**I. Adding Products/Buyers**

**1. User Interface (Front-End)**

* **HTML Form Structure**

<form id="productForm">

<label for="productName">Product Name:</label><br>

<input type="text" id="productName" name="productName" required><br>

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

<input type="text" id="category" name="category" required><br>

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

<input type="number" id="price" name="price" min="0" step="0.01" required><br>

<label for="quantity">Quantity:</label><br>

<input type="number" id="quantity" name="quantity" min="0" required><br>

<label for="description">Description:</label><br>

<textarea id="description" name="description"></textarea><br>

<input type="submit" value="Add Product">

</form>

<form id="buyerForm">

<label for="buyerName">Buyer Name:</label><br>

<input type="text" id="buyerName" name="buyerName" required><br>

<label for="email">Email:</label><br>

<input type="email" id="email" name="email" required><br>

<label for="phoneNumber">Phone Number:</label><br>

<input type="tel" id="phoneNumber" name="phoneNumber"><br>

<label for="address">Address:</label><br>

<textarea id="address" name="address"></textarea><br>

<input type="submit" value="Add Buyer">

</form>

**Validation (JavaScript)**

**function validateForm(formId) {**

**const form = document.getElementById(formId);**

**if (!form.checkValidity()) {**

**alert("Please fill in all required fields correctly.");**

**return false;**

**}**

**// Example: Email validation (you might want a more robust regex)**

**if (formId === "buyerForm") {**

**const emailField = form.querySelector("#email");**

**if (emailField && !emailField.value.includes("@")) {**

**alert("Please enter a valid email address.");**

**return false;**

**}**

**}**

**// Example: Price and Quantity validation (already done with HTML attributes, but can add more complex checks)**

**if (formId === "productForm") {**

**const price = parseFloat(form.querySelector("#price").value);**

**const quantity = parseInt(form.querySelector("#quantity").value);**

**if (price < 0 || quantity < 0) {**

**alert("Price and quantity must be positive values.");**

**return false;**

**}**

**}**

**return true;**

**}**

**Data Submission (JavaScript with Fetch API)**

**async function submitForm(event, formId, endpoint) {**

**event.preventDefault(); // Prevent default form submission**

**if (!validateForm(formId)) {**

**return;**

**}**

**const form = document.getElementById(formId);**

**const formData = new FormData(form);**

**const data = Object.fromEntries(formData.entries()); // Convert to object**

**try {**

**const response = await fetch(endpoint, {**

**method: "POST",**

**headers: {**

**"Content-Type": "application/json",**

**},**

**body: JSON.stringify(data),**

**});**

**const result = await response.json();**

**if (response.ok) {**

**alert(result.message || "Success!"); // Show success message**

**form.reset(); // Clear the form**

**} else {**

**alert(result.error || "An error occurred."); // Show error message**

**}**

**} catch (error) {**

**console.error("Error:", error);**

**alert("Failed to submit data.");**

**}**

**}**

**document.getElementById("productForm").addEventListener("submit", (event) => submitForm(event, "productForm", "/products")); // Replace '/products' with your actual server endpoint**

**document.getElementById("buyerForm").addEventListener("submit", (event) => submitForm(event, "buyerForm", "/buyers")); // Replace '/buyers' with your actual server endpoint**

**2). Back-End (Example with Node.js/Express and a conceptual database interaction)**

* **Server Setup (Node.js/Express)**

**const express = require("express");**

**const bodyParser = require("body-parser");**

**const app = express();**

**const port = 3000;**

**app.use(bodyParser.json());**

**// Conceptual Database Interaction (Replace with your actual database - e.g., PostgreSQL, MySQL, MongoDB)**

**const products = [];**

**const buyers = [];**

**Handling Product Creation**

**app.post("/products", (req, res) => {**

**const product = req.body;**

**// Basic Validation (You should enhance this with a library like 'express-validator')**

**if (!product.name || !product.category || !product.price || !product.quantity) {**

**return res.status(400).json({ error: "Missing required fields" });**

**}**

**product.id = products.length + 1; // Simple ID generation (replace with proper DB auto-increment)**

**products.push(product);**

**console.log("Product added:", product);**

**res.status(201).json({ message: "Product added successfully", product });**

**});**

**Handling Buyer Creation**

**app.post("/buyers", (req, res) => {**

**const buyer = req.body;**

**if (!buyer.name || !buyer.email) {**

**return res.status(400).json({ error: "Missing required fields" });**

**}**

**buyer.id = buyers.length + 1; // Simple ID generation**

**buyers.push(buyer);**

**console.log("Buyer added:", buyer);**

**res.status(201).json({ message: "Buyer added successfully", buyer });**

**});**

**Start the server**

**app.listen(port, () => {**

**console.log(`Server listening on port ${port}`);**

**});**

**Deleting Products/Buyers**

**Displaying Records (Front-End)**

* **HTML Table Structure**

**<h2>Products</h2>**

**<table id="productTable">**

**<thead>**

**<tr>**

**<th>ID</th>**

**<th>Name</th>**

**<th>Category</th>**

**<th>Price</th>**

**<th>Quantity</th>**

**<th>Action</th>**

**</tr>**

**</thead>**

**<tbody>**

**</tbody>**

**</table>**

**<h2>Buyers</h2>**

**<table id="buyerTable">**

**<thead>**

**<tr>**

**<th>ID</th>**

**<th>Name</th>**

**<th>Email</th>**

**<th>Phone</th>**

**<th>Action</th>**

**</tr>**

**</thead>**

**<tbody>**

**</tbody>**

**</table>**

**Fetching and Displaying Data (JavaScript)**

**async function displayRecords(tableId, endpoint, renderRow) {**

**const tableBody = document.querySelector(`#${tableId} tbody`);**

**tableBody.innerHTML = ""; // Clear existing rows**

**try {**

**const response = await fetch(endpoint);**

**const data = await response.json();**

**if (response.ok) {**

**data.forEach(item => {**

**const row = renderRow(item);**

**tableBody.appendChild(row);**

**});**

**} else {**

**alert("Failed to fetch data.");**

**}**

**} catch (error) {**

**console.error("Error:", error);**

**alert("Failed to fetch data.");**

**}**

**}**

**function renderProductRow(product) {**

**const row = document.createElement("tr");**

**row.innerHTML = `**

**<td>${product.id}</td>**

**<td>${product.name}</td>**

**<td>${product.category}</td>**

**<td>${product.price}</td>**

**<td>${product.quantity}</td>**

**<td><button class="delete-btn" data-id="${product.id}">Delete</button></td>**

**`;**

**return row;**

**}**

**function renderBuyerRow(buyer) {**

**const row = document.createElement("tr");**

**row.innerHTML = `**

**<td>${buyer.id}</td>**

**<td>${buyer.name}</td>**

**<td>${buyer.email}</td>**

**<td>${buyer.phoneNumber || ""}</td>**

**<td><button class="delete-btn" data-id="${buyer.id}">Delete</button></td>**

**`;**

**return row;**

**}**

**// Initial display of data**

**displayRecords("productTable", "/products", renderProductRow); // Replace '/products'**

**displayRecords("buyerTable", "/buyers", renderBuyerRow); // Replace '/buyers'**

**Delete Functionality (Front-End)**

**async function handleDelete(event, endpoint, refreshTable) {**

**if (!event.target.classList.contains("delete-btn")) return;**

**const id = event.target.dataset.id;**

**const confirmed = confirm("Are you sure you want to delete this record?"); // Confirmation dialog**

**if (!confirmed) return;**

**try {**

**const response = await fetch(`${endpoint}/${id}`, { method: "DELETE" }); // Send DELETE request**

**if (response.ok) {**

**alert("Record deleted successfully.");**

**refreshTable(); // Refresh the table to reflect changes**

**} else {**

**alert("Failed to delete record.");**

**}**

**} catch (error) {**

**console.error("Error:", error);**

**alert("Failed to delete record.");**

**}**

**}**

**document.getElementById("productTable").addEventListener("click", (event) => handleDelete(event, "/products", () => displayRecords("productTable", "/products", renderProductRow))); // Replace '/products'**

**document.getElementById("buyerTable").addEventListener("click", (event) => handleDelete(event, "/buyers", () => displayRecords("buyerTable", "/buyers", renderBuyerRow))); // Replace '/buyers'**

**. Back-End (Node.js/Express)**

* **Handling Product Deletion**

**app.delete("/products/:id", (req, res) => {**

**const id = parseInt(req.params.id);**

**const index = products.findIndex(p => p.id === id);**

**if (index === -1) {**

**return res.status(404).json({ error: "Product not found" });**

**}**

**products.splice(index, 1);**

**console.log("Product deleted:", id);**

**res.json({ message: "Product deleted successfully" });**

**});**

* **Handling Buyer Deletion**

**app.delete("/buyers/:id", (req, res) => {**

**const id = parseInt(req.params.id);**

**const index = buyers.findIndex(b => b.id === id);**

**if (index === -1) {**

**return res.status(404).json({ error: "Buyer not found" });**

**}**

**buyers.splice(index, 1);**

**console.log("Buyer deleted:", id);**

**res.json({ message: "Buyer deleted successfully" });**

**});**