**1) Data You’ll Need**

**1.1 Seed knowledge (curated JSON/CSV you control)**

* **Skills ontology:** id, name, domain, synonyms[].
* **Role templates:** role, core\_skills[], nice\_to\_have[], starter\_projects[].
* **Learning resources:** title, url, type, skills[], difficulty, duration, quality\_score.
* **Skill‑builder modules (DAG):** id, title, skill\_tag, level, prereqs[], duration\_hours, objectives[], practice\_tasks[], project\_brief.

**1.2 Runtime user data**

* **Resume text** (parsed) + **Job Description (JD)**.
* **Extracted skills** with confidence & level heuristics.
* **Session state:** chosen role, hours/day, completed tasks, interview history.
* **(Optional)** Audio snippets (transient only), language preference.

**1.3 Training/benchmark data (small, hackathon-friendly)**

* **ATS keyword pairs:** tiny set of (JD, resume) with human ATS scores (for calibrating weights).
* **Interview rubric snippets:** example good/poor answers per topic.
* **Intent routing examples:** 50–100 chat utterances labeled as PROFILE / MATCH\_ROLES / PLAN / INTERVIEW / ATS / ASSETS / HELP.

Storage: start with JSON files → move to SQLite/Postgres (users, profiles, resources, modules, roadmaps, ats\_results).

**2) Models to Use / Train**

Keep training minimal; lean on retrieval + prompting to move fast.

**2.1 Off‑the‑shelf (no training)**

* **Embeddings:** sentence-transformers/all-MiniLM-L6-v2 for skills/resources/RAG.
* **LLM (hosted):** for generation (plans, rewrites, interview feedback) with strict JSON guardrails.
* **STT (speech-to-text):** Whisper (local or API) or browser SpeechRecognition as fallback.
* **TTS (text-to-speech):** Cloud TTS (multi‑voice, multi‑lang) with SSML.
* **Translation:** Cloud translation API, or let LLM handle bilingual prompts.

**2.2 Light training (optional, 2–4 hrs total)**

* **Role fit scorer:** Logistic Regression/LightGBM on features:
  + coverage of core skills, cosine(JD,resume), title match, certs match → **probability of fit**.
* **Intent router:** fastText/LogReg or tiny transformer fine‑tune on your 50–100 labeled utterances (or keep it regex + heuristics).

**2.3 No training (deterministic rules)**

* **ATS scoring:** weighted formula + penalties for formatting.
* **Skill‑path synthesis:** topological sort over module DAG + calendar packing.

**3) Tools & Libraries**

**Backend (Python/Flask)**

* flask, flask-cors
* sentence-transformers, faiss-cpu, numpy, scikit-learn, lightgbm (optional)
* PDF/doc parsing: pypdf or pdfminer.six, python-docx
* Text utils: rapidfuzz (fuzzy match), python-frontmatter (for README export)
* **Real-time:** flask-sse (or custom SSE), flask-socketio (for WebRTC signaling if needed)

**Voice/Video**

* **WebRTC** in browser, simple-peer (JS) or your own RTCPeerConnection
* STT/TTS SDKs (Whisper/Cloud provider)
* (Optional) 2D avatar: react-three-fiber + viseme mapping, or iframe to a hosted avatar service

**Frontend (React + TS)**

* Vite, Tailwind, shadcn/ui, lucide-react
* State: React Query/Zustand
* Charts: Recharts
* File handling: react-dropzone
* Audio: Web Audio API; video: WebRTC APIs
* i18n: i18next (menus, static strings)

**Data/Infra**

* SQLite (dev) → Postgres (prod)
* Vector store: FAISS (in‑process)
* Deployment: Render/Fly/Cloud Run + Cloud Storage for static assets
* Logging: structlog + request IDs
* Secrets: environment variables

**4) Web Requirements (End‑to‑End)**

**4.1 Frontend UX (production‑looking)**

* **Tabs:** ATS Checker · Career Advisor · Skill Builder · Dashboard
* **Global language selector** (dropdown) → sets ui\_lang + reply\_lang.
* **Chat dock** (floating) with mic & camera toggles; stream responses; show citations.

**4.2 APIs (minimal but complete)**

* POST /ats/check → { resume\_text, jd\_text } ⇒ ATS score, coverage, gaps, formatting, rewrites (LLM).
* POST /ats/rewrite → { bullet, jd\_text, target\_lang } ⇒ improved bullet (STAR), keywords used.
* POST /advisor/analyze\_profile → extracted skills/levels.
* POST /advisor/match\_roles → roles with fit% and gaps.
* POST /advisor/roadmap → 30/60/90‑day plan (RAG‑grounded) + citations.
* POST /skills/path → module path + calendar schedule.
* POST /interview/question / POST /interview/feedback → adaptive Q&A with rubric.
* POST /translate (optional) → text ↔ chosen language.
* **Streaming:** /chat (SSE or WS) for conversational flow and tool calls.
* **Voice:** /ws/voice (WebSocket) to stream PCM/Opus; server returns chunked TTS audio URLs or base64.

**4.3 Real‑time**

* **SSE** for text tokens and tool events (fast to implement).
* **WebRTC** for voice/video sessions; use a simple signaling endpoint (WS) and TURN/STUN (e.g., Google STUN; free TURN like coturn if needed).

**4.4 Security & Privacy**

* Don’t store raw audio by default; if needed, delete after transcription.
* Strip PII before embedding (emails, phone numbers).
* CORS locked to your domains; rate limit public endpoints.
* Content safety: refuse non‑career advice (medical/financial) in prompts.

**5) How Components Use Data & Models (Pipeline)**

1. **ATS Checker**
   * Parse resume & JD → extract keywords (ontology + embeddings + exact match)
   * Compute **ATS score** (deterministic) + **cosine similarity**(resume↔JD)
   * Generate **rewrite bullets** via LLM prompt (JSON schema), optionally translate to reply\_lang
2. **Career Advisor**
   * Extract skills (regex+embed) → **role fit** (coverage + optional LogReg)
   * RAG retrieve top resources (MiniLM + FAISS)
   * LLM generates **30/60/90 day plan** (JSON) with **citations**; fallback template if JSON invalid
3. **Skill‑Builder**
   * Map gaps → modules; close prereqs; **topo sort** → calendar packing by hours/day
   * LLM stitches “why this” explanations; resources from RAG
4. **Voice/Video Multilingual**
   * STT (user\_lang) → intent route → (translate to system\_lang if needed) → tools/LLM
   * Response → (translate to user\_lang) → TTS (voice matching user gender/locale) → optional avatar lip‑sync

**6) Acceptance Criteria (what “done” looks like)**

* **ATS:** Same input gives same score (±2); ≥80% rewrite bullets include ≥2 JD keywords; export ATS‑friendly PDF.
* **Advisor:** Plan JSON validates; ≥80% steps show ≥1 citation.
* **Skill‑Builder:** No prereq violations; schedule respects hours/day.
* **Voice:** Round‑trip (speech→reply speech) under **3.5s P95**; language selector actually switches STT/TTS/LLM output.
* **Video:** Stable WebRTC call with avatar; audio‑video sync within 150ms.

**7) 2‑Day Build Plan (realistic)**

**T0–4h**

* Repo scaffold (Vite React + Flask), Tailwind/shadcn set up.
* Data seeds: skills, roles, resources, modules.
* FAISS index + /advisor/analyze\_profile, /advisor/match\_roles.

**T4–8h**

* /ats/check deterministic scoring; basic parsing; UI gauge & coverage chips.
* RAG retrieval + /advisor/roadmap (LLM) with JSON guardrails + fallback.

**T8–12h**

* Skill‑path synthesis /skills/path + calendar packing.
* UI for roadmaps & skill paths; export README & bullets (client‑side).

**T12–16h**

* SSE /chat stream; intent router; wire buttons to chat events.
* ATS rewrite endpoint (LLM), JD‑aware STAR bullets.

**T16–20h**

* Voice MVP: mic capture → /ws/voice → STT → text reply (no avatar yet).
* Language selector: plumb through STT/TTS/LLM prompts.

**T20–24h**

* Video avatar (WebRTC) container; TTS playback; basic viseme timing (optional).
* Polish, demos, sample data, screenshots for slides.

**8) Config You’ll Set (env)**

LLM\_PROVIDER=... # e.g., OpenAI/Gemini

EMBED\_MODEL=all-MiniLM-L6-v2

DB\_URL=sqlite:///app.db

STT\_PROVIDER=whisper # or browser fallback

TTS\_PROVIDER=... # multi-language voice

TURN\_URL=... # if using TURN for WebRTC

**9) Prompts (ready to paste)**

* **ATS bullet rewrite:** “Rewrite for <ROLE>; ≤28 words; quantify; include 1–2 JD keywords; JSON {original, improved, delta\_keywords[]}.”
* **Plan generation (RAG):** “Ground every task in provided context; ≤90‑min tasks; include citations[] and why\_this.”
* **Interview feedback:** JSON {score, what\_went\_well[], improve\_next\_time[], follow\_up, hint}—be specific & concise.
* **Translation wrapper (if needed):** “Translate to <LANG>, preserve tech terms; return plain text.”