























CareCove Project Structure

This document provides a comprehensive overview of the CareCove Django project structure and organization.

Root Directory Structure

carecove-django/	
 README.md	# Main project documentation
 requirements.txt	# Python dependencies
 .env.example	# Environment variables template
 .gitignore	# Git ignore rules
 LICENSE	# MIT License
 CHANGELOG.md	# Version history and updates
 CONTRIBUTING.md	# Contribution guidelines
 deployment_guide.md	# Deployment instructions
 PROJECT_STRUCTURE.md	# This file
 manage.py	# Django management script
 Procfile	# Heroku deployment configuration
 runtime.txt	# Python version specification
 gunicorn.conf.py	# Gunicorn server configuration
 Dockerfile	# Docker container configuration
 docker-compose.yml	# Docker Compose setup
 nginx.conf	# Nginx configuration
 app.json	# Heroku app configuration
 db.sqlite3	# SQLite database (development)
 venv/	# Python virtual environment
 logs/	# Application logs
 staticfiles/	# Collected static files (production)
 [Django Apps]	# Individual Django applications

Django Applications

1. shop - Core E-commerce

shop/	
 models.py	# Product, Category, Order models
 views.py	# Product listing, detail views
 admin.py	# Admin interface configuration
 urls.py	# URL routing
 forms.py	# Product and order forms
 management/commands/	# Custom management commands
 seed_data.py	# Sample data seeding
 migrations/	# Database migrations
 __pycache__/	# Python cache files

Key Features:

- Product catalog management
- Category organization
- Order processing
- Inventory tracking
- Search and filtering

2. cart - Shopping Cart

```

cart/
├── models.py           # Cart and CartItem models
├── views.py           # Cart operations (add, remove, update)
├── context_processors.py # Cart context for templates
├── forms.py           # Cart-related forms
├── migrations/        # Database migrations
└── __pycache__/       # Python cache files

```

Key Features:

- Session-based cart functionality
- AJAX cart updates
- Cart persistence
- Checkout process

3. accounts - User Management

```

accounts/
├── models.py          # User profile extensions
├── views.py           # Registration, login, profile views
├── forms.py           # User registration and profile forms
├── urls.py            # Authentication URLs
├── migrations/        # Database migrations
└── __pycache__/       # Python cache files

```

Key Features:

- User registration and authentication
- Profile management
- Order history
- Password reset functionality

4. chatbot - AI Customer Support

```

chatbot/
├── models.py          # ChatSession, ChatMessage, FAQ models
├── views.py           # Chat API endpoints
├── admin.py           # Chat management interface
├── urls.py            # API routing
├── management/commands/ # Chatbot data management
│   └── seed_chatbot_data.py # FAQ and responses seeding
├── migrations/        # Database migrations
└── __pycache__/       # Python cache files

```

Key Features:

- LLM-powered natural language processing
- WhatsApp integration
- FAQ automation
- Chat session tracking
- Quick response templates

5. 📧 newsletter - Email Marketing

```
newsletter/
├── models.py           # Subscriber model
├── views.py            # Subscription management
├── forms.py            # Newsletter signup forms
├── urls.py             # Newsletter URLs
├── migrations/         # Database migrations
└── __pycache__/        # Python cache files
```

Key Features:

- Email subscription management
- Newsletter campaigns
- Subscriber analytics

6. ★ testimonials - Customer Reviews

```
testimonials/
├── models.py           # Testimonial model
├── views.py            # Review submission and display
├── forms.py            # Testimonial forms
├── urls.py             # Review URLs
├── migrations/         # Database migrations
└── __pycache__/        # Python cache files
```

Key Features:

- Customer review collection
- Review moderation
- Rating system
- Display customization



carecove - Main Project Configuration

```
carecove/
├── settings.py         # Django settings and configuration
├── urls.py             # Main URL routing
├── wsgi.py             # WSGI application entry point
├── asgi.py             # ASGI application entry point
├── health.py           # Health check endpoints
└── __pycache__/        # Python cache files
```

Frontend Assets

Static Files

```
static/
├── css/
│   └── style.css           # Main stylesheet with Tanzanian theme
├── js/
│   └── main.js             # JavaScript functionality and chatbot
└── img/
    ├── beach-dhow.jpg      # Tanzanian dhow boat image
    ├── beach-golden.jpg    # Golden hour beach image
    ├── beach-palms.jpg     # Palm trees beach image
    └── beach-panoramic.jpg  # Panoramic beach view
```

Templates

```
templates/
├── base.html               # Base template with common structure
├── shop/
│   ├── home.html          # Homepage with product showcase
│   ├── product_list.html  # Product listing page
│   ├── product_detail.html # Individual product pages
│   ├── about.html         # About page
│   └── contact.html       # Contact page
├── cart/
│   ├── cart_detail.html   # Shopping cart page
│   ├── checkout.html     # Checkout process
│   └── payment_success.html # Order confirmation
├── accounts/
│   ├── login.html         # User login page
│   └── register.html      # User registration page
├── testimonials/
│   ├── testimonials_list.html # Customer reviews display
│   └── submit_testimonial.html # Review submission form
└── newsletter/           # Newsletter templates
```

Media Files

```
media/
├── products/              # Product images
├── categories/            # Category images
└── testimonials/         # User-uploaded review images
```

Database Schema

Core Models

Product Management:

- `Category` : Product categories with multilingual support
- `Product` : Main product model with pricing, descriptions, images
- `ProductImage` : Additional product images
- `Order` : Customer orders
- `OrderItem` : Individual items within orders

User Management:

- `User` : Django's built-in user model
- `UserProfile` : Extended user information
- `Cart` : Shopping cart sessions
- `CartItem` : Items in shopping carts

Customer Support:

- `ChatSession` : Chat conversation sessions
- `ChatMessage` : Individual chat messages
- `ChatbotFAQ` : Frequently asked questions
- `QuickResponse` : Pre-defined quick responses

Marketing:

- `Testimonial` : Customer reviews and ratings
- `Newsletter` : Email subscribers



Configuration Files

Environment Configuration

- `.env.example` : Template for environment variables
- `settings.py` : Django configuration with environment variable support

Deployment Configuration

- `Procfile` : Heroku deployment commands
- `runtime.txt` : Python version specification
- `requirements.txt` : Python package dependencies
- `gunicorn.conf.py` : Production server configuration

Containerization

- `Dockerfile` : Docker container configuration
- `docker-compose.yml` : Multi-container development setup
- `nginx.conf` : Reverse proxy configuration

Cloud Platform Support

- `app.json` : Heroku Button deployment configuration



Key Features Implementation

1. Multi-Language Support

- Translation files in `locale/` directory
- Language switching in templates
- Localized content in models

2. AI Chatbot Integration

- LLM API integration in `chatbot/views.py`
- Real-time chat interface in `static/js/main.js`
- WhatsApp handoff functionality

3. Payment Processing

- Pesapal gateway integration
- Secure payment handling
- Order confirmation workflow

4. Responsive Design

- Mobile-first CSS approach
- Tanzanian beach theme
- Golden yellow branding consistency

5. Performance Optimization

- Static file compression with WhiteNoise
- Database query optimization
- Caching configuration ready

6. Security Features

- CSRF protection
- XSS prevention
- Secure headers configuration
- Environment variable management



Monitoring & Health Checks

Health Check Endpoints

- `/health/` : Comprehensive health status
- `/health/simple/` : Basic health check for load balancers

Logging Configuration

- Structured logging setup in `settings.py`
- Log rotation and management
- Error tracking and monitoring ready



Security Considerations

Production Security

- SSL/HTTPS configuration
- Secure cookie settings
- Content Security Policy headers
- Rate limiting configuration

Data Protection

- Environment variable isolation
- Secure database connections
- File upload restrictions
- API authentication



API Endpoints

Chatbot API

- `POST /chatbot/chat/` - Chat messaging
- `GET /chatbot/quick-responses/` - Quick responses
- `GET /chatbot/history/<session_id>/` - Chat history
- `POST /chatbot/whatsapp-transfer/` - WhatsApp handoff

Shop API

- `GET /shop/products/` - Product listing
- `GET /shop/categories/` - Category listing
- `POST /cart/add/` - Add to cart
- `GET /cart/` - Cart contents



Development Workflow

Local Development

1. Virtual environment setup
2. Environment configuration
3. Database migrations
4. Sample data loading
5. Development server startup

Production Deployment

1. Environment variable configuration
2. Database setup and migration
3. Static file collection
4. SSL certificate configuration
5. Server deployment and monitoring



Documentation Files

- `README.md` : Comprehensive project overview and setup
- `deployment_guide.md` : Detailed deployment instructions
- `CONTRIBUTING.md` : Contribution guidelines and standards
- `CHANGELOG.md` : Version history and feature updates
- `LICENSE` : MIT license terms
- `PROJECT_STRUCTURE.md` : This architectural overview

This structure supports a complete e-commerce platform with modern features, scalable architecture, and production-ready deployment options. The modular design allows for easy maintenance and feature expansion while maintaining clean separation of concerns.