Generative AI-based Medical Chatbot Application-Final Project Submission

This document describes the design, implementation, and testing of a Generative AI-based medical chatbot developed to support clinicians in interpreting clinical text and answering natural language questions. The chatbot is implemented using a Gradio web interface and a lightweight pre-trained biomedical language model.

Task 1: Gradio Web Application Interface

A chatbot-style web interface was developed using the Gradio framework. The interface includes three clearly labeled text boxes: one for entering a medical report, one for entering a clinical question, and one for displaying the AI-generated answer. A button labeled “Get the Answer” triggers the chatbot response. The interface renders correctly with all components visible and easy to use.

```
import gradio as gr

report_input = gr.Textbox(lines=10, label="Enter Medical Report")
question_input = gr.Textbox(lines=2, label="Enter Your Question")
answer_output = gr.Textbox(lines=4, label="AI Answer")

iface = gr.Interface(
    fn=lambda r, q: "Test output",
    inputs=[report_input, question_input],
    outputs=answer_output,
    title="Medical Chatbot"
)

iface.launch()
```

Figure 1

Task 2: Lightweight Language Model Integration

To enable natural language understanding and generation, a lightweight sequence-to-sequence language model was integrated using the Hugging Face Transformers library. The google/flan-t5-small model was selected due to its efficiency and strong performance on question-answering tasks. The model and tokenizer were loaded using AutoTokenizer and AutoModelForSeq2SeqLM, and a text2text-generation pipeline was created. A test prompt confirmed that the model generates concise and relevant medical answers.

```python
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM, pipeline

model_name = "google/flan-t5-small"

tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)

nlp_pipeline = pipeline(
    "text2text-generation",
    model=model,
    tokenizer=tokenizer
)

# Test
test_prompt = "question: What is the diagnosis? context: The patient shows signs
result = nlp_pipeline(test_prompt)

print(result[0]["generated_text"])
```

Figure 2

Task 3: Answering Function Implementation

An answering function was implemented to connect user inputs from the Gradio interface to the language model. The function accepts a medical report and a clinical question, constructs a structured prompt in the format “question: … context: …”, and generates a response using the integrated model. The function also handles missing inputs gracefully by prompting the user to provide both required fields. Testing with a sample clinical report confirmed accurate and meaningful responses.

```python
def generate_answer(report, question):
    if not report or not question:
        return "Please enter both a medical report and a question."

    prompt = f"question: {question} context: {report}"
    result = nlp_pipeline(prompt)

    return result[0]["generated_text"]
```

Figure 3

Medical Report:

```
The patient presents with persistent cough and fever.
Chest X-ray shows signs of pneumonia.
Antibiotics have been prescribed.
```

Figure 4

Question:

```
What is the diagnosis?
```

Task 4: Application Testing and Validation

The complete chatbot workflow was tested by entering a sample medical report and a related question into the Gradio interface. Upon clicking the “Get the Answer” button, the chatbot successfully generated a clear and clinically relevant response. This confirms that the application functions correctly from input to output and is suitable for assisting clinicians in quickly extracting key information from medical text.

```
iface = gr.Interface(
    fn=generate_answer,
    inputs=[report_input, question_input],
    outputs=answer_output,
    title="Medical Chatbot"
)

iface.launch()
```

Figure 6

Overall, this project demonstrates the effective use of generative AI for clinical text interpretation through a simple, user-friendly interface. The chatbot provides reliable responses while maintaining usability and performance, making it a useful prototype for clinical decision support.