Fresh Grocery List

A PROJECT REPORT

Submitted by

Mohamed Marzuq Tharif S

(312323205136)

BACHELOR OF TECHNOLOGY

In

DEPARTMENT OF INFORMATION TECHNOLOGY

St.JOSEPH'S COLLEGE OF ENGINEERING

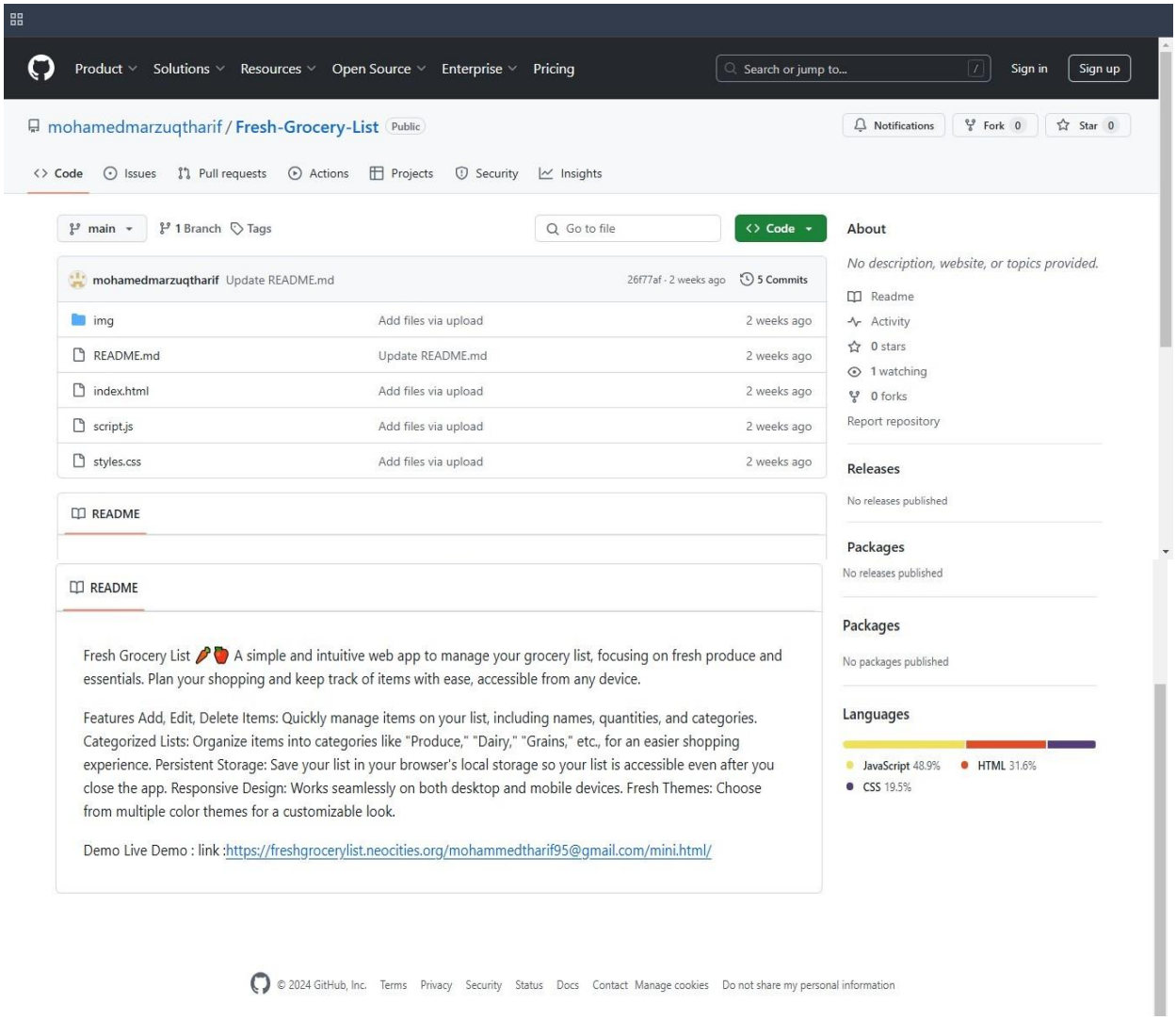
(An Autonomous Institution)

St.JOSEPH'S GROUP OF INSTITUTIONS

OMR,CHENNAI-119

Fresh Grocery List

Github :



Abstract:

The Grocery List Application is a web-based tool designed to simplify grocery shopping by allowing users to create, manage, and calculate grocery lists efficiently. Built using HTML, CSS, and JavaScript, this application organizes items into categorized sections such as Vegetables, Fruits, Drinks, Snacks, and Grocery Items, making it easy for users to navigate their list. It enables users to input specific product details, including name, quantity, unit, and price per unit, with an automatic calculation of the total cost, inclusive of a 2% GST.

Data persistence through local storage ensures that the grocery list is saved for future sessions. An interactive chart, powered by Chart.js, visually represents item quantities, providing a clear overview of shopping needs. Featuring an intuitive user interface with animations and relevant visuals, the Grocery List Application enhances the shopping experience and serves as a practical showcase of essential web development skills.

Introduction:

Grocery shopping is an essential, recurring task in daily life, yet it often becomes cumbersome and disorganized without the proper tools. Traditional methods, like handwritten lists, can lead to confusion, forgotten items, and an inefficient shopping experience. To address these challenges, the Grocery List Application has been designed as a web-based solution that simplifies and enhances grocery shopping. Built with HTML for structure, CSS for styling, and JavaScript for interactivity, this application offers key functionalities, including categorized item organization, product detail entry, total cost calculation with GST, and data persistence through local storage for easy access across sessions. An interactive chart, created with Chart.js, visually represents item quantities, providing users with a quick overview of their grocery needs. By improving shopping efficiency and reducing stress, this project not only aims to empower users in managing their groceries effectively but also demonstrates core web development skills in a practical, everyday application.

Objective:

The objective of the Grocery List Application is to provide a seamless, efficient, and user-friendly solution for managing grocery shopping. This web-based tool is designed to eliminate the common issues of traditional list-making by offering a structured, interactive, and categorized approach to grocery management. With features that allow users to add detailed product information, automatically calculate total costs including GST, and securely save data for future sessions, the application aims to streamline the entire shopping experience.

By integrating an interactive chart for visual representation of item quantities, it enables users to gain insights into their shopping needs at a glance. Ultimately, the Grocery List Application aspires to improve shopping efficiency, reduce the likelihood of overlooked items, and empower users to make well-informed, organized purchasing decisions.

Implementation:

The implementation of the Grocery List Application begins with setting up a well-organized file structure, comprising three main files: `index.html` for the core HTML layout, `styles.css` for visual styling, and `script.js` to manage functionality. The HTML structure creates an intuitive interface with input fields for product details (such as name, quantity, unit, and price), a dropdown menu for selecting item categories, and dedicated sections for each grocery category like Vegetables, Fruits, Drinks, Snacks, and Grocery Items. Additionally, display elements include a total price summary and an area for an interactive chart to provide users with a visual representation of their grocery items. Styling with CSS enhances user experience by applying a fresh green background, appropriate padding and margins for readability, and smooth animations for actions like button clicks and item additions. JavaScript handles core functionality, enabling users to add items to the list, validate inputs, and dynamically organize items under relevant categories. It also includes an automated total price calculation that updates in real-time, with a 2% GST applied, and data persistence via `localStorage` to ensure grocery lists are saved across sessions. An interactive bar chart, created with Chart.js, visually represents item quantities and updates dynamically as users add or remove items. To complete the application, each list item features a remove button to facilitate easy updates to the list, while also updating the total price and storage. Finally, thorough testing ensures all components function as expected, verifying accurate input validation, correct total price calculation, and reliable data saving and loading. The application can then be deployed on a platform such as GitHub Pages for easy access and sharing.

Technologies Used:

The Grocery List Application utilizes the following technologies:

1. HTML:Provides the structural foundation of the application, defining the layout and elements such as input fields, dropdown menus, category sections, and display areas for total price and charts.

2.CSS: Enhances the user interface with styling, including color schemes, padding, margins, and animations for a visually engaging and easy-to-navigate experience.

3. JavaScript:Implements dynamic functionality, handling tasks like adding items, validating inputs, calculating totals with GST, managing data in local storage, and updating the interactive chart.

4. Chart.js: A JavaScript library for creating interactive charts, used here to visualize item quantities in a bar chart format that updates as users add or remove items from the list.

5. Local Storage (Web Storage API):Enables data persistence by storing the grocery list in the browser, ensuring that the list remains accessible across sessions.

These technologies work together to create an organized, user-friendly application that simplifies grocery management and provides a practical demonstration of web development skills.

Project Description:

The Grocery List Application is a web-based tool designed to streamline the grocery shopping process by providing users with a structured, efficient way to manage their lists. Built using HTML, CSS, and JavaScript, this application organizes grocery items into categories such as Vegetables, Fruits, Drinks, Snacks, and Grocery Items for easy navigation. Users can add detailed product information, including name, quantity, unit, and price per unit, and the application automatically calculates the total cost, including a 2% GST. To ensure persistence, the grocery list is saved in the browser's local storage, allowing users to retrieve their list even after closing the application.

The interface includes an interactive bar chart created with Chart.js, visually displaying item quantities to give users a clear overview of their shopping needs. Designed with an intuitive layout, a green background to evoke freshness, and smooth animations, the application enhances the shopping experience, making it more organized and less stressful. This project demonstrates essential web development skills and provides a practical, everyday solution for efficient grocery management.

Source code:

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Grocery List</title>
    <link rel="stylesheet" href="styles.css">
    <link rel="icon" href="C:\Users\Hp\Desktop\mini.html\su.jpeg"
type="icon/png">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
    <div class="container">
        <img src="C:\Users\Hp\Desktop\mini.html\Grey Simple Business Corporate
LinkedIn Banner.png" alt="Supermarket" class="supermarket-image">
        <h1>Grocery List</h1>
        <input type="text" id="product-input" placeholder="Enter product name">

        <select id="category-select">
            <option value="vegetables">Vegetables</option>
            <option value="fruits">Fruits</option>
            <option value="drinks">Drinks</option>
            <option value="snacks">Snacks</option>
            <option value="groceries">Grocery Items</option>
        </select>

        <input type="number" id="quantity-input" placeholder="Quantity" min="0"
step="any">
        <select id="unit-select">
            <option value="kg">Kilograms</option>
            <option value="g">Grams</option>
            <option value="l">Liters</option>

           <option value="pcs">Pieces</option>
```

```
</select>

        <input type="number" id="price-input" placeholder="Price per unit"
min="0" step="any">

        <button id="add-button">Add to List</button>

        <h2>Grocery Items</h2>

        <!-- Separate sections for each category -->
        <div id="vegetables-section">
            <h3>Vegetables</h3>
            <ul id="vegetables-list"></ul>
        </div>

        <div id="fruits-section">
            <h3>Fruits</h3>
            <ul id="fruits-list"></ul>
        </div>

        <div id="drinks-section">
            <h3>Drinks</h3>
            <ul id="drinks-list"></ul>
        </div>

        <div id="snacks-section">
            <h3>Snacks</h3>
            <ul id="snacks-list"></ul>
        </div>

        <div id="groceries-section">
            <h3>Grocery Items</h3>
            <ul id="groceries-list"></ul>
        </div>

        <h3 id="total-price">Total Price: $0.00</h3>

        <canvas id="grocery-chart" width="400" height="200"></canvas>

        <center>
            <a href="https://www.bigbasket.com/">
                <img src="C:\Users\Hp\Desktop\mini.html\images.png"
alt="Shopping Trolley" class="trolley-image">
                <button>BUY</button>
            </a>
        </center>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

Styles.css:

```css
body {
    font-family: Arial, sans-serif;
    background-color:#0B3D3B ;
    margin: 0;
    padding: 20px;
}

.container {
    max-width: 600px;
    margin: auto;
    background:  #c8e6c9;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
    animation: fadeIn 0.5s ease;
}

.supermarket-image {
    width: 100%;
    border-radius: 8px;
    margin-bottom: 20px;
}

.trolley-image {
    width: 100%;
    border-radius: 8px;
    margin-top: 20px;
    width: 40px;
}

h1, h2 {
    text-align: center;
}

input, select, button {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border-radius: 4px;
    border: 1px solid #ccc;
    transition: border-color 0.3s;
}

input:focus, select:focus, button:hover {
    border-color: #bd6c0f;
}

ul {
    list-style-type: none;
}
```

```css
li {
    padding: 10px;
    border-bottom: 1px solid #ddd;
    transition: background-color 0.3s;
}

li:hover {
    background-color: #949292;
}

canvas {
    margin: 20px 0;
}

@keyframes fadeIn {
    from {
        opacity: 0;
        transform: translateY(-20px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}
```

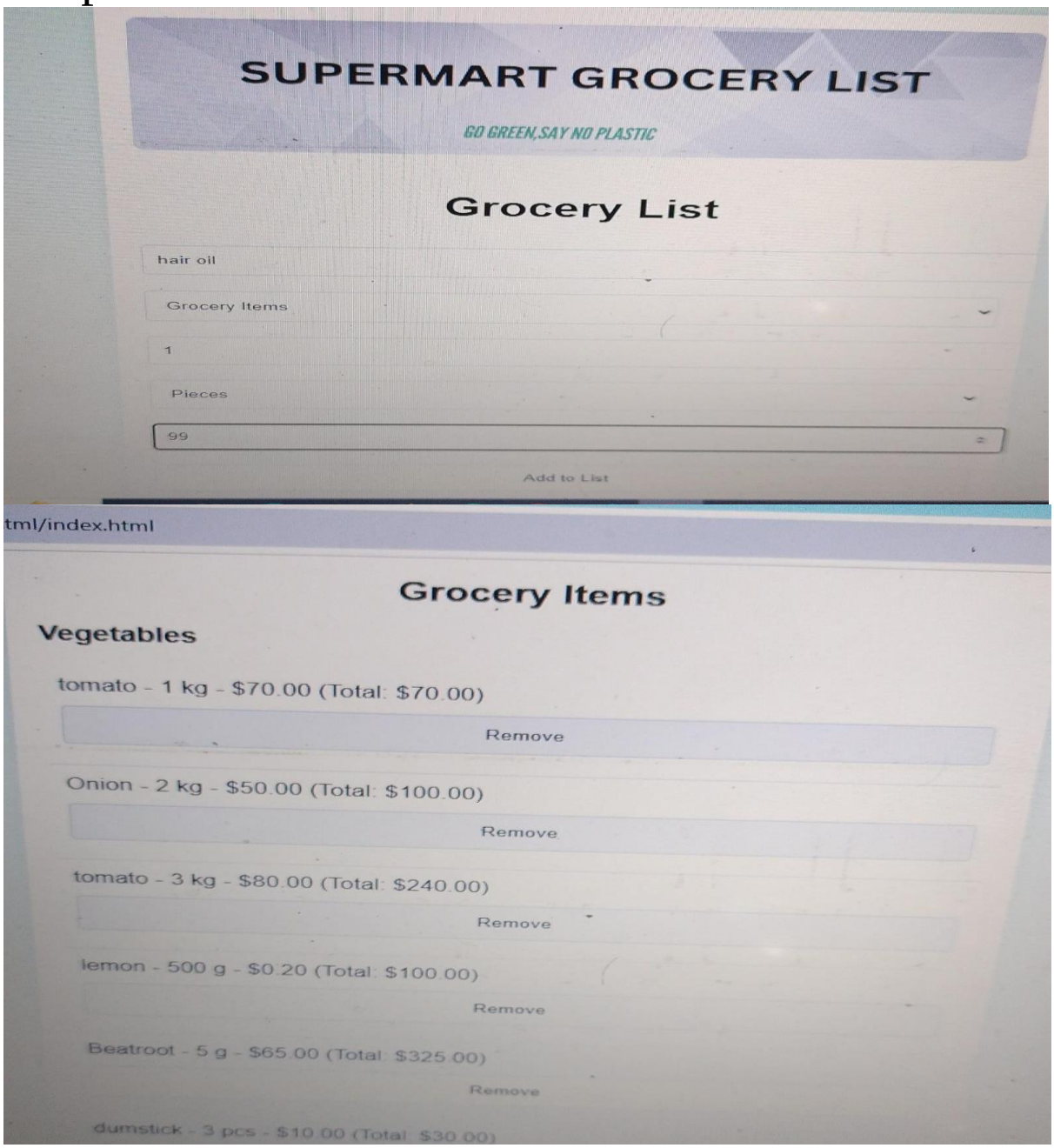
Script.js:

```javascript
let totalPrice = 0;
let chart;
const categories = ['vegetables', 'fruits', 'drinks', 'snacks', 'groceries'];


document.addEventListener('DOMContentLoaded', loadGroceryList);

document.getElementById('add-button').addEventListener('click', function() {
    const productInput = document.getElementById('product-input');
    const categorySelect = document.getElementById('category-select');
    const quantityInput = document.getElementById('quantity-input');
    const unitSelect = document.getElementById('unit-select');
    const priceInput = document.getElementById('price-input');

    const productName = productInput.value.trim();
    const category = categorySelect.value;
    const quantity = parseFloat(quantityInput.value.trim());
    const unit = unitSelect.value;
    const price = parseFloat(priceInput.value.trim());

    if (productName && !isNaN(quantity) && !isNaN(price)) {
        addGroceryItem(productName, category, quantity, unit, price);
        saveGroceryList();
        clearInputs();
    }

else {
```

```
        alert('Please fill in all fields correctly.');
    }
});


function addGroceryItem(productName, category, quantity, unit, price) {
    const groceryList = document.getElementById(`${category}-list`);
    const listItem = document.createElement('li');
    const itemTotalPrice = quantity * price;


    const removeButton = document.createElement('button');
    removeButton.textContent = 'Remove';
    removeButton.addEventListener('click', function() {
        groceryList.removeChild(listItem);
        totalPrice -= itemTotalPrice;
        updateTotalPrice();
        saveGroceryList();
    });

    listItem.textContent = `${productName} - ${quantity} ${unit} -
$${price.toFixed(2)} (Total: $${itemTotalPrice.toFixed(2)})`;
    listItem.appendChild(removeButton);
    groceryList.appendChild(listItem);


    totalPrice += itemTotalPrice;
    updateTotalPrice();
}


function updateTotalPrice() {
    const gst = totalPrice * 0.02;
    document.getElementById('total-price').textContent = `Total Price:
$${(totalPrice + gst).toFixed(2)}`;
}


function clearInputs() {
    document.getElementById('product-input').value = '';
    document.getElementById('quantity-input').value = '';
    document.getElementById('price-input').value = '';
}


function createChart() {
    const ctx = document.getElementById('grocery-chart').getContext('2d');
    chart = new Chart(ctx, {
        type: 'bar',
        data: {
            labels: [],
            datasets: [{
                label: 'Quantity',
```

```
                data: [],
                backgroundColor: 'rgba(75, 192, 192, 0.2)',
                borderColor: 'rgba(75, 192, 192, 1)',
                borderWidth: 1
            }]
        },
        options: {
            scales: {
                y: {
                    beginAtZero: true
                }
            }
        }
    });
}
function loadGroceryList() {
    const groceryData = JSON.parse(localStorage.getItem('groceryList')) || {};
    for (const category of categories) {
        const items = groceryData[category] || [];
        items.forEach(item => {
            const { name, quantity, unit, price } = item;
            addGroceryItem(name, category, quantity, unit, price);
        });
    }
}

function saveGroceryList() {
    const groceryData = {};
    categories.forEach(category => {
        groceryData[category] =
Array.from(document.querySelectorAll(`#${category}-list li`)).map(item => {
            const parts = item.firstChild.textContent.split(' - ');
            const name = parts[0];
            const quantityUnitPrice = parts[1].split(' ');
            const quantity = parseFloat(quantityUnitPrice[0]);
            const unit = quantityUnitPrice[1];
            const price = parseFloat(parts[2].match(/\$([\d.]+)/)[1]);
            return { name, quantity, unit, price };
        });
    });
    localStorage.setItem('groceryList', JSON.stringify(groceryData));
}
```

Output:



System Design:

The system design of the Grocery List Application is built to provide a modular, responsive, and user-friendly experience. The front-end architecture is structured using HTML, which defines the key elements of the application, including input fields for product details (name, quantity, unit, price) and a dropdown menu for selecting item categories. Each grocery category, such as Vegetables, Fruits, and Drinks, is organized into separate sections for easy navigation. The display area includes the total price and an interactive chart, where the quantities of items are visualizedJavaScript Functionality:

The styling is done using CSS, ensuring a responsive layout that adapts to both mobile and desktop screens. The design features a fresh green background, padding, and font adjustments to enhance readability. Additionally, smooth animations are added for button clicks and item additions to make the user interface engaging and intuitive. JavaScript handles the dynamic functionality of the application, such as adding items to the list, calculating the total price with GST, and updating the interactive chart using Chart.js. Local storage is used to persist the grocery list across sessions, ensuring that users' data is saved and can be retrieved even after the application is closed or refreshed. This design ensures that the application is not only functional but also easy to use and visually appealing.

Key Features:

The Grocery List Application offers several key features that enhance the user experience and streamline the shopping process:

1. **Categorized Grocery List:**
   Items are organized into specific categories like Vegetables, Fruits, Drinks, Snacks, and Grocery Items, making it easy for users to navigate and manage their list effectively.
2. **Product Details Input:**
   Users can add detailed information for each product, including:
   - Product name
   - Quantity (in units such as kg, g, l, pcs)
   - Price per unit This ensures that the grocery list is comprehensive and accurate.
3. **Total Cost Calculation:**
   The application automatically calculates the total cost of all listed items, including a 2% GST, helping users track their budget and expenses in real-time.
4. **Data Persistence with Local Storage:**
   Users' grocery lists are saved in the browser’s local storage, allowing them to access and update their list across sessions, even after closing or refreshing the page.
5. **Interactive Chart Visualization:**
   Using Chart.js, the application displays an interactive bar chart that visualizes the quantities of items in the list, offering a clear overview of the shopping needs and helping users identify items with higher quantities at a glance.

6. **Remove Items Functionality:**
   Each item in the list includes a "remove" button, allowing users to easily delete items. The total price and local storage are automatically updated to reflect the changes.
7. **User-Friendly Interface:**
   The application features a clean and intuitive interface with a fresh green background, smooth animations, and clear input fields, ensuring a pleasant and efficient shopping experience for users.

Future Enhancements:

Future enhancements for the Grocery List Application could include a variety of features aimed at improving functionality, user experience, and overall performance:

1. **User Authentication and Account Management:**
   Adding user authentication would allow users to create accounts, saving their grocery lists and preferences across multiple devices. This would also enable personalized recommendations and shopping history.
2. **Barcode Scanning Integration:**
   Implementing a barcode scanning feature would allow users to easily add products to their list by scanning the barcode of items, automatically filling in product details like name, price, and category.
3. **Multiple List Management:**
   Enabling users to manage multiple grocery lists (e.g., for different stores or shopping trips) would provide more flexibility, allowing users to plan their shopping in a more organized manner.
4. **Price Comparison:**
   Incorporating a price comparison tool could help users compare prices for the same product across different stores, making it easier to choose the most cost-effective option.
5. **Mobile App Version:**
   Developing a mobile application for iOS and Android would expand the reach of the project and provide users with an on-the-go solution for managing grocery shopping.
6. **Recipe Suggestions and Ingredient Lists:**
   Users could input ingredients from recipes they plan to cook, and app could generate shopping list based on those ingredients. Additionally, it could suggest recipes based the items already user.

7. **Voice Integration:**
   Adding voice control functionality would allow users to add items to their list hands-free, improving accessibility and ease of use, especially when they are busy with other tasks.
8. **Smart Notifications:**
   Integrating notifications to remind users about items they often purchase or notify them of price drops on frequently bought items could enhance the application's utility.
9. **Sharing and Collaboration:**
   Allowing users to share their grocery lists with family members or roommates and enabling collaborative list editing would streamline group shopping efforts.

These future enhancements would not only increase the application's versatility and user engagement but also position it as a comprehensive solution for grocery management.

Conclusion:

In conclusion, the Grocery List Application is a practical, user-friendly tool that simplifies and enhances the grocery shopping experience. By allowing users to easily create, organize, and manage their grocery lists with features like categorized sections, real-time total cost calculations, and interactive visualizations, the application effectively addresses common challenges faced during shopping, such as disorganization and forgotten items. With data persistence through local storage and a responsive interface, the app ensures a seamless and efficient experience. The project not only demonstrates essential web development skills but also offers a valuable, everyday solution to streamline grocery management. Future enhancements, such as user accounts, barcode scanning, and mobile integration, promise to expand its capabilities, making it even more versatile and beneficial for users in the future.

Reference:

1. **HTML Documentation:**
   Mozilla Developer Network (MDN). "HTML." MDN Web Docs.
2. **CSS Documentation:**
   Mozilla Developer Network (MDN). "CSS." MDN Web Docs.

3. **JavaScript Documentation:**
   Mozilla Developer Network (MDN). "JavaScript." MDN Web Docs.
4. **Chart.js Documentation:**
   Chart.js. "Chart.js Documentation." Chart.js.
5. **LocalStorage Documentation:**
   Mozilla Developer Network (MDN). "Web Storage API." MDN Web Docs.
6. **CSS Animations:**
   W3Schools. "CSS Animations." W3Schools.
7. **GitHub Pages Deployment Guide:**
   GitHub. "GitHub Pages." GitHub Docs.

These references provide a foundation for understanding the key technologies and libraries used in developing the Grocery List Application. Adjust or add additional resources based on any other tools or libraries used during the development process.