AI Tools Usage and Training Summary

AI Tools Used

1. ChatGPT (Free Tier) — Used in early stages for architecture decisions (App Router vs Pages Router, Drizzle vs Prisma), API research (CoinGecko, CryptoCompare, Reddit endpoints and rate limits), component scaffolding (onboarding form, dashboard layout, vote buttons), and debugging TypeScript/Next.js issues.

2. Claude (free tier) — Used in later stages for hands-on debugging and improvements: diagnosed why AI insights silently fell back to a hardcoded template (OpenRouter 429 rate limit, Gemma rejecting system role messages), added Hugging Face as a fallback LLM provider, fixed meme image filter (.jpeg URLs being filtered out), iteratively improved the AI prompt (longer insights, all tracked assets, qualitative trend descriptions instead of inaccurate numbers), replaced img tags with Next.js Image components, and generated documentation. Claude also wrote most of this document and the AI training suggestions.

3. Both models also wrote all of the copywriting and the design. I let them come up with design ideas and create a html document showcasing possible design directions but ended up not using those designs at all, and chose a minimal clean one instead. The designs are added to the project as a refernce under the docs folder, but none were used in the final product - in order to see them, open html document in browser.

4. Bonus features (prices history charts and link to the original new story from the headlines) were completely vibe coded.

Feedback Storage and Future Training

Current System: Every dashboard section has thumbs up/down buttons. Votes are stored in a votes table with userId, section, contentId, vote (+1 or -1), and timestamp. Each user can have one active vote per section+content combination.

How Feedback Influences Insights: Before generating an insight, the system queries the user's last 200 votes, aggregates them by section (e.g. insight: +12/-3), and includes these signals in the LLM prompt. This lets the model adjust tone and focus based on what the user finds valuable.

Proposed Training Pipeline:

Step 1 — Data Collection: Store generated insight text alongside votes to create (prompt, response, vote) tuples.

Step 2 — Preference Dataset: Build positive/negative pairs from upvoted vs downvoted insights, grouped by investor type.

Step 3 — Fine-Tuning: Apply DPO (Direct Preference Optimization) to fine-tune the model using preference pairs. DPO is simpler than RLHF and more practical for small teams.

Step 4 — Evaluation: A/B test fine-tuned vs base model, track upvote rates over time, monitor for quality regressions.

Practical Considerations: DPO needs hundreds to thousands of preference pairs (weeks/months of collection). Until then, prompt-based signals are a reasonable proxy. For cost efficiency, LoRA fine-tuning on a small open model would be ideal. All training data should be anonymized.