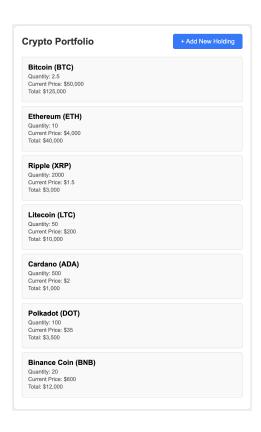
Crypto Portfolio Tracker (React)

Objective:

Develop a simple crypto portfolio tracker where users can add, view, and manage their cryptocurrency holdings.



Requirements:

1. Project Setup:

- Create a new React project.
- Use the latest stable version of React.

2. User Interface:

Home Screen:

- Display a list of all cryptocurrencies in the user's portfolio.
- Each item should show the cryptocurrency name, symbol, quantity held, current price, and total value.

• Add/Edit Holding Screen:

 Form to add or edit a holding with fields for cryptocurrency name, symbol, and quantity.

• Details Screen:

 Show detailed information about the cryptocurrency, including historical price charts.

3. Task Management:

- Allow users to add a new cryptocurrency to their portfolio.
- Allow users to edit existing holdings.
- Allow users to delete holdings.

4. Data Persistence:

• Use the local storage/ index database to store the portfolio.

5. State Management:

 Use a state management solution (e.g., Redux, zustand, mobx) to manage the state of the app.

6. User Experience:

- Implement basic input validation (e.g., quantity should be a positive number).
- Show appropriate error messages or feedback (e.g., snackbar, toast).
- Ensure the app is responsive and works well on different screen sizes.

7. Optional Enhancements:

 Fetch real-time price data from a public API (e.g., CoinGecko, CryptoCompare).

- Display a chart showing the price history of each cryptocurrency.
- Allow users to filter and sort their holdings by different criteria (e.g., value, name).

8. Documentation:

- Provide a README file with instructions on how to run the app.
- Include comments in the code to explain key parts of the implementation.

Evaluation Criteria:

1. Code Quality:

- Clean, readable, and well-organized code.
- Use of best practices and design patterns.

2. Functionality:

- The app should meet all the specified requirements.
- Smooth and bug-free user experience.

3. User Interface:

- Aesthetic and user-friendly design.
- Consistency in UI components and layout.

4. Data Management:

- Efficient and correct implementation of data persistence.
- Proper handling of state and state changes.

5. **Optional Features:**

Implementation of any optional enhancements will be considered a plus.

Submission:

- Provide the complete source code in a GitHub repository.
- Include the README file with setup instructions.
- Mention any additional features or enhancements implemented.