SCHOOL OF FUTURE TECH

ITM SKILLS UNIVERSITY

Case Study Report: WEARABLE TECH WEBSITE

PROJECT REPORT:

GARMIN INDIA – PREMIUM E-COMMERCE WEBSITE

SUBMITTED BY: Aaryan Kuchekar

1. WEBSITE DETAILS

Title of the Website

Garmin India – Premium Multi-Page E-Commerce Experience

Purpose of the Website

The purpose of this website is to design and develop a modular, scalable, multi-page e-commerce platform inspired by Garmin's product ecosystem. The website demonstrates:

- Advanced semantic HTML structure
- Modular CSS architecture
- Reusable JavaScript components
- Dynamic product modal system
- localStorage-based shopping cart
- Mega navigation overlay
- Responsive grid + flex layouts
- Performance optimizations (lazy load, intersection observer)

Target Users

- Fitness Enthusiasts
- Outdoor Adventurers
- Athletes
- Automotive & Marine Users
- Indian E-commerce Customers

Tools & Technologies Used

- HTML5 (Semantic Structure)
- CSS3 (Grid, Flexbox, Media Queries, Animations)
- JavaScript (ES6, DOM API, localStorage, Intersection Observer)
- VS Code
- Chrome DevTools

Key Features

- Fully modular navigation system (injected dynamically)
- Mega menu with hover interaction
- Horizontal scroll product filmstrip
- Quick View Modal system
- Persistent shopping cart (localStorage)
- Scroll reveal animations
- Responsive multi-breakpoint layout
- Accessibility enhancements (ARIA, keyboard focus)

2. PROJECT ARCHITECTURE

2.1 File Structure & Modularity

Your system is modular. Components are injected once and reused everywhere.

Core Files

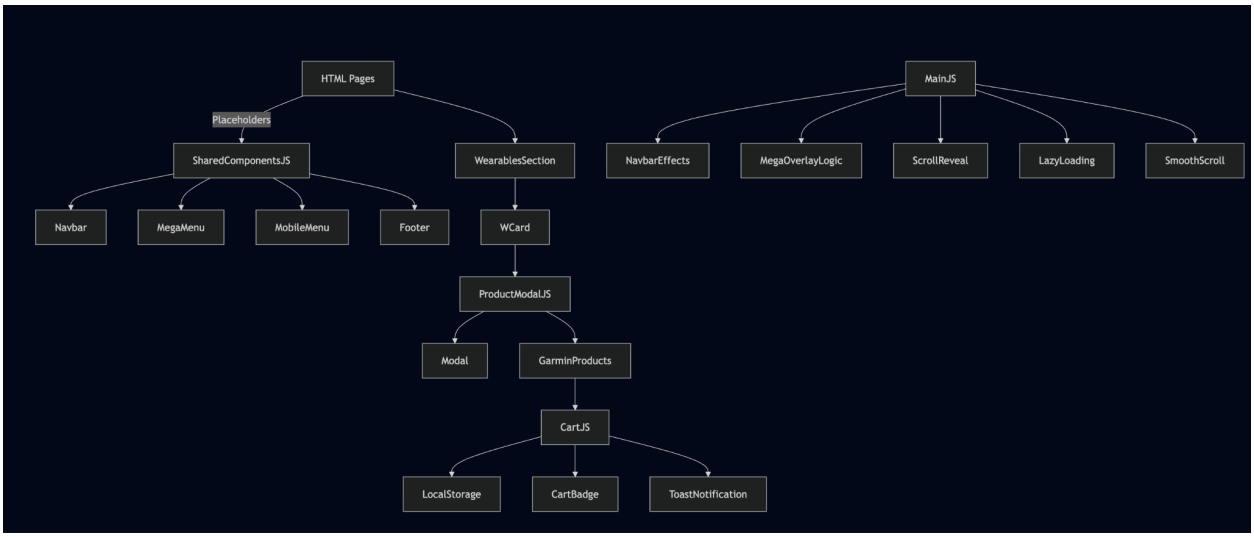
CSS Files

- `footer.css`
- `wearables.css`
- `modal.css`
- (Global styles assumed in main stylesheet)

JavaScript Files

- `main.js`
- `shared-components.js`
- `product-modal.js`
- `cart.js`

2.2 System Architecture Diagram



3. HTML STRUCTURE ANALYSIS

3.1 Semantic HTML Tags Used (Project-Specific Explanation)

Tag	Where It Is Used in This Project	Purpose in Our Website	Technical Importance
`<html>`	Root of every page with `data-depth` attribute	Stores page depth for dynamic path resolution	Enables shared-components.js to correctly generate relative links
`<head>`	All pages	Contains metadata, CSS links, JS links	Loads modular styles and scripts
`<body>`	All pages	Contains visible UI and injected components	JavaScript attaches event delegation to body

`<nav>`	Injected via `shared-components.js`	Main navigation bar (Glass navigation)	Semantic landmark for navigation; supports accessibility
`<footer>`	Injected footer component	Contains company info, links, newsletter, legal	Defines page-ending content using semantic landmark
`<section>`	Wearables, product categories, hero sections	Groups logically related content blocks	Improves semantic structure and SEO clarity
`<div>`	Used extensively in layout structure	Structural wrapper for layout (Grid/Flex containers)	Used for CSS positioning and layout control
`<main>`	Implied through skip link	Primary content area	Improves accessibility for screen readers
`<h2>`	Section titles (e.g., Wearables titles)	Defines major content headings	Helps SEO hierarchy and document structure
`<h3>`	Footer column headings	Secondary content headings	Organizes footer into semantic blocks
`<h4>`	Mega menu headings	Sub-category grouping	Maintains logical heading hierarchy
`<p>`	Newsletter text, feature descriptions	Displays descriptive content	Used inside modal features and footer

`<span>`	Price labels, badges, eyebrow text	Inline text styling	Used for dynamic cart count and feature eyebrow
`<strong>`	Cart toast product name	Emphasizes product name in notification	Semantic emphasis instead of bold styling
`<ul>`	Navigation links, footer links	Unordered list structure	Groups related links semantically
`<li>`	Inside `<ul>`	Individual navigation/footer items	Structured list organization
`<a>`	Navigation links, product links, social links	Hyperlink to pages or sections	Core navigation mechanism
`<form>`	Newsletter subscription form	Captures user email	Structured user input submission
`<input type="email">`	Newsletter form	Collects email address	Enforces email validation at browser level
`<button>`	Buy buttons, modal buttons, menu toggle	Triggers actions (JS events)	Used for interaction rather than navigation
`<img>`	Product images, modal images	Displays product visuals	Lazy-loaded using Intersection Observer

`<video>`	Hero background video	Visual engagement	Paused when not in viewport (performance optimized)
`<svg>`	Icons in nav, modal, cart, footer	Vector-based icons	Scalable, lightweight, styleable via CSS

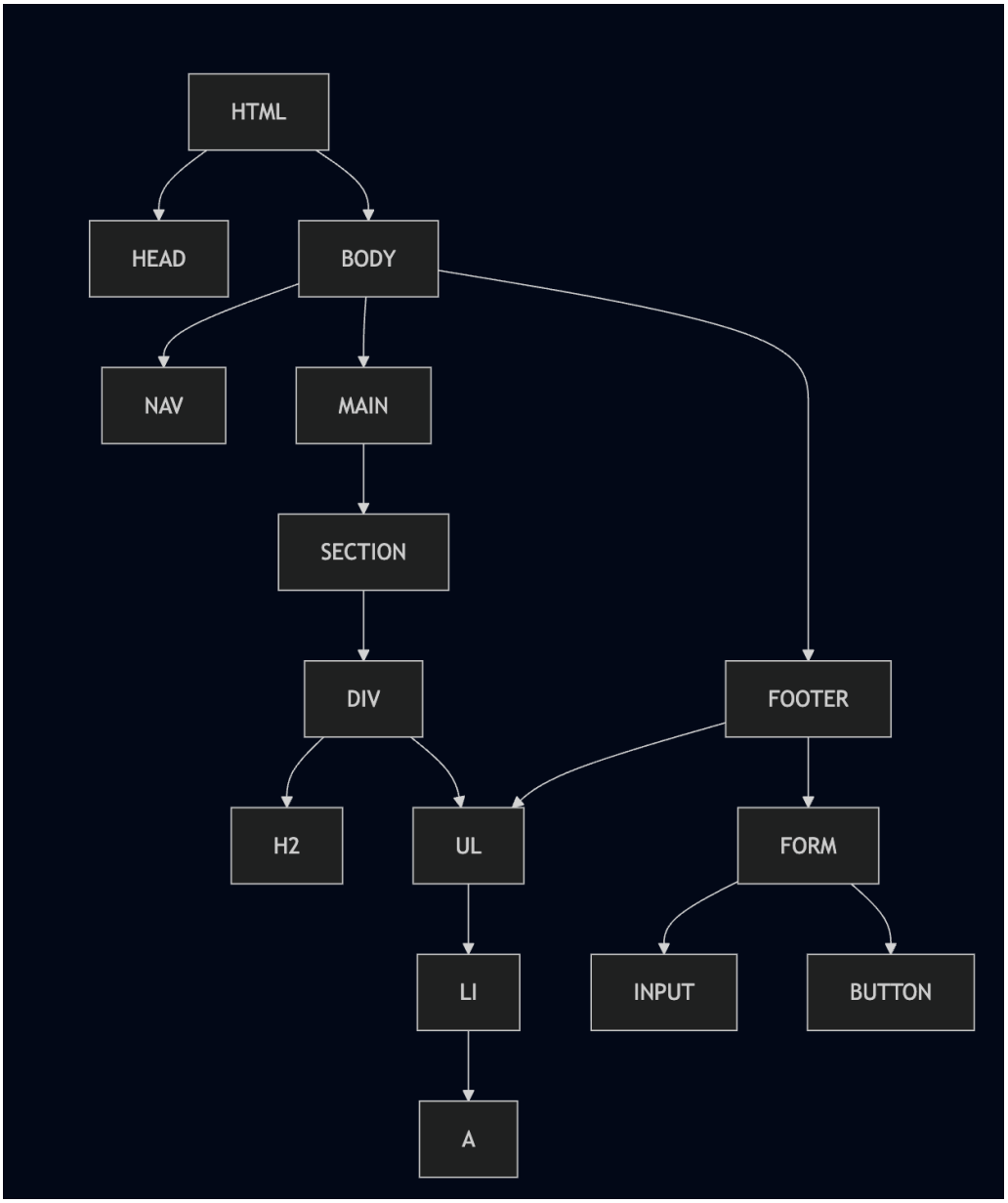
Accessibility Attributes Used

Attribute	Where Used	Purpose in This Project
`role="navigation"`	`<nav>`	Defines navigation landmark for assistive technologies
`role="dialog"`	Modal card	Identifies modal as dialog window
`aria-label`	Buttons, nav, mobile toggle	Provides descriptive label for screen readers
`aria-expanded`	Mobile menu toggle	Indicates open/closed state
`aria-hidden`	Modal backdrop	Indicates visibility state
`aria-modal="true"`	Modal	Signals that modal traps focus

`tabindex="0"`	Country selector	Makes non-button element keyboard accessible

How These Tags Work Together (Structural Hierarchy)

Here is the structural relationship in your project:



Important Insight

The project does NOT use div everywhere blindly.

we used:

• `<nav>` for navigation landmark
• `<footer>` for closing section
• `<section>` for logical grouping
• `<ul>` for grouped navigation links
• `<button>` for actions (not `<a>`)
• `<a>` strictly for navigation

3.2 Shared Components Injection Logic

Your navigation, mega menu, mobile menu and footer are injected using:

document.getElementById("nav-root").innerHTML = navHTML;

This achieves:

- DRY architecture (Don’t Repeat Yourself)
- Centralized navigation management
- Cross-page consistency
- Scalable maintainability

4. CSS ARCHITECTURE ANALYSIS

The project follows a modular CSS architecture where each major UI section is isolated into its own stylesheet:

- `globals.css` – Design system, tokens, utilities
- `hero.css` – Landing page hero section
- `wearables.css` – Product filmstrip & section styling
- `modal.css` – Quick View modal
- `footer.css` – Footer layout & styling

This separation ensures maintainability, reusability, and clarity.

4.1 Layout Systems Used

The project uses **CSS Grid and Flexbox strategically**, not randomly.

4.1.1 Grid Layout Usage

Where Grid Is Used

- `.footer-top`
- `.mega-grid`
- Multi-column footer sections
- Mega menu content columns

Example 1 – Footer Grid (`footer.css`)

```
.footer-top {
 display: grid;
 grid-template-columns: 1.2fr 1fr 1fr 1.5fr;
 gap: 80px;
}
```

What This Code Does

- Creates 4 responsive columns
- Allocates proportional width using `fr` units
- Ensures equal vertical alignment
- Adds controlled spacing with `gap`

Purpose

- Clean multi-column structure
- Professional enterprise-style footer
- Predictable content distribution
- Easy conversion to 2-column and 1-column layouts at breakpoints

---

Example 2 – Mega Menu Grid (`shared-components.js` → injected HTML + CSS)

Mega panels use:

```
.mega-grid {
 display: grid;
}
```

This creates structured product category columns.

Purpose

- Organized mega dropdown layout
- Easy scalability (add/remove columns without layout breaking)
- Clean separation between menu groups

Why Grid Was Chosen

Grid is ideal for:

- Two-dimensional layouts
- Structured columns

- Footer architecture
- Mega menu organization

It provides more control than flexbox for column-based structures.

4.1.2 Flexbox Usage

Flexbox is used for alignment, stacking, and dynamic distribution.

Where Flexbox Is Used

- Navbar container
- Modal split layout
- Product card layout
- Footer social icons
- Horizontal product filmstrip
- Card action buttons
- Mobile menu
- Newsletter form

Example 1 – Horizontal Product Filmstrip (`wearables.css`)

```
.wearables-scroll-container {
 display: flex;
 overflow-x: auto;
 scroll-snap-type: x mandatory;
}
```

What This Code Does

- Creates a horizontal row of cards
- Enables scrollable overflow
- Combines with `scroll-snap-type` for smooth snapping

Purpose

- Apple-style filmstrip experience
- Mobile-friendly horizontal scrolling
- Dynamic card expansion

Example 2 – Modal Split Layout (`modal.css`)

```
.modal-card {
 display: flex;
 flex-direction: row;
}
```

What This Code Does

- Splits modal into:
    - Left: image
    - Right: content

At mobile breakpoint:

```
@media (max-width: 768px) {
 .modal-card {
   flex-direction: column;
 }
}
```

Purpose

- Desktop: side-by-side layout
- Mobile: vertical stacking
- Adaptive UI behavior

Example 3 – Card Actions

```
.w-card-actions {
 display: flex;
 justify-content: space-between;
}
```

Purpose

- Align price on left
- Align buttons on right
- Maintain visual balance

Why Flexbox Was Chosen

Flexbox is ideal for:

- One-dimensional layouts
- Aligning items horizontally or vertically
- Distributing space dynamically
- Responsive stacking

4.2 Responsive Design Strategy

The project follows a structured breakpoint system.

Breakpoints Used

- **1024px** → Tablet/Desktop transition
- **768px** → Tablet/Mobile transition
- **480px** → Small mobile devices

Defined across all major CSS files.

Responsive Techniques Used

1. Grid → Single Column Conversion

Example (`footer.css`):

```
@media (max-width: 768px) {
 .footer-top {
   grid-template-columns: 1fr;
 }
}
```

Effect:

- Footer collapses to single column
- Improves readability
- Enhances mobile usability

---

2. Flex Direction Changes

Example (`hero.css`):

```
@media (max-width: 768px) {
 .hero-cta {
   flex-direction: column;
 }
}
```

Effect:

- Buttons stack vertically
- Prevents overflow
- Maintains spacing

---

3. Scroll Padding Adjustments

```
@media (max-width: 768px) {
 .wearables-scroll-container {
   padding: 0 var(--spacing-md);
 }
}
```

Effect:

- Maintains edge spacing on mobile
- Improves swipe interaction

---

4. Mobile Menu Transformation

Desktop:

- Mega menu (hover-based)

Mobile:

- Off-canvas slide menu
- Accordion toggles
- Backdrop overlay

Handled through:

mobileMenu.classList.toggle("active", isActive);

This completely transforms navigation behavior based on viewport.

4.3 Animations & Transitions

The project heavily relies on GPU-accelerated animations.

4.3.1 Properties Used

- `transition`
- `transform`
- `opacity`
- `box-shadow`
- `backdrop-filter`
- `scroll-snap-type`
- `cubic-bezier()`

4.3.2 Hover Animations

Example (`wearables.css`):

```
.w-card:hover {
 transform: translateY(-8px);
 box-shadow: 0 20px 40px rgba(0, 0, 0, 0.12);
}
```

What Happens

- Card lifts upward
- Shadow deepens
- Creates elevation illusion

This simulates physical movement.

4.3.3 Modal Animation

Initial State:

```
.modal-card {
 transform: scale(0.92) translateY(20px);
 opacity: 0;
}
```

Active State:

```
.modal-backdrop.active .modal-card {
 transform: scale(1) translateY(0);
 opacity: 1;
}
```

Effect

- Modal scales up smoothly
- Moves upward
- Fades in

This creates Apple-like motion physics.

4.3.4 Toast Animation (Injected via JS)

From `cart.js`:

```
.cart-toast {
 transform: translateX(100%) scale(0.95);
 opacity: 0;
}

.cart-toast.show {
 transform: translateX(0) scale(1);
 opacity: 1;
}
```

Effect

- Slides in from right
- Slight scale pop
- Fades in
- Auto-dismiss after 3 seconds

4.3.5 Parallax Background Effect

From `wearables.css`:

```
.section-bg {
 background-attachment: fixed;
}
```

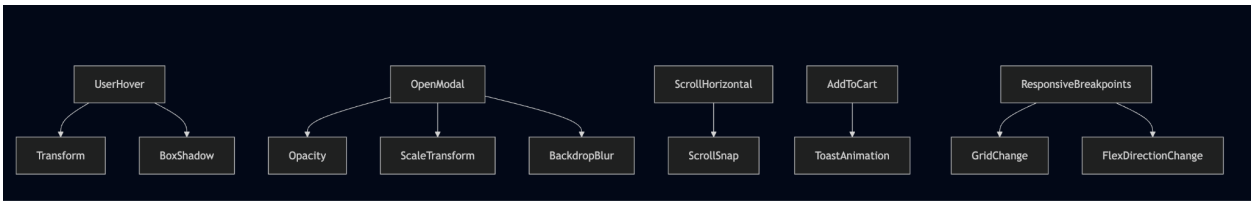
Effect

- Background remains fixed

- Content scrolls over it
- Creates depth illusion

Used for premium product section backgrounds.

CSS Motion Architecture Diagram



---

5. JAVASCRIPT SYSTEM ANALYSIS

The JavaScript architecture is modular and responsibility-driven.
Each file controls a specific behavioral domain:

- `main.js` → Core UI interactions & performance
- `shared-components.js` → Component injection system
- `product-modal.js` → Dynamic product quick-view system
- `cart.js` → LocalStorage-based cart engine

This separation ensures:

- Scalability
- Maintainability
- Reusability

- Clear separation of concerns

5.1 `main.js` Responsibilities

This file controls **global UI behavior and performance optimizations**.

5.1.1 Navbar Scroll Effect

Code

```
const navbar = document.getElementById("navbar");

const handleNavScroll = () => {
 const currentScrollY = window.scrollY;

 if (currentScrollY > 50) {
   navbar.classList.add("scrolled");
 } else {
   navbar.classList.remove("scrolled");
 }
};
```

What This Does

- Detects vertical scroll
- Adds `.scrolled` class after 50px
- Removes it when back at top

Why It Is Used

- Enables glass → solid navbar transition
- Improves readability after hero section

- Enhances premium interaction feel

Performance Optimization

```
let ticking = false;

window.addEventListener("scroll", () => {
 if (!ticking) {
   window.requestAnimationFrame(() => {
     handleNavScroll();
     ticking = false;
   });
   ticking = true;
 }
});
```

Why This Is Important

- Prevents excessive scroll event execution
- Uses `requestAnimationFrame`
- Improves performance on low-end devices

This shows performance-aware engineering.

5.1.2 Mega Overlay System

A detached fixed overlay system for desktop navigation.

Overlay Activation

```
const showOverlay = (category) => {
 megaOverlay.classList.add("active");
};
```

Category Switching

```
document.querySelectorAll(".mega-panel").forEach((panel) => {
 panel.classList.remove("active");
});

const targetPanel = document.getElementById(`panel-${category}`);
targetPanel.classList.add("active");
```

What This Does

- All panels exist in DOM
- Only visibility changes
- No DOM injection/removal

Why This Is Smart

- Faster than recreating DOM
- Avoids layout shift
- Smooth UX

Close Mechanisms

- Hover leave delay (150ms)
- Click outside detection
- Escape key support

```
if (e.key === "Escape") {
 megaOverlay.classList.remove("active");
}
```

This ensures full keyboard accessibility.

5.1.3 Mobile Menu System

Completely different navigation logic for mobile.

Toggle Logic

mobileMenu.classList.toggle("active", isActive);
document.body.style.overflow = isActive ? "hidden" : "";

What This Does

- Opens sliding mobile menu
- Activates backdrop
- Prevents background scrolling

ARIA Accessibility

mobileToggle.setAttribute("aria-expanded", isActive);

This ensures:

- Screen readers detect open/close state
- Accessibility compliance

Accordion Submenus

parentItem.classList.toggle("expanded", !isExpanded);

Only one submenu remains open at a time.

This improves usability and prevents clutter.

5.1.4 Scroll Reveal Animation

Uses modern **Intersection Observer API**.

Code

```
const revealOnScroll = new IntersectionObserver(
 (entries, observer) => {
   entries.forEach((entry, index) => {
     if (entry.isIntersecting) {
       setTimeout(() => {
         entry.target.classList.add("revealed");
       }, index * 100);
       observer.unobserve(entry.target);
     }
   });
 },
 {
   threshold: 0.15,
 }
);
```

What This Does

- Detects when elements enter viewport
- Adds `.revealed` class
- Applies staggered animation delay

Why It Is Used

- Smooth entrance animations
- No scroll polling
- Highly performant

This is modern frontend engineering.

5.1.5 Lazy Loading Images

```
const lazyImages = document.querySelectorAll("img[data-src]");

img.src = img.dataset.src;
img.removeAttribute("data-src");
```

What This Does

- Loads images only when visible
- Reduces initial page load
- Improves LCP (Largest Contentful Paint)

Performance-focused implementation.

5.1.6 Hero Video Optimization

```
const videoObserver = new IntersectionObserver(
 (entries) => {
   entries.forEach((entry) => {
     if (entry.isIntersecting) {
       heroVideo.play();
     } else {
       heroVideo.pause();
     }
   });
 },
 { threshold: 0.5 }
);
```

What This Does

- Pauses video when not visible
- Saves CPU & battery
- Prevents background playback waste

This shows advanced UX-performance awareness.

5.2 Product Modal System (`product-modal.js`)

This system is fully dynamic and reusable.

5.2.1 Central Data Store

window.GarminProducts = productData;

Why This Matters

- Single source of truth
- Modal and cart both use same data
- No duplication
- Easy product scaling

5.2.2 Event Delegation

Instead of attaching listeners to every card:

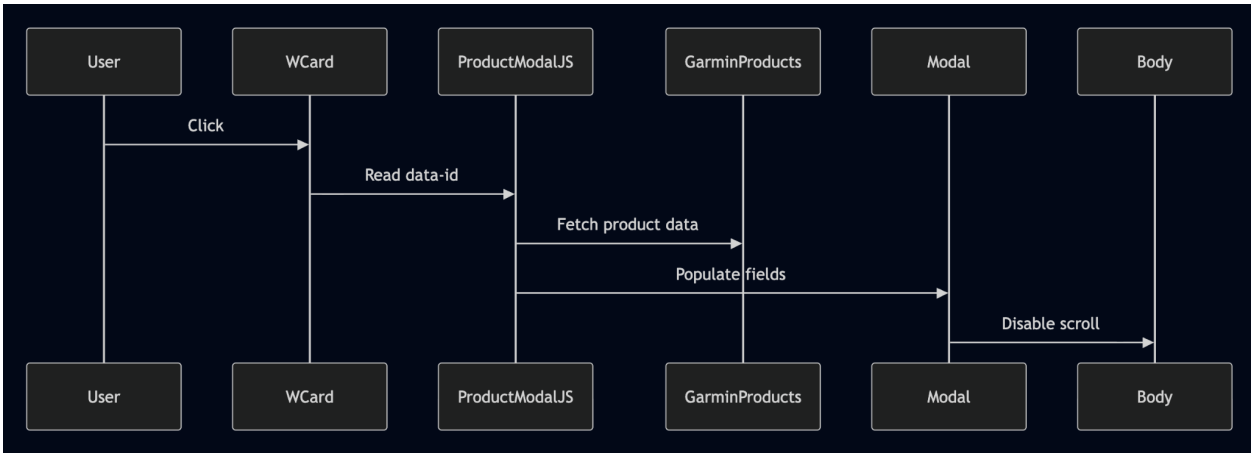
```
document.body.addEventListener("click", (e) => {
 const card = e.target.closest(".w-card");
});
```

Why This Is Better

- Only one listener
- Works for dynamically added cards
- Memory efficient

- Scalable architecture

5.2.3 Modal Lifecycle



Populate Logic

modalTitle.textContent = data.title;
modalPrice.textContent = data.price;

Why This Design Is Good

- Dynamic rendering
- No hardcoded modal HTML
- Easily extensible

5.3 Cart System (`cart.js`)

The cart system uses **localStorage-based persistence**.

5.3.1 Storage Logic

localStorage.setItem("garmin_cart", JSON.stringify(items));

What This Does

- Stores cart data locally
- Persists across page reloads
- Works without backend

5.3.2 Public API Exposure

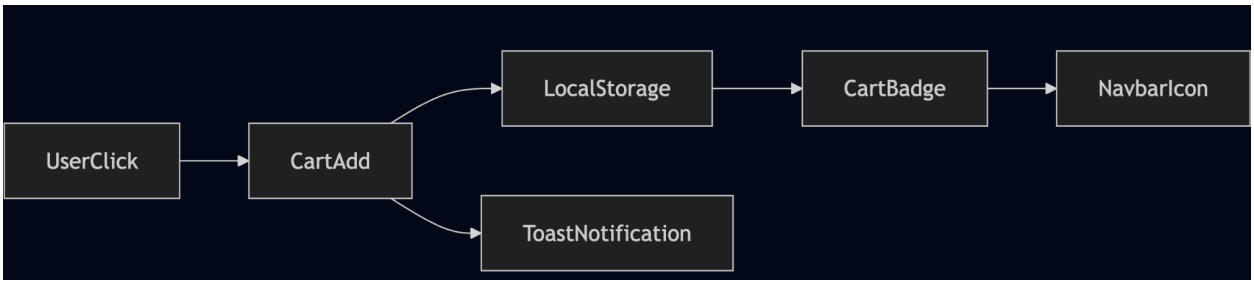
window.GarminCart = Cart;

Available Methods

- `getItems()`
- `getCount()`
- `getTotal()`
- `add()`
- `remove()`
- `updateQty()`
- `updateBadge()`

This is modular API design.

5.3.3 Cart Data Flow



5.3.4 Toast Notification System

Injected dynamically:

toast.classList.add("show");

CSS transition handles animation.

---

6. NAVIGATION LOGIC

Navigation works seamlessly due to three advanced systems.

6.1 Shared Component Injection

`shared-components.js` injects:

- Navbar
- Mega menu
- Mobile menu
- Footer

Ensures consistency across all pages.

6.2 Depth Detection System

HTML:

<html data-depth="1">

JS:

const depth = parseInt(document.documentElement.dataset.depth || "0", 10);

Why This Exists

- Calculates relative paths
- Prevents broken links in nested directories
- Makes system scalable

6.3 Dynamic Link Resolver

```
function resolve(href) {
 return base + "/" + clean + ".html";
}
```

What It Does

- Converts root-style paths to relative paths
- Supports nested folders
- Automatically resolves links

This is production-level routing thinking without a framework.

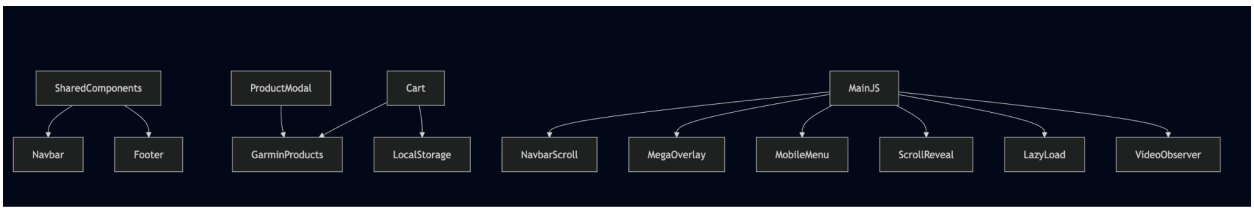
---

7. MODULARITY ACHIEVED

The project achieves modularity through:

- Component injection
- Centralized product database
- Global cart API
- Reusable CSS class naming
- Event delegation
- Separation of layout and behavior
- Performance-optimized observers
- Accessibility enhancements

System Architecture Overview

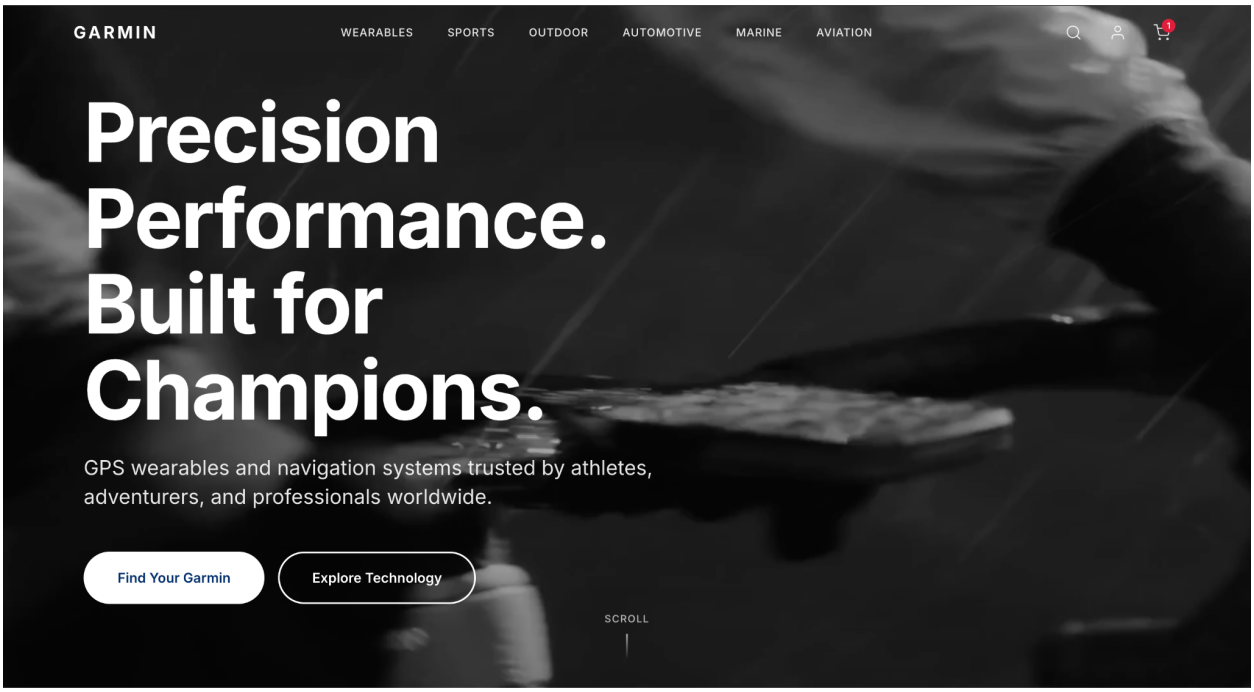


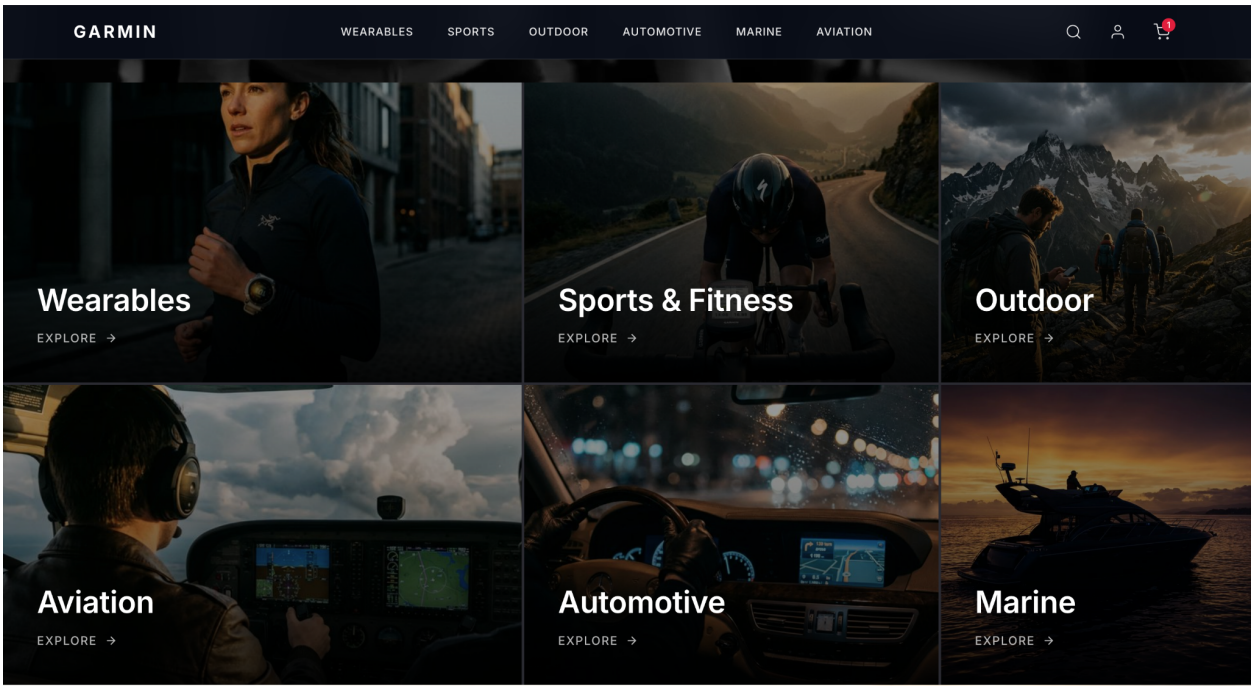
---

8. SCREENSHOTS

You should include:

- Landing page full view


GARMIN
WEARABLES
SPORTS
OUTDOOR
AUTOMOTIVE
MARINE
AVIATION
Precision Performance. Built for Champions.
GPS wearables and navigation systems trusted by athletes, adventurers, and professionals worldwide.
Find Your Garmin
Explore Technology
SCROLL


GARMIN
WEARABLES
SPORTS
OUTDOOR
AUTOMOTIVE
MARINE
AVIATION
Wearables
EXPLORE
Sports & Fitness
EXPLORE
Outdoor
EXPLORE
Aviation
EXPLORE
Automotive
EXPLORE
Marine
EXPLORE

GARMIN

WEARABLES SPORTS OUTDOOR AUTOMOTIVE MARINE AVIATION

01 · FEATURED COLLECTION

Award-Winning Wearables

Discover the power of precision GPS and advanced health monitoring to support your performance journey.

your performance journey.

NEW

Fēnix 8

Ultimate adventure smartwatch with AMOLED display and LED flashlight.

TOP PICK

Forerunner 965

Premium running smartwatch with colorful AMOLED display.

SMART & HEALTH

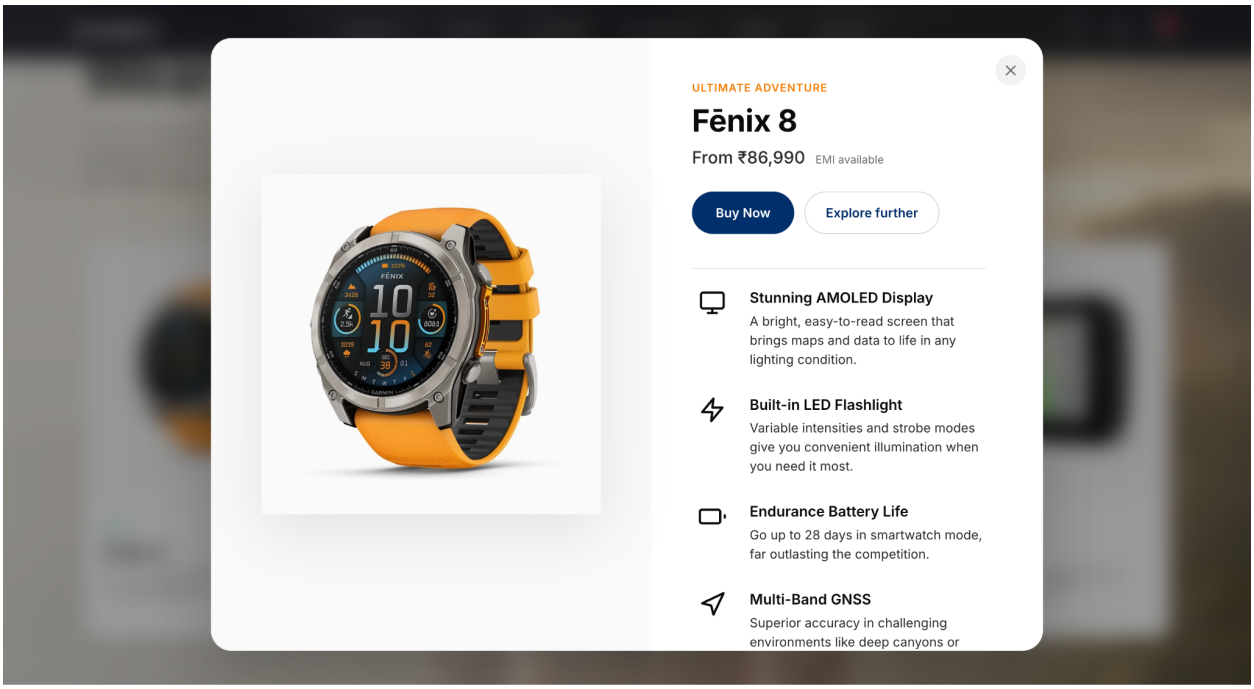
Venu 3

Know the real you with advanced health and fitness features.

CYCLING

Edge 1050

Ultimate cycling computer with the largest Edge display ever.



35+
YEARS OF INNOVATION

#1
GPS ACCURACY

90
DAYS BATTERY LIFE

300M+
DEVICES WORLDWIDE

GARMIN

WEARABLES SPORTS OUTDOOR AUTOMOTIVE MARINE AVIATION

02 · OFF-ROAD COLLECTION

Conquer Every Trail

Navigate the uncharted with rugged GPS navigators and trackers built for the most demanding off-road adventures.

ALL-TERRAIN

Tread® XL

10" all-terrain navigator built for

ALL-TERRAIN

Tread® 2

8" all-terrain navigator with group

RUGGED

Overlander®

Rugged GPS navigator with

HANDHELD GPS

Montana® 700

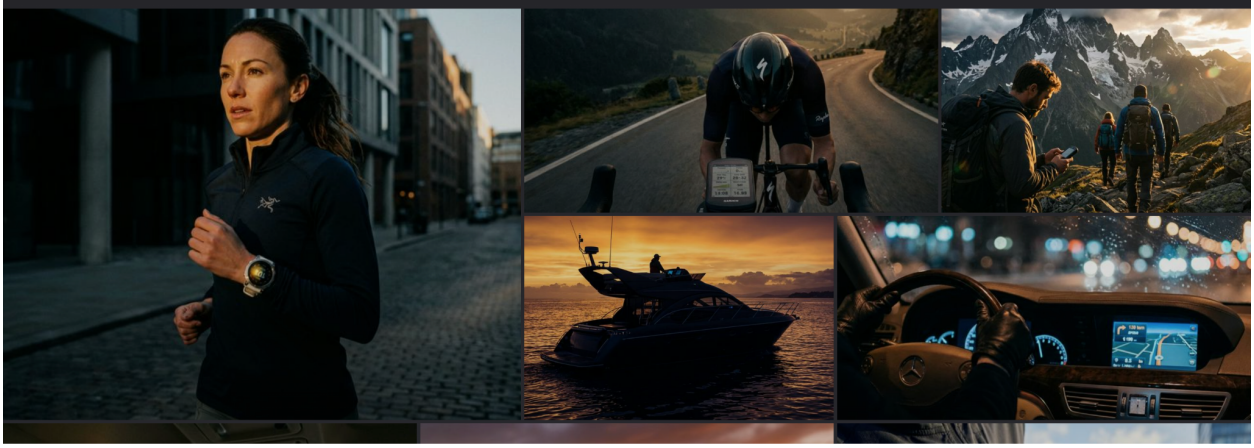
5" touchscreen handheld GPS with

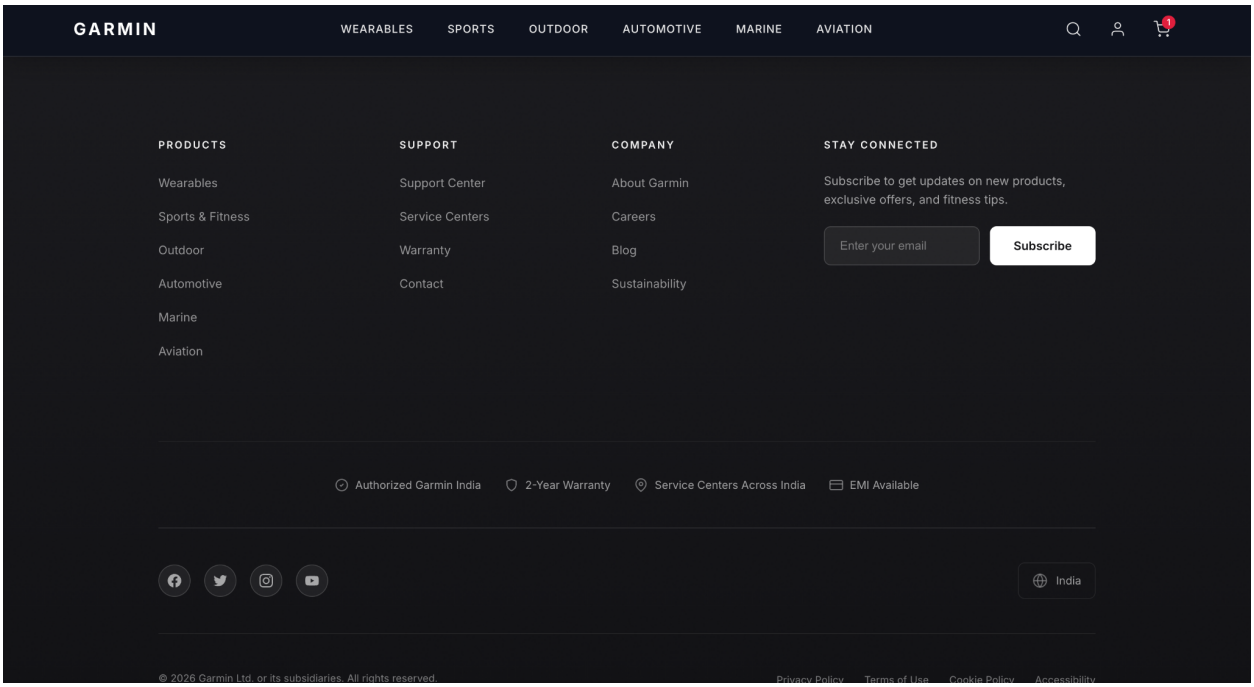
GARMIN

WEARABLES SPORTS OUTDOOR AUTOMOTIVE MARINE AVIATION

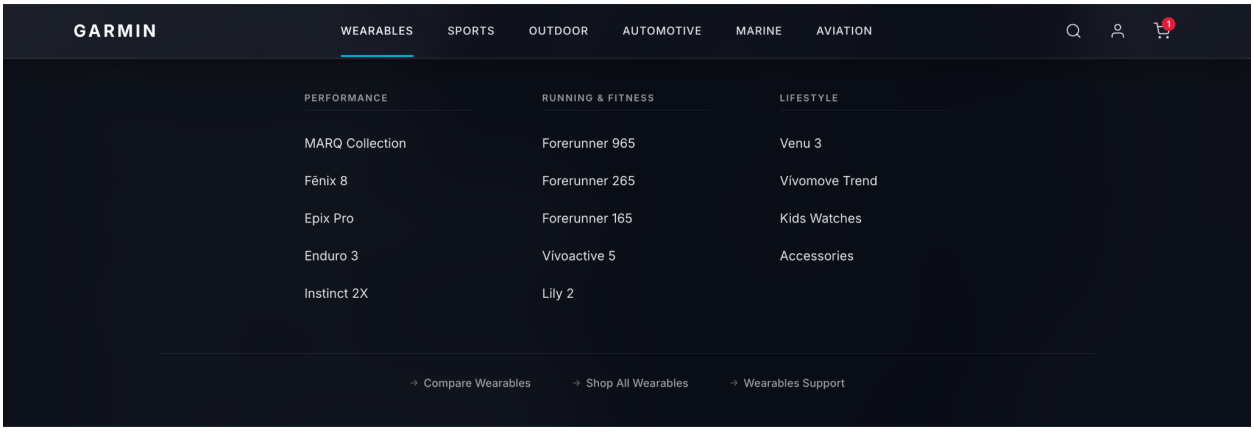
You'll Never Train Alone

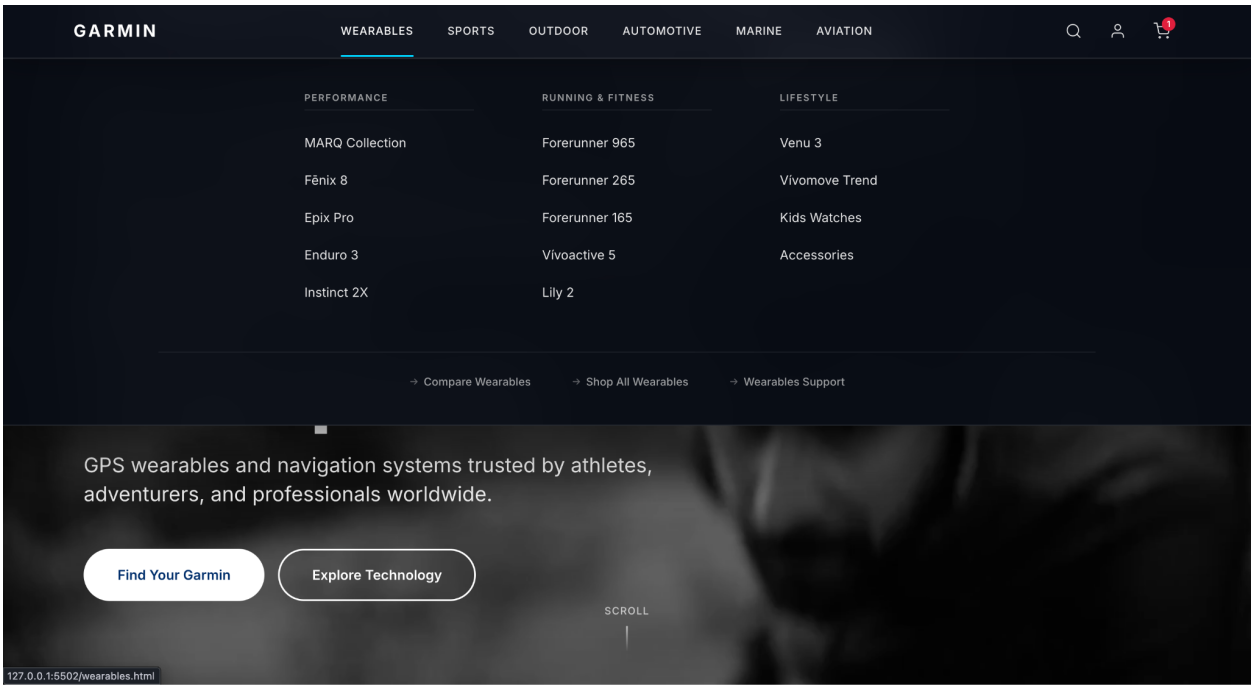
Join millions of athletes, adventurers, and professionals who trust Garmin to push their limits.



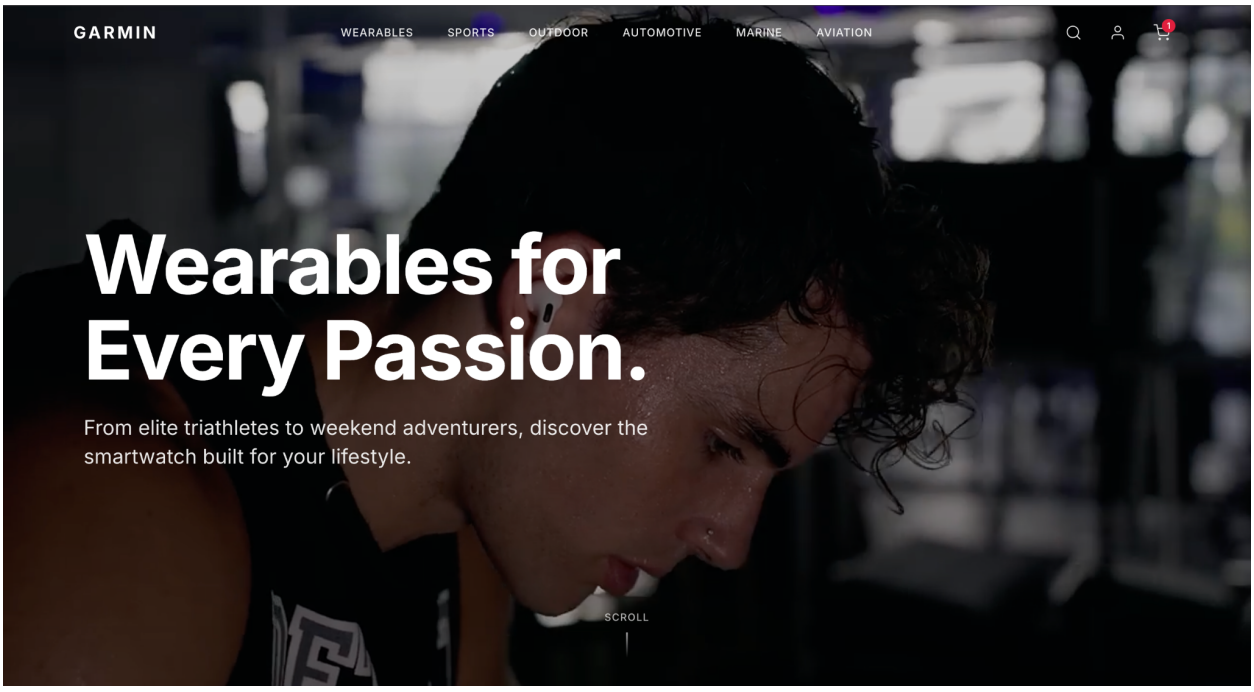


- Mega menu open state





- Wearables section scroll



GARMIN

WEARABLES SPORTS OUTDOOR AUTOMOTIVE MARINE AVIATION

Performance Watches.

Shop all Performance →

NEW

Fēnix 8

Ultimate adventure smartwatch with AMOLED display and LED flashlight.

BEST SELLER

Epix Pro

High-performance GPS watch with stunning AMOLED display.

ULTRA ENDURANCE

Enduro 2

Extreme battery life for the longest races.

RUGGED

Instinct 2X

Built tough with unlimited battery life in smartwatch mode.

GARMIN

WEARABLES SPORTS OUTDOOR AUTOMOTIVE MARINE AVIATION

Running & Fitness.

Shop all Running →

TOP PICK

Forerunner 965

Premium running smartwatch with colorful AMOLED display.

ADVANCED

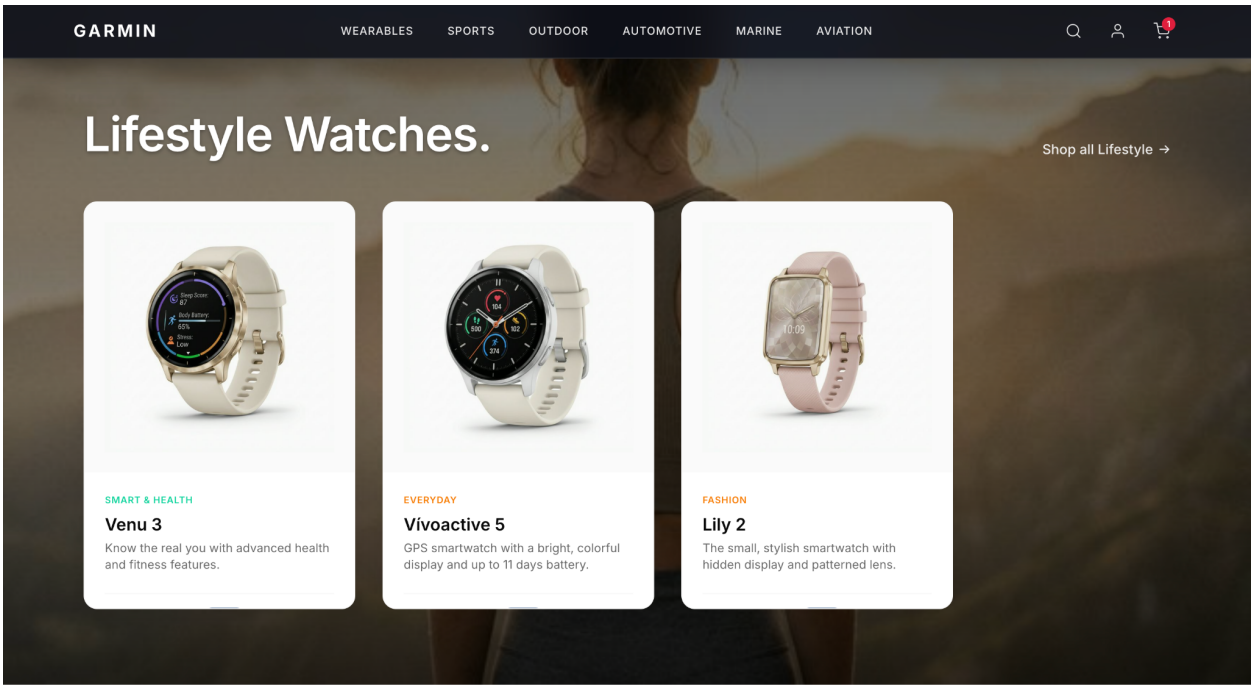
Forerunner 265

Train brilliantly with an AMOLED display and training metrics.

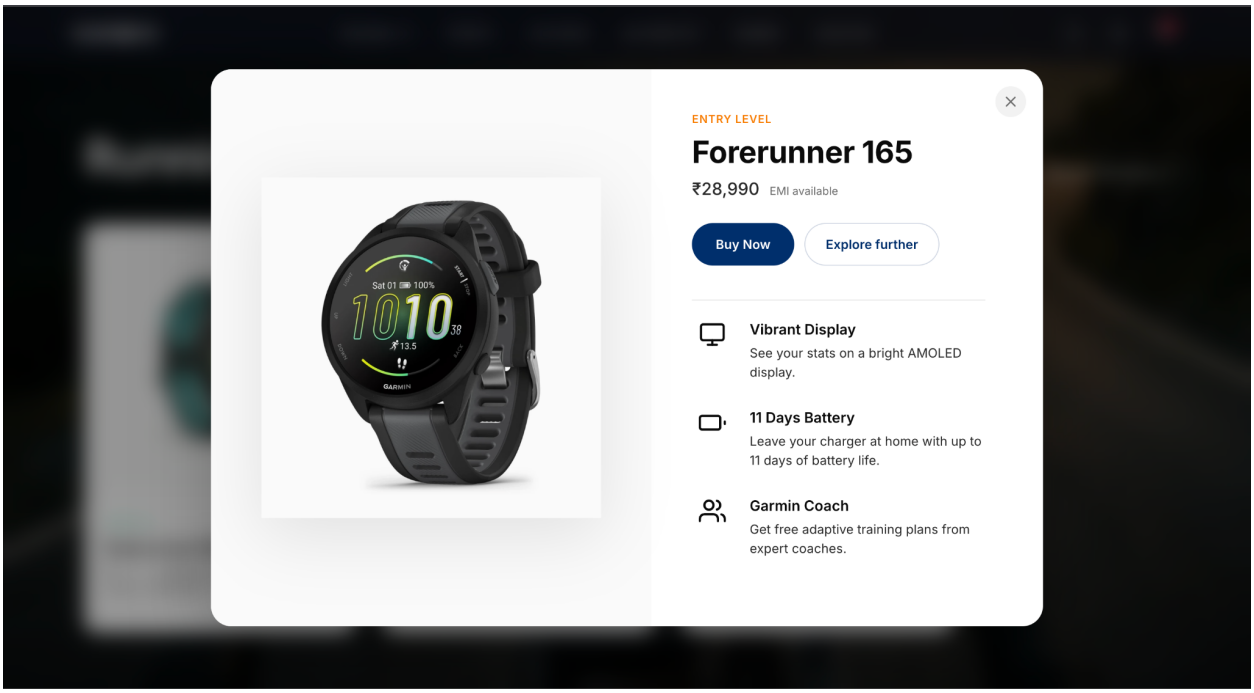
ENTRY LEVEL

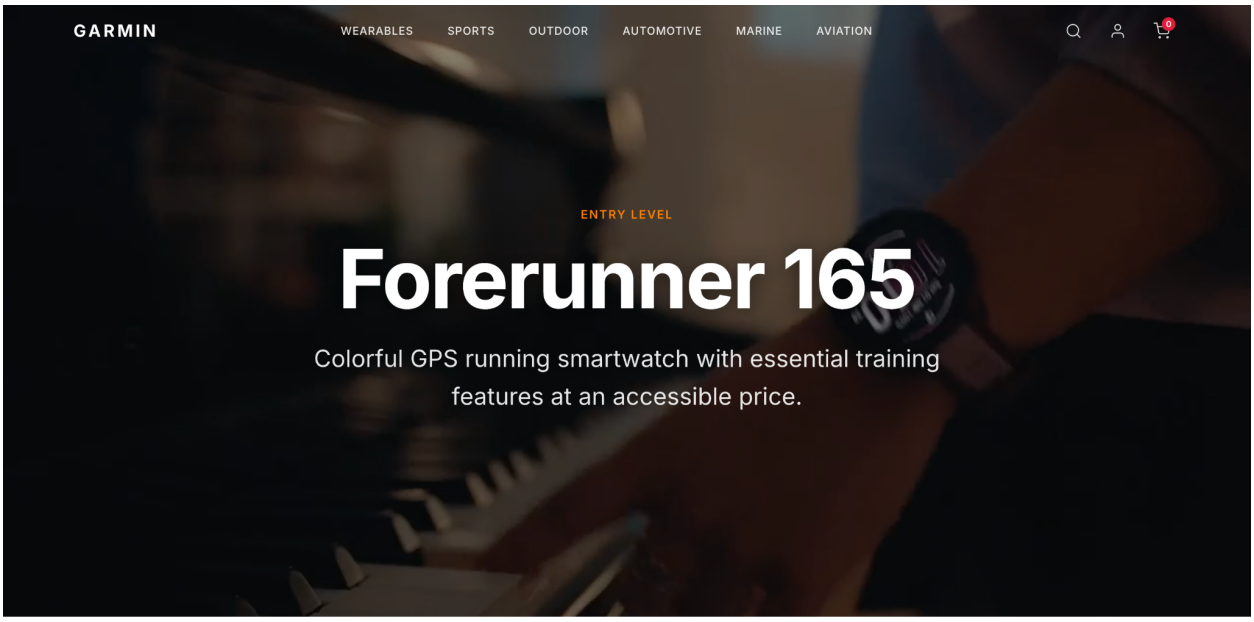
Forerunner 165

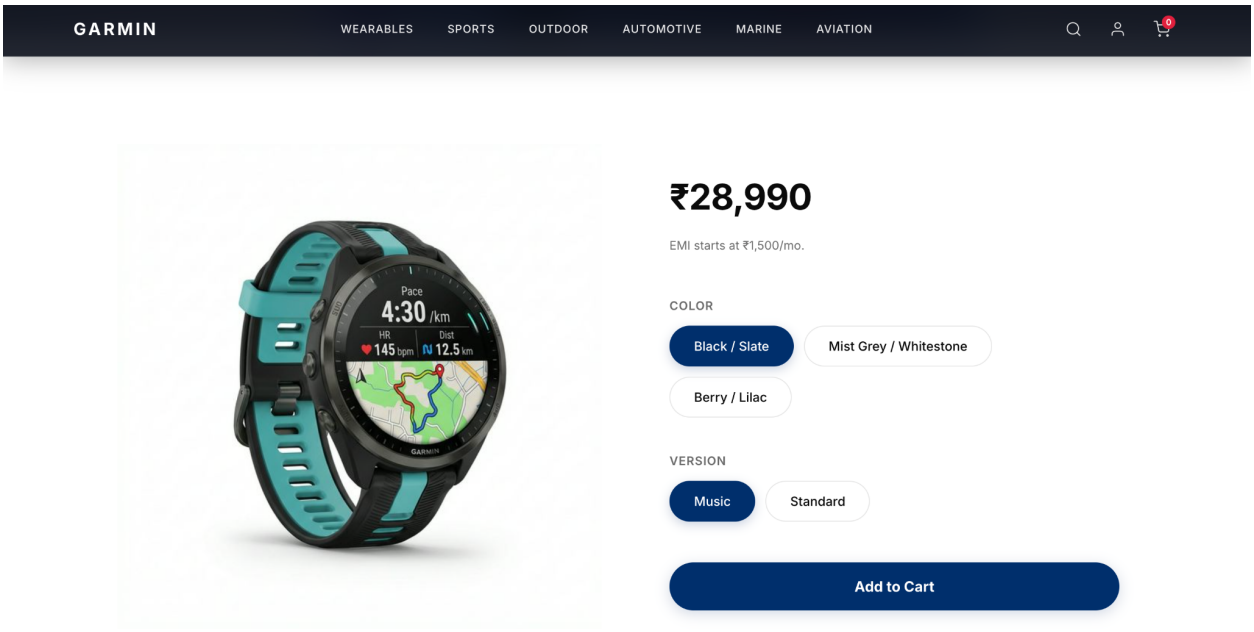
Purpose-built GPS running smartwatch with essential features.



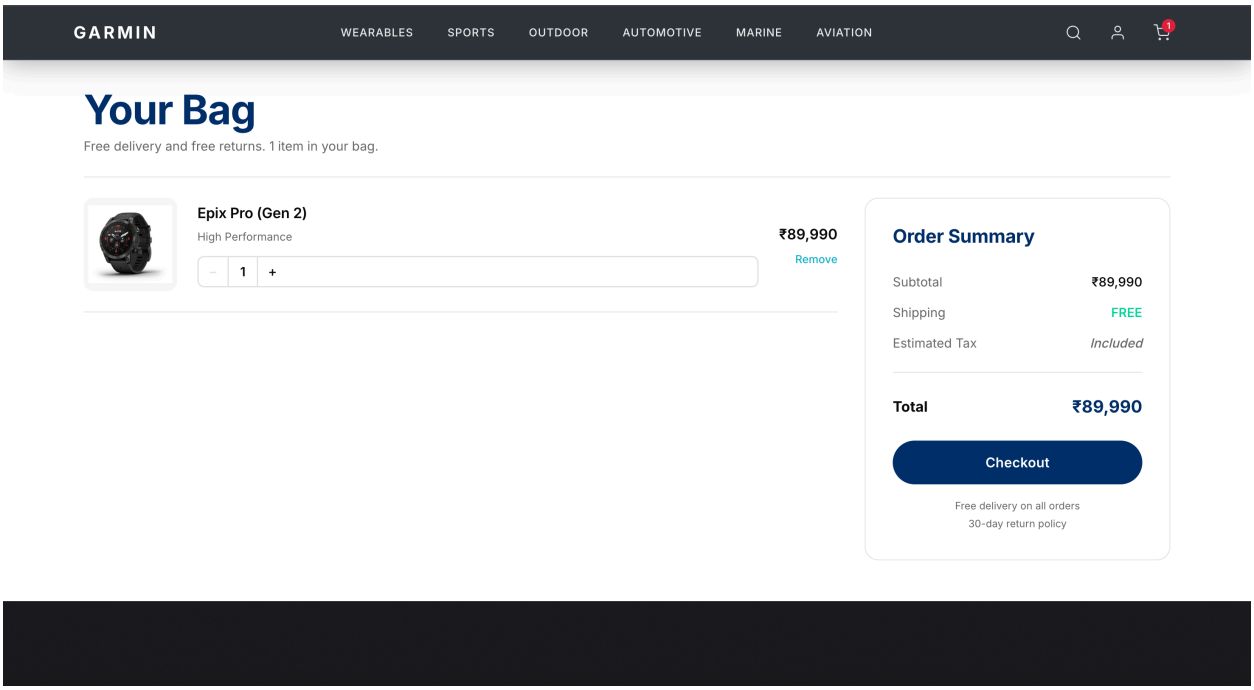
- Product modal open







- Cart toast notification



9. FUTURE SCOPE

- Backend integration (Node.js + MongoDB)
- Payment gateway integration
- User authentication
- Server-side rendering
- Product filtering & sorting
- PWA implementation
- Accessibility audit compliance

10. DECLARATION

I hereby declare that this project entitled
"Garmin India – Modular E-Commerce Website"
is my original work carried out by me. The architecture, styling, and logic have been designed and implemented independently as part of academic practical work.

Name of Student: Aaryan kuchekar
Roll Number: 048
Date: **17/02/2026**