AIAlgoTradeHits.com

Architecture Refactoring Masterplan

Combined Architecture Document & Reengineering Strategy

Document Version: 3.0

Platform Version: 2.0.0 (Current) -> 3.0.0 (Target)

Created: January 12, 2026

Status: Refactoring Required | Architecture Redesign

Reference Documents:

1. AIAlgoTradeHits Architecture Document (January 2026)

2. Economic Intelligence Platform File Architecture v2.0 (Saleem Ahmad)

3. masterquery.md v4.0 - Trading System Specifications

Executive Summary

This document presents a comprehensive refactoring plan for AIAlgoTradeHits.com, combining insights from two architectural approaches: the current React/Vite-based trading platform and Saleem Ahmad's clean architecture principles from the Economic Intelligence Platform. The goal is to transform AIAlgoTradeHits into a state-of-the-art fintech application with enterprise-grade architecture.

Key Transformation Goals:

Single Source of Truth (SSOT)	All configuration centralized in one location
Layer Cake Architecture	Config -> Engine -> Service -> API -> UI separation
TypeScript Migration	Full type safety across the codebase
Feature-Based Organization	Components grouped by feature, not type
NLP-Driven Operations	Eliminate need for external programming
Redundancy Elimination	Remove 13 redundant components (5,373 lines)

1. Architecture Comparison

1.1 Technology Stack Comparison

Layer	AIAlgoTradeHits (Current)	EI Platform (Target)	Migration
Framework	React 18 + Vite	Next.js 14 App Router	Major
Language	JavaScript (JSX)	TypeScript (TSX)	Required
Styling	Inline Styles + CSS	Tailwind CSS 3.x	Recommended
Database	Google BigQuery	BigQuery + Supabase	Optional
Charts	Lightweight Charts	Recharts 2.x	Keep Current
AI	Gemini + XGBoost	Claude API	Keep Gemini
Deployment	GCP Cloud Run	Vercel	Keep GCP
State	React useState	Context + Hooks	Enhance

1.2 Architecture Philosophy Comparison

Principle	AIAlgoTradeHits	EI Platform	Gap Analysis
Config Location	Scattered in components	lib/config/macro-config.ts	CRITICAL GAP
Calculation Logic	Mixed in components	Pure engines (no I/O)	CRITICAL GAP
Data Flow	Bidirectional	Unidirectional down only	MAJOR GAP
Import Paths	Relative (../../../)	Absolute (@/)	MODERATE GAP
Component Org	Flat (42 files)	Feature-based folders	MAJOR GAP
Type Safety	None (JavaScript)	Full TypeScript	MAJOR GAP
API Structure	Single service file	Route-based handlers	MODERATE GAP

1.3 The Layer Cake Architecture (Target)

The EI Platform uses a strict layered architecture where data flows DOWN only. This prevents circular dependencies and makes the system testable and maintainable.

Layer	Purpose	AIAlgoTradeHits Equivalent	Required Changes
USER INTERFACE	Pages & Components	src/components/*.jsx	Reorganize by feature
API ROUTES	Request Handlers	cloud_function_api/	Add frontend API layer
SERVICE LAYER	Orchestration & Logic	src/services/api.js	Split into services
ENGINE LAYER	Pure Calculations	MISSING	CREATE NEW
CONFIG LAYER	SSOT Configuration	MISSING	CREATE NEW
DATA LAYER	BigQuery, APIs	BigQuery + 5 APIs	Keep, add abstraction

2. Current State Analysis

2.1 AIAlgoTradeHits Current File Structure

The current AIAlgoTradeHits platform has grown organically, resulting in a flat component structure with 42 React components, scattered configuration, and mixed concerns within components.

```
stock-price-app/
■■■ src/
■ ■■■ components/ # FLAT: 42 components in one folder
■ ■ ■■■ TradingDashboard.jsx (1,847 lines) - Main dashboard
■ ■ ■■■ AdminPanelEnhanced.jsx (687 lines) - Admin functions
■ ■ ■■■ ProfessionalChart.jsx (612 lines) - Chart component
■ ■ ■■■ SmartDashboard.jsx (580 lines) - AI dashboard
■ ■ ■■■ ... (38 more components)
■ ■ ■■■ Navigation.jsx (740 lines) - Side navigation
■ ■
■ ■■■ services/
■ ■ ■■■ api.js (2,100+ lines) - ALL API calls in ONE file
■ ■ ■■■ marketData.js - Market data service
■ ■ ■■■ aiService.js - AI service calls
■ ■ ■■■ monitoringService.js - Monitoring
■ ■
■ ■■■ App.jsx - Main app with ALL routes (25+)
■ ■■■ App.css - Global styles
■ ■■■ main.jsx - Entry point
■
■■■ public/ - Static assets
```

2.2 Identified Redundant Components

Component	Lines	Reason for Redundancy	Action
EnhancedDashboard.jsx	821	Superseded by SmartDashboard	DELETE
NLPSearch.jsx	628	Integrated into SmartDashboard	DELETE
WeeklyReconciliation.jsx	620	Not used in routing	DELETE
AIAlgoTradeHits.jsx	564	Duplicate landing page	DELETE
AIAlgoTradeHitsReal.jsx	542	Duplicate landing page	DELETE
AdminPanel.jsx	453	Superseded by AdminPanelEnhanced	DELETE
DataDownloadControl.jsx	398	Merged into DataExportDownload	DELETE
AdvancedChart.jsx	380	Replaced by ProfessionalChart	DELETE
WeeklyAnalysis.jsx	367	Merged into WeeklyDashboard	DELETE
DataDownloadWizard.jsx	352	Duplicate of DataExportDownload	DELETE
AdvancedTradingChart.jsx	318	Replaced by TradingViewChart	DELETE
MultiPanelChart.jsx	280	Not actively used	REVIEW
FundamentalsView.jsx	250	Incomplete implementation	REVIEW
TOTAL	**5,373**	**Lines to eliminate**	**-**

2.3 Redundant Cloud Functions

Function Folder	Status	Replaced By
cloud_function_5min/	Redundant	bulletproof-fetcher
cloud_function_daily/	Redundant	bulletproof-fetcher
cloud_function_hourly/	Redundant	bulletproof-fetcher
cloud_function_stocks_5min/	Redundant	bulletproof-fetcher
cloud_function_stocks_daily/	Redundant	bulletproof-fetcher
cloud_function_stocks_hourly/	Redundant	bulletproof-fetcher
cloud_function_weekly_cryptos/	Redundant	bulletproof-fetcher
cloud_function_weekly_stocks/	Redundant	bulletproof-fetcher
cloud_function_max_quota/	Redundant	bulletproof-fetcher
cloud_function_multi_source/	Redundant	bulletproof-fetcher

3. Target Architecture Design

3.1 New Directory Structure (Following EI Platform Pattern)

```
stock-price-app/
■
■■■ src/
■   ■■■ lib/                                 # CORE LOGIC LAYER
■   ■   ■■■ config/                          # CONFIGURATION (SSOT)
■   ■   ■   ■■■ trading-config.ts            # Master config - indicators, thresholds
■   ■   ■   ■■■ scoring-config.ts            # Growth score, sentiment rules
■   ■   ■   ■■■ api-config.ts                # API endpoints, rate limits
■   ■   ■   ■■■ ui-config.ts                 # Theme, colors, chart settings
■   ■   ■
■   ■   ■■■ engines/                         # CALCULATION ENGINES (Pure functions)
■   ■   ■   ■■■ indicator-engine.ts          # Technical indicator calculations
■   ■   ■   ■■■ signal-engine.ts             # Buy/Sell/Hold signal generation
■   ■   ■   ■■■ growth-score-engine.ts       # Growth score calculations
■   ■   ■   ■■■ nested-ml-engine.ts          # ML prediction logic
■   ■   ■
■   ■   ■■■ services/                        # SERVICE ORCHESTRATION
■   ■   ■   ■■■ market-data-service.ts       # TwelveData integration
■   ■   ■   ■■■ trading-signal-service.ts    # Signal orchestration
■   ■   ■   ■■■ ai-prediction-service.ts     # ML/AI predictions
■   ■   ■   ■■■ nlp-query-service.ts         # Natural language queries
■   ■   ■   ■■■ data-export-service.ts       # Data download service
■   ■   ■
■   ■   ■■■ hooks/                           # REACT HOOKS
■   ■   ■   ■■■ useMarketData.ts             # Market data hook
■   ■   ■   ■■■ useTradingSignals.ts         # Signals hook
■   ■   ■   ■■■ useMLPredictions.ts          # ML predictions hook
■   ■   ■   ■■■ useNLPQuery.ts               # NLP query hook
■   ■   ■
■   ■   ■■■ utils/                           # UTILITIES
■   ■       ■■■ formatters.ts                # Number/date formatting
■   ■       ■■■ validators.ts                # Input validation
■   ■       ■■■ transforms.ts                # Data transformations
■   ■
■   ■■■ components/                          # UI COMPONENT LAYER
■   ■   ■■■ dashboard/                       # Dashboard-specific
■   ■   ■   ■■■ TradingDashboard.tsx
■   ■   ■   ■■■ SmartDashboard.tsx
■   ■   ■   ■■■ StatCards.tsx
■   ■   ■
■   ■   ■■■ charts/                          # Chart components
■   ■   ■   ■■■ ProfessionalChart.tsx
■   ■   ■   ■■■ VolumeChart.tsx
■   ■   ■   ■■■ IndicatorOverlay.tsx
■   ■   ■
■   ■   ■■■ signals/                         # Signal components
■   ■   ■   ■■■ NestedSignals.tsx
■   ■   ■   ■■■ AITradeSignals.tsx
■   ■   ■   ■■■ SignalCard.tsx
■   ■   ■
■   ■   ■■■ admin/                           # Admin components
■   ■   ■   ■■■ AdminPanelEnhanced.tsx
■   ■   ■   ■■■ SchedulerMonitoring.tsx
■   ■   ■   ■■■ DatabaseMonitoring.tsx
■   ■   ■
■   ■   ■■■ shared/                          # Reusable across pages
■   ■   ■   ■■■ Navigation.tsx
■   ■   ■   ■■■ SearchBar.tsx
■   ■   ■   ■■■ LoadingSpinner.tsx
■   ■   ■
■   ■   ■■■ ui/                              # Base UI primitives
■   ■       ■■■ Button.tsx
■   ■       ■■■ Card.tsx
■   ■       ■■■ Badge.tsx
■   ■
■   ■■■ pages/                               # PAGE COMPONENTS
■   ■   ■■■ Dashboard.tsx
■   ■   ■■■ Signals.tsx
■   ■   ■■■ Admin.tsx
■   ■   ■■■ Settings.tsx
■   ■
■   ■■■ types/                          # TYPE DEFINITIONS
■   ■   ■■■ trading.ts                   # Trading types
■   ■   ■■■ api.ts                       # API response types
■   ■   ■■■ index.ts                     # Re-exports
■   ■
■   ■■■ App.tsx                         # Main app (simplified routing)
■   ■■■ main.tsx                        # Entry point
■
■■■ tailwind.config.ts                  # Tailwind configuration
■■■ tsconfig.json                       # TypeScript configuration
■■■ vite.config.ts                      # Vite configuration
```

4. Config Layer Design (SSOT)

"One config to rule them all" - All business logic configuration in one place

4.1 Master Trading Config (trading-config.ts)

```
// src/lib/config/trading-config.ts

export const LOGIC_VERSION = "3.0.0";

// INDICATOR THRESHOLDS (from masterquery.md v4.0)
export const INDICATOR_THRESHOLDS = {
  RSI: { oversold: 30, overbought: 70, sweet_spot: [50, 70] },
  MACD: { signal_cross: 0, histogram_threshold: 0 },
  ADX: { weak: 20, strong: 25, very_strong: 40 },
  VOLUME: { surge_multiplier: 1.5 },
};

// GROWTH SCORE CALCULATION (0-100)
export const GROWTH_SCORE_RULES = {
  rsi_sweet_spot: 25,      // RSI 50-70 = 25 points
  macd_positive: 25,        // MACD histogram > 0 = 25 points
  strong_trend: 25,         // ADX > 25 = 25 points
  above_sma200: 25,         // Close > SMA200 = 25 points
};

// TREND REGIME CLASSIFICATION
export const TREND_REGIMES = {
  STRONG_UPTREND: { sma_condition: 'above_both', adx_min: 25 },
  WEAK_UPTREND: { sma_condition: 'above_50_200' },
  STRONG_DOWNTREND: { sma_condition: 'below_both', adx_min: 25 },
  WEAK_DOWNTREND: { sma_condition: 'below_50_200' },
  CONSOLIDATION: { default: true },
};

// EMA CYCLE DETECTION
export const EMA_CYCLES = {
  rise_cycle: { condition: 'ema12 > ema26' },
  fall_cycle: { condition: 'ema12 < ema26' },
};

// NESTED ML SIGNAL THRESHOLDS
export const NESTED_SIGNAL_THRESHOLDS = {
  ULTRA_BUY: { aligned_pct: 60, min_scores: [5, 6, 5] },
  STRONG_BUY: { aligned_pct: 50, min_scores: [4, 5, 4] },
  BUY: { daily_hourly_aligned: true, min_scores: [4, 4] },
  WEAK_BUY: { daily_bullish: true, min_score: 4 },
  HOLD: { default: true },
};

// API RATE LIMITS
export const API_LIMITS = {
  TWELVEDATA: { calls_per_min: 800, outputsize: 5000 },
  KRAKEN: { calls_per_min: 60 },
  FRED: { calls_per_day: 100 },
  FINNHUB: { calls_per_min: 60 },
  COINMARKETCAP: { calls_per_day: 333 },
};
```

4.2 Config Files Summary

Config File	Purpose	Contents
trading-config.ts	Master config (SSOT)	Indicators, thresholds, regimes, ML params
scoring-config.ts	Scoring rules	Growth score, sentiment, signal logic

api-config.ts	API configuration	Endpoints, rate limits, keys reference
ui-config.ts	UI constants	Colors, chart sizes, theme settings

5. Engine Layer Design (Pure Logic)

> *"No side effects, no I/O, just math" - Engines are pure functions*

5.1 Engine Design Principles

Engines contain all calculation logic but never make API calls, access databases, or have side effects. They take inputs and return outputs - making them fully testable and reusable.

```
// src/lib/engines/growth-score-engine.ts

import { GROWTH_SCORE_RULES, INDICATOR_THRESHOLDS } from '@/lib/config/trading-config';

interface IndicatorData {
  rsi_14: number;
  macd_histogram: number;
  adx: number;
  close: number;
  sma_200: number;
}

/**
 * Calculate Growth Score (0-100) - PURE FUNCTION
 * No API calls, no database access, just calculations
 */
export function calculateGrowthScore(data: IndicatorData): number {
  let score = 0;

  // RSI sweet spot (50-70)
  const [low, high] = INDICATOR_THRESHOLDS.RSI.sweet_spot;
  if (data.rsi_14 >= low && data.rsi_14 <= high) {
    score += GROWTH_SCORE_RULES.rsi_sweet_spot;
  }

  // MACD histogram positive
  if (data.macd_histogram > 0) {
    score += GROWTH_SCORE_RULES.macd_positive;
  }

  // Strong trend (ADX > 25)
  if (data.adx > INDICATOR_THRESHOLDS.ADX.strong) {
    score += GROWTH_SCORE_RULES.strong_trend;
  }

  // Above SMA200
  if (data.close > data.sma_200) {
    score += GROWTH_SCORE_RULES.above_sma200;
  }

  return score;
}

/**
 * Classify Trend Regime - PURE FUNCTION
 */
export function classifyTrendRegime(
  close: number,
  sma50: number,
  sma200: number,
  adx: number
): string {
  if (close > sma50 && sma50 > sma200 && adx > 25) return 'STRONG_UPTREND';
  if (close > sma50 && close > sma200) return 'WEAK_UPTREND';
  if (close < sma50 && sma50 < sma200 && adx > 25) return 'STRONG_DOWNTREND';
  if (close < sma50 && close < sma200) return 'WEAK_DOWNTREND';
  return 'CONSOLIDATION';
}
```

5.2 Engine Files Summary

Engine	Purpose	Key Functions
indicator-engine.ts	Technical indicators	calculateRSI, calculateMACD, calculateEMA
signal-engine.ts	Trade signals	generateSignal, classifyAction, getRecommendation
growth-score-engine.ts	Growth scoring	calculateGrowthScore, classifyTrendRegime
nested-ml-engine.ts	ML predictions	calculateNestedSignal, predictDirection
sentiment-engine.ts	Sentiment analysis	calculateSentiment, analyzeNews

6. Refactoring Implementation Phases

6.1 Phase Overview

Phase	Focus Area	Effort	Risk	Priority
Phase 1	Config Layer (SSOT)	Medium	Low	CRITICAL
Phase 2	Engine Layer	High	Medium	HIGH
Phase 3	TypeScript Migration	High	Medium	HIGH
Phase 4	Component Reorganization	Medium	Low	MEDIUM
Phase 5	Redundancy Cleanup	Low	Low	MEDIUM
Phase 6	NLP Enhancement	High	Medium	HIGH
Phase 7	Testing & Validation	Medium	Low	HIGH

6.2 Phase 1: Config Layer (SSOT)

Tasks:

1. Create src/lib/config/ directory structure
2. Extract all hardcoded thresholds from components into trading-config.ts
3. Move indicator calculations to scoring-config.ts
4. Centralize API endpoints and limits in api-config.ts
5. Extract UI constants (colors, sizes) to ui-config.ts
6. Update all components to import from config files
7. Add LOGIC_VERSION for cache invalidation

6.3 Phase 2: Engine Layer

Tasks:

1. Create src/lib/engines/ directory
2. Extract calculateGrowthScore from TradingDashboard.jsx
3. Extract signal generation logic from AITradeSignals.jsx
4. Create indicator-engine.ts with pure calculation functions
5. Create nested-ml-engine.ts for ML prediction logic
6. Ensure all engines have NO I/O operations
7. Add comprehensive unit tests for each engine

6.4 Phase 3: TypeScript Migration

Tasks:

1. Configure TypeScript in vite.config.ts
2. Create src/types/ directory with type definitions
3. Define Trading, API, and UI types
4. Migrate config files to TypeScript first
5. Migrate engine files to TypeScript
6. Migrate service files to TypeScript
7. Migrate components (starting with shared/)
8. Update import statements to use absolute paths (@/)

6.5 Phase 4: Component Reorganization

Current Location	Target Location	Components
components/*.jsx	components/dashboard/	TradingDashboard, SmartDashboard, StatCards
components/*.jsx	components/charts/	ProfessionalChart, TradingViewChart
components/*.jsx	components/signals/	NestedSignals, AITradeSignals, AIPredictions
components/*.jsx	components/admin/	AdminPanelEnhanced, SchedulerMonitoring
components/*.jsx	components/shared/	Navigation, SmartSearchBar, Login
components/*.jsx	components/data/	DataExportDownload, MLTestDataDownload

6.6 Phase 5: Redundancy Cleanup

Delete the 13 redundant components identified in Section 2.2. This will eliminate 5,373 lines of duplicate code and simplify maintenance. Archive files before deletion for reference.

6.7 Phase 6: NLP Enhancement

Feature	Description	Implementation
Text-to-SQL	Natural language to BigQuery	Gemini Pro + SQL templates
Voice Search	Spoken queries to actions	Web Speech API + NLP
Smart Autocomplete	Context-aware suggestions	History + ML predictions
Natural Commands	"Show AAPL daily chart"	Intent classification + routing
Report Generation	"Generate weekly report"	Template + AI summarization

7. Refactored Site Map

7.1 Core Navigation Structure

Route	Component	Description	Status
/	Dashboard	Main trading dashboard with overview	Keep
/dashboard	TradingDashboard	Detailed trading view	Keep
/signals	NestedSignals	Multi-timeframe ML signals	NEW
/ai-signals	AITradeSignals	AI-generated trade signals	Keep
/ai-predictions	AIPredictions	ML prediction display	Keep
/charts/:symbol	ProfessionalChart	Symbol-specific chart	Keep
/admin	AdminPanelEnhanced	Administration panel	Keep
/scheduler	SchedulerMonitoring	Job scheduler monitor	Keep
/database	DatabaseMonitoring	Database status monitor	Keep
/data-export	DataExportDownload	Data download wizard	Keep
/settings	Settings	User preferences	NEW
/login	Login	Authentication	Keep

7.2 Routes to Remove

Route	Reason
/enhanced-dashboard	Merged into /dashboard
/nlp-search	Integrated into SmartSearchBar
/weekly-reconciliation	Not actively used
/landing	Duplicate - use /
/data-wizard	Merged into /data-export

8. ML Model Integration Architecture

8.1 Nested Multi-Timeframe Model

The Nested Multi-Timeframe ML Model (66.2% UP accuracy) requires proper integration into the refactored architecture. This section outlines how to structure the ML components.

Component	Location	Purpose
Model Config	lib/config/trading-config.ts	ML thresholds, signal definitions
Prediction Engine	lib/engines/nested-ml-engine.ts	Pure prediction calculations
ML Service	lib/services/ai-prediction-service.ts	BigQuery ML orchestration
ML Hook	lib/hooks/useMLPredictions.ts	React integration
Signals Component	components/signals/NestedSignals.tsx	UI display

8.2 Feature Importance (from ML Model)

Feature	Importance	Layer
fivemin_price_up_pct	0.0665	5-Min (Most Predictive)
fivemin_ema_pct	0.0373	5-Min
avg_5min_score	0.0355	5-Min
fivemin_macd_pct	0.0275	5-Min
max_5min_score	0.0134	5-Min
daily_score	0.0030	Daily

9. API Integration Architecture

9.1 Data Sources (5 APIs)

API	Purpose	Rate Limit	Daily Quota
TwelveData ($229)	OHLCV + Indicators	800/min	2M records
Kraken	Buy/Sell Volume	60/min	Unlimited
FRED	Economic Indicators	100/day	100 calls
CoinMarketCap	Crypto Rankings	333/day	10K credits
Finnhub	Analyst Ratings	60/min	60 calls

9.2 Backend API Endpoints

Endpoint	Method	Purpose
/api/market-data/:symbol	GET	OHLCV data with indicators
/api/ai/trading-signals	GET	AI-generated signals
/api/ai/rise-cycle-candidates	GET	EMA crossover detection
/api/ai/ml-predictions	GET	Growth score predictions
/api/ai/nested-signals	GET	Multi-timeframe signals
/api/ai/nested-summary	GET	Nested model summary
/api/ai/text-to-sql	POST	Natural language queries
/api/data/download	GET	Data export endpoint
/api/admin/schedulers	GET	Scheduler status
/api/admin/tables	GET	BigQuery table info

10. Quick Reference Card

```
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■                      REFACTORING QUICK REFERENCE                      ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■                                                                      ■
■  CONFIGURATION (SSOT)                                                ■
■  ■■■■■■■■■■■■■■■■■■■■■                                               ■
■  Master Config:    @/lib/config/trading-config.ts                    ■
■  Scoring Rules:    @/lib/config/scoring-config.ts                    ■
■  API Config:       @/lib/config/api-config.ts                        ■
■  UI Config:        @/lib/config/ui-config.ts                         ■
■                                                                      ■
■  CALCULATION ENGINES                                                 ■
■  ■■■■■■■■■■■■■■■■■■■■■                                               ■
■  Indicators:       @/lib/engines/indicator-engine.ts                 ■
■  Signals:          @/lib/engines/signal-engine.ts                    ■
■  Growth Score:     @/lib/engines/growth-score-engine.ts              ■
■  ML Predictions:   @/lib/engines/nested-ml-engine.ts                 ■
■                                                                      ■
■  SERVICES                                                            ■
■  ■■■■■■■■■■■■■■■■■■■■■                                               ■
■  Market Data:      @/lib/services/market-data-service.ts             ■
■  Trading Signals:  @/lib/services/trading-signal-service.ts          ■
■  AI Predictions:   @/lib/services/ai-prediction-service.ts           ■
■  NLP Queries:      @/lib/services/nlp-query-service.ts               ■
■                                                                      ■
■  COMPONENTS                                                          ■
■  ■■■■■■■■■■■■■■■■■■■■■                                               ■
■  Dashboard:        @/components/dashboard/                           ■
■  Charts:           @/components/charts/                              ■
■  Signals:          @/components/signals/                             ■
■  Admin:            @/components/admin/                               ■
■  Shared:           @/components/shared/                              ■
■                                                                      ■
■  KEY METRICS                                                         ■
■  ■■■■■■■■■■■■■■■■■■■■■                                               ■
■  ML Accuracy:      66.2% UP | 70.6% DOWN | 68.4% Overall             ■
■  ROC AUC:          0.777                                             ■
■  API Budget:       $229/month (TwelveData)                           ■
■  Components:       42 -> 29 (after cleanup)                          ■
■  Lines Removed:    5,373 (redundant code)                            ■
■                                                                      ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
```

11. Conclusion & Next Steps

This refactoring plan combines the best practices from Saleem Ahmad's Economic Intelligence Platform clean architecture with the specific requirements of AIAlgoTradeHits.com trading platform. The transformation will result in a maintainable, testable, and scalable fintech application.

11.1 Immediate Actions

1. Create lib/config/ directory and migrate hardcoded values
2. Create lib/engines/ with pure calculation functions
3. Configure TypeScript and create type definitions
4. Delete 13 redundant components (5,373 lines)
5. Reorganize components into feature-based folders
6. Update import statements to use @/ absolute paths
7. Add comprehensive tests for engine functions

11.2 Success Criteria

Metric	Current	Target
TypeScript Coverage	0%	100%
Config Centralization	Scattered	SSOT
Component Organization	Flat (42)	Feature-based (29)
Engine Test Coverage	0%	80%+
Redundant Code	5,373 lines	0 lines
Build Time	~30s	<15s

AIAlgoTradeHits.com Architecture Refactoring Masterplan
Generated: January 12, 2026 at 20:31
Reference: EI Platform Architecture v2.0 + masterquery.md v4.0