Project Report:

Role-Based Access Control (RBAC) RAG Medical Chatbot

1. Project Overview

The RBAC RAG Medical Chatbot is a domain-specific AI assistant designed for healthcare environments. It utilizes Retrieval-Augmented Generation (RAG) to answer medical queries using a role-restricted document corpus. Access to content is governed by user roles such as Admin, Doctor, Nurse, and Patient, ensuring security and privacy in information dissemination.

---

2. Objective

To build a secure, scalable, and intelligent chatbot that:

- Allows Admins to upload and index role-based medical documents.
- Enables Doctors, Nurses, and Patients to query relevant healthcare information.
- Restricts data access based on user roles using RBAC.
- Provides accurate answers using a RAG pipeline with Groq LLM.

---

3. Tech Stack

Backend:

- **FastAPI**: For serving APIs
- **MongoDB Atlas**: For user authentication and role management
- **Pinecone**: Vector DB to store role-annotated document embeddings
- **Google GenAI Embeddings**: For converting text and queries into vector representations
- **Groq (LLaMA 3)**: To generate answers using context retrieved from Pinecone

Frontend:

- **Streamlit**: For user-friendly chat interface

Utilities:

- **LangChain**: To orchestrate embedding and LLM chains
- **dotenv, bcrypt**: For environment and security

---

4. System Architecture

1. **Authentication Layer**:
    - User login/signup using HTTPBasic auth
    - Passwords hashed using bcrypt
    - MongoDB stores users and roles
2. **Document Upload (Admin Only)**:
    - Admin uploads medical PDFs
    - Files are converted to text → split into chunks → embedded → stored in Pinecone
    - Metadata includes `role`, `source`, and `doc_id`
3. **Query Flow (RAG Pipeline)**:
    - User sends query via chat
    - Query is embedded
    - Pinecone returns top-k matches for user role
    - Groq LLM generates answer using LangChain prompt with context
    - Response returned to frontend

---

5. Core Modules

auth/routes.py

- `signup`: User registration with role
- `login`: Basic HTTP auth and role check
- `authenticate`: Dependency injection for role-based access control

chat/chat_query.py

- `answer_query(query, user_role)`: Embeds query, retrieves relevant docs, invokes LLM

vectordb/load_vectorstore.py

- `load_vectorstore(files, role, doc_id)`: Parses, chunks, embeds, and uploads files to Pinecone with role-based tagging

ui/chat_ui.py

- Streamlit app with role-aware chat and login UI

---

6. API Endpoints

Method	Endpoint	Description	Role
POST	`/signup`	Register a new user with role	Public
GET	`/login`	Authenticate and return role	All
POST	`/upload_docs`	Upload and index documents	Admin
POST	`/chat`	Submit query and receive role-filtered response	Doctor/Nurse/Patient

---

7. Role-Based Access Control (RBAC)

Role	Capabilities
Admin	Upload documents, create users
Doctor	Ask medical questions, access doctor-only content
Nurse	Ask support-related questions, limited content access
Patient	Ask general questions, minimal access

---

8. Future Enhancements

- JWT-based auth with refresh tokens
- Audit logs for document uploads and query history
- Granular permissions per document section
- Dashboard with analytics for Admin

9. Conclusion

This RBAC-based RAG Medical Chatbot demonstrates a secure, scalable way to use GenAI in healthcare settings. It ensures compliance with data boundaries by design while maintaining flexibility and intelligence through a modular architecture.