Certify Health Intel: Next-Generation Competitive Intelligence Platform

Executive Summary

The current VBA/Excel solution provides a solid foundation but has significant limitations in **intelligence**, **scalability**, and **actionable insight generation**. This proposal outlines a modern, AI-powered platform that transforms raw competitor data into **strategic intelligence for executive decision-making**.

Current Solution: Gap Analysis

Capability	Current State	Gap
Competitor Discovery	Bing search with static query	No semantic understanding of "who is a competitor"
Evidence Extraction	Regex-based price extraction	Misses context, product features, positioning
Intelligence Layer	None	No reasoning about competitive dynamics
Change Detection	Hash-based diff	No semantic diff (can't explain *what* changed)
Insights	Raw data tables	No synthesis into executive-ready narratives
Scalability	Single-user Excel	Cannot scale to multiple analysts or scheduled runs
Data Sources	Company websites only	Missing news, SEC filings, job postings, reviews

> [!CAUTION]
> The current regex-based approach (`\$([0-9][0-9,]*)\s*(?:per\s*year|/year|annual)`) will miss most real-world pricing patterns, feature claims, and competitive positioning language.

Recommended Solution: AI-Powered Intel Platform

Deliverable Format

> [!IMPORTANT]
> **Primary Deliverable: Web Application + Automated Reports**
>
> Instead of an Excel workbook, deliver a **modern web dashboard** with:
>
> - Scheduled automated intelligence gathering
> - AI-generated executive briefings
> - Real-time competitor monitoring alerts
> - Mobile-accessible insights

Why Move Beyond Excel?

Excel-Based	Web Platform
Manual "Run Pipeline" button	Automated scheduled runs (hourly/daily)
Single user at a time	Multi-user with role-based access
Local API keys	Secure server-side credential management
Static tables	Interactive visualizations, drill-down
No mobile access	Responsive design, executive mobile app
Limited history	Full temporal analysis, trend detection

Platform Architecture

```
flowchart TB
    subgraph Data Collection Layer
        D1[Web Scraper Engine]
        D2[News API Integration]
        D3[SEC/Public Filings Crawler]
        D4[Job Postings Monitor]
        D5[Review Sites Aggregator]
        D6[Social Media Listener]
    end

    subgraph AI Intelligence Layer
        AI1[Competitor Identification Agent]
        AI2[Claim Extraction LLM]
        AI3[Semantic Change Detector]
        AI4[Strategic Insight Generator]
        AI5[Executive Briefing Synthesizer]
    end

    subgraph Data Storage
        DB1[(PostgreSQL - Structured Data)]
        DB2[(Vector DB - Embeddings)]
        DB3[(Document Store - Raw Evidence)]
    end

    subgraph Delivery Layer
        W1[Executive Dashboard]
        W2[Automated Email Briefings]
        W3[Slack/Teams Alerts]
        W4[API for Internal Tools]
    end

    D1 & D2 & D3 & D4 & D5 & D6 --> DB3
    DB3 --> AI1 & AI2
    AI1 --> DB1
    AI2 --> DB1 & DB2
    DB1 & DB2 --> AI3 & AI4
    AI3 & AI4 --> AI5
    AI5 --> W1 & W2 & W3 & W4
```

Component 1: Intelligent Competitor Discovery

How It Knows Who the Right Competitors Are

Current approach uses a static Bing query. This misses:

- Companies that don't rank for generic terms
- Adjacent market entrants
- Private/emerging competitors

Proposed Multi-Signal Approach:

Signal	Source	Intelligence Method
Direct Search	Bing/Google	Expand query taxonomy: "patient intake software", "healthcare revenue cycle", "medical credentialing", etc.
Semantic Similarity	Company websites	Embed Certify Health's own positioning → find companies with similar embeddings
Customer Overlap	G2, Capterra reviews	Find products reviewed by same customers
Job Title Overlap	LinkedIn Jobs API	Companies hiring for same roles = same market
Investment/M&A	Crunchbase, PitchBook	Portfolio overlap with Certify Health's investors
Analyst Reports	Gartner, KLAS	Healthcare IT market maps
LLM Reasoning	GPT-4 / Claude	Given Certify Health's product description, identify competitive categories

Implementation:

```python
class CompetitorDiscoveryAgent:
    """
    Multi-signal competitor identification using LLM reasoning.
    """

    def identify_competitors(self, company_profile: dict) -> list[Competitor]:
        # Step 1: Generate search taxonomy
        taxonomy = self.llm.generate(
            prompt=f"""
            Given this company profile:
            {company_profile}

            Generate 20 search queries that would find competitors, including:
            - Direct product competitors
            - Adjacent market entrants
            - Enterprise vs. SMB alternatives
            - Regional competitors
            - Emerging startups
            """
        )

        # Step 2: Multi-source search
        candidates = []
        for query in taxonomy:
            candidates += self.search_bing(query)
            candidates += self.search_g2_category(query)
            candidates += self.search_crunchbase(query)

        # Step 3: LLM validation & ranking
        validated = self.llm.evaluate(
            prompt=f"""
            For each candidate company, rate competitive relevance 1-10:
            - 10 = Direct head-to-head competitor
            - 7-9 = Overlapping market segment
            - 4-6 = Adjacent/potential competitor
            - 1-3 = Not a real competitor

            Candidates: {candidates}
            Our company: {company_profile}
            """
        )

        return [c for c in validated if c.score >= 6]
```

Component 2: Comprehensive Data Collection

What Data Points to Search, Scrape, and Extract

Category	Data Points	Source	Extraction Method
Pricing	List prices, pricing model (per-user, per-facility, flat), tiers, discounts	Pricing pages, G2, vendor quotes	LLM extraction with structured output
Product Features	Feature lists, capabilities, integrations, certifications	Product pages, changelogs, docs	LLM summarization + embedding for comparison
Market Positioning	Target segments, value propositions, differentiators	Homepage, About page, press releases	LLM analysis of positioning language
Customer Evidence	Customer logos, case studies, testimonials, reference counts	Website, press releases	Image recognition + LLM extraction
Company Health	Funding, headcount, revenue estimates, growth rate	Crunchbase, LinkedIn, news	Structured API pulls + LLM synthesis
Product Velocity	Release notes, new feature announcements, roadmap hints	Changelogs, blogs, webinars	LLM temporal analysis
Sentiment	Review ratings, NPS proxies, complaint themes	G2, Capterra, Reddit, Glassdoor	Sentiment analysis + theme extraction
Go-to-Market	Sales motion, partner ecosystem, channel strategy	Careers page, partner page, news	LLM inference from multiple signals

Executive Team	Leadership changes, key hires, departures	LinkedIn, news, press	Named entity extraction + change tracking
Strategic Signals	M&A activity, new markets, pivots, layoffs	News, SEC filings, job postings	Event extraction + classification

Intelligent Extraction Architecture

```
class IntelligentExtractor:
    """
    LLM-powered extraction with structured output schemas.
    """

    PRICING_SCHEMA = {
        "pricing_model": "enum[per_user, per_facility, flat, usage_based, custom]",
        "base_price": "number | null",
        "price_unit": "string",  # e.g., "/user/month"
        "tiers": [{"name": "string", "price": "number", "features": ["string"]}],
        "enterprise_pricing": "string",  # e.g., "Contact sales"
        "free_tier": "boolean",
        "confidence": "number 0-1",
        "evidence_quote": "string",
        "extraction_reasoning": "string"
    }

    def extract_pricing(self, page_content: str, url: str) -> dict:
        result = self.llm.structured_output(
            prompt=f"""
            Extract pricing information from this webpage content.

            URL: {url}
            Content: {page_content}

            If pricing is not clearly stated, set confidence < 0.5 and explain in reasoning.
            Always include the exact quote that supports your extraction.
            """,
            schema=self.PRICING_SCHEMA
        )

        # Add evidence chain
        result['source_url'] = url
        result['extracted_at'] = datetime.utcnow().isoformat()
        result['content_hash'] = self.hash(page_content)

        return result
```

Component 3: Semantic Change Detection

How It Works (Beyond Hash Comparison)

Current system only knows *that* something changed, not *what* or *why*.

Proposed Approach:

```
flowchart LR
    A[Previous Snapshot] --> B[LLM Diff Analysis]
    C[Current Snapshot] --> B
    B --> D{Change Classification}
    D --> E[Pricing Change]
    D --> F[Feature Addition/Removal]
    D --> G[Positioning Shift]
    D --> H[Leadership Change]
    D --> I[Strategic Signal]

    E & F & G & H & I --> J[Impact Assessment]
    J --> K[Executive Alert Generation]
```

Change Classification Examples:

Change Type	Detection Method	Alert Severity
Price Increase >10%	Numeric diff on extracted pricing	▩ High
New Feature Launch	Feature list diff + changelog analysis	▨ Medium
New Integration	Partner page diff	▨ Medium
Executive Departure	Leadership page diff + news cross-ref	▩ High
Positioning Pivot	Embedding cosine distance on homepage	▨ Medium
Negative Press	News sentiment spike	▩ High
Funding Round	Crunchbase monitor + news	▧ Informational

Component 4: Actionable Insights Generation

From Data → Executive Intelligence

> [!IMPORTANT]
> The key differentiator of this platform is transforming raw scraped data into **narratives that drive decisions**.

Insight Types for Executive Leadership:

Insight Category	Example Output	Business Value
Competitive Threat Assessment	"Competitor X reduced enterprise pricing 20% and launched 3 Certify-competitive features in Q4. Risk: High."	Inform defensive strategy
Market Movement Alert	"Two competitors announced Epic EHR integrations this month. Certify lacks this."	Prioritize roadmap
Win/Loss Intelligence	"Competitor Y appears in 4 new customer case studies where we also competed."	Sales enablement
Positioning Opportunity	"No competitor prominently markets HIPAA BAA compliance speed. Differentiator opportunity."	Marketing messaging
Acquisition Radar	"Competitor Z showing distress signals: layoffs, leadership churn, negative reviews trending."	M&A opportunity
Pricing Intelligence	"Industry median pricing is $15/user/month. Certify at $X is [above/below] market."	Pricing strategy

Executive Briefing Generation:

```python
class ExecutiveBriefingGenerator:
    """
    Synthesizes raw intelligence into executive-ready narratives.
    """

    def generate_weekly_briefing(self, time_period: str) -> str:
        # Gather all changes and new data from period
        changes = self.db.get_changes(since=time_period)
        new_competitors = self.db.get_new_competitors(since=time_period)
        alerts = self.db.get_triggered_alerts(since=time_period)

        briefing = self.llm.generate(
            prompt=f"""
            You are a competitive intelligence analyst preparing a weekly briefing
            for Certify Health's executive team.

            CONTEXT:
            - Certify Health provides patient intake, insurance verification, and
              revenue cycle solutions for healthcare providers
            - Key segments: ambulatory, dental, specialty practices

            THIS WEEK'S INTELLIGENCE:

            New Competitors Identified: {new_competitors}
            Significant Changes Detected: {changes}
            Alerts Triggered: {alerts}

            Generate an executive briefing with:

            1. EXECUTIVE SUMMARY (3-4 sentences, most important developments)

            2. COMPETITIVE THREATS (ranked by urgency)
               - What happened
               - Why it matters to Certify
               - Recommended action

            3. OPPORTUNITIES IDENTIFIED
               - Market gaps or competitor weaknesses
               - How Certify could capitalize

            4. WATCH LIST
               - Emerging signals that aren't yet threats
               - What would elevate them to threats

            5. METRICS DASHBOARD SUMMARY
               - Competitor count by threat level
               - Price positioning vs. market
               - Feature parity score

            Write in crisp, direct executive language. No fluff.
            Lead with insights, not data.
            """
        )

        return briefing
```

Component 5: Technical Implementation

Technology Stack

Layer	Technology	Rationale
Backend	Python (FastAPI)	Rich AI/ML ecosystem, async support
Frontend	Next.js + React	Modern, responsive, SSR for SEO
Database	PostgreSQL + pgvector	Structured data + vector similarity
Document Store	S3 + metadata in Postgres	Scalable evidence storage
Task Queue	Celery + Redis	Scheduled scraping, async processing

AI/LLM	OpenAI GPT-4 / Anthropic Claude	Extraction, reasoning, synthesis
Embeddings	OpenAI text-embedding-3-large	Semantic similarity, clustering
Scraping	Playwright + requests	JS-rendered pages + simple fetches
Deployment	Docker + AWS (ECS/RDS)	Scalable, manageable

Project Structure

```
certify-intel-platform/
├── backend/
│   ├── app/
│   │   ├── api/              # FastAPI endpoints
│   │   ├── agents/           # AI agent implementations
│   │   │   ├── discovery.py      # Competitor discovery
│   │   │   ├── extraction.py     # Data extraction
│   │   │   ├── change_detection.py
│   │   │   ├── insight_generation.py
│   │   │   └── briefing.py       # Executive briefings
│   │   ├── scrapers/         # Data collection
│   │   │   ├── web.py           # Website scraping
│   │   │   ├── news.py          # News API integration
│   │   │   ├── reviews.py       # G2, Capterra
│   │   │   ├── jobs.py          # LinkedIn jobs
│   │   │   └── filings.py       # SEC, business filings
│   │   ├── models/           # SQLAlchemy models
│   │   ├── schemas/          # Pydantic schemas
│   │   ├── services/         # Business logic
│   │   └── tasks/            # Celery async tasks
│   ├── tests/
│   └── alembic/              # DB migrations
├── frontend/
│   ├── app/                  # Next.js app router
│   │   ├── dashboard/
│   │   ├── competitors/
│   │   ├── insights/
│   │   ├── alerts/
│   │   └── settings/
│   └── components/
├── docker/
├── scripts/
└── docs/
```

Database Schema (Core Tables)

```
-- Competitors
CREATE TABLE competitors (
    id UUID PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    domain VARCHAR(255) UNIQUE,
    company_url TEXT,
    threat_level VARCHAR(20),  -- high, medium, low, watch
    discovery_method VARCHAR(50),
    discovery_reasoning TEXT,
    validated_by VARCHAR(50),  -- auto, human
    status VARCHAR(20) DEFAULT 'active',
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Evidence (scraped content)
CREATE TABLE evidence (
    id UUID PRIMARY KEY,
    competitor_id UUID REFERENCES competitors(id),
    source_type VARCHAR(50),  -- website, news, review, job_posting
    source_url TEXT,
    content_text TEXT,
    content_hash VARCHAR(64),
    fetched_at TIMESTAMPTZ,
    metadata JSONB
```

```sql
);

-- Extracted Claims (structured data from evidence)
CREATE TABLE claims (
    id UUID PRIMARY KEY,
    competitor_id UUID REFERENCES competitors(id),
    evidence_id UUID REFERENCES evidence(id),
    claim_type VARCHAR(50),  -- pricing, feature, positioning, etc.
    claim_data JSONB,  -- structured extraction result
    confidence FLOAT,
    extraction_reasoning TEXT,
    validated_by VARCHAR(50),
    status VARCHAR(20),  -- active, superseded, review_required
    valid_from TIMESTAMPTZ,
    valid_to TIMESTAMPTZ,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Embeddings for semantic search
CREATE TABLE claim_embeddings (
    claim_id UUID REFERENCES claims(id),
    embedding vector(1536),
    PRIMARY KEY (claim_id)
);

-- Change Events
CREATE TABLE change_events (
    id UUID PRIMARY KEY,
    competitor_id UUID REFERENCES competitors(id),
    change_type VARCHAR(50),
    severity VARCHAR(20),
    previous_claim_id UUID REFERENCES claims(id),
    new_claim_id UUID REFERENCES claims(id),
    change_summary TEXT,
    impact_assessment TEXT,
    detected_at TIMESTAMPTZ DEFAULT NOW()
);

-- Alerts
CREATE TABLE alerts (
    id UUID PRIMARY KEY,
    change_event_id UUID REFERENCES change_events(id),
    alert_type VARCHAR(50),
    priority VARCHAR(20),
    title TEXT,
    body TEXT,
    delivery_channel VARCHAR(50),  -- dashboard, email, slack
    delivered_at TIMESTAMPTZ,
    acknowledged_by VARCHAR(50),
    acknowledged_at TIMESTAMPTZ
);

-- Executive Briefings
CREATE TABLE briefings (
    id UUID PRIMARY KEY,
    period_start DATE,
    period_end DATE,
    briefing_type VARCHAR(50),  -- weekly, monthly, ad_hoc
    content_markdown TEXT,
    generated_at TIMESTAMPTZ,
    delivered_at TIMESTAMPTZ,
    feedback_score INTEGER
);
```

Component 6: User Interface

Executive Dashboard Mockup

The dashboard should provide:

1. **Competitive Landscape Overview**
    - Visual competitor map (quadrant or tier view)
    - Threat level distribution
    - Recent movers (new entrants, exits, level changes)
2. **Alert Feed**
    - Chronological list of significant changes
    - Filterable by competitor, severity, change type
    - One-click drill-down to evidence
3. **Competitor Deep Dive**
    - Per-competitor intelligence dossier
    - Pricing, features, positioning, sentiment
    - Historical trend charts
    - Side-by-side comparison with Certify
4. **Insights Hub**
    - AI-generated strategic insights
    - Weekly/monthly briefing archive
    - Custom insight requests
5. **Configuration**
    - Add/remove tracked competitors
    - Configure alerts and thresholds
    - Manage data sources and API keys

Implementation Roadmap

Phase 1: Foundation (Weeks 1-3)

Task	Deliverable
Set up Next.js + FastAPI project structure	Skeleton codebase
Design and implement core database schema	PostgreSQL + migrations
Implement basic web scraper (Playwright)	Fetch competitor homepages
Integrate OpenAI for LLM extraction	Pricing + feature extraction
Build competitor CRUD API	Add/edit/list competitors
Create basic dashboard UI	Competitor list view

Phase 2: Intelligence Layer (Weeks 4-6)

Task	Deliverable
Implement Competitor Discovery Agent	Auto-discovery from taxonomy
Build multi-source data collection	News, reviews, jobs scrapers
Create extraction pipelines for all claim types	Structured extraction
Implement embedding-based similarity	Competitive clustering
Build change detection engine	Semantic diff + alerting
Create alert delivery system	Email + dashboard notifications

Phase 3: Insights & Polish (Weeks 7-9)

Task	Deliverable
Build Executive Briefing Generator	Weekly auto-reports
Create insight templates (threats, opportunities)	AI-generated insights
Build competitor comparison view	Side-by-side analysis
Implement historical trend analysis	Temporal visualizations
Add user authentication + roles	RBAC
Polish UI/UX	Production-ready frontend

Phase 4: Deployment & Handoff (Weeks 10-12)

Task	Deliverable

Docker containerization	Production images
AWS deployment (ECS, RDS, S3)	Live environment
Scheduled job configuration	Automated daily runs
Documentation & training	User guide, admin guide
Handoff to Certify Health ops	Knowledge transfer
30-day support window	Bug fixes, tuning

Verification Plan

Automated Tests

1. **Unit Tests**
   - Extraction output validation against known pages
   - Schema compliance for all claim types
   - Change detection logic accuracy
   - Command: `pytest backend/tests/unit/`
2. **Integration Tests**
   - End-to-end pipeline: URL → Extraction → Storage → Alert
   - API endpoint response validation
   - Database migration integrity
   - Command: `pytest backend/tests/integration/`
3. **Scraper Reliability Tests**
   - Test against saved HTML fixtures of competitor pages
   - Validate extraction consistency
   - Command: `pytest backend/tests/scrapers/`

Manual Verification

1. **Extraction Quality**
   - Manually review 20 competitor extractions for accuracy
   - Compare AI extraction vs. human reading
   - Acceptance: >90% accuracy on structured fields
2. **Dashboard Usability**
   - Executive stakeholder demo session
   - Collect feedback on insight clarity
   - Iterate on visualizations
3. **Alert Relevance**
   - Review 1 week of alerts for signal-to-noise ratio
   - Tune thresholds based on feedback

User Review Required

> [!IMPORTANT]
> **Decision Points Requiring Your Input:**

1. **Technology Confirmation**
   - Proceed with Python/FastAPI + Next.js stack?
   - Or prefer a different stack (e.g., Node.js, Python-only)?
2. **LLM Provider**
   - OpenAI GPT-4 vs. Anthropic Claude vs. hybrid?
   - Cost tolerance for API usage?
3. **Deployment Target**
   - AWS preferred? Azure? GCP? On-premise?

    - Existing infrastructure to integrate with?

4. **Data Source Priorities**
    - Which sources are highest priority? (Websites, news, reviews, jobs, SEC?)
    - Any sources to explicitly avoid (compliance reasons)?
5. **Competitor Seed List**
    - Do you have an initial list of known competitors?
    - Or should we start with pure auto-discovery?
6. **Executive Stakeholder Access**
    - Who will consume the insights?
    - What's their preferred delivery channel (dashboard, email, Slack)?
7. **Timeline Expectations**
    - Is the 12-week roadmap appropriate?
    - Any hard deadlines or milestones?

Summary: Why This Approach Is Better

Dimension	Current VBA/Excel	Proposed AI Platform
Intelligence	None (regex + hash)	LLM reasoning, semantic understanding
Coverage	Single static query	Multi-source, multi-signal
Scalability	1 user, manual	Multi-user, automated
Actionability	Raw tables	Executive narratives + recommendations
Timeliness	Manual run	Scheduled + real-time alerts
Maintainability	VBA in Excel	Modern codebase, version controlled
Extensibility	Limited	API-first, plugin architecture

This platform transforms Certify Health from passively collecting data to actively surfacing competitive intelligence that drives strategic decisions.