Doc'S' Bot Application Documentation

Objective: The use is to create a BOT demonstrate document search using Open AI. The Bot will take set of documents as input and answers user queries using information from the uploaded documents

1. Design & Approach Document

1.1System Overview

The Doc'S' Bot is a document management and query system that allows users to:

- Authenticate into the system
- Upload and manage documents
- Query documents using natural language
- Receive AI-powered responses based on document content

1.2 Architecture

Frontend Architecture

- React-based SPA using TypeScript
- State management using React hooks
- RESTful API integration using Axios

Backend Architecture

- Fast API-based REST API
- OAuth2 authentication system
- Asynchronous document processing
- Vector-based document search using embeddings
- Background task management for document processing

1.3 Key Components

Frontend Components

1. **App.tsx**: Main application container
    a. Manages authentication state
    b. Handles routing logic
    c. Coordinates component interactions
2. **Login.tsx**: Authentication interface
    a. User credential input
    b. Form validation
    c. Authentication state management
3. **FileUpload.tsx**: Document upload interface
    a. File selection
    b. Upload progress tracking
    c. Error handling
4. **FileList.tsx**: Document management
    a. Document listing
    b. Deletion functionality
    c. Status display
5. **DocumentSearch.tsx**: Query interface
    a. Question input
    b. Response display
    c. Loading state management

Backend Components

1. **Authentication Module**
    a. User verification
2. **Document Processing Pipeline**
    a. File validation
    b. Text extraction
    c. Chunk processing
    d. Embedding generation
3. **Query Processing System**
    a. Question embedding
    b. Vector similarity search
    c. Response generation

d. Context management

1.4 Data Flow

1. **Authentication Flow:**
   - User Input → Login Request → Token Generation → Client Storage → Protected Routes
2. **Document Upload Flow:**
   - File Selection → Upload Request → Background Processing → Status Updates → Completion Notification

3. **Query Flow:**
   - Question Input → Embedding Generation → Vector Search → Context Retrieval → Response Generation

1.5 Environment Setup

1.5.1 Prerequisites

- Python 3.9+
- Node.js 16+
- npm 8+
- Git
- For detailed prerequisites check “requirements.txt”

1.5.2 Environment Variables

- GOOGLE_API_KEY= your Gemini API
- PINECONE_API_KEY= your Pinecone API
- PINECONE_INDEX_NAME= your index name