**Cryptocurrency Dashboard**

**Team ID : SWTID1741245881158516**

**TEAM MEMBERS :**

**Team Size : 5**

**Team Leader :** M Samuvel

( email : [samuvelm007@gmail.com](mailto:samuvelm007@gmail.com) )

**Team member :** K Thirumoorthy

( email : [tmoorthy811@gmail.com](mailto:tmoorthy811@gmail.com) )

**Team member :**  E Ragul

( email : [eragul87@gmail.com](mailto:eragul87@gmail.com) )

**Team member :**  Pradish Mariappan

( email : [Pradishmariappan607@gmail.com](mailto:Pradishmariappan607@gmail.com) )

**Team member :**  M Sanmugasundaram

(email : [sanmugamsundaram85@gmail.com](mailto:sanmugamsundaram85@gmail.com) )

# INTRODUCTION

Cryptocurrency markets are highly volatile, and tracking price changes, market capitalization, and trading volume in real-time is essential for investors and traders. The Cryptocurrency Dashboard is a web-based application designed to provide a simple and interactive way to monitor cryptocurrency prices and trends.

This dashboard displays key details of various cryptocurrencies, including their current price, 24-hour change, market capitalization, and trading volume. The user-friendly interface ensures easy navigation and readability of data.

The goal of this project is to present cryptocurrency data in a visually appealing and structured format using HTML, CSS, and JavaScript. Currently, the data is static, but future improvements will include live updates through API integration.

# **Languages Used :**

1. **HTML (HyperText Markup Language)**

HTML is used to create the structure of the web pages. It defines elements such as headings, paragraphs, images, lists, links, and buttons. In this project, multiple HTML files are used to organize different recipe categories and pages.

# CSS (Cascading Style Sheets)

CSS is used to style the website and enhance the user experience. It controls the layout, colors, fonts, and responsive behavior of the pages. This project uses external CSS (style.css) to maintain consistency across different pages.

# JavaScript (Planned but Not Implemented Yet)

Although the provided code does not contain JavaScript, it can be used to add interactivity, such as dynamically displaying ingredients, filtering recipes, or adding animations.

# PROJECT OVERVIEW

# **1. Purpose**

# The Cryptocurrency Dashboard is designed to provide users with a simple and effective way to track cryptocurrency market trends. Cryptocurrencies like Bitcoin, Ethereum, and others experience frequent price fluctuations, making it essential for traders and investors to stay updated.

# Display essential cryptocurrency details such as price, market cap, and 24-hour changes in a structured format.

# Offer a user-friendly and responsive interface for easy navigation.

# Serve as a foundation for future enhancements like real-time data updates using APIs.

# **2. Features**

# ✅ **Cryptocurrency Price Display:** Shows the latest price of different cryptocurrencies.

# ✅ **Market Cap Information:** Displays the total market value of each cryptocurrency.

# ✅ **24-Hour Price Change:** Indicates whether the price has increased or decreased in the past 24 hours.

# ✅ **Trading Volume:** Provides insights into the trading activity of each cryptocurrency.

# ✅ **User-Friendly Interface:** Simple design with easy-to-read data.

# ✅ **Responsive Design:** Works well on different screen sizes.

# ✅ **Comparison Table:** Lists multiple cryptocurrencies in a structured format for easy comparison.

# **3. Architecture Overview**

# The application will follow a component-based architecture using React.js. It will consist of:

# UI Components (Navbar, Recipe Card, etc.)

# Pages (Homepage, Meals, Drinks, Desserts, Recipe Details)

# State Management (for handling recipes and user interactions)

# Routing (for navigation between pages

# **Component Structure**

# The Cryptocurrency Dashboard follows a structured design where different sections of the page serve specific purposes. The major components of the project include:

# ✅ Navbar Component:

# Displays navigation links (Home, About, Contact).

# Ensures easy navigation across different sections.

# ✅ Dashboard Component:

# Shows cryptocurrency details such as price, market cap, and 24-hour changes.

# Each cryptocurrency is displayed inside a card-style layout.

# ✅ Comparison Table Component:

# Displays multiple cryptocurrencies in a tabular format for easy comparison.

# Includes price changes, market cap, trading volume, and circulating supply.

# ✅ Footer Component:

# Displays copyright information and branding details.

# Each section is structured using HTML, styled using CSS, and dynamically handled with JavaScript.

# **2.State Management**

# Since this project is built using vanilla JavaScript, state management is handled using:

# JavaScript Variables & Objects: Data is stored in objects and updated dynamically.

# Event Listeners: User interactions (like filtering cryptocurrencies) are managed using event listeners.

# Local Storage (Optional): If needed, user preferences (e.g., favorite coins) can be stored locally.

# If migrating to React, Context API or Redux can be used for state management.

# **3.Routing Structure (Using React Router)**

# Currently, this project does not use routing as it is a single-page application (SPA). However, if React is implemented, React Router can be used for navigation between different pages like:

# Home Page ("/") – Displays the main dashboard.

# Details Page ("/crypto/:id") – Displays in-depth details of a selected cryptocurrency.

# About Page ("/about") – Provides information about the project.

# **4.SETUP INSTRUCTIONS**

# **1. Prerequisites**

# Before running the Cryptocurrency Dashboard, ensure you have the following installed:

# ✅ Web Browser: Google Chrome, Firefox, or Edge for viewing the dashboard.

# ✅ Code Editor: VS Code (Recommended) or any other code editor.

# ✅ Live Server Extension (Optional): If using VS Code, install the Live Server extension to preview changes instantly.

# ✅ Git (Optional): If cloning the project from a repository.

# **Installation & Setup**

# Step 1: Clone the Repository

# If the project is hosted on GitHub, use the following command to clone it:

# git clone https://github.com/your-username/cryptocurrency-dashboard.git

# cd cryptocurrency-dashboard

# (Replace your-username with your actual GitHub username.)

# Step 2: Open the Project in VS Code

# code .

# (This command opens the project in VS Code. Alternatively, you can manually open the folder.)

# Step 3: Run the Project

# Since this is a static project using HTML, CSS, and JavaScript, there is no need to install dependencies. Simply open the index.html file in a browser:

# Option 1: Double-click index.html to open it in a browser.

# Option 2 (Recommended): Use Live Server in VS Code:

# Right-click on index.html → Click "Open with Live Server".

# **5. FOLDER STRUCTURE**

# Current Structure (HTML, CSS, JavaScript)

# cryptocurrency-dashboard/ (Project root directory)

# assets/ → Contains images, icons, and other static assets

# css/ → Contains stylesheets

# styles.css → Main stylesheet

# js/ → Contains JavaScript files

# script.js → Handles DOM interactions and data fetching

# index.html → Main HTML file

# about.html → About page (if applicable)

# contact.html → Contact page (if applicable)

# README.md → Project documentation

# **Suggested React Folder Structure (If Migrated to React)**

# cryptocurrency-dashboard/ (Project root directory)

# public/ → Public assets like favicon and index.html

# src/ → Main source code directory

# assets/ → Static assets like images and icons

# components/ → Reusable UI components

# Navbar.js → Navigation bar component

# Card.js → Cryptocurrency card component

# Table.js → Cryptocurrency data table component

# pages/ → Page-level components

# Home.js → Home page (dashboard)

# About.js → About page

# Contact.js → Contact page

# utils/ → Utility functions and custom hooks

# api.js → API call functions

# format.js → Helper functions for formatting numbers, dates, etc.

# App.js → Main application component

# index.js → Entry point for the React app

# package.json → Project dependencies and scripts

# README.md → Project documentation

# **RUNNING THE APPLICATION**

# **Running the Application Locally**

# 1. Open the project folder in VS Code.

# 2. Install the Live Server extension (if not installed).

# 3. Right-click on index.html and select "Open with Live Server".

# 4. The project will run on http://127.0.0.1:5500/ or a similar local address.

# **Running the Application in a React Setup**

# If you migrate the project to React, follow these steps to run the frontend server locally:

# 1. Navigate to the project directory

# cd cryptocurrency-dashboard

# 2. Install dependencies

# npm install

# 3. Start the frontend server

# npm start

# This will start the React development server at http://localhost:3000/ by default.

# **For the React Version (If Upgrading)**

# Install Dependencies

# npm install

# 2. Start the Development Server

# npm run dev

# Open the App in Browser

# Once the server starts, open:

# http://localhost:5173/

# (If using Vite for React development)

# **COMPONENT DOCUMENTATION**

# **1. Key Components**

# Below are the major components in the Cryptocurrency Dashboard project, their purpose, and the props they receive.

# 1. Dashboard Component

# Purpose: Displays the list of cryptocurrencies with their prices, market cap, and percentage changes.

# Props:

# cryptoData (array): Contains cryptocurrency details such as name, price, and change percentage.

# 2. CryptoCard Component

# Purpose: Displays individual cryptocurrency information in a card format.

# Props:

# name (string): Name of the cryptocurrency.

# symbol (string): Symbol of the cryptocurrency.

# price (number): Current price of the cryptocurrency.

# change (number): 24-hour percentage change.

# marketCap (number): Market capitalization value.

# icon (string): URL or path to the cryptocurrency logo.

# 3. CryptoTable Component

# Purpose: Displays cryptocurrency market details in a tabular format, including price changes over different time frames.

# Props:

# cryptoList (array): List of cryptocurrencies with detailed market data.

# **2. Reusable Components**

# 1. Navbar Component

# Purpose: Displays the navigation bar with links to Home, About, and Contact pages.

# Props: None (static navigation menu).

# 2. Loader Component

# Purpose: Shows a loading animation while data is being fetched.

# Props: None (only UI-based).

# 3. Button Component

# Purpose: A reusable button used for various actions like refreshing data or navigation.

# Props:

# text (string): Button label.

# onClick (function): Function to execute on button click.

# styleClass (string): Custom styling class (optional).

# **STATE MANAGEMENT**

# **Global State Management**

# Since your Cryptocurrency Dashboard currently uses HTML, CSS, and JavaScript, you might not have a global state. However, if you migrate to React, global state management will be essential for handling data across multiple components.

# If using React with Context API or Redux, the global state would handle:

# Cryptocurrency Data: Storing API responses (prices, market caps, etc.) so all components can access the latest market data.

# Theme Preferences: If the project includes a dark/light mode feature.

# User Preferences: Settings like currency selection (USD, EUR, INR).

# **2. Local State Management**

# Local state is managed inside individual components using React's useState or JavaScript variables. Examples of local state include:

# Search Input: Managing the search query entered by the user.

# Sort & Filter Options: Storing whether the user wants to sort cryptocurrencies by market cap, price, or percentage change.

# Loading State: Showing a loading spinner while fetching data.

# **USER INTERFACE**

# **1.Home Page (Dashboard View)**

# Displays a list of cryptocurrencies with their prices, market cap, and percentage changes.

# Includes a search bar to filter cryptocurrencies.

# Example Screenshot: (Insert screenshot here)

# **2 .Detailed View (If Available)**

# Clicking on a cryptocurrency card/table row might show more details.

# Includes price trends, historical data, and additional market insights.

# Example Screenshot: (Insert screenshot here)

# **3. Search and Filter Feature**

# Users can search for a cryptocurrency by name or symbol.

# Sorting options to arrange data based on price, market cap, or percentage change.

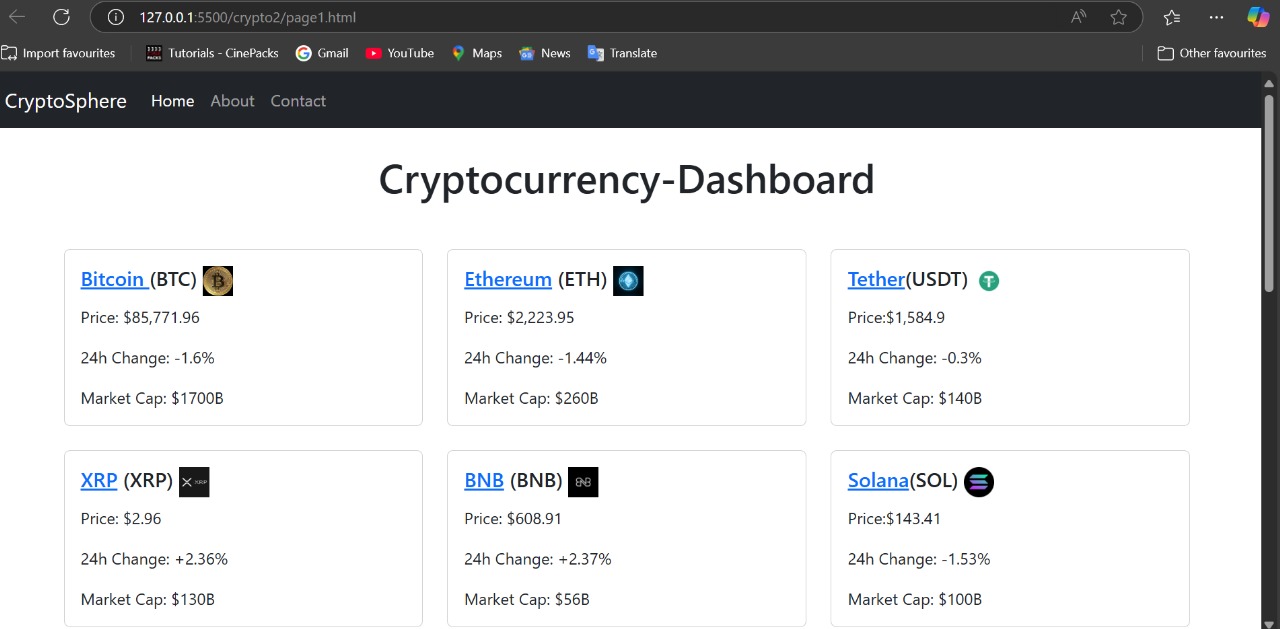
# Example GIF/Screenshot: (Insert GIF or screenshot here)

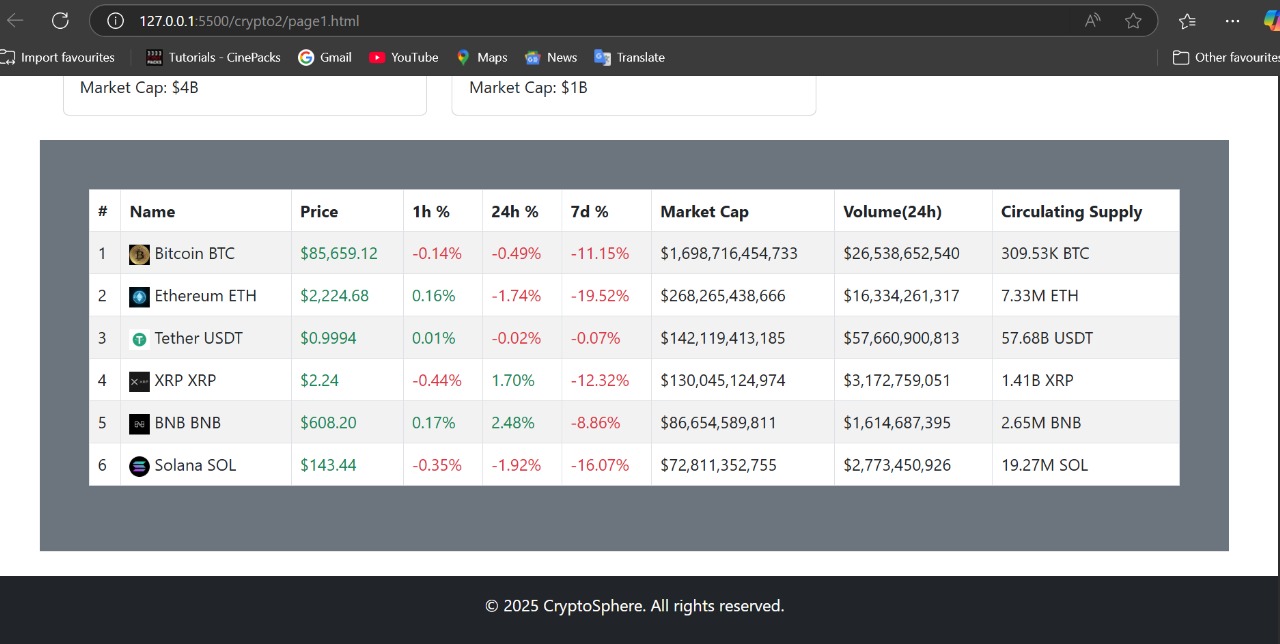
# **4 .Responsive Design**

# Works on different screen sizes (desktop, tablet, mobile).

# Uses CSS media queries for responsiveness.

# Example Screenshot (Mobile View): (Insert screenshot here)





1. **STYLING**

**1. CSS Frameworks/Libraries**

Your Cryptocurrency Dashboard is styled using CSS, and if any CSS frameworks or libraries are used, you can document them here.

CSS: The project primarily uses custom CSS for styling.

CSS Frameworks (If Used):

Bootstrap – If used, provides a grid system and pre-designed UI components.

Tailwind CSS – If used, enables utility-first styling for faster UI development.

1. **Theming**

If your project includes dark mode or a custom theme, document it here.

Custom Styling:

Color schemes, typography, and spacing are defined in the styles.css file.

Dark Mode (If Implemented):

Users can switch between light and dark themes using CSS classes or JavaScript.

1. **TESTING**

**1 .Testing Strategy**

1. Manual Testing (For Current HTML, CSS, JS Project)

Since the project is a frontend dashboard, testing can be done manually by checking:

**UI Responsiveness:** Open the site on different screen sizes (desktop, tablet, mobile).

**Functionality Testing**:

Ensure data loads correctly from APIs.

Verify that sorting and filtering features work as expected.

Test search functionality with different inputs.

**Performance Testing:** Check page speed using Lighthouse in Chrome DevTools.

**2. Automated Testing (If Migrated to React)**

If the project is later converted to React, you can use:

Jest for unit testing.

React Testing Library for component testing.

Cypress or Playwright for end-to-end (E2E) testing.

**3. Code Coverage**

If using Jest, run:

npm test -- --coverage

This generates a code coverage report showing which parts of the code are tested.

If testing manually, use Chrome DevTools to inspect elements, check errors, and verify performance.

1. **SCREENSHOT OR DEMO**

https://drive.google.com/file/d/1\_RVPFCvakalcE4jX7IFAb2XhlNQfUGGO/view?usp=sharing

1. **KNOWN ISSUES**

1. API Data Delay – Data takes time to load. Fix: Show a loading spinner.

2. No Error Handling – API failures are not displayed. Fix: Show an error message if the request fails.

3. Search Not Optimized – Case-sensitive or incomplete searches may not work well. Fix: Convert text to lowercase.

4. No Pagination – Too many cryptocurrencies slow down rendering. Fix: Implement pagination or infinite scroll.

5. No Dark Mode – Users may prefer a dark theme. Fix: Add a theme toggle.

6. UI Not Fully Responsive – Layout may break on small screens. Fix: Use CSS media queries.

1. **FUTURE ENHANCEMENTS**

1. Real-time Data Updates – Auto-refresh cryptocurrency prices without manual reload.

2. User Authentication – Allow users to save favorite coins and track portfolios.

3. Advanced Filtering & Sorting – Add filters for market cap, volume, and performance trends.

4. Dark Mode – Provide a theme toggle for better user experience.

5. Graphical Analysis – Integrate charts for price trends and historical data visualization.

6. Multi-Currency Support – Allow users to view prices in different currencies (INR, EUR, etc.).

***THANK YOU !!***