Assignment Nineteen

Please read:

- This is your nineteen assignment which you need to submit on the LMS. This is an assignment on Redux.
- Work on the questions given below and be ready with your solution. You have to submit your codesandbox link below on this page.
- Late Submission: The deadline to submit this assignment is 19th February 2024, 6:00PM IST.

Important Instructions:

1. Make sure that you follow the rules as it will help you build your muscle.
2. Do not copy from someone else as that would be cheating.

challenge

1. Introduction

Purpose

The Financial Management App is designed to help users manage their income, expenses, and savings effectively. It provides a user-friendly interface for tracking financial transactions and generating reports to gain insights into their financial health.

2. Features

Income Management

- Users can add income entries, including a description and amount.
- They can view a list of their income transactions on the Income Page with sort and filter functionality.
- Overview
    - It uses Redux to fetch and store income data from the Redux store.
    - The component includes filters for sorting income by amount and filtering by category.
    - It provides a user-friendly form (`IncomeExpenseForm`) for adding new income entries.
    - The user can sort income by amount using a checkbox and filter it by category using a dropdown menu.
    - It displays a list of income transactions, including details like category, date, description, and amount.
    - The total income is calculated and displayed at the bottom of the page.

Expense Management

- Users can add expense entries, including a description, amount, and category.
- They can view a list of their expense transactions on the Expense Page with sort and filter functionality.
- Overview
    - It uses Redux to fetch and store expense data from the Redux store.
    - The component includes filters for sorting expenses by amount and filtering them by category.
    - It provides a form (`IncomeExpenseForm`) for adding new expense entries.
    - The user can sort expenses by amount using a checkbox, and filter them by category using a dropdown menu.
    - It displays a loading message if the expense data is being fetched.
    - The expense transactions are displayed as a list, including details like category, date, description, and amount.
    - The total expenses are calculated and displayed at the bottom of the page.

Savings Management

- Users can add savings entries, including a description and amount.
- They can view a list of their savings transactions on the Savings Page with sort and filter functionality.
- Overview
    - It uses Redux to fetch and store savings data from the Redux store.
    - The component includes filters for sorting savings by amount and filtering them by category.

    - It provides a form (`IncomeExpenseForm`) for adding new savings entries.
    - The user can sort savings by amount using a checkbox, and filter them by category using a dropdown menu.
    - It displays a loading message if the saving data is being fetched.
    - The saving transactions are displayed as a list, including details like category, date, description, and amount.
    - The total savings are calculated and displayed at the bottom of the page.

Financial Reports

- Users can generate financial reports, including:
    - Income vs. Expenses: A summary of total income, total expenses, and savings.
    - Expense Breakdown: A breakdown of expenses by category.
- Overview
    - It has a dropdown for selecting the type of report: "Income vs. Expenses" or "Expense Breakdown."
    - Clicking the "Generate Report" button calculates and displays the selected report type.
    - The component fetches income and expense data from a Redux store using useSelector.
    - Depending on the selected report type, it displays total income, total expenses, savings, or an expense breakdown.

API Endpoints

1. Fetching Income Data Endpoint:
    - URL: `"https://your-server-url/income"`
    - Purpose: This endpoint is used to retrieve income data from the server.
    - HTTP Method: GET
    - Usage:
        - The `fetchIncome` action makes a GET request to this endpoint to fetch income data.
        - Upon a successful response, it dispatches an action of type `"FETCH_INCOME_SUCCESS"` with the fetched data as the payload.
        - In case of an error, it dispatches an action of type `"FETCH_INCOME_FAILURE"`.
2. Fetching Savings Data Endpoint:
    - URL: `"https://your-server-url/savings"`
    - Purpose: This endpoint is used to retrieve savings data from the server.

  - HTTP Method: GET
  - Usage:
    - The `fetchSavings` action makes a GET request to this endpoint to fetch savings data.
    - Upon a successful response, it dispatches an action of type `"FETCH_SAVINGS_SUCCESS"` with the fetched data as the payload.
    - In case of an error, it dispatches an action of type `"FETCH_SAVINGS_FAILURE"`.

3. Fetching Expenses Data Endpoint:
  - URL: `"https://your-server-url/expenses"`
  - Purpose: This endpoint is used to retrieve expenses data from the server.
  - HTTP Method: GET
  - Usage:
    - The `fetchExpenses` action makes a GET request to this endpoint to fetch expenses data.
    - Upon a successful response, it dispatches an action of type `"FETCH_EXPENSES_SUCCESS"` with the fetched data as the payload.
    - In case of an error, it dispatches an action of type `"FETCH_EXPENSES_FAILURE"`.

4. Adding Entry Endpoint:
  - URL: `"https://your-server-url/add-{entry.type}"`
  - Purpose: This endpoint is used to add a new income, expense, or savings entry to the server.
  - HTTP Method: POST
  - Headers:
    - `"Content-Type": "application/json"`: Indicates that the request body is in JSON format.
  - Request Body: The action sends the entry data as JSON in the request body.
  - Usage:
    - The `addEntry` action makes a POST request to this endpoint to add a new entry of the specified type (income or expense).
    - The `entry` object contains the necessary data (description, amount, entry type).
    - Upon a successful response it dispatches either `"ADD_INCOME_SUCCESS"` or `"ADD_EXPENSE_SUCCESS" or "ADD_SAVINGS_SUCCESS"` action based on the `entry.type`.
    - In case of an error, it dispatches an action of type `"ADD_ENTRY_FAILURE"`.