LIVE CRYPTO TRACKER

1. Overview

This project is a Python-based desktop GUI application that tracks live cryptocurrency prices using the Binance API. It displays multiple coin prices in real-time, auto-refreshes every few seconds, plots price history graphs, and triggers alerts when a coin crosses a specified price threshold.

Technology Choice: Building a Live Crypto Tracker using Python, Tkinter & Binance API

Justification:

Python was selected for its simplicity, Tkinter for GUI development, and the Binance Public API for reliable real-time cryptocurrency pricing without authentication.

End Goal:

- Fetch live crypto prices

- Display data in a GUI

- Auto-refresh after every 5 seconds

- Display professional crypto-style graphs

- Support currency selection and alerts

2. Technology Summary

Definition: This is a simple program that returns the price of various cryptocurrencies to the dollar and uses a graph to show trend.

Use cases: This program can be integrated with online traders to make it easier for users to analyze trends and make trade decisions.

Real-World Example: Such programs would be found integrated into various online trading platforms such as Metatrader and Binance.

3. System Requirements

- Python 3.8+

- Windows / Linux / macOS

- Internet connection

- Libraries: requests, matplotlib

4. Installation & Setup

1. Install Python

2. Install dependencies: pip install requests matplotlib pyinstaller

3. Run: python crypto_tracker_binance.py

4. Build EXE: pyinstaller --onefile --noconsole crypto_tracker_binance.py

5. Features

- Multi-coin tracking

- Currency selector (USD, EUR, GBP)

- Refresh interval selector

- Fixed-size trading-style graphs

- Alerts

6. AI Prompt Journal

Prompt 1 – Project Definition

"I have a project for a short course I'm doing on using AI to code that I'd like your help with."

Purpose:
To introduce the project context and seek AI assistance for an academic coding task.

Prompt 2 – Technology Selection

"For my project I'll be using python and I'll be developing a live crypto tracker."

Purpose:
To define the programming language and application domain.

Prompt 3 – Feature Requirements

"CoinGecko, GUI, track multiple coins, show price, auto refresh, graphs, alerts, beginner friendly."

Purpose:
To specify functional and non-functional requirements for the application.

Prompt 4 – Documentation Request

"I would also like comprehensive documentation that I can put into a Word document for the presentation."

Purpose:
To request structured academic documentation for submission and presentation.

Prompt 5 – Code Generation

"Now the code."

Purpose:
To request the initial implementation of the crypto tracker application.

Prompt 6 – Error Reporting

"Failed to fetch prices from CoinGecko error."

Purpose:
To diagnose API connectivity and data-fetching issues.

Prompt 7 – Debugging Feedback

"The modify fetch part of the fix is flawed."

Purpose:
To request correction of an incorrect or incomplete solution.

Prompt 8 – Documentation Format Alignment

"Could you please generate the documentation using this format."

Purpose:
To align project documentation with an institutional template.

Prompt 9 – API Reliability Concern

"The CoinGecko API is having issues. What other cost friendly API can I use?"

Purpose:
To evaluate alternative APIs due to reliability constraints.

Prompt 10 – API Migration

"Could you edit my code to use the Binance API?"

Purpose:
To refactor the application to use a more stable data source.

Prompt 11 – Runtime Issue Identification

"It's loading but not returning output."

Purpose:
To identify logic or UI update issues after API migration.

Prompt 12 – Full Rewrite Request

"Could you generate fresh code using Binance API. Not corrections, everything from scratch."

Purpose:
To eliminate accumulated bugs by rebuilding the application cleanly.

Prompt 13 – Documentation & Packaging Enhancements

"Update word documentation to match this version, add refresh period select button, provide both .py file and .exe packaged app, edit the graph to look more like a crypto graph."

Purpose:
To finalize the project for academic submission and demonstration.

Prompt 14 – Graph Styling Bug

"The graphs no longer have labels and gridlines and interval labels. How do I fix that?"

Purpose:
To restore professional chart readability after dynamic updates.

Prompt 15 – Feature Enhancement

"Add a dropdown menu to choose a currency to display and lengthen the x-axis a bit."

Purpose:
To improve usability and data visualization realism.

Prompt 16 – Graph Stability Improvement

"Could you make the graph and grid a fixed size that doesn't change with results?"

Purpose:
To achieve trading-platform–style fixed chart layouts.

Prompt 17 – Documentation Access

"Could you please get me another link to the final documentation?"

Purpose:
To obtain a stable, final version of the project documentation.

Prompt 18 – Prompt Audit

"Could you please list all prompts I've used in this project?"

Purpose:
To formally document AI usage as required in AI-assisted coursework.

7. Challenges & Solutions

- API reliability: solved by switching to Binance

- Graph distortion: solved by fixed axes

8. Conclusion

The project successfully demonstrates AI-assisted application development.

9. Project Timeline

Date	Activity
Monday 8/12/25	• Github repo creation. • Project start. • First AI prompts. • First version of program.
Tuesday 9/12/25	• Debugging. • First version upload to Github.
Wednesday 10/12/25	• Debugging.
Thursday 11/12/25	• Project restart using Binance API. • Debugging.
Friday 12/12/25	• Debugging. • Fine tuning.
Monday 15/12/25	• Debugging. • Final program upload to Github. • Documentation compilation and editing.

10.References

- Binance API Docs

- Python Tkinter Docs

- Matplotlib Docs