Inventory Management System

Project Report

Project report in partial fulfillment of the requirement of Innovative Project
In
Department of Computer Science and Engineering
Submitted By

GOPESH SHARMA
Department of Computer Science Engineering,
University Enrollment No.: 12023002001112 UEM, Kolkata
Section: B , Class Roll No.: 70

SOUBHADRA BHATTACHARYA
Dept. of Computer Science and Engineering,
University Enrollment No.: 12023002001182 UEM, Kolkata
Section: B , Class Roll No.: 71

ATRI BANERJEE
Dept. of Computer Science and Engineering,
University Enrollment No.: 12023002001058 UEM, Kolkata
Section: B , Class Roll No.: 72

UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA
University Area, Plot No. III – B/5, New Town, Action Area – III, Kolkata – 700 156

ACKNOWLEDGEMENT

We would like to take this opportunity to thank everyone whose cooperation and encouragement throughout the ongoing course of this project remains invaluable to us.

We are sincerely grateful to our guide Prof. _____ and Prof. _____ of the Department of Computer Science and Engineering , UEM, Kolkata, for their wisdom, guidance and inspiration that helped us to go through with this project and take it to where it stands now.

Last but not the least, we would like to extend our warm regards to our families and peers who have kept supporting us and always had faith in our work.

Soubhadra Bhattacharya
Atri Banerjee
Gopesh Sharma

TablTable of Contents

1. Introduction

*The **Inventory Management System** is a simple web-based application developed to manage inventory items. This system allows users to add, edit, and delete items in an inventory, as well as calculate the total cost and quantity of items. It is designed using **HTML**, **CSS**, and **JavaScript** for the frontend.*

1.1 Purpose

This document specifies the functional and nonfunctional requirements for the Inventory Management System. It highlights the core functionalities and describes the user experience.

1.2 Document Conventions

- *The document uses standard naming conventions for variables and functions.*
- *Key features are listed in a structured format.*

1.3 Intended Audience and Reading Suggestions

This report is intended for:

- ***Developers**: To understand the scope and design of the project.*
- ***Teachers/Reviewers**: For evaluation purposes.*
- ***End Users**: To get an overview of the application's capabilities.*

1.4 Project Scope

The Inventory Management System enables users to efficiently manage inventory items. It supports adding, editing, and deleting items, and provides a summary of the total cost and quantity. This application is ideal for small businesses or personal inventory tracking.

1.5 References

- *HTML5, CSS3, and JavaScript Documentation*
- *W3Schools*

2. Overall Description

2.1 Product Perspective

The Inventory Management System is a standalone application designed for client-side operation. It focuses on basic inventory operations without requiring backend support.

2.2 Product Features

- *Add items with name, quantity, and price.*
- *Edit or delete items.*
- *Display total cost and total quantity dynamically.*

2.3 User Classes and Characteristics

- ***General Users**: Small business owners or individuals who want a simple inventory tracking system.*

2.4 Operating Environment

Runs on modern web browsers like Chrome, Firefox, Edge, and Safari.

2.5 Design and Implementation Constraints

- *Uses only HTML, CSS, and JavaScript.*
- *Does not include backend functionalities.*

2.6 User Documentation

An online guide is provided with the system to assist users in operating the application.

2.7 Assumptions and Dependencies

- *Assumes that users are familiar with basic web applications.*
- *The system depends on browser compatibility.*

3. System Features

3.1 Add Items

- Users can add new items to the inventory with a name, quantity, and price.

3.2 Edit Items

- Users can modify existing item details such as name, quantity, and price.

3.3 Delete Items

- Users can remove items from the inventory.

3.4 Total Cost and Goods Calculation

- The system calculates the total cost and quantity of all items in the inventory dynamically.

4. External Interface Requirements

4.1 User Interfaces

- Input fields for item details.
- A table to display inventory data.

4.2 Hardware Interfaces

- Requires a device with a web browser.

4.3 Software Interfaces

- Works with browsers supporting HTML5, CSS3, and JavaScript.

4.4 Communications Interfaces

- Not applicable; the system is client-side only.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The system responds to user inputs within 1 second.

5.2 Safety Requirements

- Minimal safety risks as the application handles no sensitive data.

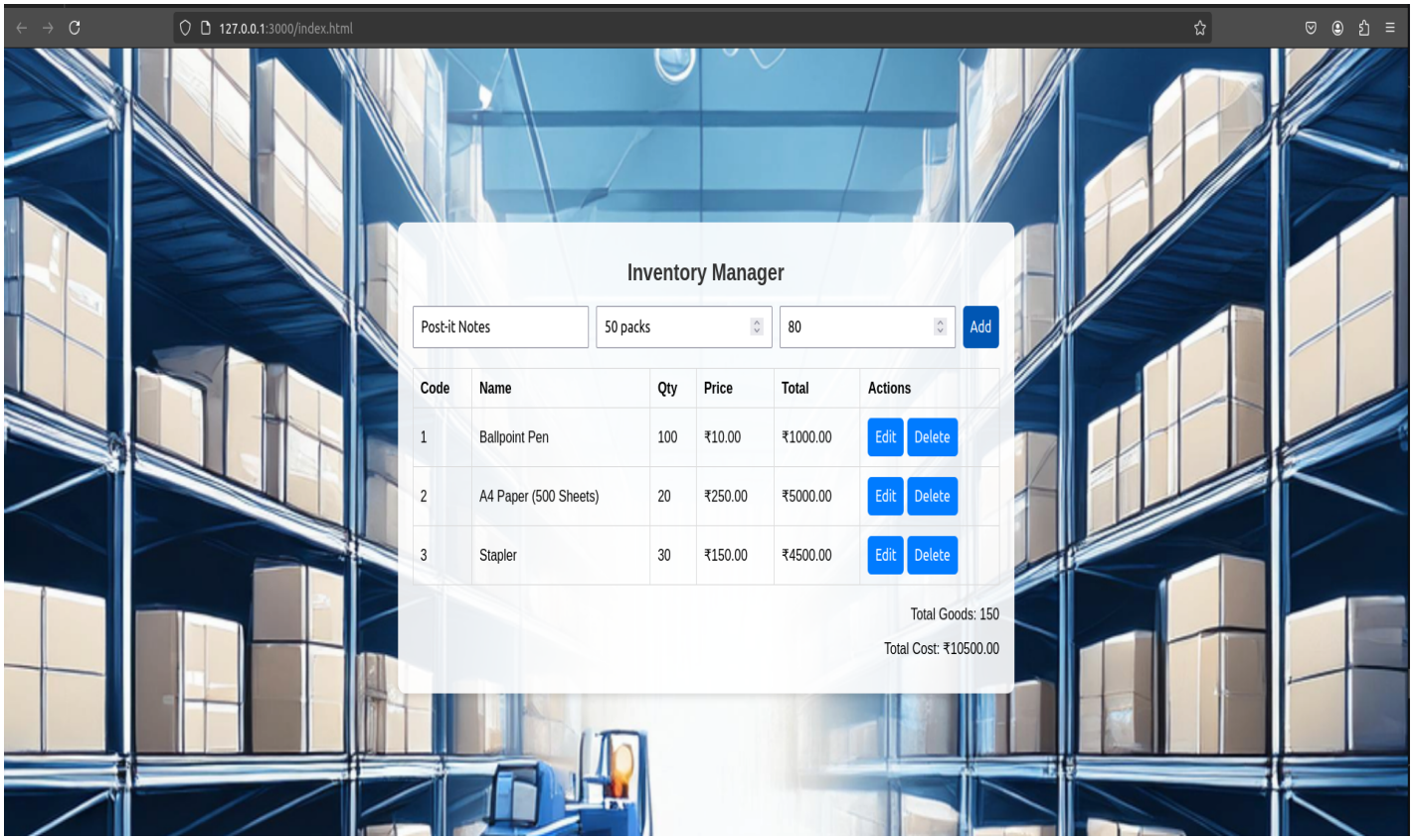
5.3 Security Requirements

- No specific security measures required due to lack of backend storage.

5.4 Software Quality Attributes

- **Usability**: Simple and intuitive for users.
- **Reliability**: Handles typical user interactions without crashes.

6.Screen shots of the developed system



- **HTML CODE SNIPPET**

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Inventory Manager</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Inventory Manager</h1>
    <form id="inventoryForm">
      <input type="text" id="itemName" placeholder="Enter Item Name" required>
      <input type="number" id="itemQuantity" placeholder="Enter Quantity" min="1" requir
      <input type="number" id="itemPrice" placeholder="Enter Price (₹)" min="0.01" step=
      <button type="button" id="addItemBtn">Add</button>
      <button type="button" id="updateItemBtn" style="display: none;">Update</button>
    </form>

    <table id="inventoryTable">
      <thead>
        <tr>
          <th>Code</th>
          <th>Name</th>
          <th>Qty</th>
          <th>Price</th>
          <th>Total</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        <!-- Items will show here -->
      </tbody>
    </table>

    <div class="summary">
      <p>Total Goods: <span id="totalGoods">0</span></p>
      <p>Total Cost: ₹<span id="totalCost">0.00</span></p>
    </div>
  </div>

  <script src="script.js"></script>
</body>
</html>
```

- **CSS CODE SNIPPETS**

```css
body {
    font-family: Arial, sans-serif;
    background: url('BACKGROUND-IMAGE.png') no-repeat center center fixed;
    background-size: cover;
    margin: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
  }

  .container {
    max-width: 800px;
    width: 100%;
    background: rgba(255, 255, 255, 0.9);
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
    text-align: center;
  }

  h1 {
    font-size: 24px;
    color: #333;
    margin-bottom: 20px;
  }

  form {
    display: flex;
    flex-wrap: wrap;
    gap: 10px;
    margin-bottom: 20px;
  }

  form input, button {
    padding: 10px;
    font-size: 16px;
  }
```

```css
button {
   background-color: #007bff;
   color: white;
   border: none;
   border-radius: 5px;
   cursor: pointer;
 }

 button:hover {
   background-color: #0056b3;
 }

 table {
   width: 100%;
   border-collapse: collapse;
   margin-bottom: 20px;
 }

 table th, table td {
   padding: 10px;
   border: 1px solid #ddd;
   text-align: left;
 }

 .summary {
   font-size: 16px;
   text-align: right;
   margin-top: 20px;
 }


 button {
   background-color: #007bff;
   color: white;
   border: none;
   border-radius: 5px;
   cursor: pointer;
 }

 button:hover {
   background-color: #0056b3;
 }
```

- **JAVASCRIPT CODE SNIPPETS**

```
// Wait until the page loads
document.addEventListener('DOMContentLoaded', function () {
  let itemId = 1;
  let totalGoods = 0;
  let totalCost = 0;

  // Buttons and inputs
  const addItemBtn = document.getElementById('addItemBtn');
  const updateItemBtn = document.getElementById('updateItemBtn');
  const inventoryTable = document.getElementById('inventoryTable').querySelector('tbody'
  let currentEditRow = null;

  addItemBtn.addEventListener('click', function () {
    let itemName = document.getElementById('itemName').value;
    let itemQuantity = parseInt(document.getElementById('itemQuantity').value);
    let itemPrice = parseFloat(document.getElementById('itemPrice').value);

    if (itemName && itemQuantity > 0 && itemPrice > 0) {
      let itemTotal = (itemQuantity * itemPrice).toFixed(2);
      let row = document.createElement('tr');

      row.setAttribute('data-id', itemId);
      row.innerHTML = `
        <td>${itemId}</td>
        <td>${itemName}</td>
        <td>${itemQuantity}</td>
        <td>₹${itemPrice.toFixed(2)}</td>
        <td>₹${itemTotal}</td>
        <td>
          <button class="editBtn">Edit</button>
          <button class="deleteBtn">Delete</button>
        </td>
      `;

      inventoryTable.appendChild(row);

      totalGoods += itemQuantity;
      totalCost += parseFloat(itemTotal);

      document.getElementById('totalGoods').textContent = totalGoods;
      document.getElementById('totalCost').textContent = totalCost.toFixed(2);

      itemId++; // Next item ID
      clearForm();
```

```
inventoryTable.addEventListener('click', function (e) {
  if (e.target.classList.contains('editBtn')) {
    let row = e.target.closest('tr');
    currentEditRow = row;

    document.getElementById('itemName').value = row.children[1].textContent;
    document.getElementById('itemQuantity').value = row.children[2].textContent;
    document.getElementById('itemPrice').value = row.children[3].textContent.slice(1);

    addItemBtn.style.display = 'none';
    updateItemBtn.style.display = 'inline-block';
  }

  if (e.target.classList.contains('deleteBtn')) {
    let row = e.target.closest('tr');
    let quantity = parseInt(row.children[2].textContent);
    let total = parseFloat(row.children[4].textContent.slice(1));

    totalGoods -= quantity;
    totalCost -= total;

    document.getElementById('totalGoods').textContent = totalGoods;
    document.getElementById('totalCost').textContent = totalCost.toFixed(2);

    row.remove();
  }
});

updateItemBtn.addEventListener('click', function () {
  if (currentEditRow) {
    let newName = document.getElementById('itemName').value;
    let newQty = parseInt(document.getElementById('itemQuantity').value);
    let newPrice = parseFloat(document.getElementById('itemPrice').value);

    if (newName && newQty > 0 && newPrice > 0) {
      let oldQty = parseInt(currentEditRow.children[2].textContent);
      let oldTotal = parseFloat(currentEditRow.children[4].textContent.slice(1));

      totalGoods = totalGoods - oldQty + newQty;
      totalCost = totalCost - oldTotal + (newQty * newPrice);
```

7. Future Scope

The Inventory Management System has significant potential for future enhancements, both in functionality and usability. The following features and improvements can be implemented to make the system more robust and versatile:

5.0.1.1 7.1 Add Backend Support for Persistent Storage

- *Currently, the system is client-side and does not save inventory data permanently. Adding backend support will allow persistent storage of items in a database such as MySQL, MongoDB, or Firebase.*
- *This enhancement will enable the system to store data across sessions, allowing users to access their inventory from any device.*
- *Backend integration will also support data synchronization, backups, and security measures for large-scale inventory systems.*

7.2 Provide User Authentication

- *Implementing a user authentication system will allow multiple users to access the application securely.*
- *Features like user registration, login, and password recovery can be added to ensure only authorized personnel can manage the inventory.*
- *This will also enable role-based access control, where certain users (e.g., managers) can edit or delete items, while others (e.g., staff) can only view the inventory.*

7.3 Enable Filtering and Sorting of Inventory Items

- *Advanced filtering options can be introduced, such as filtering by item category, quantity range, or price range.*
- *Sorting capabilities will allow users to organize items alphabetically, by price, or by quantity for easier management.*
- *These features will improve usability, especially for inventories with a large number of items.*

7.4 Integration with Reports and Analytics

- *Adding reporting features will enable users to generate detailed reports on inventory usage, sales trends, and restocking needs.*
- *Visual analytics like charts and graphs can help users track inventory performance and make informed decisions.*
- *Export options (e.g., CSV or PDF) can be added for sharing inventory reports.*

7.5 Mobile App Integration

- A dedicated mobile app can be developed to complement the web application.
- The app would allow users to manage their inventory on-the-go, with offline capabilities for updating data and syncing when back online.

- Push notifications for low stock alerts or other critical updates could further enhance functionality.

7.6 Barcode and QR Code Integration

- Adding barcode or QR code scanning will streamline item management by automating item entry and lookup.
- Users could scan codes to instantly retrieve or update item information, reducing manual errors and improving efficiency.

7.7 Multilingual and Regional Support

- Adding support for multiple languages and currencies will make the system accessible to a broader audience.
- Users will be able to view and manage their inventory in their preferred language, improving usability in diverse markets.

7.8 AI-Powered Inventory Insights

- Artificial Intelligence (AI) could be used to predict stock requirements based on historical data and trends.
- AI algorithms could also suggest optimal restocking times and quantities to reduce overstocking or stockouts.

7.9 Cloud Integration

- Cloud-based deployment will enable users to access their inventory from anywhere, on any device.
- Cloud storage solutions like AWS, Google Cloud, or Azure can ensure scalability, reliability, and security for large inventories.

7.10 Enhanced Security Measures

- To secure sensitive inventory data, advanced security features like two-factor authentication (2FA), data encryption, and activity logging can be introduced.
- Regular security audits and compliance with standards like GDPR can ensure the system remains secure and trustworthy.

8. Conclusion

The Inventory Management System is a simple yet effective tool for tracking and managing inventory. It offers core functionalities like adding, editing, and deleting items while dynamically calculating totals. Future enhancements can improve its usability and scalability.

9. Bibliography

- https://github.com/inventree/InvenTree
- W3Schools
- FreeCodeCamp
- gemini,codeum,blackbox