Project Name: SecureMediBot Backend

A secure, privacy-focused backend system for healthcare chatbot services

---

🧠 Abstract

SecureMediBot is a backend project designed to power a healthcare chatbot system with a strong emphasis on **security**, **privacy**, and **role-based access**. It ensures that sensitive health data is protected through modern security practices and that users interact within the boundaries of their access level (e.g., doctor, patient, admin). This project is built with **Node.js**, **Express.js**, and **MongoDB**, and integrates real-world security measures like **JWT authentication**, **bcrypt encryption**, **input validation**, **rate limiting**, and **role-based authorization**.

🎯 Introduction

Healthcare systems are increasingly adopting chatbots to provide instant support to patients. However, securing user health data (EHR) remains a major challenge. SecureMediBot solves this by providing a robust backend that manages:

- ✅ User authentication and authorization
- ✅ Role-based access (Doctor, Patient, Admin)
- ✅ Secure data storage of health records
- ✅ Logging, error handling, and input validation
- ✅ Preventing common security threats

This is not just a CRUD app — it reflects real-world challenges and solutions faced by healthcare backend systems.

⚙️ Tech Stack

Layer	Tools/Tech
Language	Node.js (JavaScript)
Framework	Express.js
Database	MongoDB + Mongoose
Authentication	JWT (JSON Web Tokens)
Encryption	bcrypt
Validation & Sanitization	express-validator, mongo-sanitize, helmet
Security Middleware	helmet, rate-limiter-flexible, CORS
Deployment	Render / Railway / Vercel backend
API Testing	Postman / Thunder Client
Documentation	Swagger or Markdown

🔐 Key Features

✅ Authentication & Authorization

- JWT-based login/signup
- Password hashing with bcrypt
- Role-based access control (RBAC)

🧑‍⚕️ User Roles

- **Patient**: Can view their own records, update profile
- **Doctor**: Can add diagnosis, view assigned patients
- **Admin**: Full control over users and health data

🔒 Security Measures

- Input validation & sanitization
- Rate limiting (protect from brute-force attacks)
- XSS & NoSQL injection protection (helmet, mongo-sanitize)

🧾 Health Record Management

- Patients can upload/view reports
- Doctors can update prescriptions/diagnoses
- Admins can manage all data securely

📦 RESTful API Design

- Modular routes
- Middleware-based access control
- Clean, scalable folder structure

📁 Logging & Error Handling

- Global error handler middleware
- Proper status codes and messages

📌 Use Cases

- Used as the backend for **healthcare chatbots**
- Useful for **hospital management systems**
- Can be extended to support **telemedicine** or **e-pharmacy** platforms

- Acts as a **learning platform** for backend + security

📈 Why This Project is Valuable

Category	Value
Real-World Relevance	Solves an actual problem — secure healthcare data
Depth	Beyond CRUD — introduces security layers
Interview-Worthy	Shows skills in backend, security, data design
Portfolio Impact	Can be a standout project on resume/LinkedIn
Scalable	Can be connected with AI chatbot frontend or mobile apps

---

✅ Future Improvements

- Add Docker for containerization
- Add CI/CD using GitHub Actions
- Write unit tests with Jest
- Add chatbot frontend (React or Next.js)
- Integrate OAuth login (Google, GitHub)

🧩 4. Core Modules to Implement

✅ 1. User Registration & Login

- Register patient/doctor/admin (only admin can add doctors)
- Login → JWT issued
- Hash passwords using `bcrypt`

✅ 2. JWT-based Authentication

- Protect all private routes
- Verify token in headers

✅ 3. Role-Based Authorization Middleware

- Custom middleware to allow only specific roles
- Example: `/admin/users` → Only `admin` allowed

✅ 4. Patient-Doctor Record Management

- Patients can view their records only
- Doctors can view/update assigned patient records
- Admin can view all records

✅ 5. Chat Module (optional)

- Secure chat between patient and doctor
- Save chat logs in DB

✅ 6. Security Implementation

- `helmet` → secure headers
- `express-validator` → input validation
- Rate limiting (optional)
- Sanitize all inputs

✅ 7. Error Handling & Logging

- Centralized error handler middleware
- Log errors using `winston` or basic logging

```
secure-healthbot-backend/
│
├── src/                    # 🔹 All source code lives here
│   ├── config/             # 🔹 Configuration (DB, env, etc.)
│   │   ├── db.js           # MongoDB connection
│   │   └── index.js        # Load configs
│   │
│   ├── controllers/        # 🔹 Route logic (one per feature)
│   │   ├── auth.controller.js
│   │   ├── user.controller.js
│   │   ├── record.controller.js
│   │   └── chat.controller.js
│   │
│   ├── routes/             # 🔹 Express routes (modular)
│   │   ├── auth.routes.js
│   │   ├── user.routes.js
│   │   ├── record.routes.js
│   │   └── chat.routes.js
│   │
│   ├── models/             # 🔹 Mongoose schemas
│   │   ├── User.model.js
│   │   ├── Record.model.js
│   │   └── Chat.model.js
│   │
│   ├── middleware/         # 🔹 Auth, RBAC, error, validators
│   │   ├── auth.middleware.js
│   │   ├── rbac.middleware.js
│   │   ├── error.middleware.js
│   │   └── validate.middleware.js
│   │
│   ├── services/           # 🔹 Business logic (non-controller)
│   │   ├── auth.service.js
│   │   ├── user.service.js
│   │   └── record.service.js
│   │
│   ├── utils/              # 🔹 Helpers (JWT, email, etc.)
│   │   ├── jwt.js
│   │   ├── logger.js
│   │   └── sanitizer.js
│   │
│   ├── app.js              # 🔹 Express app setup
│   └── server.js           # 🔹 Server entry point
│
├── tests/                  # 🔹 Unit & integration tests
```

```
│   └── auth.test.js
│
├── .env                    # 🔹 Environment variables
├── .gitignore
├── README.md               # 🔹 Project info & usage
├── package.json
└── package-lock.json
```