Project: A fully private, locally run, convo transcription tool for online calls or in-person discussions. Analyze conversations and have a visual map of convo topics, live. Your conversations never leave your computer.

Use cases: Need AI analysis of work calls that you don't want to send off to other company's servers. Having a convoluted argument with your significant other and you want AI analysis and to keep track of topics for your sensitive discussion. Maybe even involve an AI therapist.

1. High-Level System

Four services:

- Capture + transcription (ASR)
- Storage + transcript search
- LLM intelligence (via Parallax)
- Realtime visualization UI
  Parallax handles summaries, Q&A, topic detection. ASR runs locally (Whisper / faster-whisper).

2. Pipeline Step-by-Step

2.1 Audio Capture

Desktop app: OS microphone capture; optional system-audio for Zoom/Meet.
Web app: Browser MediaRecorder streaming audio chunks via WebSocket.
Output: small audio chunks with timestamps + optional speaker ID.

2.2 Local Transcription (Privacy)

Run Whisper / faster-whisper locally. Apply VAD, then transcribe to text + timestamps + optional speakers. Store immediately in DB.
Example:

```
{ "start": 12.4, "end": 15.9, "speaker": "A", "text": "We should talk about the hiring plan next quarter." }
```

2.3 Topic Detection & Segmentation (Parallax)

Streaming topic labeler: Every N seconds or K utterances, send last 1–2 minutes to Parallax → “What is the current topic?”. Maintain `current_topic_id`.
Topic switch detection: Prompt model with previous topic + latest text → decide if topic changed. If yes, close segment + open new one.
Example:

```
{ "topic_id": "T5", "label": "Budget for Q4 marketing", "start":
420.0, "end": 735.0 }
```

Branch / thread detection: Periodic prompt to generate a topic tree with time ranges.
Example:

```
{ "root_topic": "Weekly sync",
  "nodes": [
    { "id": "T1", "label": "Product roadmap", "parent": null, "start":
0, "end": 600 },
    { "id": "T2", "label": "Roadmap – mobile app", "parent": "T1",
"start": 120, "end": 300 }
  ]
}
```

LLM calls run through Parallax’s OpenAI-compatible API.

2.4 Summary & Q&A

Segment summaries: For each segment: 2–3 bullet summary + 1 sentence of action items.
Conversation summary: Structured summary including goals, decisions, open questions, action items.
Q&A (RAG): Store transcript; use embeddings + vector search (Qdrant) to fetch snippets; feed snippets + query to Parallax; answer strictly from retrieved text.

3. Topic Map Visualization (Live)

Timeline: Time on X-axis; colored topic segments; update on topic switch; moving cursor.
Branching map: Nodes = topics; parent/child from structure JSON; layouts: radial, vertical tree, etc.
Live updates: WebSocket events for new utterances, topic switches, structure updates. Frontend updates transcript pane, timeline, and graph.
Example stack: React + D3/VisX/React-Flow + Tailwind.

4. Privacy Guarantees

Everything runs locally or self-hosted: local Whisper, Parallax LLM, local DB/encrypted disk.
Optional modes: delete raw audio; keep only text/summaries; fully in-memory ephemeral mode.
UI messaging: “Data never leaves your devices.”

5. How Parallax Fits

Run Parallax on a local GPU machine. Load small model for topic detection + larger one for summaries/Q&A.
API usage:

```
base_url = http://<parallax-host>:<port>/v1
```

Use `/chat/completions` with JSON.
Minimal hackathon slice: Single GPU machine; web app with start/stop recording, live transcript, live topic timeline, end-of-call topic map, Q&A box.