Crypto Forex Bubble

(SRS)

Table of Contents

1. Introduction

Crypto Forex Bubble is a modern web application designed to provide **real-time visualization** of cryptocurrency and forex market data. By leveraging **interactive bubble charts**, it transforms complex datasets into engaging visual experiences, enabling traders, analysts, and enthusiasts to **quickly spot trends, anomalies, and movements**.

2. Purpose

The primary purpose of the Crypto Forex Bubble project is to deliver a **user-friendly, visually rich, and intuitive interface** for monitoring and analyzing the crypto and forex markets. The system simplifies **data-heavy feeds into interactive bubbles**, allowing users to grasp market conditions at a glance and take informed decisions.

3. Scope

The system will:

- Provide **real-time visualization** of cryptocurrency and forex assets.
- Enable **category-based exploration** (Crypto, Forex, Forex Pairs).
- Include **interactive bubble charts** with smooth animations.
- Support **search, filtering, and detailed asset insights**.
- Be **responsive and mobile-friendly**.
- Integrate with **reliable APIs** for data fetching.

The initial version is web-based, with provisions for future mobile app development.

4. System Overview

- **Frontend:** Next.js (React), Tailwind CSS, D3.js for visualization
- **Backend/API Layer:** Integration with external APIs (CoinGecko, ExchangeRate-API)
- **Deployment Environment:** Node.js-compatible servers (Vercel, Hostinger, VPS)
- **Configuration:** `.env` files for API keys and environment variables

5. Functional Requirements

5.1 Data Fetching

- Fetch **top cryptocurrencies** from CoinGecko API.
- Fetch **major forex currencies and currency pairs** from ExchangeRate-API.
- Refresh data **every 30 seconds** (configurable interval).
- Store API credentials securely using environment variables.

5.2 Visualization

- Render assets as **interactive bubbles** sized dynamically by market cap or trading volume.
- Use **D3.js physics simulation** for smooth, realistic animations.
- Allow bubbles to be **dragged, hovered, and clicked**.
- Apply **color coding**: green/red for gain/loss, categories differentiated by palettes.

5.3 User Interaction

- **Category Tabs** → Crypto, Forex, Forex Pairs.
- **Search/Filter Bar** → Search by name or symbol.
- **Bubble Click** → **Detailed Info Popup** (price, change, volume, rate).
- **Responsive UI** with dynamic resizing.

5.4 UI Components

- **Header** → Branding, category tabs, search input.
- **Main Area** → Interactive bubble chart.
- **Info Popups** → Asset details on click.
- **Error/Loading Components** → Handle async states.
- **Dashboard Page (Future)** → Placeholder for advanced analytics.
- **Prelaunch Page** → Countdown timer for marketing.

6. Non-Functional Requirements

- **Performance:** Sub-second UI responsiveness, smooth animations at 60 FPS.
- **Reliability:** Graceful handling of API failures with fallback data.
- **Security:** API keys stored in `.env.local`, never exposed in frontend code.
- **Scalability:** Horizontal scaling possible (stateless deployment).
- **Maintainability:** TypeScript, modular code, reusable components.
- **Accessibility:** WCAG 2.1 AA compliance for colors and interactions.

7. Actors & Use Cases

Actors

- **Trader/Investor** → Monitors live data, tracks trends.
- **Analyst/Researcher** → Studies market behavior, compares categories.
- **Casual Enthusiast** → Visual exploration of assets.

Use Cases

1. **UC-01:** User opens app → sees live crypto bubbles.
2. **UC-02:** User switches to Forex tab → forex currencies visualized.
3. **UC-03:** User hovers bubble → tooltip with asset details.
4. **UC-04:** User clicks bubble → popup with extended info.

5. **UC-05:** User searches “BTC” → Bitcoin bubble highlighted.
6. **UC-06:** API call fails → fallback dataset displayed + error notice.

8. Architecture

- **Next.js App Router** for layouts, pages, and routing.
- **D3.js** for rendering bubble simulations.
- **Tailwind CSS** for styling and responsive design.
- **TypeScript** for type safety.
- **Service Layer (/src/services/)** for API communication.
- **Reusable UI Components (/src/components/ui/)** for consistency.

9. Data Model

Interfaces in /src/types/:

```
interface BubbleData {
  id: string;
  symbol: string;
  name: string;
  marketCap: number;
  priceChange24h: number;
  volume24h: number;
  category: "crypto" | "forex" | "pair";
  size: number; // derived from marketCap/volume
  color: string;
  currentRate: number;
}
```

10. API Integration

- **Crypto Data:** CoinGecko API
- **Forex Data:** ExchangeRate-API
- Configurable endpoints via .env.local.

11. Deployment

- **Environment:** Node.js v18+
- **Build Commands:** npm run build → npm start
- **Supported Hosts:** Vercel, Hostinger, Netlify, or VPS with Docker.
- **CI/CD:** GitHub Actions for automated build & deploy (future).

12. Error Handling

- Show **loading spinners** during API calls.
- Display **error popups** if fetch fails.
- Serve **cached/fallback mock data** when APIs are down.

13. Security

- .env.local securely stores API keys.
- .gitignore includes .env*.
- HTTPS-only requests.
- No sensitive information stored in localStorage.

14. Future Enhancements

- User authentication and profiles.
- Personalized dashboards.
- Historical data with line/candle charts.
- Real-time alerts/notifications.
- Dedicated **mobile app** (React Native).

15. File Structure Overview

```
components.json
.eslint.config.mjs
next-env.d.ts
next.config.ts
package.json
postcss.config.mjs
README.md
SRS.md
public/
  ...assets
src/
  app/
    layout.tsx
    page.tsx
    prelaunch/page.tsx
    globals.css
  components/
    features/landing/crypto-bubble.tsx
    features/landing/dashboard.tsx
    layout/header.tsx
    ui/button.tsx
    ui/input.tsx
  lib/utils.ts
  services/coinApiService.ts
  services/forexApiService.ts
```

types/index.ts

16. Testing Strategy

- **Unit Tests:** For services and utils (Jest).
- **Integration Tests:** API + visualization interactions.
- **UI Testing:** Playwright/Cypress for end-to-end.
- **Performance Tests:** Lighthouse audits.

17. References

- Next.js Documentation
- D3.js Documentation
- Tailwind CSS Documentation
- CoinGecko API
- ExchangeRate-API