Objective: Create a simple banking transactions web application that allows users to manage multiple accounts and transfer funds between them.

Technology Requirements:

- Use **Angular** for the front-end framework.
- Use **Bootstrap** or **Material UI** for styling.

Core Functionality: The web application should allow users to perform the following operations:

1. **Create a new user account** with an initial balance.
2. **Transfer funds** from one account to another.
3. **View transaction history** for any given account.

Requirements

1. **User Interface (UI) Design**:
    - Design an intuitive, user-friendly interface.
    - Use Angular's **FormBuilder** and **FormControls** to create UI components, ensuring the forms are structured and easy to manage.
2. **Account Creation with Conditional Styling**:
    - Include an option to select the account type (Chequing or Savings) via radio buttons.
    - Ensure a **button** is conditionally rendered based on the account type selected.
    - Apply different styles for the button based on the account type.
3. **Form Validations**:
    - Apply reasonable input validators. For example:
        - **Balance** cannot be negative.
        - Use suitable input types for different fields (e.g., balance > 0, account names with min/max character limits).
4. **Reusable Components**:
    - Create a custom, reusable **button component** and incorporate it into the application.
    - Place this component in a **shared module** separate from your main module to promote modularity.

5. **Routing and Navigation**:
    - Implement Angular Router to handle navigation between:
        - **Account transactions** page (for viewing and managing transfers).
        - **Transaction history** page (to see a list of previous transactions for a given account).
6. **Modular Codebase**:
    - Structure the code in a modular way, with custom components in separate, logically organized modules.
    - Keep the reusable component in a shared module that can be imported across different parts of the application.

Implementation Guidelines

This section provides a few guidelines to help you implement the key features effectively.

1. **Account Creation Page**:
    - Set up a form for account creation that allows users to enter an initial balance and select an account type (Chequing or Savings).
    - Use **Angular's FormBuilder** to create a structured form with **FormControls** for each input.
    - Add conditional styling for the submit button based on the account type selected.
2. **Fund Transfer**:
    - Design a form to allow the user to transfer funds between accounts.
    - Ensure inputs have **validators** (e.g., amounts cannot exceed account balances and cannot be negative).
    - Use **Angular's two-way data binding** to update account balances after each transaction.
3. **Transaction History**:
    - Create a component or page that retrieves and displays the transaction history for a selected account.
    - Implement a **filter or search feature** (**optional**) to help users quickly find specific transactions.
4. **Custom Reusable Button Component**:
    - Develop a **reusable button component** that supports custom styling based on account type.
    - Place this button in a **shared module** so it can be used across different parts of the application.
5. **Navigation Using Angular Router**:

    - Set up routes for navigating between:
        - The account creation form
        - The transaction form
        - The transaction history
    - Ensure each route has a corresponding component, and use Angular's **RouterModule** for navigation.
6. **Project Structure**:
    - Organize your code into modules to separate concerns.
    - Store the reusable button component in a dedicated **SharedModule** and import it where needed.

Submission Guidelines

- **Source Code**: Share your code in a GitHub repository.
- **README File**: Include a README file with instructions on how to build and run the application, including any dependencies and setup requirements.