AiiA Stock Analysis Implementation Documentation

Overview

The AiiA application implements a comprehensive stock and cryptocurrency analysis system that combines multiple data sources, AI processing, and real-time market data to provide investment recommendations. This document outlines the complete analysis pipeline, data sources, algorithms, and implementation details.

Architecture Overview

The stock analysis system follows this high-level flow:

```
User Input (Symbol) → API Endpoint → Data Collection → AI Analysis → Database Storage → Frontend Display
```

Core Components

1. Main Analysis API Endpoint

File: `/app/api/analysis/[symbol]/route.ts`

This is the primary entry point for stock analysis requests.

Key Functions:
- `GET()` - Main handler for analysis requests
- `generateAIAnalysis()` - Core analysis generation logic
- Helper functions for technical analysis calculations

Request Parameters:
- `symbol` - Stock/crypto symbol (e.g., “AAPL”, “BTC”)
- `type` - Asset type (“stock” or “crypto”)
- `refresh` - Boolean to force new analysis generation

Response Structure:

```
{
  analysis: {
    symbol: string,
    type: 'stock' | 'crypto',
    recommendation: 'buy' | 'sell' | 'hold',
    confidence: number, // 0-100
    reasoning: string,
    technicalScore: number,
    fundamentalScore: number, // stocks only
    sentimentScore: number,
    riskLevel: 'low' | 'medium' | 'high',
    targetPrice: number,
    stopLoss: number,
    lastUpdated: Date
  }
}
```

2. Data Collection Layer

File: `/lib/market-data.ts`

This module handles data collection from multiple external sources with caching and rate limiting.

Data Sources Used:

1. **Alpha Vantage API**
   - Stock quotes and historical data
   - News sentiment analysis
   - API Key: `ALPHADVANTAGE_API_KEY`
   - Rate Limit: 5 requests/minute, 25 requests/day
   - Endpoints used:
     - `GLOBAL_QUOTE` for stock prices
     - `TIME_SERIES_DAILY` for chart data
     - `NEWS_SENTIMENT` for news with sentiment
2. **Finnhub API**
   - Stock quotes and company news
   - API Key: `FINNHUB_API_KEY`
   - Rate Limit: 60 requests/minute, 1000 requests/day
   - Endpoints used:
     - `/quote` for stock prices
     - `/company-news` for company-specific news
     - `/search` for stock symbol search
3. **News API**
   - General financial news
   - API Key: `NEWS_API_KEY`
   - Rate Limit: 1000 requests/day
   - Endpoint: `/everything` for financial news
4. **CoinGecko API**
   - Cryptocurrency data
   - No API key required (free tier)

- Rate Limit: 30 requests/minute
- Endpoints used:
  - `/simple/price` for crypto prices
  - `/coins/markets` for market movers
  - `/coins/{id}/ohlc` for chart data
  - `/search` for crypto search

5. **Yahoo Finance (Unofficial API)**
   - Stock quotes and market data
   - No API key required
   - Used as primary data source due to reliability
   - Endpoints used:
     - Chart API for stock quotes
     - Screener API for market movers
     - Search API for symbol lookup

Key Functions:

```
// Stock data retrieval
async function getStockQuote(symbol: string): Promise<Asset | null>

// Cryptocurrency data retrieval
async function getCryptoQuote(symbol: string): Promise<Asset | null>

// Asset search across multiple sources
async function searchAssets(query: string): Promise<Asset[]>

// Market movers (gainers/losers)
async function getMarketMovers(type: 'stock' | 'crypto'): Promise<{gainers:
MarketMover[], losers: MarketMover[]}>

// Historical chart data
async function getChartData(symbol: string, type: 'stock' | 'crypto', interval: string
= 'daily'): Promise<ChartData[]>

// News articles with sentiment
async function getNews(symbols?: string[]): Promise<NewsArticle[]>

// Comprehensive data aggregation
async function getComprehensiveAssetData(symbol: string, type: 'stock' | 'crypto'): Pro
mise<{
  quote: Asset | null,
  news: NewsArticle[],
  sentiment: any,
  confidence: number
}>
```

3. Web Scraping and Social Sentiment

File: `/lib/web-scraping.ts`

Provides additional data sources through web scraping and social sentiment analysis.

Functions:

```
// Yahoo Finance data scraping
async function scrapeYahooFinanceQuote(symbol: string): Promise<Asset | null>
async function scrapeYahooMarketMovers(): Promise<{gainers: MarketMover[], losers: Mar-
ketMover[]}>
async function scrapeYahooFinanceSearch(query: string): Promise<Asset[]>

// Social sentiment analysis (placeholder implementation)
async function scrapeSocialSentiment(symbol: string): Promise<SentimentData | null>
async function scrapeRedditSentiment(symbol: string): Promise<SentimentData | null>

// Data aggregation from multiple sources
async function aggregateMarketData(symbol: string, type: 'stock' | 'crypto'): Promise<{
  prices: Asset[],
  sentiment: SentimentData[],
  confidence: number
}>
```

4. AI Analysis Engine

The AI analysis is powered by **Abacus.AI's GPT-4.1-mini model** via their API.

API Configuration:
- Endpoint: `https://apps.abacus.ai/v1/chat/completions`
- API Key: `ABACUSAI_API_KEY`
- Model: `gpt-4.1-mini`
- Response Format: JSON object

Analysis Process:

1. **Data Gathering**: Collect comprehensive market data from all sources
2. **Technical Analysis**: Calculate volatility, trend direction, support/resistance
3. **Sentiment Analysis**: Aggregate news sentiment and social media sentiment
4. **AI Prompt Generation**: Create detailed analysis prompt with all collected data
5. **AI Processing**: Send data to Abacus.AI for analysis
6. **Confidence Scoring**: Calculate final confidence based on data quality and AI output
7. **Database Storage**: Save analysis results with expiration

AI System Prompt:

```
const systemPrompt = `You are a senior financial analyst with expertise in ${type === 'stock' ? 'equity' : 'cryptocurrency'} analysis. You have access to comprehensive market data from multiple sources including real-time prices, technical indicators, news sentiment, and social media sentiment.

Your analysis should provide:
1. Overall recommendation (buy/sell/hold)
2. Confidence score (0-100) - Consider data quality, market conditions, and analysis certainty
3. Technical analysis score (0-100) - Based on price action, volume, trends, and indicators
4. ${type === 'stock' ? 'Fundamental analysis score (0-100) - Based on company fundamentals and market position' : 'Market momentum score (0-100) - Based on adoption, development activity, and market dynamics'}
5. Sentiment analysis score (0-100) - Based on news sentiment and social media sentiment
6. Risk assessment (low/medium/high)
7. Target price (conservative estimate)
8. Stop loss recommendation (risk management)
9. Detailed reasoning (2-3 paragraphs explaining your analysis)

Analysis Factors to Consider:
- Technical indicators and price trends
- Volume analysis and market momentum
- News sentiment and market reaction
- Social media sentiment (if available)
- Market volatility and risk factors
- ${type === 'stock' ? 'Company fundamentals and sector performance' : 'Network activity and adoption metrics'}
- Overall market conditions

Important: Base your confidence score on the quality and quantity of available data. Higher data source count and fresher data should increase confidence.

Respond with raw JSON only. Do not include code blocks, markdown, or any other formatting.`
```

Technical Analysis Calculations:

```
// Volatility calculation using standard deviation of returns
function calculateVolatility(chartData: any[]): number {
  if (chartData.length < 2) return 0

  const returns = chartData.slice(1).map((current, index) => {
    const previous = chartData[index]
    return (current.close - previous.close) / previous.close
  })

  const avgReturn = returns.reduce((sum, ret) => sum + ret, 0) / returns.length
  const variance = returns.reduce((sum, ret) => sum + Math.pow(ret - avgReturn, 2),
0) / returns.length

  return Math.sqrt(variance) * 100 // Convert to percentage
}

// Trend direction analysis
function calculateTrend(chartData: any[]): string {
  if (chartData.length < 10) return 'neutral'

  const recent = chartData.slice(-10)
  const older = chartData.slice(-20, -10)

  const recentAvg = recent.reduce((sum, data) => sum + data.close, 0) / recent.length
  const olderAvg = older.reduce((sum, data) => sum + data.close, 0) / older.length

  const change = (recentAvg - olderAvg) / olderAvg

  if (change > 0.02) return 'bullish'
  if (change < -0.02) return 'bearish'
  return 'neutral'
}

// Support and resistance levels
function calculateSupportResistance(chartData: any[]): { support: number, resistance: n
umber } {
  if (chartData.length < 20) return { support: 0, resistance: 0 }

  const prices = chartData.map(data => data.close)
  const sortedPrices = [...prices].sort((a, b) => a - b)

  return {
    support: sortedPrices[Math.floor(sortedPrices.length * 0.2)], // 20th percentile
    resistance: sortedPrices[Math.floor(sortedPrices.length * 0.8)] // 80th percentile
  }
}
```

Confidence Score Calculation:

The final confidence score is calculated using multiple factors:

```
const baseConfidence = analysisResult.confidence || 50
const dataQualityBoost = Math.min(25, dataSourcesAvailable * 5) // Up to 25 points for
data sources
const volumeBoost = asset.volume && asset.volume > 1000000 ? 5 : 0
const newsBoost = news.length > 5 ? 5 : news.length
const sentimentBoost = sentiment.length > 0 ? 5 : 0
const volatilityPenalty = Math.abs(asset.changePercent) > 10 ? -5 : 0

const finalConfidence = Math.min(98, Math.max(20,
  baseConfidence + dataQualityBoost + volumeBoost + newsBoost + sentimentBoost + volat-
ilityPenalty
))
```

5. Database Storage

Database: PostgreSQL via Prisma ORM

Analysis Storage Schema:

```
model AIAnalysis {
  id               String   @id @default(cuid())
  symbol           String
  type             String   // 'stock' or 'crypto'
  recommendation   String   // 'buy', 'sell', 'hold'
  confidence       Int      // 0-100
  reasoning        String
  technicalScore   Int?
  fundamentalScore Int?
  sentimentScore   Int?
  riskLevel        String?  // 'low', 'medium', 'high'
  targetPrice      Float?
  stopLoss         Float?
  dataSource       Json     // Raw analysis data
  expiresAt        DateTime
  createdAt        DateTime @default(now())
  updatedAt        DateTime @updatedAt

  @@unique([symbol, type])
}
```

6. Frontend Components

Analysis Display Component

File: `/components/ai-analysis.tsx`

Displays the AI analysis results with:
- Recommendation badge with confidence score
- Technical, fundamental, and sentiment scores
- Price targets (target price and stop loss)
- Detailed reasoning
- Refresh functionality

Analysis Page

File: `/app/analyze/page.tsx`

Main analysis interface providing:
- Asset search functionality

- Real-time analysis generation
- News integration
- Interactive charts
- Watchlist management

Data Flow Diagram

```
1. User searches for symbol → Asset Search API
2. User selects asset → Analysis API called
3. Analysis API → Data Collection Layer
4. Data Collection → Multiple External APIs
   ├── Alpha Vantage (stocks, news)
   ├── Finnhub (stocks, news)
   ├── News API (general news)
   ├── CoinGecko (crypto)
   └── Yahoo Finance (stocks)
5. Collected Data → Technical Analysis Calculations
6. All Data → AI Analysis Engine (Abacus.AI)
7. AI Response → Confidence Score Calculation
8. Final Analysis → Database Storage
9. Analysis → Frontend Display
```

Caching and Performance

Caching Strategy:

- **Market Data**: 1-minute cache for quotes, 5-minute cache for market movers
- **News**: 10-minute cache
- **Search Results**: 5-minute cache
- **AI Analysis**: 1-hour expiration in database

Rate Limiting:

- Per-API rate limiting with request tracking
- Fallback mechanisms when APIs are unavailable
- Graceful degradation with reduced data sources

Error Handling

Fallback Mechanisms:

1. **Primary Data Source Failure**: Automatically try secondary sources
2. **API Rate Limits**: Use cached data or alternative sources
3. **AI Analysis Failure**: Return cached analysis or basic recommendation
4. **Network Issues**: Display cached data with staleness indicators

Error Recovery:

- Retry logic for transient failures
- Circuit breaker pattern for consistently failing services
- User-friendly error messages with retry options

Security Considerations

API Key Management:

- All API keys stored in environment variables
- Keys not exposed to frontend
- Rate limiting to prevent abuse

Data Validation:

- Input sanitization for all user inputs
- Response validation from external APIs
- SQL injection prevention via Prisma ORM

Performance Metrics

Typical Response Times:

- **Cached Analysis**: < 100ms
- **New Analysis Generation**: 2-5 seconds
- **Data Collection**: 1-3 seconds
- **AI Processing**: 1-2 seconds

Data Source Reliability:

- **Yahoo Finance**: 95% uptime, fastest response
- **Alpha Vantage**: 90% uptime, rate limited
- **Finnhub**: 95% uptime, good coverage
- **CoinGecko**: 98% uptime, crypto-focused
- **News API**: 90% uptime, comprehensive news

Future Enhancements

Planned Improvements:

1. **Real-time Social Sentiment**: Integration with Twitter/Reddit APIs
2. **Advanced Technical Indicators**: RSI, MACD, Bollinger Bands
3. **Fundamental Analysis**: P/E ratios, earnings data, financial statements
4. **Machine Learning Models**: Custom ML models for price prediction
5. **Real-time Data Streaming**: WebSocket connections for live updates
6. **Portfolio Analysis**: Correlation analysis and risk assessment

Scalability Considerations:

- Database sharding for high-volume analysis storage
- Redis caching layer for improved performance
- Microservices architecture for independent scaling
- CDN integration for static content delivery

Conclusion

The AiiA stock analysis system provides comprehensive investment analysis by combining multiple data sources, advanced AI processing, and real-time market data. The system is designed for reliability, performance, and scalability while maintaining accuracy through diverse data sources and sophisticated confidence scoring mechanisms.