

CAMPUS MARKETPLACE

System Summary

A secure peer-to-peer e-commerce platform built for NWU students

Version	1.2	Module	CMPG 323
Issued	October 2025	Platform	Next.js / Firebase / PayFast
Authors	Group 3 — NWU IT Development	Status	Production-ready

1. System Overview

What Campus Marketplace is and who it serves

Campus Marketplace is a full-stack, production-ready e-commerce web application designed exclusively for North-West University (NWU) students. It enables students to list, discover, and purchase second-hand textbooks, electronics, clothing, notes, stationery, and other items — all within a trusted campus community.

Secure Authentication	Firebase Auth with email/password & Google OAuth, enforced email verification, session timeout after 15 min inactivity.	PayFast Payments	South-African payment gateway with MD5 signature generation/validation, sandbox & production modes.
Item Listings	Sellers create listings (draft or live), upload up to 3 images, set price, condition, category & collection address.	Admin Dashboard	Analytics charts (Recharts), item approval workflow, user suspension / promotion to admin.
Search & Filter	Full-text search across titles; filter by price range, condition, category; sort by newest/oldest/price.	Image Management	Firebase Storage with drag-and-drop multi-image uploader, reordering, 5 MB per-file limit.
Shopping Cart	Real-time cart management, VAT (15%) calculated automatically, checkout via PayFast.	Route Protection	Next.js middleware validates JWT on every request; admin routes require custom claims.

2. Technology Stack

Layer	Technology & Notes
Frontend Framework	Next.js 15.4.6 with React 19.1.0 & TypeScript 5.9.2 — App Router, Server Components, Turbopack
Styling	Tailwind CSS v4 + Radix UI headless primitives + Lucide-React icons
Backend / API	Next.js API Routes + Firebase Admin SDK 13.5.0 (server-side Firestore & Auth)
Database	Cloud Firestore (NoSQL, document-based) hosted on Google Firebase
Authentication	Firebase Auth — email/password, Google OAuth, JWT in HTTP-only cookies via jose 6.1.0
Payments	PayFast (South Africa) — MD5 signature with crypto-js 4.2.0, query-string via qs 6.14.0
File Storage	Firebase Storage with security rules limiting uploads to authenticated users, max 5 MB

Layer	Technology & Notes
Charts	Recharts 3.2.1 — Pie & Bar charts for admin analytics dashboard
Form Validation	Zod 4.1.8 schemas + React Hook Form 7.62.0 + @hookform/resolvers
Notifications	Sonner 2.0.7 + react-hot-toast 2.6.0 for in-app toasts
Drag & Drop	@hello-pangea/dnd 18.0.1 — image re-ordering in uploader
Deployment	Firebase Hosting (frameworksBackend, region: europe-west1) + Vercel

3. Application Architecture

High-level structure of the Next.js App Router project

Path	Responsibility
/src/app/	All Next.js pages (App Router) — home, login, register, profile, cart, admin, item details
/src/app/api/	Server-side API route handlers: items, cart, payfast, admin, auth, users, payments
/src/components/	Reusable React components: ItemCard, AddToCartButton, MultiImageUploader, AuthButtons, ...
/src/context/	React Context providers — auth.tsx (user state + admin status) and actions.ts (server actions)
/src/firebase/	Firebase SDK config — client.ts (browser) and server.ts (Admin SDK, server-only)
/src/hooks/	Custom React hooks — useSessionTimeout.ts (15-min inactivity logout)
/src/lib/	Utility functions — imageUrlFormatter, cartCleanup, itemManagement, sessionUtils
/src/types/	TypeScript interfaces — Item, CartItem, UserDocument, API response shapes
/src/validation/	Zod schemas — itemSchema.ts, registerUser.ts for runtime validation
/src/middleware.ts	Edge middleware — JWT validation, public-route allow-list, admin route guard
/storage.rules	Firebase Storage security rules — 5 MB image limit, auth-required reads/writes
/firebase.json	Firebase Hosting + Firestore + Storage config, deployment scripts

4. Database Schema (Cloud Firestore)

items collection

Field	Type	Description
id	string	Firestore document ID (auto-generated)
title	string	Listing title — searchable
description	string	Full description (Markdown supported)
price	number	Price in cents to avoid float precision errors
condition	enum	new excellent used fair poor
category	enum	books electronics clothing notes stationery other
status	enum	draft pending for-sale sold withdrawn collected
sellerId	string	Firebase Auth UID of seller
buyerId?	string	UID of buyer — set when sold
images?	string[]	Array of Firebase Storage URLs (max 3)
collectionAddress	string	Campus pickup location

Field	Type	Description
postedAt / updatedAt	Timestamp	ISO date strings after serialisation

cartItems collection

Field	Type	Description
cartItemId	string	Document ID
cartId	string	Equals the buyer's Firebase UID
itemId	string	Reference to items document
quantity	number	Always 1 (unique items)

User data — Firebase Auth + Firestore

Field	Type	Description
uid	string	Firebase Auth UID — primary key
email	string	Login email
emailVerified	bool	Must be true before login is permitted
displayName?	string	Full name
admin	claim	Firebase custom claim — set server-side via Admin SDK

5. Authentication & Security

Auth Methods	Email/Password (verified) and Google OAuth via Firebase Auth
Token Storage	Firebase ID token stored in HTTP-only cookie (XSS-proof); refresh token also cookie-stored
Email Verification	Login blocked until email verified; unverified users redirected to /auth/verifyemail
Session Timeout	Custom hook monitors mouse/keyboard/touch/scroll; auto-logout after 15 min inactivity, cross-tab via localStorage events
Route Middleware	Next.js edge middleware decodes JWT on every request; expired tokens redirect to /login; /admin requires admin claim
Admin Claims	Server-side only — ADMIN_EMAILS env var; Admin SDK sets custom claim; validated via /api/admin/status endpoint
PayFast Signatures	Outbound: MD5(queryString + passphrase); Inbound ITN: server re-generates and compares signature
Storage Rules	Firebase Storage rules enforce auth, image MIME type, and 5 MB size cap per upload
API Error Codes	401 Unauthorized 403 Forbidden 400 Bad Request 404 Not Found 500 Server Error

6. Key API Routes

Method	Endpoint	Auth	Description
POST	/api/items/list	Public/Auth	Fetch items with filters (price, condition, category, sort, pagination)
POST	/api/items/create	Auth	Create new item listing with Zod validation
GET	/api/cart	Auth	Get cart items; auto-removes items no longer for-sale
POST	/api/items/actions/add-to-cart	Auth	Add an item to the authenticated user's cart

POST	/api/payfast/checkout	Auth	Build PayFast payment URL with MD5 signature
POST	/api/payfast/notify	PayFast	Receive ITN, validate signature, update item status to sold
GET	/api/admin/status	Admin	Return isAdmin claim for current token
POST	/api/users/suspend	Admin	Suspend / unsuspend a user account
POST	/api/auth/register	Public	Create Firebase Auth user + Firestore document
GET	/api/refresh-token	Auth	Refresh Firebase ID token; rotate cookie

7. Core User Flows

Purchasing an Item	Selling an Item
- Browse / search items on home page - Click 'View Item' for details - Click 'Add to Cart' (login required) - Review cart — subtotal + 15% VAT displayed - Agree to T&Cs; and click 'Pay Now' - Complete payment on PayFast gateway - Contact seller to arrange collection - Collect item, take selfie as proof - Mark item as 'Collected' in My Purchases - Payment released to seller	- Click 'Sell' in navigation bar - Enter title, description (Markdown), price - Set collection address, condition, category - Upload 1–3 item photos (drag to reorder) - Click 'Sell Item' to submit for review - Admin reviews and approves the listing - Item goes live on Campus Marketplace - Buyer purchases — item marked 'Sold' - Buyer contacts you to collect - Collection selfie → buyer marks 'Collected' → payment released

8. PayFast Payment Integration

Gateway	PayFast (South African Rand — ZAR); sandbox & production modes configurable via env vars
Checkout Flow	Server builds payment data object → filters empty values → URL-encodes → appends passphrase → MD5 hash → redirect URL returned to browser
ITN (Notify)	PayFast POSTs to /api/payfast/notify; server strips signature from raw body, re-hashes, compares — 400 returned on mismatch
VAT	15% South African VAT calculated client-side in cart; totalWithVat sent to PayFast as amount
Security	PAYFAST_MERCHANT_ID, PAYFAST_MERCHANT_KEY, PAYFAST_PASSPHRASE stored server-side only (no NEXT_PUBLIC_ prefix)
Return URLs	return_url → /success cancel_url → /cart notify_url → /api/payfast/notify
Sandbox	SANDBOX=true constant in checkout route for development; switch to false for production

9. Admin Dashboard & Capabilities

Capability	Details
Dashboard Metrics	Total items, items for sale, total users, revenue (turnover ÷ 100 formatted as ZAR)
Item Status Chart	Pie chart of items grouped by status: for-sale, draft, sold, collected, pending, withdrawn
Price Distribution	Bar chart with price buckets: R0–R50, R50–R200, R200–R500, R500+
User Management	Search users; suspend / remove; promote to admin; view detailed profile incl. email verification status

Capability	Details
Item Approval	Review pending listings; change condition (based on uploaded photos); approve → status changes to for-sale
Item Edit	Admin can override item condition and status for any listing
Access Control	Admin routes (/admin/*) protected at middleware level via admin JWT custom claim
Analytics Data	Fetches live from /api/items/list (pageSize 1000) and /api/admin/status on dashboard load

10. Environment Configuration

Variable	Scope	Purpose
NEXT_PUBLIC_FIREBASE_API_KEY	Client	Firebase project API key (browser-safe)
NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN	Client	OAuth redirect domain
NEXT_PUBLIC_FIREBASE_PROJECT_ID	Client	Firestore project identifier
NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET	Client	Firebase Storage bucket URL
NEXT_PUBLIC_FIREBASE_APP_ID	Client	Firebase app identifier
NEXT_PUBLIC_BASE_URL	Client	Base URL for PayFast redirect URLs
FIREBASE_PROJECT_ID	Server	Admin SDK project ID
FIREBASE_CLIENT_EMAIL	Server	Service account email
FIREBASE_PRIVATE_KEY	Server	Service account private key (PEM)
PAYFAST_MERCHANT_ID	Server	PayFast merchant ID
PAYFAST_MERCHANT_KEY	Server	PayFast merchant key
PAYFAST_PASSPHRASE	Server	Passphrase for MD5 signature generation
ADMIN_EMAILS	Server	Comma-separated admin email addresses

11. Deployment

Hosting	Firebase Hosting with Next.js frameworksBackend; backend region: europe-west1
Live URL	https://campus-marketplace-2yna.vercel.app (also deployed on Vercel)
Build Command	npm run build (Next.js production build with TypeScript & ESLint checks)
Dev Server	npm run dev — Next.js with Turbopack for fast HMR
Deploy All	firebase deploy — hosting + Firestore rules + Storage rules
Deploy Rules	firebase deploy --only firestore:rules,storage:rules
Emulators	firebase emulators:start --only firestore,storage — local dev without cloud
CI/CD	TypeScript and ESLint errors fail the build (ignoreBuildErrors: false)

12. Quick Developer Setup

1	Clone repository	<code>git clone https://github.com/Pieter246/campus-marketplace.git && cd campus-marketplace</code>
---	------------------	---

2	Install dependencies	<code>npm install</code>
3	Configure environment	<code>cp .env.example .env.local</code> – then fill in Firebase, PayFast, and admin credentials
4	Start dev server	<code>npm run dev</code> – available at <code>http://localhost:3000</code>
5	Run Firebase emulators	<code>npm run firebase:emulator</code> – local Firestore & Storage
6	Deploy to production	<code>npm run build && firebase deploy</code>

Campus Marketplace | CMPG 323 | Group 3 | NWU IT Development | October 2025