**Creating a ToDo app in React with the specified functionalities in the following manner.Here's a step-by-step guide on how to build such an app:**

**1-Set up  React environment:**

**Node.js and npm (Node Package Manager) installed . create a new React app using Create React App or set up your project manually.**

**To create a new React app, run the following commands:**

**npx create-react-app todo-app**

**cd todo-app**

**Project Structure:**

**Inside your project directory, create a components folder to organize your React components. components like TaskList.js, Task.js, and App.js**

**Create Task Component:**

**In Task.js, create a component for each task with the following structure:**

**// Task.js**

**import React, { useState } from 'react';**

**const Task = ({ task, onUpdate, onDelete }) => {**

**const [isChecked, setIsChecked] = useState(false);**

**const [isEditing, setIsEditing] = useState(false);**

**const [updatedTask, setUpdatedTask] = useState(task.text);**

**const handleToggle = () => {**

**setIsChecked(!isChecked);**

**};**

**const handleEdit = () => {**

**setIsEditing(true);**

**};**

**const handleSave = () => {**

**onUpdate(task.id, updatedTask);**

**setIsEditing(false);**

**};**

**return (**

**<div style={{ backgroundColor: task.backgroundColor }}>**

**<input type="checkbox" checked={isChecked} onChange={handleToggle} />**

**{isEditing ? (**

**<input**

**type="text"**

**value={updatedTask}**

**onChange={(e) => setUpdatedTask(e.target.value)}**

**/>**

**) : (**

**<span onClick={handleEdit}>{task.text}</span>**

**)}**

**<button onClick={() => onDelete(task.id)}>Delete</button>**

**{isEditing && <button onClick={handleSave}>Save</button>}**

**</div>**

**);**

**};**

**export default Task;**

**2-Create TaskList Component:**

**In TaskList.js, create a component to manage the list of tasks:**

**// TaskList.js**

**import React, { useState } from 'react';**

**import Task from './Task';**

**const TaskList = () => {**

**const [tasks, setTasks] = useState([]);**

**const [newTask, setNewTask] = useState('');**

**const handleAddTask = () => {**

**if (newTask.trim() === '') return;**

**const randomColor = `#${Math.floor(Math.random() \* 16777215).toString(16)}`;**

**const task = {**

**id: Date.now(),**

**text: newTask,**

**backgroundColor: randomColor,**

**};**

**setTasks([...tasks, task]);**

**setNewTask('');**

**};**

**const handleUpdateTask = (taskId, updatedText) => {**

**const updatedTasks = tasks.map((task) =>**

**task.id === taskId ? { ...task, text: updatedText } : task**

**);**

**setTasks(updatedTasks);**

**};**

**const handleDeleteTask = (taskId) => {**

**const updatedTasks = tasks.filter((task) => task.id !== taskId);**

**setTasks(updatedTasks);**

**};**

**return (**

**<div>**

**<div>**

**<input**

**type="text"**

**placeholder="Add a new task"**

**value={newTask}**

**onChange={(e) => setNewTask(e.target.value)}**

**/>**

**<button onClick={handleAddTask}>Add</button>**

**</div>**

**<div>**

**{tasks.map((task) => (**

**<Task**

**key={task.id}**

**task={task}**

**onUpdate={handleUpdateTask}**

**onDelete={handleDeleteTask}**

**/>**

**))}**

**</div>**

**</div>**

**);**

**};**

**export default TaskList;**

**Integrate TaskList into App:**

**In App.js, import and render the TaskList component:**

**// App.js**

**import React from 'react';**

**import './App.css';**

**import TaskList from './components/TaskList';**

**function App() {**

**return (**

**<div className="App">**

**<h1>ToDo App</h1>**

**<TaskList />**

**</div>**

**);**

**}**

**export default App;**

**Create App Styles:**

**Create a CSS file for your app's styles. You can name it App.css and add it to the src folder:**

**\* App.css \*/**

**body {**

**font-family: Arial, sans-serif;**

**text-align: center;**

**background-color: #f4f4f4;**

**}**

**.todo-app {**

**max-width: 400px;**

**margin: 0 auto;**

**padding: 20px;**

**background-color: white;**

**border-radius: 5px;**

**box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);**

**}**

**.task-input {**

**display: flex;**

**gap: 10px;**

**margin-bottom: 20px;**

**}**

**.task-input input {**

**flex: 1;**

**padding: 5px;**

**border: 1px solid #ccc;**

**border-radius: 3px;**

**}**

**.task-input button {**

**padding: 5px 10px;**

**background-color: #007bff;**

**color: white;**

**border: none;**

**border-radius: 3px;**

**cursor: pointer;**

**}**

**.task-list {**

**display: flex;**

**flex-direction: column;**

**gap: 10px;**

**}**

**.task {**

**display: flex;**

**align-items: center;**

**gap: 10px;**

**padding: 5px;**

**border: 1px solid #ccc;**

**border-radius: 3px;**

**}**

**.task input[type="checkbox"] {**

**cursor: pointer;**

**}**

**.task span {**

**flex: 1;**

**cursor: pointer;**

**}**

**.task button {**

**background-color: #dc3545;**

**color: white;**

**border: none;**

**border-radius: 3px;**

**cursor: pointer;**

**}**

**Update App.js:**

**Finally, update src/App.js to render the TodoApp component:**

**/ App.js**

**import React from 'react';**

**import './App.css';**

**import TodoApp from './TodoApp';**

**function App() {**

**return (**

**<div className="App">**

**<TodoApp />**

**</div>**

**);**

**}**

**export default App;**

**Local Storage:**

**To persist tasks even after refreshing the page, we can use the localStorage API. In the TaskList component, we can load tasks from localStorage when the component mounts and save tasks to localStorage whenever they change. Here's an example of how to do this:**

**// TaskList.js**

**// Load tasks from localStorage when the component mounts**

**useEffect(() => {**

**const storedTasks = JSON.parse(localStorage.getItem('tasks')) || [];**

**setTasks(storedTasks);**

**}, []);**

**// Save tasks to localStorage whenever they change**

**useEffect(() => {**

**localStorage.setItem('tasks', JSON.stringify(tasks));**

**}, [tasks]);**

**Testing:**

**npm start**

**Finally, test your app by running it with npm start and see how it works.we should be able to add, update, delete, and cross tasks off the list, and the tasks should persist after page refresh.**