Currency Calculator Application Presentation

I'm presenting the Currency Calculator application, a project designed to meet the requirements outlined in the task description. This application provides a user-friendly interface for converting currencies and managing exchange rates through a secure backend API.

Meeting the Task Requirements:

As per the task description, this project includes:

- **A REST API (built with Node.js and Express):** This API provides endpoints for:
    - Converting currencies between a base and target currency for a given amount.
    - Performing CRUD (Create, Read, Update, Delete) operations on currencies and their exchange rates.
    - Implementing a simple Authentication and Authorization mechanism to protect the CRUD operations.
- **A modern Graphical User Interface (built with React):** This interface allows users to:
    - Convert currencies using the API endpoints.
    - Log in to access protected functionalities.
    - Add and update existing currency exchange rates through forms that interact with the API's CRUD endpoints.

Technologies Used:

This application leverages the following technologies for its development:

- **Backend:**
    - **Node.js:** A JavaScript runtime environment for building the server-side application.
    - **Express:** A minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
    - **MySQL2:** 1 A MySQL client for Node.js, used for interacting with the database to store currency data and user information.
    - **jsonwebtoken (JWT):** Used for implementing JSON Web Tokens for user authentication and securing protected API endpoints.
    - **bcrypt:** A library for hashing passwords securely before storing them in the database.
    - **cors:** Enables Cross-Origin Resource Sharing, allowing the frontend to make requests to the backend API running on a different port or origin.
    - **dotenv:** Used to manage environment variables, keeping sensitive information like database credentials and the JWT secret key separate from the main codebase.
- **Frontend:**
    - **React:** A JavaScript library for building user interfaces or UI components. It allows for the creation of interactive and dynamic single-page applications.
    - **React Router DOM:** A library providing routing functionalities for React applications, enabling navigation between different views (e.g., login, signup, converter, currency management).
    - **Axios:** A promise-based HTTP client for making API requests from the frontend to the backend.

- **React Bootstrap:** A UI library that provides pre-built Bootstrap components as React components, facilitating the creation of a responsive and visually appealing interface.
- **Bootstrap:** A popular CSS framework for building responsive and mobile-first websites.

Key Functionalities Demonstration:

I will now briefly demonstrate the key functionalities of the application:

1. **Signup:** A user can create a new account through the signup form. This involves providing a username and password, which is then securely stored in the database after hashing.
2. **Login:** Existing users can log in using their registered username and password. Upon successful authentication, the backend API returns a JWT token, which is stored in the browser's local storage. This token is then used to authorize subsequent requests to protected API endpoints.
3. **Adding Currencies:** After logging in, a user can access the "Currency Form" to add new currencies and their corresponding exchange rates. This involves providing a currency code, name, and its rate relative to a base currency (implicitly managed by the user's data). This data is sent to the backend API via a POST request and stored in the database.
4. **Editing Currencies:** Similarly, logged-in users can edit the details (name and exchange rate) of existing currencies using the "Currency Form." When editing, the currency code is typically used to identify the record to be updated via a PUT request to the API.
5. **Deleting Currencies:** Logged-in users can also delete currencies they have added through the "Currency Form." This action sends a DELETE request to the API, identifying the currency to be removed by its code.
6. **Converting Currencies:** All users (even those not logged in, as this endpoint is not strictly protected) can use the "Currency Converter" to convert an amount from one selected currency to another. The frontend sends a GET request to the appropriate API endpoint, providing the base currency code, target currency code, and the amount. The API then returns the calculated converted value based on the stored exchange rates.