Optomatica's GateID Technology Solution Architecture   
**Introduction**

### **Purpose**

This document serves as the technical solution architecture guide for the development and deployment of our integrated digital contract services platform. It outlines the architectural design, technological components, and interaction flows that constitute the foundation of our digital contract signing system, ID recognition and matching eKYC system, administrative portal, and user’s mobile application. This document is intended to provide our development team, stakeholders, and future contributors with a comprehensive understanding of the system's structure and operational mechanics.

### **Scope**

The scope of this architecture encompasses the design and implementation strategies for:

* A digital ID and signing system that facilitates secure user onboarding and contract management processes.
* A state of the are eKYC system for detecting national in captured images and verify the authenticity of the captured national id document. Also It reads the text in the ID using OCR technology as well as matching ID photo with captured selfie image. It also detect fraud ID images cases.
* An administrative portal designed for tracking, analytics, and business intelligence functionalities of both systems.
* A mobile application (or SDK in partner mobile application) to capture the ID and validate selfie liveness and validate users in all onboarding steps including phone and email OTPs.

Each component is built to ensure stability, security, and cloud deployability, allowing for scalability and ease of maintenance.

## **Architectural Overview**

## Techstack Rational

The tech stack was selected with the following criterion in mind:

1. Well established and widely open source technologies
2. Stability and security
3. Cloud deployability
4. Easy to modify, customize and expand

## Techstack elements

Towards the aforementioned ends we selected the following elements:

1. [PostgreSQL](https://www.postgresql.org/) for the database
2. [Nest.js](https://nestjs.com/) framework for backend running on top of node.js
3. [FastApi](https://fastapi.tiangolo.com/) framework for eKYC backend running on top of python.
4. [React](https://reactjs.org/) for the front end
5. [Flutter](https://flutter.dev/) for mobile apps
6. [ScyllaDB](https://www.scylladb.com/) for date replication and disaster recovery

Such arrangement allows for the rapid development of the core technology using a small team of full stack developers.

With that tech stack we build an eKYC backend that handles:

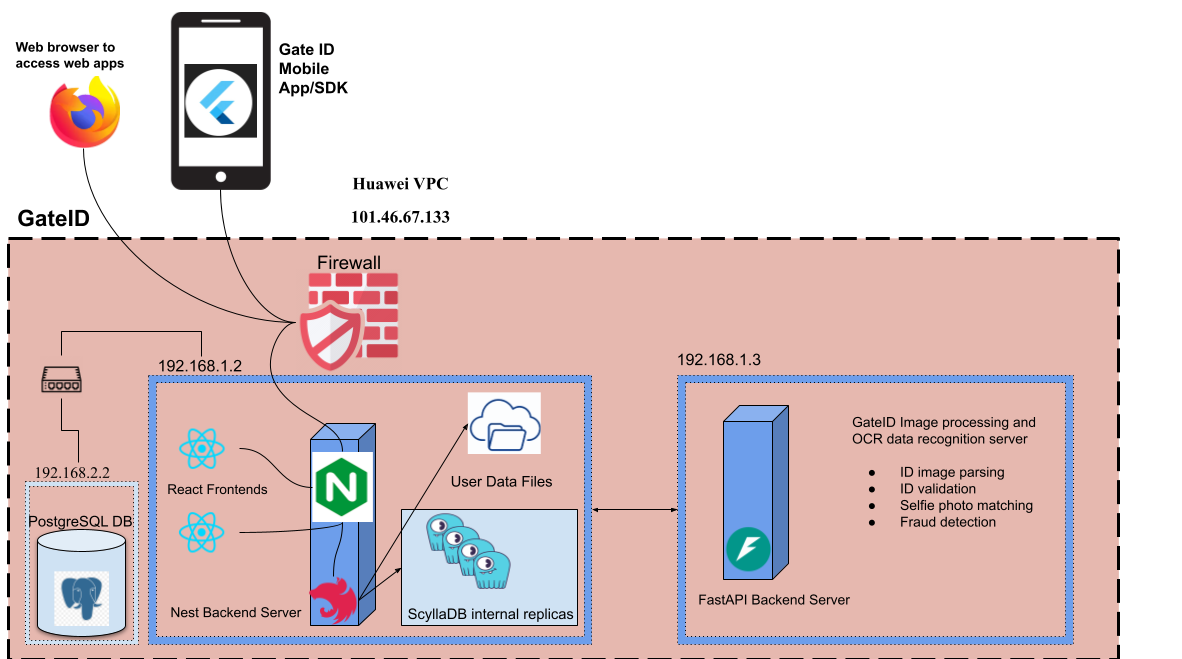
1. ID scanning, verification and selfie matching.
2. Admin portal that can be used to track all scanned id attempts with their statuses as well as providing analytics and BI functionalities

Additionally we built digital ID/signing system that comprises

1. A resilient backend the manages user IDs, contracts, and contract states
2. An App to handle client onboarding and signing

We can broadly divide the system into two major subsystems.

1. The GateID eKYC subsystem which contains the infrastructure for the ID & selfie images processing
2. The GateID backend subsystem which contains all elements related to digital identity and contract e-signing.



This is a high level network security overview of the GateID system as it currently stands. Backends, PostgresDB and ScyllaDB are hosted on Huawei Cloud in Egypt.

The two machines are protected by Kaspersky Endpoint Security which includes advanced threat protection and EDR and Cloudflare for the WAF. Both systems maintain logs of all transactions.

The GateID system communicates with the GateID eKYC system securely by a local connection and there are additional recovery mechanisms via record and document replication using ScyllaDB.

Our infrastructure is designed with a strong emphasis on security and compliance. We currently run a PostgreSQL database on a NestJS backend, all of which are securely hosted behind a firewall provided by Huawei, our hosting provider. The inclusion of a Web Application Firewall (WAF) adds an additional layer of security, safeguarding our web applications against common threats such as SQL injection and cross-site scripting (XSS). The firewall further enhances our security posture by regulating network traffic and ensuring that only authorized access is permitted to our backend services.

All our data is securely stored within Egypt, adhering to local data residency laws and regulations. This ensures that customer data, including personal information, remains within national borders, aligning with compliance requirements and enhancing trust with our stakeholders.

While our infrastructure does not include a dedicated call center or Service Level Agreements (SLAs) with our open-source database and framework providers, these elements are not currently relevant to our operational model. Our use of open-source technologies like PostgreSQL, FastAPI and NestJS is intentional, leveraging their robust community support and extensive functionality. The absence of SLAs does not impact our ability to provide reliable and secure services, as our focus remains on maintaining a secure and compliant environment through best practices and rigorous security measures.

Furthermore, the need for a call center is not a priority at this stage, given our current operational scale and the nature of our customer interactions. Our existing support mechanisms are sufficient to handle customer inquiries effectively, and we continue to prioritize digital communication channels that align with our service model.

Overall, our infrastructure is well-equipped to meet the security and compliance needs of our organization. The elements not currently in place, such as a call center or specific SLAs for open-source tools, do not affect our ability to provide secure and compliant services at this point in time. We remain committed to evaluating and adapting our strategies to ensure they align with our growth and evolving regulatory requirements.

## Data Architecture

### GateID database

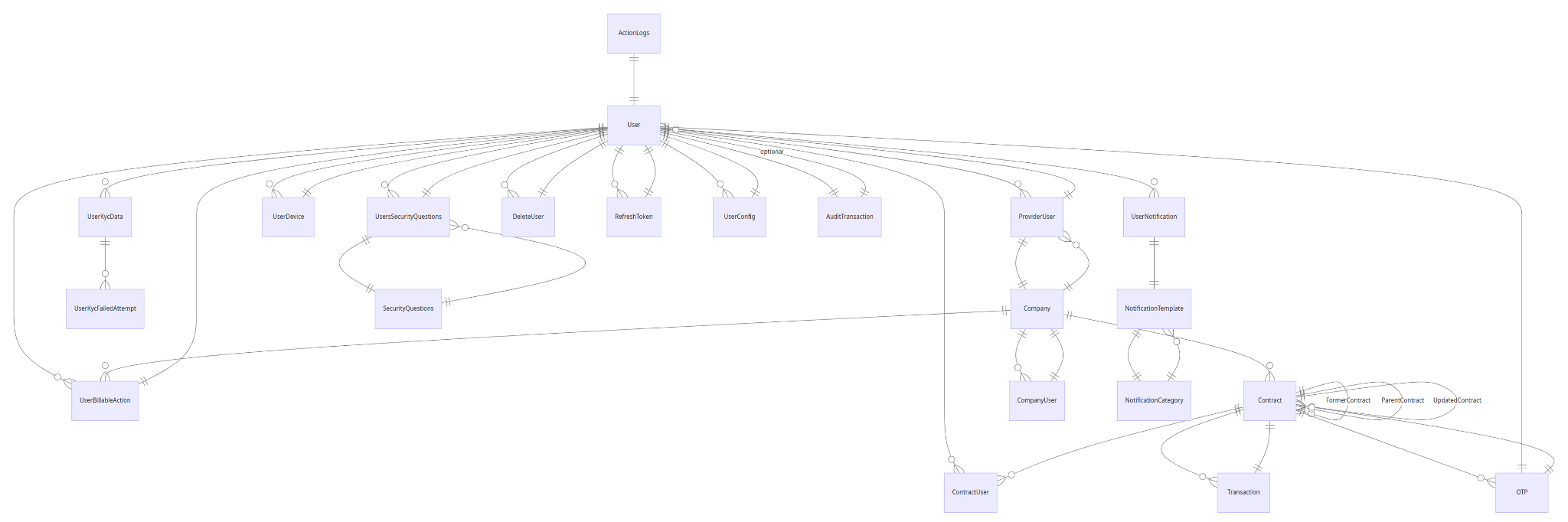
#### Models

1. **User**: Represents an individual user with details like username, password, national ID, phone number, email, and device information. Additional attributes include geolocation, IP address, and verification statuses.
2. **Contract**: Manages contractual agreements including details such as title, description, and association with a company. Supports unique identifiers and hash for security, and it is linked to transactions, users, and related contracts.
3. **ContractUser**: Links users to contracts and stores related information like user and contract identifiers, phone number, device ID, and user status in the contract context.
4. **Company**: Represents a business entity with unique identifiers and details such as name, tax card ID, and other identifying documents. Companies can have multiple contracts and users associated with them.
5. **OTP**: Handles one-time passwords for authentication, associated with users and optionally linked to contracts. It records the OTP type, user details, and expiration times.
6. **Transaction**: Represents financial or other types of transactions related to a contract, including creation and update timestamps.
7. **RefreshToken**: Manages session refresh tokens for users, ensuring each token is unique and tied to a specific user.
8. **CompanyUser**: Represents a user associated with a specific company, including unique user details and verification statuses.
9. **ProviderUser**: Represents users linked to external providers, storing details like client and company IDs, and optional user information.
10. **AdminUser**: Defines an administrative user with unique login credentials and email, including lock status and timestamps.
11. **UserKycData**: Manages Know Your Customer (KYC) data for users, including identification documents and verification statuses, linked directly to a user.
12. **UserKycFailedAttempt**: Records failed KYC attempts, storing error details and the specific step that failed.
13. **UserConfig**: Stores user-specific configurations, like preferred application language.
14. **SecurityQuestions**: Manages security questions for user verification processes.
15. **UsersSecurityQuestions**: Links security questions to specific users, including their answers.
16. **AuditTransaction**: Logs changes made within the system, recording details like the modified model, field, new value, and associated user.
17. **ActionLogs**: Logs user actions within the system, including details like the action type, associated user, and geographic location.
18. **NotificationCategory**: Categorizes notifications and links them to templates for actual notification messages.
19. **NotificationTemplate**: Defines templates for sending notifications to users, including title, body, and other options.
20. **UserNotification**: Manages notifications sent to users, including read and seen statuses.
21. **UserDevice**: Represents devices associated with users, storing details like device IDs and tokens.
22. **UserBillableAction**: Tracks billable actions performed by users, such as KYC processes or contract signings.
23. **DeleteUser**: Manages the record of users who have been deleted from the system.

These models collectively support a robust system for managing user identities, contracts, companies, and transactions with emphasis on security and traceability.

#### ERD

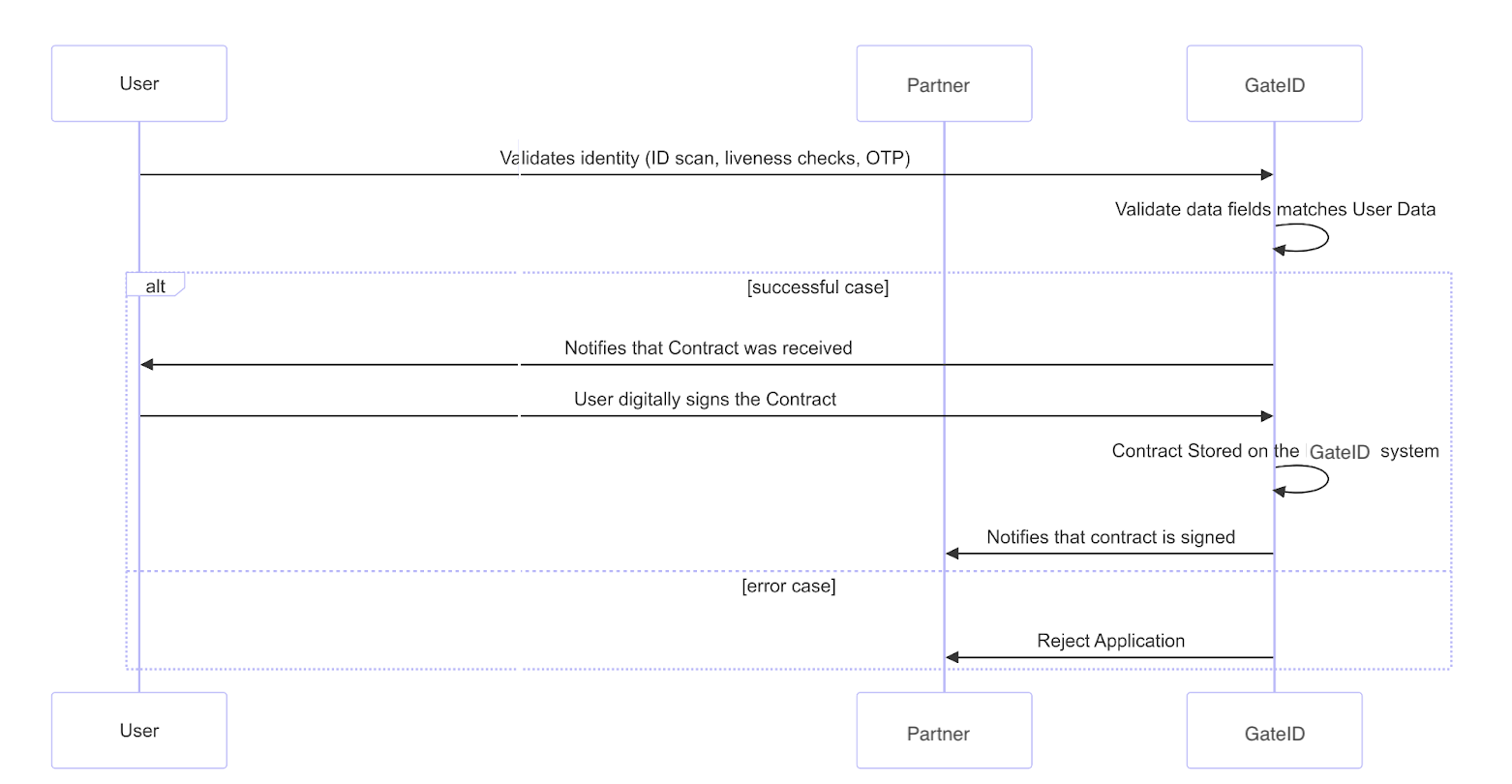
Here we are showing the relevant relationship between the models, a full listing of all the attributes is beyond the scope of this document.



## GateID onboarding and contract signing Processes

Typically GateID works with partners that need to onboard a customer digitally.

The below process diagram shows the basic case where the user is onboarded on the system until the customer signs the contract.



When a user initiates to sign a contract from the partner system, the process is as follows:

1. The partner sends the user's data (user ID, national ID, contract, etc…) to GateID.
2. The user is prompted to create a digital ID on GateID.
3. The user downloads the GateID app or GateID SDK on the partner app and validates their identity through ID scanning, liveness checks, and OTP verification.
4. GateID validates that the data fields match the user data provided.
5. If successful, the user is notified of the contract and can digitally sign it.
6. GateID stores the signed contract and notifies the partner of the signing.

Data Storage:

* In GateID: User identity information and signed contracts are stored in a centralized blockchain-like system.

The system's security architecture ensures that communications between Partner system and GateID are secure, using SSL encryption and protected by distributed firewalls. The critical contract data and digital signing records in GateID are only accessible by the Partner system using strong authentication and authorization.

Additionally, all records are replicated on ScyllaDB for disaster recovery and added safeguards. This replication ensures data integrity and provides a mechanism for fine-grained recovery if needed.

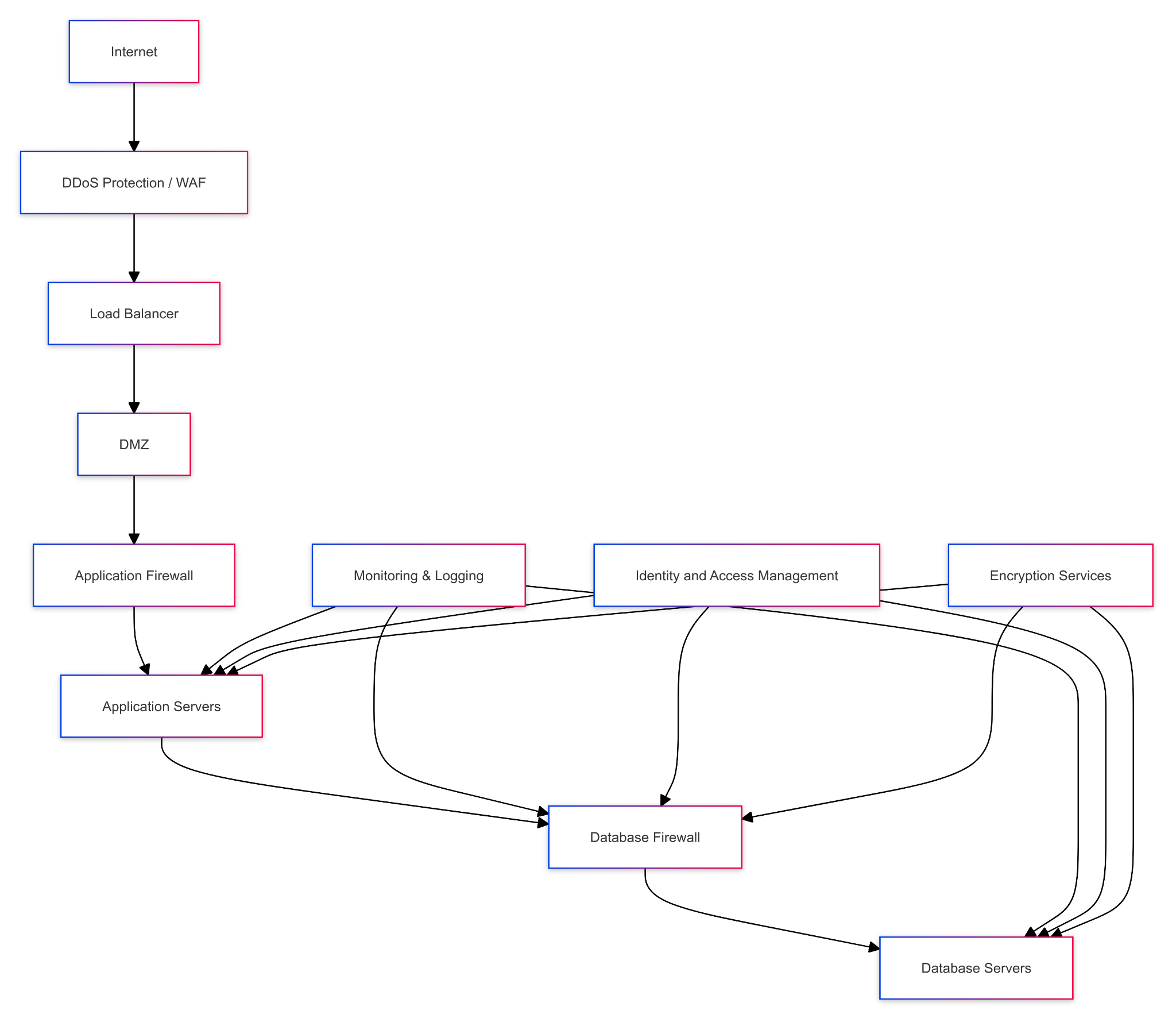
The system's design provides a robust, secure, and scalable architecture for managing contracts and user identities in various scenarios.

## 

## Security Architecture and CyberSecurity

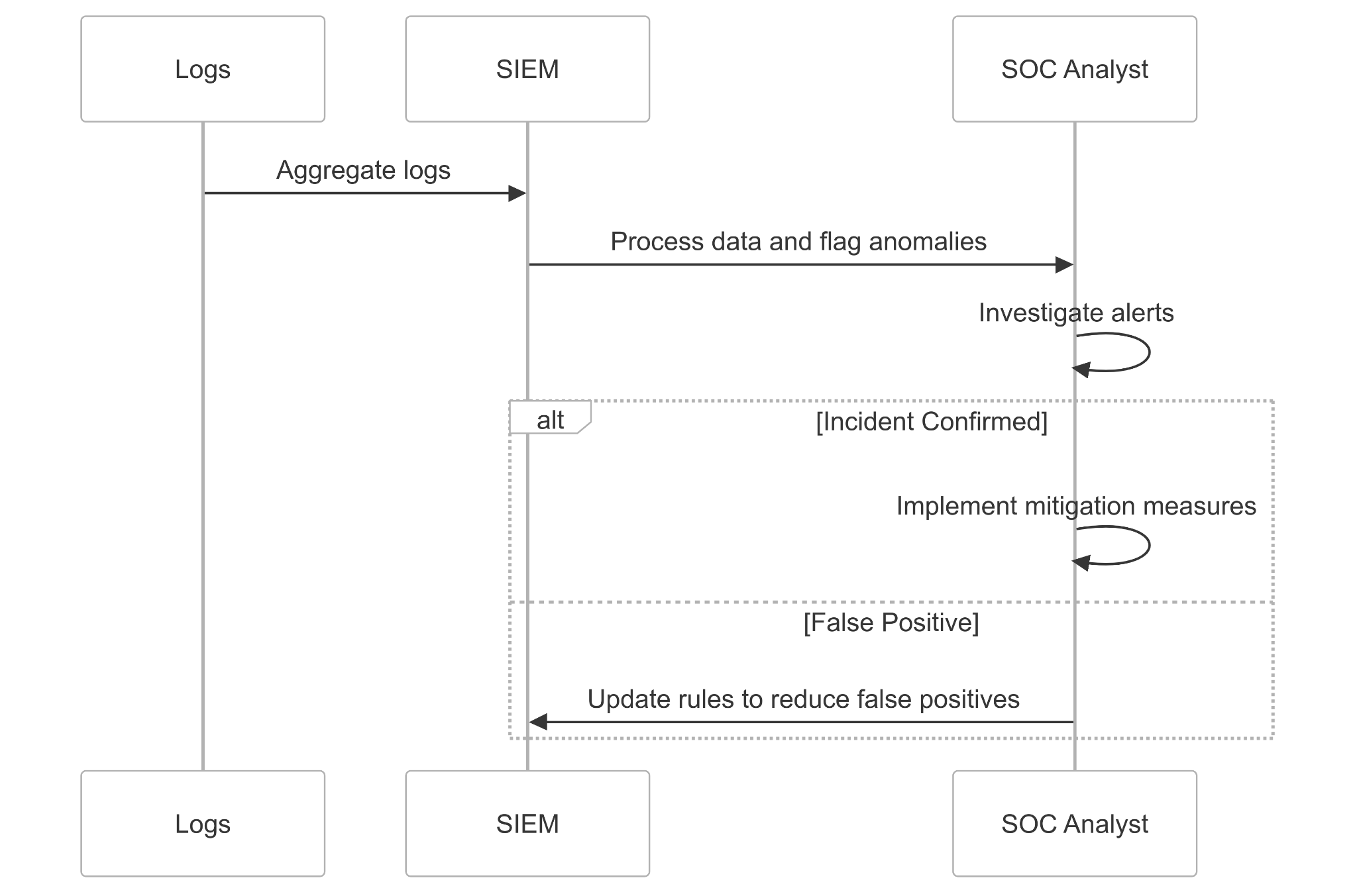
Both backends GateID and GateID eKYC communicate via RESTful interface.The RESTful interface is rigorously protected through multiple layers of security measures to ensure both the integrity and confidentiality of data. Firstly, access to the interface is exclusively available over SSL (Secure Sockets Layer), which encrypts the data transmitted between clients and the server, thus safeguarding against eavesdropping and tampering by unauthorized entities. This SSL implementation ensures that all communications maintain a high standard of security, including authentication and data integrity checks. Additionally, they are communicating strictly using the local network and not via the open internet. This advanced network security measure is strategically designed to detect and mitigate potential threats from various sources, providing a robust barrier against a wide array of cyber attacks. By integrating these security mechanisms, the architecture not only secures the data transmission but also robustly defends the server endpoints from malicious intrusions and disruptions.

Furthermore we have network security rules in place that only allows the Partner backend to communicate with the GateID backend. The critical contract data and digital signing records are not reachable by any other IP other than that of the Partner backend.

The security architecture for the GateID ecosystem, and the GateID App/SDK, employs a multi-layered approach to protect against various cyber threats while ensuring regulatory compliance. This strategy involves implementing multiple security controls throughout the system architecture.  
  


As illustrated in the Multi-layered Security Architecture diagram above, the system incorporates several key components:

1. DDoS Protection and Web Application Firewall (WAF) at the perimeter, shielding both GateID systems from volumetric attacks and common web vulnerabilities.
2. A Demilitarized Zone (DMZ) for public-facing services of both systems, providing an additional layer of separation from the internet.
3. Application firewalls to protect GateID servers from application-layer attacks.
4. Continuous monitoring and logging across all layers of both systems, providing real-time visibility into activities and potential security events.
5. Encryption services for data in transit and at rest, protecting sensitive contract and identity information handled by GateID, and the GateID App/SDK.



Our security monitoring and incident response capabilities are designed to detect and respond to threats in real-time across GateID, and the GateID App/SDK. As shown in the updated Security Monitoring and Incident Response Flow diagram, this process includes:

1. Centralized log aggregation from GateID, and GateID App/SDK components.
2. A Security Information and Event Management (SIEM) system that processes the aggregated data and flags potential anomalies or security events.
3. SOC analysts who investigate the alerts generated by the SIEM.
4. A feedback loop where SOC analysts update SIEM rules based on investigation outcomes to improve detection accuracy and reduce false positives.

When an incident is confirmed, the SOC analysts are responsible for implementing appropriate mitigation measures. This could involve actions such as isolating affected systems, blocking malicious IP addresses, or initiating predefined incident response procedures depending on the nature and severity of the incident.

This streamlined approach ensures that potential security issues are promptly detected and addressed, while continuously improving the system's ability to distinguish between genuine threats and false alarms.

We implement strong encryption measures for data at rest and in transit across all GateID systems. This includes the sensitive identity and contract information in GateID.

The GateID App implements secure authentication mechanisms to ensure secure access to user accounts and sensitive information.

Our disaster recovery strategy relies heavily on the replication of critical records to ScyllaDB, providing an additional layer of data protection and enabling efficient recovery capabilities in case of system failures or data loss incidents.

A key component of our data protection and disaster recovery strategy is the use of ScyllaDB. As mentioned in the original document, all critical records, including contracts and digital signing records, are replicated on ScyllaDB. This replication provides an additional layer of data protection and enables fine-grained disaster recovery capabilities.

The centralized blockchain-like system implemented on PostgreSQL for GateID, which stores contracts and digital signing records, is further secured by ensuring that it's only accessible by the Partner backend. This restricted access, combined with the ScyllaDB replication, provides a robust security model for protecting critical identity and contract data.

This comprehensive security architecture provides robust protection for the entire GateID ecosystem, addressing security at multiple layers from network infrastructure to application security for GateID, and the GateID App/SDK. It includes advanced measures for monitoring, incident response, and data protection, ensuring the integrity and confidentiality of both loan management and digital identity services.

### Centralized Blockchain: Implementing a Blockchain-like System for GateID

#### Overview

A "Centralized Blockchain" system built on PostgreSQL leverages the database's robustness to create a secure, verifiable chain of records, akin to a blockchain. This system is designed primarily for storing contracts as PDF documents, where each contract is linked to its predecessor via cryptographic hashes, ensuring data integrity and tamper resistance.

### **System Design**

#### **1. Database Schema**

The PostgreSQL database has a table structured to hold each contract's details, including a unique identifier, the contract's PDF stored as a binary large object (BLOB), a cryptographic hash of the contract, and a reference to the hash of the previous contract in the chain.

For example, this can be expressed in SQL as

CREATE TABLE contracts (

id SERIAL PRIMARY KEY,

contract\_data BYTEA NOT NULL,

contract\_hash CHAR(64) NOT NULL,

previous\_hash CHAR(64),

timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

#### **2. Generating Cryptographic Hashes**

Each contract's PDF will be hashed using a cryptographic hash function, such as SHA-256. This hash ensures that any alteration of the contract can be easily detected. The hash of each new contract includes:

* The hash of the contract's PDF document.
* The hash of the previous contract in the chain.

This chaining reinforces the security, as tampering with any contract would require recalculating all subsequent hashes, which is computationally prohibitive.

#### **3. Inserting a New Contract**

When a new contract is added:

* The PDF document is first converted to a binary format.
* The binary data is then hashed using SHA-256.
* The system retrieves the hash of the latest contract in the database to be used as the previous hash.
* The new contract, along with its hash and the hash of the previous contract, is inserted into the database.

sql

INSERT INTO contracts (contract\_data, contract\_hash, previous\_hash)

VALUES (

$pdf\_binary,

encode(digest($pdf\_binary, 'sha256'), 'hex'),

(SELECT contract\_hash FROM contracts ORDER BY id DESC LIMIT 1)

);

#### **4. Verifying Contract Integrity**

To verify the integrity of the blockchain:

* Start from the first contract.
* Recalculate and verify each contract's hash by using its PDF and the previous contract's hash.
* Any discrepancy in the calculated hashes would indicate a tampering incident.

#### **5. Security Measures**

* **Access Control:** Restrict database access using PostgreSQL's role-based access control to ensure only authorized users can insert or query the contracts.
* **Audit Trails:** Maintain logs of all queries and modifications.
* **Regular Backups:** Ensure data recovery capabilities through regular backups.

#### **6. Advantages of Centralized Blockchain**

* **Reliability and Scalability:** Leverage PostgreSQL’s reliability and scalability.
* **Auditing and Compliance:** Provides a clear, immutable history of contractual agreements suitable for auditing.
* **Data Integrity:** Ensures data has not been tampered with, providing a secure way to manage contracts.

This centralized blockchain model on PostgreSQL provides an innovative approach to managing contracts with high integrity and security. By using cryptographic hashes and chaining, it offers a tamper-proof system, albeit in a centralized manner, leveraging the traditional strengths of relational databases for managing complex datasets efficiently. The

The contracts table is additionally configured to be write-only. Incorporating write-only access into the Centralized Blockchain system significantly enhances its security and integrity. This approach, while restrictive, aligns well with the principles of data immutability and security inherent in blockchain technology. However, it also necessitates robust role management and auditing mechanisms to ensure that the system remains both secure and functional.

Finally, all those records are also replicated on ScyllaDB for added safeguards and fine grained disaster recovery.

### Optomatica Software Development Methodology

At Optomatica, we employ a robust Agile software development methodology based on Scrum, complemented by continuous integration and deployment (CI/CD) practices. This approach ensures rapid, iterative development while maintaining high standards of code quality and system reliability.

### Scrum Framework

Our development teams work in two-week sprints, allowing for frequent releases and adaptability to changing requirements. Each sprint begins with a planning session where the team selects items from the product backlog, breaking them down into manageable tasks. Daily stand-up meetings keep the team aligned and help identify any blockers quickly.

### Continuous Integration (CI)

We have implemented a comprehensive CI/CD pipeline that automates the build, test, and deployment processes for GateID systems:

1. **Version Control**: All code changes are managed through Git, with separate repositories for GateID and GateID eKYC components.
2. **Automated Builds**: Upon each code commit, our CI server (Jenkins) automatically triggers a build process, ensuring that new changes integrate smoothly with the existing codebase.
3. **Automated Testing**: Our CI pipeline includes several layers of automated testing:
   * Unit tests for individual components
   * Integration tests to verify interactions between different modules
   * End-to-end tests simulating user workflows
   * Security scans to identify potential vulnerabilities
4. **Code Quality Checks**: We use static code analysis tools to maintain code quality and adherence to our coding standards.

### Testing Strategy

1. **Unit Testing**:
   * Implement comprehensive unit tests for GateID
   * Use testing frameworks appropriate for each component (e.g., Jest for NestJS)
   * Aim for high test coverage, especially for critical business logic
2. **Integration Testing**:
   * Develop integration tests to verify interactions between GateID and Partner backends
   * Test API endpoints and data flow between systems
   * Simulate various scenarios, including error conditions and edge cases
3. **End-to-End Testing**:
   * Implement E2E tests using tools like Cypress or Selenium
   * Cover critical user journeys, such as user onboarding and loan application processes
   * Include mobile app testing for the GateID App using appropriate mobile testing frameworks
4. **Performance Testing**:
   * Conduct regular performance tests to ensure system scalability
   * Use tools like Apache JMeter to simulate high loads on GateID
   * Monitor database performance, especially for the centralized blockchain-like system in PostgreSQL
5. **Security Testing**:
   * Perform regular security audits and penetration testing
   * Focus on API security, authentication mechanisms, and data encryption
   * Verify the integrity of the contract storage system in GateID

### Deployment Process

Our deployment strategy follows a staged approach:

1. **Development Environment**: Developers work locally and push changes to feature branches.
2. **Test Environment**: Successfully built and tested code is automatically deployed to our test environment, which closely mimics the production setup. This environment is used for manual QA and user acceptance testing.
3. **Staging Environment**: Prior to production deployment, changes are pushed to a staging environment for final verification, including performance testing and security audits.
4. **Production Environment**: Once approved, changes are deployed to production using a blue-green deployment strategy to minimize downtime and risk.

### Monitoring and Feedback

Post-deployment, we utilize advanced monitoring tools to track system performance and user behavior. This data feeds back into our development process, informing future sprints and helping us continuously improve our services.

### Security Integration

Given the sensitive nature of contract and identity data handled by GateID, security is deeply integrated into our development process:

* Regular security training for all developers
* Automated security scans as part of the CI pipeline
* Manual penetration testing conducted on a quarterly basis
* Continuous monitoring for potential security threats

This comprehensive approach to software development ensures that we can rapidly deliver new features and improvements to GateID while maintaining the highest standards of security and reliability. It also allows us to quickly respond to changing regulatory requirements and market demands in the fintech sector.

## Optomatica Risk Framework: NIST CSF Implementation

### Overview

At Optomatica, we recognize the critical importance of cybersecurity in the fintech industry. To address this, we are gradually implementing the NIST Cybersecurity Framework (CSF) to enhance our risk management practices. This framework provides a comprehensive approach to managing and reducing cybersecurity risk by addressing five core functions: Identify, Protect, Detect, Respond, and Recover.

### Phased Implementation

Our implementation of the NIST CSF is being carried out in phases, allowing us to systematically improve our cybersecurity posture while maintaining operational efficiency.

#### Phase 1: Identify and Protect (Current Phase)

1. **Asset Management**:
   * We have cataloged all hardware and software assets related to GateID
   * Data flows between systems are mapped and documented
2. **Access Control**:
   * Role-based access control (RBAC) is implemented across all systems
   * Multi-factor authentication (MFA) is enforced for all administrative access
3. **Data Security**:
   * Encryption is applied to data in transit
   * Our centralized blockchain-like system for contract storage enhances data integrity
4. **Information Protection Processes and Procedures**:
   * Security policies and procedures are documented and communicated to all staff
   * Regular security awareness training is conducted for all employees

#### Phase 2: Detect (Next Phase)

1. **Anomalies and Events**:
   * We are in the process of implementing advanced log analysis tools
   * Behavior analytics will be integrated to detect unusual patterns
2. **Security Continuous Monitoring**:
   * Plans are in place to expand our monitoring capabilities
   * We will implement real-time alerts for potential security incidents
3. **Detection Processes**:
   * We are developing processes to ensure timely awareness of anomalous events

#### 

#### Phase 3: Respond and Recover (Future Phase)

1. **Response Planning**:
   * We will develop and test incident response plans
   * Table-top exercises will be conducted to simulate various cyber incidents
2. **Communications**:
   * Processes for reporting incidents to relevant stakeholders will be established
   * We plan to implement secure communication channels for use during incidents
3. **Recovery Planning**:
   * Disaster recovery plans will be developed and tested
   * We will establish metrics to evaluate the effectiveness of recovery efforts

### Continuous Improvement

Our implementation of the NIST CSF is an ongoing process. We regularly:

* Assess our current security posture against the framework
* Identify gaps and areas for improvement
* Update our risk management strategies based on new threats and vulnerabilities

By gradually implementing the NIST Cybersecurity Framework, we are enhancing our ability to identify, protect against, detect, respond to, and recover from potential cybersecurity incidents, ensuring the security and reliability of our services platform.

### 