🩺 Project Proposal: Medeo plus ✨

👥 Team Members

Name	UTORid	Email
Mianli Wang	`wangm246`	`mianli.wang@mail.utoronto.ca`
Steve Nguyen	`nguy3671`	`st.nguyen@mail.utoronto.ca`

🚀 Project Overview

Medeo Plus is a modern web application designed to streamline communication between patients and providers while enhancing mental health support. Within a single, secure workspace, users can exchange **messages** and join **video calls** with their clinicians. The platform is built around two core features:

- **AI-Powered Messaging**: Patients and providers can engage in one-on-one messaging. In addition, users can interact with an AI assistant capable of delivering personalized, evidence-based responses tailored to the user's medical history and uploaded files. The assistant also generates concise summaries to aid clinicians, acting as a stand-in when doctors or counselors are unavailable.
- **Live Video with Real-Time Transcription**: The video interface is embedded using a **Daily.co** iframe. Browser-captured audio is streamed via **WebSocket** to an Express server, which relays it to **AssemblyAI** for real-time transcription and translation. These captions, similar in function to services like **Mikata Health** or **Scribeberry**, support clinicians during consultations and are delivered to the user's screen through a **Socket.IO** overlay.

🏗️ System Architecture

Paste the Mermaid snippet below into the [Mermaid Live Editor]() for a visual diagram.

代码段

```
---
config:
  layout: elk
  theme: neo-dark
  look: neo
---
flowchart TD
  subgraph client["🖥️ User Client"]
    User["💻 Browser"]
  end

  subgraph nuxt["✨ Nuxt 3 SPA (3000)"]
    NuxtNode["OAuth ↔ Google<br/>Stripe Checkout<br/>Video UI (Daily iframe)"]
  end

  subgraph nodeSvc["🟢 Node.js Ecosystem"]
```

```
    Express["🚀 Express.js API (8080)<br/>/auth /payments /files /rag"]
    SocketIO["🔌 Socket.IO<br/>Real-time channel"]
    BullMQ["🐎 BullMQ Worker<br/>Async jobs<br/>(transcribe → embed)"]
    Prisma["📦 Prisma ORM"]
  end

  subgraph pySvc["🐍 Python AI Service"]
    LightRAG["🧠 LightRAG (FastAPI 5000)<br/>Hybrid KG + vector retrieval"]
  end

  subgraph core["🛠 Core Services"]
    Postgres["🐘 PostgreSQL"]
    Redis["🧰 Redis<br/>BullMQ queue"]
  end

  subgraph infra["☁ Docker Compose / Nginx"]
    Nginx["⚙ Nginx<br/>SSL / Reverse Proxy"]
    nodeSvc
    pySvc
    core
  end

  subgraph third["🌐 Third-Party Cloud"]
    Supa["📦 Supabase Storage<br/>S3-compatible (JWT RLS)"]
    Qdrant["🧠 Qdrant Cloud"]
    Stripe["💳 Stripe Checkout"]
    OpenAI["🤖 OpenAI API"]
    AssemblyAI["🎙 AssemblyAI"]
    Daily["🎥 Daily.co"]
  end

  %% client paths
  User -- "HTTPS" --> NuxtNode
  NuxtNode -- "HTTPS (Nginx)" --> Express
  NuxtNode -- "WSS (Nginx)" --> SocketIO
  NuxtNode -- "iframe" --> Daily

  %% node paths
  Express -- "Prisma" --> Postgres
  Express -- "Upload metadata" --> Supa
  Express -- "REST /rag/query" --> LightRAG
  Express -- "Add job" --> BullMQ
  Express -- "Webhook" --> Stripe
  BullMQ -- "R/W" --> Redis

  %% py paths
  LightRAG -- "Query vectors" --> Qdrant
  LightRAG -- "Embeddings / LLM" --> OpenAI

  %% transcription
  Express -- "WS proxy" --> AssemblyAI
```

🛠️ Tech Stack

Layer	Technology
🖼️ **Frontend**	Nuxt 3 (Vue 3) • TypeScript • TailwindCSS
🎨 **UI Kit**	DaisyUI
🚪 **Backend API Gateway**	Express.js (TypeScript) _
💾 **ORM**	Prisma (PostgreSQL adapter)
⚡ **Real-time**	Socket.IO
🔐 **Auth (OAuth 2.0)**	Supabase OAuth (Google)
💳 **Payments**	Stripe Checkout (test)
🧠 **LLM / RAG**	LightRAG (FastAPI) • OpenAI GPT-4o • LangChain.js *(optional post-processing)*
🌌 **Vector DB**	Qdrant Cloud *(optional optimization)*
🔄 **Queue**	BullMQ + Redis *(optional optimization)*
📹 **Video**	Daily.co
🎤 **Speech-to-Text**	AssemblyAI Streaming API _
🐳 **Deployment**	DigitalOcean VM • Docker Compose • Nginx
🚀 **DevOps**	GitHub Actions (lint / CI)

📅 Project Milestones

Alpha Version

* Architecture Validation: Achieve a stable local launch of all services using Docker Compose.

 * Debug the inter-service communication between Express and FastAPI.

 * Basic Features: Build the foundational UI and APIs for the Messages, Appointments, and Documents modules.

Beta Version

* LightRAG Implementation: Complete the RAG data indexing and retrieval pipeline, enabling personalized AI conversations.

 * Feature Completion: Integrate the real-time video transcription/translation feature and deploy the full application to a DigitalOcean VM.

 * Core Workflow: Implement the complete user flow from OAuth registration to a successful Stripe subscription payment.

Final Version

* Optimization & Bug Fixes: Resolve all identified bugs based on beta testing feedback. Optimize RAG retrieval efficiency and front-end performance.

 * Security Hardening: Conduct a thorough review of all authentication, payment, and data-handling processes.

 * Documentation & Submission: Finalize all code and documentation for submission to Gradescope.

⚖️ Legal & Ethical

*This academic prototype is **not** a certified medical device. All AI output is informational only and must not replace professional advice.*

- No real PHI or live payment credentials should be used.
- Secrets are injected **only** via GitHub Secrets or Docker secrets—no keys are committed to the repo.