Sneaker Corner

E-commerce Website for Sneakers

Web Programming
Semester Project Report

Author: Léo Floissac

Course: Web Programming

Date: December 2025

Technologies Used:

HTML5 • CSS3 • JavaScript • PHP • MySQL
MVC Architecture • AJAX • Responsive Design

Contents

1. Introduction

1.1 Project Overview

Sneaker Corner is a complete e-commerce website dedicated to premium sneakers, designed to showcase products from major brands (Nike, Adidas, Puma, New Balance, Converse, Vans) with a focus on the Vietnamese market.

The project was developed as part of the Web Programming course to demonstrate proficiency in modern web development technologies including HTML5, CSS3, JavaScript, PHP, and MySQL.

1.2 Project Objectives

The main objectives of this project are:

- Create a fully functional e-commerce website with a clean, modern design
- Implement responsive design for optimal viewing on desktop, tablet, and mobile devices
- Develop a robust product catalog with filtering and search capabilities
- Integrate user authentication system (register, login, logout, forgot password)
- Display store locations with Google Maps integration
- Apply SEO best practices for better search engine visibility
- Follow the MVC (Model-View-Controller) architectural pattern

1.3 Target Audience

The website targets sneaker enthusiasts in Vietnam who want to:

- Browse and discover authentic sneakers from premium brands
- Filter products by brand, color, and size
- Check product availability in physical stores

2. Design Choices

2.1 Architecture: MVC Pattern

The project follows the **Model-View-Controller (MVC)** architectural pattern to ensure clean separation of concerns:

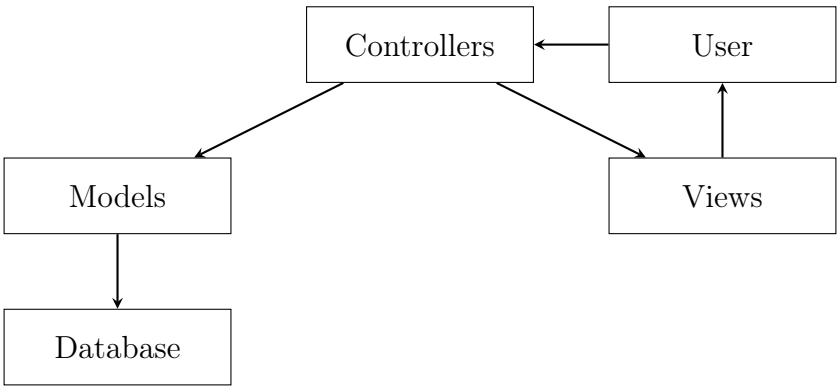

Figure 1: MVC Architecture Overview

Directory Structure:

```
sneaker_corner/
|-- app/
|   |-- controllers/     # Business logic
|   |   |-- AuthController.php
|   |   |-- ProductsController.php
|   |   |-- StoresController.php
|   |-- models/          # Database operations
|   |   |-- Product.php
|   |   |-- ProductVariant.php
|   |   |-- Color.php
|   |   |-- VariantSize.php
|   |   |-- VariantImage.php
|   |   |-- StoreInventory.php
|   |   |-- Store.php
|   |   |-- User.php
|   |-- views/           # HTML/PHP templates
|   |   |-- partials/    # Reusable components
|   |-- api/             # AJAX endpoints
|-- config/              # Configuration files
|-- css/                 # Stylesheets
|-- js/                  # JavaScript files
|-- database/            # SQL schema and data
|-- images/              # Static assets
|-- public/              # Entry point (index.php)
```

Listing 1: Project Structure

2.2 Layout and Structure

The website layout follows modern e-commerce conventions:

- **Header:** Fixed navigation with logo on the left, navigation links (Home, Products, Stores, About, Contact), and user authentication buttons on the right
- **Navigation Bar:** Horizontal menu with links to main sections
- **Body:** Content area with a maximum width of 1200px for optimal readability, centered on the page
- **Footer:** Contact information, social links, and copyright notice

Width Choice Explanation: The body width is set to `max-width: 1200px` because:

- It provides optimal line length for readability (50-75 characters per line)
- It works well on most desktop screens (1366px, 1920px)
- It allows comfortable padding on each side
- It follows common e-commerce design patterns

2.3 Color Scheme and Typography

The design uses a minimalist color palette:

- **Primary:** Black (#000000) - For text and buttons
- **Background:** White (#FFFFFF) - Clean, professional look
- **Accent:** Dark gray (#333333) - For secondary elements
- **Typography:** System fonts stack for fast loading

3. Implementation Details

3.1 Database Design

The database schema uses MySQL with the following tables:

Table 1: Database Tables Overview

Table	Records	Description
`users`	Variable	User accounts with hashed passwords
`products`	Multiple	Base product information (name, brand, slug)
`product_variants`	Multiple	Color variants with prices
`colors`	Multiple	Color definitions with slugs
`variant_sizes`	Multiple	Sizes with stock quantities
`variant_images`	Multiple	Product images with ordering
`stores`	3	Physical store locations
`store_inventory`	Multiple	Stock per store/variant/size
`categories`	Multiple	Product categories (hierarchical)

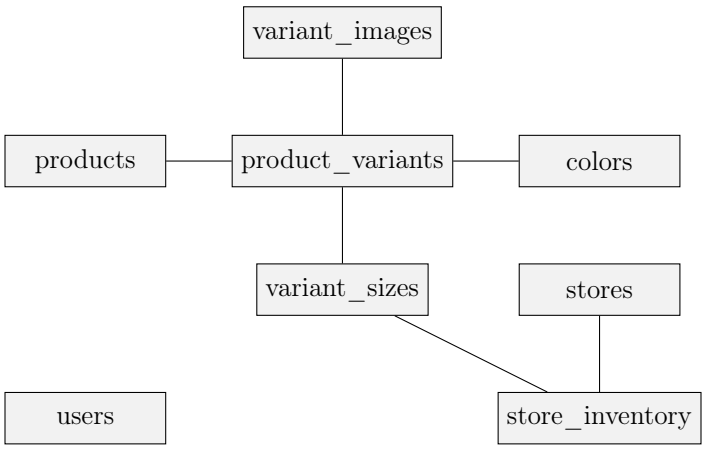

Figure 2: Entity Relationship Diagram (Simplified)

3.2 Responsive Design Implementation

The website uses CSS media queries and flexible layouts:

```
/* Mobile First Approach */
.products-grid {
    display: grid;
    grid-template-columns: 1fr;
    gap: 1rem;
}

/* Tablet (768px+) */
@media (min-width: 768px) {
    .products-grid {
        grid-template-columns: repeat(2, 1fr);
    }
}

/* Desktop (1024px+) */
@media (min-width: 1024px) {
    .products-grid {
        grid-template-columns: repeat(4, 1fr);
    }
}
```

Listing 2: Responsive CSS Example

Breakpoints used:

- Mobile: $<$ 768px (1 column layout)
- Tablet: 768px - 1023px (2 column layout)
- Desktop: 1024px+ (4 column layout)

3.3 Products Page with Lazy Loading

The products page implements lazy loading using the Intersection Observer API:

```
// Intersection Observer for lazy loading
const observer = new IntersectionObserver(function(entries) {
    entries.forEach(function(entry) {
```

```
        if (entry.isIntersecting && !isLoading) {
            loadMoreProducts();
        }
    });
}, {
    root: null,
    rootMargin: '200px', // Load before reaching trigger
    threshold: 0
});

observer.observe(document.getElementById('lazy-load-trigger'));
```

Listing 3: Lazy Loading Implementation

Features:

- Initial load: 12 products
- Load on scroll: 12 more products per batch
- Loading spinner animation
- Smooth fade-in effect for new products

3.4 AJAX Search Functionality

The search feature provides real-time suggestions as the user types:

```
// Debounced search function
searchInput.addEventListener('input', function() {
    const query = this.value.trim();

    if (searchTimeout) clearTimeout(searchTimeout);

    // Debounce: wait 300ms before searching
    searchTimeout = setTimeout(function() {
        if (query.length >= 2) {
            performAjaxSearch(query);
        }
    }, 300);
});
```

Listing 4: AJAX Search with Debouncing

Backend API Endpoint (`/app/api/search.php`):

```
<?php
header('Content-Type: application/json');

$query = isset($_GET['q']) ? trim($_GET['q']) : '';

if (strlen($query) < 2) {
    echo json_encode(['results' => [], 'count' => 0]);
    exit;
}

$productsController = new ProductsController($conn);
$results = $productsController->search($query);
```

```php

echo json_encode([
    'results' => $results,
    'count' => count($results),
    'query' => $query
]);
```

Listing 5: Search API Endpoint

3.5 Filtering System with Breadcrumbs

Products can be filtered by:

- **Brand:** Nike, Adidas, Puma, etc.
- **Color:** Black, White, Red, etc.
- **Size:** EU sizes (36, 37, 38, ..., 46)

Breadcrumb Navigation:

```
Home > Products > Brand: Nike > Color: Black
```

Listing 6: Breadcrumb Example

The URL structure uses query parameters:

`/products.php?brand=Nike&color=black&size=42`

3.6 Store Locations with Google Maps

Each product detail page shows store availability with direct links to Google Maps:

```php
// Generate Google Maps URL
if ($store['latitude'] && $store['longitude']) {
    $mapsUrl = "https://www.google.com/maps?q="
             . $store['latitude'] . "," . $store['longitude'];
} else {
    $address = urlencode($store['address'] . ", " . $store['city']);
    $mapsUrl = "https://www.google.com/maps/search/?api=1&query="
             . $address;
}
```

Listing 7: Google Maps URL Generation

3.7 User Authentication

The authentication system includes:

Table 2: Authentication Features

Feature	Implementation
Registration	Email validation, password hashing (bcrypt), duplicate check
Login	Email/password verification, session management
Logout	Session destruction, cookie cleanup
Forgot Password	Form available (email sending not implemented*)
Validation	Client-side (HTML5) + Server-side (PHP)

*Note: Email sending requires SMTP configuration not available in the local MAMP environment.

4. SEO Implementation

The following SEO practices were implemented:

4.1 1. Semantic HTML5 Structure

```
<header>
    <nav aria-label="Main navigation">...</nav>
</header>
<main class="products-page">
    <section class="featured-products">
        <article class="product-card">...</article>
    </section>
</main>
<footer>...</footer>
```

Listing 8: Semantic HTML Usage

4.2 2. SEO-Friendly URLs

Instead of: `/product.php?id=123`
We use: `/product-detail.php?slug=nike-air-max-90&color=white`

4.3 3. XML Sitemap

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
    <url>
        <loc>http://localhost/sneaker_corner/public/index.php</loc>
        <changefreq>daily</changefreq>
        <priority>1.0</priority>
    </url>
    <url>
        <loc>http://localhost/sneaker_corner/app/views/products.php</
           loc>
        <changefreq>daily</changefreq>
        <priority>0.9</priority>
```

```
    </url>
    <!-- Additional pages... -->
</urlset>
```

Listing 9: sitemap.xml

4.4 4. Meta Tags

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Sneaker Corner - Premium Sneakers in Vietnam</title>
<meta name="description" content="Discover authentic sneakers...">
```

Listing 10: Meta Tags in Header

4.5 5. Image Alt Attributes

All images include descriptive alt text:

```
<img src="nike-air-max.jpg" alt="Nike Air Max 90 - White colorway">
```

5. Technologies and Tools

Table 3: Technologies Stack

Category	Technology	Purpose
Frontend	HTML5	Semantic structure
Frontend	CSS3	Styling, animations, responsive design
Frontend	JavaScript (Vanilla)	Interactivity, AJAX, lazy loading
Backend	PHP 8.x	Server-side logic, MVC
Database	MySQL	Data storage
Server	MAMP	Local development environment
Version Control	Git/GitHub	Code versioning

Note on Libraries: This project was built without external CSS/JS frameworks (no Bootstrap, no jQuery) to demonstrate understanding of core web technologies.

6. Challenges and Solutions

6.1 Challenge 1: Product Variant System

Problem: Each sneaker can have multiple color variants, each with its own images and sizes.

Solution: Designed a normalized database schema with separate tables for products, variants, colors, sizes, and images, linked by foreign keys.

6.2 Challenge 2: AJAX Search Performance

Problem: Search requests could overload the server if sent on every keystroke.

Solution: Implemented debouncing (300ms delay) and request cancellation for previous pending requests.

6.3 Challenge 3: Store Inventory Display

Problem: Showing which stores have which sizes in stock is complex.

Solution: Created a `store_inventory` junction table linking stores, variants, and sizes with quantity tracking.

7. Lessons Learned

1. **MVC Architecture Benefits:** Separating models, views, and controllers makes the code more maintainable and testable. Each model handles one database table, following the Single Responsibility Principle.
2. **Database Normalization:** Proper normalization (3NF) prevents data redundancy and ensures data integrity, especially for complex relationships like product-variant-size-store.
3. **Performance Optimization:** Lazy loading and debouncing significantly improve user experience by reducing initial load time and server requests.
4. **SEO Matters:** Semantic HTML, proper URL structure, and sitemaps help search engines understand and index the website content.

8. Future Improvements

- **Email System:** Integrate PHPMailer with SMTP for password reset emails
- **Social Login:** Implement Google and Facebook OAuth authentication
- **Shopping Cart:** Add cart functionality and checkout process
- **Admin Panel:** Create a dashboard for product/inventory management

9. Conclusion

The Sneaker Corner project successfully demonstrates the implementation of a complete e-commerce website using modern web technologies. The project meets all the specified requirements:

Clean layout with header, navigation, body, and footer

Fully responsive design (desktop, tablet, mobile)

Products page with filtering and lazy loading

AJAX search with dropdown suggestions

Category organization with breadcrumb navigation

Store locations with Google Maps integration

User authentication (register, login, logout, forgot password)

SEO practices (semantic HTML, sitemap, friendly URLs, meta tags)

The MVC architecture ensures the codebase is maintainable and scalable for future enhancements. This project provided valuable hands-on experience in full-stack web development, from database design to responsive frontend implementation.