Coding Assignment

AI for Mental Healthcare Experts: The Manashi Project

Time: 72 hrs

December 28, 2025

Our Mission

You are applying to work on **Manashi**, a specialized AI coTherapist designed not to replace clinicians, but to augment their capabilities. Unlike general-purpose chatbots that face patients directly, Manashi operates as a **Clinical Co-Pilot** for mental healthcare experts (psychiatrists, psychologists, and counsellors). Your role involves building systems that support high-stakes workflows such as clinical documentation, session summarization, symptom tracking, and safety-aware reasoning. *Every algorithmic decision you make must prioritize expert control, interpretability, and patient safety.* We are looking for researchers who can think deeply about the trade-offs between model performance and clinical risk.

Guidelines on AI Usage (Important)

We recognize that Large Language Models (LLMs) and coding assistants are now integral to modern software engineering. Therefore, the use of tools like ChatGPT, Claude, GitHub Copilot, or Cursor is **permitted**.

Transparency Requirement

If you use AI tools to generate code, brainstorm ideas, or debug errors, you must be transparent about it. We want to evaluate your **prompt engineering skills** and your ability to verify AI-generated outputs.

Requirement: If AI tools are used, you must attach the chat links, and the LLM outputs should either match or atleast your response should be inspired from it. Blindly copying code without understanding or verification will result in disqualification.

Timeline and Expectations

This assignment is designed to be completed within **72 hours**, with an expected focused effort of approximately **10 to 12 hours**. We value the depth of your reasoning and your approach to failure handling more than a perfectly polished final product.

Part A: The Essay: Expert-Centric Design

Write a short position paper (approx. 350 words) arguing that general-purpose LLMs are insufficient for clinical mental healthcare. Address technical limitations such as context windows, hallucinations, and domain adaptability. Propose a mathematical framework or evaluation metric that penalizes critical errors (e.g., missed suicide flags) significantly more than minor ones. Even if it happened, suggest how to visualize model uncertainty in the app UI to assist clinicians without causing cognitive overload.

*Note: Kind request to put **your** thoughts, not any random GPT's.*

Part B: Session-to-Note Transformation

Design a deterministic data processing pipeline. Synthesize five (Depression, Anxiety, Schizophrenia, Bipolar disorder, OCD) raw therapy session *transcript* with *timestamps* and *speaker_labels* (Therapist/Client). Your task is to design an algorithm (in Python) that extracts structured clinical insights **without** training/using a new machine learning model.

Input Example:

```
#talk1_depression
transcript = [
  {"time": "10:00", "speaker": "Client", "text": "I haven't slept in three days."},
  {"time": "10:01", "speaker": "Therapist", "text": "That sounds exhausting. Are you feeling anxious?"},
  {"time": "10:01", "speaker": "Client", "text": "Yes, my heart keeps racing."}
]
```

Required Output Structure: Your pipeline should map the input to a JSON object containing:

1. **Subjective Observations:** What the patient reports (e.g., "Insomnia", "Palpitations").
2. **Objective Facts:** Verifiable data points (e.g., "Sleepless for 72 hours").
3. **Risk Flags:** keywords indicating immediate harm, etc.

Create a session compression mechanism that helps a mental healthcare expert quickly understand what matters in a session. This is **not** text summarisation.

Fixed Output Format

Your output must strictly follow this structure:

```
{
  "observations": [],
  "objective_facts":[],
  "emotional_trajectory": [],
```

```
  "risk_flags": [],
  "open_loops_for_next_session": []
}
```

Deliverables: Provide the algo for this transformation. Explain your choice of data structures and how you handle edge cases, such as ambiguous speaker labels or overlapping speech.

Part C: Model Dissection

Use `LLaMA-3.2-1B` (or a similar small open-source model, whichever you are comforable with) as the reference architecture.

C1. Architecture Analysis

Describe the information flow of the LLaMA architecture during a forward pass. Do not just list layers; explain the transformation of the data throughout the layers.

C2. Controlled Intervention Task

Your goal is to modify the model's behavior at inference time **without fine-tuning**. We want to reduce the model's tendency to produce confident-sounding clinical interpretations when the evidence is weak (hallucination dampening).

The Task: Choose an internal component (e.g., attention weights, MLP activation output, or the final logit distribution) and implement a "gate" or "filter." For instance, you might detect high entropy in the attention heads and suppress the generation of specific medical tokens.

C3. PyTorch Implementation

Write a minimal, executable PyTorch script that loads the model and applies your intervention. You may use the Hugging Face `transformers` library.

Code Requirement: Your code must demonstrate how to hook into the model's forward pass. Below is a skeleton to get you started. You need to fill in the logic for the `ClinicalSafetyHook`.

```python
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer

class ClinicalSafetyHook:
    def __init__(self, threshold=0.8):
        self.threshold = threshold

    def __call__(self, module, input, output):
        # TODO: Inspect the 'output' tensor.
        # If the activation magnitude or entropy suggests
        # unwarranted confidence, suppress specific dimensions.

        # Example logic (conceptual):
```

```
        # modified_output = torch.clamp(output, max=self.threshold)
        # return modified_output
        return output

# 1. Load Model
model_id = "meta-llama/Llama-3.2-1B-Instruct"
model = AutoModelForCausalLM.from_pretrained(model_id)

# 2. Register the Hook
# Identify a specific layer to intervene on (e.g., the last MLP layer)
target_layer = model.model.layers[-1].mlp
handle = target_layer.register_forward_hook(ClinicalSafetyHook(threshold
    =0.5))

# 3. Run Forward Pass
dummy_input = torch.tensor([[101, 202, 303]])
output = model(dummy_input)

print("Forward pass completed with intervention.")
```

Submission Format

Please submit your work as a single Notebook file named `YourName_Manashi_Assignment.ipynb`. The structure should be as follows:

1. Text cells containing your Part A essay, failure mode analysis.
2. Coding cells containing:
    - `pipeline` (Part B logic)
    - `model_intervention` (Part C PyTorch code)

Answer the following three questions in a text cell, one line each:

- Which part of this assignment challenged your understanding the most?
- If you had one more week to work on this, how would you improve your solutions?
- In your opinion, what is one aspect of the therapeutic relationship that can **never** be automated?