AI Engineer Take-Home Project: Patient Chat Application

Project Overview

You have been contracted to build a Django application where a patient (user) can interact with an AI bot regarding their health and care plan. The AI bot is designed to handle health-related conversations and detect patient requests for changes to their treatment or appointments, while filtering out irrelevant or sensitive topics.

The focus of this project is to allow a patient to chat with an AI bot that can respond to health-related inquiries, provide answers about their care plan, and escalate specific requests (like changes in appointments or treatment protocols) to their doctor.

Main Features

- Patient:
    - For simplicity, let us assume we have one patient (no authentication/authorization needed) that will use the app. You can either hardcode the patient data or use a db table with one patient. A patient can have the following attributes (feel free to add fields as necessary): First Name, Last Name, Date of Birth, Phone Number, Email, Medical Condition, Medication Regimen, Last Appointment DateTime, Next Appointment DateTime, Doctor's Name.
- Main View:
    - Once the app is launched, the main view should contain a chat box and should display the conversation history with date and time stamps for each interaction.
    - Patients can write their message and receive a response from the AI bot, like any modern chat application.
- AI Bot Functionality:
    - The bot should only respond to health-related topics such as:
        - General health and lifestyle inquiries
        - Questions about the patient's medical condition, medication regimen, diet, etc.
        - Requests from the patient to their doctor such as medication changes
    - The bot should filter out and ignore any unrelated, sensitive, or controversial topics.
    - Appointment and Treatment Protocol Requests:
        - If the patient makes a request to modify their appointment (e.g., "Can we reschedule the appointment to next Friday at 3 PM?"), the bot should:
            - Respond to the patient with something like, "I will convey your request to Dr. [Doctor's Name]."
            - Simultaneously, output a message saying something like, "Patient [Name] is requesting an appointment change from [current time] to

[requested time],” which should be displayed next to the chat box for review by the patient.

- Conversation History and Memory Optimization:
  - The bot should manage long conversations while optimizing memory usage to ensure efficient handling of ongoing dialogues without losing important information.
- Extract key entities from the conversation which the patient mentioned. For example, the patient's preference for appointment time, or any patient mention of a medication /diet / etc. for example if the patient says, ‘i am taking lisinopril twice a day’ then extract {medication: lisinopril, frequency: 2 times a day}.
- This extracted entities and values can be stored in knowledge graph and used in subsequent conversation (Bonus)

Bonus Points

- Use of Knowledge Graph and LLM:
  - Integrate a Knowledge Graph to dynamically query additional data about the patient (e.g., lab tests, doctor notes, weight, vital signs, medications).
- Multi-Agent System and Model Orchestrator:
  - Implement a multi-agent system that coordinates multiple models to handle different tasks within the chat.
- Conversation Summaries and Medical Insights:
  - Detect and output live conversation summaries and medical insights from ongoing conversations.
- LLM Agnostic Design and use of Langchain and Langraph (for LLM and RAG):
  - Ensure the application is LLM agnostic, so that different language models can be easily swapped out by setting environment variables (e.g., model name, API keys).

Technology Stack

- Backend: Django (required as FE/BE web framework).
- Langchain and Langgraph (for LLM and RAG)
- Database: PostgreSQL (required), Neo4j (optional).
- LLM Model: You can use any LLM model. For convenience, you may use Gemini, which offers free usage. Get your Gemini API key from: https://aistudio.google.com/app/apikey

Project Deliverables

- A functional prototype that meets the specified requirements.
- Bonus for including any additional features outlined in the Bonus Points section.
- Include a README file with clear instructions on:
  - Setting up the project locally.
  - Running the application.

    - Any assumptions made and how to use the key features.
- Git Repository: Share the code in a public GitHub repository and provide the link upon project completion.

Evaluation Criteria

- Code Structure: Is the code modular and maintainable?
- Functionality: Does the chat bot effectively manage health-related conversations and detect appointment/treatment requests?
- Memory Optimization: Is the bot designed to handle long conversations efficiently?
- Bonus Features: Are any of the bonus points (Knowledge Graph, multi-agent system, conversation summaries, etc.) implemented?
- Documentation: Is the README clear, and does it explain the setup and implementation choices?

Timeline

You are expected to complete this project within 7 to 10 days. Please share the GitHub repository link with your submission when completed.

Good luck! We look forward to reviewing your submission. Please submit your GitHub repository link to firas@dtxplus.com.

If you require more time due to existing commitments or would like to opt-out from consideration, kindly send an email to firas@dtxplus.com.