The Architectural Blueprint for an AI-Native Indian Derivative Trading Platform: A Comprehensive Research Report

1. Executive Summary: The Convergence of Vibe Coding and Quantitative Finance in 2026

The financial technology landscape of 2026 is defined by a radical convergence of two potent technological vectors: the maturation of "vibe coding" as a primary software development paradigm and the deployment of reasoning-heavy Large Language Models (LLMs) such as Anthropic's Claude Opus 4.6 and Google's Gemini 3 Pro. This report provides an exhaustive architectural analysis and implementation roadmap for constructing a professional-grade trading platform tailored specifically for the Indian stock market (NSE/BSE). The objective is to democratize institutional-grade technical analysis by leveraging these advanced tools to create a system that not only processes real-time market data but actively interprets it to generate actionable buy/sell signals for high-frequency derivatives trading.

The premise of this platform is to utilize the "vibe coding" methodology—where human architects guide autonomous AI agents through natural language to build complex software ecosystems—to rapidly deploy a system that would traditionally require a team of specialized engineers. The proposed platform integrates low-latency WebSocket data pipelines, orchestrates a multi-agent AI ensemble for real-time technical analysis, and enforces rigorous deterministic guardrails to mitigate the risk of model hallucination. By synthesizing live Open Interest (OI) dynamics, Option Greeks, and standard technical indicators (RSI, VWAP, Bollinger Bands), the system aims to provide retail traders with the cognitive edge previously reserved for proprietary trading desks.

Key findings indicates that while vibe coding drastically accelerates the development lifecycle—potentially reducing the time-to-market for a complex financial dashboard from months to days—the reliability of the financial advice generated requires a sophisticated hybrid architecture. This architecture must decouple the "creative" reasoning of LLMs from the "deterministic" validation of mathematical engines. This report details the specific prompting strategies, technical stacks, and data engineering required to achieve this, concluding with a "Master System Prompt" designed to instantiate the core platform infrastructure.

2. The Vibe Coding Paradigm: Redefining Financial Software Development

2.1 The Evolution of Vibe Coding: From Viral Tweet to Industry Standard

"Vibe coding," a term popularized around 2025, has transcended its origins as a colloquial description of prompt-driven programming to become a formalized, AI-native development methodology by 2026. Fundamentally, vibe coding represents a shift in the developer's role from syntax writer to system architect and product manager. In this paradigm, the developer communicates intent and high-level logic ("the vibes") to an AI agent, which then handles the implementation details, boilerplate generation, dependency management, and error resolution.
This shift is particularly transformative for financial engineering. Historically, building a trading platform required deep, siloed expertise: frontend developers for React dashboards, backend engineers for low-latency Python services, and quantitative analysts for the mathematical models. Vibe coding collapses these silos. A single individual with deep domain knowledge of the Indian derivative markets can now orchestrate the creation of a full-stack platform. The focus shifts from "How do I implement a WebSocket listener in Python?" to "How should the system react when Nifty crosses the VWAP with high volume?".
The efficacy of vibe coding in 2026 is driven by the capabilities of the underlying models. Tools like Replit Agent 3 and Cursor do not merely autocomplete code; they possess "agentic" capabilities. They can plan multi-step refactors, understand the context of an entire codebase, and debug their own errors by reading stack traces and documentation. This capability is crucial for financial applications where the integration of disparate libraries—such as TA-Lib for indicators, pydantic for data validation, and websockets for streaming—must be seamless and error-free.

2.2 The Tooling Landscape: A Strategic Bifurcation

For a project of this complexity, selecting the right development environment is a strategic decision. The analysis of the 2026 landscape reveals a bifurcation between cloud-native "App Generators" and local "AI-Augmented IDEs."
Replit Agent 3 (The Cloud Orchestrator): Replit has positioned itself as the premier "zero-setup" environment. Its Agent 3 is capable of taking a high-level prompt—"Build a Nifty option chain analyzer with live Greeks"—and generating a deployed application with a database, authentication, and hosting fully configured.

- **Strategic Advantage:** Replit excels at the "0 to 1" phase. It removes the friction of environment configuration, dependency hell, and deployment pipelines. For a trader acting as a developer, this allows for rapid prototyping of the dashboard UI and basic data connections.
- **Limitation:** While powerful, cloud-based agents can sometimes struggle with the granular, low-level optimization required for high-frequency data processing. The latency introduced by a cloud IDE can be a bottleneck when fine-tuning WebSocket handlers that process thousands of ticks per second.

Cursor / Windsurf (The Precision Engineers): Cursor, an AI-native fork of VS Code, represents the "local" approach. It integrates deeply with the developer's local file system and uses the entire codebase as context for its reasoning.

- **Strategic Advantage:** Cursor is superior for the "1 to 100" phase. Once the prototype exists, Cursor allows for deep architectural refactoring. It is particularly adept at "vibe coding" complex algorithmic logic, such as ensuring that the Black-Scholes calculation in Python is vectorized for performance using numpy rather than iterated in a slow loop.
- **Synergy:** The optimal workflow for building a professional trading platform involves a hybrid approach: utilizing Replit to "vibe code" the initial full-stack architecture and frontend dashboard, and then pulling the codebase locally to use Cursor for refining the

technical analysis algorithms and ensuring strict type safety.

2.3 The "Plan-Prompt-Polish" Loop in Financial Context

Successful vibe coding follows a disciplined iterative process, often described as the "Plan → Prompt → Polish" loop. In the context of building a trading system for the Indian market, this loop takes on a specific character:

1. **Plan (The Financial Logic):** The developer must articulate the trading strategy in plain but precise English. For example: "I want to identify a 'Short Covering' rally. This happens when the Nifty price rises above its 15-minute VWAP, while the Open Interest (OI) for the ATM Call option decreases significantly."
2. **Prompt (The AI Implementation):** This logic is fed to the AI agent. The prompt must be context-rich: "Implement a Python function using pandas that takes a DataFrame of option ticks. It should detect when the price column crosses above vwap and call_oi drops by more than 5% in a 5-minute window. Use TA-Lib for the VWAP calculation.".
3. **Polish (The Risk Management):** The generated code is reviewed not just for syntax, but for financial validity. Does the code handle missing data ticks? Does it crash if the API returns a null value during a volatility halt? This step involves using the AI to write unit tests that simulate extreme market conditions (e.g., a 10% market crash) to ensure robustness.

This methodology transforms the development process from a coding exercise into a financial engineering exercise, allowing the trader to embed their market intuition directly into the software architecture.

3. The Cognitive Engine: AI Model Architecture for 2026

The "brain" of the trading platform is not a single AI model but an ensemble of specialized models, each selected for its specific strengths in the 2026 AI landscape. The requirement is for a system that can "search and tell the user" what to trade, implying a need for both real-time information retrieval and deep analytical reasoning.

3.1 Claude Opus 4.6: The Analytical Sovereign

Anthropic’s Claude Opus 4.6 is identified as the "Technical Leader" in the 2026 model hierarchy, particularly for tasks requiring deep reasoning and long-context retention.

- **Capabilities:** With a 1 million token context window (in beta) and a top-tier SWE-bench score of 80.8%, Claude 4.6 excels at "agentic" workflows. It can maintain a coherent "chain of thought" over long trading sessions, remembering market context from hours ago (e.g., "The market opened gap-up but has been consolidating").
- **Role in Platform:** Claude 4.6 serves as the **Strategy Validator**. It acts as the senior analyst. It ingests the raw signals from the technical algorithms (e.g., "RSI is 75") and synthesizes them with the broader market context. Its "extended thinking" capabilities allow it to weigh conflicting data points—such as a bullish price action conflicting with a bearish global macro sentiment—before issuing a final trade recommendation.

3.2 Gemini 3 Pro: The Real-Time Market Sensor

Google's Gemini 3 Pro distinguishes itself through its massive context window and, crucially, its native integration with real-time Google Search.

- **Capabilities:** As a "multimodal powerhouse," Gemini 3 Pro can process text, images (charts), and video streams. Its ability to access live information minimizes the "knowledge cutoff" latency, a fatal flaw for trading bots relying on older models.
- **Role in Platform:** Gemini 3 Pro functions as the **Market Sensor**. It is tasked with scanning live news feeds, regulatory announcements (SEBI circulars), and Twitter/X sentiment regarding Indian stocks. It answers the "Why?" behind a move. If Nifty suddenly spikes 100 points, Gemini can instantly correlate that with a breaking news story about a corporate tax cut or an RBI policy announcement, providing context that pure technical analysis misses.

3.3 OpenAI (GPT-5.2/o-series): The Structured Executioner

OpenAI's latest iterations (referred to as GPT-5.2 or o-series in 2026) remain the gold standard for structured data output and speed.

- **Capabilities:** These models have been fine-tuned to reliably output valid JSON schemas, even when dealing with complex, nested data structures. They are less prone to formatting errors than other large models.
- **Role in Platform:** The **Signal Formatter**. While Claude thinks and Gemini searches, OpenAI's model takes their unstructured outputs and formats them into the precise BUY / SELL signals, stop-loss levels, and target prices that the frontend dashboard requires. It ensures that the "vibe" of the analysis is translated into rigorous, machine-readable code that can trigger a UI alert or even execute a trade.

3.4 The "Mixture of Experts" (MoE) Architecture

The most robust architecture for this platform utilizes an ensemble approach, often referred to as a "Mixture of Experts" (MoE) in AI system design.

- **Agent A (Gemini):** "Context Agent" - Scans for news, macro events, and sentiment.
- **Agent B (Python Algo):** "Quant Agent" - Calculates hard technicals (RSI, VWAP, Greeks) deterministically.
- **Agent C (Claude):** "Decision Agent" - Synthesizes the news from Agent A and the data from Agent B to form a trading thesis.
- **Agent D (OpenAI):** "Interface Agent" - Formats the thesis into a user-friendly alert.

This structure mimics a professional trading desk, where junior analysts (Quant/Context Agents) feed information to a portfolio manager (Decision Agent), who then instructs a trader (Interface Agent) to execute.

4. Technical Architecture: Building the Indian Trading Engine

To deliver a "functional professional" experience, the platform must move beyond simple request-response chatbots. It requires a real-time, event-driven architecture capable of handling

the high-velocity data of the Indian derivatives market.

4.1 The Stack: Speed and Reliability

- **Frontend: Next.js 15 & Shadcn/UI**
  - **Next.js 15:** The latest version of Next.js introduces advanced streaming capabilities and React Server Components (RSC). This allows the platform to maintain a persistent connection to the backend without blocking the main thread, ensuring the UI remains responsive even during high-volatility events.
  - **Shadcn/UI:** For the visual layer, Shadcn/UI provides a professional, data-dense aesthetic reminiscent of a Bloomberg Terminal or Refinitiv Eikon. Its modular nature fits perfectly with the vibe coding workflow, allowing developers to "prompt" complex components like data tables and alert panels into existence.
- **Backend: Python (FastAPI)**
  - **FastAPI:** Python is the non-negotiable language of quantitative finance due to its rich ecosystem (pandas, numpy, scipy). FastAPI is chosen for its asynchronous capabilities, which are essential for handling WebSockets. It bridges the gap between Python's analytical power and the real-time needs of the modern web.
- **State Management: Redis**
  - **In-Memory Speed:** The Nifty index generates thousands of ticks per second. Storing this flow directly in a disk-based database like PostgreSQL would introduce unacceptable latency. Redis serves as the "hot" data layer, storing the live state of the option chain and allowing the AI agents to query the latest prices in sub-millisecond time.

4.2 Data Engineering: The Lifeblood of the System

The Indian market operates on a tick-by-tick basis. Latency is the differentiator between profit and loss.

- **WebSocket Protocol:** The platform must rely on WebSockets (WSS) for data ingestion. HTTP polling (requesting data every second) is obsolete for options trading where gamma spikes (rapid changes in option delta) occur in milliseconds.
- **API Providers (NSE/BSE):**
  - **TrueData / Global Datafeeds:** These are authorized data vendors that provide authorized, low-latency feeds for NSE/BSE. Crucially, they offer specific APIs for "Option Greeks" and "Option Chain" via WebSockets, reducing the computational load on the backend.
  - **Broker APIs (Dhan/Kite):** For a retail-focused platform, integrating directly with broker APIs like Dhan HQ or Zerodha Kite Connect is efficient. Dhan, for instance, provides free API access for execution and data, making it a cost-effective choice for independent traders.
- **The "Firehose" Strategy:** Market data arrives faster than a human can read. The backend architecture must implement a "throttling" or "sampling" mechanism for the UI (updating the screen 5 times a second is sufficient for human perception), while the internal AI agents continue to analyze every single tick for anomalies.

4.3 Real-Time Greeks and Option Chain Management

Trading options without Greeks (Delta, Gamma, Theta, Vega) is akin to driving blind.

- **Ingestion vs. Calculation:** While some premium feeds (like TrueData) provide pre-calculated Greeks, a professional platform must be robust enough to calculate them if the feed fails. The backend should include a dedicated Python microservice using libraries like py_vollib (a vectorised implementation of Black-Scholes) to calculate Greeks on the fly based on the spot price and implied volatility.
- **Live State Object:** The backend maintains a "Live State" object in Redis—a structured JSON representation of the entire option chain that updates continuously. This object is the "truth" source that the AI agents query to make decisions.

4.4 Advanced Visualization: TradingView Integration

- **Library:** The platform will integrate **TradingView Lightweight Charts**. This library is the industry standard for web-based financial charting due to its canvas-based rendering engine, which ensures high performance even with thousands of data points.
- **Vibe Coding Implementation:** The prompt to the AI agent would be specific: "Implement a TradingView Lightweight Chart component in React. Use a custom useEffect hook to subscribe to the WebSocket stream and update the candlestick series in real-time without triggering a full component re-render".

5. The "Brain" Logic: Technical Analysis for Indian Markets

To instruct the AI effectively, we must define the specific trading logic valid for the Indian Nifty and BankNifty markets. General trading knowledge is insufficient; the AI must understand the nuances of this specific liquidity pool.

5.1 The "Holy Trinity" of Indian Options Strategy

The platform's analysis engine will be grounded in three core indicators that have high predictive validity in the Indian context :

Indicator	Bullish Signal Logic	Bearish Signal Logic	Context & Nuance
VWAP (Volume Weighted Avg Price)	Price crosses *above* VWAP with significant volume.	Price crosses *below* VWAP with significant volume.	The institutional benchmark. In Indian intraday markets, institutions often buy at VWAP. Price holding above VWAP is the primary filter for any long trade.
Open Interest (OI) Dynamics	Call OI Unwinding (Short Covering) + Put OI Buildup.	Put OI Unwinding (Long Unwinding) + Call OI Buildup.	The most potent signal in Nifty/BankNifty. A rise in price accompanied by a drop in Call OI indicates sellers are trapped and

Indicator	Bullish Signal Logic	Bearish Signal Logic	Context & Nuance
			covering, often leading to explosive moves.
Put-Call Ratio (PCR)	PCR rising above 1.0 or bouncing from oversold (<0.7).	PCR falling below 1.0 or rejecting from overbought (>1.3).	Acts as a sentiment gauge. Extreme readings (e.g., 0.5 or 1.6) often signal a contrarian reversal opportunity.

5.2 Synthesizing Strategy: The Confluence Engine

The AI agent is programmed to look for **Confluence**. A single indicator is merely noise; the alignment of three is a signal.

- **Scenario:** The prompt logic instructs the AI: "If BankNifty crosses above VWAP, *AND* the 45,000 Strike Call shows a 15% drop in OI (Short Covering), *AND* RSI is above 60, generate a 'Strong Buy' signal."
- **Option Buying vs. Selling:** The AI must discern the regime. If Implied Volatility (India VIX) is low (<12), it suggests option buying strategies (Long Call/Put). If VIX is high (>18), it suggests option selling (Credit Spreads) to capture Theta decay.

5.3 Technical Indicator Configurations

- **RSI (Relative Strength Index):** Standard 14-period. However, for Indian markets, the "bullish range" is often defined as 60-80 and the "bearish range" as 20-40, rather than the standard 70/30 levels. The AI must be prompted with these specific thresholds.
- **Bollinger Bands:** Used to detect volatility squeezes. A "Squeeze" (bands contracting) followed by a price breakout is a high-probability setup for options buyers as it implies an imminent Gamma explosion.

5.4 Preventing Hallucination: The Data-First Approach

A critical risk in AI-driven finance is "hallucination"—the AI inventing data.

- **Solution:** We utilize a **Data-First Context** strategy. We never ask the AI "What is the price?" Instead, we provide the price in the prompt and ask "Given that the price is X and VWAP is Y, interpret the trend."
- **Structured Enforcement:** The AI is forced to output its reasoning in a strict JSON schema. This allows the deterministic Python backend to validate the logic. If the AI outputs { "signal": "BUY", "reason": "Price above VWAP" } but the system knows price < vwap, the signal is automatically rejected before reaching the user.

6. Comprehensive Implementation Guide: The Vibe Coding Blueprint

This section provides the actionable blueprint for building the platform using vibe coding tools. It translates the architectural theory into concrete prompts and steps.

6.1 Phase 1: The Master System Prompt

This prompt is designed to be input into **Replit Agent 3** or **Cursor** to initialize the entire project structure. It establishes the persona, the strict technical constraints, and the expected outcome.
Role: You are an expert Quantitative Developer and UI/UX Architect specializing in the Indian Stock Market (NSE/BSE). You are tasked with building a professional-grade, real-time options trading dashboard.
Technical Stack:

- **Frontend:** Next.js 15 (App Router), Tailwind CSS, Shadcn/UI components, Lucide Icons.
- **Charting:** TradingView Lightweight Charts (Canvas-based rendering).
- **Backend:** Python (FastAPI) for high-performance data processing and AI orchestration.
- **Data Handling:** WebSockets for real-time tick streaming; Redis for in-memory state management.
- **AI Integration:** Use Anthropic Claude 4.6 via API for reasoning and signal generation.

Core Features & Logic:

1. **Live Market Dashboard:** Display Nifty 50 and BankNifty spot prices. Implement a WebSocket hook to update these values in real-time without page reloads.
2. **Option Chain Analyzer:** Build a data table visualizing the live option chain. Highlight "In-The-Money" (ITM) versus "Out-Of-The-Money" (OTM) rows. Columns must include: Strike, LTP, OI, Change in OI, Delta, Theta.
3. **AI Trade Signal Engine:**
    - Ingest live technical indicators: RSI (14), VWAP, Bollinger Bands (20, 2), and PCR.
    - **Bullish Logic:** If (Price > VWAP) AND (RSI > 60) AND (PCR > 1.0) -> Signal "Bullish/Buy CE".
    - **Bearish Logic:** If (Price < VWAP) AND (RSI < 40) AND (PCR < 0.8) -> Signal "Bearish/Buy PE".
    - **Narrative Generation:** The AI must explain the signal in natural language (e.g., "BankNifty is bullish due to short-covering at 45000 strike combined with RSI momentum.").
4. **Guardrails:** Never invent data. If the WebSocket feed is interrupted, the UI must display "Data Disconnected". Do not offer financial advice; label all outputs as "Technical Analysis Insights".

Visual Style:

- Default to Dark Mode (Financial terminal aesthetic).
- Use monospaced fonts (JetBrains Mono) for all numerical data to ensure readability.
- Use standard financial colors: Green (#22c55e) for bullish/up, Red (#ef4444) for bearish/down.

Execution Plan:

1. Scaffold the Next.js application with a sidebar layout.
2. Create a mock Python WebSocket server that simulates Nifty tick data (sine wave with noise) for initial testing.
3. Implement the OptionChain component using Shadcn Table.
4. Build the AI_Insight_Panel component that fetches the latest analysis from the backend every 60 seconds.

6.2 Phase 2: The "Vibe Loop" of Refinement

Once the initial codebase is generated, the developer enters the "Polish" phase of the vibe coding loop:

1. **Chart Optimization:** "The chart updates are causing a flicker. Refactor the React useEffect hook to update the candle series data efficiently without triggering a full component re-render."
2. **Contextual Nuance:** "Update the Option Chain to highlight strike prices that are multiples of 50 (for Nifty) and 100 (for BankNifty). Calculate the 'Max Pain' point based on the strike with the highest cumulative Open Interest."
3. **Real Data Connection:** "Replace the mock Python server with a real WebSocket connection to the TrueData API. Here is the documentation snippet for their WebSocket authentication...".

6.3 Phase 3: The AI Analyst Prompt (Runtime)

This prompt is embedded within the application's backend code. It is sent to the LLM (Claude/Gemini) every time a market analysis is requested.

System Prompt for Trade Analyst Agent: "You are a senior derivatives trader on the NSE desk. I will provide you with a JSON object containing the latest market data for Nifty 50. This includes: Current Price, VWAP, RSI, Total Call OI, Total Put OI, and the Change in OI for the ATM strike.

Task: Analyze this data and determine the immediate market trend (next 15-30 mins).

- **Short Covering Detection:** If Call OI is decreasing significantly while Price is rising, identify this as 'Short Covering'.
- **Long Unwinding Detection:** If Put OI is decreasing significantly while Price is falling, identify this as 'Long Unwinding'.
- **Output Format:** Return ONLY a JSON object with the following structure: { 'signal': 'BUY_CALL' | 'BUY_PUT' | 'WAIT', 'confidence': 0-100, 'reasoning': 'String', 'key_level': 'Price' }
- Keep the reasoning concise, professional, and data-backed. Do not use flowery language."

7. Data Infrastructure & Risk Management

7.1 The Real-Time Data Pipeline

To fulfill the promise of "real-time" analysis, the data architecture must be optimized for throughput.

1. **Source:** NSE Tick-by-Tick feed via an authorized vendor.
2. **Transport:** Secure WebSocket (WSS).
3. **Processing Layer:**
    - Incoming Tick -> **Update Redis Cache** (Atomic operation).
    - **Calculation Engine:** A background worker (using Python celery or asyncio) computes the new RSI, VWAP, and Greeks.
    - **Analysis Trigger:** Every 1 minute (or upon a significant price deviation), the system triggers the **Agentic Analysis**.
    - **Delivery:** The Agent's output is pushed to the Frontend via **Server-Sent Events (SSE)**, ensuring the user receives the insight instantly without polling.

7.2 Handling Hallucinations and Operational Risk

Financial AI systems operate in a high-stakes environment where errors have direct monetary consequences.

- **Deterministic Validation Layer:** The Python backend acts as a gatekeeper. It parses the JSON output from the AI. If the AI signals "BUY" but the hard-coded technical check reveals that current_price is strictly below VWAP, the system **rejects** the signal. The user never sees the hallucinated advice. This hybrid model uses AI for *context* (narrative generation) but code for *compliance* (rule enforcement).
- **Latency Arbitrage:** Vibe coding can lead to inefficient loops (e.g., asking the AI to analyze every single tick). The architecture must strictly define the analysis frequency. The AI is queried on *candle closes* (e.g., 1-minute or 5-minute intervals), while the WebSocket handles the visualization of live ticks. This balances cost, latency, and utility.
- **Regulatory Compliance:** The platform must include prominent disclaimers that it is an algorithmic analysis tool. In the Indian context, unless the user is a SEBI-registered Investment Advisor (RIA), the tool must frame its outputs as "educational insights" rather than "financial advice."

8. Strategic Roadmap & Future Outlook

8.1 From Dashboard to Autonomy (2027 Horizon)

The current platform functions as a *decision support system*. The next evolutionary step is **Autonomous Execution**. This would involve the "Interface Agent" (OpenAI) communicating directly with broker APIs (like Kite Connect) to place orders. This introduces a new layer of complexity regarding risk management (e.g., max daily loss limits, kill switches) which must be hard-coded, never AI-generated.

8.2 The "Agent Team" Evolution

Future iterations will likely employ complex "Agent Teams".

- **Risk Manager Agent:** A specialized agent that monitors portfolio Beta and volatility, with the authority to "veto" trades proposed by the Trader Agent if market conditions are too hazardous.
- **Macro Agent:** A dedicated agent monitoring global indices (Dow Jones Futures, Brent Crude) and currency pairs (USD/INR) to provide a broader context for local Nifty moves.

8.3 Conclusion

Building a functional, professional trading platform for the Indian market is no longer the exclusive domain of institutional hedge funds with massive engineering teams. The **vibe coding** revolution has lowered the barrier to entry, allowing a single knowledgeable developer to orchestrate powerful AI models like **Claude Opus 4.6** and **Gemini 3 Pro** into a cohesive system. By following this architectural blueprint—combining the creative reasoning of LLMs with the deterministic rigor of quantitative finance—traders can build bespoke tools that rival professional terminals. The success of such a platform lies not in the code itself, but in the

precision of the prompts and the robustness of the data pipelines that feed the AI.

Appendix: Technical Specifications Summary

Component	Technology	Rationale
Development Style	**Vibe Coding**	Rapid prototyping via Replit; Algorithm optimization via Cursor.
Reasoning Engine	**Claude 4.6 Opus**	Superior logic handling for complex option strategies and "chain of thought" validation.
Context Engine	**Gemini 3 Pro**	Massive context window for ingesting historical data and real-time news search.
Data Feed	**TrueData / Global Datafeeds**	Authorized, low-latency NSE/BSE WebSockets with Greeks support.
Frontend	**Next.js 15 + Shadcn/UI**	Modern, server-side optimized, reactive UI with professional aesthetics.
Backend	**Python (FastAPI)**	Async support; native integration with financial libraries (pandas, TA-Lib).
Cache Layer	**Redis**	Sub-millisecond data retrieval for live market ticks.
Visualization	**TradingView Lightweight Charts**	High-performance, canvas-based charting standard for finance.

Works cited

1. What is Vibe Coding? How To Vibe Your App to Life - Replit Blog, https://blog.replit.com/what-is-vibe-coding 2. Vibe Coding Tools Guide: Best AI App Builders 2026 - Replit, https://replit.com/discover/best-vibe-coding-tools 3. AI dev tool power rankings & comparison [Feb. 2026] - LogRocket Blog, https://blog.logrocket.com/ai-dev-tool-power-rankings/ 4. Replit vs Cursor: Which AI Coding Platform Fits Your Workflow Best?, https://replit.com/discover/replit-vs-cursor 5. How does Replit compare to Cursor and Windurf for vibe coding? - Reddit, https://www.reddit.com/r/replit/comments/1kco0xe/how_does_replit_compare_to_cursor_and_windurf_for/ 6. 7 Best Vibe Coding Tools (2025): From Prompt To Production - AceCloud, https://acecloud.ai/blog/best-vibe-coding-tools/ 7. Replit-style AI agents vs Cursor/VSCode for non-dev app builders? - Reddit, https://www.reddit.com/r/replit/comments/1pf7l4c/replitstyle_ai_agents_vs_cursorvscode_for_nondev/ 8. Can a newbie really vibe code an app? I tried Cursor and Replit to find out | ZDNET, https://www.zdnet.com/article/beginner-vibe-coding-apps-cursor-replit-hands-on/ 9. I Tested The Top 3 AIs for Vibe Coding (Shocking Winner) - YouTube, https://www.youtube.com/watch?v=277l16eJphI 10. Efficient prompting with Replit AI, https://docs.replit.com/tutorials/effective-prompting 11. 9 Vibe Coding Best Practices -

Memberstack, https://www.memberstack.com/blog/9-vibe-coding-best-practices 12. Claude Opus 4.6: New Control Plane for SDLC & IT Operations - Rapyder, https://www.rapyder.com/blog/claude-opus-4-6/ 13. Claude Opus 4.6 System Card - Anthropic, https://www-cdn.anthropic.com/0dd865075ad3132672ee0ab40b05a53f14cf5288.pdf 14. Prompting best practices - Claude API Docs, https://platform.claude.com/docs/en/build-with-claude/prompt-engineering/claude-prompting-best-practices 15. Introducing Claude Opus 4.6 - Anthropic, https://www.anthropic.com/news/claude-opus-4-6 16. Claude and Perplexity AI for Finance: All You Need to Know - Neurons Lab, https://neurons-lab.com/article/claude-perplexity-for-finance/ 17. AI Showdown: Comparative Analysis of AI Models on Hallucination, Bias, and Accuracy - Software Testing and Development Company - Shift Asia, https://shiftasia.com/column/comparative-analysis-of-ai-models-on-hallucination-bias-and-accuracy/ 18. Gemini 3 — Google DeepMind, https://deepmind.google/models/gemini/ 19. Claude vs Gemini vs GPT: Which AI Model Should Enterprises Choose? | TTMS, https://ttms.com/claude-vs-gemini-vs-gpt-which-ai-model-should-enterprises-choose-and-when/ 20. Reducing LLM Hallucinations: A Developer's Guide - Zep, https://www.getzep.com/ai-agents/reducing-llm-hallucinations/ 21. Best Practices for Controlling LLM Hallucinations at the Application Level - Parasoft, https://www.parasoft.com/blog/controlling-llm-hallucinations-application-level-best-practices/ 22. TradExpert: Revolutionizing Trading with Mixture of Expert LLMs - arXiv.org, https://arxiv.org/html/2411.00782v1 23. TradingAgents: Multi-Agents LLM Financial Trading Framework - arXiv, https://arxiv.org/html/2412.20138v5 24. Next.js AI Prompt - shadcn.io, https://www.shadcn.io/prompts/nextjs-ai 25. Interactive Dashboard with Next.js and Python | 2025 Guide - August Infotech, https://www.augustinfotech.com/blogs/building-an-interactive-dashboard-with-next-js-and-python/ 26. From Vibe Coding to Engineering: Building a Production-Ready Next.js 15 Blog with AI, https://dev.to/dumebii/the-ultimate-prompt-strategy-how-to-vibe-code-production-ready-websites-4e9 27. 20 Best Free Next.js Admin Dashboard Templates 2026 - AdminLTE.IO, https://adminlte.io/blog/nextjs-admin-dashboard-templates/ 28. Live Trading: Visualize Trades as They Happen Using WebSockets and Dash | by Filippos Tzimopoulos | Python in Plain English, https://python.plainenglish.io/live-trading-visualize-trades-as-they-happen-using-websockets-and-dash-833bb0621397 29. WebSocket Implementation with Next.js (Node.js + React in One App) - DEV Community, https://dev.to/addwebsolutionpvtltd/websocket-implementation-with-nextjs-nodejs-react-in-one-app-gb6 30. surnr/option-chain-algotrade: This files contains all the necessary function required to run the Kite Ticker API to Stream and Fetch Option Data - GitHub, https://github.com/surnr/option-chain-algotrade 31. Realtime API with WebSocket | OpenAI API, https://developers.openai.com/api/docs/guides/realtime-websocket/ 32. Real-Time data streaming using web sockets, a how-to guide | Intrinio, https://docs.intrinio.com/tutorial/websocket 33. Authorized NSE, BSE , MCX & NCDEX Stock Realtime API Provider - Global Datafeeds, https://globaldatafeeds.in/authorized-nse-bse-mcx-stock-realtime-api-provider/ 34. APIs - Real-time, historical, snapshot, delayed, end-of-day, entire exchange, option-chain, option greeks data of Indian Stock Exchange - Global Datafeeds, https://globaldatafeeds.in/asdf/ 35. Market Data API | Live, SnapChat, Options Chain, Option Greeks - TrueData, https://www.truedata.in/market-data-apis 36. Best indicators for option trading explained - Dhan, https://dhan.co/blog/option-trading/best-indicators-for-option-trading/ 37. Option Chain -

DhanHQ Ver 2.0 / API Document, https://dhanhq.co/docs/v2/option-chain/ 38. Power Your Trading and Financial Applications with Upstox Robust API Suite!, https://upstox.com/trading-api/ 39. OptionGreeks - What is OpenAlgo? | Documentation, https://docs.openalgo.in/api-documentation/v1/data-api/optiongreeks 40. tradingview/charting-library-examples: Examples of ... - GitHub, https://github.com/tradingview/charting-library-examples 41. Getting started | Lightweight Charts - GitHub Pages, https://tradingview.github.io/lightweight-charts/docs 42. Interactive Brokers API, TradingView Charts in Python - YouTube, https://www.youtube.com/watch?v=TlhDI3PforA 43. Top Indicators for Option Trading in India - Bajaj Finserv, https://www.bajajfinserv.in/options-trading-indicators 44. Indicators for Options Trading - Bajaj Broking, https://www.bajajbroking.in/blog/indicators-for-option-trading 45. Best Indicators For Option Trading | Choice - Choiceindia.com, https://choiceindia.com/blog/best-indicators-for-option-trading 46. Best Indicators for Option Trading - Most Accurate Intraday Trading Indicators - Groww, https://groww.in/blog/best-indicators-for-option-trading 47. Claude 4.6, GPT 5.3, Gemini 3 Pro and more; Top AI models that rattled stock market, https://upstox.com/news/market-news/us-stocks/claude-4-6-gpt-5-3-gemini-3-pro-and-more-top-ai-models-that-rattled-stock-market/article-189408/