CONFIDENTIAL

Business Requirements Document

Learning Center CRM

Document Version: 1.0

Date: February 8, 2026

Status: Final

Classification: Internal Use Only

Table of Contents

1. Executive Summary

1.1 Purpose

This Business Requirements Document (BRD) defines the complete functional and technical requirements for the **Learning Center CRM**—a web-based management system designed for educational institutions such as tutoring centers, language schools, and training academies. The document serves as the single source of truth for any developer to reproduce the system in its entirety.

1.2 Scope

The system covers the full operational lifecycle of a learning center:

- **Student lifecycle** — from lead capture through enrollment, payments, attendance, and graduation.
- **Financial management** — payments with month-wise allocation, expenses, invoicing, debt tracking, and salary processing.
- **Operational management** — groups, schedules, enrollments, attendance, and group transfers.
- **CRM pipeline** — lead tracking with priority, interactions, trial scheduling, and conversion.
- **Analytics & reporting** — dashboard, revenue charts, monthly reports, and audit trails.
- **Administration** — user management, role-based access, system settings, and unified audit logging.

1.3 Definitions & Acronyms

TERM	DEFINITION
CRM	Customer Relationship Management
Lead	A prospective student who has not yet enrolled
Enrollment	A student-to-group membership record with optional discount
Group	A class or course with a teacher, schedule, price, and capacity
Payment Month	A record tracking which calendar month a payment covers
Soft Delete	Marking a record's `deleted_at` timestamp instead of physically deleting it
Audit Log	An immutable record of who changed what, with before/after values
Salary Type: Fixed	Teacher receives a flat monthly salary amount
Salary Type: Per-Student	Teacher receives a percentage of collected payments from their groups

2. System Overview

2.1 Architecture & Technology Stack

LAYER	TECHNOLOGY	VERSION
Frontend	React + TypeScript	React 18, TypeScript 5.3+
Build Tool	Vite	5.x
UI Framework	Tailwind CSS + shadcn/ui (Radix)	Tailwind 3.4+
State & Data	TanStack Query (React Query)	v5
Forms	react-hook-form + Zod	—
Charts	Recharts	—
Icons	Lucide React	—
Routing	React Router	v6
Backend	PHP REST API	PHP 8+
Database	PostgreSQL with PDO	12+
Authentication	PHP Sessions + bcrypt	—

2.2 Project Structure

```
learning-center/
├── api/
│   └── index.php              # REST API (all endpoints)
├── frontend/
│   ├── src/
│   │   ├── components/        # 43 UI & business components
│   │   │   └── ui/            # shadcn/ui component library
│   │   ├── contexts/          # AuthContext (login/logout/role check)
│   │   ├── hooks/             # Custom hooks (useAmountInput, useStudents)
│   │   ├── lib/
│   │   │   ├── api.ts         # Typed API client + interfaces
│   │   │   └── utils.ts       # Utility functions
│   │   ├── pages/             # 16 page components
│   │   ├── App.tsx            # Router configuration
│   │   └── main.tsx           # Entry point
│   ├── package.json
│   └── vite.config.ts
├── config.php                 # DB config, auth, logging functions
├── router.php                 # HTTP route dispatcher
├── schema.sql                 # Base database schema
├── schema_additions.sql       # Extended tables
├── schema_v2.sql              # V2 enhancements
└── schema_source_tracking.sql # Referral source tracking
```

2.3 Deployment Model

The application is deployed as a single-server setup: the PHP backend serves the REST API, the Vite-built React frontend is served as static files, and PostgreSQL runs locally or on a networked database server. The PHP file `config.php` contains an `initDB()` function that auto-migrates the database schema on first API request, using `CREATE TABLE IF NOT EXISTS` and `ALTER TABLE ADD COLUMN IF NOT EXISTS` patterns.

3. User Roles & Authentication

3.1 Authentication Model

ASPECT	IMPLEMENTATION
Method	Session-based (PHP `$_SESSION`)
Password Hashing	bcrypt via `password_hash(PASSWORD_DEFAULT)`
Session Timeout	Configurable (default 30 minutes), checked on every API request
Deactivation Guard	Every authenticated request checks `users.is_active`; if `false`, session is destroyed and a 403 is returned
Default Admin	Seeded on first run: username `admin`, password `password`, role `admin`

3.2 User Roles

Roles are stored as a comma-separated string in `users.role` (e.g., `"admin,teacher"`). A user may hold multiple roles simultaneously.

ROLE	DESCRIPTION
admin	Full system access. Manages users, can soft-delete records, manages system settings.
manager	Manages students, groups, leads, attendance, reports. Cannot manage users or settings.
teacher	Views assigned groups, takes attendance, views student info. Read-only for most entities.
accountant	Manages payments, expenses, salary slips, and financial reports.
user	Default role. Minimal read-only access (view own profile).

3.3 Permission Matrix

MODULE / ROUTE	ADMIN	MANAGER	TEACHER	ACCOUNTANT	USER
Dashboard	✓	✓	✓	✓	—
Students	✓	✓	✓	✓	—
Teachers	✓	✓	—	✓	—
Groups	✓	✓	✓	✓	—
Leads (CRM)	✓	✓	—	—	—
Attendance	✓	✓	✓	✓	—
Payments	✓	✓	—	✓	—
Expenses	✓	✓	—	✓	—

MODULE / ROUTE	ADMIN	MANAGER	TEACHER	ACCOUNTANT	USER
Salaries	✓	—	—	✓	—
Reports	✓	✓	—	✓	—
Settings	✓	—	—	—	—
User Management	✓	—	—	—	—
Soft Delete operations	✓	—	—	—	—

4. Database Schema

4.1 Entity-Relationship Overview

The database consists of 26 tables organized into five domains:

- **Core Entities:** users, students, teachers, groups, enrollments
- **Financial:** payments, payment_months, payment_invoices, payment_plans, payment_plan_items, discounts, expenses
- **CRM:** leads, lead_interactions
- **Operations:** attendance, salary_slips, group_schedules, group_transfers
- **System:** audit_log, activity_log, notifications, settings, documents, communication_log, student_notes, password_reset_tokens

4.2 Table Definitions

4.2.1 users

COLUMN	TYPE	CONSTRAINTS	DESCRIPTION
id	SERIAL	PRIMARY KEY	Auto-incremented ID
username	VARCHAR(50)	UNIQUE NOT NULL	Login username
password	VARCHAR(255)	NOT NULL	bcrypt hash
name	VARCHAR(100)	—	Display name
role	VARCHAR(100)	DEFAULT 'user'	Comma-separated roles
teacher_id	INT	FK → teachers(id)	Link to teacher profile if applicable
email	VARCHAR(100)	—	Email address
phone	VARCHAR(20)	—	Phone number
is_active	BOOLEAN	DEFAULT true	Account active status
last_login	TIMESTAMP	—	Last login timestamp
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Record creation time

4.2.2 students

COLUMN	TYPE	CONSTRAINTS	DESCRIPTION
id	SERIAL	PRIMARY KEY	Auto-incremented ID
first_name	VARCHAR(50)	NOT NULL	First name
last_name	VARCHAR(50)	NOT NULL	Last name

COLUMN	TYPE	CONSTRAINTS	DESCRIPTION
dob	DATE	—	Date of birth
phone	VARCHAR(20)	—	Student phone
email	VARCHAR(100)	—	Student email
parent_name	VARCHAR(100)	—	Parent/guardian name
parent_phone	VARCHAR(20)	—	Parent phone
status	VARCHAR(20)	DEFAULT 'active'	active | inactive | graduated | suspended
notes	TEXT	—	General notes
source	VARCHAR(30)	NOT NULL DEFAULT 'walk_in'	How the student was acquired
referred_by_type	VARCHAR(20)	—	student | teacher | user
referred_by_id	INTEGER	—	ID of the referring entity
lead_id	INTEGER	FK → leads(id)	Source lead if converted
created_by	INTEGER	FK → users(id)	User who created the record
deleted_at	TIMESTAMP	DEFAULT NULL	Soft-delete timestamp
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Record creation time

4.2.3 teachers

COLUMN	TYPE	CONSTRAINTS	DESCRIPTION
id	SERIAL	PRIMARY KEY	Auto-incremented ID
first_name	VARCHAR(50)	NOT NULL	First name
last_name	VARCHAR(50)	NOT NULL	Last name
phone	VARCHAR(20)	—	Phone number
email	VARCHAR(100)	—	Email address
subjects	TEXT	—	Subjects taught
salary_type	VARCHAR(20)	DEFAULT 'fixed'	fixed | per_student
salary_amount	DECIMAL(10,2)	—	Fixed amount or percentage (0-100)
status	VARCHAR(20)	DEFAULT 'active'	active | inactive
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Record creation time

4.2.4 groups

COLUMN	TYPE	CONSTRAINTS	DESCRIPTION
id	SERIAL	PRIMARY KEY	Auto-incremented ID

COLUMN	TYPE	CONSTRAINTS	DESCRIPTION
name	VARCHAR(100)	NOT NULL	Group/class name
subject	VARCHAR(100)	—	Subject being taught
teacher_id	INT	FK → teachers(id)	Assigned teacher
capacity	INT	DEFAULT 15	Maximum student capacity
price	DECIMAL(10,2)	—	Monthly price per student
status	VARCHAR(20)	DEFAULT 'active'	active | inactive | completed
schedule_days	TEXT	—	Scheduled days (e.g., "Mon,Wed,Fri")
schedule_time_start	TEXT	—	Start time (HH:MM)
schedule_time_end	TEXT	—	End time (HH:MM)
room	TEXT	—	Classroom/room assignment
deleted_at	TIMESTAMP	DEFAULT NULL	Soft-delete timestamp
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Record creation time

4.2.5 enrollments

COLUMN	TYPE	CONSTRAINTS	DESCRIPTION
id	SERIAL	PRIMARY KEY	Auto-incremented ID
student_id	INT	FK → students(id) ON DELETE CASCADE	Enrolled student
group_id	INT	FK → groups(id) ON DELETE CASCADE	Enrolled group
enrolled_at	DATE	DEFAULT CURRENT_DATE	Enrollment date
discount_percentage	DECIMAL(5,2)	DEFAULT 0	Student discount (0–100%)
UNIQUE constraint on (student_id, group_id)			

4.2.6 payments

COLUMN	TYPE	CONSTRAINTS	DESCRIPTION
id	SERIAL	PRIMARY KEY	Auto-incremented ID
student_id	INT	FK → students(id)	Paying student
group_id	INT	FK → groups(id)	Group the payment is for
amount	DECIMAL(10,2)	NOT NULL	Payment amount
payment_date	DATE	DEFAULT CURRENT_DATE	Date payment was made
method	VARCHAR(20)	DEFAULT 'cash'	cash | card | transfer | other
notes	TEXT	—	Payment notes

COLUMN	TYPE	CONSTRAINTS	DESCRIPTION
deleted_at	TIMESTAMP	DEFAULT NULL	Soft-delete timestamp
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Record creation time

4.2.7 payment_months

COLUMN	TYPE	CONSTRAINTS	DESCRIPTION
id	SERIAL	PRIMARY KEY	Auto-incremented ID
payment_id	INT	FK → payments(id) ON DELETE CASCADE	Associated payment
for_month	DATE	NOT NULL	The month this payment covers (first day of month)
amount	DECIMAL(10,2)	NOT NULL	Amount allocated to this month
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Record creation time
UNIQUE constraint on (payment_id, for_month)			

4.2.8 leads

COLUMN	TYPE	CONSTRAINTS	DESCRIPTION
id	SERIAL	PRIMARY KEY	Auto-incremented ID
first_name, last_name	VARCHAR(50)	NOT NULL	Lead name
phone, email	VARCHAR	—	Contact info
parent_name, parent_phone	VARCHAR	—	Parent contact
source	VARCHAR(30)	NOT NULL DEFAULT 'walk_in'	Lead source channel
status	VARCHAR(20)	DEFAULT 'new'	Pipeline stage (see §9.2)
priority	VARCHAR(10)	DEFAULT 'warm'	hot | warm | cold
notes	TEXT	—	General notes
follow_up_date	DATE	—	Next follow-up date
interested_courses	TEXT	—	Courses of interest
trial_date	DATE	—	Scheduled trial date
trial_group_id	INT	FK → groups(id)	Trial group
birth_year	INT	—	Year of birth
preferred_schedule	TEXT	—	Preferred class times
budget	VARCHAR(50)	—	Budget range

COLUMN	TYPE	CONSTRAINTS	DESCRIPTION
loss_reason	TEXT	—	Reason for loss (if status=lost)
last_contact_date	DATE	—	Last interaction date
converted_student_id	INT	—	Student ID if converted
created_by	INTEGER	FK → users(id)	Creator
referred_by_type / referred_by_id	VARCHAR / INT	—	Referral source
deleted_at	TIMESTAMP	DEFAULT NULL	Soft-delete timestamp
created_at, updated_at	TIMESTAMP	—	Timestamps

4.2.9 Additional Tables (Summary)

TABLE	KEY COLUMNS	PURPOSE
attendance	student_id, group_id, attendance_date, status	Daily attendance (present/absent/late/excused). UNIQUE(student, group, date).
salary_slips	teacher_id, period_start/end, base/bonus/deduction/total, status, paid_at	Teacher salary records with payment tracking.
expenses	category, amount, description, expense_date, deleted_at	Center expense tracking by category.
discounts	payment_id, student_id, discount_type, amount, reason	Per-payment discount records.
payment_invoices	payment_id (UNIQUE), invoice_no	Auto-generated invoice numbers.
group_transfers	student_id, from_group_id, to_group_id, paid_month, discount_percentage	Student transfer history between groups.
group_schedules	group_id, day_of_week, start_time, end_time, classroom	Multi-day weekly schedule entries.
lead_interactions	lead_id, type, notes, scheduled_at, completed_at, created_by	CRM communication history.
audit_log	user_id, action, entity_type, entity_id, old_values, new_values, ip_address, details	Unified change history with JSONB before/after values.
activity_log	user_id, action, entity, entity_id, details	Legacy simple action log (retained for historical data).
notifications	user_id, type, title, message, link, is_read	In-app notification system.
settings	key (PK), value, description	System configuration key-value store.
documents	entity_type, entity_id, filename, file_path, mime_type	File upload metadata.
student_notes	student_id, note_type, content, created_by	Student journal / progress notes.

4.3 Cross-Cutting Database Patterns

PATTERN	DETAILS
Soft Delete	Tables with `deleted_at TIMESTAMP DEFAULT NULL`: payments, expenses, salary_slips, students, leads, groups. All queries must include `WHERE deleted_at IS NULL`.
Auto-Migration	`initDB()` uses `CREATE TABLE IF NOT EXISTS` and `ALTER TABLE ADD COLUMN IF NOT EXISTS`. No separate migration runner needed.
JSONB Audit	`audit_log.old_values` and `new_values` store full before/after snapshots as JSONB.
Composite Unique	enrollments(student_id, group_id), attendance(student_id, group_id, attendance_date), payment_months(payment_id, for_month).

5. Functional Requirements

5.1 Student Management

FR-STU-01: List Students

Display a searchable, filterable table of all non-deleted students. Filters: status (active/inactive/graduated/suspended), text search (name, phone, email), group, source. Each row shows: name, phone, email, groups list, current month debt, status.

FR-STU-02: Current Month Debt Calculation

For each student, calculate: `debt = SUM(group_price × (1 − discount_pct/100)) − SUM(payment_months.amount for current month)`. Display expected, paid, and remaining amounts.

FR-STU-03: Create Student

Required fields: first_name, last_name, source. Optional: dob, phone, email, parent_name, parent_phone, status, notes. If source is "referral", capture referred_by_type (student/teacher/user) and referred_by_id. Records the creating user's ID.

FR-STU-04: Update Student

Edit all student fields. System fetches old values before update and logs both old and new values to the audit log.

FR-STU-05: Delete Student (Soft)

Admin only. Sets `deleted_at = NOW()`. Logs old values to audit log. Student remains in database for historical reporting.

FR-STU-06: Student Detail View

Displays: personal info, parent/emergency contacts, enrollment list with discounts, payment history, group transfer history, and student notes.

FR-STU-07: Bulk Import

Import students via CSV/structured data through `POST /api/students-import`.

5.2 Teacher Management

FR-TCH-01: Create Teacher from User

A teacher is created by linking an existing user account. The system splits the user's display name into first_name/last_name and creates a teacher record. The user's `teacher_id` is set to the new teacher ID. Required input: user_id, subjects, salary_type, salary_amount.

FR-TCH-02: Salary Types

- **Fixed:** `salary_amount` is the flat monthly salary in currency.
- **Per-Student:** `salary_amount` is a percentage (0–100). Monthly salary = SUM(collected payments for teacher's groups in that month) × (salary_amount / 100).

FR-TCH-03: Update/Delete Teacher

Update: edits all fields, logs old/new to audit log. Delete: hard delete (teachers do not use soft delete), logs old values.

5.3 Group (Class) Management

FR-GRP-01: Create Group

Fields: name, subject, teacher_id, capacity, price, status, schedule_days, schedule_time_start, schedule_time_end, room. Creates audit log entry.

FR-GRP-02: List Groups

Displays groups with: name, subject, teacher name (JOINed), student count (from enrollments), capacity, price, status. Excludes soft-deleted groups.

FR-GRP-03: Group Detail

Shows: overview, student roster, attendance quick-view, schedule, payment summary by group, teacher assignment.

FR-GRP-04: Delete Group (Soft)

Admin only. Sets `deleted_at = NOW()`. Logs old values.

5.4 Enrollment & Group Transfers

FR-ENR-01: Enroll Student

Creates enrollment with student_id, group_id, and discount_percentage (0–100). Uses `ON CONFLICT DO UPDATE` to handle re-enrollment.

FR-ENR-02: Update Discount

Modify enrollment-level discount percentage. Affects future debt calculations.

FR-ENR-03: Group Transfer

Transfers a student from one group to another within a single transaction:

1. Verify student is enrolled in source group and not in target group.
2. Check if current month is paid (≥50% of expected payment = considered paid).
3. Calculate total outstanding debt in source group.
4. Delete enrollment in source group, create enrollment in target group with new discount.
5. If month was paid, create a $0 "transfer credit" payment in target group with a payment_months entry for the full target monthly rate.
6. Record transfer in `group_transfers` table.
7. Return success with warnings about any outstanding source debt.

5.5 Payment & Billing

FR-PAY-01: Create Payment

Input: student_id, group_id, amount, payment_date, method, notes, months[] (array of YYYY-MM strings).

Validation: If months are specified, calculate total remaining debt across selected months. Reject if amount exceeds total remaining debt.

Month Distribution: Distribute payment across months sequentially. For each month: calculate remaining debt = (group_price × (1 - discount%)) - already_paid. Allocate min(remaining_amount, debt_for_month). Continue until amount is fully allocated.

Invoice: Auto-generate invoice number in format `INV-YYYYMMDD-NNNN` (zero-padded payment ID).

FR-PAY-02: Debt Calculation

For a given student + group + month: `remaining_debt = (group_price × (1 - discount_pct/100)) - SUM(payment_months.amount WHERE for_month = target_month)`.

FR-PAY-03: Delete Payment (Soft)

Admin only. Sets `deleted_at = NOW()`. All report queries exclude soft-deleted payments.

5.6 Expense Management

FR-EXP-01: Record Expense

Fields: category, amount, description, expense_date. Categories are free-text (user-defined).

FR-EXP-02: Delete Expense (Soft)

Admin only. Sets `deleted_at = NOW()`.

5.7 Attendance Tracking

FR-ATT-01: Record Attendance

Input: group_id, date (cannot be future), array of {student_id, status}. Status values: present, absent, late, excused.

Uses `INSERT ... ON CONFLICT DO UPDATE` for idempotent saves. Each change is individually audit-logged with old/new values.

FR-ATT-02: View Attendance

Shows all enrolled students for a group on a given date, with their attendance status. Students without a record for that date show as unmarked.

5.8 Salary Management

FR-SAL-01: Salary Preview

Before creating a salary slip, show a preview: for fixed-salary teachers, display the fixed amount; for per-student teachers, calculate percentage of collected payments from their groups in the target month.

FR-SAL-02: Create Salary Slip

Fields: teacher_id, period_start, period_end, base_amount (auto-calculated from salary type), bonus, deduction. Total = base + bonus - deduction. Status: pending or paid.

FR-SAL-03: Pay Salary Slip

Update status from "pending" to "paid" and set paid_at timestamp. Audit-logged.

FR-SAL-04: Delete Salary Slip (Soft)

Admin only. Sets `deleted_at = NOW()`.

5.9 Lead Management (CRM)

FR-LEAD-01: Lead Pipeline

Leads progress through stages: `new → contacted → interested → trial_scheduled → trial_completed → negotiating → enrolled | lost | postponed`.

Sorting priority: active leads first (not enrolled/lost), then by priority (hot > warm > cold), then by follow_up_date ascending, then by created_at descending.

FR-LEAD-02: Lead Fields

Core: name, phone, email, parent info, source, status, priority. Enhanced: interested_courses, trial_date, trial_group_id, birth_year, preferred_schedule, budget, loss_reason, last_contact_date.

FR-LEAD-03: Lead Interactions

Log communication history with type (call, whatsapp, email, meeting, trial, note), notes, scheduled_at, completed_at. Adding an interaction auto-updates the lead's `last_contact_date`.

FR-LEAD-04: Lead Conversion

Convert a lead to a student in a single transaction: create student record copying lead fields (name, phone, email, parent info, source, referral info), set lead status to "enrolled", set `converted_student_id`. Student's `lead_id` references the original lead.

FR-LEAD-05: Lead Statistics

Dashboard metrics: count by status, follow-ups due today, overdue follow-ups, trials scheduled, hot leads count, conversions this month, leads by source.

FR-LEAD-06: Delete Lead (Soft)

Sets `deleted_at = NOW()`. Logs old values to audit log.

5.10 Dashboard & Analytics

FR-DASH-01: Summary Statistics

Cards showing: active students, active teachers, active groups, current month revenue, current month expenses, current month profit, pending leads. Each with trend indicator (percentage change vs. previous month).

FR-DASH-02: Revenue Chart

6-month bar/line chart showing monthly revenue, expenses, and profit. Excludes soft-deleted records. Calculates month-over-month growth percentage.

5.11 Reports

FR-RPT-01: Payment Report

Date-range filtered list of payments with student name, group name, method, amount. Excludes soft-deleted payments.

FR-RPT-02: Expense Report

Date-range filtered list of expenses. Excludes soft-deleted expenses.

FR-RPT-03: Income vs. Expense Summary

Aggregated totals for a date range: total income (SUM payments), total expenses (SUM expenses), net profit. Both exclude soft-deleted records.

FR-RPT-04: Monthly Detailed Report

For a selected month, per active group (excluding soft-deleted groups):

- **Student count:** enrolled active students (excluding soft-deleted students)
- **Expected amount:** SUM(group_price × (1 - discount%)) for all enrolled students
- **Collected amount:** SUM(payment_months.amount) for the month (excluding soft-deleted payments)
- **Remaining debt:** expected - collected
- **Payment percentage:** (collected / expected) × 100
- **Teacher portion:** For per-student teachers: collected × (salary_amount / 100)
- **Center portion:** collected - teacher_portion

Includes aggregate totals row across all groups.

FR-RPT-05: Overview Charts

Pie chart: expenses by category. Bar chart: payments by method.

5.12 Audit & Activity Logging

FR-AUD-01: Unified Audit Log

All entity changes (create, update, delete, soft_delete) and actions (login, logout, lead_convert, attendance_save) are recorded in a single `audit_log` table. The `activityLog()` function is a thin wrapper that writes to `audit_log` with null old/new values.

FR-AUD-02: Audit Log Fields

Each entry captures: user_id, action, entity_type, entity_id, old_values (JSONB), new_values (JSONB), ip_address, details (TEXT), created_at.

FR-AUD-03: Audit Log UI

Filterable table in Reports page: filter by date range, entity type (payment, student, lead, group, discount, attendance, salary_slip, teacher, user), action (create, update, soft_delete, delete, login, logout, lead_convert). Supports pagination (50 per page) and CSV export.

FR-AUD-04: Entities Tracked

ENTITY	ACTIONS LOGGED	VALUES CAPTURED
Student	create, update, soft_delete	Key fields (name, phone, email, status, source)
Teacher	create, update, delete	All fields except password
Group	create, update, soft_delete	name, subject, teacher_id, capacity, price, status
Lead	create, update, soft_delete	name, phone, source, status, priority
User	create, update, delete	username, name, role (never password)

ENTITY	ACTIONS LOGGED	VALUES CAPTURED
Payment	create, update, soft_delete	student_id, group_id, amount, date, method
Discount	create, update, delete	All fields
Attendance	create, update	student_id, group_id, date, status
Salary Slip	create, update, soft_delete	status, paid_at, amounts
Session	login, logout	user_id (via activityLog wrapper)

5.13 User Management

FR-USR-01: Create User

Admin only. Fields: username (unique), password (hashed with bcrypt), name, role (comma-separated), teacher_id (optional link), email, phone. Audit-logged with username, name, role (never password).

FR-USR-02: Update User

Admin only. Can modify: name, role, email, phone, is_active, teacher_id, password. Logs old/new values (excluding password).

FR-USR-03: Deactivate User

Setting `is_active = false` triggers automatic logout on next API request (the `auth()` function checks this on every call).

FR-USR-04: Delete User

Admin only. Hard delete. System prevents deletion of the currently logged-in user. Audit-logged.

5.14 Settings & Configuration

FR-SET-01: System Settings

Admin only. Key-value store for: organization_name, currency, session_timeout, payment_reminder_days, notification preferences, contact_email, contact_phone.

5.15 Notifications

FR-NOT-01: Notification System

In-app notifications with type, title, message, link. Users can mark individual notifications as read or mark all as read.

6. API Specification

6.1 General Conventions

ASPECT	CONVENTION
Base URL	`/api`
Content-Type	`application/json; charset=utf-8`
Authentication	Session cookie (credentials: 'include' in fetch)
Success Response (GET)	JSON array or object with data
Success Response (POST)	`{"id": <number>}` or `{"ok": true}`
Error Response	`{"error": "message"}` with appropriate HTTP status
Date Format	YYYY-MM-DD
Month Format	YYYY-MM
Timestamp Format	YYYY-MM-DD HH:MM:SS (ISO 8601)

6.2 Endpoint Reference

METHOD	ENDPOINT	AUTH	DESCRIPTION
Authentication			
POST	/api/login	None	Login with username/password. Returns user object.
POST	/api/logout	Any	Destroy session.
GET	/api/me	Any	Current user + last_activity timestamp.
Students			
GET	/api/students	admin, manager, teacher, accountant	List with filters (?status, ?search, ? group_id, ?source).
POST	/api/students	admin, manager, teacher, accountant	Create student. Requires source field.
PUT	/api/students/{id}	admin, manager, teacher, accountant	Update student fields.
DELETE	/api/students/{id}	admin	Soft delete.
Teachers			

METHOD	ENDPOINT	AUTH	DESCRIPTION
GET	/api/teachers	admin, manager, accountant	List all teachers.
POST	/api/teachers	admin, manager, accountant	Create teacher from user. Requires user_id.
PUT	/api/teachers/{id}	admin, manager, accountant	Update teacher fields.
DELETE	/api/teachers/{id}	admin, manager, accountant	Hard delete teacher.
Groups			
GET	/api/groups	admin, manager, teacher, accountant	List groups with teacher name & student count.
POST	/api/groups	admin, manager, teacher, accountant	Create group with schedule info.
PUT	/api/groups/{id}	admin, manager, teacher, accountant	Update group.
DELETE	/api/groups/{id}	admin	Soft delete.
Enrollments			
GET	/api/enrollments	admin, manager, teacher, accountant	List (?group_id or ?student_id).
POST	/api/enrollments	admin, manager, teacher, accountant	Enroll student in group with discount.
PUT	/api/enrollments/{id}	admin, manager, teacher, accountant	Update discount percentage.
DELETE	/api/enrollments/{id}	admin, manager, teacher, accountant	Remove enrollment.
Group Transfers			
GET	/api/group-transfers	admin, manager	List transfers (?student_id).
POST	/api/group-transfers	admin, manager	Transfer student between groups.
Payments			
GET	/api/payments	admin, manager, accountant	List with months_covered (LIMIT 500).
POST	/api/payments	admin, manager, accountant	Create payment with month allocation.
PUT	/api/payments/{id}	admin, manager, accountant	Update payment.
DELETE	/api/payments/{id}	admin	Soft delete.

METHOD	ENDPOINT	AUTH	DESCRIPTION
Discounts			
GET	/api/discounts	admin, manager, accountant	List (?payment_id or ?student_id).
POST	/api/discounts	admin, manager, accountant	Create discount on payment.
PUT	/api/discounts/{id}	admin, manager, accountant	Update discount.
DELETE	/api/discounts/{id}	admin, manager, accountant	Hard delete discount.
Expenses			
GET	/api/expenses	admin, manager, accountant	List expenses (LIMIT 500).
POST	/api/expenses	admin, manager, accountant	Create expense.
DELETE	/api/expenses/{id}	admin	Soft delete.
Leads (CRM)			
GET	/api/leads	admin, manager	List leads sorted by priority.
POST	/api/leads	admin, manager	Create lead with all CRM fields.
PUT	/api/leads/{id}	admin, manager	Dynamic field update.
DELETE	/api/leads/{id}	admin, manager	Soft delete.
POST	/api/leads/{id}/convert	admin, manager	Convert lead to student (transactional).
GET	/api/leads/{id}/interactions	admin, manager	List lead interactions.
POST	/api/leads/{id}/interactions	admin, manager	Log interaction.
GET	/api/lead-stats	admin, manager	CRM dashboard statistics.
Attendance			
GET	/api/attendance	admin, manager, teacher, accountant	Get attendance (?group_id & ?date).
POST	/api/attendance	admin, manager, teacher, accountant	Save bulk attendance (no future dates).
Salaries			
GET	/api/teacher-salary-preview	admin, accountant	Preview calculated salary (?teacher_id & ?month).
GET	/api/salary-slips	admin, accountant	List salary slips (LIMIT 200).

METHOD	ENDPOINT	AUTH	DESCRIPTION
POST	/api/salary-slips	admin, accountant	Create salary slip.
PUT	/api/salary-slips/{id}	admin, accountant	Update status/paid_at.
DELETE	/api/salary-slips/{id}	admin	Soft delete.
Dashboard			
GET	/api/dashboard/stats	admin, manager, teacher, accountant	Summary stats with trends.
GET	/api/dashboard/revenue-chart	admin, manager, teacher, accountant	6-month revenue/expense data.
Reports			
GET	/api/reports/payments	admin, manager, accountant	Payment report (?from & ?to).
GET	/api/reports/expenses	admin, manager, accountant	Expense report (?from & ?to).
GET	/api/reports/income-expense	admin, manager, accountant	Income/expense summary.
GET	/api/reports/attendance	admin, manager, accountant	Attendance report (?from, ?to, ?group_id).
GET	/api/reports/monthly	admin, manager, accountant	Monthly detailed report (?month=YYYY-MM).
Users (Admin)			
GET	/api/users	admin, manager, accountant	List users with teacher info.
POST	/api/users	admin	Create user.
PUT	/api/users/{id}	admin	Update user.
DELETE	/api/users/{id}	admin	Hard delete (cannot delete self).
Other			
GET	/api/referrers	admin, manager	Referrer list by type (?type).
GET	/api/student-debt	Any auth	Calculate debt (?student_id, ?group_id, ?month).
GET	/api/audit-log	admin, manager, accountant	Paginated audit log with filters.
GET/PUT	/api/settings	admin	System settings.
GET/PUT	/api/notifications	Any auth	User notifications.
PUT	/api/profile	Any auth	Update own profile.

METHOD	ENDPOINT	AUTH	DESCRIPTION
PUT	/api/profile/password	Any auth	Change own password.
GET/POST/PUT/DELETE	/api/schedules	admin, manager, teacher	Group schedules CRUD.
POST	/api/students-import	admin, manager	Bulk import students.
GET	/api/students-export	admin, manager	Export students.

7. Frontend Specification

7.1 Pages & Navigation

PAGE	ROUTE	KEY FEATURES
Login	/login	Bifold design, username/password with show/hide toggle, error handling.
Dashboard	/	6 stat cards with trends, 6-month revenue chart, quick actions.
Students	/students	Filterable table (status, search, group, source), create/edit dialogs, debt display, bulk import.
Student Detail	/students/{id}	Profile, enrollments, payment history, transfers, notes.
Teachers	/teachers	Directory with salary info, create from user account, status management.
Teacher Detail	/teachers/{id}	Profile, classes taught, salary details.
Groups	/groups	Table with student count/capacity, schedule info, create/edit forms.
Group Detail	/groups/{id}	Student roster, attendance view, schedule, payments.
Leads	/leads	CRM pipeline, priority badges, interaction timeline, conversion, follow-up tracking.
Payments	/payments	Payment form with month selector, debt calculator, invoice display, history.
Expenses	/expenses	Expense recording, category management, date filter.
Attendance	/attendance	Group selector, date picker, bulk status save, historical view.
Salaries	/salaries	Salary slip creation, preview, bonus/deduction, payment tracking.
Reports	/reports	5 tabs: Overview (charts), Monthly, Payments, Expenses, Change History (audit log).
Settings	/settings	Organization config, session timeout, user management (nested).
Users	/settings/users	User CRUD, role assignment, teacher linking, activation toggle.

7.2 Component Library

The UI is built on **shadcn/ui** (Radix primitives + Tailwind). 35+ components including: AlertDialog, Avatar, Badge, Button, Card, Checkbox, Dialog, DropdownMenu, Input, Label, ScrollArea, Select, Separator, Sheet, Skeleton, Switch, Table, Tabs, Textarea, Toast, Tooltip.

Custom components: DateInput, TimeInput, PhoneInput, StatsCard, RevenueChart, Timetable, LegacyAcademyLogo, ProtectedRoute, MainLayout, Header, Sidebar.

Skeleton loaders: DashboardSkeleton, TableSkeleton, StudentsSkeleton, StudentDetailSkeleton, AttendanceSkeleton, ChartSkeleton.

7.3 Form Patterns & Validation

PATTERN	IMPLEMENTATION
Form library	react-hook-form with Zod resolver
Validation	Zod schemas (compile-time + runtime type safety)
Stale data fix	Use `key` prop to force remount OR `reset()` in useEffect when editing existing records
Amount fields	Custom `useAmountInput` hook for decimal handling
Important: setValue	Never pass `undefined` to react-hook-form `setValue()`; keep non-schema fields in local state

7.4 State Management

CONCERN	SOLUTION
Server data	TanStack Query (React Query) with query keys for cache invalidation
Authentication	React Context (AuthContext) with login/logout/hasRole
Form state	react-hook-form (local to each form)
UI state	React useState/useMemo (local to each page)

8. Non-Functional Requirements

8.1 Performance

- API queries use `LIMIT` clauses (payments: 500, salary slips: 200, group transfers: 100).
- Database indexes on frequently queried columns (student_id, group_id, for_month, payment_date, expense_date, source, lead_id).
- TanStack Query caches results client-side to reduce redundant requests.
- Prepared statements (`PDO::prepare`) used for all parameterized queries.

8.2 Security

- **SQL Injection:** All database queries use PDO prepared statements with parameterized values.
- **Authentication:** bcrypt password hashing, session timeout enforcement, is_active deactivation guard.
- **Authorization:** Every API endpoint validates role permissions via `requireRole()` .
- **CORS:** Headers configured for cross-origin requests.
- **Sensitive data:** Passwords are never logged in audit log. Passwords are never returned in API responses.
- **Self-protection:** Admin cannot delete their own user account.

8.3 Data Integrity

- **Foreign keys:** Enforced at database level with appropriate CASCADE rules.
- **Unique constraints:** Username, enrollment (student+group), attendance (student+group+date), payment_months (payment+month).
- **Transactional operations:** Lead conversion and group transfers use database transactions with rollback on failure.
- **Soft delete:** Financial and student records are never physically deleted; `deleted_at` timestamp preserves data for audit and reporting.
- **Audit trail:** Immutable log entries with before/after JSONB values, user ID, IP address, and timestamp.

9. Appendices

9.1 Source Options

Used for both leads and students to track acquisition channel:

VALUE	DISPLAY LABEL
instagram	Instagram
telegram	Telegram
facebook	Facebook
website	Website
referral	Referral (requires referred_by_type + referred_by_id)
walk_in	Walk-in (default)
phone	Phone Call
flyer	Flyer/Banner
event	Event
other	Other

9.2 Lead Status Flow

```
new → contacted → interested → trial_scheduled → trial_completed → negotiating
                                                                        ↓
                                                                enrolled (converte
                                                                      OR
                                                                lost (with loss_re
                                                                      OR
                                                                postponed
```

9.3 Invoice Number Format

`INV-YYYYMMDD-NNNN`

- **YYYYMMDD** — Date the payment was recorded
- **NNNN** — Zero-padded payment ID (4 digits minimum)

Example: `INV-20260208-0042`