



**UNIVERSITY OF ADEN
FACULTY OF COMPUTER AND
INFORMATION TECHNOLOGY
IS DEPARTMENT**



**Signature Platform — A Web-Based Contract System
with Enforced Multi-Method Identity Verification**

Prepared by

Supervised by

Eng. Mohammed Bin Shahna

April 2026

Acknowledgement

First of all, the project team would like to thank Allah for giving us the patience and clarity needed to finish this work.

We would like to thank our supervisor for the time spent reviewing our progress, the practical advice during difficult parts of the development, and the steady push to keep the scope realistic for a graduation project.

We are also grateful to the staff and instructors of our college for the courses that prepared us for this project, especially the modules on web development, databases, software engineering, and information security.

Finally, we thank our families and classmates for the encouragement during the months of building, breaking, and rebuilding the system.

Abstract

Paper contracts depend on physical presence, ink signatures, and personal trust. Once both parties leave the room, the only thing tying them to the agreement is a piece of paper that can be lost, copied, or denied. Existing electronic signature services such as DocuSign, Zoho Sign, and eSignatures.com solve part of the problem but stop at the click of a single button, which is rarely enough proof when one of the parties later disputes the agreement.

This project presents a web-based contract platform that enforces multi-method identity verification on every party before a contract is sealed. The platform is built with pure PHP 8.1 and MySQL 8 on the server side, and uses HTML, CSS, and vanilla JavaScript on the client. The creator drafts a contract in a rich-text editor, invites the other parties by email, and chooses any combination of eight verification methods: drawn signature, uploaded signature image, live webcam selfie, ID document upload, email OTP, typed consent clause, geolocation and IP capture, and a short video consent recording. Each invitee walks through the steps in order. Only after all required methods are passed does the party count as signed. When every party has signed, the system computes a SHA-256 hash over the locked content and produces a PDF that contains the contract, the embedded signatures, the audit trail, and a QR code that points to a public verification page where anyone can re-check the hash.

The result is a self-hostable platform that gives both sides a piece of evidence that is much harder to dispute than a single signature. The same architecture also fits low-resource environments where regional language support and offline-friendly hosting matter more than a glossy SaaS dashboard.

List of Content

Acknowledgement.....	2
Abstract.....	3
List of Content.....	4
List of Figures.....	6
List of Tables.....	7
Chapter 1: Introduction.....	8
1.1 Overview.....	8
1.2 Problem Statement.....	8
1.3 Objectives.....	9
1.4 Advantages.....	9
1.5 Future Challenges and Limitations	10
1.6 Project Scope	10
Chapter 2: Literature Review.....	12
2.1 Introduction.....	12
2.2 Real-world Platforms Analysed.....	12
DocuSign	12
Zoho Sign.....	13
eSignatures.com	14
Comparison with the Signature Platform.....	15
Chapter 3: Methodology.....	17
3.1 Overview.....	17
3.2 Software Development Life Cycle (SDLC).....	17
3.3 SDLC Models.....	18
3.4 Selected SDLC Model — Incremental Model.....	19
Why Incremental was chosen	19
Phases of the Incremental approach.....	19
3.5 Gannt Chart	19
3.6 Functional Requirements.....	20
3.7 Non-Functional Requirements	22
3.8 Hardware Requirements.....	23
3.9 Software Requirements	23
3.10 System Technologies	24
3.11 UML Diagrams and Data Models.....	25
Use Case Diagram.....	25
Use Case Scenario — Create and Send a Contract	27
Use Case Scenario — Invitee Signs a Contract.....	28
Activity Diagram	29
Sequence Diagram	30

Class Diagram.....	31
DFD Level 0 (Context Diagram).....	32
Entity Relationship Diagram (ERD).....	33
Database Diagram	33
3.12 Data Dictionary	34
users	34
contracts	34
contract_parties.....	35
verification_requirements	36
verification_submissions	37
files.....	38
audit_events.....	38
Chapter 4: Design and Implementation	40
4.1 Overview.....	40
4.2 Authentication	41
4.3 Creator Dashboard	42
4.4 Contract Editor	43
4.5 Invitee Landing Page	45
4.6 Verification Steps.....	46
4.6.1 Drawn Signature	46
Live Selfie	47
ID Document Upload.....	48
Consent Clause.....	48
Geolocation and IP.....	48
Uploaded Signature Image	49
Chapter 5: Conclusion.....	50
5.1 Summary	50
5.2 Future Work.....	50
References.....	52

List of Figures

Figure 1 Docusign	13
Figure 2 Zoho Sign.....	14
Figure 3 e-Signature.com.....	15
Figure 4 SDLC phases	17
Figure 5 SDLC models comparison	18
Figure 6 Gantt chart	20
Figure 7 Use Case diagram	26
Figure 8 Activity diagram.....	29
Figure 9 Sequence diagram	30
Figure 10 Class diagram	31
Figure 11 DFD Level 0 (Context Diagram)	32
Figure 12 Entity Relationship diagram.....	33
Figure 13 Database Diagram	33
Figure 14 System architecture	40
Figure 15 Invitee signing flow	41
Figure 16 Login page	41
Figure 17 Creator dashboard	42
Figure 18 Contract editor.....	43
Figure 19 Verification methods picker	44
Figure 20 Verification methods overview.....	44
Figure 21 Invitee landing page	45
Figure 22 Drawn signature step	46
Figure 23 Live selfie step	47
Figure 24 ID document upload step	48

List of Tables

Table 1 Project scope.....	11
Table 2 Pros & cons of DocuSign	12
Table 3 Pros & cons of Zoho Sign.....	13
Table 4 Pros & cons of eSignatures.com	14
Table 5 Comparison of e-signature platforms with the Signature Platform.....	16
Table 6 Functional requirements	21
Table 7 Non-functional requirements	22
Table 8 Hardware requirements	23
Table 9 Software requirements	24
Table 10 Comparison of system technologies.....	24
Table 11 Use case scenario: create and send a contract.....	27
Table 12 Use case scenario: invitee signs	28
Table 13 Data dictionary: users.....	34
Table 14 Data dictionary: contracts	35
Table 15 Data dictionary: contract_parties	36
Table 16 Data dictionary: verification_requirements	37
Table 17 Data dictionary: verification_submissions.....	38
Table 18 Data dictionary: files	38
Table 19 Data dictionary: audit_events	39

Chapter 1: Introduction

1.1 Overview

Contracts shape almost every formal interaction between two parties — employment, rent, freelance work, sales, partnership, and many others. The traditional process needs both sides to be in the same place, sign two paper copies, and trust each other to keep the document safe. This works when the parties already know each other, but it fails when they are in different cities, when one side later denies signing, or when the paper is lost in a moving box.

Online signing services such as DocuSign, Zoho Sign, and eSignatures.com move the signing step into the browser. They have made remote signing common, but they treat the proof of agreement as a single click on a button. A real contract dispute usually needs more than that — the other side can claim the email account was hacked, the signature was traced from a leaked image, or the click was made by someone else.

This project tackles that gap. It is a web platform built with pure PHP, MySQL, HTML, CSS, and JavaScript. The creator drafts a contract, invites the other parties, and decides which proofs each party must complete before the contract is sealed. The platform supports eight different methods, and the creator can require any combination. A party can be asked to draw a signature, upload a signed image, take a live selfie, upload a national ID, type a consent phrase, confirm an email OTP, allow location and IP capture, and record a short video reading the agreement clause. Once every party has cleared every method, the contract is hashed, exported as a PDF with an embedded QR code, and locked. The QR code on the PDF points to a public page that recomputes the hash so anyone can confirm the file has not been altered.

1.2 Problem Statement

Two issues motivate this project. First, paper contracts are physical, easy to lose, and hard to verify after the fact. Second, current electronic signature services compress the proof of agreement into one click. When a dispute happens, the trail often comes down to an email saying the link was opened, which is not strong evidence in court or in front of a third-party arbitrator. Smaller businesses and individuals also find existing services expensive on a per-document basis, and many of them do not handle Arabic or right-to-left content well. The need is a self-hostable platform that takes the consent step

seriously, captures multiple kinds of evidence per party, and produces an artifact that is hard to deny.

1.3 Objectives

- Build a PHP/MySQL web platform that lets a creator draft, send, and track contracts between two or more parties.
- Provide eight independent verification methods so the creator can mix and match the proof level for each contract.
- Make every step block until it passes, so a party who skips a method cannot move on.
- Capture an immutable audit trail that records every event with timestamp, IP, and user agent.
- Compute a SHA-256 hash on the final content and embed it in the exported PDF together with a QR code that points to a public verification page.
- Generate an evidence pack (PDF and raw files) that the creator can download as a single ZIP.
- Support Arabic and right-to-left layout natively in the UI and in the generated PDF.
- Run on a normal PHP/MySQL hosting setup or on XAMPP for local use, without external paid services for the core flow.

1.4 Advantages

The platform offers a few practical advantages over both paper and existing online signing tools. The first is multi-method proof on the same contract, which makes denial much harder than in a single-click flow. The second is the audit trail, which is recorded on the server side rather than rebuilt from logs, so it cannot be silently changed by a party. The third is the self-hostable architecture: a small business or government office can deploy the platform on its own server and avoid sending sensitive contracts to a third-party SaaS. The fourth is the strong Arabic and RTL support, which is something most international platforms still treat as an afterthought.

1.5 Future Challenges and Limitations

The platform has limits that need to be acknowledged. The video consent and ID upload steps are stored as files inside the storage volume — they are not encrypted at rest beyond the disk-level protection, so a deployment that handles sensitive ID copies should add an encryption layer. The OTP currently uses email; SMS is not implemented. There is no built-in legal review of the contract text; the system trusts that the creator wrote a valid agreement. Browser support for MediaRecorder is good on desktop but uneven on older mobile browsers. Finally, the platform does not yet integrate with national electronic ID schemes (such as a national digital identity gateway), which would give an even higher level of identity binding.

1.6 Project Scope

Included in the project	Not included in the project
A web platform written in pure PHP 8.1+ with MySQL 8 as the database	Native mobile applications (the platform is responsive but not packaged as an app)
Account registration and login for creators	Federation with national digital identity providers
Contract creation with a rich-text editor (Quill 2.x in RTL mode) and multi-party support	Live legal review of contract content
Eight verification methods that can be mixed per contract	End-to-end encryption of stored files at rest
Tokenized invite links with HMAC signing and a 14-day expiry	Long-running blockchain anchoring of every contract
A stepper UX for invitees with one verification per step and a progress bar	Real-time chat or co-editing between the parties
Audit trail with IP, user agent, and timestamp on every event	Built-in payment processing for paid contracts
SHA-256 content hash computed on finalize	Automated translation between languages
mPDF export with embedded signatures, selfies, ID images, and audit table	Integration with third-party CRM or ERP systems
QR code on the PDF pointing to a public verification page	Multi-tenant SaaS billing and quota management

Included in the project	Not included in the project
Evidence pack download as a ZIP with PDF, raw files, and a manifest	Court submission workflow specific to one jurisdiction
Arabic and English interfaces with right-to-left layout for Arabic	Custom branding marketplaces or template stores

Table 1 Project scope

Chapter 2: Literature Review

2.1 Introduction

This chapter analyses the problem space and the system that the team built to solve it. It starts with three real-world platforms that already operate in this space (DocuSign, Zoho Sign, and eSignatures.com), pulls out their strengths and weaknesses, and then lists the requirements that came out of the comparison.

2.2 Real-world Platforms Analysed

Three reference platforms were studied during the planning phase. They all sit in the same broad category (electronic signatures) but with different priorities.

DocuSign

DocuSign is the largest platform in the e-signature market. It is a closed-source SaaS service used by large enterprises, government agencies, and law firms in many countries. The signing flow is centred on a single click on a coloured button, with optional add-ons such as ID verification or knowledge-based authentication for higher-tier plans.

Pros	Cons
Strong brand and legal recognition in many jurisdictions	Closed-source SaaS — the data lives on DocuSign servers
Very large template marketplace and integration list	Per-envelope pricing is high for small businesses and individuals
Court-tested audit trail format	Identity verification is a paid add-on, not the default
Solid mobile apps on iOS and Android	Arabic and right-to-left support is limited compared to Latin scripts
Compliant with major frameworks (eIDAS, ESIGN Act)	The standard signing experience is one click — disputes still happen

Table 2 Pros & cons of DocuSign

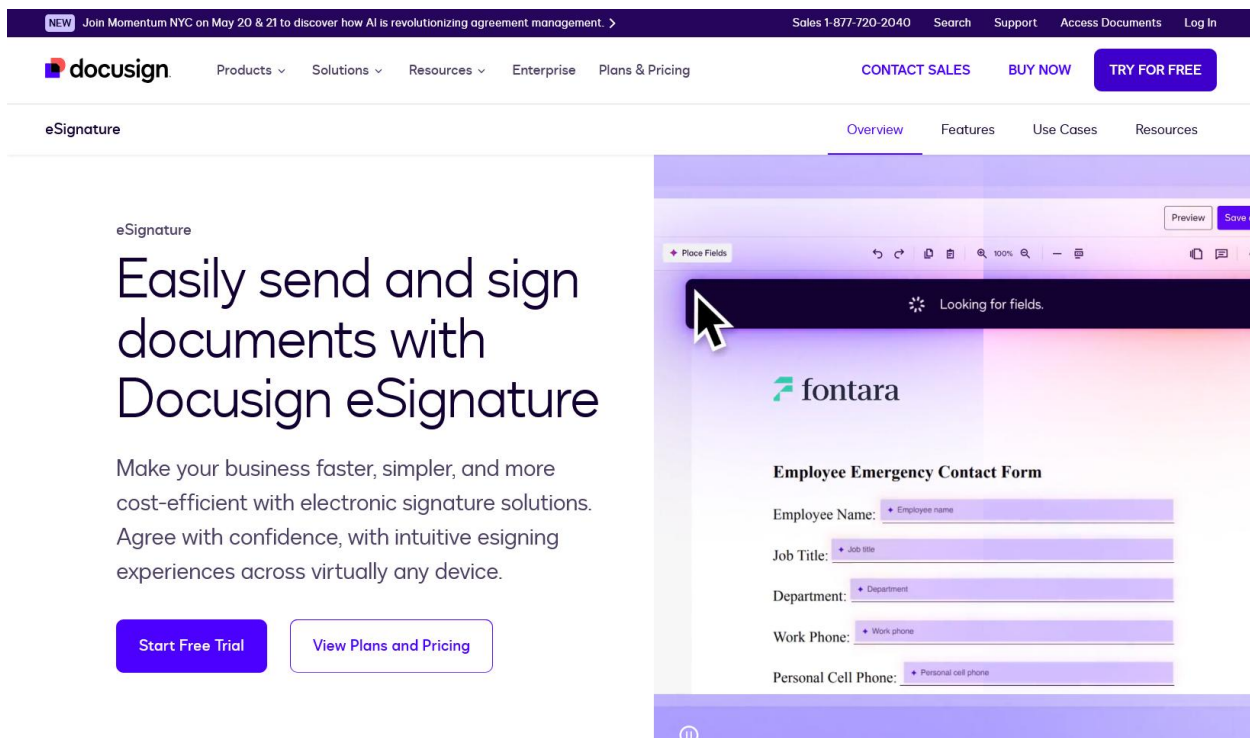


Figure 1 Docusign

Zoho Sign

Zoho Sign is part of the Zoho suite. It targets small and medium businesses that already use Zoho CRM, Books, or WorkDrive. The price per envelope is lower than DocuSign and the integration with the rest of the Zoho stack is the main selling point.

Pros	Cons
Cheaper than DocuSign for similar volume	Best value only if the business already uses other Zoho products
Tight integration with Zoho CRM, Books, and WorkDrive	Verification options are mostly OTP-based, no native video consent
Bulk send and template features	UI for non-Zoho users is less polished
Decent audit trail	Self-hosting is not an option
API for custom apps	Arabic UI exists but RTL rendering of long contracts can break

Table 3 Pros & cons of Zoho Sign

تطبيق توقيع إلكتروني متكامل للموقعين في قطاع الأعمال.

وداعًا لاستخدام الأوراق والأقلام.

حساب Zoho Sign

قل مرحبًا بالتوقعات
الإلكترونية.

هل ما زلت توفّق مستندات العمل باستخدام الأوراق والأقلام؟
ما رأيك أن توفّق المستندات في غضون دقائق بدلاً من
الانتظار لأيام أو ساعات؟ بفضل Zoho Sign، يمكنك الآن
توقيع مستندات أعمالك مباشرة من أي مكان ومن أي جهاز.

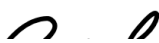


Figure 2 Zoho Sign

eSignatures.com

eSignatures.com is a smaller competitor that focuses on speed: send a contract, get a signature, done. It is popular with freelancers and small agencies because of its no-account flow for the signer and its pay-per-document pricing.

Pros	Cons
No account required for the signer	Verification is essentially a typed name + click
Pay-per-document, no monthly subscription	Audit trail is basic
Clean and fast UI	No identity proof beyond the email link
Friendly for non-technical users	Branding is limited compared to enterprise products
Good for low-stakes agreements	Not suitable for contracts that may end up in court

Table 4 Pros & cons of eSignatures.com

Sign Smarter, Stand Out

Welcome to the next generation of online signatures.

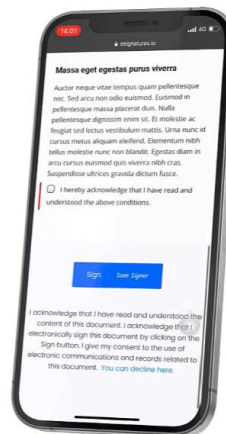


Figure 3 e-Signature.com

Comparison with the Signature Platform

Feature	DocuSign	Zoho Sign	eSignatures.com	Signature Platform
Verification methods per contract	1 default + paid add-ons	1-2 (OTP)	1 (click)	Up to 8, mix and match
Live selfie at sign time	Add-on	No	No	Yes (built-in)
ID document upload	Add-on	No	No	Yes (built-in)
Video consent recording	No	No	No	Yes (MediaRecorder)
Self-hosted	No	No	No	Yes
Source available	Closed	Closed	Closed	Yes (project source)
Arabic / RTL support	Limited	Partial	No	First-class

Feature	DocuSign	Zoho Sign	eSignatures.com	Signature Platform
SHA-256 tamper hash on PDF	Yes	Yes	Partial	Yes
QR-based public verification	No	Limited	No	Yes
Cost for individual users	High	Medium	Low	Free (self-host)

Table 5 Comparison of e-signature platforms with the Signature Platform

Chapter 3: Methodology

3.1 Overview

This chapter explains the development methodology used for the Signature Platform. It starts with a short summary of the Software Development Life Cycle, lists the well-known SDLC models, and explains why the team chose the Incremental model for this project. The chapter ends with the project Gantt chart that breaks the work into weeks.

3.2 Software Development Life Cycle (SDLC)

The Software Development Life Cycle is the structured process that a software team follows from the first idea to the deployed system and beyond. It has seven common phases: planning, requirements analysis, design, implementation, testing, deployment, and maintenance. The phases are not always strictly sequential — different SDLC models arrange them differently — but every project passes through each one at some point.

Software Development Life Cycle - Phases

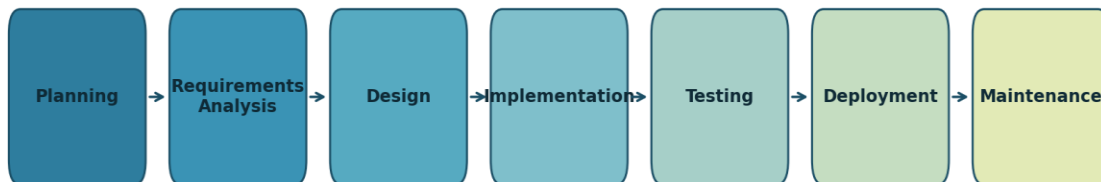


Figure 4 SDLC phases

3.3 SDLC Models

There is no single best model. Each one trades flexibility, speed, and risk handling differently. The most common ones are:

- Waterfall: a strict top-down model where each phase finishes before the next starts. Good for fixed requirements, weak when the requirements change.
- Incremental: the system is delivered in increments, each one adding more features. The team can release a usable subset early and adjust later.
- Iterative: similar to Incremental but the same scope is repeatedly refined rather than expanded with new features.
- Spiral: a risk-driven model that loops through planning, risk analysis, engineering, and evaluation.
- V-Model: a Waterfall variant where each development phase has a paired test phase.
- Agile: short cycles (sprints) with continuous feedback, used heavily in product companies.
- RAD (Rapid Application Development): focused on prototyping and quick UI iteration.

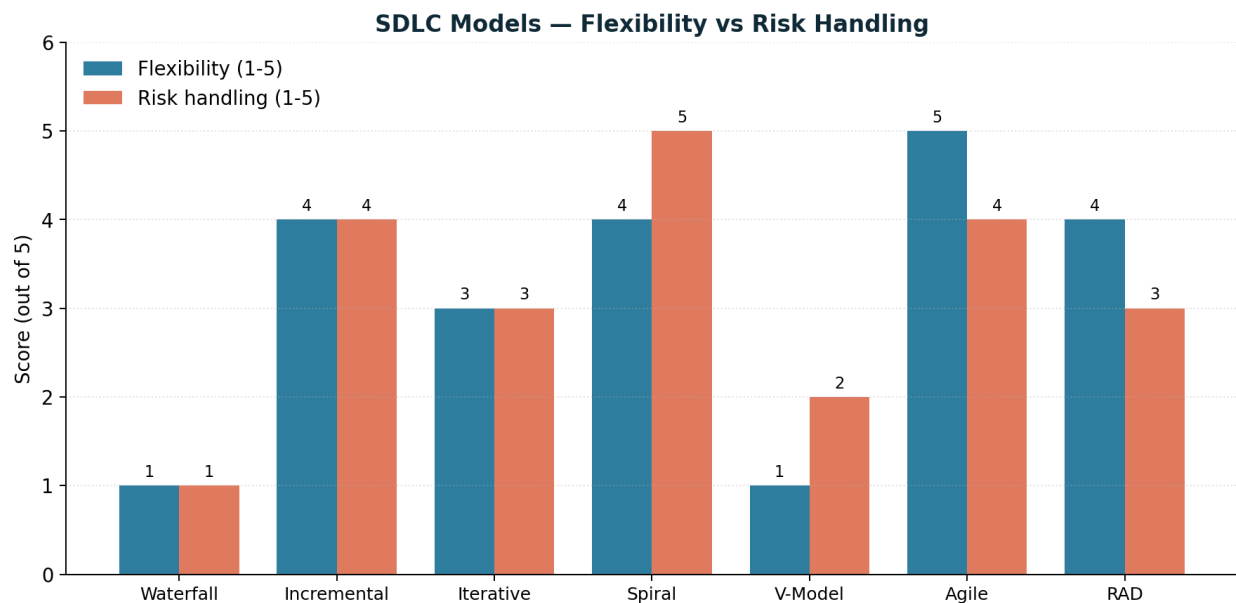


Figure 5 SDLC models comparison

3.4 Selected SDLC Model — Incremental Model

The team chose the Incremental model. The platform has a few clearly separable feature blocks (auth, contract editor, invite flow, eight verification methods, PDF export and audit) that can be built one after another, each ending in a working system that the team can demo to the supervisor. The team had limited time and needed early feedback on the verification UX, which is the most novel part of the project. Building everything as one big release would have delayed that feedback to the end of the semester.

Why Incremental was chosen

The decision came down to four reasons. First, the verification methods could be added one by one without rewriting the rest of the system. Second, an early increment with auth and a single signature method gave the supervisor something to react to before the team committed to the rest of the design. Third, the data model had to be stable from increment 2 onward, which the Incremental model handles well because each increment ends in a finished and tested slice. Fourth, the team could test each increment in isolation before adding more on top.

Phases of the Incremental approach

Increment 1 covered authentication, the basic dashboard, and contract draft creation with a single signature method. Increment 2 added the invite flow with HMAC tokens, the rejection flow, and the second verification method (uploaded signature image). Increment 3 added the remaining six verification methods and the per-contract method picker. Increment 4 added the mPDF export, the audit table, the SHA-256 hash, the QR code, and the public verification page. Each increment was tested before the next one started.

3.5 Gantt Chart

The project ran for roughly twenty weeks. The chart below shows the planned distribution of work across the four increments together with the supporting tasks (planning, design, documentation).

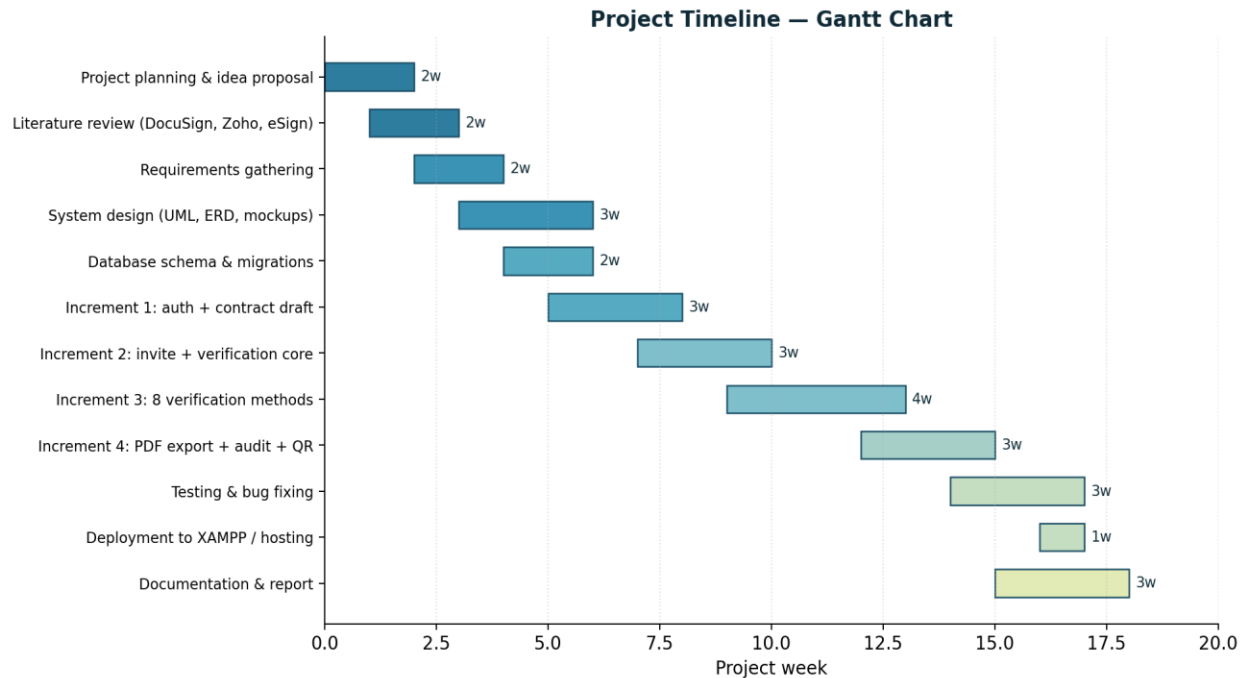


Figure 6 Gantt chart

3.6 Functional Requirements

Req. ID	Requirement description
FR-01	The platform shall let a registered creator draft a contract with rich-text content using a Quill editor in RTL mode.
FR-02	The platform shall let the creator add one or more parties to a contract with name, email, role, and signing order.
FR-03	The platform shall let the creator pick which of the eight verification methods are required for each contract.
FR-04	The platform shall send each invitee a tokenized invite link, signed with HMAC and expiring in 14 days.
FR-05	The platform shall present invitees with a stepper UX, one verification per step, with a visible progress bar.
FR-06	The platform shall accept a drawn signature on an HTML canvas and store it as a PNG.
FR-07	The platform shall accept a signature image upload (PNG or JPG, up to 2 MB).

Req. ID	Requirement description
FR-08	The platform shall capture a live webcam selfie when that method is required.
FR-09	The platform shall accept a national ID document upload (front and optional back).
FR-10	The platform shall send a 6-digit email OTP, rate-limit attempts, and verify the code before passing the step.
FR-11	The platform shall accept a typed consent clause and require it to match a fixed phrase.
FR-12	The platform shall capture geolocation (with permission) and IP at sign time.
FR-13	The platform shall record a short video consent (up to 30 seconds) using the MediaRecorder API and save it as webm.
FR-14	The platform shall block progress until every required verification method has passed.
FR-15	The platform shall let an invitee reject a contract with a reason, which is stored in the audit trail.
FR-16	The platform shall record an audit event for every action with timestamp, IP, and user agent.
FR-17	The platform shall compute a SHA-256 hash of the locked content when the last party signs.
FR-18	The platform shall export a PDF with mPDF that contains the contract, the embedded signatures, the audit table, and a QR code linking to the public verification page.
FR-19	The platform shall expose a public verification page that recomputes the hash for any given contract id.
FR-20	The platform shall let the creator download an evidence pack (PDF, raw files, manifest) as a single ZIP.
FR-21	The platform shall show a dashboard with counters for drafts, sent, completed, and incoming contracts.

Table 6 Functional requirements

3.7 Non-Functional Requirements

Req. ID	Category	Requirement description
NFR-01	Security	All passwords shall be stored using PHP password_hash with the default Bcrypt algorithm.
NFR-02	Security	All forms shall require a CSRF token tied to the session.
NFR-03	Security	All file uploads shall be stored outside the webroot and served through a controller with permission checks.
NFR-04	Security	Invite tokens shall be HMAC-signed and stored as a SHA-256 hash in the database.
NFR-05	Performance	Pages shall render within 1 second on a typical PHP/MySQL hosting environment.
NFR-06	Reliability	PDF export shall produce a valid file even when one optional file (selfie or video) is missing.
NFR-07	Usability	The UI shall work without a mouse — every action shall be reachable by keyboard and screen reader.
NFR-08	Compatibility	The platform shall run on PHP 8.1 or higher with MySQL 8 or MariaDB 10.4+.
NFR-09	Localisation	The interface and PDF shall fully support Arabic and right-to-left layout.
NFR-10	Maintainability	The code base shall not depend on a heavy framework — the entire backend is plain PHP.

Table 7 Non-functional requirements

3.8 Hardware Requirements

Hardware	Purpose	Specification
Development laptop / PC	Writing and running the platform locally	8 GB RAM, modern CPU, 50 GB free disk
Web server (production)	Hosting the platform for real users	1-2 vCPU, 2 GB RAM, 20 GB SSD, public IP
Webcam and microphone	Live selfie and video consent capture	Built-in laptop webcam or external USB camera
Smartphone (testing)	Testing the responsive UI on small screens	Any modern Android or iOS device
Printer (optional)	Printing the final PDF if the parties want a paper copy	Any A4 printer

Table 8 Hardware requirements

3.9 Software Requirements

Software	Version	Purpose
PHP	8.1 or higher	Backend programming language with PDO, GD, mbstring, fileinfo, zip extensions
MySQL	8.0+ (or MariaDB 10.4+)	Relational database
Apache	2.4+	Web server (XAMPP bundle for development)
XAMPP	Latest	Local development stack on Windows
Composer	2.x	PHP dependency manager
mPDF	8.x (via Composer)	Generating Arabic-friendly PDFs
PHPMailer	6.x	Sending invite emails and OTPs through SMTP
endroid/qr-code	5.x	Generating the QR code embedded in the PDF
Quill	2.x (CDN)	Rich-text editor for the contract body
Vanilla JavaScript	ES2020+	Signature pad, webcam capture, MediaRecorder for video

Software	Version	Purpose
Git	Latest	Version control
VS Code / PHPStorm	Latest	IDE for development

Table 9 Software requirements

3.10 System Technologies

Technology	Type	Role in the platform	Key advantage
PHP 8.1	Server language	All backend logic, routing, and templating	Universal hosting, fast for I/O-bound work, strong type hints
MySQL 8	Database	Stores users, contracts, parties, requirements, files, audit	Mature, reliable, free, supports JSON columns for payloads
mPDF	PDF library	Renders the final signed PDF with embedded images and audit	Arabic and RTL support out of the box
PHPMailer	Mail library	Sends invite emails and OTP codes through SMTP	Works with any SMTP provider (Gmail, SendGrid, MailHog)
endroid/qrcode	QR library	Generates the QR code printed on the PDF	Pure PHP, no external service
Quill 2.x	Rich-text editor	Contract body editor with RTL toolbar	Loaded from CDN, no build step needed
MediaRecorder API	Browser API	Records the video consent in webm	Built into modern browsers, no plugin
Canvas API	Browser API	Captures the drawn signature	Built into modern browsers
Web Crypto / hash_hmac	Crypto	HMAC for invite tokens, SHA-256 for content hash	Standard library on both sides

Table 10 Comparison of system technologies

3.11 UML Diagrams and Data Models

This section presents the diagrams that describe the system structure, behaviour, and data flow. The actual UML diagrams (use case, activity, sequence, class, DFD, ERD) are kept as placeholders for the team to draw with the diagram tool of choice and then drop into the document.

Use Case Diagram

The use case diagram shows the two main actors and what each one can do. The Creator is the primary actor: registered, logged in, owner of one or more contracts. Creator actions are: register, log in, create a contract, edit a draft, choose verification methods, add parties, send the contract, view the dashboard, download the evidence pack, and revoke or expire a contract. The Invitee is the secondary actor: not registered, opens the platform through a tokenized invite link. Invitee actions are: open the contract, read the body, complete each required verification step, sign the contract, or reject it with a reason. The public verification flow is also a use case but the actor is anyone in the public — they just open the verification URL and confirm the hash.

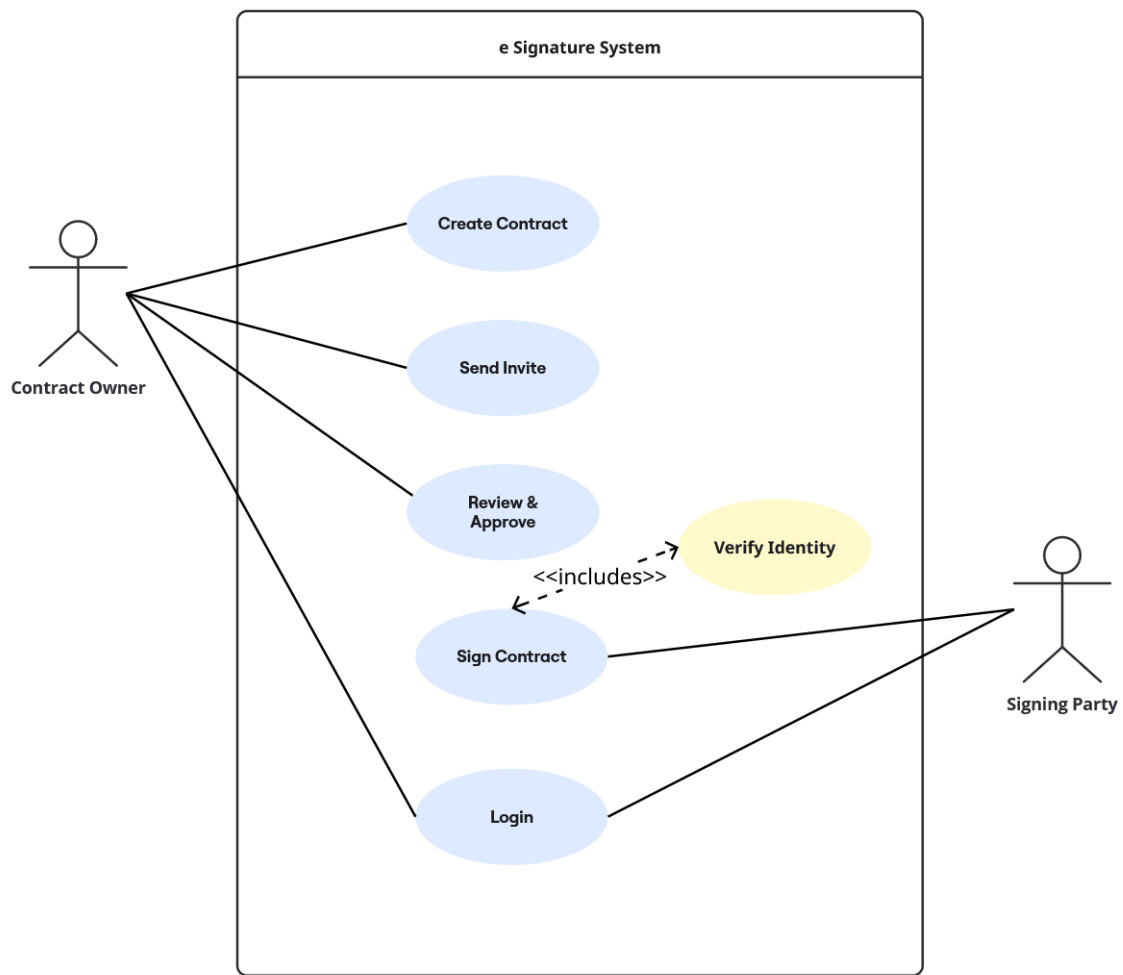


Figure 7 Use Case diagram

The two main use case scenarios are described in detail below.

Use Case Scenario — Create and Send a Contract

Use Case Title	Create and Send a Multi-Party Contract
Actor	Creator
Description	The creator drafts a contract, adds parties, picks verification methods, and sends invite links.
Preconditions	1. The creator is registered and logged in. 2. The platform is reachable on the configured APP_URL. 3. SMTP is configured (or OTP fallback is enabled).
Workflow / Steps	1. Creator opens the dashboard and clicks 'New contract'. 2. Creator enters a title and writes the body in the Quill editor. 3. Creator adds one or more parties with name, email, role, and signing order. 4. Creator picks the required verification methods for the contract. 5. Creator clicks 'Send'. 6. The platform creates an invite token for each party, signs it with HMAC, and stores its SHA-256 hash. 7. The platform sends each party an email containing the tokenized link. 8. The contract status changes from 'draft' to 'sent'. 9. The dashboard updates the counters.
Postconditions	Each invitee has a unique 14-day link they can open to sign.
Exceptions	1. SMTP failure — the invite link is shown on screen for manual copy. 2. A required field is missing — the form blocks submission and highlights the error. 3. The creator is not authenticated — the request is rejected by the auth middleware.

Table 11 Use case scenario: create and send a contract

Use Case Scenario — Invitee Signs a Contract

Use Case Title	Invitee Completes All Verification Steps and Signs
Actor	Invitee
Description	The invitee opens the link, walks through every required verification method, and signs the contract.
Preconditions	1. The invitee has received a valid token email. 2. The token has not expired and has not been used.
Workflow / Steps	1. Invitee opens the link. 2. The platform validates the token and shows the contract. 3. Invitee reads the contract body. 4. Invitee starts the stepper. 5. For each required method, the invitee completes the action and the platform records a verification submission. 6. Once all methods pass, the invitee taps 'Sign'. 7. The platform marks the party as 'signed' and writes an audit event. 8. If this was the last party, the platform finalizes the contract: hashes the body, generates the PDF, and sends the creator a confirmation.
Postconditions	The party is recorded as signed; the contract may be finalized if everyone has signed.
Exceptions	1. A method fails (wrong OTP, video too short) — the step shows an error and the invitee can retry. 2. The token has expired — the platform shows an expiry message and offers to request a new link. 3. The invitee rejects — the contract status becomes 'rejected' and the creator is notified.

Table 12 Use case scenario: invitee signs

Activity Diagram

The activity diagram captures the flow from the moment an invitee opens the link until the contract is either fully signed or rejected.

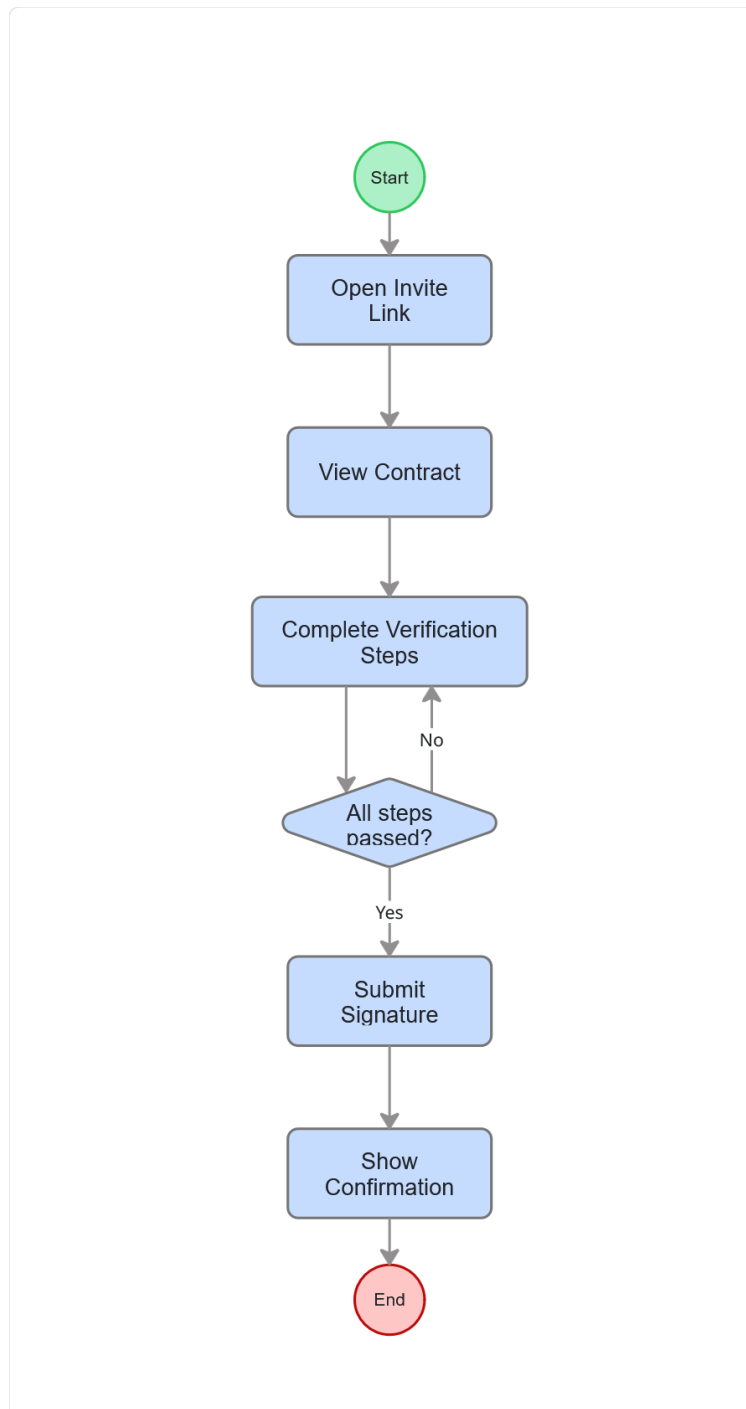


Figure 8 Activity diagram

Sequence Diagram

The sequence diagram covers two scenarios. In the first scenario, an invitee opens a valid invite link. The browser sends a GET request to the front controller. The Router resolves the route to the SignController. The SignController asks the Token library to validate the HMAC and check the expiry in the database. The Token library returns 'valid'. The SignController loads the contract, the party, and the verification requirements from the database, builds the stepper view, and returns HTML. The browser renders the page and the invitee starts the steps. Each step posts JSON to a verify endpoint, which writes a verification submission row, possibly stores an uploaded file, and returns 'passed' or 'failed'. The second scenario covers finalize: when the last submission passes, the FinalizeService computes the SHA-256 hash, the PdfService renders the PDF, the QrService generates the QR, and the Mailer sends a confirmation to the creator.

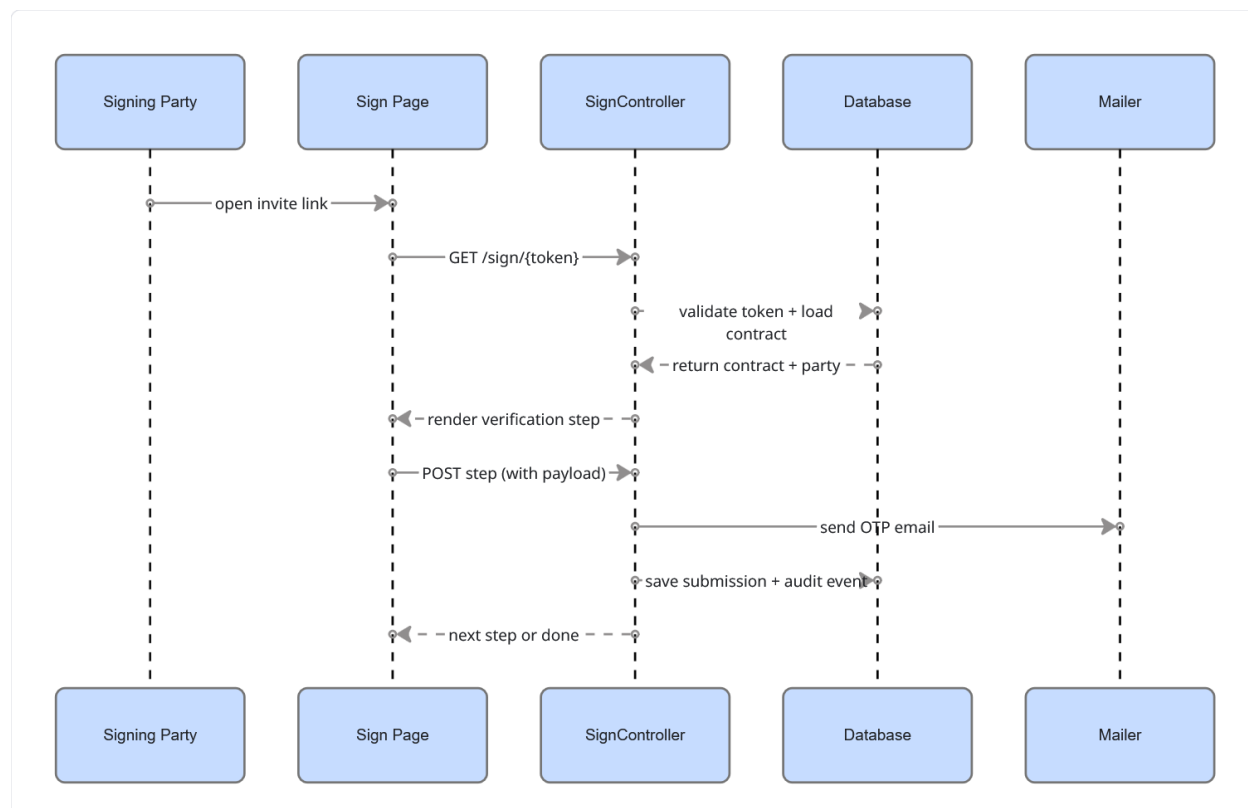


Figure 9 Sequence diagram

Class Diagram

The class diagram shows the main domain classes and their relationships. The User class represents a registered creator with id, name, email, password hash, and locale

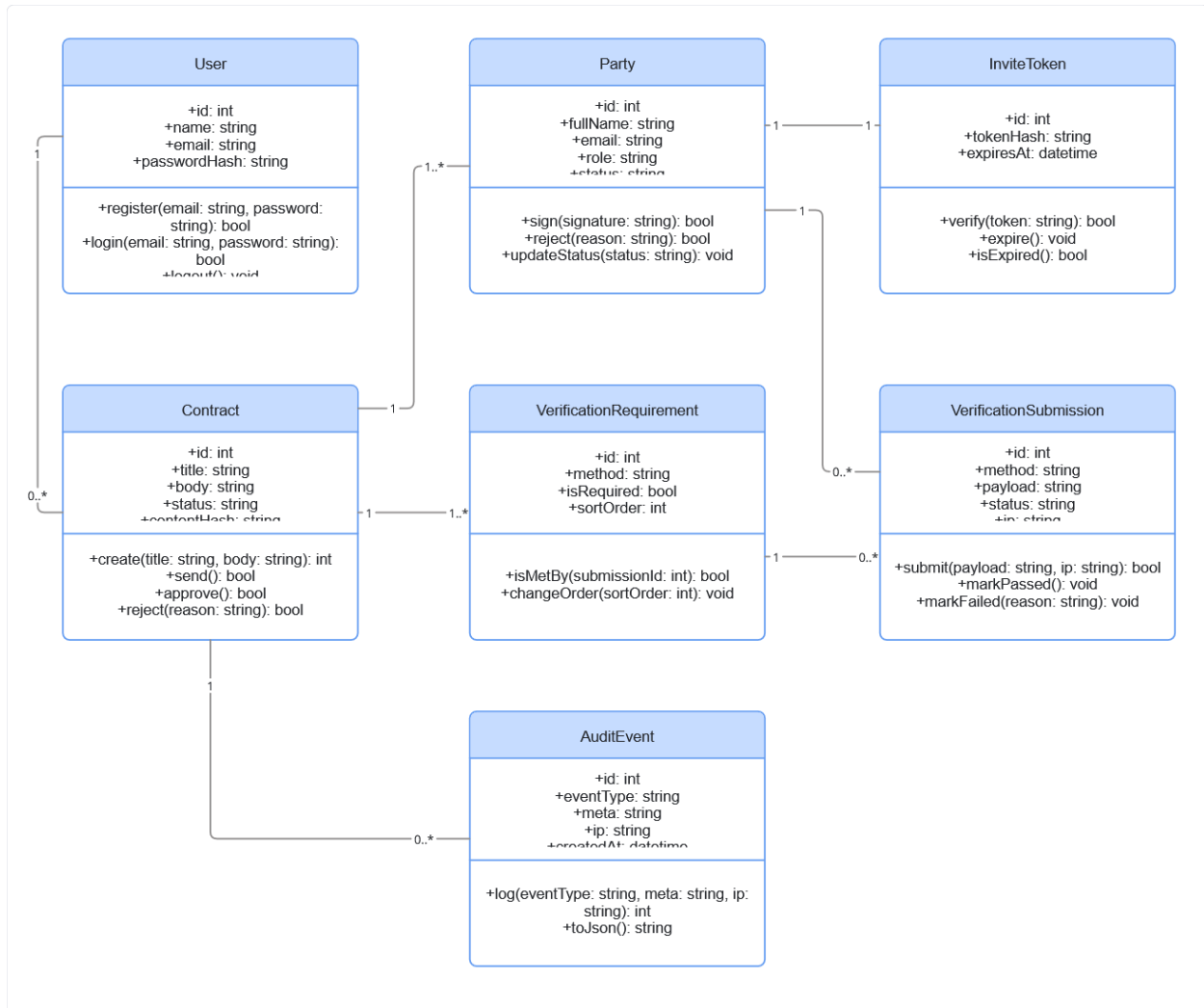


Figure 10 Class diagram

DFD Level 0 (Context Diagram)

The context diagram shows the platform as a single process with four external entities. The Creator sends contract drafts and verification choices and receives the dashboard, the evidence pack, and notifications. The Invitee sends verification submissions (signature image, selfie, ID, OTP, video) and receives the contract body, the stepper, and the final PDF. The Email Service receives outbound messages (invite, OTP, completion). The Public Verifier sends a contract id and receives a verification result that compares the stored hash to the recomputed one.

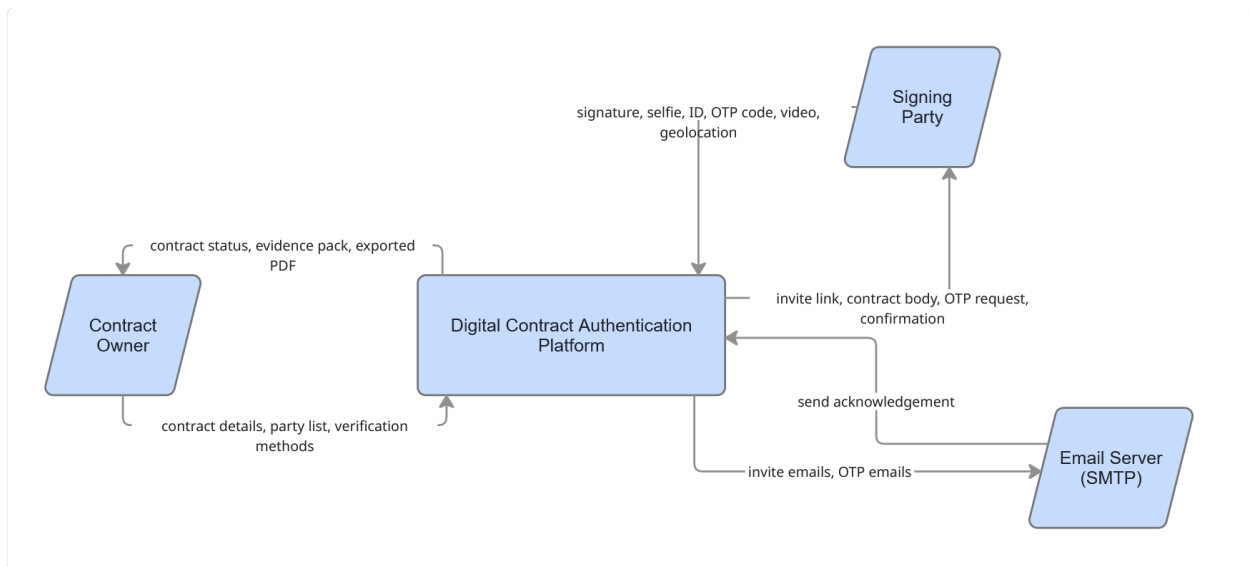


Figure 11 DFD Level 0 (Context Diagram)

Entity Relationship Diagram (ERD)

The ERD shows the eleven tables of the database and how they connect. The User entity has a one-to-many relationship with Contract (a creator can own many contracts).

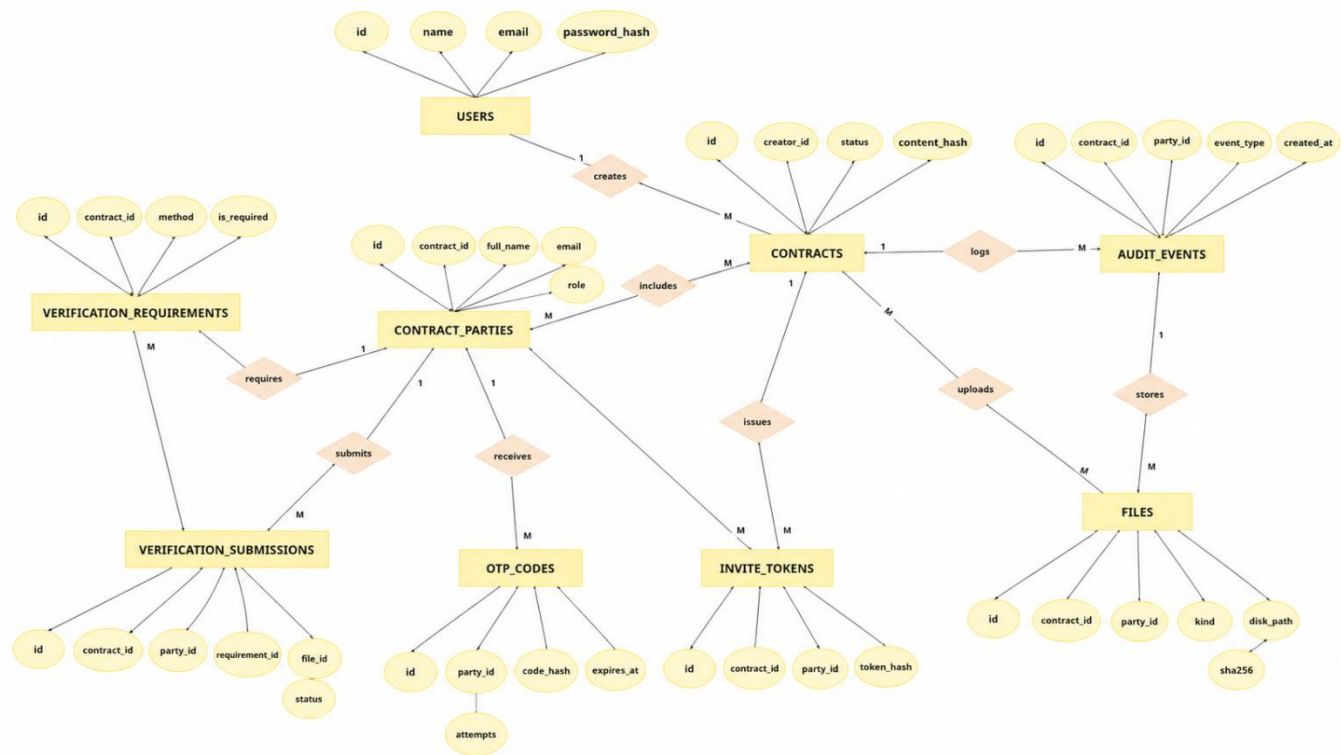


Figure 12 Entity Relationship diagram

Database Diagram

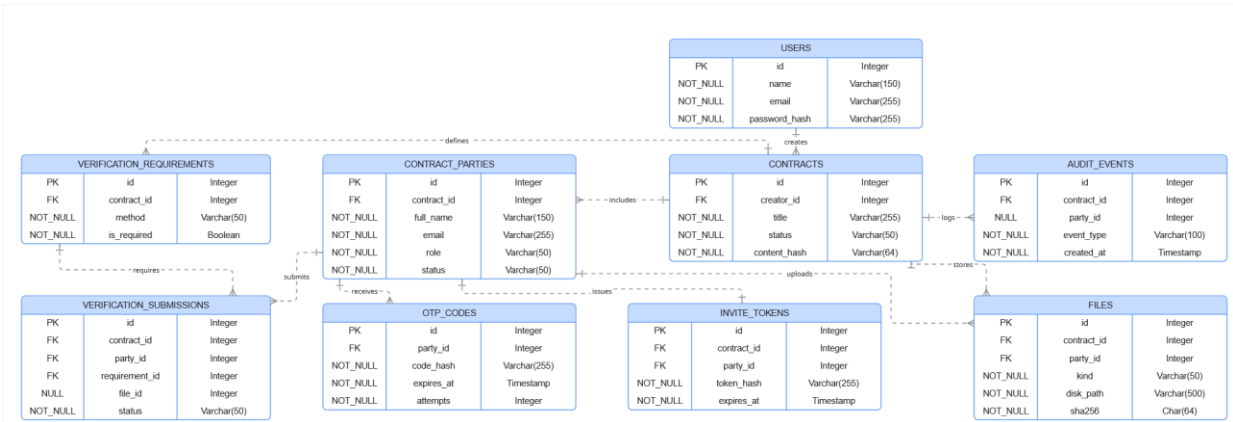


Figure 13 Database Diagram

3.12 Data Dictionary

The data dictionary describes each table and column in the MySQL schema. The DDL is the source of truth and lives in db/schema.sql. The tables below summarise the most important fields.

users

Column	Data type	Constraint	Description
id	BIGINT UNSIGNED	PK, auto increment	Primary key for the user.
name	VARCHAR(120)	NOT NULL	Display name of the creator.
email	VARCHAR(190)	NOT NULL, UNIQUE	Login email address.
password_hash	VARCHAR(255)	NULL	Bcrypt hash of the password.
email_verified_at	DATETIME	NULL	Set when the user confirms the email.
locale	CHAR(2)	NOT NULL, default 'ar'	UI language preference.
created_at	DATETIME	NOT NULL, default NOW()	Creation timestamp.
updated_at	DATETIME	NOT NULL, on update NOW()	Last update timestamp.

Table 13 Data dictionary: users

contracts

Column	Data type	Constraint	Description
id	BIGINT UNSIGNED	PK, auto increment	Primary key for the contract.
creator_id	BIGINT UNSIGNED	FK -> users(id)	Owner of the contract.
title	VARCHAR(255)	NOT NULL	Contract title shown in the dashboard.

Column	Data type	Constraint	Description
body_html	MEDIUMTEXT	NOT NULL	Quill-rendered HTML body of the contract.
body_text	MEDIUMTEXT	NOT NULL	Plain-text version of the body, used for hashing.
status	ENUM(...)	NOT NULL, default 'draft'	Contract state (draft, sent, completed, rejected, expired, canceled).
expires_at	DATETIME	NULL	Optional cut-off date.
content_hash	CHAR(64)	NULL	SHA-256 of the locked body, set on finalize.
finalized_at	DATETIME(3)	NULL	When the last party signed.
created_at	DATETIME	NOT NULL	Creation timestamp.
updated_at	DATETIME	NOT NULL	Last update timestamp.

Table 14 Data dictionary: contracts

contract_parties

Column	Data type	Constraint	Description
id	BIGINT UNSIGNED	PK	Primary key.
contract_id	BIGINT UNSIGNED	FK -> contracts(id)	The contract this party belongs to.
full_name	VARCHAR(160)	NOT NULL	Name of the invitee.
email	VARCHAR(190)	NOT NULL	Email used for the invite link and OTP.
phone	VARCHAR(40)	NULL	Optional phone number.

Column	Data type	Constraint	Description
role	ENUM('signer','witness','approver')	default 'signer'	Role of the party in the contract.
sign_order	SMALLINT UNSIGNED	default 1	Order in which the parties must sign.
status	ENUM(...)	default 'pending'	Party state (pending, viewed, in_progress, signed, rejected).
signed_at	DATETIME	NULL	When the party finished all steps.
rejected_at	DATETIME	NULL	When the party rejected.
reject_reason	TEXT	NULL	Reason given when rejecting.

Table 15 Data dictionary: contract_parties

verification_requirements

Column	Data type	Constraint	Description
id	BIGINT UNSIGNED	PK	Primary key.
contract_id	BIGINT UNSIGNED	FK	Contract this requirement belongs to.
method	ENUM(8 values)	NOT NULL	Which verification method is required.
is_required	TINYINT(1)	default 1	Whether the method is required (1) or optional.

Column	Data type	Constraint	Description
sort_order	SMALLINT UNSIGNED	default 0	Display order in the stepper.

Table 16 Data dictionary: verification_requirements

verification_submissions

Column	Data type	Constraint	Description
id	BIGINT UNSIGNED	PK	Primary key.
contract_id	BIGINT UNSIGNED	FK	Contract this submission belongs to.
party_id	BIGINT UNSIGNED	FK	Party who filled this submission.
requirement_id	BIGINT UNSIGNED	FK	Which requirement this submission is for.
method	ENUM(8 values)	NOT NULL	Method copied from the requirement (denormalised for queries).
payload_json	JSON	NULL	Method-specific data (geo coords, OTP attempt, hash of typed clause).
file_id	BIGINT UNSIGNED	FK -> files(id)	If the method produced a file, points to it.
status	ENUM('pending','passed','failed')	default 'pending'	Status of the submission.
completed_at	DATETIME	NULL	When the submission passed.
ip	VARCHAR(45)	NULL	IP address at submit time.

Column	Data type	Constraint	Description
user_agent	VARCHAR(500)	NULL	Browser user agent at submit time.

Table 17 Data dictionary: verification_submissions

files

Column	Data type	Constraint	Description
id	BIGINT UNSIGNED	PK	Primary key.
owner_user_id	BIGINT UNSIGNED	FK	Creator who owns the file (if any).
contract_id	BIGINT UNSIGNED	FK	Contract the file belongs to.
party_id	BIGINT UNSIGNED	FK	Party who uploaded the file.
kind	ENUM(7 values)	NOT NULL	What the file represents (signature_image, selfie, id_front, id_back, video_consent, drawn_signature_png, exported_pdf).
disk_path	VARCHAR(500)	NOT NULL	Relative path inside the storage volume.
mime	VARCHAR(100)	NOT NULL	MIME type of the stored file.
size_bytes	INT UNSIGNED	NOT NULL	File size.
sha256	CHAR(64)	NOT NULL	SHA-256 of the file content.
original_name	VARCHAR(255)	NULL	Original filename uploaded by the user.

Table 18 Data dictionary: files

audit_events

Column	Data type	Constraint	Description
id	BIGINT UNSIGNED	PK	Primary key.
contract_id	BIGINT UNSIGNED	FK	Contract this event belongs to.
party_id	BIGINT UNSIGNED	FK	Party that triggered the event (nullable).
actor_user_id	BIGINT UNSIGNED	FK	Creator that triggered the event (nullable).
event_type	VARCHAR(60)	NOT NULL	Event name (sent, viewed, started, step_passed, signed, rejected, finalized, downloaded).
meta_json	JSON	NULL	Extra context for the event.
ip	VARCHAR(45)	NULL	IP address at the time of the event.
user_agent	VARCHAR(500)	NULL	Browser user agent.
created_at	DATETIME(3)	NOT NULL	Millisecond-precision timestamp.

Table 19 Data dictionary: audit_events

Chapter 4: Design and Implementation

4.1 Overview

This chapter walks through the running platform. It starts with the high-level architecture and the signing flow, then shows each main screen on the creator side and on the invitee side, and explains what each one does. The screenshots are kept as placeholders so the team can paste the latest UI before submission.

System Architecture — Three-tier View

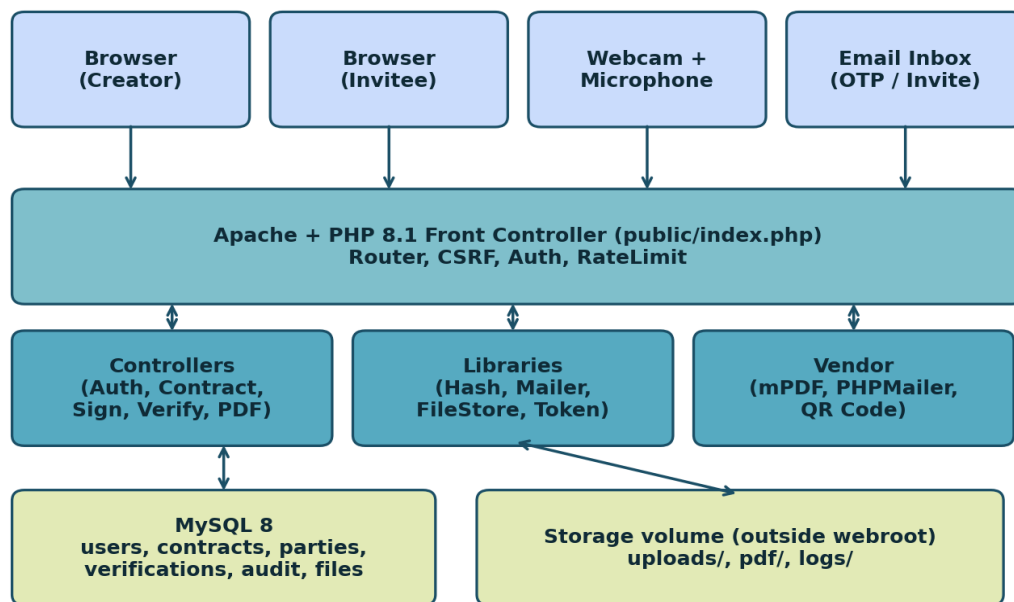


Figure 14 System architecture

The platform follows a classic three-tier layout. The browser handles the contract editor, the signature pad, the webcam, and the video recorder. Apache and PHP take care of routing, CSRF, authentication, rate limiting, file storage, and the verification flow. MySQL stores all relational data and the storage volume keeps the uploaded files outside the webroot. The third party libraries (mPDF, PHPMailer, endroid/qr-code) live in vendor/.

Invitee Signing Flow — Step-by-step Verification

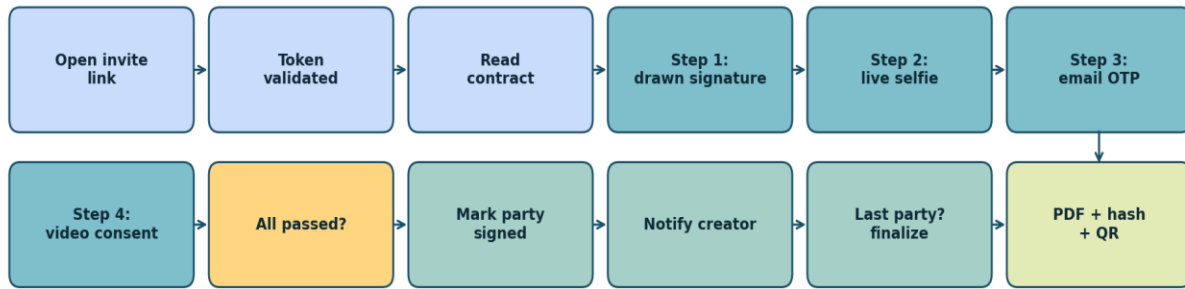


Figure 15 Invitee signing flow

The signing flow on the invitee side runs through a clear sequence: open invite link, validate token, read contract, run each verification step, mark party as signed, notify the creator, and finalize when the last party finishes.

4.2 Authentication

منصة العقود

Figure 16 Login page

4.3 Creator Dashboard

After login, the creator lands on a dashboard with four counters: drafts, sent, completed, and incoming (contracts where the creator is also a signer). Below the counters, a sortable table lists the latest contracts with title, status, parties, and last activity. A primary action button at the top opens the new-contract form.

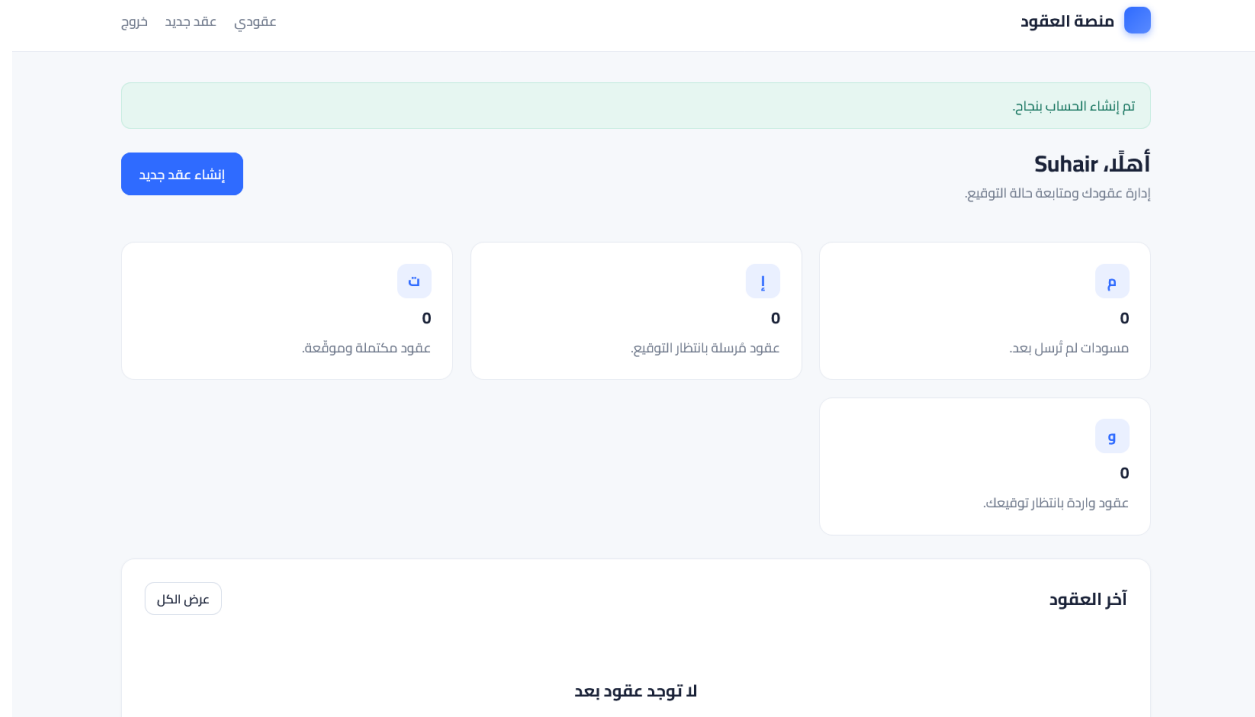


Figure 17 Creator dashboard

4.4 Contract Editor

The new-contract form is split into three blocks. The first block is the title and the rich-text body, edited with Quill 2.x in RTL mode. The second block is the parties: name, email, role (signer, witness, or approver), and sign order. The third block is the verification methods picker: eight checkboxes the creator can tick in any combination, each with a short description and the produced evidence type.

منصة العقود

عقودي عقد جديد خروج

إلغاء عقد جديد

عنوان العقد

مثال: عقد إيجار شقة سكنية

محتوى العقد

اكتب نص العقد هنا...

يمكنك التنسيق وإضافة عناوين وفواصل. لن تُحفظ السكريبتات.

تاريخ انتهاء صلاحية الدعوة (اختياري)

mm / dd / yyyy

Figure 18 Contract editor

طرق التحقق المطلوبة

حدّد الإجراءات التي يجب على الطرف الآخر إكمالها قبل أن يُحتسب توقيعه سارياً.

☐ صورة توقيع رفع صورة للتوقيع موجود مسبقاً.	☒ توقيع مرسوم يرسم الطرف توقيعه على لوحة بإصبعه أو القلم.
☐ وثيقة هوية رفع صورة الهوية الوطنية أو جواز السفر.	☐ صورة شخصية مباشرة التقاط صورة عبر الكاميرا لحظة التوقيع.
☒ نص موافقة بخط اليد كتابة جملة محدّدة لإثبات الموافقة الواعية.	☒ رمز تحقق بالبريد إرسال رمز سريّ إلى بريد الطرف يجب إدخاله للتوقيع.
☐ تسجيل فيديو موافقة تسجيل فيديو قصير ينطق فيه الطرف جملة الموافقة.	☒ الموقع وعنوان الإنترنت تسجيل صامت لإحداثيات GPS وعنوان IP عند التوقيع.

Figure 19 Verification methods picker

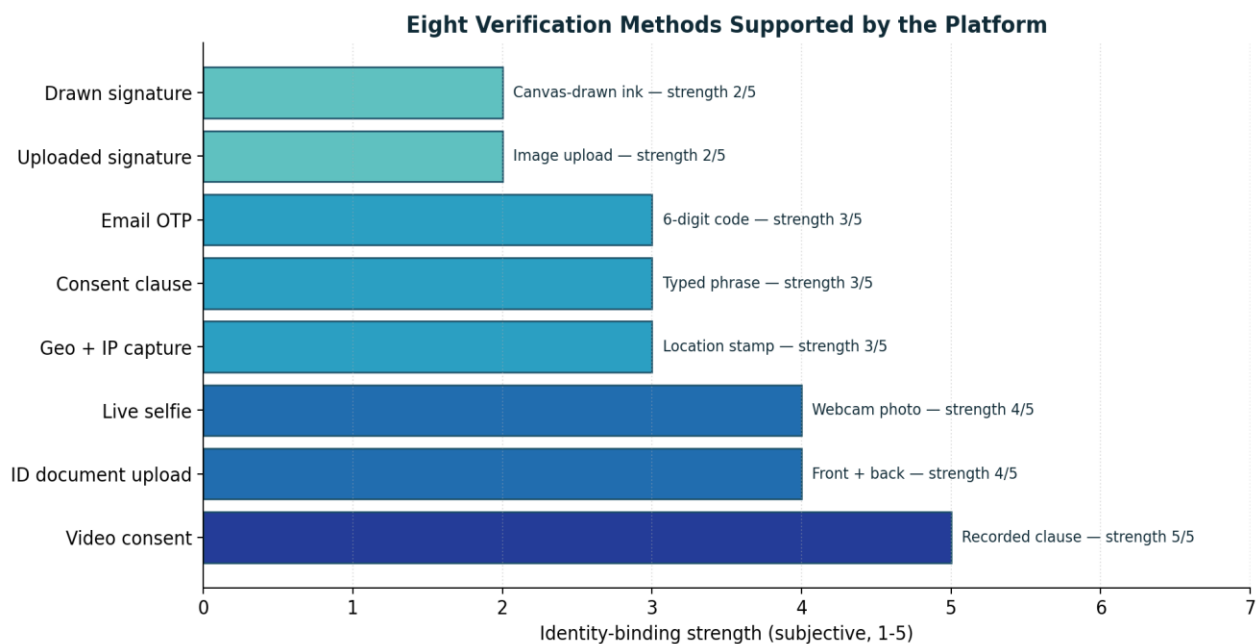


Figure 20 Verification methods overview

4.5 Invitee Landing Page

When the invitee opens the email link, the platform validates the HMAC token, checks the expiry, and serves the contract page. The page shows the contract title, the body in read-only mode, the party name, and a 'Start signing' button. A reject link is also visible so the invitee can stop early with a reason.

منصة العقود

إيجار شقة سكنية

ذُعت بصفة علي عبد القوي لمراجعة العقد وإكمال خطوات التحقق.

حذر هذا العقد بين كل من:

1- السيد / الجنسية: بطاقة رقم قومي/ إقامة:
(طرف أول - مؤجر)

2- السيد / الجنسية: بطاقة رقم قومي/ إقامة:
(طرف ثاني - مستأجر)

بعد أن أقر الطرفان بأهليتهما للتعاقد، اتفقا على التالي:

البند الأول: موضوع العقد

أجر الطرف الأول للمستأجر القابل لذلك ما هو شقة سكنية رقم (...) بالدور (...) بالعقار رقم (...) الكائن في:

الخطوات المطلوبة منك

عليك إكمال جميع الخطوات بالترتيب قبل تسجيل التوقيع.

1. توقيع مرسوم
2. صورة توقيع
3. صورة شخصية مباشرة
4. وثيقة هوية
5. رمز تحقق بالبريد
6. نص موافقة بخط اليد
7. الموقع وعنوان الإنترنت
8. تسجيل فيديو موافقة

رفض العقد

بدء خطوات التحقق

Figure 21 Invitee landing page

4.6 Verification Steps

4.6.1 Drawn Signature

The drawn signature step shows an HTML canvas with a dark stroke and a 'Clear' button. The invitee draws the signature with the mouse, finger, or pen. On 'Save', the canvas is exported to a PNG and uploaded to the file store. The file id is recorded in the verification submission.

الخطوة 1 من 8

توقيع مرسوم

صورة توقيع

صورة شخصية مباشرة

وثيقة هوية

رمز تحقق بالبريد

نص موافقة بخط اليد

الموقع وعنوان الإنترنت

تسجيل فيديو موافقة

توقيع مرسوم

ارسم توقيعك بإصبعك أو القلم في المساحة أدناه. سيتم حفظه كصورة.



مسح

تراجع

حفظ والمتابعة

العودة

Figure 22 Drawn signature step

Page 46 of 52

Live Selfie

The selfie step asks for camera permission, shows a live preview, and waits for the invitee to click 'Capture'. The captured frame is uploaded as a JPG with a 90% quality setting and a max width of 1200 px to keep file size reasonable.

The screenshot shows a registration form titled 'الخطوة 3 من 8' (Step 3 of 8). The form contains several buttons for different steps: 'رمز تحقق بالبريد' (Verify by email), 'وثيقة هوية' (Identity document), 'صورة شخصية مباشرة' (Live selfie) - which is highlighted in blue, 'صورة توقيع' (Signature photo), 'توقيع مرسوم' (Official signature), 'تسجيل فيديو موافقة' (Record video consent), 'الموقع وعنوان الإنترنت' (Website and internet address), and 'نص موافقة بخط اليد' (Handwritten consent text). Below these buttons, the section 'صورة شخصية مباشرة' (Live Selfie) is active, displaying a large black rectangle for the camera preview. A blue instruction box above the preview says 'سُيطلب منك السماح للكاميرا. التقط صورة شخصية واضحة لوجهك.' (You will be asked to allow the camera. Take a clear photo of your face.). Below the preview are buttons for 'تشغيل الكاميرا' (Turn on camera), 'التقاط الصورة' (Take photo), 'حفظ والمتابعة' (Save and continue) - highlighted in blue, and 'السابق' (Previous).

Figure 23 Live selfie step

ID Document Upload

The ID document step accepts a front image and an optional back image. The platform checks the MIME type, the size (max 5 MB per side), and the file extension. The images are stored in the same file store with the kind set to 'id_front' or 'id_back'. They are not OCR-processed in this version — that is a future improvement.

Figure 24 ID document upload step

Consent Clause

The consent clause step shows a fixed phrase such as 'I confirm that I have read and agreed to the contract above on my own free will.' The invitee types the phrase. The platform compares the typed text to the expected one ignoring leading or trailing whitespace and inner double spaces. A pass is recorded only if the strings match.

Geolocation and IP

The geo step asks the browser for the location through navigator.geolocation. If the user grants permission, the latitude, longitude, and accuracy are recorded in the payload_json column. The IP and user agent are always recorded by the server. If permission is denied, the step still passes but only with the IP fingerprint.

Uploaded Signature Image

Some users prefer to upload a previously prepared signature image. The platform accepts PNG or JPG up to 2 MB, validates the MIME type, stores the file, and links it to the verification submission.

Chapter 5: Conclusion

5.1 Summary

This project set out to build a contract platform that takes the proof of agreement seriously. The starting question was simple: if a paper contract gives both sides ink and witnesses, what should an electronic contract give them? The answer the team landed on is a stack of independent proofs — eight of them, configurable per contract, all bound to the same audit trail and the same SHA-256 hash on the final PDF.

Chapter 1 framed the problem and the scope. Chapter 2 explained the SDLC choice (Incremental) and the project timeline. Chapter 3 listed the requirements, the technologies, and the diagrams that describe the system. Chapter 4 walked through the running platform screen by screen. Together, these chapters describe a system that is small enough to host on a normal PHP/MySQL server and strong enough to stand up in a real dispute.

The key design decisions worth highlighting are the separation between requirement and submission tables, the immutable audit trail, the storage of files outside the webroot, the HMAC-signed invite tokens, and the QR-based public verification. Each one was a deliberate choice over a simpler shortcut, and each one earns its weight when the platform is challenged after the fact.

5.2 Future Work

- SMS OTP through a local provider, in addition to email OTP.
- Encryption of stored ID and video files at rest using a per-contract data encryption key.
- OCR on uploaded ID documents to extract name and number and match them to the party record.
- Liveness detection on the selfie step to block printed photos and screen replays.
- Federation with national digital identity providers for jurisdictions that offer them.
- Optional blockchain anchoring of the SHA-256 hash for contracts that need a public timestamp.
- Multi-tenant mode for organisations that want to host the platform for their teams.
- Mobile-first redesign of the stepper to handle small screens and patchy connectivity.

- Integration tests with a headless browser to lock the verification flow against regressions.
- A template marketplace where creators can pick from common contract templates and only fill the variables.

References

1. DocuSign. (2025). Electronic Signature — Learn About eSign Contracts.
<https://www.docusign.com/products/electronic-signature/learn/esign-contract>
2. Zoho. (2025). Online Contract Signature with Zoho Sign.
<https://www.zoho.com/sign/online-contract-signature.html>
3. eSignatures.com. (2025). eSignatures — Send and Sign Contracts Online.
<https://esignatures.com/>
4. The PHP Group. (2024). PHP Manual — PHP 8.1 Reference.
<https://www.php.net/manual/en/>
5. Oracle. (2024). MySQL 8.0 Reference Manual.
<https://dev.mysql.com/doc/refman/8.0/en/>
6. mPDF Team. (2024). mPDF — A PHP library generating PDF files from UTF-8 encoded HTML. <https://mpdf.github.io/>
7. PHPMailer. (2024). PHPMailer — The classic email sending library for PHP.
<https://github.com/PHPMailer/PHPMailer>
8. Quill.js. (2024). Quill Rich Text Editor. <https://quilljs.com/>
9. endroid. (2024). QR Code library for PHP. <https://github.com/endroid/qr-code>
10. Mozilla Developer Network. (2024). MediaRecorder API.
<https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder>
11. Mozilla Developer Network. (2024). Canvas API. https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
12. National Institute of Standards and Technology. (2015). Secure Hash Standard (FIPS 180-4).
13. Internet Engineering Task Force. (1997). RFC 2104 — HMAC: Keyed-Hashing for Message Authentication.
14. eIDAS Regulation. (2014). EU Regulation 910/2014 on Electronic Identification and Trust Services.
15. ESIGN Act. (2000). United States Electronic Signatures in Global and National Commerce Act.