# Functional & Non-Functional Requirements

**Project:** Atino Clone - Men's Fashion Store MVP

## 1. Project Overview

The goal is to build a frontend web application replicating the core shopping experience of **Atino.vn**. The application will simulate a real e-commerce site using Vanilla JavaScript and a Mock API for data retrieval. The design focuses on a clean, minimalist aesthetic suitable for men's fashion.

## 2. Functional Requirements

### 2.1. Navigation & Header

- **FR-01:** The website must have a responsive navigation bar including: Home, Shop (All Products), Collections (Shirts, Pants), and About Us.
- **FR-02:** A persistent "Cart" icon must be visible in the header, displaying a badge with the current number of items in the cart.

### 2.2. Homepage (Hero & Featured)

- **FR-03: Hero Section:** Display a large banner image reflecting the "Gentleman/Streetwear" style of Atino.
- **FR-04: Featured Section:** Display a grid of 4-8 "New Arrival" products fetched from the Mock API.

### 2.3. Product Browsing (PLP - Product Listing Page)

- **FR-05: Product Grid:** Render a list of products dynamically. Each card must show: Image, Name, Price (formatted in VND), and an "Add to Cart" button.
- **FR-06: Filtering:** Users must be able to filter products by:
  - Category (e.g., Polo, Shirt, Jeans, Trousers).
  - Price Range (e.g., Under 300k, 300k-500k, Above 500k).
- **FR-07: Sorting:** Users can sort products by "Price: Low to High" and "Price: High to Low".

### 2.4. Product Detail (PDP - Product Detail Page/Modal)

- *Note: For a 3-day MVP, a Quick View Modal is recommended over a separate page to simplify routing logic.*
- **FR-08:** Clicking a product card opens a detailed view showing:
  - Multiple images (or main image).
  - Size Selector (S, M, L, XL).

    - Color Selector.
    - Description.
- **FR-09:** Users must select a Size and Color before adding to the cart.

### 2.5. Shopping Cart & Checkout

- **FR-10: Add to Cart:** Clicking "Add to Cart" adds the item object to a global state array.
    - *Logic:* If the same Item ID + Size + Color exists, update quantity instead of creating a new row.
- **FR-11: Cart View:** A specific section (or drawer) to view added items.
- **FR-12: Update Quantity:** Users can increase (+) or decrease (-) the quantity of an item.
- **FR-13: Remove Item:** Users can delete an item from the cart.
- **FR-14: Total Calculation:** The total price must update dynamically whenever quantity changes.

### 2.6. Data Persistence

- **FR-15:** The application must save the Cart state to `localStorage`. If the user reloads the browser, the cart items must persist.

## 3. Technical Constraints

- **Language:** HTML5, CSS3, Vanilla JavaScript (ES6+).
- **No Frameworks:** No React, Vue, Angular, Bootstrap, or Tailwind.
- **Data Source:** Mock API (using a local `products.json` file or `json-server`).
- **Styling:** Custom CSS using Flexbox and CSS Grid to mimic Atino.vn's layout.

## 4. Data Schema (Mock API Structure)

The `products.json` should follow this structure to support the requirements:

```json
{
  "products": [
    {
      "id": 101,
      "name": "Áo Polo Basic Slimfit",
      "price": 350000,
      "category": "polo",
      "images": ["url_front.jpg", "url_back.jpg"],
      "sizes": ["S", "M", "L", "XL"],
      "colors": ["Black", "White", "Navy"],
      "isNewArrival": true
    }
  ]
}
```