MATCHSTAK — Project Documentation

[AI-Powered Career Intelligence Platform]

1. Overview

Matchstak is an end-to-end AI-driven career intelligence system built to help job seekers—especially international candidates in the U.S.—navigate a difficult and time-consuming job market more efficiently.

Over the last few years, the U.S. tech job market has been particularly challenging. Many international students and early-career engineers are stuck in a vicious loop: they rely on part-time work to survive, which leaves little time to search and apply for suitable roles, which in turn delays full-time employment. A significant portion of the job-search effort is not applying itself, but **finding the right jobs to apply to**.

Matchstak was built to address that exact problem.

By aggregating jobs from multiple sources, ranking them using semantic similarity instead of keywords, and surfacing only the most relevant opportunities, Matchstak aims to save users an **hour or two per day** that would otherwise be spent scrolling through job boards.

Although originally designed with international students in mind—particularly those needing H-1B sponsorship—it is applicable to any U.S.-based tech job seeker who wants higher signal and less noise in their job search.

---

2. Problem Framing

Key Problems Identified

1. **Job discovery is inefficient**
   Existing platforms like LinkedIn and Indeed prioritize volume over relevance. Users must manually sift through hundreds of postings to find a handful that match their profile.
2. **Sponsorship information is opaque**
   Job boards do not clearly indicate whether companies sponsor H-1B visas. This forces international candidates to waste time applying to roles they are not eligible for.
3. **Keyword matching is insufficient**
   Most platforms rely on keyword overlap, which fails to capture semantic alignment between a résumé and a job description.

4. **Application effort is misallocated**
   Applying to poorly matched jobs yields diminishing returns. What users actually need is help identifying *the best possible matches*, not more matches.

---

3. Design Principles

Matchstak was designed around a few guiding principles:

- **Explainability over black-box automation**
  Users should understand *why* a job is recommended, not just see a score.
- **Cost-aware AI usage**
  The system should minimize unnecessary LLM and embedding calls through caching and scope control.
- **Simplicity before polish**
  The priority was correctness and system behavior, not UI aesthetics.
- **End-to-end ownership**
  The platform was intentionally designed so that a single engineer could understand, debug, and evolve the entire pipeline—from ingestion to matching to delivery.

---

4. Architecture Overview

High-Level Architecture

```
Client (Next.js)
   |
   v
FastAPI Backend
   |
   +-- PostgreSQL (Jobs, Resumes, Matches, Caches)
   |
   +-- Embedding Engine (OpenAI)
   |
   +-- ATS Ingestion Layer
   |      - Greenhouse
   |      - Lever
   |      - Remotive
   |      - JSearch
   |
   +-- Scheduler (APScheduler)
          |
          +-- Daily Job Fetch
          +-- Match Generation
          +-- Email Delivery
```

Architectual Intent

- **ATS ingestion is isolated** to handle inconsistent schemas and rate limits independently of matching logic.
- **Matching results are persisted** to avoid recomputation and prevent duplicate recommendations.
- **Embeddings** are cached using **fingerprinted text hashes** to avoid recomputation for unchanged inputs.
- **Scheduled batch processing** is used instead of real-time matching to keep the system predictable and cost-efficient.

---

5. Key Features

5.1 Résumé Intelligence

Users can upload multiple résumé versions and activate one at a time depending on the role they are targeting.

Each résumé undergoes:

- PDF ingestion and parsing
- LLM-based extraction of:
  - Skills
  - Experience
  - Summary
  - Domains / industries
  - Strengths and weaknesses are inferred heuristically by the LLM and treated as advisory signals rather than hard classifications
  - Skill gaps
  - Seniority estimation
- Embedding generation for semantic matching

This design supports real-world behavior where candidates tailor résumés for different roles.

5.2 Job Aggregation Layer

Matchstak integrates four job sources:

- Greenhouse API
- Lever API
- Remotive API
- JSearch API

All jobs are normalized into a common schema to the extent allowed by source data and stored with:

- Company name
- Title
- Location
- Description
- Category
- Job type
- URL
- Embedding vector

Jobs are deduplicated and automatically cycled out after a fixed retention window to keep the dataset manageable.

5.3 Job Matching Engine

Matchstak uses **embeddings + cosine similarity** to measure semantic alignment between résumés and job descriptions.

This approach was chosen after evaluating alternatives because

- It captures semantic meaning beyond keywords
- It performs well without labeled training data
- It scales cleanly as new job sources are added
- It integrates naturally with LLM-generated explanations

Matching results are:

- Scored
- Ranked
- Stored to prevent duplicates
- Filtered by relevance and eligibility signals

5.4 Insights Engine

For each match, the system generates:

- Why the job is a good fit
- Relevant skills
- Missing skills
- Estimated difficulty
- Seniority classification
- Salary band inference (heuristic, text-based)
- Inferred company tech stack
- Similar job recommendations

Insights are generated only for the **top-N matches** to maintain quality and control cost.

5.5 Application Assistance (Not Auto-Apply)

Matchstak intentionally avoids automatic job submission due to ToS and reliability concerns.

Instead, it generates:

- Tailored cover letters
- “Why this role?” explanations
- Strength summaries
- Interview preparation bullet points

This keeps the system compliant while still providing high value.

5.6 Daily Match Emails

A scheduled job sends users:

- Top-N matches (tier-based - designed for future subscription tiers)
- Match scores
- Insights
- Résumé suggestions
- One-click access to the dashboard

This turns job search into a **push-based workflow** instead of constant manual browsing.

5.7 Dashboard

The dashboard provides:

- Résumé upload and management
- Job match list
- ATS job explorer
- Job detail views

- Insight panels
- Cover letter generation
- Applied / hidden / saved tracking

The UI is intentionally minimal to prioritize system behavior over visual polish.

---

6. ML / LLM Usage Strategy

LLMs are used where they add leverage:

- Parsing unstructured résumés
- Generating explanations and insights
- Producing tailored application material

They are *not* used for:

- Core ranking logic
- Continuous real-time matching
- Autonomous actions (auto-apply)

This hybrid approach balances reliability, explainability, and cost.

---

7. Tradeoffs & Constraints

Key tradeoffs accepted during development:

- **Scope over polish**
  Focused on building a complete, working system rather than a refined UI.
- **Semantic similarity over supervised ML**
  Avoided training models due to lack of labeled data and evaluation complexity.
- **Batch processing over real-time**
  Reduced system complexity and cost.
- **Limited job retention window**
  Prevented uncontrolled data growth.

---

8. Deployment

Matchstak is designed as a production-ready system with a clear deployment path, even though development and testing were primarily done in a local and development environment.

- Backend: FastAPI application structured for deployment on platforms like Render or Railway
- Frontend: Next.js application compatible with Vercel-style deployments
- Database: PostgreSQL supported for production, SQLite used during development
- Scheduler: APScheduler integrated as a background worker for job ingestion and email delivery

The architecture intentionally avoids vendor lock-in and keeps deployment flexible, allowing the system to move from local development to a hosted environment with minimal changes

---

9. Performance Considerations

- Embedding caching via fingerprinted text hashes
- Job and résumé deduplication
- Batched similarity computation
- Controlled LLM invocation
- Retention-based job cycling

---

10. Limitations

- UI is functional but not enterprise-grade
- Embedding similarity, not full RAG
- No active learning loop yet
- No auto-apply functionality

---

11. Future Enhancements

- Auto-apply exploration
- Recruiter-facing portal
- Preference learning from user actions
- Skill taxonomy graph

- Resume rewriting feedback loop
- Vector DB integration
- RAG-based company intelligence

Closing Note

Matchstak was built end-to-end by a single developer with the goal of solving a real, personally experienced problem. It represents an exploration of how AI, LLMs, and modern backend systems can be combined into a practical, user-facing product under real-world constraints.
