**EXPERIMENT 4.1**

**AIM:**

CRUD Operations for Product Database Using Mongoose

**CODE:**

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width,initial-scale=1" />

<title>Product CRUD (Frontend) — Express + Mongoose</title>

<style>

:root{--accent:#0b69ff}

body{font-family:system-ui,-apple-system,Segoe UI,Roboto,Helvetica,Arial;max-width:900px;margin:28px auto;padding:20px}

h1{font-size:1.4rem;margin-bottom:6px}

.card{border:1px solid #e6e6e6;border-radius:10px;padding:12px;margin:12px 0;background:#fff}

label{display:block;margin:8px 0 4px;font-size:0.9rem}

input,select,button{padding:8px;border-radius:8px;border:1px solid #ccc;font-size:0.95rem}

.row{display:flex;gap:8px}

.row> \*{flex:1}

.actions{margin-top:10px}

table{width:100%;border-collapse:collapse;margin-top:12px}

th,td{padding:8px;border-bottom:1px solid #f0f0f0;text-align:left}

.muted{color:#666;font-size:0.9rem}

.btn{cursor:pointer;background:var(--accent);color:#fff;border:none;padding:8px 10px;border-radius:8px}

.btn.ghost{background:#fff;color:var(--accent);border:1px solid var(--accent)}

.small{padding:6px 8px;font-size:0.85rem}

.inline{display:inline-block}

.danger{background:#ff4d4f}

.pill{display:inline-block;padding:6px 8px;border-radius:999px;background:#f3f4f6;border:1px solid #eaeaea;font-size:0.85rem}

.hidden{display:none}

</style>

</head>

<body>

<h1>Product CRUD — Frontend</h1>

<p class="muted">This HTML page is a simple frontend that talks to a backend REST API using the typical endpoints used with the Mongoose example:</p>

<ul class="muted">

<li><code>POST /products</code> — create</li>

<li><code>GET /products</code> — read all</li>

<li><code>GET /products/:id</code> — read one</li>

<li><code>PUT /products/:id</code> — update</li>

<li><code>DELETE /products/:id</code> — delete</li>

</ul>

<div class="card">

<form id="createForm">

<label for="name">Name</label>

<input id="name" required placeholder="e.g. Whey Protein" />

<div class="row">

<div>

<label for="price">Price</label>

<input id="price" type="number" min="0" step="0.01" required placeholder="2999.00" />

</div>

<div>

<label for="category">Category</label>

<input id="category" placeholder="Health & Nutrition" />

</div>

</div>

<label><input id="inStock" type="checkbox" checked /> In Stock</label>

<div class="actions">

<button class="btn" type="submit">Create Product</button>

<button class="btn ghost" type="button" id="refreshBtn">Refresh List</button>

</div>

</form>

</div>

<div id="status" class="muted"></div>

<div class="card">

<strong>Products</strong>

<div id="listArea"></div>

</div>

<!-- Edit area (hidden) -->

<div id="editCard" class="card hidden">

<strong>Edit Product</strong>

<form id="editForm">

<input id="editId" type="hidden" />

<label for="editName">Name</label>

<input id="editName" required />

<div class="row">

<div>

<label for="editPrice">Price</label>

<input id="editPrice" type="number" min="0" step="0.01" required />

</div>

<div>

<label for="editCategory">Category</label>

<input id="editCategory" />

</div>

</div>

<label><input id="editInStock" type="checkbox" /> In Stock</label>

<div class="actions">

<button class="btn" type="submit">Save</button>

<button class="btn ghost" type="button" id="cancelEdit">Cancel</button>

</div>

</form>

</div>

<script>

// Change this base URL if your backend runs on a different host/port

const BASE = '';

const createForm = document.getElementById('createForm');

const status = document.getElementById('status');

const listArea = document.getElementById('listArea');

const refreshBtn = document.getElementById('refreshBtn');

const editCard = document.getElementById('editCard');

const editForm = document.getElementById('editForm');

function setStatus(msg, isError=false) {

status.textContent = msg;

status.style.color = isError ? 'crimson' : '';

}

async function fetchProducts() {

setStatus('Loading products...');

try {

const res = await fetch(BASE + '/products');

if (!res.ok) throw new Error('Failed to fetch');

const data = await res.json();

renderList(data);

setStatus(`Loaded ${data.length} product(s)`);

} catch (err) {

setStatus('Could not load products — is the backend running?','error');

listArea.innerHTML = '<p class="muted">No data</p>';

console.error(err);

}

}

function renderList(items) {

if (!items || !items.length) {

listArea.innerHTML = '<p class="muted">No products yet.</p>';

return;

}

let html = '<table><thead><tr><th>Name</th><th>Price</th><th>Category</th><th>In Stock</th><th>Actions</th></tr></thead><tbody>';

for (const p of items) {

html += `<tr>

<td>${escapeHtml(p.name)}</td>

<td>${Number(p.price).toFixed(2)}</td>

<td>${escapeHtml(p.category || '')}</td>

<td>${p.inStock ? '✅' : '❌'}</td>

<td>

<button class="small" onclick="startEdit('${p.\_id}')">Edit</button>

<button class="small" onclick="removeProduct('${p.\_id}')">Delete</button>

</td>

</tr>`;

}

html += '</tbody></table>';

listArea.innerHTML = html;

}

createForm.addEventListener('submit', async (e) => {

e.preventDefault();

const payload = {

name: document.getElementById('name').value.trim(),

price: Number(document.getElementById('price').value),

category: document.getElementById('category').value.trim(),

inStock: document.getElementById('inStock').checked,

};

setStatus('Creating...');

try {

const res = await fetch(BASE + '/products', {

method: 'POST',

headers: {'Content-Type':'application/json'},

body: JSON.stringify(payload)

});

if (!res.ok) {

const err = await res.json().catch(()=>({message:'Unknown'}));

throw new Error(err.message || 'Create failed');

}

const created = await res.json();

setStatus(`Created "${created.name}"`);

createForm.reset();

fetchProducts();

} catch (err) {

setStatus('Create failed — check console', true);

console.error(err);

}

});

refreshBtn.addEventListener('click', fetchProducts);

window.startEdit = async function(id) {

setStatus('Loading product...');

try {

const res = await fetch(BASE + '/products/' + id);

if (!res.ok) throw new Error('Not found');

const p = await res.json();

document.getElementById('editId').value = p.\_id;

document.getElementById('editName').value = p.name;

document.getElementById('editPrice').value = p.price;

document.getElementById('editCategory').value = p.category || '';

document.getElementById('editInStock').checked = !!p.inStock;

editCard.classList.remove('hidden');

setStatus('Loaded product for edit');

window.scrollTo({top:document.body.scrollHeight, behavior:'smooth'});

} catch (err) {

setStatus('Could not load product', true);

console.error(err);

}

}

editForm.addEventListener('submit', async (e) => {

e.preventDefault();

const id = document.getElementById('editId').value;

const payload = {

name: document.getElementById('editName').value.trim(),

price: Number(document.getElementById('editPrice').value),

category: document.getElementById('editCategory').value.trim(),

inStock: document.getElementById('editInStock').checked,

};

setStatus('Saving...');

try {

const res = await fetch(BASE + '/products/' + id, {

method: 'PUT',

headers: {'Content-Type':'application/json'},

body: JSON.stringify(payload)

});

if (!res.ok) {

const err = await res.json().catch(()=>({message:'Update failed'}));

throw new Error(err.message || 'Update failed');

}

const updated = await res.json();

setStatus(`Updated "${updated.name}"`);

editCard.classList.add('hidden');

fetchProducts();

} catch (err) {

setStatus('Update failed', true);

console.error(err);

}

});

document.getElementById('cancelEdit').addEventListener('click', () => {

editCard.classList.add('hidden');

});

window.removeProduct = async function(id) {

if (!confirm('Delete this product?')) return;

setStatus('Deleting...');

try {

const res = await fetch(BASE + '/products/' + id, {method:'DELETE'});

if (!res.ok) throw new Error('Delete failed');

setStatus('Deleted');

fetchProducts();

} catch (err) {

setStatus('Delete failed', true);

console.error(err);

}

}

function escapeHtml(s){

if (!s) return '';

return s.replace(/[&<>\"']/g, (c)=> ({'&':'&amp;','<':'&lt;','>':'&gt;','"':'&quot;',"'":'&#39;'}[c]));

}

// initial load

fetchProducts();

</script>

</body>

</html>

**OUTPUT:**

