# AI-POWERED STOCK AND ETF SIGNAL GENERATION PLATFORM

Problem Statement

# The Challenge We Face

Investors and analysts face significant barriers in efficiently analyzing market data

### Time-Consuming Analysis

Manual analysis of large volumes of stock and ETF data is slow and inefficient

### Error-Prone Decisions

Human error in interpreting complex market indicators leads to poor trading decisions

### Complex Metrics

Users struggle to interpret technical metrics and backtesting results effectively

### Lack of Real-Time Response

No automated system to validate strategies and deliver timely trading alerts

### Strategy Validation Gap

Difficulty in backtesting and validating trading strategies before actual deployment

### No intelligent Insights

Absence of AI-driven explanations to help users understand why specific signals are generated

End-to-End Solution

# Platform Architecture

Modular Python-based platform using FastAPI backend for scalable communication between modules

**Data Ingestion**
yfinance API
60+ Indian stock tickers

**Data Pipeline**
Medallion Architecture
Bronze >> Silver >>Gold

**Storage**
Supabase PostgreSQL
Ticker & feature storage

**ML Models**
RF,XGBoost, LSTM
Buy/Sell/Hold Signals

**Backtesting**
VectorBT
5-year validation

**Gen AI**
LLM Explanations
Signal insights

**Alerts**
Gmail/SMTP
Confidence-based

**Dashboard**
Streamlit
Visual analytics

**User Authentication**
Streamlit-based login

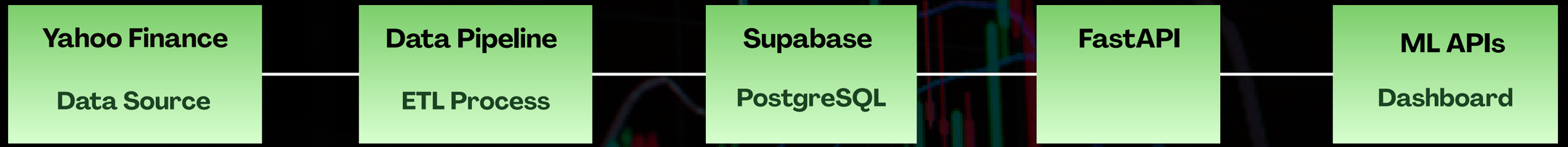
**Strategy Validation**
Backtesting confidence

**Real-Time Alerts**
Email notifications

Python FastAPI Streamlit Supabase VectorBT yfinance GenAI Gmail SMTP

# Data Ingestion & API Layer

Automated backend infrastructure for real-time stock market analysis with 60+ Indian stock tickers




## Data Pipeline Process

**Fetch Data**
OHLCV from yfinance

60+ Indian Tickers

**Clean and Validate**
Remove nulls and Duplicate

5 Years Historical

**Add Features**
RSI, SMA, MACD, etc.

6+ ML Features

## Tech Stack

yfinance

Python

Supabase

numpy

pandas

FastAPI

## Medallion Architecture

- Bronze Layer: Raw OHLCV data ingestion
- Silver Layer: Cleaned and validated data
- Gold Layer: Feature-enriched ML data

# Data Pipeline Flowchart

Data Ingestion  Storage  API Layer  ML Pipeline

1. Fetch OHLCV data
2. Clean & validate data
3. Feature engineering (RSI, SMA, Volatility, MACD etc.)
4. Sync processed data to Supabase
5. Update available data
6. Request training data via FastAPI
7. Query features from database
8. Return data to ML models
9. Deliver features for prediction
10. Train/Predict signals

## MLOps & Drift Detection Flow

**Fetch Baseline**
Last 30 days data

**Compare Window**
Last 7 days data

**Calculate Z-scores**
KS test p-value

**Alert & Retrain**
If drift detected

# Key Engineering Highlights

**Core technical decisions powering the data pipeline**

## Incremental Loading

Pipeline checks existing records and only fetches missing date ranges, optimizing data transfer

## Parallel Processing

Uses ThreadPool Executor to process multiple tickers concurrently, reducing execution time

## Dual Persistence

Data stored locally as Apache Parquet for speed and in Cloud (Supabase) for accessibility

## Drift Detection

Implements Kolmogorov-Smirnov (KS) tests to monitor feature distribution shifts

## Technical Indicators

RSI, SMA (20/50), Rolling Volatility, MACD, Volume MA for comprehensive ML features

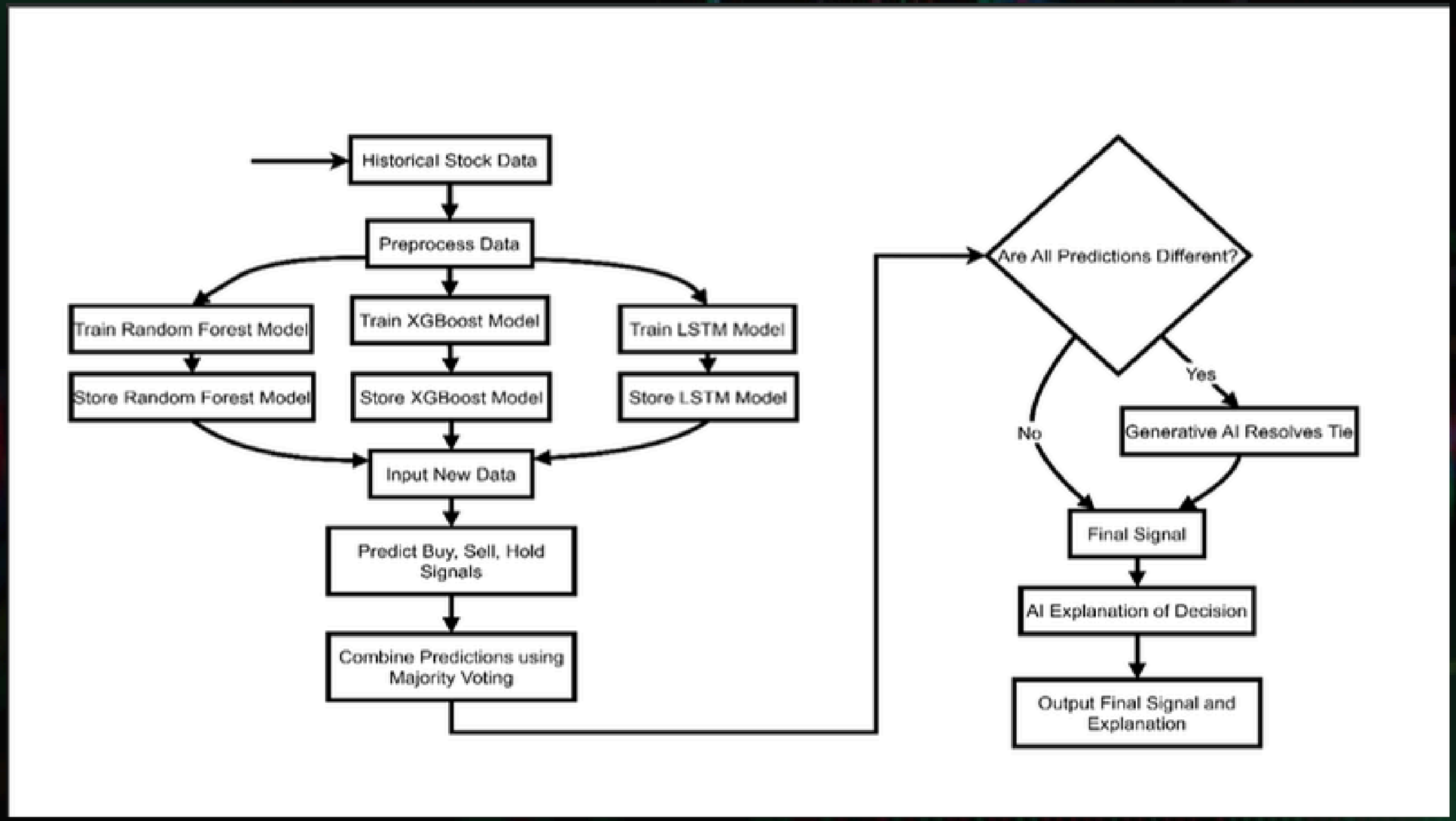
## Automated Alerts

If data drift detected (p-value < 0.05), system logs alert for model retraining

## PostgreSQL Data Model

**Stock features table with composite key (ticker, date) + B-Tree indexing**

# ML SIGNAL GENERATION ENGINE



# ML Models

| Market Condition / Topic | Random Forest | XGBoost | LSTM |
|---|---|---|---|
| **Sideways / Noisy Market** | ✅ Robust to noise via bagging | ❌ Overreacts to false momentum | ✅ Avoids trend confirmation without temporal strength |
| **Strong Trending Market** | ❌ Conservative averaging weakens trend signal | ✅ Captures nonlinear momentum | ✅ Confirms trend persistence over time |
| **Sudden Price Breakout** | ❌ Slow to react due to averaging | ✅ Detects sharp nonlinear moves | ✅ Captures early momentum sequence |
| **Long-Term Trend Continuation** | ❌ Lacks temporal memory | ❌ Feature-based, no sequence awareness | ✅ Learns long-range dependencies |
| **High Volatility / Whipsaw** | ✅ Variance reduction stabilizes signal | ✅ Adapts quickly to changing interactions | ❌ Sequence confusion due to abrupt reversals |
| **Regime Change (Trend → Range)** | ✅ Quickly adapts via re-sampling | ❌ Overfits previous regime patterns | ❌ Temporal memory tied to old regime |
| **Conflicting Technical Indicators** | ✅ Averages out indicator conflicts | ❌ Sensitive to misleading feature importance | ✅ Confirms using price-action sequence |
| **Low Liquidity / Sparse Data** | ❌ Performance degrades with sparse splits | ✅ Handles imbalance with boosting | ✅ Uses temporal continuity instead of raw density |

| Model Architecture | Parameters & Configuration |
|---|---|
| **XGBoost** | Depth: 6, LR: 0.05, 300 Estimators |
| **Random Forest** | Depth: 5, 100 Estimators |
| **Bidirectional LSTM** | 10-day lookback, Dropout layers |

### Hybrid Reliability Strategy

Eliminates **Single Point of Failure** by balancing XGBoost speed with Random Forest stability.

### Explainable AI (XAI)

Bridges the **Contextual Gap**, allowing users to understand the 'Why' behind every numeric score.

### Automated Data Integrity

Handles missing data points common in volatile financial feeds to ensure model input integrity.

# GenAI: Translating Quantitative Logic

**Top Evidence Headlines extracted from sources for verification**

**Clear Sentiment Categories providing instant visibility into market posture**

**Detailed Reasoning Narratives that explain the logic behind every signal**

**1.ML Score Input**

Meric scores and headlines are fed into the LLM synthesis engine

**2.LLM Processing**

Mistral and llama3 categorize the data into high confidence market vibes

**3.StrategOutput**

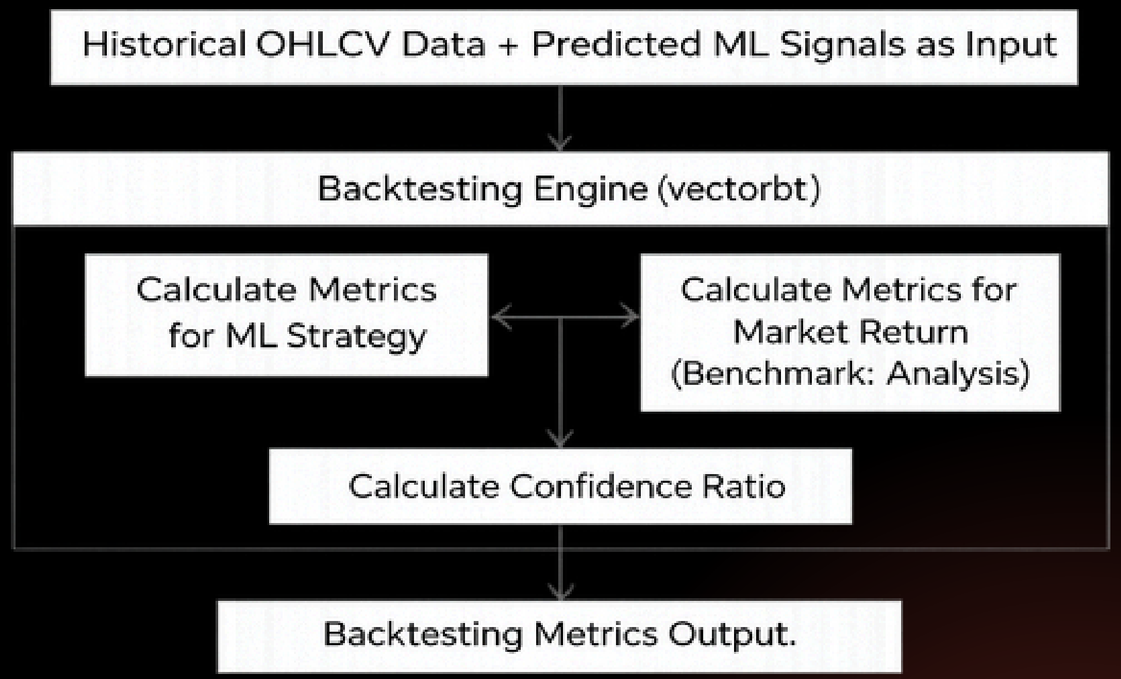
Plain English reasoning with an evidence based auidit trail

# What is **Backtesting?**

- Applying a trading strategy to historical market data to see how it would have performed
- It is the validation and evaluation layer of the  platform
- Its objective is to test ML-generated trading signals on historical market data and determine whether those signals are profitable, stable, and risk-aware

“High ML accuracy does not guarantee profits. Backtesting ensures strategies are financially viable and risk-aware before deployment“

# Flow Chart




- Historical OHLCV data along with predicted ML signals are given as input
- The engine runs the trading strategies against the historical timeline using Vectorbt
- This step simulates account balance, transaction costs, and trade execution as if they were happening in real-time
- Metrics are calculated for both ML strategy and Market return
- Confidence ratio is calculated using metrics and sent to dashboard
- This provides validation to the predicted signals and enhances user trust

# Outputs

The system generates the following key outputs

- **Total Return**
- **CAGR**
- **Volatility**
- **Sharpe Ratio**
- **Max Drawdown**
- **Win Rate**
- **Profit Factor**
- **Market Benchmark Metrics (Buy & Hold)**
- **Confidence Score**
- **Equity Curve & Trade Statistics**

“Only validated, confidence-backed, and risk-aware signals are shown to users and used for alerts”

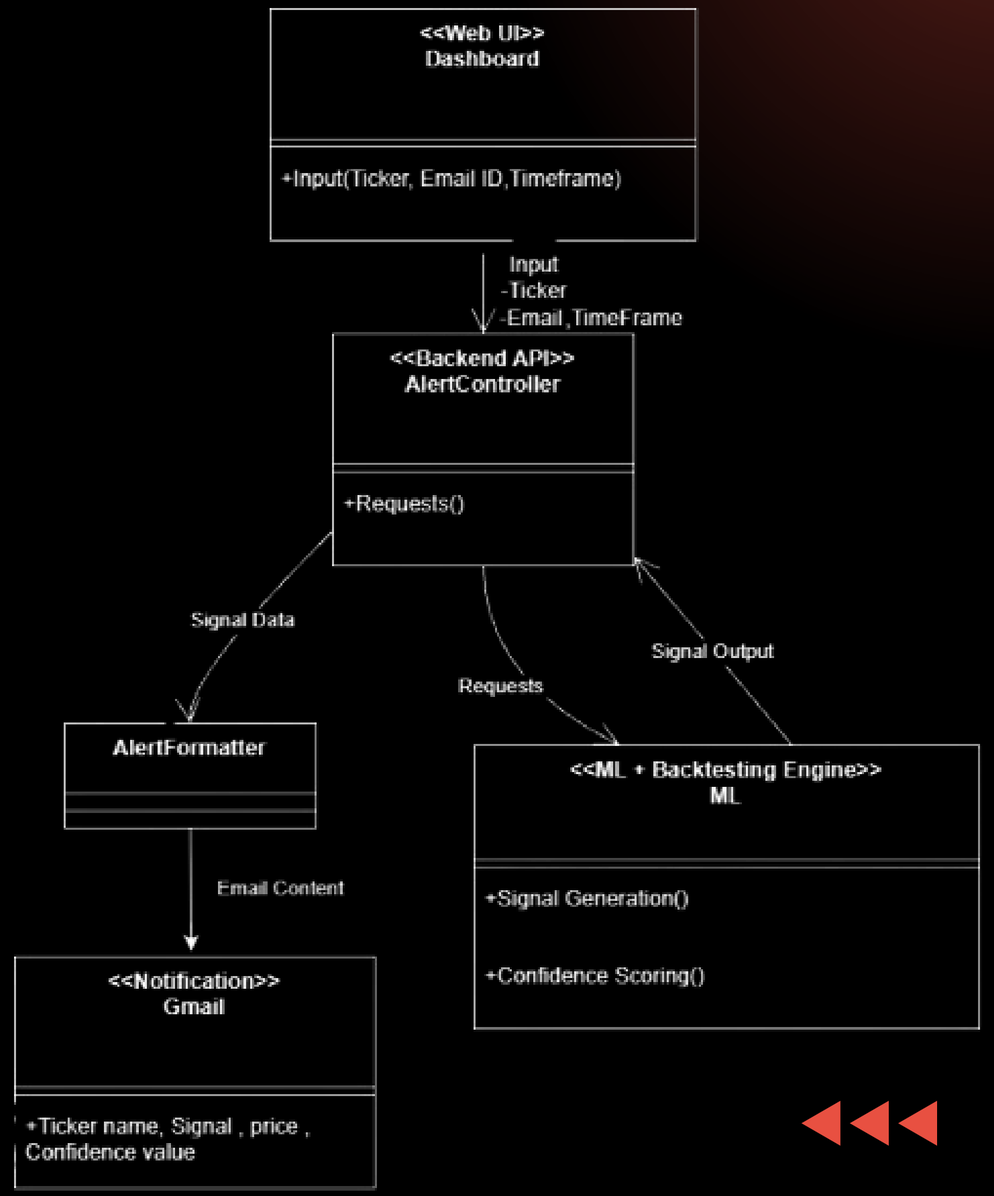
# Real-Time Alerts System

- The Alerts Service is responsible for notifying users when a strong and reliable BUY or SELL signal is detected for a stock
- Since raw ML signals can be noisy and risky, this service ensures that alerts are sent only when the ML signal is validated using historical performance and meets a confidence threshold
- User configures alert preferences via dashboard by providing ticker, email, timeframe
- Notifications are delivered via email for the selected ticker in given timeframe

# Flowchart

- The user provides ticker, email ID and timeframe and registers for alerts via dashboard
- The alert controller automatically runs the alert pipeline during the specified timeframe
- The pipeline triggers ML and backtesting API to predict the signal and validate it
- Upon validation the following Alert Content is sent via email

  - Ticker symbol
  - Signal type
  - Current price
  - Confidence score

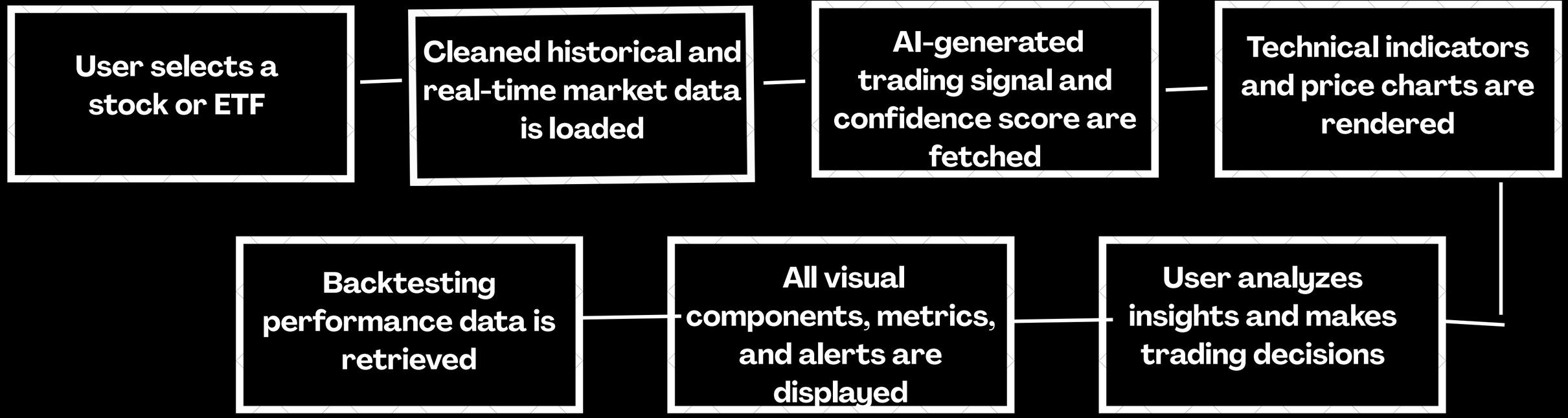


# What is Dashboard & Visualization Hub?

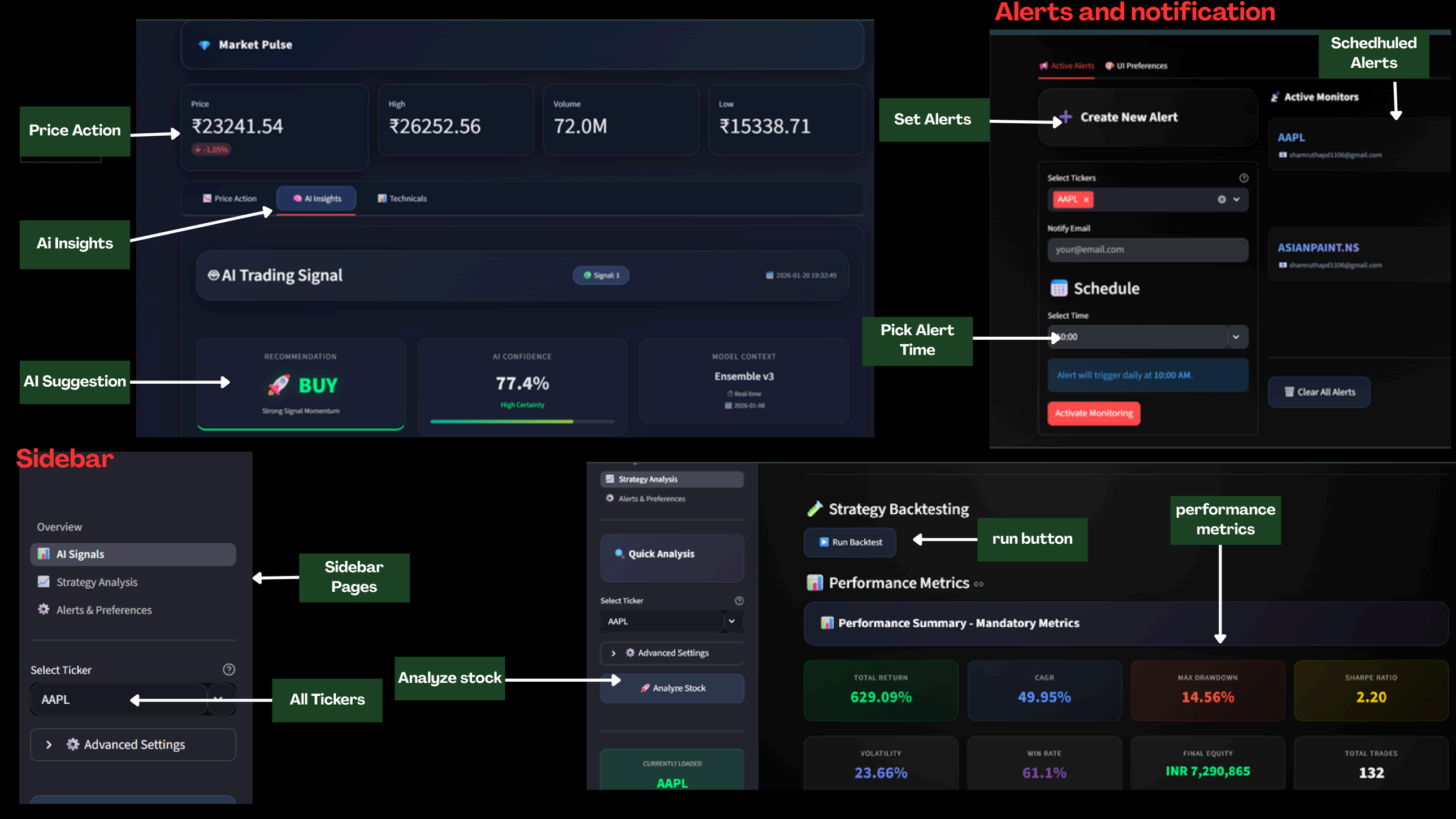
- It is the user-friendly interface of the AI-Powered Stock & ETF Signal Generation Platform that acts as the final presentation layer, converting complex ML, data engineering, and backtesting outputs into clear and actionable visual insights.
- The dashboard ensures transparency and informed decision-making by visually presenting BUY/SELL/HOLD signals, confidence scores, technical indicators, and performance metrics in a professional trading interface.

**“Even accurate ML models are ineffective if users cannot interpret or trust their outputs. The dashboard ensures transparency, clarity, and informed decision-making.”**

# Flow Chart




- Shows end-to-end user interaction with the dashboard
- Data is loaded automatically from backend APIs
- AI signals and confidence are visualized, not calculated here
- Technical indicators support signal understanding


Alerts and notification
Market Pulse
Price
₹23241.54
-1.05%
High
₹26252.56
Volume
72.0M
Low
₹15338.71
Price Action
Price Action
AI Insights
Technicals
Ai Insights
AI Trading Signal
Signal: 1
2026-01-20 19:32:49
AI Suggestion
RECOMMENDATION
BUY
Strong Signal Momentum
AI CONFIDENCE
77.4%
High Certainty
MODEL CONTEXT
Ensemble v3
Real-time
2026-01-08
Set Alerts
Active Alerts
UI Preferences
Create New Alert
Schedhuled Alerts
Active Monitors
AAPL
shamruthapd1106@gmail.com
Select Tickers
AAPL
Notify Email
your@email.com
ASIANPAINT.NS
shamruthapd1106@gmail.com
Schedule
Pick Alert Time
Select Time
10:00
Alert will trigger daily at 10:00 AM.
Activate Monitoring
Clear All Alerts
Sidebar
Overview
AI Signals
Strategy Analysis
Alerts & Preferences
Sidebar Pages
Select Ticker
AAPL
All Tickers
Advanced Settings
Analyze stock
Strategy Analysis
Alerts & Preferences
Quick Analysis
Select Ticker
AAPL
Advanced Settings
Analyze Stock
CURRENTLY LOADED
AAPL
Strategy Backtesting
Run Backtest
run button
performance metrics
Performance Metrics
Performance Summary - Mandatory Metrics
TOTAL RETURN
629.09%
CAGR
49.95%
MAX DRAWDOWN
14.56%
SHARPE RATIO
2.20
VOLATILITY
23.66%
WIN RATE
61.1%
FINAL EQUITY
INR 7,290,865
TOTAL TRADES
132

# Dashboard System Output & Tools Used

## Dashboard System Output & Data Presentation

- Selected ticker symbol
- AI-generated signal (BUY / SELL / HOLD)
- Current market price
- Confidence score
- Risk & volatility indicators
- Backtesting performance summary
- Data is fetched via APIs
- Data can be exported for reporting

## Tools Used

- Streamlit – UI development
- Plotly – interactive charts
- Python – backend logic
- Pandas & NumPy – data handling
- Figma – UI design planning