Backend Assignment: Simple Product Inventory API

Overview

This report details the successful completion of the simple product inventory REST API assignment. The objective was to create a RESTful API using Node.js and Express.js that supports standard CRUD (Create, Read, Update, Delete) operations on a list of products. All product data is stored in-memory using a JavaScript array, and all API responses are returned in JSON format.

Technologies Used

Core Technologies

- **Node.js**: The JavaScript runtime environment.
- **Express.js v4.x**: The web framework used to build the REST API.
- **JavaScript (ES6+)**: The programming language used for implementation.

Development Tools

- **Nodemon**: Utilized for automatic server restarts during development, enhancing productivity.
- **npm**: The package manager used for handling project dependencies.

Features Implemented

The API provides a comprehensive set of features to manage a product inventory.

- **REST API**: Full CRUD functionality has been implemented, enabling the creation, retrieval, updating, and deletion of product data.
- **In-Memory Storage**: Products are stored in a simple JavaScript array, eliminating the need for a separate database and fulfilling the assignment's core requirement.
- **JSON Responses**: All API endpoints return data in a standard JSON format, ensuring easy consumption by client applications.
- **Error Handling**: Robust error handling is in place, providing clear and informative error messages with appropriate HTTP status codes (e.g., 404 Not Found, 400 Bad Request).
- **CORS Support**: Cross-Origin Resource Sharing is enabled, allowing the API to be accessed from different domains.
- **Input Validation**: Essential request body fields are validated to ensure data integrity.

Bonus Tasks Completed

The following bonus tasks were successfully implemented, adding advanced functionality to the API:

1. Search Functionality

- **Endpoint**: GET /api/products/search?q=term
- **Description**: This endpoint allows users to perform a case-insensitive search on the product data. The search query (q) is matched against both the product's name and description fields.
- **Example**: A request to /api/products/search?q=shoe will return all products that have the word "shoe" in either their name or description.

2. Pagination

- **Endpoint**: GET /api/products?page=1&limit=10
- **Description**: This feature provides paginated results for the product listing endpoint. The page parameter specifies the page number, while the limit parameter defines the number of items per page. The API response includes metadata such as the total number of pages, the current page, and indicators for whether there is a next or previous page.
- **Example**: The request /api/products?page=1&limit=10 will retrieve the first 10 products from the inventory, along with relevant pagination information.

API Endpoints Summary

Method	**Endpoint**	**Description**
GET	/api/products	Get all products with optional pagination.
GET	/api/products/search?q=term	**BONUS**: Search products by name or description.
GET	/api/products/:id	Get a single product by its unique ID.
POST	/api/products	Add a new product to the inventory.
PUT	/api/products/:id	Update an existing product by its ID.
DELETE	/api/products/:id	Delete a product from the inventory by its ID.