Assignment 2

Rapid Prototyping with and without Generative AI Tools

Artificial Intelligence for Business Decisions and Transformation- CSCN8030

Course Instructor: Dr Anasuya Bhima

Group 4: Symptomsphere

Haysam, Elamin, 8953681
Paula Ramirez, 8963215
Fasalu Kottaparambu, 8991782
Christo Pananjickal Baby, 8989796

Due: October 7, 2025

Problem Definition

Our project, Symptomshpere addresses the challenge of simulating realistic patient interactions for medical training and clinical rotations. The challenge medical students and medical colleges face today is that they are not getting enough patients for clinical rotations. Medical students often lack opportunities to practice clinical communication and diagnosis skills in varied, realistic scenarios. Traditional standardized patient programs are expensive and not scalable.

To solve this pain point, we are developing a Symptomsphere, an AI patient actor chatbot that can simulate conversations from a patient's point of view. This chatbot allows students to ask questions, take history, and interact as if they were talking to a real patient. We have three main AI use cases in our project. First one is to generate the new patients with AI from existing patient pool, second one is the AI which is the conversational AI used for chat, third one is the chat evaluation and feedback AI. For this rapid prototyping assignment, we are going ahead with the priority, conversational AI use case.

Conversational AI - Rapid Prototype Solution (With Generative AI Tools)

1. Research

We did some research with help of Chat GPT on how existing conversational agents and virtual patients are developed using Large Language Models (LLMs) like GPT. Generative AI model is very good at generating contextually rich, human-like responses, simulating roles (e.g. acting as a patient, teacher, or assistant) and reducing development time by generating code, prompts, and example dialogues. We decided to use OpenAI's GPT model through the API because of its strong conversational abilities and ease of integration with Python.

2. Design

The solution which we have designed right now is a web-based chat interface that allows medical students to interact with the chatbot. The chatbot uses a generative AI model (via API) to role-play as a patient with predefined symptoms.

Key Components:

- Frontend/UI: Streamlit web app for interactive chat
- Backend: Python code to send user messages + a structured "patient scenario" prompt to the GPT model
- Environment Variables: API keys securely stored using .env file
- Scenario Prompting: Hardcoded prompt to ensure the model consistently acts as a patient with headache symptoms

3. Prototype

Tools & Technologies:

- Python
- Streamlit
- OpenAI API
- python-dotenv for loading API keys
- git
- GitHub

Steps:

1. Store API key in .env in order to avoid hardcoding the key with code.
2. Use Python code to load the key and call the GPT model via API
3. Design a conversation loop between user and model
4. Launch Streamlit UI to simulate patient interaction in real-time

This allowed us to prototype the entire chatbot quickly, without training any models locally. The entire code was generated with a single prompt in Chat GPT.

4. Documentation & Rationale

- Why generative AI - generative AI significantly reduced development time. We didn't need to train a model or write a single line of code to make this application.
- Flexibility**:** We could easily modify the patient scenario by changing the system prompt which was the idea given by the Chat GPT.
- Quality of Responses**:** GPT produced coherent, context-aware replies suitable for medical history-taking simulations.

Conversational AI - Rapid Prototype Solution (Without Generative AI Tools)

1. **Research**

For the non-generative AI approach, our idea was to explore any open-source language models that can be downloaded and run locally so that we don't need a paid key like we had in the gen AI approach.

After some web search we found a smaller and good language model which is Phi-2 by Microsoft. Phi-2 is lightweight but a high-quality model for conversations. So, we decided to use Phi-2 and run it locally using transformers package. This approach eliminates dependency on external APIs and paid models.

2. **Design**

The design is very similar to which the generative AI proposed, but we replaces the API with a locally stored model:

Key Components:

- Model Storage: Downloaded the model using transformers and saved to models/phi-2/
- Loading Locally: Used AutoModelForCausalLM and AutoTokenizer with local_files_only=True
- UI: Streamlit-based chat interface
- Prompt: Same structured patient scenario promp

3. **Prototype**

Tools & Technologies:

- Python
- Streamlit
- Hugging Face Transformers
- git
- GitHub

Steps:

1. Download the Phi-2 model locally using a separate Python script (commented at the beginning of main.py)
2. Load the model from "without_gen_ai/models/phi-2" in main.py
3. Implement a conversation loop manually using model.generate()
4. Run Streamlit to launch the chatbot interface

This allowed us to replicate the patient actor chatbot completely offline, without API keys. Some files of the model is not uploaded in git due to upload size restrictions from GitHub. The 3 files in the model are about 14GB.

4. Documentation & Rationale

- Why Local Model - Using a local model avoids API costs, privacy concerns, and internet dependency.
- Control: We could run and modify everything locally and do as may experiments we want without worrying about the API expense.
- Limitations: Response quality is lower than GPT, and generation is very slower on CPU.

Comparison

Aspect	**With Gen AI**	**Without Gen AI**
Development time	Very fast – less than 5 minutes	very slow, about 3 hours. Had to go through stack overflow, medium.com to figure out many things.
Response time	Very good – feels like normal text chat between to people	Very slow – average response time was 34 sec on my pc
Cost	Had to pay $12 for API token	Free for ever
Privacy	Data is sent to third party	Data is never shared
Hardware needs	None since the heavy lifting is done on cloud	Uses good amount of CPU and memory
Flexibility	Easy scenario change with prompts	Easy scenario change with prompts
Launch Speed	Very fast	Takes around 14 seconds to initialize the model

Github Link: https://github.com/Christo-Conestoga/ai_for_business_assignment_2.git