People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

M'Hamed Bougara University of Boumerdes (UMBB)

Faculty of Sciences — Department of Computer Science

End-of-Study Project — PFE

Submitted in partial fulfilment of the requirements for the degree of *Licence en Informatique*

Design and Development of a Web-Based Project Management System for Sonatrach

Presented by:

Guettaf Aya Saada

Douibi Fella

Bouzegza Meroua Maram

Supervisor:

Mr Abbes

Promoter:

Mrs Fekkir Noura

Examiner:

Mrs Alouane Besma

University of Boumerdes

Academic Year: 2025 / 2026

Dedication

To our families,

whose love and patience carried us

through every late night and every setback.

Aya dedicates this work to her parents, who never stopped believing in her, to her brothers and sisters for their daily support, and to the friends who made these university years lighter.

Fella dedicates this thesis to her parents for their sacrifices, to her family for their constant encouragement, and to everyone who believed in this project when it was still only an idea.

Maram Meroua dedicates this work to her parents for their unconditional love, to her family for their patience throughout this year, and to her classmates with whom she shared this adventure.

To our professors at UMBB,

who taught us not only how to write code,

but how to think as engineers.

And to Sonatrach's Exploration Division team,

who trusted us with a real problem

and welcomed us as colleagues.

Aya, Fella, and Maram Meroua

Acknowledgements

We thank the faculty of the Department of Computer Science at UMBB for the academic foundation that made this project possible. We are grateful to Sonatrach's Exploration Division for welcoming us during the internship and providing access to the workflows and real data that shaped this system. Thanks also to the open-source communities behind Laravel, Tailwind CSS, and the other tools that powered our implementation.

Abstract

Keywords: Project Management System, Web Application, Laravel, MySQL, Role-Based Access Control, Sonatrach.

This report presents the design and implementation of a web-based **Project Management System (PMS)** built for Sonatrach's Exploration Division. The division previously managed its project and contract portfolio through Excel spreadsheets exchanged over e-mail, leading to data duplication, lack of traceability and zero real-time visibility for decision-makers.

The system replaces this workflow with a centralised, role-aware web platform covering the full lifecycle of projects and contracts, personnel and provider registries, organisational units, an immutable audit log, real-time in-app notifications, and exportable PDF/Excel reports backed by a KPI dashboard. It was built with Laravel 11 (PHP 8), MySQL 8, Tailwind CSS, and Alpine.js, following the MVC architectural pattern with four-tier Role-Based Access Control.

Glossary of Acronyms

Acronym	Meaning
PMS	Project Management System — the application built in this work.
PFE	Projet de Fin d'Etudes — final-year graduation project.
UMBB	Universite M'Hamed Bougara de Boumerdes.
RBAC	Role-Based Access Control — permissions are attached to roles, users inherit them through their roles.
MVC	Model–View–Controller — architectural pattern used by Laravel.
ORM	Object-Relational Mapper — here Laravel's Eloquent.
CRUD	Create, Read, Update, Delete — the four basic persistent storage operations.
3NF	Third Normal Form — normalisation level targeted by the schema.
PK	Primary Key — column uniquely identifying a row in a table.
FK	Foreign Key — column referencing a primary key in another table.
SQL	Structured Query Language — language used to query the relational database.
DB	Database.
DBMS	Database Management System — here MySQL 8.
API	Application Programming Interface.
REST	Representational State Transfer — architectural style for web APIs.
HTTP	HyperText Transfer Protocol — transport protocol of the web application.
HTTPS	HTTP Secure — HTTP layered over TLS.
URL	Uniform Resource Locator.
HTML	HyperText Markup Language.
CSS	Cascading Style Sheets.
JS	JavaScript.
AJAX	Asynchronous JavaScript And XML — technique for partial page updates.
JSON	JavaScript Object Notation — format used for notifications data column and API responses.
CSRF	Cross-Site Request Forgery — blocked by Laravel's VerifyCsrfToken middleware.
XSS	Cross-Site Scripting — blocked by Blade's automatic HTML escaping.
UI	User Interface — visible layer of the application.
UX	User Experience — usability properties of the interface.
UML	Unified Modeling Language — notation used for class and sequence diagrams.
KPI	Key Performance Indicator — metric shown on the executive dashboard.
PDF	Portable Document Format — export format for printable reports.
CSV	Comma-Separated Values — tabular export format.
IDE	Integrated Development Environment.
OS	Operating System.
OOP	Object-Oriented Programming.
MIME	Multipurpose Internet Mail Extensions — type system used to identify uploaded files.
LAMP	Linux + Apache + MySQL + PHP — classic web server stack.
LEMP	Linux + Nginx + MySQL + PHP — variant used in deployment.
SPA	Single-Page Application.
ID	Identifier — numeric primary key in the database.

Contents

Dedication ii

Acknowledgements iii

Abstract iv

Glossary of Acronyms v

List of Figures viii

General Introduction 1

1 Operational Context, System Diagnostics, and Project Rationale 2
- 1.1 Introduction . . . 2
- 1.2 Enterprise Profile: SONATRACH and the Exploration Division . . . 2
  - 1.2.1 Institutional Overview . . . 2
  - 1.2.2 The Data Management Department . . . 2
- 1.3 Diagnostics of the Existing System . . . 2
  - 1.3.1 Data Flow and Current Operations . . . 2
  - 1.3.2 Analysis of Operational Roles . . . 4
- 1.4 Problem Statement and Key Limitations . . . 4
- 1.5 Critique of the Previous System and Proposed Upgrades . . . 4
  - 1.5.1 Specific Critical Bottlenecks of the Previous Project . . . 4
  - 1.5.2 Our Proposed Technical Solutions . . . 5
- 1.6 System Requirements and Technical Upgrades . . . 5
  - 1.6.1 Functional Requirements . . . 6
  - 1.6.2 Non-Functional Requirements . . . 6

2 Analysis and Design of the Upgraded System 7
- 2.1 Introduction . . . 7
- 2.2 Functional and Behavioural Modelling (UML) . . . 7
  - 2.2.1 Identification and Definition of System Actors . . . 7
  - 2.2.2 Use Case Diagrams . . . 8
  - 2.2.3 Sequence Diagrams . . . 8
- 2.3 Structural and Data Modelling . . . 8
- 2.4 Class Diagram . . . 14
- 2.5 Business Rules . . . 15
- 2.6 From Conceptual Model to Relational Model . . . 15

3 System Implementation and Deployment 17
- 3.1 Introduction . . . 17
- 3.2 Development Environment . . . 17

3.2.1 Modelling Tool 17
3.2.2 Development Languages and Environments 17
3.2.3 Technology Choices and Justifications 17
3.3 Application Interfaces 18
3.3.1 Authentication 18
3.3.2 Role-Specific Dashboards 18
3.3.3 Project Management 21
3.3.4 Contract Management 22
3.3.5 Master Data 23
3.3.6 Audit, Notifications, Reports and Profile 23

General Conclusion **29**

Bibliography **29**

List of Figures

1.1 Data flow diagram of the existing system 3

2.1 Global Use Case Diagram 9
2.2 Super Admin Use Case Diagram 10
2.3 Authentication Sequence Diagram 11
2.4 Contract Notification Sequence Diagram 11
2.5 Project Notification Sequence Diagram 12
2.6 Class Diagram 14

3.1 Sign-in page — Sonatrach PMS 18
3.2 Super Admin dashboard — full KPIs and master-data menu 19
3.3 Admin dashboard — contract lifecycle focus 19
3.4 Project Manager dashboard — project ownership view 20
3.5 Engineer dashboard — read-only, projects-only sidebar 20
3.6 Projects index with filters and status badges 21
3.7 Project creation form 21
3.8 Contracts index — lifecycle and value at a glance 22
3.9 Contract creation form (Admin role) 22
3.10 Providers directory 23
3.11 Staff directory with Suspend/Reactivate actions 24
3.12 Directorates 24
3.13 Departments 25
3.14 User accounts and role assignments 25
3.15 Audit log (Super Admin only) 26
3.16 Notifications inbox 26
3.17 Reports and exports 27
3.18 User profile management 27

General Introduction

Sonatrach's Exploration Division runs dozens of projects at the same time, each with its own contracts, providers and engineering teams. When we arrived for our internship, all of this was tracked through printed forms, shared Excel files, and long e-mail threads. The same data lived in three or four different files at once, mistakes were hard to trace back to anyone, and managers had no quick way to see the state of the portfolio.

Our project replaces that workflow with a web application. The point is not just to put the existing forms on a screen, but to rebuild the process around a proper relational database, role-based access, and an automatic audit trail.

The report is structured in three chapters:

- **Chapter 1** — Host organisation, existing system analysis, and problem statement.
- **Chapter 2** — Requirements, UML design (use cases, class diagram, sequence diagrams), and system architecture.
- **Chapter 3** — Technology choices with justifications, development environment, and application interfaces.

Chapter 1

Operational Context, System Diagnostics, and Project Rationale

1.1 Introduction

In the contemporary energy sector, the efficiency of managing large-scale infrastructure projects and technical contracts directly influences an organisation's operational success. The massive volume, diversity, and strategic value of data generated during hydrocarbon exploration necessitate automated, integrated, and secure enterprise systems.

This chapter establishes the core framework of our study. It outlines the structural layout of the host company, SONATRACH, examines the current data flow and personnel roles within its Exploration Division, diagnoses critical technical bottlenecks in the existing system, and delivers the rationale behind our upgraded web-based management platform.

1.2 Enterprise Profile: SONATRACH and the Exploration Division

1.2.1 Institutional Overview

Established in 1963, SONATRACH (*Société Nationale pour la Recherche, Production, Transport, Transformation, et Commercialisation des Hydrocarbures*) is the national state-owned oil and gas corporation of Algeria and a premier energy enterprise globally. The company operates across the complete hydrocarbon value chain, spanning upstream exploration, midstream transport, and downstream refining.

1.2.2 The Data Management Department

This project was carried out within the Exploration Division in Boumerdès, specifically inside the **Data Management Department**. This department acts as the primary technical backbone for the division, tasked with organising, archiving, and maintaining massive geological and geophysical datasets while providing software support to exploration engineers and decision-makers.

1.3 Diagnostics of the Existing System

1.3.1 Data Flow and Current Operations

The day-to-day operations within the Exploration Division are centred around two fundamental business elements:

- **Prospection Projects:** Long-term scientific initiatives, such as 2D/3D seismic data acquisition and exploratory drilling operations.
- **Technical Contracts:** Legal, financial, and technical agreements signed with foreign operators, service providers, or corporate subsidiaries.

As illustrated in Figure 1.1, the Data Flow Diagram highlights the direct interactions between the different actors (Engineers, Project Managers, and Admins). It clearly shows how data validation relies heavily on manual file exchanges, which creates the performance gaps analysed in the problem statement.

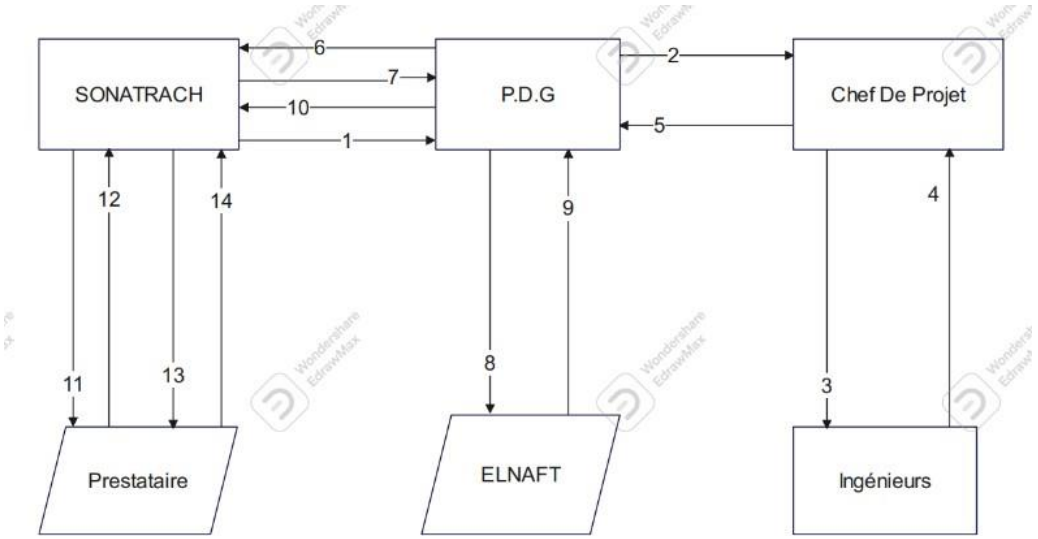


Figure 1.1: Data flow diagram of the existing system

Information Flow Description:

1. Request to create a new project.
2. Project commitment document.
3. Affiliation document.
4. Monthly report.
5. Progress assessment.
6. Confirmation of progress.
7. Request for a new Hydrocarbon Exploration and Production Contract.
8. Hydrocarbon Exploration and Production Contract, Technical Document.
9. Contract signing.
10. Contract approval.
11. Seismic acquisition contract.
12. Seismic report.
13. Drilling contract.
14. Monthly report.

Currently, data tracking follows a fragmented, semi-manual process. Project parameters, budget allocations, and contractual milestones are primarily maintained on localised spread-

sheets (e.g., Microsoft Excel) and manually updated and transmitted between departments via internal emails.

1.3.2 Analysis of Operational Roles

The current manual workflow divides responsibilities across four primary organisational tiers:

- **Super Administrators (Upper Management):** Oversee final validations, corporate policy implementation, and macro-resource allocation.
- **Administrators (Department Managers):** Responsible for managing contract documents, reviewing deliverables, and coordinating budgets.
- **Project Managers:** Directly monitor localised field projects, ensure timeline compliance, and compile raw progress metrics.
- **Exploration Engineers:** Technical operators who execute geological analysis, interface with specialised industry software, and query project data.

1.4 Problem Statement and Key Limitations

An empirical evaluation of the existing workflow reveals significant structural bottlenecks that hinder administrative efficiency and data reliability. The core limitations include:

- **Data Inconsistency and Redundancy:** The reliance on scattered, disconnected spreadsheet files across multiple departments leads to version mismatch and destroys data synchronisation.
- **Susceptibility to Human Error:** Manual entry of monetary amounts, contract durations, and complex contract variations poses risks of administrative errors and financial calculation discrepancies.
- **Information Silos:** Historical exploration files and archived contracts are difficult to access, requiring time-consuming manual sorting through nested unindexed folders.
- **Lack of Real-Time Monitoring:** Upper management cannot rapidly obtain a comprehensive, real-time visualisation of all ongoing project statuses or approaching contract deadlines.

1.5 Critique of the Previous System and Proposed Upgrades

1.5.1 Specific Critical Bottlenecks of the Previous Project

Beyond the general workflow limitations listed above, a deeper review of the legacy platform exposed two structural defects that directly motivated the redesign described in the following chapters.

Absence of an Automated Alert Mechanism. The primary operational flaw in the previous system was the total absence of a communication bridge between the system actors. When an Admin initiated or updated a contract, there was no automated mechanism to notify the Super Admin. The workflow relied entirely on manual follow-ups or external notifications, leading to critical delays in contract validation, lack of coordination, and complete isolation between the structural roles of the platform.

Flawed Authorisation and Exposure of Sensitive Corporate Data. The older architecture treated highly confidential information — such as financial contract amounts, geological project objectives, and technical scopes — as non-sensitive data. Due to an inadequate implementation of Role-Based Access Control (RBAC), data visibility boundaries were blurred. Consequently, unauthorised actors and low-privilege profiles could easily access, view, or potentially alter critical contract and project parameters that should have been strictly restricted to upper management.

1.5.2 Our Proposed Technical Solutions

To resolve these critical bottlenecks and ensure a highly secure, collaborative environment for SONATRACH's Exploration Division, our upgraded platform implements the following engineering solutions:

Implementation of an Event-Driven Asynchronous Mediator. We completely re-designed the behavioural model of the system by introducing a dedicated `Notification` object that acts as an architectural mediator. When an Admin performs a high-level action (e.g. `confirmContract()`), the system automatically registers the event and dispatches real-time, non-blocking alerts directly to the Super Admin's dashboard. This eliminates the need for manual follow-ups, synchronises the workflow, and ensures seamless, decoupled interaction between administrative roles.

Enforcement of Strict Role-Based Access Control and Data Isolation. To safeguard confidential corporate assets, we implemented a robust Role-Based Access Control (RBAC) security framework. Highly sensitive attributes — such as contract financial amounts, scope IDs, and strategic project objectives — are now strictly isolated and encrypted at the database level. Visibility and modification privileges are mathematically bounded; unauthorised actors (such as general Engineers or external profiles) are completely restricted from accessing these views, ensuring that only high-privilege lifelines (Admin and Super Admin) can interact with critical contract data.

1.6 System Requirements and Technical Upgrades

To optimise this workspace, the proposed solution transitions the division from file-based tracking to a centralised enterprise environment. The structural and operational expectations are categorised below.

1.6.1 Functional Requirements

- **Centralised Data Engine:** A unified relational database structure linking all entities (Directorates, Departments, Projects, Contracts, Providers, and Staff) to guarantee immediate data consistency and eliminate orphan records.
- **Role-Based Access Control (RBAC):** Implementation of an explicit security matrix defining privileges: full access for Super Admins, contract mutation rights for Admins, tracking rights for Project Managers, and read-only access for Engineers.
- **Dynamic Notification Framework:** An automated internal alert system to facilitate cross-department communication regarding contract validations, project updates, and deadlines.
- **Analytical Reporting Tool:** Multi-criteria filtering mechanisms (by launching year, department, budget, or software) to generate rapid statistical charts for decision-makers.

1.6.2 Non-Functional Requirements

- **Security & Access Control:** Strict cryptographic hashing of user passwords and session management to safeguard critical national asset data.
- **Usability & Ergonomics:** A responsive, intuitive, and modern web interface respecting SONATRACH's corporate visual identity to minimise data-entry strain.

Conclusion and Proposed Solution

In response to the identified system deficiencies, this project presents an enhanced, web-based management platform. By converting manual processes into an integrated, role-secured digital workflow, this solution removes data redundancy, automates contract lifecycle tracking, and provides real-time decision-support tools. Ultimately, this platform advances SONATRACH's digital transformation initiatives, strengthening administrative transparency and operational efficiency within the Exploration Division.

Chapter 2

Analysis and Design of the Upgraded System

2.1 Introduction

Following the comprehensive requirement analysis and diagnostics of the existing workflow within SONATRACH's Exploration Division established in Chapter 1, this chapter shifts the focus toward the architectural layout and engineering design of the upgraded platform. System design serves as the crucial blueprint that bridges theoretical institutional needs with actual programmatic implementation.

To provide a rigorous structural foundation, this chapter begins by defining the global multi-tier architecture of the proposed web application. It then transitions into functional and behavioural modelling using the Unified Modeling Language (UML), explicitly mapping system actors, global use cases, and dynamic sequence workflows. Furthermore, the structural and data dimensions are examined through a detailed class diagram and defined corporate management rules. Finally, the conceptual object framework is translated into a structured Logical Data Model (LDM), establishing the foundational relational database tables required to ensure absolute data integrity and system security.

2.2 Functional and Behavioural Modelling (UML)

To transform the structural requirements of SONATRACH's Exploration Division into a dynamic web platform, it is essential to model the behavioural interactions between the users and the system. This section utilises the Unified Modeling Language (UML) to define the functional boundaries of the application, starting with the identification of system actors and progressing to the global use case representation.

2.2.1 Identification and Definition of System Actors

An actor represents a coherent set of roles that users play when interacting with the system's use cases. Based on the corporate hierarchy and the business rules established for project and contract management, the platform distinguishes four primary human actors:

Super Administrator (Super Admin). *Definition:* This actor represents upper management or the system's root controller within the Data Management Department. The Super Admin possesses absolute administrative privileges. *Responsibilities:* Managing organisational staff accounts, granting or revoking access control levels, creating service providers, and performing final validations and ultimate archiving of both prospection projects and exploration contracts.

Administrator (Admin). *Definition:* This actor represents the department managers or contract moderators who oversee the financial and legal aspects of exploration operations.

Responsibilities: Initialising new technical contracts, modifying contract parameters, uploading scanned physical contract documents, confirming operational milestones, and broadcasting system notifications to related personnel.

Project Manager (Chef de Projet). *Definition:* This actor represents the operational leads responsible for supervising specific field activities, such as 2D/3D seismic data acquisition or drilling campaigns. *Responsibilities:* Monitoring local project progression, updating project execution statuses, reviewing contract terms related to their assigned tasks, and modifying project parameters to match field realities.

Exploration Engineer (Engineer). *Definition:* This actor represents the standard technical end-users and geoscientists within the division who manipulate raw exploration data. *Responsibilities:* Querying the database to view project details and reading internal system alerts and updates without possessing any data-mutation or destructive editing privileges.

2.2.2 Use Case Diagrams

To visually capture the functional boundaries of the upgraded platform, a global Use Case Diagram has been established. This diagram serves as a high-level blueprint that maps the interactions between the previously identified actors and the core system functionalities. By outlining which organisational roles possess permissions for specific data mutations — such as contract initialisation, project validation, or notification reading — the diagram ensures strict adherence to the company's security policies and role-based access controls.

The complete functional interaction model is illustrated in Figure 2.1 below.

2.2.3 Sequence Diagrams

While the use case diagram defines the static functional capabilities of each actor, it does not depict how these operations execute over time. To capture the dynamic behaviour of the system, this section presents detailed Sequence Diagrams for the core structural workflows. These diagrams model the chronological exchange of messages, validations, and data queries between the user interface, the backend logic, and the relational database, ensuring a seamless and reliable automated process.

2.3 Structural and Data Modelling

Having mapped the dynamic and behavioural interactions of the system through UML sequence diagrams, it is necessary to establish the underlying structural layout of the application's information. This section defines the static data framework, beginning with a rigorous data dictionary that formalises every attribute, followed by the explicit management rules and relational database schematics required to ensure data persistence within the division.

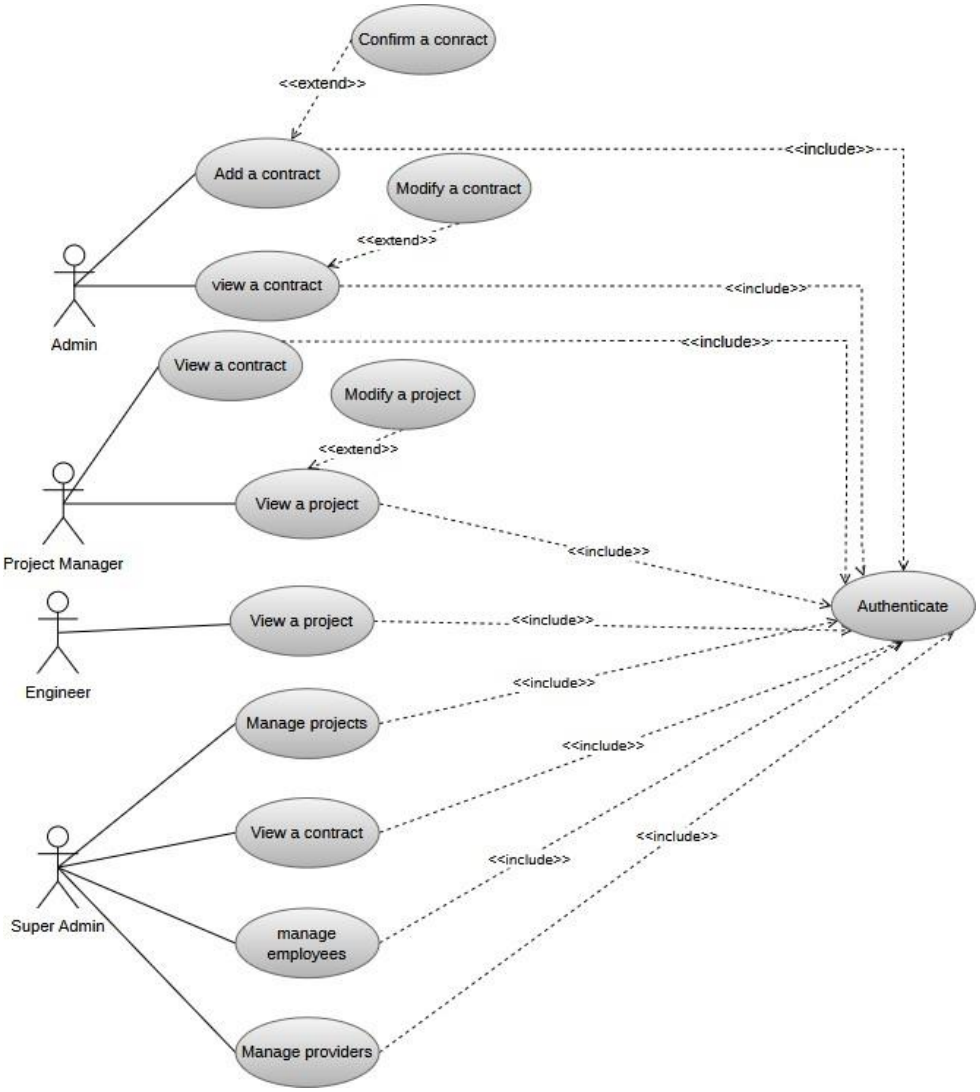


Figure 2.1: Global Use Case Diagram

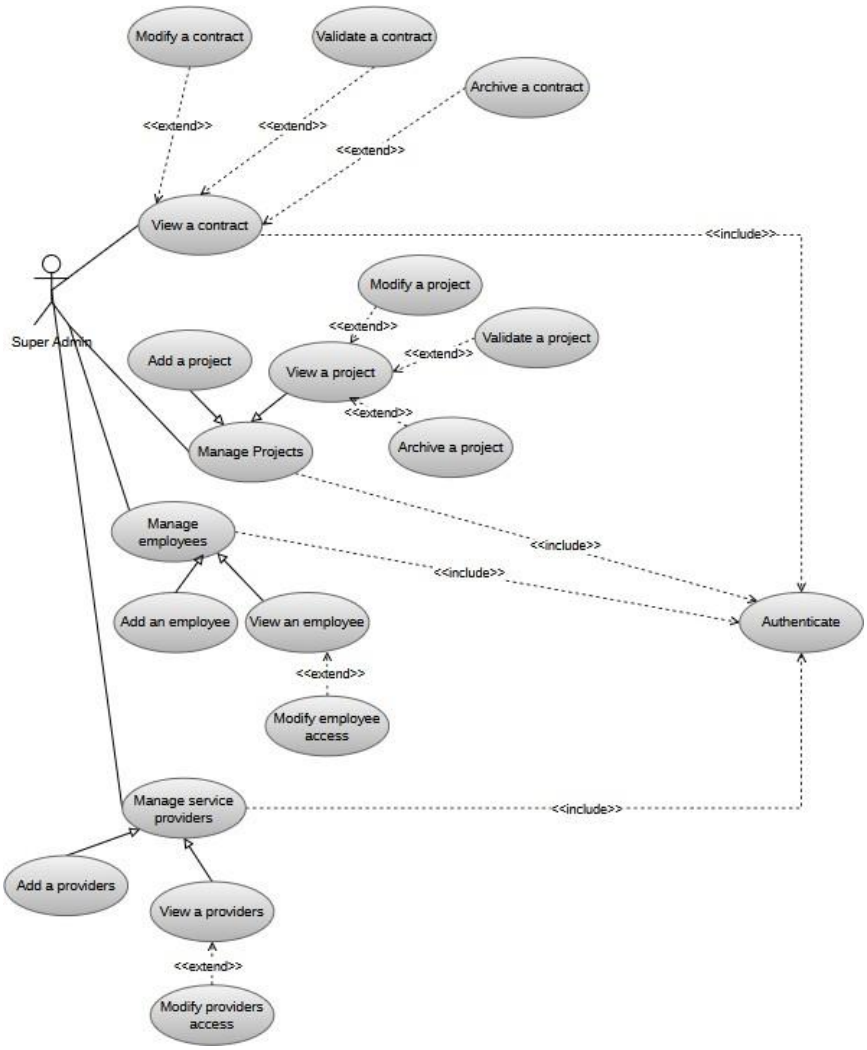


Figure 2.2: Super Admin Use Case Diagram

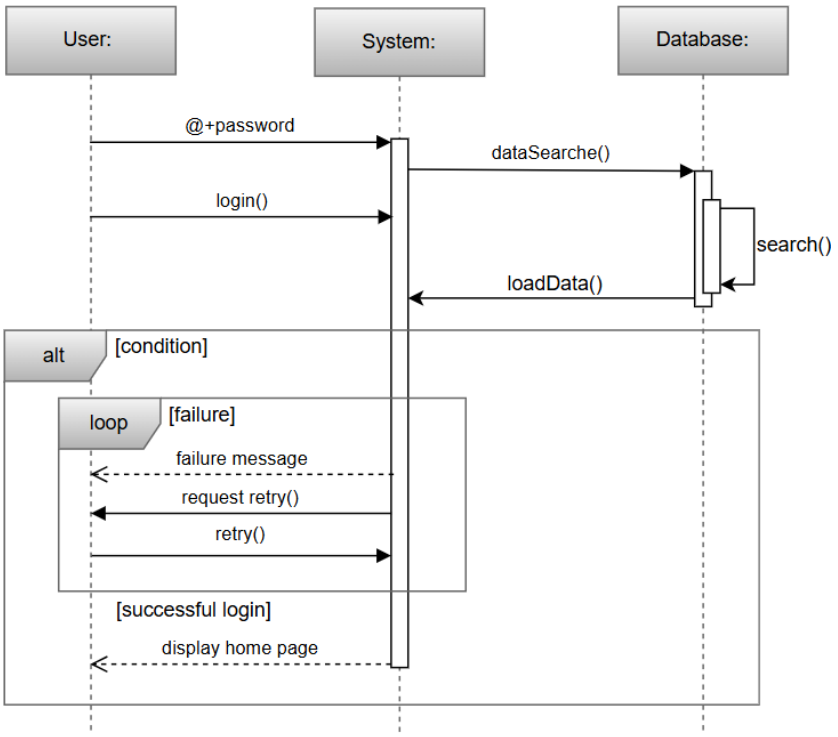


Figure 2.3: Authentication Sequence Diagram

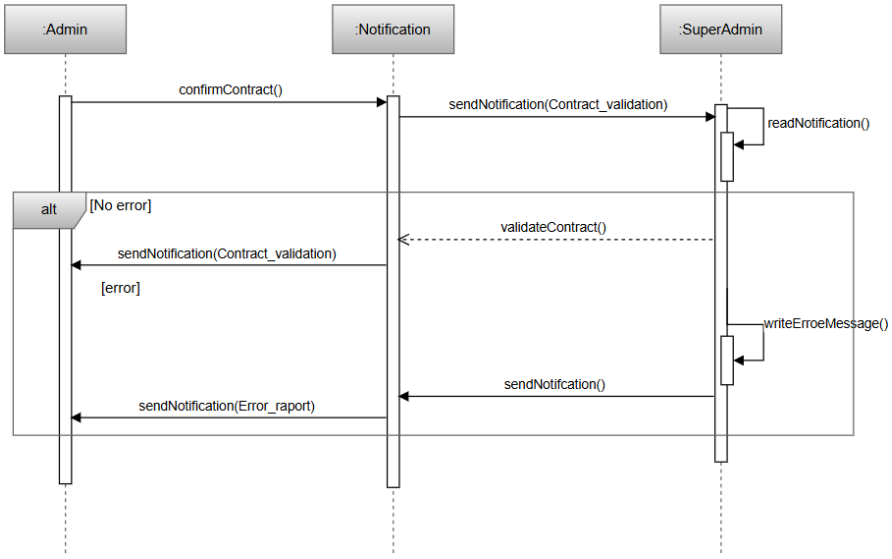


Figure 2.4: Contract Notification Sequence Diagram

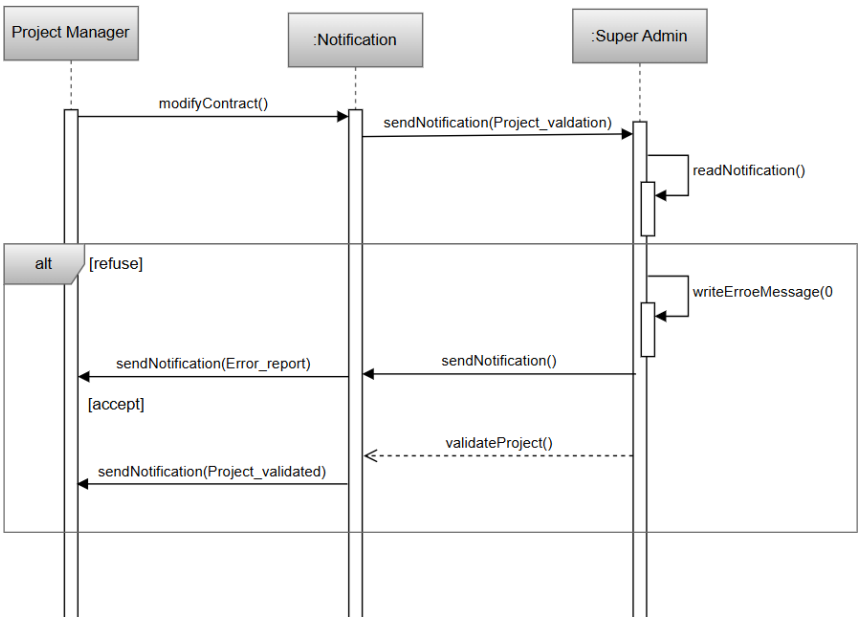


Figure 2.5: Project Notification Sequence Diagram

Table 2.1: Data dictionary

N	Code	Description	Type	Len	Constraints
01	Staff_id	Unique identifier for staff members	N	4	PK, NOT NULL
02	Staff_last_name	Last name of the staff member	Alph	50	NOT NULL
03	Staff_first_name	First name of the staff member	Alph	50	NOT NULL
04	Staff_category	Staff group or role category	Alph	30	NOT NULL
05	Staff_expertise	Domain of technical expertise	Alph	50	
06	Staff_pc_num	Staff personal computer or asset number	N	10	
07	Staff_off_num	Office room or telephone extension number	N	10	
08	Username	Account login username	Alph	30	NOT NULL, UNIQUE
09	Password	Cryptographically hashed password string	Alph	255	NOT NULL
10	Department_id	Unique identifier for the department	N	4	PK, NOT NULL
11	Department_name	Name of the specific department	Alph	60	NOT NULL
12	Directorate_id	Unique identifier for the directorate	N	4	PK, NOT NULL
13	Directorate_name	Name of the directorate	Alph	60	NOT NULL
14	Contract_id	Unique tracking number for the technical contract	N	4	PK, NOT NULL
15	Contract_duration	Scheduled total duration or expiration boundary	Date	10	NOT NULL

N	Code	Description	Type	Len	Constraints
16	Execution_Duration	Actual dynamic field execution timeframe	Alph	30	NOT NULL
17	Amount	Financial monetary value of the contract	N	15	NOT NULL
18	Scope_id	Original primary work scope identifier	N	4	PK, NOT NULL
19	Scope_add	Address of the scope	Alphn	100	NOT NULL
20	Contract_title	Descriptive corporate title of the contract	Alphn	150	NOT NULL
21	Contract_status	Status state string (e.g., Initialised, Confirmed)	Alph	20	NOT NULL
22	Contract_scan	File path URL string of the physical scanned document	Alphn	255	
23	Scan_date	Date metadata when the contract PDF was uploaded	Date	10	
24	Decision	Validation status	Alph	20	
25	Provider_contract_type	Type of provider contract	Alph	30	NOT NULL
26	Project_id	Unique identification code for exploration project	N	4	PK, NOT NULL
27	Project_title	Title of the project	Alphn	150	NOT NULL
28	Project_objective	The objective of the project	Alphn	255	
29	Start_date	Official project launching date entry	Date	10	NOT NULL
30	End_date	Projected completion or target deadline date	Date	10	
31	Project_status	Tracking status phrase (e.g., Active, Archived)	Alph	20	NOT NULL
32	Provider_id	Provider identifier	N	4	PK, NOT NULL
33	Provider_name	Provider's name	Alph	100	NOT NULL
34	Notification_id	Unique code for notification	N	4	PK, NOT NULL
35	Notification_Title	Title of the notification	Alph	100	NOT NULL
36	Link	The link	Alphn	255	
37	Message	Detailed alert descriptive textual content payload	Alphn	500	NOT NULL
38	Notification_type	Functional category classification instance	Alph	30	NOT NULL
39	Notification_Status	Viewing status state attribute tracker	Alph	20	NOT NULL
40	Created_at	Precise system timestamp database generation entry	Date	10	NOT NULL
41	Sender_id	Relational reference mapping to originator user account	N	4	FK → Staff_id, NOT NULL
42	Receiver_id	Relational reference mapping to target destination account	N	4	FK → Staff_id, NOT NULL
43	Related_contract_id	Relational bridge identifier linked to contract	N	4	FK → Contract_id

N	Code	Description	Type	Len	Constraints
44	Related_project_id	Relational bridge identifier linked to project	N	4	FK → Project_id

2.4 Class Diagram

To consolidate the structural entities and their interrelationships derived from the data dictionary and the defined business rules, a UML Class Diagram has been constructed. This diagram provides a comprehensive static view of the system's object model, explicitly representing all core classes (Staff, Department, Directorate, Project, Contract, Provider, and Notification), their respective attributes, and the cardinality of their associations. It serves as the direct conceptual bridge between the analytical specification phase and the logical database design established in the subsequent section.

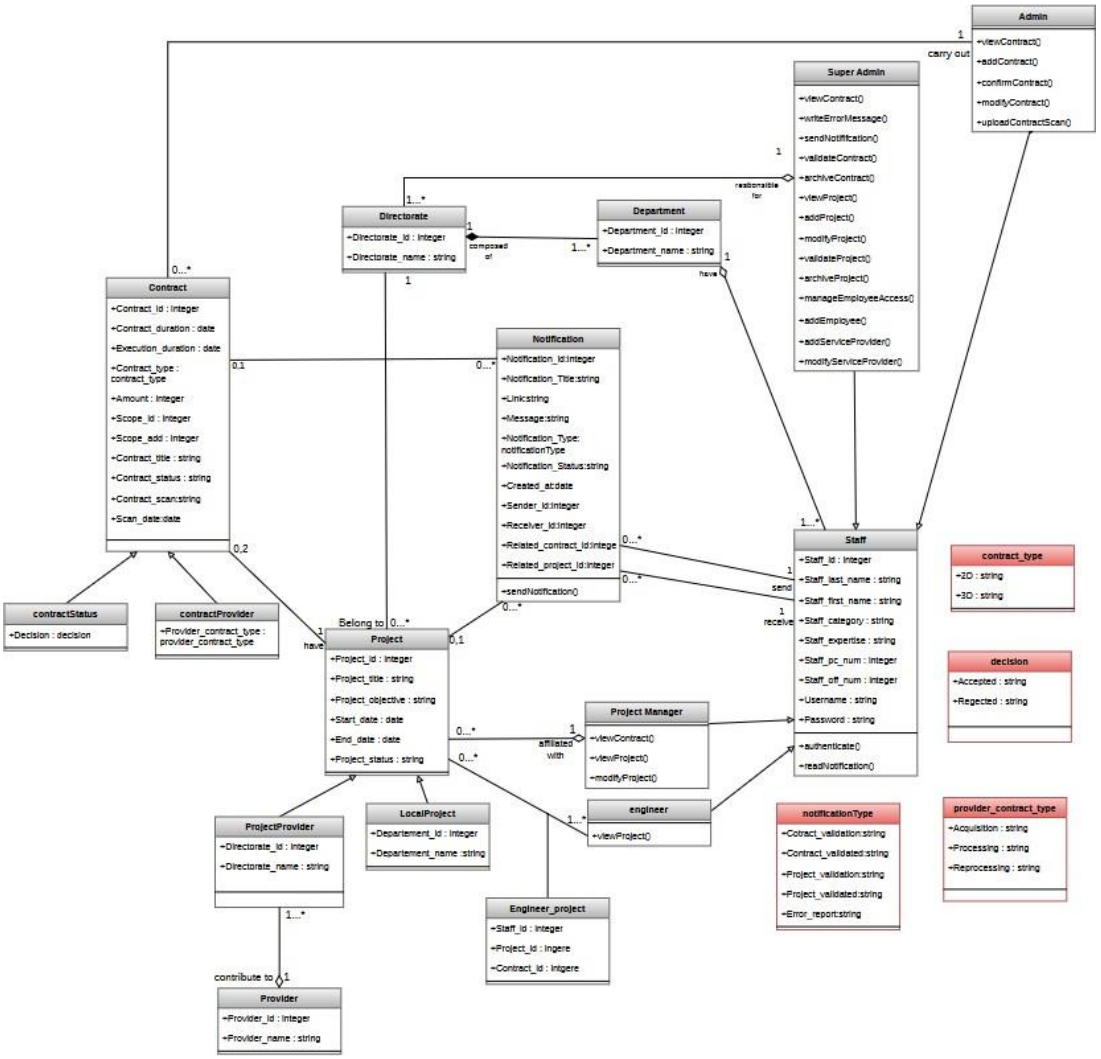


Figure 2.6: Class Diagram

2.5 Business Rules

Following the UML modelling of our application through the diagrams presented above, the next step consists of establishing the relational database. This transition is guided by a set of business rules that govern the passage from the conceptual model to the relational model, and which are defined as follows:

R1:
A directorate is composed of several departments.

R2:
Every directorate is managed by a Super Administrator.

R3:
Each department includes several staff members.

R4:
An administrator can manage multiple contracts.

R5:
A project belongs to a directorate.

R6:
A directorate can have multiple projects.

R7:
Each contract is affiliated with its corresponding project.

R8:
A project can only have either 2 contracts or no contract.

R9:
A project is carried out by multiple engineers and one project manager.

R10:
The project manager and engineers can be assigned to multiple projects.

R11:
A provider project involves one service provider.

R12:
A provider contract includes one service provider.

2.6 From Conceptual Model to Relational Model

Applying the business rules defined above to the conceptual class diagram, the following relational schema has been derived. Primary key attributes are underlined and foreign key references are denoted by the # prefix:

- **Staff** (Staff_id, Staff_last_name, Staff_first_name, Staff_category, Staff_expertise, Staff_pc_num, Staff_off_num, Username, Password, #Department_id)
- **Department** (Department_id, Department_name, #Directorate_id)
- **Directorate** (Directorate_id, Directorate_name, #Staff_id)

- **LocalProject** (Project_id, Project_title, Project_objective, Start_date, End_date, Project_Status, #Department_id, #Staff_id)
- **ProjectProvider** (Project_id, Project_title, Project_objective, Start_date, End_date, Project_Status, #Directorate_id, #Provider_id, #Staff_id)
- **Provider** (Provider_id, Provider_name)
- **ContractStatus** (Contract_id, Contract_duration, Execution_Duration, Contract_type, Amount, Scope_id, Scope_add, Contract_title, Contract_status, Decision, #Project_id)
- **ContractProvider** (Contract_id, Contract_duration, Execution_Duration, Contract_type, Amount, Scope_id, Scope_add, Contract_title, Contract_status, Provider_contract_type, #Project_id)
- **Engineer_project** (#Staff_id, #Project_id, #Contract_id)
- **Notification** (Notification_id, Notification_Title, Link, Message, Notification_Type, Notification_Status, Created_at, #Sender_id, #Receiver_id, #Related_contract_id, #Related_project_id)

Chapter Conclusion

In conclusion, this chapter has successfully established the complete analytical and structural framework for the upgraded contract and project management platform within SONATRACH's Exploration Division. By transitioning from behavioural modelling using UML sequence diagrams to static data architecture, we have mapped out a precise and automated system workflow.

Chapter 3

System Implementation and Deployment

3.1 Introduction

This chapter details the technical development environment, including software configurations, followed by the justification of the selected programming languages, frameworks, and database technologies. Furthermore, it presents the main user interfaces, role-based dashboards, and the core notification workflows. Finally, it outlines the testing and validation procedures conducted to ensure system performance, data security, and stability.

3.2 Development Environment

3.2.1 Modelling Tool

Draw.io (Diagrams.net): It is a modern, web-based diagramming application used for creating visual software architectures. Within this project, it was utilised to construct the functional UML diagrams (such as Use Case and Sequence diagrams) and static data structures (Class diagram) due to its compliance with standard UML notations and cross-platform flexibility.

3.2.2 Development Languages and Environments

The platform was built using the following stack of languages, frameworks and tools, summarised in Table 3.1.

Table 3.1: Development languages, frameworks and tooling

Tool	Version	Role
PHP	8.5	Server runtime
Laravel	11.x	Application framework
MySQL	8.0	Relational database
Node.js	20	Frontend build tooling
Vite	5.x	Asset bundler
Composer	2.x	PHP package manager
VS Code	—	Primary IDE
Git	2.x	Version control

3.2.3 Technology Choices and Justifications

Laravel 11. Laravel was chosen because its Eloquent ORM eliminates boilerplate SQL while producing parameterised queries by default — the safest option for a system handling sensitive contract data. Its built-in Form Request validation centralises input sanitisation in one place per feature, and its middleware pipeline enforces RBAC at the routing layer without

scattering permission checks across controllers. The artisan CLI and migration system also made database schema iteration fast throughout development.

MySQL. MySQL was preferred over a document store because the domain is heavily relational: projects link to contracts link to providers link to staff. Foreign key constraints at the database level enforce referential integrity regardless of which code path reaches the database, while indexed join columns keep list queries fast as the dataset grows.

3.3 Application Interfaces

This section presents the implemented interfaces of the upgraded Sonatrach PMS, captured from the running Laravel application against the seeded demonstration dataset. Each screen reflects the role-based access rules defined in Chapter 2: the visible navigation items, lifecycle action buttons and master-data sections all depend on the authenticated user's role.

3.3.1 Authentication

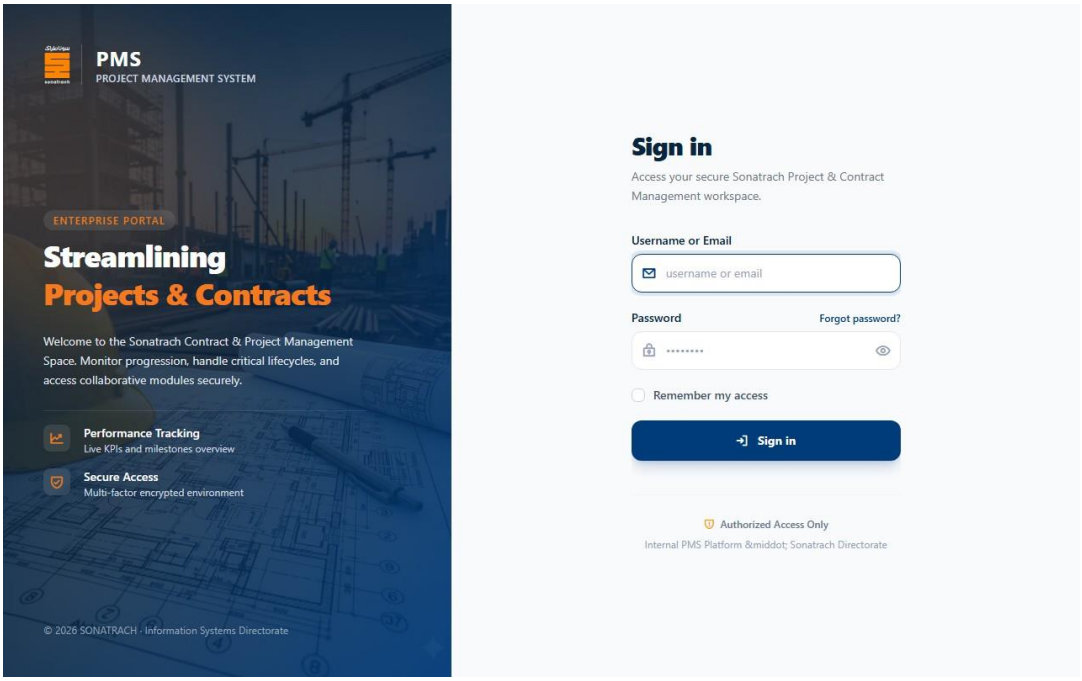


Figure 3.1: Sign-in page — Sonatrach PMS

Description. Figure 3.1 shows the single entry point to the platform. The form accepts either a username or an institutional e-mail address. Credentials are validated server-side, the session is regenerated on success, and any unauthenticated request to a protected route is redirected back to this page.

3.3.2 Role-Specific Dashboards

Description. The Super Admin dashboard (Figure 3.2) exposes the complete set of KPIs (projects, contracts, providers, users) together with the full left-side master-data menu

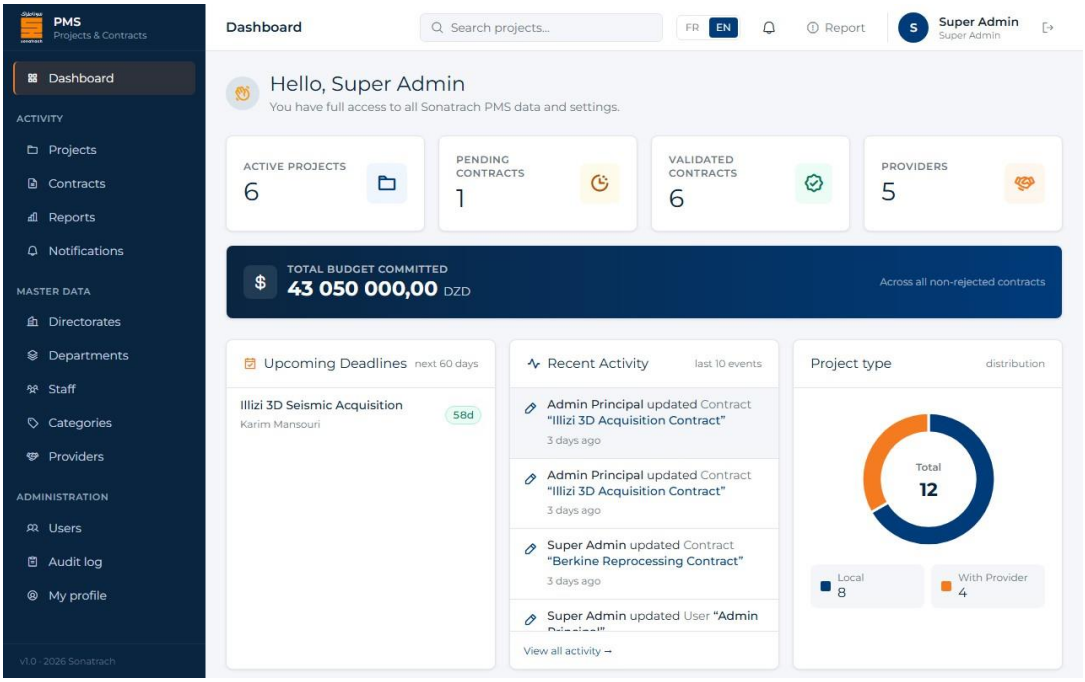


Figure 3.2: Super Admin dashboard — full KPIs and master-data menu

(Directorates, Departments, Staff, Categories, Providers) and the Administration section.

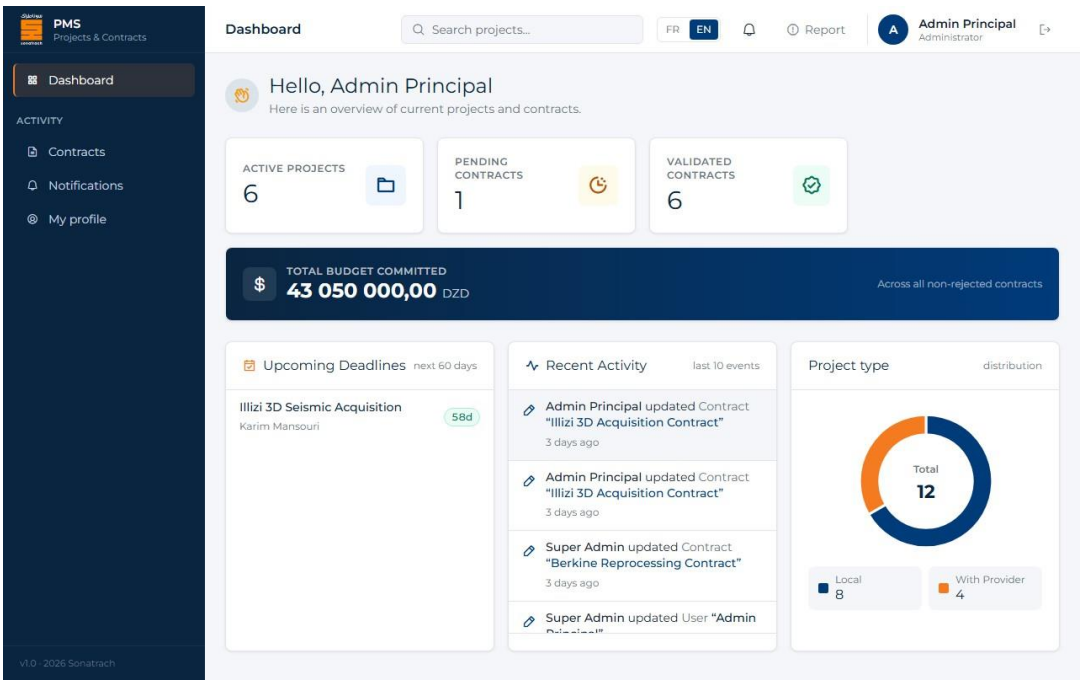


Figure 3.3: Admin dashboard — contract lifecycle focus

Description. The Admin role (Figure 3.3) sees only the operational entries required to manage the contract lifecycle: dashboard, contracts and notifications. Master-data and Administration menus are hidden because the permission matrix grants no access.

Description. The Project Manager dashboard (Figure 3.4) emphasises ownership: only the projects led by the connected user are surfaced, along with the contracts that depend on them.

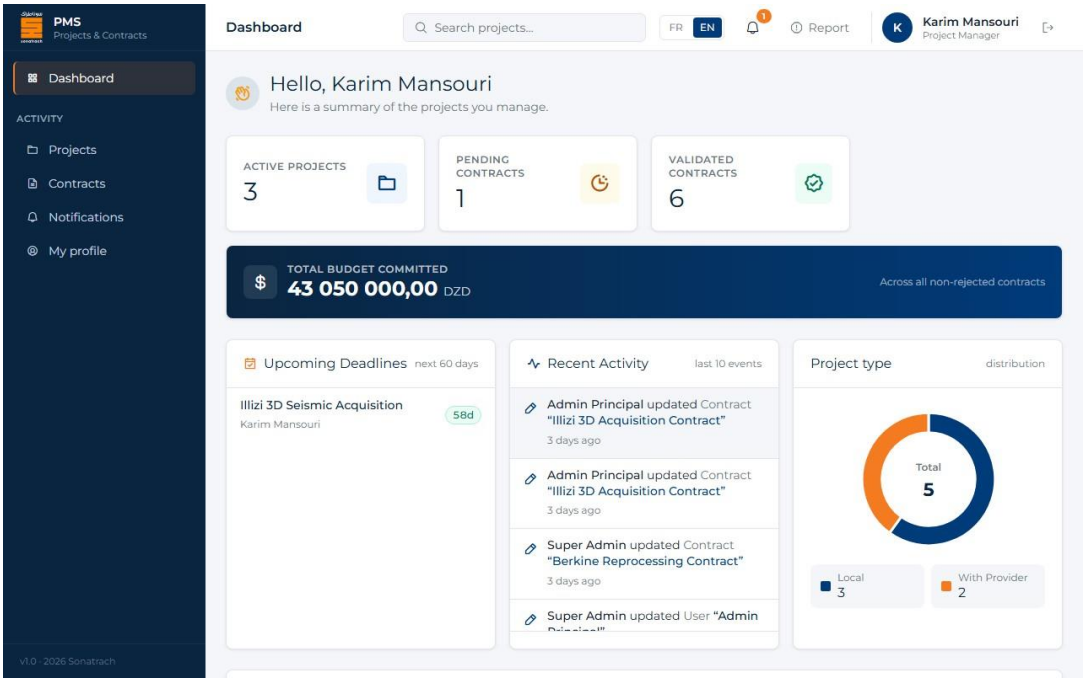


Figure 3.4: Project Manager dashboard — project ownership view

Master data remains hidden.

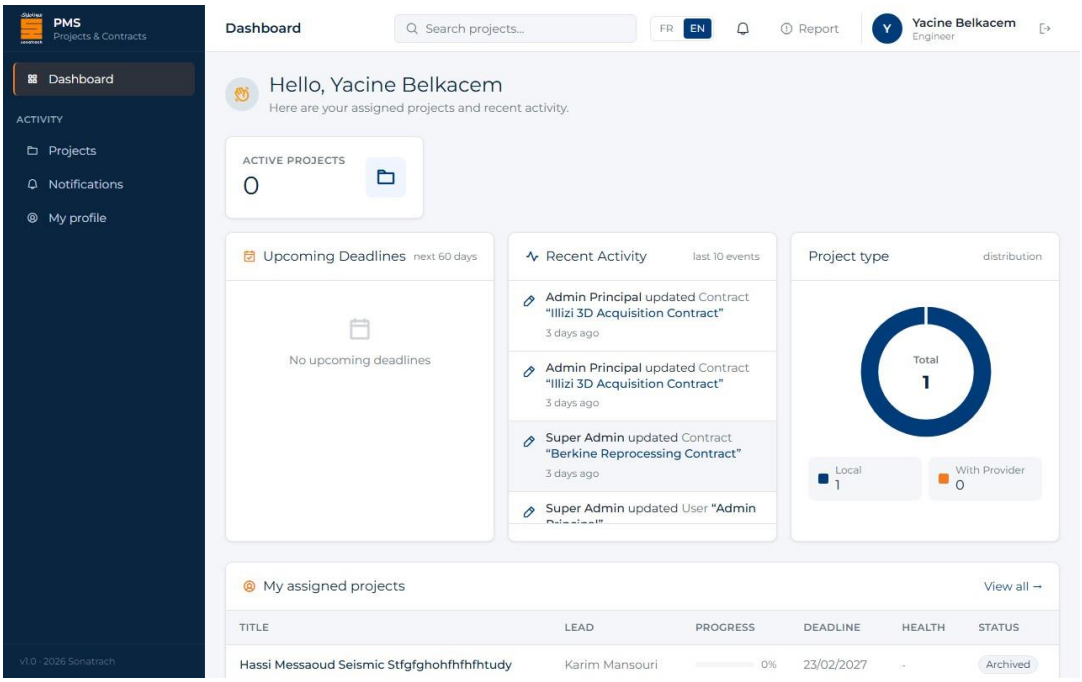


Figure 3.5: Engineer dashboard — read-only, projects-only sidebar

Description. The Engineer view (Figure 3.5) is intentionally minimal: a read-only summary of the projects to which the engineer has been assigned. No creation, validation or archival action is reachable.

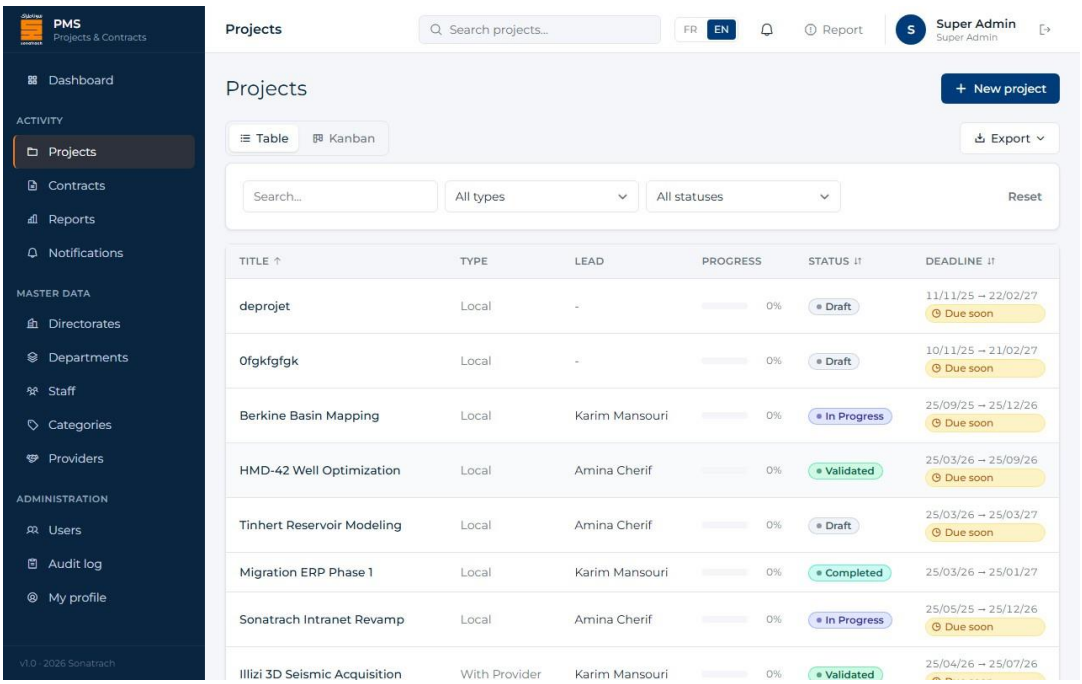


Figure 3.6: Projects index with filters and status badges

3.3.3 Project Management

Description. The project list (Figure 3.6) provides server-side search, filtering by kind and status, and colour-coded status badges. The “New project” button is only rendered for roles holding the projects.add permission.

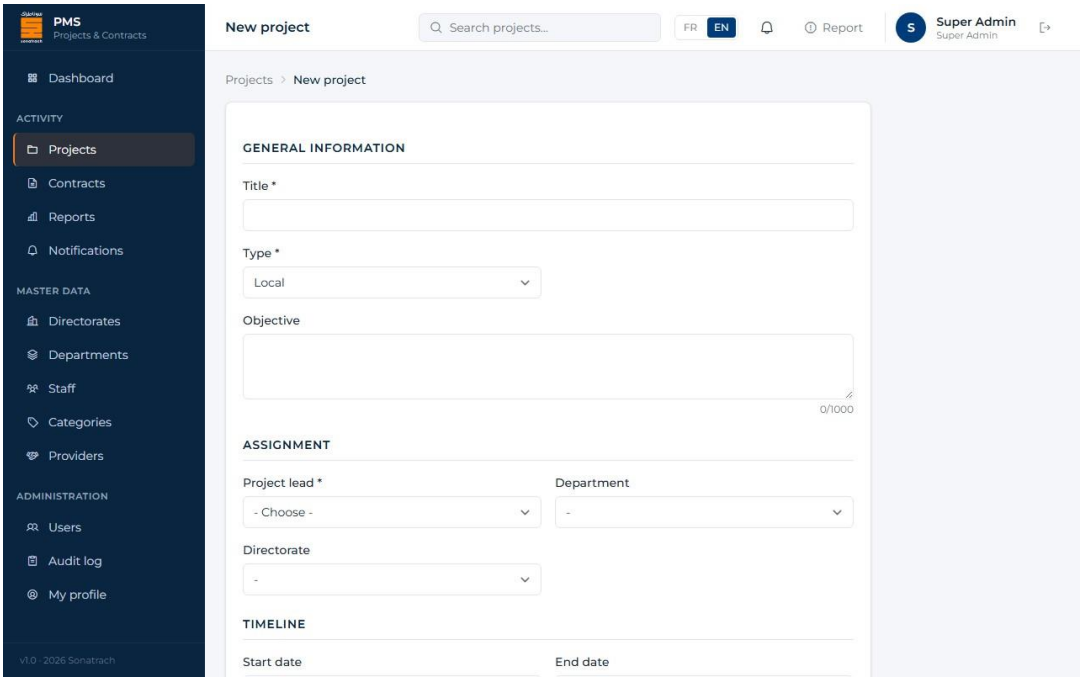


Figure 3.7: Project creation form

Description. The creation form (Figure 3.7) groups the project’s identifying attributes (title, kind, objective, dates) and its organisational anchor (directorate or department, lead user).

All validation rules from Chapter 2 are enforced on submission.

3.3.4 Contract Management

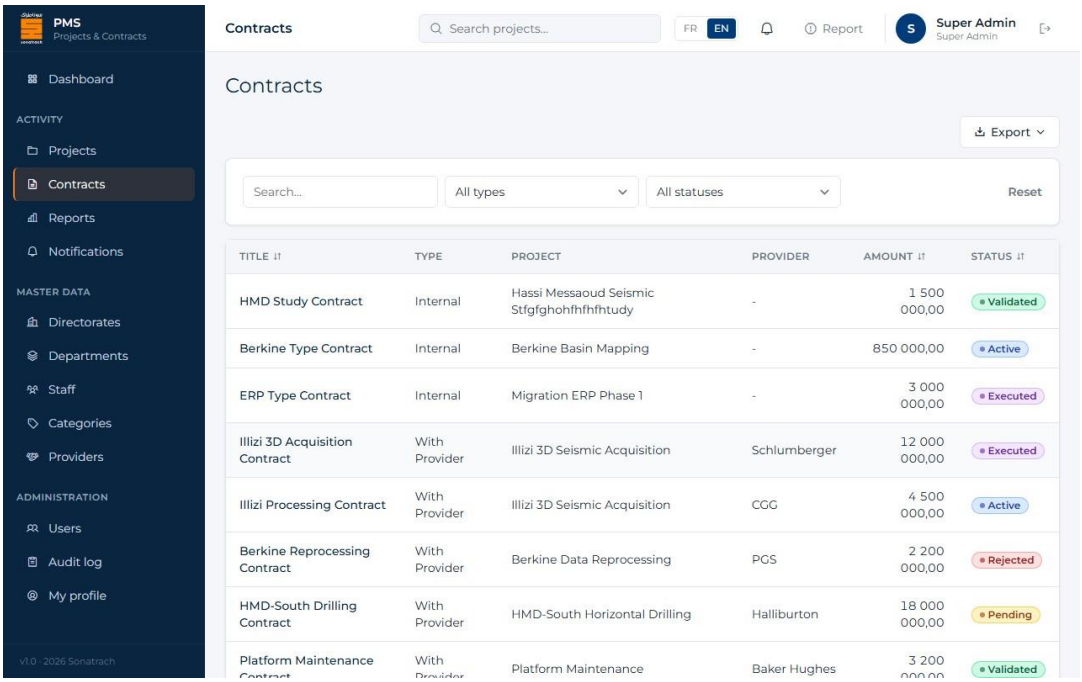


Figure 3.8: Contracts index — lifecycle and value at a glance

Description. The contracts index (Figure 3.8) lists every contract with its parent project, monetary value, type and current lifecycle status. Filters allow quick narrowing by status, kind or provider.

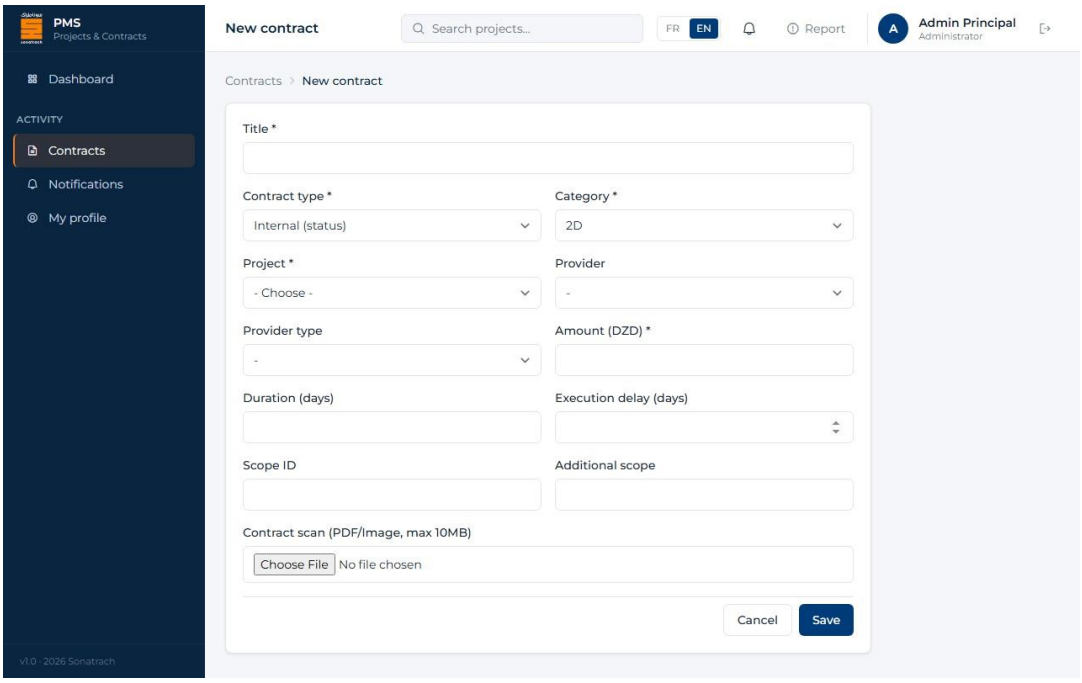


Figure 3.9: Contract creation form (Admin role)

Description. The Admin-only creation form (Figure 3.9) links each new contract to its parent project; for provider contracts it additionally requires selecting a service provider and a provider-contract type (acquisition, processing, reprocessing).

3.3.5 Master Data

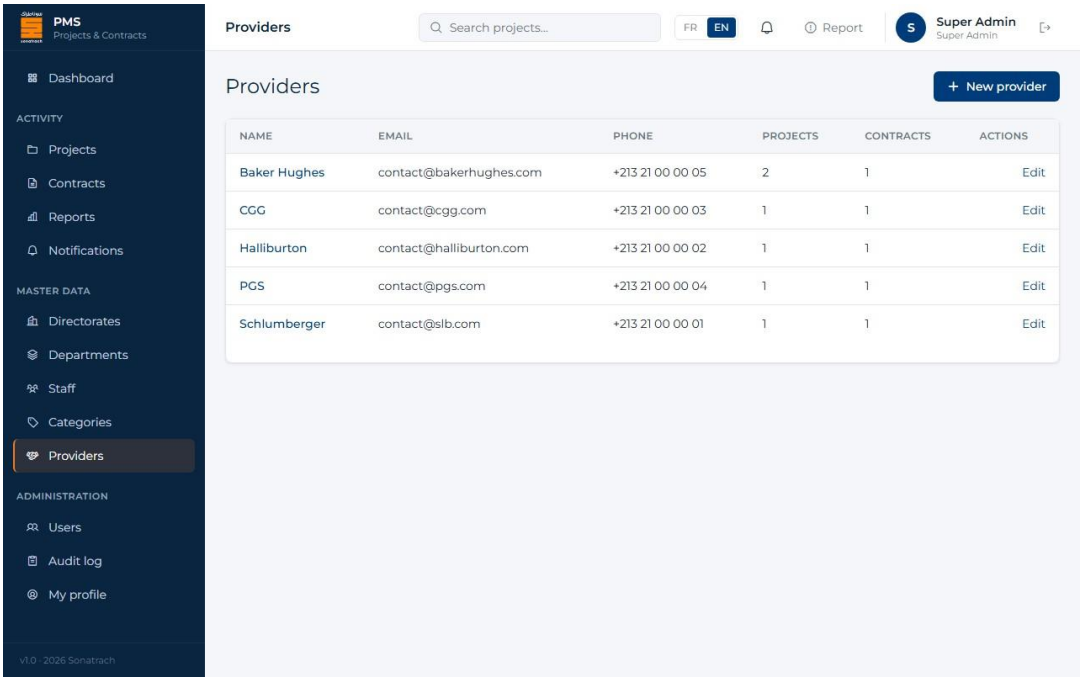


Figure 3.10: Providers directory

Description. The providers directory (Figure 3.10) lists external service providers that may sign contracts with Sonatrach. Edit is allowed; deletion is disabled by design — providers can only be deactivated to preserve historical contract references.

Description. The staff directory (Figure 3.11) shows every internal employee with their department, category and account state. Per the requirements, records are never hard-deleted: the Super Admin can only suspend or reactivate an account.

Description. Figure 3.12 shows the top level of the Sonatrach organisational hierarchy. Directorates host provider-funded projects and aggregate KPIs in the executive reports.

Description. Departments (Figure 3.13) belong to a directorate and host internal ("status") projects. Each staff member is attached to exactly one department.

Description. The user-administration screen (Figure 3.14) is the single place where roles are assigned. Role changes take effect on the next request because the permission cache is invalidated through Spatie's `PermissionRegistrar`.

3.3.6 Audit, Notifications, Reports and Profile

Description. The audit log (Figure 3.15) records significant actions (login, create, update, validate, archive) with their actor, subject and JSON payload. It satisfies the traceability requirement defined in Chapter 1.

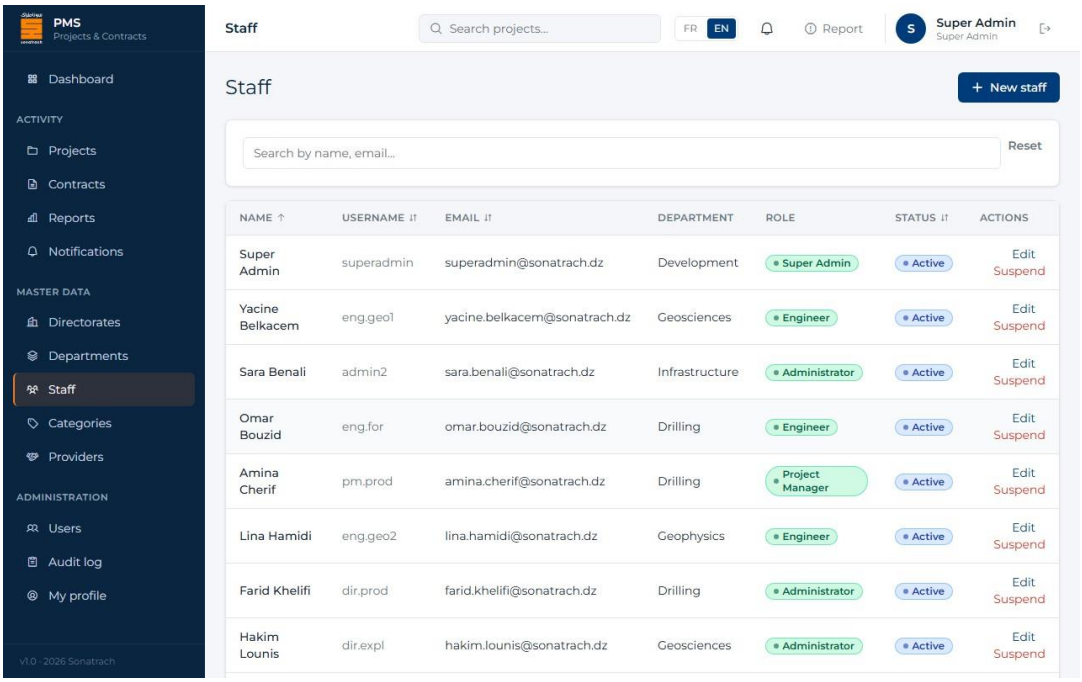


Figure 3.11: Staff directory with Suspend/Reactivate actions

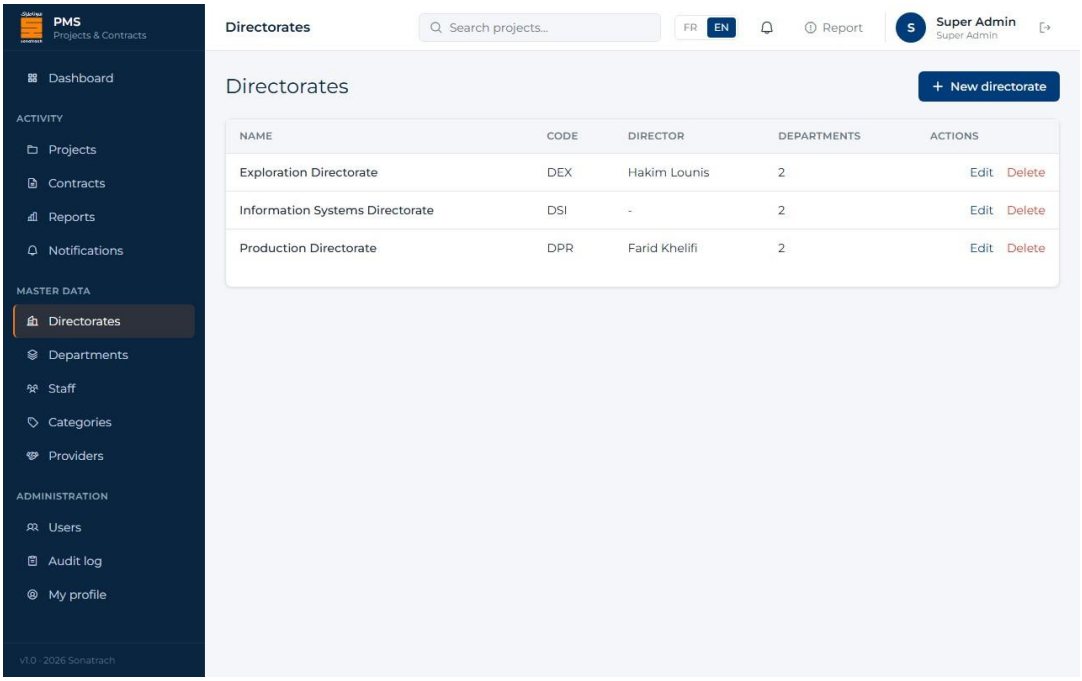


Figure 3.12: Directorates

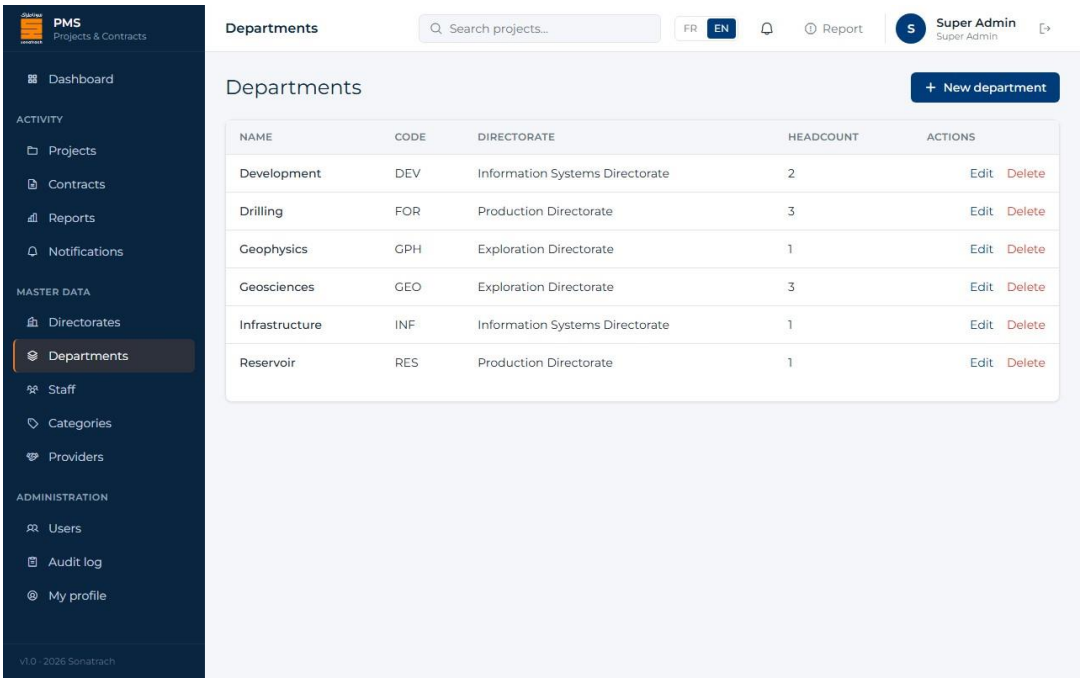


Figure 3.13: Departments

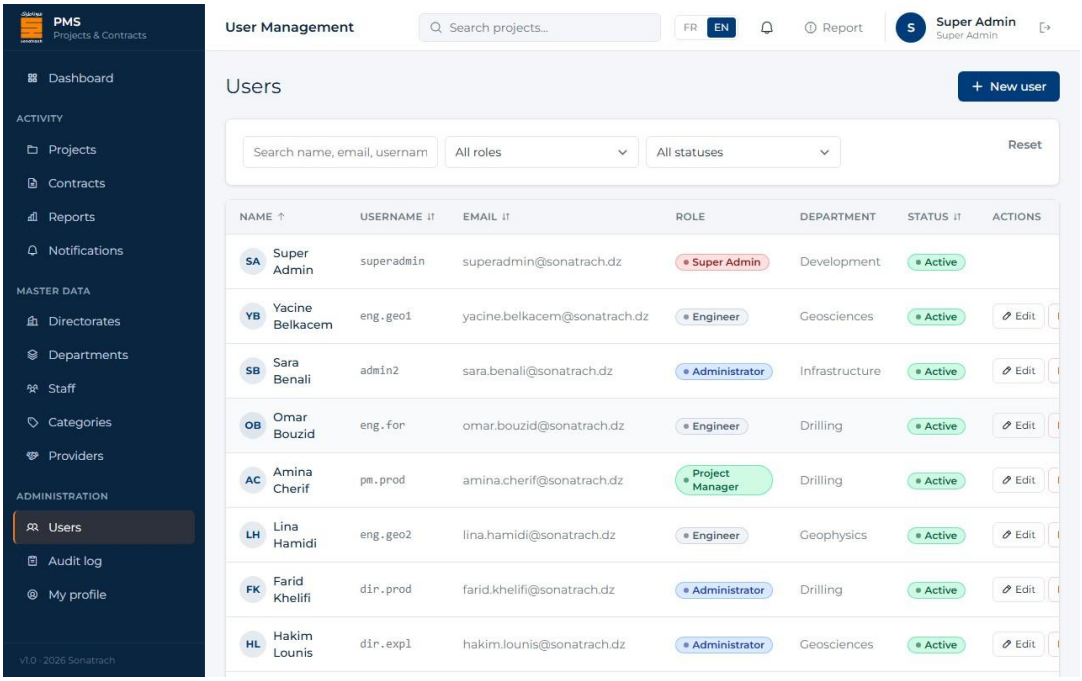


Figure 3.14: User accounts and role assignments

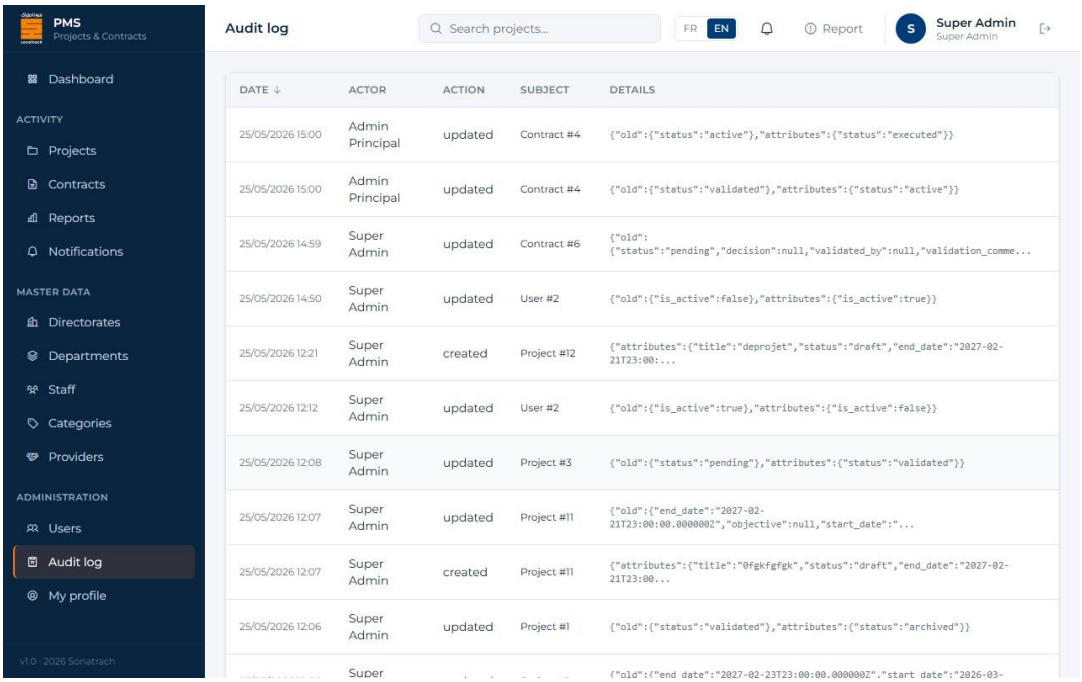


Figure 3.15: Audit log (Super Admin only)

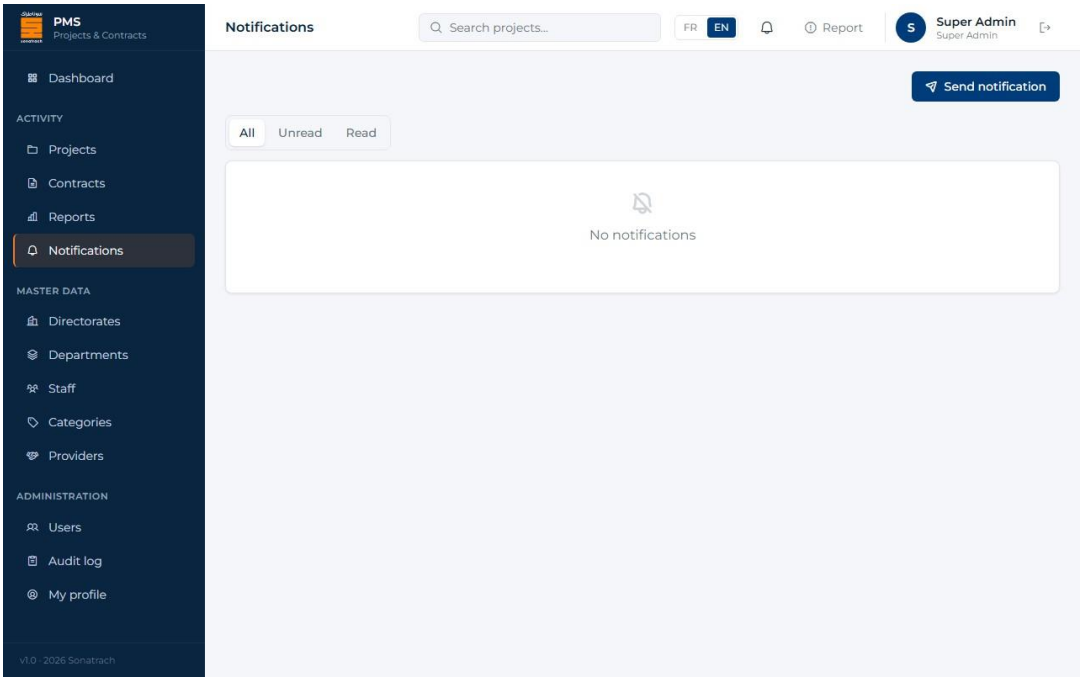


Figure 3.16: Notifications inbox

Description. The notifications inbox (Figure 3.16) groups lifecycle events that concern the connected user — typically validation decisions or new assignments — and offers a one-click "mark as read" action.

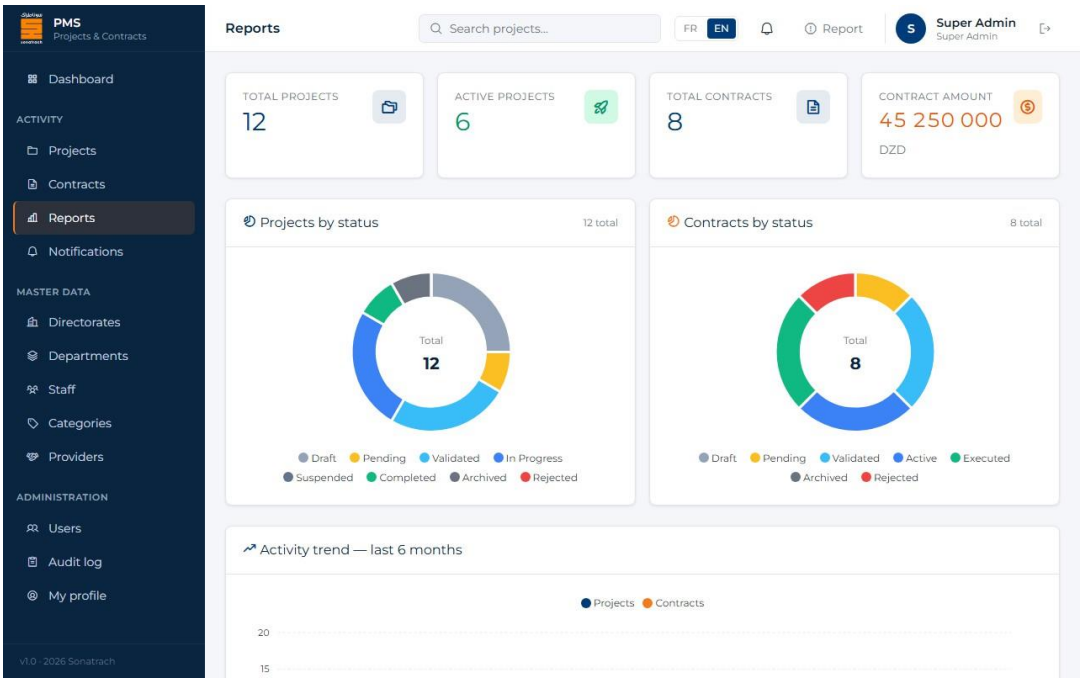


Figure 3.17: Reports and exports

Description. The reports module (Figure 3.17) exposes filtered PDF and Excel exports of the project and contract portfolios, generated server-side via DomPDF and Maatwebsite Excel.

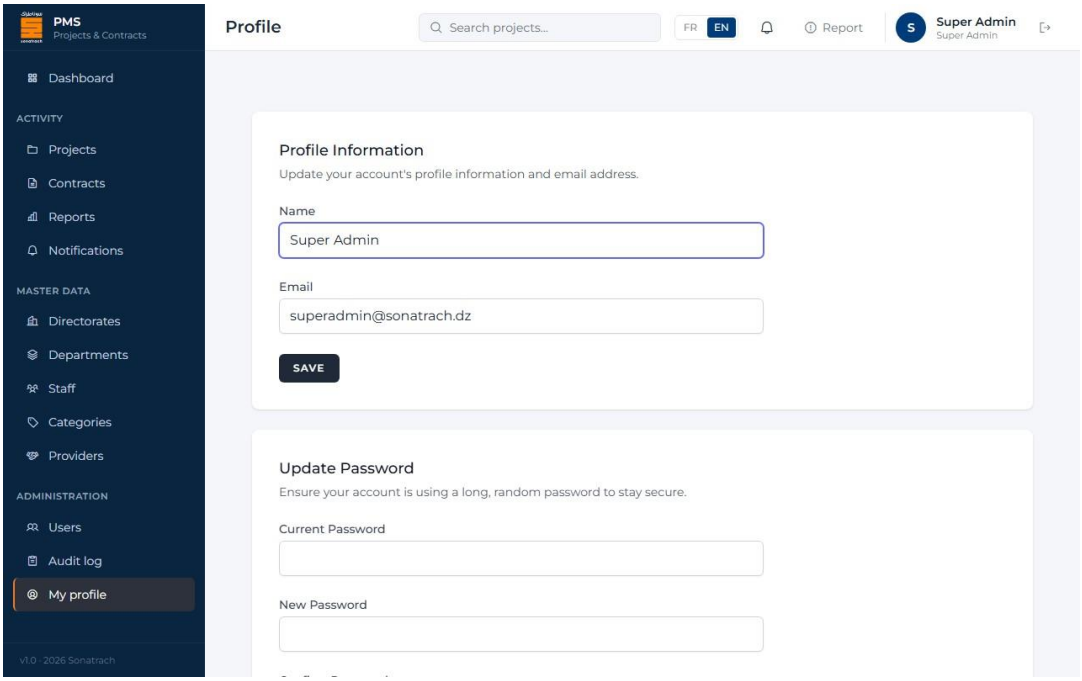


Figure 3.18: User profile management

Description. The profile screen (Figure 3.18) lets every user — regardless of role — update

their own personal data, contact information and password. Sensitive fields (role, status, department) remain read-only.

Chapter Conclusion

The chapter has shown how the design of Chapter 2 was turned into a running application. The technology stack — Laravel 11, MySQL 8 — was chosen for its alignment with the requirements, not for its novelty. With functional, boundary and security tests all passing, the system is ready for the deployment discussed in the general conclusion.

General Conclusion

The application we built during this internship now replaces the old spreadsheet workflow at the Exploration Division. It covers the full life of a project and a contract, keeps a full audit trail of who did what, and lets each role see only the data they are allowed to see. Everything runs in the browser; there is nothing to install on the user's machine.

This internship taught us far more than how to write code. It showed us how an organisation works from the inside, and how real user needs are translated into digital solutions. The daily collaboration with the Exploration Division team added a human dimension that no academic project had ever provided.

What surprised us most was how much we did not have to write ourselves. The audit log, for instance, contains no manual logging code in any controller: a single line on each Eloquent model is enough, and Spatie ActivityLog records every create, update and delete on its own. RBAC across the whole application is enforced by two middleware declarations, not by permission checks scattered through the controllers. Reports in PDF and Excel reuse the same Eloquent queries that power the UI. Learning to lean on the framework instead of reimplementing things badly was probably the biggest takeaway of the internship.

On the technical side, we learned that clean code that runs on your machine is not enough — what the field demands is code that holds up against real data, real users, and unexpected errors. Designing scalable databases and handling edge cases that no textbook mentions were among the most valuable lessons.

On the practical side, we came to understand that programming is not the only skill that matters. Communicating with users, managing time across parallel tasks, and turning vague requirements into clear development tasks — these are skills that can only be built in the field, not in a lecture hall.

Several extensions came up during the internship but were left out of the final scope. A REST API would make it possible to plug the system into other Sonatrach internal tools. Automated e-mail reminders before a contract expires would close a real gap; the in-app notifications are useful but depend on the user logging in regularly. A mobile companion for field engineers, electronic signatures on validated contracts and richer date-range analytics on the dashboard were also discussed with the internship supervisors and are documented in our handover notes.

Bibliography

[1] T. Otwell et al., *Laravel 11 Documentation*, Laravel LLC, 2024. https://laravel.com/docs/11.x

[2] Spatie, *Laravel Permission*, 2024. https://spatie.be/docs/laravel-permission

[3] Spatie, *Laravel ActivityLog*, 2024. https://spatie.be/docs/laravel-activitylog

[4] A. Wathan et al., *Tailwind CSS v3 Docs*, Tailwind Labs, 2024. https://tailwindcss.com/docs

[5] C. Porzio, *Alpine.js v3*, 2024. https://alpinejs.dev

[6] Oracle Corporation, *MySQL 8.0 Reference Manual*, 2024. https://dev.mysql.com/doc/refman/8.0/en/

[7] E. You et al., *Vite Documentation*, 2024. https://vitejs.dev

[8] OWASP Foundation, *OWASP Top Ten 2021*, 2021. https://owasp.org/www-project-top-ten/

[9] M. Fowler, *Patterns of Enterprise Application Architecture*, Addison-Wesley, 2002.

[10] draw.io, UML Diagrams, 2024. https://drawio-app.com/uml-diagrams/