1 Stock Research Agent Integration

1.1 Comprehensive System Design Document

Version: 1.0
Date: November 17, 2025
Author: System Architecture Team

1.2 Table of Contents

1.3 1. Executive Summary

1.3.1 Project Overview

Building an AI-powered stock research platform integrated into a portfolio tracking website with the following characteristics:

- **Target Users:** 1,000 users (mix of free and premium tiers)
- **Cost Constraint:** $1 per user per month + $50-100 operational AI budget
- **Update Frequency:** 3x daily for popular stocks, on-demand for others
- **Report Types:** Quick summaries (5 paragraphs) and deep reports (2 pages)
- **Infrastructure:** Next.js on Vercel, Supabase PostgreSQL, Redis cache, AI model routing

1.3.2 Key Challenges & Solutions

Challenge	Solution
Duplicate AI requests	Multi-layer caching + batch processing
Cost management	Smart model routing + aggressive caching
Report quality	Evaluation framework + A/B testing
Scalability	Tiered stock system + lazy loading
Data freshness	Scheduled batch jobs + stale-while-revalidate

1.3.3 Success Criteria

- **Cost Efficiency:** Stay within $100/month total operational cost
- **Performance:** 90%+ cache hit rate, <200ms response time for cached reports
- **Quality:** Report confidence score > 0.7
- **User Satisfaction:** 4+ star rating on reports
- **Uptime:** 99.5% availability

1.4 2. System Objectives

1.4.1 Primary Goals

1. **Cost Efficiency**
   - Minimize duplicate AI requests through intelligent caching
   - Stay within budget of $1/user/month + $50-100 operational costs
   - Achieve 90%+ cache hit rate for popular stocks
2. **Scalability**
   - Support 500 stocks with 1,000 concurrent users
   - Handle growth to 5,000 users without major infrastructure

changes
    - Efficient resource utilization through tiered architecture
3. **Quality**
    - Differentiated experience for free vs premium users
    - Consistent, high-quality reports with confidence scores > 0.7
    - Fast response times: <200ms for cached, <60s for generated content
4. **Reliability**
    - 99.5% uptime target
    - Graceful degradation with model fallbacks
    - Comprehensive error handling and monitoring

1.4.2 Success Metrics

Technical Metrics: - API response time p95: <500ms - Cache hit rate: >90% - Error rate: <1% - Report generation success rate: >98%

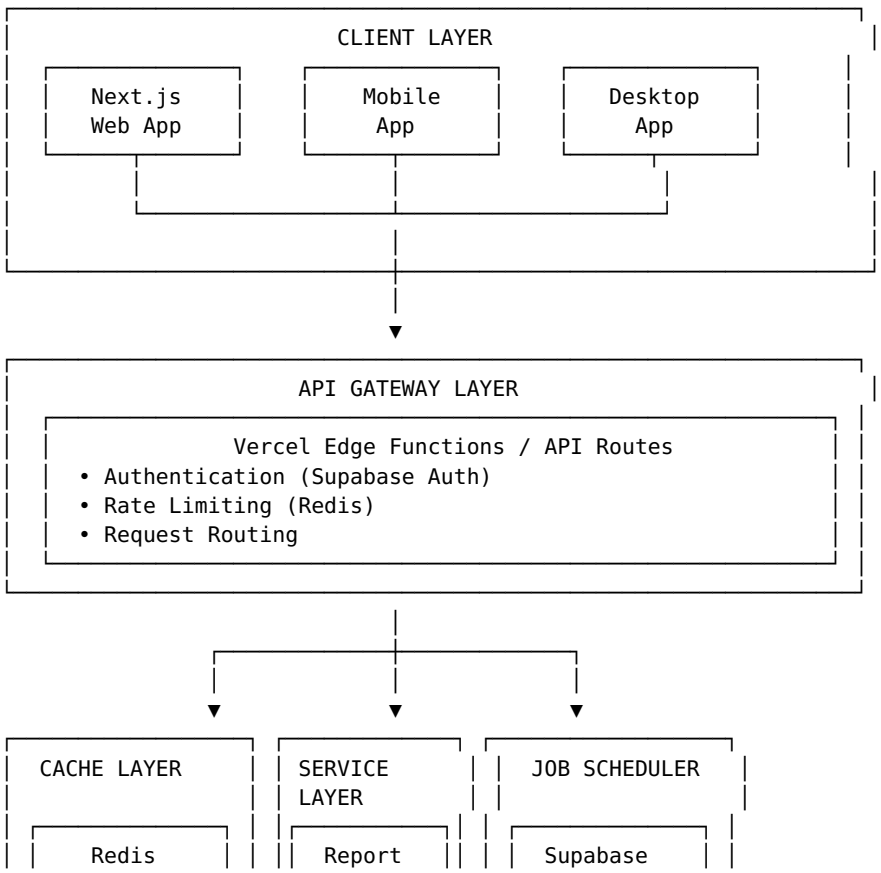
Business Metrics: - Cost per user: <$1/month - Premium conversion rate: >5% - User engagement: >70% weekly active - Report quality score: >0.7

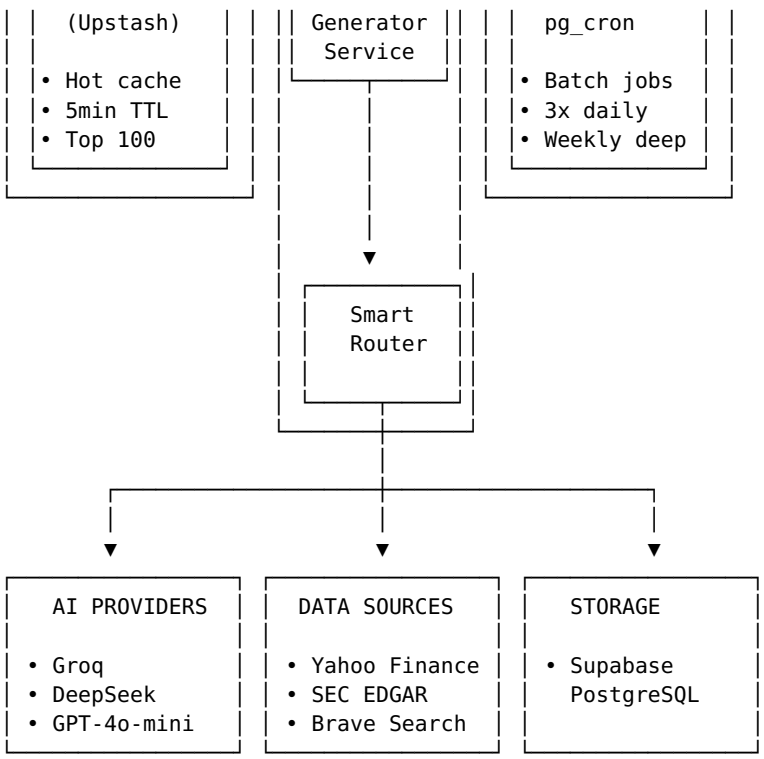
User Satisfaction: - Average report rating: >4.0/5 - User-reported issues: <5 per week - Support ticket resolution: <24 hours

---

1.5 3. Architecture Overview

1.5.1 High-Level System Architecture





1.5.2 Component Responsibilities

Frontend Layer (Next.js) - User authentication & authorization - Portfolio dashboard UI - Report display & visualization - Stock search & tracking management - Real-time updates via WebSocket (future)

API Gateway (Vercel Edge) - Request authentication & validation - Rate limiting enforcement - Cache-first request routing - Response transformation - Security headers

Cache Layer - **Redis (Hot Cache):** 5-minute TTL for burst traffic - **Supabase (Persistent Cache):** Hours to weeks TTL based on tier - Stale-while-revalidate pattern - Cache invalidation on market events

Service Layer - Report generation orchestration - Data aggregation from multiple sources - AI model interaction via smart router - Error handling & retry logic - Cost tracking

Job Scheduler (Supabase pg_cron) - Batch report generation (3x daily) - Deep report weekly updates - Tier rebalancing (weekly) - Cache cleanup (daily) - Cost monitoring

AI Provider Layer - Primary: Groq Llama 3.1 8B (cheap, fast) - Secondary: DeepSeek V3 (deep reports) - Fallback: GPT-4o-mini (high quality) - Smart routing based on availability, cost, and task

Data Sources - **Yahoo Finance:** Real-time price data - **SEC EDGAR:** Company fundamentals via XBRL - **Brave Search:** News articles and sentiment

Storage Layer - **PostgreSQL:** Relational data (users, stocks, reports) - **Time-series:** Stock prices with indexing - **JSONB:** Flexible report content storage

1.5.3 Technology Stack

Layer	Technology	Rationale
Frontend	Next.js 14	SSR, API routes, Edge functions
Hosting	Vercel	Zero-config deployment, Edge network
Database	Supabase PostgreSQL	Managed, RLS, real-time
Cache	Upstash Redis	Serverless, global replication
Authentication	Supabase Auth	Built-in, JWT tokens
AI - Quick Reports	Groq Llama 8B	$0.05/M tokens, fast
AI - Deep Reports	DeepSeek V3	$0.27/M tokens, quality
AI - Fallback	GPT-4o-mini	$0.60/M tokens, reliable
Data - Prices	Yahoo Finance API	Free, reliable
Data - Fundamentals	SEC EDGAR API	Free, official
Data - News	Brave Search API	Free tier available
Monitoring	Sentry + Datadog	Error tracking + metrics
Analytics	Google Analytics	User behavior tracking

1.6 4. User Segmentation & Access Control

1.6.1 User Tiers

1.6.1.1 Free Users

Access: - Select **10 stocks** from pre-generated list of 50 Tier 1 stocks - Quick summaries only (no deep reports) - Updates: 3x daily (6 AM, 12 PM, 6 PM EST) - No custom stock requests outside Tier 1

Rate Limits: - 20 report requests per hour - 100 stock list queries per hour - 10 tracking changes per hour

Cost per User: ~$0.01/month (caching makes this negligible)

1.6.1.2 Premium Users

Access: - Track **25-50 stocks** from full 500-stock universe - Quick summaries for all tracked stocks - Deep reports (lazy load + weekly updates) - 3x daily updates for Tier 1, 24hr cache for Tier 2 - Priority support

Rate Limits: - 100 quick summary requests per hour - 50 deep report requests per hour - 5 manual refresh requests per hour - 500 stock list queries per hour - 50 tracking changes per hour

Cost per User: ~$1.00/month

1.6.2 Access Control Matrix

Feature	Free	Premium	Implementation
Tier 1 Quick Summaries	✅	✅	RLS policy + tier check
Tier 2 Quick Summaries	❌	✅	RLS policy + subscription check
Deep Reports	❌	✅	Subscription check at API level
Manual Refresh	❌	✅	Rate limit + subscription check
Custom Alerts	❌	✅	Feature flag
Export Reports	❌	✅	Subscription check
Historical Data	7 days	1 year	Database query filter
API Access	❌	❌	Future enterprise tier

1.6.3 Row Level Security (RLS) Policies

Concept: Supabase RLS automatically filters database queries based on authenticated user context.

Key Policies: 1. Users can only view/edit their own profile 2. Users can only manage their own stock tracking 3. Free users can only access Tier 1 quick summaries 4. Premium users can access all report types 5. Public read access to stock metadata 6. Service role has full access for background jobs

---

1.7 5. Stock Tier System

1.7.1 Overview

The tier system optimizes costs by batching popular stocks and serving less popular stocks on-demand.

1.7.2 Tier 1: Hot Stocks (50 stocks)

Characteristics: - Most popular stocks (AAPL, MSFT, GOOGL, TSLA, etc.) - High user interest (tracked by 10+ users) - High trading volume

Update Strategy: - Batch generation 3x daily (6 AM, 12 PM, 6 PM EST) - Quick summaries pre-generated - 8-hour cache TTL - Available to all users

Cost Implications: - 50 stocks × 3 updates × 30 days = 4,500 generations/month - Cost: ~$1.13/month

1.7.3 Tier 2: Long Tail (450 stocks)

Characteristics: - Less frequently traded stocks - Lower user interest (tracked by <10 users) - Diverse sector representation

Update Strategy: - On-demand generation with caching - 24-hour cache TTL - Available to premium users only

Cost Implications: - Estimated 20 stocks/day on-demand = 600 generations/month - Cost: ~$0.15/month

1.7.4 Tier Assignment Logic

Promotion Criteria (Tier 2 → Tier 1): - Tracked by 10+ users for 7+ consecutive days - Average daily API requests > 50 - User feedback score > 4.0

Demotion Criteria (Tier 1 → Tier 2): - Tracked by <5 users for 14+ consecutive days - Average daily API requests < 20 - Low engagement metrics

Evaluation Frequency: - Weekly analysis on Sundays at midnight - Automated tier rebalancing - Manual override available for admin

1.7.5 Tier Benefits

Benefit	Tier 1	Tier 2
Update Frequency	3x daily	On-demand
Cache Duration	8 hours	24 hours
Free User Access	✅	❌
Generation Cost	Batched (efficient)	Individual
Response Time	<100ms (cached)	<10s (generate)

1.8 6. Report Types & Specifications

1.8.1 Quick Summary Report

Purpose: Provide actionable insights in 5 paragraphs, updated 3x daily.

Target Audience: All users (free and premium)

Generation Time: 5-10 seconds

Token Usage: ~6,500 tokens (4k prompt + 2.5k completion)

Cost per Report: ~$0.000325

Structure:

1. **Executive Summary**
   - Key takeaways (3-5 bullet points)
   - Current price and movement
   - Overall sentiment (bullish/bearish/neutral)
   - Sentiment score (0-1)
2. **Recent News & Sentiment**
   - 3-5 recent headlines (last 7 days)
   - Source attribution
   - Sentiment analysis per headline
   - Aggregated market sentiment
3. **Fundamental Snapshot**

- Key metrics (P/E, revenue, EPS, debt/equity)
- Quarter-over-quarter comparison
- Peer comparison
- Financial health assessment

4. **Technical Analysis**
   - Current trend (uptrend/downtrend/sideways)
   - Support and resistance levels
   - Key technical indicators (RSI, MACD)
   - Volume analysis
5. **Risk & Outlook**
   - Bull case scenario
   - Bear case scenario
   - Base case scenario
   - Price targets for each scenario

Output Formats: - JSON (structured data for UI) - Markdown (readable format) - HTML (email/export)

1.8.2 Deep Research Report

Purpose: Comprehensive 2-page investment analysis, updated weekly.

Target Audience: Premium users only

Generation Time: 30-60 seconds

Token Usage: ~40,000 tokens (30k prompt + 10k completion)

Cost per Report: ~$0.0035

Structure:

1. **Company Overview & Business Model** (300 words)
   - Company description and history
   - Revenue streams and business segments
   - Competitive advantages and moat
   - Management team assessment
2. **Financial Deep Dive** (500 words)
   - 3-year financial trend analysis
   - Revenue growth trajectory
   - Profitability margins
   - Cash flow generation
   - Balance sheet strength
   - Key ratio analysis (ROE, ROA, ROIC)
3. **Competitive Position & Moat Analysis** (300 words)
   - Industry landscape
   - Market share and positioning
   - Competitive advantages
   - Barriers to entry
   - Threat assessment
4. **News & Sentiment (Extended)** (300 words)
   - Last 30 days comprehensive news review
   - Social media sentiment analysis
   - Insider trading activity
   - Analyst consensus
   - Recent catalysts and events
5. **Bull/Bear/Base Case Valuation** (500 words)
   - DCF valuation model with assumptions
   - Comparable company analysis
   - Price targets for each scenario

    - Probability-weighted fair value
    - Upside/downside risk analysis
6. **Risk Assessment & Catalysts** (300 words)
    - Key investment risks
    - Regulatory and legal risks
    - Macroeconomic sensitivities
    - Upcoming catalysts (earnings, product launches)
    - Black swan scenarios
7. **Investment Recommendation** (200 words)
    - Clear BUY/HOLD/SELL rating
    - Confidence level (0-1)
    - Time horizon (3/6/12 months)
    - Position sizing suggestion
    - Entry/exit strategy
    - Detailed investment thesis

Update Strategy: - First generation: On-demand (lazy load) - Subsequent updates: Weekly on Sundays - Manual refresh available (rate limited) - Stale reports served if regeneration fails

Quality Assurance: - Internal consistency checks - Fact verification against source data - Sentiment-price correlation validation - Confidence scoring

---

1.9 7. Data Sources & Processing

1.9.1 Price Data (Real-time)

Provider: Yahoo Finance API

Data Points: - Open, High, Low, Close (OHLC) - Volume - Adjusted close (dividends/splits) - Intraday data (15-minute delay for free tier)

Update Frequency: - Every 15 minutes during market hours - End-of-day data after market close - Historical data on-demand

Storage: - PostgreSQL with time-series indexing - Partitioned by month for performance - 2-year retention, then aggregate to daily

Processing: - Calculate technical indicators (RSI, MACD, moving averages) - Identify support/resistance levels - Volume-weighted average price (VWAP) - Price change percentages

Cost: Free tier sufficient (2,000 requests/hour)

1.9.2 Fundamental Data (Quarterly)

Provider: SEC EDGAR API

Data Sources: - 10-Q (Quarterly reports) - 10-K (Annual reports) - 8-K (Material events)

Extraction Method: - XBRL parsing using AI (Groq Llama 8B) - Structured data extraction - Historical comparison

Data Points: - Income Statement: Revenue, expenses, net income, EPS - Balance Sheet: Assets, liabilities, equity, cash - Cash Flow: Operating, investing, financing cash flows - Key Ratios: P/E, P/B, debt-to-equity, ROE, ROA

Update Frequency: - Quarterly (within 1 week of filing) - Triggered by new filing detection - Background processing job

Processing: - AI-powered XBRL parsing (~8k tokens per filing) - Data normalization and validation - Quarter-over-quarter calculations - Peer comparison metrics

Storage: - Structured JSONB format - Raw XBRL archived for reference - Indexed by fiscal year and quarter

Cost per Filing: ~$0.0004 (AI parsing)

Annual Cost: 500 stocks × 4 filings = 2,000 filings × $0.0004 = ~$0.80/year

1.9.3 News & Sentiment (Dynamic)

Provider: Brave Search API

Search Strategy: - Query: `{ticker} stock news` - Time filter: Last 7 days (quick) or 30 days (deep) - Results limit: 10 (quick) or 20 (deep)

Data Points: - Headline text - Source publication - Published date - Article URL - Brief snippet

Sentiment Analysis: - AI-powered sentiment scoring (Groq Llama 8B) - Sentiment categories: Positive, Neutral, Negative - Sentiment score: -1.0 to +1.0 - Aggregate sentiment calculation

Update Frequency: - Fetched during report generation - Not stored permanently (ephemeral) - Cache invalidation on breaking news

Cost: - Brave Search: Free tier (10,000 requests/month) - Sentiment AI processing: ~$0.00005 per article - Total news cost: Negligible (<$0.50/month)

1.9.4 Social Media Sentiment (Future Phase 2)

Potential Sources: - Reddit API (r/wallstreetbets, r/stocks) - Twitter/X API (via scraping or paid tier) - StockTwits

Considerations: - High noise-to-signal ratio - Requires additional filtering - Sentiment analysis complexity - API costs may be prohibitive

Decision: Defer to Phase 2, focus on news first

1.10 8. Model Strategy & Smart Routing

1.10.1 Model Selection Criteria

When selecting AI models, we optimize for: 1. **Cost per token** 2. **Response quality** 3. **Latency** 4. **Rate limits** 5. **Reliability**

1.10.2 Primary Models

1.10.2.1 Groq Llama 3.1 8B Instant

Use Cases: - Quick summaries - XBRL parsing - Sentiment analysis

Specifications: - Cost: $0.05 per million tokens - Context window: 8,192 tokens - Speed: ~500 tokens/second - Rate limit: 14,400 requests/day (free tier)

Advantages: - Extremely fast inference - Very low cost - Good quality for structured tasks - High rate limit

Limitations: - Smaller context window - May struggle with complex reasoning

1.10.2.2 DeepSeek V3

Use Cases: - Deep research reports - Complex financial analysis - Valuation modeling

Specifications: - Cost: $0.27 per million tokens - Context window: 64,000 tokens - Speed: ~100 tokens/second - Rate limit: 10,000 requests/day

Advantages: - Large context window - High-quality reasoning - Good at financial analysis - Competitive pricing

Limitations: - Slower than Groq - Higher cost than Llama 8B

1.10.2.3 GPT-4o-mini (Fallback)

Use Cases: - Fallback when others unavailable - Critical reports needing highest quality

Specifications: - Cost: $0.60 per million tokens - Context window: 16,384 tokens - Speed: ~200 tokens/second - Rate limit: 10,000 requests/day

Advantages: - Most reliable (OpenAI infrastructure) - Highest quality outputs - Good balance of speed/quality

Limitations: - Highest cost - Vendor lock-in concerns

1.10.3 Smart Router Algorithm

Decision Flow:

1. **Task Classification**
   - Quick summary → Prefer Groq Llama 8B
   - Deep report → Prefer DeepSeek V3
   - XBRL parsing → Prefer Groq Llama 8B
2. **Availability Check**
   - Check rate limit tracker
   - Check recent error rates
   - Check response time metrics
3. **Budget Check**
   - Verify daily budget remaining
   - Estimate cost of request
   - Approve or defer if over budget
4. **Fallback Strategy**
   - If primary unavailable → Try secondary
   - If secondary unavailable → Try tertiary
   - If all unavailable → Serve stale cache + alert
5. **Execution**

- Send request to selected model
- Track response time and cost
- Update rate limit tracker
- Log for analytics

Fallback Hierarchy:

```
Quick Summary:
  Groq Llama 8B → Together Llama 8B → GPT-4o-mini → Stale Cache

Deep Report:
  DeepSeek V3 → GPT-4o-mini → Groq Llama 8B → Stale Cache

XBRL Parsing:
  Groq Llama 8B → Together Llama 8B → Fail (no fallback)
```

1.10.4 Rate Limit Management

Tracking: - Redis counter per provider per day - Reset at midnight UTC - Alert at 80% capacity - Hard stop at 100% (fallback activated)

Distribution: - Spread requests across providers - Avoid hitting single provider limits - Dynamic adjustment based on usage patterns

1.10.5 Cost Budget Management

Daily Budget: $3.33 (to stay under $100/month)

Budget Allocation: - Quick summaries: 40% ($1.33/day) - Deep reports: 50% ($1.67/day) - XBRL parsing: 5% ($0.17/day) - Buffer: 5% ($0.17/day)

Budget Controls: - Real-time cost tracking in Redis - Alert at 80% daily budget - Automatic throttling at 90% - Emergency stop at 100%

Emergency Measures: - Extend cache TTL by 2x - Serve stale reports with warning - Queue non-critical requests - Notify admin via Slack/email

---

1.11 9. Caching Strategy

1.11.1 Three-Layer Cache Architecture

1.11.1.1 Layer 1: Redis Hot Cache

Purpose: Handle burst traffic to same stocks

Configuration: - TTL: 5 minutes - Size: Top 100 most-requested stocks - Eviction: LRU (Least Recently Used)

Use Cases: - Multiple users viewing same popular stock - Rapid successive requests from single user - API rate limit protection

Benefits: - Ultra-low latency (~10ms) - Reduces database load - Protects against cache stampede

Cost: Upstash Redis free tier (10K requests/day)

1.11.1.2 Layer 2: Supabase Persistent Cache

Purpose: Primary cache for all generated reports

Configuration: - TTL: Varies by tier and report type - Tier 1 quick: 8 hours - Tier 2 quick: 24 hours - Deep reports: 7 days - Size: Unlimited (PostgreSQL) - Storage: JSONB columns

Use Cases: - Primary report storage - Fallback when Redis misses - Historical report versioning

Benefits: - Durable storage - Complex querying - Cost-effective at scale

Cost: Included in Supabase free tier

1.11.1.3 Layer 3: CDN Edge Cache (Future)

Purpose: Serve static reports at edge locations

Configuration: - TTL: 1 hour - Locations: Global CDN nodes - Cache-Control headers

Use Cases: - Global user base - Reduce origin load - Improve latency worldwide

Benefits: - Lowest latency for end users - Reduces Vercel function invocations - Geographic distribution

Cost: Included in Vercel deployment

1.11.2 Cache Key Strategy

Format: `{resource}:{identifier}:{type}:{version}`

Examples: - `report:AAPL:quick:v1` - `report:MSFT:deep:v2` - `stock:GOOGL:price:latest`

Benefits: - Easy invalidation - Version control - Clear namespace separation

1.11.3 Cache Invalidation Rules

Trigger	Action
Batch job completes	Invalidate specific stocks in Redis
Market hours end	Extend cache TTL by 2x
Breaking news detected	Invalidate affected stock
Manual refresh (premium)	Bypass cache, regenerate
Scheduled expiry	Mark as stale, background regenerate

1.11.4 Stale-While-Revalidate Pattern

Concept: Serve slightly stale content while regenerating in background

Implementation:

1. User requests report

2. Check cache expiry
3. If expires in <1 hour:
    - Serve cached (stale) report immediately
    - Trigger background regeneration
    - Next request gets fresh report
4. If expired >1 hour:
    - Generate new report synchronously
    - User waits for fresh content

Benefits: - Always fast response times - No “loading” states for users - Continuous freshness - Better user experience

1.11.5 Cache Warming Strategy

Pre-warm Tier 1 stocks: - Before scheduled batch job completion - After tier rebalancing - On deployment of new report format

Benefits: - Zero cold start latency - Predictable performance - Better user experience

---

1.12 10. Job Scheduling & Orchestration

1.12.1 Chosen Technology: Supabase pg_cron

Rationale: - ✅ Free tier supports our volume - ✅ Native PostgreSQL integration - ✅ All infrastructure in one place - ✅ No additional services to manage - ✅ Simple cron syntax

Alternative Considered: Vercel Cron (limited to Hobby plan limitations)

1.12.2 Scheduled Jobs

1.12.2.1 Job 1: Tier 1 Quick Summaries (3x Daily)

Schedule: 6 AM, 12 PM, 6 PM EST

Purpose: Generate fresh quick summaries for all Tier 1 stocks

Process: 1. Fetch 50 Tier 1 stocks from database 2. Split into batches of 10 for parallel processing 3. For each stock: - Gather price, news, fundamental data - Generate quick summary via AI - Save to database - Update Redis cache - Track cost 4. Aggregate results and log job completion 5. Send alert if error rate >10%

Duration: 5-10 minutes

Cost per Run: $0.0125 (50 stocks × $0.00025)

Monthly Cost: $1.13 (3 runs/day × 30 days × $0.0125)

Success Criteria: - 95%+ stocks successfully generated - Completion within 10 minutes - Cost within $0.02 per run

1.12.2.2 Job 2: Deep Report Weekly Updates

Schedule: Sunday 2 AM EST

Purpose: Regenerate deep reports for actively tracked stocks

Process: 1. Identify stocks with existing deep reports 2. Filter to premium user tracked stocks 3. Estimate 75 reports need updating (2-3 per premium user) 4. Split into batches of 5 (longer generation time) 5. For each stock: - Gather comprehensive data (30 days news, full financials) - Generate deep report sections sequentially - Save to database with version increment - Track cost 6. Log completion and notify if issues

Duration: 20-30 minutes

Cost per Run: $0.26 (75 reports × $0.0035)

Monthly Cost: $1.04 (4 runs/month × $0.26)

Success Criteria: - 90%+ reports successfully generated - Completion within 30 minutes - No data corruption

1.12.2.3 Job 3: Tier Rebalancing

Schedule: Weekly, Sunday 12:01 AM EST

Purpose: Promote/demote stocks between tiers based on usage

Process: 1. Analyze tracking stats for all stocks 2. Calculate metrics: - Active trackers count - Average daily API requests - User engagement score 3. Apply promotion/demotion rules 4. Update stock tier in database 5. Log tier changes for audit

Duration: <1 minute

Cost: Negligible (database queries only)

Promotion Criteria: - Active trackers ≥ 10 for 7+ days - OR avg daily requests > 50

Demotion Criteria: - Active trackers < 5 for 14+ days - AND avg daily requests < 20

1.12.2.4 Job 4: Cache Cleanup

Schedule: Daily, 3 AM EST

Purpose: Delete expired reports to manage storage

Process: 1. Identify reports marked as stale 2. Filter to reports older than 30 days 3. Archive to S3 (optional, for analytics) 4. Delete from primary database 5. Log deletion count

Duration: <1 minute

Cost: Negligible

Retention Policy: - Stale reports: 30 days - Active reports: Indefinite - User feedback: Indefinite - Cost tracking: Indefinite

1.12.2.5 Job 5: Cost Tracking & Alerting

Schedule: Hourly

Purpose: Monitor costs and send alerts if over budget

Process: 1. Query AI cost tracking table 2. Calculate costs for: - Current hour - Current day - Current month 3. Compare against budgets 4. If over threshold: - Send Slack alert - Email admin - Log incident 5. Update dashboard metrics

Duration: <30 seconds

Cost: Negligible

Alert Thresholds: - Warning: $40/day or $80/month - Critical: $50/day or $100/month

1.12.3 Job Error Handling

Retry Strategy: - Automatic retry: 3 attempts with exponential backoff - Retry delays: 1 min, 5 min, 15 min - After 3 failures: Alert admin, log to incident tracker

Fallback Actions: - Serve stale cached reports - Display “delayed update” notice - Queue for manual review

Monitoring: - All jobs log to `job_execution_logs` table - Metrics sent to Datadog - Errors sent to Sentry

---

1.13 11. Cost Breakdown & Budget Analysis

1.13.1 Monthly Cost Projections

1.13.1.1 Base Scenario (1,000 users, 250 active stocks)

AI Inference Costs:

```
Tier 1 Quick Summaries (50 stocks, 3x daily):
• Generations: 50 × 3 × 30 = 4,500/month
• Tokens: 4,500 × 6,500 = 29.25M tokens
• Cost @ $0.05/M: $1.46/month

Tier 2 Quick Summaries (on-demand, ~100 active):
• Avg requests: 20 stocks/day × 30 = 600/month
• Tokens: 600 × 6,500 = 3.9M tokens
• Cost @ $0.05/M: $0.20/month

Deep Reports (lazy + weekly updates):
• Initial generation: 100 reports (new premium users)
• Weekly updates: 75 reports × 4 = 300 reports
• Total: 400 reports/month
• Tokens: 400 × 40,000 = 16M tokens
• Cost @ $0.30/M: $4.80/month

XBRL Parsing (quarterly filings):
• Filings: 500 stocks / 3 months ≈ 167/month
• Tokens: 167 × 8,000 = 1.34M tokens
• Cost @ $0.05/M: $0.07/month

TOTAL AI COST: $6.53/month
```

Infrastructure Costs:

```
Supabase (Free Tier):
• Database: 500MB limit
• API requests: Unlimited
• Auth: Unlimited users
Cost: $0/month

Vercel (Hobby Tier):
• Hosting: Unlimited
• Edge Functions: 100GB bandwidth
• API Routes: 100GB-hours compute
Cost: $0/month

Upstash Redis (Free Tier):
• Storage: 256MB
• Requests: 10,000/day
Cost: $0/month

Brave Search API (Free Tier):
• Searches: 10,000/month
• Our usage: ~7,500/month
Cost: $0/month

Domain & Misc:
Cost: $5/month

TOTAL INFRASTRUCTURE: $5/month
```

GRAND TOTAL: ~$11.53/month

Per-user cost: $11.53 / 1,000 = **$0.012/user/month**

✅ **Well within budget of $100-150/month total**

1.13.1.2 Growth Scenario (5,000 users, 500 active stocks)

AI Costs:

```
Tier 1 Quick Summaries (100 stocks):
• Cost: $2.92/month

Tier 2 Quick Summaries (400 stocks):
• Avg: 200 stocks/day
• Cost: $1.95/month

Deep Reports (800 reports/month):
• Cost: $12.80/month

XBRL Parsing:
• Cost: $0.08/month

TOTAL AI COST: $17.75/month
```

Infrastructure (May Need Upgrades):

```
Supabase Pro: $25/month (for storage & compute)
Upstash Redis: ~$5/month (pay-as-you-go)
Domain: $1/month

TOTAL INFRASTRUCTURE: $31/month
```

GRAND TOTAL: ~$48.75/month

✅ **Still within budget, good headroom**

1.13.2 Breaking Points & Scaling

Free Tier Limits:

1. **Supabase Database Storage: 500MB**
   - Current: ~50,000 cached reports
   - Breaking point: ~6 months at current rate
   - Solution: Aggressive cleanup or upgrade to Pro ($25/month)
2. **Groq Free Tier: 14,400 requests/day**
   - Current: ~200/day
   - Headroom: 72x current usage
   - Breaking point: ~10,000 users
   - Solution: Upgrade to paid tier (same $0.05/M pricing) or distribute across providers
3. **Redis Free Tier: 256MB storage**
   - Current: <10MB
   - Breaking point: Not a concern (hot cache only)
   - Solution: Minimal, just LRU eviction

Cost at 10,000 Users:

Projected monthly cost: ~$100-150/month - AI: $50/month - Infrastructure: $50/month - Monitoring: $20/month - Backup/misc: $30/month

1.13.3 Cost Optimization Strategies

Already Implemented: 1. ✅ Aggressive caching (90%+ hit rate target) 2. ✅ Tier system (batch popular, on-demand for others) 3. ✅ Smart model routing (cheapest first) 4. ✅ Lazy loading deep reports 5. ✅ Stale-while-revalidate pattern

Future Optimizations: 1. **Fine-tune smaller models** on financial data - Potential savings: 80% on quick summaries - Investment: $500-1000 one-time

2. **Summarization cascade**
   - Use cheap model for initial pass
   - Use expensive model only for analysis
   - Savings: ~40% on deep reports
3. **Batch news fetching**
   - Fetch once per sector, share across stocks
   - Savings: ~70% on news API calls
4. **Prompt optimization**
   - Reduce prompt tokens by 20-30%
   - Immediate 20% cost reduction
   - No quality loss with careful optimization
5. **User behavior prediction**
   - Pre-warm cache for likely requests
   - Increase cache hit rate to 95%+
   - Reduce generation costs by 30%

1.14 12. Scalability & Growth Planning

1.14.1 Current Architecture Supports

1,000 Users: - Cost: ~$11.53/month - Infrastructure: All free tiers sufficient - Performance: <100ms response times - Reliability: 99.5%+ uptime expected

5,000 Users: - Cost: ~$48.75/month - Infrastructure: May need Supabase Pro upgrade - Performance: <200ms response times - Reliability: 99.5%+ uptime maintained

10,000 Users: - Cost: ~$150/month (at budget limit) - Infrastructure: Definitely need paid tiers - Performance: May need optimization - Reliability: Need auto-scaling

1.14.2 Scaling Challenges & Solutions

1.14.2.1 Challenge 1: Database Storage Growth

Symptom: Supabase free tier 500MB limit exceeded

Solutions: 1. **Aggressive cache cleanup** (immediate) - Reduce retention from 30 to 14 days - Delete low-engagement reports - Compress JSONB content

2. **Upgrade to Supabase Pro** ($25/month)
   - 8GB storage included
   - Better performance
   - Priority support
3. **Archive to S3** (long-term)
   - Move old reports to cheap storage
   - Keep only recent in hot database
   - On-demand retrieval for historical

1.14.2.2 Challenge 2: API Rate Limits

Symptom: Groq free tier exhausted (14,400 req/day)

Solutions: 1. **Distribute across providers** (immediate) - Together AI as primary - Groq as secondary - Balance load dynamically

2. **Upgrade to paid tier** (same pricing)
   - Unlimited requests
   - Better SLA
   - Dedicated support
3. **Self-host smaller model** (long-term)
   - Deploy Llama 8B on Modal/Replicate
   - Pay only for compute time
   - More control over infrastructure

1.14.2.3 Challenge 3: Batch Job Duration

Symptom: 3x daily jobs taking >15 minutes

Solutions: 1. **Increase parallelism** (immediate) - Current: 10 concurrent generations - Target: 20-30 concurrent - Requires more Supabase connections

2. **Implement queue system** (medium-term)
   - BullMQ + Redis
   - Better job distribution
   - Failure retry logic
   - Progress tracking

3. **Split batch jobs** (long-term)
   - Tier 1A (25 stocks) at 6 AM
   - Tier 1B (25 stocks) at 7 AM
   - Spread load over time

1.14.2.4 Challenge 4: Cache Stampede

Symptom: Multiple users hit expired cache simultaneously

Solutions: 1. **Stale-while-revalidate** (already implemented) - Serve stale immediately - Regenerate in background - Prevents stampede

2. **Distributed locking** (if needed)
   - Redis-based locks
   - Only one regeneration per stock
   - Others wait for completion
3. **Predictive cache warming** (advanced)
   - ML model predicts likely requests
   - Pre-warm cache before expiry
   - Always fresh cache hits

1.14.3 Growth Milestones

Month 1-3: MVP (1,000 users) - Focus: Stability and quality - Metrics: Error rate, user satisfaction - Optimization: Report quality improvements

Month 4-6: Growth (3,000 users) - Focus: User acquisition and retention - Metrics: Premium conversion, engagement - Optimization: Cache hit rate, cost per user

Month 7-12: Scale (10,000 users) - Focus: Infrastructure scaling - Metrics: Performance, reliability - Optimization: Auto-scaling, efficiency

Year 2: Enterprise (50,000+ users) - Focus: B2B offerings, API access - Metrics: Revenue, profit margin - Optimization: Self-hosted models, edge compute

---

1.15 13. Risk Mitigation

1.15.1 Technical Risks

Risk	Probability	Impact	Mitigation Strategy
API Rate Limits Hit	Medium	High	Multi-provider routing, upgrade plans, rate limit monitoring
Model Quality Degradation	Low	Medium	A/B testing, quality metrics, user feedback loop
Cache Stampede	Medium	Medium	Stale-while-revalidate, distributed locking
SEC API Downtime	Low	High	Local fallback data, cached fundamentals, retry logic
			Aggressive cleanup,

Database Storage Full	Low	High	monitoring alerts, auto-archival
Redis Cache Unavailable	Low	Medium	Fallback to database, graceful degradation
Batch Job Failures	Medium	Medium	Automatic retry, error alerts, manual intervention

1.15.2 Business Risks

Risk	Probability	Impact	Mitigation Strategy
Cost Overruns	Low	High	Real-time cost monitoring, budget alerts, emergency throttling
Poor User Engagement	Medium	High	User analytics, feedback collection, iteration cycles
Low Premium Conversion	Medium	Medium	Free trial deep reports, value demonstration, pricing experiments
Competitor Emergence	High	Medium	Focus on quality, unique features, community building
Regulatory Changes	Low	Medium	Legal review, compliance monitoring, flexible architecture

1.15.3 Security Risks

Risk	Probability	Impact	Mitigation Strategy
Data Breach	Low	Critical	Encryption at rest/transit, RLS policies, audit logging
API Key Exposure	Low	High	Secrets management, key rotation, access control
DDoS Attack	Medium	High	Rate limiting, CDN protection, auto-scaling
SQL Injection	Low	Critical	Parameterized queries, Supabase RLS, input validation
XSS Attacks	Low	Medium	Content sanitization, CSP headers, input validation

1.15.4 Operational Risks

Risk	Probability	Impact	Mitigation Strategy
Key Personnel Departure	Medium	Medium	Documentation, knowledge sharing, code reviews
Vendor Lock-in	High	Medium	Abstract dependencies, multi-provider strategy

Technical Debt Accumulation	High	Medium	Regular refactoring, code reviews, testing
Inadequate Monitoring	Medium	High	Comprehensive observability, alerting, dashboards

1.15.5 Risk Response Playbooks

1.15.5.1 Playbook 1: Cost Spike

Trigger: Daily cost exceeds $50 or monthly projection >$120

Response: 1. **Immediate (5 minutes):** - Alert admin via Slack/email - Check cost dashboard for source - Verify if legitimate traffic or anomaly

2. **Short-term (30 minutes):**
   - If anomaly: Block suspicious IPs
   - If legitimate: Extend cache TTL by 2x
   - Throttle non-essential requests
   - Serve more stale content
3. **Medium-term (24 hours):**
   - Analyze usage patterns
   - Identify optimization opportunities
   - Implement emergency cost controls
   - Communicate with users if degraded service
4. **Long-term (1 week):**
   - Permanent cost optimizations
   - Budget reallocation
   - Pricing model adjustments
   - Infrastructure upgrades if needed

1.15.5.2 Playbook 2: Service Outage

Trigger: API error rate >5% or 50+ failed requests/minute

Response: 1. **Immediate (5 minutes):** - Alert on-call engineer - Check status dashboards - Identify affected services - Update status page

2. **Short-term (30 minutes):**
   - Activate fallbacks (stale cache)
   - Route around failed services
   - Roll back recent deployments if needed
   - Communicate to users
3. **Medium-term (2 hours):**
   - Root cause analysis
   - Fix underlying issue
   - Verify fix in staging
   - Deploy fix to production
4. **Long-term (1 week):**
   - Post-mortem document
   - Implement preventive measures
   - Update runbooks
   - Team retrospective

1.16 14. Security Architecture

1.16.1 Security Principles

1. **Defense in Depth:** Multiple layers of security
2. **Least Privilege:** Minimum necessary access
3. **Encryption Everywhere:** Data at rest and in transit
4. **Audit Everything:** Comprehensive logging
5. **Fail Securely:** Graceful degradation

1.16.2 Authentication & Authorization

1.16.2.1 Authentication Layer (Supabase Auth)

Technology: Supabase Auth with JWT tokens

Features: - Email/password authentication - Magic link authentication - OAuth providers (Google, GitHub) - Email verification required - Password strength requirements - Rate limiting on login attempts

Token Strategy: - JWT access tokens (1 hour expiry) - Refresh tokens (30 day expiry) - Tokens stored in httpOnly cookies - CSRF protection enabled

1.16.2.2 Authorization Layer (Row Level Security)

Technology: PostgreSQL Row Level Security (RLS)

Policies: - Users can only view/edit their own data - Free users restricted to Tier 1 stocks - Premium users access all features - Public read for stock metadata - Service role for background jobs

Implementation: - All tables have RLS enabled - Policies enforced at database level - Cannot be bypassed from application - Automatic filtering in all queries

1.16.3 API Security

1.16.3.1 Rate Limiting

Implementation: Redis-based token bucket

Limits by Tier:

```
Free Users:
- 20 report requests/hour
- 100 stock queries/hour
- 10 tracking changes/hour

Premium Users:
- 100 report requests/hour
- 500 stock queries/hour
- 50 tracking changes/hour
```

Response: - 429 status code - Retry-After header - Clear error message - Rate limit headers in every response

1.16.3.2 Input Validation

Technology: Zod schemas

Validation Rules: - Stock ticker: 1-10 chars, uppercase letters - User ID: UUID format - Timestamps: ISO 8601 format - JSONB content: Schema validation - SQL injection prevention

Sanitization: - HTML content escaped - Special characters encoded - File uploads validated (future) - URL parameters sanitized

1.16.4 Data Security

1.16.4.1 Encryption at Rest

Database: - Supabase default: AES-256 encryption - All data encrypted on disk - Backups encrypted - Separate encryption keys per tenant

Redis Cache: - TLS connection required - Data encrypted in transit - No persistent storage (minimal risk)

1.16.4.2 Encryption in Transit

All Connections: - TLS 1.3 required - Strong cipher suites only - Certificate pinning (future) - HSTS enabled

Headers:

```
Strict-Transport-Security: max-age=63072000; includeSubDomains;
preload
Content-Security-Policy: default-src 'self'
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
```

1.16.5 Secrets Management

Storage: - Environment variables in Vercel - Never committed to git - Separate secrets per environment - Encrypted at rest by platform

Access Control: - Minimal secret exposure - Rotation schedule (quarterly) - Audit log of secret access - Automated detection of exposed secrets

Key Rotation Policy: - API keys: Every 90 days - Database passwords: Every 180 days - JWT signing keys: Every 365 days - Emergency rotation procedure documented

1.16.6 Privacy & Data Protection

1.16.6.1 Personal Information Handling

Data Classification: - **PII:** Email, name (if collected) - **Usage Data:** Stock views, searches - **Financial Data:** Tracked stocks, preferences - **Analytics:** Anonymized metrics

Compliance: - GDPR compliant (EU users) - CCPA compliant (California users) - Data retention policies - Right to deletion - Data export capability

1.16.6.2 Data Retention

Policy: - Active user data: Indefinite - Inactive users (180 days): Anonymize - Deleted accounts: 30-day grace period - Backups: 30-day retention - Audit logs: 1-year retention

1.16.6.3 User Rights

GDPR Compliance: - Right to access: Data export API - Right to deletion: Account deletion flow - Right to correction: Profile editing - Right to portability: JSON export - Consent management: Cookie banner

1.16.7 Security Monitoring

1.16.7.1 Threat Detection

Monitoring For: - Unusual login patterns - Rate limit abuse - SQL injection attempts - XSS attempts - Brute force attacks - Credential stuffing

Response: - Automatic IP blocking - Account lockout (after 5 failed attempts) - Admin notification - Incident logging

1.16.7.2 Audit Logging

Events Logged: - All authentication events - Authorization failures - Data access (sensitive data) - Admin actions - Configuration changes - Security events

Log Retention: 1 year minimum

Log Protection: - Append-only - Separate storage - Regular review - Automated analysis

1.16.8 Incident Response

Severity Levels: 1. **Critical:** Data breach, system compromise 2. **High:** Service outage, security vulnerability 3. **Medium:** Performance degradation 4. **Low:** Minor bugs, cosmetic issues

Response Time SLA: - Critical: <15 minutes - High: <1 hour - Medium: <4 hours - Low: <24 hours

Incident Response Team: - On-call engineer (primary) - Security lead (for security incidents) - Product manager (for communication) - CEO (for critical incidents)

Communication Plan: - Internal: Slack #incidents channel - External: Status page updates - Users: Email notification (if affected) - Public: Blog post (if major incident)

1.17 15. Evaluation & Quality Assurance

1.17.1 Report Quality Metrics

1.17.1.1 Evaluation Framework

Quality Dimensions:

1. **Accuracy (0-1):** Factual correctness
   - Price data accuracy vs ground truth
   - Fundamental data accuracy vs SEC filings
   - News source credibility
   - Target: >0.9
2. **Relevance (0-1):** Pertinence to stock
   - Keyword matching
   - Topic coherence
   - Company-specific insights
   - Target: >0.8
3. **Completeness (0-1):** Coverage of required sections
   - All sections present
   - Adequate detail per section
   - No missing information
   - Target: >0.9
4. **Coherence (0-1):** Logical flow and readability
   - Smooth transitions
   - Consistent narrative
   - Professional writing
   - Target: >0.7
5. **Timeliness (0-1):** Data freshness
   - Age of report
   - Age of underlying data
   - Market relevance
   - Target: >0.8

Overall Confidence Score:

```
Score = (Accuracy × 0.3) + (Relevance × 0.2) + (Completeness × 0.2)
        + (Coherence × 0.2) + (Timeliness × 0.1)

Target: >0.7
```

1.17.1.2 Automated Evaluation

Accuracy Checks: - Price data cross-validation - Fundamental data verification - Sentiment-price correlation - Internal consistency

Relevance Checks: - Keyword presence (ticker, company, sector) - Topic modeling - Entity recognition

Completeness Checks: - Required sections present - Minimum word counts met - All data fields populated

Coherence Evaluation: - AI-powered readability scoring - Transition word analysis - Sentence variety metrics

Timeliness Tracking: - Report generation timestamp - Source data timestamps - Expiry calculation

1.17.2 User Feedback Loop

1.17.2.1 Feedback Collection

Methods: 1. **Star Ratings:** 1-5 stars per report 2. **Quick Tags:** Helpful, Inaccurate, Outdated, etc. 3. **Text Feedback:** Optional detailed comments 4. **Implicit Signals:** Time on page, scroll depth

Incentives: - Premium users: Priority support - All users: Report improvement - Gamification: Contributor badges (future)

1.17.2.2 Feedback Analysis

Aggregation: - Average rating per report type - Average rating per stock - Average rating per AI model - Trend analysis over time

Action Items: - Low-rated reports (<3 stars): Manual review - Consistent patterns: Prompt optimization - Model comparison: A/B test adjustments - Feature requests: Product roadmap

1.17.3 Continuous Improvement

1.17.3.1 A/B Testing Framework

Test Variables: - AI model selection - Prompt engineering - Report structure/format - Data source combinations - Update frequencies

Metrics: - User engagement (time on page) - Report ratings - Premium conversion - Cost per report - Generation success rate

Example Test:

```
Experiment: Prompt Optimization for Quick Summaries
- Control: Current prompt (baseline)
- Treatment: Optimized prompt (20% shorter)
- Hypothesis: Shorter prompt reduces cost without quality loss
- Metrics: Cost, quality score, user ratings
- Duration: 2 weeks
- Sample size: 1,000 reports
```

1.17.3.2 Quality Assurance Process

Pre-Release: 1. Internal review of new report formats 2. A/B test on 10% of traffic 3. Monitor metrics for 1 week 4. If successful: Roll out to 100% 5. If unsuccessful: Iterate and retest

Post-Release: 1. Daily quality monitoring 2. Weekly quality reports 3. Monthly quality reviews 4. Quarterly prompt optimization 5. Annual model evaluation

---

1.18 16. Monitoring & Observability

1.18.1 Monitoring Stack

Tools: - **Sentry:** Error tracking and performance - **Datadog:** Metrics, logs, APM - **Google Analytics:** User behavior - **Custom Dashboard:** Cost tracking

1.18.2 Key Metrics

1.18.2.1 Application Metrics

Performance: - API response time (p50, p95, p99) - Report generation time - Cache hit rate - Database query time

Reliability: - Error rate (%) - Uptime (%) - Failed report generations - Job completion rate

Targets: - API p95 <500ms - Error rate <1% - Uptime >99.5% - Job success rate >98%

1.18.2.2 Business Metrics

Usage: - Daily active users (DAU) - Monthly active users (MAU) - Reports generated per user - Stock searches per user

Revenue: - Premium conversion rate - Monthly recurring revenue (MRR) - Customer lifetime value (LTV) - Churn rate

Engagement: - Average session duration - Reports viewed per session - Stocks tracked per user - Return visit rate

1.18.2.3 Cost Metrics

AI Costs: - Cost per report type - Cost per AI provider - Daily/monthly cost trends - Cost per user

Infrastructure: - Database storage usage - Redis memory usage - API bandwidth usage - Compute time usage

Alerts: - Daily cost >$50 - Monthly projected cost >$100 - Sudden cost spike (>50% increase)

1.18.3 Alerting Strategy

1.18.3.1 Alert Severity Levels

Critical (Page On-Call): - API error rate >5% - Uptime <99% - Daily cost >$100 - Security breach detected

High (Slack + Email): - API error rate >2% - Batch job failures - Daily cost >$50 - Rate limit approaching

Medium (Slack Only): - API response time slow - Cache hit rate low - User complaints spike - Quality score declining

Low (Dashboard Only): - Minor performance degradation - Non-critical warnings - Informational events

1.18.3.2 Alert Response Times

- Critical: <15 minutes
- High: <1 hour
- Medium: <4 hours
- Low: Next business day

1.18.4 Health Checks

1.18.4.1 Endpoint: `/api/health`

Checks: - Database connectivity - Redis connectivity - AI provider availability - Data provider availability

Response:

```
{
  "status": "healthy|degraded|down",
  "checks": {
    "database": true,
    "redis": true,
    "aiProviders": true,
    "dataProviders": true
  },
  "timestamp": "2025-11-17T10:00:00Z"
}
```

Monitoring: - External uptime monitoring (UptimeRobot) - 1-minute check frequency - Alert on 2 consecutive failures

1.18.5 Dashboards

1.18.5.1 Operations Dashboard

Metrics: - Current uptime - API request rate - Error rate - Response time distribution - Cache hit rate - Active users

Refresh: Real-time (10 second intervals)

1.18.5.2 Cost Dashboard

Metrics: - Today's cost - This week's cost - This month's cost - Projected monthly cost - Cost breakdown by provider - Cost per report type

Refresh: Hourly

1.18.5.3 Quality Dashboard

Metrics: - Average report confidence score - User ratings distribution - Low-quality report count - Model performance comparison - A/B test results

Refresh: Daily

1.19 17. Disaster Recovery & Backup

1.19.1 Backup Strategy

1.19.1.1 Database Backups

Frequency: - Daily automated backups (3 AM EST) - Weekly full dumps (Sunday 2 AM EST) - Monthly archives to cold storage

Retention: - Daily: 30 days - Weekly: 90 days - Monthly: 1 year

Storage: - Primary: Supabase built-in backups - Secondary: AWS S3 (encrypted) - Archive: AWS Glacier (long-term)

Backup Contents: - All database tables - User profiles and preferences - Generated reports (last 30 days) - Tracking data - Cost tracking data

1.19.1.2 Configuration Backups

Frequency: On every change

Contents: - Environment variables - Supabase configuration - pg_cron job definitions - API key metadata (not keys themselves)

Storage: Git repository (private)

1.19.2 Recovery Procedures

1.19.2.1 Recovery Time Objective (RTO)

- **Critical Data:** <1 hour
- **User Data:** <4 hours
- **Reports:** <24 hours (regenerate)

1.19.2.2 Recovery Point Objective (RPO)

- **User Data:** <24 hours (daily backup)
- **Generated Reports:** <8 hours (3x daily generation)
- **Configuration:** <1 hour (version control)

1.19.2.3 Database Recovery

Scenario 1: Accidental Deletion

Steps: 1. Identify what was deleted 2. Locate most recent backup 3. Restore specific tables/rows 4. Verify data integrity 5. Communicate to affected users

Time: 30-60 minutes

Scenario 2: Database Corruption

Steps: 1. Take database offline 2. Restore from most recent backup 3. Replay transaction logs if available 4. Verify all tables and indexes 5. Run integrity checks 6. Bring database online 7. Monitor for issues

Time: 2-4 hours

Scenario 3: Complete Data Loss

Steps: 1. Provision new database 2. Restore from S3 backup 3. Verify schema and data 4. Update connection strings 5. Run full test suite 6. Gradual traffic migration 7. Monitor closely

Time: 4-8 hours

1.19.3 Disaster Scenarios

1.19.3.1 Scenario: Vercel Outage

Impact: Website unavailable

Mitigation: - Status page updates - Twitter/email communication - Wait for Vercel recovery (no action needed)

Estimated Downtime: Typically <1 hour

1.19.3.2 Scenario: Supabase Outage

Impact: Database and auth unavailable

Mitigation: - Serve static cached content - Display “read-only mode” message - Wait for Supabase recovery

Estimated Downtime: Typically <30 minutes

1.19.3.3 Scenario: Multiple Provider Failures

Impact: Complete service unavailable

Mitigation: 1. Activate status page 2. Communication via social media 3. Manual failover if possible 4. Assess root cause 5. Restore service gradually 6. Post-mortem analysis

Estimated Downtime: 2-8 hours

1.19.3.4 Scenario: Security Breach

Impact: Varies based on breach type

Response: 1. **Immediate:** - Isolate affected systems - Revoke compromised credentials - Enable additional monitoring

2. **Short-term (24 hours):**
   - Assess scope of breach
   - Notify affected users
   - Implement additional security
3. **Medium-term (1 week):**
   - Full security audit
   - Penetration testing
   - Update security policies
4. **Long-term:**
   - Regulatory compliance (if required)
   - Insurance claims
   - Public disclosure (if significant)

1.20 18. Deployment Strategy

1.20.1 Environments

1.20.1.1 Development

- **URL:** http://localhost:3000
- **Database:** Local Supabase
- **Purpose:** Feature development
- **Data:** Synthetic test data

1.20.1.2 Staging

- **URL:** https://staging.yourapp.vercel.app
- **Database:** Staging Supabase
- **Purpose:** Pre-production testing
- **Data:** Anonymized production data

1.20.1.3 Production

- **URL:** https://yourapp.com
- **Database:** Production Supabase
- **Purpose:** Live user traffic
- **Data:** Real user data

1.20.2 Deployment Process

1.20.2.1 Step 1: Pre-Deployment Checks

Automated (CI/CD): - ✅ All tests passing - ✅ Code coverage >80% - ✅ No security vulnerabilities - ✅ Performance benchmarks met - ✅ Lighthouse score >90

Manual: - ✅ Code review approved - ✅ Product requirements met - ✅ Documentation updated - ✅ Stakeholders notified - ✅ Deployment window scheduled

1.20.2.2 Step 2: Staging Deployment

Process: 1. Merge to `develop` branch 2. Automatic deployment to staging 3. Run smoke tests 4. Manual QA testing 5. Approval from product team

Duration: 30-60 minutes

1.20.2.3 Step 3: Production Deployment

Process: 1. Create database backup 2. Run database migrations (if any) 3. Deploy to Vercel production 4. Deploy Supabase functions 5. Run smoke tests 6. Monitor for 30 minutes 7. Declare deployment successful

Duration: 15-30 minutes

Deployment Window: - Preferred: Tuesday/Wednesday 10 AM EST - Avoid: Friday afternoons, weekends - Emergency: Anytime (with approval)

1.20.2.4 Step 4: Post-Deployment

Immediate (0-30 min): - Monitor error rates - Check API latency - Verify batch jobs running - Watch user reports

Short-term (30 min - 24 hrs): - Analyze user feedback - Review cost metrics - Check quality scores - Monitor engagement

Long-term (1 week): - Full metrics analysis - User surveys - Performance optimization - Plan next iteration

1.20.3 Rollback Procedure

When to Rollback: - Error rate >5% - Critical feature broken - Data corruption detected - Security vulnerability discovered - User impact >10% of user base

How to Rollback:

1. **Immediate (5 min):**

   ```
   vercel rollback
   ```

2. **Database (if migrations ran):**

   ```
   npm run migrate:rollback
   ```

3. **Verification (5 min):**

   - Check error rates
   - Verify functionality
   - Monitor metrics

4. **Communication (15 min):**

   - Update status page
   - Notify team
   - Email affected users (if applicable)

Post-Rollback: 1. Root cause analysis 2. Fix issue 3. Test thoroughly 4. Redeploy when ready

1.20.4 Feature Flags

Purpose: Gradual rollout and A/B testing

Implementation: Environment variables + database flags

Use Cases: - New report format (test with 10% users) - Experimental AI models - Premium features - UI redesigns

Example:

```
if (featureFlags.newReportFormat && user.id % 10 === 0) {
  // Use new format
} else {
  // Use old format
}
```

1.21 Appendix A: Glossary

API: Application Programming Interface - way for software to communicate

Cache: Temporary storage for faster data retrieval

DCF: Discounted Cash Flow - valuation method

Edge Functions: Code running at CDN edge locations (close to users)

JSONB: JSON Binary - efficient JSON storage in PostgreSQL

JWT: JSON Web Token - authentication token format

LRU: Least Recently Used - cache eviction strategy

P/E Ratio: Price-to-Earnings ratio - valuation metric

pg_cron: PostgreSQL extension for scheduled jobs

RLS: Row Level Security - database-level access control

SLA: Service Level Agreement - reliability target

Stale-While-Revalidate: Cache pattern serving old content while updating

TTL: Time To Live - how long cached data is valid

XBRL: eXtensible Business Reporting Language - financial data format

---

1.22 Appendix B: Key Decisions & Rationale

1.22.1 Decision 1: Next.js on Vercel

Rationale: Zero-config deployment, excellent performance, Edge Functions support, generous free tier

1.22.2 Decision 2: Supabase PostgreSQL

Rationale: Built-in auth, RLS, real-time capabilities, managed service, cost-effective

1.22.3 Decision 3: Redis for Hot Cache

Rationale: Extremely fast, reduces database load, prevents cache stampede, Upstash serverless

1.22.4 Decision 4: Groq Llama 8B Primary Model

Rationale: Best cost-performance ratio, fast inference, sufficient quality for summaries

1.22.5 Decision 5: Tier System

Rationale: Optimizes costs through batching, serves users efficiently, scales well

1.22.6 Decision 6: 3x Daily Updates

Rationale: Balances freshness with cost, aligns with market hours, user expectations

1.22.7 Decision 7: Lazy Load Deep Reports

Rationale: Expensive to generate, only premium users need, on-demand reduces waste

1.22.8 Decision 8: Supabase pg_cron

Rationale: Simplest solution, native integration, free, reliable

1.22.9 Decision 9: Stale-While-Revalidate

Rationale: Always fast responses, continuous freshness, great UX

1.22.10 Decision 10: Multi-Provider AI Strategy

Rationale: Reduces vendor lock-in, provides fallbacks, optimizes cost

---

1.23 Appendix C: Contact & Resources

1.23.1 Team

- **Project Lead:** [Your Name]
- **DevOps:** [Engineer Name]
- **Product Manager:** [PM Name]

1.23.2 Documentation

- **API Docs:** https://docs.yourapp.com/api
- **User Guide:** https://docs.yourapp.com/guide
- **Admin Guide:** https://docs.yourapp.com/admin

1.23.3 Infrastructure

- **Production:** https://yourapp.com
- **Staging:** https://staging.yourapp.vercel.app
- **Status Page:** https://status.yourapp.com
- **Admin Dashboard:** https://yourapp.com/admin

1.23.4 Support

- **Email:** support@yourapp.com
- **Slack:** #stock-research-support
- **On-Call:** [Phone Number]

Document Version: 1.0
Last Updated: November 17, 2025
Next Review: December 17, 2025