**Course:** MAC172

**Final Project**: Inventory Web Application

**Student Name:** Meiyan Ye

**Professor:** Praveenkumar Khethavath

**Date:** February 19, 2025

1. **Introduction**

A diagram of a stock market

Description automatically generatedA web-based application designed to help businesses track and manage their inventory efficiently.

Features include adding items, updating stock levels, generating reports, and visualizing inventory data.

Built using HTML, CSS, and JavaScript, with data persistence using *localStorage*.

*LocalStorage*is a web storage feature that allows websites to store data in the user's browser persistently, without expiration. This data is saved in key-value pairs and is accessible across page reloads or browser restarts. The data stored in localStorage is accessible only by the same domain that saved it, and it persists until it is explicitly deleted by the user or the website.

1. **Key Features of the System**

* Add Items:

Users can add new items to the inventory with details like name, stock quantity, target quantity, and an image.

* Edit Items:

Users can update target quantities, sold quantities, and add quantities dynamically.

* Real-Time Calculations:

The system calculates the "Need Quantity" in real-time based on stock and target levels.

* Generate Reports:

Users can generate a report summarizing inventory updates, including timestamps.

* Image Support:

Each item can have an associated image, either uploaded by the user or loaded from a URL.

* Data Persistence:

Inventory data is saved in the browser's localStorage, ensuring it persists even after the page is refreshed or closed.

**3.Technical Overview:**

* Built with HTML for structure, CSS for styling, and JavaScript or interactivity.
* Image Handling:

Images can be uploaded as files. Uploaded images are converted to Base64 for storage.

The image file is selected via an <input type="file"> element. Then, it is read and converted into a Blob using JavaScript.

A Blob represents the raw binary data of the image. Once we have the Blob, we can convert it to Base64.

Base64 encoding turns the binary data of the image into a text string that can be safely stored or transmitted over text-based systems.

* Dynamic Updates:

The system dynamically updates stock levels, need quantities, and visual indicators (e.g., red for low stock).

**4.User Interface:Login to inventory login screen

Description automatically generated**

**A screenshot of a inventory management

Description automatically generated**

1. Inventory Form:

- Users can input item details and upload images.

2. Inventory Table:

- Displays all items with editable fields for target, sold, and add quantities.

3. Report Section:

- Shows a summary of inventory updates with timestamps.

4. Visual Indicators:

- Highlights low stock levels in red and overstocked items in orange.

---

**5.Code Highlights:**

- Adding Items:

function addItemToInventory(name, stockQty, targetQty, imageFile, imageUrl) {

// Creates a new row in the table and populates it with item details

}

- Saving Data:

function saveInventory() {

// Saves inventory data to localStorage

}

- Loading Data:

function loadInventory() {

// Loads inventory data from localStorage

}

- Generating Reports:

document.getElementById('report').addEventListener('click', function() {

// Updates quantities and displays a report

});

- Form submission handler

document.getElementById('inventoryForm').addEventListener('submit', function(event){}

-Logout button functionality

document.getElementById('logout-button').addEventListener('click', function(){}

-Load the stored inventory list when login to the page

document.addEventListener('DOMContentLoaded', loadInventory);

**6.Challenges:**

1. Handling image uploads and storage.

- Solution: Used FileReader to convert images to Base64 and stored them in localStorage.

FileReader is a JavaScript API that allows web applications to read the contents of files stored on a user's device, such as images, text files, or other types of data.

2. Ensuring data persistence across page reloads.

- Solution: Implemented localStorage to save and load inventory data.

3. Real-time updates for stock and need quantities.

- Solution: Used JavaScript to dynamically recalculate values and update the UI.

**7.Future Enhancements**

1. Backend Integration:

- Replace *localStorage* with a database for scalability.

2. User Authentication:

- Add login and registration features to support multiple users.

3. Advanced Reporting:

- Include charts and graphs for better visualization of inventory trends.

4. Mobile App:

- Develop a mobile version of the system using frameworks like React Native.

**8. Conclusion**

- The **Inventory Management System** is a powerful tool for businesses to track and manage their inventory efficiently.

- It demonstrates the use of modern web technologies to create a user-friendly and functional application.

- Future enhancements will make it even more robust and scalable.

CITE WORKS

Mozilla Contributors. (n.d.). *FileReader* [JavaScript MDN Documentation]. Mozilla Developer Network. Retrieved February 19, 2025, from <https://developer.mozilla.org/en-US/docs/Web/API/FileReader>

Mozilla Contributors. (n.d.). *Blob* [JavaScript MDN Documentation]. Mozilla Developer Network. Retrieved February 19, 2025, from <https://developer.mozilla.org/en-US/docs/Web/API/Blob>

Mozilla Contributors. (n.d.). *Base64* [JavaScript MDN Documentation]. Mozilla Developer Network. Retrieved February 19, 2025, from <https://developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/btoa>