ANNA ADARSH COLLEGE FOR WOMEN

DEPARTMENT OF BCA

COLLEGE CODE : 1353

TEAM ID : SWTID1741089474

1.INTRODUCTION

PROJECT TITLE : CRYPTOVERSE

TEAM MEMBERS :

PRIYADHARSHINI R (LEADER) - CODING

DHARSHINI SRI P (MEMBER) - DOCUMENTATION

LEKHASRI D (MEMBER) - VOICEOVER

HEPZIBHA SHALOMI B (MEMBER) - CODING

AAFIYA BANU Y (MEMBER) - VIDEO MAKING

2.PROJECT OVERVIEW

PURPOSE :

The Cryptoverse project aims to create a comprehensive platform that provides real-time cryptocurrency market data, price analysis, and insights. The goal is to help users stay informed about the latest trends in the crypto world, track their favorite digital assets, and make data-driven investment

decisions. Cryptoverse will offer features like live price updates, historical data visualization, news aggregation, and portfolio management to enhance user experience and engagement in the dynamic cryptocurrency market.

FEATURES :

1. User Interface (UI) & User Experience (UX)

- Clean, modern, and intuitive design for smooth navigation.
- Responsive layout to ensure accessibility on all devices (desktop, tablet, mobile).
- Dark mode/light mode toggle for enhanced user experience.

2. Real-Time Crypto Dashboard

- Displays live prices, market cap, and trading volume of top cryptocurrencies.
- Search bar for quick coin lookup.
- Sorting and filtering options (by price, market cap, percentage change, etc.).

3. Interactive Charts & Historical Data

- Graphical representation of cryptocurrency price trends over different timeframes (1D, 7D, 1M, 1Y).
- Dynamic charts powered by libraries like Chart.js or Recharts.

4. Cryptocurrency Details Page

- Detailed information on individual cryptocurrencies, including real-time price updates, historical performance, and market data.
- Coin comparison feature to analyze multiple cryptocurrencies.

5. Watchlist & Favorites

- Users can save favorite cryptocurrencies to a personalized watchlist.
- Quick access to frequently monitored assets.

6. Portfolio Tracking

- Allows users to add, edit, and track their crypto holdings.

- Calculates total portfolio value and profit/loss based on market prices.

7. Crypto News Integration

- Aggregated latest news from top crypto sources (CoinDesk, CoinTelegraph, etc.).
- Helps users stay informed about market trends and updates.

8. User Authentication & Personalization

- Secure login/signup for personalized experiences (JWT authentication or Firebase).
- Profile settings for customizing preferences like currency, theme, and notifications.

9. Price Alerts & Notifications

- Users can set price alerts for their selected cryptocurrencies.
- Push/email notifications when the price reaches a specified threshold.

10. Exchange Listings & Buy/Sell Links

- Shows available exchanges where users can trade a selected cryptocurrency.
- Provides direct links to external trading platforms.

3.ARCHITECTURE

COMPONENT STRUCTURE :

The frontend of Cryptoverse follows a modular React component-based architecture, ensuring reusability, scalability, and maintainability. Below is an outline of the major React components and their interactions.

1. Main Component Structure

src/
components/ *(Reusable UI components)*
pages/ *(Different pages of the app)*
context/ *(Global state management, e.g., Context API or Redux)*
services/ *(API calls and data fetching)*
assets/ *(Images, icons, and styles)*

2. Major React Components & Their Interaction

(1) App.js (Main Entry Point)

- Contains BrowserRouter to handle navigation.
- Renders the Navbar, Routes, and Footer.

(2) Navbar.js *(Global Navigation Bar)*

- Displays links for Home, Market, Portfolio, News, etc.
- Includes a search bar to find cryptocurrencies.
- Dark mode toggle (optional).

(3) Footer.js *(Global Footer Section)*

- Provides quick links and social media icons.

(4) Home.js (Dashboard Page)

- Displays real-time market data using CryptoList component.
- Shows top trending cryptocurrencies.
- Contains a search bar and filters.
- Uses CryptoCard for each cryptocurrency.

(5) CryptoList.js (Crypto Market Overview)

- Fetches and displays a list of top cryptocurrencies.
- Uses CryptoCard.js to render each crypto in the list.

(6) CryptoCard.js (Reusable Card Component for Coins)

- Displays name, logo, price, % change, and market cap.
- Clicking it navigates to the detailed view (CryptoDetails.js).

(7) CryptoDetails.js (Detailed Cryptocurrency Page)

- Shows real-time data & price updates.
- Displays an interactive price chart (using Chart.js or Recharts).

- Provides exchange listings, historical performance, and news.
- Uses:

    A) PriceChart.js (for historical data visualization).
    B) ExchangeList.js (to show where the coin is available).

(8) Watchlist.js (User's Favorite Cryptos)

- Displays user's saved cryptocurrencies.
- Uses local storage or database to store selected coins.

(9) Portfolio.js (User Portfolio Tracker)

- Allows users to add, remove, and manage crypto investments.
- Calculates total holdings, profits/losses, and market trends.
- Uses:

    A) PortfolioItem.js (for individual asset tracking).

(10) News.js (Crypto News Aggregation)

- Fetches news from crypto news APIs (CoinGecko, NewsAPI, etc.).
- Uses:

    A) NewsCard.js (to display individual news articles).

(11) Auth.js (User Authentication Page)

- Handles Login / Signup / Logout.
- Uses Firebase or JWT authentication.

(12) Context.js (Global State Management - Context API / Redux)

- Manages authentication state, watchlist, portfolio data, and API fetching state.

3. Component Interaction Flow

1. App.js renders Navbar.js and routes different pages using React Router.
2. Home.js fetches and displays market data using CryptoList.js, which uses CryptoCard.js.
3. Clicking on a coin redirects to CryptoDetails.js, which includes PriceChart.js and ExchangeList.js.

4. Watchlist.js and Portfolio.js fetch user-specific data from Context API / Redux.
5. News.js fetches and displays the latest crypto news via NewsCard.js.
6. Auth.js manages authentication, interacting with Context API or Firebase.

4. Tech Stack

- React.js (Frontend UI)
- React Router (Navigation)
- Context API / Redux (State Management)
- Axios / Fetch API (Data fetching)
- Chart.js / Recharts (Graphs & charts)
- CSS / Tailwind / Styled Components (Styling)

STATE MANAGEMENT :

Cryptoverse uses Context API with useReducer for state management. This approach provides a lightweight and efficient way to manage global state across the application without the complexity of external libraries like Redux.

Key States Managed

🔹 User Authentication – Tracks login/logout status and user info.
🔹 Crypto Market Data – Stores real-time cryptocurrency prices and details.
🔹 User Watchlist – Manages favorite cryptocurrencies saved by the user.
🔹 Portfolio Management – Keeps track of user investments, holdings, and profit/loss calculations.

How It Works?

1. The Context Provider is wrapped around the entire app to make global state accessible.
2. Each component accesses the state using Context API and updates it via dispatch actions.
3. Reducers handle different actions like updating the watchlist, fetching crypto data, and managing the portfolio.

Alternative: Redux Toolkit (For Larger Apps)

For larger applications that need better state management, Redux Toolkit can be used. It provides:

A) Efficient state updates with optimized reducers.
B) Built-in async API handling for real-time crypto data.
C) Better debugging with Redux DevTools.

ROUTING :

Cryptoverse uses React Router for navigation, enabling seamless transitions between different pages while maintaining a single-page application (SPA) experience.

Why React Router?

A) Client-side navigation – No full page reloads, making the app fast and smooth.
B) Dynamic routing – Supports URLs based on crypto details (e.g., /crypto/bitcoin).
C) Nested routes – Helps organize pages efficiently.
D) Protected routes – Can restrict access to certain pages (e.g., portfolio access only for logged-in users).

Routing Structure

1. Home (/) – Displays an overview of cryptocurrencies and market trends.
2. Crypto Details (/crypto/:id) – Shows detailed information about a selected cryptocurrency.
3. Watchlist (/watchlist) – Displays cryptocurrencies saved by the user.
4. Portfolio (/portfolio) – Tracks user investments (requires authentication).
5. Login/Register (/login) – Handles user authentication.
6. Error Page (*) – A fallback page for invalid URLs.

Routing Flow

1. User lands on Home (/) → Sees a list of trending cryptocurrencies.
2. Clicks on a coin → Navigates to /crypto/:id for detailed insights.
3. Wants to track coins → Adds them to the watchlist (/watchlist).
4. Logs in to manage portfolio → Redirects to /portfolio if authenticated.
5. Tries an invalid URL → Redirected to the error page.

Additional Features

A) Dynamic Routes – Uses parameters (e.g., /crypto/bitcoin) to load specific coin details.
B) Protected Routes – Ensures that portfolio access is restricted to logged-in users.
C) Navigation Bar – Allows users to switch between pages easily.

4.SETUP INSTRUCTIONS

PREREQUISITES :

Before setting up Cryptoverse, ensure you have the following dependencies installed:

A) Node.js (LTS Version) – Required to run the React app and manage dependencies.
B) npm or yarn – Package manager for installing required libraries.
C) Git – Version control system (optional but recommended).
D) Code Editor (VS Code Recommended) – For development and debugging.

2. Required Libraries & Dependencies

1. React – Core framework for building the frontend.
2. React Router – Handles navigation and routing within the app.
3. Axios – Fetches real-time cryptocurrency data from APIs.
4. Context API / Redux Toolkit – Manages global state efficiently.
5. Tailwind CSS / Material UI – Styles the UI with modern components.
6. React Icons – Provides icons for a better visual experience.

INSTALLATION :

Follow these steps to set up Cryptoverse on your local machine.

1. Clone the Repository

Download the project from the GitHub repository and navigate to the project folder.

2. Install Dependencies

Ensure Node.js is installed, then install all required project dependencies using the package manager.

3. Configure Environment Variables

Create an environment file and add the necessary API keys for fetching real-time cryptocurrency data.

4. Start the Development Server

Run the project to launch Cryptoverse in the browser for development.

5. Build for Production (Optional)

Generate optimized files for deployment.

6. Deployment (Optional)

The project can be deployed using platforms like Netlify, Vercel, or GitHub Pages.

5.FOLDER STRUCTURE

CLIENT :

The React application is structured in a modular and organized way to ensure scalability and maintainability. Below is an overview of the key folders and their purposes:

1. src/ (Main Source Folder)

This folder contains all the essential files for the React application.

A) components/

- Stores reusable UI components like buttons, headers, footers, and card elements.
- Ensures modularity by keeping commonly used elements in one place.

B) pages/

- Contains different pages of the application, such as Home, Crypto Details, Watchlist, and Portfolio.
- Each page is a separate component, making navigation and routing easier.

C) assets/

- Stores static files like images, icons, and styling assets.
- Helps in maintaining a clean structure by keeping media files separate.

D) context/

- Manages global state using Context API (or Redux if implemented).
- Ensures smooth state management for authentication, crypto data, and user preferences.

E) services/

- Handles API requests and data fetching logic.
- Centralizes API calls to keep components clean and focused.

F) routes/

- Defines and manages application navigation using React Router.
- Helps structure page transitions effectively.

G) styles/

- Stores global styles, custom themes, or Tailwind/MUI configurations.
- Keeps styling separate from component logic.

H) utils/

- Contains utility functions for formatting numbers, dates, or currency conversions.
- Improves code reusability across the application.

2. public/ Folder

- Stores the main index.html file, where the React app is injected.
- Includes static assets like the favicon, logos, and manifest files.

3. package.json & Configuration Files

- Contains project dependencies, scripts, and metadata.
- Ensures smooth project setup and execution.

UTILITIES :

Cryptoverse includes various utility functions, helper classes, and custom hooks to enhance functionality, improve code reusability, and maintain a clean structure.

1. Helper Functions (utils/ folder)

These functions simplify common operations and improve code efficiency.

A) Data Formatting:

- Converts large numbers into a readable format (e.g., 1,000,000 → 1M).
- Formats currency values for different regions.

B) Date & Time Handling:

- Converts UNIX timestamps to human-readable dates.
- Displays historical crypto trends based on time intervals.

C) API Data Processing:

- Filters and structures API responses for better readability.
- Converts percentage changes into user-friendly labels (positive/negative).

2. Custom Hooks (hooks/ folder)

Custom React hooks improve state and effect management while keeping components clean.

A) useFetchCryptoData

- Handles API calls to fetch real-time crypto market data.
- Manages loading, success, and error states efficiently.

B) useLocalStorage

- Stores and retrieves user preferences (e.g., watchlist, dark mode) from local storage.
- Ensures data persistence even after page reloads.

C) useAuth

- Manages user authentication status (logged-in or not).
- Handles login/logout state changes.

3. Utility Classes (if applicable)

If using Tailwind CSS or SCSS, predefined utility classes are used to maintain consistency in styling.

- Example: Common button styles, card layouts, and responsive grid configurations.

6.RUNNING THE APPLICATION

To start the frontend server locally, follow these steps:

1. Ensure Prerequisites are Installed

- Node.js (LTS version)
- npm or yarn
- Required dependencies installed

2. Start the Development Server

Navigate to the project folder and run the appropriate command:

- If using npm → Start the development server
- If using yarn → Start the development server

3. Open the Application in Browser

Once the server is running, open the browser and go to:
A) http://localhost:3000/

The application should now be accessible for testing and development.

4. Stop the Server (If Needed)

To stop the server, use:

- CTRL + C in the terminal

FRONTEND :

Follow these steps to start the frontend server locally:

1. Navigate to the Client Directory – Ensure you are inside the project's frontend folder.

2. Start the Development Server – Use the appropriate command based on your package manager (npm or yarn).

3. Access the Application – Once the server starts, open a browser and go to http://localhost:3000/ to view the project.

4. Stop the Server (If Needed) – To stop the running server, use the terminal shortcut to exit.

7.COMPONENT DOCUMENTATION

KEY COMPONENTS :

The Cryptoverse frontend is built using React, with a modular component structure for better maintainability and reusability. Below are the key components, their purpose, and the props they receive.

1. Header Component

A) Purpose:

- Displays the site logo and navigation menu.
- Allows users to switch themes (dark/light mode).

B) Props:

- theme – Controls the current theme mode.
- onThemeToggle – Function to switch between themes.

2. CryptoList Component

A) Purpose:

- Fetches and displays a list of cryptocurrencies.
- Shows real-time price, percentage change, and market cap.

B) Props:

- data – Array of cryptocurrency details.
- isLoading – Boolean to indicate loading state.
- onSelectCrypto – Function to handle crypto selection.

3. CryptoDetails Component

A) Purpose:

- Shows detailed information about a selected cryptocurrency.
- Displays price charts, historical trends, and relevant stats.

B) Props:

- cryptoId – Unique ID of the selected cryptocurrency.
- data – Object containing details of the selected crypto.

4. SearchBar Component

A) Purpose:

- Allows users to search for specific cryptocurrencies.
- Filters results dynamically as the user types.

B) Props:

- onSearch – Function to handle search input updates.

5. Watchlist Component

A) Purpose:

- Displays the user's favorite cryptocurrencies for easy tracking.
- Allows adding/removing cryptos from the watchlist.

B) Props:

- watchlist – Array of saved cryptocurrencies.
- onRemove – Function to remove a crypto from the list.

6. Chart Component

A) Purpose:

- Displays historical price trends using a graph/chart.
- Supports different time intervals (1 day, 1 week, 1 month).

B) Props:

- cryptoId – Cryptocurrency ID for fetching chart data.
- timeframe – Selected time range for the chart.

7. Footer Component

A) Purpose:

- Displays links to social media and additional resources.
- Provides copyright information.

B) Props:

- None (static content).

REUSABLE COMPONENTS :

To maintain a clean and scalable architecture, Cryptoverse includes several reusable components that can be used across different sections of the application. Below are the key reusable components along with their purpose and configurations.

1. Button Component

A) Purpose:

- A standardized button used throughout the application for actions like submitting forms, navigation, and toggling features.

B) Configurations:

- Type – Primary, secondary, or outlined styles.
- Size – Small, medium, or large.
- Disabled – Option to disable interactions.
- Click Event – Executes the provided function on click.

2. Card Component

A) Purpose:

- Used to display cryptocurrency details in a structured and visually appealing manner.

B) Configurations:

- Title – Displays the crypto name.
- Image/Icon – Shows the crypto logo.
- Data Fields – Price, percentage change, market cap, etc.
- Clickable – Option to make the card interactive.

3.Modal Component

A) Purpose:

- Displays pop-ups for alerts, confirmations, or detailed views without navigating to a new page.

B) Configurations:

- Visibility – Controls whether the modal is open or closed.
- Header & Content – Customizable title and body.
- Actions – Buttons for user interaction (e.g., Confirm, Cancel).

4. Loader Component

A) Purpose:

- A spinning animation or skeleton loader to indicate data is being fetched.

B) Configurations:

- Size – Small, medium, or full-page loader.
- Type – Circular spinner or content skeleton.

5. SearchBar Component

A) Purpose:

- Allows users to search for cryptocurrencies dynamically.

B) Configurations:

- Placeholder Text – Customizable input prompt.
- OnChange Event – Calls a function whenever the user types.
- Debounce Option – Reduces API calls by delaying execution.

6. ToggleSwitch Component

A) Purpose:

- Used for enabling/disabling features like dark mode or price alerts.

B) Configurations:

- State – On or Off.
- Size – Small, medium, or large.
- OnToggle Function – Handles state changes.

7. Tooltip Component

A) Purpose:

- Provides additional information when users hover over an element.

B) Configurations:

- Text Content – Message to display.
- Position – Top, bottom, left, or right.

8.STATE MANAGEMENT

GLOBAL STATE :

State management is crucial for handling and sharing data efficiently across the Cryptoverse application. The project uses a global state management approach to maintain consistency and optimize performance.

1. Global State Management Approach

Cryptoverse uses Context API (or Redux, depending on the complexity) to manage and share state across multiple components without excessive prop drilling.

A) Why Global State?

- Ensures consistency across different components.
- Prevents unnecessary API calls by storing fetched data.
- Allows seamless communication between unrelated components.

2. State Flow in the Application

A) Fetching Data – API requests retrieve cryptocurrency data and store it in the global state.
B) Storing Data – The fetched data is kept in the global state so multiple components can access it without refetching.
C) Updating State – User actions like adding a crypto to a watchlist or changing the theme update the state dynamically.

D) Sharing State – Components subscribe to the global state and react to changes instantly.

3. Example Use Cases of Global State

A) User Preferences: Stores theme selection (dark/light mode).
B) Cryptocurrency Data: Maintains a list of cryptos, prices, and historical data.
C) Watchlist: Saves user-selected cryptocurrencies for tracking.
D) Search & Filtering: Shares search input across different sections of the app.

4. Component Interaction with State

- Header Component – Reads global theme state.
- CryptoList Component – Fetches and displays crypto data from the state.
- Watchlist Component – Uses state to display user's saved cryptos.
- SearchBar Component – Updates the state when a user searches for a crypto.

LOCAL STATE :

While global state management is used for sharing data across the application, local state is handled within individual components for temporary or component-specific data.

1. What is Local State?

Local state refers to data that is managed within a single component and does not need to be shared across multiple components. It is typically used for UI interactions, form handling, and temporary component behavior.

2. How Local State is Used in Cryptoverse

A) SearchBar Component

- Stores user input while typing a search query.
- Updates results dynamically without affecting the global state.

B) Modal Component

- Manages whether a pop-up/modal is open or closed.

- Ensures smooth animations and proper UI interactions.

C) ToggleSwitch Component

- Controls the on/off state of switches like theme mode or notifications.
- Prevents unnecessary updates to the global state.

D) Forms (e.g., Login, Registration)

- Stores user-entered values temporarily before submission.
- Clears fields after successful form submission.

E) Dropdowns & Menus

- Tracks whether a dropdown menu is expanded or collapsed.
- Used in navigation and filter selections.

3. Why Use Local State?

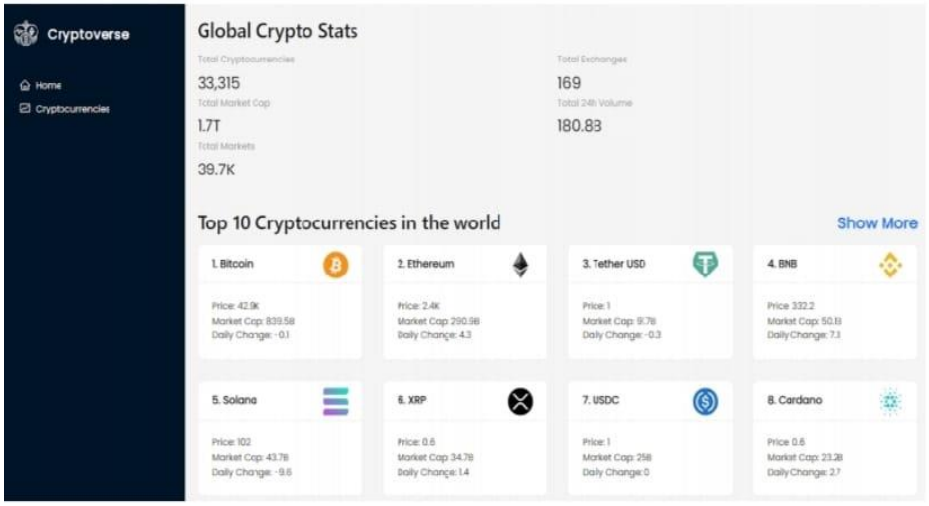
A) Optimized Performance – Prevents unnecessary re-renders by keeping temporary data isolated.
B) Better Component Encapsulation – Keeps UI-specific logic within the relevant component.
C) Faster Updates – Local state changes do not affect other parts of the app, making UI interactions smoother.

4. When to Use Local vs. Global State?

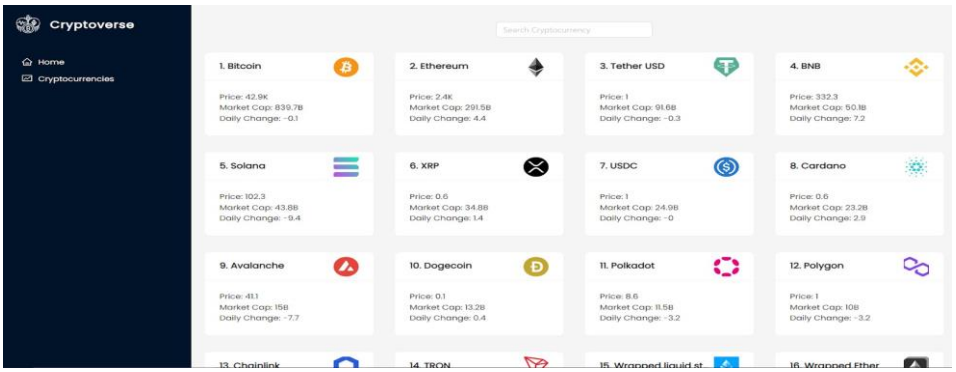
A) Use Local State when data is temporary and component-specific (e.g., UI toggles, form inputs).
B) Use Global State when data needs to be shared across multiple components (e.g., user preferences, fetched crypto data).

9.USER INTERFACE

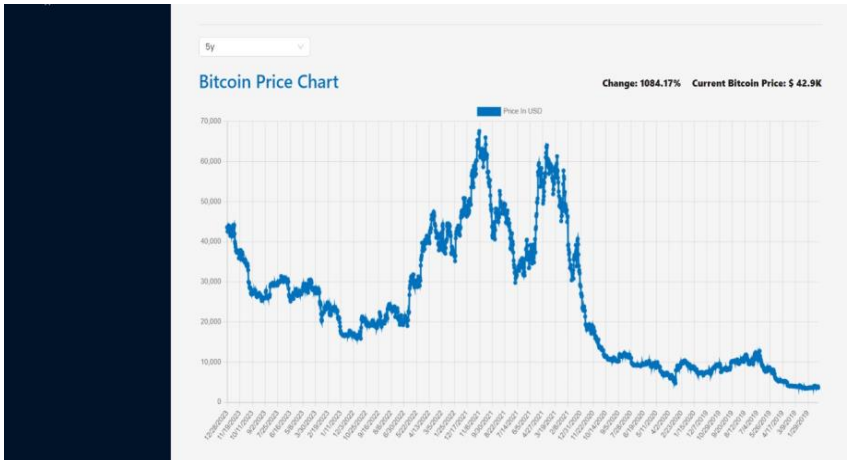
Home page : This pages consists of stats of global crypto like total cryptocurrencies, total exchanges, market cap etc. Also consist of top 10 cryptocurrencies in the world.



Crypto currencies page : This pages contains all cryptocurrencies which are currently in flow in the world. There is also a search feature where users can search and find out about their desired cryptocurrency.



Crypto currency details page : This page contains the line chart with data representation of price of cryptocurrencies. Also contains statistics and website links of cryptocurrencies.



10.STYLING

CSS FRAMEWORKS/LIBRARIES :

The styling approach in Cryptoverse ensures a clean, modern, and responsive user interface. Below is a breakdown of the CSS tools and methodologies used in the project.

1. CSS Frameworks & Libraries Used

A) Tailwind CSS (if applicable)

- Utility-first CSS framework for rapid styling.
- Enables flexible and reusable styles without writing custom CSS.

B) Styled-Components (if applicable)

- Enables writing component-scoped styles in JavaScript.
- Supports dynamic styling based on component props.

C) Sass/SCSS (if applicable)

- Used for variables, mixins, and nested styling for better maintainability.

D) CSS Modules (if applicable)

- Helps scope styles to individual components to prevent global conflicts.

2. Styling Approach & Best Practices

A) Responsive Design:

- Uses Flexbox and CSS Grid for layout structuring.
- Media queries ensure proper scaling for mobile, tablet, and desktop views.

B) Dark Mode Support:

- Theme styles dynamically adjust based on user preference.
- Implemented using CSS variables or global state management.

C) Consistent UI Elements:

- Buttons, cards, inputs, and modals follow a uniform design system.
- Custom utility classes (if using Tailwind or CSS Modules) for reusable styles.

D) Animations & Transitions:

- CSS animations for smooth hover effects.
- Framer Motion (if used) for advanced UI interactions.

3. Example Styling Implementations

A) Global Styles: Define typography, color schemes, and base styles.
B) Component-Specific Styles: Encapsulated styles for modularity and reusability.
C) Theme Customization: Light/dark mode switch with stored user preferences.

THEMING :

The Cryptoverse application includes theming support to enhance user experience, ensuring visual consistency and customization options.

1. Theming Approach

A) Dark Mode & Light Mode Support

- Users can toggle between dark and light modes.
- Theme preferences are stored (e.g., in local storage or global state).

- Styles dynamically change based on user selection.

B) Custom Design System (if applicable)

- Uses a predefined color palette, typography, and spacing for consistency.
- Standardized buttons, cards, and form elements across the UI.
- Reusable components maintain uniformity and reduce duplicate styling.

C) CSS Variables or Tailwind Configuration (if applicable)

- Theme colors, fonts, and spacing are defined using CSS variables.
- Dynamic theming is achieved through Tailwind's theme extension or Styled-Components.

2.Theming Implementation

A) Color Scheme:

- Light Mode: Bright backgrounds, dark text.
- Dark Mode: Dark backgrounds, light text.
- Accent colors for primary UI elements (buttons, links).

B) Typography & Spacing:

- Consistent font sizes, margins, and paddings.
- Readability-focused font choices.

C) Component-Based Theming:

- All UI components (buttons, cards, navbars) follow the theme settings.

D) User Preferences Persistence:

- Once a user selects a theme, the choice is saved (e.g., localStorage) and applied automatically on the next visit.

11.TESTING

TESTING STRATEGIES :

To ensure the stability, reliability, and performance of the Cryptoverse application, a structured testing approach is followed. This includes unit testing, integration testing, and end-to-end (E2E) testing using modern testing frameworks.

1. Types of Testing Used

A) Unit Testing

- Focuses on isolated components and functions.
- Ensures that individual components (e.g., buttons, search bars) work correctly.
- Tools Used: Jest, React Testing Library

B) Integration Testing

- Tests how multiple components interact.
- Verifies that APIs and state management (e.g., Redux, Context API) work correctly.
- Tools Used: Jest, React Testing Library, MSW (Mock Service Worker)

C) End-to-End (E2E) Testing

- Simulates real user interactions from page navigation to API calls.
- Ensures the application functions correctly across different devices and browsers.
- Tools Used: Cypress, Playwright, Selenium

2. Testing Tools & Frameworks

A) Jest – A popular testing framework for running unit and integration tests in React.
B) React Testing Library – Used to test React components in a user-centric way.
C) Cypress – Automates browser testing for E2E tests.
D) Mock Service Worker (MSW) – Mocks API responses for testing without relying on a real backend.

3. Example Testing Scenarios

A) Unit Testing:

- Test if the SearchBar component updates input correctly.
- Check if the CryptoCard component renders data properly.
- Validate that the theme toggle button updates UI modes.

B) Integration Testing:

- Verify that the API fetching logic correctly updates the UI.
- Test if the state management system (Redux/Context API) correctly modifies global state.
- Check if the navigation links work properly.

C) End-to-End (E2E) Testing:

- Simulate a user searching for a cryptocurrency and checking details.
- Ensure that a user can toggle dark mode and see immediate UI changes.
- Validate that a user can navigate between pages smoothly.

4. Why This Testing Approach?

A) Prevents Bugs Before Deployment – Early issue detection avoids production failures.
B) Improves Code Reliability – Ensures components function as expected across updates.
C) Enhances User Experience – Guarantees smooth navigation and accurate data fetching.

CODE COVERAGE :

To ensure comprehensive test coverage, the Cryptoverse project uses tools and techniques to measure how much of the codebase is tested. Code coverage helps identify untested parts of the application, ensuring robust and reliable software.

1. Code Coverage Tools Used

A) Jest & Istanbul (Default with Jest)

- Jest provides built-in code coverage reports using Istanbul.
- Measures coverage for functions, statements, branches, and lines.
- Generates a detailed report showing untested parts of the code.

B) Cypress Code Coverage Plugin (for E2E tests)

- Used for tracking frontend test coverage during Cypress end-to-end tests.
- Helps verify if all user interactions are covered in E2E tests.

C) Codecov (Optional, for CI/CD)

- A cloud-based code coverage tool that integrates with GitHub Actions.
- Helps track test coverage trends over time in a project.

2. Key Code Coverage Metrics

A) Statement Coverage: Ensures all executable statements run at least once.
B) Branch Coverage: Verifies that every possible decision path (if-else) is tested.
C) Function Coverage: Confirms that all functions are called at least once.
D) Line Coverage: Checks that every line of code is executed in tests.

3. Techniques to Ensure High Code Coverage

A) Write Unit Tests for Critical Components

- Test UI components like buttons, forms, and search bars.
- Ensure Redux/Context API actions update global state correctly.

B) Mock API Calls to Test Edge Cases

- Use Mock Service Worker (MSW) to test different API responses (success, failure).

C) Perform Integration Testing

- Validate that state updates reflect correctly across multiple components.

D) Run End-to-End (E2E) Tests

- Automate real user flows, including navigation, data fetching, and UI updates.

E) Monitor Test Coverage Reports

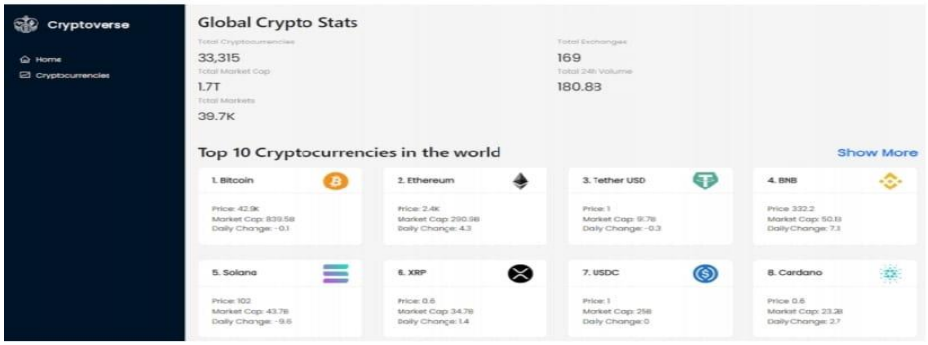
- Use Jest's --coverage flag to generate reports.
- Continuously improve low-coverage areas.

4. Why Code Coverage Matters?

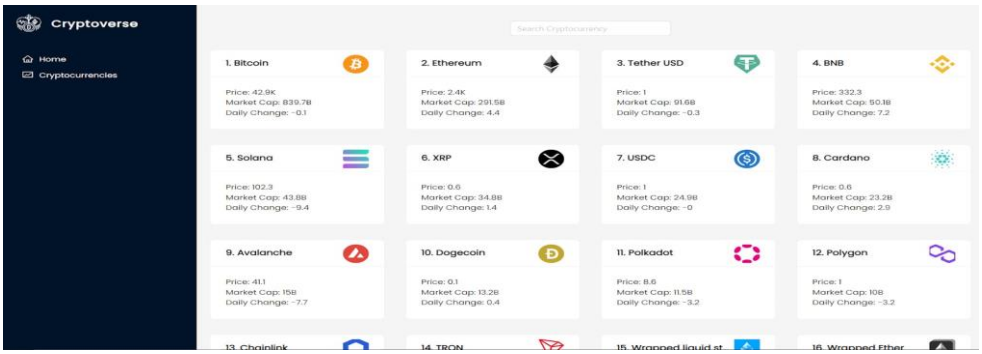
A) Prevents Bugs & Regressions – Ensures all critical paths are tested.
B) Improves Code Quality – Detects missing test cases and improves maintainability.
C) Enhances Confidence in Deployments – Higher coverage means fewer unexpected failures.

12.SCREENSHOTS/DEMO

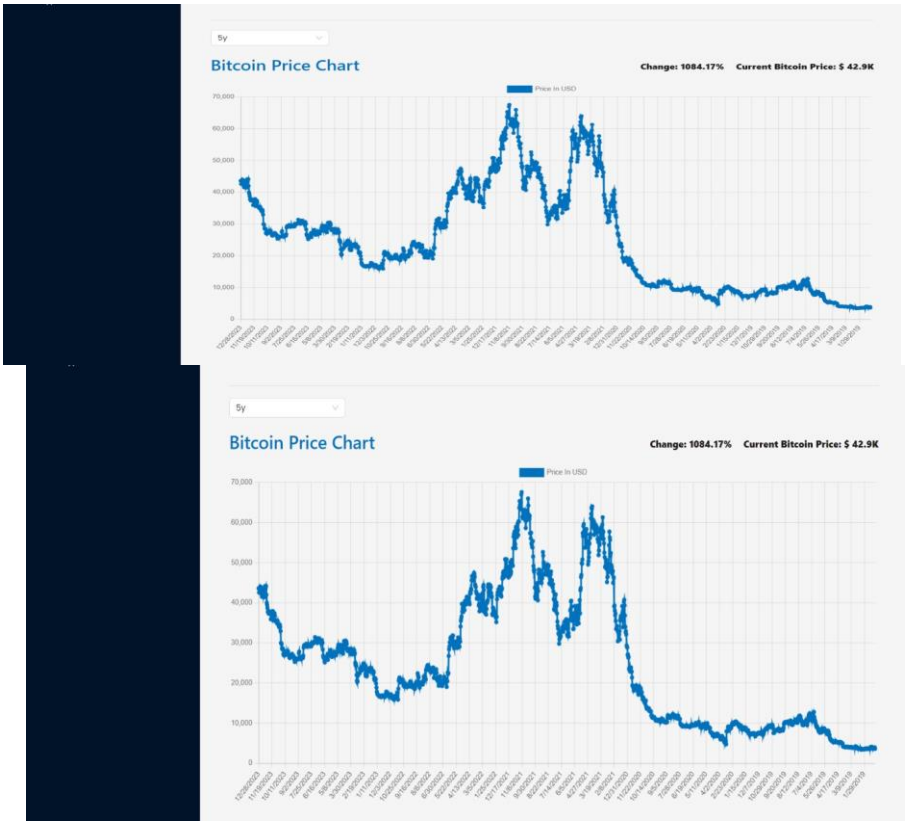
Home page : This pages consists of stats of global crypto like total cryptocurrencies, total exchanges, market cap etc. Also consist of top 10 cryptocurrencies in the world.



Crypto currencies page : This pages contains all cryptocurrencies which are currently in flow in the world. There is also a search feature where users can search and find out about their desired cryptocurrency.



Crypto currency details page : This page contains the line chart with data representation of price of cryptocurrencies. Also contains statistics and website links of cryptocurrencies.



13.KNOWN ISSUES

Below are some known bugs, limitations, and issues that users or developers should be aware of while using or working on Cryptoverse.

1. UI/UX Issues

A) Slow Data Loading – Cryptocurrency market data might take time to load, causing a delay in UI updates.
B) Dark Mode Persistence Issue – The selected theme may reset after a page refresh.
C) Mobile Responsiveness – Some charts and tables might not display correctly on smaller screens.

2. Functional Issues

A) Search Function Sensitivity – Searching for cryptocurrencies may be case-sensitive, leading to unexpected results.
B) API Rate Limits – Excessive requests may trigger API restrictions, preventing real-time data updates.
C) Chart Data Rendering Errors – Historical price data may sometimes not render properly in charts.

3. Performance Issues

A) Heavy API Requests – Large API responses may cause slow page loading and performance drops.
B) Unnecessary Re-renders – Some state updates may cause unnecessary component re-renders, affecting efficiency.

4. Security & Deployment Issues

A) API Key Exposure Risk – API keys must be stored securely to prevent unauthorized access.
B) CORS Restrictions – Some API requests might fail due to Cross-Origin Resource Sharing (CORS) limitations.

5. Suggested Workarounds & Fixes

A) Implement lazy loading and loading indicators to improve UX.
B) Use debouncing in the search bar to reduce unnecessary API calls.
C) Store theme preferences in local storage to persist dark mode settings.
D) Optimize state management to minimize unnecessary re-renders.

14.FUTURE ENHANCEMENTS

Below are some potential features and improvements that can be added to Cryptoverse to enhance its functionality, user experience, and performance.

1. UI/UX Improvements

A) Advanced Animations & Transitions – Add smooth animations for better visual appeal using Framer Motion.
B) Improved Mobile Responsiveness – Optimize charts and tables for smaller screens.
C) Customizable Themes – Allow users to personalize the UI with different color schemes.

2. New Features & Functionalities

A) Portfolio Tracking – Enable users to create a portfolio and track their investments.
B) Real-Time WebSocket Data – Implement live price updates instead of periodic API calls.
C) Crypto News Integration – Display the latest cryptocurrency news and trends.
D) Price Alerts – Allow users to set price alerts for specific cryptocurrencies.

3. Performance & Optimization

A) Reduce API Calls with Caching – Use local storage or IndexedDB to reduce redundant API requests.
B) Optimize Global State Management – Improve efficiency in handling global states with Redux Toolkit.
C) Lazy Load Components – Load heavy components only when needed to improve speed.

---

4. Security & Accessibility Enhancements

A) Secure API Keys – Implement better security practices to protect API keys.
B) Dark Mode Persistence Fix – Ensure theme settings are stored properly across sessions. 362Improve Keyboard Navigation &
C)Accessibility – Enhance support for screen readers and keyboard shortcuts.

5. Deployment & Scalability Enhancements

A) Multi-Language Support – Add translations to support a global audience.
B) Progressive Web App (PWA) – Convert Cryptoverse into a PWA for offline access.
C) User Authentication – Implement login/signup for personalized dashboards.