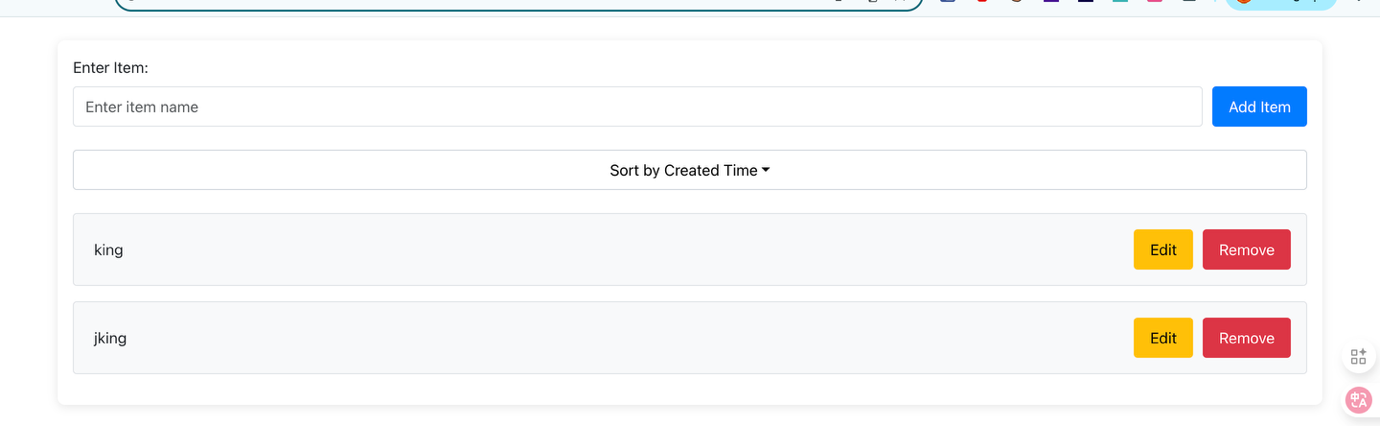
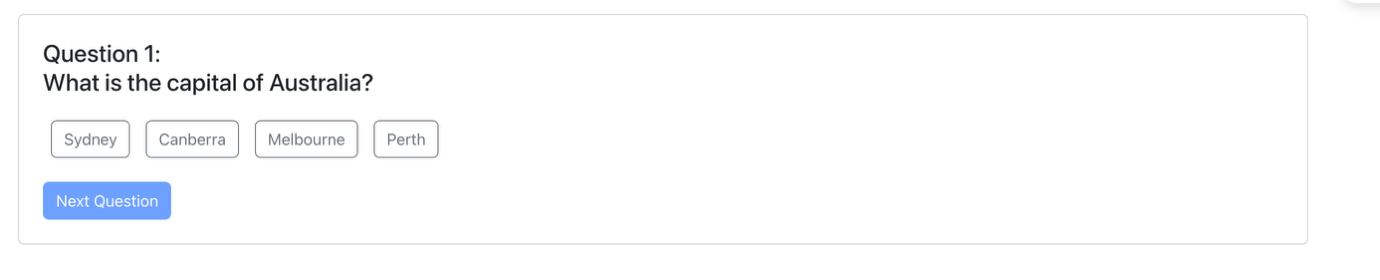
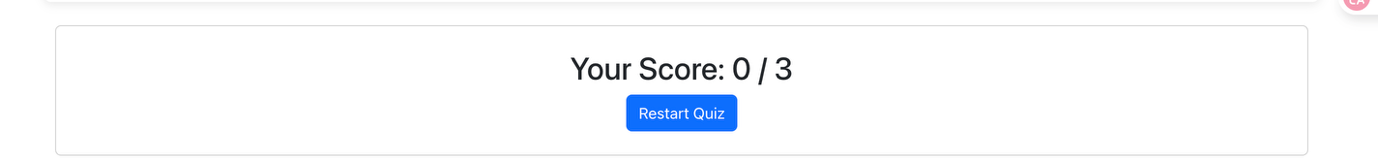
Ex: 4:



import *React*, { *useReducer*, *useState* } from "react";  
import { *Button*, Form, *Container*, *Row*, *Col*, ListGroup, InputGroup, Dropdown } 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),  
 };  
 case "EDIT\_ITEM":  
 return {  
 ...state,  
 items: state.items.map((item) =>  
 item.id === action.id ? { ...item, name: action.name } : item  
 ),  
 };  
 default:  
 return state;  
 }  
}  
  
const initialState = {  
 items: [],  
};  
  
function *ItemList*() {  
 const [state, dispatch] = *useReducer*(listReducer, initialState);  
 const [newItemName, setNewItemName] = *useState*("");  
 const [editItemId, setEditItemId] = *useState*(null);  
 const [editItemName, setEditItemName] = *useState*("");  
 const [sortType, setSortType] = *useState*("created"); // Default to "Sort by Created Time" as per image  
  
 const handleAddItem = () => {  
 if (newItemName.trim()) {  
 const newItem = { id: *Date*.now(), name: newItemName.trim(), createdAt: new Date() };  
 dispatch({ type: "ADD\_ITEM", item: newItem });  
 setNewItemName("");  
 }  
 };  
  
 const handleRemoveItem = (id) => {  
 dispatch({ type: "REMOVE\_ITEM", id });  
 };  
  
 const handleEditItem = (id, name) => {  
 setEditItemId(id);  
 setEditItemName(name);  
 };  
  
 const handleSaveEdit = (id) => {  
 if (editItemName.trim()) {  
 dispatch({ type: "EDIT\_ITEM", id, name: editItemName.trim() });  
 setEditItemId(null);  
 setEditItemName("");  
 }  
 };  
  
 const handleSort = (type) => {  
 setSortType(type);  
 };  
  
 const sortedItems = state.items.sort((a, b) => {  
 if (sortType === "alphabetical") return a.name.localeCompare(b.name);  
 return sortType === "created" ? a.createdAt - b.createdAt : 0;  
 });  
  
 return (  
 <Container className="mt-4 p-3" style={{ backgroundColor: "#fff", borderRadius: "8px", boxShadow: "0 2px 10px rgba(0,0,0,0.1)" }}>  
 <Row>  
 <Col xs={12}>  
 <Form>  
 <Form.Group controlId="formItem" className="mb-4">  
 <Form.Label>Enter Item:</Form.Label>  
 <InputGroup>  
 <Form.Control  
 type="text"  
 value={newItemName}  
 onChange={(e) => setNewItemName(e.target.value)}  
 placeholder="Enter item name"  
 style={{ borderRadius: "4px" }}  
 />  
 <Button  
 variant="primary"  
 onClick={handleAddItem}  
 style={{ backgroundColor: "#007bff", borderColor: "#007bff", padding: "8px 16px", borderRadius: "4px", marginLeft: "10px" }}  
 >  
 Add Item  
 </Button>  
 </InputGroup>  
 </Form.Group>  
 <Form.Group controlId="formSort" className="mb-4">  
 <Dropdown onSelect={handleSort}>  
 <Dropdown.Toggle  
 variant="light"  
 id="dropdown-sort"  
 style={{ width: "100%", border: "1px solid #ced4da", borderRadius: "4px", backgroundColor: "#fff", color: "#000", padding: "8px" }}  
 >  
 Sort by {sortType === "alphabetical" ? "Name" : "Created Time"}  
 </Dropdown.Toggle>  
 <Dropdown.Menu>  
 <Dropdown.Item eventKey="alphabetical">Sort by Name</Dropdown.Item>  
 <Dropdown.Item eventKey="created">Sort by Created Time</Dropdown.Item>  
 </Dropdown.Menu>  
 </Dropdown>  
 </Form.Group>  
 </Form>  
  
 <ListGroup>  
 {sortedItems.map((item) => (  
 <ListGroup.Item  
 key={item.id}  
 className="d-flex justify-content-between align-items-center mb-3 p-3"  
 style={{ borderRadius: "4px", border: "1px solid #dee2e6", backgroundColor: "#f8f9fa" }}  
 >  
 {editItemId === item.id ? (  
 <InputGroup style={{ flexGrow: 1 }}>  
 <Form.Control  
 type="text"  
 value={editItemName}  
 onChange={(e) => setEditItemName(e.target.value)}  
 style={{ borderRadius: "4px" }}  
 />  
 <Button  
 variant="success"  
 onClick={() => handleSaveEdit(item.id)}  
 style={{ backgroundColor: "#28a745", borderColor: "#28a745", padding: "8px 16px", borderRadius: "4px", marginLeft: "10px" }}  
 >  
 Save  
 </Button>  
 <Button  
 variant="secondary"  
 onClick={() => setEditItemId(null)}  
 style={{ backgroundColor: "#6c757d", borderColor: "#6c757d", padding: "8px 16px", borderRadius: "4px", marginLeft: "10px" }}  
 >  
 Cancel  
 </Button>  
 </InputGroup>  
 ) : (  
 <>  
 <span style={{ flexGrow: 1, fontSize: "1rem", padding: "5px" }}>{item.name}</span>  
 <div>  
 <Button  
 variant="warning"  
 onClick={() => handleEditItem(item.id, item.name)}  
 style={{ backgroundColor: "#ffc107", borderColor: "#ffc107", padding: "8px 16px", borderRadius: "4px", marginRight: "10px" }}  
 >  
 Edit  
 </Button>  
 <Button  
 variant="danger"  
 onClick={() => handleRemoveItem(item.id)}  
 style={{ backgroundColor: "#dc3545", borderColor: "#dc3545", padding: "8px 16px", borderRadius: "4px" }}  
 >  
 Remove  
 </Button>  
 </div>  
 </>  
 )}  
 </ListGroup.Item>  
 ))}  
 </ListGroup>  
 </Col>  
 </Row>  
 </Container>  
 );  
}  
  
export default *ItemList*;

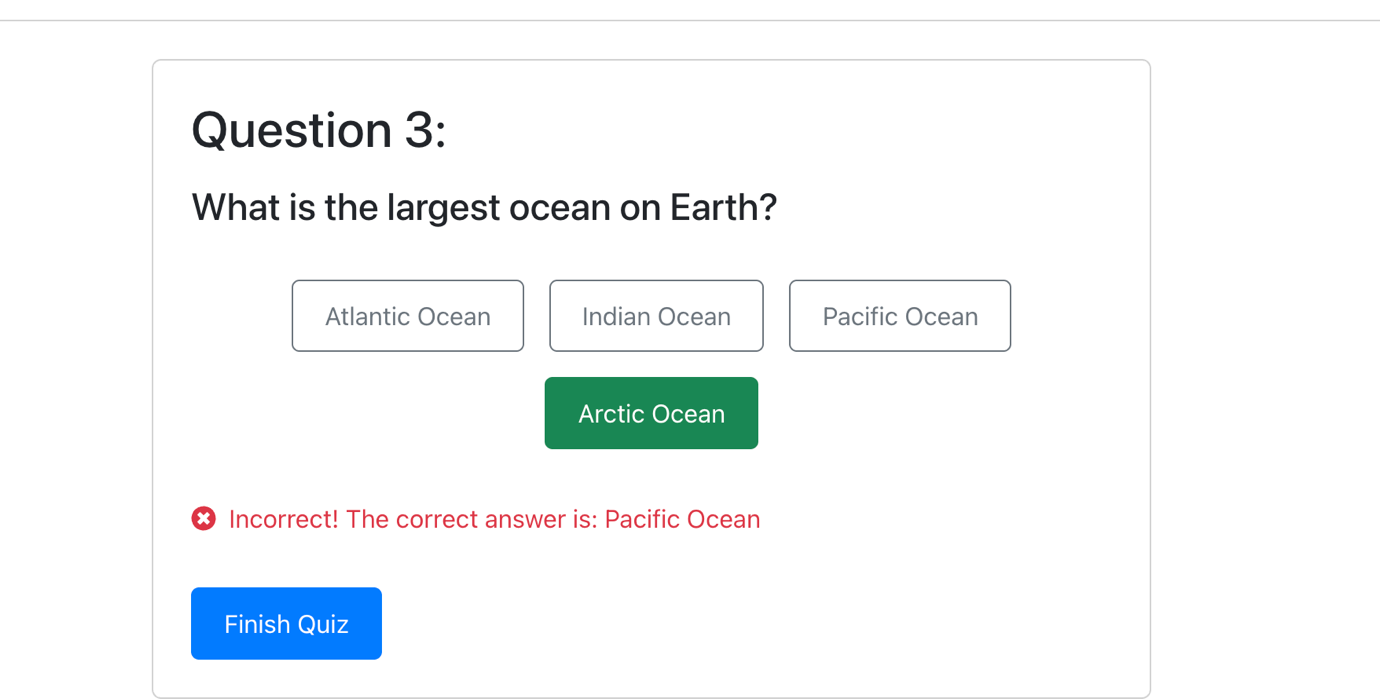
Exercise 5:

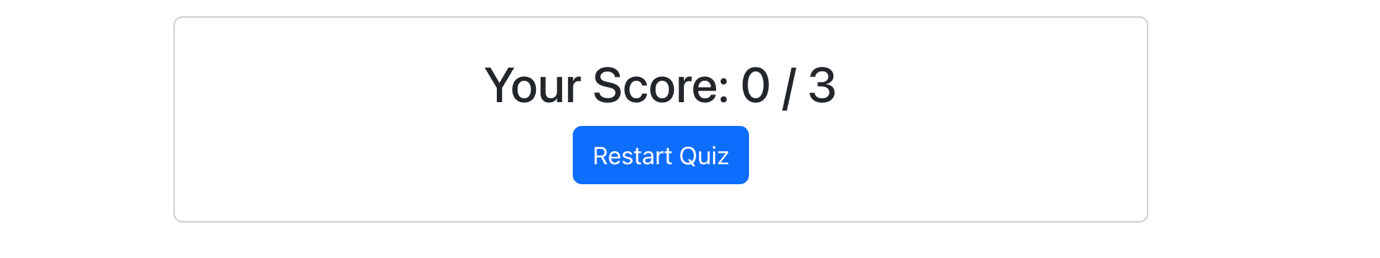




import *React*, { *useReducer*, *useState* } from "react";  
import { *Button*, Form, *Container*, *Row*, *Col*, ListGroup, InputGroup, Dropdown } from "react-bootstrap";  
  
function listReducer(state, action) {  
 switch (action.type) {  
 case "ADD\_ITEM":  
 return { ...state, items: [...state.items, action.item] };  
 case "REMOVE\_ITEM":  
 return {  
 ...state,  
 items: state.items.filter((item) => item.id !== action.id),  
 };  
 case "EDIT\_ITEM":  
 return {  
 ...state,  
 items: state.items.map((item) =>  
 item.id === action.id ? { ...item, name: action.name } : item  
 ),  
 };  
 default:  
 return state;  
 }  
}  
  
const initialState = {  
 items: [],  
};  
  
function *ItemList*() {  
 const [state, dispatch] = *useReducer*(listReducer, initialState);  
 const [newItemName, setNewItemName] = *useState*("");  
 const [editItemId, setEditItemId] = *useState*(null);  
 const [editItemName, setEditItemName] = *useState*("");  
 const [sortType, setSortType] = *useState*("created"); // Default to "Sort by Created Time" as per image  
  
 const handleAddItem = () => {  
 if (newItemName.trim()) {  
 const newItem = { id: *Date*.now(), name: newItemName.trim(), createdAt: new Date() };  
 dispatch({ type: "ADD\_ITEM", item: newItem });  
 setNewItemName("");  
 }  
 };  
  
 const handleRemoveItem = (id) => {  
 dispatch({ type: "REMOVE\_ITEM", id });  
 };  
  
 const handleEditItem = (id, name) => {  
 setEditItemId(id);  
 setEditItemName(name);  
 };  
  
 const handleSaveEdit = (id) => {  
 if (editItemName.trim()) {  
 dispatch({ type: "EDIT\_ITEM", id, name: editItemName.trim() });  
 setEditItemId(null);  
 setEditItemName("");  
 }  
 };  
  
 const handleSort = (type) => {  
 setSortType(type);  
 };  
  
 const sortedItems = state.items.sort((a, b) => {  
 if (sortType === "alphabetical") return a.name.localeCompare(b.name);  
 return sortType === "created" ? a.createdAt - b.createdAt : 0;  
 });  
  
 return (  
 <Container className="mt-4 p-3" style={{ backgroundColor: "#fff", borderRadius: "8px", boxShadow: "0 2px 10px rgba(0,0,0,0.1)" }}>  
 <Row>  
 <Col xs={12}>  
 <Form>  
 <Form.Group controlId="formItem" className="mb-4">  
 <Form.Label>Enter Item:</Form.Label>  
 <InputGroup>  
 <Form.Control  
 type="text"  
 value={newItemName}  
 onChange={(e) => setNewItemName(e.target.value)}  
 placeholder="Enter item name"  
 style={{ borderRadius: "4px" }}  
 />  
 <Button  
 variant="primary"  
 onClick={handleAddItem}  
 style={{ backgroundColor: "#007bff", borderColor: "#007bff", padding: "8px 16px", borderRadius: "4px", marginLeft: "10px" }}  
 >  
 Add Item  
 </Button>  
 </InputGroup>  
 </Form.Group>  
 <Form.Group controlId="formSort" className="mb-4">  
 <Dropdown onSelect={handleSort}>  
 <Dropdown.Toggle  
 variant="light"  
 id="dropdown-sort"  
 style={{ width: "100%", border: "1px solid #ced4da", borderRadius: "4px", backgroundColor: "#fff", color: "#000", padding: "8px" }}  
 >  
 Sort by {sortType === "alphabetical" ? "Name" : "Created Time"}  
 </Dropdown.Toggle>  
 <Dropdown.Menu>  
 <Dropdown.Item eventKey="alphabetical">Sort by Name</Dropdown.Item>  
 <Dropdown.Item eventKey="created">Sort by Created Time</Dropdown.Item>  
 </Dropdown.Menu>  
 </Dropdown>  
 </Form.Group>  
 </Form>  
  
 <ListGroup>  
 {sortedItems.map((item) => (  
 <ListGroup.Item  
 key={item.id}  
 className="d-flex justify-content-between align-items-center mb-3 p-3"  
 style={{ borderRadius: "4px", border: "1px solid #dee2e6", backgroundColor: "#f8f9fa" }}  
 >  
 {editItemId === item.id ? (  
 <InputGroup style={{ flexGrow: 1 }}>  
 <Form.Control  
 type="text"  
 value={editItemName}  
 onChange={(e) => setEditItemName(e.target.value)}  
 style={{ borderRadius: "4px" }}  
 />  
 <Button  
 variant="success"  
 onClick={() => handleSaveEdit(item.id)}  
 style={{ backgroundColor: "#28a745", borderColor: "#28a745", padding: "8px 16px", borderRadius: "4px", marginLeft: "10px" }}  
 >  
 Save  
 </Button>  
 <Button  
 variant="secondary"  
 onClick={() => setEditItemId(null)}  
 style={{ backgroundColor: "#6c757d", borderColor: "#6c757d", padding: "8px 16px", borderRadius: "4px", marginLeft: "10px" }}  
 >  
 Cancel  
 </Button>  
 </InputGroup>  
 ) : (  
 <>  
 <span style={{ flexGrow: 1, fontSize: "1rem", padding: "5px" }}>{item.name}</span>  
 <div>  
 <Button  
 variant="warning"  
 onClick={() => handleEditItem(item.id, item.name)}  
 style={{ backgroundColor: "#ffc107", borderColor: "#ffc107", padding: "8px 16px", borderRadius: "4px", marginRight: "10px" }}  
 >  
 Edit  
 </Button>  
 <Button  
 variant="danger"  
 onClick={() => handleRemoveItem(item.id)}  
 style={{ backgroundColor: "#dc3545", borderColor: "#dc3545", padding: "8px 16px", borderRadius: "4px" }}  
 >  
 Remove  
 </Button>  
 </div>  
 </>  
 )}  
 </ListGroup.Item>  
 ))}  
 </ListGroup>  
 </Col>  
 </Row>  
 </Container>  
 );  
}  
  
export default *ItemList*;

Ex6:





import *React*, { *useReducer*, *useState* } from "react";  
import { *Button*, *Container*, Card, *Row*, *Col* } from "react-bootstrap";  
import { *FaTimesCircle* } from "react-icons/fa";  
  
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,  
 feedback: null,  
};  
  
function quizReducer(state, action) {  
 switch (action.type) {  
 case "SELECT\_OPTION":  
 const isCorrect = action.payload === state.questions[state.currentQuestion].answer;  
 return {  
 ...state,  
 selectedOption: action.payload,  
 feedback: isCorrect  
 ? { correct: true, message: "Correct! 🎉" }  
 : { correct: false, message: `Incorrect! The correct answer is: ${state.questions[state.currentQuestion].answer}` },  
 };  
 case "NEXT\_QUESTION":  
 return {  
 ...state,  
 currentQuestion: state.currentQuestion + 1,  
 selectedOption: "",  
 feedback: null,  
 showScore: state.currentQuestion + 1 === state.questions.length,  
 };  
 case "RESTART\_QUIZ":  
 return { ...initialState };  
 default:  
 return state;  
 }  
}  
  
function *QuestionBankAdvanced*() {  
 const [state, dispatch] = *useReducer*(quizReducer, initialState);  
 const { questions, currentQuestion, selectedOption, showScore, feedback } = 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">  
 <Row>  
 <Col xs={12} md={8} lg={6} className="mx-auto">  
 <Card className="p-4">  
 {showScore ? (  
 <div className="text-center">  
 <h2>Your Score: 0 / {questions.length}</h2>  
 <Button variant="primary" onClick={handleRestartQuiz}>  
 Restart Quiz  
 </Button>  
 </div>  
 ) : (  
 <div>  
 <h2 className="mb-3">Question {questions[currentQuestion].id}:</h2>  
 <h4 className="mb-4">{questions[currentQuestion].question}</h4>  
 <div className="d-flex flex-wrap justify-content-center mb-4">  
 {questions[currentQuestion].options.map((option, index) => (  
 <Button  
 key={index}  
 variant={selectedOption === option ? "success" : "outline-secondary"}  
 className="m-2"  
 style={{  
 padding: "10px 20px",  
 borderRadius: "5px",  
 minWidth: "120px",  
 }}  
 onClick={() => handleOptionSelect(option)}  
 >  
 {option}  
 </Button>  
 ))}  
 </div>  
 {feedback && (  
 <p className="text-danger d-flex align-items-center mb-3">  
 <FaTimesCircle className="me-2" />  
 {feedback.message}  
 </p>  
 )}  
 <Button  
 variant="primary"  
 className="mt-3"  
 disabled={!selectedOption}  
 onClick={handleNextQuestion}  
 style={{ backgroundColor: "#007bff", borderColor: "#007bff", padding: "10px 20px", borderRadius: "5px" }}  
 >  
 {currentQuestion === questions.length - 1 ? "Finish Quiz" : "Next Question"}  
 </Button>  
 </div>  
 )}  
 </Card>  
 </Col>  
 </Row>  
 </Container>  
 );  
}  
  
export default *QuestionBankAdvanced*;