DUNWARE SOLUTIONS

Campus
Product Whitepaper

On-premise school management for a single school, and — new — Campus Central, a Ministry/district head-office console for overseeing many schools at once.

PostgreSQL, on your own server | Field-level AES-256-GCM encryption | MFA mandatory for staff | Flat one-time licence, no per-student fee | No AI, no third-party data sharing | Append-only audit log

Describes: Campus 0.9.0 and Campus Central (first release), both as built 11 October 2026.
Prepared: Sunday, October 11, 2026 — this is the first whitepaper written for this product family.

Contents

1. Executive summary 3
2. What Campus is 3
3. What Campus Central is 3
4. Architecture 4
5. The sync model — how a school reaches Central 4
6. Security & privacy 5
7. Module catalog — Campus (single school) 5
8. Module catalog — Campus Central (Ministry/district) 6
9. Delivery: installers, backups & remote access 6
10. Pricing & licensing 7
11. Testing and quality 7
12. Known limits, stated plainly 7
13. Technical reference 8

1. Executive summary

Campus is a single-school operations platform — registration, scheduling, attendance, grades and report cards, billing with optional online payments, communication, booking, IEPs and a parent/student portal — that runs entirely on a school's own server, with nothing sent to a third-party cloud by default. **Campus Central** is a second, separate product built on top of it: a Ministry or district head-office console that any number of individual Campus installs can optionally link to, giving a head office cross-school search, aggregate dashboards (including a district-level illness/health metric) and on-demand case-file detail, without ever holding a school's full, unfiltered database.

The two products are deliberately kept apart rather than merged into one "multi-tenant" system: a school's own Campus install is exactly as self-contained and private as it has always been, whether or not any Ministry ever links to it. Central is opt-in, read-mostly, and receives only a curated, field-limited export — never a school's raw database.

> **This is the first whitepaper written for this product family.** Campus itself has shipped an Overview & User Guide and a Technical & Troubleshooting Guide since September 2026; Campus Central, built across early-to-mid October 2026, had no dedicated product document at all until this one. Every figure in this paper was checked against the actual repository and, where it is a test count, by actually running the test suite — not carried forward from an earlier draft.

2. What Campus is

A single authenticated web console plus a restricted parent/student portal, for a daycare through secondary school to run day-to-day operations. One role per login (superadmin, admin, front desk, teacher/tutor, parent/guardian, student), each seeing only the sections and records their role permits. Built on Django 5.2 (REST API) and a Next.js static-export console, both frozen into one Windows installer with a bundled PostgreSQL database and a bundled Caddy reverse proxy for local HTTPS — no separate Python or Node runtime on the target machine, no Docker in production.

3. What Campus Central is

A Ministry/district head-office product, sold and installed separately from Campus itself (`Campus-Central-Setup.exe` , its own PyInstaller freeze, its own Next.js console on its own port, its own PostgreSQL database on its own port). A head office registers each participating school, which then links to Central from its own install (a companion window, not a web form) and begins sending it a curated snapshot on a schedule plus on demand. Central's own console gives Ministry/ district staff:

- **A schools registry** and per-school sync status.
- **Cross-school search** over students, guardians and staff by name — matches a full "First Last" query, not just one half of it, and shows which school each result belongs to.
- **On-demand case-file detail** for one student or guardian at a time — enrolment, attendance, guardians, consents, report-card status — reachable instantly when genuinely needed (for example, assisting a police inquiry about a specific family without the Ministry needing to contact the school directly), never held or broadcast in bulk.
- **Aggregate dashboards** — attendance, enrolment, billing (summary totals only, never per-family), and a district-level **illness/health rate** (normalized per 100 enrolled students, so a raw count never unfairly favours a smaller district) — by school or rolled up by district, with CSV export.
- **Ministry account management** — more than one real administrator account can now be created (role: Ministry Admin, Ministry Analyst or District Officer), not only the one break-glass account the very first release shipped with.

Three roles, deliberately asymmetric: **Ministry Admin** and **District Officer** can search and open case files (District Officer scoped to their own assigned schools); **Ministry Analyst** is refused individual search and case-file access outright, at the server, not just hidden in the interface — by design, an analyst role only ever sees aggregates.

4. Architecture

Layer	Campus (per school)	Campus Central (per Ministry/district)
Backend	Django 5.2 + DRF, `waitress` WSGI, PyInstaller-frozen (`campus_app.py`)	A genuinely **separate** Django project (`backend/central/` , `central_app.py`) — reuses only the shared `apps.accounts` / `apps.audit` / `apps.core` code, never a school's own operational apps
Frontend	Next.js (`output: 'export'` , static), `frontend/`	A separate Next.js project, `frontend_central/` (own port 3100 in dev, own build) — intentionally indigo-accented so it never looks like the same screen as Campus
Database	Bundled PostgreSQL 16, one per school install	Its own bundled PostgreSQL, on port **5433** specifically so a Campus+Central box never collides on the default port
Front door / TLS	Bundled Caddy, local HTTPS	Its own bundled Caddy, default port **8443** (moved off 443/80, which Campus's own Caddy already owns on a shared box) with `admin off` and `auto_https disable_redirects` so neither Caddy instance can collide with the other even on one machine
Background jobs	`django-q2` , now registered as a 4th Windows service ("Campus Scheduler") — previously configured but never actually run (§12)	None — Central has no scheduled jobs of its own yet; sync is driven from the school side (§5)
Packaging	`Campus-Setup.exe` , Inno Setup	`Campus-Central-Setup.exe` , Inno Setup, same pattern, its own `.env.central` secrets file

Central can never share a database, a process, or a port with any school's own install — this is structural, not configuration. A school's `SiteConfiguration.deployment_mode` can be `SINGLE` or `SCHOOL` ; it is refused outright, at the model's own `save()` and at every place that sets it, from ever being set to `HEADQUARTERS` — that mode only exists as Central's own separate installer.

5. The sync model — how a school reaches Central

A school links to Central from its own server (a companion window, **Campus — Central Link Setup**, not the browser console) with a URL and a bearer token Central issued when the school was registered. From then on:

- **What's sent** is a curated, allowlist export built directly from the source models — never a wider "export everything" serializer trimmed down, to avoid a future field silently leaking through. Included: students, guardians, staff, attendance/incident/billing *summaries* (per-term totals, never a per-family invoice or application), consent status, report-card status. **Excluded by default**: health/medical data, custody notes, government IDs, narrative text (observation notes, incident descriptions), per-family billing or application detail, and (deliberately deferred, not yet built at all) who is authorized to pick a child up.
- **Full vs incremental.** A sync is incremental by default (only rows changed since the last one), forced to a full reconcile when a school has never fully synced yet or it's been 7 or more days since the last full one — row-level tables only ever sweep (detect) deletions on a full sync, since an incremental bundle's absence of a row proves nothing; the four aggregate tables always send in full either way.
- **Transport.** Server-to-server bearer-token auth (no session/CSRF — this is not a browser call); the bundle's own SHA-256 content hash is recomputed on arrival to catch tampering or corruption in transit; a schema-version check refuses a bundle Central doesn't yet know how to ingest.
- **On a timer, honestly: not yet.** The sync agent and the "on-demand sync" button both work today (`manage central_sync` , or the companion window's **Sync now** button). What doesn't exist yet is an actual scheduled job that calls it automatically on a recurring basis — the worker process to run one now exists (§9's Campus Scheduler fix), but no `Schedule` row has been created for it. Stated here rather than implied, since "nightly sync" is the eventual intent but not the current reality.

6. Security & privacy

Campus's own security posture (built in from Phase 1, not a later pass): field-level AES-256-GCM encryption at rest for medical/allergy/medication data, custody and legal notes, government IDs, IEP content and uploaded documents; mandatory TOTP for every staff role; an append-only audit log of every write *and every sensitive read*; deny-by-default, object-level permissions (a parent sees only their own linked children; a teacher only their own groups); no card data ever reaches Campus (hosted-checkout payments only); no SMS path anywhere, email only.

Campus Central inherits the same account/audit/MFA foundation (the shared `apps.accounts` / `apps.audit` code), plus its own additions specific to holding cross-school data:

- **Deny-by-default school scoping.** A District Officer's queries are restricted to their assigned schools at the query layer itself (`schools_visible_to()`), not filtered after the fact.
- **Case-file access is on-demand only, never bulk-held** — the explicit governing principle: Central should not constantly broadcast or hold full student detail by default, but when it genuinely needs detail (the police-inquiry scenario), that detail must be reachable instantly, not blocked behind contacting the school first.
- **A continuous disk-encryption check** — BitLocker was previously verified only once, at install time; it's now re-checked on every `manage check` run, fails safe on any error, frozen builds only.
- **Its own encrypted, drilled backup/restore pipeline** — mirrors Campus's own GPG-encrypted `pg_dump` approach exactly; Central had no backup tooling at all before this was built.

> **Open question, stated honestly rather than built wrong.** Central's stored student/guardian PII (name, email, student number) is *not* field-encrypted the way Campus's own health/custody/government-ID data is. This was a deliberate decision, not an oversight: Central's cross-school search does direct SQL `icontains` matching on exactly those fields, and field-encrypting them would silently break that search outright — the same "find a student by name for a police inquiry" capability this paper describes as a core Central feature. A real fix needs searchable/blind-indexed encryption, which is its own, separately-scoped cryptographic engineering effort. Flagged here as an open item for a future release, not quietly shipped as "encrypted" when it isn't for these particular fields.

7. Module catalog — Campus (single school)

Dashboard — role-scoped snapshot: attendance, academics, wellbeing (incidents), enrolment, billing, consent/booking/messages, and (administrators) staffing & system health.

Students — profile, guardians (with portal-login management), emergency contacts & authorized pickups, consents (versioned, emailed to guardians), a real class-by-class timetable (not a single homeroom block), classes, attendance history, health (encrypted), documents, IEP and billing tabs.

Registration — enquiry → application → offer/waitlist → enrolment, versioned consent, document upload.

Scheduling — session templates, generated dated sessions with real rosters, whole-school closures and early dismissals (a shorter day, not no school — deliberately modeled as its own concept), terms/academic years, rooms.

Attendance — check-in/out tied to authorized pickups; anyone else is refused on purpose.

Lessons — curriculum units, lesson plans (draft→published), resources that can stand for a whole group, not just one dated lesson.

Grades & report cards — rubric/narrative/marks assessment, four ready-made grading-scheme presets (Bahamas 4.0 GPA, Ontario elementary 4-level, South Africa NCS 7-point, Botswana letter bands) or a custom one, generated cards locked to whichever scheme was active at generation time, review → preview → generate → release workflow, school letterhead and principal's signature.

IEPs — Individual Education Plans, encrypted, any teacher of the student can read/write, a generated letterhead document, visible to guardians on the portal.

Communication — announcements, staff↔parent message threads, incident reports with per-guardian tracked acknowledgement, merge-field email templates — email only, no SMS.

Staff chat — direct messages between staff and role-restricted broadcasts, structurally unreachable from the parent/student portal.

Billing — per-family invoices, a real human-readable invoice number, hosted-checkout online payments (Stripe, Paystack or Flutterwave — the school's own merchant account; Campus never sees a card number), combine several children's invoices into one real charge, payment links for families without a portal login, manual record-payment for cash/cheque/other, per-invoice currency, credits for refunds.

Booking — tutoring/extra-curricular offerings, availability, capacity-limited dated slots, automatic waitlist promotion, ICS.

Staff — directory with a real status beyond active/inactive (sick leave, on leave, sedentary duty, transferred, suspended, terminated), invite flow, group assignment that controls what a teacher can see elsewhere.

School settings, backups, remote access, audit log — branding/letterhead, region & fees (country/currency/locale, billing on/off), on-demand encrypted backups with a health summary, remote-access status, and the append-only audit log.

Parent/student portal — a restricted view: own-children only, real interactive schedule, attendance strip, released report cards, billing with online Pay, message threads, announcements/consents/incidents/IEP summary/bookings where they apply, guardian-initiated change requests that staff must approve.

8. Module catalog — Campus Central (Ministry/district)

Schools registry CENTRAL — register a school, issue/rotate its sync token, see its own sync status.

Cross-school search CENTRAL — students, guardians and staff by name across every school the viewer's role covers; refused entirely for Ministry Analyst.

Case files CENTRAL — on-demand single-student/guardian detail: school, enrolment, attendance, guardians, consents, report-card status.

Dashboards CENTRAL — attendance/enrolment/billing aggregates and the district illness-rate metric, by school or by district, with CSV export.

Ministry administrators CENTRAL — create and manage more than the one original break-glass account; Ministry Admin, Ministry Analyst and District Officer roles, the latter scoped to assigned districts/schools.

Backups CENTRAL — its own encrypted backup/restore scripts (no in-console "Backups" page yet — the scripts' own output is the signal, meant for a scheduled task).

9. Delivery: installers, backups & remote access

Both products install like a desktop application: one `.exe`, Windows services registered, boots on startup, no separate Python or Node install, no Docker. `Campus-Setup.exe` registers four services (PostgreSQL, App, Proxy, and — new — **Campus Scheduler**, the background-jobs

worker that was configured but never actually registered before this pass: nothing on a timer could run for any feature until this was fixed, a product-wide gap, not specific to Central). `Campus-Central-Setup.exe` registers the Central equivalent, minus a scheduler service (Central has no scheduled jobs of its own yet).

Remote access (Campus) offers four modes: a Cloudflare named tunnel, WireGuard VPN, a plain gateway with your own domain, and — new — a **Quick test tunnel** (no Cloudflare account or domain needed, a random `*.trycloudflare.com` address) added after the named-tunnel flow was found to hang indefinitely for an account with no domain already added as a zone. The quick mode's own tradeoff is stated plainly to whoever enables it: the address changes every restart and no access policy can be attached to it — fine for a one-off test, not for ongoing access.

Backups (both products) use the same GPG-encrypted `pg_dump` pipeline, including the `.env` 's encryption key inside the encrypted archive so a restore onto a fresh install (which would otherwise generate a different key) can actually read the encrypted fields back. Neither product schedules a backup automatically — that is an operator's own Task Scheduler entry, documented in the Technical & Troubleshooting Guide.

10. Pricing & licensing

Campus: $1,450 flat, one-time, no per-student or per-user fee, ever — the base package covers every module on one server. Add-ons: LAN access for multiple devices (+$250), remote access (+$500, all three modes priced the same since they differ only in exposure, not cost), data migration (from $350), staff training ($150/hr or $600/half-day), custom branding (+$300), off-box backup destination setup (+$150), priority install (+$200). Buying more than one school's licence at the same time: 40% off base for the 2nd and every one after. Optional ongoing support: 18%/year of licence cost, or $95/hr ad hoc. Full current figures: `Campus_Pricing.pdf`.

> **Campus Central has no published price yet.** It is a genuinely separate product with its own installer, and this paper does not invent a figure for it — that is an open business decision for Dunware Solutions, not a technical one, and is stated here as still open rather than guessed at.

11. Testing and quality

Verified by actually running the suites on 11 October 2026, not carried forward from an earlier pass:

- **Campus's own backend: 425 tests pass.**
- **Campus Central's own backend (`apps/central`): 73 tests pass** (an earlier internal note said 76 — today's direct count supersedes it).
- Both exes have been re-frozen and smoke-tested against real throwaway databases after every fix in this pass, not just imported successfully — a frozen `manage check` was run clean on both, and the real end-to-end Campus↔Central link (register a school, link it, sync, search, open a case file) was driven live on a real machine, not mocked.
- `ruff` clean across the whole backend; `tsc` / `next build` clean on both frontends.

12. Known limits, stated plainly

- Campus Central's sync is not yet on an automatic schedule — the worker now exists (§9), but no recurring job has actually been created for it; sync today is on-demand (manual button / command) only.
- Campus Central does not field-encrypt the student/guardian PII its own cross-school search matches against (§6) — a deliberate, flagged trade-off against breaking that search, not yet resolved with searchable encryption.
- Pickup-authorization data (who may collect a child) is not synced to Central at all — explicitly deferred by its own decision, not yet designed.
- Campus Central has no in-console "Backups" page — only the scripts' own output, meant for a scheduled task.
- The Kanoo (CaribPay) payment gateway for Bahamas-registered merchants is not built; Stripe is documented as unavailable to Bahamas-registered merchants, so a Bahamas install records online card payments only through Paystack/Flutterwave today, or by hand.
- No multi-school *import* tool and no self-serve licensing (e.g. LemonSqueezy) exist yet — every install today is set up directly by Dunware Solutions.
- Translation/i18n is investigated but deliberately parked — Django's i18n flag is on but unused, the Next.js frontend has no translation scaffolding and well over a thousand hardcoded English strings.

- No macOS build exists for either product; both are Windows-only today.
- Campus Central has no published price (§10).

13. Technical reference

Component	Campus	Campus Central
Backend	Django 5.2 LTS + DRF, Python, `waitress` WSGI	Separate Django project, same framework version
Frontend	Next.js (App Router, TS, Tailwind), static export	Separate Next.js project, same stack
Database	PostgreSQL 16, bundled, port 5432	PostgreSQL 16, bundled, port 5433
Reverse proxy / TLS	Caddy, bundled, local HTTPS	Caddy, bundled, default port 8443
Background jobs	`django-q2`, now a registered Windows service	None yet
Packaging	PyInstaller + Inno Setup 7, `Campus-Setup.exe`	PyInstaller + Inno Setup 7, `Campus-Central-Setup.exe`
PDF generation	WeasyPrint + a bundled GTK3 runtime (report cards, IEP documents, these PDFs)	Not used
Sync transport	HTTPS + bearer token (`httpx` , TLS verification delegated to the OS trust store via `truststore`)	Same, receiving end
Tests	425 (pytest, real database)	73 (pytest, real database)