# **Concise Project Report — Autonomous Job Application Assistant**

## **1) Scope & Goals**

**Goal:** Continuously discover relevant job openings, curate them in one place, let the user approve applications, and auto-tailor resume/cover letters per job — with full transparency and an approval step for each field the agent fills.

**Key outcomes**

* Fresh, deduped job feed from LinkedIn, Indeed, Y Combinator, and major boards.
* Human-in-the-loop approvals at every critical step (listing selection, form fields, resume edits).
* Versioned artifacts (resume + cover letter) stored with job metadata and timestamps.
* Functional, auditable UI first; agentic automation second.

⚠️ **Compliance note:** Automated form submission may violate some sites’ Terms of Service and trigger anti-bot defenses. Design for **user-initiated** submissions with automation “assist,” and use official APIs/RSS where available.

## **2) System Architecture (high level)**

* **Frontend (Web UI):** Next.js/React, Tailwind, shadcn/ui; WebSocket streams for live agent progress.
* **Orchestrator/API:** FastAPI (Python) or Node (Express) to expose endpoints, manage sessions, and broker approvals.
* **RAG & LLM:**
  + **Local** (user preference): Ollama (e.g., Llama 3.1 8B) for generation; bge-small/e5-small for embeddings.
  + **Vector store:** SQLite + FAISS (lightweight, M1-friendly).
* **Browser automation:** Playwright (headed mode) with a “controlled browser” and DOM diff overlay; fall back to user-click automation (semi-auto).
* **Scheduler:** APScheduler (interval jobs with user-defined cadence).
* **Storage:**
  + **DB (SQLite/Postgres):** jobs, applications, approvals, artifacts, knowledge docs.
  + **Blob store:** local FS (/data/artifacts/…) for PDFs/DOCs; checksum for dedupe.
* **Logging & Audit:** Structured logs (JSON), per-step traces, field-level provenance.

## **3) Core Modules**

1. **Ingestion & Discovery**
   * Sources: YC Work at a Startup API/feeds, Indeed RSS (where available), company career pages, user-provided searches.
   * Scraper adapters normalize to a common schema; dedupe by URL + title + company + location.
2. **Ranking & Match Scoring**
   * **Score(job, profile)** =  
      0.4·semantic\_sim(resume, JD) + 0.25·keyword\_coverage(JD, resume) + 0.15·recency(job\_posted) + 0.1·eligibility(location, visa) + 0.1·seniority\_fit
   * Semantic via sentence embeddings; coverage via tf-idf/keyword hit; configurable weights.
3. **RAG Knowledge Base**
   * Sources: user’s resumes, bullets, projects, metrics, portfolio links, testimonials.
   * Chunk to ~512–800 tokens; store embeddings + metadata (dates, roles, skills).
   * Retrieval prompts generate **justifications** and **citations** per field.
4. **Resume/Cover Tailoring**
   * Strategy: constrained editing with **diff view**.
   * Preserve ATS compatibility: reverse-chronological, consistent headers, plain bullets with quantified impact.
   * Safety rails: jargon cap, length caps, “no hallucinated employers” rule, citations to RAG chunks.
5. **Form-Fill & Live Approval**
   * For each detected field: show **Question**, **Proposed Answer**, **Provenance (RAG chunks)**, **Risk flag** (low/high).
   * User can **Accept**, **Edit**, or **Reject** per field. Accepted fields auto-fill via Playwright.
6. **Submission & Evidence**
   * Store submitted artifacts + timestamp + JD snapshot + site + status.
   * Keep **proof bundle**: PDF of tailored resume, cover letter, applied answers (JSON), screenshots hash (optional).

## **4) Functional UI (focus)**

**Pages & key components**

* **Jobs Listing:** filters (role, level, location, remote), sort by **Match Score**; badges for source & freshness.
* **Job Detail Drawer:** JD text, **Match breakdown** (semantic, keyword, eligibility), “Tailor Resume” CTA.
* **Tailor Workspace:** split view  
  + Left: JD + extracted requirements (bulleted).
  + Right Tabs:  
    - **Resume Diff** (accept/revert hunks).
    - **Cover Letter Draft** (structured, editable).
    - **Form Fill Queue**: table of fields → [Question | Proposed | Source | Actions].
* **Live Browser Panel:** embedded or external window; step log with Playwright screenshots; “Pause / Resume / Step”.
* **History & Artifacts:** searchable timeline; per application card links to files, JD snapshot, and status.

## **5) Data Model (concise)**

**jobs**(id, title, company, location, remote, url, source, posted\_at, jd\_text, hash)  
 **profiles**(id, user\_id, name, base\_resume\_path, skills[], eligibility{work\_auth, location})  
 **knowledge\_docs**(id, profile\_id, type, content, embedding, source\_meta, updated\_at)  
 **match\_scores**(job\_id, profile\_id, semantic, keywords, eligibility, seniority, total)  
 **applications**(id, job\_id, profile\_id, status{draft, pending, submitted, rejected, interview}, submitted\_at)  
 **artifacts**(id, application\_id, kind{resume, cover, qa\_json, screenshots}, path, checksum, created\_at)  
 **field\_proposals**(id, application\_id, site, question, proposed, provenance\_doc\_ids[], state{proposed, accepted, edited, rejected}, final\_value)

## **6) Agent Flow (E2E)**

1. **Schedule tick** → ingest new jobs → rank & dedupe → create/refresh listing.
2. User opens app → reviews **curated jobs** → selects job(s).
3. **Tailor step** → RAG retrieves relevant bullets → LLM proposes edits → user approves per hunk.
4. **Form-fill step** → agent reads fields → proposes values with citations → user approves/edits → agent fills.
5. **Submit (manual or auto)** → store artifacts + status → show success & log.
6. **Follow-ups** (optional): reminders for assessments/emails; update status via links or mailbox hooks.

## **7) Prompts & Guardrails (essentials)**

* **Retrieval prompt**: “Given JD and profile chunks, return bullet candidates with source IDs; do not invent employers/titles; quantify impact.”
* **Edit prompt**: “Apply only these diffs; preserve structure; cap section length; maintain consistent dates.”
* **Form-fill prompt**: “Answer in short prose; cite source chunk IDs; mark as ‘Unknown’ if not in KB.”
* **Validation**: regex/date checks; “No Hallucination” filter rejects content without provenance.

## **8) Privacy, Security, & Compliance**

* Local-first processing; no third-party calls by default (Ollama).
* Secrets vault for site credentials; never store raw passwords in DB.
* Respect site ToS; prefer user-driven “Assist Mode” over fully autonomous submit.
* Optional **CAPTCHA** handling: always human-in-the-loop.

## **9) Performance & Footprint (M1 Air, 16 GB)**

* Models: Llama 3.1 8B (generation), **bge-small** or **e5-small** (embeddings).
* Batch ingestion with rate limiting; incremental vector updates.
* Pre-compute JD embeddings on ingest; cache match scores.
* PDFs via pypdf/docx conversion kept minimal; heavy OCR avoided.

## **10) Quality & Evaluation**

* **Offline eval set**: 50 JDs × 3 base resumes; measure:  
  + Match score correlation with interviewer judgments.
  + Edit precision (accepted hunk %) & provenance coverage %.
  + Submission success rate (no validation errors).
* **A/B**: different weighting schemas; diff acceptance friction.

## **11) Milestones (indicative)**

**Week 1** — Ingestion MVP (YC + RSS), schema, listings UI.  
 **Week 2** — RAG KB + match scoring; Tailor Workspace (diff).  
 **Week 3** — Form-Fill Queue with provenance + approvals; headed Playwright.  
 **Week 4** — Artifact store + History; scheduling; logs & alerts.  
 **Week 5** — Hardening: validation, edge cases, retries, ToS “Assist Mode”.  
 **Week 6** — Polish UI, metrics dashboard, export/share.

## **12) Minimal API Surface (sample)**

* GET /jobs?filter=…
* POST /applications/{jobId}/tailor → returns resume diff, cover draft
* POST /applications/{id}/fields/propose → list of {question, proposed, provenance}
* POST /applications/{id}/fields/{fieldId}/accept|edit|reject
* POST /applications/{id}/submit
* GET /applications/{id}/artifacts

## **13) Risks & Mitigations**

* **Site changes/anti-bot:** adapter isolation, schema validation, “assist” mode.
* **Hallucination:** provenance gating, “Unknown” fallback, strict validators.
* **ATS formatting:** templates locked, lints before export.
* **User trust:** mandatory per-field approval; full audit trail.

## **14) What to build first (functional UI slice)**

1. Jobs Listing with **Match Score** and filters.
2. Tailor Workspace with **Resume Diff** and **Form-Fill Queue** (approval toggles).
3. Submission log + artifacts view.