!`
    );
    setCart([]);
  };

  useEffect(() => {
    if (deliveryMessage) {
      const timer = setTimeout(() => {
        navigate("/landing", { state: { userData } });
      }, 3000);
      return () => clearTimeout(timer);
    }
  }, [deliveryMessage, navigate, userData]);

  const totalPrice = cart
    .reduce((acc, item) => acc + item.price * item.quantity, 0)
    .toFixed(2);
  return (
    <div className="cart-container">
      <h1>Your Cart</h1>
      {cart.length === 0 ? (
        <p>Your cart is empty.</p>
      ) : (
        <>
          <ul>
            {cart.map((item) => (
              <li className="card-li" key={item.id}>
                <h3>{item.name}</h3>
                <p>Quantity: {item.quantity}</p>
                <p> Rs {item.price * item.quantity}</p>
              </li>
            ))}
          </ul>
          <h3>Total: Rs {totalPrice}</h3>
          <button onClick={handleCheckout}>Proceed to Checkout</button>
        </>
      )}
      {deliveryMessage && <p
className="delivery-message">{deliveryMessage}</p>}
      <button onClick={handleBackToMedicine}>Back to Medicines</button>
    </div>
  );
};

export default CartPage;
```

This is the cart page where users can see all the medicines that they have added to cart by selecting in the medicines page, here users can see the price of individual medicine as well as the total cart price and also have an option for going back to medicines list to add more medicines to cart or to remove few medicines as required.

From here the users can go to the payment gateway which will be implemented in the later phases.

BackendCode:

Server:

```
const express = require("express");
const mongoose = require("mongoose");
const bodyParser = require("body-parser");
const cors = require("cors");
const bcrypt = require("bcrypt");

const app = express();
const PORT = process.env.PORT || 5000;

// Middleware
app.use(bodyParser.json());
app.use(cors());

// MongoDB Connection
mongoose
  .connect(

"mongodb+srv://gnevercodes:seproject@softwareengineeringdb.xli4x.mongodb.n
et/mydatabase?retryWrites=true&w=majority",
    { useNewUrlParser: true, useUnifiedTopology: true }
  )
  .then(() => console.log("Connected to MongoDB successfully!"))
  .catch((error) => console.error("Failed to connect to MongoDB:",
error));

// Event listeners for MongoDB connection
mongoose.connection.on("connected", () =>
  console.log("Mongoose is connected.")
);
mongoose.connection.on("error", (err) => console.error("Mongoose error:",
err));
mongoose.connection.on("disconnected", () =>
  console.log("Mongoose disconnected.")
);

// User Schema and Model
```

```
const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true, match: /.+\@.+\..+/
},
  password: { type: String, required: true },
  phoneNumber: { type: String, required: true, match: /^\d{10}$/ },
  dateOfBirth: { type: Date, required: true },
  bloodGroup: {
    type: String,
    enum: ["A+", "A-", "B+", "B-", "AB+", "AB-", "O+", "O-"],
    required: true,
  },
  address: {
    street: { type: String, required: true },
    city: { type: String, required: true },
    state: { type: String, required: true },
    zipCode: { type: String, required: true },
  },
  emergencyContact: {
    name: { type: String, required: true },
    phoneNumber: { type: String, required: true, match: /^\d{10}$/ },
  },
  role: { type: String, enum: ["user", "admin"], default: "user" },
  createdAt: { type: Date, default: Date.now },
  updatedAt: { type: Date, default: Date.now },
});

// Middleware to update `updatedAt` before save
userSchema.pre("save", function (next) {
  this.updatedAt = Date.now();
  next();
});

const User = mongoose.model("User", userSchema);

// Signup Route
app.post("/api/signup", async (req, res) => {
  try {
    const {
      name,
      email,
      password,
      phone: phoneNumber,
      birthDate: dateOfBirth,
      bloodGroup,
      street,
      city,
      state,
      zipCode,
      emergencyName,
      emergencyPhone,
    } = req.body;

    // Create new user
    const newUser = new User({
      name,
      email,
      password: await bcrypt.hash(password, 10),
      phoneNumber,
      dateOfBirth,
      bloodGroup,
      address: {
        street,
        city,
        state,
        zipCode,
      },
      emergencyContact: {
        name: emergencyName,
        phoneNumber: emergencyPhone,
      },
    });

    await newUser.save();
    res.status(201).json({ message: "User registered successfully!" });
  } catch (error) {
    console.error("Signup error:", error);
    res.status(500).json({ error: "Signup failed." });
  }
});
```

```
app.post("/api/login", async (req, res) => {
  const { email, password } = req.body;

  try {
    console.log("Attempting login for email:", email);

    const user = await User.findOne({ email });
    console.log("User found:", user);

    if (!user) {
      console.log("User not found");
      return res.status(400).json({ error: "Invalid email or password."
});
    }

    // Check password
    const isPasswordValid = await bcrypt.compare(password, user.password);
    console.log("Password valid:", isPasswordValid);

    if (!isPasswordValid) {
      console.log("Password incorrect");
      return res.status(400).json({ error: "Invalid email or password."
});
    }

    // If successful, respond with user information
    res.json({
      message: "Login successful",
      user: { user },
    });
  } catch (error) {
    console.error("Login error:", error);
    res.status(500).json({ error: "Login failed." });
  }
});

// Export the app for testing purposes
module.exports = app;
```

```
// Start Server
if (require.main === module) {
  app.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
  });
}
```

Firstly this code provides the connection to the mongodb that we created using mongoose.

Here we create the user schema which will be stored in the database, which will be used to store user data on successful signup and this data is used for login purpose in the login page in the front end

This also has 2 apis where one is called by signup component in the frontend which upon success helps in storing the user details in the users schema in the database.

Another api is called by login component where email and password is provided to the api and this api checks if there is any user with the provided email and authenticate if the provided credentials match.

Test Cases for SignUp and Login:

```
const request = require("supertest");
const mongoose = require("mongoose");
const app = require("../server");

// beforeAll(async () => {
//   // Connect to the test database before running the tests
//   await mongoose.connect("mongodb://localhost:27017/test_db", {
//     useNewUrlParser: true,
//     useUnifiedTopology: true,
//   });
// });

// afterAll(async () => {
//   // Clean up and close the database connection after all tests
//   await mongoose.connection.dropDatabase(); // Optional: Drop test DB
after tests
//   await mongoose.connection.close();
// });
```

```
describe("POST /api/signup", () => {
  test("should successfully register a new user", async () => {
    const response = await request(app).post("/api/signup").send({
      name: "Test User",
      email: "testuser@example.com",
      password: "TestPassword123",
      phone: "1234567890",
      birthDate: "1990-01-01",
      bloodGroup: "O+",
      street: "123 Main St",
      city: "Test City",
      state: "Test State",
      zipCode: "12345",
      emergencyName: "Emergency Contact",
      emergencyPhone: "0987654321",
    });
    expect(response.statusCode).toBe(201);
    expect(response.body.message).toBe("User registered successfully!");
  });

  test("should fail to register if the email is already in use", async ()
=> {
    const response = await request(app).post("/api/signup").send({
      name: "Test User",
      email: "testuser@example.com",
      password: "TestPassword123",
      phone: "1234567890",
      birthDate: "1990-01-01",
      bloodGroup: "O+",
      street: "123 Main St",
      city: "Test City",
      state: "Test State",
      zipCode: "12345",
      emergencyName: "Emergency Contact",
      emergencyPhone: "0987654321",
    });
    expect(response.statusCode).toBe(500);
    expect(response.body.error).toBe("Signup failed.");
  });
});
```

```
describe("POST /api/login", () => {
  test("should successfully log in a user with correct credentials", async
() => {
    const response = await request(app).post("/api/login").send({
      email: "testuser@example.com",
      password: "TestPassword123",
    });
    expect(response.statusCode).toBe(200);
    expect(response.body.message).toBe("Login successful");
  });

  test("should fail to log in a user with incorrect password", async () =>
{
    const response = await request(app).post("/api/login").send({
      email: "testuser@example.com",
      password: "WrongPassword123",
    });
    expect(response.statusCode).toBe(400);
    expect(response.body.error).toBe("Invalid email or password.");
  });

  test("should fail to log in a user with non-existing email", async () =>
{
    const response = await request(app).post("/api/login").send({
      email: "nonexistentuser@example.com",
      password: "SomePassword123",
    });
    expect(response.statusCode).toBe(400);
    expect(response.body.error).toBe("Invalid email or password.");
  });
});
```

This file is used for testing the apis in the server.js where various cases are given such as valid signup, valid login, invalid user, duplicate user signup, valid email but invalid password