eProject Report: Alberto Watch Company Website

Acknowledgements

We would like to express our sincere gratitude to:

- React and Vite development teams for providing excellent frameworks
- The open-source community for various libraries and tools
- All contributors and testers who provided valuable feedback during development

Problem Definition

Alberto Watch Company, a full-service watch repair, appraisal, and retail store specializing in luxury timepieces, requires a modern, responsive Single-Page-Application (SPA) website to showcase their extensive collection of vintage, luxury, and smart watches. The website must provide an intuitive user experience with product categorization, detailed modal views, interactive gallery, store locator with geolocation, technology showcase, and customer support features, while incorporating visitor tracking, scrolling ticker with real-time information, and seamless navigation across all devices and major browsers.

---

eProject Synopsis

Project Name: Alberto Watch Company Website
Technology Stack: React 19.2.0, Vite 7.2.2, React Router DOM 7.9.5
Project Type: Single Page Application (SPA) E-commerce Website
Development Period: 2025
Prepared By: Osayemi Olamilekan Daniel
Student Number: 1638566

Key Features:

- Product catalog with 36+ watches across 3 categories
- Interactive photo gallery with modal lightbox
- Store locator with geolocation integration
- Technology showcase
- Responsive design with mobile-first approach
- 404 error handling
- Visitor counter with localStorage

- Smooth scroll navigation

Project Requirements

Index

Introduction

The thirst for learning, upgrading technical skills and applying the concepts in real life environment at a fast pace is what the industry demands from IT professionals today. However busy work schedules, far-flung locations, unavailability of convenient time-slots pose as major barriers when it comes to applying the concepts into realism. And hence the need to look out for alternative means of implementation in the form of laddered approach.

The above truly pose as constraints especially for our students too! With their busy schedules, it is indeed difficult for our students to keep up with the genuine and constant need for integrated application which can be seen live especially so in the field of IT education where technology can change on the spur of a moment. Well, technology does come to our rescue at such times!!

Keeping the above in mind and in tune with our constant endeavour to use Technology in our training model, we at Aptech have thought of revolutionizing the way our students learn and implement the concepts using tools themselves by providing a live and synchronous eProject learning environment!

So what is this eProject?

eProject is a step-by-step learning environment that closely simulates the classroom and Lab based learning environment into actual implementation. It is a project implementation at your fingertips!! An electronic, live juncture on the machine that allows you to

- Practice step by step i.e. laddered approach.
- Build a larger more robust application.
- Usage of certain utilities in applications designed by user.
- Single program to unified code leading to a complete application.
- Learn implementation of concepts in a phased manner.
- Enhance skills and add value.
- Work on real life projects.
- Give a real life scenario and help to create applications more complicated and useful.
- Mentoring through email support.

The students at the centre are expected to complete this eProject and send complete project along with the documentation to eProjects Team

Looking forward to a positive response from your end!!

Objectives of the project

The Objective of this program is to give a sample project to work on real life projects. These applications help you build a larger more robust application.

The objective is not to teach you JSON/JavaScript/Dreamweaver/HTML5 but to provide you with a real life scenario and help you create basic applications using the tools.

You can revise the chapters before you start with the project.

This project is meant for students who have completed the module of HTML5. These programs should be done in the Lab sessions with assistance of the faculty if required.

It is very essential that a student has a clear understanding of the subject. Students should go through the project and solve the assignments as per requirements given.

Kindly get back to eProjects Team in case of any doubts regarding the application or its objectives.

Problem Statement

Alberto Watch Company is a full-service watch repair, appraisal, and retail store – offering the finest selection of luxury timepieces. Offering an unparalleled selection with first-class service.
Alberto also carries a wide selection of brand new watches such as Michael Kors, Rolex, Citizen Eco-Drive, Bulova and more!

You are supposed to create a Single-Page-Application and responsive Website for them with the below mentioned requirement specifications.

The website should work well in all leading browsers including Chrome, IE, Firefox etc.

Functional Requirement Specification:

The portal will be designed as a Single-Page-Application and responsive Website with a set of pages and menus that represent choice of activities to be performed. The pages, menus, and other visual elements must be designed in a visually appealing manner with attractive fonts, colors, and animations.

All of these should also be laid out in a responsive manner

The Web site is to be created based on the following requirements.

- The Top of the Page should be presented with a suitable logo, and page should have decent colour combination.
- The site should display a menu which will contain the options as Products, Technology, Store Locator, Support etc.
- Various sections having information about different types of watches and clocks, Technology used in the watches etc. these sections should display products accordingly.
- The information should be categorized in sections according to different Product Line-up e.g. "Vintage", "Luxury", "Smart Watches" etc.
- When a user selects any product line-up, complete list of watches for that product line-up will be displayed.
- Details of the watches should be displayed on the popup window.
- Price List for all watches should be available.
- Gallery can be added for viewing different images.
- Site map, Gallery, about us, Contact us link must be added.
- About Us and Contact Us: This menu option should display Email id, address, and contact number of the organization.

Over and above this, the portal should implement the following functionalities:

- Display a continuous scrolling ticker at the bottom of the page with current date, time, and location (hint: Use geolocation features of HTML5).
- Display a visitor count at the top right corner of the page beside a logo image.
- The menu options should change color on hover and also after clicking.
- Fade in and fade out options can be used for the menus.

Hardware/ Software Requirements

Hardware

- Intel Core i3/i5 Processor or higher
- 8 GB RAM or above
- Color SVGA
- 500 GB Hard Disk space
- Mouse
- Keyboard

Software

- Technologies to be used:
    - Frontend: HTML5, CSS, Bootstrap, JavaScript, jQuery, React/AngularJS, Figma, XML
    - Data Store: JSON files or TXT files

Other Requirements:

- Operating Portal: Windows
- Browsers: Edge, Chrome, Mozilla Firefox, Safari

Project Deliverables
You will design and build the project and submit it along with a complete project report that includes:

- Problem Definition
- Design specifications
- Diagrams such as flowcharts for various activities, Data Flow Diagrams etc.
- Source Code
- Test Data Used in the Project
- Project Installation Instructions (if any)

Documentation is considered as a very important part of the project. Ensure that documentation is complete and comprehensive. The consolidated project will be submitted as a zip file with a ReadMe.doc file listing assumptions (if any) made.
Submit a video clip demonstrating the working of the Website. Optionally, a live hosted URL can be supplied for the site. Over and above the given specifications, you can apply your creativity and logic to improve the portal.

---

eProject Analysis

Problem Definition

Alberto Watch Company, a full-service watch repair, appraisal, and retail store specializing in luxury timepieces, requires a modern, responsive Single-Page-Application (SPA) website to showcase their extensive collection of vintage, luxury, and smart watches. The website must provide an intuitive user experience with product categorization, detailed modal views, interactive gallery, store locator with geolocation, technology showcase, and customer support features, while incorporating visitor tracking, scrolling ticker with real-time information, and seamless navigation across all devices and major browsers.

System Requirements

- **Frontend Framework:** React 19.2.0
- **Build Tool:** Vite 7.2.2
- **Routing:** React Router DOM 7.9.5
- **Browser Support:** Modern browsers (Chrome, Firefox, Safari, Edge)
- **Responsive Design:** Mobile, Tablet, Desktop

Functional Requirements

1. Display product catalog with filtering capabilities
2. Show detailed product information in modal
3. Gallery with category filtering

4. Store location finder with user geolocation
5. Contact form and company information
6. Technology and features showcase
7. Responsive navigation menu
8. 404 page for invalid routes

Non-Functional Requirements

- Fast page load times (< 3 seconds)
- Mobile-responsive design
- Cross-browser compatibility
- SEO-friendly structure
- Accessible UI components

---

eProject Design

System Architecture

The application follows a component-based architecture using React:

App.jsx (Root Component)

├── Header Component (Navigation)

├── Main Content Sections

│ ├── Home Page

│ ├── Products Page (with Modal)

│ ├── Technology Page

│ ├── Store Locator Page

│ ├── Support Page

│ ├── Gallery Page

│ ├── About Page

│ ├── Contact Page

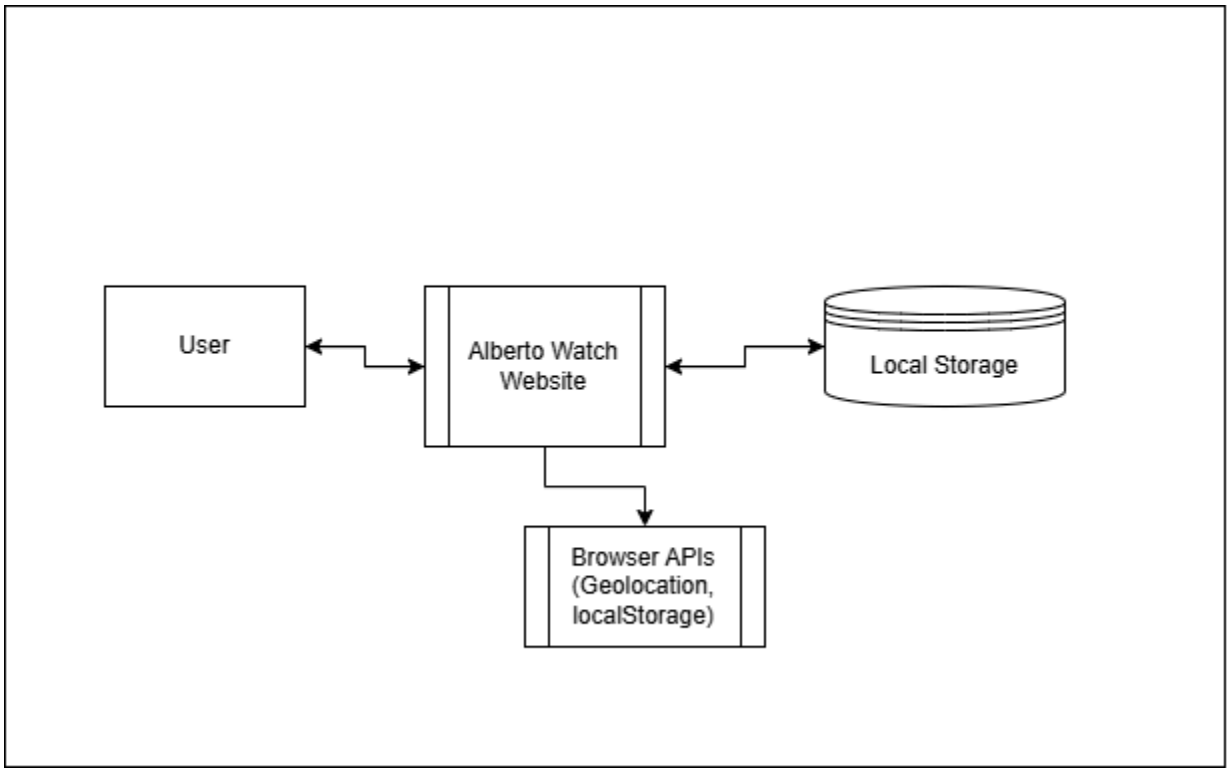
│ └── Sitemap Page

├── NotFound Page (404)

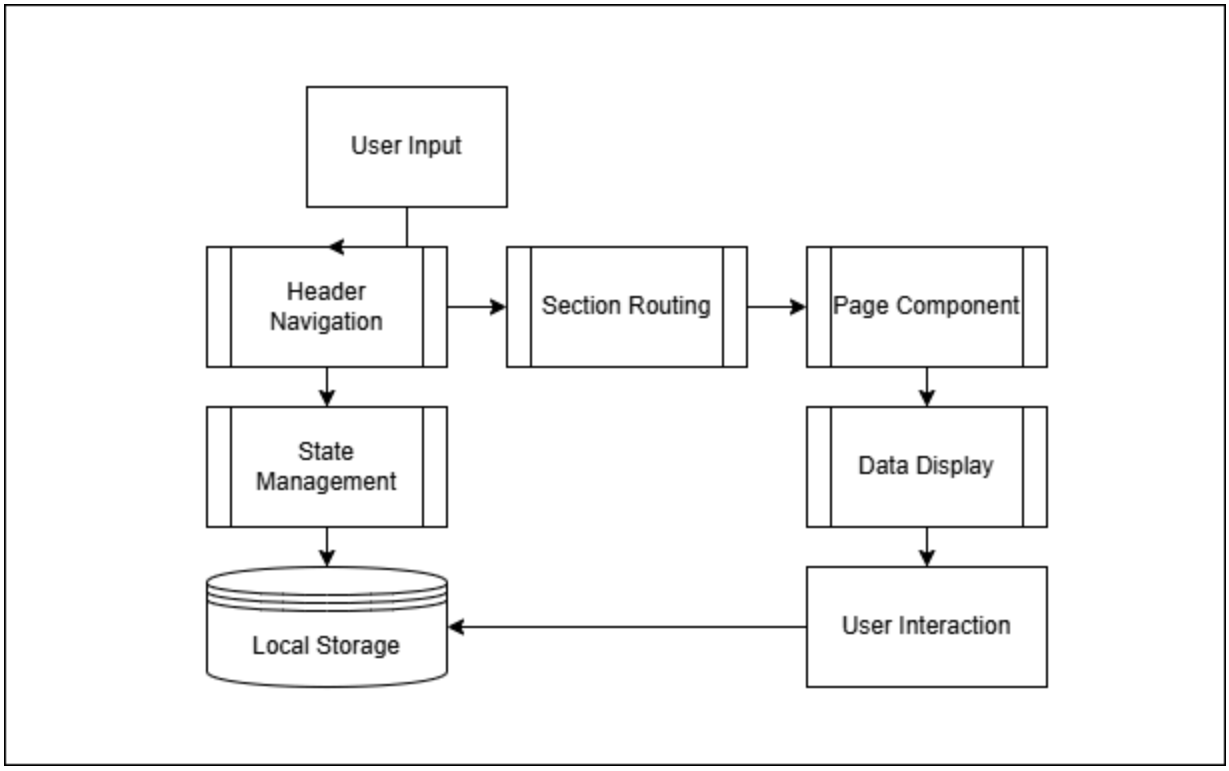
└── Ticker Component (Footer)

Data Flow Diagram (DFD)

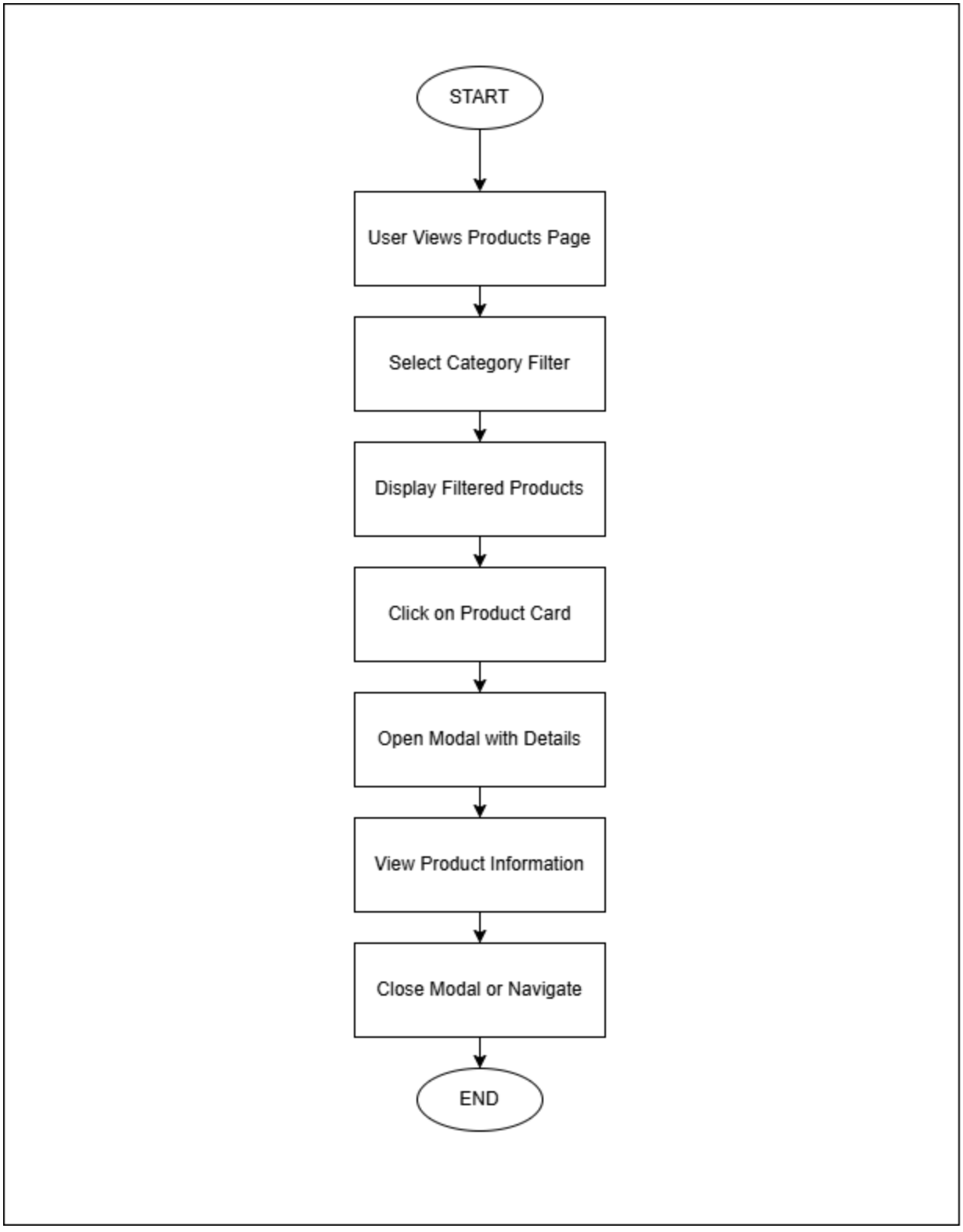
Level 0 - Context Diagram



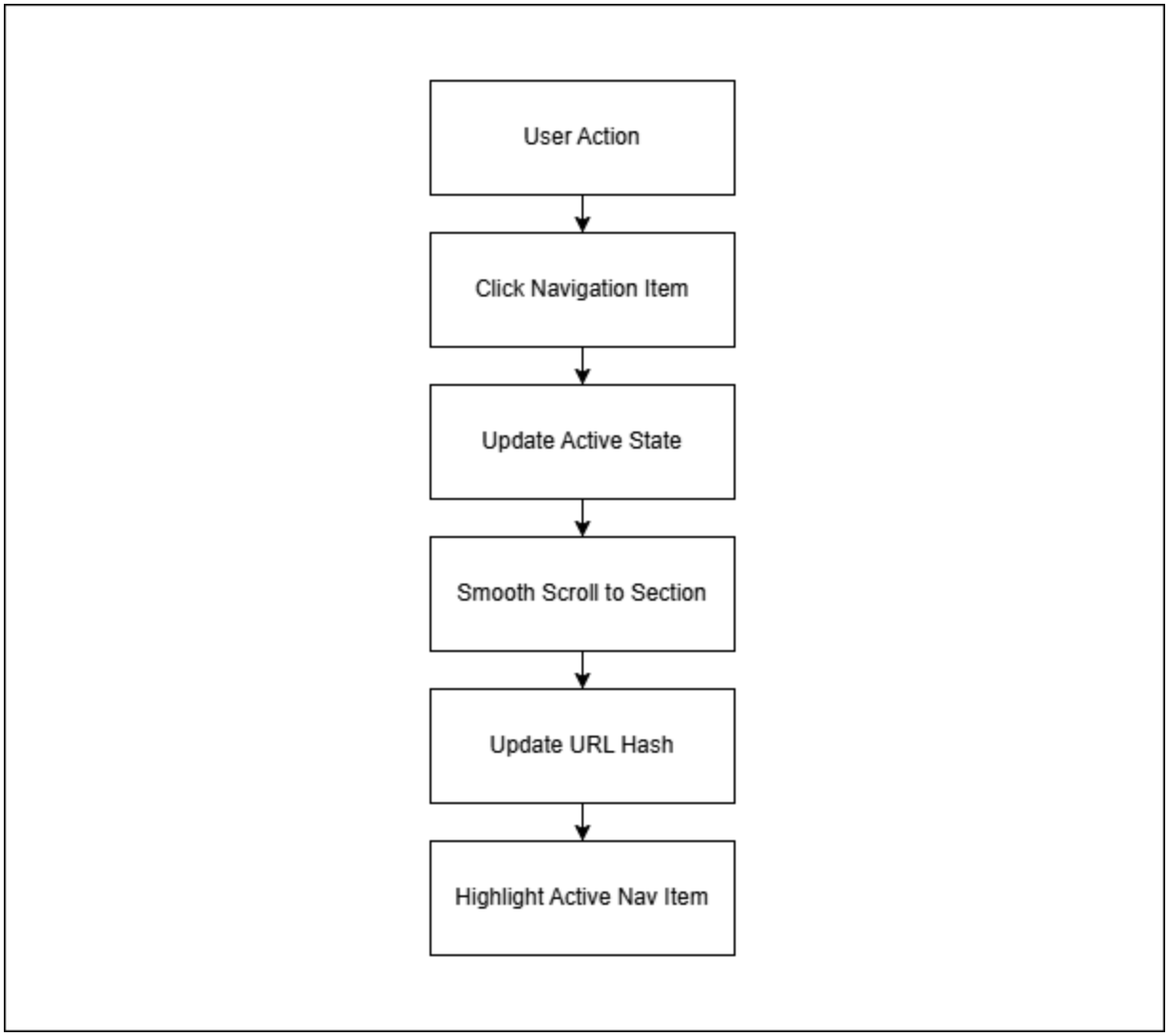
Level 1 - Main Processes



Flowchart: Product Selection Process



Process Diagram: Navigation Flow



Database Design / Structure

Note: This is a frontend-only application using local data storage.

Data Structure:

```
watches = {
  vintage: [
    {
      id: Number,
      name: String,
```

```
    price: String,
    img: String (path),
    desc: String,
    properties: {
      waterResistance: String,
      movement: String,
      caseMaterial: String,
      crystal: String
    }
  }
 ],
 luxury: [...],
 smart: [...]
}
```

Local Storage Structure:

- visitorsCount: Number (stored as string, converted to number)

Image Storage:

- Location: /public/images/
- Format: JPG, PNG, AVIF
- Total Images: 36 product images + assets

Design Specifications

UI/UX Design

- **Colors:** White/Light Gray backgrounds, Dark Gray text, Gold accents

- **Typography:** Playfair Display (headings), Montserrat (body)
- **Layout:** Max-width 1200px, CSS Grid/Flexbox, responsive breakpoints (Mobile: <768px, Tablet: 768-1024px, Desktop: >1024px)
- **Components:** Fixed header (80px), Product grid (3/2/1 columns), Modal overlay, Bottom ticker (40px)
- **Interactions:** Smooth scrolling, hover effects, category filtering, mobile hamburger menu
- **Accessibility:** ARIA labels, keyboard navigation, WCAG AA contrast, semantic HTML
- **Performance:** <3s load time, lazy loading, code splitting, optimized bundle

Data Structure

Product Schema:

```
{

  id: Number, name: String, price: String, img: String, desc: String,

  properties: { waterResistance, movement, caseMaterial, crystal }

}
```

Categories: vintage (10 items), luxury (14 items), smart (12 items)
Local Storage: visitorsCount (String, converted to Number)

State Management

- React Hooks (useState, useEffect) for local component state
- Key states: visitorCount, selectedCategory, selectedProduct, isMobileOpen, activeNavItem

API Integration

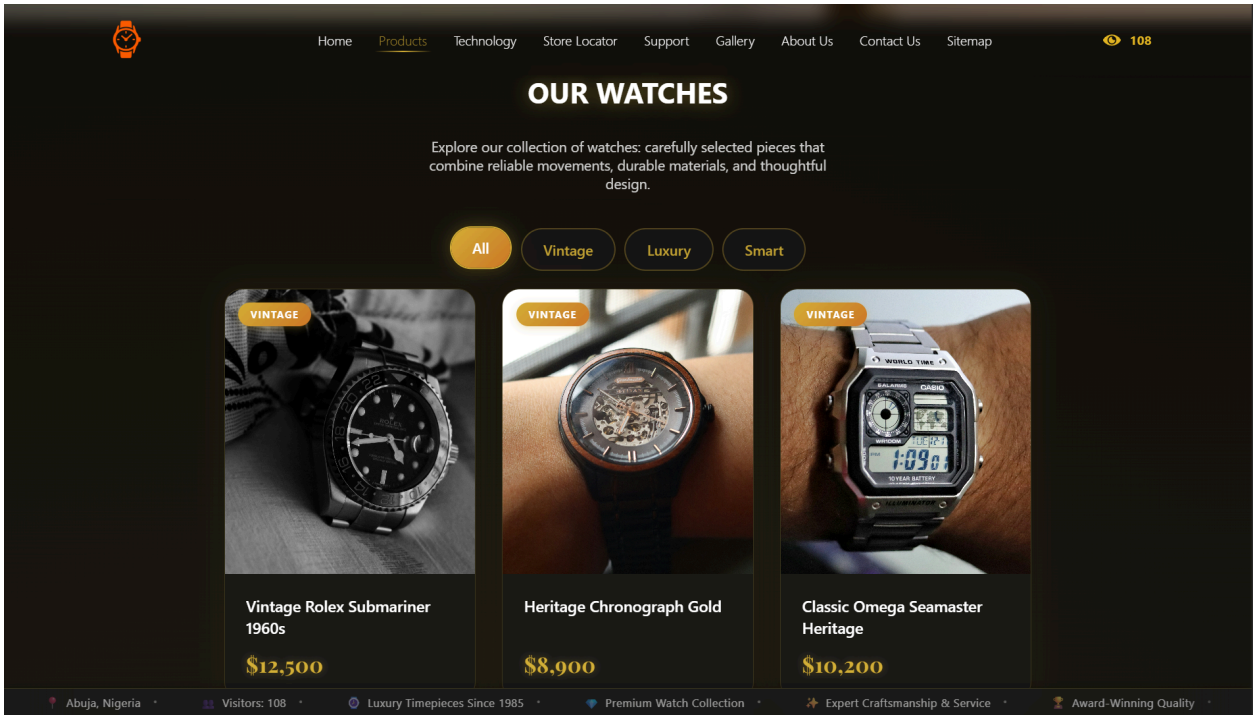
- **Geolocation:** Browser navigator.geolocation.getCurrentPosition()
- **Reverse Geocoding:** api.bigdatacloud.net for city/country lookup

---

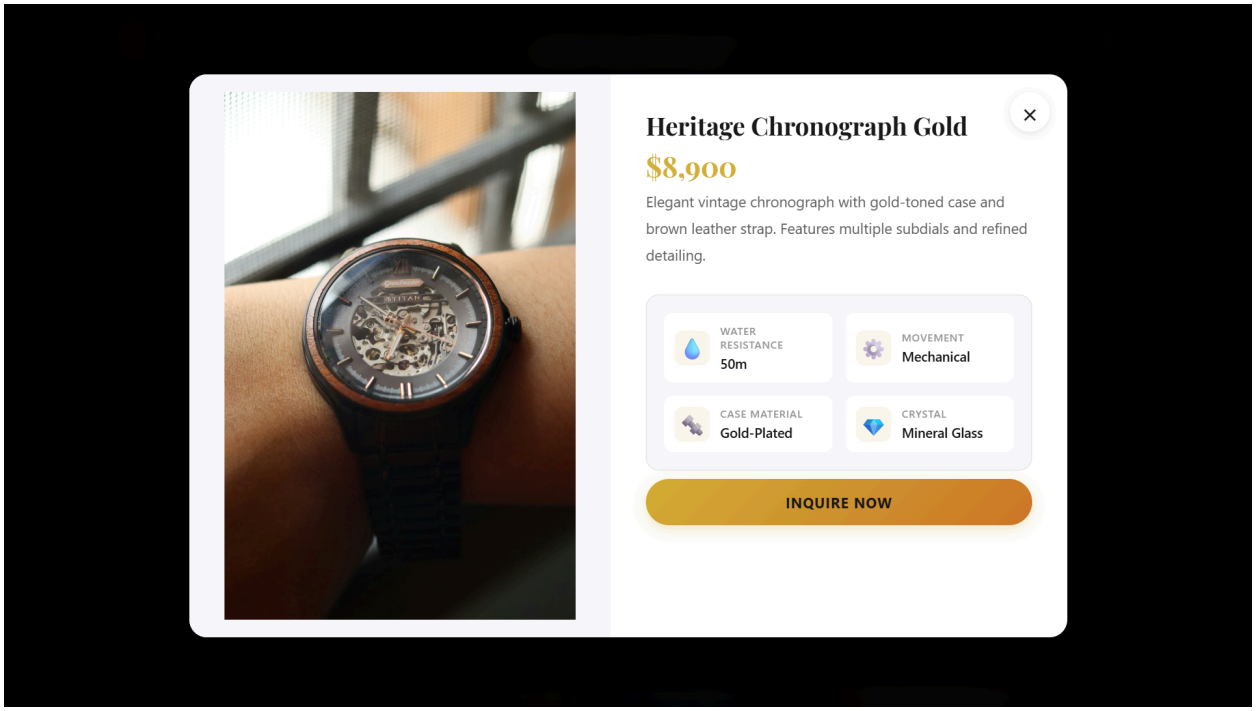
Screen Shots

1. Home page with hero section

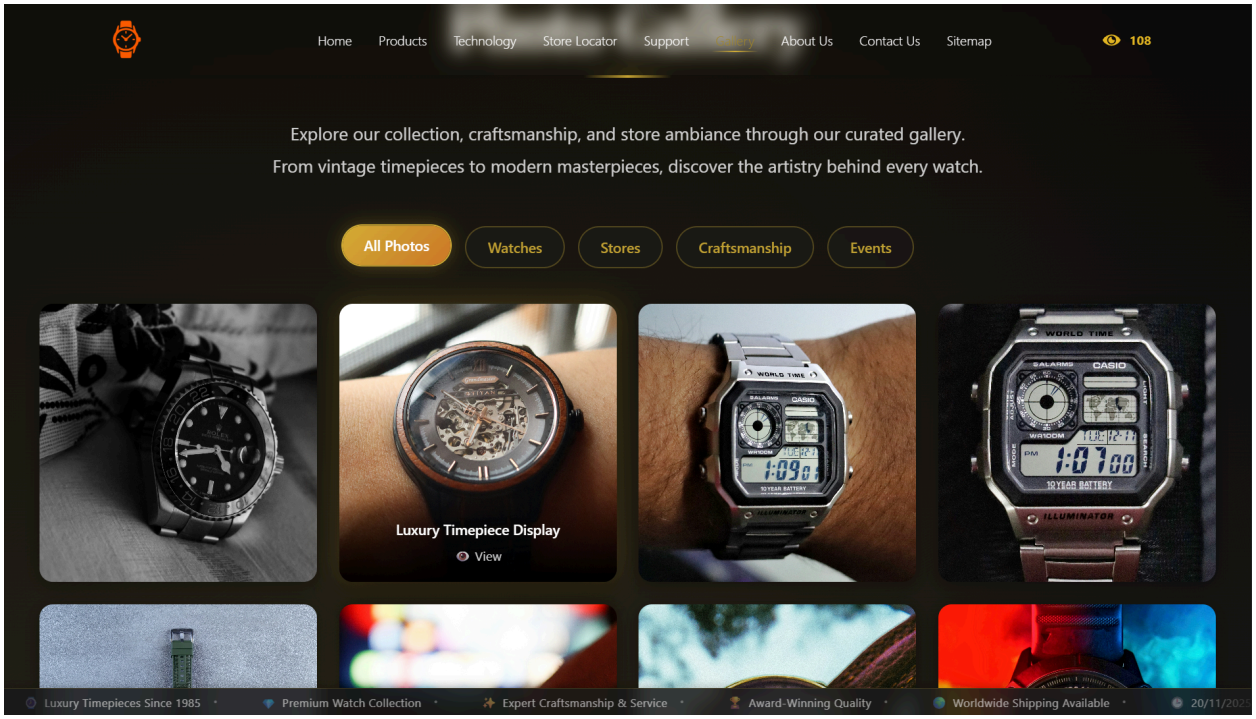
2. Products page with filter options



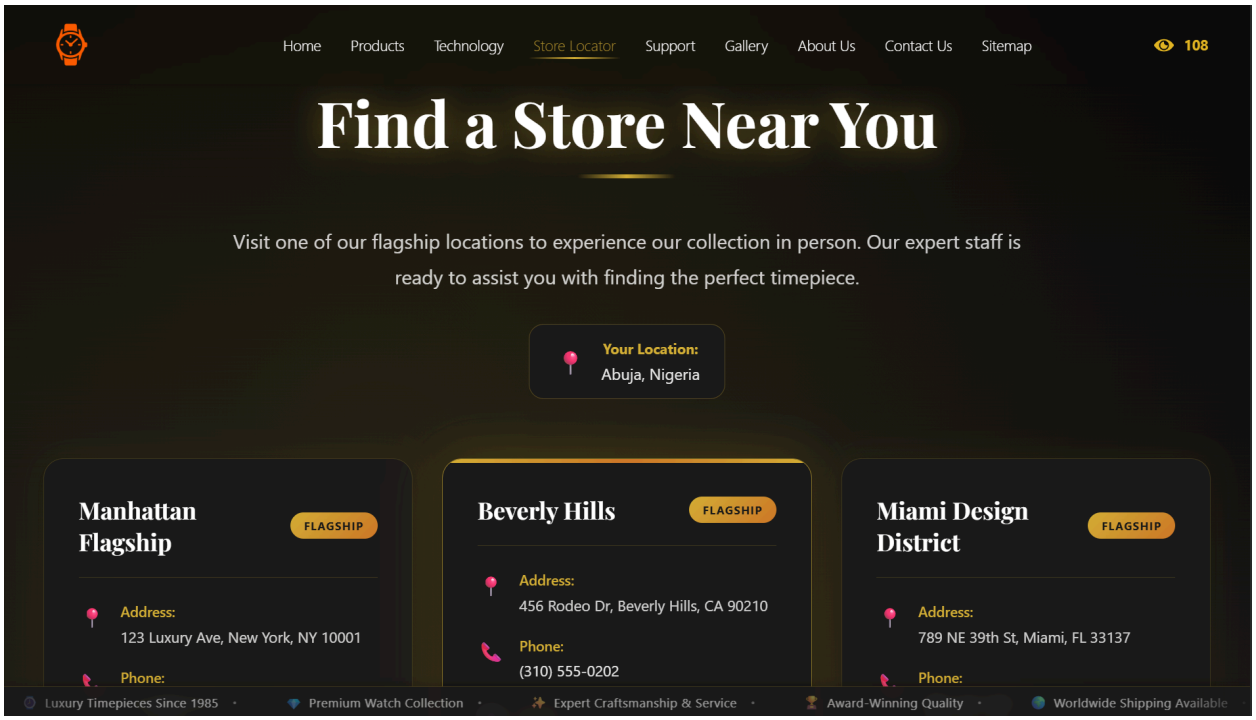
3. Product modal with details

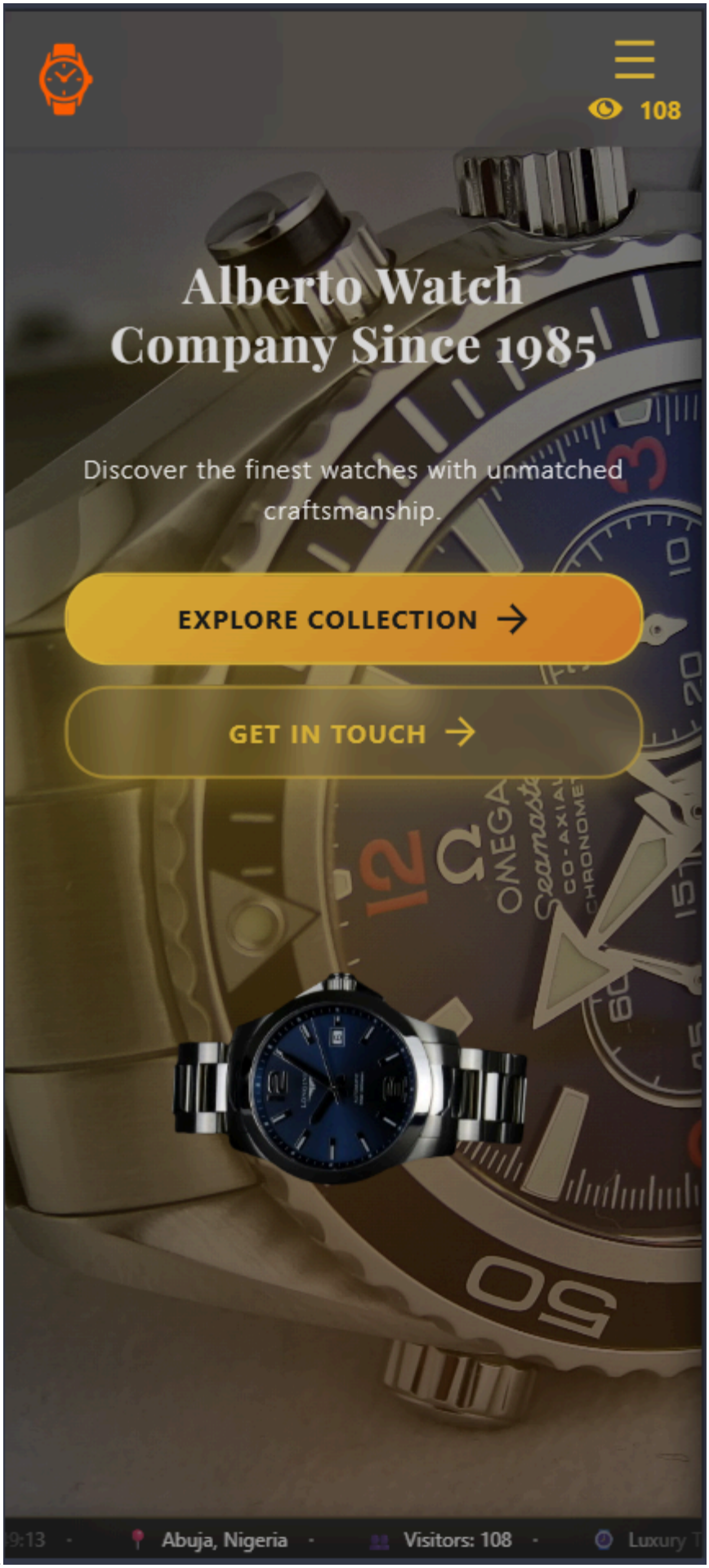


4. Gallery page with lightbox



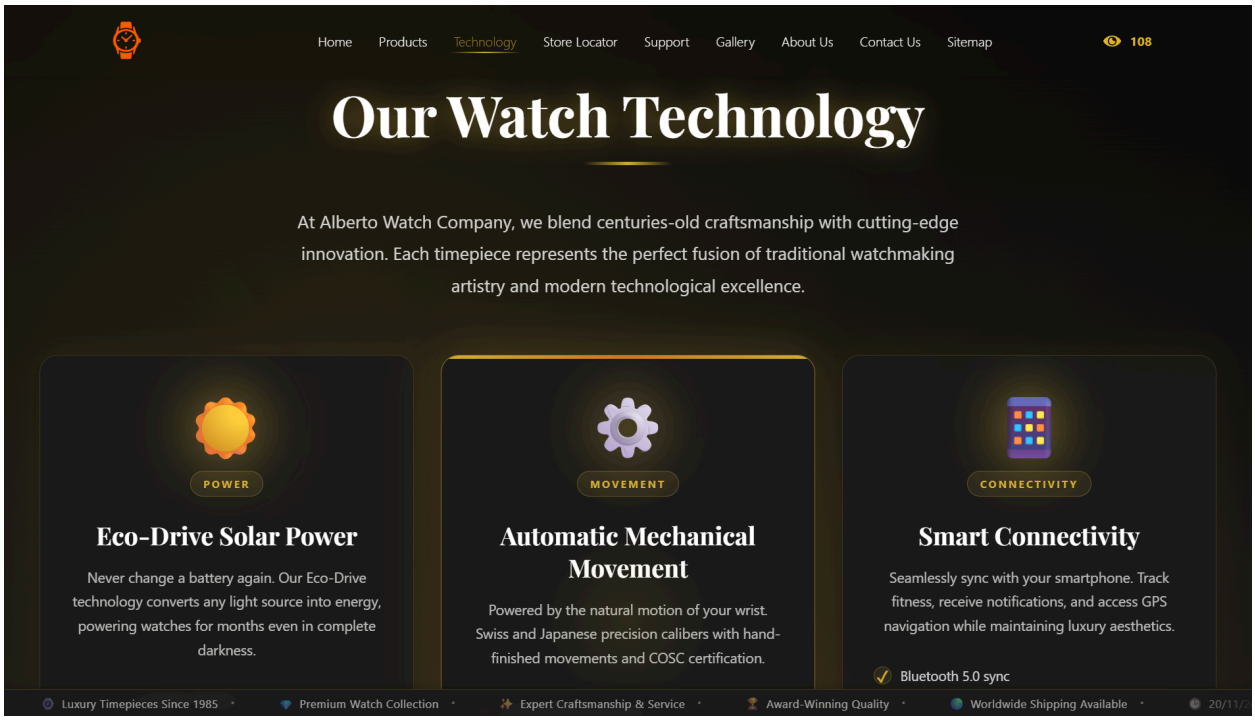
5. Store locator with map integration



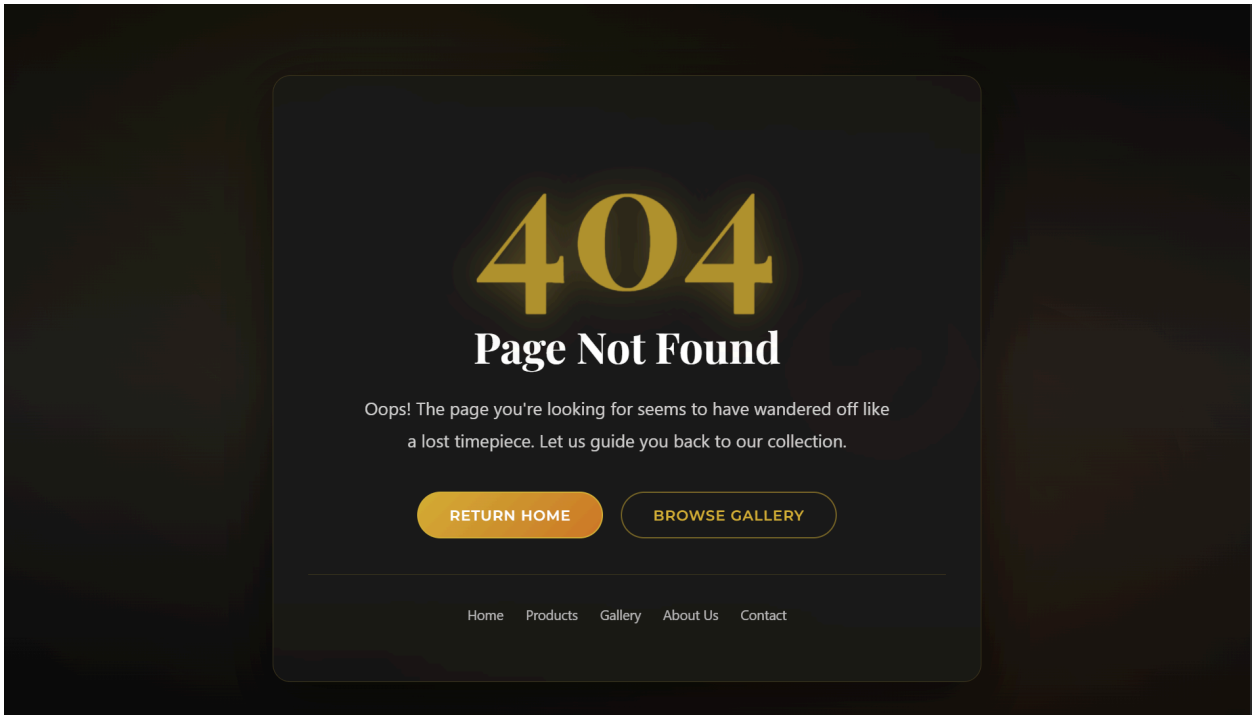


6. Mobile responsive menu

7. Technology showcase page



8. 404 error page



Source Code with Comments

Main Entry Point (src/main.jsx)

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import { BrowserRouter } from 'react-router-dom'
import './index.css'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </StrictMode>,
)
```

Root Component (src/App.jsx)

```
import { Routes, Route } from 'react-router-dom';
import Header from './components/Header';
import Ticker from './components/Ticker';
// ... other imports

function MainContent() {
  return (
    <>
```

```
      <Header />
      <main>
        <section id="home"><Home /></section>
        <section id="products"><Products /></section>
        {/* ... other sections */}
      </main>
      <Ticker />
    </>
  );
}

export default function App() {
  return (
    <div className="app">
      <Routes>
        <Route path="/" element={<MainContent />} />
        <Route path="*" element={<NotFound />} />
      </Routes>
    </div>
  );
}
```

Key Components

Header (src/components/Header.jsx): Navigation with visitor counter, mobile menu, smooth scrolling, active section detection

Products (src/pages/Products.jsx): Category filtering, product grid, modal integration

Modal (src/components/Modal.jsx): Product details display, scroll locking, portal rendering

Ticker (src/components/Ticker.jsx): Real-time clock, geolocation, visitor count, scrolling ticker

Data (src/data/watches.js): 36 watches across 3 categories (vintage, luxury, smart) with properties

---

Test Data Used in the Project

Product Test Data

Total: 36 watches across 3 categories stored in src/data/watches.js

- **Vintage (10 items, IDs 1-10):** Price range $5,200 - $15,300
    - Examples: Vintage Rolex Submariner ($12,500), Antique Patek Philippe ($15,300)
- **Luxury (14 items, IDs 11-24):** Price range $38,000 - $180,000
    - Examples: Diamond-Encrusted Audemars Piguet ($120,000), Luxury Tourbillon ($180,000)
- **Smart (12 items, IDs 25-36):** Price range $299 - $1,200
    - Examples: Premium Black Smartwatch ($650), Luxury Smartwatch Edition ($1,200)

Data Structure: Each product includes id, name, price, img, desc, and properties (waterResistance, movement, caseMaterial, crystal)

Image Test Data

Location: /public/images/ | **Total:** 36 images (JPG, PNG) | **Naming:** Sequential (1.jpg - 39.jpg)

Test Scenarios

1. **Category Filtering:** Filter by Vintage/Luxury/Smart/All (10/14/12/36 products)
2. **Product Modal:** Click product card → Display details with properties
3. **Visitor Counter:** Increments on page load, persists in localStorage
4. **Geolocation:** Shows city/country or error message based on permissions
5. **Navigation:** Smooth scroll to sections, active state updates
6. **Responsive:** Mobile (<768px), Tablet (768-1024px), Desktop (>1024px)
7. **Gallery:** Filter by category (All Photos, Watches, Stores, etc.)
8. **Store Locator:** Calculate distances, open Google Maps

Local Storage

Key: visitorsCount (String, converted to Number) | **Behavior:** Increments +1 per visit

API Test Data

Geolocation: Browser API for coordinates | **Reverse Geocoding:** api.bigdatacloud.net for city/country lookup

---

Project Installation Instructions

Prerequisites

- **Node.js** v18+ (includes npm) - Download from https://nodejs.org/
- **Code Editor** (VS Code recommended)
- **Modern Browser** (Chrome, Firefox, Safari, Edge)

Installation Steps

Navigate to project directory:
cd Alberto-Project

1.

Install dependencies:
npm install

2. Installs ~250 packages including React, Vite, React Router

Verify installation:
npm list --depth=0

3.

Running the Project

Development Mode:

npm run dev

- Runs on http://localhost:5173/
- Hot Module Replacement enabled

- Press Ctrl + C to stop

Production Build:

npm run build

- Creates optimized dist/ folder
- Minified and optimized for production

Preview Build:

npm run preview

- Serves production build locally

Project Structure

Alberto-Project/

├── node_modules/ # Dependencies

├── public/images/ # Product images

├── src/

│ ├── components/ # React components

│ ├── pages/ # Page components

│ ├── data/ # watches.js

│ └── styles/ # CSS files

├── package.json

└── vite.config.js

Troubleshooting

- **Port in use:** npm run dev -- --port 3000
- **Module not found:** Delete node_modules and package-lock.json, then npm install
- **Images not loading:** Verify paths in watches.js match /public/images/ filenames

- **Geolocation:** Requires HTTPS or localhost, check browser permissions

Deployment

1. Run npm run build
2. Upload dist/ folder to hosting service
3. Configure server to serve index.html for all routes (SPA routing)

User Guide

Getting Started

1. **Accessing the Website:**
    - Open the website in a modern web browser
    - The homepage will load automatically
2. **Navigation:**
    - Use the header menu to navigate between sections
    - Click on any menu item to smoothly scroll to that section
    - On mobile devices, tap the hamburger menu (☰) to open navigation
3. **Browsing Products:**
    - Navigate to the "Products" section
    - Use category filters (All, Vintage, Luxury, Smart) to filter watches
    - Click on any product card to view detailed information in a modal
    - Close the modal by clicking the X button or outside the modal
4. **Viewing Gallery:**
    - Go to the "Gallery" section
    - Use category filters (All Photos, Watches, Stores, Craftsmanship, Events)
    - Click on any image to open it in a full-screen lightbox
    - Click outside the image or the close button to exit
5. **Finding Stores:**
    - Visit the "Store Locator" section
    - Allow location access when prompted to see nearest stores
    - Click "Get Directions" to open Google Maps
6. **Contacting:**
    - Navigate to "Contact Us" section
    - Fill out the contact form
    - Submit your inquiry

Mobile Usage

- Tap the menu icon (☰) in the top right to open navigation

- The background will blur when the menu is open
- Tap outside the menu or the close button (✕) to close
- All features are fully functional on mobile devices

Developer's Guide

Module Descriptions

1. App.jsx - Root Application Component

Purpose: Main application entry point with routing configuration
Key Functions:

- Sets up React Router for client-side routing
- Renders MainContent component for home route
- Handles 404 errors with NotFound component

Dependencies: React Router DOM

2. Header.jsx - Navigation Component

Purpose: Provides site-wide navigation and visitor tracking
Key Features:

- Responsive navigation menu with mobile support
- Active section highlighting based on scroll position
- Visitor counter using localStorage
- Mobile menu with backdrop blur effect
- Smooth scroll navigation

State Management:

- visitorCount: Tracks total visitors
- isMobileOpen: Controls mobile menu visibility
- activeNavItem: Tracks currently active navigation item

Key Functions:

- handleNavClick(): Handles navigation clicks with smooth scrolling
- Scroll detection for active nav highlighting
- Body scroll blocking when mobile menu is open

3. Products.jsx - Product Catalog Page

Purpose: Displays and filters watch products
Key Features:

- Category-based filtering (All, Vintage, Luxury, Smart)
- Product grid display
- Modal integration for product details

State Management:

- selectedCategory: Current filter selection
- selectedProduct: Product to display in modal

Data Source: src/data/watches.js - Contains 36 watch products

4. Gallerypage.jsx - Photo Gallery Component

Purpose: Interactive image gallery with lightbox
Key Features:

- Category filtering (Watches, Stores, Craftsmanship, Events)
- Modal lightbox for full-screen image viewing
- Scroll blocking when modal is open
- Image lazy loading

State Management:

- selectedImg: Currently selected image for lightbox
- selectedCategory: Active gallery filter

Image Source: Local images from /public/images folder

5. Modal.jsx - Product Detail Modal

Purpose: Displays detailed product information
Key Features:

- Portal rendering for proper z-index layering
- Body scroll blocking when open
- Product properties display
- Close functionality

Props:

- product: Product object with details
- onClose: Callback function to close modal

6. StoreLocator.jsx - Store Finder Component

Purpose: Displays store locations with geolocation
Key Features:

- User geolocation detection
- Store list with details
- Google Maps integration for directions
- Distance calculation (if implemented)

State Management:

- userLocation: User's coordinates
- userCity: User's city name
- selectedStore: Currently selected store

7. Ticker.jsx - Footer Ticker Component

Purpose: Displays dynamic information ticker
Key Features:

- Real-time clock display
- Visitor count display
- Location information
- Company information scrolling

8. watches.js - Product Data Module

Purpose: Centralized product data storage
Structure:

- Three main categories: vintage, luxury, smart
- Each product contains: id, name, price, img, desc, properties
- Exports: watches (categorized) and allWatches (flat array)

File Structure

src/

├── components/ # Reusable React components

│ ├── Header.jsx # Navigation header

│ ├── Modal.jsx # Product detail modal

│ └── Ticker.jsx # Footer ticker

├── pages/ # Page components

```
│   ├── Home.jsx
│   ├── Products.jsx
│   ├── Gallerypage.jsx
│   ├── Technology.jsx
│   ├── StoreLocator.jsx
│   ├── Support.jsx
│   ├── About.jsx
│   ├── Contact.jsx
│   ├── Sitemap.jsx
│   └── NotFound.jsx
├── data/              # Data files
│   └── watches.js     # Product catalog data
├── styles/            # CSS stylesheets
│   └── [Component].css # Individual component styles
├── assets/            # Static assets
│   └── logo.png
├── App.jsx            # Root component
└── main.jsx           # Application entry point
```

Styling Architecture

- **Component-based CSS:** Each component has its own CSS file
- **Responsive Design:** Mobile-first approach with media queries
- **CSS Variables:** Used for consistent theming (colors, spacing)
- **Modern CSS:** Uses backdrop-filter, CSS Grid, Flexbox

Key Technologies

1. **React 19.2.0:** Component-based UI library
2. **Vite 7.2.2:** Fast build tool and dev server
3. **React Router DOM 7.9.5:** Client-side routing
4. **Local Storage API:** Visitor tracking
5. **Geolocation API:** Store locator functionality

Development Workflow

Installation:
npm install

1.

Development Server:
npm run dev

2.

Build for Production:
npm run build

3.

Preview Production Build:
npm run preview

4.

Adding New Features

1. **New Page:** Create component in src/pages/, add route in App.jsx, add section in MainContent
2. **New Component:** Create in src/components/, import where needed
3. **New Product:** Add to src/data/watches.js in appropriate category
4. **Styling:** Create corresponding CSS file in src/styles/

Best Practices

- Use functional components with hooks
- Keep components focused and reusable
- Use proper state management
- Implement error boundaries
- Optimize images and assets
- Follow React best practices for performance

Report Generated: 2025
Version: 1.0