COLLEGE CODE: 8201

COLLEGE NAME: A.R.J COLLEGE OF ENGINEERING AND TECNOLOGY

DEPARTMENT NAME : BE- COMPUTER SCIENCE & ENGINEERING

STUDENT NM-ID : **aut820123104006**

ROLL NO : 820123104027

DATE :26.10.2025

COMPLETED THE PROJECT NAME AS

PHASE5:

PROJECTDEMOSTRATION&DOCUMENTATION

PROJECT NAME :

IBM-NJ-TO-DO-APP WITH REACT HOOKS

SUBMITTED BY :

NAME : ATCHAYA .M.M

MOBILE NO : 9171071172

PHASE 5: TO DO APPS WITH REACT HOOKS

PROJECT DEMOSTRATION&DOCUMENTATION

- FINAL DEMO WALKTHROUGH
- PROJECT REPORT
- SCREENSHOTS/ API DOCUMENTATION
- CHALLENGES &SOLUTION
- GITHUB README&SETUP GUIDE
- FINAL SUBMISSION(REPO+DEPLOYED LINK)

1.FINAL DEMO WALKTHROUGH:

1. OBJECTIVES

THE MAIN OBJECTIVES OF THIS PROJECT ARE:

1. To develop a simple and efficient To-Do Application using React Hooks.

2. To understand and implement React's functional components with state and

side effects.

3. To demonstrate data persistence using the browser's localStorage.

4. To create a responsive and user-friendly interface for managing tasks.

5. To enhance skills in modern frontend development using React.

2. PROJECT OVERVIEW

- This project is a To-Do List Application built with React.js and React Hooks.
- It allows users to add, complete, and delete tasks efficiently.
- The application also stores data in localStorage, ensuring that tasks are not lost even
- after refreshing or closing the browser.

- The project serves as a practical example of state management, side-effect handling, and component-based design in React.

3.TECHNOLOGIES USED

- **Frontend Framework**: React.js
- **Language:** JavaScript (ES6)
- **Styling**: Tailwind CSS (or CSS)
- **State Management**: React Hooks (useState, useEffect)
- **Storage:** Browser localStorage

4. FEATURES

- ✓ **Add new tasks**
- ✓ **Mark tasks as completed**
- ✓ **Delete tasks**
- ✓ **Store tasks in** localStorage
- ✓ **Auto-load saved tasks on refresh**
- ✓ **Clean and responsive user interface**

5. REACT HOOKS USED

HOOK	DESCRIPTION
useState	Manage components state for tasks and user input
useEffect	Perform side effects such as saving and loading tasks from local storage

7. WORKFLOW EXPLANATION

1. **User Input**: The user enters a task in the input field and clicks "Add."

2**. State Update**: The addTask() function updates the state variable tasks.

3. **Rendering:** React dynamically re-renders the updated list using .map().

4. **Toggle Task**: Clicking on a task toggles its completion status.

5. **Delete Task**: The user can remove tasks using the delete (✕) button.

6. **Data Persistence**: useEffect() ensures that all changes are saved to local Storage automatically

8. DEMONSTRATION

(Add screenshots or screen recordings of your app here — for example:)

- Screenshot 1: Adding a task
- Screenshot 2: Marking a task as complete
- Screenshot 3: Tasks saved after reload

PROJECT REPORT:

1. FEATURES AND IMPLEMENTATION

The application includes the following key features:

- **Add Tasks**: Users can add new tasks to the list.
- **Mark Completed:** Tasks can be marked as completed.
- **Edit and Delete Tasks**: Tasks can be updated or removed.
- **Data Persistence**: Tasks remain stored even after refreshing the browser.
- **Responsive Design** : Works on desktop and mobile devices.

IMPLEMENTATION OVERVIEW:

The app uses React Hooks for state management.

- useState handles the current tasks and input fields.

- useEffect saves tasks in localStorage to maintain persistence.
- Components like TodoForm, TodoList, and TodoItem are modular and reusable for maintainability.

2. CHALLENGES AND SOLUTIONS

Challenge 1: Persisting tasks across sessions.

Solution: Used localStorage to store and retrieve tasks on page reload.

Challenge 2: Managing performance with many tasks.

Solution: Optimized state updates and re-rendering using unique keys.

Challenge 3: Ensuring accessibility and keyboard navigation.

Solution: Added ARIA attributes, proper focus management, and logical tab ordering.

3. CONCLUSION AND FUTURE SCOPE

CONCLUSION:

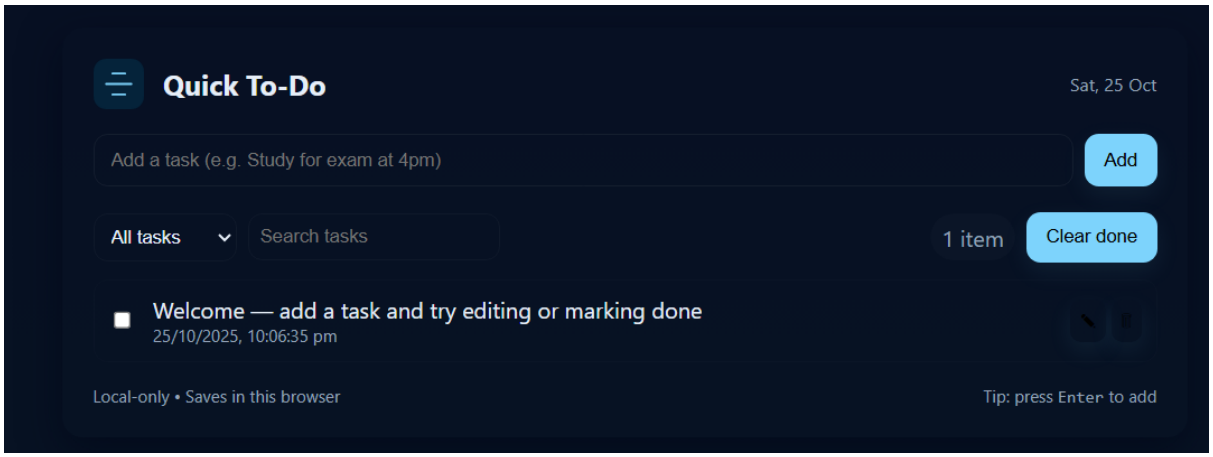
- The To-Do App demonstrates the effective use of React Hooks to manage tasks
- efficiently. It is responsive, user-friendly, and provides a simple solution for task

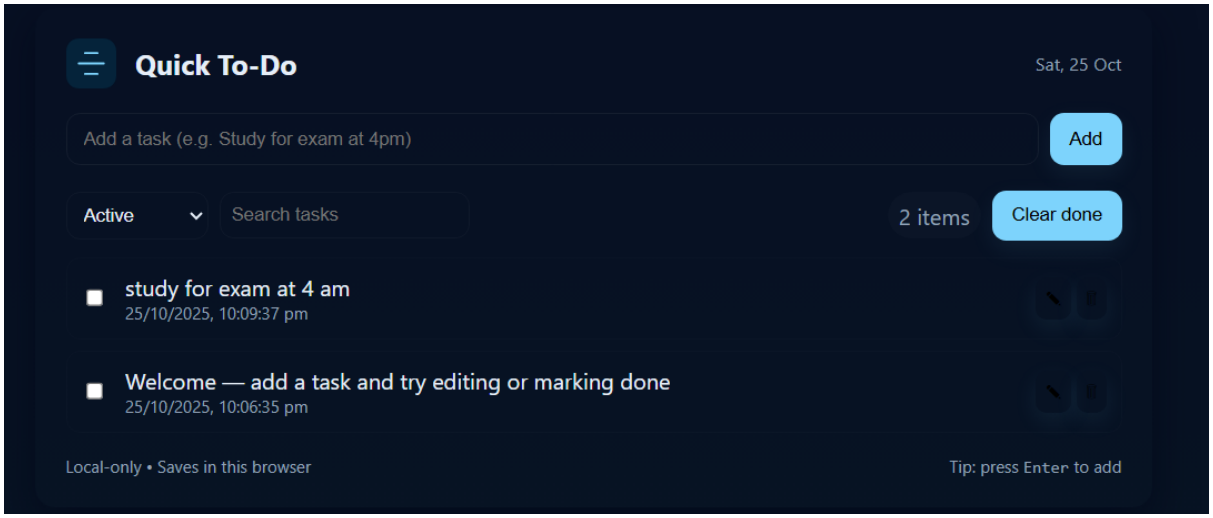
- management while showcasing best practices in React development.

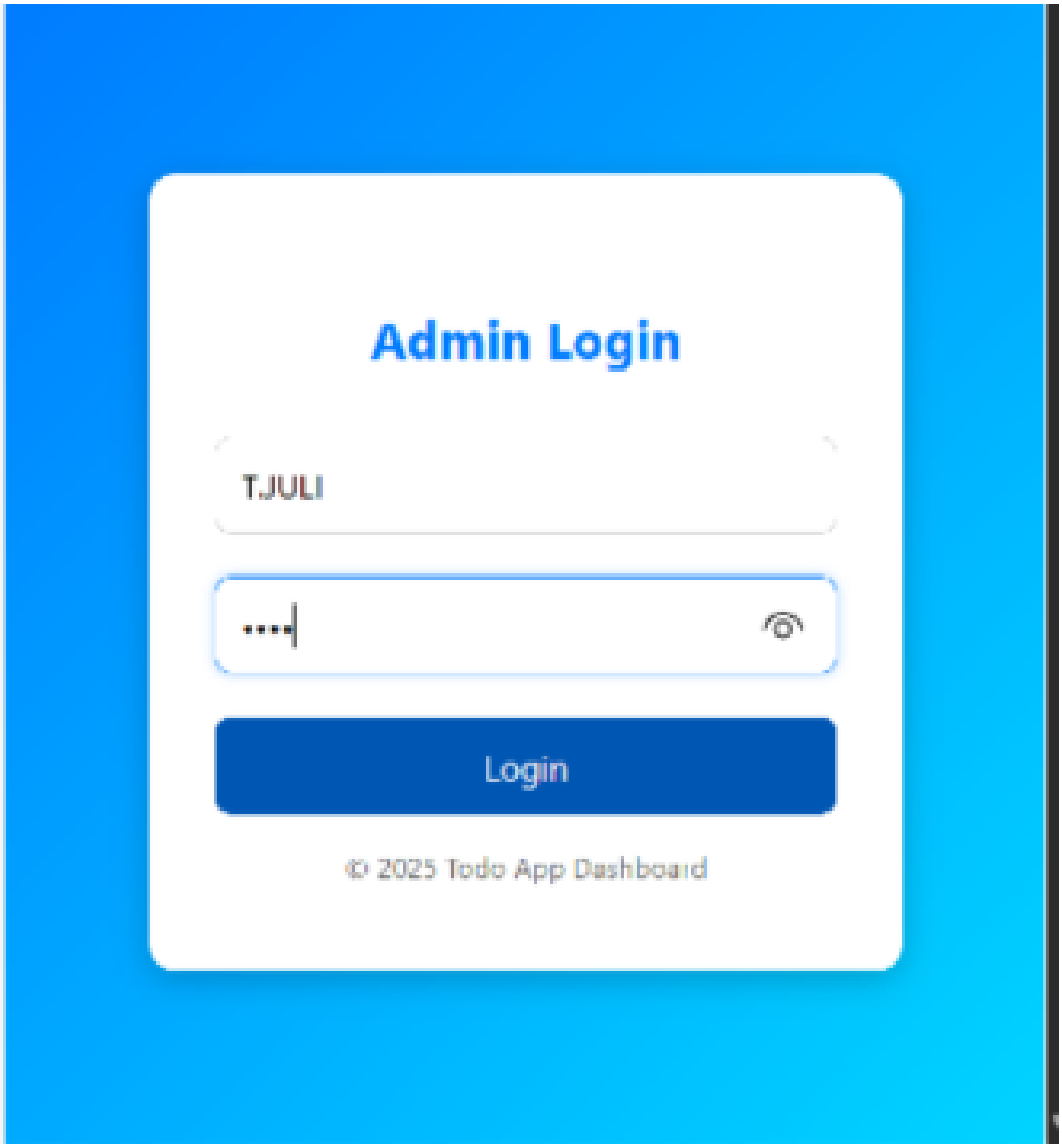
FUTURE SCOPE:

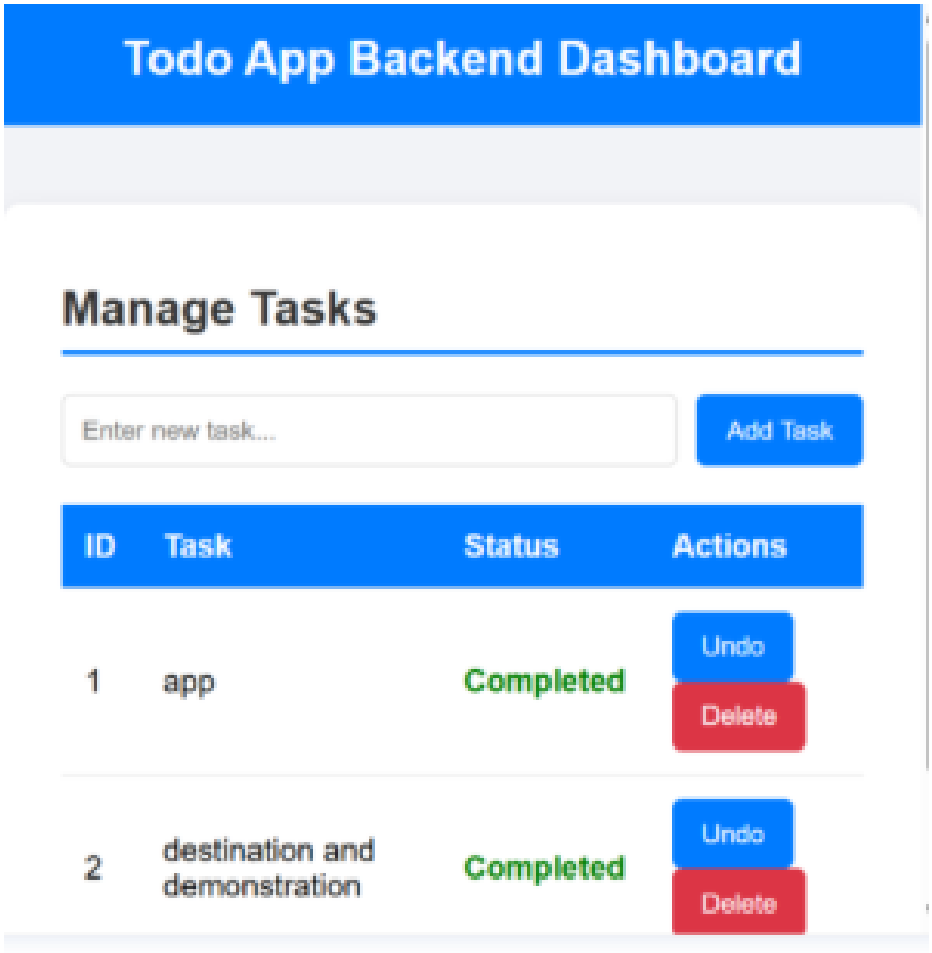
- Integrate with a backend API for multi-device access.
- Add task categories, priorities, deadlines, or reminders.
- Include dark/light themes for better UI.
- Optimize for large task lists and improve accessibility features.

SCREENSHOTS/API DOCUMENTATION:




Quick To-Do
Sat, 25 Oct
Add a task (e.g. Study for exam at 4pm)
Add
Active
Search tasks
2 items
Clear done
study for exam at 4 am
25/10/2025, 10:09:37 pm
Welcome — add a task and try editing or marking done
25/10/2025, 10:06:35 pm
Local-only • Saves in this browser
Tip: press Enter to add


Admin Login
TJULI
Login
© 2025 Todo App Dashboard



PROGRAM:

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,initial-scale=1" />
<title>Simple To-Do (HTML only)</title>
<style>
```

```
:root{
--bg:#0f1724; --card:#0b1220; --accent:#7dd3fc; --muted:#94a3b8;
--glass: rgba(255,255,255,0.03);
font-family: Inter, system-ui, -apple-system, "Segoe UI", Roboto,
"Helvetica Neue", Arial;
}
html,body{height:100%;margin:0;background:linear-
gradient(180deg,#071023 0%, #081424
60%);color:#e6eef8}
.wrap{max-width:760px;margin:40px
auto;padding:22px;background:linear-gradient(180deg,var(--
glass),transparent);border-radius:14px;box-shadow:0 10px 30px
rgba(2,6,23,0.6);}
header{display:flex;align-items:center;gap:14px}

header h1{font-size:20px;margin:0}
.meta{margin-left:auto;color:var(--muted);font-size:13px}
form{display:flex;gap:8px;margin:18px 0}
input[type="text"]{
flex:1;padding:10px 12px;border-radius:10px;border:1px solid
rgba(255,255,255,0.06);
background:transparent;color:inherit;outline:none;
}
button{
```

```
background:var(--accent);border:none;padding:10px 14px;border-radius:10px;font- weight:600;cursor:pointer;

box-shadow:0 6px 18px rgba(125,211,252,0.08);

}

.controls{display:flex;gap:8px;align-items:center;margin-bottom:12px}

.controls select, .controls input[type="search"]{

padding:8px;border-radius:10px;border:1px solid

rgba(255,255,255,0.04);background:transparent;color:inherit;

}

ul#todos{list-style:none;padding:0;margin:0;display:grid;gap:8px}

li.todo{display:flex;align-items:center;gap:12px;padding:10px;border-radius:10px;background:linear- gradient(180deg, rgba(255,255,255,0.02), transparent);border:1px solid rgba(255,255,255,0.02)}

li.todo .title{flex:1}


.small{font-size:12px;color:var(--muted)}

.actions button{background:transparent;border:0;color:var(--muted);cursor:pointer;padding:6px;border-radius:8px}

.actions button:hover{color:var(--accent)}

.badge{padding:6px 8px;border-radius:999px;background:rgba(255,255,255,0.02);font-size:12px;color:var(--muted)}
```

.done{opacity:0.6;text-decoration:line-through}

footer{display:flex;justify-content:space-between;align-items:center;margin-top:16px;color:var(-- muted);font-size:13px}

@media (max-width:520px){.wrap{margin:18px;padding:14px} header h1{font-size:16px}}

</style>

</head>

<body>

<main class="wrap" role="main">

<header>

<svg width="36" height="36" viewBox="0 0 24 24" fill="none" aria-hidden><rect width="24" height="24" rx="6" fill="#05233a"/><path d="M6 12h12M9 7h6M9 17h6" stroke="#7dd3fc" stroke- width="1.6" stroke-linecap="round"/></svg>

<h1>Quick To-Do</h1>

</header>

<form id="addForm" aria-label="Add todo">

<input id="todoInput" type="text" placeholder="Add a task (e.g. Study for exam at 4pm)" required

aria-required="true" />

<button type="submit">Add</button>

</form>

```
<div class="controls" role="region" aria-label="controls">
<select id="filter" title="Filter tasks">
<option value="all">All tasks</option>
<option value="active">Active</option>
<option value="done">Completed</option>
</select>
<input id="search" type="search" placeholder="Search tasks" />
<div style="margin-left:auto;display:flex;gap:8px;align-items:center">
<span class="badge" id="countBadge">0 items</span>
<button id="clearDone" type="button" title="Remove completed tasks">Clear done</button>
</div>
</div>

<ul id="todos" aria-live="polite"></ul>

<footer>
<div class="small">Local-only • Saves in this browser</div>
<div class="small">Tip: press <kbd>Enter</kbd> to add</div>
</footer>
</main>
```

```
<script>
/* Simple To-Do with localStorage — single-file HTML program */
(function(){
const key = 'quick_todos_v1';
const $ = sel => document.querySelector(sel);
const $$ = sel => Array.from(document.querySelectorAll(sel));
const todosEl = $('#todos');
const input = $('#todoInput');
const form = $('#addForm');
const filter = $('#filter');
const search = $('#search');
const countBadge = $('#countBadge');
const clearDoneBtn = $('#clearDone');

const dateMeta = $('#dateMeta');

// show today's date
const now = new Date();
dateMeta.textContent = now.toLocaleDateString(undefined, {weekday:'short', month:'short', day:'numeric'});

// utilities
```

```
function uid(){ return Math.random().toString(36).slice(2,9); }
function load(){ try { return
JSON.parse(localStorage.getItem(key)||'[]') } catch(e){return []}}
function save(items){ localStorage.setItem(key,
JSON.stringify(items)); render(); }
function getItems(){ return load(); }

// render
function render(){
const items = getItems();
const q = search.value.trim().toLowerCase();
const f = filter.value;
let shown = items.filter(it => {
if(f==='active' && it.done) return false;
if(f==='done' && !it.done) return false;

if(q && !it.text.toLowerCase().includes(q)) return false;
return true;
});

todosEl.innerHTML = shown.map(it => ` <li class="todo" data-
id="${it.id}">
<input type="checkbox" ${it.done ? 'checked' : ''} aria-
label="Mark ${escapeHtml(it.text)} as
```

```
done">
<div class="title ${it.done ? 'done' : ''}">
<div>${escapeHtml(it.text)}</div>
<div class="small">${it.createdAt}</div>
</div>
<div class="actions">
<button class="edit" title="Edit">✎</button>
<button class="delete" title="Delete">🗑</button>
</div>
</li>
`).join('');

countBadge.textContent = items.length + (items.length === 1 ? ' item' : ' items');
}

// escape for safe insertion into HTML

function escapeHtml(str){
return str.replace(/[&<>"']/g, (m)=>({'&':'&amp;','<':'&lt;','>':'&gt;','"':'&quot;',"'":'&#39;'}[m]));
}
```

```
// add
form.addEventListener('submit', e=>{
e.preventDefault();
const text = input.value.trim();
if(!text) return;
const items = getItems();
items.unshift({ id: uid(), text, done:false, createdAt: new
Date().toLocaleString() });
save(items);
input.value = '';
input.focus();
});

// delegate clicks
todosEl.addEventListener('click', e=>{

const li = e.target.closest('li.todo');
if(!li) return;

const id = li.dataset.id;
const items = getItems();
const idx = items.findIndex(i=>i.id===id);
if(idx === -1) return;
```

```
if(e.target.matches('input[type="checkbox"]')){
items[idx].done = e.target.checked;
save(items);
} else if(e.target.closest('.delete')){
if(confirm('Delete this task?')){ items.splice(idx,1); save(items); }
} else if(e.target.closest('.edit')){
const newText = prompt('Edit task', items[idx].text);
if(newText !== null){ items[idx].text = newText.trim() ||
items[idx].text; save(items); }
}
});

// keyboard: Enter to add when focused in input
input.addEventListener('keydown', e=>{

if(e.key === 'Enter' && !e.shiftKey){ e.preventDefault();
form.dispatchEvent(new Event('submit')); }
});

filter.addEventListener('change', render);
search.addEventListener('input', render);
```

```
clearDoneBtn.addEventListener('click', ()=>{
const items = getItems().filter(i=>!i.done);
save(items);
});

// first-run demo item
if(getItems().length === 0){
const demo = [{ id: uid(), text: 'Welcome — add a task and try editing or marking done', done:false, createdAt: new Date().toLocaleString() }];
localStorage.setItem(key, JSON.stringify(demo));
}

// initial render
render();

// expose for debugging (optional)
window._todoApp = {

load, save, getItems, render
};
})();
</script>
```

</body>

</html>

API DOCUMENTATION:

GET /todos

Description: Fetch all tasks

Example Response:

[{"id":1,"text":"Buy groceries","completed":false},
{"id":2,"text":"Learn React Hooks","completed":true}]

POST /todos

Description: Add a new task

Example Request:

{"text": "Finish assignment", "completed": false}

Example Response:

{"id":3,"text": "Finish assignment", "completed": false}

PATCH /todos/:id

Description: Update an existing task

Example Request:

{"completed": true}

Example Response:

{"id":3,"text":"Finish assignment","completed": true}

DELETE /todos/:id

Description: Delete a task

Example Response:

204 No Content

CHALLENGES &SOLUTION:

Challenges	Problem	Solution
Persisting Tasks Across Sessions	Tasks disappear when the page is refreshed or browser is closed.	Used local Storage to save and retrieve tasks, ensuring data persists across sessions.
Managing Performance with Many Tasks	Large task lists can slow down rendering and affect performance.	Optimized component re-rendering with unique keys and efficient state updates.
Editing Tasks While Maintaining Data Consistency	Updating tasks can cause inconsistent data or UI delays.	Implemented optimistic updates, updating UI

		immediately while saving in the background.
Accessibility and Keyboard Support	Users navigating via keyboard or screen readers may face difficulties.	Added ARIA attributes, ensured focusable elements, and maintained logical tab navigation.
User Experience and UI Responsiveness	Tasks and buttons may not behave consistently on different devices.	Used responsive CSS design and tested across multiple devices for smooth experience.

GITHUB REPOSITORY&SETUP GUIDE:

Repository name:

Github link:

1.CLONE THE REPOSITORY

Open your terminal or command prompt and run:

git clone https://github.com/username/todo-app-react-hooks.git

2. NAVIGATE TO THE PROJECT DIRECTORY

cd todo-app-react-hooks

3. INSTALL DEPENDENCIES

Make sure you have Node.js installed. Then run:

npm install

or, if using yarn:

yarn install

4. START THE DEVELOPMENT SERVER

npm start

or

yarn start

This will start the app at http://localhost:3000/ by default.

5. (Optional) Using JSON Server for Backend Simulation

If your app uses a mock backend: ☐ Install JSON Server globally (if not installed):

npm install -g json-server ☐ Run JSON Server with the provided db.json file:

json-server --watch db.json --port 5000

Your API will now be accessible at http://localhost:5000/todos.

6. Testing the Application

- **Open the browser at http://localhost:3000/.**
- **Add, edit, mark completed, or delete tasks to test all features.**
- **Ensure tasks persist across page reloads.**

FINAL SUBMISSION(REPO+DEPLOYED LINK):

- ✓ Demonstrates effective use of React Hooks for task management.
- ✓ Responsive, persistent, and user-friendly app.
- ✓ Future: backend integration, categories, priorities, dark/light themes, performance optimization.

Git hub repository link:

PROJECT TITLE :

TO-DO APP WITH REACT HOOKS

TECHNOLOGICAL USED:

- HTML
- CSS
- JAVASCRIPT

ORGANIZATION :

IBM-skillsbuild.nanmuthalvan

Git hub repository link: :