School ERP SaaS — Build Plan & Database Design

Nov 22, 2026 · @Jay

Overview

We are building a hosted (SaaS) school management system in Laravel and MySQL. Every school uses one shared app on our server. Schools start on a Basic plan (Students, Academic, Fees) and buy more modules as add-ons, which we switch on per school.

How this plan is organised: this tab is the overview, core design and timeline. The Students, Academic and Fees tabs hold the final detailed design of each Basic module; where a table or screen differs between this tab and a module tab, the module tab is correct. The Decision log at the end lists every choice made so far.

Decision	Choice	Why
Delivery model	SaaS, hosted on our server	No installation wizard, license system or piracy; one update reaches every school
Stack	Laravel 12, PHP 8.3, MySQL 8	Skills we already have; runs on cPanel or a small VPS
Tenancy	One database, `school_id` on every school table	Simple to host, back up and update
School address	`schoolname.ourapp.in` (wildcard subdomain)	Custom domains can come later
Basic plan	Students, Academic, Fees	What every school needs on day one
Add-ons	Attendance, Exams, Finance, HR, Library, Assets, Vehicles, Inventory, Website CMS, Alumni, Hostel, Reports	Switched on per school after payment
Branches	Single and multi-branch from day one	`branch_id` on branch-owned tables; adding it later means a rewrite
Access control	RBAC with custom roles, enforced on the server	Schools create their own roles with a module × action grid

Decision	Choice	Why
Admin levels	Platform Owner, School Super Admin, custom roles	Platform Owner and School Super Admin skip all permission checks in their own scope
Custom IDs	Format per school, per entity	For example `ABC-2026-0001` for students

Out of scope for now: installation wizard, license activation, module zip upload, themes (one admin template only).

Architecture

The app is one Laravel project where each feature area is a self-contained module, and every request is tied to one school by its subdomain.

Packages

Package	Purpose
`nwidart/laravel-modules`	Each module has its own routes, controllers, views, migrations and permissions
`spatie/laravel-permission` (teams mode, team = school)	Roles and permissions, with roles separate per school
`spatie/laravel-activitylog`	Activity log of who changed what
Tabler or AdminLTE (Bootstrap)	One admin template; no theme system

Folder structure

```
app/                    core: tenancy, auth, platform panel, RBAC, settings,
IDs
Modules/
  Students/             basic
  Academic/             basic
  Fees/                 basic
  Attendance/           add-on (later)
  Exams/                add-on (later)
  ...
    Config/permissions.php   actions shown in the role grid
    Config/menu.php          sidebar entries
    Database/Migrations/
    Http/Controllers/  Models/  Policies/  Resources/views/  Routes/web.php
```

How a request finds its school

1. Middleware reads the subdomain (`abc.ourapp.in`) and loads the school. Unknown or suspended school: stop with an error page.
2. The current school is stored for the request and set as the Spatie team, so roles and permissions load for that school only.
3. A `BelongsToSchool` trait on every school model adds `WHERE school_id = ?` to every query and fills `school_id` on create. No controller code sets `school_id` by hand.
4. A `BelongsToBranch` trait adds the user's branch filter based on their role's data scope (see Access control).

Turning modules on and off

- All module code is deployed once. A school's enabled modules come from `school_modules`.
- `EnsureModuleEnabled` middleware on each module's routes returns 403 when the school has not bought the module or it has expired.
- The sidebar and the role grid only show enabled modules.
- Core modules never depend on add-ons. Add-ons may depend on core modules (Exams needs Academic and Students).

Hosting

- Start on our cPanel hosting or a VPS at about ₹500–1,500 a month, enough for the first 20–50 schools.
- Wildcard DNS and a wildcard SSL certificate for `*.ourapp.in`.
- A separate staging site to test every update before production.
- Queue worker and scheduler (cron) for reminders, expiry checks and backups.

Access control

Every request passes five server-side checks, and the screen only shows what the server would allow; hiding a button is never the protection.

Admin levels

Level	Who	Can do	Limits
Platform Owner	Us	Everything across all schools: create schools, switch modules, plans, billing, "login as" for support	Separate login (`platform` guard), separate table
School Super Admin	The school owner	Everything inside their school: all branches, settings, roles, records	Cannot be deleted or demoted; at least one per school
Custom roles	Principal, accountant, teacher, clerk and so on	Only what the role grid allows, within the role's data scope	Cannot grant more than they have; cannot edit their own role

Permission names use `module.action`:

- Standard actions for every module: `view`, `create`, `edit`, `delete`, `export`, `print`
- Extra actions where needed: `students.promote`, `students.import`, `fees.collect`, `fees.cancel_receipt`, `fees.discount`, `fees.refund`, `roles.manage`, `settings.manage`
- Each module lists its actions in `Config/permissions.php`; a sync command writes them to the `permissions` table, so a new module appears in the role grid without code changes.

Creating a custom role

1. Enter the role name, description and data scope.
2. Tick a grid: modules down the side, actions across the top, with select-all per row and column. Only modules the school has enabled appear.
3. Or copy an existing role and adjust it.
4. System roles (School Super Admin) are locked.

Data scope (which records a role sees)

Scope	Sees	Typical role
`all_branches`	Every branch of the school	Principal, owner
`assigned_branches`	Branches listed in `branch_user` for that user	Branch admin, accountant
`own_records`	Records linked to the user (their sections, their child)	Teacher, parent, student

Where the checks live

1. Route middleware: `permission:students.create`.
2. Module middleware: the school owns the module and it has not expired.
3. Policies: `$this->authorize('update', $student)` checks this record, not just the action.
4. Global query scopes: `school_id` and branch filters on every query, including exports, searches and dropdowns. Changing an ID in the URL cannot open another branch's record.
5. Form requests: reject `school_id`, `branch_id` or `role_id` values the user is not allowed to set.

Super admins pass through `Gate::before`. Permissions are cached and cleared when a role changes. Denied attempts are written to the activity log.

Database conventions

Every table follows the same rules, so the tables below only list each table's own columns in full plus these standard ones.

- Engine InnoDB, charset `utf8mb4`, collation `utf8mb4_unicode_ci`.
- Table names plural `snake_case`; foreign keys `<singular>_id` with real foreign key constraints.
- Primary key `id` BIGINT UNSIGNED auto-increment.
- Money is `DECIMAL(12,2)`, never FLOAT.
- Timestamps are stored in UTC and shown in the school's timezone (default `Asia/Kolkata`).
- Status columns are VARCHAR(20) with allowed values checked in the model, so new values need no migration.
- Personal ID numbers (Aadhaar) are encrypted with Laravel's `encrypted` cast.

- Unique rules always include `school_id`, for example `UNIQUE(school_id, admission_no)`.
- Financial records (receipts, payments) are never deleted; they are cancelled with a reason.

Standard columns (marked "+ standard" in the tables below)

Column	Type	Null	Notes
`school_id`	BIGINT UNSIGNED, FK `schools`	No	On every school-owned table; indexed; filled by the `BelongsToSchool` trait
`branch_id`	BIGINT UNSIGNED, FK `branches`	Varies	Only on branch-owned tables; indexed
`created_by`	BIGINT UNSIGNED, FK `users`	Yes	Who created the row
`updated_by`	BIGINT UNSIGNED, FK `users`	Yes	Who last changed the row
`created_at`	TIMESTAMP	Yes	Laravel timestamps
`updated_at`	TIMESTAMP	Yes	Laravel timestamps
`deleted_at`	TIMESTAMP	Yes	Soft delete on business tables (not on logs or pivots)

Platform tables

These five tables belong to us, the Platform Owner, and have no `school_id` filter except where a column links to a school.

`platform_users`: our own admin logins, separate from school users.

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(100)	No	
`email`	VARCHAR(150)	No	Unique
`password`	VARCHAR(255)	No	Hashed

Column	Type	Null	Notes
`two_factor_secret`	TEXT	Yes	Encrypted; 2FA required for platform users
`is_active`	BOOLEAN	No	Default true
`last_login_at`	TIMESTAMP	Yes	
`last_login_ip`	VARCHAR(45)	Yes	IPv4 or IPv6
`remember_token`	VARCHAR(100)	Yes	
`created_at`, `updated_at`	TIMESTAMP	Yes	

`plans`: what we sell.

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(50)	No	Basic, Standard, Premium
`slug`	VARCHAR(50)	No	Unique
`price_monthly`	DECIMAL(12,2)	No	INR
`price_yearly`	DECIMAL(12,2)	No	INR
`max_students`	INT UNSIGNED	Yes	Null = unlimited
`max_branches`	INT UNSIGNED	Yes	Null = unlimited
`included_modules`	JSON	No	Module slugs, for example `["students","academic","fees"]`
`is_active`	BOOLEAN	No	Default true
`created_at`, `updated_at`	TIMESTAMP	Yes	

`modules`: every module the app has, core and add-on.

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	

Column	Type	Null	Notes
`slug`	VARCHAR(50)	No	Unique; matches the folder name in lower case, for example `attendance`
`name`	VARCHAR(100)	No	Shown in menus and the role grid
`description`	TEXT	Yes	Shown on the "buy add-on" page
`is_core`	BOOLEAN	No	True for Students, Academic, Fees
`price_monthly`	DECIMAL(12,2)	Yes	Add-on price; null for core
`depends_on`	JSON	Yes	Slugs it needs, for example `["students","academic"]`
`sort_order`	SMALLINT UNSIGNED	No	Menu and grid order
`is_active`	BOOLEAN	No	False hides it from every school
`created_at`, `updated_at`	TIMESTAMP	Yes	

`school_modules`: which school has which module.

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`school_id`	FK `schools`	No	
`module_id`	FK `modules`	No	Unique with `school_id`
`status`	VARCHAR(20)	No	`trial`, `active`, `expired`, `disabled`
`starts_at`	DATE	No	
`expires_at`	DATE	Yes	Null = no expiry (core modules); expired add-ons turn read-only, data kept
`activated_by`	FK `platform_users`	Yes	
`created_at`, `updated_at`	TIMESTAMP	Yes	

`school_subscriptions`: billing history.

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`school_id`	FK `schools`	No	Indexed
`plan_id`	FK `plans`	No	
`billing_cycle`	VARCHAR(20)	No	`monthly`, `yearly`
`amount`	DECIMAL(12,2)	No	Amount charged, after any discount
`starts_at`	DATE	No	
`ends_at`	DATE	No	
`status`	VARCHAR(20)	No	`pending`, `paid`, `failed`, `cancelled`
`payment_ref`	VARCHAR(100)	Yes	Razorpay or bank reference
`created_at`, `updated_at`	TIMESTAMP	Yes	

Core tables

These tables exist for every school regardless of plan. Spatie's team column is renamed to `school_id` in `config/permission.php` (`team_foreign_key`).

`schools`

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`uuid`	CHAR(36)	No	Unique; used in links instead of the numeric id
`name`	VARCHAR(150)	No	
`code`	VARCHAR(10)	No	Unique short code, used as the default ID prefix (`ABC`)
`subdomain`	VARCHAR(50)	No	Unique; `abc` in `abc.ourapp.in`

Column	Type	Null	Notes
`custom_domain`	VARCHAR(150)	Yes	Unique; later feature
`email`	VARCHAR(150)	Yes	
`phone`	VARCHAR(15)	Yes	
`address`	VARCHAR(255)	Yes	
`city`	VARCHAR(100)	Yes	
`state`	VARCHAR(100)	Yes	
`pincode`	VARCHAR(10)	Yes	
`logo_path`	VARCHAR(255)	Yes	Storage path
`plan_id`	FK `plans`	No	
`status`	VARCHAR(20)	No	`trial`, `active`, `suspended`, `cancelled`; suspended blocks login
`trial_ends_at`	DATE	Yes	
`timezone`	VARCHAR(50)	No	Default `Asia/Kolkata`
`currency`	CHAR(3)	No	Default `INR`
`is_multi_branch`	BOOLEAN	No	False hides branch menus and switcher
`created_at`, `updated_at`, `deleted_at`	TIMESTAMP	Yes	

`branches` (+ standard, no `branch_id`)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(150)	No	
`code`	VARCHAR(10)	No	Unique with `school_id`; usable in ID formats as `{BRANCH}`
`email`	VARCHAR(150)	Yes	
`phone`	VARCHAR(15)	Yes	

Column	Type	Null	Notes
`address`	VARCHAR(255)	Yes	
`city`	VARCHAR(100)	Yes	
`state`	VARCHAR(100)	Yes	
`pincode`	VARCHAR(10)	Yes	
`is_main`	BOOLEAN	No	Exactly one per school; created automatically with the school
`status`	VARCHAR(20)	No	`active`, `inactive`

`users`: everyone who logs in to a school (+ standard, no `branch_id`)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(100)	No	
`username`	VARCHAR(50)	No	Unique with `school_id`; login with username or email
`email`	VARCHAR(150)	Yes	Unique with `school_id`
`phone`	VARCHAR(15)	Yes	
`password`	VARCHAR(255)	No	Hashed
`user_type`	VARCHAR(20)	No	`staff`, `student`, `parent`
`is_super_admin`	BOOLEAN	No	School Super Admin; skips all permission checks in this school
`avatar_path`	VARCHAR(255)	Yes	
`status`	VARCHAR(20)	No	`active`, `inactive`, `locked`
`two_factor_secret`	TEXT	Yes	Encrypted
`two_factor_confirmed_at`	TIMESTAMP	Yes	
`last_login_at`	TIMESTAMP	Yes	
`last_login_ip`	VARCHAR(45)	Yes	
`remember_token`	VARCHAR(100)	Yes	

Module plans add two columns to `users`: `must_change_password` (BOOLEAN, Students: first login of parents and students) and `is_teaching_staff` (BOOLEAN, Academic: teacher dropdowns). They also add `custom_fields` and `custom_field_values` (Students), `academic_lists` (Academic), and new `id_formats` entities `refund`.

`branch_user`: which branches a user works in.

Column	Type	Null	Notes
`user_id`	FK `users`	No	Primary key with `branch_id`
`branch_id`	FK `branches`	No	
`is_default`	BOOLEAN	No	Branch selected after login

`roles`: Spatie table with extra columns.

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`school_id`	FK `schools`	Yes	Spatie team column; null only for platform templates copied into new schools
`name`	VARCHAR(100)	No	Unique with `school_id`, `guard_name`
`guard_name`	VARCHAR(50)	No	`web`
`description`	VARCHAR(255)	Yes	
`data_scope`	VARCHAR(30)	No	`all_branches`, `assigned_branches`, `own_records`
`is_system`	BOOLEAN	No	Locked roles cannot be edited or deleted
`created_by`	FK `users`	Yes	
`created_at`, `updated_at`	TIMESTAMP	Yes	

`permissions`: Spatie table, global (same list for every school).

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(100)	No	`students.create`; unique with `guard_name`
`guard_name`	VARCHAR(50)	No	`web`
`module_slug`	VARCHAR(50)	No	Grid row; links to `modules.slug`
`action`	VARCHAR(50)	No	Grid column: `view`, `create`, `edit`, `delete`, `export`, `print` or a special action
`label`	VARCHAR(100)	No	Shown in the grid, for example "Collect fees"
`sort_order`	SMALLINT UNSIGNED	No	
`created_at`, `updated_at`	TIMESTAMP	Yes	

Spatie pivots (unchanged apart from the team column): `role_has_permissions` (`permission_id`, `role_id`), `model_has_roles` (`role_id`, `model_type`, `model_id`, `school_id`), `model_has_permissions` (`permission_id`, `model_type`, `model_id`, `school_id`; kept for rare per-user extras).

`settings` (no soft delete)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`school_id`	FK `schools`	No	
`branch_id`	FK `branches`	Yes	Null = school-wide; a branch row overrides it
`group`	VARCHAR(50)	No	`general`, `academic`, `fees`, `notifications`
`key`	VARCHAR(100)	No	Unique with `school_id`, `branch_id`; for example `fees.receipt_footer`

Column	Type	Null	Notes
`value`	TEXT	Yes	JSON-encoded when not a plain string
`updated_by`	FK `users`	Yes	
`created_at`, `updated_at`	TIMESTAMP	Yes	

`id_formats`: custom ID rules.

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`school_id`	FK `schools`	No	
`branch_id`	FK `branches`	Yes	Null = shared counter for all branches
`entity`	VARCHAR(30)	No	`student`, `staff`, `admission`, `receipt`; unique with `school_id`, `branch_id`
`prefix`	VARCHAR(20)	No	`ABC`
`pattern`	VARCHAR(100)	No	Tokens `{PREFIX}`, `{BRANCH}`, `{YYYY}`, `{YY}`, `{SEQ}`; for example `{PREFIX}-{YYYY}-{SEQ}`
`seq_padding`	TINYINT UNSIGNED	No	4 gives `0001`
`next_number`	INT UNSIGNED	No	Read and increased inside a transaction with a row lock, so two clerks never get the same number
`reset_policy`	VARCHAR(20)	No	`never`, `yearly`
`last_reset_year`	SMALLINT UNSIGNED	Yes	
`created_at`, `updated_at`	TIMESTAMP	Yes	

`activity_logs`: from `spatie/laravel-activitylog`, with school and branch added. Never edited or deleted from the app.

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`school_id`	FK `schools`	Yes	Null for platform-level actions; index with `created_at`
`branch_id`	FK `branches`	Yes	
`log_name`	VARCHAR(50)	Yes	Module slug
`description`	VARCHAR(255)	No	`created`, `updated`, `deleted`, `denied`, `login`
`subject_type`, `subject_id`	VARCHAR(255), BIGINT UNSIGNED	Yes	The record that changed
`causer_type`, `causer_id`	VARCHAR(255), BIGINT UNSIGNED	Yes	A user or a platform user
`properties`	JSON	Yes	Old and new values
`ip_address`	VARCHAR(45)	Yes	
`user_agent`	VARCHAR(255)	Yes	
`created_at`, `updated_at`	TIMESTAMP	Yes	

Academic module

Academic holds the structure that students and fees hang on: years, classes, sections and subjects. Classes and sections stay the same each year; which student sits in which class each year is stored in `student_enrollments` (Students module).

Final design, replaces this section where they differ (streams, electives, teachers, timetable, copy to next year): Academic module plan

`academic_years` (+ standard, no `branch_id`)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(20)	No	`2026-27`; unique with `school_id`
`start_date`	DATE	No	

Column	Type	Null	Notes
`end_date`	DATE	No	After `start_date`
`is_current`	BOOLEAN	No	Exactly one per school; the default in every screen
`status`	VARCHAR(20)	No	`upcoming`, `active`, `closed`; closed years are read-only

`classes` (+ standard)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(50)	No	`Nursery`, `Class 1`; unique with `school_id`, `branch_id`
`code`	VARCHAR(10)	Yes	`C1`
`numeric_order`	SMALLINT	No	Sort order and promotion order (Class 1 then Class 2)
`next_class_id`	FK `classes`	Yes	Target class for promotion; null for the final class
`status`	VARCHAR(20)	No	`active`, `inactive`

`sections` (+ standard)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`class_id`	FK `classes`	No	
`name`	VARCHAR(20)	No	`A`, `B`; unique with `class_id`
`capacity`	SMALLINT UNSIGNED	Yes	Warn on admission when full
`class_teacher_id`	FK `users`	Yes	Used by the `own_records` data scope
`room_no`	VARCHAR(20)	Yes	
`status`	VARCHAR(20)	No	`active`, `inactive`

`subjects` (+ standard, no `branch_id`)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(100)	No	`Mathematics`; unique with `school_id`
`code`	VARCHAR(20)	Yes	`MATH`
`type`	VARCHAR(20)	No	`theory`, `practical`, `both`
`status`	VARCHAR(20)	No	`active`, `inactive`

`class_subjects`: which subjects each class studies in a year, and who teaches them (+ standard).

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`academic_year_id`	FK `academic_years`	No	
`class_id`	FK `classes`	No	
`section_id`	FK `sections`	Yes	Null = all sections of the class
`subject_id`	FK `subjects`	No	Unique with year, class, section
`teacher_id`	FK `users`	Yes	Used by the `own_records` data scope

Column	Type	Null	Notes
`is_optional`	BOOLEAN	No	Optional subjects are picked per student later (Exams add-on)

Students module

A student's personal details live in `students` once; their class for each year lives in `student_enrollments`, which keeps a full promotion history.

Final design, replaces this section where they differ (flows, form fields, logins, custom fields, import): Students module plan

`students` (+ standard)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`student_code`	VARCHAR(30)	No	Custom ID from `id_formats`; unique with `school_id`
`admission_no`	VARCHAR(30)	No	Unique with `school_id`; school's own register number, may also be generated
`admission_date`	DATE	No	
`first_name`	VARCHAR(100)	No	
`last_name`	VARCHAR(100)	Yes	
`gender`	VARCHAR(10)	No	`male`, `female`, `other`
`date_of_birth`	DATE	No	
`blood_group`	VARCHAR(5)	Yes	
`religion`	VARCHAR(50)	Yes	
`category`	VARCHAR(20)	Yes	`general`, `obc`, `sc`, `st`, `ews`
`mother_tongue`	VARCHAR(50)	Yes	
`nationality`	VARCHAR(50)	No	Default `Indian`
`aadhaar_no`	TEXT	Yes	Encrypted; shown masked (`XXXX-XXXX-1234`)

Column	Type	Null	Notes
`photo_path`	VARCHAR(255)	Yes	
`phone`	VARCHAR(15)	Yes	Student's own number, if any
`email`	VARCHAR(150)	Yes	
`address`	VARCHAR(255)	Yes	
`city`	VARCHAR(100)	Yes	
`state`	VARCHAR(100)	Yes	
`pincode`	VARCHAR(10)	Yes	
`previous_school`	VARCHAR(150)	Yes	
`tc_no`	VARCHAR(50)	Yes	Transfer certificate received on admission
`user_id`	FK `users`	Yes	Student login, if enabled
`status`	VARCHAR(20)	No	`active`, `left`, `transferred`, `passed_out`, `suspended`
`left_date`	DATE	Yes	
`left_reason`	VARCHAR(255)	Yes	

`guardians` (+ standard, no `branch_id`): one row per parent; siblings share the same guardian rows.

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(100)	No	
`phone`	VARCHAR(15)	No	Main contact for WhatsApp and SMS; indexed for sibling search
`alternate_phone`	VARCHAR(15)	Yes	
`email`	VARCHAR(150)	Yes	
`occupation`	VARCHAR(100)	Yes	
`annual_income`	DECIMAL(12,2)	Yes	Some schools need it for concessions

Column	Type	Null	Notes
`address`	VARCHAR(255)	Yes	
`user_id`	FK `users`	Yes	Parent login, if enabled

`guardian_student`

Column	Type	Null	Notes
`guardian_id`	FK `guardians`	No	Primary key with `student_id`
`student_id`	FK `students`	No	
`relation`	VARCHAR(20)	No	`father`, `mother`, `guardian`
`is_primary`	BOOLEAN	No	Receives fee reminders; one per student

`student_enrollments` (+ standard)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`student_id`	FK `students`	No	Unique with `academic_year_id`
`academic_year_id`	FK `academic_years`	No	
`class_id`	FK `classes`	No	
`section_id`	FK `sections`	No	
`roll_no`	VARCHAR(10)	Yes	Unique with year and section
`status`	VARCHAR(20)	No	`studying`, `promoted`, `detained`, `left`
`previous_enrollment_id`	FK `student_enrollments`	Yes	Links the promotion chain

`student_documents` (+ standard)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`student_id`	FK `students`	No	
`type`	VARCHAR(30)	No	`birth_certificate`, `tc`, `aadhaar`, `photo`, `other`
`title`	VARCHAR(150)	No	
`file_path`	VARCHAR(255)	No	Private storage; served only through a permission-checked route
`file_size`	INT UNSIGNED	No	Bytes

Fees module

Fees work in three steps: define a fee structure per class, generate a due line per student per period (`student_fees`), then record receipts that pay off those lines (`fee_payments` and `fee_payment_items`). Every unpaid list, reminder and report reads from `student_fees` .

Final design, replaces this section where they differ (installments, concessions, fines, Razorpay, refunds, carry forward): Fees module plan

`fee_heads` (+ standard, no `branch_id`)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(100)	No	`Tuition`, `Admission`, `Transport`, `Books` ; unique with `school_id`
`code`	VARCHAR(20)	Yes	
`type`	VARCHAR(20)	No	`recurring`, `one_time`
`is_refundable`	BOOLEAN	No	For example a caution deposit
`status`	VARCHAR(20)	No	`active`, `inactive`

`fee_structures` (+ standard)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`academic_year_id`	FK `academic_years`	No	
`class_id`	FK `classes`	No	Unique with year and branch
`name`	VARCHAR(100)	No	`Class 1 fees 2026-27`
`status`	VARCHAR(20)	No	`draft`, `published`; only published structures generate dues

`fee_structure_items`

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`fee_structure_id`	FK `fee_structures`	No	
`fee_head_id`	FK `fee_heads`	No	Unique with `fee_structure_id`
`amount`	DECIMAL(12,2)	No	Per period
`frequency`	VARCHAR(20)	No	`monthly`, `quarterly`, `half_yearly`, `yearly`, `one_time`
`due_day`	TINYINT UNSIGNED	No	Day of month the period falls due, for example 10
`created_at`, `updated_at`	TIMESTAMP	Yes	

`fee_discounts`: concession types (+ standard, no `branch_id`).

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(100)	No	`Sibling`, `Staff ward`, `Merit`
`type`	VARCHAR(10)	No	`percent`, `fixed`
`value`	DECIMAL(12,2)	No	10 = 10% or ₹10

Column	Type	Null	Notes
`fee_head_id`	FK `fee_heads`	Yes	Null = applies to all heads
`status`	VARCHAR(20)	No	`active`, `inactive`

`student_discounts` (+ standard)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`student_id`	FK `students`	No	
`academic_year_id`	FK `academic_years`	No	
`fee_discount_id`	FK `fee_discounts`	No	Unique with student and year
`approved_by`	FK `users`	No	Needs `fees.discount` permission
`remarks`	VARCHAR(255)	Yes	

`fine_rules` (+ standard, no `branch_id`)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(100)	No	
`type`	VARCHAR(20)	No	`per_day`, `fixed`
`amount`	DECIMAL(12,2)	No	
`grace_days`	TINYINT UNSIGNED	No	Days after the due date before a fine starts
`max_amount`	DECIMAL(12,2)	Yes	Cap per due line
`status`	VARCHAR(20)	No	`active`, `inactive`

`student_fees`: one due line per student, head and period (+ standard).

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`student_id`	FK `students`	No	
`academic_year_id`	FK `academic_years`	No	

Column	Type	Null	Notes
`fee_head_id`	FK `fee_heads`	No	
`fee_structure_item_id`	FK `fee_structure_items`	Yes	Null for manual charges
`period_label`	VARCHAR(30)	No	`Jun 2026`, `Q1`, `Annual`; unique with student, head, year
`due_date`	DATE	No	
`amount`	DECIMAL(12,2)	No	Before discount
`discount_amount`	DECIMAL(12,2)	No	Default 0
`fine_amount`	DECIMAL(12,2)	No	Default 0; updated by the daily fine job
`paid_amount`	DECIMAL(12,2)	No	Default 0; sum of active payment items
`balance`	DECIMAL(12,2)	No	Generated column: amount − discount + fine − paid
`status`	VARCHAR(20)	No	`unpaid`, `partial`, `paid`, `waived`; index with `school_id`, `due_date`

`fee_payments`: one row per receipt (+ standard).

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`receipt_no`	VARCHAR(30)	No	Custom ID from `id_formats`; unique with `school_id`
`student_id`	FK `students`	No	
`payment_date`	DATE	No	
`amount`	DECIMAL(12,2)	No	Total of its items
`mode`	VARCHAR(20)	No	`cash`, `upi`, `card`, `bank_transfer`, `cheque`, `online`
`reference_no`	VARCHAR(100)	Yes	UPI or bank reference, cheque number

Column	Type	Null	Notes
`cheque_date`	DATE	Yes	
`bank_name`	VARCHAR(100)	Yes	
`remarks`	VARCHAR(255)	Yes	
`collected_by`	FK `users`	No	
`status`	VARCHAR(20)	No	`success`, `cancelled`; never deleted
`cancelled_by`	FK `users`	Yes	Needs `fees.cancel_receipt`
`cancelled_at`	TIMESTAMP	Yes	
`cancel_reason`	VARCHAR(255)	Yes	Required when cancelling

`fee_payment_items`: how one receipt is split across due lines.

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`fee_payment_id`	FK `fee_payments`	No	
`student_fee_id`	FK `student_fees`	No	
`amount`	DECIMAL(12,2)	No	Includes any fine paid
`created_at`, `updated_at`	TIMESTAMP	Yes	

`fee_reminders` (+ standard)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`student_id`	FK `students`	No	
`guardian_id`	FK `guardians`	No	
`channel`	VARCHAR(20)	No	`whatsapp_link` , `sms` , `email`
`message`	TEXT	No	Final text sent
`amount_due`	DECIMAL(12,2)	No	Balance at the time of sending
`status`	VARCHAR(20)	No	`sent` , `failed`
`sent_at`	TIMESTAMP	No	

Screens and pages

This table is the first-release screen list for Platform and Core. The module tabs list their own screens in full: Students 7 sub-menu items, Academic 10, Fees 12, plus the parent and student portal. Mockups of the main screens are on the design canvas.

Area	Screen	Main permission
Platform	Login with 2FA	Platform guard
Platform	Dashboard: schools, active modules, renewals due	Platform guard
Platform	Schools: list, create (with subdomain and main branch), edit, suspend	Platform guard
Platform	School detail: switch modules on or off, set expiry and trial	Platform guard
Platform	Plans and module prices	Platform guard
Platform	Subscriptions and payments	Platform guard
Platform	"Login as" school super admin, with a banner and return button	Platform guard
Platform	Platform users	Platform guard
Core	Login, forgot password, 2FA	Public

Area	Screen	Main permission
Core	Dashboard (widgets per module)	Any logged-in user
Core	Branch switcher in the top bar	`branch_user` or `all_branches` scope
Core	Branches: list, create, edit	`branches.*`
Core	Users: list, create, edit, reset password, assign roles and branches	`users.*`
Core	Roles: list, create or copy, permission grid, data scope	`roles.manage`
Core	Settings: general, academic, fees, notifications	`settings.manage`
Core	ID formats with a live preview	`settings.manage`
Core	Activity log with filters	`activity_logs.view`
Core	My profile and change password	Any logged-in user
Core	Module store: enabled and available add-ons	School super admin
Core	403 and "module not enabled" pages	—
Academic	Academic years, set current year	`academic.*`
Academic	Classes, with promotion order	`academic.*`
Academic	Sections and class teachers	`academic.*`
Academic	Subjects	`academic.*`
Academic	Class subjects and teachers	`academic.*`
Academic	Year close (locks the old year)	`academic.edit`
Academic	Academic setup wizard for a new school	School super admin
Students	Student list with filters by class, section and status	`students.view`
Students	Admission form: student, guardians (with sibling search), enrollment, documents	`students.create`
Students	Student profile: details, guardians, class history, documents, fees tab	`students.view`
Students	Edit student	`students.edit`

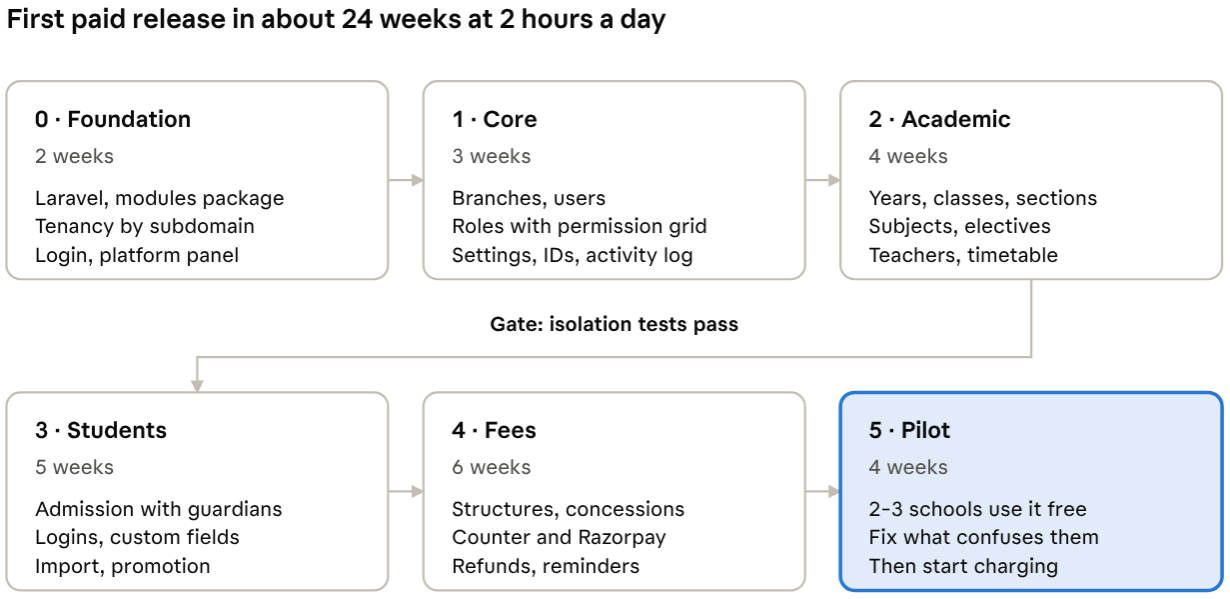
Area	Screen	Main permission
Students	Bulk import from Excel, with an error report	`students.import`
Students	Promotion: pick class, mark promote or detain, move to next year	`students.promote`
Students	Mark left or transferred	`students.edit`
Fees	Fee heads	`fees.*`
Fees	Fee structures per class and year, publish	`fees.*`
Fees	Generate dues for a class or year	`fees.create`
Fees	Discounts and student concessions	`fees.discount`
Fees	Fine rules	`fees.edit`
Fees	Collect fees: search student, tick dues, choose mode, print receipt	`fees.collect`
Fees	Receipts list, reprint, cancel with reason	`fees.view`, `fees.cancel_receipt`
Fees	Unpaid list with one-click WhatsApp reminders	`fees.view`
Fees	Reports: daily collection, mode-wise, head-wise, class-wise dues	`fees.export`

Build phases

With everything decided in the module tabs, the Basic plan is ready for a free pilot in about 20 weeks and for paying schools in about 24 weeks, working 2 hours a day alongside the job.

Phase	Weeks	Main work	Detail
0 Foundation	2	Laravel, modules package, tenancy by subdomain, login, Platform panel	This tab
1 Core	3	Branches, users, roles with permission grid, settings, ID formats, activity log	This tab
2 Academic	4	Years, classes, streams, sections, subjects, electives, teachers, timetable, copy to next year	Academic tab

Phase	Weeks	Main work	Detail
3 Students	5	Admission, guardians and siblings, custom fields, logins and portal, import, promotion	Students tab
4 Fees	6	Structures and installments, concessions, fines, counter, Razorpay, credit, refunds, reports	Fees tab
5 Pilot	4	2–3 schools use it free; fix what confuses them; then start charging	—
Total	**24**		



build phases · about 24 weeks at 2 hours a day

Students work does not start until tests prove that one school, branch or role cannot reach another's data. After the paid launch, add-ons follow in this order: Attendance, Exams with report cards, Finance, HR and Payroll, then Library, Hostel, Vehicles, Inventory, Assets, Alumni, Website CMS and Reports.

Security, testing and backups

The one failure we cannot afford is one school seeing another school's data, so tenant isolation gets automated tests before any feature ships.

Security

- [ ] Every school model uses `BelongsToSchool`; a test fails the build if one does not
- [ ] Every module route has `auth`, module and permission middleware
- [ ] Policies on every record-level action (view, edit, delete one record)

- [ ] Form requests never accept `school_id`; `branch_id` only from the user's allowed branches
- [ ] HTTPS only, secure and HttpOnly cookies, session cookie scoped per subdomain
- [ ] Login rate limit (5 tries a minute) and account lock after repeated failures
- [ ] 2FA required for platform users, optional for school users
- [ ] Uploaded files in private storage, type and size checked, served through a permission-checked route
- [ ] Aadhaar and 2FA secrets encrypted; `APP_KEY` backed up separately
- [ ] Activity log on create, edit, delete, login, denied access and receipt cancellation

Testing

- [ ] Tenant test: a user of school A gets 404 for every school B record URL
- [ ] Permission test: each route returns 403 without its permission
- [ ] Data scope test: a branch user cannot list, export or open another branch's records
- [ ] Module test: a disabled module's routes return 403
- [ ] Fees tests: balance, partial payment, fine, discount and receipt cancellation totals
- [ ] ID generator test: 50 parallel requests produce 50 unique numbers

Backups and releases

- [ ] Daily database backup at night, kept 30 days, copied to a different provider
- [ ] Weekly backup of uploaded files
- [ ] Restore tested once a month on the staging site
- [ ] Every release goes to staging first, then production outside school hours
- [ ] Uptime monitor and error tracking (Sentry free plan or Laravel log alerts)

Decision log

Every choice made while planning, in the order it was made.

#	Topic	Decision
1	Business	School management software for schools and coaching centres in our own district first
2	Stack	Laravel and MySQL
3	Plans	Basic plan (Students, Academic, Fees); all other modules are paid add-ons

#	Topic	Decision
4	Delivery	SaaS hosted on our servers; no installation wizard, license system or module zip upload for now
5	Add-ons	All module code is deployed once; add-ons are switched on per school
6	Super admin	Platform Owner (us) and School Super Admin can do everything in their scope
7	RBAC	Custom roles per school with a module × action grid; enforced on the server on every request
8	Branches	Single and multi-branch from day one
9	Admission	Direct admission by staff
10	Logins	Parent and student logins on admission, in a separate portal
11	Custom fields	Schools add their own fields to students, guardians, classes, sections and subjects
12	Students extras	Bulk import and sibling linking in version 1
13	Printing	ID cards and certificates become a separate Certificate Management add-on
14	Promotion	Checkbox select-all and bulk promote, detain or leave
15	Streams	Classes 11 and 12 have streams
16	Academic extras	Copy to next year, elective subjects per student, class and subject teachers, timetable
17	Quick setup	Board templates, bulk create, inline editing, paste lists, grids for subjects
18	Online payment	Razorpay in the parent portal, into each school's own Razorpay account
19	Installments	Each fee structure picks monthly, quarterly, term-wise, annual or custom
20	Fees extras	Late fines, concessions and scholarships, refunds and advance credit, carry forward of unpaid dues

Students module plan

Nov 22, 2026 · @Jay

Decisions and scope

Version 1 of Students covers direct admission by staff, parent and student logins, school-defined custom fields, bulk import from Excel and sibling linking.

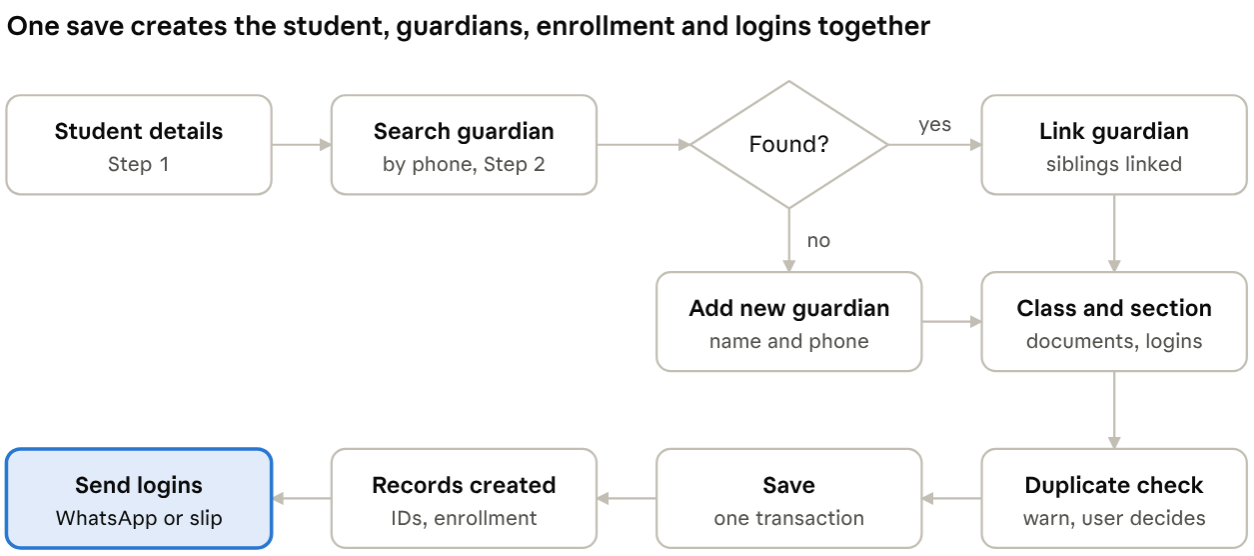
Topic	Decision
Admission	Direct admission by staff through one form
Logins	Parent and student accounts are created on admission
Custom fields	Each school adds its own fields to the admission form, list, import and profile
Bulk import	Excel upload with a preview and an error report before anything is saved
Siblings	Students share guardian records; finding a guardian by phone links siblings automatically
Printing	ID cards, transfer certificates and bonafide certificates go to a separate **Certificate Management** add-on

Not in version 1

- Enquiries and follow-ups (possible later add-on: Admissions CRM)
- Online application form for parents
- ID cards and certificates (Certificate Management add-on)
- Attendance, marks and timetable views in the parent app (they arrive with their own add-ons)

Admission flow

Staff fill one form; a single save creates the student, links or creates guardians, enrolls the student in a class and creates logins.



admission flow · one form, one save

If anything fails during the save, everything is rolled back and the form keeps what was typed.

Admission form

The form has five steps on one page, saved together in a single database transaction, so a failed save leaves nothing half-created. Custom fields appear inside whichever step the school assigns them to.

Step 1: Student details

Field	Required	Validation and notes
First name	Yes	2–100 letters, spaces, dots
Last name	No	Up to 100
Gender	Yes	Male, female, other
Date of birth	Yes	In the past; warning if age is unusual for the chosen class
Blood group	No	Fixed list (A+, A−, B+, …)
Religion	No	List from settings
Category	No	General, OBC, SC, ST, EWS
Mother tongue	No	

Field	Required	Validation and notes
Nationality	Yes	Default Indian
Aadhaar number	No (school setting can make it required)	12 digits with checksum; unique in the school; stored encrypted, shown masked
Photo	No	JPG or PNG, up to 1 MB, cropped square in the browser
Phone, email	No	Student's own; 10-digit Indian mobile, valid email
Address, city, state, pincode	No (setting)	Pincode 6 digits; "Same as guardian" copies the guardian's address
Previous school, TC number	No	

Step 2: Guardians (rules in Guardians and siblings)

Field	Required	Validation and notes
Search by phone	—	First box in the step; finds an existing guardian in this school
Father: name, phone, email, occupation, income	At least one guardian with name and phone	10-digit mobile
Mother: same fields		
Other guardian: same fields plus relation		Used when neither parent is the contact
Primary contact	Yes	Exactly one; gets reminders and the parent login

Step 3: Admission and class

Field	Required	Validation and notes
Branch	Yes	Only branches the user may access; hidden for single-branch schools
Academic year	Yes	Default current year; closed years not allowed

Field	Required	Validation and notes
Class, section	Yes	Warning when the section is at capacity
Admission date	Yes	Not in the future
Admission number	Yes	Auto-generated from `id_formats`; editable only if the school setting allows; unique
Student ID	—	Auto-generated on save, shown as a preview
Roll number	No	Default next free number in the section; unique per section and year

Step 4: Documents

Field	Required	Validation and notes
Type, title, file	No	PDF, JPG, PNG, up to 2 MB each; several allowed

Step 5: Logins

Field	Required	Validation and notes
Create student login	—	Default from school setting
Create parent login	—	Default on; skipped when the primary guardian already has one
Send login details by	—	SMS, WhatsApp link or print slip

Before saving, the system warns about a possible duplicate when a student with the same name, date of birth and guardian phone already exists. The user can cancel or continue.

Guardians and siblings

Siblings are not linked to each other directly; they share the same guardian records, so one parent login and one phone number cover all their children.

Rules

1. The guardian step starts with a phone search. A match shows the guardian's name and their current children, and the user clicks "Use this guardian".

2. A phone number is unique per school among guardians. Saving a new guardian with an existing number is blocked, with a link to use the existing one.
3. Searching works across all branches of the school, so siblings in different branches still share a guardian.
4. Editing a shared guardian (new phone, new address) updates it for every linked child. The form says "Linked to 2 students" before saving.
5. Each student has exactly one primary contact. Siblings may have different primary contacts.
6. A student's profile shows a Siblings panel: all other active students who share any guardian.
7. Guardians from an existing student can be linked later from the profile ("Link sibling"), for students admitted before the sibling existed.
8. Removing a guardian from a student never deletes the guardian if other students still use it.

Edge cases

Case	Handling
Two parents share one phone	Store it on one guardian; the other gets no phone or an alternate number
Parents separated, different contacts per child	Different primary contacts per student
Guardian changes phone number	Edit once; all linked students and the parent login use the new number
Same phone, different family (wrong entry)	Staff can unlink and create a new guardian; the change is logged
Sibling discount in Fees	Reads siblings from shared guardians, so no separate sibling table is needed

Parent and student logins

Parents and students log in on the same school address but land in a simple, mobile-first portal (`/portal`), not the staff admin panel.

	Parent	Student
Account created	On admission, for the primary guardian, if they have no account yet	On admission, if the school setting or the form says so
Username	Guardian's mobile number	Student ID (`ABC-2026-0001`)
First password	Random 8 characters; must change on first login	Same
Role	System role **Parent**, data scope `own_records`	System role **Student**, data scope `own_records`
Sees in version 1	Each child's profile, class and section, guardians, fee dues, receipts	Own profile, class and section, fee dues
Several children	One login; a child switcher at the top	—
Can edit	Own password and phone (later: request profile corrections)	Own password
Disabled when	None of their children is active	The student leaves, transfers or passes out

Rules

- Parent and Student are locked system roles. Their permissions are a fixed `portal.*` set; schools cannot give them staff permissions.
- Every portal query is filtered to the logged-in parent's own children or the student themself, enforced by the data scope and policies, not by hidden links.
- Staff with `students.manage_logins` can reset a password, disable or enable a login and resend login details.
- Login details are sent by WhatsApp link (free), printed slip, or SMS once an SMS gateway is connected.
- Imported students get logins in bulk from the import screen or later from the student list.
- Later add-ons (Attendance, Exams, Timetable) add their own pages to the portal.

Core table change: add `must_change_password` (BOOLEAN, default false) to `users`. Parent users link through `guardians.user_id`; student users through `students.user_id`.

Custom fields

Schools add their own fields (House, Bus stop, Caste certificate number) from Settings without any code change. Definitions go in `custom_fields`, and each student's answers go in `custom_field_values`. The same design will serve Staff and other modules later.

Field types: short text, long text, number, date, dropdown, multi-select, yes/no.

Rules

- Up to 30 custom fields per entity per school.
- A field with saved values can only be deactivated, not deleted. Its type cannot change once values exist.
- Dropdown options are stored as fixed keys with labels, so renaming a label does not break saved answers.
- Each field chooses where it appears: which form step, the student list and filters, the import template, and the parent portal.
- Custom values are validated server-side by the same rules as built-in fields.

`custom_fields` (+ standard, no `branch_id`)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`entity`	VARCHAR(30)	No	`student`, `guardian`; later `staff`
`field_key`	VARCHAR(50)	No	Slug, for example `house`; unique with `school_id`, `entity`; used as the import column header
`label`	VARCHAR(100)	No	Shown on the form
`type`	VARCHAR(20)	No	`text`, `textarea`, `number`, `date`, `dropdown`, `multi_select`, `boolean`
`options`	JSON	Yes	Dropdown choices: `[{"key":"red","label":"Red House"}]`
`is_required`	BOOLEAN	No	
`validation`	JSON	Yes	`min`, `max`, `max_length`, `pattern`
`default_value`	VARCHAR(255)	Yes	
`placeholder`	VARCHAR(100)	Yes	

Column	Type	Null	Notes
`help_text`	VARCHAR(255)	Yes	
`form_step`	VARCHAR(30)	No	`student_details`, `guardians`, `admission`
`show_in_list`	BOOLEAN	No	Column in the student list
`show_in_filters`	BOOLEAN	No	Dropdown and yes/no fields only
`show_in_import`	BOOLEAN	No	Column in the Excel template
`show_in_portal`	BOOLEAN	No	Visible to parents and students
`sort_order`	SMALLINT UNSIGNED	No	Order within the step
`is_active`	BOOLEAN	No	

`custom_field_values`

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`school_id`	FK `schools`	No	
`custom_field_id`	FK `custom_fields`	No	
`entity_id`	BIGINT UNSIGNED	No	The student or guardian id; unique with `custom_field_id`
`value`	TEXT	Yes	Stored as text; multi-select as a JSON array of keys
`created_at`, `updated_at`	TIMESTAMP	Yes	

Index `(custom_field_id, value(50))` so filters on dropdown fields stay fast.

Bulk import from Excel

Import is a two-pass process: every row is checked and shown in a preview first, and nothing is saved until the user confirms. Large files run in the background queue so the browser never times out.

1. **Download template.** Built for this school: the standard columns plus custom fields marked "show in import", one row per student, with father, mother and guardian columns side by side. A second sheet lists valid classes, sections and dropdown values.
2. **Upload.** XLSX or CSV, up to 5 MB and 2,000 rows. Choose branch and academic year once for the whole file.
3. **Validate.** Each row is checked with the same rules as the admission form: required fields, formats, class and section names exist, admission number not taken, Aadhaar unique, duplicates inside the file.
4. **Match guardians.** A guardian phone that already exists in the school, or repeats in the file, links to the same guardian, so siblings are linked automatically.
5. **Preview.** Counts of valid and invalid rows; invalid rows show each error next to its cell. The user can download an error file, fix it and upload again.
6. **Import.** Valid rows are saved in batches of 100, each batch in its own transaction. Student IDs, admission numbers (when blank) and roll numbers are generated.
7. **Result.** Imported count, failed rows as a downloadable file, and a button to create parent and student logins for this batch.

Package: `maatwebsite/excel` (or `openspout` for lower memory on shared hosting).

`student_imports` (+ standard)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`academic_year_id`	FK `academic_years`	No	
`original_name`	VARCHAR(255)	No	
`file_path`	VARCHAR(255)	No	Private storage; deleted after 30 days
`status`	VARCHAR(20)	No	`uploaded`, `validating`, `validated`, `importing`, `completed`, `failed`
`total_rows`	INT UNSIGNED	No	
`valid_rows`	INT UNSIGNED	No	
`error_rows`	INT UNSIGNED	No	

Column	Type	Null	Notes
`imported_rows`	INT UNSIGNED	No	
`create_logins`	BOOLEAN	No	
`error_file_path`	VARCHAR(255)	Yes	
`started_at`, `finished_at`	TIMESTAMP	Yes	

`student_import_rows`

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`student_import_id`	FK `student_imports`	No	Indexed with `status`
`row_number`	INT UNSIGNED	No	Row in the uploaded sheet
`data`	JSON	No	The row as read
`errors`	JSON	Yes	`{"column": "message"}`
`status`	VARCHAR(20)	No	`valid`, `invalid`, `imported`, `failed`
`student_id`	FK `students`	Yes	Set once imported
`created_at`, `updated_at`	TIMESTAMP	Yes	

List, profile, status and promotion

Student list

- Filters: branch, academic year, class, section, status, gender, category, and custom fields marked as filters.
- Search: name, student ID, admission number, guardian phone, last 4 digits of Aadhaar.
- Columns: user picks which to show, including custom fields; the choice is remembered per user.
- 25, 50 or 100 per page, sorted by class, section, then roll number.

- Export to Excel or CSV with the current filters, only for users with `students.export`, and always within their data scope.
- Bulk actions on ticked rows: change section, create logins, export.

Student profile tabs: Overview, Guardians and siblings, Class history, Documents, Fees (when the Fees module is on and the user has `fees.view`), Login, Activity log.

Status changes

New status	Set from	What happens
Left	Profile, with date and reason	Current enrollment marked `left`; logins disabled; unpaid dues stay, future dues are cancelled
Transferred	Profile, with date, reason and destination school	Same as Left; the TC itself comes from the Certificate Management add-on
Suspended	Profile, with reason	Stays in class; logins disabled until reactivated
Passed out	Promotion of the final class	Logins disabled; record kept for alumni
Active again (re-admission)	Profile	New enrollment in the chosen year and class; same student ID and guardians

Changing section during the year edits the current enrollment and is written to the activity log.

Promotion (end of year)

1. Choose the current year, the next year, a class and section.
2. Every student is listed with "Promote" ticked; the user can switch any to "Detain" or "Leave".
3. Choose the target section in the next class (one for all, or per student).
4. Confirm: each student gets a new enrollment in the next year; the old one is marked `promoted`, `detained` or `left`. Students in the final class become `passed_out`.
5. Roll numbers in the new year can be assigned by name order or kept blank.
6. A promotion batch can be reverted until fee dues are generated for the new year.

The next year must exist and not be closed. Promotion never deletes or changes anything in the old year.

Permissions

The Students module adds 13 permissions to the role grid, plus 2 fixed portal permissions for the Parent and Student roles.

Permission	Allows
`students.view`	Student list and profiles within the role's data scope
`students.create`	Admission form
`students.edit`	Edit student and guardian details, change section
`students.delete`	Soft-delete a wrongly entered student; blocked if any fee payment exists
`students.export`	Export lists to Excel or CSV
`students.import`	Bulk import
`students.promote`	Promotion and reverting a promotion batch
`students.change_status`	Left, transferred, suspended, re-admission
`students.manage_logins`	Create, reset, disable and resend parent and student logins
`students.manage_documents`	Upload and delete documents
`students.view_sensitive`	See full Aadhaar number and guardian income; others see them masked
`students.link_guardians`	Link or unlink guardians and siblings after admission
`custom_fields.manage`	Add and edit custom fields (Settings)
`portal.view_profile`	Parent and Student roles only
`portal.view_fees`	Parent and Student roles only; needs the Fees module

Build order and estimate

With logins, the portal, custom fields and import added, Students grows from 2 weeks to about 5 weeks at 2 hours a day. That moves the first paying school from about 15 to about 18 weeks.

1. **Week 1:** migrations, models and traits for students, guardians, enrollments and documents; guardian phone search and linking.
2. **Week 2:** admission form, student list with filters, profile page.
3. **Week 3:** custom fields (settings screen, form, list, profile), status changes, promotion.
4. **Week 4:** parent and student logins, the portal (profile and switcher), login management screens.
5. **Week 5:** bulk import with preview and error file; tests for scope, permissions, promotion and import.

Fees comes after this, so the portal's fees page is switched on in the Fees phase.

Academic module plan

Nov 22, 2026 · @Jay

Decisions and scope

Version 1 of Academic covers years, classes, streams, sections, subjects, electives, class and subject teachers, a weekly timetable, and copying the whole setup into the next year.

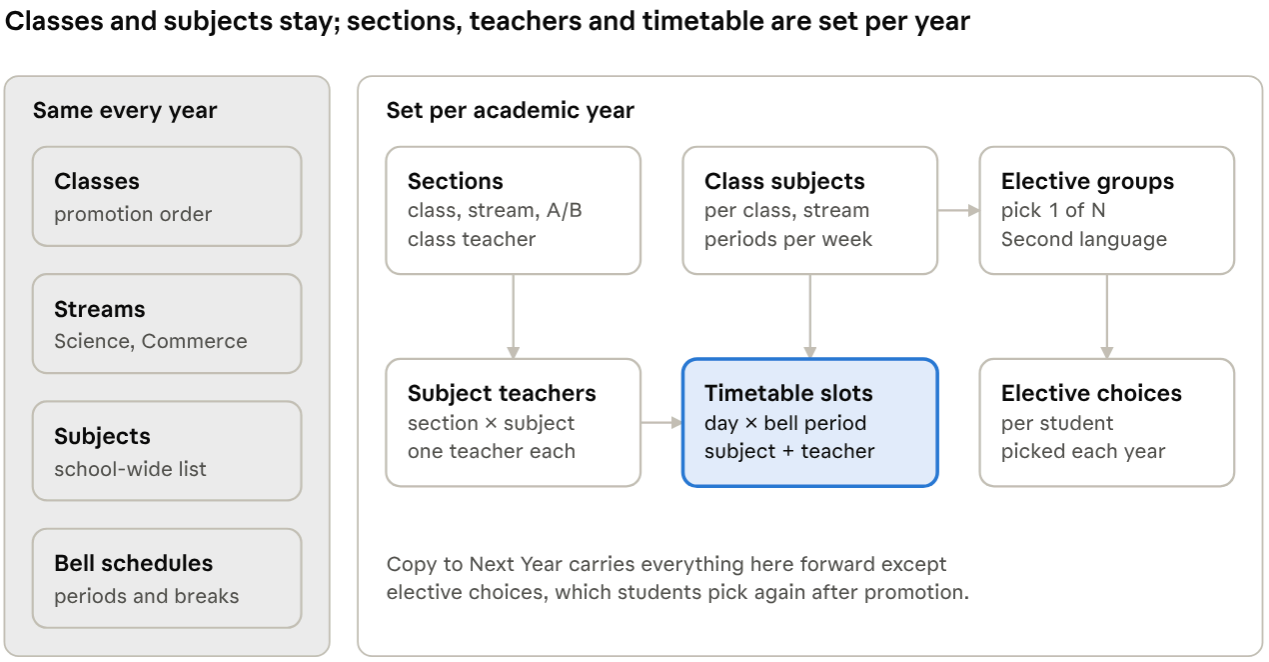
Topic	Decision
Streams	Classes 11 and 12 can have streams (Science, Commerce, Arts, or MPC, BiPC, MEC, CEC); each school names its own
Sections	Belong to one academic year, so each year can have different sections, capacities and class teachers
Classes	Stay the same across years (Class 1 to Class 12) and define the promotion order
Electives	Subjects are grouped ("Second language: pick 1"); each student's choices are stored per year
Teachers	A class teacher per section and a teacher per subject per section; teachers are staff users marked as teaching staff
Timetable	A weekly grid per section built from bell schedules, with teacher clash checks
Copy to next year	One screen copies sections, subjects, electives, teachers and the timetable; bell schedules are shared across years

Change to the main design: `sections` gain `academic_year_id` and `stream_id`, and `class_subjects` gain stream, elective and timetable columns (full list under Tables).

Not in version 1: automatic timetable generation, substitution management for absent teachers, lesson plans and syllabus tracking.

How the parts fit

Classes, streams, subjects and bell schedules are set up once; everything that changes year to year hangs off the academic year and ends in the timetable.



academic structure · what stays and what is set each year

Sections and class subjects together decide who teaches what; the timetable places those teacher-subject pairs into bell periods for each day.

Academic years and copy to next year

A new year starts empty; one copy screen fills it from the previous year in a single step, and it must run before students are promoted, because promotion needs the new year's sections.

Academic year rules

- Name like `2027-28`, start and end dates, no overlap with another year of the same school.
- Status moves `upcoming` → `active` → `closed`. Exactly one year is current; switching the current year is a Super Admin action with a confirmation.
- A closed year is read-only everywhere: no admissions, edits, timetable changes or fee dues.

Copy setup to next year

1. Choose the source year and the target year (the target must be `upcoming` or `active`).
2. Tick what to copy. Each item lists what it depends on:

Item	Copies	Needs
Sections	Names, streams, capacities, rooms	—
Class teachers	Class teacher per section	Sections
Class subjects	Subjects per class and stream, periods per week	—
Elective groups	Groups and their rules	Class subjects
Subject teachers	Teacher per subject per section	Sections, class subjects
Bell schedules	Not copied: shared by all years	—
Timetable	Every slot of every section	Sections, class subjects (uses the shared bell schedules)

3. Preview: counts per item, and anything that will be skipped (already exists in the target, or the teacher is now inactive).
4. Copy runs in one transaction and is recorded in `academic_copy_runs`. Running it twice only adds what is missing; it never overwrites.

Student elective choices are not copied: they are picked again after promotion, because subjects often change between classes.

Classes, streams and sections

A student's stream comes from their section, so "Class 11 – Science – A" needs no extra column on the student or the enrollment.

Classes

- Created once per branch: Nursery, LKG, UKG, Class 1 … Class 12. Order sets the promotion path (`next_class_id`).
- A class marked `has_streams` requires every section to pick a stream.
- A class can be deactivated but never deleted once any enrollment uses it.

Streams

- School-wide list with name and short code: Science (SCI), Commerce (COM), Arts (ART), or MPC, BiPC, MEC, CEC for intermediate-style schools.
- Used by sections, class subjects and fee structures (Fees can set different fees per stream later).

Sections

- Belong to one academic year, one class and, for stream classes, one stream.
- Name unique within year, class and stream: Class 11 can have "Science – A", "Science – B" and "Commerce – A".
- Capacity, room and class teacher per section; admission warns when the section is full.
- Display name built automatically: `Class 11 - SCI - A`.

Promotion with streams: students moving from Class 10 into Class 11 pick a target section per student, which also sets their stream. Students moving from 11 to 12 default to a section of the same stream.

Subjects and electives

Subjects are a school-wide list; class subjects say which subjects a class (and stream) studies in a given year; elective groups say where students choose.

Subjects: name, code, type (theory, practical, both). Example: Mathematics, Physics, Hindi, Sanskrit, Computer Science.

Class subjects (per year, class and stream)

- Core subjects apply to every student in the class.
- Elective subjects belong to an elective group.
- Each has a target number of periods per week, which the timetable editor counts against.

Elective groups

Example group	Class	Subjects in group	Pick
Second language	Class 6	Hindi, Telugu, Sanskrit	Exactly 1
Additional subject	Class 11 Science	Computer Science, Biology, Physical Education	Exactly 1
Skill subject	Class 9	IT, Retail, Tourism	0 or 1

Choosing electives

1. Open a section, see every student with one dropdown per elective group.
2. Pick for each student, or apply one choice to all ticked students.
3. Saving checks each group's minimum and maximum.
4. Students with a missing choice are flagged on the section screen, the student profile and the promotion screen.

Choices are stored per enrollment in `student_subjects`, so a student's history shows what they studied each year. The Exams add-on will only create marks entries for subjects a student actually takes.

Customisation and quick setup

Schools name and describe classes, sections and subjects their own way, and can create a whole school's structure in minutes instead of one record at a time.

What each school can customise

Item	Customisable fields
Classes	Any name (Nursery, PP1, Grade 1, Class I), short name, order, wing (Pre-primary, Primary, Middle, Secondary, Senior Secondary, or the school's own), has streams, default bell schedule, custom fields
Sections	Any name (A/B, 1/2, or names like Rose and Lily), medium (English, Telugu, Hindi), shift (Morning, Afternoon), gender (Co-ed, Boys, Girls), capacity, room, class teacher, custom fields
Subjects	Name, short name for timetable cells (up to 6 letters), code, category (Language, Core, Elective, Co-scholastic, Activity, or the school's own), theory or practical, colour, custom fields

Wings, mediums, shifts and subject categories are lists the school edits in Settings, so nothing is hard-coded. The same custom fields system used for students also works for classes, sections and subjects.

Ways to create quickly

1. **Board template (first setup):** pick CBSE, ICSE, Andhra Pradesh State, Telangana State or Blank. The template fills classes, wings and common subjects per class; the school reviews and edits everything before saving.

2. **Bulk add classes:** a range such as 1 to 10 with a naming pattern: `Class {n}`, `Grade {n}` or Roman numerals (`Class I`).
3. **Bulk add sections:** tick classes, choose names (A–D, 1–3, or a custom list), medium, shift and capacity; a preview shows the count, for example "30 sections in 10 classes".
4. **Inline editing:** every list is an editable table: type in a new row at the bottom, edit cells in place, drag rows to reorder, press Enter to save.
5. **Paste a list:** paste subject names one per line to create them all at once.
6. **Assign subjects by grid:** classes across, subjects down, tick boxes to assign; or copy one class's subjects to other classes.
7. **Duplicate:** a section, or a class with its subjects and elective groups.
8. **Excel import** for classes, sections and subjects, with the same preview and error file as student import.

Safe changes: renaming never breaks anything because records link by id. Anything already used by students, fees or the timetable can be deactivated but not deleted.

Table additions

Table	Column	Type	Notes
`academic_lists` (new)	`id`, `school_id`, `type`, `name`, `sort_order`, `is_active`, + standard	`type` VARCHAR(30)	`type` = `wing`, `medium`, `shift`, `subject_category`; unique `school_id`, `type`, `name`
`classes`	`short_name`	VARCHAR(10)	`C1`, `X`
`classes`	`wing_id`	FK `academic_lists`	
`sections`	`medium_id`	FK `academic_lists`, null	
`sections`	`shift_id`	FK `academic_lists`, null	
`sections`	`gender`	VARCHAR(10)	`coed`, `boys`, `girls`; default `coed`
`subjects`	`short_name`	VARCHAR(6)	Shown in timetable cells

Table	Column	Type	Notes
`subjects`	`category_id`	FK `academic_lists`, null	
`subjects`	`color`	CHAR(7)	Hex colour for timetable and reports
`academic_templates` (new, platform)	`id`, `name`, `board`, `data` (JSON), `is_active`, timestamps		Maintained by us; copied into a school on first setup
`custom_fields`	`entity`	—	Adds `class`, `section`, `subject`

Class teachers and subject teachers

Teacher assignments drive both the timetable and the `own_records` data scope: a teacher sees the students of the sections they are class teacher of or teach a subject in.

- **Who is a teacher:** any staff user with `is_teaching_staff` ticked. Until the HR add-on exists, staff are created from Users.
- **Class teacher:** one per section per year (`sections.class_teacher_id`). One teacher may be class teacher of only one section in a year; the screen warns otherwise.
- **Subject teacher:** one teacher per subject per section (`subject_teachers`). A teacher can teach the same subject in many sections, or several subjects.
- **Assign screen:** a grid per class, sections across the top and subjects down the side, a teacher dropdown in each cell, and each teacher's weekly period total shown as you assign.
- **Teacher view:** "My classes" lists sections and subjects; "My timetable" shows their week.
- **Leaving teacher:** deactivating a user lists their assignments and timetable slots so they can be reassigned first.

Change to the main design: `class_subjects` no longer hold `section_id` or `teacher_id`; subject teachers move to their own table so one class subject can have a different teacher in each section.

Timetable

The timetable is a weekly grid per section (days across, periods down) built by hand, with the system blocking teacher clashes and counting periods against each subject's weekly target.

Bell schedules (period sets)

- A school defines one or more period sets, for example "Regular" (Monday to Friday) and "Saturday half day".
- Each set lists its periods in order: name, start time, end time, and whether it is a break (Assembly, Lunch).
- Working days come from settings; each set says which weekdays it covers.
- Sections use the school's default set unless a class picks another (for example, primary classes ending earlier).

Building a section's timetable

1. Open the section; the grid shows its days and periods, breaks greyed out.
2. Click a cell, pick a subject; the teacher fills in from subject teachers and can be changed.
3. A side panel shows each subject's periods placed against its target, for example "Mathematics 5 of 6".
4. Copy a day to another day, or clear a day.

Checks on save

Check	Result
Teacher already teaching another section at an overlapping time	Blocked, showing the other section
Room already used at an overlapping time (when rooms are entered)	Warning
Subject over its weekly target	Warning
Elective group periods	All subjects in a group share the same slot; each teacher takes their own students

Views and output

- Section timetable, teacher timetable, and a "free teachers" view for any day and period (useful for arranging cover).
- Print and PDF per section or per teacher.

- Parents and students see their section's timetable in the portal; teachers see their own week.

Tables

These replace the Academic section of the main plan where they differ. "+ standard" means the standard columns from the main plan. MySQL treats NULLs as different in unique keys, so tables with an optional `stream_id` add a generated column `stream_key` = `IFNULL(stream_id, 0)` and use it in the unique key.

Changes to existing tables

Table	Column	Type	Notes
`classes`	`has_streams`	BOOLEAN	Sections of this class must have a stream
`classes`	`period_set_id`	FK `period_sets`, null	Default bell schedule for the class
`users`	`is_teaching_staff`	BOOLEAN	Shows the user in teacher dropdowns

`streams` (+ standard, no `branch_id`)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(50)	No	Science, Commerce, MPC; unique with `school_id`
`code`	VARCHAR(10)	No	SCI, COM, MPC; used in section names
`sort_order`	SMALLINT UNSIGNED	No	
`status`	VARCHAR(20)	No	`active`, `inactive`

`sections` (+ standard; replaces the earlier version)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`academic_year_id`	FK `academic_years`	No	
`class_id`	FK `classes`	No	
`stream_id`	FK `streams`	Yes	Required when the class has streams
`name`	VARCHAR(20)	No	A, B; unique with year, class, `stream_key`
`capacity`	SMALLINT UNSIGNED	Yes	
`room_no`	VARCHAR(20)	Yes	Default room for the timetable
`class_teacher_id`	FK `users`	Yes	
`period_set_id`	FK `period_sets`	Yes	Overrides the class default
`status`	VARCHAR(20)	No	`active`, `inactive`

`class_subjects` (+ standard; replaces the earlier version)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`academic_year_id`	FK `academic_years`	No	
`class_id`	FK `classes`	No	
`stream_id`	FK `streams`	Yes	Null = all streams of the class
`subject_id`	FK `subjects`	No	Unique with year, class, `stream_key`
`elective_group_id`	FK `elective_groups`	Yes	Null = core subject
`periods_per_week`	TINYINT UNSIGNED	No	Target for the timetable; 0 = not timetabled
`sort_order`	SMALLINT UNSIGNED	No	Order on report cards and screens

`elective_groups` (+ standard)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`academic_year_id`	FK `academic_years`	No	
`class_id`	FK `classes`	No	
`stream_id`	FK `streams`	Yes	
`name`	VARCHAR(100)	No	Second language
`min_select`	TINYINT UNSIGNED	No	0 = optional group
`max_select`	TINYINT UNSIGNED	No	At least `min_select` and at least 1
`sort_order`	SMALLINT UNSIGNED	No	

`student_subjects`: elective choices only; core subjects are implied.

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`school_id`	FK `schools`	No	
`student_enrollment_id`	FK `student_enrollments`	No	Unique with `class_subject_id`
`class_subject_id`	FK `class_subjects`	No	Must belong to an elective group of the enrollment's class and stream
`created_by`	FK `users`	Yes	
`created_at`, `updated_at`	TIMESTAMP	Yes	

`subject_teachers`

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`school_id`	FK `schools`	No	
`section_id`	FK `sections`	No	Unique with `class_subject_id`

Column	Type	Null	Notes
`class_subject_id`	FK `class_subjects`	No	
`teacher_id`	FK `users`	No	Must be active teaching staff; indexed
`created_by`, `updated_by`	FK `users`	Yes	
`created_at`, `updated_at`	TIMESTAMP	Yes	

`period_sets` (+ standard)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(50)	No	Regular, Saturday half day
`weekdays`	JSON	No	`[1,2,3,4,5]`, 1 = Monday
`is_default`	BOOLEAN	No	One default per branch
`status`	VARCHAR(20)	No	`active`, `inactive`

`periods`

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`period_set_id`	FK `period_sets`	No	
`name`	VARCHAR(30)	No	Period 1, Lunch
`start_time`	TIME	No	
`end_time`	TIME	No	After `start_time`; no overlap within the set
`is_break`	BOOLEAN	No	Breaks cannot hold subjects
`sort_order`	SMALLINT UNSIGNED	No	

Column	Type	Null	Notes
`created_at`, `updated_at`	TIMESTAMP	Yes	

`timetable_slots`

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`school_id`	FK `schools`	No	
`academic_year_id`	FK `academic_years`	No	
`section_id`	FK `sections`	No	Unique with `weekday`, `period_id`
`weekday`	TINYINT UNSIGNED	No	1 = Monday … 7 = Sunday
`period_id`	FK `periods`	No	
`class_subject_id`	FK `class_subjects`	Yes	Exactly one of this and `elective_group_id`
`elective_group_id`	FK `elective_groups`	Yes	Whole group shares the slot; teachers come from `subject_teachers`
`teacher_id`	FK `users`	Yes	For a single subject; index with `weekday` for clash checks
`room_no`	VARCHAR(20)	Yes	Default from the section
`created_by`, `updated_by`	FK `users`	Yes	
`created_at`, `updated_at`	TIMESTAMP	Yes	

`academic_copy_runs`

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`school_id`	FK `schools`	No	

Column	Type	Null	Notes
`from_year_id`	FK `academic_years`	No	
`to_year_id`	FK `academic_years`	No	
`items`	JSON	No	What was ticked
`result`	JSON	No	Counts copied and skipped, with reasons
`run_by`	FK `users`	No	
`created_at`	TIMESTAMP	No	

Screens and permissions

The Academic menu gets 10 sub-menu items; Timetable sits inside Academic rather than as its own menu.

Sub-menu	What it does	Permission
Academic Years	List, create, set current, close	`academic.manage_years`
Copy to Next Year	The copy wizard with preview	`academic.copy_setup`
Classes & Streams	Classes in promotion order, has-streams switch, stream list	`academic.manage_structure`
Sections	Per year: sections by class and stream, capacity, room, class teacher	`academic.manage_structure`
Subjects	School-wide subject list	`academic.manage_subjects`
Class Subjects	Per class and stream: core and elective subjects, periods per week, elective groups	`academic.manage_subjects`
Teacher Assignment	Grid of sections × subjects with teacher dropdowns and period totals	`academic.assign_teachers`
Elective Choices	Per section: students × elective groups, bulk apply	`academic.manage_electives`
Bell Schedules	Period sets and periods	`timetable.edit`

Sub-menu	What it does	Permission
Timetable	Section grid editor, teacher view, free teachers, print	`timetable.view`, `timetable.edit`, `timetable.print`

`academic.view` lets a user see all of the above without changing anything. Teachers see "My classes" and "My timetable" through their `own_records` scope. Parents and students get `portal.view_timetable`.

Build order and estimate

With streams, electives, teachers and the timetable added, Academic grows from 1 week to about 4 weeks at 2 hours a day. With the Students increase, the first paying school moves from about 18 to about 21 weeks.

1. **Week 1:** years, classes, streams, sections (per year), subjects, class subjects.
2. **Week 2:** elective groups and choices, teaching staff flag, class and subject teacher assignment, copy to next year.
3. **Week 3:** bell schedules, timetable grid editor with clash checks and period counts.
4. **Week 4:** teacher timetable, free teachers view, print and PDF, portal timetable; tests for clashes, copy and electives.

Academic must be built before Students, because admission needs years, classes and sections. The order in the main plan stays: Core, Academic, Students, Fees.

Fees module plan

Nov 22, 2026 · @Jay

Decisions and scope

Version 1 of Fees covers fee structures with installments the school chooses, concessions, late fines, counter and online payment, advance payments, refunds, and carrying unpaid dues into the next year.

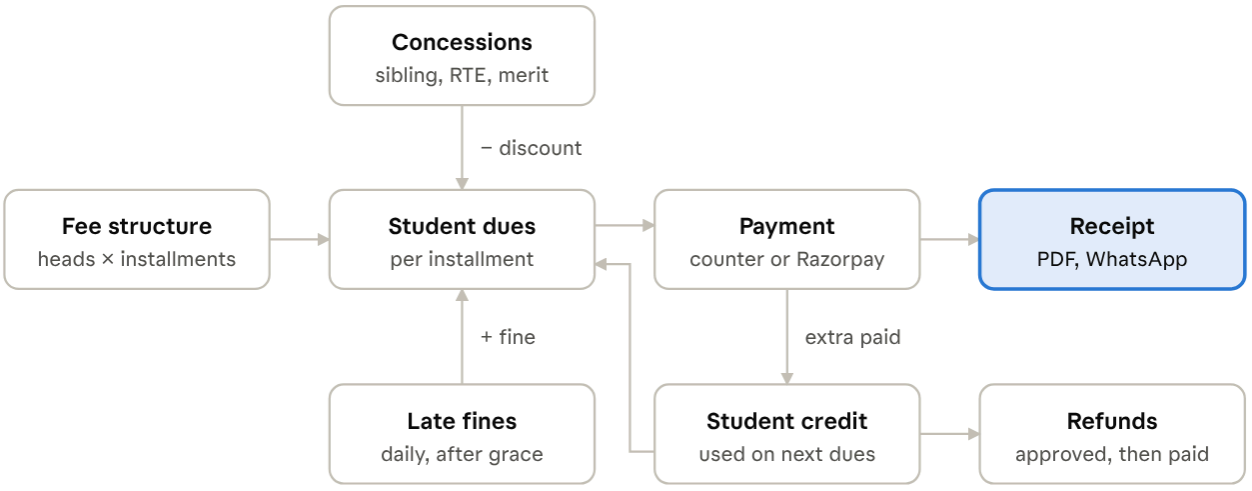
Topic	Decision
Installments	Each fee structure picks its plan: monthly, quarterly, term-wise, annual or custom dates and amounts
Online payment	Parents pay in the portal through Razorpay (UPI, cards, netbanking); the receipt is created automatically
Whose Razorpay account	Each school connects its own Razorpay account, so money goes straight to the school and we never hold it
Concessions	Sibling, staff ward, merit, RTE and scholarships, as percent or fixed amount, with approval
Late fines	Per day, fixed or slab, with grace days and a cap; waiving needs a permission
Advance and refunds	Extra money becomes student credit used against future dues; refunds for deposits and credit
Carry forward	Unpaid balance from last year becomes a "Previous year dues" line in the new year
Money rules	Receipts and refunds are never deleted, only cancelled with a reason; every change is in the activity log

Not in version 1: automatic SMS or WhatsApp sending through a paid provider (version 1 uses one-click WhatsApp links and email), Tally export, GST invoices, payment links for non-portal parents, transport fee by route (comes with the Vehicles add-on).

How money flows

A structure creates dues; concessions lower them and fines raise them; a payment from the counter or Razorpay pays them off and creates a receipt.

Every rupee moves from dues to a receipt; extra becomes credit



fee money flow · structure to receipt

Any amount above the dues goes into the student's credit, which pays future dues automatically or is refunded after approval. Unpaid dues at year end move into the next year as "Previous year dues".

Fee structures and installments

A fee structure is a grid: fee heads down the side, installments across the top, an amount in each cell. One structure is made per class (and stream) per year, and can be copied to other classes.

Fee heads are the school's own list: Tuition, Admission, Annual charges, Books, Lab, Computer, Caution deposit. Each head has:

- Type: recurring (split across installments) or one-time (charged once).
- Refundable or not (a caution deposit is refundable).
- Priority, which decides the order a payment pays off heads when it doesn't cover everything.
- Whether concessions may apply to it.

Installment plans

Plan	Installments	Example due dates
Monthly	12	10th of every month, June to May
Quarterly	4	10 Jun, 10 Sep, 10 Dec, 10 Mar
Term-wise	3	10 Jun, 10 Oct, 10 Jan
Annual	1	10 Jun
Custom	Any number	School types names, dates and amounts

Building a structure

1. Pick year, class and stream (or "all streams").
2. Pick a plan; installment columns and due dates fill in and stay editable.
3. Enter each head's yearly total; recurring heads split equally across installments, and any cell can be overridden.
4. One-time heads go into the installment the school picks (usually the first).
5. Totals per installment and per year show live.
6. Publish. Only published structures create dues; editing a published structure shows how many students it affects and changes only their unpaid future installments.

Copy structure to other classes or to next year, with an optional percentage increase (for example +8%) rounded to the nearest ₹10.

Dues and carry forward

Dues are the student's bill: one `student_fees` line per head per installment. Every unpaid list, reminder, report and the parent portal reads from these lines.

When dues are created

Event	What happens
Structure published	Dues for every active student in that class and stream, after a preview with counts
New admission	Dues from the class structure, in the same save as the admission
Section or stream change	Unpaid future lines are replaced from the new structure; paid lines stay

Event	What happens
Student leaves	Future unpaid lines are cancelled; past unpaid lines stay as dues
Manual charge	Staff add a one-off line (lost ID card, trip) with a reason

Mid-year admission: a setting decides whether installments with a due date before the admission date are charged (default: charge from the current installment; one-time heads always charged).

Approved concessions are applied as each line is created, so the line already shows the discount.

Carry forward to the new year

1. Runs from Fees when the old year closes, or automatically on promotion.
2. Each student's total unpaid balance from the old year becomes one line in the new year under the system head "Previous year dues", due immediately.
3. Old lines are marked `carried` and link to the new line, so the old year's reports still balance.
4. Students who left keep their dues in the old year and appear in the "Former students with dues" report.
5. Paying the carried line counts as collection in the new year, and the receipt shows it came from the previous year.

Concessions and scholarships

A concession lowers what the school charges; a scholarship paid by someone else (a government scheme or a trust) is recorded as a payment from that payer, so the student's bill stays honest.

Concession types (school's own list): name, category (Sibling, Staff ward, Merit, RTE, Need-based, Other), percent or fixed amount, which heads it applies to, whether it needs approval, and whether a document is required.

Giving a concession

1. From the student profile or the Concessions screen, pick the type and the year; optionally override the value.
2. Types that need approval go to `pending`; a user with `fees.approve_concession` approves or rejects with a remark.

3. When approved, the discount is applied to unpaid lines from that point on. Paid lines are never changed.
4. Sibling concessions can be suggested automatically for students who share a guardian (from the Students module).

RTE students: a 100% concession on chosen heads, with the RTE category so the RTE report lists them.

Scholarships paid by others: the payment mode `scholarship` records the payer name and reference. Partial scholarships leave the rest for the parent to pay.

Late fines

Fines are calculated by a daily job, shown as a separate amount on each overdue line, and can be waived by users with permission.

Rule type	How it works	Example
Per day	Amount × days late after grace days	₹10 a day after 5 grace days
Fixed	One amount once the grace days pass	₹100 once overdue
Slab	Different amounts by days late	₹50 for 1–15 days, ₹150 for 16–30, ₹300 after

- A rule can apply to all heads or chosen heads, and to all classes or chosen classes.
- Each rule has a maximum fine per line.
- The fine stops growing once the line is paid; a partial payment does not stop it.
- Waiving a fine (fully or partly) needs `fees.waive_fine` and a reason, and is logged.
- The receipt shows the fine separately from the fee.

Collecting at the counter

The counter screen is built for speed: find the student, tick or type an amount, choose the mode, print or share the receipt, usually in under a minute.

1. **Find the student** by name, student ID, admission number or guardian phone. Siblings appear together, so a parent can pay for all children in one visit (one receipt per student).

2. **Dues list:** every unpaid line, oldest first, with fee, concession, fine and balance. Overdue lines are highlighted.
3. **Choose what to pay:** tick lines, or type an amount and let the system allocate it, oldest due first and by head priority. Any amount above the dues becomes student credit.
4. **Use credit:** if the student has credit, it is offered first.
5. **Mode:** cash, UPI, card, bank transfer, cheque, scholarship. Cheque needs number, date and bank; UPI and transfer need a reference.
6. **Save:** creates the receipt (number from ID formats) and its allocation lines in one transaction.
7. **Receipt:** print (A4 or A5), PDF, or share on WhatsApp; it shows heads, installments, concession, fine, mode and balance left.

Cheque bounce: marking a cheque bounced cancels its receipt, puts the lines back to unpaid, and can add a bounce charge.

Cancelling a receipt needs `fees.cancel_receipt` and a reason. The receipt stays with status `cancelled`, allocations are reversed, and the day's collection report shows the cancellation.

Day close: each collector sees their own total by mode for the day, to hand over cash.

Online payment with Razorpay

Parents pay from the portal; the money goes into the school's own Razorpay account and a receipt is created only after Razorpay confirms the payment on our server.

Setup (per school): the school enters its Razorpay key ID, key secret and webhook secret in Settings (stored encrypted), and chooses whether to absorb the gateway charges or add a convenience fee shown to the parent before paying.

Payment flow

1. Parent opens Fees in the portal, sees each child's dues and ticks what to pay (or "Pay all due").
2. Our server re-calculates the amount (never trusting the browser), creates a Razorpay order and an `online_payments` row with status `created`.
3. Razorpay Checkout opens: UPI, cards, netbanking, wallets.
4. On success, the browser sends the payment id and signature; the server verifies the signature.
5. Razorpay's webhook (`payment.captured`) also arrives; whichever comes first creates the receipt, and the other is ignored, so a payment never makes two receipts.

6. The parent sees the receipt at once and can download it; the school sees it in the day's collection under mode "Online".

Safety checks

- If dues changed between ordering and paying (for example staff collected at the counter meanwhile), the extra becomes student credit rather than being lost.
- A job every 15 minutes asks Razorpay about orders still `created` after 30 minutes and settles or fails them.
- An online payments screen lists every order with its Razorpay ids for matching against settlements.
- Refunds of online payments can be started from our screen and are sent to Razorpay.

Advance payments, credit and refunds

Each student has a credit balance kept as a ledger: every rupee in or out of credit is one row, so the balance can always be explained line by line.

Credit comes from: paying more than the dues, paying ahead for future installments, an online payment whose dues were already paid, or a cancelled-and-reissued receipt.

Credit is used: automatically when new dues are created (a school setting, default on), or offered first at the counter and in the portal.

Refunds

Refund of	Example	Rules
Refundable head	Caution deposit when the student leaves	Only up to what was paid on that head
Student credit	Advance left after the student leaves	Only up to the credit balance
Online payment	Wrong amount paid in the portal	Sent back through Razorpay to the same card or UPI

1. Staff with `fees.refund` create a refund request with amount, reason and mode (cash, bank transfer, Razorpay).
2. A user with `fees.approve_refund` approves it; the same person cannot do both.
3. Paying it out creates a refund voucher with its own number from ID formats, and updates the head or the credit ledger.
4. Refunds appear in the day's collection as money out.

Reminders and reports

Reminders

- Unpaid list filtered by class, section, installment, days overdue or amount.
- One-click WhatsApp per guardian with the amount, due date and a portal payment link filled in; or tick many and open them one after another.
- Email reminders in bulk; automatic reminders before and after the due date by email.
- Automatic SMS or WhatsApp through a paid provider can be added later; the settings screen reserves the place for it.
- Every reminder is stored in `fee_reminders`, so staff can see who was reminded and when.

Reports (all filtered by the user's data scope; export to Excel or PDF with `fees.export`)

Report	Shows
Day collection	Every receipt and refund of a day, by collector and by mode, with cancellations
Collection summary	Totals by date range, mode, head, class
Dues and defaulters	Who owes what, by class and installment, with days overdue
Student ledger	Every due, payment, concession, fine, credit and refund for one student
Concessions	Concessions given by type and class, including RTE
Fines	Fines charged, collected and waived
Online payments	Razorpay orders and their status, for matching settlements
Refunds	Refund requests, approvals and payouts
Carry forward	Balances moved from last year and what has been collected

Tables

These replace the Fees section of the main plan. "+ standard" means the standard columns from the main plan. All money columns are `DECIMAL(12,2)`. Razorpay keys live in `settings` (encrypted): `fees.razorpay_key_id`, `fees.razorpay_key_secret`,

`fees.razorpay_webhook_secret`, `fees.convenience_fee_mode`. `id_formats` gains the entity `refund`.

`fee_heads` (+ standard, no `branch_id`)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(100)	No	Unique with `school_id`
`code`	VARCHAR(20)	Yes	
`type`	VARCHAR(20)	No	`recurring`, `one_time`
`is_refundable`	BOOLEAN	No	
`priority`	SMALLINT UNSIGNED	No	Lower is paid first when allocating
`allow_concession`	BOOLEAN	No	
`is_system`	BOOLEAN	No	"Previous year dues"; cannot be edited or deleted
`status`	VARCHAR(20)	No	`active`, `inactive`

`fee_structures` (+ standard)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`academic_year_id`	FK `academic_years`	No	
`class_id`	FK `classes`	No	
`stream_id`	FK `streams`	Yes	Unique with year, class, `stream_key`, branch
`name`	VARCHAR(100)	No	
`plan_type`	VARCHAR(20)	No	`monthly`, `quarterly`, `term`, `annual`, `custom`
`status`	VARCHAR(20)	No	`draft`, `published`
`published_at`	TIMESTAMP	Yes	

Column	Type	Null	Notes
`published_by`	FK `users`	Yes	

`fee_installments`

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`fee_structure_id`	FK `fee_structures`	No	
`name`	VARCHAR(50)	No	June, Term 1
`due_date`	DATE	No	
`sort_order`	SMALLINT UNSIGNED	No	
`created_at`, `updated_at`	TIMESTAMP	Yes	

`fee_structure_items`: one row per head in a structure.

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`fee_structure_id`	FK `fee_structures`	No	Unique with `fee_head_id`
`fee_head_id`	FK `fee_heads`	No	
`annual_amount`	DECIMAL(12,2)	No	Sum of its installment amounts
`created_at`, `updated_at`	TIMESTAMP	Yes	

`fee_item_amounts`: the cells of the grid.

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`fee_structure_item_id`	FK `fee_structure_items`	No	Unique with `fee_installment_id`
`fee_installment_id`	FK `fee_installments`	No	
`amount`	DECIMAL(12,2)	No	0 allowed

`concession_types` (+ standard, no `branch_id`)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(100)	No	
`category`	VARCHAR(30)	No	`sibling`, `staff_ward`, `merit`, `rte`, `need_based`, `other`
`value_type`	VARCHAR(10)	No	`percent`, `fixed`
`value`	DECIMAL(12,2)	No	
`head_ids`	JSON	Yes	Null = all heads that allow concessions
`requires_approval`	BOOLEAN	No	
`requires_document`	BOOLEAN	No	
`status`	VARCHAR(20)	No	`active`, `inactive`

`student_concessions` (+ standard)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`student_id`	FK `students`	No	
`academic_year_id`	FK `academic_years`	No	
`concession_type_id`	FK `concession_types`	No	
`value_override`	DECIMAL(12,2)	Yes	Replaces the type's value for this student
`status`	VARCHAR(20)	No	`pending`, `approved`, `rejected`, `revoked`
`approved_by`	FK `users`	Yes	Needs `fees.approve_concession`
`approved_at`	TIMESTAMP	Yes	
`document_path`	VARCHAR(255)	Yes	Private storage
`remarks`	VARCHAR(255)	Yes	

`fine_rules` (+ standard, no `branch_id`)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`name`	VARCHAR(100)	No	
`type`	VARCHAR(20)	No	`per_day`, `fixed`, `slab`
`amount`	DECIMAL(12,2)	Yes	For `per_day` and `fixed`
`slabs`	JSON	Yes	`[{"from":1,"to":15,"amount":50}, …]`
`grace_days`	TINYINT UNSIGNED	No	
`max_amount`	DECIMAL(12,2)	Yes	Per due line
`head_ids`	JSON	Yes	Null = all heads
`class_ids`	JSON	Yes	Null = all classes
`status`	VARCHAR(20)	No	`active`, `inactive`

`student_fees`: one due line (+ standard; replaces the earlier version).

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`student_id`	FK `students`	No	Index with `status`, `due_date`
`academic_year_id`	FK `academic_years`	No	
`fee_head_id`	FK `fee_heads`	No	
`fee_installment_id`	FK `fee_installments`	Yes	Null for manual and carried lines
`source`	VARCHAR(20)	No	`structure`, `manual`, `carry_forward`
`label`	VARCHAR(100)	No	"Tuition · June"
`due_date`	DATE	No	
`amount`	DECIMAL(12,2)	No	
`concession_amount`	DECIMAL(12,2)	No	Default 0

Column	Type	Null	Notes
`fine_amount`	DECIMAL(12,2)	No	Default 0; daily job
`fine_waived`	DECIMAL(12,2)	No	Default 0
`paid_amount`	DECIMAL(12,2)	No	Default 0
`balance`	DECIMAL(12,2)	No	Generated: amount − concession + fine − waived − paid
`status`	VARCHAR(20)	No	`unpaid`, `partial`, `paid`, `waived`, `cancelled`, `carried`
`carried_to_id`	FK `student_fees`	Yes	The new-year line it moved to
`remarks`	VARCHAR(255)	Yes	Required for manual lines

`fee_payments`: one receipt (+ standard; replaces the earlier version).

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`receipt_no`	VARCHAR(30)	No	Unique with `school_id`
`student_id`	FK `students`	No	
`academic_year_id`	FK `academic_years`	No	
`payment_date`	DATE	No	
`amount`	DECIMAL(12,2)	No	Money received
`credit_used`	DECIMAL(12,2)	No	Default 0
`credit_added`	DECIMAL(12,2)	No	Default 0; extra above dues
`mode`	VARCHAR(20)	No	`cash`, `upi`, `card`, `bank_transfer`, `cheque`, `scholarship`, `online`, `credit`
`reference_no`	VARCHAR(100)	Yes	
`cheque_no`, `cheque_date`, `bank_name`	VARCHAR(20), DATE, VARCHAR(100)	Yes	Cheque only
`payer_name`	VARCHAR(150)	Yes	Scholarship payer

Column	Type	Null	Notes
`online_payment_id`	FK `online_payments`	Yes	
`collected_by`	FK `users`	Yes	Null for online
`status`	VARCHAR(20)	No	`success`, `cancelled`, `bounced`
`cancelled_by`, `cancelled_at`, `cancel_reason`	FK, TIMESTAMP, VARCHAR(255)	Yes	
`remarks`	VARCHAR(255)	Yes	

`fee_payment_items`

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`fee_payment_id`	FK `fee_payments`	No	
`student_fee_id`	FK `student_fees`	No	
`fee_amount`	DECIMAL(12,2)	No	
`fine_amount`	DECIMAL(12,2)	No	Default 0
`created_at`, `updated_at`	TIMESTAMP	Yes	

`student_credit_ledger`: never updated, only added to.

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`school_id`	FK `schools`	No	
`student_id`	FK `students`	No	Index with `id` for the latest balance
`entry_type`	VARCHAR(10)	No	`credit`, `debit`
`amount`	DECIMAL(12,2)	No	Always positive
`balance_after`	DECIMAL(12,2)	No	Never below 0

Column	Type	Null	Notes
`source_type`	VARCHAR(30)	No	`payment`, `dues_applied`, `refund`, `adjustment`
`source_id`	BIGINT UNSIGNED	Yes	
`remarks`	VARCHAR(255)	Yes	
`created_by`	FK `users`	Yes	
`created_at`	TIMESTAMP	No	

`fee_refunds` (+ standard)

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`refund_no`	VARCHAR(30)	Yes	Set when paid; from ID formats
`student_id`	FK `students`	No	
`refund_of`	VARCHAR(20)	No	`head`, `credit`, `online`
`student_fee_id`	FK `student_fees`	Yes	For a refundable head
`fee_payment_id`	FK `fee_payments`	Yes	For an online refund
`amount`	DECIMAL(12,2)	No	
`reason`	VARCHAR(255)	No	
`mode`	VARCHAR(20)	No	`cash`, `bank_transfer`, `razorpay`
`reference_no`	VARCHAR(100)	Yes	
`gateway_refund_id`	VARCHAR(100)	Yes	
`status`	VARCHAR(20)	No	`requested`, `approved`, `rejected`, `paid`
`requested_by`, `approved_by`	FK `users`	Yes	Must be different users
`approved_at`, `paid_at`	TIMESTAMP	Yes	

`online_payments`: one Razorpay order; may cover siblings (one receipt per student).

Column	Type	Null	Notes
`id`	BIGINT UNSIGNED PK	No	
`school_id`	FK `schools`	No	
`guardian_user_id`	FK `users`	No	
`amount`	DECIMAL(12,2)	No	Fees part
`convenience_fee`	DECIMAL(12,2)	No	Default 0
`gateway_order_id`	VARCHAR(100)	No	Unique
`gateway_payment_id`	VARCHAR(100)	Yes	Unique
`selection`	JSON	No	Due lines and amounts chosen, per student
`status`	VARCHAR(20)	No	`created`, `paid`, `failed`, `expired`, `refunded`
`raw_payload`	JSON	Yes	Last gateway response or webhook
`paid_at`	TIMESTAMP	Yes	
`created_at`, `updated_at`	TIMESTAMP	Yes	

`fee_reminders` stays as in the main plan, with `channel` values `whatsapp_link`, `email`, `sms`.

Screens and permissions

The Fees menu has 12 sub-menu items, with Collect Fees first because it is used most.

Sub-menu	What it does	Permission
Collect Fees	Counter screen: find student, dues, pay, receipt	`fees.collect`
Receipts	Search, reprint, share, cancel, cheque bounce	`fees.view`, `fees.cancel_receipt`
Unpaid & Reminders	Defaulters list, WhatsApp and email reminders	`fees.view`

Sub-menu	What it does	Permission
Fee Structures	Heads × installments grid, publish, copy	`fees.manage_structures`
Fee Heads	The school's list of heads	`fees.manage_structures`
Concessions	Types, student concessions, approvals	`fees.concession`, `fees.approve_concession`
Fine Rules	Rules and waivers	`fees.manage_structures`, `fees.waive_fine`
Refunds	Requests, approval, payout	`fees.refund`, `fees.approve_refund`
Online Payments	Razorpay orders and status	`fees.view`
Carry Forward	Move last year's balances	`fees.carry_forward`
Fee Reports	All reports in Reminders and reports	`fees.export`
Fee Settings	Razorpay keys, receipt layout, allocation and credit rules	`fees.settings`

Other permissions: `fees.generate_dues` (publish a structure and create dues), `fees.manual_charge` (one-off lines). Portal: `portal.view_fees` and `portal.pay_fees` for Parent and Student roles.

Build order and estimate

With installments, concessions, fines, Razorpay, credit, refunds and carry forward, Fees grows from 3 weeks to about 6 weeks at 2 hours a day. The first paying school moves from about 21 to about 24 weeks.

1. **Week 1:** fee heads, structures with the installments grid, publish and copy.
2. **Week 2:** dues generation (publish, admission, section change, leaving), concessions with approval, fine rules and the daily fine job.
3. **Week 3:** Collect Fees screen, allocation, receipts (PDF and WhatsApp), cancel and cheque bounce, credit ledger.
4. **Week 4:** Razorpay: settings, orders, checkout in the portal, signature check, webhook, reconciliation job.
5. **Week 5:** refunds with approval, carry forward, unpaid list and reminders.

6. **Week 6:** reports; tests that every receipt, refund and credit entry adds up, and that one Razorpay payment can never create two receipts.

Test with Razorpay's test mode on the staging site before any school goes live with online payment.