DocSpot - Complete Project Code Documentation

Frontend (ReactJS) - App.js

```
import React from "react";
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Login from "./components/Login";
import Register from "./components/Register";
import Dashboard from "./components/Dashboard";
import BookingForm from "./components/BookingForm";

function App() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<Login />} />
        <Route path="/register" element={<Register />} />
        <Route path="/dashboard" element={<Dashboard />} />
        <Route path="/book/:doctorId" element={<BookingForm />} />
      </Routes>
    </Router>
  );
}

export default App;
```

Backend (Node.js + Express) - server.js

```
const express = require("express");
const mongoose = require("mongoose");
const userRoutes = require("./routes/userRoutes");
const appointmentRoutes = require("./routes/appointmentRoutes");
const cors = require("cors");

const app = express();
app.use(cors());
app.use(express.json());

mongoose.connect("mongodb://localhost:27017/docspot", {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => console.log("MongoDB connected"))
  .catch(err => console.log(err));

app.use("/api/users", userRoutes);
app.use("/api", appointmentRoutes);

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Backend Route - userRoutes.js

```
const express = require("express");
const router = express.Router();
const User = require("../models/User");
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");

router.post("/register", async (req, res) => {
  const { name, email, password, role } = req.body;
  const hashedPassword = await bcrypt.hash(password, 10);
  const user = new User({ name, email, password: hashedPassword, role });
  await user.save();
  res.status(201).json({ message: "User registered" });
});

router.post("/login", async (req, res) => {
  const { email, password } = req.body;
  const user = await User.findOne({ email });
  if (!user) return res.status(400).json({ message: "Invalid email" });
  const isMatch = await bcrypt.compare(password, user.password);
  if (!isMatch) return res.status(400).json({ message: "Invalid password" });
  const token = jwt.sign({ id: user._id, role: user.role }, "secretKey");
  res.json({ token });
});

module.exports = router;
```

Model - User.js

```
const mongoose = require("mongoose");

const UserSchema = new mongoose.Schema({
  name: String,
  email: String,
  password: String,
  role: String  // 'patient', 'doctor', or 'admin'
});

module.exports = mongoose.model("User", UserSchema);
```

Model - Appointment.js

```
const mongoose = require("mongoose");

const AppointmentSchema = new mongoose.Schema({
  doctorId: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },
  patientId: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },
  date: String,
  timeSlot: String,
  status: { type: String, default: "booked" }
});
```

```
module.exports = mongoose.model("Appointment", AppointmentSchema);
```