Technical Design and Architecture Document (TDAD)

Project Name: Excellent Fashion Wares E-commerce Platform
Version: 1.0
System Owner: [Your Name/Team]

1. Technology Stack

Component	Technology	Rationale
Backend Framework	**Django 5.2.1** (Python)	Full-featured, secure, and robust web framework with built-in ORM and Admin interface.
Database	**MySQL (**3306**)**	Configured via DATABASES for high-volume, transactional data storage (preferred over SQLite for production).
Templating	Django Templates, **Crispy Forms**, **Widget Tweaks**	Use of crispy-bootstrap5 for rapid, clean, and responsive form rendering.
Payment Gateway	**Paystack**	Primary payment integration using configured public and secret keys.
Logging	django_db_logger	Centralized logging of application, request, and security events directly to the database for ease of access and audit.
Development/Hosting	Localhost, ngrok, potential Docker/Cloud deployment.	ALLOWED_HOSTS and CSRF_TRUSTED_ORIGINS are set up to support local dev and remote tunneling (e.g., ngrok).

2. High-Level Architecture (Monolithic)

The system utilizes a **monolithic architecture** based on the Django framework. All functionality (product catalog, cart, checkout, user profile) resides within a single codebase.

Data Flow for Conversion (Purchase)

1. **User Interaction:** User clicks "Pay" on the final checkout page.
2. **View Processing:** Request hits the checkout app's view.
3. **Payment Gateway:** The view initiates a transaction with **Paystack** using the configured API keys.
4. **Order Creation:** Upon successful payment validation/callback, an Order object is created and linked to the user and Address model.
5. **Audit/Logging:** Events are recorded via django_db_logger and the analytics app tables.

3. Backend Module Breakdown

The application is decomposed into several modular Django apps, ensuring separation of concerns:

Django App	Key Data Models & Responsibility	Dependencies
products	Product, Category. Manages inventory and catalog presentation.	None
cart	Cart, CartItem. Handles session-based storage of items before checkout.	products
checkout	Order, OrderItem, Address. Handles payment processing and finalizes transactions.	cart
wishlist	WishlistItem. Tracks items saved by the user.	products, Django Auth
core	Custom User Model (implied by SignUpView), general utilities, contact forms.	Django Auth
analytics	PageView, ConversionEvent. Captures metrics for business intelligence.	None
ecomstore (Project)	settings.py, urls.py, context_processors.py. Central configuration and URL routing.	All apps

4. Security Considerations

- **Authentication:** Uses robust Django built-in authentication views (auth_views.LoginView, etc.).
- **Password Management:** Full password reset and change functionality is implemented in urls.py.
- **Database Credentials:** Currently hardcoded in settings.py for MySQL; **MUST** be moved to environment variables (e.g., .env file using python-decouple) before production

deployment.

- **CSRF/XSS:** Default Django middleware protection (CsrfViewMiddleware) is active.