LCMS – Comprehensive Functional & Technical Specification

# Legal Case Management System (LCMS)

Full Project Analysis & Customer-Facing Documentation

## Contents

* [1. Introduction & Purpose](#intro)
* [2. Architectural Overview](#architecture)
* [3. User Roles & Access Matrix](#roles)
* [4. Core Modules (one-by-one)](#modules)
  + [4.1 Case Management](#module-cases)
  + [4.2 Appeals & Cassation](#module-appeals)
  + [4.3 Supervisor Approval](#module-supervision)
  + [4.4 Audit Log](#module-audit)
  + [4.5 Notification Engine](#module-notify)
  + [4.6 Reporting & Analytics](#module-reports)
  + [4.7 Administration & Settings](#module-settings)
* [5. Case Type Specifications](#case-types)
* [6. Database Schema Overview](#db)
* [7. Security & Compliance](#security)
* [8. Appendix: Endpoints & Screens](#appendix)

## 1. Introduction & Purpose

This document provides an exhaustive description of the Legal Case Management System (LCMS). It is tailored for organisational stakeholders, external auditors, and future developers. Every functional module, workflow, screen and database artifact is documented in detail to facilitate understanding, onboarding, regulatory review, and change impact analysis.

## 2. Architectural Overview

### 2.1 Technology Stack

| Layer | Technology | Purpose |
| --- | --- | --- |
| Frontend | Laravel Blade, TailwindCSS, Alpine.js | Component-based UI with reactive widgets |
| Backend | Laravel Framework v12 (PHP 8.2) | RESTful controllers, Eloquent ORM, jobs, notifications |
| Database | MySQL 8 | Transactional data store |
| Auth | Laravel Breeze + Sanctum | Password & token-based authentication |
| Files | Laravel Storage (Local / S3) | Evidence & document uploads |
| Notifications | Laravel Notification Channels | Email (SMTP) + in-app |

### 2.2 High-Level Component Diagram

*(For MS Word export, insert architecture image here.)*

## 3. User Roles & Access Matrix

The LCMS uses role-based access control (RBAC). Each permission can be fine-tuned in the permissions table via the Admin UI.

| Module / Activity | Role Permission | | | |
| --- | --- | --- | --- | --- |
| Lawyer | Supervisor | Admin | SysAdmin |
| Create Case | ✓ Own | ✓ All | ✓ All | — |
| Edit Case | ✓ Own | ✓ All | ✓ All | — |
| Approve Execution | — | ✓ | ✓ | — |
| Manage Users | — | — | ✓ | ✓ |
| System Backup | — | — | ✓ | ✓ |

## 4. Core Modules

### 4.1 Case Management

Central module where lawyers record the complete lifecycle of each litigation matter.

1. **Case Entry Form** – Dynamically loads fields as per selected case type. Mandatory validation enforced server-side and via HTML5.
2. **Case Detail View** – Accordion layout showing Overview, Appeal stages, Parties, Progress Updates, Appointments and Type-Specific panels.
3. **Progress Updates** – Inline form allows attachment upload (max 8MB, configurable). Saved to progress\_updates table.

### 4.2 Appeals & Cassation

Implements strict ordered flow: Direct Suit → Appeal → Second Appeal → Cassation. Duplicate safeguards include Eloquent validation and DB unique index (case\_file\_id, level).

* Form Path: GET /lawyer/cases/{id}/appeals/create
* Controller: AppealController@store (handles flow enforcement)
* Status timeline auto-logs per stage.

### 4.3 Supervisor Approval

The Supervisor Dashboard lists cases in Pending Execution. Approval acts as a final gate, ensuring compliance with corporate policy before enforcement.

### 4.4 Audit Log

Trait Auditable hooks into creating, updating, deleting. Audit entries retain *before* & *after* JSON snapshots for legal defensibility. Logs older than 5 years can be archived.

### 4.5 Notification Engine

Notifications stored in notifications table and dispatched via Laravel queues. Templates:

* New Case Assignment
* Upcoming Hearing Reminder
* Supervisor Decision Notice

### 4.6 Reporting & Analytics

KPI dashboards powered by Blade + Chart.js. Export endpoints deliver CSV/XLSX.

### 4.7 Administration & Settings

Admin menu items: Branches, Work Units, Court Codes, Backup, Roles, Permissions. Changes utilise optimistic locking to avoid config races.

## 4. Core Modules – Exhaustive Breakdown

### 4.1 Case Management

The Case Management module is the nucleus of LCMS. It governs every CRUD operation related to case\_files and their child entities. The module is constructed around a REST-style controller (Lawyer\CaseController) supported by Service classes for heavy business logic.

#### 4.1.1 Functional Scope

* Create, edit, view and archive cases
* Dynamic form rendering based on case\_type\_id
* Attachment handling (drag-and-drop, multi-upload, virus scan hook)
* Automatic file number generator adhering to business prefix codes

#### 4.1.2 UI Flow

1. **Dashboard → “New Case”**: Lawyer selects case type.
2. **Dynamic Entry Form**: Case-type specific fields appear via Alpine.js component case-form.
3. **Validation**: Laravel Form Request + client-side constraints.
4. **Persist**: Transactional save across case\_files, related plaintiffs/defendants and type-specific tables.
5. **Redirect** to Case Detail page with toast confirmation.

#### 4.1.3 Data Model

| Table | Key Fields | Relations |
| --- | --- | --- |
| case\_files | id, file\_number, status, opened\_at | 1-to-\* appeals, progress\_updates, appointments |
| plaintiffs | name, contact\_info | FK case\_file\_id |

#### 4.1.4 Exception Handling

Every DB transaction is wrapped in try–catch; failures logged via Monolog channel *cases*.

### 4.2 Appeals & Cassation

This module enforces sequential appeal progression and maintains jurisprudential history.

#### 4.2.1 Business Rules

* Allowed Levels: *Appeal* ⇒ *Second* ⇒ *Cassation*.
* Exactly one record per level per case (DB unique index).
* Attempt to skip or duplicate a level triggers 422 error with explicit message.

#### 4.2.2 Controller Responsibilities

AppealController@store  
├─ Validate payload  
├─ Verify sequence integrity  
├─ Persist to appeals table  
└─ Dispatch ProgressUpdate("Appeal Filed")

#### 4.2.3 DB Indices & Constraints

| Index Name | Columns | Purpose |
| --- | --- | --- |
| appeals\_case\_level\_unique | case\_file\_id, level | Prevents duplication |

### 4.3 Supervisor Approval

The Supervision layer adds managerial oversight. The approval routine is exposed via Supervisor\CaseController@approveClosure.

#### Process Steps

1. Supervisor opens /supervisor/cases/approvals listing.
2. Clicks “Review” → Modal shows judgment summary & attachments.
3. Decision recorded (approved | follow\_up).
4. System logs progress update and fires notification to Lawyer.

### 4.4 Audit Logging

Audit entries are created via global Auditable trait attached to all major models. JSON diff is generated using spatie/laravel-activitylog.

#### Stored Columns

* event (created|updated|deleted)
* user\_id
* url
* ip\_address
* old\_values
* new\_values

### 4.5 Notification Engine

Notifications utilise Laravel queue workers. All templates are localised via lang/{locale}/notifications.php.

#### Channels

| Channel | Driver | Use-Case |
| --- | --- | --- |
| Email | SMTP | Case assignments, hearing reminders |
| Database | In-app | Supervisor approvals, document comments |

### 4.6 Reporting & Analytics

Reports are generated on-demand; heavy queries are cached for 30 minutes using Redis.

#### Key Reports

* Appeal Performance (stage-wise win/loss)
* Case Ageing Buckets (0-90, 91-180, 180+ days)
* Recovery Rate per Branch

### 4.7 Administration & Settings

The Settings module manipulates reference data: Branches, Work Units, Courts, Roles, Permissions. Each entity uses standard resource controllers and Livewire Datatables for inline editing.

#### Backup & Restore

Admins can trigger on-demand database dump; files stored in storage/backups with timestamped naming convention.

## 5. Case Type Specifications

Below tables list *every field* rendered on entry/edit screens for each case type. Validation rules refer to Laravel form request classes.

| Field (Clean Loan) | Type | Validation | | Description |
| --- | --- | --- | --- | --- |
| Rule | Error Msg. |
| Branch | Select | required | "Select branch" | Originating bank branch |
| Internal File # | Text | required|unique | "Duplicate file #" | Org ref no. |
| Outstanding Amount | Decimal | required|min:0 | "Amount invalid" | Remaining principal |
| Claimed Amount | Decimal | required|min:0 | "Amount invalid" | Court claim value |
| Court Name | Select | required | "Select court" | Entraining court |
| Docs | File | nullable|mimes:pdf,docx|max:8192 | "Invalid file" | Supporting files |

*Repeat similar expanded tables for Labor, Civil, Criminal...*

## 6. Database Schema Overview

Key tables and relationships (PK ►, FK ↘):

* **case\_files ►** ↘ appeals, progress\_updates, appointments, documents
* **appeals ►** (case\_file\_id) unique per level
* **progress\_updates ►** (case\_file\_id)
* **audit\_logs ►** polymorphic auditable\_id/auditable\_type

## 7. Security & Compliance

* **Authentication:** HTTPS enforced, bcrypt hashed passwords, optional 2FA.
* **Authorisation:** Middleware CheckRole + Policy classes.
* **Data Protection:** S3/Local encryption at rest; document access signed URLs.
* **Audit Compliance:** Immutable logs meet ISO 27001 evidence requirements.

## 8. Appendix: Key Endpoints & Screenshots

### 8.1 API / Web Routes

| Route Name | URI | Method | Role Access |
| --- | --- | --- | --- |
| lawyer.cases.index | /lawyer/cases | GET | Lawyer |
| cases.appeals.store | /lawyer/cases/{id}/appeals | POST | Lawyer |
| supervisor.cases.approve | /cases/{id}/approve | POST | Supervisor |

### 8.2 Screen References

*Embed or link UI screenshots here for the customer PDF/Word version.*