ElderTech Assistant – Full-Stack App Blueprint

Purpose

Provide a friendly, voice-driven assistant that helps older adults master everyday technology through a simple “video chat” interface.

---

1. High-Level User Flow

1. **Launch**: User navigates to the web app on a tablet/desktop. First-time visitors see an onboarding walkthrough with large visuals and voice narration.
2. **Conversation**: A video tile featuring a warm, human-like avatar invites the user to speak. The user presses a large **“Hold to Talk”** button (or uses wake-word).
3. **Transcription**: Audio is streamed to the backend via WebRTC → Whisper API instantly transcribes.
4. **NLP Processing**: Back-end sends conversation context + transcript to GPT-4o with an age-appropriate, empathy-focused system prompt.
5. **Response Rendering**:
6. GPT returns markdown (plain text + optional list/images).
7. FastAPI triggers D-ID *Talk* → returns MP4 of avatar speaking the text.
8. ElevenLabs (or Azure TTS) generates fallback audio for screen readers.
9. **Playback**: Front-end displays subtitles in 24-pt font and plays the avatar video. A **“Replay / Slower / Faster”** control appears.
10. **Logging & Learning**: Q&A pair persisted in MongoDB → nightly batch groups similar questions, pushes new exemplars to a *FAQs* collection and RAG index.

---

2. Tech Stack

Layer	Choice	Rationale
Frontend	Next.js 14 (React Server Components) | Tailwind CSS | Zustand	SEO-ready, fast bundling; Tailwind for quick accessible UI; lightweight global state.
Real-time	WebRTC (native) | Socket.IO fallback	Direct peer media when supported; fallback for restrictive networks.
Backend API	FastAPI + Pydantic v2	Async, type-safe, great docs; easy OpenAPI spec generation.
Database	MongoDB Atlas (Collections: `users`, `sessions`, `messages`, `faqs`)	Flexible document model fits conversational data.
AI Services	OpenAI Whisper & GPT-4o; D-ID Talk; ElevenLabs TTS	Best-in-class transcription, reasoning, and avatar synthesis.

Layer	Choice	Rationale
DevOps	Docker Compose (dev) → Render.com / Fly.io / Vercel Edge (prod)	Simple local spin-up; global edge functions for low latency.

3. System Architecture

```
┌─────────────┐   audio    ┌───────────────┐    text    ┌─────────────┐
│  Browser    │ ────────▶ │  FastAPI API  │────────▶ │  GPT-4o      │
│ (Next.js)   │            │ /whisper      │            │             │
│             │ ◀──────── │  /chat        │ ◀──────  │             │
└─────────────┘   video    └───────────────┘    MP4     └─────────────┘
       │                           │                          ▲
       │                           │                          │ nightly
       │                           ▼                          │ batch
       │                   ┌───────────────┐                  │
       │                   │ MongoDB Atlas │──────────────────┘
       ▼                   └───────────────┘
┌───────────────┐                             clustered FAQs
│  D-ID Talk    │                             & embeddings
└───────────────┘
```

4. Frontend Breakdown (Next.js `/app` directory)

```
/app
  ├─ layout.tsx             # Global providers (Zustand, Theme)
  ├─ page.tsx               # Video-chat landing
  ├─ components/
  │    ├─ AvatarPlayer.tsx # Plays returned MP4 + captions
  │    ├─ MicButton.tsx    # Press-&-hold recording, large accessible
  │    ├─ Onboarding.tsx   # Carousel tutorial
  │    └─ HelpCenter.tsx   # Themed FAQ explorer
  └─ lib/
       ├─ webrtc.ts         # Mic capture + mediaRecorder
       ├─ api.ts           # fetcher (SWR) for /chat, /faqs
       └─ validators.ts    # zod schemas
```

Key UI Patterns

- **Contrast & Size**: 18-24 pt base font, 44 px target touch areas.
- **Voice Everywhere**: All buttons have `aria-label` + optional spoken hints.

• **Frustration-free**: Always provide *Repeat*, *Slower*, *Examples* buttons.

---

5. Backend Breakdown (FastAPI)

```
backend/
  ├─ main.py              # FastAPI app factory
  ├─ routers/
  │    ├─ chat.py         # /chat POST {transcript}
  │    ├─ whisper.py      # /whisper WS stream
  │    ├─ faqs.py         # CRUD on FAQs
  │    └─ auth.py         # JWT /family login
  ├─ services/
  │    ├─ openai.py       # GPT-4 & Whisper calls
  │    ├─ did.py          # D-ID Talk wrapper
  │    └─ db.py           # Motor (async Mongo) client
  └─ scripts/
       └─ nightly_faq_clustering.py
```

Sample `/chat` Endpoint

```
@router.post("/chat", response_model=ChatResponse)
async def chat(req: ChatRequest, user: User = Depends(require_user)):
    # 1. Generate answer
    system = "You are a patient tech coach..."
    msgs = [
        {"role": "system", "content": system},
        *req.history,
        {"role": "user", "content": req.transcript},
    ]
    gpt_resp = await openai.chat(messages=msgs, model="gpt-4o")
    answer = gpt_resp.choices[0].message.content

    # 2. Create avatar video
    mp4_url = await did.talk(answer)

    # 3. Persist
    await db.messages.insert_one({
        "user_id": user.id,
        "q": req.transcript,
        "a": answer,
        "video": mp4_url,
        "ts": datetime.utcnow()
    })
    return {"answer": answer, "video": mp4_url}
```

---

6. Database Schema (Mongo)

```
users: {
  _id, name, role: "elder" | "family", language, createdAt
}
sessions: {
  _id, user_id, startedAt, endedAt
}
messages: {
  _id, session_id, q, a, video, ts
}
faqs: {
  _id, question, answer_md, tags: ["email", "scams"], updatedAt
}
```

---

7. Accessibility & UX Principles

- **WCAG 2.2 AA** compliant colors, text sizes, captions.
- **Keyboard & switch-control** navigation (focus rings, `tabindex`).
- **Latency budget**: <400 ms TTFB, <2 s full answer (cached path).

---

8. Onboarding Experience

1. Welcome screen with avatar: “Hi! I’m here to help you with tech.”
2. Guided demo: presses mic, asks “What’s an app?” → shows sample reply.
3. Brief quiz: User tries asking; gets celebratory animation.
4. Option to view “common topics” grid or start conversation.

---

9. Family Portal (Optional)

- Auth0-protected dashboard.
- Pre-load FAQs, view transcripts, flag incorrect answers.
- Email digests of weekly usage.

---

10. Deployment & Ops

- **Dev**: `docker compose up` spins Next.js + FastAPI + Mongo.
- **Staging**: Vercel (frontend) ➝ Render (backend) ➝ MongoDB Atlas.
- **Observability**: Sentry (front & back), Prometheus + Grafana metrics.
- **CI/CD**: GitHub Actions → lint, type-check, Playwright e2e.

---

11. Next Steps

1. Scaffold repo with `pnpm create next-app@latest` + `fastapi-project-generator`.
2. Implement Whisper streaming endpoint.
3. Build AvatarPlayer component and connect to /chat.
4. Conduct hallway tests with 2–3 seniors; iterate on font sizes & flow.
5. Hard-code FAQ JSON; later hook nightly clustering job.

–– End of Blueprint ––