Medicare+

A Medical Store Front

Professor: Ramsey Muvva

INF651: Front End Web Development I

Dec 9, 2025

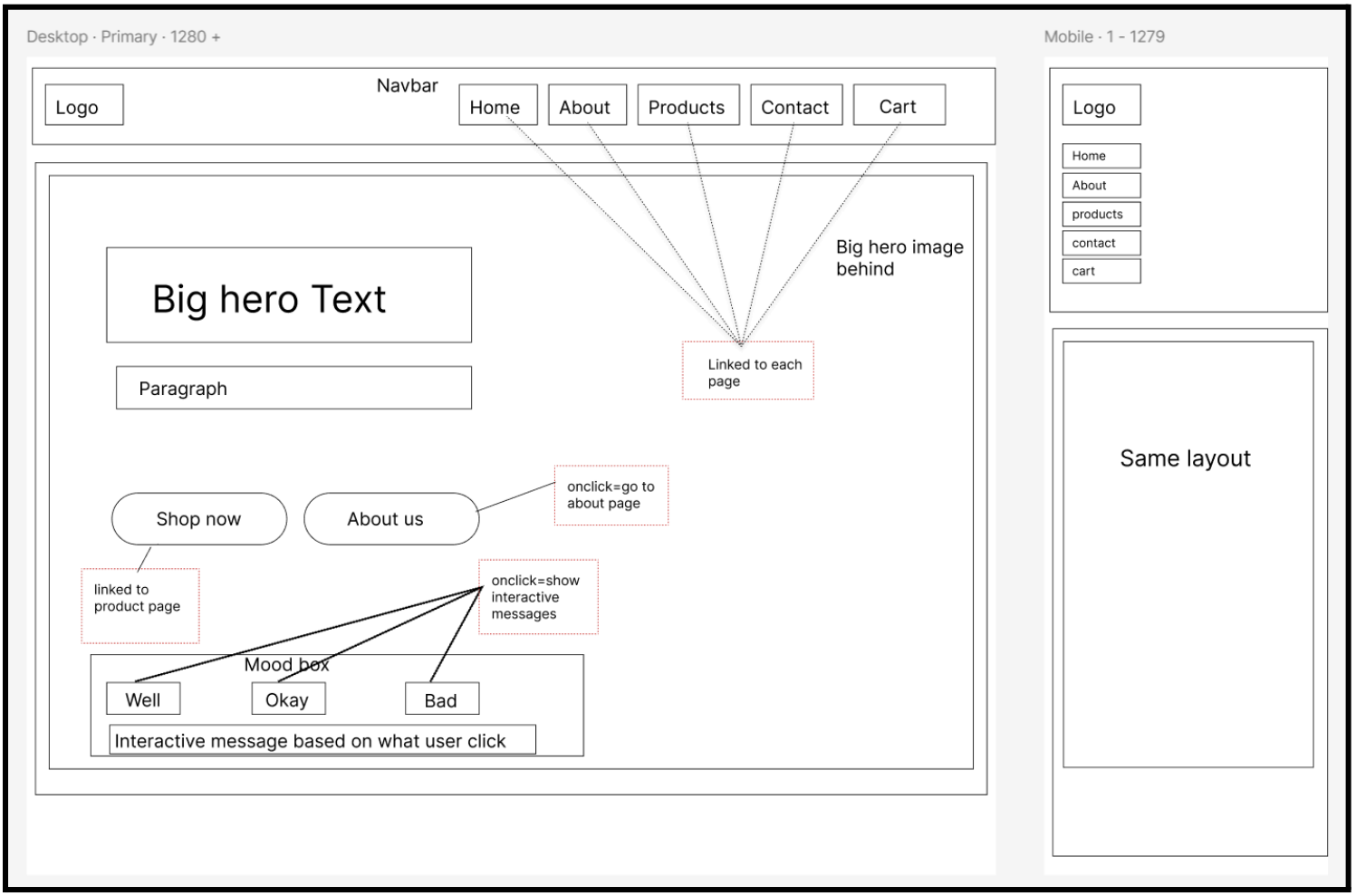
Student: Chivithrokal Chan

Table of Content

Wireframe design

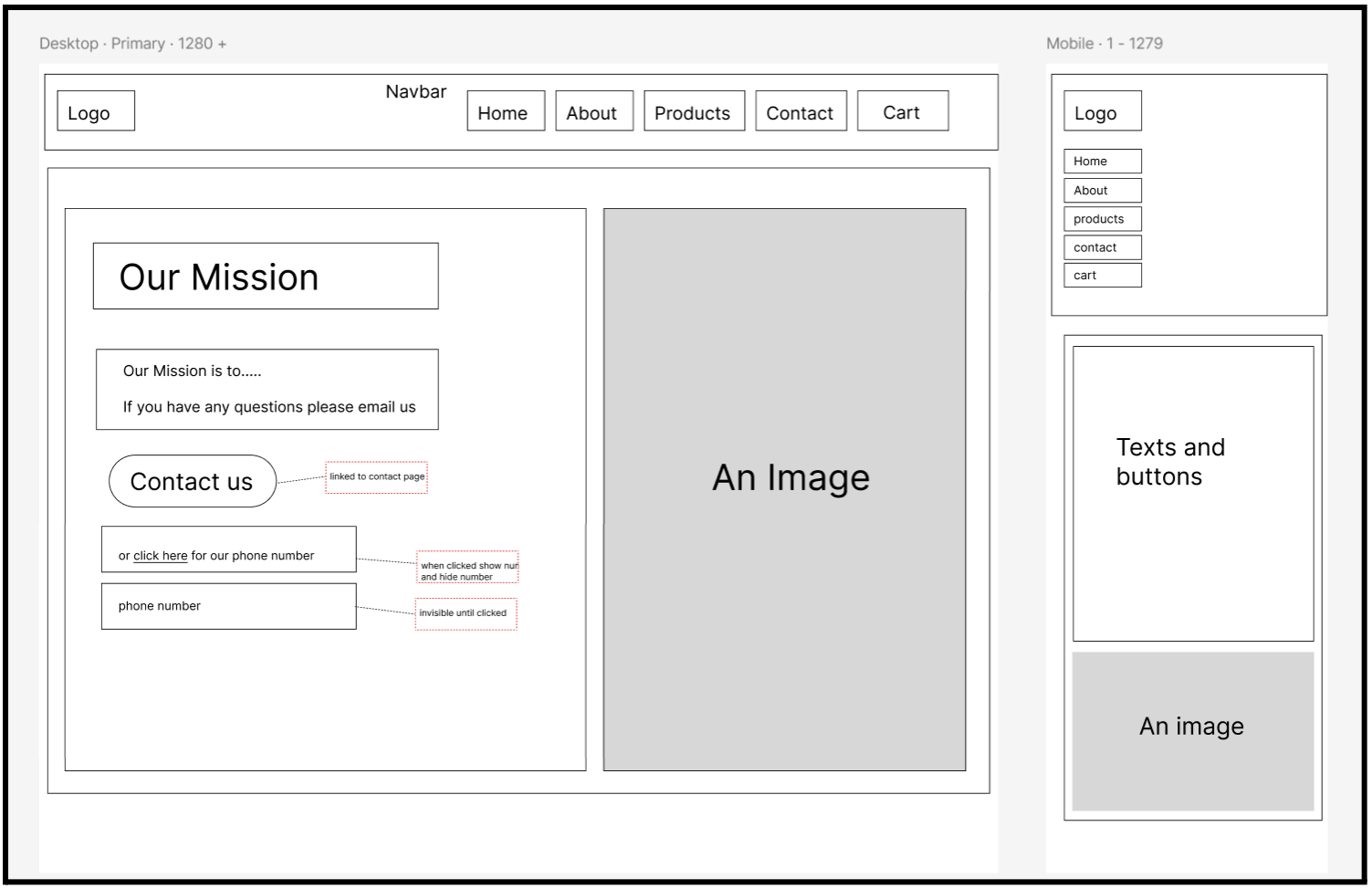
Index/Home page

On the left is the desktop view and the right is the mobile/small screen's view.



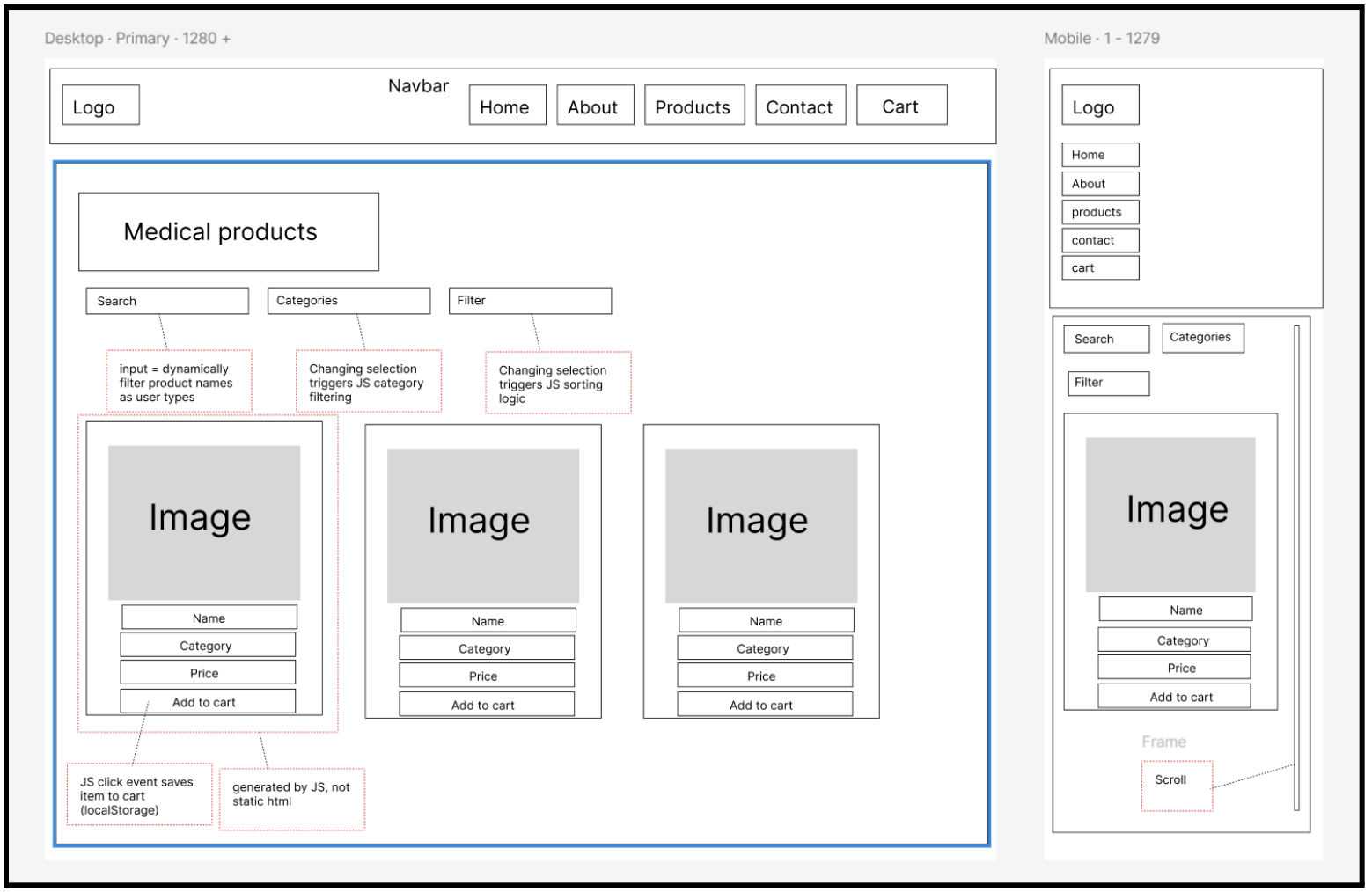
About Page

On the left is the desktop view and the right is the mobile/small screen's view.



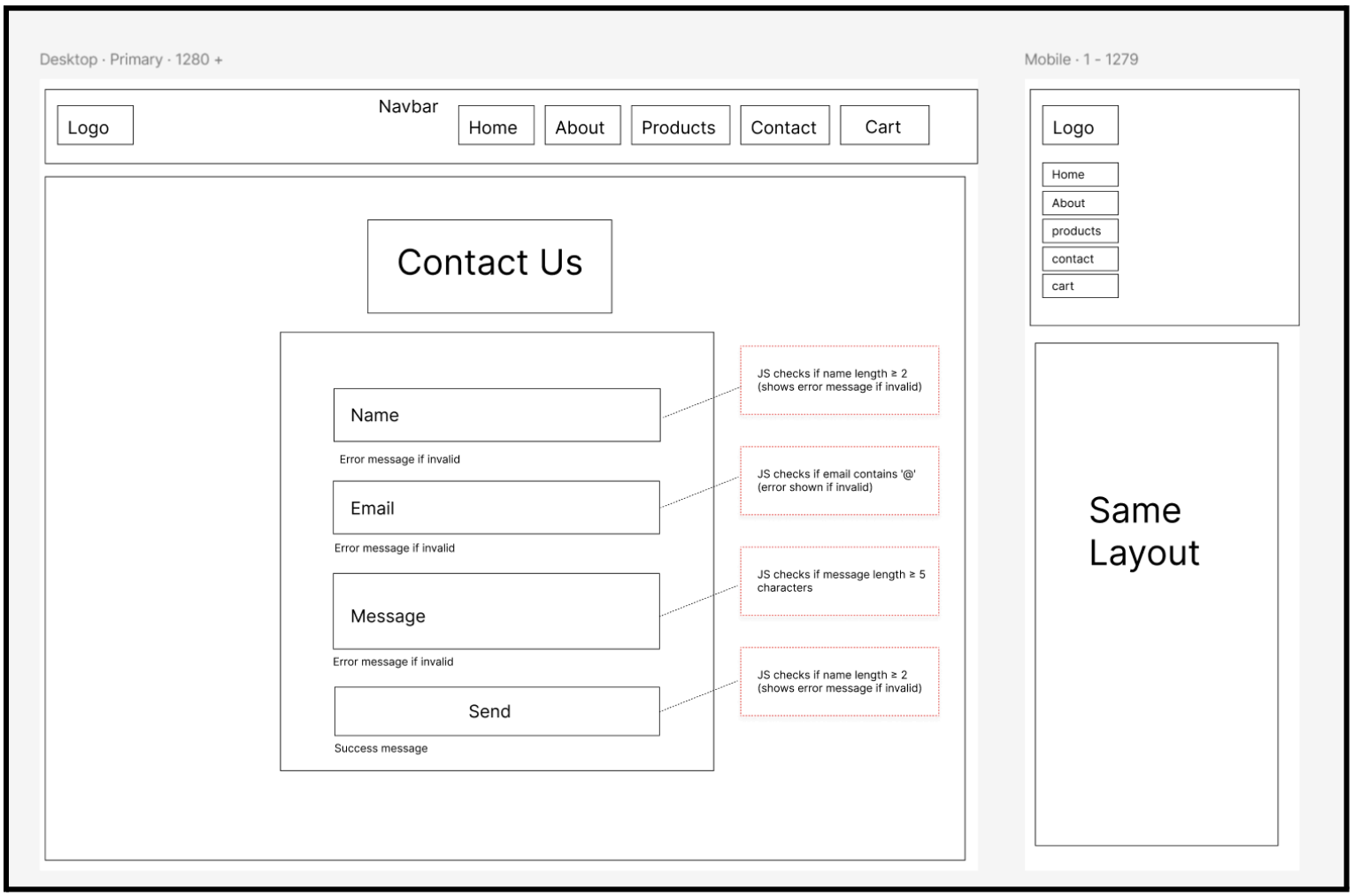
Products Page

On the left is the desktop view and the right is the mobile/small screen's view.



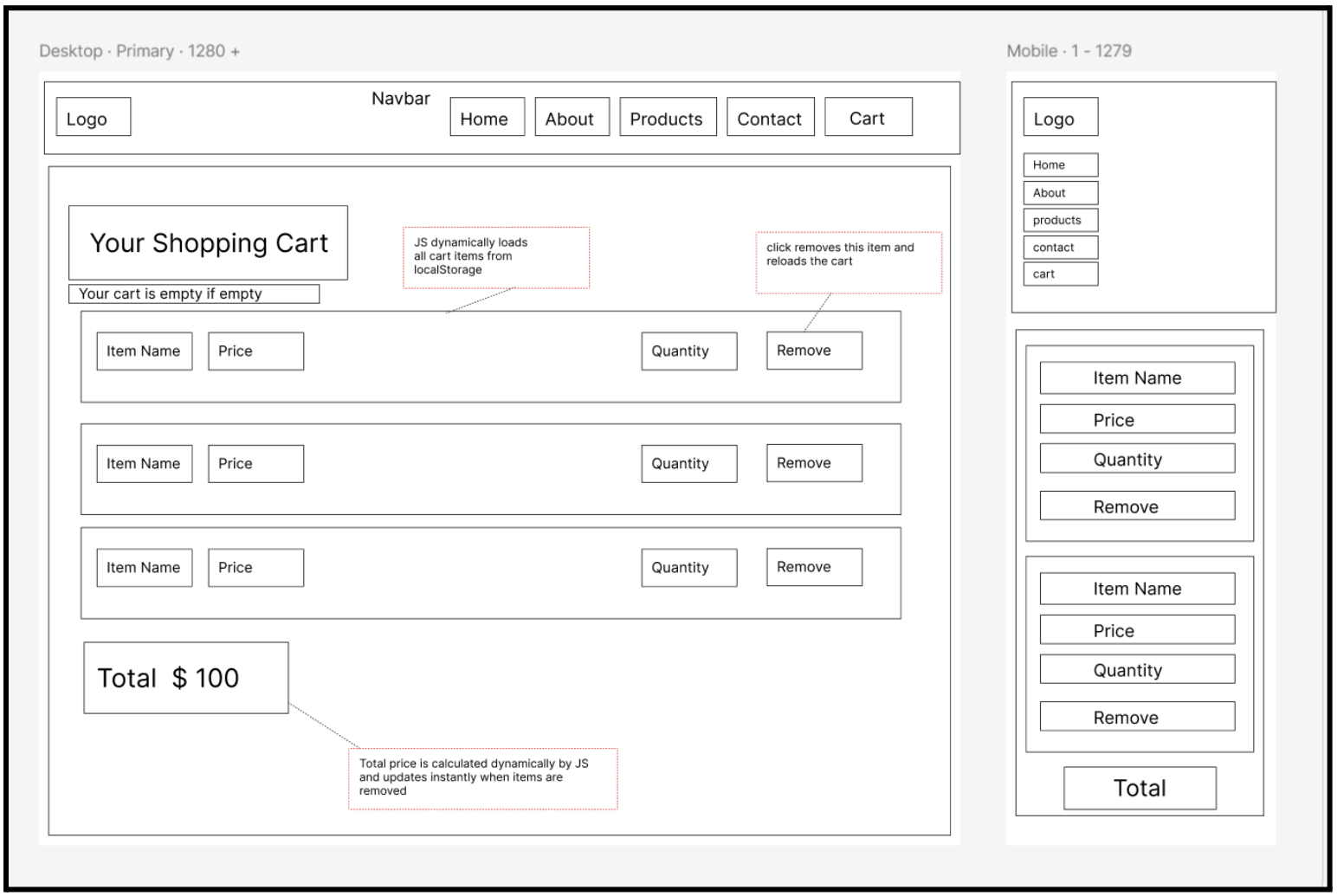
Contact Page

On the left is the desktop view and the right is the mobile/small screen's view.



Cart Page

On the left is the desktop view and the right is the mobile/small screen's view.



1. Features and Functionality

Medicare+ is a multi-page JavaScript-powered web application designed to simulate an online storefront for medical supplies, wellness products, and small medical devices. Throughout the development process, I focused on creating a clean, minimalistic user interface while still incorporating a meaningful level of interactivity. Users can browse through a variety of products, apply filters and sorting options, manage a shopping cart, read about the company, and send inquiries through a fully validated contact form. At this stage, the website consists of five pages that work together to provide a smooth browsing experience.

Unlike a static website, Medicare+ uses JavaScript to enhance usability and engagement. Dynamic product rendering, interactive UI components, persistent cart storage using localStorage, and real-time form validation all contribute to making the application feel more responsive and user-friendly. These features allowed me to explore practical front-end development concepts while also applying JavaScript in ways that directly improve the user experience.

1.1. Home Page (index.html)

The homepage serves as the user's first impression of the Medicare+ platform, and it was designed to be simple, welcoming, and easy to navigate. It includes a clean hero section and a small interactive feature to make the page feel more personal and engaging.

Key Features:

- Hero banner with background overlay to keep the text readable
- "SHOP NOW" button linking to product listings
- "ABOUT US" button redirect handled by JavaScript
- Mood Interaction Widget, allowing users to select how they feel and receive a personalized message.
- JavaScript dynamically updates the mood message using event listeners bound to three buttons. ("Good", "Okay", "Not Well")

1.2. About Page (about.html)

The About page gives users a quick overview of the company's mission and provides different ways for them to reach out. One of the small interactive touches that was added is a phone number toggle, that can be revealed or hidden based on the user's click.

Key Features:

- A toggle button that reveals or hides the company phone number.
- Clicking "click here" shows or hides the support phone number, and the link text updates accordingly. The reveal/hide action is handled via JS class toggling and text replacement.
- "Contact Us" button is linked to the contact page for users who prefer reaching out through the form rather than by phone.

1.3. Products Page (products.html)

The Products page is the most dynamic and interactive part of the entire application. Instead of writing out each product manually in HTML, all product cards are generated, displayed, and updated entirely through JavaScript. With this, I was able to practice working with arrays, loops, DOM manipulation, and event-driven filtering.

Key Features:

Dynamic Product Rendering

- All product information is stored in a JavaScript array of objects (script.js), where each product includes its name, category, price, and image path.
- Using DOM manipulation, JavaScript creates each product card on the fly and inserts it into the page.
- This approach makes the product list much easier to update or expand in the future.

Interactive Filters

Users can refine product results using:

- Search bar (real-time text filtering)
- Category filter (Pain Relief, Supplements, Allergy, etc.)

- Sort options (Price Low to High, Price High to Low, A-Z)
- Every time the user changes one of these filters, the renderProducts() function rebuilds the entire product grid to reflect the new criteria.

Add to Cart System

Each product card includes an "Add to Cart" button. When clicked, JavaScript:

- Saves the selected product into localStorage
- Checks whether the item already exists and increases the quantity if needed
- Gives the user quick feedback through a simple alert message
- Ensures that the cart remains saved even after reloading the page or navigating to another section of the site

This combination of filtering, sorting, and cart management makes the Products page feel much more like a real e-commerce platform, despite being entirely frontend-based.

1.4. Contact Page (contact.html)

The Contact page provides users with a simple way to reach out, but it also gave me an opportunity to practice implementing real-time form validation using JavaScript. Instead of allowing the form to submit immediately, the page checks the user's input and provides helpful feedback to ensure all required information is entered correctly.

Key Features:

User Input Form

The page contains three main input fields:

- Name
- Email
- Message

Real-Time JavaScript Validation

As users type, JavaScript checks for potential issues and displays small error messages directly below the input fields. This helps guide users before they attempt to submit the form.

Successful Submission Behavior

When all fields pass validation:

- A clear success message appears to confirm that the form was sent

- The form automatically resets, clearing all inputs for the next message

Validation Rules

JavaScript checks for:

- A minimum name length
- A valid email structure (at least containing "@")
- A message that meets the minimum length requirement

This feature teaches users to correct mistakes immediately, which improves the overall experience while also preventing incomplete or invalid submissions.

1.5. Cart Page (cart.html)

The Cart page is where the application brings together all the products the user has added while browsing. Since this project does not include a backend or database, the entire cart system is powered by localStorage, which allows items to remain saved even if the user refreshes the page or closes and reopens the browser. This gave me practical experience working with client-side data persistence.

Key Features:

Dynamic Cart Retrieval

When the user opens the Cart page, JavaScript reads all saved items from localStorage and builds the cart interface. Each cart entry displays:

- Product Name
- Price
- Quantity
- Remove Button
- Auto-calculated total price based on quantity

Real-Time Updates Through JavaScript

The cart is fully interactive:

- Removing an item immediately updates the localStorage data
- The UI then re-renders to reflect the updated cart contents
- If the cart becomes empty, the page shows a simple message: "Your cart is empty."

Frontend-Only Cart System

Even without a backend or database, the cart functions smoothly by relying entirely on the browser as storage. This approach allowed me to simulate a small e-commerce experience while staying within the scope of a frontend-only project.

2. How JavaScript Enhances the User Experience

For this project, JavaScript plays a central role in bringing the Medicare+ website to life. Without it, the site would function as a collection of static pages. With JavaScript, however, the application becomes dynamic, interactive, and much more engaging for users. It allows the interface to respond instantly to user actions, which makes the experience feel smoother and more modern.

2.1. Dynamic Content Rendering

One of the best uses of JavaScript in this project is the dynamic rendering on the Products page where instead of writing out each product manually in HTML, JavaScript creates all product cards directly in the browser. This gives the page the flexibility needed in a real shopping environment.

This approach allows:

- Instant re-rendering when users filter or sort every time the user types in the search bar, selects a category, or changes a sorting option, JavaScript completely rebuilds the product grid to match the new criteria.
- No duplicate HTML code where products exist only once in a JavaScript array, and the UI is generated from that data. This makes the code cleaner and easier to maintain.
- Faster and smoother browsing experience
- Easy scalability because adding a new product simply means adding another object to the product array and no extra HTML required. The page automatically updates to display it.

2.2. Real-Time Search, Filtering, and Sorting

Every user action including typing in search, changing category, adjusting sort all updates the product grid instantly. This creates a fast, intuitive browsing experience.

2.3. Client-Side Cart Persistence

Another major enhancement JavaScript brings to the application is the ability to save the user's shopping cart directly in the browser. By using localStorage, the site can remember what items the user added even after refreshing the page or closing the browser. This help deal with the need for a backend database while still providing a smooth shopping experience.

This enables:

- Cart saving across page reloads where users don't lose their selections if they accidentally refresh or navigate away.
- Cart restoration when the user returns, the items remain in the cart as long as the browser data is not cleared.
- A consistent shopping experience without needing a backend, very helpful for projects where server-side storage is not required or available.

Overall, this feature significantly improves usability and makes the site feel more like a real online store.

2.4. Form Validation and User Feedback

JavaScript also improves the user experience on the Contact page by ensuring users enter valid information before the form is submitted. Instead of waiting for a server response, the page provides immediate feedback, which helps users correct mistakes quickly and easily.

This enables:

- Real-time validation where each input is checked as the user types, catching issues early.
- Friendly inline error messages that are small and clear appearing under fields that need corrections.
- Prevention of incomplete or incorrect submissions because the form will not be sent unless all fields meet the requirements.
- Success confirmation and automatic form reset. Once everything is valid, a confirmation message appears, and the fields clear themselves for the next use.

These features help guide the user smoothly through the form, reducing frustration and improving overall clarity.

2.5. Event-Driven User Interactions

Across the entire website, JavaScript is constantly responding to user actions. This event-driven approach makes the interface feel more responsive and alive than static.

This enables:

- Button clicks for actions like adding items to the cart, selecting moods, or navigating.
- Input changes used for live filtering in the search bar and form validation.
- Form submissions ensure proper validation before sending messages.
- Page load events help initialize things like cart contents or product rendering.

By reacting instantly to these events, JavaScript creates an interactive experience that enhances how users navigate and interact with the site.

3. Challenges Faced and Solutions

One of the biggest challenges I faced was figuring out how to store the users' cart data. I don't have much experience working with databases yet, so I needed a way for the cart to "remember" items even after the user switched pages or refreshed the browser. At first, I wasn't sure how to approach this because everything in my project is entirely frontend, no backend, no database.

Solution:

I decided to use localStorage as a simple, temporary storage system for the cart. Instead of relying on a full database, I stored all cart items inside an array and saved that array as a JSON string in localStorage.

Here's how it works:

- When a user adds a product, JavaScript checks if that product already exists in the saved cart.
- If it does, the quantity increases, if not, a new entry is added.
- After every update, the cart array is saved back into localStorage.
- When the cart page loads, the app reads from localStorage to rebuild the cart UI.

This let me create a small but functional e-commerce cart system without needing any backend experience, and it helped me understand how data persistence can still work in a frontend-only project.

4. Plans for Additional Features & Backend Integration

While the current version of Medicare+ operates entirely on the frontend, there are future implementations that can be done to extend its functionality and make the application feel even closer to a real e-commerce platform. These include improving the existing cart system, eventually integrating a backend for persistent data storage, and more advanced features.

4.1 Planned Additional Features

4.1.1 Improved Cart Functionality

To make the shopping cart system better, more intuitive and interactive, the following improvements are planned:

- **Quantity increase/decrease buttons** that allow users to adjust item quantities directly from the cart page instead of relying only on repeated clicks from the product list.
- **Subtotal calculation per item** that displays a subtotal for each product (price × quantity) would give users a clearer picture of how their total is calculated.
- **Styled notifications instead of alerts** replacing basic JavaScript alert() messages with small pop-ups would provide a cleaner and more modern user experience.

4.1.2 Product Detail Pages

Currently, the site only displays products in a grid format. A future enhancement is to give each product its own dedicated page, featuring:

- **Larger, high-quality images** to give users a clearer view of the item

- **More detailed descriptions** explaining what the product is used for and any important information a customer might need

- **Additional metadata,** such as ingredients, dosage instructions, product specifications, or usage warnings

This would help the site feel more realistic and give users more information before adding items to their cart.

4.1.3 Wishlist or Favorites System

Users could mark products they are interested in and save them locally for later viewing.

4.1.4 User Accounts

A more advanced version of Medicare+ could allow users to create personal accounts, enabling features such as:

- **Sign in / Sign up** to create a personalized shopping experience
- **Save their carts** so items follow them across devices instead of being limited to localStorage
- **View purchase history or track orders,** giving them a record of past activity

These features would help greatly expand the functionality of the site, but they also require a backend to store user data securely and manage authentication.

4.2 Backend Integration Plans

If expanded to a full application, backend features may include:

4.2.1 Database-Backed Product Management

Admins could log in and manage products through a dashboard, including:

- Adding new products
- Editing existing product details
- Removing outdated or sold-out items

A database would give consistent and secure product storage.

4.2.2 Secure Cart & Checkout System

A backend would allow the application to implement:

- Server-side cart storage tied to user accounts
- Payment integration (possibly Stripe or PayPal)
- Order confirmation and digital receipts

This would move Medicare+ from just a demo project into something that works more like a real e-commerce site.

4.2.3 Email Handler for Contact Form

Instead of showing only a frontend success message, the contact form could:

- Send emails directly to the company
- Log customer inquiries for future reference

This would make communication more reliable and professional.

4.2.4 Inventory & Stock Tracking

With a backend, the platform could track product availability and prevent users from ordering items that are out of stock, essential for an actual e-commerce system.

Conclusion

Overall, the Medicare+ application included what I've learned about JavaScript, especially in areas like DOM manipulation, form validation, event-driven programming, and managing data on the client side. Even though the project is fully frontend-based, it still manages to create a smooth, interactive, and user-friendly storefront experience.

Though the website is not perfect, it still has a solid foundation. If I choose to continue developing it, the current setup would make it much easier to expand into a full-stack application with backend support and more advanced e-commerce features.

Working on Medicare+ really helped me practice JavaScript, especially things like DOM manipulation, form checking, event handling, and saving data on the client side. It

made me think about user experience, data handling, and scalability in ways I had only skimmed before. Overall, this project turned out to be a really valuable learning experience for me. I can genuinely say that I came out feeling much more confident in my abilities than when I started and it was really gratifying when everything started to work. Thank you!