Ambr Technical Assessment - Full-Stack Engineer

Overview

Build a tool that analyzes meeting transcripts using AI and presents actionable insights to users.

This is designed to take 3-4 hours, though you're welcome to spend more time if you want to add polish. We want to see:

- How you integrate AI into a product
- Your API design and error handling
- Your UI/UX instincts
- How you handle ambiguity and make product decisions

Important: We expect and encourage you to use AI coding tools (Cursor, Claude Code, Copilot, etc.) to work efficiently. This reflects how we actually work.

Core Requirements

Backend

Build an API that:

- **Accepts transcript text** via a POST endpoint
- **Calls an LLM** (of your choice) to extract:
  - Action items with owners/deadlines
  - Key decisions that were made
  - Overall meeting sentiment/tone
- **Stores the analysis results** in a Postgres database
- Returns structured, typed responses

Frontend

Build a simple React interface that:

- Lets users paste/submit a meeting transcript
- Shows the AI-generated insights in a clear, scannable way
- Handles loading and error states appropriately

Data Layer

- Use **Prisma** as your ORM with **Postgres**
- Design your schema to store transcripts and analysis results
- Make it easy to retrieve past analyses

Required Tech Stack

Use the following to match our production stack:

- **API:** ts-rest (or tRPC if you prefer)
- **Validation:** Zod schemas for all inputs/outputs
- **Database:** Prisma + PostgreSQL
- **Frontend:** React (your choice of framework - Next.js, Vite, etc.)
- **Data fetching:** React Query or similar
- **LLM:** Your choice - if you need an LLM API key, we can provide one for Vertex or Anthropic console - just let us know.
- **Language:** TypeScript throughout

Feel free to add any other libraries that make sense.

What We're Looking For

We're evaluating:

✅ **Does it work?** - Core functionality should be solid and reliable
✅ **Code quality** - Clean, typed, well-structured code
✅ **Product thinking** - Thoughtful UX decisions, useful insights presentation
✅ **Error handling** - What happens when the AI fails? When input is too long?
✅ **Decision-making** - How did you handle the ambiguous parts?

Intentionally Ambiguous Areas

These are *your* decisions to make:

- How should insights be displayed to be most useful?
- What's the right prompt strategy for the LLM?
- How do you handle transcripts that are too long?
- What makes a good vs. bad meeting analysis?
- What additional features would make this genuinely useful?

Good candidates will make thoughtful choices and maybe add a nice touch or two that we didn't specify.

Submission

1. **Push your code to GitHub** (public or private repo - if private, add jamiewooduk & huzaifahj)
2. **Include a README.md** with:
    - Setup instructions (env vars, DB setup, etc.)
    - How to run the project
    - Any decisions you made and why
    - What you'd improve with more time
3. **Make sure it runs** - we should be able to clone, setup, and run it

Note on API keys: Don't commit API keys. Use environment variables and document what's needed in the README.

Evaluation Timeframe

Once you submit, we'll review within 2-3 business days and get back to you with next steps.

Questions? Email jamie@ambr.ai and huzaifah@ambr.ai - we're happy to clarify anything about the task or tech stack.

Good luck! We're excited to see what you build.

Sample Meeting Transcript (for testing)

```
None
Hey team, thanks for joining. I wanted to kick off this project
planning session for Q1.
Sarah, you mentioned last week you'd have the API design doc
ready - do you have an update on that?
```

```
Sarah: Yes, I finished it yesterday. The main decision we need to
make is whether to use REST or GraphQL. I'm leaning toward REST
for simplicity, but I want to hear thoughts.

Mike: I agree with REST. We can always add GraphQL later if we
need it. When can you start implementation?

Sarah: I can start next Monday. I'll need about 2 weeks for the
initial version.

Great, let's move forward with REST. Mike, can you handle the
frontend integration once Sarah's API is ready?

Mike: Yep, I'll block out the last week of January for that.

Perfect. One more thing - we need to nail down our database
choice. Are we going with Postgres or MongoDB?

Sarah: I vote Postgres. We'll need transactions and relational
data.

Mike: Agreed on Postgres.

Alright, Postgres it is. Sarah, can you set up the initial schema
by end of week?

Sarah: Will do.

Awesome. Let's reconvene next Wednesday to check progress. Thanks
everyone!
```

We want to respect your time - we've designed this assessment to be completable in 3-4 hours while giving you room to showcase your skills.