BTC Technical Analysis Web App

Francisco Moya

Contents

Chapter 1

Introduction

1.1 Context

The **BTC Technical Analysis Web App** is a client-side application developed using standard web technologies (HTML, CSS, and JavaScript) with the goal of providing an accessible and lightweight platform for cryptocurrency analysis. The system connects to the **Binance API** in real time to fetch market data and calculates several key technical indicators commonly used by traders, such as **Simple Moving Average (SMA)**, **Exponential Moving Average (EMA)**, **Relative Strength Index (RSI)**, and **Moving Average Convergence Divergence (MACD)**.

1.2 Motivation

The motivation for building this application was to design a tool that enables users to explore the dynamics of financial markets without depending on heavy third-party platforms or broker software. By focusing on the BTC/USDT pair, the project offers a clear and practical case study that balances technical accuracy with usability.

1.3 Educational Perspective

Beyond visualization, the system allows users to **simulate potential signals** by defining entries, stop-losses, and take-profit levels. These simulations are stored locally in the browser, allowing for the creation of a personal log of decisions. This feature is particularly relevant for educational purposes, as it encourages users to reflect on their strategies and measure hypothetical outcomes.

1.4 Project Evolution

Ultimately, this project not only demonstrates technical proficiency in integrating APIs and visualizing financial data but also represents the foundation that later inspired the development of a more advanced system: the **Crypto Signal Generator in Python**, which extended the analysis to multiple assets and introduced automated alerts via Telegram.

Chapter 2

Objectives

The main objectives of the **BTC Technical Analysis Web App** were defined to ensure that the project delivers both technical accuracy and practical usability. These objectives guided the design, implementation, and evaluation of the system.

2.1 Integration of Real-Time Market Data

- Connect to the Binance API and retrieve live BTC/USDT data streams.
- Ensure automatic updates without requiring manual refreshes.

2.2 Calculation of Key Technical Indicators

- Implement core indicators widely used in trading: SMA(10), EMA(10), RSI, and MACD.
- Provide visual representation of indicators directly overlaid on price charts.

2.3 Visualization of Market Information

- Generate interactive and responsive charts displaying price action, indicators, volume, histograms, and potential signals.
- Offer users the ability to switch between different timeframes (e.g., 1m, 5m, 15m, 1h).

2.4 Signal Simulation and Record Keeping

- Allow users to preview a potential trade setup, including entry price, stop-loss, and take-profit levels.

- Save simulated trades locally in the browser using LocalStorage, enabling historical review of past decisions.

2.5 Usability and Accessibility

- Develop the platform entirely with client-side technologies (HTML, CSS, JS), ensuring that it can run in any modern browser without installation.
- Provide a clean and intuitive interface accessible for both beginners and advanced users.

2.6 Educational Purpose

- Encourage self-directed learning by enabling users to analyze data, simulate signals, and reflect on outcomes.
- Position the application as a foundation for further research and development in cryptocurrency trading systems.

Chapter 3

Methodology / Design

The design of the **BTC Technical Analysis Web App** follows a modular and structured methodology, ensuring clarity in both development and future extensibility. The system can be divided into four main components: Data Fetching, Indicator Calculation, Visualization, and Simulation & Storage.

3.1 System Architecture

The overall workflow of the system can be summarized as follows:

1. **Data Fetching** – Retrieve real-time BTC/USDT price and volume data from the Binance API.
2. **Indicator Calculation** – Compute SMA, EMA, RSI, MACD, and other relevant metrics directly on the client side.
3. **Visualization** – Render interactive charts using Chart.js to display price action, indicators, volume, histograms, and potential signals.
4. **Simulation & Storage** – Allow users to test hypothetical signals and store results locally via the browser's LocalStorage.

This architecture ensures that all calculations and visualizations remain lightweight, browser-based, and independent from external servers.

3.2 Data Fetching Module

- The system uses the **Binance public REST API** to retrieve OHLC (Open, High, Low, Close) and volume data.
- Timeframes are configurable (1m, 5m, 15m, 1h, etc.).

- Data requests are repeated periodically to maintain live updates, creating a pseudo-streaming experience without complex infrastructure.

3.3 Indicator Calculation Module

The application computes key trading indicators in JavaScript:

- **SMA(10)**: arithmetic mean of the last 10 closing prices.
- **EMA(10)**: weighted average emphasizing recent prices.
- **RSI(14)**: momentum oscillator detecting overbought/oversold conditions.
- **MACD (12,26,9)**: difference between two EMAs with signal line and histogram for trend confirmation.

All computations are performed client-side, demonstrating the feasibility of browser-based quantitative analysis.

3.4 Visualization Module

- **Chart.js** is used as the primary library for rendering interactive charts.
- Candlestick charts are enhanced with overlays (SMA, EMA) and oscillators (RSI, MACD).
- Volume and MACD histogram are displayed as complementary panels.
- Charts are fully responsive and auto-update in real time.

3.5 Simulation and Storage Module

- Users may simulate trades by selecting entry, stop-loss (SL), and take-profit (TP) levels.
- The application automatically calculates the **risk/reward ratio (R/R)**.
- Simulations are logged into **LocalStorage**, ensuring persistence even after browser closure.
- A history panel displays past simulations with date, entry, SL, TP, and R/R values.

3.6 Design Principles

- **Simplicity:** Implemented with pure HTML, CSS, and JS (no heavy frameworks).
- **Accessibility:** Works on any modern browser without installation.
- **Transparency:** Indicator values and signal explanations are visible and easy to interpret.
- **Extensibility:** Modular design allows adding new indicators, assets, or features with minimal effort.

Chapter 4

Technical Implementation

The **BTC Technical Analysis Web App** was implemented entirely with HTML, CSS, and JavaScript, ensuring portability and ease of use across any modern browser. This chapter describes the technical aspects that enable the system to fetch, process, and display market data in real time.

4.1 Data Source: Binance API

The application connects to the Binance REST API to fetch candlestick (*kline*) data for BTC/USDT. Each request provides Open, High, Low, Close, and Volume (OHLCV) information for the selected timeframe.

Example request

```
GET https://api.binance.com/api/v3/klines?symbol=BTCUSDT&interval=1m&limit=100
```

Sample response (truncated)

```
[
  [
    1672444800000,   // Open time
    "16850.01",      // Open
    "16860.50",      // High
    "16847.90",      // Low
    "16859.20",      // Close
    "152.123",       // Volume
    1672444859999,   // Close time
    "2560000.45",    // Quote asset volume
    245,             // Number of trades
    "75.324",        // Taker buy base asset volume
    "1250000.50",    // Taker buy quote asset volume
    "0"              // Ignore
  ]
]
```

This JSON response is parsed into arrays of prices and volumes for indicator calculation and chart rendering.

4.2 Front-End Structure

The web application is divided into three main components:

- **HTML:** Semantic structure for charts, controls (timeframe selection, simulation form), and history table.
- **CSS:** Responsive design and styling for readability across devices.
- **JavaScript:** Core logic for API calls, indicator computations, chart updates, and local storage management.

4.3 Indicator Computation

Technical indicators are computed directly in JavaScript. For example:

Simple Moving Average (SMA)

```
function calculateSMA(data, period = 10) {
  return data.map((_, index, arr) => {
    if (index < period) return null;
    const slice = arr.slice(index - period, index);
    const sum = slice.reduce((a, b) => a + b, 0);
    return sum / period;
  });
}
```

Formulas

$$SMA_t = \frac{P_t + P_{t-1} + \cdots + P_{t-n+1}}{n} \tag{4.1}$$

$$EMA_t = P_t \cdot \alpha + EMA_{t-1} \cdot (1 - \alpha), \quad \alpha = \frac{2}{n+1} \tag{4.2}$$

$$RSI = 100 - \frac{100}{1 + RS}, \quad RS = \frac{\text{Avg Gain}}{\text{Avg Loss}} \tag{4.3}$$

$$MACD = EMA_{12} - EMA_{26} \tag{4.4}$$

$$\text{Signal Line} = EMA_9(MACD) \tag{4.5}$$

$$\text{Histogram} = MACD - \text{Signal Line} \tag{4.6}$$

These computations are optimized for real-time updates whenever new data is fetched.

4.4 Visualization with Chart.js

- **Main chart:** candlestick or line chart with SMA and EMA overlays.
- **Secondary panels:** RSI oscillator, MACD histogram with signal line, and volume bars.
- **Interactivity:** tooltips, zoom, and timeframe selection.
- Charts auto-refresh periodically (e.g., every 10 seconds).

4.5 Signal Simulation

- Users can define a potential entry point, Stop-Loss (SL), and Take-Profit (TP).
- The system calculates the risk/reward ratio (R/R) and potential profit/loss.
- Simulations are stored in LocalStorage as JSON objects.

Example simulation record

```
{
  "date": "2025-09-05 14:20",
  "entry": 27250.0,
  "stopLoss": 27000.0,
  "takeProfit": 28000.0,
  "rrRatio": 3.0
}
```

4.6 Update and Refresh Mechanism

- JavaScript `setInterval()` triggers API fetches periodically.
- New data is appended to arrays, indicators are recalculated, and charts are re-rendered.
- Old data points are truncated to avoid performance issues.

Chapter 5

Results

The implementation of the **BTC Technical Analysis Web App** successfully demonstrated the feasibility of creating a browser-based platform for real-time cryptocurrency monitoring and educational trading analysis. The results can be evaluated across three main dimensions: chart visualization, indicator computation, and signal simulation.

5.1 Chart Visualization

The application produced fully interactive charts that integrate price action with technical indicators and volume data.

- **Main Price Chart:** Accurate rendering of BTC/USDT candlesticks in real time with SMA(10) and EMA(10) overlays.
- **Indicator Panels:** RSI graph highlighting overbought/oversold zones, MACD line with signal line and histogram for momentum shifts.
- **Volume:** Bars aligned with candlesticks to show correlation between price movement and market activity.

5.2 Indicator Computation

The JavaScript-based computation of indicators produced reliable values consistent with external trading platforms.

- SMA and EMA values tracked closely with reference values from Binance and TradingView.
- RSI correctly identified periods of strong upward or downward momentum.
- MACD histogram effectively highlighted bullish and bearish crossovers.

5.3 Signal Simulation

The simulation feature allowed users to create hypothetical trade scenarios.

- Inputs: entry price, Stop-Loss (SL), Take-Profit (TP).
- Outputs: risk/reward ratio (R/R) and potential gain/loss percentage.
- Simulations saved in LocalStorage as JSON records.

Example Simulation Record

```
{
  "date": "2025-09-05 14:20",
  "entry": 27250.0,
  "stopLoss": 27000.0,
  "takeProfit": 28000.0,
  "rrRatio": 3.0
}
```

5.4 Overall Performance

- **Responsiveness:** Charts updated seamlessly every 10 seconds without requiring manual refresh.
- **Accessibility:** The app ran smoothly on Chrome, Firefox, and Edge without additional installations.
- **Educational Value:** Users could explore timeframes, indicators, and trade simulations in a safe, risk-free environment.

Chapter 6

Discussion

The development and deployment of the **BTC Technical Analysis Web App** highlight both the strengths and limitations of browser-based financial analysis tools. This chapter reflects on the project's contributions, challenges, and areas for future improvement.

6.1 Benefits

- **Lightweight and Accessible:** The application runs entirely in the browser with no backend infrastructure, making it universally accessible on any modern device.
- **Educational Value:** By calculating and visualizing SMA, EMA, RSI, and MACD, the system reinforces technical analysis concepts while allowing users to experiment with simulated trades.
- **Real-Time Responsiveness:** Continuous integration with the Binance API provides near real-time monitoring of BTC price movements.
- **Extensibility:** Modular architecture enables future additions such as new indicators, assets, or export features with minimal restructuring.

6.2 Limitations

- **Single Asset Focus:** The current version only analyzes BTC/USDT, limiting its scope for multi-asset traders.
- **Client-Side Storage:** Simulations are stored locally via LocalStorage, restricting persistence to a single device and browser.
- **Lack of Automated Alerts:** Signals are only visualized; there are no push notifications or external alerts.

- **Performance Constraints:** Since all computations are client-side, analyzing multiple assets or very large datasets could impact performance.

6.3 Opportunities for Improvement

- **Multi-Asset Expansion:** Extend analysis beyond BTC to other cryptocurrencies such as ETH, BNB, and SOL.
- **Cloud Storage and Authentication:** Store simulations in a backend database to support cross-device access and user profiles.
- **Export Features:** Allow exporting simulation logs and charts to CSV or PDF for external reporting.
- **Advanced Indicators:** Add tools like Bollinger Bands, Ichimoku Cloud, or Stochastic Oscillator for more comprehensive analysis.
- **Automated Alerts:** Integrate push notifications or Telegram alerts to notify users when specific conditions are met.

Chapter 7

Conclusion

The **BTC Technical Analysis Web App** successfully demonstrated the feasibility of implementing a fully browser-based system for cryptocurrency analysis using only client-side technologies. By integrating real-time data from the Binance API, calculating widely recognized technical indicators, and providing simulation and history logging features, the project achieved its objective of creating an educational and accessible trading analysis platform.

One of the key contributions of this work is its balance between simplicity and functionality. While it does not aim to replace professional trading platforms, it provides users with the essential tools to understand market dynamics, experiment with technical indicators, and reflect on trading strategies through simulated scenarios. The project highlights how core concepts such as SMA, EMA, RSI, and MACD can be implemented in JavaScript and effectively visualized through interactive charts.

The development process also revealed important limitations, including the reliance on local storage, the single-asset focus on BTC, and the absence of automated alerting mechanisms. Nevertheless, these constraints provided valuable insights into potential extensions, such as multi-asset support, cloud-based storage, and real-time notifications.

Ultimately, this project not only stands as a functional educational tool but also served as the inspiration for the subsequent **Crypto Signal Generator in Python**. The lessons learned from handling real-time data, computing indicators, and managing user simulations laid the foundation for a more advanced, multi-asset system that integrates Telegram alerts and supports broader trading workflows.

Chapter 8

References / Appendix

8.1 References

- Binance Developers. *Official API Documentation.* Available at: `https://binance-docs.github.io/apidocs/spot/en/`
- Chart.js Contributors. *Chart.js Open-Source Charting Library.* Available at: `https://www.chartjs.org/docs/latest/`
- Wilder, J. W. *New Concepts in Technical Trading Systems.* Trend Research, 1978.
- Murphy, J. J. *Technical Analysis of the Financial Markets.* New York Institute of Finance, 1999.

8.2 Key Indicator Formulas

Simple Moving Average (SMA)

$$SMA_t = \frac{P_t + P_{t-1} + \cdots + P_{t-n+1}}{n} \tag{8.1}$$

where P = closing price, n = number of periods.

Exponential Moving Average (EMA)

$$EMA_t = P_t \cdot \alpha + EMA_{t-1} \cdot (1 - \alpha) \tag{8.2}$$

with $\alpha = \frac{2}{n+1}$.

Relative Strength Index (RSI)

$$RSI = 100 - \left(\frac{100}{1 + RS}\right) \tag{8.3}$$

where $RS = \frac{\text{Average Gain}}{\text{Average Loss}}$ over n periods.

Moving Average Convergence Divergence (MACD)

$$MACD = EMA_{12} - EMA_{26} \tag{8.4}$$

$$Signal\ Line = EMA_9(MACD) \tag{8.5}$$

$$Histogram = MACD - Signal\ Line \tag{8.6}$$

8.3 Sample Binance JSON Structure

Example response structure for OHLCV data retrieved from the Binance API:

```
[
  [
    1672444800000,    // Open time
    "16850.01",       // Open
    "16860.50",       // High
    "16847.90",       // Low
    "16859.20",       // Close
    "152.123",        // Volume
    1672444859999,    // Close time
    "2560000.45",     // Quote asset volume
    245,              // Number of trades
    "75.324",         // Taker buy base asset volume
    "1250000.50",     // Taker buy quote asset volume
    "0"               // Ignore
  ]
]
```