1. Customer Interviews

CUSTOMER INTERVIEWS

1. Alex – “The Sniper”

MBA Candidate | Target: Elite firms (McKinsey, Google, Goldman)

Profile intent: Few, very high-stakes applications; reputation is currency.

Chloe: Alex, take me into the moment right before you hit “Submit” on an application. What’s happening?

Alex: It feels like defusing a bomb. My heart rate’s up. I zoom into the PDF, check every line. Did I leave a placeholder? Did I misspell the firm’s name? Did I reuse a line from another application that doesn’t quite fit here?

Chloe: Why does it feel that intense?

Alex: Because at this level, reputation is everything. If I mess up on a cover letter to a no-name company, fine. But if I send something sloppy or misleading to a McKinsey or Google, that’s my name attached to it. I worry about being quietly blacklisted.

Chloe: How does that change your attitude toward AI tools?

Alex: It makes me cautious. If an AI hallucinates that I led a $10M project when it was actually $100k, and I don’t catch it, I’m in trouble. I’d rather miss an opportunity than go into a room based on false claims.

Chloe: When you imagine an “Auto-Apply” button, how do you feel?

Alex: Terrified. I don’t want a bot speaking for me. I want something more like a Super-Editor. A tool that says, “Alex, I changed this bullet to match the JD’s keywords, but I kept your metrics exactly. Approve?” I want to pull the trigger myself, every time.

Chloe: What would make you trust an AI assistant at all?

Alex: Clear, visible constraints. One: it never changes numbers or scope on its own. Two: every change is shown in a diff so I can see exactly what was done. Three: it asks for confirmation when it’s unsure—especially around quantitative claims.

Chloe: What’s your ideal “success state” with a product like this?

Alex: I send fewer, sharper applications. Each one feels like my best self, but never crosses the line into exaggeration. If the tool can tighten the language, align it to the JD, and guarantee it doesn't invent anything, that's a win.

Chloe: What would absolutely break your trust?

Alex: If I ever discover it changed a number or invented a responsibility without flagging it. One breach like that and I'd never use it again.

---

2. Sam – "The Shotgunner"

Laid-off Tech Worker | Target: Any PM role above a certain comp floor

Profile intent: High volume, high fatigue, wants scale without burnout.

Chloe: Describe your current weekly routine around job hunting.

Sam: It's like playing a slot machine. I wake up, see 50 new PM jobs on LinkedIn. I know I should tailor my resume for each one, but I physically can't. I carefully tailor the first two, then I get tired and end up sending the same generic PDF to the next 48.

Chloe: How does that feel emotionally?

Sam: Draining. The worst part is the "black hole." I spend four hours tailoring, I hit submit, and then… silence. No feedback, no signal. Just nothing. It starts to feel like I'm wasting my energy.

Chloe: What's the specific emotion when you realize you used a generic resume on a great role?

Sam: Regret. I'll see a job three days later and think, "That was perfect for me, and I sent a half-baked version because I was exhausted." I feel like I burned that lead because I didn't have the energy.

Chloe: If a tool could only do one thing for you, what would it be?

Sam: Scale my best self. Take the kind of application I'd write if I had an hour per job, and make it possible to send that level of quality to dozens of jobs per week with just a few clicks.

Chloe: How do you feel about automation applying on your behalf?

Sam: I'm open to it, but with guardrails. I don't want it to apply to completely irrelevant roles or to send obviously wrong content—like a cover letter with the wrong company name. I want a system where I can quickly approve a batch, not micromanage each line.

Chloe: What does a "good day" in your search look like?

Sam: In an hour or two, I'd like to identify 20–30 solid roles, have tailored resumes and basic cover letters generated for them, and then approve and send most of them. If my interview rate even slightly improves, I'd feel like I'm back in control.

Chloe: What's non-negotiable for you?

Sam: Speed. If using the tool takes as long as doing it myself, I won't stick with it. Also, no embarrassing errors—wrong names, repeated obvious mistakes. That's where I draw the line.

---

3. Bella – "The Pivoter"

Career Switcher | Architecture → UX / Product / Design

Profile intent: Reframe existing experience into a new language and role.

Chloe: When you look at UX or product job descriptions, what happens in your head?

Bella: I feel like I'm reading a different dialect of English. They talk about "Agile ceremonies," "design systems," "wireframes," "user flows." I've done similar things in architecture—iterative design, stakeholder meetings—but using totally different terms.

Chloe: How does that affect your willingness to change your resume language?

Bella: I'm hesitant. I'll think, "I ran weekly site meetings with contractors, that's basically a standup," but I'm scared to call it a "daily standup" in case I misuse the term. I don't want to look like I'm pretending I've done something I haven't.

Chloe: So what do you need from an AI assistant?

Bella: A translator. Something that looks at my past work and says, "You actually have this skill; here's the industry term for it." And then it explains why that term fits. I want it to teach me the language while it rewrites.

Chloe: How do you feel about it applying automatically on your behalf?

Bella: For now, no. Each application feels like a small identity shift. I want to see how I'm being described. Maybe once I trust it and see consistent patterns, I'd let it handle lower-priority roles more autonomously.

Chloe: What would make you feel safe using such a tool?

Bella: Explanations. If it changes "Client Presentation" to "Stakeholder Management," I want a tooltip that says, "We changed this because the JD emphasizes stakeholder alignment." I want to understand the why, not just see new words.

Chloe: And what does success look like for you with this product?

Bella: I feel like I have a coherent story that makes sense in the new field, and I understand the jargon I'm using. Even if I don't get every job, I'm confident my applications represent who I am and where I'm going.

---

4. Priya – "The Anxious International"

International MBA / Visa Holder | Target: Sponsorship-friendly employers

Profile intent: Highly constrained shots; needs filtering and linguistic safety.

Chloe: For you, what's at stake in this job search?

Priya: It's not just a career choice—it's my ability to stay in the country. I have a limited time window and a limited number of realistic options. Every application feels like using one of my "shots."

Chloe: How does that change the way you approach job postings?

Priya: I'm hyper-cautious. I don't want to waste time on companies that never sponsor. I spend a lot of time just figuring out who actually hires internationals before I even worry about tailoring.

Chloe: What worries you most about using AI in this process?

Priya: Nuance. I'm worried it will use idioms or phrases that don't sound like me. Then I get to the interview and I sound different. I also worry it might apply to companies that don't sponsor, or to roles I'm not legally eligible for.

Chloe: If a tool could do one thing perfectly for you, what would it be?

Priya: Filter ruthlessly. Don't even show me jobs from companies that don't sponsor, or roles clearly unsuited for my visa situation. Then, help me make sure my resume is grammatically perfect and uses appropriate keywords without sounding unnatural.

Chloe: How do you feel about automation submitting applications for you?

Priya: I'd prefer to submit myself, at least early on. Each opportunity is too important. I'd be okay with the tool preparing everything and guiding me step by step, but I want to press the final button.

Chloe: What does "confidence" look like for you in this product?

Priya: I see that every job in my dashboard meets my non-negotiables—sponsorship, location, level. My resume feels polished and professional, and the language still feels like mine. If that happens, I'll feel like I'm not wasting time or chances.

---

5. Daniel – "The Stealth Candidate"

Employed at Strong Brand | Quietly exploring new roles

Profile intent: Discreet, controlled search with strong brand protection.

Chloe: Why does job searching feel risky to you?

Daniel: Because I'm not desperate. I have a good job, a mortgage, responsibilities. I'm looking for a 20–30% improvement, not a lifeline. If my current employer suspects I'm looking, that can change internal dynamics and even put me at risk.

Chloe: How does that affect your behavior with tools?

Daniel: I'm paranoid about anything that could act on my behalf without my explicit consent. The idea of "Auto-Apply" makes me nervous: applying to a competitor, a partner, or some subsidiary could blow back on me.

Chloe: What would you need to feel safe using an AI assistant?

Daniel: A clear blocklist. "Never apply to these companies, their subsidiaries, or specific domains." Also, a mode where the product only generates drafts—PDFs, text, etc.—and I decide where and how to submit.

Chloe: So you prioritize discretion over speed?

Daniel: Exactly. I'm fine applying to just a few roles per week as long as each one is intentional and high quality. I'd rather have three carefully crafted applications than thirty risky ones.

Chloe: Would you still find automation useful in any form?

Daniel: Yes, on the preparation side. If it can generate tailored resumes and cover letters based on job descriptions, that saves me time. But I want a detailed log of what's generated and I want to be the only one who sends anything out.

Chloe: What would immediately break your trust?

Daniel: If it applied to a company on my blocklist, or any company without my explicit approval. Also, if I couldn't see an exact history of what was generated and where it was supposed to go.

Chloe: Describe your ideal workflow with such a tool.

Daniel: I'd feed in a handful of roles, get back high-quality drafts, edit them lightly, download, and then submit manually on my timeline. The system helps me think and write; it never acts in the world without me.

---

6. Jason – "The Power User"

Engineer / System Thinker | Treats job search as an optimization problem

Profile intent: Wants a configurable system, not a black box.

Chloe: How do you currently manage your job search?

Jason: Like a mini experiment. I have a spreadsheet tracking roles, keywords, resume versions, responses. I'll change my headline or tweak a section to see if response rates change. It's very manual.

Chloe: What frustrates you about existing AI tools?

Jason: They're black boxes. I upload a resume and a JD, it spits out a new resume, and I have no idea why it chose certain phrases or emphasized certain skills. I can't tweak the logic; I'm just supposed to trust it.

Chloe: What would an ideal AI assistant look like for you?

Jason: A controllable agent. Give me knobs and dials: sliders for how much to emphasize leadership vs technical depth vs domain knowledge. Let me set an “Aggressiveness” level—how much it can stretch the narrative within factual boundaries.

Chloe: How do you feel about “Apply” buttons?

Jason: A simple “Apply” button does nothing for me. I need to define the rules under which the agent operates. For example: “For mid-tier companies above 85% match, use this resume template and apply automatically. For top-tier or dream companies, draft only.”

Chloe: Would you spend time configuring those rules?

Jason: Absolutely. I already spend that time in spreadsheets. If the system lets me encode my strategy into its behavior, that’s a huge win.

Chloe: What would you want to see after it tailors a resume?

Jason: A breakdown. “We emphasized Python because the JD mentions it 7 times vs Java once. We added this leadership bullet because the role is tagged as ‘senior’.” I want to understand the reasoning, not just the result.

Chloe: What’s non-negotiable for you?

Jason: Transparency and override. I should be able to see the rules, adjust them, and override any decision. If I can’t inspect or influence the logic, I’ll never fully trust it.

Chloe: What does success look like with this product?

Jason: I codify my job search strategy into the system, and it executes it faster than I could. My time shifts from micro-editing to adjusting strategy and reviewing edge cases.

2. Market Research

MARKET RESEARCH

High-Stakes Job Search Copilot

Final Market & Product Strategy Report

1. Purpose and Stance

We are not building a mass-market auto-apply bot.

We are building a **safety-first AI copilot for high-stakes applications**, starting with **MBA "Snipers" who are international and pivoting**.

Strategic choices:

1. Focus on **concurrent active seekers** who have money and urgency (not everyone with a résumé).
2. Optimize for **Snipers, Internationals, and Pivoters**; treat "Shotgunners" as a later, constrained use case.
3. Compete on **Safety, Translation, and Visa-aware intelligence**, not on tracking or brute-force volume.

Everything else follows from these decisions.

2. Market Sizing (Serviceable Obtainable Market)

We ignore "Total Addressable Market" and focus on **people currently in pain and realistically reachable**.

2.1 Segments and Sizes (Concurrent Active Seekers)

These numbers are directional, not precise; they are to guide strategy.

Segment (Persona)	Est. Active Size	Strategic Interpretation
Tier 1 – MBA Snipers (Alex)	~75k–80k	Top ~40% of active MBA/early alumni seekers, aiming for elite roles. Highest willingness to pay; reputation safety is key.
Tier 2 – MBA Internationals (Priya)	~65k–75k	Subset of MBA pool; visa-constrained. Landing a job is a legal/immigration necessity. Sponsorship risk is existential.

Segment (Persona)	**Est. Active Size**	**Strategic Interpretation**
Tier 3 – MBA Pivoters (Bella)	~120k–130k	MBAs switching industry/function (e.g., Architecture → UX, Engineer → PM). Majority of MBA pool; they need skill translation.
Tier 4 – Stealth Professionals (Daniel)	~8M–12M	Employed white-collar workers quietly looking. Privacy and brand protection matter more than speed. Good for subscription.
Tier 5 – Shotgunners (Sam)	~9M–10M	High-volume, low-tailoring applicants. Large but low-ARPU and heavy LLM cost. Useful later for data, not initial revenue.

Key points:

- **Tiers 1–3 overlap heavily**. Our true beachhead is the intersection of Sniper + International + Pivoter.
- **Stealth behavior is pervasive**, not niche; privacy features are platform-level, not only for Daniel.
- **Shotgunners** are deliberately **de-prioritized** for early monetization and product shaping.

3. The "Golden Wedge": Our Super-Persona

Our MVP focuses on the **intersection** of Alex, Priya, and Bella.

3.1 Super-Persona: "Priya the Pivoting Sniper"

- International MBA at a strong/top program.
- Pivoting into PM / strategy / consulting / product-adjacent roles.
- Targeting elite or selective employers.
- Visa-constrained; each application is a scarce "shot."
- Afraid of AI hallucinating claims or pushing her into roles that will reject her on sponsorship/legal grounds.

3.2 What She Buys

1. Reputation Protection (Alex component)

- Behaves like a Sniper: every application feels like "defusing a bomb."
- Refuses fully blind auto-apply.

- Wants a **Super-Editor**: precise tailoring plus a diff view that proves no metrics or facts were quietly changed.

2. Survival Filtering (Priya component)

- Job outcome affects legal right to stay.
- Cannot waste time on non-sponsoring or “citizens only” roles.
- Needs a **visa-aware filter** that reduces false positives and highlights risks.

3. Career Translation (Bella component)

- Reads JDs as a different dialect.
- Afraid to misuse jargon (e.g., calling ad hoc meetings “Scrum ceremonies”).
- Needs a **Translator** that:
    - Re-labels her experience into the target field’s language.
    - Explains *why* each translation is valid so she can defend it in interviews.

If we build a product that satisfies her, we automatically cover:

- Domestic Snipers (Alex) once visa constraints are turned off.
- Non-international Pivoters (Bella) with the same translation engine.
- Stealth candidates (Daniel) by layering in blocklists and draft-only workflows.

4. Positioning and Competitive Stance

4.1 What We Are Not

We are **not**:

- A generic job tracker (Teal, Simplify, etc. already own “organize and track”).
- A pure auto-apply bot bragging about “100+ applications per day.”
- A black-box AI résumé writer that silently invents achievements.

4.2 What We Are

> **The safety-first AI copilot for high-stakes applications.**

Four pillars:

1. Truth-First Generation

- All content anchored to a user-verified fact base (résumé + structured Q&A).
- No silent invention of projects, titles, or metrics.
- Any expansion or numerical change is highlighted and requires explicit approval.

2. Visa- and Risk-Aware “Sponsorship Intelligence”

- We do **not** assume “company X sponsors” ⇒ “every job at X is safe.”
- Instead:
  - Employer-level sponsorship history is a **prior**, not a decision.
  - **NLP on each job description** detects:
    - Negative signals (“no sponsorship,” “US persons only,” “must be authorized to work without sponsorship”).
    - Ambiguous or missing signals.
  - Output is a **risk label** (“likely sponsor,” “unclear,” “likely no sponsor”) plus explanation.
- Priya can set rules:
  - “Hide likely-no-sponsor roles.”
  - “Warn but still show ‘unclear’ roles.”

3. Role and Language Translation With Education

- Map past experience into target-role framing:
  - “Client presentations” → “stakeholder management.”
  - “Site coordination” → “cross-functional execution.”
- Each significant change accompanied by a rationale:
  - “We changed A → B because the JD emphasizes C.”
- Product becomes both **tailoring engine** and **learning tool**.

4. Reputation and Privacy as Defaults

- Side-by-side diffs for every tailored artifact.
- Explicit warnings for metric changes, company names, or scope shifts.
- Blocklists and draft-only modes available to all.
- No hidden auto-apply; the system never acts behind the user’s back.

5. Product Roadmap

We execute in three phases: **Trust** → **Privacy** → **Controlled Scale**.

Phase 1 – Safe Harbor MVP (Months 1–3)

Target: Super-Persona (Priya + Alex + Bella) in MBA programs and early alumni.

Positioning:

“The only AI copilot I trust for MBB/FAANG/PE applications.”

Must-Have Capabilities:

1. **Sponsorship Intelligence v1**
    - Employer-level sponsorship prior (where available).
    - JD-level NLP for sponsorship disclaimers and legal restrictions.
    - Risk labels and explanations; user-configurable rules for hiding/showing roles.
2. **Super-Editor With Diff Dashboard**
    - Upload résumé → build fact base.
    - Per-job tailoring of résumé and cover letter.
    - Side-by-side diff view:
        - Highlight wording changes.
        - Special flags for metric/scope changes.
    - User approval required per artifact.
3. **Translator With Educational Tooltips**
    - Automatic mapping of old-role language to new-role language.
    - Tooltip on each key translation:
        - “We changed X → Y because the JD emphasizes Z.”
    - Improves applications and prepares the candidate for interviews.
4. **Application Support (Assisted, Not Blind Auto-Apply)**
    - Export tailored documents.
    - Optional browser extension to help with form-filling; **user presses final “Submit.”**
    - Clear log of what was generated for which job.

Phase 1 Success Criteria:

- High trust scores:
    - “I would send this as-is.”
    - “I understand every change.”
    - “I feel safer using this than writing manually.”
- Early signs of improved interview rate for pilot users vs. their baseline.

Phase 2 – Stealth Expansion (Months 4–6)

Target: Daniel-type employed professionals and alumni.

Positioning:

“Look for a better job without your boss ever knowing.”

Must-Have Capabilities:

1. **Blocklists and Guardrails**
    - "Never apply to: [current employer], [subsidiaries], [named partners/competitors]."
    - Clear UI for managing exclusions.
2. **Draft-Only / "Career Insurance" Mode**
    - Ongoing intake of JDs (via links, boards, or monitored firms).
    - For each promising role:
        - Prepare safe drafts using Phase-1 safety + translation logic.
    - Notify user; they decide if/when to submit.
    - Monetized as subscription ("career insurance" for high-stakes professionals).
3. **Light Analytics & Positioning Tests**
    - Feedback on which positioning (e.g., "Product Strategy" vs "BizOps") yields better responses.
    - Transparent, interpretable summaries (no black-box "magic score").

Phase 3 – Controlled Volume (Month 6+)

Target: Sam-type users and busy professionals needing more throughput without sacrificing safety.

Positioning:

"Scale your best self, not your worst résumé."

Capabilities (Tightly Constrained):

1. **Safe Batching**
    - User selects multiple jobs; system prepares tailored drafts for all.
    - Streamlined review UI for quick approval while still exposing key diffs and risk flags.
2. **Optional, Bounded Auto-Apply (If Ever)**
    - Explicit opt-in.
    - Limited to roles above a fit threshold and below a stakes threshold.
    - Hard caps on applications per week.
    - Full logs of all submissions.
    - Convenience feature, **not** core value proposition.

We do **not** design the core product around "spray and pray" behavior.

6. Go-to-Market Implications

6.1 Beachhead GTM (MBAs)

- **Focus:** Top global MBA programs + strong regional schools.
- **Channels:** Career centers, MBA clubs (tech/consulting/product), international student groups, alumni networks.
- **Messaging:**
    - “AI you can trust for elite applications.”
    - “Use AI without risking your visa or your reputation.”

6.2 Pricing and Packaging

- Premium positioning; price closer to coaching / career services than generic apps.
- Offer school/cohort deals and referral programs to accelerate adoption.

6.3 Brand Values

- **Safety over speed.**
- **Truth over hype.**
- **Control over automation.**

This must show up in product UX (diffs, warnings, logs) and in marketing copy.

7. Final Recommendation

Proceed with **Phase 1 – Safe Harbor MVP** targeted at the Super-Persona: international, pivoting MBA Snipers.

Near-term actions:

- Scope and prototype **Sponsorship Intelligence** (employer prior + JD NLP + risk labels + user rules).
- Design and validate the **Diff Dashboard** UX with real MBA résumés and job descriptions.
- Implement the **Translator + tooltip** experience so candidates learn as the system rewrites.

If we win trust and usage with this narrow, high-stakes audience, we establish a defensible wedge that generic trackers and auto-apply bots cannot match, because our architecture and brand are built around one thing: **safety and integrity for high-stakes candidates.**

3. PRD

PATHFINDER – SAFE HARBOR COPILOT
PRODUCT REQUIREMENTS DOCUMENT (FINAL MVP, FIREBASE / NEXT / TS)

1. Product summary

Pathfinder is a free, desktop-first web app that helps high-stakes candidates (MBAs, internationals, pivoters) generate multiple truthful, visa-aware, high-quality resumes for specific jobs with minimal friction.

Core idea:

- User signs in with Google.
- Uploads and confirms their resume once (Fact Base).
- Searches or pastes jobs, sees visa risk and match potential.
- Selects up to 5 jobs and generates tailored resumes sequentially (top → bottom).
- Reviews differences where needed, approves, downloads.
- History tracks which jobs were generated/downloaded/applied.

2. Objectives

- Reduce time to create tailored resumes for 3–5 roles in a session.
- Increase trust that AI does not change facts (companies, dates, metrics).
- Help internationals avoid non-sponsoring or risky roles.
- Keep UX simple: after onboarding, users mainly use a single Jobs screen.

3. Users and personas

Primary:

- "Pivoting Snipers": international and/or pivoting MBAs applying to selective roles.

Secondary:

- Domestic MBAs targeting elite roles (reputation sensitive).
- Stealth candidates quietly exploring new roles.

Key needs:

- Avoid visa traps.
- Understand potential (fit) per position.
- Scale tailoring without rewriting everything.
- See exactly what changed and why.

4. Value proposition and principles

Value proposition:
"Upload your resume once, then safely generate tailored resumes for up to five jobs at a time, with clear visa risk, match potential, and full visibility into every change."

Product principles:

- Safety over automation: no hidden changes to facts.
- Simplicity: single primary Jobs surface post-onboarding.
- Transparency: explicit diff and reasoning.
- Controlled scale: max 5 jobs per batch, sequential processing.
- RAG + AI crew: use LangChain and a crew-style architecture to improve quality using the resume, Fact Base, and history.

5. Core user flows

5.1 Onboarding and Fact Lock (first login)

- User logs in via Google SSO.
- If no resume present:
  - Sees “Welcome to Pathfinder” with upload card (drag-and-drop + upload button).
  - Uploads resume (PDF/DOCX).
- System parses resume into structured Fact Base and shows Fact Audit:
  - Left: profile summary (name, visa status, target roles), editable.
  - Right: experiences list (company, role, dates, bullets), metrics highlighted, low-confidence items flagged.
  - Sticky footer explains “truth lock” and provides “Confirm & lock facts”.
- On confirming:
  - Fact Base is LOCKED.
  - Resume and facts indexed into vector store for RAG.
  - User is redirected to Jobs.

5.2 Resume update

- Profile & Resume screen has “Update resume & Fact Base”:
  - Upload → parse → Fact Audit → lock (same flow).
  - New Fact Base becomes active; RAG indices updated.
  - Existing history remains tied to old versions.

5.3 Jobs flow (daily usage)

- Jobs screen is default home:
  - Top: search and filters.
  - Middle: jobs table with visa risk and match potential.
  - Bottom/top sticky bar: batch generate.
- User:
  - Searches (title/company, location, remote filter) or pastes JD/URL.
  - Uses filters (“Hide high visa risk”, “Hide low match”).
  - Sees Visa Risk pill (LOW/HIGH/UNCLEAR) and Match Score (percentage + bar, High/Medium/Low label).
  - Selects jobs via checkboxes (up to 5).
  - Clicks “Generate X resumes”.

5.4 Generation, review and download

- Backend:
  - Creates a Batch for the user + selected jobs.
  - Enqueues generation tasks per job in UI order.
  - Processes jobs sequentially (1 → 2 → …).
- User:
  - Sees per-row status (“Queued”, “Generating…”, “Ready”, “Error”) and batch progress (“2 of 5 generated”).
  - For a “Ready” row:
    - Presses “Download” (quick path) or “Review” (opens diff).
- Diff view:
  - Shows JD, original snapshot, tailored resume with red/green diff.

    - Shows change summary (metric changes, key translations).
    - User can accept/reject changes, edit text, and “Save & approve”.
- From diff or Jobs:
    - User downloads resume (PDF/DOCX/text).

5.5 History and application tracking

- History screen:
    - Table: job, company, location, date generated, visa risk pill, match score, status (Generated/Downloaded/Applied), actions (View diff, Download), toggle “Applied”.
- User:
    - Marks “Applied” for jobs they submitted.
    - Re-opens diff and re-downloads.

6. Functional requirements by module

6.1 Authentication and user profile

- Google SSO only.
- Store:
    - userId, email, displayName from Google.
    - currentFactBaseId.
- Handle logout and account deletion (removes Fact Base, vector entries, drafts, history).

6.2 Resume ingest, Fact Base, and RAG

- Parse uploaded resume into structured TypeScript models:
    - Experience: { id, company, role, location, startDate, endDate, bullets[] }.

    - Metrics: { id, value, unit, context, experienceId }.
    - Skills: string[].
- Fact Audit UI shows this structure; user can:
    - Edit fields inline.
    - See metrics/dates highlighted.
    - See low-confidence fields flagged.
- When user clicks Lock:
    - Fact Base marked LOCKED.
    - Hard constraints:
        - Metrics, dates, and company names cannot be changed by the AI.
    - Backend writes Fact Base into Firestore and pushes embeddings to vector store.
- RAG:
    - Store resume text, Fact Base segments, and possibly approved drafts as documents in a vector store.
    - LangChain retriever uses this for tailoring and matching.

6.3 Job ingestion and enrichment

- Dual ingest:
    - Search:
        - API or scraper-backed JobSource that returns job list with JDs.
    - Manual:
        - “Paste job description” dialog:
            - Paste URL.

        - Or paste full JD text.
    - Creates a job entry with jdText.
- For each job:
  - Visa Guard agent:
    - Reads jdText and classifies sponsorshipRisk = LOW / HIGH / UNCLEAR.
    - Returns flaggedText array with any concerning sentences.
  - Match Analyzer agent:
    - Uses RAG to retrieve relevant facts.
    - Computes matchScore (0–100) and label (High/Medium/Low).
    - Generates fitReasons: 2–3 short bullets.

6.4 Jobs table and selection (UI requirements)

- Columns:
  - Selection checkbox.
  - Job Title (blue link-style) and Company.
  - Location.
  - Visa Risk pill (green/amber/red) + tooltip with flaggedText.
  - Match Score: percentage + bar; label High/Med/Low.
  - Posted Date.
  - Actions (Generate/Generating/Review/Download/Retry).
- Selection logic:
  - User can check multiple rows.
  - When 5 rows are selected:

        - Additional checkboxes disabled with tooltip “Limit 5 per batch”.

- Batch bar:
    - Appears when >0 jobs selected.
    - Shows “Generate X resumes” and caption “Jobs will be processed one by one from top to bottom.”
    - Shows batch progress while generating.

6.5 Batch processing and AI crew

- On “Generate X resumes”:
    - Create Batch document (userId, factBaseId, jobIds[], createdAt).
    - Create Draft placeholder documents and enqueue tasks in order.
- Per job in batch:
    - Retriever agent (LangChain) queries vector store with job JD.
    - Analyzer agent validates matchScore & fitReasons.
    - Tailor agent:
        - Generates tailored resume text using Fact Base + retrieved snippets + JD.
        - Adheres to hard constraints (no new companies, no changed metrics).
    - QA agent:
        - Compares original vs tailored resume.
        - Identifies:
            - bulletsChanged (count),
            - metricsChanged (boolean),
            - metricChanges[] (from/to),

            - translationAlerts[] (from → to, reason).
        - Sets safetySignals (metricIntegrity: PASS/ALERT, hallucinationRisk).
    - Persist updated Draft; update job row status.

6.6 Diff review and approval

- Diff view:
    - Header: job title, company, status pill (“Metric alert” or “Verified safe”), actions (“Cancel”, “Save & approve”, “Download”).
    - Left column: JD card.
    - Right: original snapshot and tailored resume with red/green diff styling.
    - Change summary area listing:
        - Metric changes with original vs suggested and Accept/Reject toggles.
        - Key phrase translations with reason and Accept/Reject or edit.
- On approve:
    - Draft status set to APPROVED.
    - Safety pill becomes “Verified safe”.
    - User can export.

6.7 Export and history

- Export:
    - Generate PDF, DOCX, and plain text from approved or ready drafts.
- History:
    - Each job they’ve generated for is retained:
        - jobId, batchId, timestamps, visa risk, match score, status, applied flag.

    - History table view supports:
        - Viewing diff.
        - Downloading exports.
        - Toggling “Applied”.

7. Non-functional requirements

- Performance:
    - Fact Audit appears within a few seconds of upload.
    - Job search and enrichment responds within a few seconds per query.
    - Batch of 5 resumes completes in ~90 seconds or less, with per-job updates.
- Reliability:
    - If search fails, show error and offer manual JD paste as fallback.
    - If a single draft generation fails, that job shows ERROR with “Retry”.
- Security:
    - Use Google SSO; no password management.
    - Encrypt data at rest and in transit.
    - Support account deletion and data purge.
- Safety:
    - Hard constraint enforcement at both prompt and code level.
    - QA agent checks and flags any metric change or hallucination risk.

8. Technical implementation (for Firebase / Next / TypeScript)

Stack decisions:

- Framework: Next.js 14 with App Router (app/ directory pattern).

- Language: TypeScript (strict). Shared domain types for FactBase, Job, Batch, Draft.
- Directory: use src/ (src/app, src/components, src/lib, src/server).
- Styling: Tailwind CSS.
- Linting: ESLint enabled.

Deployment and runtime:

- Frontend: Next.js app deployed via Firebase Hosting with SSR using Next's App Router.
- Backend:
    - Next.js server components and route handlers for SSR and simple APIs.
    - Firebase Cloud Functions (Node/TS) as needed for long-running or queued tasks (generation).
    - Cloud Tasks or an equivalent queue for batch generation per job.
- Data:
    - Firestore for:
        - users, factBases, jobs, batches, drafts, history.
    - Storage (e.g., Cloud Storage) for uploaded resume files and generated artifacts if needed.
    - Vector store:
        - Use a Firestore-backed or dedicated vector store integrated via LangChain (e.g., stored embedding vectors keyed by documentId).

App Router structure (high-level):

- src/app/(app)/jobs/page.tsx
    - Main Jobs screen: search, filters, table, batch bar.
- src/app/(app)/history/page.tsx

    - History table.
- src/app/(app)/profile/page.tsx
    - Profile & Resume view, “Update resume & Fact Base”.
- src/app/api/resume/upload/route.ts
    - Handles file upload and triggers parsing.
- src/app/api/factbase/lock/route.ts
    - Locks Fact Base.
- src/app/api/jobs/search/route.ts
    - Job search endpoint using JobSource.
- src/app/api/jobs/paste/route.ts
    - Manual JD ingestion.
- src/app/api/batches/create/route.ts
    - Creates batch and enqueues tasks.
- src/app/api/drafts/[id]/route.ts
    - Fetch/update draft, change summary, approvals, exports.

Code organization:

- src/lib/types:
    - Domain interfaces: FactBase, Experience, Metric, Job, Enrichment, Batch, Draft, ChangeSummary.
- src/lib/rag:
    - Embedding and vector store helpers.
- src/lib/agents:

    - VisaGuardAgent, MatchAnalyzerAgent, TailorAgent, QAAgent built with LangChain.
- src/lib/jobSource:
    - JobSource interface and implementations for integrated search and manual paste.
- src/components:
    - UI primitives: JobTable, FactAuditForm, DiffView, HistoryTable, Pills, Buttons.

ESLint:

- Enforce TypeScript best practices, no implicit any, no unused vars.
- Enforce clean separation between server and client components in App Router.

Tailwind:

- Use Tailwind for layout and styling of the dashboard, tables, pills, and diff UI.
- Define utility classes for visa risk pills, match score bars, and diff highlights (green added, red removed).

AI integration:

- LangChain (TypeScript) orchestrates:
    - Retriever (RAG queries).
    - Analyzer (match potential).
    - Tailor (resume generation).
    - QA (diff analysis and safety).
- Models can be Google AI / Gemini plugged via LangChain from Firebase environment.

This updated PRD reflects the chosen stack (Firebase Studio, Next.js 14 App Router, TypeScript, Tailwind, ESLint) and provides enough context for product, UX, and engineering to collaborate and for Google Firebase Studio to scaffold code aligned with the product intent.

4. Technical Dive-in

Pathfinder – Safe Harbor Copilot
Technical Design / Dive-in (MVP)

1. Scope and assumptions

This document is for engineering (frontend, backend, AI) and for Firebase Studio scaffolding. It expands the PRD into concrete architecture, data models, APIs, and flows.

Scope:

- MVP web app, desktop-first, built with:
  - Next.js 14 (App Router, TypeScript, Tailwind, ESLint, src/ directory).
  - Firebase: Hosting, Authentication, Firestore, Cloud Functions, Cloud Storage (optional), Cloud Tasks (for queue).
  - LangChain (TypeScript) orchestrating a small “crew” of AI agents.
- AI models assumed to be Gemini or similar LLMs accessible from the Node.js environment.

Non-goals (MVP):

- No auto-apply to job boards.
- No multi-tenant admin dashboards.
- No advanced analytics / reporting for users (basic history only).
- No mobile app (responsive web only, optimized for desktop).

---

1. High-level architecture

1.1 Frontend

- Next.js 14 with App Router using src/app.
- Pages (route segments):
  - src/app/(app)/jobs/page.tsx → Jobs main screen.
  - src/app/(app)/history/page.tsx → History.
  - src/app/(app)/profile/page.tsx → Profile & Resume.
  - src/app/(auth)/signin/page.tsx → Sign-in redirect/guard (optional).
- UI components:
  - Layout components: AppShell (sidebar, header), ToastProvider.
  - Domain components:
    - FactAuditForm
    - JobsTable
    - JobFilters
    - BatchBar
    - DiffView
    - HistoryTable
    - ProfileSummary
- Styling: Tailwind CSS, with design tokens for pills, match bars, diff colors.

1.2 Backend

- Next route handlers for simple API endpoints:
  - src/app/api/resume/upload/route.ts
  - src/app/api/factbase/lock/route.ts
  - src/app/api/jobs/search/route.ts

    - src/app/api/jobs/paste/route.ts
    - src/app/api/batches/create/route.ts
    - src/app/api/drafts/[draftId]/route.ts
    - src/app/api/history/route.ts
- Cloud Functions (Node/TS) for:
    - Long-running AI tasks (batch generation worker).
    - Vector embedding & similarity search (if we don’t want to run heavy compute in route handlers).
- Cloud Tasks (or Firestore-trigger based queue) for:
    - Job-level draft generation, ensuring sequential processing per batch.

1.3 Data & AI

- Firestore collections:
    - users
    - factBases
    - jobs
    - batches
    - drafts
    - history (or history embedded in drafts/batches)
- Vector store:
    - Per-user documents (resume, Fact Base segments, maybe successful drafts) with embeddings stored in Firestore documents.
    - Similarity search implemented by:
        - Retrieving candidate docs for a user.

        - Computing cosine similarity in a worker (dataset per user is small).

- LangChain in Cloud Functions:
    - VisaGuardAgent
    - MatchAnalyzerAgent
    - TailorAgent
    - QAAgent

---

2. Authentication and authorization

2.1 Google SSO

- Firebase Auth with Google provider.
- Frontend:
    - Next.js auth client integrates with Firebase Auth SDK.
    - On sign-in: store Firebase ID token; Next route handlers verify it via Admin SDK.
- Route guards:
    - Protected app routes (jobs, profile, history) verify user is authenticated.
    - Redirect unauthenticated users to sign-in page.

2.2 Authorization model

- userId = auth.uid from Firebase Auth.
- Firestore documents always include userId field.
- Firestore security rules:
    - users: user can read/write only their own profile document.

    - factBases/jobs/batches/drafts/history: user can read/write docs where doc.userId == auth.uid.
- Backend (Cloud Functions, route handlers) verify userId from ID token and enforce doc ownership.

---

3. Data model and schemas (TypeScript interfaces)

TypeScript interfaces will live in src/lib/types.

3.1 User

```
interface UserProfile {
 id: string; // auth.uid
 email: string;
 displayName: string | null;
 currentFactBaseId: string | null;
 createdAt: string; // ISO
 updatedAt: string; // ISO
 }
```

3.2 Fact Base

```
interface FactBase {
 id: string;
 userId: string;
 version: number;
 status: 'LOCKED' | 'DRAFT';
 lockedAt?: string;
 profile: {
 name: string;
 visaStatus?: string;
 targetRoles: string[];
 };
 experiences: Experience[];
 metrics: Metric[];
 skills: string[];
 }
```

```
interface Experience {
 id: string;
 company: string;
 role: string;
 location?: string;
 startDate: string; // ISO
 endDate?: string; // ISO or 'Present'
 bullets: string[];
 }

interface Metric {
 id: string;
 value: string; // e.g. "20%", "$1.2M"
 unit?: string;
 context: string; // "Revenue growth", "Team size"
 experienceId: string;
 }
```

3.3 Job and enrichment

```
interface Job {
 id: string;
 userId: string;
 source: 'SEARCH' | 'MANUAL';
 externalId?: string; // external job board id
 title: string;
 company: string;
 location?: string;
 url?: string;
 postedDate?: string; // ISO
 jdText: string; // full JD text
 enrichment?: JobEnrichment;
 createdAt: string;
 updatedAt: string;
 }

interface JobEnrichment {
 sponsorshipRisk: 'LOW' | 'HIGH' | 'UNCLEAR';
 flaggedText: string[]; // sentences about citizenship/visa
 matchScore: number; // 0–100
 matchLabel: 'HIGH' | 'MEDIUM' | 'LOW';
 fitReasons: string[]; // explanations, short
 }
```

3.4 Batch and draft

```
interface Batch {
 id: string;
 userId: string;
 factBaseId: string;
 jobIds: string[];
 createdAt: string;
 status: 'PENDING' | 'IN_PROGRESS' | 'COMPLETED' | 'ERROR';
 }

interface Draft {
 id: string;
 userId: string;
 batchId: string;
 jobId: string;
 factBaseId: string;
 status: 'QUEUED' | 'PROCESSING' | 'READY' | 'APPROVED' | 'ERROR';
 errorMessage?: string;
 resumeOriginal: string; // snapshot
 resumeTailored: string; // current tailored text
 changeSummary?: ChangeSummary;
 safetySignals?: SafetySignals;
 createdAt: string;
 updatedAt: string;
 }

interface ChangeSummary {
 bulletsChanged: number;
 metricsChanged: boolean;
 metricChanges: MetricChange[];
 translationAlerts: TranslationAlert[];
 }

interface MetricChange {
 metricId: string;
 from: string;
 to: string;
 }

interface TranslationAlert {
 id: string;
 from: string;
 to: string;
 reason: string;
 }

interface SafetySignals {
 metricIntegrity: 'PASS' | 'ALERT';
 hallucinationRisk: 'LOW' | 'MEDIUM' | 'HIGH';
 }
```

3.5 History (option 1: separate collection)

```
interface HistoryEntry {
 id: string;
 userId: string;
 jobId: string;
 draftId: string;
 generatedAt: string;
 lastDownloadedAt?: string;
 applied: boolean;
 }
```

---

4. RAG and AI crew

4.1 Documents for RAG

Per user:

- Resume full text.
- Fact Base segments:
  - Each experience text.
  - Each bullet.
  - Important metrics context.
- Potentially approved drafts for learning (later).

We store:

```
interface RAGDocument {
 id: string;
 userId: string;
 type: 'RESUME' | 'EXPERIENCE' | 'BULLET';
 sourceId?: string; // experienceId, etc.
 text: string;
 embedding: number[];
 createdAt: string;
}
```

4.2 Vector storage

MVP approach:

- Store RAGDocument and embedding vector in Firestore (small per user).
- When retrieving:
  - Pull all docs for user (or subset by type).
  - Compute cosine similarity in the worker function.
  - Select top K (e.g., 10).

4.3 AI agents (LangChain)

We implement a small crew:

VisaGuardAgent:

- Input: Job.jdText.
- Output: sponsorshipRisk, flaggedText[].

MatchAnalyzerAgent:

- Input: Job.jdText + top K RAG docs + Fact Base summary.
- Output: matchScore, matchLabel, fitReasons[].

TailorAgent:

- Input: Fact Base, top K RAG docs, Job.jdText.
- Output: tailored resume text; structured info about translation (for QA if possible).

QAAgent:

- Input: original resume text (or relevant sections), tailored text, Fact Base.
- Output: ChangeSummary, SafetySignals; ensures metric integrity and no new companies.

Each agent is implemented as a LangChain chain or tool, with system prompts enforcing:

- No new metrics/dates/companies.
- Must flag any change to metric phrasing.

---

5. API design and route handlers

All routes are under src/app/api with App Router.

Authentication:

- Every route that touches user data expects an Authorization header with Firebase ID token.
- Route handler verifies token with Admin SDK and resolves userId.

Key endpoints (conceptual):

POST /api/resume/upload

- Body: form-data with file.
- Actions:
    - Store raw resume file (optional) in Cloud Storage.
    - Extract text via parser.
    - Create Fact Base (status = DRAFT) with parsed content.
    - Return FactBase JSON for Fact Audit.

POST /api/factbase/lock

- Body: edited FactBase JSON.
- Actions:
  - Validate and write to Firestore with status = LOCKED, increment version.
  - Generate or update RAG documents + embeddings.
  - Update user.currentFactBaseId.
  - Return locked FactBase.

GET /api/jobs/search

- Query: q, location, remoteOnly, paging.
- Actions:
  - Call JobSource (search) integration.
  - For each job:
    - Create Job doc (if not exists).
    - If jdText present, call VisaGuardAgent and MatchAnalyzerAgent.
  - Return list of Job with enrichment.

POST /api/jobs/paste

- Body: { title, company, location, url?, jdText }.
- Actions:
  - Create Job doc.
  - Call VisaGuardAgent and MatchAnalyzerAgent.
  - Return Job.

POST /api/batches/create

- Body: { jobIds: string[] }.
- Actions:
    - Validate ≤5 jobs, all owned by user.
    - Create Batch doc.
    - For each job:
        - Create Draft doc with status QUEUED.
        - Enqueue Cloud Task for that job (includes batchId, draftId).
    - Return Batch and Draft IDs.

GET /api/drafts/[draftId]

- Returns Draft with changeSummary and safetySignals.

PATCH /api/drafts/[draftId]

- Body: updated resumeTailored, Accept/Reject decisions for metrics/translations, status (APPROVED).
- Actions:
    - Validate user ownership.
    - Update draft and recalc safetySignals if needed.

GET /api/history

- Returns History entries + resolved Job title/company.

POST /api/history/applied

- Body: { historyId, applied }.
- Sets applied flag.

---

6. Batch processing and queue details

We use Cloud Tasks to ensure sequential processing per batch.

Task payload:

- { userId, batchId, draftId, jobId }

Worker Cloud Function:

- Verifies userId, loads FactBase, Job, Draft.
- Marks Draft status PROCESSING, Batch status IN_PROGRESS.
- Steps:
    - If Job.enrichment missing or outdated, call VisaGuardAgent + MatchAnalyzerAgent.
    - Call Retriever (RAG) for user + job.
    - Call TailorAgent to generate tailored resume.
    - Call QAAgent to compute changeSummary + safetySignals.
- Save Draft:
    - resumeOriginal snapshot (from user's base resume).
    - resumeTailored from TailorAgent.
    - changeSummary, safetySignals.
    - status READY or ERROR.
- If all Drafts in Batch are READY/ERROR:
    - Batch status COMPLETED or ERROR.

Sequential constraint:

- Options:
    - Use a small orchestration:
        - First task processes job1 and, on completion, enqueues job2, etc.
    - Or use a “position” field and a Cloud Function that only processes the lowest position with status QUEUED.
- For simplicity, MVP can accept “logically sequential” (we process one by one per batch via orchestration function) while still allowing multiple batches from different users in parallel.

---

7. Frontend architecture

7.1 Layout and routing

- AppShell component handles sidebar, header, toasts.
- (app)/jobs/page.tsx:
    - Fetch Fact Base (to know lock state).
    - Fetch Jobs (filtered).
    - Render Filters + JobsTable + BatchBar.
    - Use client-side data fetching (React Query or custom hooks) hitting API routes.
- (app)/history/page.tsx:
    - Fetch history, join with Jobs.
- (app)/profile/page.tsx:
    - Fetch FactBase + UserProfile.
    - Provide “Update resume & Fact Base” entry point.

7.2 State management

- Use React Query (or similar) for:
    - Jobs, Drafts, History, FactBase.
- Use local state for:
    - Current selections (jobIds).
    - Modal open/closed (diff view, paste JD dialog).

7.3 UI components

Key components:

- FactAuditForm:
    - Expects FactBase data and emits edited FactBase JSON.
- JobsTable:
    - Props: jobs[], selection state, onSelectChange, onActionClick.
- BatchBar:
    - Props: selectedCount, onGenerate, batchProgress.
- DiffView:
    - Props: draft (original, tailored, changeSummary, safetySignals).
    - Allows inline editing and acceptance/rejection toggles.
- HistoryTable:
    - Props: history records, onAppliedToggle.

---

8. Observability, config, testing

8.1 Config

- Environment variables:
    - FIREBASE_PROJECT_ID, FIREBASE_PRIVATE_KEY, etc.
    - AI_MODEL_API_KEY / endpoint.
- Config for queue:
    - Cloud Tasks queue name.
    - Rate limits (e.g., max tasks/sec).

8.2 Logging

- Cloud Functions log:
    - Each generation step (VisaGuard, MatchAnalyzer, Tailor, QA).
    - Errors tagged with userId and jobId (no PII beyond IDs).
- Frontend:
    - Minimal console logging; errors posted to backend logging endpoint if needed.

8.3 Testing

- Unit tests:
    - Pure functions in src/lib (diff computation, changeSummary generation).
- Integration tests:
    - Route handlers with mocked Firebase Admin and AI clients.
- Manual flows:
    - End-to-end sanity for:
        - First-time onboarding → Fact Audit → Jobs.
        - Search and manual JD paste.

- Generate 1 and 5 jobs.
- Diff view with metric alert.
- History and applied toggle.

5. UI/UX

PATHFINDER – SAFE HARBOR COPILOT
UI/UX DESIGN SPEC (MVP)

Owner: Chief of UX/Design
Audience: Design + Frontend + PM + Firebase Studio

---

1. Design goals and principles

1.1 Experience goals

- Feel “safe and in control”: users must feel AI is assisting them, not acting on their behalf behind their back.
- Minimize friction: once onboarding is done, the main workflow should be obvious from the Jobs screen.
- Make risk and potential obvious: visa risk and match potential should be instantly scannable from the jobs list.
- Support both “quick download” and “deep review”: some users will just trust and download; others will want to inspect every change.

1.2 UX principles

- One primary surface post-onboarding: Jobs is the “home base”.
- Progressive detail: summary first (pills, scores), details on demand (diff view, tooltips).
- Opinionated defaults, few choices: user should not need to configure complex options to get value.
- Honest visuals: don’t hide warnings; make risk and changes visually explicit.

1.3 UI principles

- Desktop-first dashboard layout (left sidebar, right content).
- Clear visual hierarchy: large titles, consistent section headings, table headers.
- Consistent components: one pill style, one button style family, one table style.
- Use color to signal risk and status, not decoration.

---

2. Information architecture

Main navigation (left sidebar):

- Dashboard (placeholder/future; may redirect to Jobs for now)
- Jobs (default after onboarding)
- History
- Profile & Resume
- Settings/Help (can be minimal in MVP)

Primary user journeys:

- First-time: Sign in → Upload resume → Fact Audit (lock) → Jobs.
- Returning: Sign in → Jobs → Search/select → Generate → Review/download → History.

Screens (top-level):

1. Onboarding & Fact Audit
2. Jobs
3. Diff Review (can be full-screen or large modal on top of Jobs)
4. History
5. Profile & Resume

All within a shared App Shell (sidebar + header).

---

3. Visual language and tokens

3.1 Layout

- Desktop width target: design for 1440px width; main content max-width ~1200px inside shell.
- Layout structure:
    - Left: 240–280px sidebar.
    - Right: content with 24–32px padding.
- Use column-based layout for content:
    - Top: title + filters.
    - Middle: tables/cards.
    - Bottom: sticky batch bar or footer actions where needed.

3.2 Color (suggested tokens, not exact hex values)

- Background: light gray (e.g., #F5F5F7).
- Surface: white (cards, tables).
- Primary: blue/teal (e.g., #2563EB / #0F766E; choose one primary).
- Text:
    - Primary text: near-black (#111827).
    - Secondary text: gray (#6B7280).
- Status / pills:
    - Visa-safe / Verified: green (#10B981).
    - Visa-risk / Metric Alert: red (#EF4444).
    - Unclear / Amber: amber (#F59E0B).

- Diff:
  - Additions: very light green background (#ECFDF3) + darker green text.
  - Deletions: very light red background (#FEE2E2) + darker red + strikethrough.

3.3 Typography

- Font: Inter or similar.
- Sizes:
  - Page title: 24–28px, semibold.
  - Section headings: 18–20px, semibold.
  - Table headers: 12–14px, medium, uppercase or small caps.
  - Body: 14–16px regular.
  - Helper/annotation: 12–13px.
- Line height: 1.4–1.6.

3.4 Components
We should define reusable components and Tailwind utility patterns for:

- Buttons:
  - Primary (filled, blue).
  - Secondary (outline or subtle).
  - Tertiary/link (text).
- Pills/Badges:
  - Status variants: green, red, amber, gray.
- Inputs:
  - Text field, search field, dropdown, toggle.

- Table:
    - Header row (bold, slightly shaded background).
    - Row states (hover, selected, disabled checkbox).
- Cards:
    - Simple card with title, body, actions.
- Toasts:
    - Top-right, short text, severity variants.

---

4. Screen-by-screen design

4.1 App Shell

Sidebar:

- Top area:
    - Small logo mark + “Pathfinder” wordmark.
    - Subtitle text: “Safe Harbor Copilot” (smaller, muted).
- Nav items:
    - Each item: icon + label; active state with solid or left border highlight.
    - Items: Dashboard, Jobs, History, Profile & Resume, Settings (optional).
- Scrollable if content grows (future).

Header (top of content):

- Left: context-specific title (“Jobs”, “History”, etc.).
- Right:
    - “Signed in with Google” text in small font.

    - Avatar circle with user initial.
    - Notification bell icon.

Global:

- Toast region top-right.
- Optional global progress bar under header for long-running operations.

4.2 Onboarding – Upload & Fact Audit

Upload screen:

- Single central card:
    - H1: “Welcome to Pathfinder”.
    - Body text: 1–2 lines about control and safety.
    - Drag-and-drop zone with dashed border and icon.
    - Primary button: “Upload resume (PDF/DOCX)”.
    - Small helper text: “We’ll show you what we understood before using it.”

Fact Audit screen:

- Header within content:
    - Title: “Review your experience”.
    - Subtext: “Confirm what we’ll use as your source of truth.”
- Layout:
    - Two columns:
        - Left column (fixed width ~320px):
            - Card: “Profile”:

            - Name field.
            - Visa status field (text or dropdown).
            - Target role tags (chips).
            - Badge: “Facts not locked yet” (amber) or “Facts locked” (green after lock).
        - Right column (flex):
            - Scrollable list of “Experience” cards.
            - Each card:
                - Company (bold), role and location.
                - Dates (start – end).
                - Bullets, vertically stacked.
                - Metrics highlighted in-line (e.g., bold/colored).
                - Low-confidence fields with a small orange icon and tooltip “Parsed with low confidence, please check”.
                - Edit icon per field or in-place editing.
- Footer (sticky at bottom):
    - Left: small explanatory text about not changing companies/dates/metrics without approval.
    - Right: “Back” (secondary) and “Confirm & lock facts” (primary).

After “lock”:

- Return user to Jobs screen with a one-time toast: “Facts locked. You can update them anytime from Profile & Resume.”

4.3 Jobs Screen (core)

Header row:

- Title: “Jobs”.
- Subtitle: “Search roles, see visa risk and match potential, then generate tailored resumes.”

Filters/search:

- Horizontal row:
    - Search text input: “Search title or company…”.
    - Location input.
    - Toggle: “Remote only”.
    - Filter chips/toggles:
        - “Hide high visa risk”.
        - “Hide low match potential”.
    - Button: “Paste job description” (opens small modal with title/company/location + JD text fields).
    - Right side label: “You can generate up to 5 resumes at once.”

Jobs table:

- Structure:
    - Column 1: checkbox.
    - Column 2: Job title (link style) and smaller company name below or beside.
    - Column 3: Location.
    - Column 4: Visa Risk pill.
    - Column 5: Match Score (number + bar).
    - Column 6: Posted date.

    - Column 7: Actions.

Visa Risk pill:

- Text:
    - “Visa-safe”.
    - “Risky”.
    - “Unclear”.
- Color: green / red / amber accordingly.
- Tooltip: show flagged sentence, e.g., “U.S. citizenship required.”

Match Score:

- Show “82%” plus a horizontal bar representing the score.
- Optional small label under the percent: “High match” / “Medium” / “Low”.

Row status:

- Subtle secondary text under actions or near them: “Not generated”, “Queued”, “Generating…”, “Ready”, “Error”.
- For “Generating…” show a small spinner inline.

Actions:

- Not generated: primary small button “Generate”.
- Generating: disabled “Generating…” button.
- Ready: “Review” (secondary) + “Download” (primary).
- Error: “Retry” text button with error icon.

Selection:

- Checkbox behavior:
    - When 0 selected: all checkboxes enabled.
    - When 1–4 selected: all other checkboxes enabled.
    - When 5 selected: any additional checkbox is disabled and shows tooltip “Limit 5 per batch”.
- Selected rows have a subtle background highlight.

Batch bar:

- Sticky bar at bottom (above footer):
    - Visible only when ≥1 job selected.
    - Left: selection summary (“3 jobs selected”).
    - Center (optional): text “Jobs will be processed one by one from top to bottom.”
    - Right: primary button “Generate 3 resumes”.
- During generation:
    - Bar shows progress: “Generating 2 of 5 resumes…” with simple progress indicator.

Empty / error states:

- If no jobs found:
    - Illustration or icon, text “No jobs found. Try adjusting filters or paste a job description.”
    - “Paste job description” button.
- If search API error:
    - Red banner above table: “We couldn’t fetch jobs right now. Please try again or paste a job description manually.”

4.4 Diff Review (Resume Review)

Trigger:

- From Jobs table: clicking “Review” on a ready job row.
- From History: clicking “View diff”.

Layout (full screen or large centered modal):

- Header:
    - Title: “Review tailored resume”.
    - Subheading: “[Job Title] at [Company]”.
    - Status pill: “Metric alert” (red) or “Verified safe” (green).
    - Right-side buttons: “Cancel” (secondary), “Save & approve” (primary).
- Body:
    - Left column:
        - JD card:
            - Job title and company.
            - Scrollable JD text.
            - Key phrases highlighted (e.g., responsibilities).
    - Right column (split):
        - Top: “Original resume” (read-only).
            - Only show relevant section(s) – e.g., one or two roles.
        - Bottom: “Tailored resume” (editable, diff-styled).
            - Bullets / paragraphs where:
                - Deleted text: red, strikethrough, with leading “–”.

            - Added text: green highlight, with leading “+”.
            - Unchanged: neutral.
        - Allow direct text editing.

Change summary panel:

- Either on the right side or below the tailored card:
    - Section “Metric changes”:
        - Each row: label, from → to, Accept/Reject toggle (or simple checkboxes).
        - If no metric changes, show “No metric changes detected.”
    - Section “Key translations”:
        - Each row: from phrase → to phrase, short reason (“Aligns to JD term ‘stakeholders’”).
        - Accept/Reject or editable link.
- Aim: give user a quick mental model of what changed without reading full text.

Footer:

- Full-width “Approve changes” / “Save & approve” button.
- Secondary “Download PDF” if appropriate.

State after approval:

- Back in Jobs/History:
    - Status pill becomes “Verified safe”.
    - Export actions remain.

4.5 History

Layout:

- Title: “History”.
- Subtitle: “See where you’ve generated resumes and track your applications.”
- Table structure:
    - Job (title + company).
    - Location.
    - Date generated.
    - Visa Risk pill.
    - Match Score.
    - Status (Generated / Downloaded / Applied).
    - “Applied” toggle (checkbox or small button).
    - Actions: “View diff”, “Download”.

Empty state:

- Message: “No history yet. Generate your first tailored resume from the Jobs page.”
- Button: “Go to Jobs”.

4.6 Profile & Resume

Layout:

- Title: “Profile & Resume”.
- Main card:
    - Name, email, visa status, target roles (chips).
    - Badge: “Facts locked”.
    - Short paragraph explaining:

        - "These locked facts are used as your single source of truth. We won't change your companies, dates, or numbers without your approval."
    - Button: "Update resume & Fact Base".
- On clicking update:
    - Reopen the upload + Fact Audit flow.

---

5. Microcopy and tone

Tone:

- Calm, direct, professional.
- Avoid hype; emphasize control and clarity.

Examples:

- Upload helper: "You stay in control of every change."
- Fact lock explanation: "We'll never change your companies, dates, or numbers without your approval."
- Visa risk tooltip: "This job description mentions 'U.S. citizens only'. This may not be suitable if you require sponsorship."
- Match explanation: "Strong match: aligns with your stakeholder management and data analysis experience."

Error messages:

- Clear and actionable: "We couldn't generate this resume. Please try again. If the problem persists, adjust your resume or job description."

---

6. Interaction details and states

- Buttons:

    - Include disabled states for loading/generation.
- Checkboxes:
    - Visual feedback for hover, focus, disabled.
- Modals:
    - Standard overlay with ESC and click-outside dismissal (except for critical actions like fact lock – require explicit choice).
- Loading:
    - Use inline spinners for row-level actions and a thin header bar for long global operations.
- Keyboard:
    - Support TAB navigation between inputs.
    - ENTER to submit forms where logical (search, filters, fact lock).

---

7. Accessibility

- Color contrast:
    - Ensure all text and pill combinations meet WCAG AA contrast.
- Keyboard navigation:
    - All interactive elements tabbable.
    - Visible focus rings.
- Semantic structure:
    - Use proper headings and ARIA roles for tables and dialogs.
- Screen reader hints:
    - Badge/pill labels should be meaningful when read without color.

8. Handoff notes for engineering

- Use component-first approach:
  - Extract: AppShell, JobsTable, FactAuditForm, DiffView, HistoryTable, ProfileCard, Banner, Toast.
- Tailwind:
  - Define utility classes or small component wrappers for:
    - Pills: base + variant classes.
    - Buttons: primary/secondary/tertiary.
    - Diff highlight spans (added/removed).
- Use consistent spacing:
  - 8px grid (8, 12, 16, 24, 32).
- Provide Figma frames:
  - For each screen with:
    - Default state.
    - Loading state.
    - Empty state.
    - Error example.
- Document primary flows in Figma prototype:
  - Onboarding → Jobs → Generate → Diff → History.