# Technical Architecture

Goa - GEL Blockchain based License Approval Platform

**Confidentiality Notice**

This document contains proprietary and confidential information intended solely for the use of authorized personnel within McLaren Strategic Solutions and its designated stakeholders. The contents of this document, including all data, designs, diagrams, and intellectual property, are protected by applicable laws and regulations.
Unauthorized disclosure, reproduction, distribution, or use of this document or any part of it is strictly prohibited. If you have received this document in error, please
notify MSS immediately and delete all copies of it.

By accessing or reviewing this document, you agree to comply with the confidentiality requirements outlined herein.

For any questions or permissions regarding this document, please contact:

**Mahendrababu D V**
**Lead Blockchain Engineer**
Mahendrababu.dv@mclarensv.com
**+1 782 882 7272**

### Revision History

| Revision | Description | Author | Revision On | Revision Notes |
|---|---|---|---|---|
| **1.0** | Initial Draft | Mahendrababu D V | 03/02/26 | |
| | | | | |
| | | | | |
| | | | | |

## Summary

The Goa GEL Blockchain based License approval Platform represents a pioneering initiative to leverage blockchain technology for secure, transparent, and tamper-proof document verification and workflow management.

This document provides comprehensive technical architecture, detailing blockchain infrastructure, application layer, workflow engine, and deployment strategy using Hyperledger Besu with QBFT consensus.

### Key Features

- End-to-end document verification using blockchain
- Multi-department approval workflows with state management
- ERC-721 NFT-based license verification
- Custodial wallet management with HSM progression path
- Integrated document storage via MinIO and Redis caching
- RESTful API for seamless system integration

# 1. Introduction

## 1.1 Background & Problem Statement

Government document verification processes in India face significant challenges:

- Document forgery and tampering risks
- Opaque, siloed approval workflows across departments
- Lack of audit trails and accountability
- Manual processes prone to delays and errors
- Limited interoperability between government systems

## 1.2 Solution Overview

The Goa GEL platform addresses these challenges through blockchain-based document verification:

- Immutable ledger for document records and approvals
- NFTs for non-transferable license credentials
- Smart contracts for automated approval workflows
- Cryptographic proof of data integrity
- Complete audit trails for regulatory compliance

## 1.3 Alignment with NBFLite

Platform adheres to India's National Blockchain Framework guidelines:

- Permissioned blockchain (Hyperledger Besu) for controlled access
- QBFT consensus for Byzantine fault tolerance
- Public-key cryptography for secure transactions
- Interoperability with existing e-governance systems

# 2. System Architecture

## 2.1 High-Level Overview

Three-tier architecture with clear separation of concerns:

| Layer | Description |
|---|---|
| Blockchain Layer | Hyperledger Besu with smart contracts for immutable record-keeping |
| Application Layer | NestJS backend, Next.js frontend, PostgreSQL, MinIO, Redis |
| Integration Layer | RESTful APIs, webhooks, DigiLocker integration |

## 2.3 Container Architecture

Platform containerized using Docker:

| Service | Technology | Port | Purpose |
|---|---|---|---|
| Backend API | NestJS/TypeScript | 3001 | Core application logic |
| Frontend | Next.js 14 + shadcn/ui | 3000 | Web UI |
| Database | PostgreSQL | 5432 | Relational data |
| Document Storage | MinIO | 9000 | S3-compatible storage |
| Cache | Redis | 6379 | Key-value caching |
| Blockchain | Hyperledger Besu | 8545/8546 | QBFT validators |

# 3. Blockchain Layer

## 3.1 Hyperledger Besu with QBFT Consensus

Hyperledger Besu chosen for:

- Enterprise-grade stability and security
- EVM compatibility for Ethereum smart contracts
- Multiple consensus mechanisms (QBFT for private networks)
- Privacy features including private transactions

**QBFT (Quorum Byzantine Fault Tolerance)** ensures Byzantine fault tolerance with guaranteed finality. Ideal for permissioned networks where validators are known and trusted.

## 3.2 4-Validator Node Network

Network consists of 4 validator nodes,

| Node |
| --- |
| Boot |
| Public |
| Validator0 |
| Validator1 |

**Byzantine Fault Tolerance:** With 4 validators, network tolerates 1 faulty node (f=1, 3f+1=4). All transactions require consensus from minimum 3 validators.

## 3.3 Smart Contract Architecture

Four core smart contracts manage document verification:

### 3.3.1 LicenseRequestNFT (ERC-721 )

**Purpose:** Issues non-transferable NFT credentials upon approval

- Non-transferable () tokens
- Embeds metadata: applicant address, approval date, license type

- Revocation capability for expired licenses

### 3.3.2 ApprovalManager

**Purpose:** Orchestrates multi-department approvals

- Records department approval decisions
- Manages approval sequence and parallel stages
- Stores approval metadata: timestamp, approver ID, remarks

### 3.3.3 DepartmentRegistry

**Purpose:** Maintains list of authorized departments

- Role-based access control (RBAC)
- Department metadata: address, name, approval sequence

### 3.3.4 WorkflowRegistry

**Purpose:** Defines and manages approval workflows

- Defines sequential and parallel approval stages
- Workflow templates for different license types

## 3.4 On-Chain vs Off-Chain Data

Optimized storage strategy:

| Data Type | On-Chain | Off-Chain |
|---|---|---|
| Approvals | Hash, timestamp, approver ID | Full records, remarks |
| Documents | Content hash (SHA-256) | PDFs, images |
| NFT Metadata | License ID, holder, expiry | Additional details, JSON |
| Request Status | State transitions | Full workflow history |

# 4. Application Layer

## 4.1 Backend (NestJS)

Progressive TypeScript framework for scalable server-side applications.

**Core Modules:**

- AuthModule: API key + secret authentication
- RequestModule: License request CRUD operations
- ApprovalModule: Department approval workflows
- BlockchainModule: Besu interaction
- DocumentModule: MinIO storage

## 4.3 PostgreSQL Schema

Key database tables:

| Table | Purpose |
|---|---|
| users | User accounts (applicants, staff, admins) |
| requests | License requests with status, timestamps |
| approvals | Department approvals with decision, remarks |
| documents | Document metadata, paths, hashes |
| departments | Department registry with roles |
| nfts | Issued NFT metadata |
| audit_logs | Complete audit trail |

## 4.4 Document Storage (MinIO)

S3-compatible object storage for large files.

- Bucket organization: requests/{requestId}/documents/
- Immutable object mode
- Versioning for audit trails

- Access control via signed URLs

### 4.5 Cache Layer (Redis)

High-speed key-value caching for:

- Session management
- Approval state
- Request workflow cache
- API rate limiting

# 5. Workflow Engine

## 5.1 Multi-Department Approval

Resort license approval requires sequential approvals:

1. Tourism Department: Resort legitimacy and classification
2. Health Department: Sanitation and medical facilities
3. Environment Department: Ecological compliance
4. Forest Department: Forest area protection

## 5.2 Sequential vs Parallel Stages

**Sequential:** Each department completes approval before next begins. Tourism -> Health -> Environment -> Forest.
**Parallel:** Independent reviews conducted simultaneously for non-dependent workflows.

## 5.3 State Machine

Request states follow defined state machine:

| State | Transition | Trigger | Actions |
|---|---|---|---|
| DRAFT | SUBMITTED | Applicant submits | Log entry, notify |
| SUBMITTED | UNDER_REVIEW | First dept starts | Update dashboard |
| UNDER_REVIEW | APPROVED/REJECTED | Department decision | Record approval |
| APPROVED | NFT_ISSUED | All depts approve | Mint NFT |

## 5.4 Document Versioning

**Document Versioning:** Each update creates new version with hashing. Previous versions retained.
**Approval Invalidation:** Document updates after approval invalidate dependent approvals.

# 6. API Design

## 6.1 RESTful Structure

All APIs follow REST conventions with JSON payloads. Base: http://localhost:3001/api/v1

## 6.2 Key Endpoints

### Requests

- POST /requests: Create license request
- GET /requests: List requests
- GET /requests/:id: Request details
- POST /requests/:id/submit: Submit for approval

### Approvals

- GET /approvals: Pending approvals
- POST /approvals/:id/approve: Approve
- POST /approvals/:id/reject: Reject

### Blockchain

- GET /blockchain/nft/:tokenId: NFT metadata
- POST /blockchain/nft/:requestId/mint: Mint NFT

## 6.3 Webhooks

Real-time department notifications:

- request.submitted
- approval.approved
- approval.rejected
- nft.issued

# 7. Security Architecture

## 7.1 Authentication

POC authentication uses API Key + Secret:

- X-API-Key and X-API-Secret headers
- JWT token issued on success
- Subsequent requests use JWT bearer

**Future:** OAuth 2.0 with OIDC for AADHAR integration.

## 7.2 Custodial Wallet Management

**POC:** Keys in environment variables
**Production:** Keys in Hardware Security Module (HSM)

## 7.3 Mock DigiLocker

POC purposes - simulates document verification:

- Validates document signatures
- Returns: VERIFIED, FAILED, NOT_FOUND

## 7.4 Key Storage Progression

| Phase | Storage | Level | Use |
|---|---|---|---|
| POC | Env Vars | Low | Development |
| Pilot | HashiCorp Vault | Medium | Controlled |

# 8. Deployment Architecture

## 8.1 Docker Compose Setup

POC deployed with all services containerized:

- Backend (NestJS) port 3001
- Frontend (Next.js/React.js) port 3000
- PostgreSQL port 5432
- MinIO port 9000
- Redis port 6379
- Besu nodes ports 8545-8548
- Prometheus port 9090
- Grafana port 3100

## 8.2 Service Topology

| Service | Port | Protocol | Dependencies | Check |
|---|---|---|---|---|
| Backend | 3001 | HTTP/JSON | All | /health |
| Frontend | 3000 | HTTP | Backend | GET / |
| PostgreSQL | 5432 | TCP | None | psql |
| MinIO | 9000 | HTTP | None | health |
| Redis | 6379 | TCP | None | PING |

## 8.3 Monitoring & Logging

**Prometheus Metrics:** Request latency, errors, database connections, block time, consensus
**Grafana Dashboards:** System overview, application health, blockchain status
**Logging:** Structured JSON with timestamps, audit logs for approvals and NFT minting

# 9. POC Scope

## 9.1 In-Scope Features

- Resort license application submission
- Multi-department approval workflow
- Document upload and verification
- Blockchain approval recording
- ERC-721  NFT minting
- RESTful API
- Dashboard for request tracking
- Audit logs and blockchain tracking

## 9.2 Out-of-Scope (Future)

- OAuth 2.0 / OIDC with AADHAR
- Actual DigiLocker integration
- Hardware Security Module
- Private smart contracts
- Cross-chain interoperability
- Analytics dashboards
- Mobile app

## 9.3 Demo Scenario

5. Applicant logs in at http://localhost:3000
6. Fills resort license form, uploads documents
7. Backend records in PostgreSQL, stores in MinIO
8. Submits; status becomes SUBMITTED
9. Tourism Dept reviews and approves
10. Health, Environment, Forest Depts complete reviews
11. All approvals trigger NFT mint on Besu
12. Status becomes NFT_ISSUED
13. Audit logs show complete chain

# 10. Appendices

## 10.1 Glossary

| Term | Definition |
|---|---|
| QBFT | Quorum Byzantine Fault Tolerance - consensus for permissioned networks |
| NFT | Non-transferable token bound to wallet, cannot be traded |
| Smart Contract | Self-executing code on blockchain enforcing rules |
| Validator Node | Node with authority to validate blocks in QBFT |
| MinIO | S3-compatible object storage for distributed systems |
| DigiLocker | India's digital document storage and verification service |
| NBFLite | National Blockchain Framework (Lite) for government adoption |

## Document Information

**Title:** Goa - GEL Blockchain based License Approval Platform - Technical Architecture
**Version:** 1.1 POC
**Date:** February 4, 2026
**Classification:** Technical

Comprehensive technical architecture covering blockchain infrastructure, application design, security measures, and deployment strategy for the Goa GEL platform.