NEXYPHER

AI-Powered Crypto Trading Platform

Detailed Project Report

Project:	NEXYPHER v2
Type:	AI Crypto Trading & Intelligence Platform
Tech Stack:	Python / FastAPI / Gemini AI / Supabase / Blockchain
Deployment:	Vercel (Frontend) + Render (Backend)
Live URL:	https://NEXYPHER.vercel.app
Date:	February 20, 2026

1. Executive Summary

NEXYPHER is a full-stack, AI-powered cryptocurrency trading and intelligence platform that combines real-time market data, Machine Learning, Gemini AI-driven analysis, walk-forward backtesting, and an always-on autonomous trading bot - all accessible through a premium web dashboard.

Key Metrics

Metric	Value
Total Codebase	~500,000+ lines across 80+ files
Backend	Python 3.11+ / FastAPI / Uvicorn
Frontend	Single-page HTML/CSS/JS (428KB)
AI Engine	Google Gemini 2.0 Flash
Database	Supabase (PostgreSQL)
Blockchain	Base L2 / Ethereum (on-chain TX recording)
Deployment	Vercel (frontend) + Render (backend)
Live URL	https://NEXYPHER.vercel.app

2. System Architecture

The platform follows a modular monolithic architecture with clear separation between data collection, AI analysis, trading execution, and user-facing API layers.

Architecture Components

Frontend Layer

Web Dashboard (index.html - 428KB) + AlgoTrader page (algo.html - 178KB). Premium dark mode UI with glassmorphism, real-time token feed, trading dashboard.

API Layer

FastAPI backend (web_app.py - 3,049 lines, 170+ REST endpoints). Handles auth, market data, trading, AI analysis, wallet management.

AI Engine

Gemini 2.0 Flash client, orchestrator, learning loop, confidence scorer, NLG engine, prompt templates. Self-improving via prediction tracking.

Trading Engine

Always-on auto-trader bot (2-min cycles), all-in strategy, risk management (stop-loss, take-profit, 4h auto-exit).

Data Collectors

CoinGecko (top coins + trending), DexScreener (DEX pairs), News (sentiment analysis), Social (Reddit/Twitter), Technical (RSI/MACD/BB).

Backtest Engine

Walk-forward backtesting (78KB). Tests LONG/SHORT/RANGE strategies on 180 days of historical data with in-sample/out-of-sample split.

Infrastructure

Supabase (PostgreSQL DB), Base L2/Ethereum blockchain, SMTP email alerts, in-memory caching.

3. Module Breakdown

3.1 Web Application (web_app.py - 3,049 lines)

The monolithic FastAPI server serving 170+ REST API endpoints:

Category	Endpoints	Description
Auth	/api/auth/*	JWT auth, register, login, bcrypt
Market	/api/market/*	Live prices, trending, search
Trading	/api/trader/*	Buy, sell, positions, P&L
AI	/api/ai/*	Token analysis, recommendations
Wallet	/api/wallet/*	Balance, deposit, withdraw
Blockchain	/api/blockchain/*	On-chain TX verification
Backtest	/api/backtest/*	Walk-forward results
Feed	/api/feed/*	Real-time token feed

3.2 Trading Engine (trading_engine.py - 1,135 lines)

Core autonomous trading system with always-on auto-trader bot, all-in single trade strategy, and comprehensive risk management.

Auto-Trader Bot Features:

- 2-minute trading cycles - continuously scans markets for opportunities
- All-in strategy - picks single best coin, invests entire balance
- Aggressive risk profile - bypasses backtest gate for faster execution
- 4-hour auto-exit - positions automatically sold after max holding period
- Stablecoin filter - excludes USDT, USDC, DAI, USD1 (they don't move)
- DEX token filter - only buys CoinGecko-listed coins with reliable prices
- Stop-loss at -6% and Take-profit at +20%
- Daily loss limit of 10% to prevent catastrophic losses

Coin Scoring Algorithm (0-100 points)

Factor	Points	Condition
24h Momentum	+25	3-15% price gain in last 24h
Volume/MCap Ratio	+20	Ratio > 0.3 (high activity)
Trending Status	+15	Appears on CoinGecko trending
Large Cap	+10	Market cap > $10 billion
ATH Distance	+10	30-70% below all-time high
Backtest Verified	+10	Walk-forward test passed
Heavy Decline	-10	24h change below -10%

3.3 AI Engine (src/ai_engine/ - 10 files)

The AI engine uses Google Gemini 2.0 Flash for real-time token analysis, market summaries, and trading recommendations. It features a self-improving learning loop that tracks prediction accuracy over time.

File	Size	Purpose
orchestrator.py	47KB	Central coordinator - merges all data sources
learning_loop.py	28KB	Self-improving AI - tracks & evaluates predictions
confidence_scorer.py	21KB	Multi-factor confidence scoring (0-10)
prompt_templates.py	18KB	Gemini prompt engineering templates
nlg_engine.py	17KB	Natural language generation for reports
models.py	16KB	Pydantic data models for AI pipeline
gemini_client.py	9KB	Google Gemini 2.0 Flash API client
conflict_detector.py	7KB	Detects conflicting signals between sources
gpt_client.py	7KB	OpenAI GPT fallback client

Learning Loop (Self-Improving AI):

- RECORD - Logs every prediction with token, verdict, confidence, and price
- EVALUATE - After holding period, checks actual price movement vs prediction
- LEARN - Adjusts confidence thresholds and strategy weights based on outcomes
- STORE - Persists in SQLite DB (nexypher_learning.db) for cross-session memory

3.4 Backtest Engine (src/backtest_engine.py - 78KB)

The most sophisticated module - a full walk-forward backtesting system that tests trading strategies against 180 days of historical CoinGecko OHLCV data.

- 180-day historical data from CoinGecko OHLCV API
- 3 strategies tested automatically: LONG, SHORT, RANGE
- Walk-forward analysis: In-sample (90 days) + Out-of-sample (89 days) split
- Strategy cascade: Tests trend-appropriate strategy first, then alternatives
- Threshold gates: Win rate, total return, Sharpe ratio, max drawdown
- Token tier detection: Major, mid, small cap with different thresholds
- Trend detection: Uptrend, downtrend, sideways classification

3.5 Data Collectors (src/data_collectors/ - 7 files)

Collector	Source	What It Provides
CoinGecko	CoinGecko API	Top 30 coins, trending, OHLCV, simple prices
DexScreener	DexScreener API	DEX pair data, liquidity, buy/sell ratios
News	NewsAPI	Keyword sentiment analysis (bullish/bearish words)
Social	Reddit/Twitter	Social sentiment, trending mentions
Technical	Computed	RSI, MACD, Bollinger Bands, EMA indicators
Pipeline	Aggregator	Merges all collectors into unified data stream

3.6 Authentication (auth.py - 694 lines)

- User registration with email, username, bcrypt password hashing
- JWT token authentication (HS256) with configurable expiry
- Bank account management - IFSC code validation, account verification
- Wallet balance - deposit, withdraw with full transaction logging
- User preferences - risk profile, AI sensitivity, notification settings
- Rs. 10,000 signup bonus credited automatically on registration

3.7 Blockchain Service (blockchain_service.py - 378 lines)

Every trade in NEXYPHER is recorded on the blockchain for transparency and verification.

- Chains supported: Base L2 (primary), Ethereum (fallback)
- Smart Contract: TransactionRegistry - records SHA-256 hashes on-chain
- Transaction types: BUY, SELL, DEPOSIT, WITHDRAW, SIGNUP_BONUS
- Async recording - non-blocking background thread for performance
- Explorer links - BaseScan / Etherscan verification URLs for every trade

3.8 Frontend (web/index.html - 428KB)

- Single-page application with premium dark mode design
- Glassmorphism effects and smooth micro-animations
- Real-time token feed with sparkline price charts
- Trading dashboard - open positions, P&L tracking, trade history
- AI recommendations panel with confidence scores and analysis
- Wallet management - balance display, deposit/withdraw, bank accounts
- Token detail modal with backtest results, AI analysis, and trade button
- Responsive design for desktop and mobile

4. Technology Stack

Layer	Technology	Purpose
Backend	FastAPI + Uvicorn	Async REST API server
AI/LLM	Gemini 2.0 Flash	Token analysis, market summaries
Database	Supabase (PostgreSQL)	Users, trades, positions, settings
Local DB	SQLite	AI learning loop, parameter optimization
Blockchain	Web3.py + Base L2	On-chain transaction verification
Auth	JWT (PyJWT) + bcrypt	Token-based authentication
Email	SMTP (Gmail)	Security alerts, login notifications
Caching	In-memory (TTL)	API response and backtest caching
Frontend	HTML/CSS/JS	Premium SPA dashboard
Hosting	Vercel + Render	Frontend + Backend deployment

5. Database Schema (Supabase)

Table	Purpose
users	User accounts (email, username, password hash, created_at)
wallet_balance	INR/USD balance per user
wallet_transactions	Deposit, withdraw, and trade transaction history
trade_positions	Open/closed positions with P&L, stop-loss, take-profit
trade_orders	Individual buy/sell orders with AI reasoning
trade_settings	Per-user trading config (risk level, limits)
trade_stats	Aggregate stats (total PnL, win rate, best/worst)
user_preferences	Risk profile, AI sensitivity, notification prefs
wallets	Linked crypto wallet addresses (ETH, SOL, etc.)
watchlist	User's watched coin IDs
bank_accounts	Verified Indian bank accounts (IFSC, account number)
trade_log	Detailed event log for debugging and auditing

6. Trading Data Flow

The auto-trader executes the following cycle every 2 minutes:

- 1. FETCH - Retrieve top 30 coins + trending from CoinGecko API
- 2. SCORE - Rate each coin (0-100) based on momentum, volume, cap, trending status
- 3. FILTER - Remove stablecoins, DEX tokens, coins with price <= $0.001
- 4. ANALYZE - Send top 5 coins to Gemini AI for detailed analysis
- 5. BACKTEST - Run walk-forward backtest (180 days) for verification
- 6. SELECT - Pick the #1 highest-scoring coin
- 7. EXECUTE - All-in buy order with entire balance
- 8. RECORD - Store position in Supabase + record TX hash on blockchain
- 9. MONITOR - Check stop-loss (-6%), take-profit (+20%), 4h auto-exit
- 10. EXIT - Auto-sell when conditions met, then restart cycle

7. Key Files Summary

File	Lines	Size	Role
web_app.py	3,049	127KB	Main FastAPI server, all API endpoints
web/index.html	-	428KB	Frontend SPA dashboard
trading_engine.py	1,135	52KB	Auto-trader bot, trade execution
backtest_engine.py	-	78KB	Walk-forward backtesting system
orchestrator.py	-	47KB	AI analysis coordinator
auth.py	694	28KB	Auth, users, bank accounts, wallet
learning_loop.py	-	28KB	Self-improving AI predictions
smtp_service.py	-	28KB	Email notification service
technical_analyzer.py	-	27KB	RSI, MACD, Bollinger Bands
web/algo.html	-	178KB	AlgoTrader page
blockchain_service.py	378	15KB	On-chain TX recording (Base L2)

8. Security Features

- bcrypt password hashing with unique salt per user
- JWT token authentication (HS256 algorithm) with configurable expiry
- CORS middleware with configurable allowed origins
- Rate limiting via custom middleware to prevent abuse
- IFSC code and account number validation for bank accounts
- Email-based login alerts with IP address, location, and device info
- Blockchain verification - every trade hash recorded on Base L2 chain
- Environment variables for all secrets (API keys never in code)

9. Deployment

Local Development

Command: python run_web.py

Server starts at: http://localhost:8000

Production Deployment

- Frontend: Deployed on Vercel -> https://NEXYPHER.vercel.app
- Backend: Deployed on Render with render.yaml configuration
- Database: Hosted on Supabase (managed PostgreSQL)
- Environment Variables: GEMINI_API_KEY, SUPABASE_URL, SUPABASE_KEY, SECRET_KEY

10. Current Auto-Trader Configuration

Setting	Value
User Account	ID: 1 (primary user account)
Balance	Rs. 1,00,00,000 (Rs. 1 Crore / $10M)
Risk Level	Aggressive (backtest gate bypassed)
Trading Interval	2 minutes between cycles
Strategy	All-in on single best coin
Max Hold Time	4 hours (auto-exit)
Stop-Loss	-6% from entry price
Take-Profit	+20% from entry price
Daily Loss Limit	10% of total balance
Filters	No DEX tokens, no stablecoins, price > $0.001