Enhanced Dell E-commerce Inventory Management System(Object Destructuring)

Problem Statement

Design and develop a functional **inventory management system** for a Dell E-commerce application. This system will serve as a foundational backend component for a website, leveraging **modern JavaScript features** to ensure product information is accurate and up-to-date. The product catalog will be managed using a **JavaScript array of objects**, dynamically populated from an **external JSON file**.

Core Information Requirements

The system must efficiently handle and store the following information for each product:

- **id**: A unique identifier for the product (e.g., a number or string).
- **name**: The name of the product (e.g., 'Dell XPS 15').
- **inStock**: A **boolean** value indicating whether a product is currently in stock (true) or not (false).
- **price**: The price of the product (numeric value).
- **category**: A string representing the product category (e.g., 'Laptop', 'Monitor', 'Accessory').

Functional Requirements

The inventory management system must provide the following interactive capabilities:

- **Display Inventory:** 📋
  - The user should be able to view a **complete list** of all products in the inventory.
  - All properties (id, name, inStock, price, and category) for each product must be displayed.

- The inventory should be presented in a **tabular format** for clarity.
- The full inventory must be displayed **by default** when the application loads.

- **Check Availability:** 🔍
  - The user can input a product's **ID or Name** (case-insensitive search).
  - The system will then check its availability (in stock or out of stock) and report the current status.
  - The availability message should appear **temporarily** (e.g., for 3-5 seconds) and then automatically disappear.

Technical Specifications

- **Data Source:** 📁
  - The product catalog will be stored in an **external JSON file named products.json**.
  - This file must contain at least **50 product objects**, each conforming to the Core Information Requirements.
- **Data Structure:** 📦
  - The core of the inventory system must be implemented using a **JavaScript array of objects**.
  - This array will be populated dynamically by **fetching and parsing** the products.json file.
  - Each object in the array will represent a Dell product with the properties specified in the Core Information Requirements.
- **User Interface:** 🖥️
  - An interface built purely with **HTML and CSS** to allow user interaction and display information.
- **Programming Language:** 🚀

- **JavaScript** for all application logic, including fetching and parsing the JSON data.
- **Mandatory use of modern JavaScript features:**
  - **Arrow Functions**: For concise function definitions.
  - **Object Destructuring**: **Must be used** when accessing properties from product objects within functions or loops to improve code readability. For example, to access a product's name and price, you must use a syntax like { name, price } = product;.
  - **JavaScript Array Functions**: Utilize methods like map(), filter(), find(), forEach() for efficient array manipulation and iteration.

Testing Requirements

The following test cases must be performed to ensure the system functions correctly:

Display Inventory Test

- **Test Case 1.1:** Load the application with an initial list of products read from products.json.
  - **Expected Result:** All initial products and their properties are displayed correctly in a table.
- **Test Case 1.2:** (Implicit for Read-Only): Verify the table content matches the products.json data.

Check Stock Test

- **Test Case 2.1:** Check for a product that is in the inventory (by ID or Name).
  - **Expected Result:** A message confirming the product is in stock is displayed temporarily.
- **Test Case 2.2:** Check for a product that is not in the inventory (by ID or Name).
  - **Expected Result:** A message indicating the product is not found or is out of stock is displayed temporarily.

- **Test Case 2.3:** Check with an empty input.
    - **Expected Result:** A message prompting the user to enter a product ID/Name is displayed temporarily.

Learning Objectives

Upon completion of this project, students will be able to:

- **Core Concepts:** Understand and apply the fundamental concepts of JavaScript arrays and objects, including working with **arrays of objects**.
- **JSON Handling:** Master fetching and **parsing external JSON data** into JavaScript objects.
- **Modern JavaScript Functions:** Effectively write and use **arrow functions**, implement **object destructuring**, and leverage powerful **JavaScript array functions** (e.g., map, filter, find).
- **Event Handling:** Use event handling to make the web application interactive, responding to user actions like button clicks and form submissions.
- **CRUD Operations (Read):** Implement robust Read operations on an array of objects data structure.
- **HTML/CSS/JS Integration:** Seamlessly integrate HTML for structure, CSS for styling, and JavaScript for dynamic functionality to create a working web application.
- **Problem-Solving:** Break down a larger problem (inventory management) into smaller, manageable tasks and implement solutions for each.
- **Logic and Conditionals:** Utilize conditional statements (if/else) to handle different scenarios, such as checking for existing items or handling invalid inputs.
- **Asynchronous JavaScript:** Understand the basics of asynchronous operations (e.g., fetch with Promise or async/await) for loading data.