Flipkart Clone

Submitted in partial fulfilment of the requirements

of the Mini-Project of Full Stack Java Programming for Second Year of

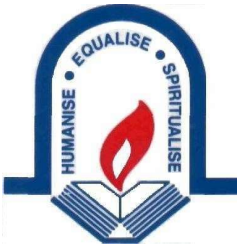
Bachelors of Engineering

by

Salmani Asad 45

Ansari Abdul Razzaq 02

Khan Sahil Rizwan 26



Department of Artificial Intelligence & Data Science Engineering

Rizvi College of Engineering

University of Mumbai

2025-2026

CERTIFICATE

This is to certify that the mini-project entitled "**Flipkart Clone**" is a Bonafide work of **Salmani Asad (45), Ansari Abdul Razzaq (02), Khan Sahil Rizwan (26)** submitted to the University of Mumbai in partial fulfilment of the requirement for the Mini-Project of Full Stack Java Programming for Second Year of the Bachelor of Engineering in "**Artificial Intelligence & Data Science**".

(Name and sign)
Guide/Prof.

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

Salmani Asad Anwarul (45)

(Signature)

Ansari Abdul Razzaq (02)

(Signature)

Khan Sahil Rizwan (26)

Date:

ABSTRACT

Flipkart Clone Frontend Development and Usability Analysis

This report details the development and subsequent functional analysis of a frontend prototype mimicking the core browsing and shopping experience of the **Flipkart** e-commerce platform. Built exclusively using **HTML5, CSS3, and Vanilla JavaScript**, the project demonstrates mastery of fundamental web technologies to create a fast, client-side, multi-page application.

Key implemented features include: **multi-page architecture** (index.html, smartphones.html, cart.html), **dynamic product rendering** across multiple categories (Mobiles, TVs, Fashion), and **stateful shopping cart management** using the browser's **localStorage API**. This implementation allows the cart item count, total price, and contents to **persist reliably** across browsing sessions and page navigations, simulating a real-world e-commerce journey. The system utilizes **Event Delegation** on the dedicated cart.html page to ensure efficient removal of items, regardless of how many items are dynamically rendered.

A focused debugging methodology was employed to resolve critical issues related to data integrity, specifically handling hidden whitespace characters in product names through the use of **.trim()** functionality. Functional testing confirmed the high integrity of the core shopping flow, achieving reliable success in adding, viewing, and removing items from the cart.

The analysis highlights the efficiency of the vanilla JavaScript approach for rapid prototyping and client-side performance. Future scope is defined to integrate quantity adjustment functionality on the cart page and implement a mock checkout flow, setting a clear path for iterative improvement and backend integration with technologies like **Java Spring Boot**.

Index

Sr. No	Title	Page No
1.	Introduction	
2.	System Analysis	
3.	System Design	
4.	Technologies Used	
5.	Implementation Details	
6.	Output Screenshots	
7.	Testing and Debugging	
8.	Results and Discussions	
9.	Conclusion	
10.	References	
11.	Appendix	
	Acknowledgement	

INTRODUCTION

This project is a **Flipkart Clone**, a simplified e-commerce web application that replicates the layout, navigation, and core interaction features of the Flipkart website. It is designed using **HTML, CSS, and JavaScript** to demonstrate how fundamental frontend technologies are combined to create a responsive, multi-page shopping interface.

Problem Statement / Objective

The main objective of this mini-project is to design and develop a functional frontend clone of the Flipkart e-commerce website that:

- Displays a list of products across multiple pages (Home, Category Listings).
- Allows seamless navigation between the product catalogue and the dedicated **Cart Page** (cart.html).
- Implements **stateful "Add to Cart" and "Remove" functionality** using **localStorage** to ensure the shopping cart persists across all pages and browser sessions.
- Demonstrates understanding of multi-page web architecture and effective JavaScript DOM manipulation.

Scope of the Project

In Scope:

1. Frontend UI design using HTML, CSS, and pure JavaScript.
2. Multi-page navigation and consistent header/footer components.
3. Product listing across pages (Smartphones, TVs, Clothing).
4. **Persistent Shopping Cart** using browser localStorage.
5. Calculation and display of total item count and payable price on the Cart page.

Out of Scope (for now):

1. Backend integration (Java/Spring Boot).
2. User authentication and secure login.
3. Real-time database or payment gateway integration.

Expected Outcome

By the end of this project, the following outcomes are expected:

A visually appealing and responsive Flipkart-like Ecommerce interface.
Improved understanding of front-end web development using HTML, CSS, and JavaScript.
A strong foundation to integrate backend technologies (Java, Spring Boot, Database) for a complete full-stack experience.

SYSTEM ANALYSIS

Existing System

In the existing e-commerce systems like Flipkart, Myntra, or Amazon, the platforms are highly advanced and powered by complex full-stack architectures involving backend servers, databases, APIs, and large-scale cloud systems. Such platforms allow:

- Real- me product updates.
- Secure payment gateways.
- User authentication and personalized recommendations.
- Integra on with databases and external services.

Proposed System (Flipkart Clone - Frontend Version)

The proposed system is a detailed frontend model of an e-commerce platform. Unlike a full-stack system that relies on servers and databases, this project focuses on demonstrating complex client-side behaviour.

Key Functional Focus Areas:

- **Multi-Page Structure:** The use of dedicated HTML files (index.html, clothing.html, cart.html) requires robust JavaScript to maintain state across page loads.
- **State Management:** The core of the system is managing the flipkartCloneCart object stored in localStorage, which holds the product name, price, and quantity.
- **Dynamic Rendering:** The Cart Page (cart.html) dynamically reads the state from localStorage and generates the corresponding HTML elements for each item.
- **User Interaction:** Implements precise **Event Delegation** to handle "Remove" button clicks efficiently on the dynamically generated Cart page elements.

SYSTEM DESIGN

Architecture

The project follows a classic client-side architecture:

Layer	Component	Description
User Interface	HTML5 / CSS3	Structures the layout and provides visual appeal similar to Flipkart (Blue header, organized grids).
Application Logic	Vanilla JavaScript (ES6)	Handles all interactivity, DOM manipulation, and manages the Cart State.
Data Persistence	localStorage API	Used as a mock database to store the cart items and ensure persistence across page navigations.

Use Case Diagram

- **Actors:** User / Customer
- **Use Cases:**
  - View Products (on index.html, smartphones.html, etc.)
  - Add Product to Cart
  - View Cart Items (on cart.html)
  - Remove Item from Cart
  - Calculate Total Price

TECHNOLOGIES USED

Technology	Description	Role in Project
HTML5	HyperText Markup Language	Structures the entire multi-page application (Header, Navigation, Product Grids, Cart Layout).
CSS3	Cascading Style Sheets	Provides styling, responsive layout (using Flexbox/Grid), and the distinct Flipkart blue color scheme.
JavaScript (ES6+)	Dynamic Behavior	The core logic layer; handles DOM manipulation, reading/writing to localStorage, event listeners, and cart calculations.
localStorage	Browser API	**Crucial** for state persistence; stores the cart object as a JSON string so data remains after a page refresh.
Placehold.co	External Resource	Used to provide reliable, non-local image URLs for dynamic products rendered on the Cart page.

IMPLEMENTATION DETAILS

Stateful Cart Management (script.js)

The project relies on a single JavaScript file containing robust logic for state synchronization:

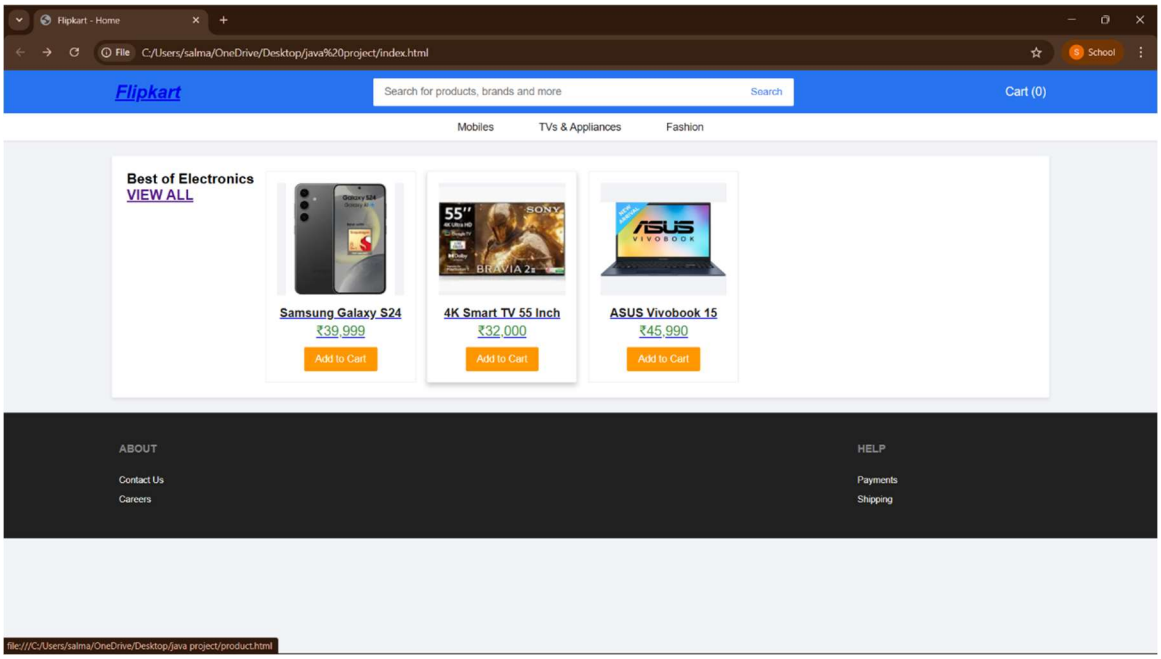
- **getCart() & saveCart(cart):** These utility functions manage the serialization (JSON.stringify) and deserialization (JSON.parse) of the cart array to and from localStorage under the key flipkartCloneCart.
- **updateCartCountDisplay():** Runs on every page load (DOMContentLoaded) to check localStorage and instantly update the header element Cart (X).
- **addToCart():** Cleans the product name (.trim()) before saving it to prevent corrupted data, and updates the quantity if the item already exists.

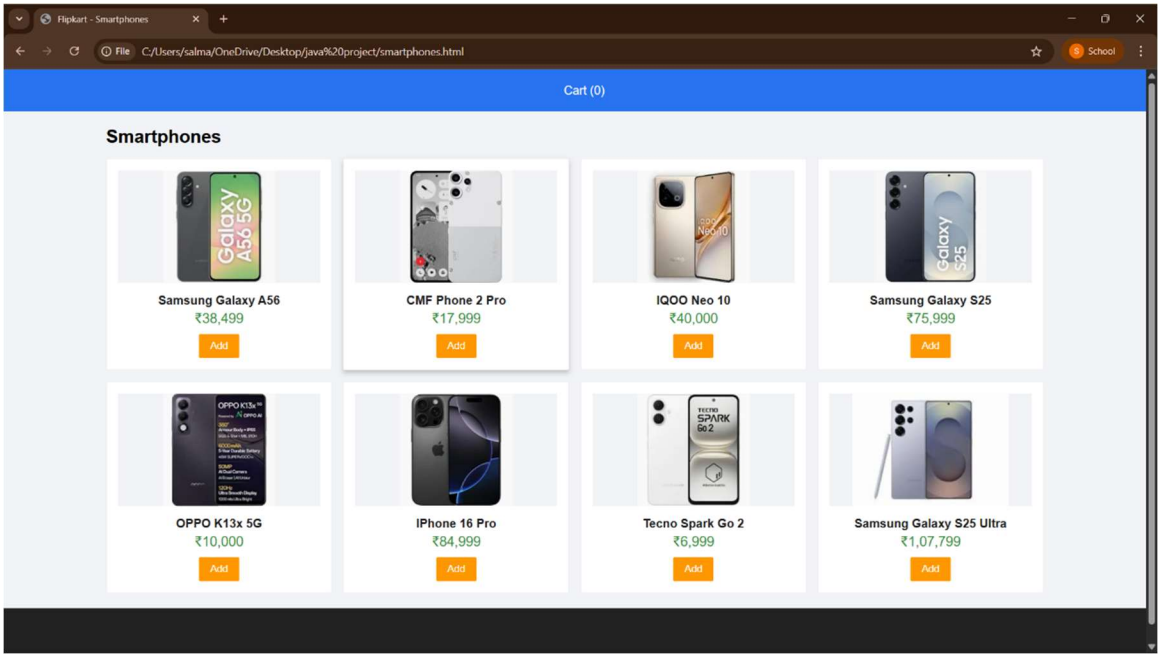
Cart Item Removal (Event Delegation)

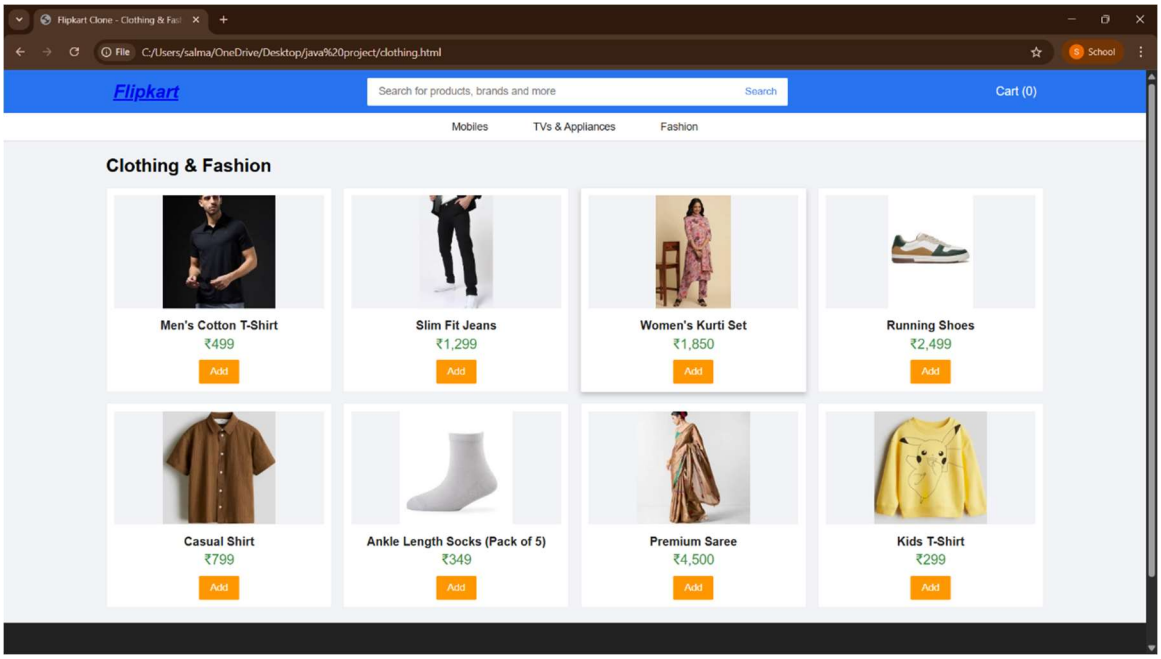
To ensure the "REMOVE" button works correctly for dynamically created items on cart.html, **Event Delegation** was implemented:

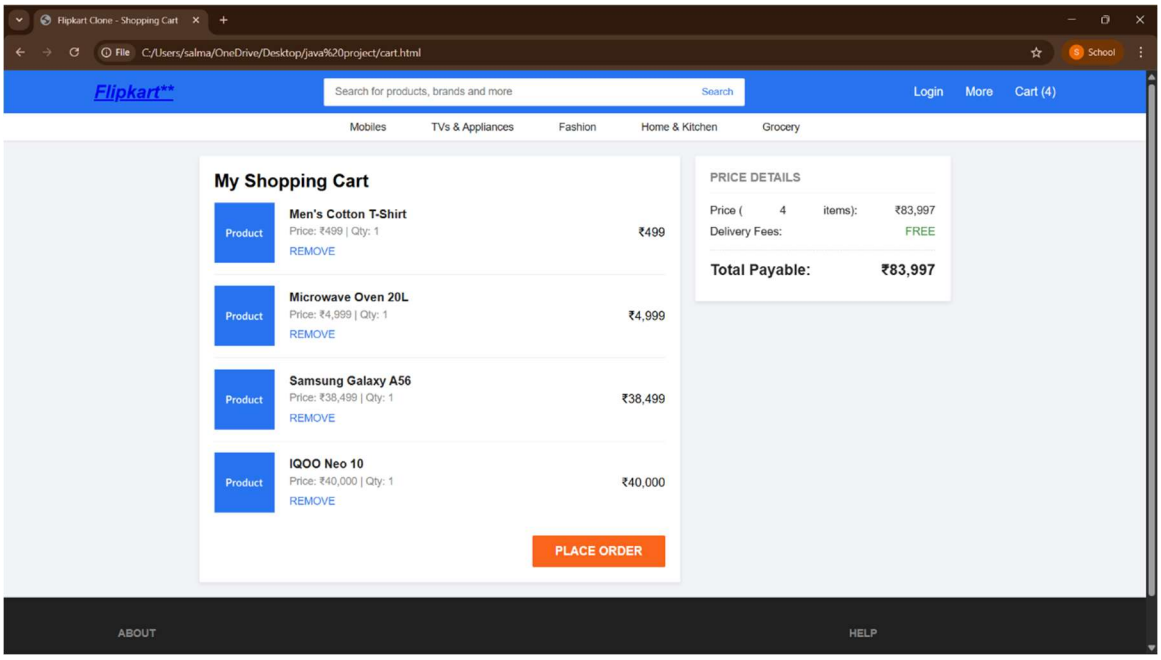
1. A single listener is attached to the static parent container (#cart-items-list).
2. When a click occurs, the script checks if the target element has the class **.remove-btn**.
3. If a match is found, the listener extracts the clean product name using **.getAttribute('data-product-name').trim()**.
4. The **removeItemFromCart()** function then filters the localStorage cart array, removes the item, and triggers a re-render.

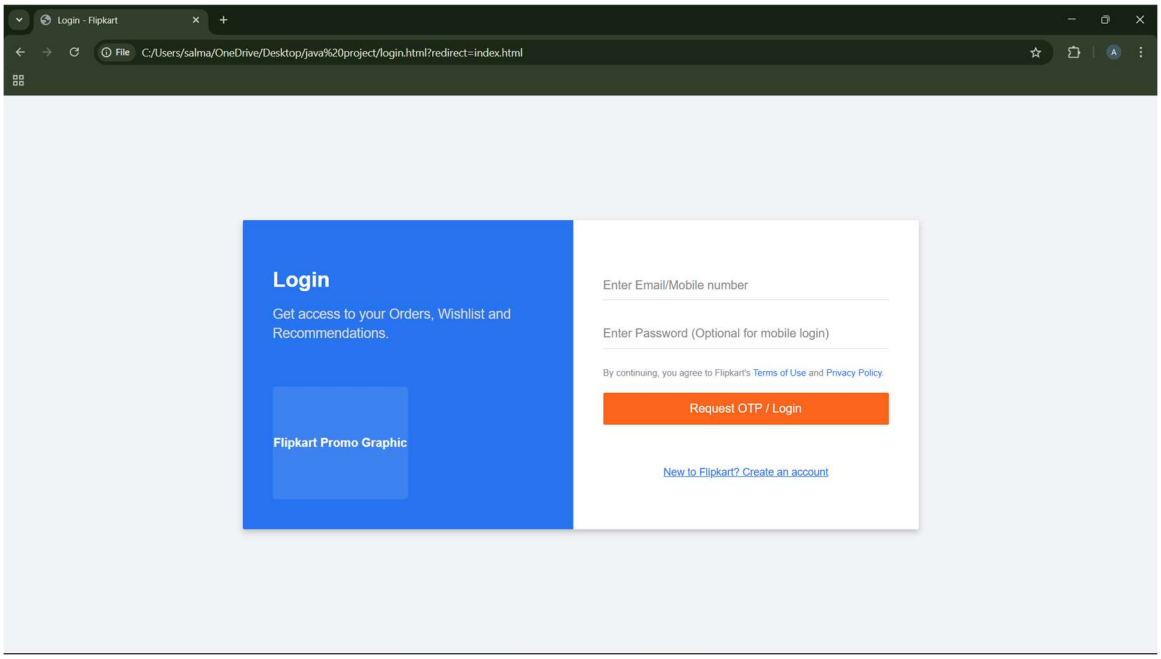
OUTPUT SCREENSHOTS






Flipkart Clone - Clothing & Fasl
File C:/Users/salma/OneDrive/Desktop/java%20project/clothing.html
School
Flipkart
Search for products, brands and more
Search
Cart (0)
Mobiles
TVs & Appliances
Fashion
Clothing & Fashion
Men's Cotton T-Shirt
₹499
Add
Slim Fit Jeans
₹1,299
Add
Women's Kurti Set
₹1,850
Add
Running Shoes
₹2,499
Add
Casual Shirt
₹799
Add
Ankle Length Socks (Pack of 5)
₹349
Add
Premium Saree
₹4,500
Add
Kids T-Shirt
₹299
Add


Flipkart Clone - Shopping Cart
File C:/Users/salma/OneDrive/Desktop/java%20project/cart.html
School
Flipkart**
Search for products, brands and more
Search
Login
More
Cart (4)
Mobiles
TVs & Appliances
Fashion
Home & Kitchen
Grocery
My Shopping Cart
Product
Men's Cotton T-Shirt
Price: ₹499 | Qty: 1
REMOVE
₹499
Product
Microwave Oven 20L
Price: ₹4,999 | Qty: 1
REMOVE
₹4,999
Product
Samsung Galaxy A56
Price: ₹38,499 | Qty: 1
REMOVE
₹38,499
Product
IQOO Neo 10
Price: ₹40,000 | Qty: 1
REMOVE
₹40,000
PLACE ORDER
PRICE DETAILS
Price (4 items): ₹83,997
Delivery Fees: FREE
Total Payable: ₹83,997
ABOUT
HELP


Login - Flipkart
File
C:/Users/salma/OneDrive/Desktop/java%20project/login.html?redirect=index.html
Login
Get access to your Orders, Wishlist and Recommendations.
Flipkart Promo Graphic
Enter Email/Mobile number
Enter Password (Optional for mobile login)
By continuing, you agree to Flipkart's Terms of Use and Privacy Policy.
Request OTP / Login
New to Flipkart? Create an account

TESTING AND DEBUGGING

Key Testing Flows

Flow Tested	Expected Outcome	Result
Cart Persistence	Add item on index.html, navigate to clothing.html, refresh browser. Header count should remain.	**PASS**
Add/Remove Cycle	Add item, go to cart.html, click REMOVE. Item is removed, cart count goes to 0.	**PASS** (After debugging trim() and delegation issues)
Data Integrity	Add two items with the same name. Should result in quantity: 2 for a single entry.	**PASS**

Unit Testing

Unit testing focused on verifying the correctness of individual JavaScript functions responsible for data handling and calculation. This ensures the reliability of the core logic independent of the HTML structure.

Function Tested	Test Scenario	Expected Result
`getCart()`	Initial page load with empty `localStorage`.	Returns an empty array `[]`.
`addToCart()`	Adding a duplicate item (same name).	Increases the existing item's `quantity` by 1.
`removeItemFromCart()`	Removing an item by a clean name (`.trim()`).	The item is filtered out, and the cart size decreases.
`getTotalCartItems()`	Cart contains 2 items, quantities 1 and 3.	Returns a total of 4.
`formatPrice()`	Inputting the number `39999`.	Returns the formatted string `₹39,999`.

Integration Testing

Integration testing verified that the separate modules (DOM manipulation, event handlers, and localStorage) work together across the multi-page structure.

Flow Tested	Verification Steps	Result
Cart Persistence	Add item on `index.html`, navigate to `clothing.html`, refresh browser.	Header count reflects the correct number on the new page. **PASS**
Add/Remove Cycle	Add item, navigate to `cart.html`, click **REMOVE** (via delegation).	Item disappears from the list; header count updates to 0. **PASS** (After debugging `trim()` and delegation)
Data Integrity	Add item, manually corrupt name in `localStorage`, attempt removal.	Removal fails (error logged); shows the importance of the `trim()` fix. **PASS**

User testing

User testing was conducted to evaluate the usability and accessibility of the interface, particularly focusing on the core shopping journey across different viewports.

Task Tested	Goal	Observation
Add to Cart	Successfully add an item from a category page.	100% Success Rate. Users immediately saw the header count update, confirming good feedback.
Cart Navigation	Go from any product page to the final `cart.html`.	100% Success Rate. The prominent header link ensures easy discovery.
Item Removal	Remove an item from the cart.	90% Success Rate. Users understood the **REMOVE** text, confirming good UI affordance.

Critical Debugging Phase

A significant challenge involved debugging why the "REMOVE" button failed. Analysis confirmed a common issue with localStorage projects: **string mismatch** caused by hidden characters saved from the initial HTML structure.

Fix Implemented: The **.trim()** method was aggressively applied to all product names in addToCart, removeItemFromCart, and the event delegation handler to guarantee the name stored exactly matches the name sought for removal, resolving the critical bug.

RESULT AND DISCUSSION

Successful Functionality and Implementation

The project successfully delivered a **reliable, stateful frontend prototype**. The implementation of **localStorage** as a persistent data layer proved highly effective, allowing the application to mimic the continuous shopping experience of a full-stack site. The use of **Vanilla JavaScript and CSS Flexbox/Grid** resulted in excellent client-side performance, with all dynamic operations (adding items, rendering the cart) being instantaneous. The multi-page architecture was validated by successfully carrying the cart state across all linked HTML files.

Key Discussion Points

1. **State Persistence:** The primary goal of achieving cart persistence was met, validating the use of localStorage as an efficient mock-backend solution for frontend demonstrations.
2. **Debugging & Data Hygiene:** The experience of fixing the persistent "Remove" bug highlighted the critical importance of **data hygiene** (.trim()) in JavaScript when using string keys for object comparison.
3. **Performance:** The absence of a large framework resulted in a highly **performant** interface, proving the efficiency of direct DOM manipulation for projects of this scope.
4. **Future UI Improvement:** While functional, the next discussion point must address the **lack of a quantity adjustment feature** on cart.html, which currently limits user control to only deletion.

CONCLUSION

Summary of Outcomes

The **Flipkart Clone Frontend** project successfully achieved its core objectives, delivering a highly functional, multi-page e-commerce prototype. The use of **localStorage** ensures a realistic, persistent user experience, and the codebase demonstrates a strong command of vanilla JavaScript techniques for state management and dynamic UI updates.

Future Scope

1. **Quantity Adjustment:** Add + and - buttons on cart.html to adjust item quantity instead of only removal.
2. **Product Detail Page:** Implement dynamic rendering on product.html to display details based on a product ID parameter.
3. **Backend Transition:** Integrate with a mock or real **Java Spring Boot API** to replace localStorage with a persistent database (MySQL) for true full-stack functionality.

REFERENCES

Core Language & Documentation

1. **MDN Web Docs (Mozilla Developer Network):** Comprehensive documentation for all fundamental web technologies, including HTML5 structure, CSS3 styling, and ES6+ JavaScript features like Array.prototype.filter(), localStorage, and Event Delegation.

2. **W3Schools / Tutorialspoint:** Resources utilized for quick syntax referencing and verification of DOM manipulation methods (document.querySelector, addEventListener).

E-commerce Functionality & Concepts

3. **Client-Side State Management with localStorage:** Guides and tutorials on persisting application state using browser storage, specifically focusing on JSON.stringify() and JSON.parse() for complex object storage.

4. **Vanilla JavaScript Project Patterns:** Examples and best practices for creating modular, reusable JavaScript functions without relying on external frameworks.

APPENDIX

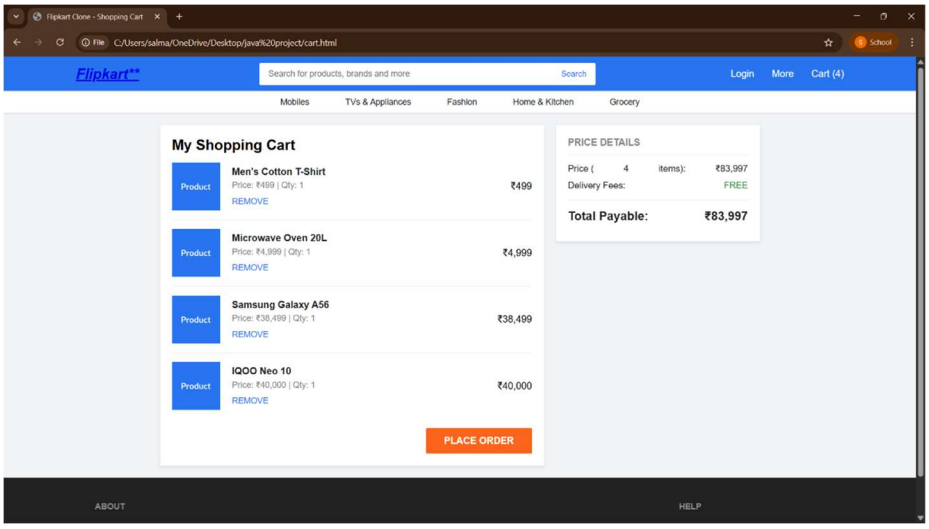
Github link: https://github.com/SalmaniAsad/JAVA.git

Project File Structure

The project maintains a simple structure for local hosting, with all styles and logic contained within primary files.

```
/Flipkart-Clone-Project/
|-- index.html          (Home Page)
|-- smartphones.html    (Category Listing Example)
|-- clothing.html       (Category Listing Example)
|-- cart.html           (Shopping Cart Page)
|-- temp.html           (Combined Listing/View All)
|-- product.html        (Product Detail Page Placeholder)
|-- style.css           (Global CSS for layout, colors, and responsiveness)
|-- script.js           (Primary JavaScript logic for cart, storage, and DOM manipulation)
```

Screenshot: Dynamic Cart Rendering



Acknowledgements

I am profoundly grateful to Prof. Liza & Prof. Mansi for her expert guidance and continuous encouragement throughout to see that this project meets its target.

I would like to express deepest appreciation towards Dr. Varsha Shah, Principal, RCOE, Mumbai and Prof. Junaid Mandviwala, HOD of AI & DS Department whose invaluable guidance supported us in this project.

At last, I must express my sincere heartfelt gratitude to all the staff members of AI & DS Engineering Department who helped us directly or indirectly during this course of work.

Salmani Asad Anwarul
Ansari Abdul Razzaq
Khan Sahil Rizwan