BueaDelights Django Project Setup Guide

"Local Flavors at Your Fingertips"

Table of Contents

Prerequisites

Before starting, ensure you have:

- Python 3.9+ installed
- Node.js and npm installed (for Tailwind CSS)
- Git installed
- A code editor (VS Code recommended)

1. Virtual Environment Setup

Step 1: Create Project Directory

bash

```bash
# Create main project directory
mkdir bueadelights-project
cd bueadelights-project

# Create virtual environment
python -m venv venv

# Activate virtual environment
# On Windows:
venv\Scripts\activate

# On macOS/Linux:
source venv/bin/activate
```

Step 2: Install Core Dependencies

bash

```bash
# Upgrade pip
pip install --upgrade pip

# Install Django and essential packages
pip install django==4.2
pip install pillow
pip install python-decouple
pip install django-cors-headers
pip install whitenoise
pip install gunicorn
pip install psycopg2-binary

# Create requirements.txt
pip freeze > requirements.txt
```

2. Django Project Creation

Step 1: Create Django Project

bash

```bash
# Create Django project
django-admin startproject bueadelights .

# Navigate to project directory
cd bueadelights
```

Step 2: Create Backend App

bash

```
# Create backend application
python manage.py startapp backend

# Return to root directory
cd ..
```

Step 3: Configure Initial Settings

Create `.env` file in root directory:

env

```
# .env
SECRET_KEY=your-secret-key-here
DEBUG=True
ALLOWED_HOSTS=localhost,127.0.0.1
DATABASE_URL=sqlite:///db.sqlite3

# WhatsApp Configuration
WHATSAPP_NUMBER=+237699808260
BUSINESS_NAME=BueaDelights
BUSINESS_EMAIL=info@bueadelights.com

# Payment Configuration
NOUPIA_API_KEY=your-noupia-api-key
NOUPIA_MERCHANT_ID=your-merchant-id

# Email Configuration
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USE_TLS=True
EMAIL_HOST_USER=your-email@gmail.com
EMAIL_HOST_PASSWORD=your-app-password
```

Step 4: Update Django Settings

Update `bueadelights/settings.py`:

python

```python
import os
from decouple import config
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = config('SECRET_KEY', default='django-insecure-change-me')

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = config('DEBUG', default=True, cast=bool)

ALLOWED_HOSTS = config('ALLOWED_HOSTS', default='localhost,127.0.0.1').split(',')

# Application definition
DJANGO_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

THIRD_PARTY_APPS = [
    'corsheaders',
]

LOCAL_APPS = [
    'backend',
]

INSTALLED_APPS = DJANGO_APPS + THIRD_PARTY_APPS + LOCAL_APPS

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
```

```
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'bueadelights.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'backend.context_processors.business_context',
            ],
        },
    },
]

WSGI_APPLICATION = 'bueadelights.wsgi.application'

# Database
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

# Password validation
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

```
# Internationalization
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'Africa/Douala'
USE_I18N = True
USE_TZ = True

# Static files (CSS, JavaScript, Images)
STATIC_URL = '/static/'
STATIC_ROOT = BASE_DIR / 'staticfiles'
STATICFILES_DIRS = [
    BASE_DIR / 'static',
]

# Media files
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'

# Default primary key field type
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

# Business Configuration
WHATSAPP_NUMBER = config('WHATSAPP_NUMBER', default='+237699808260')
BUSINESS_NAME = config('BUSINESS_NAME', default='BueaDelights')
BUSINESS_EMAIL = config('BUSINESS_EMAIL', default='info@bueadelights.com')

# Email Configuration
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = config('EMAIL_HOST', default='smtp.gmail.com')
EMAIL_PORT = config('EMAIL_PORT', default=587, cast=int)
EMAIL_USE_TLS = config('EMAIL_USE_TLS', default=True, cast=bool)
EMAIL_HOST_USER = config('EMAIL_HOST_USER', default='')
EMAIL_HOST_PASSWORD = config('EMAIL_HOST_PASSWORD', default='')

# Payment Configuration
NOUPIA_API_KEY = config('NOUPIA_API_KEY', default='')
NOUPIA_MERCHANT_ID = config('NOUPIA_MERCHANT_ID', default='')

# Security Settings for Production
if not DEBUG:
    SECURE_BROWSER_XSS_FILTER = True
    SECURE_CONTENT_TYPE_NOSNIFF = True
    SECURE_HSTS_INCLUDE_SUBDOMAINS = True
    SECURE_HSTS_SECONDS = 86400
    SECURE_REDIRECT_EXEMPT = []
    SECURE_SSL_REDIRECT = True
```

```
SESSION_COOKIE_SECURE = True
CSRF_COOKIE_SECURE = True
```

3. Backend App Configuration

Step 1: Create Backend App Structure

bash

```
# Create directories for backend app
mkdir -p backend/templates/backend
mkdir -p backend/static/backend/css
mkdir -p backend/static/backend/js
mkdir -p backend/static/backend/images
mkdir -p backend/management/commands
mkdir -p backend/templatetags
```

Step 2: Create Models

Update `backend/models.py`:

python

```python
from django.db import models
from django.utils.text import slugify
from django.urls import reverse
import uuid

class Category(models.Model):
    name = models.CharField(max_length=100)
    slug = models.SlugField(unique=True, blank=True)
    description = models.TextField(blank=True)
    image = models.ImageField(upload_to='categories/', blank=True, null=True)
    is_active = models.BooleanField(default=True)
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        verbose_name_plural = "Categories"
        ordering = ['name']

    def save(self, *args, **kwargs):
        if not self.slug:
            self.slug = slugify(self.name)
        super().save(*args, **kwargs)

    def __str__(self):
        return self.name

class Product(models.Model):
    name = models.CharField(max_length=200)
    slug = models.SlugField(unique=True, blank=True)
    description = models.TextField()
    price = models.DecimalField(max_digits=10, decimal_places=2)
    category = models.ForeignKey(Category, on_delete=models.CASCADE, related_name='products')
    image = models.ImageField(upload_to='products/')
    is_available = models.BooleanField(default=True)
    is_featured = models.BooleanField(default=False)
    stock_quantity = models.PositiveIntegerField(default=0)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        ordering = ['-created_at']

    def save(self, *args, **kwargs):
        if not self.slug:
            self.slug = slugify(self.name)
        super().save(*args, **kwargs)
```

```
        super().save(*args, **kwargs)

    def get_absolute_url(self):
        return reverse('backend:product_detail', kwargs={'slug': self.slug})

    def __str__(self):
        return self.name

class ProductImage(models.Model):
    product = models.ForeignKey(Product, on_delete=models.CASCADE, related_name='gallery')
    image = models.ImageField(upload_to='products/gallery/')
    alt_text = models.CharField(max_length=200, blank=True)
    is_primary = models.BooleanField(default=False)

    def __str__(self):
        return f"{self.product.name} - Gallery Image"

class Order(models.Model):
    ORDER_STATUS_CHOICES = [
        ('pending', 'Pending'),
        ('confirmed', 'Confirmed'),
        ('preparing', 'Preparing'),
        ('ready', 'Ready'),
        ('delivered', 'Delivered'),
        ('cancelled', 'Cancelled'),
    ]

    PAYMENT_STATUS_CHOICES = [
        ('pending', 'Pending'),
        ('completed', 'Completed'),
        ('failed', 'Failed'),
        ('refunded', 'Refunded'),
    ]

    PAYMENT_METHOD_CHOICES = [
        ('mobile_money', 'Mobile Money'),
        ('cash_on_delivery', 'Cash on Delivery'),
    ]

    order_id = models.CharField(max_length=20, unique=True, blank=True)
    customer_name = models.CharField(max_length=100)
    customer_phone = models.CharField(max_length=20)
    customer_email = models.EmailField(blank=True, null=True)
    customer_location = models.TextField()
    total_amount = models.DecimalField(max_digits=10, decimal_places=2)
    delivery_fee = models.DecimalField(max_digits=6, decimal_places=2, default=1500)
    payment_method = models.CharField(max_length=20, choices=PAYMENT_METHOD_CHOICES)
```

```
    payment_status = models.CharField(max_length=20, choices=PAYMENT_STATUS_CHOICES, default='p
    order_status = models.CharField(max_length=20, choices=ORDER_STATUS_CHOICES, default='pendi
    special_instructions = models.TextField(blank=True)
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ['-created_at']

    def save(self, *args, **kwargs):
        if not self.order_id:
            self.order_id = f"BD{uuid.uuid4().hex[:8].upper()}"
        super().save(*args, **kwargs)

    def __str__(self):
        return f"Order {self.order_id} - {self.customer_name}"

class OrderItem(models.Model):
    order = models.ForeignKey(Order, on_delete=models.CASCADE, related_name='items')
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField()
    unit_price = models.DecimalField(max_digits=10, decimal_places=2)
    total_price = models.DecimalField(max_digits=10, decimal_places=2)

    def save(self, *args, **kwargs):
        self.total_price = self.quantity * self.unit_price
        super().save(*args, **kwargs)

    def __str__(self):
        return f"{self.quantity}x {self.product.name}"

class ContactMessage(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField()
    phone = models.CharField(max_length=20, blank=True)
    subject = models.CharField(max_length=200)
    message = models.TextField()
    is_read = models.BooleanField(default=False)
    admin_response = models.TextField(blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    responded_at = models.DateTimeField(blank=True, null=True)

    class Meta:
        ordering = ['-created_at']

    def __str__(self):
        return f"{self.name} - {self.subject}"
```

```python
class CateringInquiry(models.Model):
    EVENT_TYPE_CHOICES = [
        ('wedding', 'Wedding'),
        ('birthday', 'Birthday'),
        ('anniversary', 'Anniversary'),
        ('corporate', 'Corporate Event'),
        ('traditional', 'Traditional Ceremony'),
        ('graduation', 'Graduation'),
        ('religious', 'Religious Celebration'),
        ('family', 'Family Gathering'),
        ('other', 'Other'),
    ]

    STATUS_CHOICES = [
        ('new', 'New'),
        ('contacted', 'Contacted'),
        ('quoted', 'Quoted'),
        ('confirmed', 'Confirmed'),
        ('completed', 'Completed'),
        ('cancelled', 'Cancelled'),
    ]

    name = models.CharField(max_length=100)
    phone = models.CharField(max_length=20)
    email = models.EmailField(blank=True)
    event_type = models.CharField(max_length=20, choices=EVENT_TYPE_CHOICES)
    guest_count = models.PositiveIntegerField()
    preferred_date = models.DateField()
    preferred_time = models.TimeField()
    location = models.TextField()
    special_requirements = models.TextField(blank=True)
    status = models.CharField(max_length=20, choices=STATUS_CHOICES, default='new')
    admin_notes = models.TextField(blank=True)
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ['-created_at']

    def __str__(self):
        return f"{self.name} - {self.event_type} ({self.guest_count} guests)"

class BusinessSettings(models.Model):
    business_name = models.CharField(max_length=200, default='BueaDelights')
    business_description = models.TextField(default='Local Flavors at Your Fingertips')
    phone = models.CharField(max_length=20, default='+237699808260')
    email = models.EmailField(default='info@bueadelights.com')
```

```
    address = models.TextField(default='Buea, Southwest Region, Cameroon')
    operating_hours = models.TextField(default='Monday - Sunday: 8:00 AM - 10:00 PM')
    delivery_fee = models.DecimalField(max_digits=6, decimal_places=2, default=1500)
    delivery_areas = models.TextField(default='Buea, Limbe, Tiko, Douala')
    is_accepting_orders = models.BooleanField(default=True)

    class Meta:
        verbose_name = "Business Settings"
        verbose_name_plural = "Business Settings"

    def __str__(self):
        return self.business_name

    def save(self, *args, **kwargs):
        if not self.pk and BusinessSettings.objects.exists():
            raise ValueError('There can be only one BusinessSettings instance')
        return super().save(*args, **kwargs)
```

Step 3: Create Context Processor

Create `backend/context_processors.py`:

python

```
from .models import BusinessSettings, Category

def business_context(request):
    """Add business settings and common data to all templates."""
    try:
        business = BusinessSettings.objects.first()
    except BusinessSettings.DoesNotExist:
        business = None

    categories = Category.objects.filter(is_active=True)

    return {
        'business': business,
        'categories': categories,
    }
```

Step 4: Create Template Tags

Create `backend/templatetags/__init__.py` (empty file)

Create `backend/templatetags/cart_tags.py`:

python

```python
from django import template

register = template.Library()

@register.filter
def multiply(value, arg):
    """Multiply the value by the argument."""
    try:
        return float(value) * float(arg)
    except (ValueError, TypeError):
        return 0

@register.filter
def cart_item_count(cart):
    """Return the total number of items in cart."""
    if not cart:
        return 0
    return sum(item['quantity'] for item in cart.values())

@register.filter
def cart_total(cart):
    """Return the total price of items in cart."""
    if not cart:
        return 0
    total = 0
    for item in cart.values():
        total += item['price'] * item['quantity']
    return total

@register.simple_tag
def whatsapp_message(order_details):
    """Generate WhatsApp message for order."""
    message = f"Hello! I'd like to place an order:\n\n"
    message += f"Order Details:\n{order_details}\n\n"
    message += "Please confirm my order. Thank you!"
    return message
```

4. Tailwind CSS Integration

Step 1: Initialize Node.js Project

bash

```bash
# Initialize npm project
npm init -y

# Install Tailwind CSS
npm install -D tailwindcss
npm install -D @tailwindcss/forms
npm install -D autoprefixer

# Initialize Tailwind
npx tailwindcss init
```

Step 2: Configure Tailwind

Update `tailwind.config.js`:

javascript

```javascript
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    './templates/**/*.html',
    './backend/templates/**/*.html',
    './backend/static/**/*.js',
  ],
  theme: {
    extend: {
      colors: {
        'forest-green': '#228B22',
        'warm-red': '#DC143C',
        'golden-yellow': '#FFD700',
        'neutral-gray': '#F5F5F5',
        'cameroon-green': '#009639',
        'cameroon-red': '#CE1126',
        'cameroon-yellow': '#FCDD09',
      },
      fontFamily: {
        'inter': ['Inter', 'sans-serif'],
        'roboto': ['Roboto', 'sans-serif'],
        'playfair': ['Playfair Display', 'serif'],
      },
      animation: {
        'fade-in': 'fadeIn 0.5s ease-in-out',
        'slide-up': 'slideUp 0.3s ease-out',
        'bounce-gentle': 'bounceGentle 2s infinite',
      },
      keyframes: {
        fadeIn: {
          '0%': { opacity: '0' },
          '100%': { opacity: '1' },
        },
        slideUp: {
          '0%': { transform: 'translateY(10px)', opacity: '0' },
          '100%': { transform: 'translateY(0)', opacity: '1' },
        },
        bounceGentle: {
          '0%, 20%, 50%, 80%, 100%': {
            transform: 'translateY(0)',
          },
          '40%': {
            transform: 'translateY(-10px)',
          },
```

```
            60%': {
              transform: 'translateY(-5px)',
            },
          },
        },
      },
    },
  },
  plugins: [
    require('@tailwindcss/forms'),
  ],
}
```

Step 3: Create CSS Files

Create `static/css/input.css`:

```css
@tailwind base;
@tailwind components;
@tailwind utilities;

/* Custom Components */
@layer components {
  .btn {
    @apply px-4 py-2 rounded-lg font-medium transition-all duration-300 ease-in-out;
  }

  .btn-primary {
    @apply bg-forest-green text-white hover:bg-green-700 focus:ring-4 focus:ring-green-300;
  }

  .btn-secondary {
    @apply bg-warm-red text-white hover:bg-red-700 focus:ring-4 focus:ring-red-300;
  }

  .btn-outline {
    @apply border-2 border-forest-green text-forest-green hover:bg-forest-green hover:text-whit
  }

  .card {
    @apply bg-white rounded-xl shadow-lg hover:shadow-xl transition-shadow duration-300;
  }

  .input-field {
    @apply w-full px-4 py-3 border border-gray-300 rounded-lg focus:ring-2 focus:ring-forest-gr
  }

  .product-card {
    @apply card overflow-hidden hover:scale-105 transition-transform duration-300;
  }

  .hero-section {
    @apply bg-gradient-to-r from-forest-green to-green-600 text-white py-20;
  }

  .section-title {
    @apply text-3xl md:text-4xl font-bold text-gray-800 mb-8 text-center;
  }

  .whatsapp-float {
    @apply fixed bottom-6 right-6 bg-green-500 text-white p-4 rounded-full shadow-lg hover:bg-g
  }
```

```
  }

  .notification {
    @apply fixed top-4 right-4 p-4 rounded-lg shadow-lg z-50 animate-slide-up;
  }

  .notification-success {
    @apply notification bg-green-500 text-white;
  }

  .notification-error {
    @apply notification bg-red-500 text-white;
  }

  .loader {
    @apply inline-block w-4 h-4 border-2 border-white border-t-transparent rounded-full animate
  }
}

/* Custom animations for mobile-first design */
@layer utilities {
  .animate-pulse-gentle {
    animation: pulse 3s cubic-bezier(0.4, 0, 0.6, 1) infinite;
  }

  .hover-lift {
    transition: transform 0.3s ease;
  }

  .hover-lift:hover {
    transform: translateY(-5px);
  }
}

/* Mobile-first responsive utilities */
@media (max-width: 768px) {
  .mobile-menu {
    @apply transform transition-transform duration-300 ease-in-out;
  }

  .mobile-menu.hidden {
    @apply -translate-x-full;
  }

  .mobile-menu.visible {
    @apply translate-x-0;
  }
```

```
}

/* Print styles */
@media print {
  .no-print {
    display: none !important;
  }
}
```

Step 4: Build Script

Add to `package.json`:

```
json

{
  "scripts": {
    "build-css": "tailwindcss -i ./static/css/input.css -o ./static/css/output.css --watch",
    "build-css-prod": "tailwindcss -i ./static/css/input.css -o ./static/css/output.css --minif
  }
}
```

5. Hot Reload Configuration

Step 1: Install Django Extensions

```
bash

pip install django-extensions
pip install watchdog
```

Step 2: Update Settings

Add to `INSTALLED_APPS` in settings.py:

```
python

INSTALLED_APPS = [
    # ... other apps
    'django_extensions',
]
```

Step 3: Create Development Script

Create `dev_server.py`:

python

```python
#!/usr/bin/env python
import os
import sys
import subprocess
import threading
import time

def run_django():
    """Run Django development server."""
    os.system('python manage.py runserver 0.0.0.0:8000')

def build_css():
    """Build Tailwind CSS in watch mode."""
    os.system('npm run build-css')

def main():
    """Run both Django and Tailwind build in parallel."""
    print("🚀 Starting BueaDelights Development Server...")
    print("📱 Django: http://localhost:8000")
    print("🎨 Tailwind CSS: Watching for changes...")

    # Start CSS build in background
    css_thread = threading.Thread(target=build_css)
    css_thread.daemon = True
    css_thread.start()

    # Give CSS build time to start
    time.sleep(2)

    # Start Django server
    run_django()

if __name__ == '__main__':
    main()
```

Make it executable:

bash

```bash
chmod +x dev_server.py
```

6. Development Environment

Step 1: Create Base Templates

Create `templates/base.html`:

html

```
<!DOCTYPE html>
<html lang="en" class="scroll-smooth">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="{% block meta_description %}BueaDelights - Local Flavors
    <meta name="keywords" content="{% block meta_keywords %}Buea food delivery, Cameroonian cui

    <title>{% block title %}BueaDelights - Local Flavors at Your Fingertips{% endblock %}</titl

    <!-- Google Fonts -->
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&family=

    <!-- Tailwind CSS -->
    {% load static %}
    <link href="{% static 'css/output.css' %}" rel="stylesheet">

    <!-- Favicon -->
    <link rel="icon" type="image/png" href="{% static 'backend/images/favicon.png' %}">

    <!-- Additional CSS -->
    {% block extra_css %}{% endblock %}

    <!-- Meta tags for social sharing -->
    <meta property="og:title" content="{% block og_title %}BueaDelights - Local Flavors at Your
    <meta property="og:description" content="{% block og_description %}Authentic Cameroonian cu
    <meta property="og:image" content="{% block og_image %}{% static 'backend/images/og-image.j
    <meta property="og:url" content="{{ request.build_absolute_uri }}">
    <meta name="twitter:card" content="summary_large_image">
</head>
<body class="font-inter bg-gray-50 min-h-screen">
    <!-- Navigation -->
    {% include 'includes/navbar.html' %}

    <!-- Main Content -->
    <main class="{% block main_class %}{% endblock %}">
        {% block content %}{% endblock %}
    </main>

    <!-- Footer -->
    {% include 'includes/footer.html' %}

    <!-- WhatsApp Float Button -->
```

```
    <!-- WhatsApp Float Button -->
    {% include 'includes/whatsapp_float.html' %}

    <!-- Notifications -->
    <div id="notifications" class="fixed top-4 right-4 z-50 space-y-2"></div>

    <!-- JavaScript -->
    <script src="{% static 'backend/js/main.js' %}"></script>
    {% block extra_js %}{% endblock %}

    <!-- Hot Reload for Development -->
    {% if debug %}
    <script>
        // Simple auto-reload for development
        if (window.location.hostname === 'localhost' || window.location.hostname === '127.0.0.1
            let lastModified = document.lastModified;
            setInterval(() => {
                fetch(window.location.href, { method: 'HEAD' })
                    .then(response => {
                        if (response.headers.get('last-modified') !== lastModified) {
                            window.location.reload();
                        }
                    })
                    .catch(() => {}); // Ignore errors
            }, 1000);
        }
    </script>
    {% endif %}
</body>
</html>
```

Step 2: Create Navigation Template

Create `templates/includes/navbar.html`:

html

```
{% load static %}
{% load cart_tags %}

<nav class="bg-white shadow-lg sticky top-0 z-40">
    <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
        <div class="flex justify-between items-center h-16">
            <!-- Logo -->
            <div class="flex-shrink-0 flex items-center">
                <a href="{% url 'backend:home' %}" class="flex items-center space-x-2">
                    <img src="{% static 'backend/images/logo.png' %}" alt="BueaDelights" class=
                    <span class="font-playfair font-bold text-xl text-forest-green">BueaDelight
                </a>
            </div>

            <!-- Desktop Navigation -->
            <div class="hidden md:block">
                <div class="ml-10 flex items-baseline space-x-4">
                    <a href="{% url 'backend:home' %}" class="nav-link">Home</a>
                    <a href="{% url 'backend:products' %}" class="nav-link">Menu</a>
                    <a href="{% url 'backend:catering' %}" class="nav-link">Catering</a>
                    <a href="{% url 'backend:about' %}" class="nav-link">About</a>
                    <a href="{% url 'backend:contact' %}" class="nav-link">Contact</a>
                </div>
            </div>

            <!-- Cart and Mobile Menu -->
            <div class="flex items-center space-x-4">
                <!-- Cart -->
                <a href="{% url 'backend:cart' %}" class="relative p-2 text-gray-600 hover:text
                    <svg class="w-6 h-6" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d
                    </svg>
                    <span id="cart-count" class="absolute -top-1 -right-1 bg-warm-red text-whit
                        {{ request.session.cart|cart_item_count }}
                    </span>
                </a>

                <!-- Mobile Menu Button -->
                <button id="mobile-menu-btn" class="md:hidden p-2 text-gray-600 hover:text-fore
                    <svg class="w-6 h-6" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                        <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d
                    </svg>
                </button>
            </div>
        </div>
```

```
        </div>
    </div>

    <!-- Mobile Navigation -->
    <div id="mobile-menu" class="md:hidden hidden bg-white border-t">
        <div class="px-2 pt-2 pb-3 space-y-1 sm:px-3">
            <a href="{% url 'backend:home' %}" class="mobile-nav-link">Home</a>
            <a href="{% url 'backend:products' %}" class="mobile-nav-link">Menu</a>
            <a href="{% url 'backend:catering' %}" class="mobile-nav-link">Catering</a>
            <a href="{% url 'backend:about' %}" class="mobile-nav-link">About</a>
            <a href="{% url 'backend:contact' %}" class="mobile-nav-link">Contact</a>
        </div>
    </div>
</nav>

<style>
.nav-link {
    @apply px-3 py-2 rounded-md text-sm font-medium text-gray-700 hover:text-forest-green hover
}

.mobile-nav-link {
    @apply block px-3 py-2 rounded-md text-base font-medium text-gray-700 hover:text-forest-gre
}
</style>
```

Step 3: Create WhatsApp Float Button

Create `templates/includes/whatsapp_float.html`:

```
html

{% load static %}

<div id="whatsapp-float" class="whatsapp-float hover:scale-110 transition-transform">
    <a href="https://wa.me/{{ business.phone|default:'+237699808260' }}?text=Hello%20BueaDeligh
       target="_blank"
       rel="noopener noreferrer"
       class="flex items-center justify-center w-14 h-14">
        <svg class="w-8 h-8" fill="currentColor" viewBox="0 0 24 24">
            <path d="M17.472 14.382c-.297-.149-1.758-.867-2.03-.967-.273-.099-.471-.148-.67.15-
        </svg>
    </a>
</div>
```

Step 4: Create Main JavaScript

Create `backend/static/backend/js/main.js`:

javascript

```javascript
// BueaDelights Main JavaScript

document.addEventListener('DOMContentLoaded', function() {
    // Initialize mobile menu
    initMobileMenu();

    // Initialize cart functionality
    initCart();

    // Initialize notifications
    initNotifications();

    // Initialize smooth scrolling
    initSmoothScrolling();

    // Initialize form animations
    initFormAnimations();
});

// Mobile Menu Toggle
function initMobileMenu() {
    const mobileMenuBtn = document.getElementById('mobile-menu-btn');
    const mobileMenu = document.getElementById('mobile-menu');

    if (mobileMenuBtn && mobileMenu) {
        mobileMenuBtn.addEventListener('click', function() {
            mobileMenu.classList.toggle('hidden');
        });
    }
}

// Cart Functionality
function initCart() {
    // Add to cart buttons
    const addToCartBtns = document.querySelectorAll('.add-to-cart');

    addToCartBtns.forEach(btn => {
        btn.addEventListener('click', function(e) {
            e.preventDefault();

            const productId = this.dataset.productId;
            const productName = this.dataset.productName;
            const productPrice = this.dataset.productPrice;
            const quantity = 1;
```

```
            addToCart(productId, productName, productPrice, quantity);
        });
    });

    // Update cart count on page load
    updateCartCount();
}

function addToCart(productId, productName, productPrice, quantity) {
    const csrfToken = document.querySelector('[name=csrfmiddlewaretoken]').value;

    fetch('/cart/add/', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'X-CSRFToken': csrfToken
        },
        body: JSON.stringify({
            product_id: productId,
            quantity: quantity
        })
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            showNotification('Product added to cart!', 'success');
            updateCartCount();
        } else {
            showNotification('Error adding product to cart', 'error');
        }
    })
    .catch(error => {
        console.error('Error:', error);
        showNotification('Error adding product to cart', 'error');
    });
}

function updateCartCount() {
    fetch('/cart/count/')
    .then(response => response.json())
    .then(data => {
        const cartCount = document.getElementById('cart-count');
        if (cartCount) {
            cartCount.textContent = data.count;
        }
    })
```

```
    .catch(error => {
        console.error('Error updating cart count:', error);
    });
}

// Notifications
function initNotifications() {
    // Auto-hide notifications after 5 seconds
    setTimeout(() => {
        const notifications = document.querySelectorAll('.notification');
        notifications.forEach(notification => {
            notification.style.opacity = '0';
            setTimeout(() => {
                notification.remove();
            }, 300);
        });
    }, 5000);
}

function showNotification(message, type = 'success') {
    const notificationsContainer = document.getElementById('notifications');

    const notification = document.createElement('div');
    notification.className = `notification notification-${type}`;
    notification.innerHTML = `
        <div class="flex items-center">
            <span>${message}</span>
            <button onclick="this.parentElement.parentElement.remove()" class="ml-4 text-white
                <svg class="w-4 h-4" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                    <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M6
                </svg>
            </button>
        </div>
    `;

    notificationsContainer.appendChild(notification);

    // Auto-remove after 5 seconds
    setTimeout(() => {
        if (notification.parentElement) {
            notification.style.opacity = '0';
            setTimeout(() => {
                notification.remove();
            }, 300);
        }
    }, 5000);
```

```
}

// Smooth Scrolling
function initSmoothScrolling() {
    const links = document.querySelectorAll('a[href^="#"]');

    links.forEach(link => {
        link.addEventListener('click', function(e) {
            e.preventDefault();

            const targetId = this.getAttribute('href');
            const targetElement = document.querySelector(targetId);

            if (targetElement) {
                targetElement.scrollIntoView({
                    behavior: 'smooth',
                    block: 'start'
                });
            }
        });
    });
}

// Form Animations
function initFormAnimations() {
    const inputs = document.querySelectorAll('.input-field');

    inputs.forEach(input => {
        input.addEventListener('focus', function() {
            this.parentElement.classList.add('focused');
        });

        input.addEventListener('blur', function() {
            if (!this.value) {
                this.parentElement.classList.remove('focused');
            }
        });
    });
}

// WhatsApp Integration
function sendWhatsAppMessage(message) {
    const phoneNumber = '+237699808260';
    const encodedMessage = encodeURIComponent(message);
    const whatsappUrl = `https://wa.me/${phoneNumber}?text=${encodedMessage}`;

    window.open(whatsappUrl, '_blank');
```

```
}

// Loading States
function showLoading(element) {
    const originalText = element.textContent;
    element.setAttribute('data-original-text', originalText);
    element.innerHTML = '<span class="loader"></span> Loading...';
    element.disabled = true;
}

function hideLoading(element) {
    const originalText = element.getAttribute('data-original-text');
    element.textContent = originalText;
    element.disabled = false;
    element.removeAttribute('data-original-text');
}

// Image Lazy Loading
function initLazyLoading() {
    if ('IntersectionObserver' in window) {
        const imageObserver = new IntersectionObserver((entries, observer) => {
            entries.forEach(entry => {
                if (entry.isIntersecting) {
                    const img = entry.target;
                    img.src = img.dataset.src;
                    img.classList.remove('lazy');
                    imageObserver.unobserve(img);
                }
            });
        });

        document.querySelectorAll('img[data-src]').forEach(img => {
            imageObserver.observe(img);
        });
    }
}

// Initialize Lazy Loading after DOM is loaded
document.addEventListener('DOMContentLoaded', initLazyLoading);

// Utility Functions
const utils = {
    formatCurrency: (amount) => {
        return new Intl.NumberFormat('fr-CM', {
            style: 'currency',
            currency: 'XAF',
```

```
            minimumFractionDigits: 0
        }).format(amount);
    },

    formatPhoneNumber: (phone) => {
        // Format Cameroon phone numbers
        return phone.replace(/(\d{3})(\d{2})(\d{2})(\d{2})(\d{2})/, '+$1 $2 $3 $4 $5');
    },

    debounce: (func, wait) => {
        let timeout;
        return function executedFunction(...args) {
            const later = () => {
                clearTimeout(timeout);
                func(...args);
            };
            clearTimeout(timeout);
            timeout = setTimeout(later, wait);
        };
    }
};

// Export for use in other scripts
window.BueaDelights = {
    addToCart,
    showNotification,
    sendWhatsAppMessage,
    showLoading,
    hideLoading,
    utils
};
```

7. Production Configuration

Step 1: Production Settings

Create `bueadelights/settings_prod.py`:

python

```python
from .settings import *
import dj_database_url

# Production settings
DEBUG = False
ALLOWED_HOSTS = ['your-app-name.onrender.com', 'www.bueadelights.com', 'bueadelights.com']

# Database for production (PostgreSQL on Render)
DATABASES = {
    'default': dj_database_url.config(
        default=config('DATABASE_URL', default='sqlite:///db.sqlite3')
    )
}

# Static files configuration for production
STATIC_ROOT = BASE_DIR / 'staticfiles'
STATICFILES_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'

# Security settings
SECURE_BROWSER_XSS_FILTER = True
SECURE_CONTENT_TYPE_NOSNIFF = True
SECURE_HSTS_INCLUDE_SUBDOMAINS = True
SECURE_HSTS_SECONDS = 86400
SECURE_SSL_REDIRECT = True
SESSION_COOKIE_SECURE = True
CSRF_COOKIE_SECURE = True

# Logging
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'file': {
            'level': 'INFO',
            'class': 'logging.FileHandler',
            'filename': BASE_DIR / 'debug.log',
        },
        'console': {
            'level': 'INFO',
            'class': 'logging.StreamHandler',
        },
    },
    'loggers': {
        'django': {
```

```
            'handlers': ['file', 'console'],
            'level': 'INFO',
            'propagate': True,
        },
    },
}
```

Step 2: Update Requirements for Production

Add to `requirements.txt`:

txt

```
dj-database-url==1.3.0
psycopg2-binary==2.9.7
gunicorn==21.2.0
whitenoise==6.5.0
```

8. Render.com Deployment Setup

Step 1: Create Build Script

Create `build.sh`:

bash

```bash
#!/usr/bin/env bash
# Exit on error
set -o errexit

# Install dependencies
pip install -r requirements.txt

# Build CSS
npm install
npm run build-css-prod

# Collect static files
python manage.py collectstatic --no-input

# Run migrations
python manage.py migrate

# Create superuser if needed
python manage.py shell -c "
from django.contrib.auth import get_user_model
User = get_user_model()
if not User.objects.filter(username='admin').exists():
    User.objects.create_superuser('admin', 'admin@bueadelights.com', 'your-secure-password')
"

# Create business settings
python manage.py shell -c "
from backend.models import BusinessSettings
if not BusinessSettings.objects.exists():
    BusinessSettings.objects.create()
"
```

Make it executable:

bash

```bash
chmod +x build.sh
```

Step 2: Create Render Configuration

Create `render.yaml`:

yaml

```yaml
services:
  - type: web
    name: bueadelights
    env: python
    buildCommand: "./build.sh"
    startCommand: "gunicorn bueadelights.wsgi:application"
    envVars:
      - key: PYTHON_VERSION
        value: 3.9.16
      - key: DJANGO_SETTINGS_MODULE
        value: bueadelights.settings_prod
      - key: SECRET_KEY
        generateValue: true
      - key: DEBUG
        value: False
      - key: ALLOWED_HOSTS
        value: your-app-name.onrender.com

databases:
  - name: bueadelights-db
    plan: free
```

Step 3: Update URL Configuration

Create `bueadelights/urls.py`:

python

```python
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('backend.urls')),
]

# Serve media files in development
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
    urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Create `backend/urls.py`:

python

```python
from django.urls import path
from . import views

app_name = 'backend'

urlpatterns = [
    # Homepage
    path('', views.HomeView.as_view(), name='home'),

    # Products
    path('menu/', views.ProductListView.as_view(), name='products'),
    path('menu/<slug:slug>/', views.ProductDetailView.as_view(), name='product_detail'),
    path('category/<slug:slug>/', views.CategoryProductsView.as_view(), name='category_products

    # Cart
    path('cart/', views.CartView.as_view(), name='cart'),
    path('cart/add/', views.add_to_cart, name='add_to_cart'),
    path('cart/remove/', views.remove_from_cart, name='remove_from_cart'),
    path('cart/count/', views.cart_count, name='cart_count'),

    # Orders
    path('checkout/', views.CheckoutView.as_view(), name='checkout'),
    path('order-confirmation/<str:order_id>/', views.OrderConfirmationView.as_view(), name='orc

    # Pages
    path('about/', views.AboutView.as_view(), name='about'),
    path('contact/', views.ContactView.as_view(), name='contact'),
    path('catering/', views.CateringView.as_view(), name='catering'),

    # AJAX endpoints
    path('api/search/', views.search_products, name='search_products'),
]
```

9. WhatsApp Integration Setup

Step 1: Create WhatsApp Helper

Create `backend/utils.py`:

python

```python
import urllib.parse
from django.conf import settings

class WhatsAppHelper:
    @staticmethod
    def create_order_message(order):
        """Create WhatsApp message for order."""
        message = f"🍽️ *New Order from BueaDelights*\n\n"
        message += f"📋 Order ID: {order.order_id}\n"
        message += f"👤 Customer: {order.customer_name}\n"
        message += f"📱 Phone: {order.customer_phone}\n"
        message += f"📍 Location: {order.customer_location}\n\n"

        message += "🛒 *Order Items:*\n"
        for item in order.items.all():
            message += f"• {item.quantity}x {item.product.name} - {item.total_price} FCFA\n"

        message += f"\n💰 *Total: {order.total_amount + order.delivery_fee} FCFA*\n"
        message += f"🚚 Delivery Fee: {order.delivery_fee} FCFA\n"
        message += f"💳 Payment: {order.get_payment_method_display()}\n"

        if order.special_instructions:
            message += f"\n📝 Special Instructions:\n{order.special_instructions}\n"

        message += f"\n⏰ Order placed: {order.created_at.strftime('%d/%m/%Y at %H:%M')}"

        return message

    @staticmethod
    def create_whatsapp_url(phone_number, message):
        """Create WhatsApp URL with message."""
        encoded_message = urllib.parse.quote(message)
        return f"https://wa.me/{phone_number}?text={encoded_message}"

    @staticmethod
    def get_business_whatsapp_url(message=""):
        """Get business WhatsApp URL."""
        phone = settings.WHATSAPP_NUMBER.replace('+', '').replace(' ', '')
        default_message = "Hello BueaDelights! I'm interested in your services."
        final_message = message if message else default_message
        return WhatsAppHelper.create_whatsapp_url(phone, final_message)
```

Step 2: Create Template Filter for WhatsApp

Add to `backend/templatetags/cart_tags.py`:

python

```python
from django.template import Library
from ..utils import WhatsAppHelper

register = Library()

@register.simple_tag
def whatsapp_url(message=""):
    """Generate WhatsApp URL."""
    return WhatsAppHelper.get_business_whatsapp_url(message)

@register.simple_tag
def whatsapp_order_url(order):
    """Generate WhatsApp URL for order."""
    message = WhatsAppHelper.create_order_message(order)
    return WhatsAppHelper.get_business_whatsapp_url(message)
```

10. Development Commands

Step 1: Run Development Server

bash

```bash
# Activate virtual environment
source venv/bin/activate  # On macOS/Linux
# or
venv\Scripts\activate     # On Windows

# Run migrations
python manage.py makemigrations
python manage.py migrate

# Create superuser
python manage.py createsuperuser

# Start development server with hot reload
python dev_server.py
```

Step 2: Useful Django Commands

bash

```bash
# Create new migration
python manage.py makemigrations backend

# Apply migrations
python manage.py migrate

# Collect static files
python manage.py collectstatic

# Create sample data
python manage.py shell -c "
from backend.models import Category, Product
cat = Category.objects.create(name='Traditional Dishes', description='Authentic Cameroonian cui
Product.objects.create(name='Ndolé', description='Traditional Cameroonian dish with groundnuts
"
```

Next Steps

1. **Setup Development Environment**: Follow steps 1-6 to get your development environment running
2. **Create Views and Templates**: Implement the views and templates for your application
3. **Add Sample Data**: Create categories and products through the admin interface
4. **Test Functionality**: Test all features including cart, orders, and WhatsApp integration
5. **Deploy to Render**: Follow the deployment steps to go live

Additional Resources

- Django Documentation
- Tailwind CSS Documentation
- Render.com Django Guide
- WhatsApp Business API

Remember to update your `.env` file with actual values before deploying to production!