Frontend Interaction Features Specification

Overview

This document outlines the JavaScript functionality, implementation approach, and team task division for interactive UI features: - Smooth scrolling - CTA click feedback - Sticky navigation - Scroll-triggered animations

---

2. Smooth Scrolling for Navigation Links

Purpose: Provide animated scrolling when users click navigation links instead of jumping instantly.

Recommended APIs / Methods - `Element.scrollIntoView()` - `window.scrollTo()` - Event listeners

Implementation Pattern - Attach click listeners to nav links - Prevent default jump - Scroll smoothly to target section

Key Concept: Navigation links should trigger scripted scroll behavior rather than default anchor behavior.

---

3. CTA Button Click Feedback

Purpose: Give users visual confirmation that a button was pressed.

Recommended APIs / Methods - `addEventListener("click")` - `classList.add/remove/toggle` - `setTimeout()`

Possible Effects - Ripple animation - Scale press effect - Temporary color change - Loading state

Key Concept: Interaction should feel responsive and immediate.

---

4. Sticky Navigation on Scroll

Purpose: Keep navigation visible while scrolling.

Recommended APIs / Methods - `window.addEventListener("scroll")` - `window.scrollY` - `classList.toggle()`

Logic Flow - Detect scroll position - Add sticky class after threshold - Remove when above threshold

Modern Alternative (Preferred) - `IntersectionObserver` watching a sentinel element

Key Concept: Sticky behavior should activate only when necessary to avoid unnecessary processing.

---

5. Scroll-Based Class Toggles (Animations)

Purpose: Trigger animations when elements enter the viewport.

Recommended API - `IntersectionObserver` (preferred over scroll events)

Why Use It - High performance - Runs only when visibility changes - Avoids constant scroll calculations

Implementation Pattern - Observe hidden elements - Add visible class when intersecting

Key Concept: Animations should only run when visible to users.

Suggested Project Structure

```
/js
   nav.js
   interactions.js
/css
   nav.css
   animations.css
```

Workload Division (2 Developers)

Developer A — Navigation Logic

Responsible for: - Smooth scrolling - Sticky navbar - Anchor link behavior - Navbar styling classes - Accessibility attributes

Focus Area: Structure + navigation systems

Developer B — Interaction & Motion

Responsible for: - CTA feedback - Scroll animations - Animation timing - Motion polish - Performance tuning

Focus Area: Visual feedback + animation

Shared Class Naming Contract

Both developers must agree on class names before implementation:

```
.hidden   → element before entering viewport
.show     → element after entering viewport
.sticky   → applied to navbar when fixed
.clicked  → applied to CTA on press
```

Reason: Prevents merge conflicts and logic mismatches.

Recommended Stack

- Vanilla JavaScript
- CSS transitions
- IntersectionObserver API

Frameworks are unnecessary unless animation complexity increases significantly.

Implementation Strategy (Recommended Order)

1. Build scroll observer system
2. Reuse observer logic for sticky nav
3. Add CTA feedback using same class toggle approach

Goal: One reusable logic system powering multiple features.

Development Principle

> Build reusable behavior systems rather than isolated feature scripts.

This approach improves maintainability, scalability, and performance.