Seedgent

```
seedgent.config.js

export default {
role: 'full-stack',
agents: ['scout', 'matcher', 'analyzer'],
stage: ['seed', 'pre-series-a'],
stack: ['TypeScript', 'Next.js', 'Supabase'
remote: 'preferred',
frequency: 'weekly'
};
```

Seedgent

Building an AI Career Radar

Seedgent - Career radar for seed-stage developers

Seedgent

Contents

Executive Summary

Like most developers, I was tired of manually sifting through job boards looking for quality seed-stage opportunities. Most "startup" jobs are actually Series B+ companies with limited equity upside, and existing tools don't understand your unique background or filter for company stage effectively.

So I built Seedgent - an AI-powered career radar that continuously monitors the startup job market and delivers weekly email digests with 3 highly-relevant seed-stage roles. It's designed to work passively in the background, letting busy developers focus on shipping code while never missing their next career opportunity.

This project served two purposes: learning modern AI/ML development and solving my own job discovery problem. The result is a production-ready platform that demonstrates both product thinking and technical execution.

Part 1

Product & Business Understanding

Understanding the Market Problem

The target user is a full-stack developer who wants equity opportunities at early-stage startups but lacks time for traditional job hunting. Current solutions fail because they require manual effort and don't understand the nuances of seed-stage opportunities.

The core problem isn't cost - it's matching quality. LinkedIn Premium still shows Series C companies as "startups". WellFound has good startup focus but requires active browsing. Most alerts are poorly matched because they don't understand your unique background and career angle.

My insight: The best opportunities emerge when you're not actively looking. Stealth startups raise seed rounds, find product-market fit, and need to scale quickly - none of this aligns with personal job search timelines.

Product Strategy & User Experience

I designed Seedgent around the "silent radar" concept - a system that works continuously in the background and only surfaces opportunities worth your attention.

Key design decisions:

- **Email-first delivery** integrates seamlessly into existing workflows
- **3-match weekly limit** ensures quality over quantity (better to send nothing than waste time)
- **One-click feedback** (Save/Dismiss) improves matching over time without cognitive overhead
- **Magic link interactions** eliminate authentication friction when making quick decisions

The UX prioritizes the "30-second decision" - can you determine if a role is worth exploring in under 30 seconds? This drove information hierarchy: match scores upfront, technical stacks visible, equity potential clear.

Business Model & Economics

I chose a freemium model with lifetime pricing for several strategic reasons:

Why freemium works: Developers need to trust the matching quality before paying. A free tier with meaningful value builds trust over months while they experience the product during non-hunting periods.

Why lifetime pricing: Job search is episodic. Developers are highly active for 3-6 months, then largely dormant for 2-3 years. Lifetime pricing ($89) aligns with this usage pattern better than monthly subscriptions.

Economics: Currently built for personal use, but designed with productization in mind. Fixed infrastructure costs would favor scale, with cost controls like automatic user deactivation. The pricing model reflects how I think job search actually works - episodic rather than continuous.

Part 2

Technical Implementation

AI-Assisted Development Approach

This project was largely built using AI coding tools, but with 15 years of software engineering judgment guiding every decision. I didn't blindly accept suggestions - I reviewed, refined, and ensured quality. This approach let me move 5-10x faster while actually increasing code quality.

Modern AI engineering isn't about replacing developer judgment - it's about amplifying it. This project demonstrates how experienced developers can leverage AI tools to accelerate implementation while maintaining high standards.

Architecture & Technology Choices

I chose a hybrid architecture optimized for rapid development and cost efficiency while maintaining production-grade reliability.

Core Stack:

- **FastAPI** handles authentication, APIs, and user sessions
- **Modal** manages heavy AI/ML processing and job scraping with serverless compute
- **Next.js** provides minimal UI for onboarding and saved roles
- **Supabase** offers PostgreSQL with pgvector for semantic search and auth

Why this works:
Python ecosystem for AI, serverless compute without infrastructure overhead, modern frontend patterns. Fast to build, predictable costs.

AI/ML Implementation:

The core technical challenge was building reliable AI systems that work in production, not just demos.

Multi-provider LLM strategy: I built a factory pattern supporting OpenAI, Gemini, and other providers. This prevents vendor lock-in and enables cost optimization - I can use expensive models for critical tasks and cheaper ones for bulk processing.

BaseAgent pattern: All AI agents inherit from a 400+ line base class that handles retry logic, structured validation, error hierarchies, and comprehensive logging. This isn't quick prototype code - it's enterprise-grade reliability patterns applied to LLM workflows.

Semantic matching: User skills and job requirements are converted to 1536-dimensional embeddings. The matching algorithm combines vector similarity with preference weighting and company stage filtering. PostgreSQL's pgvector handles the heavy lifting at scale.

Production reliability: The system includes exponential backoff retry logic, circuit breaker patterns, input validation, and graceful degradation. LLM outputs are validated using Pydantic schemas with fallback strategies when AI responses don't match expected formats.

Engineering Excellence & Scalability

Serverless orchestration: I use Modal for scheduling (5 cron jobs) combined with Prefect for local workflow management. This gives me enterprise workflow benefits without paying for cloud orchestration services - a cost-conscious architecture decision that scales.

Database resilience: The repository pattern includes connection health checks, automatic client refresh, and intelligent retry logic. The system can handle database connection issues gracefully without user-facing errors.

Cost monitoring: Built-in token usage tracking and cost optimization patterns. I monitor AI operation costs in real-time and can adjust provider selection based on performance vs cost trade-offs.

Code organization: 60+ structured modules with clean separation of concerns. Repository pattern for data access, service pattern for business logic, agent pattern for AI workflows. This architecture supports team growth from solo developer to 10+ engineers.

Key insight: I focused on building systems that are reliable and maintainable, not just functional. The AI-assisted approach enabled rapid implementation of proven patterns while maintaining engineering excellence.

Building Seedgent taught me that the best products come from scratching your own itch.

I needed better job discovery, so I built it. I wanted to learn modern AI development, so I used it as my learning vehicle.

The result is a tool I actually use daily and an approach to AI-assisted development that makes me significantly more effective. It turns out building something you actually need is the best way to learn new technologies while creating something genuinely useful.

Code & Demo:

GitHub Repository (private - ask for invite)

Live Demo

Seedgent

Career radar for seed-stage developers