**Using useReducer**

by traltb@fe.edu.vn

useReducer là một Hook trong React cho phép bạn quản lý state phức tạp trong component một cách hiệu quả. Đó là một sự thay thế cho useState, đặc biệt hữu ích khi state có nhiều trạng thái con hoặc logic cập nhật state phức tạp.

Cấu trúc cơ bản của useReducer như sau:

const [state, dispatch] = useReducer(reducer, initialState, init);

* reducer: Một hàm reducer, nhận vào state hiện tại và một action, trả về state mới.
* initialState: Giá trị khởi tạo cho state.
* init (tùy chọn): Một hàm khởi tạo, nhận vào initialState và trả về state khởi tạo.
* **state**: State hiện tại.
* **dispatch**: Một hàm để gửi action đến reducer.

**Tạo hàm reducer:** Hàm reducer có kiểu sau:

function reducer(state, action) {

    switch (action.type) {

      case 'ACTION\_TYPE\_1':

        // Logic cập nhật state cho action type 1

        return { ...state, /\* cập nhật state \*/ };

      case 'ACTION\_TYPE\_2':

        // Logic cập nhật state cho action type 2

        return { ...state, /\* cập nhật state \*/ };

      // ... các case khác

      default:

        return state;

    }

  }

* state là trạng thái hiện tại.
* action là một đối tượng chứa thông tin về hành động, ví dụ như action.type có thể là "ADD\_ITEM" hoặc "REMOVE\_ITEM".
* Hàm reducer nhận vào state hiện tại và action, sau đó trả về state mới dựa trên action.
* Sử dụng switch để xử lý các loại action khác nhau.
* Luôn trả về một state mới (không thay đổi state cũ) để đảm bảo tính immutable.

Tạo mới ứng dụng react tên exercise15-usereducer. Tạo mới thư mục components trong thư mục src

Làm các bài tập dưới đây:

**Exercise 1**: (Exercise 15)

Use useReducer to manage state in a simple counter component.

**Tạo file Counter.js trong thư mục src\components\Counter.js**

**Bước 1**: xác định hàm counterReducer sẽ nhận vào 2 tham số:

* state: Trạng thái hiện tại (bao gồm giá trị count).
* action: Một đối tượng hành động chứa thông tin về hành động cần thực hiện, ví dụ như INCREMENT, DECREMENT, hoặc RESET.

Bước 2: Xây dựng cấu trúc của hàm counterReducer:

// Định nghĩa hàm reducer để xử lý các hành động

function counterReducer(state, action) {

  switch (action.type) {

    case "INCREMENT":

      return { count: state.count + 1 };

    case "DECREMENT":

      return { count: state.count - 1 };

    case "RESET":

      return { count: 0 };

    default:

      return state;

  }

}

Bước 3: **Sử dụng useReducer trong component:**

Sử dụng useReducer để quản lý trạng thái count. Khi gọi useReducer, bạn cần truyền vào reducer (counterReducer) và giá trị trạng thái ban đầu ({ count: 0 }).

function Counter() {

  // Sử dụng useReducer để quản lý trạng thái count

  const [state, dispatch] = useReducer(counterReducer, { count: 0 });

  return (

    <div>

      <h1>Counter: {state.count}</h1>

      <button onClick={() => dispatch({ type: "INCREMENT" })}>+</button>

      <button onClick={() => dispatch({ type: "DECREMENT" })}>-</button>

      <button onClick={() => dispatch({ type: "RESET" })}>Reset</button>

    </div>

  );

}

export default Counter;

**Giải thích các phần trong mã nguồn:**

* useReducer(counterReducer, { count: 0 }):
  + counterReducer: Đây là hàm reducer, quản lý logic cập nhật trạng thái dựa trên hành động.
  + { count: 0 }: Đây là giá trị trạng thái ban đầu (số đếm bắt đầu là 0).
  + useReducer trả về một mảng có hai phần tử: state (trạng thái hiện tại) và dispatch (hàm gửi hành động).
* dispatch({ type: 'INCREMENT' }):
  + Khi người dùng nhấn nút "+", chúng ta gọi dispatch với hành động có type là 'INCREMENT', giúp tăng giá trị count.
* dispatch({ type: 'DECREMENT' }):
  + Khi người dùng nhấn nút "-", chúng ta gọi dispatch với hành động có type là 'DECREMENT', giúp giảm giá trị count.
* dispatch({ type: 'RESET' }):
  + Khi người dùng nhấn nút "Reset", chúng ta gọi dispatch với hành động có type là 'RESET', giúp đặt lại giá trị count về 0.

**Mã nguồn đầy đủ của file Counter.js:**

import React, { useReducer } from "react";

// Định nghĩa hàm reducer để xử lý các hành động

function counterReducer(state, action) {

  switch (action.type) {

    case "INCREMENT":

      return { count: state.count + 1 };

    case "DECREMENT":

      return { count: state.count - 1 };

    case "RESET":

      return { count: 0 };

    default:

      return state;

  }

}

function Counter() {

  // Sử dụng useReducer để quản lý trạng thái count

  const [state, dispatch] = useReducer(counterReducer, { count: 0 });

  return (

    <div>

      <h1>Counter: {state.count}</h1>

      <button onClick={() => dispatch({ type: "INCREMENT" })}>+</button>

      <button onClick={() => dispatch({ type: "DECREMENT" })}>-</button>

      <button onClick={() => dispatch({ type: "RESET" })}>Reset</button>

    </div>

  );

}

export default Counter;

* Chỉnh sửa file src/App.js như sau:

import React from "react";

import Counter from "./Counter";

function App() {

  return (

    <div>

      <Counter />

    </div>

  );

}

export default App;

Kết quả:

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 2: Tạo 1 form như sau: cho phép người dùng thay đổi Name, Age**

**A screenshot of a phone

AI-generated content may be incorrect.**

**Bước 1: Tạo file src/components/ChangeNameAge.js với nội dung sau:**

Xác định hàm Reducer

function formReducer(state, action) {

  switch (action.type) {

    case "SET\_NAME":

      return { ...state, name: action.value }; // Sử dụng value thay vì payload

    case "SET\_AGE":

      return { ...state, age: action.value }; // Sử dụng value thay vì payload

    default:

      return state;

  }

}

Bước 2: Sử dụng hàm useReducer trong component

function ChangeNameAge() {

  // Khởi tạo useReducer với reducer và initial state

  const [state, dispatch] = useReducer(formReducer, { name: "", age: "" });

  // Xử lý khi người dùng thay đổi input

  const handleNameChange = (e) => {

    dispatch({ type: "SET\_NAME", value: e.target.value }); // Truyền value vào action

  };

  const handleAgeChange = (e) => {

    dispatch({ type: "SET\_AGE", value: e.target.value }); // Truyền value vào action

  };

  return (

    <div>

      <div>

        <label>Name:</label>

        <input

          type="text"

          value={state.name}

          onChange={handleNameChange}

          placeholder="Input name"

        />

      </div>

      <div>

        <label>Age:</label>

        <input

          type="text"

          value={state.age}

          onChange={handleAgeChange}

          placeholder="Input age"

        />

      </div>

      <div>

        <h3>Name: {state.name}</h3>

        <h3>Age: {state.age}</h3>

      </div>

    </div>

  );

}

export default ChangeNameAge;

**Giải thích code:**

* **useReducer(reducer, initialState): Khởi tạo state và dispatch function.**
* handleChangeName và handleChangeAge: Hai hàm này được gọi khi giá trị của input thay đổi.
* event.target.value: Lấy giá trị mới từ input.
* dispatch: Gửi một action đến reducer. Action này có type là "SET\_NAME" hoặc "SET\_AGE" và value là giá trị mới từ input.
* **dispatch({ type: 'SET\_NAME', value: e.target.value }):**
* Khi người dùng thay đổi trường "Name", sự kiện onChange được gọi. Chúng ta dispatch hành động với type: 'SET\_NAME' và truyền value là giá trị từ trường nhập liệu (ví dụ: e.target.value).
* **dispatch({ type: 'SET\_AGE', value: e.target.value }):**
* Tương tự, khi người dùng thay đổi trường "Age", chúng ta dispatch hành động với type: 'SET\_AGE' và truyền value là giá trị từ trường nhập liệu.
* **Reducer cập nhật trạng thái:**
* Reducer nhận vào action.value và cập nhật giá trị tương ứng trong trạng thái, ví dụ, cập nhật state.name khi hành động SET\_NAME được gọi.

**Code đầy đủ:**

import React, { useReducer } from 'react';

function formReducer(state, action) {

    switch (action.type) {

      case "SET\_NAME":

        return { ...state, name: action.value };

      case "SET\_AGE":

        return { ...state, age: action.value };

      default:

        return state;

    }

  }

function ChangeNameAge() {

  // Khởi tạo useReducer với reducer và initial state

  const [state, dispatch] = useReducer(formReducer, { name: '', age: '' });

  // Xử lý khi người dùng thay đổi input

  const handleNameChange = (e) => {

    dispatch({ type: 'SET\_NAME', value: e.target.value }); // Truyền value vào action

  };

  const handleAgeChange = (e) => {

    dispatch({ type: 'SET\_AGE', value: e.target.value }); // Truyền value vào action

  };

  return (

    <div>

      <div>

        <label>Name:</label>

        <input

          type="text"

          value={state.name}

          onChange={handleNameChange}

          placeholder="Input name"

        />

      </div>

      <div>

        <label>Age:</label>

        <input

          type="text"

          value={state.age}

          onChange={handleAgeChange}

          placeholder="Input age"

        />

      </div>

      <div>

        <h3>Name: {state.name}</h3>

        <h3>Age: {state.age}</h3>

      </div>

    </div>

  );

}

export default ChangeNameAge;

Kết quả:

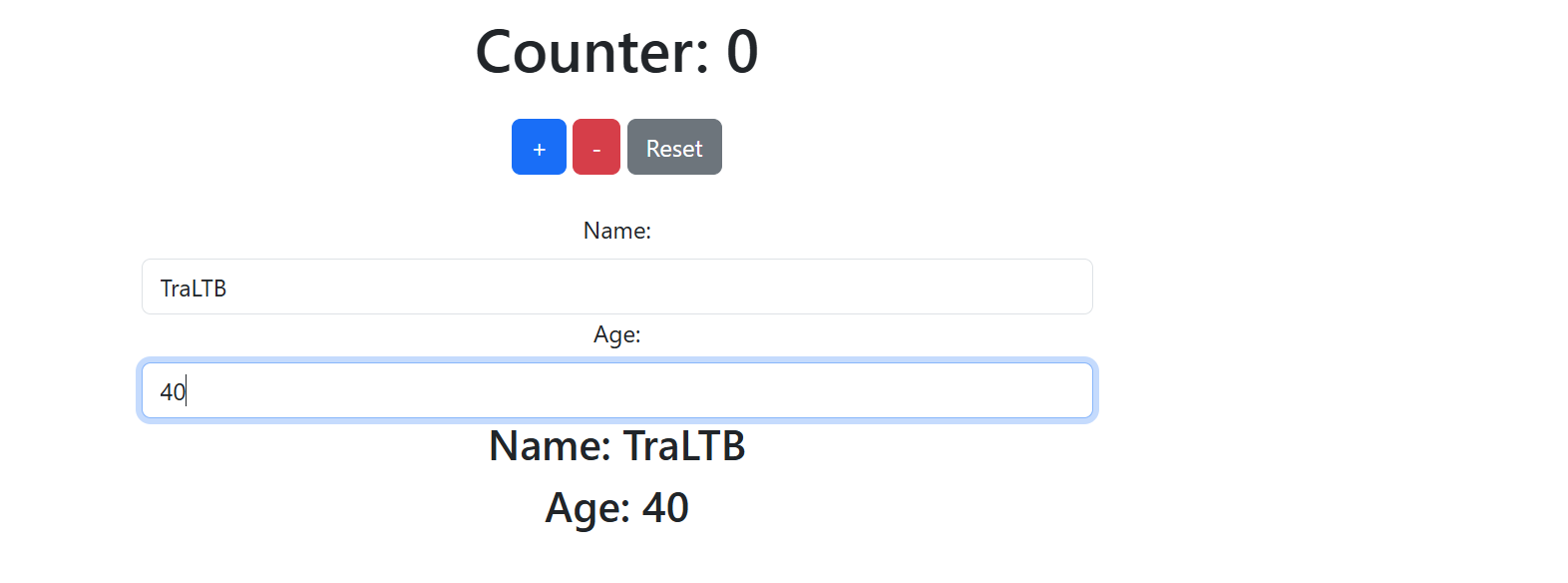
A screenshot of a computer

AI-generated content may be incorrect.

Hiệu chỉnh bài này, dùng react bootstrap để tạo ra giao diện Responsive

* Cài đặt react-bootstrap: npm install react-bootstrap bootstrap
* Import boostrap vào file index.js hoặc App.js:

import 'bootstrap/dist/css/bootstrap.min.css';



Exercise 3 Tạo mới một ứng dụng như sau:

**Tạo 1 form có ô input và button Add Item, cho phép:**

* **Thêm item (Add Item):**
  + Người dùng có thể nhập tên item vào một ô input.
  + Khi nhấn nút "Add Item", item mới sẽ được thêm vào danh sách với một id duy nhất.
  + Sau khi thêm, ô input sẽ được xóa trống.
* **Xóa item (Remove Item):**
  + Mỗi item trong danh sách có một nút "Remove".
  + Nhấn nút "Remove" sẽ xóa item tương ứng khỏi danh sách.
* **Hiển thị danh sách (Display List):**
  + Ứng dụng hiển thị danh sách tất cả các item đã thêm.
  + Mỗi item được hiển thị kèm theo nút "Remove".

Yêu cầu sử dụng:

* useReducer:

Được sử dụng để quản lý trạng thái của danh sách item.

Hàm listReducer xử lý các action ADD\_ITEM và REMOVE\_ITEM để cập nhật danh sách.

* useState:

Được sử dụng để quản lý giá trị của ô input (newItemName).

Kết quả:

A screenshot of a computer

AI-generated content may be incorrect.

Bước 1: Tạo ItemList.js trong src/components:

Bước 2: Định nghĩa hàm reducer để xử lý các hành động:

function listReducer(state, action) {

  switch (action.type) {

    case "ADD\_ITEM":

      return { ...state, items: [...state.items, action.item] };

    case "REMOVE\_ITEM":

      return {

        ...state,

        items: state.items.filter((item) => item.id !== action.id),

      };

    default:

      return state;

  }

}

Bước 2: Khởi tạo giá trị ban đầu của state

const initialState = {

  items: [],

};

Bước 3: Sử dụng useReducer để quản lý danh sách:

  const [state, dispatch] = useReducer(listReducer, initialState);

Bước 4: Sử dụng useState để lưu giá trị input của người dùng:

  const [newItemName, setNewItemName] = useState("");

Bước 5: Hàm xử lý hành động khi nhấn nút Add Item, Remove Item:

// Thêm một mục mới vào danh sách

  const handleAddItem = () => {

    if (newItemName) {

      const newItem = { id: Date.now(), name: newItemName }; // Tạo item với id duy nhất

      dispatch({ type: "ADD\_ITEM", item: newItem });

      setNewItemName(""); // Reset input sau khi thêm mục

    }

  };

  // Xóa một mục khỏi danh sách

  const handleRemoveItem = (id) => {

    dispatch({ type: "REMOVE\_ITEM", id });

  };

Bước 7: Sử dụng useReducer trong component

<Form>

            <Form.Group controlId="formItem">

              <Form.Label>Enter Item:</Form.Label>

              <Form.Control

                type="text"

                value={newItemName}

                onChange={(e) => setNewItemName(e.target.value)}

                placeholder="Enter item name"

              />

            </Form.Group>

            <Button variant="primary" onClick={handleAddItem}>

              Add Item

            </Button>

          </Form>

          <h3 className="mt-4">Item List:</h3>

          <ListGroup>

            {state.items.map((item) => (

              <ListGroup.Item

                key={item.id}

                className="d-flex justify-content-between"

              >

                {item.name}

                <Button

                  variant="danger"

                  onClick={() => handleRemoveItem(item.id)}

                >

                  Remove

                </Button>

              </ListGroup.Item>

            ))}

          </ListGroup>

Code đầy đủ:

import React, { useReducer, useState } from "react";

import { Button, Form, Container, Row, Col, ListGroup } from "react-bootstrap";

function listReducer(state, action) {

  switch (action.type) {

    case "ADD\_ITEM":

      return { ...state, items: [...state.items, action.item] };

    case "REMOVE\_ITEM":

      return {

        ...state,

        items: state.items.filter((item) => item.id !== action.id),

      };

    default:

      return state;

  }

}

const initialState = {

  items: [],

};

function ItemList() {

  // Khởi tạo useReducer với reducer và initial state

  const [state, dispatch] = useReducer(listReducer, initialState);

  // State quản lý input cho item name

  const [newItemName, setNewItemName] = useState("");

  // Thêm một mục mới vào danh sách

  const handleAddItem = () => {

    if (newItemName) {

      const newItem = { id: Date.now(), name: newItemName }; // Tạo item với id duy nhất

      dispatch({ type: "ADD\_ITEM", item: newItem });

      setNewItemName(""); // Reset input sau khi thêm mục

    }

  };

  // Xóa một mục khỏi danh sách

  const handleRemoveItem = (id) => {

    dispatch({ type: "REMOVE\_ITEM", id });

  };

  return (

    <Container className="mt-4">

      <Row>

        <Col md={6} className="offset-md-3">

          <Form>

            <Form.Group controlId="formItem">

              <Form.Label>Enter Item:</Form.Label>

              <Form.Control

                type="text"

                value={newItemName}

                onChange={(e) => setNewItemName(e.target.value)}

                placeholder="Enter item name"

              />

            </Form.Group>

            <Button variant="primary" onClick={handleAddItem}>

              Add Item

            </Button>

          </Form>

          <h3 className="mt-4">Item List:</h3>

          <ListGroup>

            {state.items.map((item) => (

              <ListGroup.Item

                key={item.id}

                className="d-flex justify-content-between"

              >

                {item.name}

                <Button

                  variant="danger"

                  onClick={() => handleRemoveItem(item.id)}

                >

                  Remove

                </Button>

              </ListGroup.Item>

            ))}

          </ListGroup>

        </Col>

      </Row>

    </Container>

  );

}

export default ItemList;

Exercise 4: Hiệu chỉnh yêu cầu trên.

* **Chỉnh sửa item (Edit Item):** Cho phép người dùng chỉnh sửa nội dung của item sau khi đã thêm vào danh sách.
* **Sắp xếp item (Sort Items):** Cung cấp chức năng sắp xếp danh sách item theo thứ tự bảng chữ cái (alphabetical), theo thời gian thêm vào (created time)
* **Lọc item (Filter Items):** Cho phép tìm kiếm item

A screenshot of a computer

AI-generated content may be incorrect.

Exercise 5 (Bài 2 trong Exercise 15\_useReducer)

Bước 1: **Tạo Component QuestionBank.js**

Tạo một file mới src/components/QuestionBank.js

Bước 2: Khởi tạo giá trị ban đầu của state

const initialState = {

    questions: [

      {

        id: 1,

        question: 'What is the capital of Australia?',

        options: ['Sydney', 'Canberra', 'Melbourne', 'Perth'],

        answer: 'Canberra',

      },

      {

        id: 2,

        question: 'Which planet is known as the Red Planet?',

        options: ['Venus', 'Mars', 'Jupiter', 'Saturn'],

        answer: 'Mars',

      },

      {

        id: 3,

        question: 'What is the largest ocean on Earth?',

        options: ['Atlantic Ocean', 'Indian Ocean', 'Pacific Ocean', 'Arctic Ocean'],

        answer: 'Pacific Ocean',

      },

    ],

    currentQuestion: 0,

    selectedOption: '',

    score: 0,

    showScore: false,

  };

* **questions**: Danh sách các câu hỏi trắc nghiệm, mỗi câu có:
  + question: Nội dung câu hỏi.
  + options: Các phương án trả lời.
  + answer: Đáp án đúng.
* **currentQuestion**: Theo dõi số thứ tự câu hỏi hiện tại.
* **selectedOption**: Lưu phương án người dùng đã chọn.
* **score**: Điểm số của người chơi.
* **showScore**: Nếu true, sẽ hiển thị điểm số khi làm xong bài kiểm tra.

**Bước 3**: Định nghĩa reducer function (quizReducer)

Reducer này quản lý các trạng thái như chọn đáp án, chuyển câu tiếp theo, và reset bài kiểm tra.

const initialState = {

    questions: [

      {

        id: 1,

        question: 'What is the capital of Australia?',

        options: ['Sydney', 'Canberra', 'Melbourne', 'Perth'],

        answer: 'Canberra',

      },

      {

        id: 2,

        question: 'Which planet is known as the Red Planet?',

        options: ['Venus', 'Mars', 'Jupiter', 'Saturn'],

        answer: 'Mars',

      },

      {

        id: 3,

        question: 'What is the largest ocean on Earth?',

        options: ['Atlantic Ocean', 'Indian Ocean', 'Pacific Ocean', 'Arctic Ocean'],

        answer: 'Pacific Ocean',

      },

    ],

    currentQuestion: 0,

    selectedOption: '',

    score: 0,

    showScore: false,

  };

  function quizReducer(state, action) {

    switch (action.type) {

      case 'SELECT\_OPTION':

        return { ...state, selectedOption: action.payload };

      case 'NEXT\_QUESTION':

        const isCorrect = state.selectedOption === state.questions[state.currentQuestion].answer;

        return {

          ...state,

          score: isCorrect ? state.score + 1 : state.score,

          currentQuestion: state.currentQuestion + 1,

          selectedOption: '',

          showScore: state.currentQuestion + 1 === state.questions.length, // Hiển thị điểm khi hết câu hỏi

        };

      case 'RESTART\_QUIZ':

        return {

          ...initialState, // Reset về trạng thái ban đầu

        };

      default:

        return state;

    }

  }

* SELECT\_OPTION

Khi người dùng chọn một phương án, cập nhật selectedOption.

* NEXT\_QUESTION
* Kiểm tra câu trả lời đúng hay sai và tăng điểm (score) nếu đúng.
* Tăng currentQuestion để chuyển sang câu tiếp theo.
* Reset selectedOption về "".
* Nếu đã đến câu cuối cùng (currentQuestion + 1 === questions.length), đặt showScore = true.
* RESTART\_QUIZ

Khi người dùng nhấn nút Restart, đặt lại trạng thái về initialState.

Bước 4: Sử dụng hàm reducer trong component QuestionBank

function QuestionBank() {

  const [state, dispatch] = useReducer(quizReducer, initialState);

  const { questions, currentQuestion, selectedOption, score, showScore } =

    state;

* **useReducer(quizReducer, initialState)**:
  + state: Trạng thái hiện tại.
  + dispatch: Hàm để gửi hành động (action) cập nhật trạng thái.

**Bước 5: Hiển thị kết quả nếu bài kiểm tra kết thúc**

{showScore ? (

          <div className="text-center">

            <h2>

              Your Score: {score} / {questions.length}

            </h2>

            <Button variant="primary" onClick={handleRestartQuiz}>

              Restart Quiz

            </Button>

          </div>

 Nếu showScore = true, hiển thị điểm số.

 Nút "Restart Quiz" gọi dispatch({ type: "RESTART\_QUIZ" }) để làm lại bài kiểm tra.

**Ngược lại:**

**Hiển thị câu hỏi và các phương án lựa chọn**

) : (

          <div>

            <h4>

              Question {questions[currentQuestion].id}:<br />

              {questions[currentQuestion].question}

            </h4>

            <div className="mt-3">

              {questions[currentQuestion].options.map((option, index) => (

                <Button

                  key={index}

                  variant={

                    selectedOption === option ? "success" : "outline-secondary"

                  }

                  className="m-2"

                  onClick={() => handleOptionSelect(option)}

                >

                  {option}

                </Button>

              ))}

            </div>

 Hiển thị **câu hỏi hiện tại**.

 Duyệt qua options để tạo danh sách các nút chọn đáp án.

 Khi người dùng chọn một phương án, dispatch({ type: "SELECT\_OPTION", option }) sẽ cập nhật selectedOption.

**Bước 6: Hiển thị nút "Next Question" hoặc "Finish Quiz"**

<Button

              variant="primary"

              className="mt-3"

              disabled={!selectedOption}

              onClick={handleNextQuestion}

            >

              {currentQuestion === questions.length - 1

                ? "Finish Quiz"

                : "Next Question"}

            </Button>

          </div>

        )}

* **disabled={!selectedOption}**: Chỉ cho phép bấm khi đã chọn phương án.
* Nếu là câu cuối cùng, nút sẽ hiển thị **"Finish Quiz"**, nếu không sẽ là **"Next Question"**.
* Khi bấm, dispatch({ type: "NEXT\_QUESTION" }) sẽ chuyển sang câu tiếp theo.

Kết quả:

A screenshot of a question

AI-generated content may be incorrect.

A screenshot of a quiz

AI-generated content may be incorrect.

Code đầy đủ:

import React, { useReducer } from "react";

import { Button, Container, Card } from "react-bootstrap";

const initialState = {

  questions: [

    {

      id: 1,

      question: "What is the capital of Australia?",

      options: ["Sydney", "Canberra", "Melbourne", "Perth"],

      answer: "Canberra",

    },

    {

      id: 2,

      question: "Which planet is known as the Red Planet?",

      options: ["Venus", "Mars", "Jupiter", "Saturn"],

      answer: "Mars",

    },

    {

      id: 3,

      question: "What is the largest ocean on Earth?",

      options: [

        "Atlantic Ocean",

        "Indian Ocean",

        "Pacific Ocean",

        "Arctic Ocean",

      ],

      answer: "Pacific Ocean",

    },

  ],

  currentQuestion: 0,

  selectedOption: "",

  score: 0,

  showScore: false,

};

function quizReducer(state, action) {

  switch (action.type) {

    case "SELECT\_OPTION":

      return { ...state, selectedOption: action.payload };

    case "NEXT\_QUESTION":

      const isCorrect =

        state.selectedOption === state.questions[state.currentQuestion].answer;

      return {

        ...state,

        score: isCorrect ? state.score + 1 : state.score,

        currentQuestion: state.currentQuestion + 1,

        selectedOption: "",

        showScore: state.currentQuestion + 1 === state.questions.length, // Hiển thị điểm khi hết câu hỏi

      };

    case "RESTART\_QUIZ":

      return {

        ...initialState, // Reset về trạng thái ban đầu

      };

    default:

      return state;

  }

}

// Component chính

function QuestionBank() {

  const [state, dispatch] = useReducer(quizReducer, initialState);

  const { questions, currentQuestion, selectedOption, score, showScore } =

    state;

  const handleOptionSelect = (option) => {

    dispatch({ type: "SELECT\_OPTION", payload: option });

  };

  const handleNextQuestion = () => {

    dispatch({ type: "NEXT\_QUESTION" });

  };

  const handleRestartQuiz = () => {

    dispatch({ type: "RESTART\_QUIZ" });

  };

  return (

    <Container className="mt-4">

      <Card className="p-4">

        {showScore ? (

          <div className="text-center">

            <h2>

              Your Score: {score} / {questions.length}

            </h2>

            <Button variant="primary" onClick={handleRestartQuiz}>

              Restart Quiz

            </Button>

          </div>

        ) : (

          <div>

            <h4>

              Question {questions[currentQuestion].id}:<br />

              {questions[currentQuestion].question}

            </h4>

            <div className="mt-3">

              {questions[currentQuestion].options.map((option, index) => (

                <Button

                  key={index}

                  variant={

                    selectedOption === option ? "success" : "outline-secondary"

                  }

                  className="m-2"

                  onClick={() => handleOptionSelect(option)}

                >

                  {option}

                </Button>

              ))}

            </div>

            <Button

              variant="primary"

              className="mt-3"

              disabled={!selectedOption}

              onClick={handleNextQuestion}

            >

              {currentQuestion === questions.length - 1

                ? "Finish Quiz"

                : "Next Question"}

            </Button>

          </div>

        )}

      </Card>

    </Container>

  );

}

export default QuestionBank;

Exercise 6 Hiệu chỉnh bài 5, Thêm một số tính năng vào bài QuestionBanks như sau:

* + 1. Hiển thị phản hồi đúng:

 Nếu đúng: ✅ Hiển thị **"Correct! 🎉"**

 Nếu sai: ❌ Hiển thị **"Incorrect! The correct answer is ..."**

**A screenshot of a quiz

AI-generated content may be incorrect.**

**Hoặc có thể sử dụng icon từ thư viện react-icons, hãy cài đặt bằng lệnh: npm install react-icons**

import { FaCheckCircle, FaTimesCircle } from "react-icons/fa";

* + 1. Hiển thị tiến trình làm bài, mỗi câu hỏi dạng 1/3 hoặc 2/3, 3/3
    2. Thêm đồng hồ đếm ngược cho mỗi câu, mỗi câu 10s, nếu <5s thì sẽ hiển thị text màu đỏ để cảnh báo sắp hết giờ.
    3. Lưu điểm cao vào Local Storage, hiển thị khi kết thúc quiz.

Exercise 7: Làm bài Lab 4 (All About Hook)