Project Title: Product Catalog and Shopping Cart System using Angular and TypeScript

Course: BTOE461T5P – Advanced JavaScript Frontend Frameworks

Team Members:
Udit R Singh (2462165),
Samuel Biju (2462141),
Anson Mathew Allan (2462043),
Pavithra Alexander (2462123)

Date: January 23, 2026

---

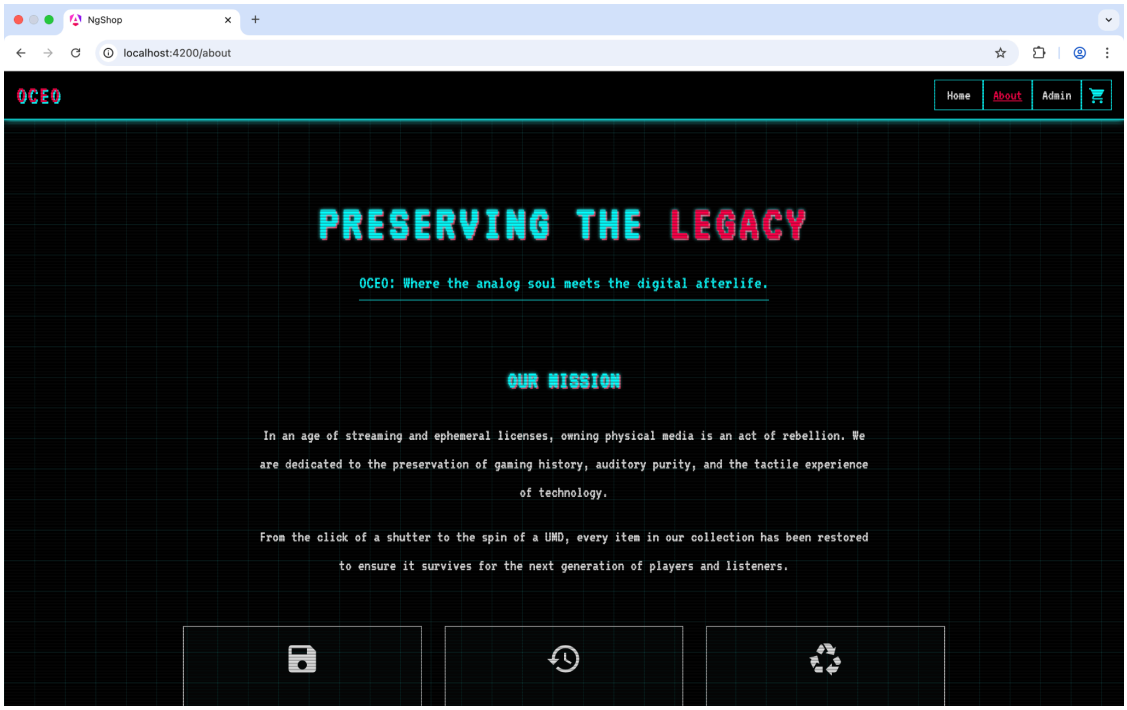
1. Introduction

For this assignment, our team developed **OCEO**, an e-commerce Single Page Application (SPA) designed to serve as a digital storefront for retro technology and physical media. Our primary objective was to explore the intersection of cutting-edge frontend architecture—specifically the new features in Angular 19—with a nostalgic, "retro-futurist" aesthetic. We wanted to build something that felt like a technological archive from the late 90s or early 2000s but performed like a modern progressive web app.

2. Design Philosophy: Analog Soul, Digital Afterlife

The most distinct feature of OCEO is its visual identity. We collectively decided to move away from the clean, white minimalism that dominates modern e-commerce. Instead, we adopted a **brutalist UI approach**.

- **Color Palette:** We utilized a high-contrast dark mode scheme—pitch black backgrounds (#000) with neon green and hot pink accent borders. This mimics the aesthetic of early terminal interfaces and cyberpunk media.
- **Typography:** The font choice is monospaced and blocky, reinforcing the feeling of interacting with a legacy computer system or a game console BIOS.



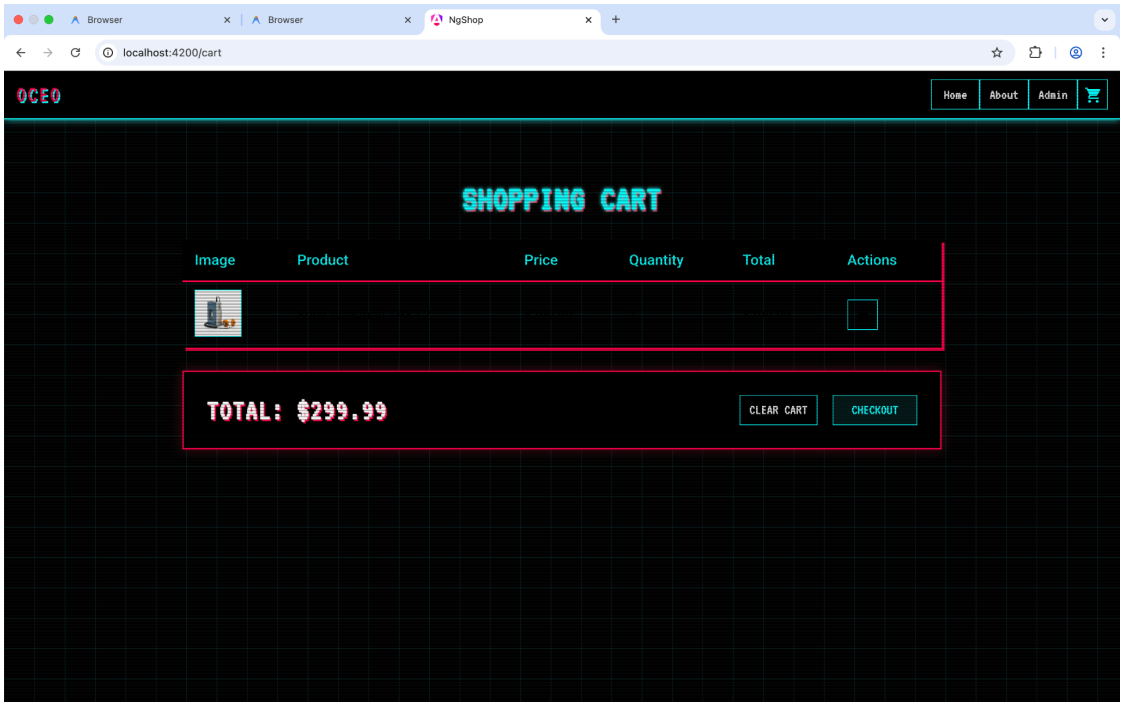
The "About" page frames the site not just as a shop, but as a preservation effort ("Where the analog soul meets the digital afterlife"), giving the user a narrative context for our design choices.

3. Technical Implementation: Angular Signals

On the technical side, we utilized **Angular 19**, specifically focusing on **Signals** for state management. This was a shift from the traditional RxJS BehaviorSubject approach we have used in previous coursework.

The most practical application of this was in the **Shopping Cart** logic:

1. We implemented a CartService that holds a writable signal for the list of items.
2. The "Total Price" in the cart view is a computed signal. This ensures the UI updates reactively and instantly the moment an item is added or removed, without us needing to manually subscribe to streams or trigger change detection cycles.



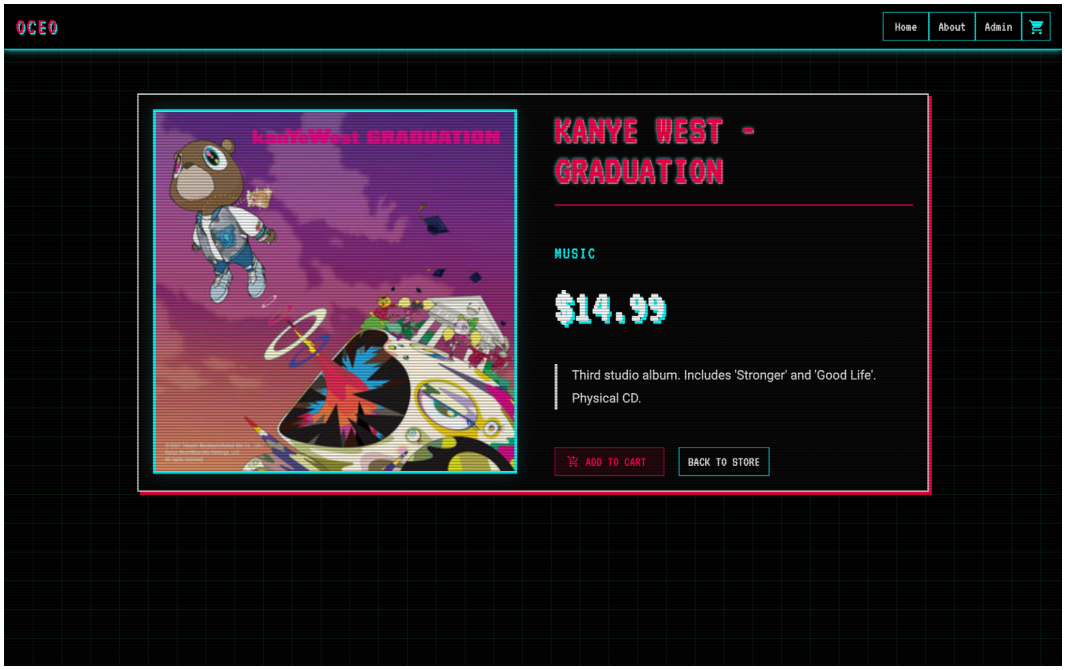
4. User Journey Walkthrough

To demonstrate the application flow, we designed the user journey to feel like browsing a physical record store.

Step 1: The Archive (Landing) The user lands on the main catalog. The sidebar navigation allows for immediate filtering by category.

[INSERT SCREENSHOT 3 HERE: The main Homepage showing the grid of products] *Figure 3: The main product archive featuring a grid layout and sidebar navigation.*

Step 2: Filtering and Selection If a user is looking specifically for audio media, they click the "Music" category. Clicking on an item, such as **Kanye West's *Graduation*** or the **Sony Walkman**, routes the user to a dedicated 'Details' view. We designed this page to isolate the product with a large hero image and specific metadata.



Step 3: Checkout Navigating to the Cart page shows a tabular breakdown of the order (Image, Product, Price). The "Checkout" button simulates the purchase process, clearing the signal state and resetting the cart.

5. Future Development

While our core functionality is stable, we have identified two key areas for the next iteration:

- **Asset Optimization:** Currently, we are serving product images locally. We plan to migrate these to a cloud storage solution (like AWS S3 or Firebase Storage) to improve load times and reduce the bundle size.
- **Database Expansion:** The current product set is a static JSON mock. Our next step is connecting this to a real backend (Node.js/Express) to allow for dynamic inventory management, so the "Out of Stock" logic can be fully implemented.
