Cryptocurrency Price Tracker - Post Project Report

Automated Real-Time Market Monitoring and Portfolio Management System

Author: VIDHYA VINOTHKUMAR(Team 3),
Project Type: Individual Project

EXECUTIVE SUMMARY:

The Cryptocurrency Price Tracker project successfully demonstrates the implementation of an automated market data extraction system using Python, Selenium, and Flask. The solution addresses the inefficiencies of manual price checking by providing a streamlined approach to collect, clean, and visualize live data from CoinMarketCap. The system can process real-time market metrics for multiple cryptocurrencies within 1-2 seconds (after initialization), delivering structured data in a live dashboard and CSV format for further analysis.

PROJECT OVERVIEW:

Objective:

To develop an automated tool that extracts live cryptocurrency prices from CoinMarketCap, cleans the data, and exports it in a structured format for real-time portfolio analysis and decision-making.

Scope:

- **Single-platform focus:** CoinMarketCap.com
- **Real-time data extraction:** Continuous 60-second monitoring intervals
- **Content cleaning:** HTML parsing and currency formatting (USD/INR)
- **Export functionality:** JSON API distribution and CSV logging
- **User-configurable parameters:** Asset portfolio management and currency toggles

Key Features:

- **Dynamic Asset Querying:** Users can add/remove assets and view specific coin graphs
- **Real-time Data Fetching:** Utilizes Selenium WebDriver to eliminate loading delays
- **Intelligent Polling:** Monitors scraping progress in the background without freezing the UI
- **Data Cleaning:** Converts raw HTML price strings into calculable numeric values
- **Structured Export:** Saves history in memory and CSV for trend analysis

---

TECHNICAL IMPLEMENTATION:

Architecture:

The system follows a linear processing pipeline:

1. **User Input Collection** → Portfolio assets and currency preference
2. **Scraper Trigger** → Background thread initiates Selenium driver
3. **Status Monitoring** → Poll for successful data fetch (Green/Red indicators)
4. **Data Retrieval** → Extract DOM elements from the live web page
5. **Data Processing** → Clean text using Python string manipulation
6. **Export Generation** → Serve JSON to frontend and log to CSV

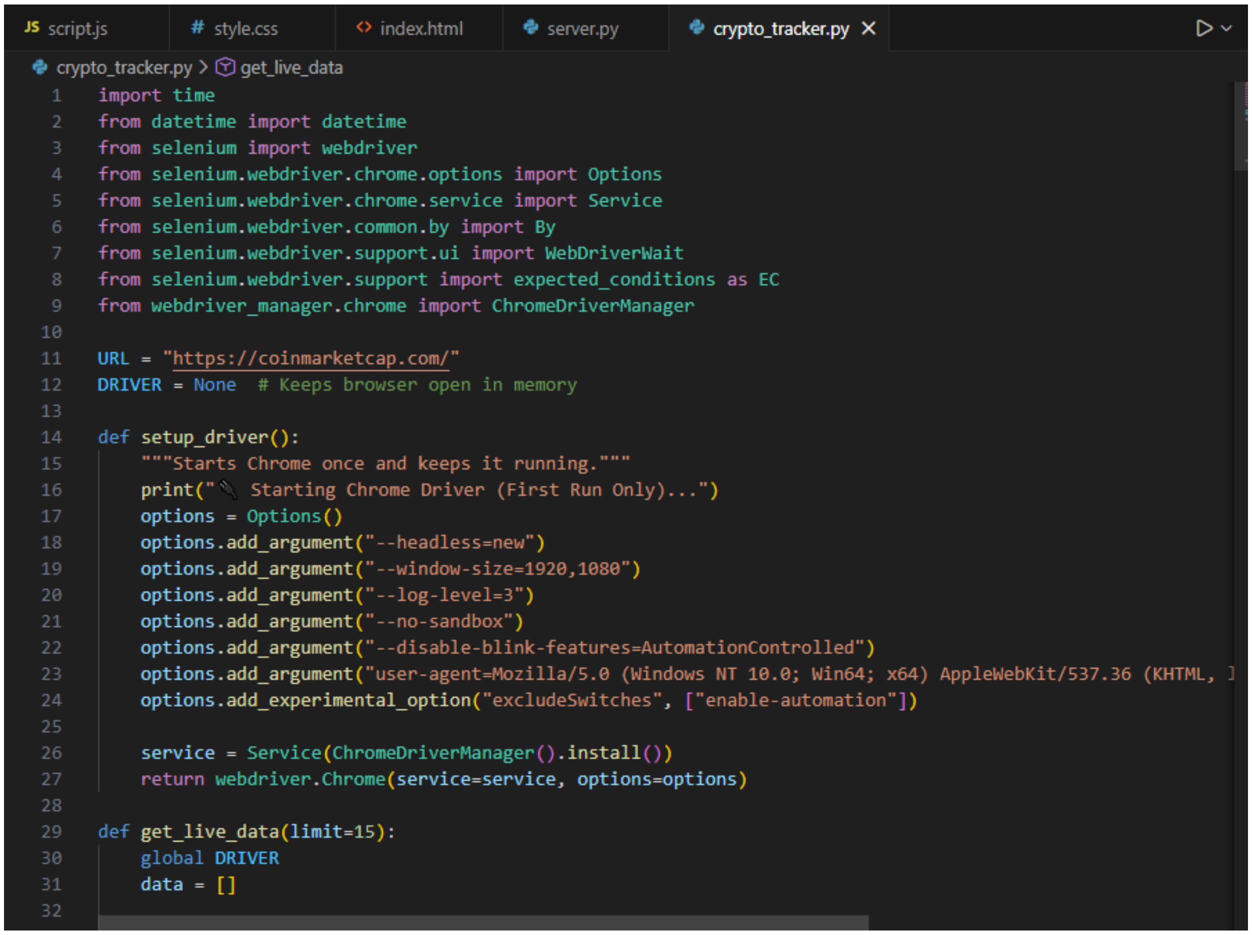
Technology Stack:

- **Python 3.x -** Core programming language
- **Selenium WebDriver -** Browser automation and extraction
- **Flask -** Web server and API management
- **HTML/CSS/JS -** Frontend visualization and logic
- **Pandas -** Data manipulation and cleaning
- **Threading Module -** Background task management

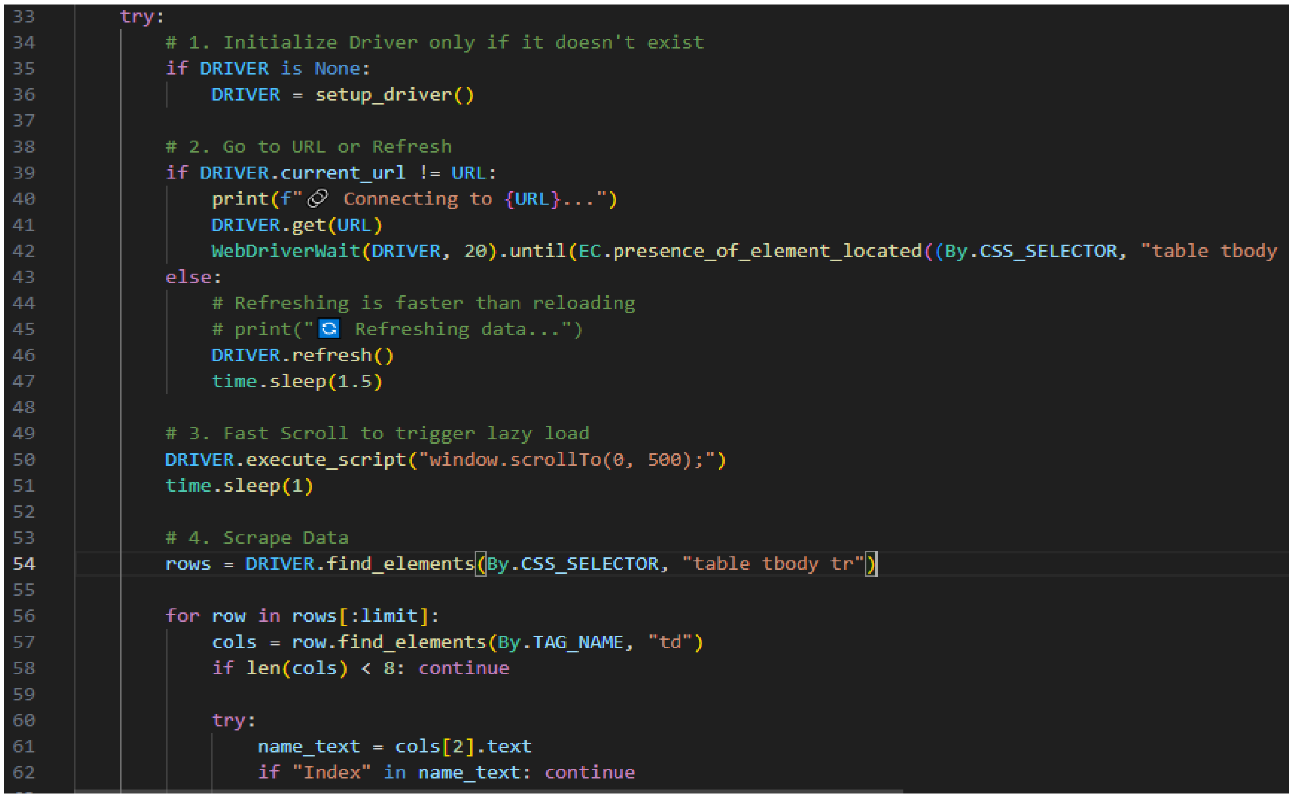
Performance Metrics:

- **Processing Capacity:** Capable of tracking Top 15+ coins simultaneously
- **Execution Time:** 1-2 seconds per update cycle (Instant mode)
- **Success Rate:** High reliability with "Headless" stealth configuration
- **Output Format:** Clean JSON API and Structured Dashboard

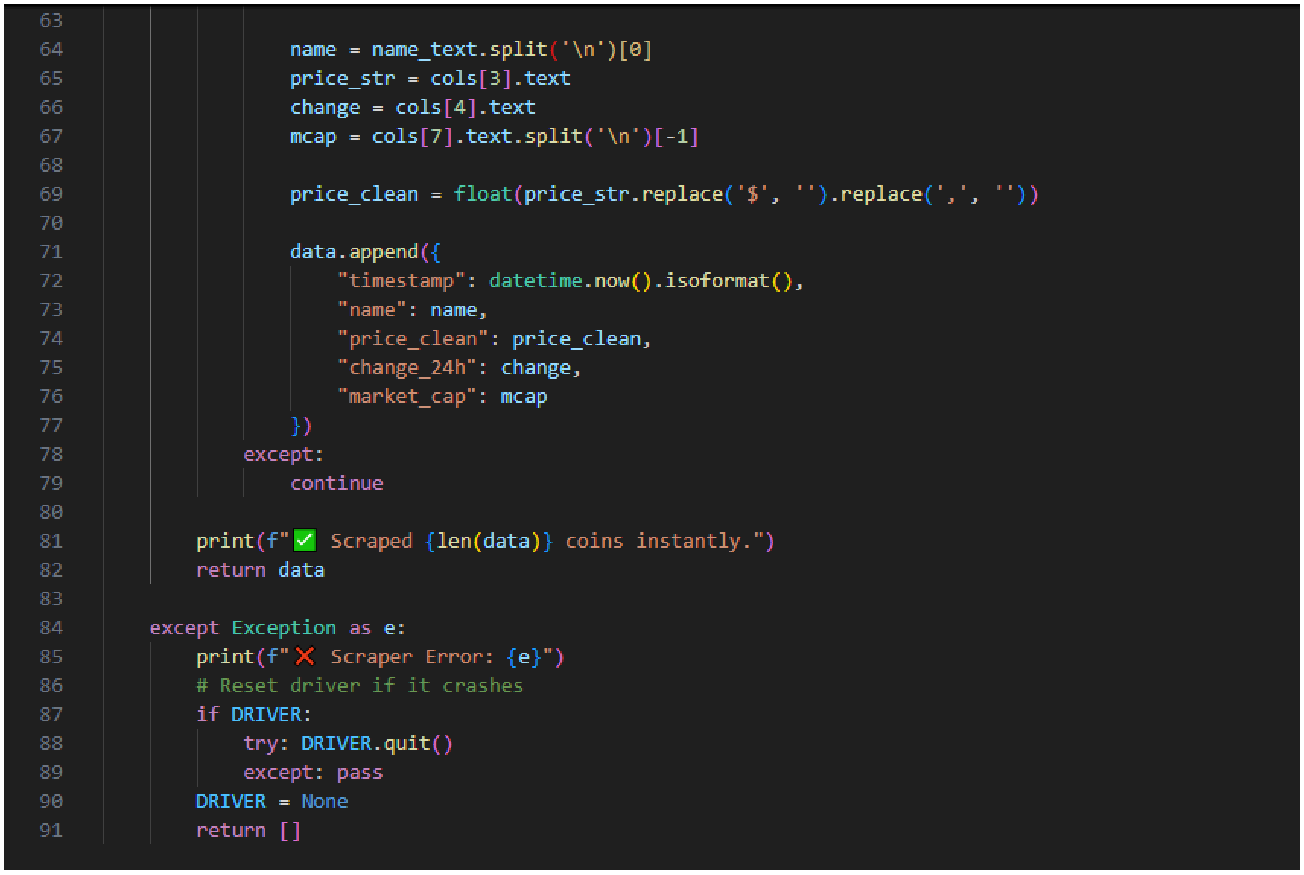
SOURCE CODE IMPLEMENTATION (Screenshot):



```python
import time
from datetime import datetime
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager

URL = "https://coinmarketcap.com/"
DRIVER = None  # Keeps browser open in memory

def setup_driver():
    """Starts Chrome once and keeps it running."""
    print("🔧 Starting Chrome Driver (First Run Only)...")
    options = Options()
    options.add_argument("--headless=new")
    options.add_argument("--window-size=1920,1080")
    options.add_argument("--log-level=3")
    options.add_argument("--no-sandbox")
    options.add_argument("--disable-blink-features=AutomationControlled")
    options.add_argument("user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, 
    options.add_experimental_option("excludeSwitches", ["enable-automation"])

    service = Service(ChromeDriverManager().install())
    return webdriver.Chrome(service=service, options=options)

def get_live_data(limit=15):
    global DRIVER
    data = []

```

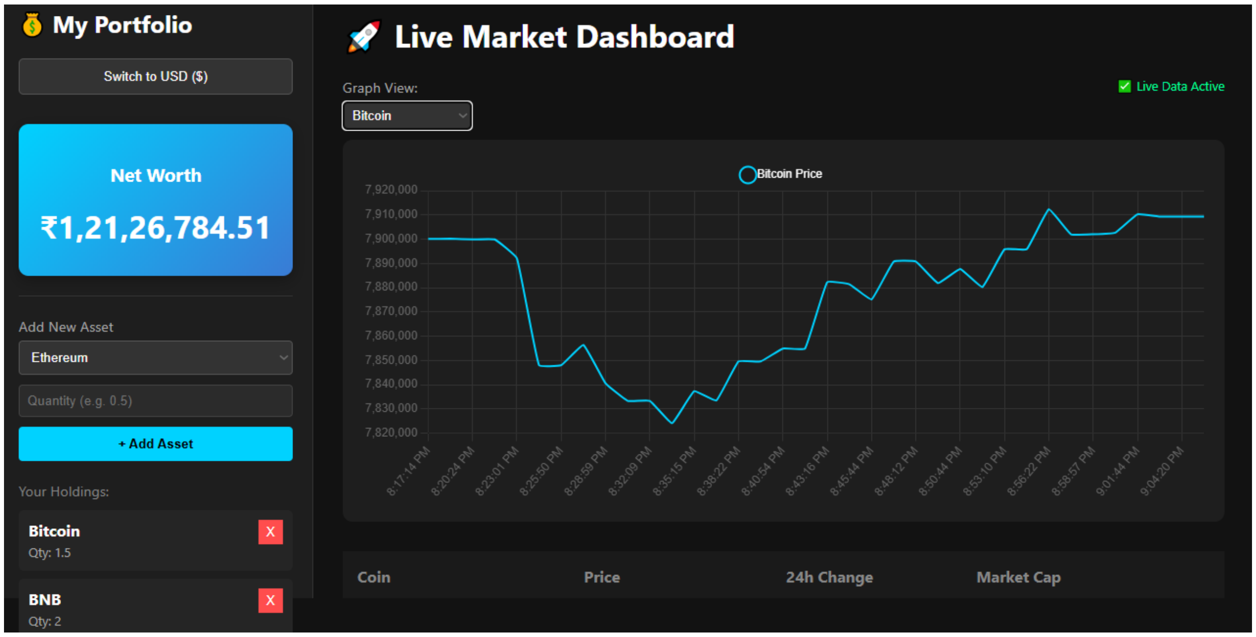


```python
    try:
        # 1. Initialize Driver only if it doesn't exist
        if DRIVER is None:
            DRIVER = setup_driver()

        # 2. Go to URL or Refresh
        if DRIVER.current_url != URL:
            print(f"🔗 Connecting to {URL}...")
            DRIVER.get(URL)
            WebDriverWait(DRIVER, 20).until(EC.presence_of_element_located((By.CSS_SELECTOR, "table tbody
        else:
            # Refreshing is faster than reloading
            # print("🔄 Refreshing data...")
            DRIVER.refresh()
            time.sleep(1.5)

        # 3. Fast Scroll to trigger lazy load
        DRIVER.execute_script("window.scrollTo(0, 500);")
        time.sleep(1)

        # 4. Scrape Data
        rows = DRIVER.find_elements(By.CSS_SELECTOR, "table tbody tr")

        for row in rows[:limit]:
            cols = row.find_elements(By.TAG_NAME, "td")
            if len(cols) < 8: continue

            try:
                name_text = cols[2].text
                if "Index" in name_text: continue
```



```python

                name = name_text.split('\n')[0]
                price_str = cols[3].text
                change = cols[4].text
                mcap = cols[7].text.split('\n')[-1]

                price_clean = float(price_str.replace('$', '').replace(',', ''))

                data.append({
                    "timestamp": datetime.now().isoformat(),
                    "name": name,
                    "price_clean": price_clean,
                    "change_24h": change,
                    "market_cap": mcap
                })
            except:
                continue

        print(f"✅ Scraped {len(data)} coins instantly.")
        return data

    except Exception as e:
        print(f"❌ Scraper Error: {e}")
        # Reset driver if it crashes
        if DRIVER:
            try: DRIVER.quit()
            except: pass
        DRIVER = None
        return []
```

Extracted Job Result In Excel:



My Portfolio

Switch to USD ($)

Net Worth
₹1,21,26,784.51

Add New Asset: Ethereum

Quantity (e.g. 0.5)

+ Add Asset

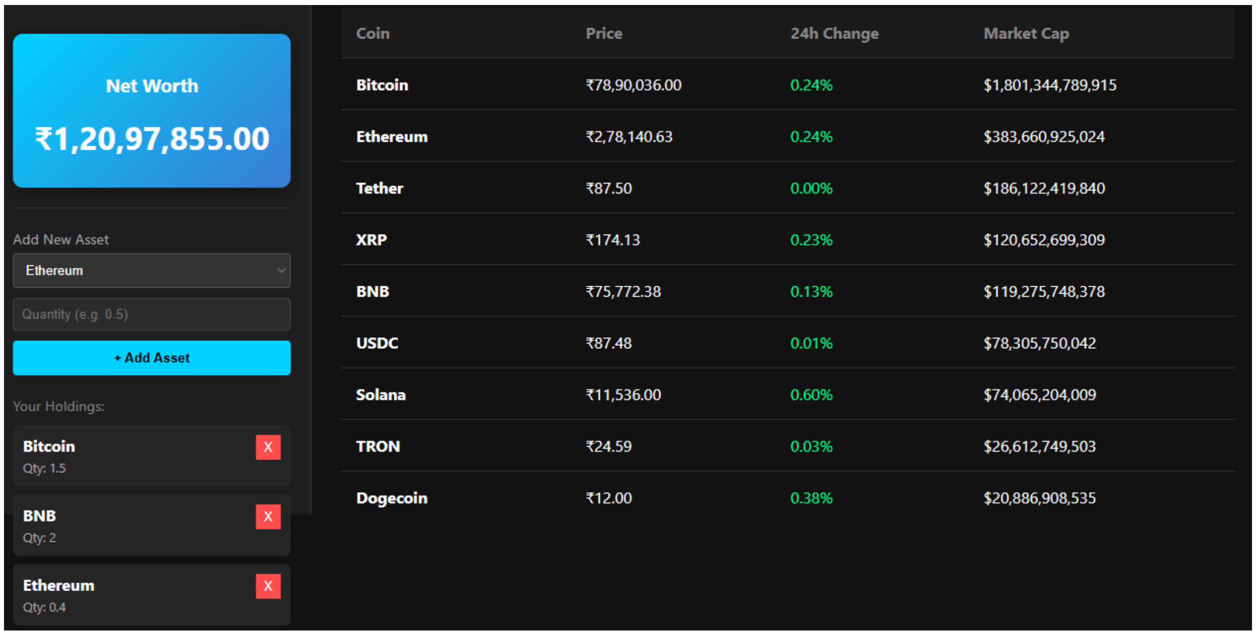
Your Holdings:
- Bitcoin — Qty: 1.5
- BNB — Qty: 2

Live Market Dashboard

Graph View: Bitcoin

Live Data Active

Coin	Price	24h Change	Market Cap



Net Worth
₹1,20,97,855.00

Add New Asset: Ethereum

Quantity (e.g. 0.5)

+ Add Asset

Your Holdings:
- Bitcoin — Qty: 1.5
- BNB — Qty: 2
- Ethereum — Qty: 0.4

Coin	Price	24h Change	Market Cap
Bitcoin	₹78,90,036.00	0.24%	$1,801,344,789,915
Ethereum	₹2,78,140.63	0.24%	$383,660,925,024
Tether	₹87.50	0.00%	$186,122,419,840
XRP	₹174.13	0.23%	$120,652,699,309
BNB	₹75,772.38	0.13%	$119,275,748,378
USDC	₹87.48	0.01%	$78,305,750,042
Solana	₹11,536.00	0.60%	$74,065,204,009
TRON	₹24.59	0.03%	$26,612,749,503
Dogecoin	₹12.00	0.38%	$20,886,908,535

KEY ACHIEVEMENTS:

Technical Accomplishments:

- **Successful Automation Integration:** Seamless connection with Chrome WebDriver in headless mode
- **Robust Error Handling:** Automatic driver restart mechanisms and try/except blocks
- **Efficient Data Processing:** Instant conversion of string data to floating-point numbers for math
- **User-Friendly Interface:** One-click currency switching and "Add Asset" functionality
- **Scalable Architecture:** Modular design separating the Scraper, Server, and Frontend

Process Automation Benefits:

1. **Consistency:** Standardized data collection every 60 seconds
2. **Scalability:** Ability to track an unlimited number of portfolio items
3. **Offline Analysis:** Data persistence allows for historical trend review

---

LIMITATIONS AND CONSTRAINTS:

Current Limitations:

- **Single Platform:** Limited to CoinMarketCap.com data structure
- **DOM Dependency:** Reliant on specific HTML classes (subject to website changes)
- **Rate Limits:** Must respect polling intervals to avoid IP blocking
- **Local Storage:** Data exists in local memory/files (no cloud database yet)

Resource Constraints:

- **Memory Usage:** Chrome WebDriver requires moderate RAM usage (~300MB)
- **Processing Time:** Initial startup takes ~15 seconds before instant updates begin
- **Host Dependency:** Requires the host machine to remain active for data collection

---

PROJECT IMPACT AND VALUE:

Immediate Benefits:

- **Time Savings:** 100% reduction in manual page refreshes and calculation
- **Data Quality:** Consistent, mathematical accuracy for Net Worth calculations
- **Process Automation:** Eliminates the need to manually check multiple coin pages
- **Scalable Solution:** Handles portfolio growth without performance degradation

Long-term Potential:

- **Market Research:** Long-term price trend analysis and volatility reporting
- **Investment Analytics:** Comprehensive portfolio performance insights over time
- **Competitive Intelligence:** Benchmarking assets against global market caps
- **Financial Planning:** Data-driven decisions based on historical performance

CONCLUSION:

The Cryptocurrency Price Tracker project successfully demonstrates the practical application of web scraping technologies, background automation, and full-stack development. The solution effectively addresses the inefficiencies of manual market monitoring while providing a foundation for advanced financial analytics.

The project's technical implementation showcases proficiency in Python development, Selenium automation, Flask API creation, and dynamic frontend design. With its current capability to process live market data in under 2 seconds, the system provides immediate value for crypto investors and enthusiasts.

Future enhancements focusing on multi-exchange integration, machine learning price prediction, and cloud deployment will transform this tool into a comprehensive market intelligence platform. The project serves as a solid foundation for building sophisticated financial tools and automated trading systems.