Full Stack Banking Application Project

Create a full-stack banking application that includes a React frontend and an Express backend using MongoDB Atlas for data storage. The frontend should be deployed on Netlify and the backend on Render or Cyclic.

Backend Development

1. Setup MongoDB Atlas

- Create a new MongoDB Atlas database.
- Set up a connection between your Express application and MongoDB.

2. Mongoose Integration

- Design Mongoose schemas to represent users (fields: ID, cash, credit, etc.).
- Implement CRUD operations for user management.

3. API Endpoints

- Develop the following RESTful endpoints:
    - **Add User**: Create a new user with default cash and credit values.
    - **Deposit**: Enable cash deposits into user accounts.
    - **Update Credit**: Allow updates to a user's credit limit.
    - **Withdraw Money**: Facilitate cash withdrawals from user accounts.
    - **Transfer Funds**: Enable money transfers between users.
    - **User Details**: Retrieve details of a specific user.
    - **All Users**: Fetch details of all users.
    - **Filter Users**: Filter users based on cash amount.
- Ensure error handling for scenarios like non-existent users or insufficient funds.

4. Deployment

- Deploy the Express server on Render or Cyclic, ensuring the database is connected.

Frontend Development

1. React Application Setup

- Create a new React application using `Vite`.
- Setup React Router and create at least three pages: Home, User Details, Transactions.

2. Frontend Features

- Build forms and components for user operations (adding users, making deposits, etc.).
- Use **Axios** or Fetch API to connect the frontend to your Express backend.
- Implement state management using React Context if needed.

3. User Interface

- Design a responsive and user-friendly interface.
- Optionally, use Material-UI to enhance the appearance of your UI components.

4. Deployment

- Deploy the React application on Netlify, ensuring it communicates with the backend.

Extra Challenges (Ninja Tasks)

User Authentication

- Add a basic authentication system (login/logout) to the application.

- Implement protected routes in React that require user authentication.
- Handle session management in both frontend and backend.

None Admin Users

- Create a none admin users, they will be able to:
  - Withdraw money.
  - Deposit cash to the account.
  - Transfer money to someone using his personal id(not a unique mongo id).

Additional Features

- Implement the "IsActive" property in the user model, impacting user transaction capabilities.
- Develop advanced user filtering and sorting capabilities in the frontend.

Submission Guidelines

- Submit GitHub repositories with detailed README files for both frontend and backend.
- Include clear documentation on API usage and frontend-backend integration.

Extra info

- You have access to the Q&A session from last week, which is still relevant for this project. Feel free to review it as needed.