**Recipe Manager Application - Detailed Report**

The Recipe Manager Application is a React-based web application designed to help users manage recipes efficiently. It includes features like adding, editing, deleting, filtering, sorting, and sharing recipes. The application uses several React components and hooks to create a dynamic and responsive user interface. This report provides a comprehensive explanation of the code, detailing the purpose and functionality of each component and the usage of React hooks and features.

**React Components Overview**

The application is divided into modular React components, each serving a specific purpose. These components are described below:

**1. App Component**

The App component serves as the main entry point for the application. It manages the overall layout, routing, and core logic.

**Key Features:**

* **Routing**: Utilizes react-router-dom to provide navigation between different sections of the app.
* **State Management**: Uses useState to manage application states like recipes, filters, selected recipes, and pagination.
* **Data Fetching**: Fetches recipes from a JSON server using useEffect.
* **Pagination**: Implements logic to display a fixed number of recipes per page.
* **Recipe Sharing**: Allows users to share selected recipes via email using the emailjs library.

**Hooks Used:**

* **useState**: Manages states like recipes, search query, selected filters, and pagination.
* **useEffect**: Handles side effects like data fetching and resetting the current page when filters or sorting options change.

**2. CreateRecipe Component**

This component provides a form for users to add new recipes to the database.

**Key Features:**

* **Interactive Form**: Includes fields for recipe title, description, ingredients, steps, tags, and difficulty.
* **Validation**: Ensures users cannot add more than three tags and removes empty entries from ingredients and steps.
* **Data Submission**: Sends the new recipe to the server using an HTTP POST request.

**Hooks Used:**

* **useState**: Manages the form data and ensures it is reset after submission.

**3. RecipeCard Component**

Displays individual recipes in a visually appealing card format.

**Key Features:**

* **Details Display**: Shows recipe title, description, ingredients, steps, tags, and difficulty level.
* **Edit Mode**: Allows users to edit recipes in place.
* **Hover Actions**: Displays buttons for editing and deleting when the user hovers over the card.
* **Selection**: Provides a checkbox to select recipes for sharing.

**Hooks Used:**

* **useState**: Tracks hover status, edit mode, and selection status of the recipe card.

**4. FeaturedRecipe Component**

Highlights the most recently added recipe in the database.

**Key Features:**

* **Real-time Updates**: Periodically fetches data from the server to ensure the featured recipe is always up-to-date.
* **Detailed View**: Displays all key details of the featured recipe, including ingredients and steps.

**Hooks Used:**

* **useState**: Stores the featured recipe data.
* **useEffect**: Fetches data periodically using a timer.

**5. Contact Component**

Provides a form for users to send messages to the admin or support team.

**Key Features:**

* **Form Submission**: Collects user input for subject, email, and message.
* **Data Handling**: Sends the form data to the server via an HTTP POST request.

**Hooks Used:**

* **useState**: Manages form input fields.

**6. Pagination Implementation**

Divides the list of recipes into manageable pages for improved usability.

**Key Features:**

* **Dynamic Page Calculation**: Determines the number of pages based on the total recipes and recipes per page.
* **Navigation Controls**: Includes "Previous" and "Next" buttons and numeric page buttons.
* **Active Page Highlight**: Highlights the current page number for better user experience.

**Hooks Used:**

* **useState**: Tracks the current page.
* **useEffect**: Resets the current page when filters or sorting options change.

**Key Functionalities and Code Explanation**

**1. Data Fetching**

* **Code Snippet**:
* useEffect(() => {
* fetch(apiUrl)
* .then((response) => response.json())
* .then((data) => setRecipes(data))
* .catch((error) => console.error("Error fetching recipes:", error));
* }, []);
* **Explanation**: The useEffect hook fetches recipes from the JSON server when the component mounts. The data is then stored in the recipes state.

**2. Pagination Logic**

* **Code Snippet**:
* const indexOfLastRecipe = currentPage \* recipesPerPage;
* const indexOfFirstRecipe = indexOfLastRecipe - recipesPerPage;
* const currentRecipes = filteredRecipes.slice(indexOfFirstRecipe, indexOfLastRecipe);
* **Explanation**: Calculates the recipes to display on the current page by slicing the filteredRecipes array based on the current page and recipes per page.

**3. Recipe Sharing**

* **Code Snippet**:
* const shareRecipes = () => {
* if (selectedRecipes.length === 0) {
* alert("Please select recipes to share.");
* return;
* }
* const emailParams = {
* to\_email: userEmail,
* recipes: JSON.stringify(selectedRecipes, null, 2),
* };
* emailjs.send("service\_id", "template\_id", emailParams, "user\_id")
* .then(() => alert("Recipes shared successfully!"))
* .catch((error) => console.error("Error sending email:", error));
* };
* **Explanation**: Validates user input and sends selected recipes to the specified email using the emailjs library.

**4. Sorting and Filtering**

* **Code Snippet**:
* const filteredRecipes = sortRecipes(
* recipes.filter((recipe) => {
* return (
* recipe.title.toLowerCase().includes(searchQuery.toLowerCase()) &&
* (selectedTag === "" || recipe.tags.includes(selectedTag)) &&
* (selectedDifficulty === "" || recipe.difficulty === selectedDifficulty)
* );
* }),
* sortOption
* );
* **Explanation**: Filters recipes based on search query, tags, and difficulty. The filtered recipes are then sorted according to the selected sorting option.

**Styling and User Interface**

* **Pagination**: Styled for accessibility and responsiveness. Active page buttons are highlighted.
* **Forms and Buttons**: Designed to provide a clean and intuitive user experience.
* **Hover Effects**: Enhances interactivity for recipe cards.