# **Project overview**

This project is a React web application that gives a cryptocurrency database basic CRUD (Create, Read, Update, Delete) capability. In order to make sure that only individuals with permission to do so can access and edit the database, the application also has a user authentication system.

This application's function is to provide users with the ability to browse, add, modify, and delete cryptocurrencies. It offers a straightforward and user-friendly interface that makes it simple for users to manage the data.

# **Installation and Setup**

You must have Node.js and npm (Node Package Manager) installed on your computer in order to use this project. To start up the project, follow these steps:

1. GitHub clone the project repository.
2. On your terminal, go to the project directory.
3. To install the necessary dependencies, run npm install.
4. To launch the development server, type npm start.
5. To access the program, go to http://localhost:3000 in your web browser.

# **Authentication**

JSON Web Tokens (JWTs) are used by the application to authenticate users. The server creates a JWT and sends it back to the client when a user logs in. The JWT is then kept by the client in local storage and used to verify the user's identity with each future API request.

The steps involved in the authentication process are as follows:

1. The user fills out the login form with their email address and password (Login component).
2. The user's credentials are included in a POST request that the client makes to the /auth/login endpoint.
3. The server creates a JWT after verifying the user's credentials.
4. The JWT is returned to the client by the server.
5. The JWT is kept locally by the client.
6. For user authentication, the client provides the JWT along with each subsequent API call.

The client directs the user to the login page if the JWT is missing or invalid, and the server issues a 401 Unauthorized error.

The application employs a higher-order component (HOC) called withAuth to secure routes that demand authentication. withAuth verifies the existence of a valid JWT and, if necessary, redirects the user to the login page. This HOC can be used to wrap any component that needs authentication to guarantee that only authorized users can access it.

# **API Endpoints**

When a user wants to conduct CRUD operations on the bitcoin database, the application interacts with a server-side API. It makes use of the following API endpoints:

* GET /cryptocurrencies: Gets a database list of all cryptocurrencies.
* A single coin is retrieved from the database using the GET /cryptocurrencies/:id command.
* POST /cryptocurrencies: Updates the database with a new cryptocurrency.
* PUT /cryptocurrencies/:id updates a cryptocurrency by its ID that already exists in the database.
* A coin is removed from the database by its ID with the command DELETE /cryptocurrencies/:id.
* POST /auth/login: Returns a JWT after user authentication.

# **CRUD Operations**

For managing cryptocurrency, the program offers the following CRUD operations:

* Create: By completing the cryptocurrency form (CryptoForm component) and sending it to the server, users can add new coins to the database. The database record is added after the server has verified the data. The cryptocurrency list view then shows the new addition (CryptoList component).
* Read: The coin list view allows users to see a list of all cryptocurrencies in the database (CryptoList component). By selecting a particular cryptocurrency from the list view, which gives a detailed view of it, users may also view details about that particular coin (CryptoForm component).
* Update: By selecting a cryptocurrency in the list view, users can change its details by opening the cryptocurrency form (CryptoForm component), which already has the data they need filled in. After making any necessary changes, the user can submit the form to update the database entry.
* Delete: On the cryptocurrency list view, users can remove an existing cryptocurrency by clicking the delete button (CryptoList component). The database entry is deleted by the server, and the list display is updated to reflect the modification.

The API queries made to the server are used to implement all of these CRUD processes. The api.js file in the utils directory contains definitions for the API functions.

# **Conclusion**

In conclusion, this React project offers a straightforward and approachable interface for controlling a cryptocurrency database. It enables basic CRUD capability for handling cryptocurrency and has a user authentication system to ensure that only authorized users may access and alter the database. To further enhance the user experience, the project can be expanded to include further features like search and filtering capabilities.