SURANA COLLEGE-AUTONOMOUS

Affiliated to Bangalore University

Recognized under 2(f) and 12 B of UGC, Certified by ISO, Re-accredited by NAAC with A+

No 16, South End Road, Bengaluru-560 004

WEB PROGRAMMING REPORT

Project title: CRUD OPERATIONS ON PRODUCT INVENTORY MANAGEMENT

Course: Bachelor of Computer Applications (BCA)

Submitted By,

ABHISHEK. V - U03ME23S0037 (5th Semester BCA 'A'),

NITHISH S - U03ME23S0046 (5th Semester BCA 'A')

Under the Guidance of,

Prof. Vidya. A

Assistant Professor, Dept of Computer Science SURANA-COLLEGE-Autonomous, Bengaluru- 560 004

` **2025-2026**

DECLARATION

ABHISHEK. V

NOVEMBER 2025

BCA

I, **ABHISHEK V** hereby declare that the project report, entitled **"CRUD OPERATIONS ON Product Inventory Management"** submitted to **SURANA COLLEGE-AUTONOMOUS, i**n partial fulfilment for the curriculum requirement for **BACHELOR OF COMPUTER APPLICATIONS** done by me during 2025-2026 under the guidance of **P r o f. V i d y a. A,** Assistant Professor, Department of Computer Science, SURANA COLLEGE-Autonomous, Bengaluru. It has not formed the basis for the award any Degree/Diploma/Associateship/Fellowship or other similar title to any candidate in any College/University.

ABHISHEK. V

DECLARATION

NITHISH. S

NOVEMBER 2025

BCA

I**, NITHISH. S** hereby declare that the project report, entitled **"CRUD OPERATIONS ON Product Inventory Management" submitted** to **SURANA COLLEGE-AUTONOMOUS, i**n partial fulfilment for the curriculum requirement for **BACHELOR OF COMPUTER APPLICATIONS** done by me during 2025-2026 under the guidance of **prof. V i d y a. A,** Assistant Professor, Department of Computer Science, SURANA COLLEGE-Autonomous, Bengaluru. It has not formed the basis for the award any Degree/Diploma/Associateship/Fellowship or other similar title to any candidate in any College/University.

NITHISH. S

ACKNOWLEDGEMENT

We are extremely delighted and grateful to thank every single person who helped me in completing my dissertation successfully. I thank the almighty for bestowing me with his blessings. I'm very much indebted and extend my deep sense of gratitude to **P r o f .Vidya.A,** Assistant Professor, Department of Computer Science, for her consistent words of motivation, encouragement and filling me completely with insight regarding the topic throughout the period of my study. She was present whenever I needed her the most during the completion of my Project.

My sincere thanks are to the faculty members of Department of Computer Science, for their encouragement in my academic endeavors.

I thank my parents and for being very liberal towards me and for having complete trust in me during this project.

ABHISHEK V
NITHISH S

TABLE CONTENTS

CHAPTER 1: ABSTRACT

This project is a fundamental Inventory Management System designed to teach and demonstrate core database operations. Built using PHP for server-side logic and MySQL for data storage (accessed via phpMyAdmin), the system focuses on implementing the essential CRUD (Create, Read, Update, Delete) functionalities required for managing product inventory. The application allows users to effectively add new products, display current stock records (including pricing and supplier details), modify existing entries (such as updating price or quantity), and delete obsolete items. The frontend utilizes HTML, CSS, and JavaScript to provide a responsive and accessible interface.

By integrating these technologies, the system offers practical, hands-on experience in PHP-MySQL integration, dynamic form handling, and applying basic inventory logic, making it an ideal exercise in foundational database-driven web development.

Functionally, the application allows users to add new products, view all inventory records (including stock and pricing), modify existing product details (like quantity or price), and delete obsolete entries.

It serves as a core, practical exercise demonstrating effective PHP-MySQL integration, dynamic form handling, and the application of basic inventory logic within a data-driven web environment. This blend of technologies not only delivers a functional system but also provides hands-on experience across the full stack from interface design and dynamic forms to robust backend scripting and database interaction—serving as a crucial foundation for aspiring backend and full-stack developers.

CHAPTER 2: INTRODUCTION

Overview of the Project:

The modern business landscape, characterized by high volumes of transactions and diverse product lines, necessitates sophisticated and reliable methods for managing physical and digital assets. This project, Product Inventory Management System, addresses this need by creating a foundational, database-driven web application designed to efficiently track and manage product stock. Built using the LAMP stack specifically PHP for server-side execution, MySQL for data persistence, and HTML, CSS, and JavaScript for the responsive user interface the system provides a comprehensive platform for the systematic entry, retrieval, modification, and deletion of inventory records.

Importance of CRUD Operations in Applications:

At the heart of every functional data-driven application are the four fundamental operations: Create, Read, Update, and Delete (CRUD). These operations constitute the lifeblood of data interaction, defining how information flows between the client layer and the persistent storage layer (the database). The successful implementation of CRUD is crucial as it governs data integrity, enables real-time changes to be recorded, and ensures the application accurately reflects the current state of the inventory. Therefore, mastering the technical implementation of CRUD via server-side scripting is essential for understanding the architecture and flow of full-stack development.

Aim and Scope of the Project:

The primary aim of this project is to successfully implement all four CRUD operations for managing product inventory data. This involves establishing a stable, secure connection between the PHP backend and the MySQL database, utilizing effective SQL commands to manipulate data, and presenting the information clearly to the user. The scope of this project is focused specifically on the management of core product attributes, including unique identifiers, product name, current quantity in stock, unit price, and primary supplier information.

This exercise serves as a critical foundation in database interaction and dynamic web content generation, demonstrating core skills required for developing more complex enterprise applications.

CHAPTER 3: OBJECTIVE

The primary objective of this project is to build a functional, database-driven Product Inventory Management System capable of executing all four fundamental CRUD operations. This involves designing a responsive user interface and a corresponding MySQL database schema to ensure accurate data storage. Furthermore, the goals extend beyond basic functionality to include promoting data integrity through robust validation, automating the tracking of stock levels to minimize manual errors, and developing a clear Managerial Dashboard for full control over product and supplier data. The overarching goal is to gain practical experience in integrating client-side design with PHP server-side logic, showcasing how technology can transform logistical challenges into efficient, data-driven processes.

The specific objectives of this project are as follows:

- Interface Design: To design an interactive, user-friendly, and responsive interface that simplifies navigation for the Inventory Manager, ensuring smooth access across various devices.
- CRUD Implementation: To implement Create, Read, Update, and Delete (CRUD) operations effectively, allowing managers to manage Product Listings and Supplier Profiles.
- Security and Validation: To establish robust data validation mechanisms that protect the database from invalid entries (e.g., negative stock, non-numeric prices) and ensure system integrity.
- Database Integration: To integrate a robust MySQL database for efficient data storage, retrieval, and management, ensuring consistency and reliability of all product and supplier information.
- Dashboard Development: To develop a clear dashboard for the Inventory Manager that allows for full CRUD management of all product listings and Read and Update access to supplier details.

- Data Integrity and Traceability: To promote accuracy and trust by ensuring every stock change is accurately recorded and that the system prevents data conflicts, such as entering negative stock.
- Technical Awareness: To enhance technical awareness by showcasing how modern web technologies can transform traditional logistical challenges into smart, data-driven platforms for efficient business operations.
- Scalability: To create a scalable and extendable platform that can be further improved by integrating advanced features like Barcode Scanning, Stock In/Stock Out tracking, or Advanced Reporting in future iterations.

CHAPTER 4: SYSTEM REQUIREMENTS

To develop and test the CRUD-based Student Management System, the following system configuration was used. These requirements ensure smooth performance, reliable database connectivity, and proper execution of all web functionalities.

Since the system is built using PHP and MySQL within a XAMPP environment It can be implemented efficiently on any standard computer system with moderate specifications

4.1 Hardware Requirements

The project was developed on a machine with the following specifications:

- Processor: Intel Core i5 / AMD Ryzen 5 (or equivalent)
- RAM: Minimum 16 GB (Recommended for multitasking, running the local server, and
- browser testing)
- Storage: At least 30 GB of free disk space for project files, XAMPP installation, and
- database storage.
- Hard Disk: 1024 GB SSD (Solid State Drive) preferred for better performance.
- Display: 1024 × 768 resolution or higher for proper interface visualization.
- Input Devices: Standard keyboard and mouse for navigation and data entry.
- Network Connectivity: Stable internet or local network connection for hosting and
- accessing the application during testing.

The above configuration is sufficient for developing and running the web-based application in a local or institutional environment. For large-scale implementation, higher specifications and dedicated server systems are recommended.

4.2 Software Requirements

The software stack and tools used include:

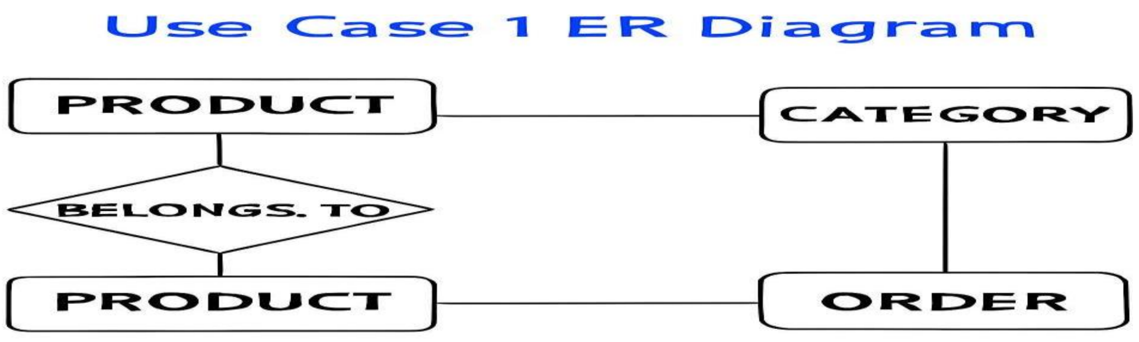
- Operating System: Windows 10 or later / macOS / Linux (Ubuntu preferred for open-
- source environment).
- Development Environment: Visual Studio Code, php Storm, or Sublime Text (for
- editing HTML, CSS, PHP, and JavaScript files).
- Local Server Package: XAMPP (includes Apache, PHP, and MySQL) for local
- hosting and database management.
- Backend Technology: PHP 8.0+ for server-side scripting and logic execution.
- Database: MySQL (5.7+ or MariaDB 10.0+) for storing and retrieving user, profile,
- and donation data.
- Frontend Technologies: HTML5, CSS3, and JavaScript for user interface design and
- interactivity.
- Browser: Google Chrome, Mozilla Firefox, or Microsoft Edge for running and testing
- the application.
- Version Control (Optional): Git/GitHub for project version tracking and
- collaboration.

These software components ensure that the project runs efficiently, allowing the developer to design, test, and deploy all modules within a controlled and secure environment.

CHAPTER 5: SYSTEM DESIGN

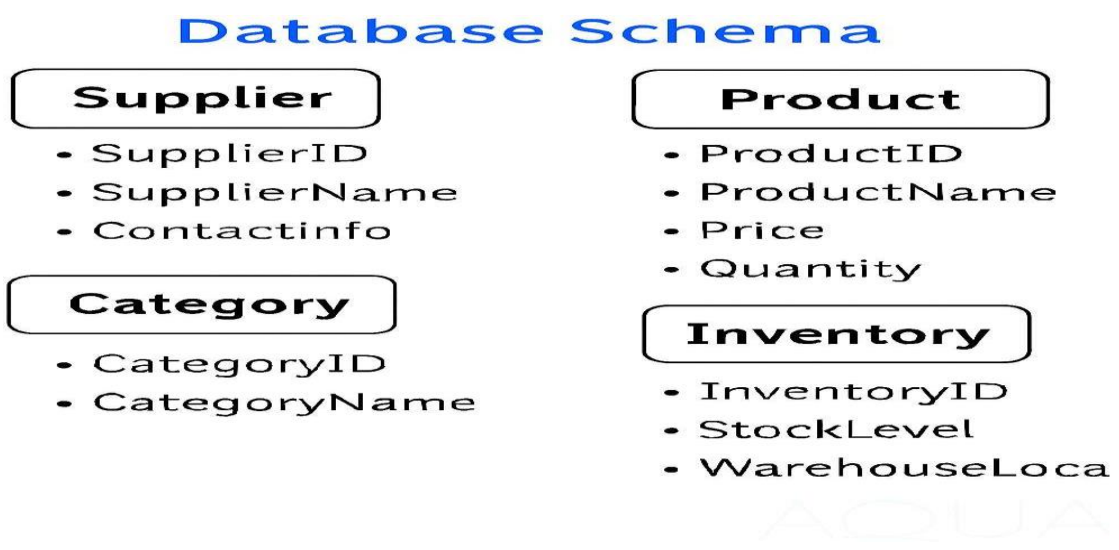
5.1 ER Diagram (Entity Relationship Diagram)

The Entity-Relationship Diagram represents the logical relationships between different entities involved in the “Product Inventory Management”. It illustrates how data such as users, products, and the donations themselves are connected within the system database. This diagram illustrates the logical structure of the database, showing the entities, their attributes.



Entity	Attributes	Key	Relationship
User	User ID, Username, Full Name, Email, Password Hash	PRIMARY KEY: User ID	1: N with Product (One User can manage Many Products)
Product	Product ID, User ID, SKU, Name, Price, Stock Quantity, Category, Summary, Image URL	PRIMARY KEY: Product ID, FOREIGN-KEY: User ID	N:1 with User

5.2 Database Schema:



This is the SQL representation of the ER diagram, defining the tables and their columns.

Table: Users (Stores user authentication and profile information)

```
CREATE TABLE Users (
    User id        INT PRIMARY KEY AUTO_INCREMENT,
    username       VARCHAR (50) NOT NULL UNIQUE,
    password hash  VARCHAR (255) NOT NULL, -- Storing a hash of the password
    email          VARCHAR (100) UNIQUE,
    full name      VARCHAR (100),
    created at     TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Table: Products (Stores all inventory items managed by a user)

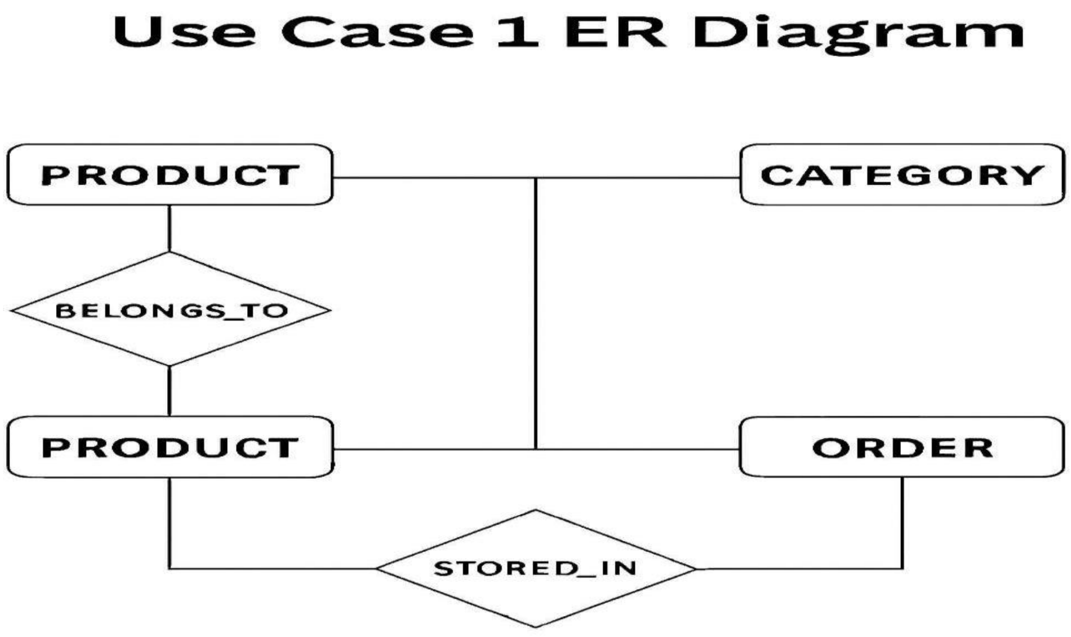
```
CREATE TABLE Products (
    Product id     INT PRIMARY KEY AUTO_INCREMENT,
    user_ id       INT NOT NULL,
    sku            VARCHAR (50) NOT NULL UNIQUE, -- Stock Keeping Unit
    name           VARCHAR (255) NOT NULL,
    price          DECIMAL (10, 2) NOT NULL,
    stock quantity   INT NOT NULL DEFAULT 0,
   category        VARCHAR (100),
    summary         TEXT,
    image URL        VARCHAR (255),
    created at       TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated  at              TIMESTAMP  DEFAULT  CURRENT_TIMESTAMP  ON  UPDATE
CURRENT_TIMESTAMP,
    Define Foreign Key relationship
    FOREIGN KEY (user id) REFERENCES Users (user id) ON DELETE CASCADE
);.
```

5.3. Use Case Diagrams (2 Use Cases):

Use Case diagrams illustrate the functional requirements of the system by showing how different actors interact with the system.

5.3.1 Use Case Diagram 1: User Authentication & Account Management:

This diagram focuses on the access control and user profile features of the system.



PRODUCT: Represents individual items available for sale or stock. Attributes typically include Productid, ProductName, Price, and Quantity.

- CATEGORY: Defines logical groupings of products (e.g., Electronics, Apparel, Groceries). Attributes include Category id and Category Name.
- INVENTORY: Captures physical storage details such as Inventory id, Stock Level, and Warehouse Location.
- Relationships:
- PRODUCT → BELONGS_TO → CATEGORY

  This relationship ensures that every product is assigned to a specific category. It supports:

Easier filtering and browsing by category.

Category-level analytics (e.g., sales trends, stock turnover).

Streamlined product management for suppliers and managers.

- PRODUCT → STORED_IN → INVENTORY

  This relationship maps each product to its physical storage location. It enables:
- Real-time stock level tracking.
- Efficient warehouse operations (e.g., restocking, retrieval).
- Location-based inventory audits and optimization.

Strategic Relevance:

This ER model is crucial for systems that require both logical classification and physical traceability of products. It supports:

- Multi-category product listings.
- Distributed inventory systems across multiple warehouses.
- Integration with order processing and supplier modules.
-

```
Graph TD:
subgraph System Boundary: Inventory Tracker
        UC1[Sign Up]
        UC2[Log In]
        UC3[Log Out]
        UC4[Forgot Password]
        end
```

A[User] -- interacts with --> UC1

A -- interacts with --> UC2

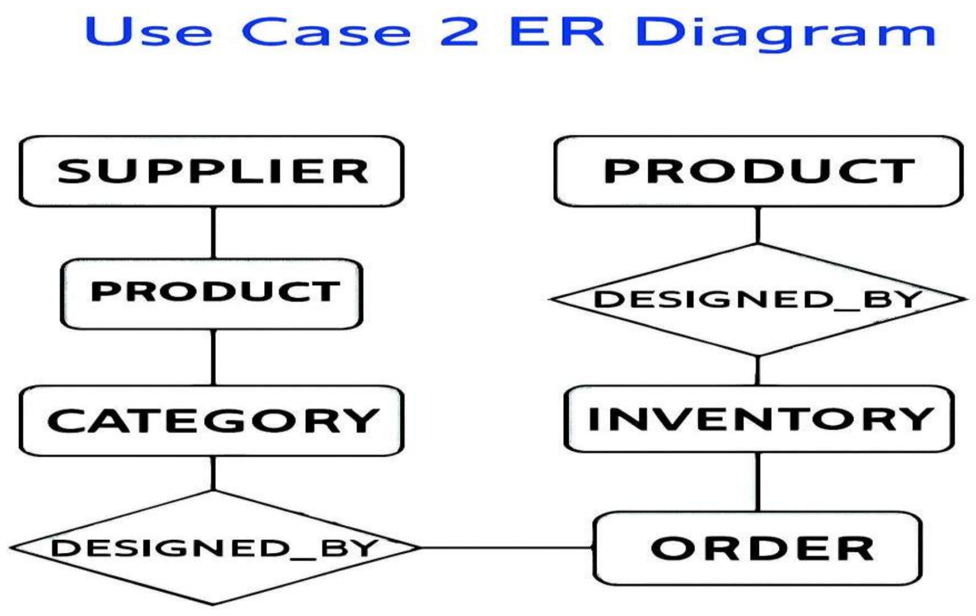
A -- interacts with --> UC3

A -- interacts with --> UC4

UC2 --> Logged in Session (Session Started)): <<includes>>

5.3.2 Use Case Diagram 2: Inventory Management (CRUD Operations):

This diagram focuses on the core functionality of managing product data within the inventory**.**



This expanded ER diagram captures the broader operational flow involving Supplier, Product, Order, Category, and Inventory.

- Suppliers are connected to Products through the SUPPLIES relationship, denoting sourcing origins.
- Each Product is linked to an Order via PRODUCED_BY, and the Order is further tied to a Category through DESIGNED_BY, reflecting product planning and classification.

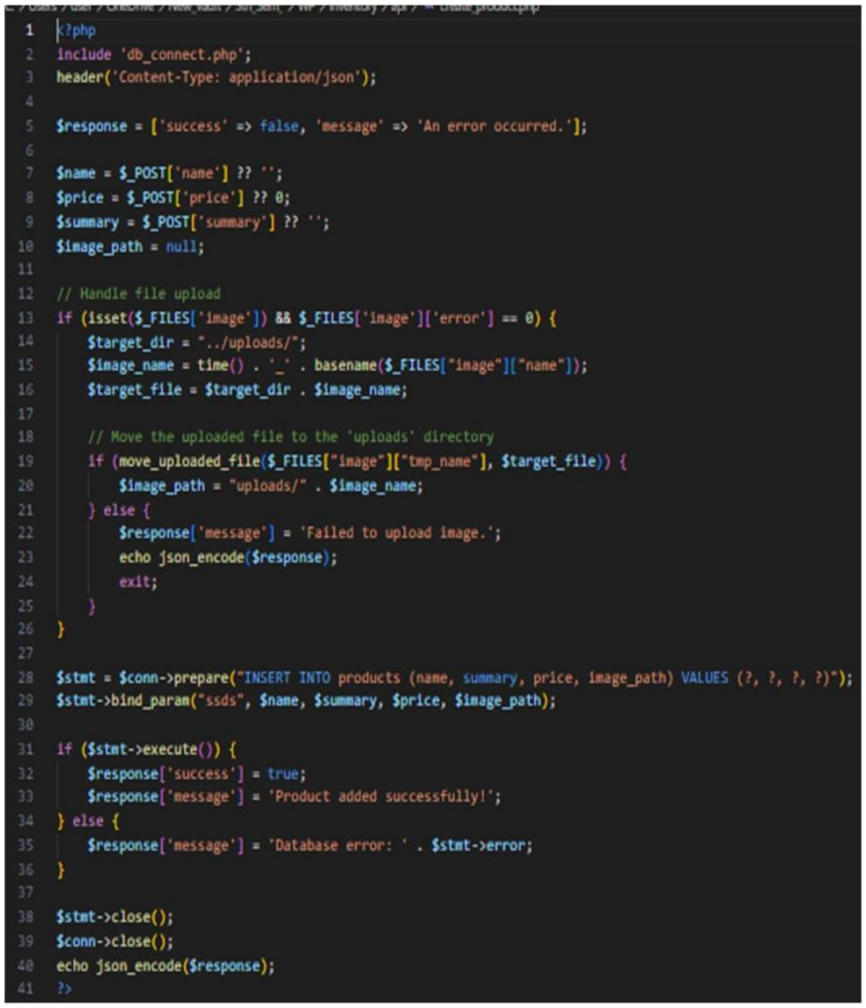
- The Inventory entity is connected to Product through CONTAINS, indicating stock levels and warehouse placement.
  This structure supports procurement, order tracking, and inventory control across multiple dimensions.

```
Graph TD:
    subgraph System Boundary: Product Inventory Management
      UC1[Manage Products]
      UC2[View Dashboard/Reports]
      UC3[Search/Filter Products]

      UC1 -- <<includes>> --> C [Add Product (Create)]
      UC1 -- <<includes>> --> R [View Products (Read)]
      UC1 -- <<includes>> --> U [Update Product Details (Update)]
      UC1 -- <<includes>> --> D [Delete Product (Delete)]
     end

   B [Authenticated User] -- interacts with --> UC1
   B -- interacts with --> UC2
   B -- interacts with --> UC3.
```

CHAPTER 6: IMPLEMENTATION

CRUD Operation Details:

6.1. Create: How New Records are Added



```php
<?php
include 'db_connect.php';
header('Content-Type: application/json');

$response = ['success' => false, 'message' => 'An error occurred.'];

$name = $_POST['name'] ?? '';
$price = $_POST['price'] ?? 0;
$summary = $_POST['summary'] ?? '';
$image_path = null;

// Handle file upload
if (isset($_FILES['image']) && $_FILES['image']['error'] == 0) {
    $target_dir = "../uploads/";
    $image_name = time() . '_' . basename($_FILES["image"]["name"]);
    $target_file = $target_dir . $image_name;

    // Move the uploaded file to the 'uploads' directory
    if (move_uploaded_file($_FILES["image"]["tmp_name"], $target_file)) {
        $image_path = "uploads/" . $image_name;
    } else {
        $response['message'] = 'Failed to upload image.';
        echo json_encode($response);
        exit;
    }
}

$stmt = $conn->prepare("INSERT INTO products (name, summary, price, image_path) VALUES (?, ?, ?, ?)");
$stmt->bind_param("ssds", $name, $summary, $price, $image_path);

if ($stmt->execute()) {
    $response['success'] = true;
    $response['message'] = 'Product added successfully!';
} else {
    $response['message'] = 'Database error: ' . $stmt->error;
}

$stmt->close();
$conn->close();
echo json_encode($response);
?>
```

Fig1.1 Create Database:

6.1. Create: How New Records are Added

The Create operation adds a new Product record to the database.

❖ User Action: The user fills out the form in add.html and clicks the "Add Product" button.

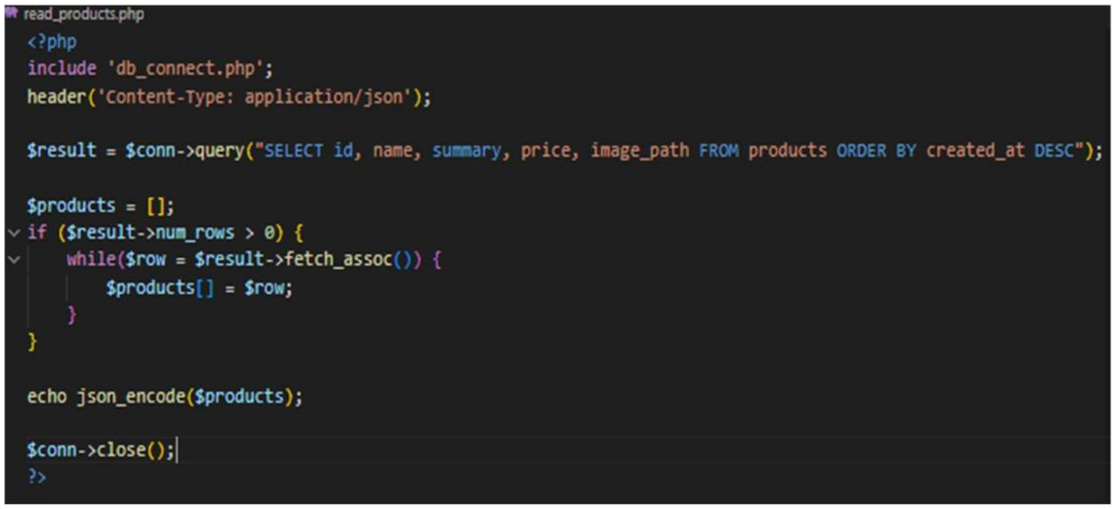
❖ Client-Side (JavaScript - from add.html):

  ➢ The form data, including text fields and the Product Image file, is packaged into a Form Data object.

  ➢ An asynchronous fetch request is sent using the POST method.

- ➢ The request target is the backend PHP script responsible for creation: Api /create product.

6.2. Read: How Data is Displayed or Fetched:



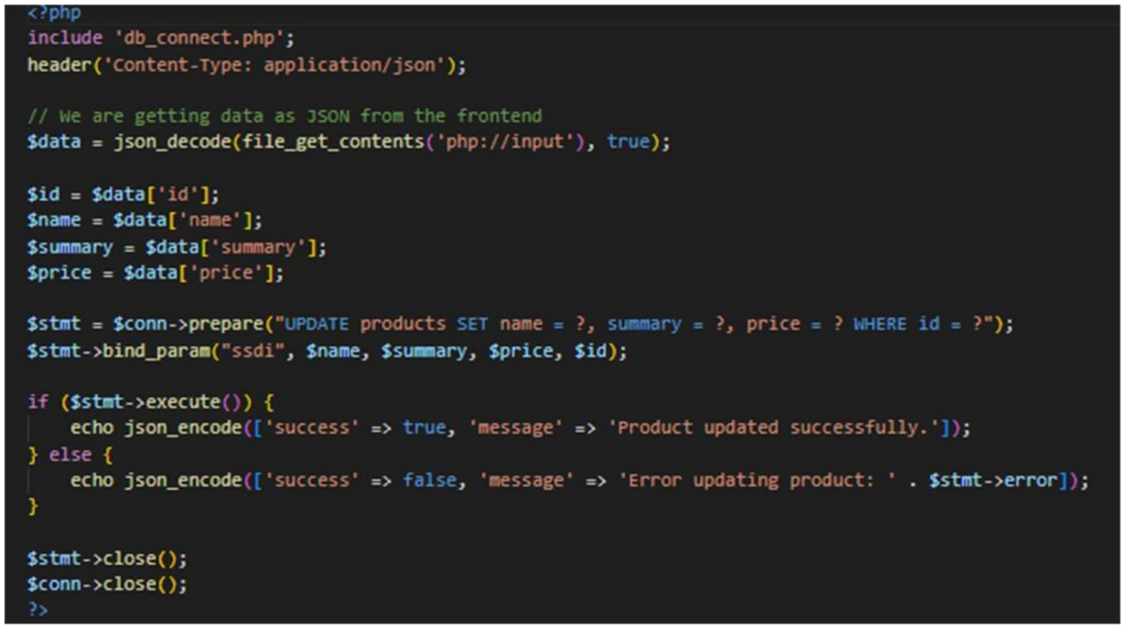
```php
read_products.php
<?php
include 'db_connect.php';
header('Content-Type: application/json');

$result = $conn->query("SELECT id, name, summary, price, image_path FROM products ORDER BY created_at DESC");

$products = [];
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        $products[] = $row;
    }
}

echo json_encode($products);

$conn->close();
?>
```

Fig 1.2 Read Data

Client-Side (JavaScript - from sim.html):

- ➢ A function (load Products or equivalent) is called on page load.
- ➢ An asynchronous fetch request is sent using the GET method to a dedicated read endpoint.
- ➢ The received JSON data (an array of product objects) is iterated over, and corresponding HTML table rows (<tr>) are dynamically constructed and inserted into the product table.

6.3. Update: How Existing Data is Modified

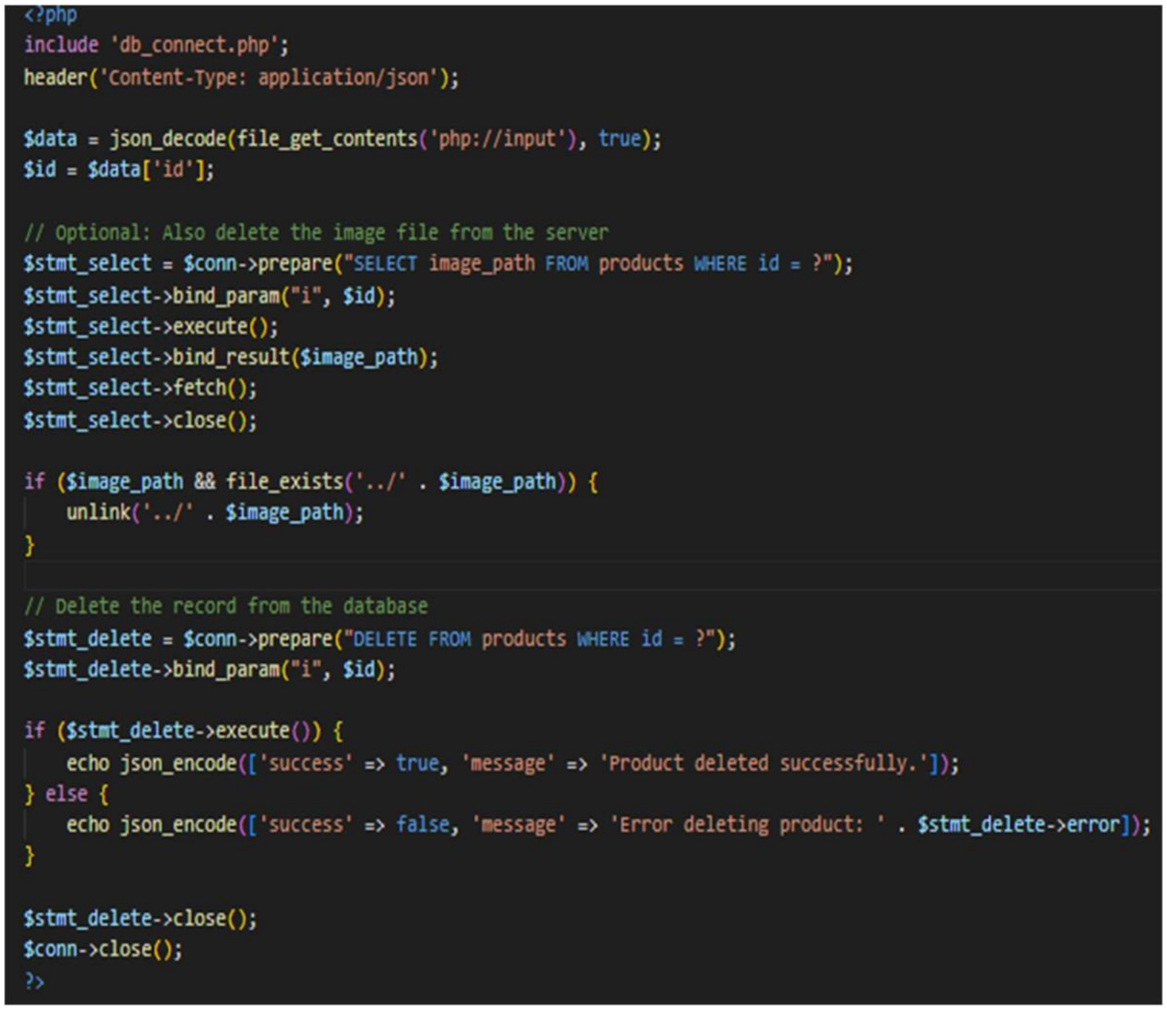


```php
<?php
include 'db_connect.php';
header('Content-Type: application/json');

// We are getting data as JSON from the frontend
$data = json_decode(file_get_contents('php://input'), true);

$id = $data['id'];
$name = $data['name'];
$summary = $data['summary'];
$price = $data['price'];

$stmt = $conn->prepare("UPDATE products SET name = ?, summary = ?, price = ? WHERE id = ?");
$stmt->bind_param("ssdi", $name, $summary, $price, $id);

if ($stmt->execute()) {
    echo json_encode(['success' => true, 'message' => 'Product updated successfully.']);
} else {
    echo json_encode(['success' => false, 'message' => 'Error updating product: ' . $stmt->error]);
}

$stmt->close();
$conn->close();
?>
```

Fig 1.3 Update Database

- ❖ User Action: The user clicks the Edit button in the product table, an edit modal pops up, the user changes fields, and clicks Save Changes.
- ❖ Client-Side (JavaScript - from sim.html):
- ➢ The "Save Changes" button listener collects the updated data from the modal form fields.
- ➢ The data (including the hidden id of the product) is converted into a JSON string.
- ➢ An asynchronous fetch request is sent using the POST method to the update endpoint: Api/update, product. php

6.4. Delete: How Data is Removed:



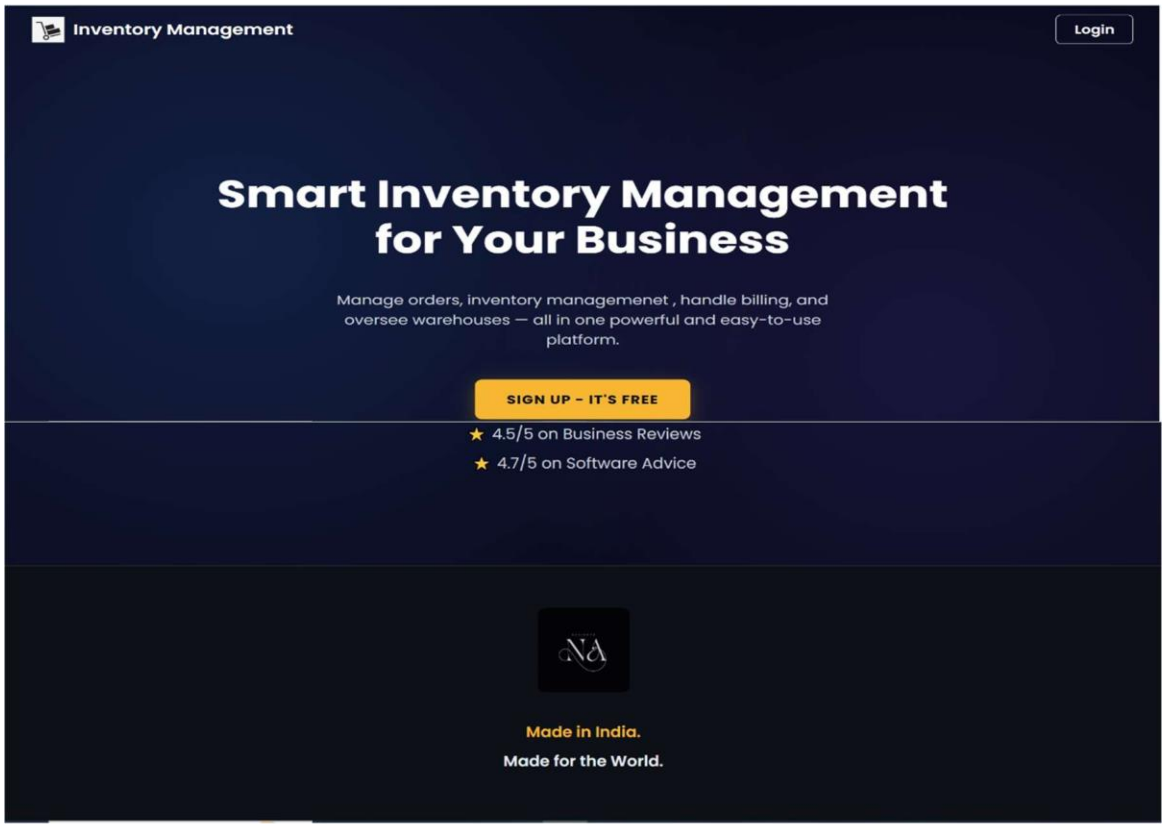
```php
<?php
include 'db_connect.php';
header('Content-Type: application/json');

$data = json_decode(file_get_contents('php://input'), true);
$id = $data['id'];

// Optional: Also delete the image file from the server
$stmt_select = $conn->prepare("SELECT image_path FROM products WHERE id = ?");
$stmt_select->bind_param("i", $id);
$stmt_select->execute();
$stmt_select->bind_result($image_path);
$stmt_select->fetch();
$stmt_select->close();

if ($image_path && file_exists('../' . $image_path)) {
    unlink('../' . $image_path);
}

// Delete the record from the database
$stmt_delete = $conn->prepare("DELETE FROM products WHERE id = ?");
$stmt_delete->bind_param("i", $id);

if ($stmt_delete->execute()) {
    echo json_encode(['success' => true, 'message' => 'Product deleted successfully.']);
} else {
    echo json_encode(['success' => false, 'message' => 'Error deleting product: ' . $stmt_delete->error]);
}

$stmt_delete->close();
$conn->close();
?>
```

Fig1.4 Delete Database

- User Action: The user clicks the Delete button next to a product on the table.
- Client-Side (JavaScript - Inferred logic from sim.html):
- The Product ID is retrieved from the button's data attribute (data-id).
- A confirmation dialog is typically displayed for safety.
- An asynchronous fetch request is sent to the delete endpoint (e.g., Api/delete product. php), typically including the ID in the request body or URL.

CHAPTER 7: RESULTS AND OUTPUT

Fig 7.1



Smart Inventory Management for Your Business

Manage orders, inventory managemenet , handle billing, and oversee warehouses — all in one powerful and easy-to-use platform.

★ 4.5/5 on Business Reviews

★ 4.7/5 on Software Advice

Made in India.

Made for the World.

Fig 7.2

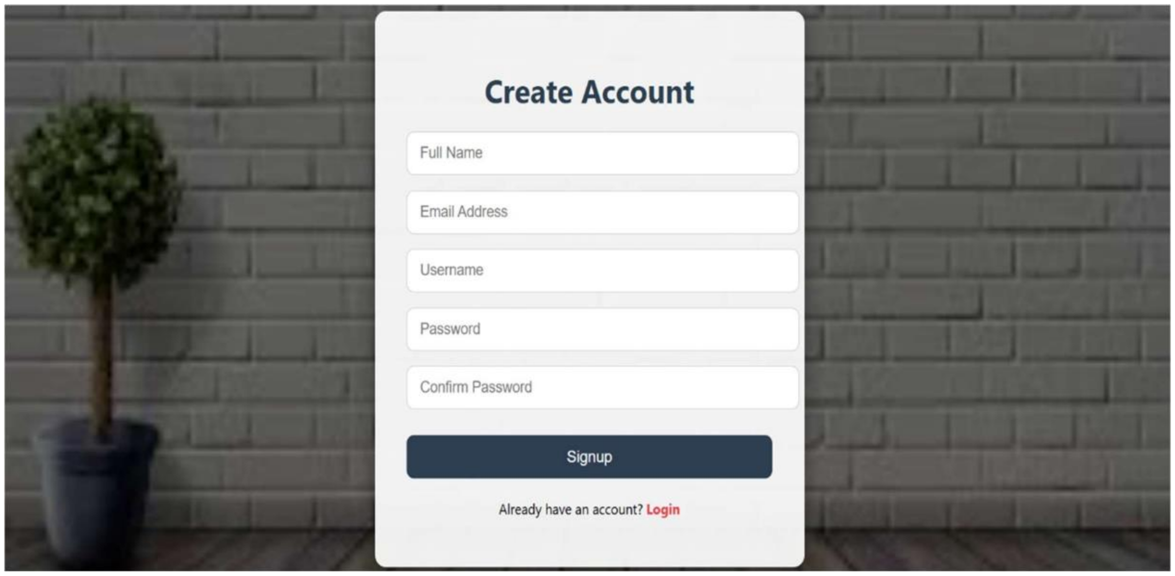


Fig 7.3

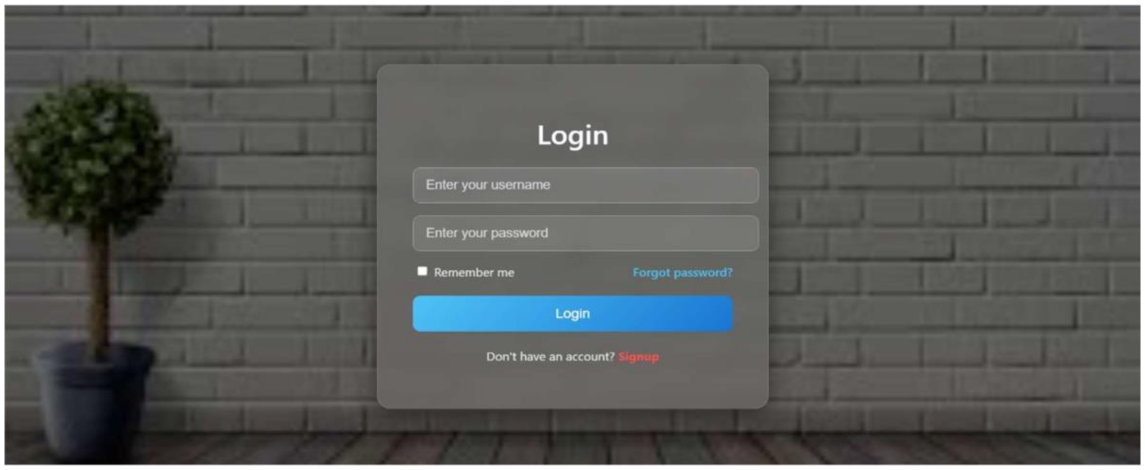
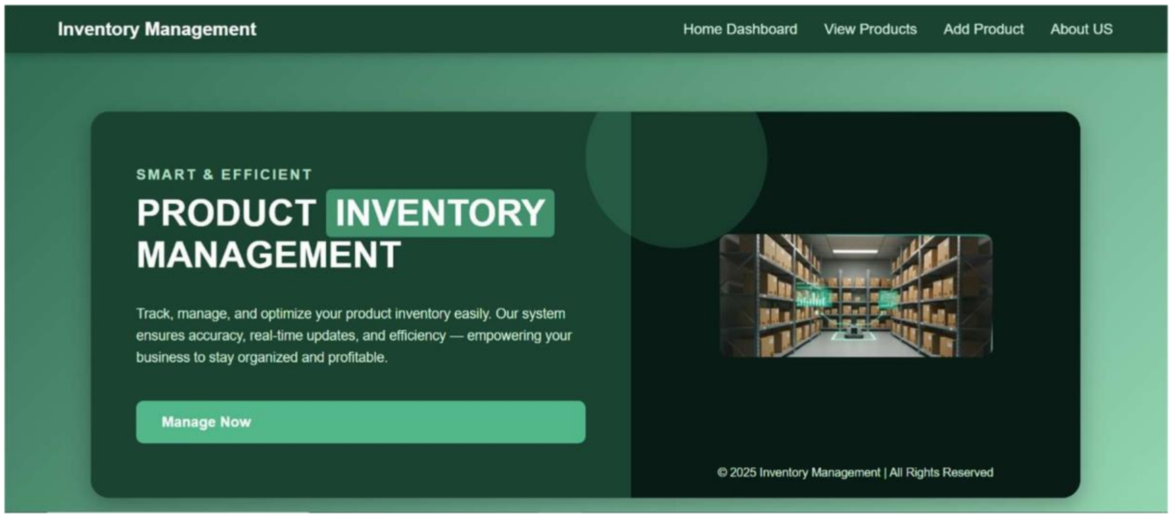


Fig 7.4



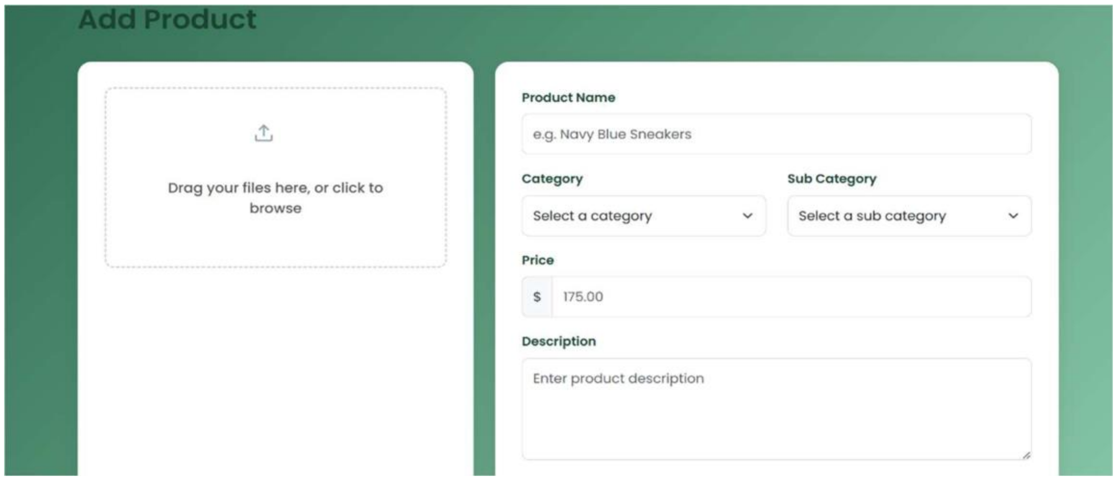
SMART & EFFICIENT

PRODUCT INVENTORY MANAGEMENT

Track, manage, and optimize your product inventory easily. Our system ensures accuracy, real-time updates, and efficiency — empowering your business to stay organized and profitable.

© 2025 Inventory Management | All Rights Reserved

Fig 7.5


Add Product

Drag your files here, or click to browse

Product Name: e.g. Navy Blue Sneakers

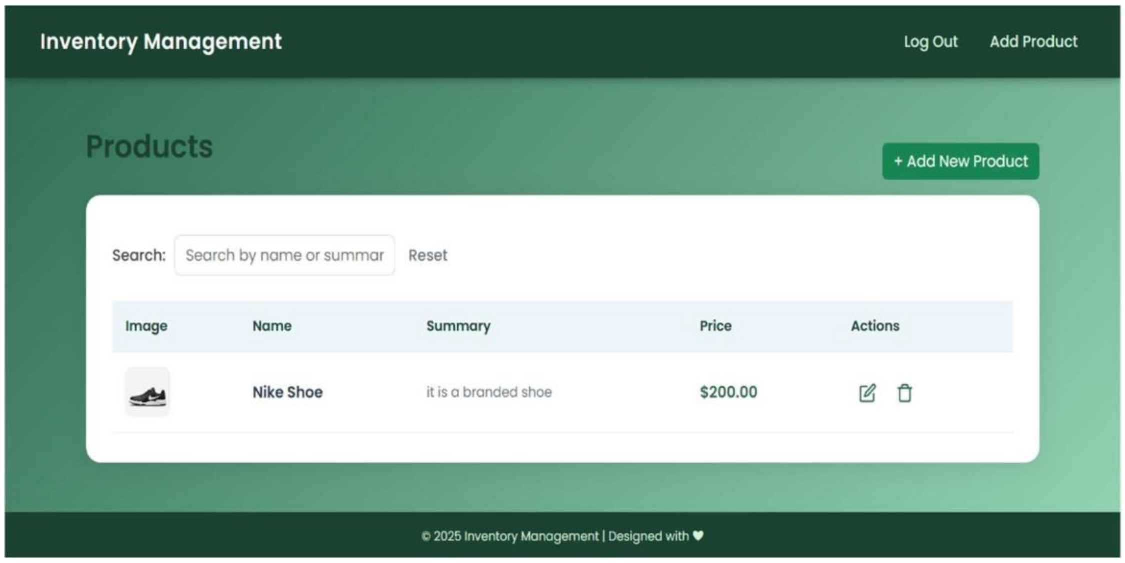
Category: Select a category

Sub Category: Select a sub category

Price: $ 175.00

Description: Enter product description

Fig 7.6



Inventory Management Log Out Add Product

Products

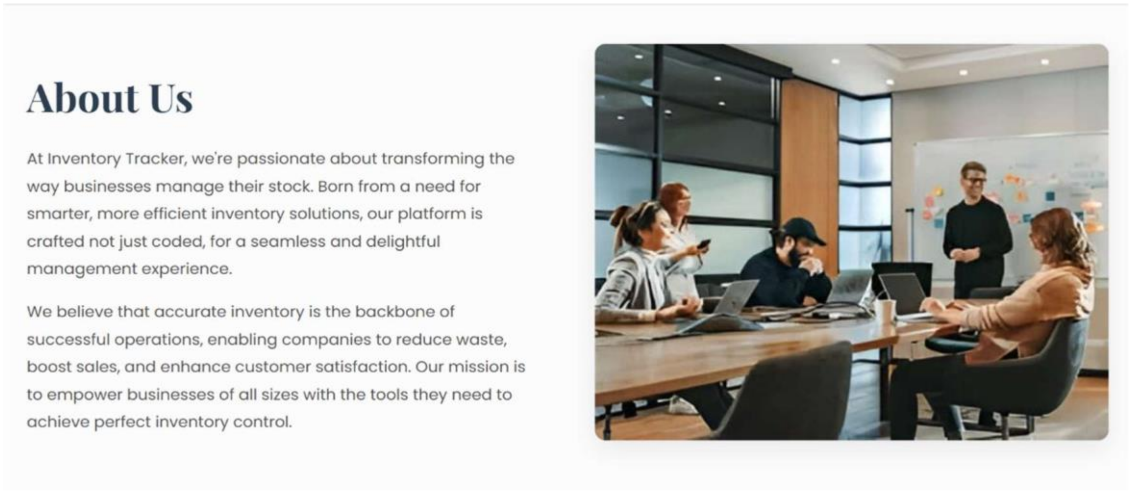
+ Add New Product

Search: Search by name or summar Reset

Image	Name	Summary	Price	Actions
	Nike Shoe	it is a branded shoe	$200.00	

© 2025 Inventory Management | Designed with ♥

Fig 7.7

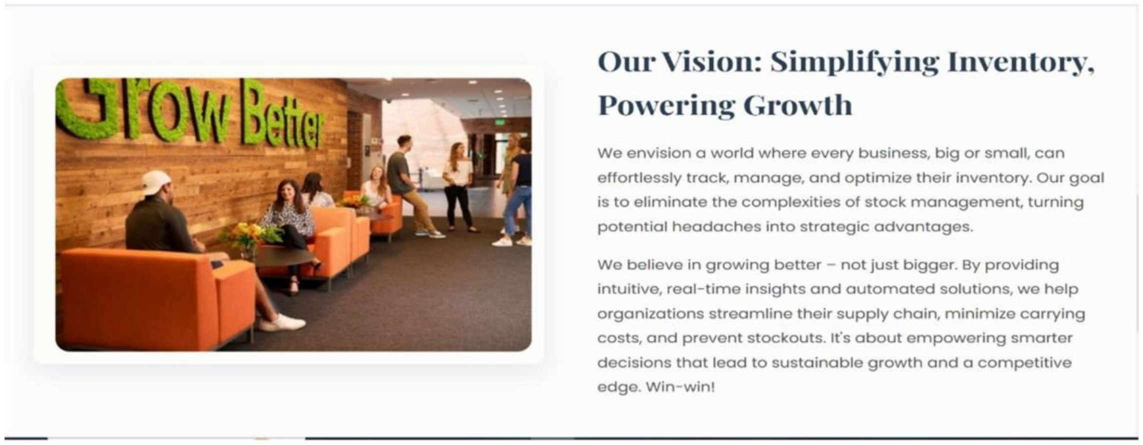


About Us

At Inventory Tracker, we're passionate about transforming the way businesses manage their stock. Born from a need for smarter, more efficient inventory solutions, our platform is crafted not just coded, for a seamless and delightful management experience.

We believe that accurate inventory is the backbone of successful operations, enabling companies to reduce waste, boost sales, and enhance customer satisfaction. Our mission is to empower businesses of all sizes with the tools they need to achieve perfect inventory control.

Fig 7.8



Our Vision: Simplifying Inventory, Powering Growth

We envision a world where every business, big or small, can effortlessly track, manage, and optimize their inventory. Our goal is to eliminate the complexities of stock management, turning potential headaches into strategic advantages.

We believe in growing better – not just bigger. By providing intuitive, real-time insights and automated solutions, we help organizations streamline their supply chain, minimize carrying costs, and prevent stockouts. It's about empowering smarter decisions that lead to sustainable growth and a competitive edge. Win-win!

CHAPTER 8: CONCLUSION

❖ **Key Learning Outcomes:**

- The project successfully demonstrated the complete lifecycle of a web application: designing a database schema, building an interactive user interface, and implementing the data persistence layer. The key takeaway is the practical understanding of Full-Stack Integration, specifically how client-side events (JavaScript) trigger server-side logic (PHP) to manipulate data stored in a relational database (MySQL).
- Project Significance
- This system serves as a foundational exercise, proving competence in the CRUD paradigm the functional core of nearly all database-driven applications. It shows mastery over AJAX/JSON data handling and the essential requirement of separating client-side presentation from secure server-side processing.
- Scope for Future Enhancement
- The project is ready for advanced development in three major areas:
- Security: Implement robust server-side input sanitization and transition from simple alerts to secure Session/JWT-based authentication.
- Scalability: Separate the product data by introducing a Category entity to improve filtering and reporting.
- Advanced Features: Develop reporting tools (e.g., total value calculation, low-stock alerts) and implement a sophisticated Stock In/Stock Out tracking system to manage inventory levels beyond a simple quantity field.

CHAPTER 9: REFERENCES

The following websites, books, and documentation were utilized during the development of the Inventory Management System, specifically for understanding and implementing the full-stack architecture and CRUD operations:

Category	Reference	Purpose
Official Documentation	MDN Documentation (Mozilla Developer Network)	Comprehensive guides on JavaScript syntax, especially the fetch API for asynchronous communication and DOM manipulation.
Official Documentation	https://www.mysql.com	Reference for SQL syntax, database design principles, and structuring the Users and Products tables.
Official Documentation	https://getbootstrap.com	Documentation for the Bootstrap 5 framework, used for responsive styling, components (like the navigation bar and modals in sim.html), and general layout.
Tutorials & Education	https://www.w3schools.com	Quick reference for fundamental HTML structure, CSS properties, and basic PHP/MySQL data handling examples.
Framework & Language	PHP Manual (php.net)	Documentation for server-side logic, handling POST requests, managing Form Data (especially file uploads in Api/create-product-php), and interacting with the database layer