Technical Proposal

Corporate Governance and Performance Monitoring Information System (CGAPMIS) Hackathon

27-01-2025

By Vigilan Link Pty Ltd

Executive Summary

The Corporate Governance and Performance Monitoring Information System (CGAPMIS) proposal outlines the development of a secure, scalable, and efficient platform designed to manage, track corporate governance and performance of SOEs. The system aims to improve transparency, accountability, and efficiency in calculating and handling SOE performance and governance ensuring compliance with legal and regulatory frameworks.

Project Overview

The CGAPMIS will serve as a centralized repository and management tool to enhance its operational efficiency, PEEPA intends to digitize its governance and performance monitoring channels, subsequently creating an effectively balanced data channels as well as a transparent stakeholder engagement process and reporting channels.

Objectives

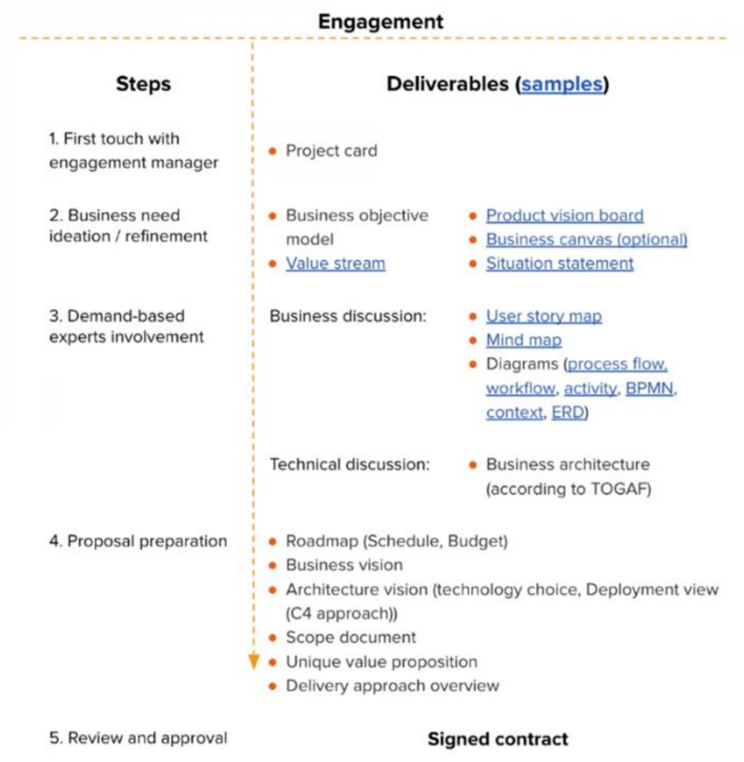
- Centralization: Establish a single, unified platform for managing SOE corporate governance and performance.
- Transparency: Improve visibility into the status and handling of SOE corporate governance and performance through detailed tracking and reporting.
- Efficiency: Streamline processes related to corporate governance and performance to reduce administrative overhead and costs.
- Compliance: Ensure adherence to legal and regulatory requirements governing the management of SOE governance and performance.
- Security: Implement robust security measures to protect sensitive data and prevent unauthorized access.

Project Timeline

- Month 1-2: Requirements Gathering and Planning
- Month 3-4: System Design and Architecture
- Month 5-8: Development and Integration
- Month 9-10: Testing and Quality Assurance
- Month 11: Deployment
- Month 12: Training, Go live and Support
- Year 2 & 3: Perfective/Adaptive/Corrective Maintenance & Support

Methodology

The Agile methodology will be adopted to provide flexibility and iterative progress throughout the project. This approach allows for continuous feedback, adaptation to changes, and ensures stakeholder involvement at every stage.



Engagement

Steps	Deliverables (samples)	
1. First touch with engagement manager	• Project card	
2. Business need ideation / refinement	• Business objective model • Value stream	• Product vision board • Business canvas (optional) • Situation statement
3. Demand-based experts involvement	Business discussion:	• User story map • Mind map • Diagrams (process flow, workflow, activity, BPMN, context, ERD)
	Technical discussion:	• Business architecture (according to TOGAF)
4. Proposal preparation	• Roadmap (Schedule, Budget) • Business vision • Architecture vision (technology choice, Deployment view (C4 approach)) • Scope document • Unique value proposition • Delivery approach overview	
5. Review and approval	**Signed contract**	

Introduction

Mandate

- The mandate of PEEPA is to advise the government of Botswana on privatisation, strategies and to implement privatisation programme, promote corporate governance as well as monitoring the performance of parastatals.

Expected Outcome

- Fully automated self-service system platform.
- A versatile and flexible workflow management solution. Enhance timely responses to stakeholder information. requests.
- Fully automated inventory management capability.
- Centralized and secure electronic document management.
- Reduced occurrence of discrepancies.
- Reduced occurrence of discrepancies.
- Data collection and accurate analysis for reporting of data.
- Enhanced planning for future vacancies.
- Reduced turnaround time for board selection process, parastatal and Boards performance monitoring and evaluation.
- Capability to Search, sort, calculate, report and share information.
- Digitized and improved agency's existing business processes.
- Capability to Perform mathematical and statistical calculations on the data to support queries submitted by users.
- Highly secure system with restricted access to data based upon usernames and passwords as well as other security features
- Prompt end user support platform within the system
- Automated alerts, notifications and feedback mechanism.

Challenges

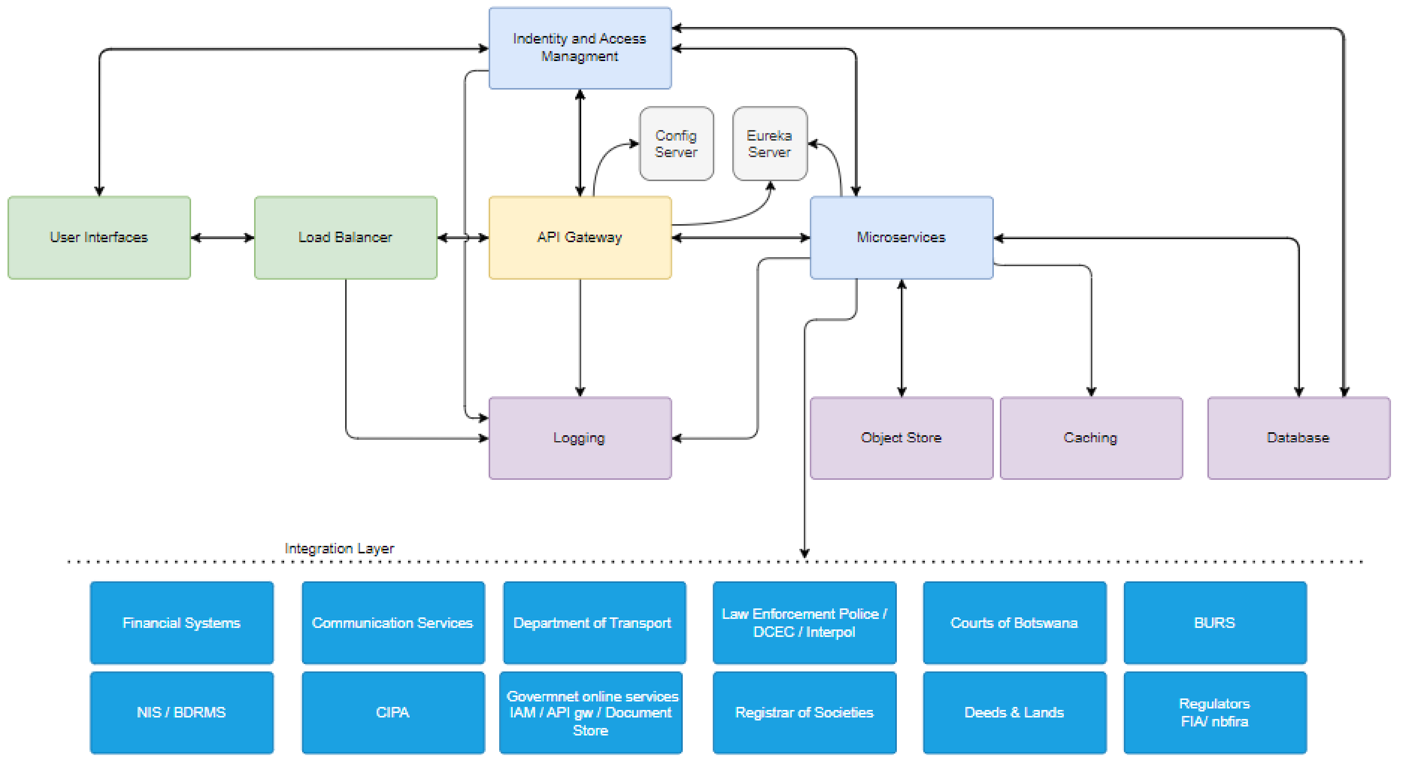
- PEEPA still relies on manual processes in carrying out these functions and this manifest itself as:
    - Manual Data handling – the reliance on spreadsheets and paper-based reports increases the risk of errors and delays in decision making and presentation of results There is limited access to information and the team members do not work in a well-coordinated environment.
    - Lack of Real-Time Insights: PEEPA struggles to obtain real-time data for performance tracking, for instance, the directors' database is not updated regularly leading to missed opportunities for timely reporting and possible recommendations and board appointment tracking. There are also delays in producing performance reports.
- All these lead to inefficiencies, inaccuracies, delays and lack of transparency in the reporting of the results.
- PEEPA wishes to leverage on innovative technology solutions to improve on its data management channels and wishes to develop a Corporate Governance and Performance Monitoring information System (CGAPMIS).

Solution Design

High-Level Architecture

- **Frontend** – Self-service portal & Management Interface.
- **Backend** – Services to fulfil the core business functions.
- **Database**, Object Store & Caching – Persist and secure data of any type.
- **Cloud hosting** - Leverage of managed services for high availability and resilience. **Security** – solution designed with security as a core feature
- **APIs** - RESTful APIs for communication between frontend and backend, external system integrations.
- **Message queuing brokering & log aggregation-** manage messaging and logs useful for auditing.

Component Diagram



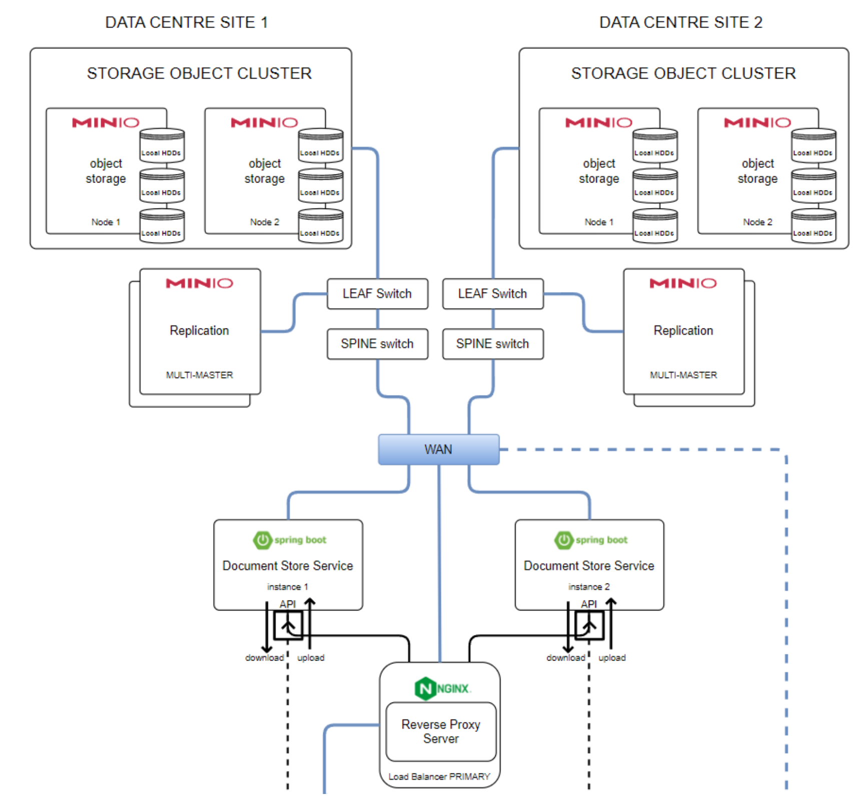
SECURITY

High Availability of data persistent storages

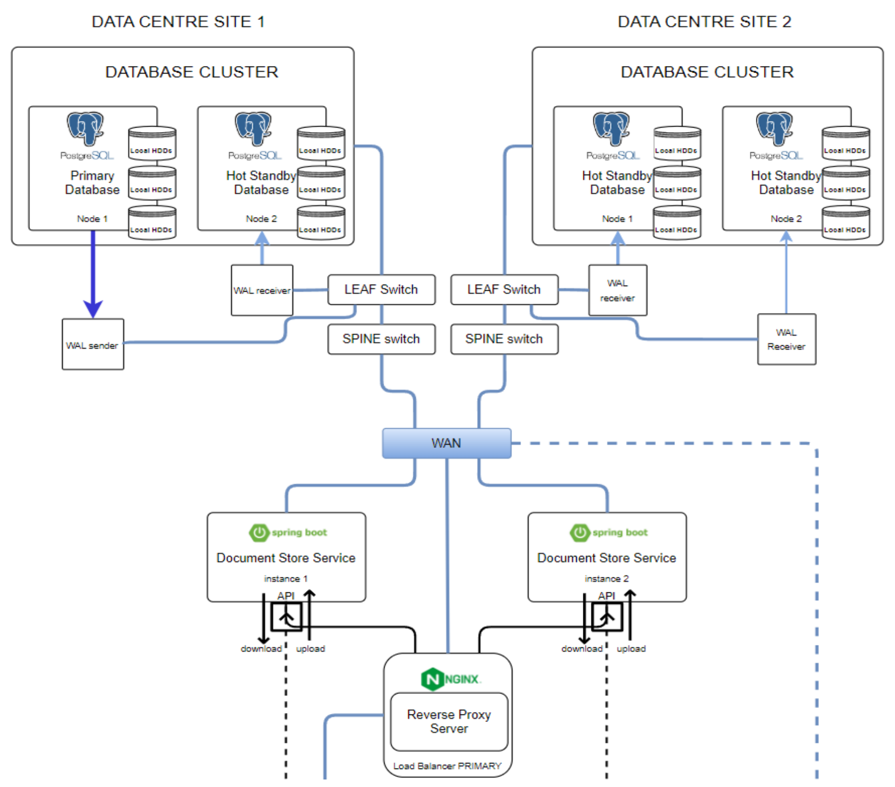
Multiple Servers: multiple servers or virtual machines (at least 4) to set up a distributed clusters.
Networking: All servers should be able to communicate with each other over the network.
Storage: Each server should have its own storage. You can use local disks or network-attached storage.

Moreover, data at rest and in transit will be encrypted

Document Stores Architecture



Database Architecture



Microservice - agility & modularity

Java (spring) and Golang will be used to build all micro applications will run on replica set, and managed by Kubernetes, in an event service dies a new one will be spun up to maintain the set number of instances running

Load balancer – security

NGINX will be used to distribute network or application traffic across multiple servers to ensure no single server becomes overwhelmed. It helps improve the reliability, scalability, and performance of applications, providing high availability by ensuring traffic can still be processed even if one or more servers fail.

Backups

All databases will be backed up on scheduled frequencies and database verification performed on monthly basis.

Databases & Document store

PostgreSQL and MySQL will be used in a highly secured set up, and the following techniques will be used

- Data Redaction
- Data encryption
- No direct access to database / use APIs

MinIO will be used to store documents

Solution Design

Frontend
The frontend will be a responsive web application
- Dashboard: Overview of asset status and key metrics.
- Asset Management: Forms and views for entering, updating, and viewing asset details.
- Reports: Generate and view detailed reports on asset status, movements, and dispositions.
- User Management: Admin interface for managing user roles and permissions.

API Gateways & Load balancers
All request will be routed through load balancers, and via API gateways. API gateways provide a single secure/trusted entry to the onegov services

Identity and Access Management
CGAPMIS solution will leverage on this for all authentication (Users, Roles, groups and resource access control). The integration will be via existing APIs.

Backend
The backend micro services will be developed providing RESTful APIs for frontend communication. Key services include:
- Performance Monitoring: Manage asset information, status updates, and lifecycle events.
- User Service: Handle authentication, user profiles, and role management.
- Reporting Service: Generate and deliver reports based on asset data.
- Notification Service: Send alerts and notifications related to asset events.

Caching
For caching data in memory for fast store and retrievals, OTPS, encryption keys, tokens

Logging, Aggregation and visualization
For processing the varies logs generated in the entire CGAPMIS, this logs are useful for audits and troubleshooting the system.

Message queueing/brokering
acts as a mediator between producers of messages and consumers who receive and process those messages. It facilitates the communication between different components of an application by enabling asynchronous messaging and decoupling the sender and receiver.

Communications services [SMS & EMAILS]
For CGAPMIS to send any communication.

Object Storage
A Central Document storage. The store is capable of handling documents of all types, eg videos, pictures, text documents of any format

Database
PostgreSQL will be used for relational data management. Key tables include:
- Assets: Stores asset details such as type, value, location, and status.
- Users: Stores user information, roles, and permissions.
- Transactions: Records asset movements and status changes.
- Reports: Stores generated reports and analytics data.

Integration
- The system will integrate with external systems through RESTful APIs:
- Law Enforcement Databases: To import and update asset information.
- Financial Systems: For handling financial assets and transactions.
- Auction Platforms: To facilitate the disposition of assets.
- National Identity, CIPA, BURS, Birth & Death Registration systems

FEASIBILITY

Solution stacks

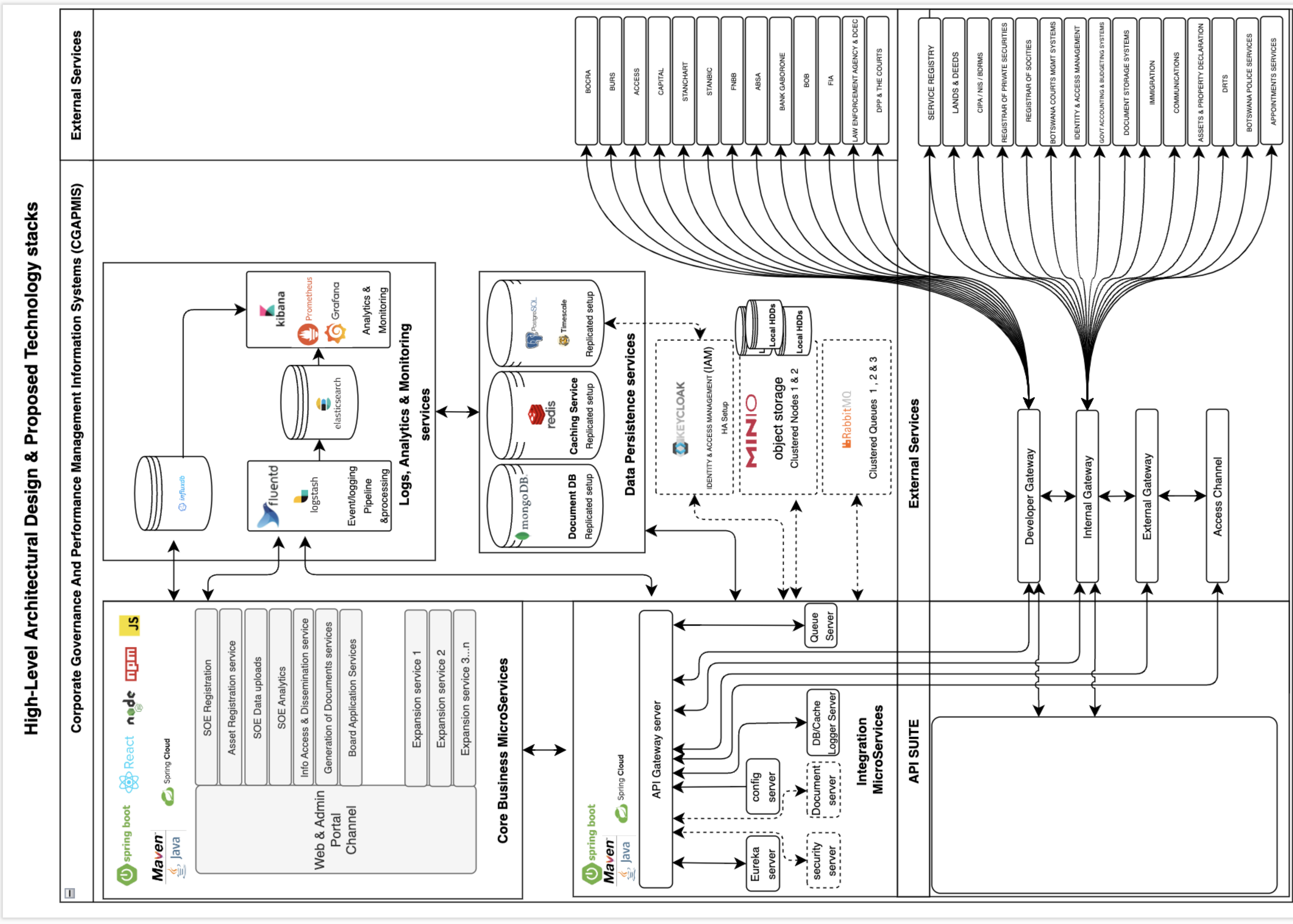
The CGAPMIS will be developed using a modular architecture, leveraging on cloud technologies to ensure scalability and reliability. The system will utilize a microservices-based approach, enabling independent deployment and scaling of components.

Data protection and data privacy remains a critical issue and will be build right from the start

Moreover, open-source technologies enables innovation and low cost of operation and ownership.

- Cloud Infrastructure: Local Datacenters(e.g. BTC) for hosting and infrastructure management.
  - Local hosting in compliance with data protection act and data residency
- Deployment : Linux Nodes running Docker Containers orchestrated with Kubernetes
  - Containerized for agility for scalability running on Linux OS
- Database Management: PostgreSQL for relational database management.
  - Powerful and open source
- Caching Management: Redis
  - robust open-source in-memory data structure store.
- Message Queuing and Brokering: RabbitMQ
  - lightweight and easy-to-deploy open-source message broker that supports multiple messaging protocols, such as AMQP, STOMP, and MQTT.
- Document management: minIO
  - a secure, highly available high-performance, S3 compatible object store
- Frontend Development: React.js
  - for building a responsive and user-friendly interface.
- Backend Development: Golang, Java and Spring Framework, Node.js and Express.js
  - for developing scalable server-side applications.
- Security: Red Hat keycloak , Load balancer Nginx
  - for implementation of industry-standard security protocols, including encryption, single sign on, multi-factor authentication, and role-based access control.
- API-first approach:
  - Industry open APIs for seamless and agile integrations with internal, third-party and legacy systems
- Elasticsearch, Logstash, and Kibana.
  - gives the ability to aggregate logs from all microservices, analyze these logs, and create visualizations for application and infrastructure monitoring, faster troubleshooting, security analytics, and more.

High Level Architecture



Deployment Strategy & Maintenance and Support

Staging Environment:
Deploy to a staging environment for thorough testing.
Production Deployment:
Use CI/CD pipelines to ensure smooth, error-free deployments.
Rollback Procedures:
Implement rollback mechanisms in case of deployment failures.

Monitoring:
Use Prometheus and Government online monitoring capabilities for system performance and health monitoring.
Updates:
Regular updates and patches to address bugs and security vulnerabilities.
Support:
Provide ongoing technical support and user training.

Technologies

Jenkins: automates pipelines to enable continuous integration and continuous delivery (CI/CD), automating the various stages of software development such as test, build, and deployment.

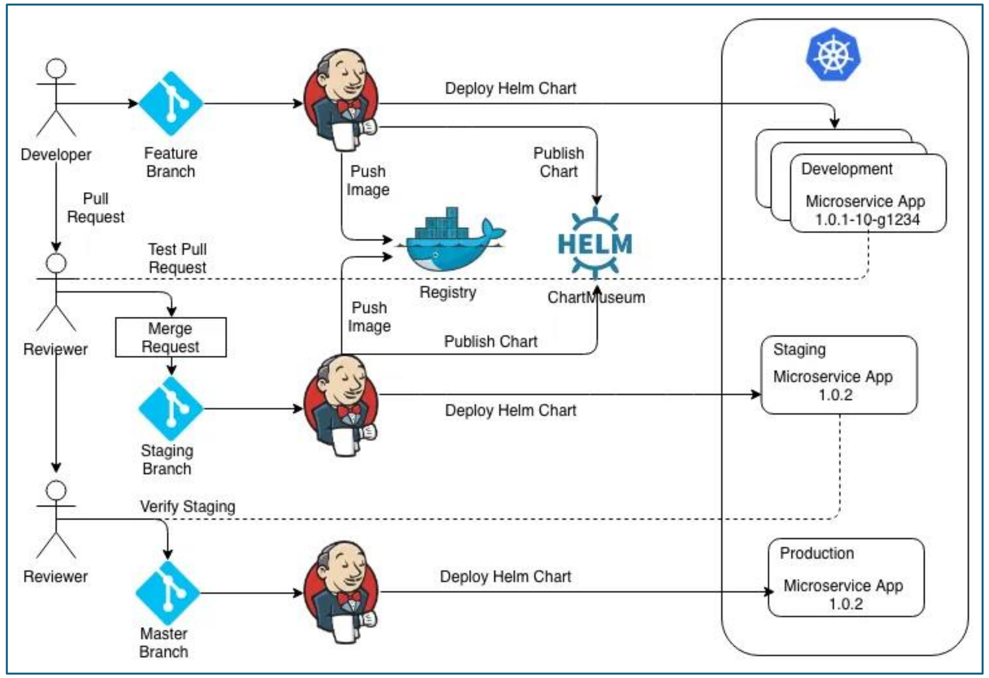
Git & Docker Registry: **Git** is a distributed version control system that tracks versions of files. Used to control source code by programmers collaboratively developing software. Docker Registry stores container images.

Helm: Helm helps manage Kubernetes applications — Helm Charts help you define, install, and upgrade even the most complex Kubernetes application.

Prometheus: systems monitoring and alerting toolkit

Kubernetes & Docker: Docker is an engine for building containerized applications while K8s automating deployment, scaling, and management of containerized applications.

Argo CD: automates the deployment of the desired application states in the specified target environments. Application deployments can track updates to branches, tags, or pinned to a specific version of manifests at a Git commit.



FUNCTIONALITY

Workflow & Processes

Corporate Governance

<u>SOE Creation</u>

Functions – Manage Financials, Documents, Assets, board and vacancies

Initiation: User selects "Create New SOE" from the dashboard.

Data Entry: User fills in SOE details in the registration form.

Validation: System validates the entered data.

Submission: User submits the form to create a new SOE.

Confirmation: System displays a confirmation message, and the new SOE is added to the database.

<u>SOE Update</u>

Functions – Manage Financials, Documents, Assets, board and vacancie**s**

Selection: User selects a SOE from the SOE list.

Editing: User updates the necessary fields in the SOE detail's view.

Validation: System validates the updated data.

Submission: User submits the form to save changes.

Confirmation: System displays a confirmation message and updates the SOE in the database.

<u>SOE Search and Filter</u>

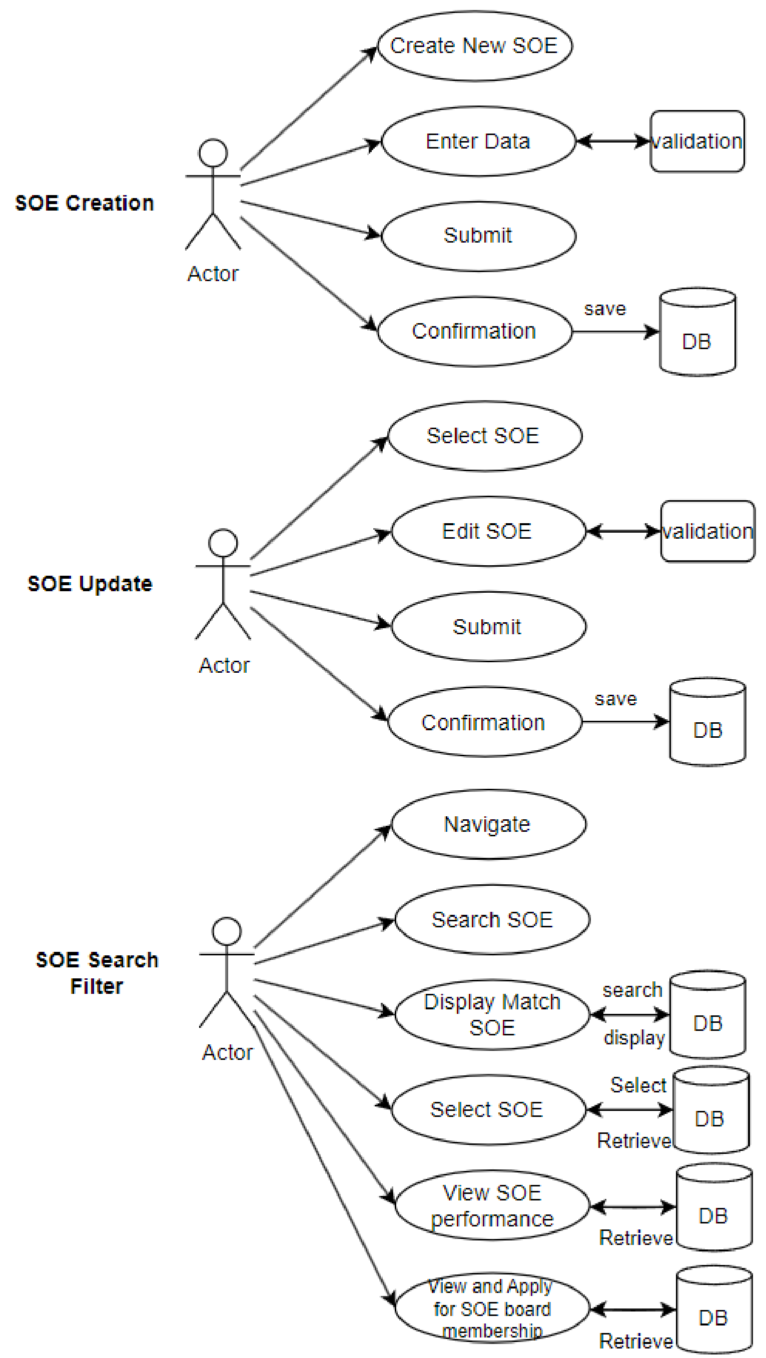
Functions – View and download reports on Financials, Documents, Assets, Board and vacancies, view SOE performance

Access: User navigates to the SOE search page.

Search: User enters search criteria and initiates the search.

Results: System displays matching SOE in a list.

Selection: User selects a SOE from the search results to view details.



Lesson Learnt

1. Importance of Clear Requirements

- Lesson: Clearly defined and documented requirements are crucial for project success.
- Action: Invest more time in initial requirements gathering and validation to avoid scope creep and ensure all stakeholders have a shared understanding of project goals.

2. Agile Methodology Benefits

- Lesson: The Agile methodology facilitated flexibility and adaptability, allowing the team to respond to changes and feedback quickly.
- Action: Continue using Agile practices such as sprints, daily stand-ups, and retrospectives to maintain a high level of collaboration and continuous improvement.

3. Risk Management

- Lesson: Early identification and proactive management of risks prevented many potential issues.
- Action: Implement a more formal risk management framework with regular risk assessment meetings and predefined mitigation strategies.

4. Continuous Feedback

- Lesson: Regular feedback from stakeholders helped align the project with user expectations and business needs.
- Action: Schedule frequent demos and feedback sessions with stakeholders to ensure their requirements are continuously met and any issues are promptly addressed.

5. Security Best Practices

- Lesson: Implementing security best practices from the beginning helped in mitigating potential vulnerabilities.
- Action: Continue to prioritize security in the development process, including regular security audits and adherence to best practices for data protection.

Thank you