```
<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport"
content="width=devicewidth,initialscale=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:lineargradient(180d
eg,#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;borderradius:10px;border:1px solid
rgba(255,255,255,0.06);
background:transparent;color:inherit;outline:none;
} button{
backgrou
nd:var(accent);b
order:non e;paddin
g:10px
14px;bor
derradius :10px;fon
t-
weight:60
0;cursor: pointer;
boxshadow:0
6px 18px
rgba(125,
211,252,0
.08);
}
.controls{display:flex;gap:8px;alignitems:center;marginbottom:12
px}
```

```
.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{liststyle:none;padding:0;margin:0;display:grid;gap:8px
}
li.todo{display:flex;alignitems:center;gap:12px;padding:10px;bor
d erradius: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;borderradius:999px;background:rgba(255,255,255,0.02);font-
size:12px;color:var(--muted)}
.done{opacity:0.6;textdecoration:line-through}
footer{display:flex;justifycontent:spacebetween;alignitems:center;
margintop: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"
strokelinecap="round"/></svg>

<h1>Quick To-Do</h1>

<div class="meta" id="dateMeta" aria-live="polite"></div>

</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="marginleft:auto;display:flex;gap:8px;alignitems: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"
dataid="${it.id}">
<input type="checkbox" ${it.done ? 'checked' : ''} arialabel="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>
```

CSS:

```
 Body {
  Font-family: “Segoe UI”, sans-serif;
  Background: #f5f6fa;
  Display: flex;
  Justify-content: center;
  Align-items: center;
  Height: 100vh;
  Margin: 0;
}

#app {
  Width: 380px;
  Background: white;
  Padding: 25px;
  Border-radius: 15px;
  Box-shadow: 0 5px 15px rgba(0, 0, 0, 0.15);
}

.screen {
  Text-align: center;
}

.hidden {
  Display: none;
}

Input {
  Width: 80%;
  Padding: 10px;
  Margin: 8px 0;
  Border: 1px solid #ccc;
  Border-radius: 8px;
  Font-size: 16px;
}

Button {
  Padding: 10px 15px;
  Background: #007bff;
  Color: white;
  Border: none;
  Border-radius: 8px;
  Cursor: pointer;
  Font-size: 15px;
}

Button:hover {   Background: #0056b3;
}

Ul {
  List-style: none;
  Padding: 0;
}

Li {
  Display: flex;
  Justify-content: space-between;
  Align-items: center;
  Background: #f1f3f6;
  Margin: 6px 0;
  Padding: 10px;
  Border-radius: 8px;
  Font-size: 16px;
}

li.completed span {   textdecoration:
line-through;   color:
gray;  }

.top-bar {
  Display: flex;
  Justify-content: space-between;
  Align-items: center;
  Margin-bottom: 15px;
```

```
}


#loginError {
  Color: red;
  Margin-top: 10px;
  Font-size: 14px;
}
Script.Js:
   Const API_URL = http://localhost:5000/api;
Const loginScreen = document.getElementById("login-screen");
Const todoScreen = document.getElementById("todo-screen");
Const usernameInput = document.getElementById("username");
Const passwordInput = document.getElementById("password");
Const loginBtn = document.getElementById("loginBtn");
Const loginError = document.getElementById("loginError");
Const newTodoInput = document.getElementById("newTodo");
Const addBtn = document.getElementById("addBtn");
Const todoList = document.getElementById("todoList");
Const welcomeUser = document.getElementById("welcomeUser");
Const logoutBtn = document.getElementById("logoutBtn");


Let token = localStorage.getItem("token");
Let username = localStorage.getItem("username");
```

```
// Auto-login if token exists If
(token) {
showTodoScreen();   fetchTodos();
}

// Handle login loginBtn.addEventListener("click",
async () => {   const username =
usernameInput.value.trim();   const password =
passwordInput.value.trim();

  const res = await fetch(`${API_URL}/login`, {
method: "POST",     headers: { "Content-Type":
"application/json" },     body: JSON.stringify({ username,
password }),
  });

  If (res.ok) {
    Const data = await res.json();     localStorage.setItem("token",
data.token);     localStorage.setItem("username",
data.username);     showTodoScreen();     fetchTodos();
  } else {
    loginError.textContent = "❌ Invalid username or password";
  }
});

Function showTodoScreen() {
loginScreen.classList.add(“hidden”);
todoScreen.classList.remove(“hidden”);
welcomeUser.textContent = `Welcome,
${localStorage.getItem(“username”)}!`;
}


// Fetch todos
Async function fetchTodos() {
todoList.innerHTML = “Loading…”;   const res
= await fetch(`${API_URL}/todos`, {     headers:
{ Authorization: `Bearer
${localStorage.getItem(“token”)}` },

  });
  Const todos = await res.json();


  todoList.innerHTML = “”;   todos.forEach(renderTodo);
}


// Render single todo item
```

```
Function renderTodo(todo) {
  Const li = document.createElement("li");
li.classList.toggle("completed", todo.completed);

  const span = document.createElement("span");
span.textContent = todo.title;   span.addEventListener("click",
() => toggleTodo(todo));

  const delBtn = document.createElement("button");
delBtn.textContent = "❌";   delBtn.addEventListener("click",
() => deleteTodo(todo.id));

  li.appendChild(span);
li.appendChild(delBtn);
todoList.appendChild(li);
}

// Add new todo addBtn.addEventListener("click",
async () => {   const title = newTodoInput.value.trim();
  if (!title) return;

  const res = await fetch(`${API_URL}/todos`, {     method:
"POST",
    headers: {
      “Content-Type”: “application/json”,
      Authorization: `Bearer ${localStorage.getItem(“token”)}`,
    },
    Body: JSON.stringify({ title }),
  });

  If (res.ok) {
    Const newTodo = await res.json();
renderTodo(newTodo);     newTodoInput.value
= “”;
  }
});

// Toggle complete
Async function toggleTodo(todo) {
  Await fetch(`${API_URL}/todos/${todo.id}`, {
    Method: “PUT”,
    Headers: {
      “Content-Type”: “application/json”,
      Authorization: `Bearer ${localStorage.getItem(“token”)}`,
    },
    Body: JSON.stringify({ completed: !todo.completed }),
  });
  fetchTodos();
}

// Delete todo
Async function deleteTodo(id) {
  Await fetch(`${API_URL}/todos/${id}`, {
    Method: “DELETE”,
    Headers: { Authorization: `Bearer
${localStorage.getItem(“token”)}` },
  });
  fetchTodos();
}

// Logout logoutBtn.addEventListener(“click”,
() => {   localStorage.clear();   todoScreen.classList.add(“hidden”);
loginScreen.classList.remove(“hidden”);
});
```

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