Nice — below is a complete, **phase-by-phase roadmap** for building **AI Voice-Guided Website Tours** from scratch. I've broken it into clear milestones, the concrete tasks for each, deliverables you can demo at the end of each phase, acceptance criteria, recommended tech choices, and notes about how to scale/extend later. I avoided time estimates (per your constraints) — treat each phase as a discrete milestone you finish and demo.

Phase 0 — Project Setup & Research (Foundations)

Goal

Establish the problem scope, pick target websites and users, choose the stack and create a reproducible dev environment.

Tasks

- Define target personas & use-cases (first-time visitor, shopper, visually impaired user, dashboard power-user).
- Select 3–5 demo websites (one e-commerce, one university, one SaaS dashboard, one government) for development + testing.
- Choose tech stack:
    - Frontend: React / Next.js
    - Voice STT/TTS: Web Speech API (prototype); for backend option: Whisper / local models
    - LLM: WebLLM for on-device inference (privacy demo) + OpenAI/GPT for server-side if needed
    - Backend: FastAPI + LangChain
    - DB: MongoDB for user prefs & metadata; Vector DB (Weaviate/Pinecone) for embeddings
    - Dev ops: Docker, Vercel/Netlify for frontend, AWS/Render for backend
- Create repo scaffold, branch strategy, issue tracker templates, CI basics (linting, tests).
- Make a minimal UI mockup and demo flow diagrams (user presses mic → hears tour → page scrolls and highlights).

Deliverables

- Project repo with README, architecture sketch, and UI mockups.
- List of demo websites and user stories.
- Minimal "hello world" React app and FastAPI skeleton.

Acceptance Criteria

- Team agrees on personas and demo sites.
- Repo and basic CI in place.
- Simple mock UI ready to demo.

Phase 1 — Core MVP: Voice Tour Basic (Local-first)

Goal

Build a working prototype that converts voice commands to basic guided tours on a page: detect sections, read out summaries, scroll & highlight.

Tasks

- Frontend:
  - Implement microphone capture (Web Speech API) with push-to-talk UI.
  - Implement TTS (Web Speech API) to speak tour content.
  - Write a DOM analyzer module that:
    - Extracts headings, main content blocks, images, buttons, forms, tables.
    - Produces a simple structure: `[ {id, type, title, textSnippet, boundingRect} ]`.
  - Implement the highlight/scroll UI (overlay card + CSS highlight).
- Backend/Local LLM:
  - Build a small “tour generator” function: input = DOM structure + user intent; output = tour steps (title, text to speak, actions like scroll/highlight).
  - Start with deterministic heuristics and template text (no LLM yet).
- Integration:
  - Flow: user says “Give me a tour” → capture → call tour generator → frontend executes actions and reads content step-by-step.
- UX:
  - Show a mini transcript and “next / repeat / stop” controls.
  - Allow “skip to section ”.

Deliverables

- Live demo: Ask for a tour → page scrolls & highlights sections while voice speaks.

- Repo: frontend + small backend (or fully frontend) with DOM analyzer + speech integration.

Acceptance Criteria

- Demo site properly analyzed and the system can highlight and read at least 5 sections.
- Voice commands “next”, “previous”, “repeat”, and “stop” work.

Phase 2 — LLM Understanding & Intent Mapping (Smart Actions)

Goal

Replace heuristics with an LLM-based understanding layer that maps natural language commands to DOM actions and generates natural-sounding narrations.

Tasks

- Choose LLM path:
  - On-device WebLLM for privacy/latency demos.
  - Server-side LLM (OpenAI or similar) for complex reasoning (can be hybrid).
- Build a prompt engineering layer:
  - Template: `You are a webpage tour assistant. Given the DOM JSON and intent, choose which elements to show and produce X-step tour with short narration for each step. Output must be JSON: {steps:[{element_id, action, narration}]}`
- Implement LangChain-like agent (or custom planner) that:
  - Parses user utterance to intent (goal, mode: beginner/goal/expert).
  - Maps synonyms (pricing, plans, cost) to elements.
  - Breaks multi-step goals (e.g., “Compare pricing and highlight differences”) into actions.
- Add safety & constraints: maximum characters per narration, do-not-click on login buttons, etc.
- Build a small cache of past tours & reuse to speed up repeated requests.

Deliverables

- Demo where user asks diverse questions (goal-based): “Show pricing”, “Compare Basic and Pro”, “Explain the signup flow”.
- Examples of LLM prompt templates and example outputs.

Acceptance Criteria

- The LLM layer correctly maps common intents to the right DOM elements on 80% of test phrases (define test set).
- Tour outputs are human-readable and actionable (JSON steps).

Phase 3 — Data, Personalization & Memory

Goal

Make tours personalized and remember user preferences and history so tours become faster and more relevant.

Tasks

- Data model:
  - MongoDB collections: `users`, `sessions`, `tours`, `preferences`.
  - Vector DB for embedding DOM sections + user queries.
- Build embedding pipeline:
  - Convert element text + metadata into embeddings.
  - Store semantic index for fast retrieval.
- Personalization:
  - Use stored user preferences (e.g., “I prefer short tours”, “I’m an advanced user”) to choose tour mode.
  - Memory: recall past tours and adapt (e.g., skip sections user already visited).
- Add metrics collection (which tour steps were accepted, skipped, clicked).
- Implement a small UI for user profile and tour preference toggles.

Deliverables

- Personalized demo: user with preferences receives a different style of narration and shorter/longer tours.
- Vector search demo: ask complex queries that retrieve specific DOM elements across pages.

Acceptance Criteria

- System returns personalized tours that reflect stored preferences.
- Vector search retrieves relevant sections for semantic queries.

Phase 4 — Advanced Agent Actions & Multi-step Automation

Goal

Turn the tour system into an agent that can perform multi-step tasks (click, fill forms, extract tables, run comparisons) safely.

Tasks

- Agent capabilities:
    - Read-only actions (highlight, scroll, read).
    - Safe interactive actions (click non-sensitive buttons).
    - Extract & summarize tables and lists.
    - Fill forms using user-approved data (with explicit consent).
- Action planner:
    - Use LLM + tools pattern: planner → verifier → executor.
    - Add human-in-the-loop confirmations for sensitive actions (submitting forms, payments).
- Implement DOM sandboxing to prevent the agent from performing harmful operations.
- Build result presentation: inline comparison tables, CSV export, PDF snapshot.

Deliverables

- Demo multi-step flow: “Find internships, filter by Chennai, extract the job titles & contact emails, and compare benefits.”
- Evidence of safe execution: agent asks confirmation before form submit.

Acceptance Criteria

- Agent reliably performs non-destructive multi-step tasks on demo sites.
- Safety verifications occur for all sensitive operations.

Phase 5 — On-Device Privacy Mode & Offline LLMs

Goal

Make a privacy-first variant: run everything in-browser (WebLLM), no server calls, suitable for sensitive sites.

Tasks

- Integrate a compact LLM that can run in the browser (WebLLM / smaller open models).
- Convert tour-generation pipeline to local inference:
    - DOM parsing, embedding generation (approximate), LLM prompt execution locally.
- Implement efficient caching and memory management to handle model size limits.
- Provide fallback: if local model fails, show “Server assisted mode” and ask user to opt-in.

Deliverables

- Demo where all operations (including understanding and narration) run in the browser and no backend is contacted (show devtools/network panel).
- Documentation of privacy guarantees and limitations.

Acceptance Criteria

- Local-only mode produces coherent tours on small/medium pages.
- Network calls can be disabled and user sees offline behavior.

Phase 6 — Accessibility & Multimodal Extensions

Goal

Extend the product to support visually impaired users and multimodal inputs (gaze, gestures, keyboard navigation).

Tasks

- Accessibility improvements:
    - ARIA-friendly highlights.
    - Better TTS semantics (richer pauses, SSML support).
    - Screen-reader compatibility.
- Multimodal inputs:
    - Basic webcam-based gaze detection (WebGazer.js) to focus UI elements.
    - Simple gesture recognition to confirm actions (hand wave to continue).
    - Keyboard-first shortcuts & full keyboard navigation.
- Add language support & translation layer.

Deliverables

- Accessibility demo: full tour usable by a screen-reader user.
- Multimodal demo: gaze focus + voice commands working together.

Acceptance Criteria

- Audited accessibility score improvement on Lighthouse / a11y tools.
- Multimodal inputs trigger expected actions reliably in demo.

Phase 7 — Production Hardening, Monitoring & Scalability

Goal

Prepare the product for wider user testing, reliability, and low-latency scale.

Tasks

- Create robust API rate-limiting and auth (OAuth for multi-site agents).
- Add logging, observability, and metrics (tour success rate, latency, error counts).
- Add feature flags and canary rollout for new agent capabilities.
- Implement caching of tours & precomputed embeddings to reduce inference costs.
- CI/CD: automatic tests (unit, integration), end-to-end demo tests.
- Prepare docs and developer API so other sites can integrate the widget.

Deliverables

- Production-ready deployment with monitoring dashboards.
- Onboarding docs for third-party sites and APIs (embed script snippet).

Acceptance Criteria

- Stable release deployed behind monitoring.
- Performance targets met for demo usage pattern.

Phase 8 — Hackathon Polish & Pitch Materials

Goal

Craft a compelling demo, slides, video, and a live-judging plan.

Tasks

- Distill feature set into a tight demo script (30–90 seconds per judge).
- Create 6–8 slides: Problem → Solution → Demo (screenshots) → Architecture → Novelty → Roadmap → Ask.
- Build short pre-recorded demo video for fallbacks.
- Prepare a live scenario script for each persona (first-time, shopper, visually impaired).
- Prepare answers for likely judge questions: privacy, latency, IP, scaling, and commercial use.

Deliverables

- Slide deck and 2–3 minute demo video.
- Live demo checklist and fallbacks.
- Ready answers to FAQs.

Acceptance Criteria

- Demo runs reliably in a 3–5 minute slot.
- Judges see clear differentiation vs. existing navigation methods.

Cross-cutting Concerns (apply across phases)

- **Evaluation metrics**: user satisfaction (qualitative), task success rate (did user reach target), average tour length, action accuracy (intent → correct DOM element), latency.
- **Privacy & Security**: avoid collecting PII unless explicitly permitted; require explicit consent for form-filling and data submission; provide local-only mode.
- **Safety**: never auto-submit payments or logins; whitelist demoable DOM actions.
- **Testing**: unit tests for DOM parser, LLM prompt outputs (schema validation), end-to-end tests for voice commands.
- **Reusability**: make the DOM analyzer modular; export tour generator as a microservice with a clear JSON contract.

Concrete artifacts I can create next (pick any)

- FastAPI endpoint specs (request/response JSON examples).
- React components: MicButton, TourOverlay, ElementHighlighter, TourTranscript.
- LLM prompt templates and example prompts + expected JSON output.
- MongoDB schema & vector index design.
- Demo script & slide deck.
- Minimal code scaffold for Phase 1 to run locally (I can drop code stubs).