**1.Introduction:**

**Project Title**: Cryptoverse: A Cryptocurrency Dashboard.

**Team Members: Mail Id**

DHARSHINI.D - Team leader(Implementation) **dharshinikiki264@gmail.com**

DIVYA.C – Team member1 (Implementation) **divyadks19@gmail.com**

DEVADARSHINI.S - Team member2(Documentation**) shodarshkum11@gmail.com**

DHARSHINI PRIYA.Y- Team member3(Voice over**) prabavathiyoge@gmail.com**

DEVI.P- Team member4(Voice over) **devip11177@gmail.com**

1. **Project Overview**

**Purpose**: Cryptoverse is a cryptocurrency dashboard that provides real-time data on various digital currencies. It allows users to track prices, market trends, and news related to the crypto space.

**Features**

* + Historical market data with interactive charts
  + News and updates from trusted sources
  + Search and filter options for cryptocurrencies
  + User authentication (if applicable)
  + Dark mode support

1. **Architecture**

**Component Structure**:

* + **Dashboard**: Main page showing crypto stats
  + **CryptoList**: List of available cryptocurrencies
  + **CryptoDetails**: Detailed view of a selected cryptocurrency
  + **News**: Latest crypto-related news
  + **Footer**: Additional links and information

**State Management**:

* + Uses **Redux Toolkit** for global state management
  + API data is fetched and stored in Redux store
  + Local states managed with **useState** for UI interactions.

**Routing**: React Router is used for navigation.

1. **Setup Instructions**

**Prerequisites:**

* + Node.js (Latest LTS)
  + NPM/Yarn
  + API keys for market data and news sources

**Installation:**

1.Clone the repositorygit clone[**https://github.com/your-repo/cryptoverse.git**](https://github.com/your-repo/cryptoverse.git)

**2.**Install dependencies**:** npm install

3. Set up environment variables (.env file):

4.Start the development server:

1. **Folder Structure:**

**Client**: The react application follows a structured folder organization for maintainability and scalability.

**Utility**: The /utils folder contains reusable helper functions and custom hooks to simplify development.

**Helper Functions:**

|  |  |
| --- | --- |
| File | Purpose |
| formatCurrency.js | Converts numbers into currency format. |
| dateFormatter.js | Formats timestamps into readable dates. |
| fetchData.js | Handles API requests with error handling. |

**Custom Hooks:**

|  |  |
| --- | --- |
| Hook | Purpose |
| useFetch.js | Handles API calls and manages loading state. |
| useDarkMode.js | Manages theme switching (light/dark mode). |
| useLocalStorage.js | Saves and retrieves data from local storage. |

**7.Running the Application**

* To start the application locally, use the following command:

npm start

* + - **Frontend**: npm start in the client directory.

1. **Component Documentation**

**Key Components**:

CryptoList: Fetches and displays the list of cryptocurrencies.

CryptoDetails: Shows detailed data for a selected cryptocurrency.

News: Fetches and displays crypto-related news

**Reusable Components**:

Card: Used to display individual cryptocurrency data.

SearchBar: Allows users to filter cryptocurrencies .

Loader: Displays while fetching data.

**State Management**

* + **Global State**: Used for shared data across components.

Example:

cryptoSlice.js → Manages cryptocurrency data.

newsSlice.js → Stores latest crypto news.

themeSlice.js → Handles dark/light mode.

* + **Local State**: Used for component-specific data.

Example:

* + SearchBar.js → Manages user input.
  + CryptoDetails.js → Handles tab selection.
  + Provide screenshots or GIFs showcasing different UI features, such as pages, forms, or interactions.

1. **Styling:**

**CSS Frameworks/Libraries**:

* Tailwind CSS → Utility-based styling for fast UI development.
* Styled-Components → Component-level dynamic styling.
* CSS Modules → Scoped styles to prevent conflicts.

**Theming**:

* Light & Dark Mode using CSS variables + Redux.
* User preference stored in local Storage.

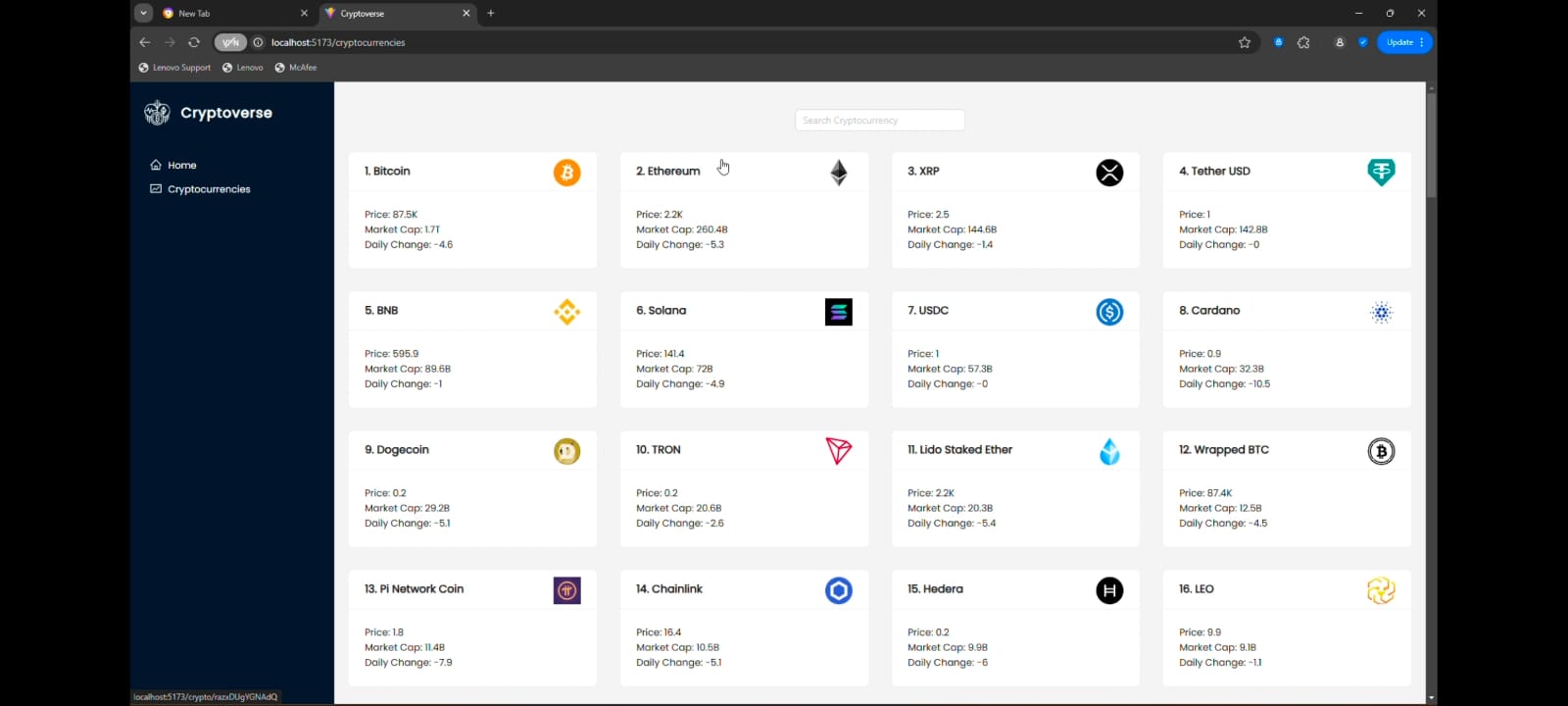
11. **Testing**

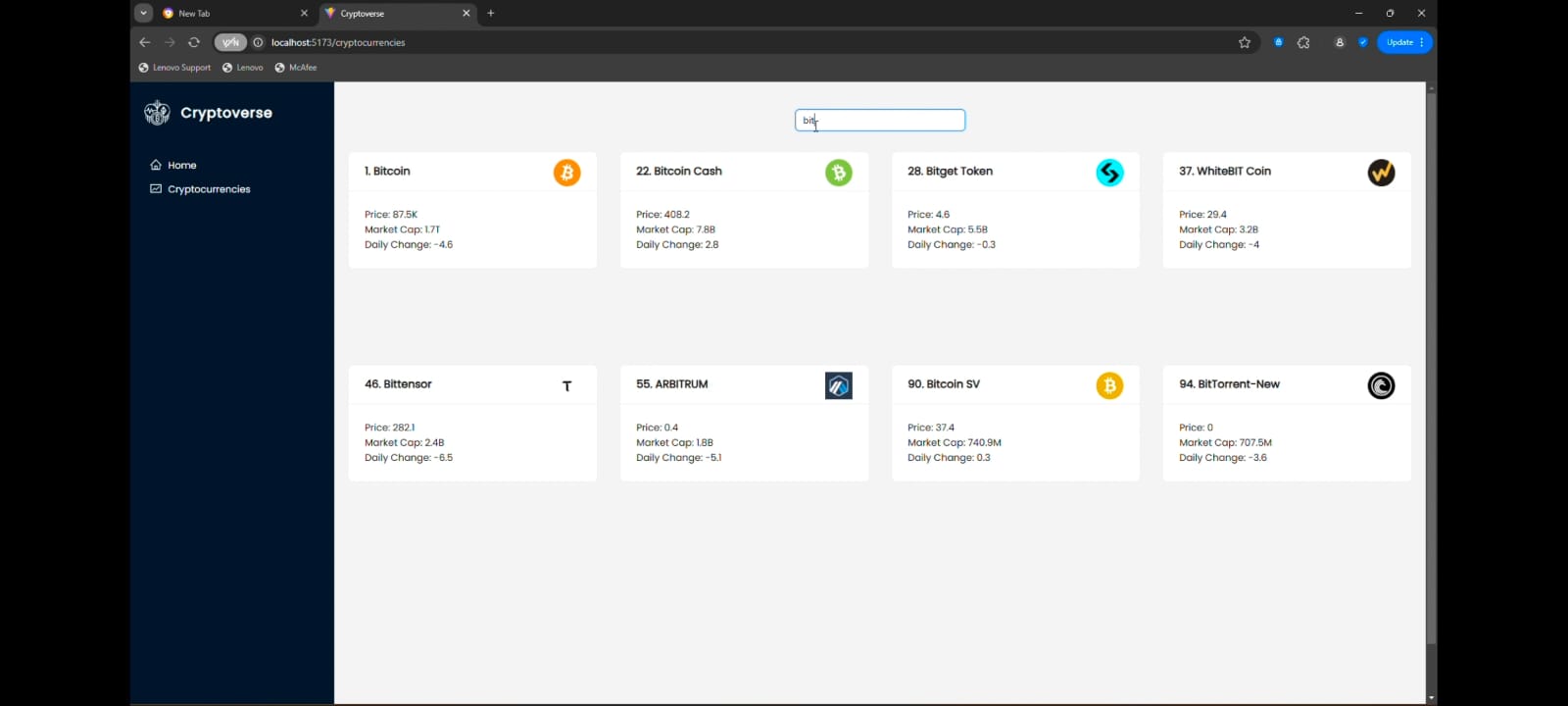
**Testing Strategy:**

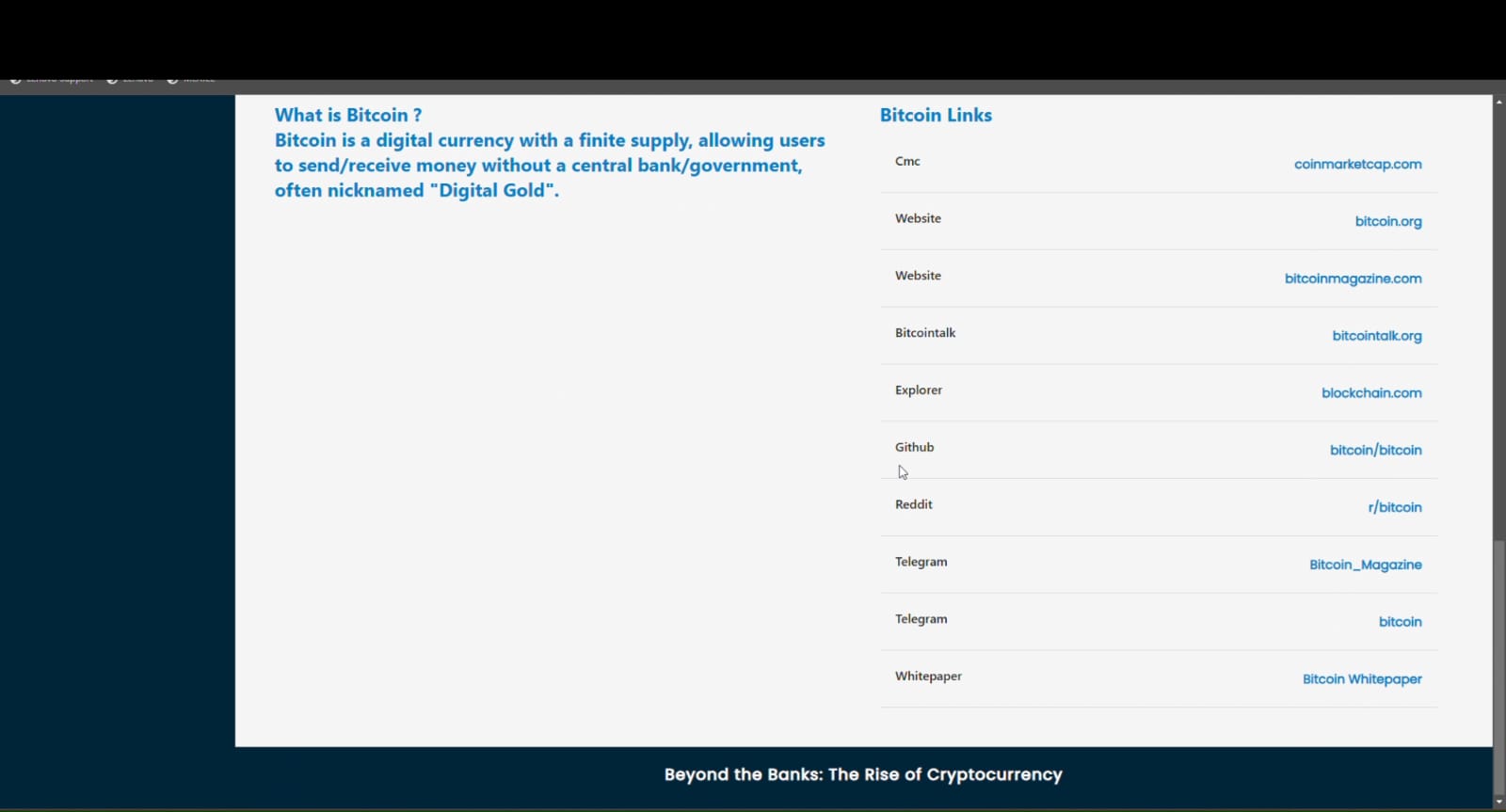
* Unit Tests → Jest + React Testing Library for component testing.
* Integration Tests → Ensures multiple components work together.
* End-to-End (E2E) Tests → Cypress for user interaction testing.

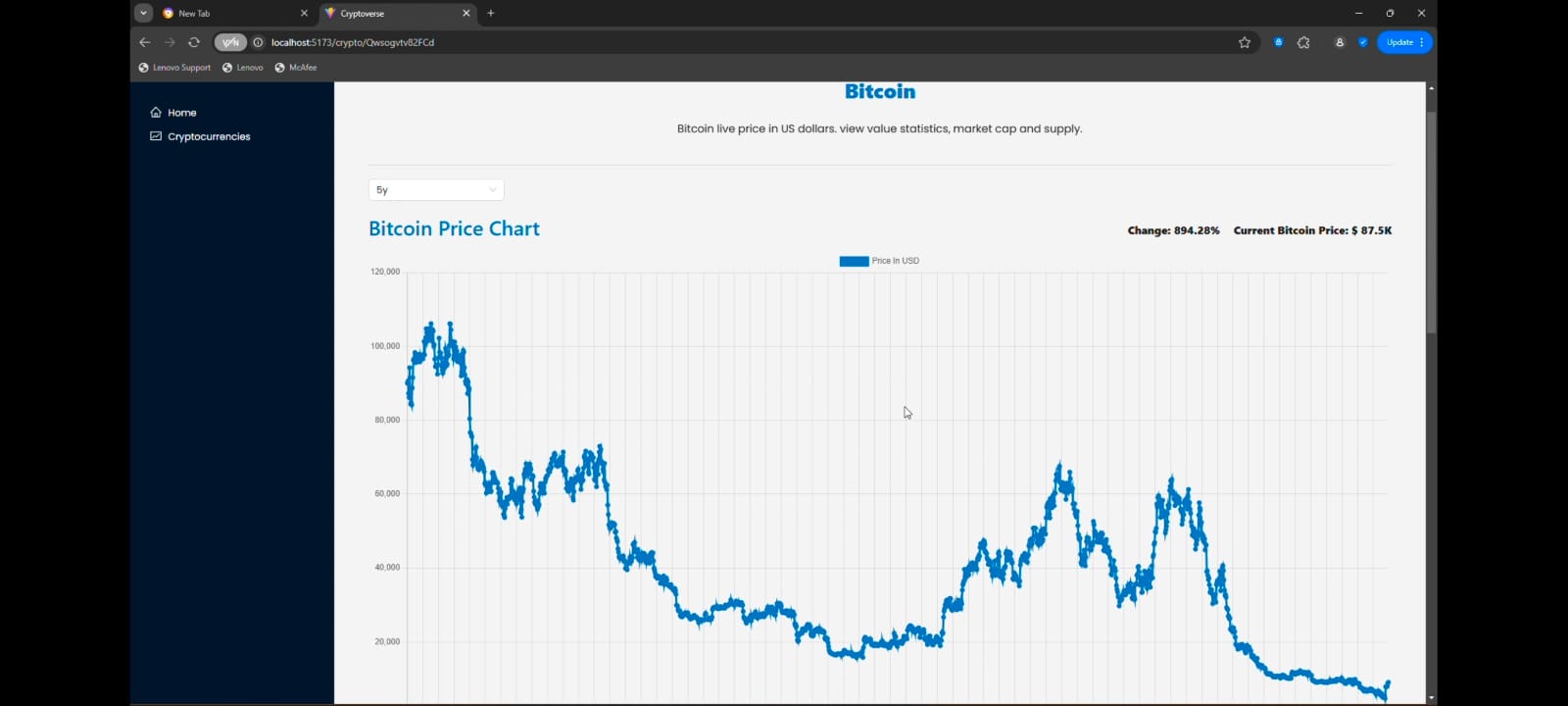
**Code Coverage:**

* Jest Coverage Tool (--coverage) is used to track tested lines of code.
* Thresholds are set to ensure minimum required test coverage.
* Cypress Dashboard monitors E2E test results.

**12. Screenshots: **



****

****

**13.Known Issues:**

* Slow Load Time → API calls delay initial loading. (Fix: Implement caching/lazy loading).
* Dark Mode Flicker → Theme resets briefly on refresh. (Fix: Apply stored theme earlier).
* Search Lag → Delayed response for large datasets. (Fix: Add debounce to search input).
* Mobile UI Issues → Some elements break on small screens. (Fix: Adjust Tailwind breakpoints).
* Infinite Scroll Bug → Stops fetching new data sometimes. (Fix: Improve API error handling)
* Price Update Delay → Crypto prices don’t refresh instantly. (Fix: Use WebSockets for real-time data)

**14.Future Enhancements**

* User authentication and portfolio tracking
* Price alerts and notifications
* Additional data sources for more insights