PRD

Final PRD — Janapremi World School

Product Requirements Document & System Logic Map Madhyapur Thimi, Bhaktapur · `janapremi.edu.np`

Prepared as the authoritative pre-development and build reference for the new Janapremi World School website. This document supersedes all earlier drafts and conversations.

Revision 1. This is the first authoritative revision. It draws on a previous institutional build by the same team and its post-build conformance audit, adopting what proved correct there, correcting what did not, and specifying from the outset several things that project discovered mid-build. Where a decision differs from that project's, the difference is stated and the reason given.

How to Read This Document

Part I (Sections 1–9) describes what the system is, who may do what, and why each significant decision was made. It is the part to bring to a management meeting.

Part II (Sections 10–14) specifies the data model, the security model, the storage architecture, and the shared infrastructure that governs every page. It is written for whoever builds and later maintains the system.

Part III (Sections 15–41) breaks down every page and endpoint using a consistent five-part structure: route and visibility, user experience, frontend behaviour, backend behaviour, and security constraints.

Part IV (Sections 42–47) covers operations, the go-live cutover, the build sequence, and the items still requiring a decision from the school.

Decisions taken once and applied in many places are stated once, in Section 7, and referenced rather than re-argued on each page. The operating rules that govern how the code is built live in `CLAUDE.md` at the repository root; **this document is the spec, that file is the working discipline**, and the two are deliberately kept separate so they do not drift.

PART I — FOUNDATIONS

1. Purpose and Scope

Janapremi World School is an established institution in Madhyapur Thimi, Bhaktapur, teaching from early years through Bachelor level, with roughly 1,600 students and over 100 staff. Its current website is an ageing PHP site built over an abandoned WordPress installation, with contradictory content, dead navigation links, and no meaningful search presence.

This project replaces it with a fast, modern, maintainable web presence whose three jobs are:

1. **Present the institution accurately and well.** A parent or prospective student should form a correct, favourable impression — programmes, faculty, achievements, facilities, student voices — on a site that loads quickly and looks current on a phone.
2. **Convert visitors into enquirers.** The admission enquiry forms are the site's primary conversion action, and every design decision touching them is settled in their favour: *the school should never lose a lead to friction or to a technical failure.* Where that principle conflicts with tidiness, this document chooses the lead, and says so.
3. **Serve as the school's recruitment channel.** The career module is a first-class part of this build, not an afterthought, and it handles the most sensitive data in the system.

Unlike a civic accountability system, this site holds no public record that must be defended against its own operators. Its content is promotional, its submissions are private and staff-reviewed, and nothing a visitor submits ever appears publicly or triggers an automated public action. This materially simplifies the security model — but it does not simplify the *privacy* model, because the system holds job applicants' CVs and admission enquirers' contact details, and both belong to people who are not its users.

1.1 In Scope

Public marketing site with: homepage · twelve programme pages across three levels · about page · a full team directory (leadership, faculty, administration, support staff) · admission procedure with two enquiry forms · scholarships · achievements · student testimonials · photo gallery · news, events and notices · downloadable resources · a career module with published vacancies and CV-based applications · a contact page · site-wide level filtering · sequential homepage announcements · and a complete three-tier staff administration area.

1.2 Explicitly Out of Scope for This Build

Public user accounts of any kind · applicant login or application-status tracking · online payment · fee portal · SMS · document issuance · online examination or results lookup · a student or parent portal · a learning-management system · integration with any external registry · a mobile application · multilingual content (the site is English-only, Decision 3) · an activity or audit log (Decision 21).

Any future interactive service requiring public accounts — a fee portal, a student portal, results lookup, a grievance channel — will be built as a **separate application on a subdomain** (for example `portal.janapremi.edu.np`) and linked from this site with a button. It will not share this codebase (Decision 1). This is a deliberate architectural boundary, not a deferral.

1.3 Expected Scale

A single institution. Content volume in the low hundreds of rows per table. Traffic is strongly seasonal, peaking during SEE and NEB result publication and the admission periods that follow. Realistic peak demand is approximately 30,000 pageviews in an admission month; the architecture carries roughly five times that (Section 9).

Several decisions here are correct at this scale and would be wrong at a multi-campus or national scale. Each is marked with the condition under which it should be revisited.

1.4 Findings From the Existing Site

Three things were established by examining the current site, and all three carry into the school's obligations in Section 47:

The school's own content contradicts itself. The homepage claims establishment in 2008, "15+ years", and a 100% pass rate. The introduction page states establishment in 1990 AD (2046 BS), "after 31 years", 1,600+ students, and a 99% SLC success rate. The "31 years" phrasing also dates the copy to approximately 2021. Every one of these figures is a homepage statistic in this build, and the school must supply corrected numbers before launch.

The affiliation logo strip requires verification. The current site displays British Council, Cambridge, UNESCO, and NIEPA marks alongside NEB. NEB and TU affiliations are verifiable. The others require written confirmation from the school before reproduction. A false affiliation claim on a site built by Relentless Lab is a risk that must not be carried silently.

The legacy URL structure must be mapped. Pages are `.php`; images sit under `/wp-content/uploads/`. A 301 map from the old URLs to their new equivalents is a go-live deliverable (Section 43).

---

2. Roles and Permissions

Three roles, and the boundary between them is a data boundary, not a seniority one.

Role	Who	How assigned
Guest	Any visitor, never signed in	Default and only state for the public
Admin	Content managers — the people who post news, upload photos, maintain the team directory	By an Owner, through the Users page
Senior Admin	Staff the institution trusts with personal information — admission leads, enquiries, job applications	By an Owner, through the Users page
Owner	Two accounts: the school's head, and Relentless Lab under the maintenance agreement	Seeded by hand; thereafter granted by an existing Owner

There is **no self-signup anywhere in this system.** A Guest has no account and no path to obtain one.

2.1 The Governing Principle

An Admin manages what the public sees. A Senior Admin handles what the public sends.

That single sentence defines the tier boundary and should be used to settle any future question about where a new module belongs. It is a real principle, not an arbitrary ranking, and it is explicable to the school in one line.

Capability	Admin	Senior Admin	Owner
News, Gallery, Programmes, Team, Achievements, Testimonials, Scholarships, Downloads	Yes	Yes	Yes
Announcements, homepage statistics	Yes	Yes	Yes
Admission forms — *configuration* (publish, options, deadlines)	Yes	Yes	Yes
Vacancies — *postings* (create, edit, close)	Yes	Yes	Yes
Admission submissions	No	Yes	Yes
Contact messages	No	Yes	Yes
Career applications and CVs	No	Yes	Yes
Settings	No	Yes	Yes
User management	No	No	Yes

2.2 The Configuration/Content Split

The subtle part of the table above: **configuring a form and reading what it collects are different permissions.**

An Admin may open or close admissions, edit the stream list, set a deadline, write a vacancy, and close it when filled. An Admin may not read a single admission enquiry, contact message, or CV. Same for Careers: posting a job is public content; reading applications is personal information.

This is the distinction the previous project's flat model could not express, and it is the main reason this build has three tiers rather than two.

2.3 Why Settings Sits at Senior Admin

Settings holds `office_email` and `career_email`. A content Admin able to edit those could redirect every admission lead and every job application to an address they control — reading personal information without ever holding personal-information permissions.

Settings is therefore Senior Admin and above. The two items a content manager genuinely needs — **homepage statistics and announcements** — were removed from Settings for exactly this reason and given their own Admin-accessible pages (Decision 20).

2.4 Owner Rules

- **Two Owner accounts**, held by the school's head and by Relentless Lab.
- **At least one active Owner must always exist**, enforced in the database. The last active Owner cannot be deactivated or demoted by anyone.
- An Owner **cannot deactivate or demote themselves.** The school cannot lock itself out through a single mistake.
- Either Owner may grant or revoke the Owner role on another account.
- The **first Owner is seeded by hand** — initially the developer's own address, so the login and password-reset flow is provable on a real inbox. The school's Owner account is created afterward.

Handover consequence, stated deliberately: because the school's Owner account already exists and already holds the role, ending the maintenance agreement is a matter of Relentless Lab revoking its own Owner status — permitted, since another active Owner remains. The school is left owning its site outright with nothing to coordinate. This is why the "at least one Owner" rule exists rather than an atomic transfer mechanism.

2.5 Deactivation, Not Deletion

An account may be deactivated by an Owner. A deactivated account cannot sign in and cannot write. It is not destroyed, and deactivation is reversible. Because enforcement reads the role live from the database (Section 11.2), deactivation takes effect on the account's very next action.

2.6 Documented Upgrade Path

If the school later separates recruitment from the Senior Admin tier — for example, an HR officer who should see CVs but not admission leads — the addition is a `can_review_applications` boolean on `profiles`, granted by an Owner: one column and one policy branch. It is documented here so the decision has a written home rather than being rediscovered mid-build.

---

3. The Enforcement Model

This is the most important technical concept in the document, stated once here rather than repeated on every page.

The public can read published content from the database, but can never write to it directly. Every public write goes through the server.

Because Guests have no account, no session, and no database credentials, the only key their browser could hold is the anonymous key — and in this system **that key is granted read access to published content and nothing else.** It cannot insert, update, or delete any row in any table.

Every public submission — an admission enquiry, a contact message, a job application — is sent to a **Next.js Server Action or route handler** that runs on the server, holds the privileged service key (which never reaches the browser), performs the bot-protection checks (Section 14.5), and only then writes the row. There is exactly one write path into public-submitted data, and it sits on the server behind the checks.

Four layers exist, and the meaning of "authoritative" is precise:

- **Layer 1 — Middleware.** Runs before admin pages load. Confirms a valid, active staff session exists and redirects anyone without one to `/login`. It gates exactly `/admin/:path*` and `/account/:path*`. It checks **session only, never role** (Section 11.2).
- **Layer 2 — Admin layout and live role check.** The `/admin` layout confirms the session belongs to an active staff account, reading the role **live from** `profiles` on every request, never from the session token.
- **Layer 3 — Route-level role gates.** Each module's page confirms the acting user meets its `minRole` before rendering. Navigation hides what a role cannot use.
- **Layer 4 — Row Level Security in PostgreSQL.** Every table carries policies. The anonymous role may `SELECT` only published rows of public content tables and may write nothing, anywhere. Staff writes are permitted by policy, scoped to tier. These policies run inside the database, beneath the website, and cannot be bypassed by inspecting the site in a browser.

Only Layer 4 is authoritative. The practical consequence that governs the whole build: **hiding a control is never a security measure.** A restriction exists only if a database policy enforces it. A content Admin who types `/admin/submissions` directly is redirected by Layer 3 *and* returned nothing by Layer 4 — and the second is the one that matters.

One exception, and it is deliberate: private file access is not RLS-governed, because private files do not live in the database (Section 12). CV access is enforced by an admin-only server route that names an *application*, never a file, and issues a short-lived presigned URL only if that application is readable under normal role rules. The enforcement is equivalent; the mechanism differs, and Section 13.4 states it precisely so no maintainer assumes RLS is covering it.

> **Contrast worth recording for maintainers.** In an account-based civic system, signed-in users hold tokens and can address the database directly, so a website-side bot check is worthless and RLS must do everything. Here the opposite holds: there are no public accounts, the anonymous key is read-only, and so a server-side bot check is *genuine, un-bypassable* protection. Do not "simplify" this later by letting the browser insert submissions with the anonymous key — that would reintroduce a direct write path and render every bot check decorative.

---

4. Level — The Site-Wide Vocabulary

This build introduces one shared concept the previous project did not have, and it is worth stating before the modules that use it.

Three levels, defined once:

Value	Covers
`school`	Early Years through Grade 10
`plus_two`	Grades 11–12, all four streams
`bachelor`	BBS and BASW

Applied to: News, Downloads, Achievements, Gallery albums, Scholarships, Team (faculty), Programmes, and Career vacancies.

Stored as a required array with at least one value. "All levels" means all three selected — **not** null-means-everything. Null is the ambiguity that quietly breaks a filter query six months later. The admin form provides a "select all" convenience, so it is one click either way.

This is what lets a parent filter the whole site by "my child is in +2", and it is the single largest usability improvement over the previous project. It is deliberately a coarse vocabulary: programme pages remain granular (Early Years, Primary & Middle, Secondary), while the three buckets exist for *filtering*, not for describing academics.

Testimonials deliberately carry no level. Testimonials are aspirational content — a school parent reading a Bachelor graduate's testimonial is a good outcome, and filtering would work against the module's purpose.

---

5. Language and Dates

The site is **English-only** (Decision 3). There is no language toggle, no parallel-field storage, and no machine translation anywhere. Content is authored once, in English.

A **Devanagari-capable fallback font** is still specified, because the school's name and occasional proper nouns will appear in Nepali script within copy and image alt text. Without it, those render as empty boxes on some devices.

All timestamps are stored timezone-aware and rendered in **Nepal Standard Time (UTC+05:45)**. The 45-minute offset is not cosmetic: naive handling shows the wrong day for anything recorded between 18:15 and midnight local time.

All displayed dates use the **Gregorian** calendar (Decision 4). There is no Bikram Sambat rendering in this build. Where the school's own authored prose references BS years, that is authored text and is left as written.

---

6. Technology Stack

Layer	Choice	Notes
Application framework	Next.js (App Router) + TypeScript	Server-rendered; public writes via Server Actions and route handlers
UI	React + Tailwind CSS	Orange-led design system (Section 8)
Database & authentication	Supabase (PostgreSQL)	**Database and auth only.** No file storage
Image storage & delivery	Cloudinary	Transformed delivery only, never originals
Document storage	Backblaze B2	Two buckets — public and private. S3-compatible API
Edge & DNS	Cloudflare	DNS, CDN, and a Worker serving public documents
Transactional email	Resend	Verified sending subdomain at go-live
Bot protection	Honeypot + timing trap + database-backed per-IP rate limiting	No challenge widget at launch (Decision 12)
Maps	Static Google Maps embed	Find-us only; low stakes
Hosting	Vercel, with Netlify and Cloudflare as documented alternatives	See Decision 22

Layer	Choice	Notes
Source control	GitHub, **private** repository	Handles applicant and enquirer personal data

---

7. Decisions and Rationale

Every decision below is applied throughout the document and is not re-argued on the pages it affects.

Decision 1 — No public accounts. Any future portal is a separate subdomain application.

This build has three staff roles and no self-signup. The entire account subsystem a portal would need — signup, email verification, public session handling, password recovery for the public — is absent, because nothing a visitor does here requires an identity. If the institution later wants a fee portal, a student portal, or results lookup, that is a distinct application with a distinct risk profile, built at `portal.janapremi.edu.np` and linked with a button.

Keeping the two apart prevents a marketing site and an account-holding service from sharing a codebase and a blast radius. The current site's condition was not bad luck; it was the predictable result of an ageing, plugin-heavy stack. This build avoids that whole class of problem from the first commit.

Decision 2 — Three roles, with a data boundary between content and personal information.

The previous project shipped with two roles in its PRD and three in its code, and the conformance audit correctly identified that standing contradiction as the worst possible outcome. This document adopts the tier as spec from the outset, and extends it: the boundary is not seniority but *access to personal information*.

The role is stored as an **enum**, not a boolean, so a further tier is a new value plus a few policy branches. Every access check routes through one of three `SECURITY DEFINER` functions (Section 11.1), so adding a tier is never a rewrite.

Decision 3 — The site is English-only.

The institution's audience and existing content are predominantly English. Parallel bilingual storage doubles authoring effort for content that changes constantly, and doubles every admin form. It is not justified here. Every text field is single, not paired.

Decision 4 — Gregorian dates throughout.

No Bikram Sambat rendering, no dual-calendar display, no conversion library. The school's audience reads Gregorian dates on notices without difficulty, and dual-calendar support is a persistent source of off-by-one errors for a benefit this audience does not need.

Decision 5 — Two admission forms, split by level. Fields fixed in code; options admin-managed.

The school admits at three levels but takes online enquiries at two. **School admission (Early Years to Grade 10) is handled at campus** and has no form — a decision that reflects the school's actual process rather than forcing a channel it does not use.

The two forms are **+2** and **Bachelor**, each a row in `admission_forms` carrying its own publish toggle, editable heading and description, and optional informational deadline. The form's fields are fixed in code, keyed by the form's id — there is no dynamic form-builder, which would be high-effort, high-risk flexibility the school does not need.

Each form has exactly **one admin-managed picker**: stream for +2 (Science, Management, Law, Humanities), programme for Bachelor (BBS, BASW). Because forms are rows, adding a School form later — if the school ever opens online school admissions — is a row plus a code schema, not new plumbing.

Decision 6 — Admission enquiry forms accept no document uploads at all.

This is a lead-capture form, not an application. The office phones back; the family brings documents to campus, as they already do.

Three consequences, all good:

- **The largest projected storage workload in the system disappears.** Several hundred families uploading marksheets and citizenship, most of whom never enrol, would have dwarfed everything else the site stores.
- **The form gets faster and shorter** — no upload widget, no progress state, no orphaned-file problem on abandonment, no "my scan won't upload" phone call during admission season. This serves the never-lose-a-lead promise better than an upload toggle ever would.
- **One less category of family personal information** held on behalf of people who never enrol.

Stated plainly so it is not "helpfully" added back later: **admission enquiry is lead capture. Documents are brought to campus.** If the school later wants true online applications with document intake, that is portal-shaped and lives on the subdomain app.

Decision 7 — Admission form options are retired, never deleted.

The stream and programme lists change year to year, and mid-season a stream fills up. The *field* is fixed in code; its *options* are data an Admin edits.

Removing an option means **retiring it** — an `is_available` flag hides it from the applicant picker immediately, while every past submission keeps its label intact (stored as text) and remains filterable by it. Hard-deleting an option would silently rewrite the meaning of enquiries already made under it. The same flag doubles as "this stream is full."

The option lists ship empty. The admission in-charge populates them before publishing. The system guards against a published-but-empty form (Section 27): rather than render a broken empty picker, it shows an "admissions opening soon" state and warns on the admin dashboard.

Decision 8 — Career is a first-class module, and it is the only anonymous write door into storage.

The school publishes vacancies; candidates apply with a CV. This is the first module in any Relentless Lab build handling third-party personal information from anonymous submitters, and it is specified accordingly:

- **Five applicant fields only** — full name, email, phone, address, CV. Everything else is in the CV. Every additional field is friction on a form whose whole purpose is to be completed.
- **CV is PDF only, strictly 2 MB.** The cap is defended in the error message ("most CVs are well under 2 MB — if yours is larger, it is probably a scan") rather than as a bare rejection. This is the one place where the never-lose-an-applicant principle and a hard technical limit genuinely conflict, and the resolution is a clear message rather than a silent failure.
- **No citizenship, no academic certificates, no supporting documents.** Citizenship is an employment document, not an application document; the school verifies identity at appointment, in person, as it already does. Storing scanned identity documents for hundreds of people the school will not hire is liability with no operational payoff.
- **This is the system's sole anonymous upload path.** Every other upload purpose requires an authenticated staff session. It is PDF-only, size-capped, and rate-limited per IP.

Decision 9 — CVs are purged six months after a vacancy closes; the application record is kept.

The file is deleted; the name, email, phone, status, and staff note remain as hiring history. This is stated on the privacy page, warned about on the admin dashboard **fourteen days before** each purge, and executed by a scheduled job.

A retention job that deletes files without warning is the kind of automation that gets discovered badly. The fourteen-day notice makes it a policy rather than a surprise, and gives the Senior Admin a last chance to act on anything still live.

Decision 10 — Vacancy open/closed has exactly one definition, in the database.

Accepting applications = published AND not manually closed AND (no deadline OR deadline not passed). Defined once, read everywhere.

A list page deriving status from dates while a detail page reads an independent toggle is two sources of truth that will eventually disagree, showing a candidate one answer on one page and another on the next. The manual close matters separately from the deadline: a position filled early must stop accepting applications without waiting for a date.

Career deadlines are real and auto-close. Admission deadlines are informational and never auto-close (Decision 11). Two different semantics in one system is a maintenance trap unless stated explicitly, so it is stated here and labelled in both admin interfaces.

Decision 11 — Admission deadlines are informational only.

A school rarely wants to hard-stop enquiries; a late enquirer is still a lead. The deadline displays to set expectations and does nothing else. The admin field is labelled to say so, and closing admissions is a separate explicit toggle — per form, plus a **master "close all admissions"** for the off-season.

Decision 12 — Bot protection is a lightweight, invisible stack: honeypot, timing trap, and database-backed per-IP rate limiting. No challenge widget at launch.

The entire cost of a bot getting through is one junk row in a staff-reviewed pipeline. Nothing a visitor submits appears publicly or triggers an automated action. The bar is therefore "keep the inbox usable without ever risking a real lead," not "stop every bot."

- **Honeypot** — a decoy field hidden from people, filled by naive bots. Named and marked `autocomplete="off"` so password managers do not populate it and flag a real person.
- **Timing trap** — submissions arriving implausibly fast are rejected. The threshold is set **generously**; it catches only the obviously automated.
- **Per-IP rate limiting** — the load-bearing one, and the only one that also protects the anonymous CV upload endpoint from being used to burn storage.

Rate limiting is database-backed, not in-memory. In-memory limits on serverless hosting are per-instance and degrade into a meaningless approximation under exactly the load they exist to handle. A `rate_limits` table with a `SECURITY DEFINER` function holds correctly across every instance.

The posture on verification failure is fail-open with a flag (Decision 12a): an ambiguous check accepts the submission tagged `unverified_review` rather than dropping it, so the worst case for a real enquirer is a two-second dismissal by staff, never a silently lost lead.

A challenge widget remains a documented, ready-to-add upgrade if reviewed spam ever exceeds a level staff find tolerable. It is not built now.

Decision 13 — Three storage services, each holding what it is actually good at.

Holds	Where	Why
Images	Cloudinary	Transformation and format conversion is its actual value
Public documents — Downloads, brochure	Backblaze B2 public bucket	Documents gain nothing from an image CDN
Private documents — CVs	Backblaze B2 private bucket	Isolated from public content by bucket, not by prefix
Everything else	Supabase	Database and auth only

The reasoning that produced this split is worth recording, because the intuition most people have is wrong: **the constraint on media services is bandwidth, not storage.** A single results PDF fetched by a thousand families moves more data than every image the site will ever hold. Routing documents through an image service would have been the most expensive possible arrangement.

Delivery is layered so no origin service sees visitor traffic directly:

- Images: Cloudinary → framework image optimisation → Cloudflare edge → visitor
- Public documents: B2 → Cloudflare Worker on `files.janapremi.edu.np` → Cloudflare edge → visitor
- Pages: statically generated → Cloudflare edge → visitor

Each layer caches. Origin services see roughly one request per unique asset per edge location, not one per visitor.

Decision 14 — All uploads are signed; delivery type is decided by the server, per purpose.

Signing an upload controls *who may put a file into the account*; delivery type controls *who may look at it once there*. These are independent, and conflating them is a common and consequential error.

Every upload is signed. The server maps each stated purpose to a fixed configuration — bucket or folder, format allow-list, size limit, delivery type — and the browser cannot influence it. A browser cannot request that a CV be stored publicly.

Content	Delivery	Cap
Gallery photos, news covers, team photos, achievement images, testimonial photos, programme covers	Public (Cloudinary, transformed)	4 MB, downscaled in browser first
Downloads, brochure	Public (B2 public bucket)	4 MB
CVs	Private (B2 private bucket)	2 MB, PDF only

Public images are served through a resizing transformation, never as originals — which strips embedded GPS metadata from phone photos and protects the media quota.

Every image is downscaled in the browser before upload, long edge capped at approximately 1920px. This cuts upload time on Nepali connections, cuts storage, and cuts every derived asset's cost. It is also what makes the school's instruction to "compress everything" real: it happens before any service sees the file.

Decision 15 — Private file access names a submission, never a file.

The request identifies an *application*; the server reads that application under normal role rules and, only if it exists and the caller may read it, issues a **five-minute presigned URL** for the CV recorded on that row. The browser never names a file, and the server never signs a file the browser named.

Under a design where any signed-in user may request a link for any file identifier, a staff member who obtained or guessed another identifier would receive a valid link to a document they should not see. Naming the submission means the existing role rules do all the work, with no special cases.

Presigned URLs are generated locally by the SDK, never through an API call, and are never cached at the edge.

Decision 16 — All storage access goes through one adapter.

`lib/storage/` exposes `upload()`, `getSignedUrl()`, `delete()`, and `list()`. Drivers are selected by environment variable.

This is the same discipline as a configurable sending address: the decision is isolated once, so changing storage provider is configuration plus a file migration, never a refactor touching routes, policies, or admin pages. Given that this project's storage arrangement was itself revised three times before build, the abstraction has already paid for itself.

Three implementation rules live in the adapter and are stated here so they are not lost:

1. **Use the S3-compatible endpoint, never the native provider API.** Native APIs typically require a separate authorisation call per session, which is a metered transaction on a request path.
2. **Generate presigned URLs locally from the secret key.** No API call, no transaction.
3. **Never list a bucket in a request path.** Listing belongs only in scheduled jobs.

Decision 17 — Leadership and Team are one table.

The previous project ended with two parallel faculty tables — two tables of identical shape — and its audit correctly flagged the cost. Here, one `team_members` table serves the About page's leadership block, the homepage carousel, the full team directory, and every programme page's faculty list.

A teacher who teaches both +2 Science and Bachelor exists **once**. A department head who is also leadership exists once. Programme pages read this table filtered by level and department; there is no second roster anywhere in the system.

Four categories, fixed in code because they drive page structure and their rate of change is zero: **Leadership · Faculty · Administration · Support Staff.**

One point of institutional care: support staff are titled with dignity — "Office Assistant" not "Peon", "Security Officer" not "Guard", "Housekeeping Supervisor" not "Sweeper". These names sit on a public page beside the Principal's, and getting it right costs nothing.

Decision 18 — Programmes are two-level throughout, and structure is uniform.

Parent	Children	Level
School	Early Years · Primary & Middle (Grades 1–8) · Secondary (Grades 9–10)	`school`
Plus Two	Science · Management · Law · Humanities	`plus_two`
Bachelor	BBS · BASW	`bachelor`

Three parents, nine children, twelve rows in one self-referencing table. A child inherits its parent's level.

The Grade 8 boundary is not arbitrary — it is the actual structural break in Nepal's school system. The school's current site splits this four ways (Pre-Primary, Elementary, Middle, Secondary); collapsing to three means one fewer page for the school to write, with the Primary & Middle page carrying internal sections for each phase.

The School parent page becomes genuinely useful: it is where the largest audience segment lands, and where the School Admissions message (Decision 5) naturally sits alongside the three child cards.

Decision 19 — Markdown in, opinionated rendering out.

Programme bodies, news bodies, and leadership messages are markdown. A rich WYSIWYG editor would let staff paste styled HTML from Word and the site would degrade within two years.

The renderer is heavily opinionated instead (Section 8.3): tables become designed components, blockquotes become callouts, headings generate a sticky section navigation, lists get styled markers. Staff write plain markdown and get a designed page.

Structured facts do **not** live in markdown. Each programme carries an **at-a-glance strip** of 3–5 admin-defined label/value pairs, rendered consistently across all twelve pages. If duration and affiliation lived in the body, one page would say "Duration: 2 years", another "The programme runs for two academic years", and a third would forget.

Decision 20 — Homepage statistics and announcements sit outside Settings.

Both are content a content manager should change seasonally, and neither is an escalation vector. They get their own **Admin-accessible** pages, leaving Settings holding only escalation-sensitive keys at Senior Admin.

Statistics move from fixed settings keys into a small table, so the school can relabel and reorder them without a migration.

Announcements are two, sequential. The second renders when the first is dismissed. Dismissal is keyed to each announcement's file identifier independently with a three-day window, so replacing an image re-reaches people who dismissed the previous one.

The SEO trade is recorded rather than discovered: search engines treat stacked mobile interstitials as an intrusive-interstitial signal, and search visibility is load-bearing for a school with no existing presence. The recommendation to the school is to run two simultaneously only during admission season. The capability is built because it is the school's call to make.

Decision 21 — No activity log.

Every row carries `created_by` and `updated_by`, which answers "who last touched this." A full append-only event stream is a real build cost, grows without bound, and — in a system with no adversarial relationship between the institution and its own record — is the kind of thing that gets built and never read.

This is a genuine difference from a civic accountability system, where the log *is* the product. Here it is not.

Decision 22 — Hosting is portable by construction.

Vercel is the first choice. Netlify and Cloudflare's paid tier are documented alternatives, and the school's site should be movable between them within a day.

The build therefore avoids host-specific APIs where a portable equivalent exists. This is cheap insurance: a marketing site should never be captive to one platform's terms.

Decision 23 — `janapremi.edu.np` is canonical; `jws.edu.np` redirects and keeps the mail.

The school controls both domains. Neither has meaningful existing search presence, so the choice is purely branding, and the institution's real name wins.

`jws.edu.np` **301-redirects to the canonical, path-preserving**, so legacy URLs land on their correct new pages rather than dumping everyone on the homepage.

Mailboxes stay on `jws.edu.np`, untouched. This removes the single most dangerous step in the go-live cutover — breaking school email by failing to recreate MX records — because nothing about the mail configuration changes. Printed materials, signboards, and years of saved contacts stay valid.

Transactional email sends from a verified subdomain of the canonical domain, with `Reply-To` **set to the real mailbox**, so a candidate's reply reaches a human rather than a send-only address.

Decision 24 — A backup strategy is required before launch.

The database tier in use includes no automated backups and pauses after roughly a week of inactivity. For a site holding admission leads and job applications during season, neither is acceptable.

A scheduled keepalive prevents pausing. **That mitigates the symptom, not the absence of backups.** Before launch the institution either funds a tier with automated backups or commits to a scheduled, tested export with a stated acceptable data-loss window. **An untested backup is not a backup.**

---

8. Design Direction

The frontend is a complete departure from the previous project's. This section is direction, not final design; the visual system is produced in a design pass after this document.

8.1 Orange, Used With Restraint

The school's identity is orange. The discipline that separates a premium school site from a fast-food aesthetic is **where orange is allowed to appear.**

- **Deep or burnt orange** for text, links, and interactive elements — pure orange fails contrast against white for body text
- **Bright orange** reserved strictly for calls to action and accents
- **Warm neutral base** with near-black text carrying the majority of the page

Orange everywhere reads cheap. Orange held back reads confident.

8.2 Typography

Most of the perceived quality of a content page is typographic, before any component work:

- Body text at 18–19px, line height ~1.7, measure capped around 68 characters. The audience reads on phones; cramped text is the single biggest thing that makes a page feel cheap
- A display face for headings, distinct from the body face
- Generous vertical rhythm between blocks — whitespace reads as confidence
- A Devanagari-capable fallback in the stack (Section 5)

8.3 The Markdown Renderer

One component, used by Programmes, News, and leadership messages. Hard-styled overrides for:

- **Tables** — accented header row, zebra striping, row hover, horizontally scrollable on mobile with a fade edge. Curriculum tables are the most important content on programme pages and the ugliest by default; this is the single biggest visual win available
- **Blockquotes → callouts** — three variants driven by a leading marker (Note, Important, Tip), rendered as accented cards
- **Headings** — `h2` with real breathing room and an accent marker; `h3` tighter, as a sub-label
- **Lists** — accent markers instead of bullets, more spacing, slightly larger text
- **Section navigation** — `h2` headings parsed into a sticky table of contents with scroll-spy. Zero admin effort, and it is what makes a long page feel navigable rather than endless

The admin editor previews through the identical renderer. That is what prevents the "why does it look different on the site" support call.

8.4 Photo Normalisation

Team photos will be patchy — good for leadership, mixed for faculty, poor or missing for support staff. The design absorbs this rather than exposing it:

- Fixed square crop with face-detection gravity, so a badly framed photo still centres on the person
- Consistent card treatment — same corners, same border, same hover behaviour — which does a surprising amount to make mismatched lighting and backgrounds read as one set
- **Initials placeholder in the brand palette** where a photo is missing, never a grey silhouette or a broken image

8.5 Motion

Sections fade and rise slightly on scroll, once. Statistics count up. Nothing loops, nothing bounces. This is the difference between "modern" and "template".

---

9. Capacity and Its Ceilings

A full analysis is held separately. The findings that constrain design are summarised here because they should inform decisions, not be discovered after them.

Measure	Figure
Sustainable monthly pageviews	~150,000
Realistic peak demand	~30,000 in an admission month
Concurrent public visitors	Effectively unbounded — pages are static and edge-cached
Public document downloads	~500,000/month before any limit
Content ceiling at year three	~40 gallery albums · ~600 news posts · ~150 team members · ~200 downloads

Traffic is not the constraint. A statically generated site behind layered caching scales almost independently of visitor numbers. What binds are per-item ceilings that do not grow with audience:

1. **Transactional email daily cap** — the tightest ceiling in the system. A popular teaching vacancy attracting fifty applications in one day generates a hundred emails (office notification plus applicant acknowledgement), which sits at the limit. The failure is silent: the office simply stops being notified. Mitigations, in order: drop the applicant acknowledgement first under load; batch office notifications into a digest above a threshold; raise the allowance.
2. **Image transformation count** — bounded by the 30-photo album cap (Decision 25, Section 32). Relaxing that cap moves this ceiling up the list.
3. **Nothing else**, at any modelled scenario.

The intended property: a school site should not become more expensive to run because more parents visited it. Every remaining ceiling is bounded by policy — a retention purge, an album cap, a notification strategy — rather than by popularity.

---

10. What This System Deliberately Does Not Do

- It does **not** create public accounts, log anyone in, or let an applicant track a submission (Decision 1).
- It does **not** take payment.
- It does **not** issue documents, admit letters, offer letters, or results.
- It does **not** send SMS. Email only.
- It does **not** verify that a submitted document is genuine.
- It does **not** publish anything a visitor submits. All submissions are private and staff-reviewed.
- It does **not** accept documents on admission enquiry forms (Decision 6).
- It does **not** keep an activity log (Decision 21).
- It does **not** guarantee that every automated bot is stopped (Decision 12) — it guarantees that a real enquirer is never turned away, and that bot noise is bounded and reviewed.
- It does **not** retain CVs indefinitely (Decision 9).

PART II — SYSTEM ARCHITECTURE

11. Security Model

11.1 The Three Role Functions

Every "may this staff member do this?" check routes through one of three dedicated `SECURITY DEFINER` functions, each declared with `set search_path = ''`:

Function	True for	Gates
`current_user_is_active_admin()`	admin, senior_admin, owner	Public content policies
`current_user_is_senior_admin()`	senior_admin, owner	Personal-information tables and Settings
`current_user_is_owner()`	owner	User management

Each reads the current user's row from `profiles` and returns true only if the role qualifies **and** `is_active` is true. They run with elevated rights so they are not themselves subject to the policies they inform — which is what prevents infinite recursion

when a policy on `profiles` needs to ask `profiles` about the current user's role.

The roles nest, so most policies need no branching. Written once, used everywhere. **They read live from the table, never from the sign-in token.**

11.2 Staff Authentication and the Live Role Check

- **No signup, anywhere.** No signup route, no trigger creating profiles from `auth.users`, no INSERT policy on `profiles` for anonymous or authenticated users, no DELETE policy for anyone.
- **The first Owner is seeded by hand.** One auth record created in the dashboard, one profile row inserted by SQL. This is manual because no existing Owner exists to authorise it.
- **The live role check is load-bearing.** Middleware confirms only that a session exists. The `/admin` layout then reads `profiles` **live on every request**, never trusting the role embedded in the session token. This is what makes a deactivation or demotion take effect on the account's next action rather than up to an hour later when the token would refresh.
- `profiles` **policies.** SELECT: a user may read their own row; a senior admin may read all. UPDATE: a user may change only their own `full_name` — never their own `role`, `is_active`, `email`, or `id`. The edit-anyone policy lives only on the Users page and is written against `current_user_is_owner()`.

11.3 The Read-Only-Public Rule

The anonymous role is granted `SELECT` on **published rows** of public content tables and nothing else — no INSERT, UPDATE, or DELETE on any table, ever. Unpublished drafts are excluded by policy, not hidden by the page.

This single rule is what makes the server the only write door for the public.

11.4 Server-Mediated Submissions

Admission enquiries, contact messages, and job applications are written by a server route holding the service key, which:

1. Runs the bot-protection checks (Section 14.5)
2. On an ambiguous check, accepts the row flagged `unverified_review` rather than dropping it
3. Validates the answers against the form's code-defined schema
4. Sets reference, status, verification, and timestamps itself, never accepting them from the browser
5. Records the chosen picker option **as text**, not a foreign key, so a later retirement cannot orphan the row

The browser never writes directly.

11.5 The Service-Role Key

The privileged service-role key lives only in `lib/supabase/service.ts` — which begins with `import 'server-only'`, so any attempt to import it into client code becomes a **build error rather than a silent browser leak** — and in server routes. It bypasses all RLS and is used only after bot and permission checks. It is never exposed to the browser, never logged, and never committed.

Storage credentials follow the same rule and live only in `lib/storage/`.

11.6 Media Authorisation

Upload signing maps a stated purpose to a fixed server-side configuration; the browser cannot request that a CV be stored publicly. **The CV upload purpose is the only purpose open to anonymous callers** and is rate-limited per IP. Every other purpose requires an active staff session of the appropriate tier.

Private-file viewing names an *application*, not a file, and is restricted to Senior Admin and above (Decision 15).

---

12. Storage Architecture

12.1 The Three-Way Split

Holds	Service	Delivery path
Images	Cloudinary	Framework image optimisation → Cloudflare edge
Public documents	B2 public bucket	Cloudflare Worker on `files.janapremi.edu.np` → Cloudflare edge
Private documents (CVs)	B2 private bucket	Admin-only route → five-minute presigned URL, never cached
Database and auth	Supabase	—

Supabase holds no files. Its storage allowance is unused, leaving its full capacity for the database.

12.2 The Worker

A small Cloudflare Worker on `files.janapremi.edu.np` fetches an object from the public bucket, sets a long cache header, and returns it. Cloudflare caches the response at the edge; the second request onward never reaches origin.

Two reasons for a Worker rather than an application route: the bytes never pass through the application host, and it keeps document delivery working unchanged if the application moves hosting provider (Decision 22).

The Worker also **increments the download counter** for the requested file before returning it.

12.3 Buckets, Not Prefixes

Public and private documents live in **separate buckets**, not in separate folders of one bucket. A misconfiguration on one cannot expose the other. The private bucket has public access disabled entirely.

12.4 Orphans

The browser uploads before the record is saved. If a submission is abandoned, the file remains with nothing referencing it. A monthly reconciliation removes unreferenced objects.

---

13. Data Model

Roughly twenty tables. Every table has `created_at`; editable tables also have `updated_at`, refreshed by the database rather than the application. Every content table records `created_by` and `updated_by`.

13.1 Accounts

`profiles` — one row per staff account.

Field	Notes
`id`	References `auth.users(id)` on delete cascade
`email`	Login identity and staff-page display
`full_name`	Editable by the account holder
`role`	Enum: `admin` · `senior_admin` · `owner`. Default `admin`
`is_active`	Defaults true. False blocks sign-in and all writes
`created_at`, `updated_at`	

Database constraint: **at least one active Owner must exist at all times.**

13.2 Public Content

`posts` — `title`, `slug`, `type` (news/event/notice), `level` (array), `excerpt`, `body` (markdown), `cover_image` (nullable), `event_date` (nullable, events only), `is_published`, `published_at`. One table serves all three types, distinguished by `type`. Events sort by `event_date`; everything else by `published_at`.

`gallery_albums` — `slug`, `title`, `cover_photo`, `event_date`, `level` (array), `display_order`, `is_published`.

`gallery_photos` — `album_id`, `photo`, `caption` (nullable), `display_order`. Capped at 30 per album.

`achievements` — `title`, `description`, `image` (nullable), `level` (array), `achieved_on`, `display_order`, `is_published`.

`testimonials` — `student_name`, `programme` (free text), `quote` (≤60 words), `photo` (nullable), `display_order`, `is_published`. No level, by design.

`scholarships` — `title`, `description` (markdown), `criteria` (markdown), `level` (array), `display_order`, `is_published`.

`downloads` — `title`, `description` (≤100 words), `level` (array), `file`, `file_size`, `file_type`, `download_count`, `display_order`, `is_published`, `published_at`.

`programmes` — `slug`, `title`, `level`, `parent_id` (nullable, self-referencing), `intro`, `body` (markdown), `cover_image`, `gallery_album_id` (nullable), `is_published`, `display_order`.

`programme_facts` — `programme_id`, `label` (short), `value` (short), `display_order`. Maximum five per programme.

`team_members` — `full_name`, `designation`, `category` (enum), `level` (array, faculty only), `department` (free text, faculty only), `qualification` (nullable), `short_description` (≤30 words, nullable), `full_message` (markdown, leadership only), `photo` (nullable), `display_order`, `show_on_homepage`, `is_published`.

13.3 Admissions

`admission_forms` — `id` (fixed text key: `plus_two`, `bachelor`), `title`, `description`, `option_label`, `is_published`, `deadline` (nullable, informational), `display_order`.

`admission_form_options` — `form_id`, `name`, `display_order`, `is_available`. Ships empty. `is_available = false` retires an option without deleting it. **No delete path is exposed.**

`admission_submissions` —

Field	Notes
`reference`	`JWS-P2-2026-00042` / `JWS-BACH-2026-00042`, database counter, never reissued
`form_id`	
`full_name`, `phone`, `email`, `address`	Promoted to columns for a readable pipeline
`option_choice`	The chosen stream or programme, **as text**
`previous_institution`	Optional
`gpa`, `results_awaited`	GPA required unless awaited
`payload`	Remaining per-form answers, shape fixed in code
`status`	New / Reviewed / Contacted / Enrolled / Archived
`verification`	`verified` or `unverified_review`

13.4 Careers

`vacancies` — `title`, `slug`, `category`, `level` (array, nullable for non-teaching), `department`, `employment_type`, `positions`, `excerpt`, `body` (markdown), `salary_text` (default "Negotiable"), `deadline` (nullable), `is_published`, `is_closed`, `closed_at`.

`applications` — `reference` (`JWS-CAR-2026-00042`), `vacancy_id`, `full_name`, `email`, `phone`, `address`, `cv_key`, `cv_purge_after`, `status` (New / Shortlisted / Hired / Rejected), `staff_note`, `verification`.

13.5 Communications

`contact_messages` — `name`, `email`, `phone`, `message`, `status` (New / Read / Archived), `verification`.

13.6 Homepage and Configuration

`homepage_stats` — `label`, `value`, `display_order`, `is_active`. Maximum four active.

`announcements` — `image`, `link_url` (nullable), `alt_text`, `display_order`, `is_active`. Maximum two active, enforced in the admin.

`settings` — key/value store, keys fixed by migration with no INSERT or DELETE from the application: `office_email` · `career_email` · `brochure_file` · `brochure_label` · `school_admissions_text` · `career_empty_state_text`.

`rate_limits` — supports Decision 12's database-backed limiting.

Reference counters — one per prefix, never reissuing a number.

13.7 What There Is No Table For

No public counter, dashboard, or tracker — this is a marketing site, not an accountability system. **No status-history table** — status plus `updated_at` adequately serves a lead pipeline (Decision 21). **No activity log. No second faculty roster** (Decision 17). **No admission document table** (Decision 6).

---

14. Shared Infrastructure

14.1 Session Handling

Middleware runs before every `/admin/:path*` and `/account/:path*` request, refreshes an expiring staff session, and redirects an unauthenticated or deactivated visitor to `/login`. It does not gate public pages and does not check role.

14.2 Email Link Handling

A route at `/auth/confirm` receives every link the system emails — staff password reset and staff invitation — validates the single-use token, establishes a session, and forwards onward. Email templates must point here rather than at their defaults. There is no signup-confirmation flow, because there is no public signup.

14.3 Email Sending

`RESEND_FROM` and `RESEND_REPLY_TO` are environment variables. During development the sending identity is the provider's onboarding domain, which requires no DNS and delivers only to the account owner's verified address — exactly what is wanted while testing. At go-live it flips to a verified subdomain address, a one-line change.

`RESEND_REPLY_TO` **points at the school's real mailbox on** `jws.edu.np` (Decision 23), so replies reach a human.

Notification recipients are configuration, not hardcoded: `office_email` receives contact enquiries and admission submissions; `career_email` receives job applications.

14.4 Media Handling

Upload authorisation — `/api/media/sign-upload`**.** The browser states a purpose and nothing else. The server confirms the caller may use that purpose, maps it to a fixed bucket, folder, format allow-list, size limit, and delivery type, signs that, and returns it. The browser never names the folder, filename, or delivery type.

Viewing authorisation — `/api/media/sign-view`**.** Senior Admin and above. The request names an *application*; the server reads it under normal rules and, if readable, returns a five-minute presigned URL for the CV recorded on that row.

Uploads go **browser-to-storage directly**, never through the application server. Images are downscaled in the browser first (Decision 14).

14.5 Bot Protection

Applied on every public write path — admission submissions, the contact form, and the CV upload endpoint — as the invisible stack of Decision 12. Fail-open with a flag. No visible challenge to a real person at launch.

14.6 Content Freshness

Public pages are pre-rendered and cached. Every staff action changing public content explicitly refreshes the affected pages — publishing a news post refreshes the news list and the homepage; adding a gallery album refreshes the gallery and any programme linking it. Admin pages render fresh per request.

14.7 Link Previews, SEO, and Structured Data

Every public page carries Open Graph tags generated from its own content, falling back to the school's logo and a standard description.

The site generates a sitemap and a robots file. The previous project promised both and shipped neither; here they are **Phase 1 deliverables**, verified before the phase closes.

Structured data where it earns its place:

- `JobPosting` on every vacancy — this is what places the school in job search results, and no competitor school in Bhaktapur will have it
- `Event` on events carrying a date and location
- `EducationalOrganization` on the homepage

14.8 Data Export

Every staff list page carries an export-to-spreadsheet action producing a file from what is on screen. **Export inherits its list's role restriction** — a content Admin cannot export what they cannot read.

14.9 Environments and Configuration

- **Local development runs against live cloud services.** There is no public deployment until go-live; every milestone, including the email milestone, is provable locally.
- **Secrets live only in environment variables.** `.env.example` is committed with empty values. Nothing sensitive is ever committed.
- **The repository is private**, because the system handles applicant and enquirer personal data. This is defence in depth, not a substitute for the security model.
- **The spec and operating rules live in the repo:** this document at `/docs/JWS_PRD.md`, the operating discipline at `/CLAUDE.md`.

PART III — PAGES AND ENDPOINTS

Each entry follows the same five-part structure. Rules established in Parts I and II are not repeated.

15. Homepage — `/`

1. **Route & Visibility.** Public.
2. **Experience.** Institutional hero with a primary call to action to `/apply` and a secondary brochure download — **no carousel**, which is measurably ignored and slows the most important page on the site. Then: the four homepage statistics, animating on scroll; a programmes overview with three cards for School, +2, and Bachelor; a leadership carousel; an achievements strip; a mixed news and events feed; student testimonials; an icon-led "Why Janapremi" facilities block; and a find-us block carrying the School Admissions message. A sticky apply call to action persists; up to two sequential announcements may appear.
3. **Frontend.** Navigation and calls to action. Statistics animate once. Announcements are dismissible by button, backdrop, or Escape.
4. **Backend.** Reads: active homepage statistics; published programmes at parent level; team members flagged for the homepage; recent achievements; latest posts across all types; published testimonials; active announcements.
5. **Security.** All reads public, published rows only.

Achievements sit above the news feed. News is the school's most *recent* content; achievements are its most *persuasive*. A parent choosing between three schools is moved by a district topper, not by a holiday notice.

Every section hides itself when empty — no empty headings, no placeholder grids. The site launches with several modules unpopulated and must look deliberate throughout.

Performance target: under 400 KB on first load. Hero image priority-loaded; everything else lazy.

16. About — `/about`

Public. Institutional history, vision, and mission as static prose, plus the leadership block read from `team_members` filtered to the Leadership category. The Principal's message lives in exactly one place — this block — and is not duplicated as separate

static copy.

17. Programmes Index — `/programmes`

Public. Three parent cards. One read of published parent programmes in display order.

18. Programme Parent — `/programmes/[slug]`

Public. Intro, shared content, cards for each child programme, and an apply call to action where that level's form is open. The **School** parent page additionally carries the School Admissions message (Section 27).

19. Programme Detail — `/programmes/[parent]/[slug]`

1. **Route & Visibility.** Public.
2. **Experience.** Hero and cover; the **at-a-glance strip** of 3–5 facts; an apply link where relevant; the markdown body with a sticky section navigation generated from its headings; the faculty roster; and, where an album is linked, a gallery strip.
3. **Frontend.** Section navigation with scroll-spy. Gallery strip is a horizontal scroll opening the shared lightbox.
4. **Backend.** One read of the programme and its facts; one read of `team_members` filtered by level and department; one read of the linked album's photos if present.
5. **Security.** Published rows only.

No related downloads and no related news. Deliberate — the page is a programme description, not a hub.

20. Team — `/team`

1. **Route & Visibility.** Public.
2. **Experience.** Category tabs — Leadership, Faculty, Administration, Support Staff — **defaulting to Leadership**, which has the strongest photos and the fewest cards. Leadership cards carry a short excerpt; every other card shows photo, name, and `designation · department`. All cards are clickable. Faculty adds a level filter and a name/department search.
3. **Frontend.** Leadership opens a large modal with photo, name, designation, qualification, and full message. Everyone else opens a compact modal with photo, name, designation, level, department, qualification, and short description — degrading cleanly when fields are blank. Photos for non-default tabs load only when that tab opens.
4. **Backend.** One read of published team members in category and display order.
5. **Security.** Public read. **No contact details are published for any team member.**

21. News, Events & Notices — `/news`

1. **Route & Visibility.** Public.
2. **Experience.** A mixed feed by default, newest first. Two filter rows: type (All / News / Events / Notices) and level (All / School / +2 / Bachelor). The Events view splits into Upcoming and Past. Cards show title, type badge, level badges, date, excerpt, and cover where present.
3. **Frontend.** Filter state is held in the URL, so a parent can bookmark or share "school notices".
4. **Backend.** Server-side pagination, twelve per page. Events ordered by `event_date`, others by `published_at`.
5. **Security.** Published rows only.

Cards are designed to work without a cover image — the no-cover card is a distinct layout, not a broken version of the cover card.

22. News Detail — `/news/[slug]`

Public. Cover where present, title, type and level badges, date (or event date, time, and location for events), markdown body through the shared renderer, a share action whose Open Graph tags render the link as an official-looking card, and three related posts matched by level.

23. Gallery — `/gallery`

Public. Album cards with cover, title, date, and photo count, with a level filter. Ready to group by year once volume justifies it — `event_date` already exists, so no schema change is required later.

24. Gallery Album — `/gallery/[slug]`

Public. A masonry grid of photographs, thumbnails at approximately 400px, lazy-loaded in batches of twenty, full size loaded only when the lightbox opens. Captions optional; the lightbox is designed to look correct without them. Photographs served through a resizing transformation, stripping location metadata and protecting the media quota.

25. Achievements — `/achievements`

Public. Timeline-style list, newest first, level filter. Title, description, image where present, and date. No distinction between student and institutional achievements.

26. Student's Voice — `/students-voice`

Public. Staggered quote cards — student name, programme, quote, optional photo with initials fallback. No level filter (Section 4).

27. Admissions — `/admissions` and `/apply`

1. **Route & Visibility.** Public.
2. **Experience.** The stable procedure prose, then cards for each currently published admission form. If none is published, a clear "admissions are not currently open" state.

   Both pages also carry the **School Admissions block**:

   > **School Admissions (Early Years – Grade 10)** School admission is handled at our school. Call us on **01-6630967** or follow the link for general enquiry and location.

   Phone as tap-to-call, link to `/contact`. Copy editable from Settings.
3. **Frontend.** Navigation only.
4. **Backend.** One read of published admission forms.
5. **Security.** Public read.

This block is not a placeholder for a missing feature. It is the school's actual process, and a parent of a Grade 3 child must never reach a dead end.

28. Admission Enquiry Form — `/apply/[form]`

1. **Route & Visibility.** Public. Limited to `plus_two` and `bachelor`; anything else is not found.
2. **Experience.** The form's editable heading and description, then its code-fixed fields:

   Shared — student's full name, contact phone, email, address, the form's picker, previous school or college (optional).

   +2 adds — stream picker; SEE GPA with a "Results awaited" checkbox; parent or guardian phone (optional).

   Bachelor adds — programme picker; +2 subject (free text); +2 GPA with a "Results awaited" checkbox.

   A consent line above the submit button, **not a checkbox**: *"By submitting, you agree that Janapremi World School may contact you about admission."*

   On success, a reference number and confirmation that the office will be in touch.

   Guard for an empty option list. If a form is published but has no available options — the launch state until the in-charge fills the list — the form does not render an empty picker. It shows an "admissions opening soon" message, and the admin dashboard flags the condition.
3. **Frontend.** The invisible bot stack is present. Checking "Results awaited" disables and clears the GPA field. **No file upload appears anywhere on this form.** Client-side validation is friendly and permissive — never block a real enquirer over a formatting quibble.
4. **Backend.** The server runs the bot checks (fail-open with flag), validates against the form's schema, records the chosen option as text, sets reference, status, verification and timestamps itself, writes one row, and emails `office_email`.
5. **Security.** No account required and none created. The browser cannot write directly. The endpoint is rate-limited per IP.

29. Careers — `/careers`

1. **Route & Visibility.** Public.

2. **Experience.** Currently open vacancies as cards — title, category, level, employment type, deadline. Filters by category and level.

   When no vacancies are open, the page does not dead-end. It shows an editable "why work at Janapremi" block, the live team grid pulled from `team_members`, and a direct email call to action with a suggested subject line. The page always feels alive, costs nothing extra, and doubles as employer branding a candidate reads *before* a vacancy opens.
3. **Frontend.** Filters work on the loaded list.
4. **Backend.** One read of vacancies where the single open/closed definition (Decision 10) returns true.
5. **Security.** Public read.

There is no spontaneous-application path. No CV upload exists when no vacancy is open — which also removes a permanent stream of unsolicited documents the school would have to store and eventually purge.

30. Vacancy Detail — `/careers/[slug]`

1. **Route & Visibility.** Public. A closed vacancy **keeps a live detail page** with a clear closed state; it simply drops off the listing.
2. **Experience.** Title, category, level, department, employment type, positions available, salary text, deadline, and the markdown body. Then the application form.
3. **Frontend.** `JobPosting` structured data. Where closed, the form is replaced by the closed notice.
4. **Backend.** One read.
5. **Security.** Public read.

Keeping closed pages live matters: job listings must remain reachable through their validity window to index correctly, and a dead link from a shared social post reflects on the school.

31. Job Application — part of `/careers/[slug]`

1. **Route & Visibility.** Public, where the vacancy is accepting applications.
2. **Experience.** Five fields — full name, email, phone, address — and one CV upload, **PDF only, 2 MB**. On success, a reference number and confirmation.
3. **Frontend.** The bot stack is present. The browser requests upload permission (Section 14.4) and uploads directly to storage, then submits the answers plus the file reference. **The size limit is stated before upload**, and the rejection message names it and suggests the file is likely a scan.
4. **Backend.** The server runs the bot checks, validates, writes one row with a purge date six months beyond the vacancy's close, emails `career_email`, and sends the applicant an acknowledgement containing the reference. **Status-change emails are never automatic** (Section 39).
5. **Security.** The CV uses private delivery in a dedicated bucket. **This endpoint is the system's only anonymous upload path** and is rate-limited per IP.

A duplicate email applying twice to the same vacancy is **flagged in the admin list, never blocked** — without accounts, duplicate detection cannot be reliable, and blocking would reject a legitimate resubmission after a failed upload.

32. Scholarships — `/scholarships`

Public. Expandable cards — title and summary visible, criteria on expand — with a level filter, ending in a link to the relevant admission form. A scholarship page that does not lead to applying is a dead end.

33. Downloads — `/downloads`

Public. A flat list newest first, with a level filter and a title search. Each row shows title, description, file type badge, file size, publication date, download count, and a "New" badge for seven days. File size is displayed because a parent on mobile data deserves to know before tapping.

Files are served through the Worker on `files.janapremi.edu.np`.

34. Contact — `/contact`

1. **Route & Visibility.** Public.
2. **Experience.** Address, phone numbers (tap-to-call), the school email, office hours, social links, and a static map. Then a general enquiry form: name, email, phone, message.

3. **Frontend.** The invisible bot stack.
4. **Backend.** The server runs the checks, writes one `contact_messages` row, and emails `office_email`.
5. **Security.** Browser cannot write directly; rate-limited per IP.

35. Privacy — `/privacy`

Public, static prose. States what the site stores — admission enquiries, contact messages, job applications and CVs — that these are visible only to authorised staff, that nothing submitted is ever published, **that CVs are deleted six months after a vacancy closes**, that admission enquiries carry no documents, and how a person may request correction or removal.

The privacy page must accurately reflect this document. Where the two disagree, this document is wrong until corrected.

36. Login and Password Reset — `/login`, `/reset-password`

Staff only. A failed login shows an inline generic message ("Invalid email or password") without revealing whether an address is registered. Password reset uses the built-in flow; the emailed link lands at `/auth/confirm`, which validates the single-use token, establishes a session, and forwards to the change page.

37. Admin Landing — `/admin`

1. **Route & Visibility.** Any active staff account.
2. **Experience.** Role-scoped navigation and an attention summary. Quick actions: add news, add team member, upload download, new vacancy.

   The attention summary is hidden entirely from Admin, not shown as zero — a count is itself information about volume.

   Warnings:

Warning	Who sees it
Published admission form with no available options	Admin+
Vacancy past deadline still published	Admin+
More than two announcements active	Admin+
Storage above 80% of ceiling	Senior Admin+
Submissions unreviewed over 7 days	Senior Admin+
Applications unreviewed over 7 days	Senior Admin+
CVs scheduled for purge within 14 days	Senior Admin+

   A **storage readout is always visible** — "412 MB of 10 GB" — not only when warning. A number that appears only once it is a problem teaches nobody; a number always present means the school develops a feel for what a thirty-photo album costs.
3. **Frontend.** No data changes here.
4. **Backend.** Counts, scoped by role.
5. **Security.** The live role check runs in the layout. Every module below independently enforces its own policies.

38. Admin — Content Modules

*News, Gallery, Programmes, Team, Achievements, Testimonials, Scholarships, Downloads, Announcements, and Homepage Statistics share a structure. All are **Admin and above**.*

- **News.** CRUD, publish, markdown editor with live preview, level multi-select with "select all", event date/time/location where type is event.
- **Gallery.** Albums with covers and level; **batch photo upload** with per-file skip and continue and progress, capped at thirty per album.
- **Programmes.** Edit intro, body, cover, linked album, and the at-a-glance facts (max five). Publish and reorder. **No faculty editing here** — faculty lives in Team.
- **Team.** One filterable list with inline reorder, plus a **quick-add mode** — a compact repeating row for entering many people in one sitting without a page load each. Designation is a controlled list per category with an "Other" escape; department autocompletes from existing values.

- **Achievements / Testimonials / Scholarships / Downloads.** Standard CRUD with level, images or files where relevant, and display order.
- **Announcements.** Maximum two active, enforced at save.
- **Homepage Statistics.** Maximum four active, label and value editable, reorderable.

Each write records the acting user and refreshes affected public pages.

Quick-add and batch upload are not conveniences. Ninety-five team members and thirty-photo albums entered one full form at a time is the single most common reason these modules ship half-populated.

39. Admin — Admissions and Careers Configuration

Admin and above.

`/admin/admissions` — the two forms as manageable rows: publish, edit heading and description, set an informational deadline, plus a **master close-all toggle**. An options panel per form: add, reorder, and **retire** (flag off — the option leaves the picker, past submissions keep their label, no hard delete). A warning shows if a published form has no available options.

`/admin/vacancies` — CRUD on vacancies: title, category, level, department, employment type, positions, salary text, deadline, body. Publish and manual close.

Neither page shows a single submission or application. That is the Section 2.2 boundary made concrete.

40. Admin — Submissions, Messages, and Applications

Senior Admin and above.

1. **Route & Visibility.** `/admin/submissions`, `/admin/contact-messages`, `/admin/applications`.
2. **Experience.**
   - **Submissions** — every enquiry across both forms, filterable by form, status, chosen option, and results-awaited; searchable by reference; exportable. Each shows contact details, the answers, the verification flag (an `unverified_review` row is obvious and dismissable in seconds), and the status, which staff advance.
   - **Contact messages** — name, email, phone, message, status, **searchable and exportable**. Both were promised in the previous project's PRD and never shipped; here they are baseline.
   - **Applications** — filterable by vacancy and status, searchable by reference, exportable. Each shows the five applicant fields, a **CV viewer opening inline** rather than downloading, a **staff note** written after reading, a duplicate-email flag, and the purge date.
3. **Frontend.** CV viewing follows Section 14.4 — name the application, receive a five-minute link. Status changes are inline. Status-change emails to applicants are a **separate explicit action**, never automatic: a rejection firing the moment a dropdown changes is a reputational decision the school should make deliberately, and staff often change status in bulk while tidying up.
4. **Backend.** One read each, unrestricted by ownership — these are staff-only records. Status changes write status and `updated_at`.
5. **Security.** Senior Admin and above at both the route and the database. CVs are reachable only through the admin-only view path. Archiving rather than deleting is the office convention.

The staff note on applications exists for a specific reason. With only five applicant fields, the list has nothing to triage on — staff would open every CV, every time. A note written once after the first read ("MSc Physics, 8 years, Budhanilkantha") makes the list scannable thereafter. It is admin-authored and never applicant-facing.

41. Admin — Settings, Users, and Account

`/admin/settings` — *Senior Admin and above.* `office_email`, `career_email`, brochure file and label, and the two editable copy blocks (School Admissions message, career empty state), both markdown rendered through the shared renderer.

`/admin/users` — *Owner only.* All accounts with name, email, role, active status, created date, and last sign-in. Create a staff account (invitation via email, landing at `/auth/confirm`), assign role, deactivate and reactivate, grant and revoke Owner. Guards: no self-deactivation, no self-demotion, at least one active Owner enforced in the database.

Carries a plain-language note, because a school Admin will otherwise see a role they cannot reach and wonder why:

> *Site ownership is held jointly by the school and Relentless Lab under the maintenance agreement. Contact us to transfer ownership or change administrators.*

`/account` — any signed-in staff member edits their own name and password. Email is shown, not edited. The database rule restricts every account to its own row and never to `role` or `is_active`.

PART IV — DELIVERY

42. Operations

42.1 Backups — Required Before Launch

The database tier has no automated backups and pauses after inactivity (Decision 24). **Recommended:** fund a tier with daily backups and no pausing. **Minimum acceptable:** a scheduled export, tested by actually restoring it once, with a stated acceptable-loss window.

A scheduled keepalive prevents pausing but does not substitute for backups.

42.2 Scheduled Jobs

Job	Frequency	Purpose
Database keepalive	Daily	Prevents tier pausing
CV retention purge	Weekly	Deletes CVs six months past vacancy close; dashboard warns 14 days ahead
Orphan sweep	Monthly	Removes uploaded-then-abandoned objects
Storage usage recompute	Daily	Feeds the dashboard readout

42.3 Monitoring

Error reporting on the live site. Monthly image-service quota check. Monthly storage check. Daily email-volume awareness during hiring periods (Section 9).

And the one that matters most: **during admission season, a periodic confirmation that submission emails are actually arriving.** A silently broken notification is the highest-cost failure this site has, because it loses leads invisibly and no quota, log, or dashboard will report it. Test it by observation, and re-test it when the season opens.

42.4 Storage Caps and Alerts

Where the storage provider supports spend caps, **set every cap to zero** and enable alerts at 75%. With no payment method on file there is nothing to bill, but an explicit zero cap means the account blocks rather than accrues, and the alert gives warning long before anything stops. This is a standing configuration recorded in the handover document.

42.5 Handover

Required before launch: written credential custody for every service; a documented deployment process; this document kept current in `/docs`; the school's Owner account created and verified; and at least two people at the school who know how to reach the developer.

Ownership model, stated for the contract as much as the spec:

What	Held by
Asset ownership — domains, database, storage, code, content	The school, outright
Operational control — Owner account, deployment access	Relentless Lab, under the maintenance agreement
Attribution — a small credit in the footer	Perpetual

Because the school's Owner account already exists and already holds the role, ending the maintenance agreement means Relentless Lab revoking its own Owner status. The exit is a two-click action rather than a database intervention, so neither party depends on goodwill.

43. Go-Live: Domain, DNS, and Email Cutover

The site is developed against live cloud services with no public deployment. Go-live is a deliberate, sequenced cutover.

`janapremi.edu.np` **is canonical.** `jws.edu.np` **redirects and keeps the mail** (Decision 23).

1. **Inventory the existing** `jws.edu.np` **zone before touching nameservers** — every record, and in particular the **MX and SPF/TXT records for the existing mailboxes**. This is the single most forgettable step and the one that breaks school email if missed. It remains in the runbook even though the mail configuration is not changing.
2. **Build the 301 map** from legacy `.php` URLs and `/wp-content/uploads/` image paths to their new equivalents.
3. **Deploy** with production environment variables, `RESEND_FROM` still pointing at the onboarding domain.
4. **Move both domains' DNS to Cloudflare.**
   - `janapremi.edu.np` : site records; `files.janapremi.edu.np` for the Worker; Resend's sending records for `mail.janapremi.edu.np` (DKIM, SPF, optional DMARC).
   - `jws.edu.np` : **its existing MX and SPF records copied across exactly**, plus one path-preserving redirect rule to the canonical domain. **No site records.**
5. **Add the custom domain** to the hosting platform.
6. **Verify the sending domain**, then flip `RESEND_FROM` to the verified address and confirm `RESEND_REPLY_TO` points at the school's real mailbox.
7. **Confirm end to end:** the site loads on the canonical domain; legacy URLs redirect to their correct destinations; a test admission enquiry emails the office from the verified address; a test job application emails the career address and acknowledges the applicant; a test download serves through the Worker; and `info@jws.edu.np` **still sends and receives.**

Optional, free, and worth recording: email routing on `janapremi.edu.np` can forward `info@janapremi.edu.np` to the real mailbox, so mail sent to the new domain does not bounce as new signage and printed material appear. Not required at launch.

44. Build Sequence

The riskiest surface — Careers, with an anonymous signed upload, private delivery, a retention job, and third-party personal data — is built last, after every technique it needs is proven in cheaper form.

Phase 0 — Foundation. Profiles; the three role functions and RLS; seeded Owner accounts; login, session, and logout with the live role check; password reset and `/auth/confirm` ; the storage adapter; the Cloudflare Worker; database-backed rate limiting; and the **contact form** as the first end-to-end public write.

Definition of done: a stranger on their own phone sends a contact message that lands in the admin list and emails the office, and staff can sign in and reset a password unaided. **Prove this before proceeding** — everything downstream reuses this exact pattern.

Phase 1 — Public shell and core content. The design system and markdown renderer; homepage; News; Gallery; Downloads; static pages; **sitemap, robots, and Open Graph, verified before the phase closes.** No sensitive data anywhere.

Phase 2 — Institutional content. Programmes with the at-a-glance strip and section navigation; Team with quick-add; Achievements; Testimonials; Scholarships; Announcements; Homepage statistics; Settings; Users.

Phase 3 — Admissions. Two forms; admin-managed option lists with the empty-state guard; the submission pipeline; office notification; export.

Phase 4 — Careers. Vacancies with structured data; the CV pipeline with private delivery; the applicant acknowledgement; the retention job with its dashboard warning; the empty state.

If time runs short, the lowest-cost deferrals are additive rather than load-bearing: download counts, the "New" badge, and the programme gallery strip. **None of them touches the promise that a real enquirer is never turned away.**

45. Risks

Risk	Mitigation
Content, not code, delays launch — the school owes corrected statistics, verified affiliations, team photos, and programme copy	Section 47 lists these against phases. The previous project's schedule risk was almost entirely content

Risk	Mitigation
Silent notification failure during admission season	Explicit periodic testing (Section 42.3). This is the highest-cost failure in the system
Email daily cap during a hiring spike	Documented mitigations in Section 9, applied in order
Team module ships half-populated	Quick-add mode; initials placeholders; category-specific card layouts that look correct with sparse data
The 4 MB download cap is hit during results week	Error message names the limit and advises re-scanning at lower resolution; raising the cap is a one-line change if it recurs
Legacy URLs lost at cutover	The 301 map is a go-live deliverable, verified in step 7

46. What Changed From the Previous Project, and Why

Recorded so the differences are deliberate rather than accidental.

Area	Previous project	Janapremi	Why
Roles	Two in spec, three in code	Three, specified	The audit's most consequential finding, resolved in advance
Role boundary	Seniority	Access to personal information	Expressible, explicable, and enforceable
Faculty	Two parallel roster tables	One `team_members` table	The audit's second finding
Leadership	Static in PRD, CMS in code	CMS, specified	Decision 10's rate-of-change line, drawn correctly
Level filtering	None	Site-wide vocabulary	Largest usability gain
Admission documents	Optional per-form upload	None at all	Removes the largest storage load and the most family PII
Storage	Cloudinary for everything	Three services by fitness	Bandwidth, not storage, is the constraint
Rate limiting	In-memory	Database-backed	In-memory is per-instance and fails under load
Sitemap and robots	Promised, not built	Phase 1 deliverable	The audit's clearest gap
Contact search and export	Promised, not built	Baseline	Same
Career module	None	First-class	New requirement, and the highest-risk module

47. Decisions Required From the School

Required before Phase 1:

1. **Corrected institutional history and statistics.** The current site's homepage and introduction page contradict each other on founding year, years of operation, and pass rate (Section 1.4). These become the four homepage statistics and the About page's history.
2. **Written confirmation of affiliations.** NEB and TU are verifiable. British Council, Cambridge, UNESCO, and NIEPA marks require confirmation before reproduction.
3. **Static content** — About page history, vision, and mission; the "Why Janapremi" facilities list.
4. **Brand assets** — logo files, and any existing colour or typography guidance.
5. **A named content owner** at the school. One person responsible for supplying copy and photographs.

Required before Phase 2:

6. **Team roster and photographs** — approximately ninety-five records across four categories, with photographs where available.
7. **Programme copy** for all twelve pages, including curriculum tables and the three-to-five at-a-glance facts per programme.

8. **Confirmation of support-staff titles** (Decision 17).

Required before Phase 3:

9. **The two notification addresses** — `office_email` and `career_email`.
10. **Initial stream and programme option lists**, and which are open this session.

Required before launch:

11. **Backups** — fund a tier with automated backups, or approve a tested scheduled export with a stated acceptable-loss window (Decision 24).
12. **Hosting tier funding**, appropriate to a commercial project (Decision 22).
13. **Content migration scope** — which existing pages, news posts, and images carry over and which start fresh.
14. **Privacy page confirmation** — that Section 35's statement is accurate and acceptable, particularly the six-month CV retention.
15. **Credential custody** — who at the school holds the Owner account, and which two people can reach the developer.

---

48. Closing Note

This build is larger than the previous project's, and most of the additional weight sits in three places: a career module handling documents belonging to people who are not the school's students, a team directory of roughly a hundred real people with their photographs on a public page, and a level vocabulary that ties eight modules together.

Two commitments run through all of it.

The first is the promise inherited from the previous project, unchanged: the institution must never lose a real enquirer to friction or to a technical failure. Every decision touching a form was settled in its favour — the invisible bot stack over a visible challenge, fail-open over fail-closed, a generous timing trap, permissive client validation, five fields on the career form rather than fifteen, and no document upload on an enquiry form at all. A parent who fills a form should reach the office every time; a bot that slips through costs a staff member two seconds. That asymmetry is the whole design, and it is the right way round.

The second is new, and it belongs to the career module. This system holds CVs from people who applied for a job they will mostly not get. They are not the school's students, not its customers, and they will never log in to check what is held about them. The protections around that data — a dedicated private bucket, access that names an application rather than a file, a role tier that keeps content managers out, a purge with a date on it and a warning before it fires — exist because nobody in that group is in a position to ask for them.

The third thing worth defending is the boundary that keeps this a marketing site and pushes any future account-holding service onto its own subdomain. It costs a little ambition now — no portal, no logins, no fee payment — and buys a great deal of safety and simplicity. The current site's condition was not bad luck; it was the predictable result of an ageing, plugin-heavy setup that advertised its own software and was hard to keep patched. This build is designed to avoid that entire class of problem from the first commit: one clean custom system, a read-only public, secrets that live only on the server, and a repository kept private not because the design needs secrecy but because a school's data deserves the care.