Tava Health x Gauntlet AI Engineer Challenge - AI Mental Health Treatment Plans

AI-Assisted Mental Health Treatment Plans

1. Context

Tava Health is a digital mental health company that connects people with licensed therapists through modern, accessible technology. In real-world care, therapists create **treatment plans** that capture a client's goals, diagnoses (when appropriate), interventions, and progress over time.

Creating and updating these plans is:

- Time-consuming
- Often inconsistent between providers
- Usually written in language that is either too clinical for clients or too simplistic for clinicians

Your challenge is to build an AI-powered, full-stack experience that takes therapy session transcripts (audio → text, at minimum) and helps generate high-quality, personalized treatment plans that can be shared between therapist and client.

---

2. High-Level Problem

Design and implement a full-stack application that:

1. Accepts session data (at minimum, an audio transcript; optionally audio/video files).
2. Uses AI to parse and understand the session content.
3. Generates two tailored treatment plan views:

    - A therapist-facing treatment plan (clinical detail, ICD language, interventions, risk factors, etc.).
    - A client-facing treatment plan (plain-language, strengths-based, motivational).
4. Provides a web UI where both therapist and client can log in and view their respective plans.

You should think about:

- How to structure mental health data (goals, interventions, progress, risk).
- How to keep the client at the center: accessible language, empowerment.
- How to make the therapist's life easier: drafts, editing, quick updates, version history.

---

3. Core Requirements (Must-Haves)

Your solution must include at least the following:

3.1 Full-Stack Application

- A working backend (API) and frontend UI.
- Persistent data store (e.g., Postgres, MongoDB, SQLite) for users, sessions, and treatment plans.
- Use AI models (via an API or local model) as a core part of the treatment plan generation flow.

Tech stack is your choice. Examples:

- TypeScript + React/Next.js + Node/Express/NestJS
- Python + FastAPI/Flask + React/Vue/Svelte
- Anything else modern and reasonable

3.2 Session Input (Transcripts Required)

- Accept audio transcripts of mental health sessions as the primary input.
    - You can simulate this by:

    - Uploading a `.txt` file
    - Pasting text into a form
    - Or integrating a speech-to-text API (if you'd like)
- Each transcript should be associated with:
    - A client
    - A therapist
    - A session (date/time, basic metadata)

> Note: Use only synthetic or fully de-identified transcripts. No real patient data.

3.3 AI-Generated Treatment Plans

- Use an AI model to parse transcripts and generate a structured treatment plan with fields like:
    - Presenting concerns
    - Clinical impressions (optional)
    - Goals (short-term, long-term)
    - Interventions / approaches (e.g., CBT, ACT, DBT, etc.)
    - Homework / between-session actions
    - Strengths and protective factors
    - Risk indicators / red flags (if present in transcript)
- Generate two versions of the plan from the same underlying data:
    - Therapist view: More clinical, structured, potentially documentation-ready.
    - Client view: Plain-language, supportive, action-oriented.

3.4 Role-Based UI (Therapist vs Client)

- Basic auth or session handling (does not need to be production-grade, but should demonstrate the idea).
- Therapist dashboard:

    - View clients and their sessions
    - Upload / select a transcript
    - Trigger plan generation
    - Review & edit the therapist-facing plan
- Client dashboard:
    - View their client-facing plan
    - See goals, next steps, and homework in clear language

3.5 Documentation & DX

- A README that includes:
    - Tech stack and architecture overview
    - How to run the project locally
    - How the AI integration works (prompting strategy, model choice)
    - Any limitations and future ideas
- At least a few automated tests (unit or integration) around a core piece of logic (e.g., parsing model output into structured objects).

---

4. Recommended Requirements (Strongly Encouraged)

These aren't strictly required but will make your submission stand out:

4.1 Treatment Plan Lifecycle

- Ability to update an existing treatment plan when a new session transcript is added.
- Maintain plan history / versioning so therapists can see how the plan evolved over time.
- Show a simple timeline or "last updated" metadata.

4.2 Session Summaries

- Generate a session summary using AI that is:

- Clinical in the therapist view (e.g., for note-taking support).
- Friendly and encouraging in the client view (e.g., “Here’s what we worked on together today.”)

4.3 Safety & Ethics Awareness

- Basic checks that detect crisis language or high-risk statements (e.g., self-harm, harm to others) in transcripts and flag them in the therapist view.
- Clear disclaimers in the UI that this is not a substitute for clinical judgment.

---

5. Bonus / Stretch Objectives

These are intentionally challenging; use them to flex your creativity and AI engineering skills. You do not need to do all of them.

5.1 Model Evaluation & Experimentation

- Implement a small evaluation harness to compare two or more models or prompts:
  - For example, measure which model produces more structured outputs or fewer parsing errors.
  - Or run automatic checks on reading level for client-facing plans.
- Visualize evaluation results in a small dashboard or report.

5.2 Therapist-Guided Model “Training”

- Allow therapists to adjust how the model behaves over time, for example:
  - Preferences for certain therapeutic modalities (CBT vs ACT vs DBT).
  - Example “golden” treatment plans they like.
- Use this feedback to:
  - Adapt prompts (prompt engineering + stored preferences), or
  - Maintain a small few-shot library per therapist, or
  - (Ambitious) Fine-tune a small open-source model with synthetic data.

5.3 Plan Editing & Multi-Source Updates

- If a user already has a treatment plan:
  - Allow manual editing of any section.
  - Allow regeneration or updating from:
    - A new audio/audio transcript
    - A video transcript (if implemented)
  - Show diffs when updating: what changed vs the previous version.

5.4 Video Support

- If you start with audio + transcript, add:
  - A video upload and playback experience for sessions.
  - Optional: use video metadata (e.g., timestamps, captions) in your pipeline.

5.5 Mobile Experience

- A responsive, mobile-friendly UI for clients on their phones.
- Extra credit for:
  - A dedicated mobile layout or PWA
  - Mobile-first flows for viewing goals and homework

5.6 Advanced AI Features

- Multi-language support (e.g., detect language and generate plan in the same language).
- Explainability: let the therapist click on a plan element and see which transcript snippets influenced it.
- “What-if” or “copilot” features:
  - Therapist can ask the system: “Suggest 3 alternative interventions for this goal.”
  - Client can ask: “Explain this goal in simpler terms.”

---

6. Data & Privacy (Important)

Even though this is a challenge project, treat it like a mental health product:

- No real PHI (Protected Health Information). Use only mock or synthetic data.
- No real Tava Health production data should be used.
- Add visible privacy disclaimers in the UI to show you're thinking about ethics and safety.

---

7. Evaluation Criteria

We will look at:

1. Problem Understanding
    - Do you demonstrate understanding of mental health workflows and treatment planning?
    - Are therapist and client needs reflected in the product design?
2. AI Use & Quality
    - Is AI clearly central to the experience (not just a single "magic button")?
    - Is the prompt/model design thoughtful and documented?
    - Are outputs structured, useful, and reasonably safe?
3. Product & UX
    - Does the app feel coherent and purposeful?
    - Is the UI usable for both therapist and client personas?
    - Are complex flows (plan creation, updates, viewing history) handled gracefully?
4. Technical Execution
    - Clean architecture and code organization
    - Testing on at least some critical paths
    - Proper handling of model errors, timeouts, or malformed outputs
5. Ambition & Creativity

- Did you go beyond the basics in a meaningful way?
- Did you explore interesting AI features, evaluation, or personalization?
- Are there clear ideas for how this could evolve into a production-ready system?

---

8. Deliverables

Your submission should include:

1. Running application (local or hosted)
   - Instructions in the README to run locally (and a demo link if hosted).
2. Source code repository
   - Clear project structure
   - Brief comments where helpful
3. Documentation
   - README with setup, stack, and architecture
   - Short write-up (~1–2 pages in the repo or README) covering:
     - AI system design (models, prompts, data flow)
     - Key product decisions and tradeoffs
     - What you'd build next with more time
4. (Optional but Appreciated) Demo
   - Short Loom / screen recording walkthrough of the main flows.

---

9. Getting Started (Suggested)

You are free to approach this however you like. Here is one possible path:

1. Define your data models: User, Therapist, Client, Session, Transcript, TreatmentPlan, TreatmentPlanVersion.
2. Build basic auth and two dashboards (therapist, client).
3. Create a simple "Upload or paste transcript → Generate plan" flow.

4. Implement structured AI output: goals, interventions, homework, etc.
5. Split it into therapist-facing and client-facing views.
6. Add editing, updates from new sessions, and history.
7. Choose at least one bonus objective and go deep on it.

Push the edges. Think like someone building the future of digital mental health.

Good luck, we’re excited to see what you create.