

AI Agent Engr Task 1: AI Companion Video Call & Streaming

Overview

Develop an interactive web application enabling users to select from multiple AI companions/avatars and initiate real-time, peer-to-peer video calls. The platform serves as a foundational building block towards creating human-like, multi-agent conversational AI tutors integrated with real-time video and knowledgeable assistance.

Features should include:

- **Companion selection**: Users browse a list of avatars/AI personas — images, metadata, and Voice (ElevenLabs voice ID) included (fetched from a backend proxy endpoint).
- **WebRTC-based two-way video**: Browser → browser streaming for user←>companion call using negotiated peer connections (not server-proxied media).
- **Chat panel**: Instant messages, sent via the same signaling WebSocket (or fallback REST).
- **Call controls**: Mute mic, toggle camera, call timer, captions option.

Frontend: Next.js + TypeScript

Backend: Python (FastAPI), LangMem, Google Gemini, Redis

Code must be built from scratch during hackathon.

Backend API Specification

Method	Endpoint	Description	Response Example
POST	`/api/video/rooms`	Create new video room	`{ roomId, companionId, userId, expiresAt }`
GET	`/api/video/rooms/:roomId`	Fetch or validate room info	`{ roomId, status: 'active'
GET	`/api/webrtc/config`	Provide ICE server configuration	`{ iceServers: [{ urls: ['stun:stun.l.google.com:19302'] }, { urls: 'turn:global.turn.twilio.com:3478', username: '...', credential: '...' }] }`
GET	`/api/companions`	List available companions with images and metadata	Should proxy from `https://persona-fetcher-api.up.railway.app/personas` which returns all companion details including voice IDs, notably voice ID of eleven lab)
POST	`/api/video/recordings` (optional)	Upload recorded session video file	`{ recordingId, roomId, url }`

WebSocket Signaling (Socket.IO)

- Endpoint: `ws://<host>/signal`
- Events:
  - `join` : `{ roomId, userId, role }`
  - `offer` : `{ roomId, from, sdp }`
  - `answer` : `{ roomId, from, sdp }`
  - `candidate` : `{ roomId, from, candidate }`

- `leave`: `{ roomId, userId }`
- `end`: `{ roomId, reason? }`

Frontend Contracts

Component: `VideoCallModal.tsx`

Props:

- `roomId: string`
- `userId: string`
- `companion: { id: string; name: string; avatarUrl: string }`
- `onEnd(recordingBlob?: Blob): void`
- `onMessage?(msg: { from: 'user' | 'companion'; text: string }): void` (optional)
- `captions?: boolean` (optional)

Hook: `useWebRTC.ts`

Methods:

- `createRoom(): Promise<{ roomId: string }>`
- `joinRoom(roomId: string, role: 'user' | 'companion'): Promise<void>`
- `startLocalMedia(constraints?: MediaStreamConstraints): Promise<MediaStream>`
- `toggleMic(enabled: boolean): void`
- `toggleCamera(enabled: boolean): void`
- `startRecording(): void`
- `stopRecording(): Promise<Blob>`
- `endCall(): Promise<void>`

Chat

- Real-time chat via WebSocket or fallback REST POST to `/api/chat/messages` with `{ roomId, from, text, ts }`

Infrastructure and Provider Recommendations

- Use public STUN server `stun:stun.l.google.com:19302`
- Recommend TURN services for connectivity (Twilio, LiveKit, Agora)
- Optional: Managed SFU solutions allowed with justification

Deliverables Checklist

- **README.md:** setup instructions, `.env.example`, scripts, TURN server notes, demo steps
- **docs/architecture.md:** sequence diagrams (signaling, media flow); design decisions (TURN/SFU usage)
- **Frontend repo:**
  - `components/VideoCallModal.tsx`
  - `hooks/useWebRTC.ts`
  - `pages/companions/index.tsx` (uses your `/api/companions`)
- **Backend repo:**
  - FastAPI server with REST and WebSocket handlers
  - Companion proxy API for `/api/companions` forwarding data from external `https://persona-fetcher-api.up.railway.app/personas`
  - Optional recording upload endpoint

Important Usage Notes

- Participants must use **your** `/api/companions` backend endpoint to get companion data. This endpoint **must** proxy the external API at `https://persona-fetcher-api.up.railway.app/personas`.
- Companion data includes images, metadata, and voice IDs with voice ID ("Eleven Labs") as a provided voice option.

Tools to Explore & Integrate

For building the **AI Companion with real-time video call and streaming capabilities**, the following tools can be leveraged:

- **Akool** – Offers generative AI capabilities for personalized video avatars and dynamic content creation.
- **D-ID** – Specializes in creating AI-driven talking avatars, enabling face-to-face interactions in real time.
- **Anam** – Provides infrastructure for voice and avatar-based AI companions, focused on conversational experiences.
- **Tavus** – Enables scalable personalized video generation, useful for companion-driven video outputs at scale.