AI Customer Support Chat Platform (Simplified)

Objective

Build a simplified AI-based customer support chat application where users can interact with a virtual assistant powered by OpenAI (GPT-4 or GPT-3.5) through a responsive chat UI.

Tech Stack

Frontend: React (with MobX for state management)
Backend: Node.js + Express
Database: MongoDB
AI Integration: Azure OpenAI (GPT-4 or GPT-3.5)

Key Features

1. User Interface (Frontend)
- A clean chat interface where users can type and send messages.
- Display both user and bot messages in chat bubbles.
- Auto-scroll to latest message.
- MobX should be used for state management (e.g., storing messages, loading states).

2. Backend APIs
- API endpoint to receive user messages and return GPT response.
- Store conversations in MongoDB (session-based or user-based).
- Provide an endpoint to retrieve previous chat history.

3. AI Integration
- Use Azure OpenAI API (GPT-3.5 or GPT-4) to generate bot replies.
- Include basic prompt engineering to maintain conversational context.
- (Optional) Add a system prompt to define assistant personality (e.g., "You are a helpful support agent...").

4. Database (MongoDB)
- Collections:
  - conversations: Contains userId, messages, timestamp
  - faqs or companyData: Contains uploaded content used for contextual responses

5. FAQ/Company Data Upload & Contextual Response
- Provide an interface where an admin can upload FAQs or company-specific documents (PDF or plain text).
- Backend should process and store this data (either as text chunks or embeddings).
- When a visitor asks a question, the AI should first try to answer using this uploaded data.
  - You can use basic string matching or (bonus) implement embeddings-based similarity search using Azure Cognitive Search or langchain.

AI Customer Support Chat Platform (Simplified)

Bonus (Not Mandatory)

- Add a simple login screen with dummy credentials (JWT-based).
- Add a typing indicator ("Agent is typing...").
- Allow users to download chat as a .txt or .pdf file.

Expectations

- Clean, well-commented code.
- GitHub repository with README and clear instructions to run.
- Frontend and backend should run independently with .env files for configs.
- Use modern component architecture and proper async/await handling.

Time Limit

Estimated completion: 3-5 days
Let us know if you need clarification at any point.