TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

UNIVERSITÉ DE TUNIS
جامعة تونس

Floucna Mina Fina

System Architecture and High-Level Design

Instructor:
Dr. Manel Abdelkader

Team Members:
Louey Dridi
Dhia Chedly
Khalil Louhichi
Leila Baccouche
Mohamed Belgacem

Academic Year: 2025 – 2026

Abstract

This document presents the High-Level Design (HLD) and system architecture for **Floucna Mina Fina**, a secure micro-lending portal developed as part of Project 15. The system facilitates a streamlined process where borrowers can request small loans, and administrators can approve them, resulting in the automatic generation of a legally traceable loan agreement PDF.

The primary cybersecurity focus of this project is the **document compliance layer**. To ensure a secure and trusted environment, the platform integrates Keycloak for OAuth2/OpenID Connect authentication, enforcing strict role-based access control (RBAC). Identity verification is prioritized through a dual-mode Know Your Customer (KYC) system, utilizing Didit as the primary provider with a local fallback mechanism for academic demonstration.

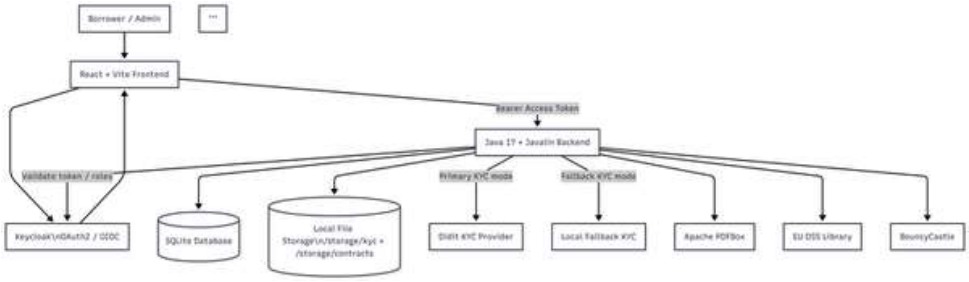
Crucially, the platform ensures document integrity and non-repudiation by applying **PAdES-compliant digital signatures** using self-signed certificates and integrating trusted timestamping to prove the time of signing. An administrative compliance dashboard, powered by the EU Digital Signature Service (DSS) library, is implemented to verify signature validity, document integrity, and timestamp presence, demonstrating a robust defense against tampering in digital financial agreements.

Contents

1. System Architecture and Technical Design

1.1 High-Level Architecture

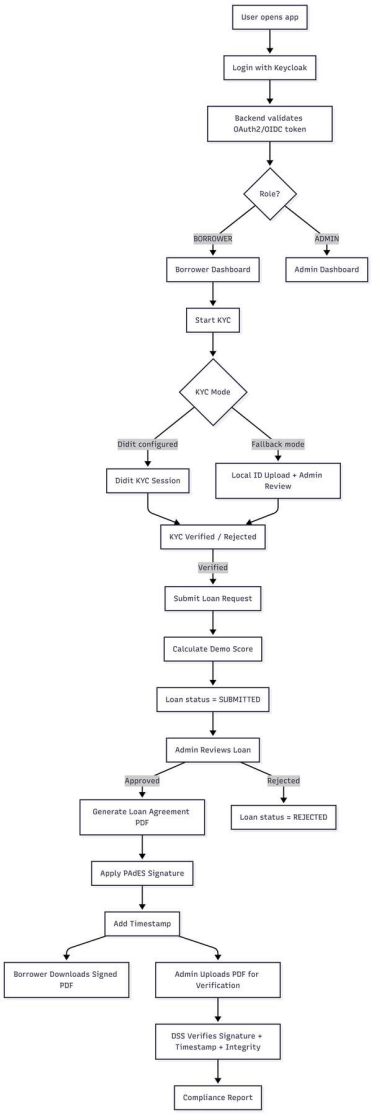
The Floucna platform employs a modern web application architecture, decoupling the frontend user interface from the backend services and relying on external providers for authentication and identity verification.



- **Frontend (React + Vite):** A Single Page Application (SPA) providing dashboards for Borrowers and Admins, communicating with the backend via Bearer Access Tokens.
- **Authentication (Keycloak):** Handles OAuth2/OpenID Connect authentication and manages user roles.
- **Backend (Java 17 + Javalin):** The core API layer that validates tokens, manages business logic, and integrates with cryptographic libraries (PDFBox, EU DSS, BouncyCastle).
- **Data & Storage:** Utilizes a SQLite database for structured data and local file storage for KYC documents and generated contracts.
- **External KYC (Didit):** The primary provider for liveness detection and identity verification.

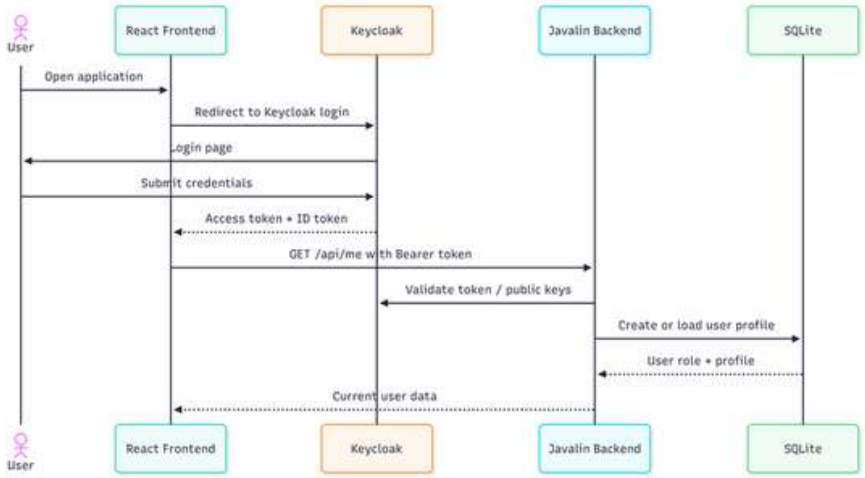
1.2 Main End-to-End Flow

The overall user journey begins with authentication via Keycloak and splits based on the user's role (BORROWER or ADMIN). Borrowers proceed through KYC validation and submit loan requests, while Admins review requests and trigger contract generation, signing, and compliance verification.



1.3 Authentication Flow

Floucna does not store local passwords. All authentication is delegated to Keycloak. The backend strictly validates Keycloak-issued access tokens and enforces role-based access before processing any request.

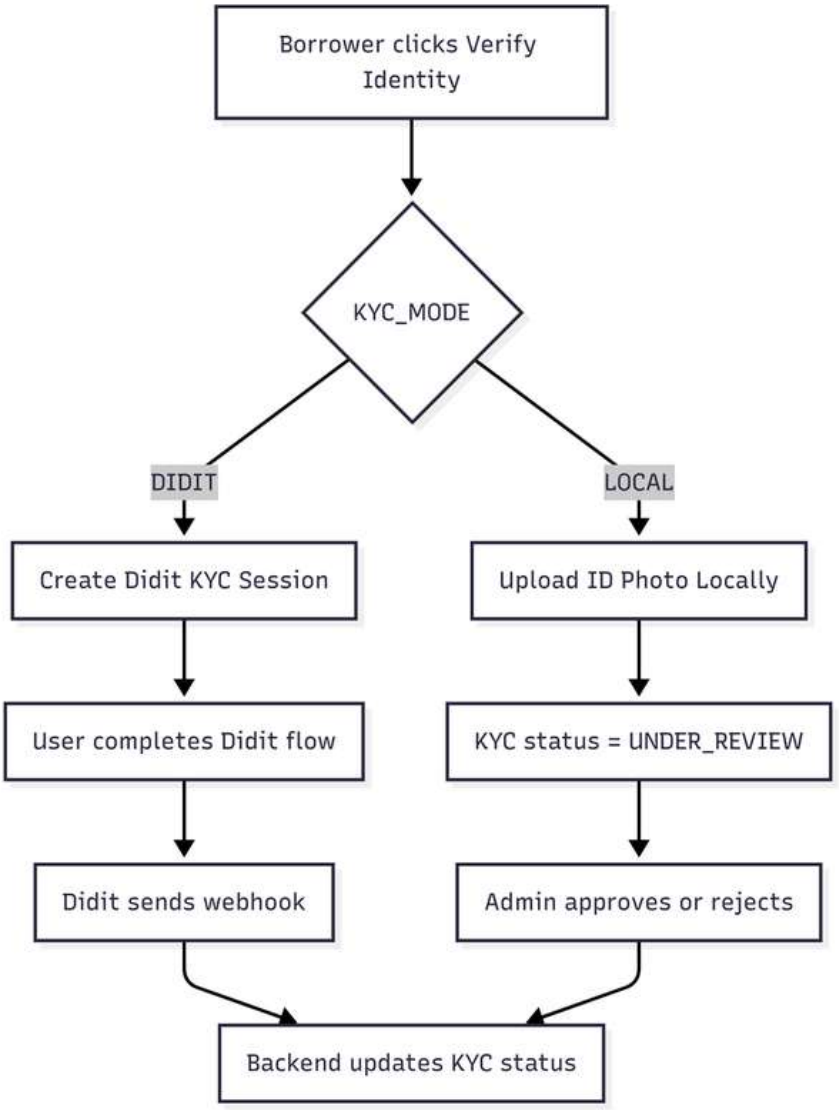


The system defines two primary roles: **BORROWER** and **ADMIN**. Access to backend endpoints is governed by the rule: `Access = valid token + required role`.

1.4 KYC (Know Your Customer) Design

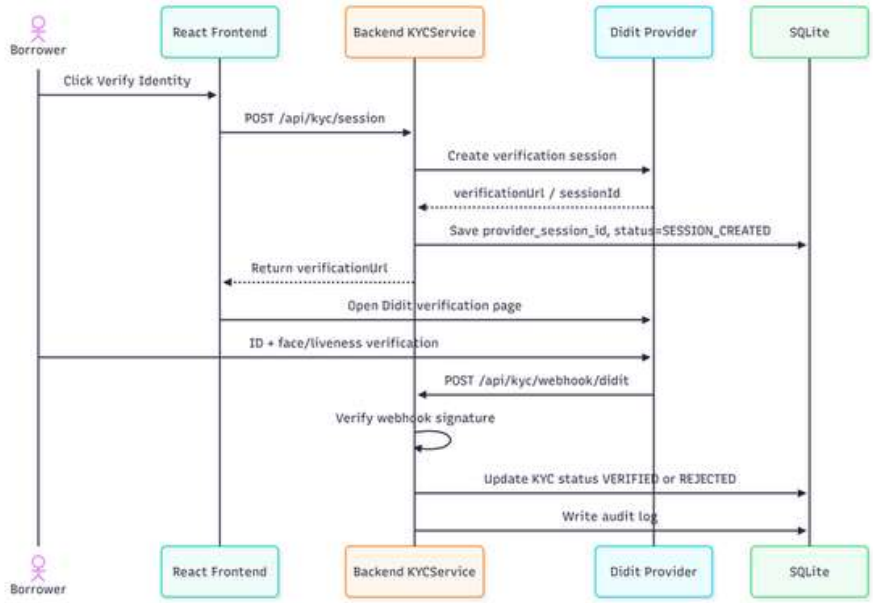
Identity verification is critical before loan submission. The system implements a flexible, two-mode KYC approach.

1.4.1 KYC Mode Decision Flow



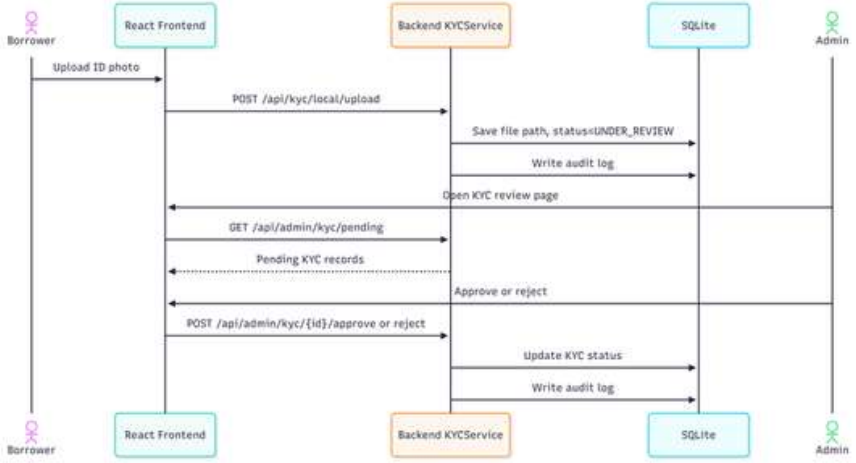
1.4.2 Didit KYC Flow

The primary flow redirects users to the Didit provider for professional verification (ID + Face Match). Upon completion, Didit sends a secure webhook to the backend to update the status.



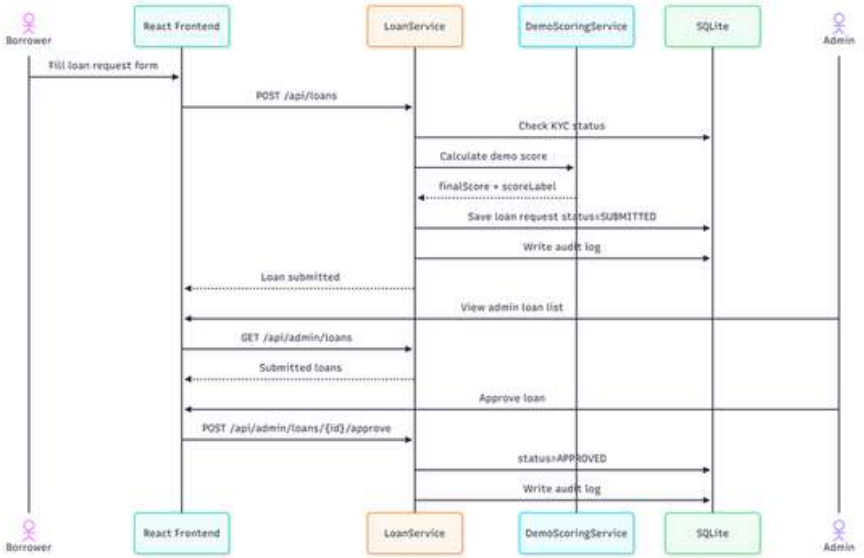
1.4.3 Local Fallback KYC Flow

An academic alternative where users upload an ID photo locally. The status is set to `UNDER_REVIEW`, requiring an Administrator to manually approve or reject the submission.



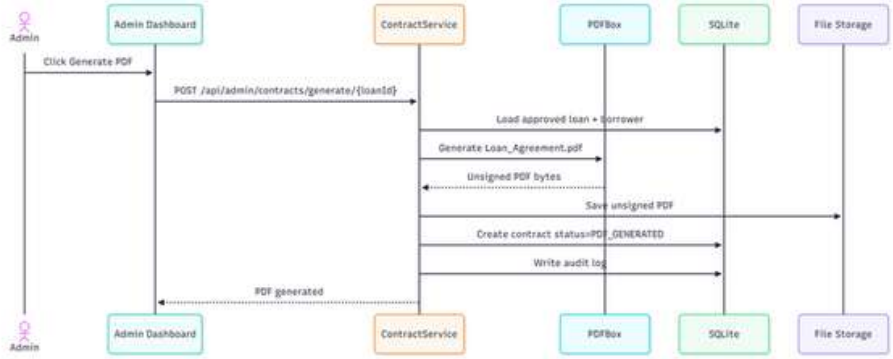
1.5 Loan Request Sequence

Borrowers with a `VERIFIED` KYC status can submit small loan requests. The system employs a simple demo scoring layer to simulate a fintech credit-risk model. The final score informs the Admin's decision to approve or reject the request.



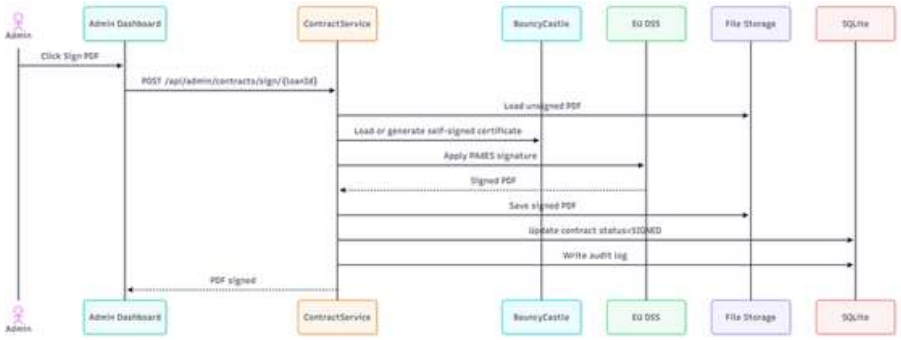
1.6 Contract Generation Flow

Upon loan approval, the backend automatically generates a comprehensive PDF loan agreement using Apache PDFBox. This document contains borrower details, the demo score, loan terms, and a signature section.



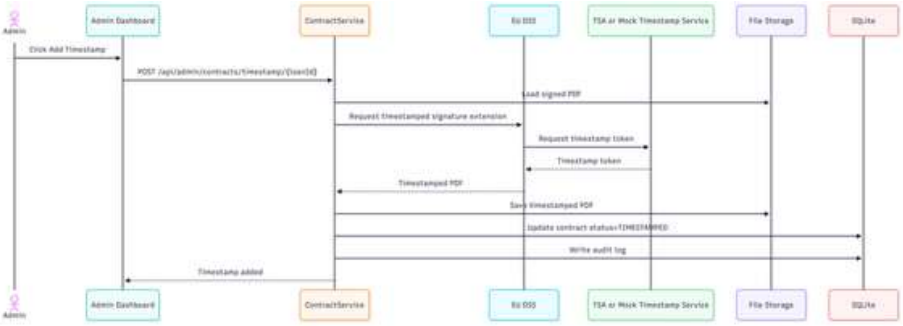
1.7 PAdES Signature Sequence

To establish legal traceability, the system signs the generated PDF using a self-signed certificate managed by BouncyCastle. A basic PAdES (PDF Advanced Electronic Signatures) signature is applied using the EU DSS library, ensuring the cryptographic binding of the identity to the document.



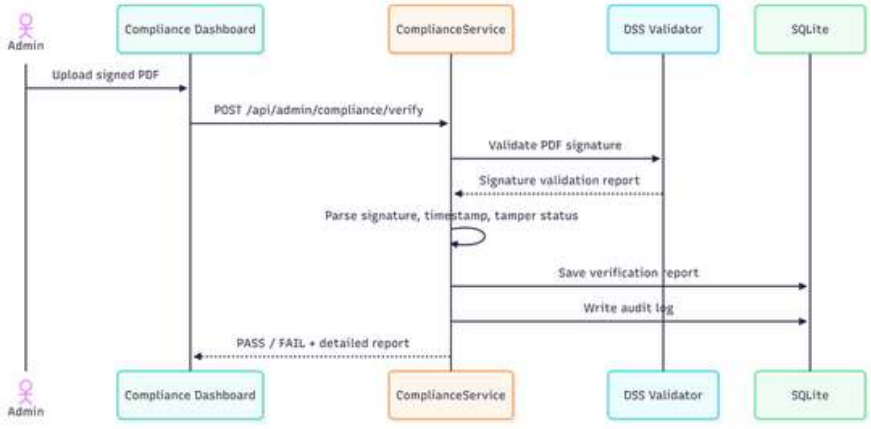
1.8 Timestamp Sequence

To prove the exact time the signature was applied and protect against certificate expiration issues, a timestamp is added to the signed PDF. The DSS library requests a timestamp token from a Time Stamping Authority (TSA) or a mock service.



1.9 Compliance Verification Sequence

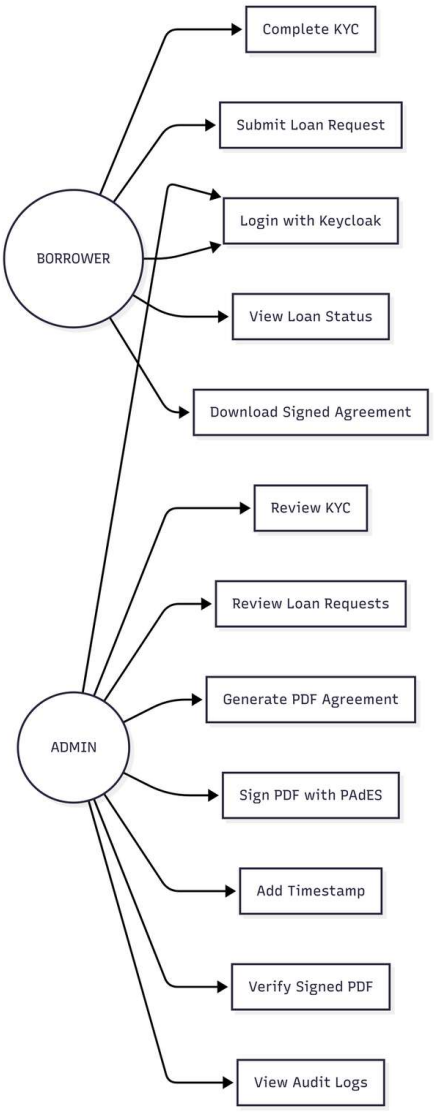
The final layer of the system's security architecture is the Compliance Dashboard. This allows Administrators to verify the integrity and origin of any signed loan agreement via the DSS Validator.



Checks include signature presence/validity, document integrity (detecting tampering), and timestamp validation.

1.10 Use Case Diagram

The system features distinct capabilities for Borrowers (loan requests, KYC submission) and Admins (reviewing applications, generating and signing PDFs, and compliance checks).



1.11 Class Diagram

The system's backend is structured around core entities such as Users, KycRecords, LoanRequests, Contracts, and AuditLogs, managed by their respective services.

1.12 Entity Relationship Diagram (ERD)

The database schema reinforces the system's architecture, employing strict relationships between users, KYC documents, loans, and their associated digital contracts.