CRYPTOCURRENCY DASHBOARD

INTRODUCTION:

A crypto currency dashboard that displays historical price data over the past five years is a powerful tool for investors seeking a comprehensive understanding of market dynamics. This feature-rich interface offers users a detailed historical perspective on the performance of various crypto currencies, enabling insightful analysis and informed decision-making. Through visually intuitive charts and graphs, the dashboard allows for effective comparisons of multiple crypto currencies, aiding in the identification of top performers and overall market trends. Users can customize timeframes for a more granular examination of price movements, facilitating in-depth volatility analysis and risk assessment. This historical data not only supports investors in making data-driven decisions but also assists in recognizing recurring patterns and cycles. Beyond its role in optimizing crypto currency portfolios, the dashboard serves as an educational resource, empowering users to grasp the evolving nature of crypto currency markets and the nuanced factors shaping price movements over an extended period.

Team Members & Their Role:

TAMIL SELVAN .S[ncas2225sz1078@ncas.in] - Oversees development, Ensures deadlines are met, Manages collaboration, Designs and implements UI, Works with CSS frameworks (Tailwind, Material UI, etc.) Manages global state (Redux API, Context API), Handles API calls and data flow

Praison Rajadurai .P [ncas2225sz1053@ncas.in] - Frontend Developer (Component Development), builds reusable components, Implements dynamic UI interactions, Conducts audits using Lighthouse & Webpage Test, Optimizes React performance (lazy loading, memorization)

Shyam .S [ncas2225sz1096@ncas.in] - Writes unit tests (Jest, React Testing Library), Performs manual testing Accessibility & Performance Expert (Secondary Role for UI Developer), Ensures UI meets WCAG accessibility standards

Siva. M [ncas2225sz1061@ncas.in] - DevOps & Deployment Specialist (Secondary Role for a Developer), Manages CI/CD pipelines (GitHub Actions, Vercel, Netlify), Handles hosting & deployment, Ensures automated testing is integrated

Robert Sam Raj .R[ncas2225sz1039@ncas.in] - Documentation & Knowledge Manager (Good for a Team Member Who Likes Writing & Organizing), Maintains project documentation. Creates a Wiki or Notion page for the team. Ensures team members follow coding conventions

Project Overview:

Purpose: A crypto currency dashboard serves as a centralized platform to manage, analyze, and monitor various aspects of digital currency investments and market trends. It helps users make informed decisions and track their portfolio effectively.

Features:

1. Real-time price tracking and charts.
2. Portfolio management with profit/loss analysis.
3. Market news and updates integration.
4. Multi-crypto currency support.
5. Wallet and exchange connectivity.
6. Alerts for price changes and trends.
7. Security features like two-factor authentication.

CRYPTOCURRENCY DASHBOARD

8. Transaction history and tracking

Architecture:

- Built using frameworks like React, Angular, or Vue.js.
- Displays real-time data such as crypto currency prices, portfolio stats, and market trends.
- Provides user-friendly navigation and interactive charts.
- Developed with technologies like Node.js.
- Handles API requests, user authentication, and data processing.
- Integrates with third-party APIs for fetching crypto currency data.
- Fetch real-time crypto currency prices and market data from services like Coin Gecko or Coin Market Cap.
- Wallet and exchange integration APIs for seamless transactions

Setup Instruction:

Here are the key prerequisites for developing a frontend application using

React.js: ✓ Node.js and npm:

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the local environment. It provides a scalable and efficient platform for building network applications. Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side

✓ React.js:

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

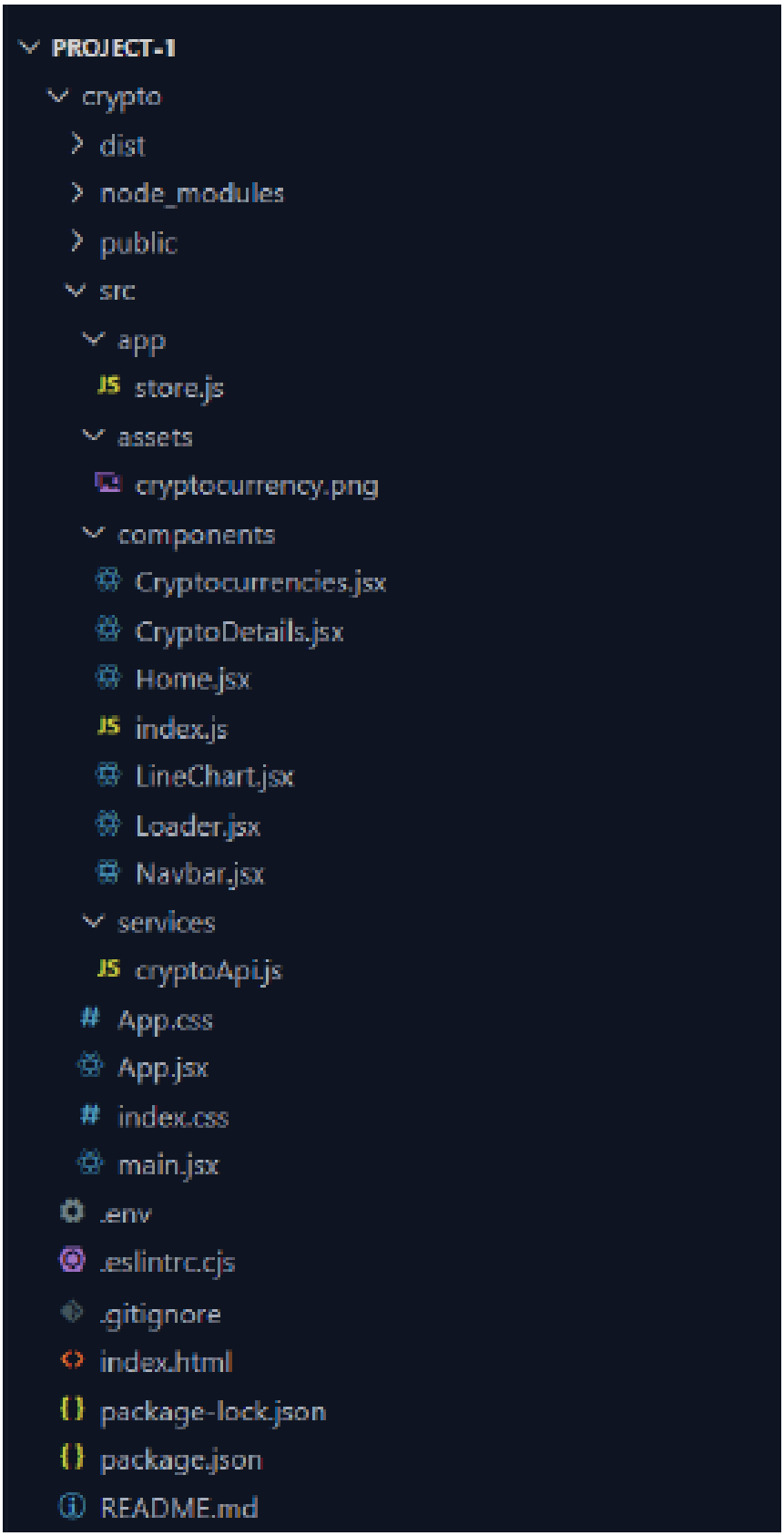
- Create a new React app: npx create-react-app my-react-app Replace my-react-app with your preferred project name.
- Navigate to the project directory: cd my-react-app
- Running the React App: With the React app created, you can now start the development server and see your React application in action.
- Start the development server: npm start

This command launches the development server, and you can access your React app at http://localhost:5173in your web browser.

✓ HTML, CSS, and JavaScript: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

CRYPTOCURRENCY DASHBOARD

Folder Structure:



CRYPTOCURRENCY DASHBOARD

Top-Level Directories and Files:

- **Folders**: `crypto`
- **Files**: `.env`, `.eslintrc.cjs`, `.gitignore`, `index.html`, `package-lock.json`, `package.json`, `README.md`

Inside the `crypto` Folder:

1. **Subfolders**:
    - `dist.`
    - `node modules`
    - `public`
    - `src`
        - **Subfolders**:
            - `App`: Contains `store.js`.
            - `Assets`: Includes an image file named `cryptocurrency.png`.
            - `Components`: Holds React components like `Cryptocurrencies.jsx`, `CryptoDetails.jsx`, `Home.jsx`, `LineChart.jsx`, `Loader.jsx`, `Navbar.jsx`, and the `index.js`.
            - `Services`: Contains `cryptoApis.js`.
        - **Standalone Files**:
            - `App.css`, `App.jsx`, `index.css`, and `main.jsx`.

Running the Application:

1. **Navigate to the Project Directory**: Open a terminal or command prompt and go to your project folder. For example:

   ```
   cd path-to-your-project-folder
   ```

2. **Install Dependencies**: Run the following command to install the required node modules:

   ```
   npm install
   ```

3. **Start the Development Server**: Launch your React app using:

   ```
   npm start or npm run dev.
   ```

   This will start the app on a development server, usually accessible at `http://localhost:3000` or `http://localhost:5173` in your browser.

4. **Access the Application**: Open your browser and visit http://localhost:3000 or `http://localhost:5173` to see your app in action.

State Management

Use **Redux Toolkit** for state management, as it simplifies state handling and integrates seamlessly with React.

CRYPTOCURRENCY DASHBOARD

State Management Workflow

1. **Global State**:
   - Centralize data (e.g., cryptocurrency stats, user preferences, and API responses).
   - Handle application-wide state through Redux or Context API.
2. **Slices for Modular State**:
   - Divide state into slices, such as:
     - `cryptoSlice`: Stores cryptocurrency data.
     - `userSlice`: Manages user-related data (e.g., settings or authentication).
     - `uiSlice`: Tracks UI state (e.g., loading, errors).
3. **Async Thunks for API Calls**:
   - Use Redux Toolkit's `createAsyncThunk` to handle API requests (e.g., fetching cryptocurrency prices or details).
4. **Dispatch and Select**:
   - Dispatch actions to update the state (e.g., `dispatch(fetchCryptoData())`).
   - Use selectors to extract state for specific components (e.g., `useSelector((state) => state.crypto.data)`).

File Structure

Here's how the state management files can be organized:

```
src/
├── app/
│   ├── store.js          # Configures the Redux store
├── features/
│   ├── crypto/
│   │   ├── cryptoSlice.js  # Handles cryptocurrency-related state
│   │   ├── cryptoApis.js   # API calls for fetching cryptocurrency data
│   ├── user/
│   │   ├── userSlice.js    # Manages user-specific state
├── components/            # React components
├── services/              # Utility serv
```

User Interface:

To design a user interface (UI) for your cryptocurrency dashboard website project, you should aim for a modern, intuitive, and responsive layout. Here's a structured plan:

1. Design Principles

- **Clarity**: Ensure information is presented clearly.
- **Responsiveness**: The UI should adapt to different screen sizes.
- **Consistency**: Use a uniform color scheme, typography, and layout.
- **User-Centric**: Highlight frequently-used features prominently.

2. Layout Components

1. **Header/Navbar**:
   - Fixed navigation bar for easy access to key sections.
   - Include links like "Home," "Cryptocurrencies," "Exchanges," and "About."

CRYPTOCURRENCY DASHBOARD

    - Search bar for quickly locating cryptocurrencies.
2. **Dashboard**:
    - Display key statistics, such as:
        - Total cryptocurrencies.
        - Market capitalization.
        - Top gainers/losers.
    - Use cards for visual distinction.
3. **Cryptocurrency List**:
    - A table or grid showing:
        - Cryptocurrency name, symbol, logo.
        - Price, 24-hour change (%), and market cap.
        - Sorting and filtering options (e.g., by name, price).
4. **Details Page**:
    - For specific cryptocurrencies:
        - Price charts (line chart showing historical data).
        - Key metrics (e.g., supply, volume, rank).
        - News feed about the selected cryptocurrency.
5. **Footer**:
    - Include links like "Contact Us," "Privacy Policy," and "Terms of Service."
    - Social media icons.

3. Tools and Technologies

- **React Components**: Break down the UI into reusable pieces.
- **CSS Frameworks**: Use libraries like Tailwind CSS or Bootstrap for styling.
- **Charting Library**: Integrate libraries like Chart.js or Recharts for visualizations.
- **Responsive Design**: Use CSS media queries and flexbox/grid layout.

4. Example Structure

Here’s a potential folder structure for the UI:

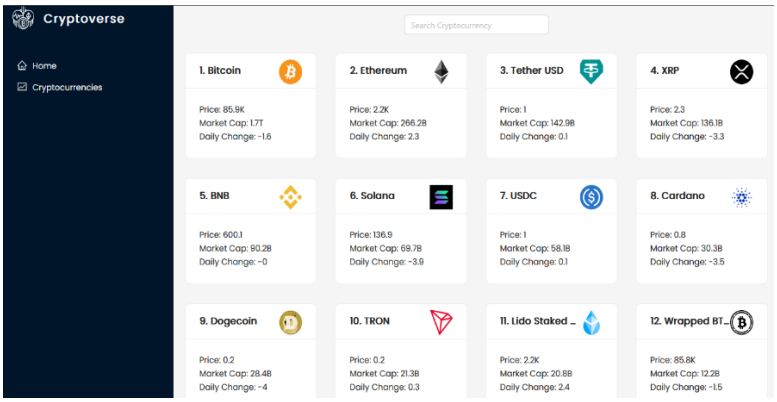
```
src/
├── components/
│   ├── Navbar.jsx          # Navigation bar
│   ├── Dashboard.jsx       # Main dashboard
│   ├── CryptoTable.jsx     # Cryptocurrency list
│   ├── CryptoDetails.jsx   # Detailed view of a cryptocurrency
│   ├── Footer.jsx          # Footer component
│   ├── LineChart.jsx       # Chart component
```

5. Wireframe Example

If visualized:

- **Top Navbar**: Fixed at the top with links and a search bar.
- **Main Section**: Dashboard stats at the top, followed by the cryptocurrency table.

CRYPTOCURRENCY DASHBOARD



Styling:

For styling cryptocurrency dashboard, you can achieve a clean and modern look by using CSS or popular frameworks like Tailwind CSS or Bootstrap. Here's how you can approach styling for various components:

1. Styling Frameworks

- **Tailwind CSS**: Provides utility-first classes for rapid styling.
- **Bootstrap**: Offers pre-designed components like cards, tables, and grids.
- **Styled-Components**: If you prefer CSS-in-JS for scoped styles.
- **CSS Modules**: Write CSS specific to each component.

2. Core Styling Principles

- **Typography**: Use clear fonts like Roboto, Open Sans, or Inter. Set a consistent font size and color scheme.
- **Color Palette**: Choose a palette that reflects professionalism, such as:
  - Primary: Blue shades for accents.
  - Secondary: Subtle greys or whites.
  - Alerts: Green for gains, red for losses.
- **Spacing**: Use padding and margins consistently for breathing room between elements.
- **Responsive Design**: Ensure the layout adapts to screen sizes using media queries or framework utilities.

3. Example Styling Code:

CSS

CRYPTOCURRENCY DASHBOARD

```
/* Fonts and Colors */
body {
  font-family: 'Roboto', sans-serif;
  background-color: #f5f5f5;
  color: #333;
  margin: 0;
  padding: 0;
}

/* Buttons */
button {
  background-color: #007bff;
  color: white;
  padding: 0.5rem 1rem;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

button:hover {
  background-color: #0056b3;
}

/* Cards */
.card {
  background: #fff;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  border-radius: 10px;
  padding: 1rem;
  margin: 1rem;
}

/* Charts */
.chart-container {
  padding: 1rem;
  background-color: white;
  border-radius: 10px;
  box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);
}
```

Tailwind CSS Example

Using Tailwind classes, you can quickly style your components directly in JSX:

jsx

```
<div className="bg-gray-100 min-h-screen flex flex-col items-center">
  <header className="w-full bg-blue-600 text-white py-4 text-center">
    <h1 className="text-2xl font-bold">Cryptocurrency Dashboard</h1>
  </header>

  <main className="flex-grow w-full max-w-4xl p-4">
    <div className="grid gap-4 md:grid-cols-2 lg:grid-cols-3">
      <div className="card bg-white shadow-lg p-4 rounded">
        <h2 className="font-semibold text-lg">Bitcoin (BTC)</h2>
        <p className="text-green-600">$45,000</p>
      </div>
      <!-- More cards -->
    </div>
  </main>

  <footer className="w-full bg-gray-800 text-white py-2 text-center">
    <p>&copy; 2025 Crypto Dashboard</p>
```

```
  </footer>
</div>
```

Testing:

1. Unit Testing

- Focus on testing individual components (e.g., React components, utility functions).
- Tools: **Jest** and **React Testing Library**.
- Example:

  javascript

```
import { render, screen } from '@testing-library/react';
import Navbar from './Navbar';

test('renders the Navbar component', () => {
  render(<Navbar />);
  expect(screen.getByText(/Home/)).toBeInTheDocument();
});
```

2. Integration Testing

- Verify the interaction between multiple components (e.g., API calls and data display).
- Tools: **Jest**, **Testing Library**, or **Cypress**.
- Example: Test if data fetched from the API correctly displays in a component:

  javascript

```
import { fetchCryptoData } from './cryptoSlice';

test('fetches and sets cryptocurrency data', async () => {
  const data = await fetchCryptoData();
  expect(data.length).toBeGreaterThan(0);
});
```

3. End-to-End (E2E) Testing

- Simulate user interactions to verify the app behaves as expected.
- Tools: **Cypress**, **Playwright**, or **Selenium**.
- Example:
  - Test navigation and data display:

  javascript

```
describe('Cryptocurrency Dashboard Tests', () => {
  it('Loads homepage and navigates to details page', () => {
    cy.visit('/');
    cy.contains('Bitcoin').click();
    cy.url().should('include', '/cryptocurrencies/bitcoin');
  });
});
```

4. Performance Testing

- Ensure the app loads quickly and runs efficiently.
- Tools: **Lighthouse** (built into Chrome DevTools).
- Check for:
  - Page load times.
  - API response times.
  - Browser rendering performance.

5. Usability Testing

- Collect feedback from real users or testers on:
  - Ease of navigation.
  - Readability of information.
  - Overall user experience.

6. Security Testing

- Ensure sensitive data (e.g., API keys) is secure.
- Check for vulnerabilities like SQL injection or XSS attacks.
- Tools: **OWASP ZAP**, **Postman** (for API security checks).

Future Enhancement:

1. Features Enhancements

- **User Authentication**:
  - Add user login and signup functionality.
  - Enable personalized dashboards for registered users.
- **Watchlist**:
  - Allow users to add cryptocurrencies to a watchlist for quick tracking.
- **Dark Mode**:
  - Implement a theme toggle for dark and light mode to improve user experience.
- **Real-Time Updates**:
  - Use WebSockets or polling to fetch real-time cryptocurrency data.
- **Multi-Language Support**:
  - Add internationalization (i18n) to cater to global users.

2. Data Visualization

- **Advanced Charts**:
  - Use libraries like D3.js or Highcharts for more dynamic and interactive charts.
  - Include candlestick charts for advanced users to analyze trends.
- **Heatmaps**:
  - Display a heatmap showing the performance of top cryptocurrencies.

3. Analytics and Insights

- **AI/ML-Powered Insights**:
  - Use predictive analytics to provide price predictions or trend analysis.

    - Suggest investment strategies based on historical data.
- **Portfolio Management**:
    - Allow users to input their crypto holdings and track portfolio performance.

4. Additional Data Integration

- **News Feed**:
    - Integrate cryptocurrency-related news using APIs like NewsAPI.
    - Display the latest updates and headlines about specific coins.
- **Exchanges Integration**:
    - Add data from various crypto exchanges to show price comparisons.
- **Fiat Currency Conversion**:
    - Enable users to view cryptocurrency values in different fiat currencies.

5. UI/UX Improvements

- **Mobile-Friendly Design**:
    - Ensure the dashboard is fully optimized for mobile and tablet devices.
- **Animations**:
    - Add subtle animations for loading data or interactive elements to enhance user engagement.
- **Customizable Dashboard**:
    - Allow users to rearrange widgets or components on their dashboard.

6. Security Enhancements

- **Two-Factor Authentication**:
    - Add an extra layer of security for user accounts.
- **Secure API Keys**:
    - Use environment variables for API keys and avoid exposing them in the frontend.
- **Data Encryption**:
    - Encrypt sensitive user data like watchlists and portfolio details.

7. Performance Improvements

- **Caching**:
    - Implement caching for API responses to reduce load times.
- **Lazy Loading**:
    - Load only the necessary components and data to optimize performance.
- **CDN for Assets**:
    - Serve images and static files through a content delivery network (CDN) for faster loading.

8. Gamification:

- Introduce user rewards for certain actions (e.g., consistent usage, inviting friends).
- Add badges for milestones (e.g., tracking 50 cryptocurrencies)