Assignment 3

Advanced Web Technologies

Submitted by:

Sumaira Nasir

21007105029

Submitted to:

Sir Safiullah

BSIT Batch 7

Semester: Spring 2024

University of Management and Technology Lahore, Sialkot Campus

Introduction

Building a single-page to-do list application using React.js. This application is offering features to manage our daily tasks effectively.

Creating react todoapp:

```
Creating a new React app in D:\ToDoList\todoapp.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


added 1476 packages in 6m

258 packages are looking for funding
  run `npm fund` for details
Git repo not initialized Error: Command failed: git --version
    at genericNodeError (node:internal/errors:984:15)
```

Starting react app:

```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\ToDoList> cd todoapp
PS D:\ToDoList\todoapp> npm start
```

```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Compiled successfully!

You can now view todoapp in the browser.

  Local:            http://localhost:3004
  On Your Network:  http://192.168.56.1:3004

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

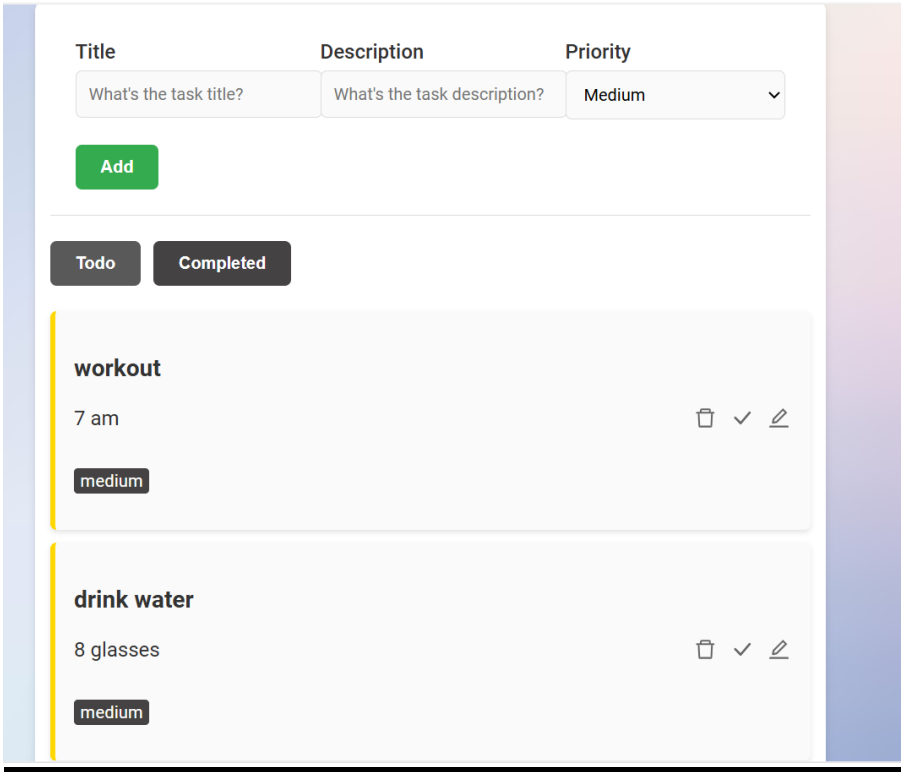
Features

- Utilizing local storage
- Adding new todos
- Editing existing todos
- Marking todos as complete
- Deleting todos
- Viewing completed todos
- Clearing completed todos

Implementation Details

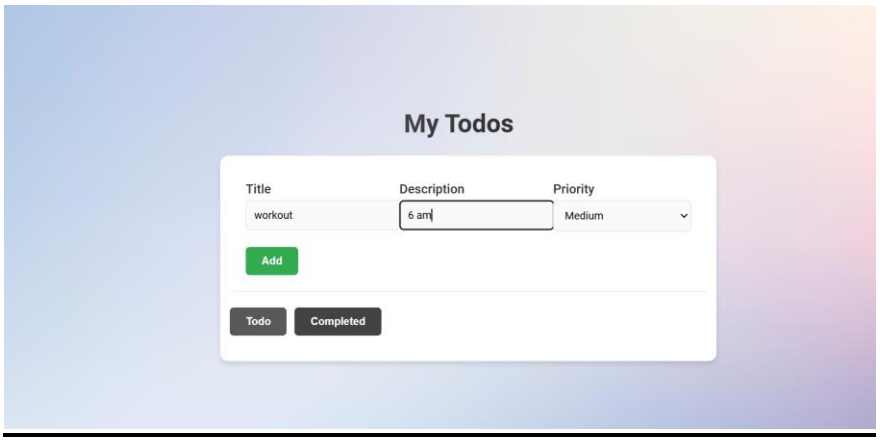
1. Utilizing local storage

Utilizing local storage won't lose any tasks even if I close the app or refresh the page.

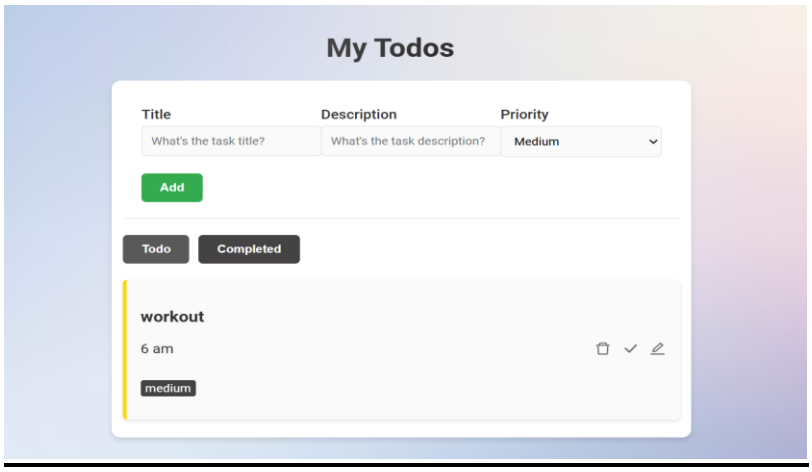


2. Adding New Todos

I have added workout task to my to-do list by entering the task title, description, and priority level in the input fields.

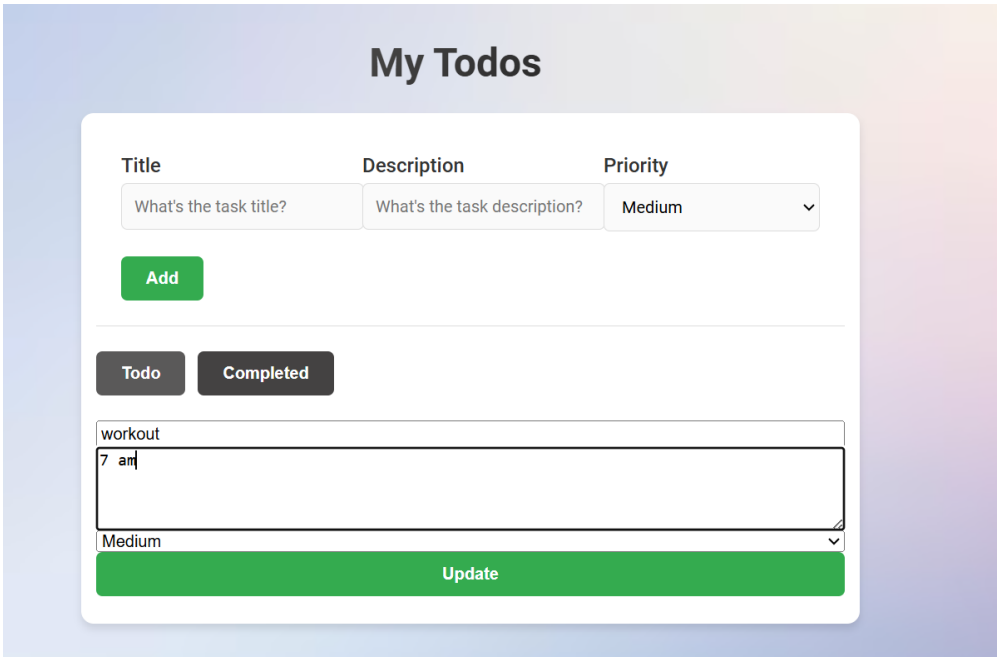


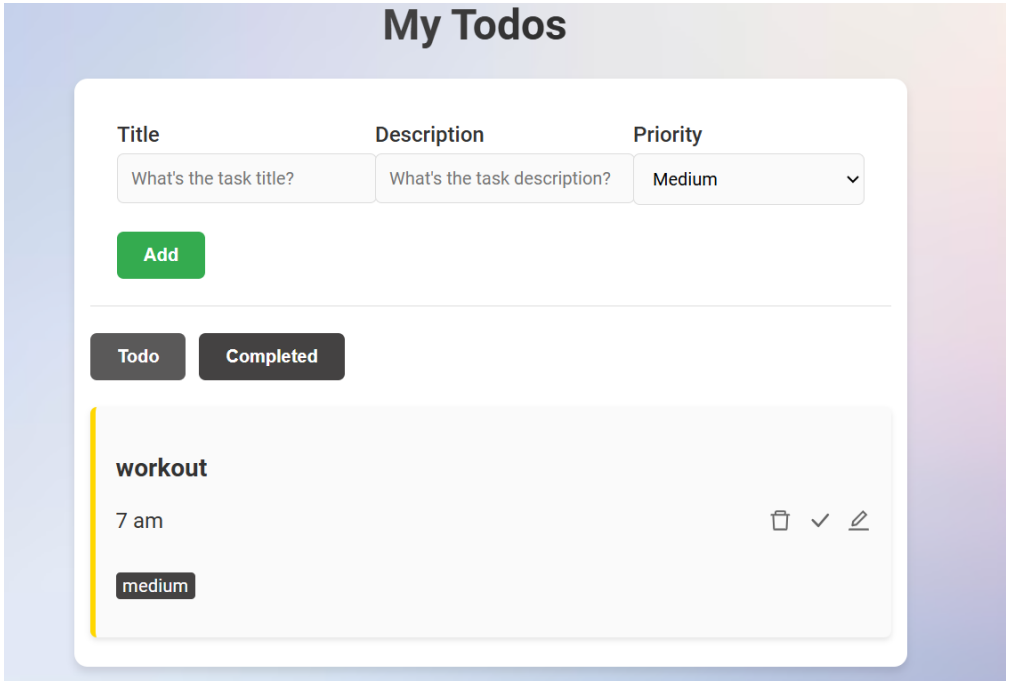
Upon clicking on Add button, the new task (workout) is added to the list.



3. Editing Existing Todos

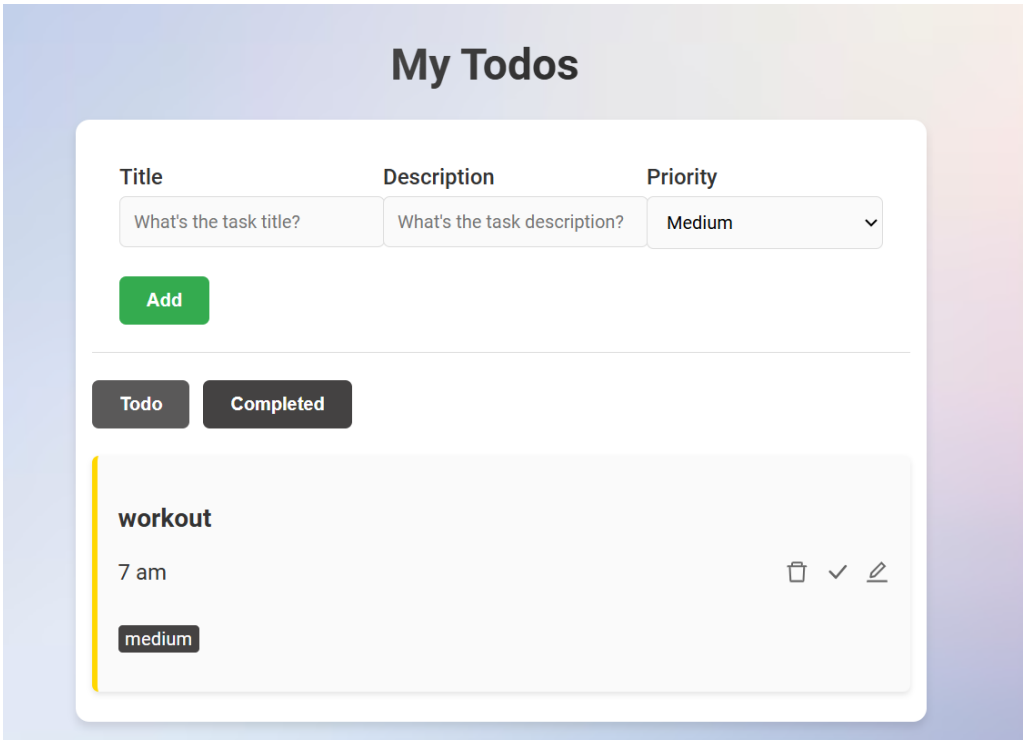
By clicking on the edit icon next to the task, I have edited existing task (workout) description from 6 am to 7 am.



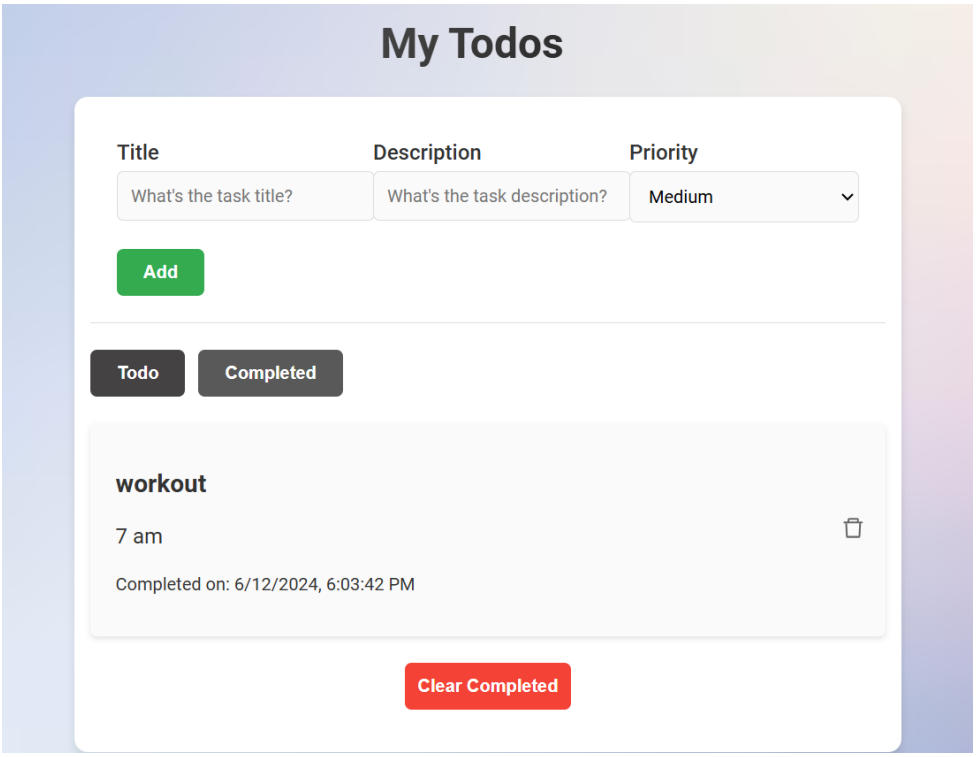


4. Marking Todos as Complete

I have marked workout task as complete by clicking on the checkbox next to the task.

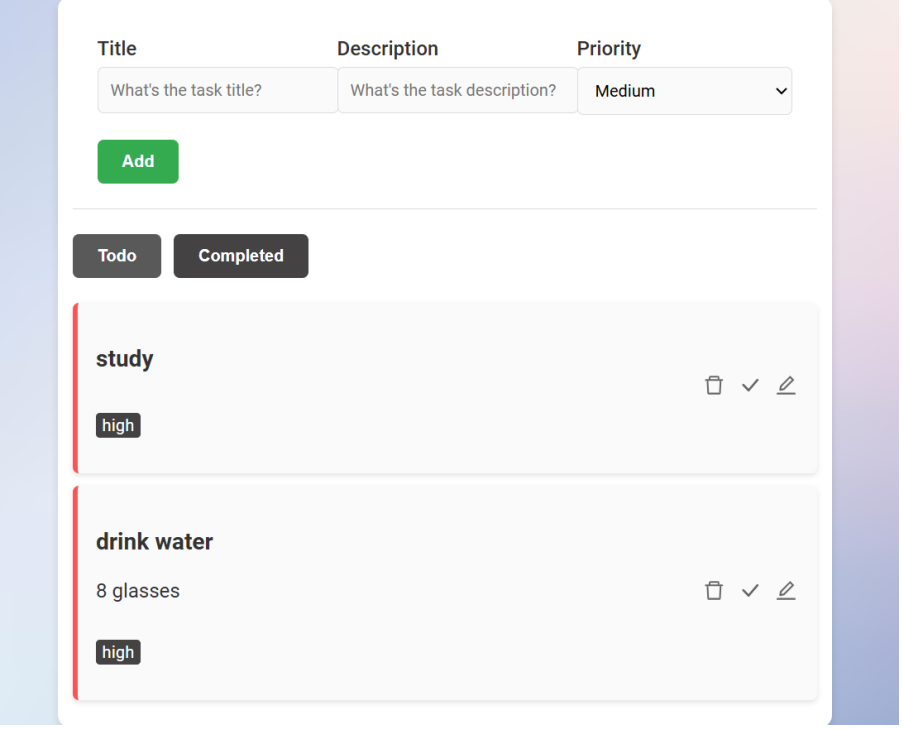


This action has moved task to the completed tasks section.

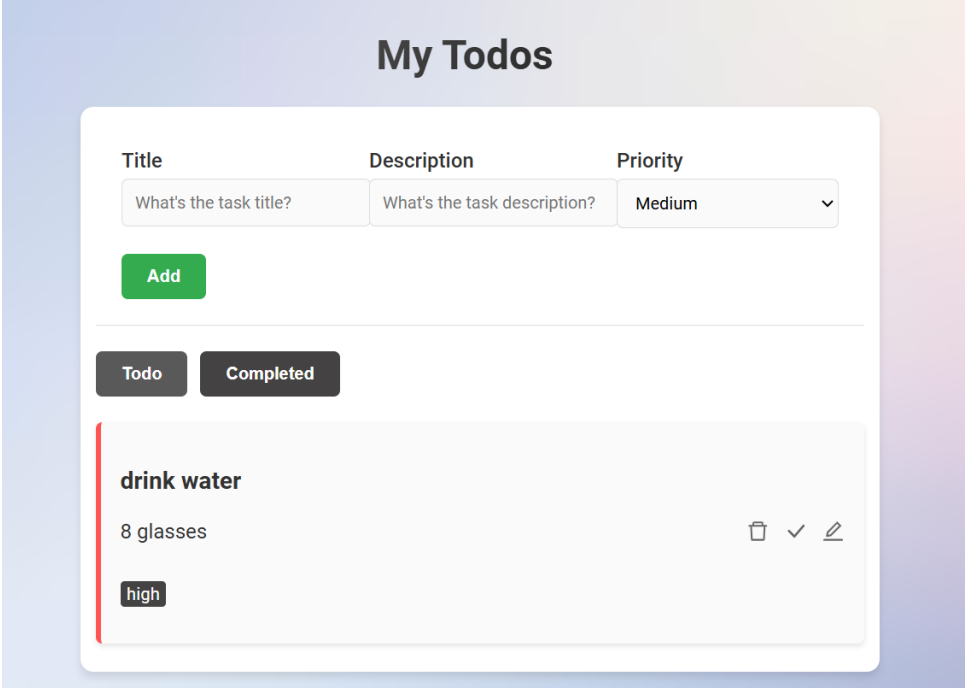


5. Deleting Todos

To remove a task from the to-do list, I have clicked on the delete icon associated with the task.

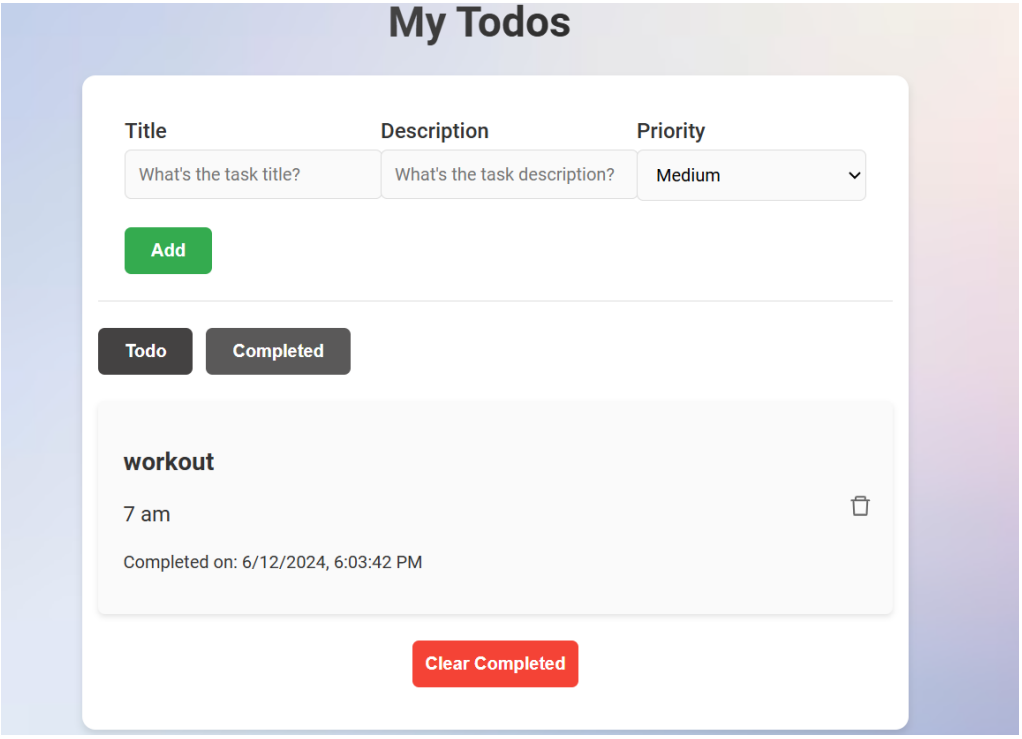


This action permanently deletes the task from the list.



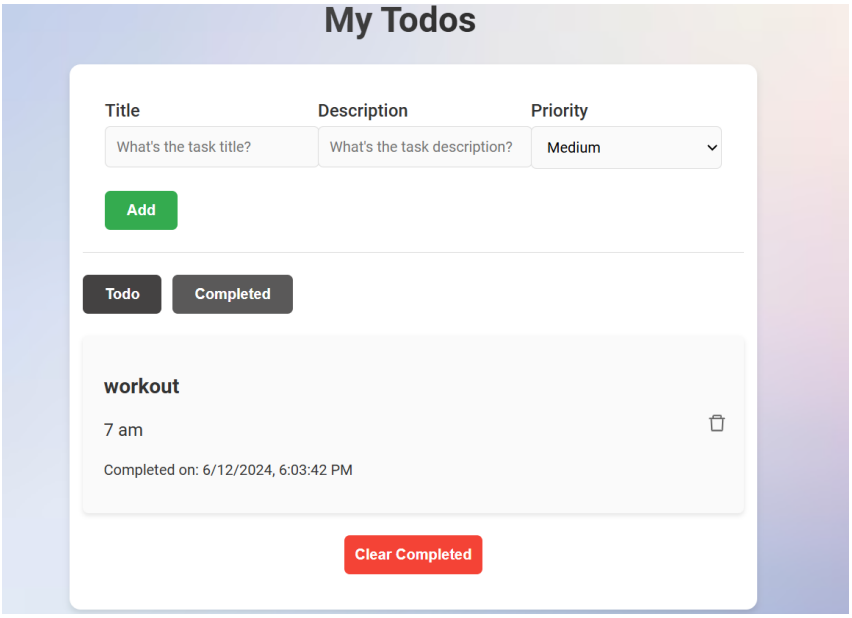
6. Viewing Completed Todos

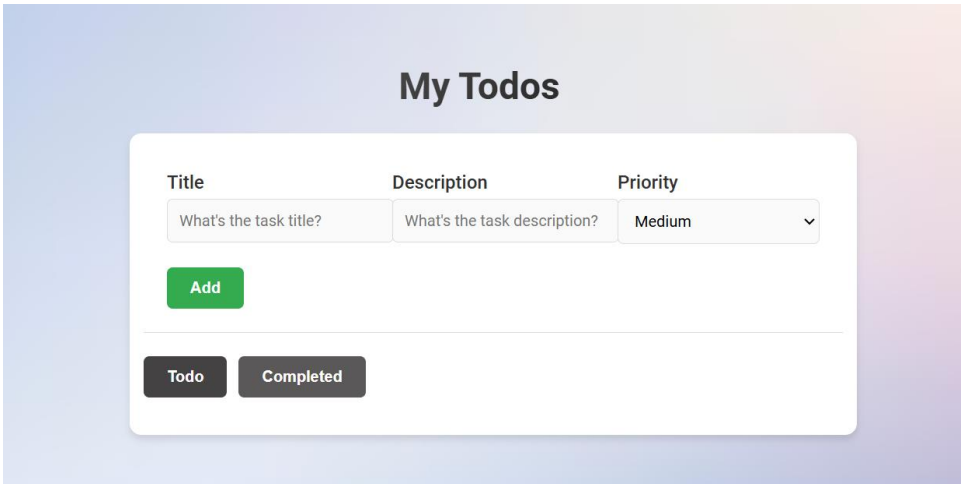
Completed tasks are displayed separately from active tasks.



7. Clearing Completed Todos

A "Clear Completed" button is provided to allow users to remove all completed tasks from the list at once.





Code:

- **App.js**

```
import React, { useState, useEffect } from 'react';
import './App.css';
import { AiOutlineDelete, AiOutlineEdit } from 'react-icons/ai';
import { BsCheckLg } from 'react-icons/bs';

function App() {
  const [isCompleteScreen, setIsCompleteScreen] = useState(false);
  const [allTodos, setTodos] = useState([]);
  const [newTitle, setNewTitle] = useState('');
  const [newDescription, setNewDescription] = useState('');
  const [newPriority, setNewPriority] = useState('medium');
  const [completedTodos, setCompletedTodos] = useState([]);
  const [currentEdit, setCurrentEdit] = useState('');
  const [currentEditedItem, setCurrentEditedItem] = useState('');

  useEffect(() => {
    const savedTodo = JSON.parse(localStorage.getItem('todolist'));
    const savedCompletedTodo =
JSON.parse(localStorage.getItem('completedTodos'));

    if (savedTodo) {
      setTodos(savedTodo);
    }

    if (savedCompletedTodo) {
      setCompletedTodos(savedCompletedTodo);
```

```
    }
  }, []);

  useEffect(() => {
    localStorage.setItem('todolist', JSON.stringify(allTodos));
    localStorage.setItem('completedTodos', JSON.stringify(completedTodos));
  }, [allTodos, completedTodos]);

  const handleAddTodo = () => {
    let newTodoItem = {
      title: newTitle,
      description: newDescription,
      priority: newPriority,
      completed: false,
    };

    let updatedTodoArr = [...allTodos];
    updatedTodoArr.push(newTodoItem);
    setTodos(updatedTodoArr);
    setNewTitle('');
    setNewDescription('');
    setNewPriority('medium');

    setTimeout(() => {
      const items = document.querySelectorAll('.todo-list-item');
      if (items.length) {
        items[items.length - 1].classList.add('add-animation');
      }
    }, 100);
  };

  const handleDeleteTodo = index => {
    let reducedTodo = [...allTodos];
    const item = reducedTodo[index];
    document.querySelectorAll('.todo-list-item')[index].classList.add('delete-
animation');

    setTimeout(() => {
      reducedTodo.splice(index, 1);
      setTodos(reducedTodo);
    }, 500);
  };

  const handleComplete = index => {
    let now = new Date();
```

```
  let completedOn = now.toLocaleString();

  let filteredItem = {
    ...allTodos[index],
    completedOn: completedOn,
    completed: true,
  };

  let updatedCompletedArr = [...completedTodos];
  updatedCompletedArr.push(filteredItem);
  setCompletedTodos(updatedCompletedArr);

  const item = document.querySelectorAll('.todo-list-item')[index];
  item.classList.add('complete-animation');

  setTimeout(() => {
    handleDeleteTodo(index);
  }, 500);
};

const handleDeleteCompletedTodo = index => {
  let reducedCompletedTodo = [...completedTodos];
  reducedCompletedTodo.splice(index, 1);
  setCompletedTodos(reducedCompletedTodo);
};

const handleClearCompleted = () => {
  setCompletedTodos([]);
};

const handleEdit = (index, item) => {
  setCurrentEdit(index);
  setCurrentEditedItem(item);
};

const handleUpdateTitle = newTitle => {
  setCurrentEditedItem({ ...currentEditedItem, title: newTitle });
};

const handleUpdateDescription = newDescription => {
  setCurrentEditedItem({ ...currentEditedItem, description: newDescription });
};

const handleUpdatePriority = newPriority => {
  setCurrentEditedItem({ ...currentEditedItem, priority: newPriority });
```

```
  };

  const handleUpdateToDo = () => {
    const updatedTodos = allTodos.map((item, index) => {
      if (index === currentEdit) {
        return currentEditedItem;
      }
      return item;
    });
    setTodos(updatedTodos);
    setCurrentEdit(null);
    setCurrentEditedItem(null);
  };

  return (
    <div className="App">
<h1 className="todo-title">My Todos</h1>

      <div className="todo-wrapper">
        <div className="todo-input">
          <div className="todo-input-item">
            <label>Title</label>
            <input
              type="text"
              value={newTitle}
              onChange={e => setNewTitle(e.target.value)}
              placeholder="What's the task title?"
            />
          </div>
          <div className="todo-input-item">
            <label>Description</label>
            <input
              type="text"
              value={newDescription}
              onChange={e => setNewDescription(e.target.value)}
              placeholder="What's the task description?"
            />
          </div>
          <div className="todo-input-item">
            <label>Priority</label>
            <select value={newPriority} onChange={e =>
setNewPriority(e.target.value)}>
              <option value="high">High</option>
              <option value="medium">Medium</option>
              <option value="low">Low</option>
```

```
            </select>
          </div>
          <div className="todo-input-item">
            <button type="button" onClick={handleAddTodo} className="primaryBtn">
              Add
            </button>
          </div>
        </div>

        <div className="btn-area">
          <button
            className={`secondaryBtn ${isCompleteScreen === false && 'active'}`}
            onClick={() => setIsCompleteScreen(false)}
          >
            Todo
          </button>
          <button
            className={`secondaryBtn ${isCompleteScreen === true && 'active'}`}
            onClick={() => setIsCompleteScreen(true)}
          >
            Completed
          </button>
        </div>

        <div className="todo-list">
          {isCompleteScreen === false &&
            allTodos.map((item, index) => {
              if (currentEdit === index) {
                return (
                  <div className="edit__wrapper" key={index}>
                    <input
                      placeholder="Updated Title"
                      onChange={e => handleUpdateTitle(e.target.value)}
                      value={currentEditedItem.title}
                    />
                    <textarea
                      placeholder="Updated Description"
                      rows={4}
                      onChange={e => handleUpdateDescription(e.target.value)}
                      value={currentEditedItem.description}
                    />
                    <select value={currentEditedItem.priority} onChange={e =>
handleUpdatePriority(e.target.value)}>
                      <option value="high">High</option>
                      <option value="medium">Medium</option>
```

```
                    <option value="low">Low</option>
                  </select>
                  <button type="button" onClick={handleUpdateToDo}
classNameName="primaryBtn">
                    Update
                  </button>
                </div>
              );
            } else {
              return (
                <div classNameName={`todo-list-item ${item.priority}`} key={index}>
                  <div>
                    <h3>{item.title}</h3>
                    <p>{item.description}</p>
                    <p classNameName="priority-label">{item.priority}</p>
                  </div>

                  <div>
                    <AiOutlineDelete
                      classNameName="icon"
                      onClick={() => handleDeleteTodo(index)}
                      title="Delete?"
                    />
                    <BsCheckLg
                      classNameName="check-icon"
                      onClick={() => handleComplete(index)}
                      title="Complete?"
                    />
                    <AiOutlineEdit
                      classNameName="check-icon"
                      onClick={() => handleEdit(index, item)}
                      title="Edit?"
                    />
                  </div>
                </div>
              );
            }
          })}

        {isCompleteScreen === true &&
          completedTodos.map((item, index) => {
            return (
              <div classNameName="todo-list-item" key={index}>
                <div>
                  <h3>{item.title}</h3>
```

```
                  <p>{item.description}</p>
                  <p><small>Completed on: {item.completedOn}</small></p>
                </div>

                <div>
                  <AiOutlineDelete
                    className="icon"
                    onClick={() => handleDeleteCompletedTodo(index)}
                    title="Delete?"
                  />
                </div>
              </div>
            );
          })}
        {isCompleteScreen && completedTodos.length > 0 && (
          <button className="clearBtn" onClick={handleClearCompleted}>
            Clear Completed
          </button>
        )}
      </div>
    </div>
  </div>
  );
}

export default App;
```

- **App.css**

```
/* Import Google Font */
@import
url('https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&display=swa
p');

/* Global Styles */
body {
  background-color: #e0e0e0;
  background-image: url('https://img.freepik.com/free-vector/winter-blue-pink-
gradient-background-vector_53876-117276.jpg');
  background-size: cover;
  font-family: 'Roboto', sans-serif;
  color: #333;
```

```
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
}


h1 {
  text-align: center;
  color: #444;
}

/* Todo Wrapper */
.todo-wrapper {
  background: #fff;
  padding: 2%;
  width: 100%;
  max-width: 600px;
  margin-top: 3%;
  border-radius: 10px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  overflow-y: auto;
}
.todo-title {
  background: linear-gradient(135deg, #6a6969, #080808);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
}

.todo-input {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
  padding: 20px;
  border-bottom: 1px solid #ddd;
  margin-bottom: 20px;
}

.todo-input-item {
  flex: 1 1 calc(30% - 10px);
}

.priority-container {
```

```
  flex: 1 1 calc(5% - 5px);
  margin-top: 10px;
}


.todo-input-item input,
.todo-input-item textarea,
.todo-input-item select {
  width: 100%;
  padding: 10px;
  border: 1px solid #ddd;
  border-radius: 5px;
  background: #fafafa;
  font-family: 'Roboto', sans-serif;
}

.todo-input-item label {
  display: block;
  margin-bottom: 5px;
  font-weight: 500;
}

.primaryBtn,
.secondaryBtn {
  padding: 10px 20px;
  border-radius: 5px;
  border: none;
  color: white;
  font-weight: bold;
  cursor: pointer;
  transition: background 0.3s ease;
}

.primaryBtn {
  background: #34ab4f;
}

.primaryBtn:hover {
  background: #2a8e3b;
}

.secondaryBtn {
  background: #444242;
  margin-right: 10px;
}
```

```css
.secondaryBtn:hover {
  background: #363434;
}

.active {
  background: #5a5959;
}

/* Todo List */
.todo-list {
  margin-top: 20px;
}

.todo-list-item {
  background: #fafafa;
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding: 15px;
  margin-bottom: 10px;
  border-radius: 5px;
  border-left: 4px solid transparent;
  transition: background 0.3s ease, border-color 0.3s ease;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.todo-list-item:hover {
  background: #f1f1f1;
}

.high {
  border-left-color: #ff5252;
}

.medium {
  border-left-color: #ffd700;
}

.low {
  border-left-color: #4caf50;
}

.priority-label {
  font-size: 0.85em;
```

```
  background: #444242;
  padding: 2px 5px;
  border-radius: 3px;
  color: white;
  display: inline-block;
}

.icon,
.check-icon {
  margin-left: 10px;
  cursor: pointer;
  font-size: 1.2em;
  color: #666;
  transition: color 0.3s ease;
}

.icon:hover,
.check-icon:hover {
  color: #333;
}

.clearBtn {
  background: #f44336;
  padding: 10px;
  border-radius: 5px;
  border: none;
  color: white;
  font-weight: bold;
  cursor: pointer;
  display: block;
  margin: 20px auto;
  transition: background 0.3s ease;
}

.clearBtn:hover {
  background: #d7352c;
}

/* Animation Classes */
@keyframes slideIn {
  from {
    opacity: 0;
    transform: translateY(-20px);
  }
  to {
```

```
    opacity: 1;
    transform: translateY(0);
  }
}

@keyframes slideOut {
  from {
    opacity: 1;
    transform: translateY(0);
  }
  to {
    opacity: 0;
    transform: translateY(20px);
  }
}

@keyframes fadeOut {
  from {
    opacity: 1;
  }
  to {
    opacity: 0;
  }
}

@keyframes complete {
  0% {
    background-color: #fafafa;
  }
  100% {
    background-color: #e0ffe0;
  }
}

.add-animation {
  animation: slideIn 0.5s ease forwards;
}

.delete-animation {
  animation: slideOut 0.5s ease forwards;
}

.complete-animation {
  animation: complete 0.5s ease forwards;
}
```

```css
.edit__wrapper {
  display: flex;
  flex-direction: column;
  margin-bottom: 10px;
  animation: slideIn 0.3s ease forwards;
}
```