dolesecommerce Django Project Documentation

Overview

This is a modern e-commerce web application built with Django. It supports product management, user authentication, payments (M-Pesa, Pi Coin), reviews, shipping, and customer support features. The UI is styled with Bootstrap 5 and includes a dark navigation bar, logo, and modern forms.

Features

- Product catalog with categories, brands, and multiple images per product
- User registration, login, profile editing (with avatar, phone, address)
- Shopping cart, Buy Now, and order management
- Payment integration: M-Pesa (Daraja API), Pi Coin
- Review and rating system
- Shipping management
- Admin dashboard for managing products, orders, users
- Customer service page with live chat, FAQ, feedback form
- Email notifications for order and registration events
- Responsive, modern UI with custom colors and logo

Project Structure

```
db.sqlite3
manage.py
core/           # Home, navigation, support views
products/       # Product models, views, templates
orders/         # Order models, views, templates
payments/       # Payment models, views, templates
reviews/        # Review models, views
shipping/       # Shipping models, views
users/          # User models, views, templates
static/         # Static files (images, CSS, JS)
media/          # Uploaded product images
templates/      # Shared base templates
```

Models

- **Category**: name, description
- **Brand**: name, description
- **Product**: name, description, price, category, brand, stock, is_active, image, timestamps
- **ProductImage**: product (FK), image, uploaded_at
- **UserProfile**: user (FK), phone, address, avatar
- **Order**: user (FK), products, status, timestamps
- **Payment**: order (FK), method, status, transaction details
- **Review**: product (FK), user (FK), rating, comment
- **Shipping**: order (FK), address, status

Key Templates

- `base.html`: Main layout, navigation bar, logo, favicon
- `products/product_list.html`: Product catalog
- `products/product_detail.html`: Product details, Buy Now, Add to Cart
- `users/register.html`, `login.html`, `profile.html`, `profile_edit.html`: User management
- `core/customer_service.html`: Support, live chat, FAQ, feedback

Static & Media Files

- Place images in `static/images/` (e.g., `logo.png`, `default-avatar.png`)
- Product images are uploaded to `media/product_images/`
- Configure `STATIC_URL`, `MEDIA_URL`, `STATICFILES_DIRS`, and `MEDIA_ROOT` in `settings.py`

Payment Integration

- **M-Pesa**: Uses Daraja API for mobile payments
- **Pi Coin**: Cryptocurrency payment support

Customer Service

- Dedicated page with sections for FAQ, feedback, live chat (Tawk.to), and contact info

Running the Project

Pi Coin Payments Integration

To enable Pi Coin payments in your app, use the official Pi Network JS SDK. This allows users to pay with Pi directly from the Pi Wallet.

Steps to Integrate Pi Coin Payments

1. **Add the Pi SDK to your template:**
   Add the following script tags to your main template (e.g., `base.html`):

```
<script src="https://sdk.minepi.com/pi-sdk.js"></script>
<script>Pi.init({ version: "2.0" })</script>
```

2. **Authenticate the user:**
   Before requesting payments, authenticate the user:

```
const scopes = ['payments'];
Pi.authenticate(scopes, function onIncompletePaymentFound(payment) {
  // Handle incomplete payments
}).then(function(auth) {
  console.log("User authenticated for Pi payments.");
}).catch(function(error) {
  console.error(error);
});
```

3. **Request a payment:**
   Use the SDK to prompt the user for payment:

```
Pi.createPayment({
  amount: 3.14, // Amount of Pi
  memo: "Order #1234", // Description for the user
  metadata: { orderId: 1234 }
}, {
  onReadyForServerApproval: function(paymentId) { /* ... */ },
  onReadyForServerCompletion: function(paymentId, txid) { /* ... */ },
  onCancel: function(paymentId) { /* ... */ },
  onError: function(error, payment) { /* ... */ }
});
```

4. **Handle server-side approval/completion:**
   For advanced flows (App-to-User payments, server confirmation), refer to the Pi Platform Docs (https://github.com/pi-apps/pi-platform-docs) and their `payments_advanced.md` and `platform_API.md` files.

Resources

- Pi Platform Docs (https://github.com/pi-apps/pi-platform-docs)
- SDK Reference (https://github.com/pi-apps/pi-platform-docs/blob/master/SDK_reference.md)
- Payments Guide (https://github.com/pi-apps/pi-platform-docs/blob/master/payments.md)

This integration allows your Django app to accept Pi Coin payments securely and easily.

Running the Project

1. Install dependencies: `pip install -r requirements.txt`
2. Apply migrations: `python manage.py migrate`
3. Create superuser: `python manage.py createsuperuser`
4. Run server: `python manage.py runserver`
5. Access site at `http://localhost:8000/`

Admin Access

- Login at `/admin/` with superuser credentials

Notes

- Ensure all static and media files are correctly named and placed
- Use `{% load static %}` and `{% static 'images/logo.png' %}` in templates
- For support, check `core/customer_service.html` and related views

Example Admin/User Credentials

- Admin: Use the credentials created with `python manage.py createsuperuser` to log in at `/admin/`.
- Users: Register via `/users/register/` or create users in the admin panel.

Sample Data Creation

To quickly add sample products, categories, and brands:

1. Log in to the Django admin at `/admin/`.
2. Add categories, brands, and products using the admin interface.
3. Upload product images via the product or product image forms.
4. Create test orders and payments to see the workflow.

Troubleshooting Common Errors

- **Static/Media 404s**: Ensure files are named correctly and placed in the right folders. Run `python manage.py collectstatic` if needed.
- **Template Errors**: Always use `{% load static %}` at the top of templates using static files.
- **NoReverseMatch**: Check that your URLs and template tags match the namespaced URL patterns.
- **Image Not Displaying**: Confirm the image path and filename match what is referenced in the template.
- **Database Issues**: Run migrations with `python manage.py migrate` and check for missing fields.

More Details on Models and Views

- **Category/Brand/Product**: Organize products for easy browsing and filtering.
- **ProductImage**: Allows multiple images per product for better presentation.
- **UserProfile**: Stores extra user info (phone, address, avatar) for personalized experience.
- **Order/Payment/Shipping**: Handles the full purchase and delivery workflow.
- **Review**: Lets users rate and review products.
- **Customer Service**: Provides support, live chat, FAQ, and feedback forms.

Learning Resources

- [Django Documentation (https://docs.djangoproject.com/)](https://docs.djangoproject.com/)
- [Bootstrap Documentation (https://getbootstrap.com/)](https://getbootstrap.com/)
- [Django Admin Guide (https://docs.djangoproject.com/en/5.2/ref/contrib/admin/)](https://docs.djangoproject.com/en/5.2/ref/contrib/admin/)

For further customization or feature requests, contact the project maintainer.