**Ministry Higher Education and Scientific Research**

TEKUP UNIVERSITY

TECHNICAL REPORT

# OrganicaFarm

**E-Commerce Platform**

*A Comprehensive Frontend Implementation*

**Subject:**

Web Development & Frontend Engineering

**Author:**

Nour ZAGHOUANI

Academic Year 2024-2025

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Project Overview

**OrganicaFarm** is a modern, fully responsive e-commerce web application specifically designed for showcasing and selling fresh, organic fruits and vegetables. The project represents a comprehensive frontend implementation that bridges the gap between sustainable farming practices and digital consumer engagement.

The platform provides an intuitive user interface that allows customers to:

- Browse a complete catalog of organic products
- Filter products by categories (Fruits, Vegetables)
- Add products to a persistent shopping cart
- View seasonal availability information
- Submit feedback and contact inquiries
- Learn about the farm's mission and practices

## 1.2 Project Objectives

The development of this platform was guided by the following core objectives:

1. **User Experience**: Design an intuitive, aesthetically pleasing interface that reflects organic farming values
2. **Accessibility**: Ensure WCAG 2.1 AA compliance for users with disabilities
3. **Responsiveness**: Guarantee flawless functionality across all devices (mobile, tablet, desktop)
4. **Code Quality**: Implement clean, modular, and maintainable code architecture
5. **SEO Optimization**: Maximize search engine visibility through proper meta tags and semantic HTML
6. **Performance**: Achieve fast loading times and smooth animations

## 1.3 Scope and Limitations

### 1.3.1 Current Scope

This project is a **frontend-only implementation** with the following characteristics:

- Client-side rendering using vanilla JavaScript (ES6+)
- LocalStorage-based data persistence for shopping cart
- Static HTML pages with dynamic content manipulation
- No server-side processing or database integration
- Simulated e-commerce workflows (checkout process)

### 1.3.2 Technical Stack

- **HTML5**: Semantic markup and structure
- **CSS3**: Advanced styling with custom properties (CSS Variables)
- **JavaScript (ES6+)**: Modern ECMAScript features including classes, arrow functions, and template literals
- **Bootstrap 5.3.0**: Responsive grid system and UI components
- **Font Awesome 6.4.0**: Comprehensive icon library
- **Google Fonts**: Montserrat & Playfair Display typography

# Chapter 2

# Requirements Analysis

## 2.1 Functional Requirements

### 2.1.1 Product Management

- **FR1**: Display product catalog with images, names, descriptions, and prices
- **FR2**: Implement category-based filtering system (All, Vegetables, Fruits)
- **FR3**: Show seasonal availability for each product
- **FR4**: Support dynamic product card rendering

### 2.1.2 Shopping Cart Functionality

- **FR5**: Add products to cart with quantity management
- **FR6**: Update item quantities (increase/decrease)
- **FR7**: Remove items from cart
- **FR8**: Calculate and display total price
- **FR9**: Persist cart data across page refreshes using LocalStorage
- **FR10**: Display real-time cart item count in navigation

### 2.1.3 User Interaction

- **FR11**: Provide contact form with real-time validation
- **FR12**: Display Google Maps integration for farm location
- **FR13**: Show customer testimonials and feedback
- **FR14**: Implement smooth page navigation

## 2.2 Non-Functional Requirements

### 2.2.1 Performance Requirements

- **NFR1**: Page load time < 3 seconds on standard broadband connection
- **NFR2**: Smooth animations at 60 FPS
- **NFR3**: Optimized images with lazy loading where applicable

### 2.2.2 Usability Requirements

- **NFR4**: Intuitive navigation requiring no training
- **NFR5**: Clear visual feedback for all user actions
- **NFR6**: Consistent design language across all pages

### 2.2.3 Accessibility Requirements

- **NFR7**: WCAG 2.1 Level AA compliance
- **NFR8**: Keyboard navigation support
- **NFR9**: ARIA labels for screen readers
- **NFR10**: Sufficient color contrast ratios (4.5:1 minimum)
- **NFR11**: Skip-to-content link for keyboard users

### 2.2.4 Browser Compatibility

- **NFR12**: Chrome (Latest 2 versions)
- **NFR13**: Firefox (Latest 2 versions)
- **NFR14**: Edge (Latest 2 versions)
- **NFR15**: Safari (Latest version)

# Chapter 3

# System Design & Architecture

## 3.1 Architectural Pattern

The application follows a **Static Website Architecture** enhanced with client-side dynamic rendering. This architecture provides several advantages:

1. **Simplicity**: No server infrastructure required for deployment
2. **Cost-Effectiveness**: Can be hosted on static hosting services (GitHub Pages, Netlify)
3. **Security**: Reduced attack surface due to no backend
4. **Performance**: Fast delivery through CDN distribution

## 3.2 File Structure

The project is organized following separation of concerns principle:

```
OrganicaFarm/
 index.html              # Homepage
 accueil.html            # Welcome page
 aboutus.html            # About Us page
 products.html           # Product catalog
 feedback.html           # Customer testimonials
 contact.html            # Contact form
 checkout.html           # Checkout process

 css/
    main.css           # Global styles (9.6 KB)
    contact.css        # Contact page styles
    accueil.css        # Welcome page styles
    checkout.css       # Checkout page styles

 js/
    main.js            # Shopping cart logic
    products.js        # Product filtering
```

```
   contact.js         # Form validation
   template-loader.js # Component loading

templates/
   navigation.html    # Reusable navbar
   footer.html        # Reusable footer

images/               # Product images and assets
```

## 3.3 Site Structure & Navigation

### 3.3.1 Information Architecture

The website consists of 6 main pages organized in a flat hierarchy:

| **Page** | **File** | **Purpose** |
|---|---|---|
| Home | index.html | Landing page, hero section, features |
| Welcome | accueil.html | Extended introduction |
| About Us | aboutus.html | Farm history and mission |
| Products | products.html | Full catalog with filtering |
| Feedback | feedback.html | Customer testimonials |
| Contact | contact.html | Contact form & map |
| Checkout | checkout.html | Order summary & payment |

# Chapter 4

# Detailed Page-by-Page Analysis

## 4.1 Homepage (index.html)

### 4.1.1 Purpose

The homepage serves as the primary landing page, introducing visitors to OrganicaFarm's value proposition and core offerings.

### 4.1.2 Key Sections

#### Hero Section

- **Visual**: Full-width background image with overlay gradient
- **Content**: Primary headline, value proposition, and call-to-action
- **Implementation**: CSS background-image with linear-gradient overlay
- **Responsiveness**: Padding adjusts from 120px to 80px on mobile devices

#### Features Section

Displays six key value propositions using feature cards:

1. Sustainable Farming
2. 100% Organic
3. Water Conservation
4. Natural Pest Management
5. Free Delivery
6. Fresh Daily Harvest

Each feature card includes:

- Font Awesome icon (2.5rem size)
- Feature title (h3 heading)

- Descriptive paragraph
- Hover effect (translateY(-5px) with box-shadow)

### 4.1.3 SEO Implementation

```html
<title>OrganicaFarm - Fresh Organic Products | Bizerte, Tunisia</title>
<meta name="description" content="Discover our 100% organic fruits and
vegetables sustainably grown in Bizerte. Family farm since 1985.">
<meta name="keywords" content="organic, fruits, vegetables, Bizerte,
Tunisia, sustainable farming">

<!-- Open Graph -->
<meta property="og:type" content="website">
<meta property="og:title" content="OrganicaFarm - Fresh Organic
    Products">
<meta property="og:description" content="100% organic produce">
```

Listing 4.1: Homepage Meta Tags

## 4.2 Products Page (products.html)

### 4.2.1 Product Filtering System

The products page implements an advanced client-side filtering mechanism using the `ProductFilter` class.

#### Filter Buttons Implementation

```javascript
class ProductFilter {
    constructor() {
        this.filterButtons = document.querySelectorAll('[data-filter]')
    ;
        this.products = document.querySelectorAll('#product-grid > div[
    data-category]');
        this.init();
    }

    filterProducts(category) {
        this.products.forEach(product => {
            const productCategory = product.getAttribute('data-category
    ');
            const visible = category === 'all' ||
                          productCategory.includes(category);
            product.style.display = visible ? 'block' : 'none';
        });
    }
}
```

Listing 4.2: Product Filter Class

### 4.2.2 Product Card Structure

Each product card contains:

- Product image (200px height, object-fit: cover)
- Category badge (absolute positioning)
- Product name (h5 heading)
- Short description
- Price with currency symbol
- "Add to Cart" button

### 4.2.3 Seasonal Availability Table

The products page features an advanced HTML table demonstrating:

- **rowspan**: Merging category cells vertically
- **colspan**: Indicating year-round availability
- **Bootstrap badges**: Visual status indicators
- **Semantic structure**: Proper thead/tbody organization

```html
<tr>
    <th scope="row" rowspan="6" class="align-middle bg-warning">
        <strong>FRUITS</strong>
    </th>
    <th scope="row">Strawberries</th>
    <td class="text-center">-</td>
    <td class="text-center"><span class="badge bg-success">Peak</span><
  /td>
</tr>
```

Listing 4.3: Table with rowspan Example

## 4.3 Contact Page (contact.html)

### 4.3.1 Contact Form Design

The contact form implements comprehensive client-side validation using the `FormValidator` class.

**Form Fields**

1. **Full Name**:
   - Minimum 3 characters
   - Letters only (regex: `/^[a-zA-ZÀ-ỹ`
     `s`
     `-']+$/`)
   - Required field
2. **Email**:
   - Email format validation
   - Regex: `/^[^`
     `s@]+@[^`
     `s@]+`
     `.[^`
     `s@]+$/`
   - Required field
3. **Phone**:
   - Optional field
   - International format support
   - Regex pattern for validation
4. **Message**:
   - Minimum 10 characters
   - Textarea with 6 rows
   - Required field

**Real-Time Validation**

```
field.element.addEventListener('blur', () => this.validateField(name));
field.element.addEventListener('input', () => this.clearError(name));
```

Listing 4.4: Validation on Blur Event

### 4.3.2 Google Maps Integration

The contact page embeds an interactive Google Map using an iframe:

- **Location**: Bizerte, Tunisia (Henna Bio)
- **Attributes**: allowfullscreen, loading="lazy"
- **Accessibility**: title="Google Maps location"
- **Privacy**: referrerpolicy="no-referrer-when-downgrade"

### 4.3.3 Contact Information Display

Structured presentation of contact details:

- Address with Font Awesome map-marker icon
- Clickable phone number (tel: protocol)
- Clickable email (mailto: protocol)
- Icon-based visual hierarchy

## 4.4 Feedback Page (feedback.html)

### 4.4.1 Testimonial Cards

Customer testimonials are displayed using Bootstrap card components featuring:

- Customer profile image (70x70px, border-radius: 50%)
- Star rating visualization
- Customer name and location
- Testimonial text
- Consistent card height for grid alignment

## 4.5 Checkout Page (checkout.html)

The checkout page simulates the order finalization process with:

- Order summary display
- Cart items review
- Total calculation
- Simulated payment form
- Order confirmation workflow

# Chapter 5

# Technical Implementation Details

## 5.1 Navigation Bar Specification

### 5.1.1 Structure

The navigation bar is implemented using Bootstrap 5 Navbar component with the following specifications:

- **Class**: `navbar navbar-expand-lg navbar-light fixed-top`
- **Position**: Fixed at top (position: fixed)
- **Background**: White with subtle box-shadow
- **Collapse breakpoint**: Large screens (lg - 992px)

### 5.1.2 Brand Logo

```html
<a class="navbar-brand" href="index.html">
    <i class="fas fa-leaf me-2" aria-hidden="true"></i>
    Organica Farm
</a>
```

Listing 5.1: Navbar Brand Implementation

**Styling**:

- Font: Playfair Display, serif
- Weight: 700 (bold)
- Size: 1.8rem
- Color: Primary green (#4a8e3c)

### 5.1.3 Navigation Links

| Label | Href | Purpose |
|---|---|---|
| Home | index.html | Navigate to homepage |
| About Us | aboutus.html | Farm information |
| Products | products.html | Product catalog |
| Feedback | feedback.html | Customer reviews |
| Contact | contact.html | Contact form |

### 5.1.4 Cart Button

```html
<a class="btn btn-outline-primary" href="#" aria-label="Shopping cart">
    <i class="fas fa-shopping-cart me-1"></i> Cart (0)
</a>
```

Listing 5.2: Shopping Cart Button

Dynamic count update via JavaScript:

```javascript
updateDisplay() {
    const cartElements = document.querySelectorAll('.btn-outline-
primary');
    const totalItems = this.getTotalItems();
    cartElements.forEach(element => {
        if (element.querySelector('.fa-shopping-cart')) {
            element.innerHTML = `<i class="fas fa-shopping-cart me
-1"></i>
                                 Cart (${totalItems})`;
        }
    });
}
```

Listing 5.3: Cart Count Update

### 5.1.5 Responsive Behavior

- **Desktop ($>$ 992px)**: Horizontal layout with all links visible
- **Tablet/Mobile ($<$ 992px)**: Collapsible hamburger menu
- **Hamburger Icon**: Bootstrap navbar-toggler component

## 5.2 Footer Specification

### 5.2.1 Footer Structure

The footer is divided into sections using Bootstrap grid system:

```html
<footer role="contentinfo">
    <div class="container">
        <div class="row">
            <div class="col-lg-4 mb-4">
                <!-- Brand & Social Icons -->
            </div>
            <div class="col-lg-2 col-md-4 mb-4">
```

```
                <!-- Google Maps -->
            </div>
        </div>
        <div class="text-center copyright">
            <!-- Copyright information -->
        </div>
    </div>
</footer>
```

Listing 5.4: Footer Layout Structure

### 5.2.2 Social Media Integration

**Supported Platforms**:

- Facebook (fab fa-facebook)
- Instagram (fab fa-instagram)
- Twitter (fab fa-twitter)
- Pinterest (fab fa-pinterest)

**Styling**:

```
.social-icons a {
    color: white;
    font-size: 1.5rem;
    margin-right: 15px;
    transition: color 0.3s ease;
}

.social-icons a:hover {
    color: var(--secondary-color);
}
```

Listing 5.5: Social Icons Hover Effect

### 5.2.3 Opening Hours Display

- Monday - Friday: 8AM $\rightarrow$ 5PM
- Saturday: 8AM $\rightarrow$ 2PM
- Sunday: Closed

## 5.3 Button Specifications

### 5.3.1 Primary Buttons

**Default State**:

```css
.btn-primary {
    background-color: var(--primary-color);
    border-color: var(--primary-color);
    transition: all 0.3s ease;
}
```

Listing 5.6: Primary Button Styling

**Hover State**:

- Background: Darker green (#3a7230)
- Transform: Slight scale increase
- Cursor: Pointer

**Focus State (Accessibility)**:

```css
.btn-primary:focus {
    box-shadow: 0 0 0 0.25rem rgba(74, 142, 60, 0.25);
    border-color: var(--primary-color);
    outline: none;
}
```

Listing 5.7: Focus Visible Outline

### 5.3.2 Outline Buttons

Used for secondary actions (Continue Shopping, etc.):

- Border: 1px solid primary color
- Background: Transparent
- Text: Primary color
- Hover: Filled background with white text

### 5.3.3 Quantity Control Buttons

Circular buttons for cart quantity manipulation:

```css
.quantity-btn {
    width: 32px;
    height: 32px;
    border: 1px solid var(--primary-color);
    background: white;
    border-radius: 50%;
    cursor: pointer;
    transition: all 0.3s ease;
}

.quantity-btn:hover {
    background-color: var(--primary-color);
    color: white;
    transform: scale(1.1);
}
```

Listing 5.8: Quantity Button Design

## 5.4 Icon Implementation (Font Awesome)

### 5.4.1 Icon Categories Used

| Category | Icons | Usage Context |
|---|---|---|
| Navigation | fa-leaf, fa-shopping-cart | Brand, Cart button |
| Features | fa-seedling, fa-tint, fa-bug | Feature cards |
| Social | fa-facebook, fa-instagram | Footer links |
| Actions | fa-paper-plane, fa-trash | Forms, Cart |
| UI | fa-check-circle, fa-exclamation | Notifications |

### 5.4.2 Icon Accessibility

All decorative icons use `aria-hidden="true"`:

```
<i class="fas fa-leaf me-2" aria-hidden="true"></i>
```

Listing 5.9: Icon Accessibility

For functional icons, descriptive `aria-label` is added to parent element.

## 5.5 Image Optimization

### 5.5.1 Product Images

- **Display size**: 200px height
- **Object-fit**: cover (maintains aspect ratio)
- **Loading**: lazy (where supported)
- **Alt text**: Descriptive for accessibility

### 5.5.2 Hero Background

- High-resolution image with gradient overlay
- CSS background-size: cover
- background-position: center
- Fallback color: Dark green

## 5.6 Form Implementation & Validation

### 5.6.1 Validation Architecture

The `FormValidator` class implements a modular validation system:

```
this.fields = {
    fullName: {
        element: this.form.querySelector('#fullName'),
        validators: [
            {
                test: (val) => val.trim().length >= 3,
                message: 'Name must contain at least 3 characters'
            },
            {
                test: (val) => /^[a-zA-Z\u00C0-\u00FF\s\-']+$/.test(val
),
                message: 'Name can only contain letters'
            }
        ]
    }
}
```

Listing 5.10: Validator Structure

### 5.6.2 Validation Timing

- **On Blur**: Full validation when user leaves field
- **On Input**: Clear existing errors while typing
- **On Submit**: Validate all fields before submission

### 5.6.3 Error Display

Bootstrap's `.is-invalid` class with custom feedback:

```
showError(fieldName, message) {
    const field = this.fields[fieldName].element;
    field.classList.add('is-invalid');

    let errorDiv = document.createElement('div');
    errorDiv.className = 'invalid-feedback';
    errorDiv.textContent = message;
    field.parentNode.insertBefore(errorDiv, field.nextSibling);
}
```

Listing 5.11: Error Display Logic

## 5.7 SEO Implementation

### 5.7.1 Meta Tags Strategy

Each page includes comprehensive meta tags:

#### Basic SEO

```
<title>Page Title - OrganicaFarm | Bizerte, Tunisia</title>
<meta name="description" content="Detailed page description">
<meta name="keywords" content="relevant, keywords, here">
```

```html
<meta name="author" content="Nour ZAGHOUANI">
```
Listing 5.12: Standard Meta Tags

#### Open Graph (Social Media)

```html
<meta property="og:type" content="website">
<meta property="og:url" content="https://organicafarm.tn/">
<meta property="og:title" content="OrganicaFarm">
<meta property="og:description" content="Fresh organic products">
<meta property="og:image" content="og-image.jpg">
```
Listing 5.13: Open Graph Tags

#### Twitter Cards

```html
<meta name="twitter:card" content="summary_large_image">
<meta name="twitter:title" content="OrganicaFarm">
<meta name="twitter:description" content="Organic produce">
```
Listing 5.14: Twitter Meta Tags

### 5.7.2 Semantic HTML

Proper use of HTML5 semantic elements:

- `<header>`, `<nav>`, `<main>`, `<footer>`
- `<section>`, `<article>` for content organization
- Heading hierarchy (h1 → h2 → h3) maintained
- Single `<h1>` per page

## 5.8 Code Separation (HTML/CSS/JS)

### 5.8.1 Separation of Concerns Principle

The project strictly adheres to separation of concerns:

#### HTML - Structure

- Contains only semantic markup
- No inline styles (except where absolutely necessary)
- No inline JavaScript event handlers
- Uses data attributes for JavaScript hooks

**CSS - Presentation**

- All styling in external `.css` files
- CSS Variables for theming (`:root`)
- Modular approach (page-specific stylesheets)
- BEM-like naming convention where applicable

**JavaScript - Behavior**

- All logic in external `.js` files
- Event listeners attached programmatically
- No inline `onclick` attributes
- Object-Oriented Programming with ES6 classes

### 5.8.2 CSS Architecture

**main.css** serves as the global stylesheet:

```css
:root {
    --primary-color: #4a8e3c;
    --primary-dark: #3a7230;
    --secondary-color: #f8a33c;
    --light-color: #f9f9f9;
    --dark-color: #333;
    --text-color: #444;
    --white: #ffffff;
}
```

Listing 5.15: CSS Variables Definition

**Benefits**:

- Easy theme customization
- Consistent color usage
- Centralized configuration
- Runtime color changes possible

### 5.8.3 JavaScript Architecture

**Object-Oriented Approach**:

1. `ShoppingCart` class (main.js)
2. `ProductFilter` class (products.js)
3. `FormValidator` class (contact.js)

**Encapsulation Example**:

```
class ShoppingCart {
    constructor() {
        this.items = this.loadCart();
        this.updateDisplay();
        this.attachCartButtonListener();
    }

    // Private method (by convention)
    loadCart() {
        const saved = localStorage.getItem('organica-cart');
        return saved ? JSON.parse(saved) : [];
    }

    // Public method
    addItem(product) {
        // Implementation
    }
}
```

Listing 5.16: Class-Based Architecture

# Chapter 6

# Future Enhancements

## 6.1 Backend Integration

### 6.1.1 Proposed Technology Stack

- **Backend Framework**: Node.js with Express.js or Python with Django/Flask
- **Database**: MongoDB (NoSQL) or PostgreSQL (SQL)
- **API**: RESTful API or GraphQL
- **Authentication**: JWT (JSON Web Tokens)

### 6.1.2 Required Backend Endpoints

| **Endpoint** | **Method** | **Purpose** |
|---|---|---|
| /api/products | GET | Fetch product catalog |
| /api/products/:id | GET | Get single product |
| /api/cart | POST | Save cart to database |
| /api/orders | POST | Submit order |
| /api/users/register | POST | User registration |
| /api/users/login | POST | User authentication |

## 6.2 Additional Features

### 6.2.1 User Account System

- User registration and login
- Profile management
- Order history
- Saved addresses
- Wishlist functionality

### 6.2.2 Payment Integration

- Stripe payment gateway
- PayPal integration
- Multiple currency support
- Invoice generation

### 6.2.3 Advanced Features

- Product search functionality
- Advanced filtering (price range, availability)
- Product reviews and ratings
- Email notifications
- Newsletter subscription
- Inventory management system

# Chapter 7

# Conclusion

## 7.1 Project Achievements

The OrganicaFarm e-commerce platform successfully demonstrates the implementation of a professional, fully-functional frontend application. Key achievements include:

1. **Complete E-Commerce Experience**: All essential features of an e-commerce platform have been implemented on the client side, including product browsing, filtering, shopping cart management, and checkout simulation.
2. **Modern Web Standards**: The project adheres to current web development best practices, utilizing semantic HTML5, CSS3 with custom properties, and modern JavaScript (ES6+).
3. **Accessibility Focus**: WCAG 2.1 Level AA compliance ensures the application is usable by individuals with disabilities, demonstrating social responsibility.
4. **Responsive Design**: Flawless functionality across all device types provides an optimal user experience regardless of screen size.
5. **Code Quality**: Clean, modular, and well-documented code follows industry standards and separation of concerns principles.
6. **SEO Optimization**: Comprehensive meta tag implementation and semantic markup ensure maximum search engine visibility.

## 7.2 Learning Outcomes

Throughout the development of this project, valuable skills and knowledge were acquired:

- Advanced CSS layouts using Bootstrap Grid System
- Object-Oriented JavaScript programming
- Client-side state management with LocalStorage
- Form validation and user input handling
- Web accessibility implementation

- SEO best practices
- Responsive design principles
- Cross-browser compatibility techniques

## 7.3 Final Remarks

OrganicaFarm represents a solid foundation for a full-stack e-commerce application. The current frontend implementation is production-ready for static deployment and provides an excellent user experience. With the addition of backend services and database integration, this platform can evolve into a complete e-commerce solution capable of handling real transactions and user accounts.

The project successfully demonstrates that modern web applications can be built with clean, maintainable code while adhering to web standards, accessibility guidelines, and user experience best practices.

# Appendix A

# Code Samples

## A.1 Shopping Cart Class (Complete)

```
class ShoppingCart {
    constructor() {
        this.items = this.loadCart();
        this.updateDisplay();
        this.attachCartButtonListener();
    }

    loadCart() {
        const saved = localStorage.getItem('organica-cart');
        return saved ? JSON.parse(saved) : [];
    }

    saveCart() {
        localStorage.setItem('organica-cart', JSON.stringify(this.items
    ));
    }

    addItem(product) {
        const existing = this.items.find(item => item.id === product.id
    );

        if (existing) {
            existing.quantity++;
        } else {
            this.items.push({
                id: product.id,
                name: product.name,
                price: product.price,
                quantity: 1,
                image: product.image || 'default.jpg'
            });
        }

        this.saveCart();
        this.updateDisplay();
        this.showNotification(`${product.name} added to cart`);
    }

    removeItem(productId) {
        this.items = this.items.filter(item => item.id !== productId);
        this.saveCart();
        this.updateDisplay();
        this.renderCartModal();
    }

    updateQuantity(productId, newQuantity) {
        const item = this.items.find(item => item.id === productId);
        if (item) {
            if (newQuantity <= 0) {
                this.removeItem(productId);
            } else {
                item.quantity = newQuantity;
                this.saveCart();
                this.updateDisplay();
                this.renderCartModal();
            }
        }
    }

    getTotalItems() {
        return this.items.reduce((sum, item) => sum + item.quantity, 0);
    }

    getTotalPrice() {
        return this.items.reduce((sum, item) =>
            sum + (item.price * item.quantity), 0
        ).toFixed(2);
    }

    updateDisplay() {
        const cartElements = document.querySelectorAll('.btn-outline-primary');
        const totalItems = this.getTotalItems();

        cartElements.forEach(element => {
            if (element.querySelector('.fa-shopping-cart')) {
                element.innerHTML =
                `<i class="fas fa-shopping-cart me-1"></i> Cart (${totalItems})`;
            }
        });
    }
}

// Initialize cart when DOM is ready
document.addEventListener('DOMContentLoaded', function () {
    window.cart = new ShoppingCart();
});
```

Listing A.1: ShoppingCart Complete Implementation

## A.2 Product Filter Class (Complete)

```
class ProductFilter {
    constructor() {
        this.filterButtons = document.querySelectorAll('[data-filter]');
        this.products = document.querySelectorAll('#product-grid > div[data-category]');
        this.init();
    }

    init() {
        if (this.filterButtons.length === 0) return;
        this.attachEventListeners();
    }

    attachEventListeners() {
        this.filterButtons.forEach(button => {
            button.addEventListener('click', (e) => this.filter(e));
        });
    }

    filter(event) {
        const category = event.target.getAttribute('data-filter');
        this.updateActiveButton(event.target);
        this.filterProducts(category);
    }

    updateActiveButton(activeBtn) {
        this.filterButtons.forEach(btn => btn.classList.remove('active'));
        activeBtn.classList.add('active');
    }

    filterProducts(category) {
        this.products.forEach(product => {
            const productCategory = product.getAttribute('data-category');
            const visible = category === 'all' ||
                           productCategory.includes(category);
            product.style.display = visible ? 'block' : 'none';
        });
    }
}
```

Listing A.2: ProductFilter Implementation

## A.3 Form Validator Class (Complete)

```
class FormValidator {
    constructor(formSelector) {
        this.form = document.querySelector(formSelector);
        if (!this.form) return;

        this.fields = {
            fullName: {
                element: this.form.querySelector('#fullName'),
                validators: [
                    {
                        test: (val) => val.trim().length >= 3,
                        message: 'Name must contain at least 3 characters'
                    },
                    {
                        test: (val) => /^[a-zA-Z\u00C0-\u00FF\s\-']+$/.test(val),
                        message: 'Name can only contain letters'
                    }
                ]
            },
            email: {
                element: this.form.querySelector('#email'),
                validators: [
                    {
                        test: (val) => /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(val),
                        message: 'Please enter a valid email address'
                    }
                ]
            },
            message: {
                element: this.form.querySelector('#message'),
                validators: [
                    {
                        test: (val) => val.trim().length >= 10,
                        message: 'Message must contain at least 10 characters'
                    }
                ]
            }
        };

        this.init();
    }

    validateField(fieldName) {
        const field = this.fields[fieldName];
        if (!field || !field.element) return true;

        const value = field.element.value;

        for (const validator of field.validators) {
            if (!validator.test(value)) {
                this.showError(fieldName, validator.message);
                return false;
            }
        }

        this.clearError(fieldName);
        return true;
    }
}
```

Listing A.3: FormValidator Implementation

# Appendix B

# CSS Specifications

## B.1 Color Palette

| Variable Name | Hex Value | Usage |
|---|---|---|
| –primary-color | #4a8e3c | Buttons, links, brand |
| –primary-dark | #3a7230 | Hover states |
| –secondary-color | #f8a33c | Accents, badges |
| –light-color | #f9f9f9 | Backgrounds |
| –dark-color | #333 | Headings, text |
| –text-color | #444 | Body text |
| –white | #ffffff | Backgrounds, text |

## B.2 Typography System

### B.2.1 Font Families

- **Headings**: 'Playfair Display', serif
- **Body**: 'Montserrat', sans-serif

### B.2.2 Font Sizes

| Element | Size |
|---|---|
| h1 | 2.5rem (40px) |
| h2 | 2rem (32px) |
| h3 | 1.75rem (28px) |
| h4 | 1.5rem (24px) |
| h5 | 1.25rem (20px) |
| body | 1rem (16px) |
| small | 0.875rem (14px) |

# Appendix C

# Project Statistics

## C.1 File Statistics

| **File Type** | **Count** | **Total Size** |
|---|---:|---:|
| HTML Files | 7 | 55 KB |
| CSS Files | 4 | 15 KB |
| JavaScript Files | 4 | 16 KB |
| Image Files | Variable | N/A |
| **Total** | **15+** | **86 KB** |

## C.2 Code Metrics

- **Total Lines of Code**: 1,500+
- **JavaScript Classes**: 3
- **CSS Rules**: 200+
- **Pages**: 7

# Appendix D

# References & Resources

## D.1 Technologies & Frameworks

1. Bootstrap 5.3.0 - https://getbootstrap.com/
2. Font Awesome 6.4.0 - https://fontawesome.com/
3. Google Fonts - https://fonts.google.com/

## D.2 Standards & Guidelines

1. WCAG 2.1 Guidelines - https://www.w3.org/WAI/WCAG21/quickref/
2. HTML5 Specification - https://html.spec.whatwg.org/
3. CSS3 Specification - https://www.w3.org/Style/CSS/
4. ECMAScript 2015+ - https://www.ecma-international.org/

## D.3 Validation Tools

1. W3C HTML Validator - https://validator.w3.org/
2. W3C CSS Validator - https://jigsaw.w3.org/css-validator/
3. Google Lighthouse - https://developers.google.com/web/tools/lighthouse