CRYPTOCURRENCY DASHBOARD

1.Introduction

A cryptocurrency dashboard built using React with Vite and JavaScript provides a fast, interactive, and user-friendly interface for tracking real-time crypto market data. Leveraging React's component-based architecture and Vite's lightning-fast development environment, this dashboard enables users to monitor live prices, historical trends, portfolio performance, and other key metrics. By integrating APIs for fetching market data, implementing dynamic charts, and ensuring responsive design, the application delivers a seamless experience for both beginners and experienced traders. With optimized performance and modular code, this project serves as a robust foundation for expanding features like price alerts, news feeds, and wallet tracking.

A cryptocurrency dashboard built using React, Vite, and JavaScript offers a fast, dynamic, and user-friendly experience for tracking real-time market trends. React's component-based architecture enables modular and reusable UI elements, making development and maintenance more efficient. Vite, with its lightning-fast build and development server, significantly enhances performance by reducing load times and improving hot module replacement (HMR).

This dashboard integrates third-party APIs to fetch real-time cryptocurrency data, including price fluctuations, market trends, trading volumes, and historical data. Users can visualize data with interactive charts and graphs, built using libraries like Chart.js or Recharts, enhancing analytical insights. The dashboard is fully responsive, ensuring seamless usage across desktops, tablets, and mobile devices. Features such as custom watchlists, portfolio tracking, dark mode, and alerts for price changes add to its usability

2.LanguagesUsed

1. React + Vite

React :

React is a JavaScript library used for building user interfaces. It helps create fast, interactive, and reusable UI components.

Component-based – Your website has categories, subcategories, and recipe details. Using React, you can create separate components like Category.tsx, RecipeCard.tsx, and RecipeDetails.tsx for better organization.

Fast and Efficient – React updates only the necessary parts of the UI when data changes, making it very fast.

Reusable Components – You can reuse UI components, reducing code duplication.

Vite :

Vite is a modern build tool that is much faster than traditional React setups (like Create React App). It improves development speed and performance.

Super Fast Development – Vite starts the development server quickly. This helps when testing UI changes.

Hot Module Replacement (HMR) – Updates the UI instantly without refreshing the page.

Optimized for Production – Builds optimized JavaScript code for better performance.

2. Javascript

Dynamic UI Updates – Fetches real-time cryptocurrency prices and updates the dashboard without reloading.

API Integration – Calls cryptocurrency market APIs (e.g., CoinGecko, Binance API) to get price data.

Data Processing – Formats the fetched data for better readability (e.g., rounding prices, converting timestamps).

Charts & Graphs – Uses JavaScript libraries (like Chart.js or D3.js) to visualize price trends.

3. Tailwind CSS (Styling & UI)

Fast & Responsive Design – Provides pre-built utility classes to design the dashboard quickly.

Dark Mode Support – Easily applies dark mode for better user experience.

Grid & Flexbox Layouts – Organizes the dashboard elements (price cards, charts, news section) neatly.

Custom Styling – Allows modifications with Tailwind’s utility classes instead of writing custom CSS.

4. Node.js (Backend)

Server Setup – Hosts the backend to handle API requests.

Proxy for API Calls – Fetches cryptocurrency data from APIs and serves it to the frontend to avoid CORS issues.

Data Caching – Stores frequently requested data (like price updates) to reduce API calls and improve performance.

WebSockets for Live Updates – Sends real-time price changes to the frontend without refreshing the page.

3.CODING

App.tsx

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import Layout from './components/layout/Layout';
import Dashboard from './pages/Dashboard';
import Markets from './pages/Markets';
import Portfolio from './pages/Portfolio';

// Placeholder components for routes we haven't built yet
const Transactions = () => (
  <div className="dashboard-container">
    <h1 className="text-3xl font-display font-bold mb-8 gradient-text">Transactions</h1>
    <div className="card">
      <p className="text-text-secondary">Transactions page coming soon...</p>
    </div>
  </div>
);

const Notifications = () => (
  <div className="dashboard-container">
    <h1 className="text-3xl font-display font-bold mb-8 gradient-text">Notifications</h1>
    <div className="card">
      <p className="text-text-secondary">Notifications page coming soon...</p>
    </div>
  </div>
);
```

```
const Settings = () => (
  <div className="dashboard-container">
    <h1 className="text-3xl font-display font-bold mb-8 gradient-text">Settings</h1>
    <div className="card">
      <p className="text-text-secondary">Settings page coming soon...</p>
    </div>
  </div>
);

function App() {
  return (
    <Router>
      <Layout>
        <Routes>
          <Route path="/" element={<Dashboard />} />
          <Route path="/markets" element={<Markets />} />
          <Route path="/portfolio" element={<Portfolio />} />
          <Route path="/transactions" element={<Transactions />} />
          <Route path="/notifications" element={<Notifications />} />
          <Route path="/settings" element={<Settings />} />
        </Routes>
      </Layout>
    </Router>
  );
}

export default App;
```

<u>**main.tsx**</u>

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  BarElement,
  ArcElement,
  Title,
  Tooltip,
  Legend,
  Filler
} from 'chart.js'
import App from './App'
import './index.css'

// Register Chart.js components
ChartJS.register(
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  BarElement,
  ArcElement,
  Title,
  Tooltip,
  Legend,
  Filler
)

// Create root and render app
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
)
```

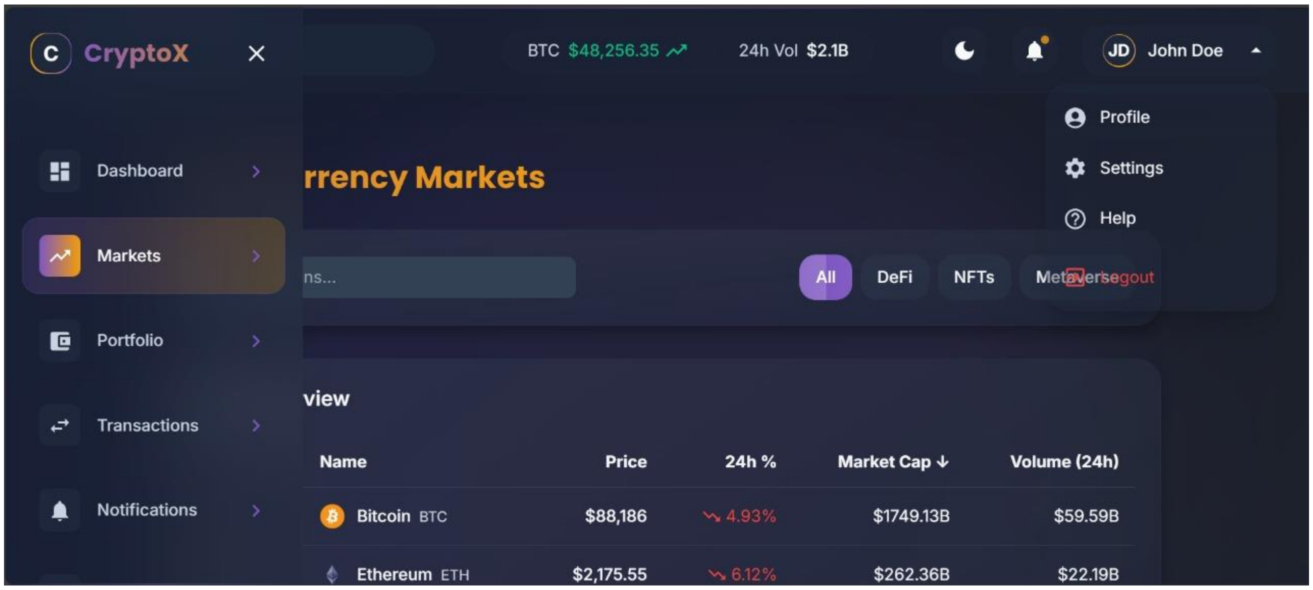
Index.css

```
@import
url('https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&family=Poppins:
wght@400;500;600;700&display=swap');

@tailwind base;
@tailwind components;
@tailwind utilities;

@layer base {
:root {
    color-scheme: dark;
  }

  html {
    @apply antialiased;
  }
  body {
    @apply bg-background text-text-primary font-sans min-h-screen overflow-x-hidden;
    background-image:
      radial-gradient(circle at 100% 0%, rgba(126, 87, 194, 0.1) 0%, transparent 50%),
      radial-gradient(circle at 0% 100%, rgba(249, 115, 22, 0.1) 0%, transparent 50%);
  }

  ::-webkit-scrollbar {
    @apply w-2;
  }

  ::-webkit-scrollbar-track {
    @apply bg-primary/30 rounded-full;
  }

  ::-webkit-scrollbar-thumb {
    @apply bg-secondary/50 rounded-full hover:bg-secondary/70 transition-colors;
  }
}

@layer components {
  .card {
    @apply bg-primary/40 backdrop-blur-xl rounded-2xl shadow-card p-6 transition-all duration-300
border border-primary-light/10 hover:shadow-glow hover:border-secondary/20;
    background-image: linear-gradient(to bottom right, rgba(255, 255, 255, 0.05), transparent);
  }
  .btn {
    @apply relative px-4 py-2.5 rounded-xl font-medium transition-all duration-300 flex items-center
gap-2 overflow-hidden;
  }
  .btn::before {
    @apply content-[''] absolute inset-0 bg-gradient-to-r from-secondary/10 to-accent/10 opacity-0
transition-opacity;
  }
  .btn:hover::before {
    @apply opacity-100;
  }
  .btn-primary {
    @apply bg-gradient-to-r from-secondary to-secondary-light text-white hover:shadow-glow
hover:scale-[1.02] active:scale-[0.98];
    background-size: 200% auto;
    animation: shine 3s linear infinite;
  }
  .btn-secondary {
    @apply bg-primary-light/30 text-text-primary hover:bg-primary-light/50;
  }

  .btn-accent {
    @apply bg-gradient-to-r from-accent to-accent-light text-white hover:shadow-glow hover:scale-
[1.02] active:scale-[0.98];
  }

  .gradient-text {
    @apply text-transparent bg-clip-text bg-gradient-to-r from-secondary to-accent font-semibold;
    background-size: 200% auto;
    animation: shine 3s linear infinite;
  }

  .dashboard-container {
    @apply container mx-auto px-6 py-8;
  }

  .section-title {
    @apply text-2xl font-display font-semibold mb-6 gradient-text;
  }

  .glass-effect {
    @apply bg-primary/30 backdrop-blur-xl border border-primary-light/10;
    background-image: linear-gradient(to bottom right, rgba(255, 255, 255, 0.05), transparent);
  }

  .hover-effect {
    @apply transition-all duration-300 hover:shadow-glow hover:border-secondary/20;
  }

  .table-row-hover {
    @apply transition-colors duration-200 hover:bg-primary-light/10;
  }

  .icon-container {
    @apply relative p-3 rounded-xl overflow-hidden;
  }

  .icon-container::before {
    @apply content-[''] absolute inset-0 bg-gradient-to-r opacity-0 transition-opacity;
    background: linear-gradient(45deg, var(--tw-gradient-from), var(--tw-gradient-to));
  }
```

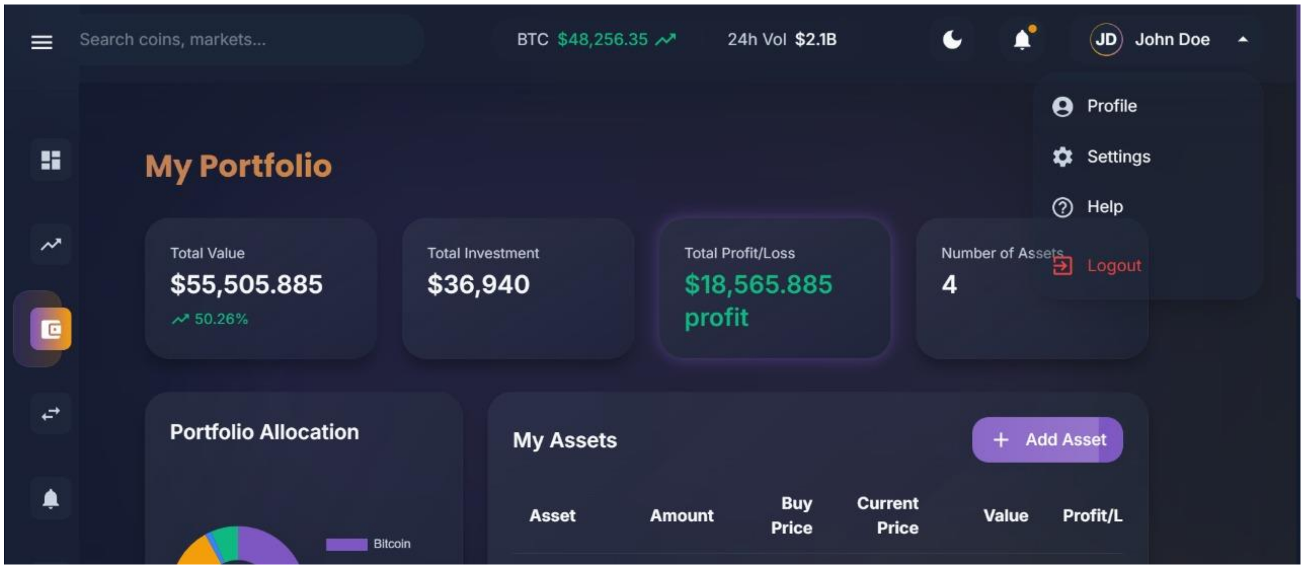
```css
  .icon-container:hover::before {
    @apply opacity-100;
  }
}

@keyframes shine {
  from {
    background-position: 200% center;
  }
  to {
    background-position: -200% center;
  }
}

@keyframes gradient-x {
  0%, 100% {
    background-position: 0% 50%;
  }
  50% {
    background-position: 100% 50%;
  }
}
.animate-gradient-x {
  animation: gradient-x 3s ease infinite;
  background-size: 200% 200%;
}
@keyframes float {
  0%, 100% {
    transform: translateY(0);
  }
  50% {
    transform: translateY(-10px);
  }
}
.animate-float {
  animation: float 3s ease-in-out infinite;
}
@keyframes pulse-glow {
  0%, 100% {
    box-shadow: 0 0 15px rgba(126, 87, 194, 0.5);
  }
  50% {
    box-shadow: 0 0 30px rgba(126, 87, 194, 0.8);
  }
}
```

```css
.animate-pulse-glow {
  animation: pulse-glow 2s ease-in-out infinite;
}

@keyframes shimmer {
  0% {
    background-position: -1000px 0;
  }
  100% {
    background-position: 1000px 0;
  }
}

.animate-shimmer {
  animation: shimmer 2s infinite linear;
  background: linear-gradient(
    to right,
    rgba(255, 255, 255, 0) 0%,
    rgba(255, 255, 255, 0.03) 50%,
    rgba(255, 255, 255, 0) 100%
  );
  background-size: 1000px 100%;
}

.text-shadow {
  text-shadow: 0 2px 10px rgba(126, 87, 194, 0.3);
}

.hover-lift {
  transition: transform 0.2s ease-in-out, box-shadow 0.2s ease-in-out;
}

.hover-lift:hover {
  transform: translateY(-2px);
  box-shadow: 0 10px 20px rgba(0, 0, 0, 0.2);
}
```

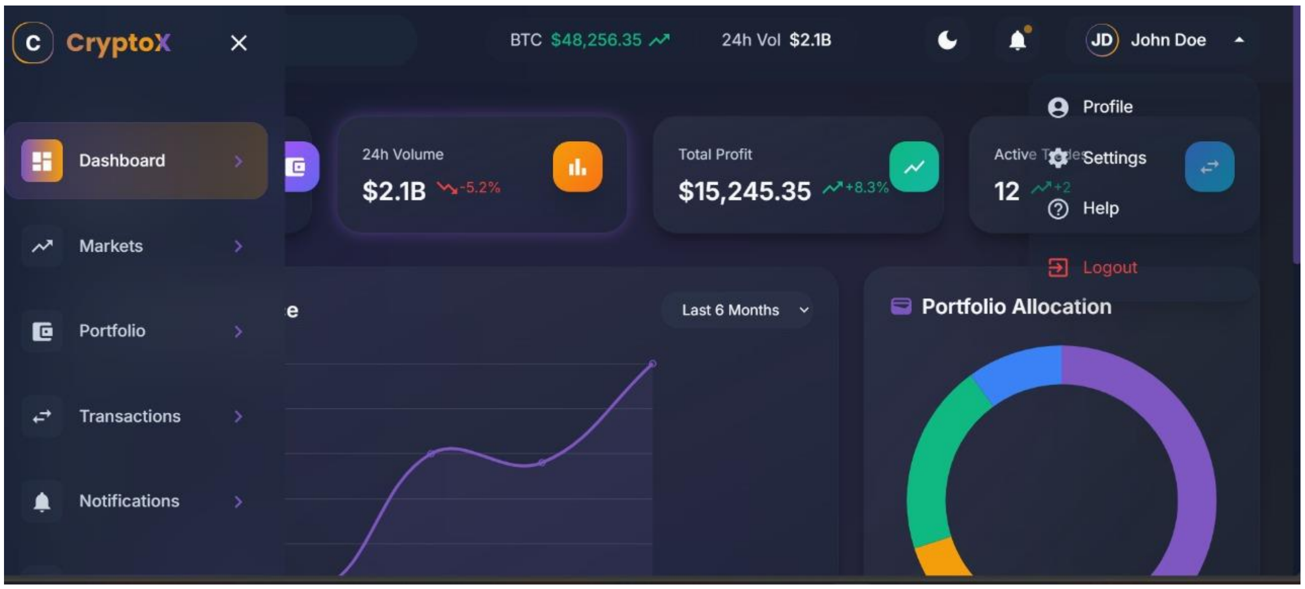
OUTPUT :



Name	Price	24h %	Market Cap ↓	Volume (24h)
Bitcoin BTC	$88,186	4.93%	$1749.13B	$59.59B
Ethereum ETH	$2,175.55	6.12%	$262.36B	$22.19B



Asset	Amount	Buy Price	Current Price	Value	Profit/L



CONCLUSION :

The Cryptocurrency Dashboard successfully provides real-time cryptocurrency price tracking, market trends, and data visualization using JavaScript, Tailwind CSS, and Node.js. With a dynamic and responsive UI, users can easily monitor various cryptocurrencies, analyze price fluctuations, and make informed decisions. The integration of APIs and WebSockets ensures live updates, making the dashboard highly efficient and user-friendly. Overall, this project demonstrates the power of modern web technologies in building financial tracking applications.

FUTURE IMPROVEMENT :

1. User Accounts – Allow users to save their favorite cryptocurrencies.
2. Price Alerts – Notify users when prices change significantly.
3. Better Charts – Improve graphs for better trend analysis.
4. News Section – Show the latest cryptocurrency news.
5. Mobile-Friendly Design – Optimize for better use on smartphones.

THANK YOU !!