Link for Front end- https://github.com/JOSHUA-MUANLAL/ATG-4-frontend.git

This project is a Cryptocurrency Alert System designed to allow users to register, set alerts on specific cryptocurrencies, and receive email notifications when the price of their selected cryptocurrencies crosses a specified target. The back-end is built using Node.js, Express, MongoDB, and Socket.IO, while the front-end is implemented using Vue.js via CDN.

Key Components

1. **User Registration and Authentication:**
    - Users can register by providing their email, name, number, and address.
    - Passwords are securely hashed using `bcryptjs` before storage.
    - JWT tokens are used for authentication, allowing users to securely access their accounts and set alerts.
2. **Vue.js Front-End:**
    - The front-end is developed using Vue.js, loaded via CDN. This allows for a dynamic and reactive user interface without requiring a complex build process.
    - Users can interact with the system through a responsive UI that communicates with the back-end via API endpoints.
    - Real-time updates on cryptocurrency prices are displayed using Vue.js, providing a seamless experience for the user.
3. **Email Notifications:**
    - The system sends email notifications for user registration, OTP for password reset, and when a price alert target is reached.
    - Emails are sent using a custom `sendmail` function, ensuring timely and reliable delivery.
4. **Cryptocurrency Price Tracking:**
    - The application fetches cryptocurrency prices from the CoinGecko API.
    - Prices are cached using `node-cache` to reduce API calls and improve performance.
    - Users can set alerts for specific cryptocurrencies, and the system checks prices periodically to trigger alerts.
5. **Real-time Communication:**
    - Socket.IO is used to establish real-time communication between the server and client, enabling live updates of cryptocurrency prices on the front-end.

    - Users receive instant updates on price changes without needing to refresh the page.
6. **Alerting System:**
    - Users can set alerts for specific cryptocurrency prices via the Vue.js front-end.
    - The system checks prices against user-defined targets every 60 seconds and sends an email alert if the target price is reached.
    - After an alert is triggered, it is removed from the database to avoid repeated notifications.

Challenges Faced and Solutions

1. **Integrating Vue.js with a Non-Build Front-End Setup:**
    - **Challenge:** Loading Vue.js via CDN can limit the use of advanced Vue features and might complicate state management across components.
    - **Solution:** Kept the front-end structure simple, leveraging Vue.js directives for dynamic updates and ensuring that all core functionalities (like form submissions and data rendering) are handled efficiently. Used Vue.js reactivity system to bind data and update the UI in real-time.
2. **Handling Large Volumes of API Requests:**
    - **Challenge:** Frequent API requests to fetch cryptocurrency prices could lead to performance issues and API rate limiting.
    - **Solution:** Implemented caching with `node-cache` to store and reuse the latest fetched prices, reducing the frequency of API calls. Prices are fetched in intervals, ensuring efficient API usage.
3. **Real-time Price Updates with Vue.js and Socket.IO:**
    - **Challenge:** Providing real-time updates on cryptocurrency prices in the front-end using Vue.js.
    - **Solution:** Socket.IO was integrated with Vue.js to enable real-time communication between the server and client. This ensures that users receive live updates on cryptocurrency prices without any delays.
4. **Secure User Authentication:**
    - **Challenge:** Ensuring that user credentials are stored securely and preventing unauthorized access.
    - **Solution:** Used `bcryptjs` for password hashing and JWT tokens for secure authentication. Vue.js communicates with the back-end to handle user sessions securely, ensuring that sensitive data is protected.
5. **Managing Concurrent Alerts:**

   - **Challenge:** Handling multiple users setting alerts on various cryptocurrencies could potentially overload the system.
   - **Solution:** Systematically checked alerts using a periodic interval. Alerts are checked every 60 seconds, and once triggered, they are deleted from the database, reducing the load on the system and ensuring efficient operation.
6. **Email Notification Reliability:**
   - **Challenge:** Ensuring timely delivery of email notifications for various user actions.
   - **Solution:** Developed a robust email sending function (`sendmail`) that handles different types of email notifications. Implemented error handling to log issues during email sending, ensuring that the server remains stable.