Introduction

Cryptocurrency has become a widely followed and data-driven financial ecosystem, yet many newcomers find it challenging to interpret raw market information. Crypto Tracker for Beginners is a web application developed using the Streamlit framework, designed specifically to help new users explore cryptocurrency trends in an accessible and visually engaging way.

The primary purpose of this app is to display real-time and historical cryptocurrency data using interactive charts, tables, and visual aids. It retrieves data from the CoinGecko API, a free and publicly available service that provides detailed information on thousands of cryptocurrencies.

This application supports beginner users by presenting only essential data—such as price changes, market capitalization, and historical price charts—in a streamlined and user-friendly interface. The inclusion of features like searchable coin lists, customizable chart views, and an interactive map further enhances user engagement and encourages exploration.

Crypto Tracker not only meets the technical and UI/UX requirements of the project but also adheres to human-computer interaction (HCI) principles, with a focus on simplicity, feedback, visibility, and consistency. The result is an intuitive tool that provides a smooth on-ramp for anyone new to the world of crypto markets.

Usability Goals

The Crypto Tracker for Beginners application was designed with several key usability goals in mind, based on foundational Human-Computer Interaction (HCI) principles. These goals directly influenced the structure, layout, and interactivity of the application to ensure a positive and efficient user experience.

1. Simplicity

The app focuses on displaying only essential information relevant to beginner users, such as price, market cap, and 24-hour price changes. Complex metrics, excessive tabs, or overly technical language were intentionally avoided to reduce cognitive load.

2. Learnability

New users can quickly understand and use the interface. Controls like a text input for search, dropdowns, sliders, and buttons follow common design patterns, making them intuitive even for first-time visitors.

3. Efficiency

Users can easily search for a cryptocurrency by name, view its current and historical data, and change chart types or time ranges—all with minimal effort and without reloading the page unnecessarily. Caching and a reload button improve performance and control.

4. Feedback

The app gives clear, contextual feedback to the user. Status messages such as st.success(), st.warning(), and st.info() confirm when data is loaded, missing, or unavailable due to API limits.

5. Visual Clarity

Data is presented using readable fonts, logical layout sections, and emojis to provide visual anchors for sections. Users can also toggle raw data visibility for transparency.

6. Accessibility

A sidebar is used to group configuration controls, minimizing clutter and improving navigation. Input fields, sliders, and buttons have clear labels. Although accessibility tools like screen readers or dark mode toggles weren't implemented, the app's clean layout and simplicity benefit usability across a broad audience.

Design Process

The design and development of Crypto Tracker for Beginners followed an iterative and goal-driven process, emphasizing both technical functionality and user-centered design.

Step 1: Topic and API Selection

The project began with exploration of APIs listed on https://github.com/public-apis/public-apis. After reviewing options, the CoinGecko API was chosen due to:

- Its extensive cryptocurrency data
- Public accessibility without requiring authentication
- Clear documentation
- This API aligned well with the project's goals of visualizing financial trends for new users.

Step 2: Defining the User

The target user group was defined as:

- Beginners in cryptocurrency
- Users seeking simple, readable market insights
- People interested in real-time or short-term crypto trends

This informed the app's minimalistic layout, choice of metrics, and use of plain language.

Step 3: Interface Planning

Initial planning focused on structuring the app into clear sections:

- Sidebar for controls (search, selection)
- Main view for current stats, historical charts, and maps
- Additional controls (reload, raw data checkbox)
- The layout was chosen for readability, with future extensibility in mind.

Step 4: Development and Prototyping

Using Streamlit, the app was developed incrementally:

- Fetching and displaying current market data
- Implementing historical price charts with chart-type toggles
- Adding map visualizations and interactivity widgets
- Improving with user search filtering, chart customization, and feedback messages
- Frequent testing ensured each feature was functional and beginner-friendly.

Step 5: Error Handling and Performance

To deal with API rate limits and missing data, caching was introduced using st.cache_data. The app provides visible feedback if data is unavailable or limited, maintaining transparency and user control.

API Integration

This project integrates live cryptocurrency data using the CoinGecko API, a public and free-to-use RESTful API that provides real-time and historical cryptocurrency market data. The integration was done entirely through Python using the requests library.

CoinGecko API Endpoints Used

Purpose	Endpoint
Fetch list of all supported coins	`/api/v3/coins/list`
Get current market stats	`/api/v3/coins/markets`
Get historical price data	`/api/v3/coins/{id}/market_chart`

Each of these endpoints returns JSON data, which is parsed into Pandas DataFrames for display and plotting.

Caching and Performance

To avoid exceeding CoinGecko's free-tier rate limits, the app uses:

- @st.cache_data(ttl=300) to cache responses for 5 minutes
- A time.sleep(1.5) delay between certain API calls

This caching mechanism improves app responsiveness and prevents the user from triggering rate limits through rapid interactions.

Data Parsing and Transformation

- Coin metadata (name, id) is stored and filtered based on a search input field.
- Market data is loaded into a DataFrame, with key fields (name, symbol, current_price, market_cap, and price_change_percentage_24h) extracted for tabular display.
- Historical price data is converted into timestamped DataFrames to support plotting via st.line_chart, st.area_chart, or st.bar_chart.

These transformations allow raw API data to be meaningfully presented to users in visual formats.

Error Handling

The app detects and handles:

- Missing data fields in responses
- Invalid user input (e.g., coin not found)
- Empty responses or rate-limiting (429 errors)
- API downtime or malformed responses

Informative status messages (st.warning, st.info, etc.) help users understand what's happening.

Interactive Widgets

To enhance usability and support a smooth user experience, the app includes a variety of Streamlit widgets that allow users to control and personalize the data they see. These widgets were selected for their clarity, simplicity, and alignment with the app's target users: cryptocurrency beginners.

1. **st.text_input("🔍 Search coin name")**

- Allows users to filter the list of available cryptocurrencies by typing a name or partial name.
- Reduces clutter in the dropdown and helps users quickly find specific coins.

2. **st.selectbox("Select a Cryptocurrency", ...)**

- Used to select a coin from the filtered list.
- Dropdown behavior ensures the user can only pick from valid entries.

3. **st.slider("Select number of days to display", ...)**

- Lets users select how much historical price data they want to view (1–90 days).
- Offers visual, intuitive control over the data time span.

4. **st.radio("Select chart type", ...)**

- Enables users to switch between Line Chart, Area Chart, and Bar Chart visualizations.
- Helps users interpret data in a format they find most comfortable.

5. **st.button("🔄 Reload Data")**

- Forces a manual refresh of cached data.
- Useful if the user suspects data is outdated or rate limits have passed.

6. **st.checkbox("Show raw API response")**

- Optional toggle that reveals the raw JSON response from the API.

- Adds transparency for users interested in the underlying data or debugging.

<u>HCI Design Principles</u>

1. Visibility of System Status

- The app provides immediate feedback through st.success(), st.warning(), and st.info() messages.
- Users are informed when data loads successfully, when the API response is empty, or when rate limits are exceeded.

2. Match Between System and Real World

- Uses plain, beginner-friendly language (e.g., "Select a Cryptocurrency" instead of technical jargon).
- Presents data in a familiar format (tables, charts, and maps) that users associate with financial tracking apps.

3. User Control and Freedom

- Users can search, select, and filter cryptocurrencies at any time without being locked into workflows.
- A reload button gives users control over when to refresh data.

4. Consistency and Standards

- All input controls are located in the sidebar, following a consistent and expected UI pattern.
- Fonts, emoji-enhanced headers, and visual cues are used uniformly across sections.

5. Error Prevention and Handling

- Caching reduces unnecessary API calls, lowering the risk of hitting rate limits.
-

- Search input filters the coin list, reducing the risk of invalid selections.
- Error messages are displayed clearly when data cannot be fetched.

6. Aesthetic and Minimalist Design

- The layout is clean and logically organized:
- Sidebar for controls
- Main area for charts, data, and maps
- Only essential data is shown; unnecessary or advanced metrics are excluded to reduce clutter.

7. Recognition Rather Than Recall

- Searchable dropdowns, pre-defined chart types, and labeled sliders eliminate the need for users to remember syntax or commands.
- Users always see available choices rather than having to guess or recall inputs.

Conclusion & Reflection

The Crypto Tracker for Beginners project successfully meets the core objectives of the assignment by combining real-world data, interactive visualization, and a beginner-friendly interface through Streamlit. It demonstrates how human-centered design, public APIs, and effective data visualization techniques can work together to create an informative and usable web application.

Through this project, I gained practical experience in:

- Working with public REST APIs
- Handling real-time and historical data
- Structuring responsive UIs using Streamlit
- Applying HCI principles to interface design

Managing API rate limits and caching in a web app context

Challenges Faced

- Rate limits imposed by CoinGecko required implementing caching and wait logic.
- Some coins lacked expected data fields, so conditional checks and fallback messages were added.
- Choosing which metrics to present involved prioritizing simplicity over technical depth, to keep the interface approachable.

Future Improvements

- If more time or resources were available, future iterations could include:
- User authentication to save watchlists
- Dark mode or mobile optimization
- Additional charts for volume or volatility
- Live price updates without needing a reload

Ultimately, this project not only demonstrates technical proficiency but also showcases how thoughtful design choices can make complex data approachable for new users. It serves as a solid foundation for future enhancements or full-scale deployment.