EXAMTRACKER INDIA

Admin Dashboard Architecture

Technical Reference • February 2026 • v1.0 • Solo Founder Setup

---

Overview & Purpose

The admin dashboard is an internal tool — only accessible to you (the founder). It is built as a protected section of the main Next.js app at **/admin/*** behind a middleware guard that checks for a specific admin role in the Supabase session.

The dashboard has **one primary job**: get verified exam data into the database as fast as possible, so users receive accurate notifications. Everything else is secondary.

BUILD ORDER	Build the admin dashboard BEFORE the scraper and BEFORE the PDF pipeline. Sprint 1 is manually entering 50 exams into this form. This validates the schema, teaches you what fields are easy/hard to find in real PDFs, and gets the product in front of users fast. The automation comes later and pre-fills this same form.

The dashboard has 5 sections:

Section	Route	Purpose
Exam Queue	*/admin/queue*	PDF review queue — exams waiting for approval. The main daily-use screen.
Add / Edit Exam	*/admin/exams/new and /admin/exams/[id]*	The data entry form. 8 field groups. Used manually and pre-filled by automation.
Exam List	*/admin/exams*	Browse all exam records. Search, filter, bulk actions.
Analytics	*/admin/analytics*	User counts, notification stats, exam coverage gaps.
Site Health	*/admin/health*	Scraper status per site, last successful scrape, error log.

Access Control — Admin Only

The entire /admin/* route tree is protected by Next.js middleware. No public user should ever reach these routes.

How admin auth works:

- In Supabase, set a custom claim on your own user account: app_metadata.role = "admin". This is done once via the Supabase dashboard → Authentication → Users → your account → Edit.

- Next.js middleware at middleware.ts checks every request to /admin/*. It reads the Supabase session JWT, checks app_metadata.role === "admin". If not → redirect to /404 (not /login — never confirm that /admin exists to non-admins).
- The check happens server-side in middleware — not client-side. No admin UI is ever rendered for non-admins.
- In future if you hire a data editor: create their Supabase account, set role = "editor" (a separate lower-privilege role). Editors can add/edit exams but not delete or access analytics/user data.

Section 1: The Exam Queue

This is the screen you open every morning. It shows all exam records in DRAFT or NEEDS_REVIEW status, sorted by urgency (application_end date soonest first). Your job: review each record and either Approve or Reject.

QUEUE	**/admin/queue** *Daily working screen — typically 5–20 records at launch, growing with automation*

Queue card — what each record shows:

Each exam in the queue is shown as a card with:

- Exam name + conducting body + category badge (SSC / UPSC / BANKING / STATE etc.)
- Application end date — colour coded: RED if < 3 days, ORANGE if < 7 days, GREEN otherwise
- Source: MANUAL or AUTO (AUTO means pipeline pre-filled it) + confidence score if AUTO
- Status badge: DRAFT / NEEDS_REVIEW / APPROVED / REJECTED
- Quick action buttons: "Review & Approve" (opens full form) + "Quick Approve" (if confidence score > 0.90) + "Reject"
- Link to original PDF — opens in new tab so you can verify data without leaving the queue

QUICK APPROVE	If the pipeline confidence score is > 0.90 AND application_end is > 7 days away, show a "Quick Approve" button that approves the record without opening the full form. This lets you process high-confidence records in 5 seconds. At MVP (100% manual), this button is hidden — every record goes through the full form.

Queue filters:

- Filter by status: DRAFT / NEEDS_REVIEW / APPROVED / REJECTED / ALL
- Filter by source: MANUAL / AUTO
- Filter by category: SSC / UPSC / BANKING / RAILWAY / STATE / PSU / DEFENCE
- Filter by urgency: closing in 7 days / 14 days / 30 days
- Sort by: application_end (default), created_at, confidence_score

Section 2: The Add / Edit Exam Form

This is the most important UI in the entire dashboard. It is used in two modes: (1) Manual entry — you fill everything from scratch reading the PDF. (2) Auto-review — pipeline pre-fills most fields, you verify and approve.

The form is divided into **8 field groups** that match the database schema exactly. Each group is a collapsible section. When automation pre-fills a field, it shows the confidence score and source snippet alongside the value so you can verify without opening the PDF.

#	Group Name	Fields	Notes for you
Group 1	**Core Identity**	Exam name, short name, conducting body, category, level (central/state/PSU), state code (for state exams)	*Always required. Usually the easiest to fill — it's in the PDF title/header.*
Group 2	**Key Dates**	Notification date, application start, application end ⚡, exam date, admit card date, result date, age cutoff date	*Application end is the most critical field — highlighted in red if missing. All dates use a date picker with DD/MM/YYYY format (how Indian PDFs show dates).*
Group 3	**Vacancy Data**	Total vacancies, vacancies_by_category JSONB (UR/OBC/SC/ST/EWS/PwBD/ESM)	*The vacancy table UI is a mini-spreadsheet: 7 columns, one row per vacancy category. If only total is known, fill total and leave category table empty.*
Group 4	**Age Eligibility**	Min age, max age per category (General/OBC/SC_ST/EWS/PwBD/Ex-Serviceman), age cutoff date	*7 separate fields — one per category. Do not auto-compute. Always enter the exact number from the PDF. "As per GOI rules" → look up the standard table.*
Group 5	**Education**	Required qualification (dropdown: Class 10 / 12 / ITI / Diploma / Graduation / Eng Graduation / PG), streams (multi-select), min marks %, allows final year, additional qualifications	*Qualification dropdown must map to the ENUM exactly. Streams are important for engineering exams.*
Group 6	**Other Eligibility**	Nationality, gender restriction (leave blank = no restriction), physical requirements (height/vision — JSONB), marital status, domicile required toggle	*Physical requirements only for Defence/Police exams. Most exams: leave blank.*
Group 7	**Application Fees**	Fee for General, SC/ST, PwBD, Women — separate fields. Fee payment modes (checkboxes).	*Very commonly ₹100 General / ₹0 SC/ST/PwBD. Quick fill buttons: "Standard GOI fees" auto-fills 100/0/0/100.*
Group 8	**Source & Verification**	Official notification URL, data source (MANUAL/AUTO), notes field	*URL is already filled from scraper or you paste it. Notes field for anything unusual about this notification (e.g. "Corrigendum — updates vacancy count from original notification")*

Auto-fill field display (when pipeline pre-fills):

Each pre-filled field shows three things beside the value: the confidence score (green/orange/red dot), the extraction method (REGEX / HAIKU / INFERRED), and a collapsible "Source snippet" — the exact text from the PDF this was extracted from. This lets you verify a field in 3 seconds without opening the PDF.

Validation rules:

- application_end is required — form cannot be submitted without it
- max_age_obc must be ≥ max_age_general (warn if not)
- exam_date must be after application_end (warn if before)
- total_vacancies must be positive integer
- vacancies_by_category values must sum to ≤ total_vacancies (not equal, because reserved categories sometimes add up differently)
- official_notification_url must be a valid URL (not just any text)

Form submission actions:

- "Save as Draft" — saves without triggering any notifications. For incomplete records.
- "Approve & Notify" — sets notification_verified = TRUE, is_active = TRUE. Immediately triggers the new_exam_notification BullMQ job. Users get emails within 2 hours.

- "Approve without Notify" — sets notification_verified = TRUE but does NOT trigger notification job. Use when updating/correcting an already-live exam rather than adding a new one.
- "Reject" — sets status = REJECTED. Adds rejection reason (free text field). Rejected records are archived, never shown to users.

Section 3: Exam List

A searchable table of all exam records in the database. Primarily used to find and edit existing records rather than for daily data entry.

LIST	**/admin/exams** *Browse, search and manage all exam records*

Table columns:

- Exam name (clickable — opens edit form)
- Category badge + Level (CENTRAL/STATE/PSU)
- Application end date + days remaining
- Status (DRAFT / APPROVED / REJECTED / EXPIRED)
- Vacancies (total + has_category_breakdown indicator)
- Data source (MANUAL / AUTO) + confidence score
- Actions: Edit | Duplicate | Archive

Search & filter:

- Full-text search on exam name (uses PostgreSQL pg_trgm index — already set up in schema)
- Filter by category, level, status, state, date range
- Bulk actions: Bulk approve, bulk archive, bulk export (CSV for your records)

DUPLICATE FEATURE	The Duplicate button creates a new DRAFT copy of an existing exam with all fields pre-filled. Critical for annual exams like SSC CGL — SSC CGL 2026 is 90% identical to SSC CGL 2025. Duplicate, update the dates and vacancy numbers, approve. Saves 5 minutes of re-entry.

Section 4: Analytics

A simple read-only dashboard showing product health metrics. Pulls data from PostgreSQL — no external analytics tool needed at MVP.

ANALYTICS	**/admin/analytics** *Key metrics at a glance — updated every hour*

Metrics shown:

Metric	Source Query	Why it matters
Total registered users	*SELECT COUNT(*) FROM users WHERE is_deleted = FALSE*	How many people have signed up

Metric	Source Query	Why it matters
Users who completed onboarding	*SELECT COUNT(*) FROM user_profiles WHERE onboarding_completed = TRUE*	Onboarding completion rate
Daily active users (last 7d)	*SELECT COUNT(DISTINCT user_id) FROM user_sessions WHERE created_at > NOW()-7d*	Product engagement signal
Total active exams in DB	*SELECT COUNT(*) FROM exams WHERE is_active = TRUE AND notification_verified = TRUE*	Data coverage — are you tracking enough?
Exams closing in next 7 days	*SELECT COUNT(*) WHERE application_end BETWEEN TODAY AND TODAY+7*	Urgency — are important deadlines covered?
Emails sent today	*SELECT COUNT(*) FROM notification_log WHERE sent_at::date = TODAY*	Notification system health check
Email open rate (7d)	*opened_at IS NOT NULL / total sent*	Deliverability and engagement quality
Notification failures (24h)	*SELECT COUNT(*) FROM notification_log WHERE status = 'FAILED'*	Pipeline health — any Resend issues?

Section 5: Site Health

Shows the scraping status for each of the 142 monitored sites. This is your early warning system when a government site changes its layout and breaks a scraper.

HEALTH **/admin/health**
Scraper monitoring — spot broken scrapers before users notice missing data

Per-site status table:

Column	Type	Example	Source	Alert if
Site name + URL	Link	*SSC — ssc.gov.in*	sites_final.json id	—
Last scraped	Timestamp	*2 hours ago*	scraper_log table	> scrape_interval_hrs × 2
Last new PDF found	Timestamp	*3 days ago*	pdf_ingestion_log	> 30 days (possible gap)
Status	Enum badge	*OK / ERROR / BLOCKED*	scraper_log.status	ERROR or BLOCKED
Error message	Text	*HTTP 403 Forbidden*	scraper_log.error_message	Any non-null error
Success rate (7d)	%	*94%*	COUNT success/total	< 80%

ALERT RULE If any P0 or P1 site has status = ERROR for > 2 consecutive scrape cycles, send yourself an email (via Resend) immediately. P0 sites (SSC, UPSC, IBPS, SBI, RRB) must never go unmonitored for more than 4 hours. A broken scraper on SSC during CGL season = users miss a notification = trust destroyed.

Technical Implementation

Framework & Stack

- Built as a /admin/* section of the main Next.js 14 app — not a separate service
- Protected by Next.js middleware checking Supabase session role
- UI: shadcn/ui components + Tailwind — already in the main app stack
- Data: All queries via Supabase JS client or server-side fetch with service_role key
- No separate admin database — reads and writes to the same PostgreSQL database

The scraper_log Table (needed for Site Health section)

Add this table to the schema:

```
CREATE TABLE scraper_log (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  source_id VARCHAR(100) NOT NULL,  -- matches id in sites_final.json
  scraped_at TIMESTAMPTZ DEFAULT NOW(),
  status VARCHAR(20) NOT NULL,      -- OK | ERROR | BLOCKED | NO_NEW
  new_urls_found INTEGER DEFAULT 0,
  error_message TEXT,
  duration_ms INTEGER
);
```

Key Routes (Next.js App Router)

Route	Type	What it does
/admin (redirect)	Page — redirect	Redirects to /admin/queue
/admin/queue	Page (server)	Fetches pending records. Shows queue cards.
/admin/exams/new	Page (client)	Empty exam form. On submit: POST /api/admin/exams
/admin/exams/[id]	Page (server + client)	Fetches exam by id, renders pre-filled form
/admin/exams	Page (server)	Paginated exam list with search/filter
/admin/analytics	Page (server)	Runs analytics queries, renders read-only dashboard
/admin/health	Page (server)	Fetches scraper_log, renders site status table
/api/admin/exams (POST)	API Route	Creates new exam record. Validates all fields. Returns exam id.
/api/admin/exams/[id] (PATCH)	API Route	Updates exam record. Handles approve action → triggers notification job.
/api/admin/exams/[id] (DELETE)	API Route	Soft-deletes exam (sets is_deleted = TRUE). Hard delete not allowed.
/api/admin/approve/[id] (POST)	API Route	Sets notification_verified = TRUE, fires BullMQ new_exam_notification job.

Build Order Within the Dashboard

Build the admin dashboard in this exact order. Each step is usable before the next is built.

Order	What to build	When it is "done enough to use"
1st	**Auth middleware**	The /admin/* route is protected. You can reach it. Others cannot.
2nd	**Add Exam form (Groups 1–3)**	You can manually enter: exam name, dates, and total vacancies. Enough to create the first 10 records and test the dashboard → user eligibility → email notification loop end-to-end.
3rd	**Add Exam form (Groups 4–8)**	Full form complete. You can enter all eligibility data. Start manually adding the 50 central exams.
4th	**Exam List + basic search**	You can find and edit existing records. Duplicating annual exams becomes fast.
5th	**Queue view**	Once the scraper and PDF pipeline start sending records, you need the queue. Build it when you have the pipeline ready.
6th	**Analytics**	Build last — only meaningful once you have real users. A week 1 priority is exam data, not analytics.
7th	**Site Health**	Build when scraper is live. No point before.

THE SINGLE MOST IMPORTANT THING Before writing any scraper, pipeline, or notification code — build the Add Exam form (steps 1–3 above) and manually enter 5 real exam records. Then trigger a notification for a test user account. Verify the email arrives, the eligibility matching works, and the data looks right in the user dashboard. This end-to-end validation in week 1 will catch schema problems early, before you have 1,000 scraped records in the wrong format.