PEAK SOLUTIONS

Iloilo City Council Digital Access for Legislative Information (DALI) Portal

ARCHITECTURE DOCUMENT

VERSION 1.0

NOVEMBER 26, 2025

Revision History

Date (DD/MM/YYYY)	Version	Description
10/11/2025	0.1	Initial draft (Chapters 1-3)
10/12/2025	0.2	Finalization of initial draft (Chapters 1-3)
11/12/2025	0.3	Updated Chapters 1-3 and added Chapter 4 (Solution Strategy)
11/15/2025	0.4	Add Chapter 5
11/16/2025	0.5	Add Chapter 6 & 7
11/17/2025	0.6	Update diagrams
11/19/2025	0.7	Update chapter 3 and chapter 7
11/20/2025	0.8	Update chapter 5 diagrams & add chapter 8
11/21/2025	0.8.1	Update layouts and chapter 8
11/22/2025	0.9	Update chapters 2, 4, 8 and add chapters 9 and 12
11/26/2025	1.0	Finalize document

PEAK SOLUTIONS

Table of Contents

Revision History 1
Table of Contents 2
1. Introduction and Goals 4
1.1 Requirements Overview 4
1.2 Quality Goals 6
1.3 Stakeholders 8
2. Constraints 9
2.1 Technical Constraints 9
2.2 Organizational Constraints 9
2.3 Political Constraints 10
2.4 Conventions 10
3. Context and Scope 12
3.1 Business Context 12
3.2 Technical Context 14
4. Solution Strategy 16
5. Building Blocks 18
5.1 Building Block View: Level 1 18
5.1.1 Whitebox: DALI Portal 18
5.1.2 Contained Blackboxes 19
5.2 Building Block View: Level 2 20
5.2.1 Blackbox of Public Portal 20
5.2.3 Blackbox of Internal Management System 21
5.3 Building Block View: Level 3 23
5.3.1 Blackbox of Document Service 23
5.3.2 Blackbox of Session Service 24
5.3.3 Blackbox of Booking Service 25
5.3.4 Blackbox of Visitor Service 26
5.3.5 Blackbox of Legislative Document Archive 27
5.3.6 Blackbox of Inquiries Ticket System 28
5.3.7 Blackbox of Session Information Display 29
6. Runtime View 30
6.1 Runtime Scenario 1: Citizen Submits an Inquiry (Public Portal) 31
6.2 Runtime Scenario 2: Staff Processes an Inquiry Ticket (Internal Workflow) 32
6.3 Runtime Scenario 3: Document Tracking 34
7. Deployment View 36

7.1 Overview of the Deployment View 36
7.1.1 Elements of the Deployment View 37
8. Crosscutting Concepts 38
8.1 Domain Concepts 39
8.1.1 Domain Models 39
8.1.2 Domain Processes 42
8.2 Security and Role-Based Access Control (RBAC) 45
8.2.1 Authentication 45
8.2.2 Role-Based Access Control (RBAC) 46
8.2.3 Abuse Prevention (Anti-Bot) 48
8.3 Deployment and Infrastructure 48
8.3.1 Cloud Based Architecture 48
8.3.2 Containerization and Orchestration 49
8.3.3 Reverse Proxy and Routing 50
8.3.4 Network Security Layer 50
8.3.5 Email Service Integration 51
8.4 Frontend Design Patterns 52
8.4.1 Client Side Rendering 52
8.4.2 Server Side Rendering 52
8.5 User Interface / User Experience 53
8.5.1 Theme 53
8.5.2 Design System 54
9. Architectural Decisions 58
9.1 Next.js for Public Portal and Internal System 58
9.2 NestJS for Backend API 59
9.3 Supabase for Authentication, Database, and Storage Provider 60
9.4 Resend for Transactional Email 60
9.5 Virtual Private Server(VPS) over Serverless hosting 61
9.6 Using Digital Ocean Droplet for VPS 62
9.7 Coolify as VPS Management Platform 62
9.8 Server Side Rendering for Public Portal 63
9.9 Client Side Rendering for Internal Management System 64
9.10 Cloudflare as Security Edge Layer 64
9.11 Zustand as State Management for Internal Management System 65
9.12 Client-Side Uploading of Files to Supabase Storage 66
9.13 Shadcn for UI components 66
12. Glossary 68

1. Introduction and Goals

The Iloilo City Council Digital Access for Legislative Information (DALI) Portal is a unified web platform designed to digitize and streamline the Iloilo City Council Offices' core operations: document management, session management, citizen assistance, beneficiary tracking, and conference booking, while making legislative documents and information easily accessible to the public.

1.1 Requirements Overview

ID	Functional Requirement	Description
FR-01	Public Transparency Portal	A citizen-facing website providing open access to legislative records and council information.
FR-02	Legislative Document Archive	A searchable repository for Ordinances, Resolutions, and Committee Reports. Citizens can filter by type, committee, or date and view documents via an integrated PDF Previewer without downloading.
FR-03	Session Information Hub	A digital board displaying upcoming and past sessions. It allows citizens to view a session's agenda items and download the "Minutes of the meeting" and "Journals" of past sessions.
FR-04	Inquiry Hub	A formal correspondence interface for citizens to submit requests (Financial Assistance, Scheduling, Follow up on Document, Other). It uses a secure "Ticket Reference Number" + Email validation system instead of a public login.

FR-05	Council Directory	A visual organizational chart displaying the Vice Mayor, City Councilors, and Legislative Staff with their names and committee chairmanships.
FR-06	Internal Tasks Dashboard	The landing page for staff, featuring personalized "To-Do" queues based on their role (e.g., "Documents for Initial" for Head Admin, "My Active Tickets" for OVM Staff).
FR-07	Legislative Document Tracker	A digital tracker for all Legislative, Administrative, and Invitation documents. It shall replace the manual logbook by enforcing standardized status workflows (Received → Initialed → Signed) and maintaining an immutable audit trail for all document actions.
FR-08	Session Management Module	A specialized workspace for the Legislative Staff to build the Session Agenda using a drag-and-drop interface. It includes a "Presentation Mode" for projecting the agenda live during council sessions.
FR-09	Inquiry Ticketing System	The workspace for OVM (Office of the Vice Mayor) Staff to manage citizen requests. It features a threaded conversation view, automated status updates (Open, Waiting for Citizen, Rejected, or Resolved), and mandatory closing remarks for transparency.
FR-10	Visitor & Beneficiary Hub	An internal CRM for tracking walk-in visitors and financial aid

		applicants. It enforces a "Search First" workflow to link individuals to a "Household Profile," preventing duplicate aid distribution.
FR-11	Conference Room Booking	A centralized calendar for managing facility usage. It employs a "Direct Booking" model for Staff and a "Request-Approval" model for Councilors, eliminating conflicting schedules.
FR-12	User Administration	A restricted module for the IT Admin to provision staff accounts via email invitations and manage Role-Based Access Control (RBAC) assignments.
FR-13	Secure Authentication	The system delegates identity management to a third-party provider (Supabase Auth), ensuring secure login sessions without storing sensitive credentials on the application server.

1.2 Quality Goals

No.	Quality	Motivation	Scenario
1	Security	Sensitive records must be protected through user authentication and role-based access.	An unauthorized user attempts to access the "Beneficiary List" via a direct URL. The system detects the lack of the `OVM_STAFF` role and immediately blocks the request, redirecting them to the login page.

2	Usability	The interface must be intuitive enough that citizens can find ordinances or file inquiries without requiring a manual or training.	A senior citizen visits the website on their mobile phone to look for a specific "Curfew Ordinance." Using the search bar, they type "curfew" and are able to view and download the PDF within three clicks.
3	Operability	The system must streamline their workflow by automating repetitive tasks and providing clear visibility on what needs attention.	An ADMIN_HEAD logs in at the start of the day. The dashboard automatically filters and displays only the documents waiting for their initial, allowing them to process the backlog immediately without searching.
4	Reliability	The system must remain operational during office hours and be able to recover quickly from any failures without losing critical data.	Throughout the entire fiscal year, the system maintains a 99.9% uptime record. Even during peak usage periods or minor infrastructure updates, the Public Portal remains accessible to citizens, ensuring that legislative records are always available when needed.
5	Maintainability	As government requirements change, the system should be easily modified, updated, and fixed when needed.	The Vice Mayor requests a new "Priority" tag for inquiries. The developers are able to add this feature to the InquiryTicket component and deploy the update without disrupting the Document Tracking module.

1.3 Stakeholders

Roles/Names	Expectations
Hon. Lady Julie Grace “Love Love” Baronda	Serves as the client and main project sponsor. She oversees the implementation of the system and ensures it aligns with the goals of transparency, efficiency, and public service in the Vice Mayor’s Office.
Vice Mayor’s Office	The primary office that handles public requests and inquiries, facilitates communication with citizens, and oversees beneficiary-related concerns and assistance programs.
Vice Mayor’s Admin Office	The supporting office responsible for managing incoming and outgoing documents, recording invitations, and maintaining official records. It ensures that all documents are properly processed, reviewed, and endorsed to the Vice Mayor for appropriate action.
City Councilors and their Staff	Use the system to access official communications, meeting details, and legislative documents for coordination and reference.
Iloilo City Residents	Serve as end users who interact with the system through the public portal for inquiries, requests, and viewing of public documents.
Sir Anfernee Joan Ng	Serves as the adviser for the Software Architecture class. He expects a comprehensive Software Architecture document as part of his requirements.
Ma’am Ariessa Lane Ko	Serves as the project adviser. She oversees the project and expects a clean development process and a complete list

		of her requirements.
	PEAK Solutions Team	Composed of five members responsible for the design, development, testing, and deployment of the software system.

2. Constraints

2.1 Technical Constraints

	Constraints	Background and/or Motivation
TC-1	Deployed to a Cloud Server	The current legacy systems use a single in-house server "black box." This risks data loss. To mitigate this risk and ensure availability, the project is constrained to a modern and secure cloud-based deployment model.
TC-2	Browser-Based Access	The users will not install custom software. The system must be fully functional on standard web browsers (Chrome/Edge) on existing office desktops (Windows) and personal mobile devices (Android/iOS).
TC-3	Standardized Data Formats	Certain data used in the system is expected to work with other government offices' format, this constrains the format of the data.

2.2 Organizational Constraints

	Constraints	Background and/or Motivation
OC-1	Team Structure	The team is composed of five members who are responsible for undertaking the development and completion of the software

		system.
OC-2	Deadline	The project was started in September 2025 and is expected to finish by May 2026.
OC-3	Client Requirement	The scope of the software system is based on the requirements set by the client.

2.3 Political Constraints

	Constraints	Background and/or Motivation
PC-1	Data Privacy Act of 2012	The system handles sensitive personal and constituent data within a political office, making compliance with the Data Privacy Act of 2012 essential.
PC-2	Transparency and Accountability Requirements	As a public office, certain data (e.g., legislative documents) must be transparent and accessible to citizens.
PC-3	Administrative Hierarchy and Approval Flow	The system must grant a configurable role-based access control that follows the office's current chain of command.
PC-4	Government Procurement Reform Act (R.A. 9184)	The solution must prioritize cost-effectiveness and long-term sustainability to maximize the use of taxpayers' money.

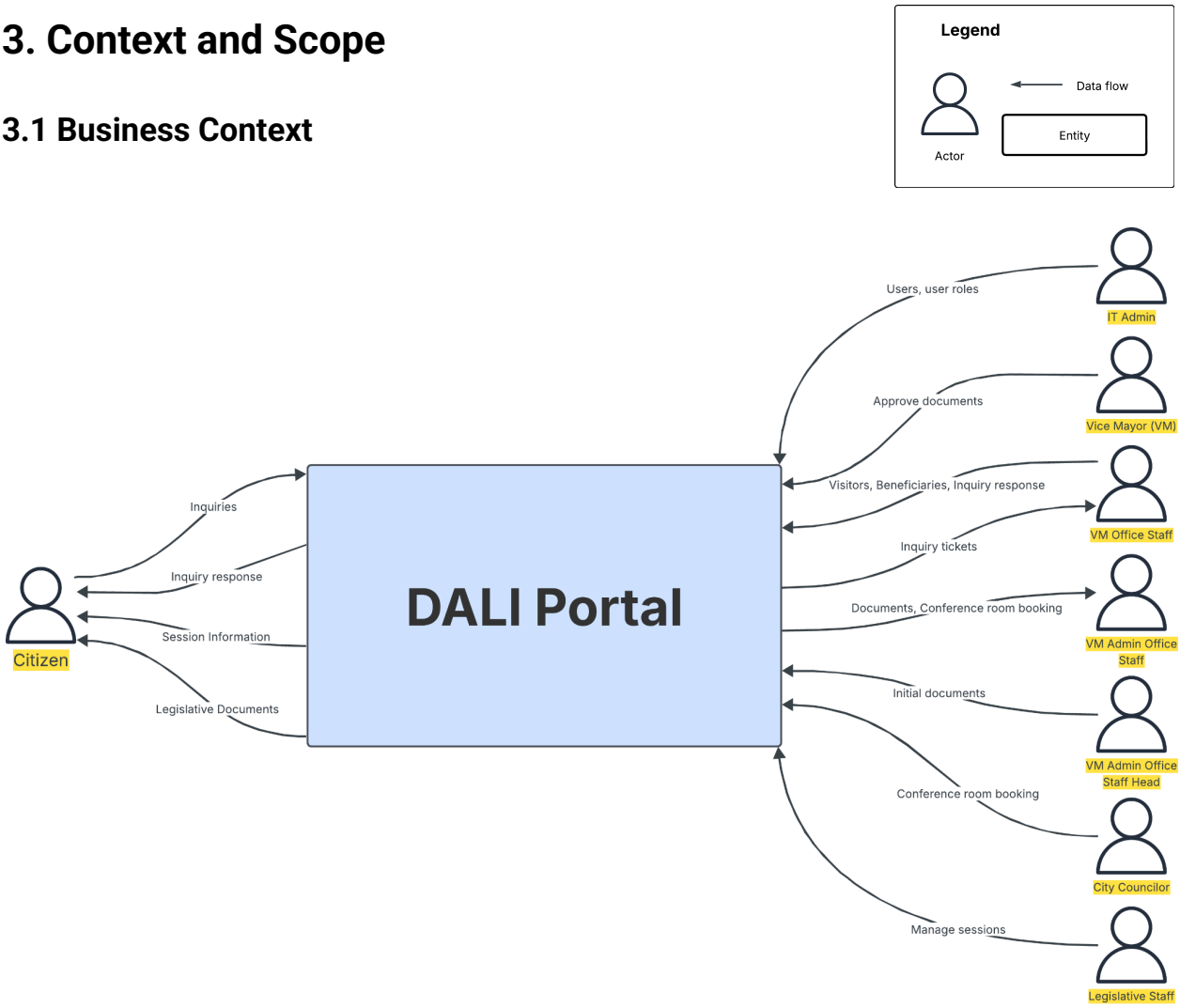
2.4 Conventions

	Convention	Background and/or Motivation
C-1	arc42 Documentation Format	The architecture documentation will follow the terminology and structure of arc42.
C-2	Naming Conventions	Use PascalCase for React Components

		Use camelCase for naming functions and variables.
C-3	Folder structure	The project will use the NextJS App router folder structure and will be organized in clearly defined sections within the src folder. Within the src folder, we'll include the following subfolders:
		App - This folder contains the routes and pages for the NextJS application.
		Components - This folder contains the reusable UI components used throughout the project.
		Lib - This folder contains the environment variables of the app.
		Utils - Used to contain the utilities used throughout the app.
C-4	Commit Message	docs - changes to existing documentation or creating new documentation.
		build - changes related to the build process, dependencies, or adding new dependencies.
		test - changes related to tests (e.g., adding new tests or modifying existing ones).
		ci - changes to the configuration of CI/CD workflows (e.g., GitHub Actions workflow).
		chore - changes that do not fit into any of the above categories, utility changes
C-5	Pull Request	For consistency, pull requests will follow an agreed upon template.

3. Context and Scope

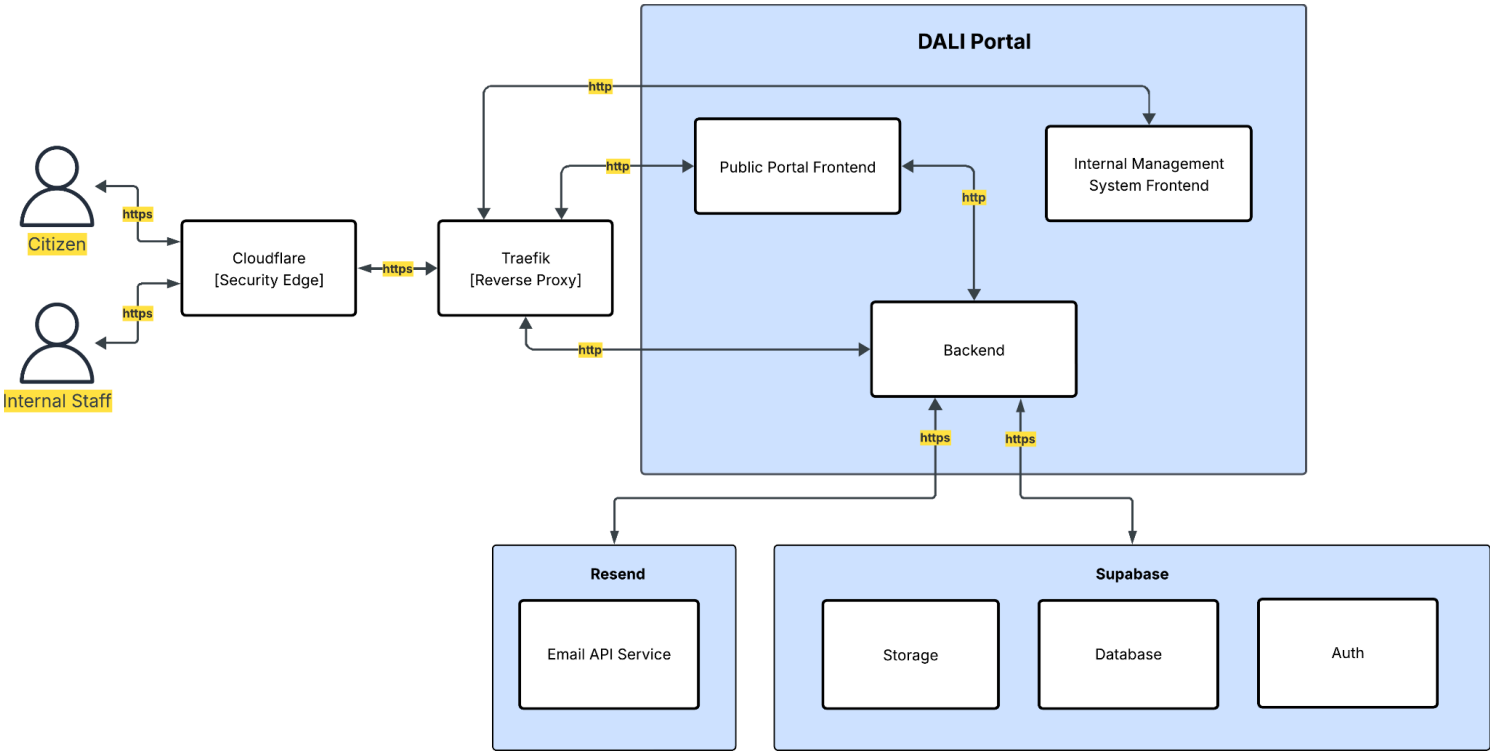
3.1 Business Context



	Business Elements	Description
BE-1	Public	Represents the citizens of Iloilo City who interact with the Public Portal. They can view ordinances, resolutions, session schedules, minutes, and journals. They can also submit inquiries or assistance requests and track the status of their tickets.
BE-2	IT Admin	The main person managing the system. Has full access to the features for general management.

		Their main goal is to assign each user their roles. These roles are defined from BE3-BE8.
BE-3	Vice Mayor(VM)	The highest-level internal user who oversees operations and decision-making. Their primary system interactions involve approving "initialed" documents and reviewing/responding to public inquiries escalated to their level.
BE-4	VM Office Staff	Staff members who are responsible for the "public-facing" operations. They handle inquiry responses, review digital caller slips, track beneficiary records, and approve conference room bookings.
BE-5	VM Admin Office Staff	Staff responsible for day-to-day administrative duties. They use the system to log and track incoming/outgoing documents, request "initial" status for documents, generate caller slips, and manage conference room bookings.
BE-6	VM Admin Office Staff Head	The Head Admin Staff with elevated privileges compared to standard Admin Staff. Their primary unique function is the authority to review and approve documents requested for "Initial" status before they reach the Vice Mayor.
BE-7	City Councilor	Legislative members who use the system primarily to view session files during weekly sessions and request bookings for the conference room.
BE-8	Legislative Staff	Has elevated privilege compared to the standard Admin Office Staff, with the ability to manage sessions, build agendas, and upload session minutes and journals.
BE-9	DALI Portal	The core system that connects the public portal and the internal management system. It serves as a centralized platform for document transparency, workflow automation, and digital interaction between the public and government offices.

3.2 Technical Context



	Technical Elements	Description
TE-1	Citizen	The external client environment (Desktop or Mobile Web Browser) used by the public to access the public portal. It executes the client-side bundle and handles UI rendering.
TE-2	Internal Staff	The internal client environment (Desktop/Laptop Web Browser) used by government personnel to access the secure IMS.
TE-3	Cloudflare	Acts as the Security Edge. It routes incoming traffic, provides DDoS protection, IP-based rate limiting, and Web Application Firewall (WAF) capabilities before traffic reaches the core server.
TE-4	Traefik	A reverse proxy hosted on the Virtual Private Server (VPS). It manages internal routing,

		directing traffic to the correct Docker containers (Next.js Frontend or NestJS Backend) via a private HTTP network.
TE-5	Public Portal Frontend	A containerized Next.js application responsible for rendering the public-facing website. It provides open access to legislative archives and the inquiry submission form.
TE-6	Internal Management System Frontend	A containerized Next.js application responsible for the secure, authenticated dashboard used by staff. It handles the UI for document tracking, session management, and ticketing.
TE-7	Backend	A containerized NestJS application. It serves as the API layer handling business logic that cannot be offloaded directly to the database, communicating with Supabase and the Frontend.
TE-8	Email Service API	An external service used to send automated transactional emails, such as inquiry confirmations, tracking numbers, and booking notifications.
TE-9	Supabase Storage	Managed file storage service used to store and serve large binary files, specifically PDFs for ordinances/resolutions, and image attachments for inquiries.
TE-10	Supabase Database	A managed PostgreSQL database. It stores all structured system data (user records, logs, document metadata, ticket history) and implements Row-Level Security (RLS) for access control.
TE-11	Supabase Auth	Managed authentication service. It handles user identity verification, secure sign-in, session management, and JWT issuance for internal staff and administrators.

4. Solution Strategy

This chapter and the table below outlines architectural approaches taken by the developers to address each quality goal.

Num	Quality Goal	Architectural Approach	Details
1	Security	**Cloudflare** is utilized for DDoS protection, WAF, and enforcing HTTPS encryption.	*See 8.2 Security and Role-Based Access Control* *See 8.3 Deployment and Infrastructure*
		Passwords and sessions are managed via **Supabase Auth** (Bcrypt hashing, JWT tokens) to avoid custom auth vulnerabilities.	
		The system enforces strict **Role-Based Access Control (RBAC)**. Additionally, **Row-Level Security (RLS)** is enabled on the PostgreSQL database, ensuring that even if the API is bypassed, a user cannot query data they are not explicitly permitted to see.	
		Public forms integrate **CAPTCHA** to prevent bot attacks.	
2	Usability	To accommodate the general public (BE-1), the system employs a Responsive Web Design using Tailwind CSS. The design adheres to strict color contrast and clear labeling standards. **Server-Side Rendering (SSR)** is used for the Public Portal to ensure content loads immediately for citizens on slower mobile connections, improving perceived performance.	*See 8.4.2 Server Side Rendering* *See 8.5 User Experience*
3	Operability	To streamline staff operations, the Internal Management System (IMS) utilizes **Client-Side Rendering (CSR)**. The interface is dynamic: "Vice Mayor"	*See 8.2.2 Role-Based Access Control*

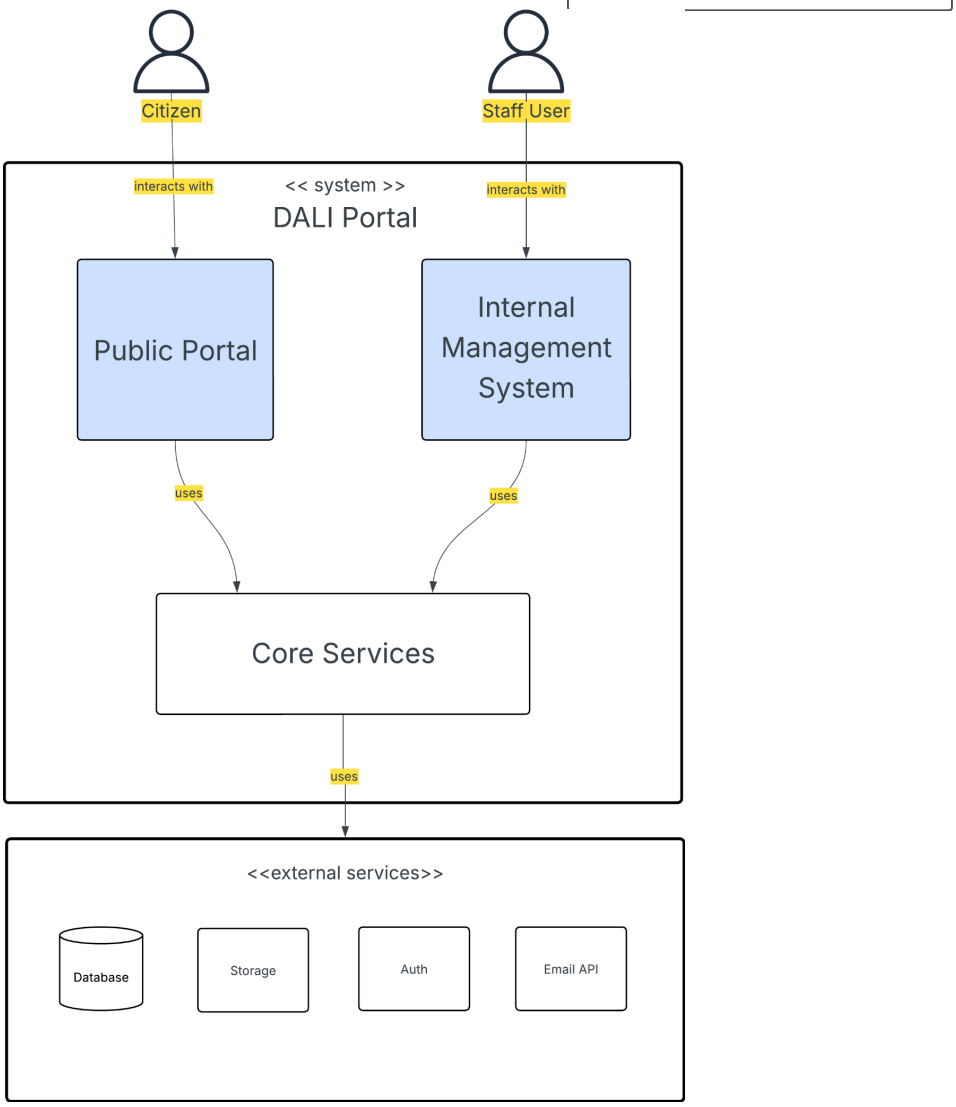
		sees approval queues, while "Head Admin" sees documents pending initials. This role-specific filtering automates the management of backlogs and ensures staff only see relevant data.	*See 8.4.1 Client Side Rendering*
4	Reliability	The architecture is distributed to avoid single points of failure. **Supabase** handles data persistence with automated backups.	*See 8.3.1 Cloud Based Architecture*
		The application logic runs in isolated **Docker** containers managed by **Coolify**, ensuring that if one service fails, it can be restarted independently without taking down the database or static assets.	*See 8.3.2 Containerization and Orchestration*
		Resend ensures reliable transactional email delivery.	*See 8.3.5 Email Service Integration*
5	Maintainability	The frontend is built with **React (Next.js)** using small, reusable components that adhere to the Single Responsibility Principle (SRP). This modular design simplifies debugging, testing, and future UI updates.	
		The backend utilizes **NestJS**, providing a structured, opinionated architecture that enforces strict separation between controllers, services, and repositories, making logic modifications safer and easier.	*See 8.3.2 Containerization and Orchestration*
		The use of **Docker** ensures consistent environments across development and production, making it easier to deploy updates or fix bugs without configuration drift.	*See 8.4 Frontend Design Patterns*

5. Building Blocks

This chapter describes the main parts of the DALI Portal and how they work together as one connected platform. The system is divided into two areas: the public portal and the internal management system.

5.1 Building Block View: Level 1

5.1.1 Whitebox: DALI Portal

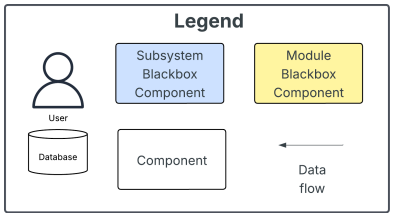


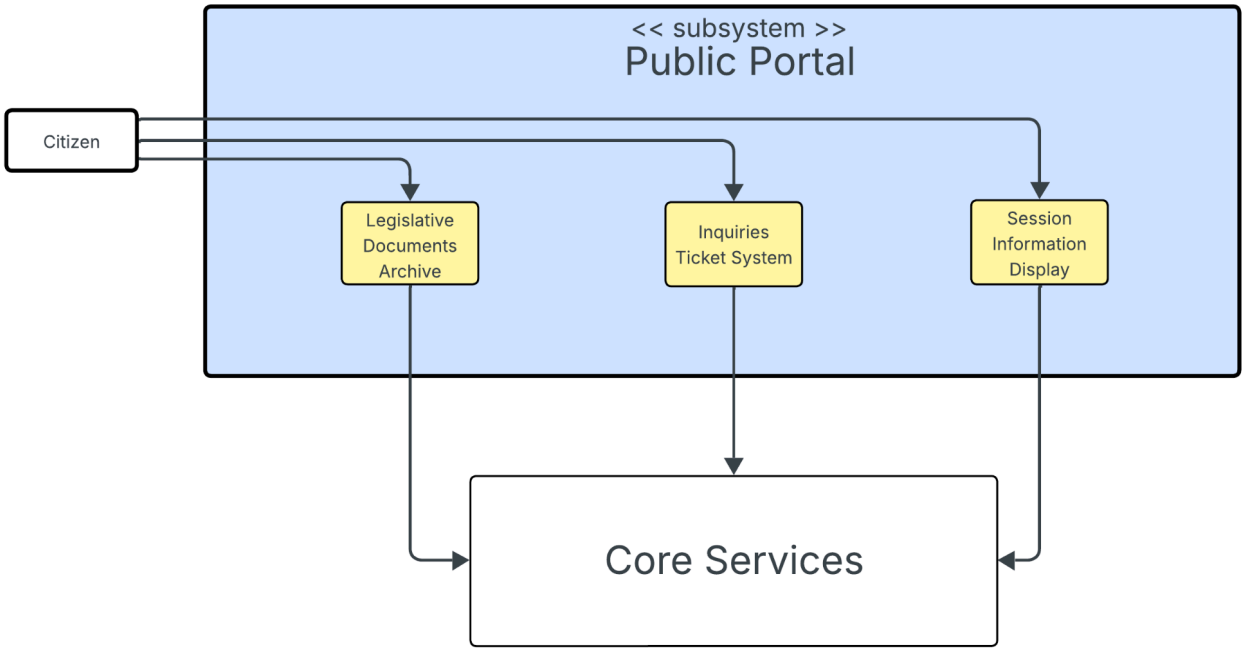
5.1.2 Contained Blackboxes

Blackbox	Intent/Responsibility
Public Portal	The interface for citizens. Its responsibilities are limited to displaying public data and submitting inquiries.
Internal Management System	A separate and secure application for all internal Staff Users. It provides the complete interface for all operational workflows.
Core Services	The backend that serves as the "brain" for both interfaces. It enforces all business logic.
Database	Storage service that stores all structured application data, such as user roles and document metadata.
Storage	File storage service responsible for securely storing and retrieving all files uploaded to the application, primarily the scanned PDFs of official documents.
Auth	External authentication service responsible for managing user credentials, handling the login/logout process, issuing secure tokens, and managing user sessions.
Email API	External email service for sending automated emails from the system.

5.2 Building Block View: Level 2

5.2.1 Blackbox of Public Portal

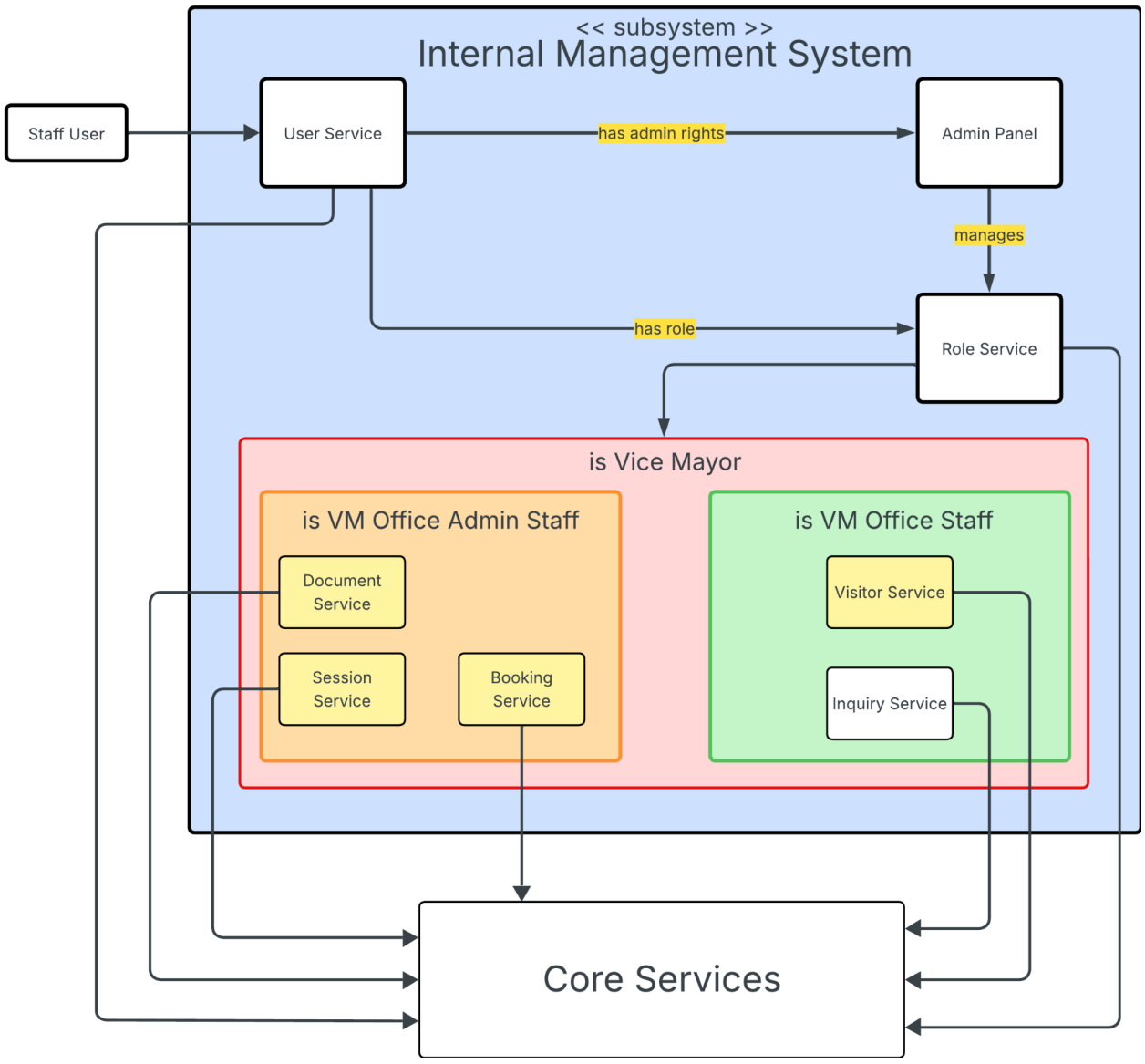




Contained Blackboxes

Blackbox	Intent/Responsibility
Legislative Documents Archive	Allows the public to search, filter, and access legislative documents, including ordinances, resolutions, and other official records.
Inquiries Ticket System	Enables citizens to submit inquiries or requests to the office of the Vice Mayor.
Session Display Information	Displays information about official sessions, such as schedules and agendas.

5.2.3 Blackbox of Internal Management System



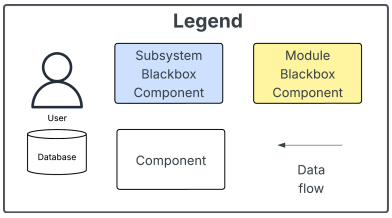
Contained Blackboxes

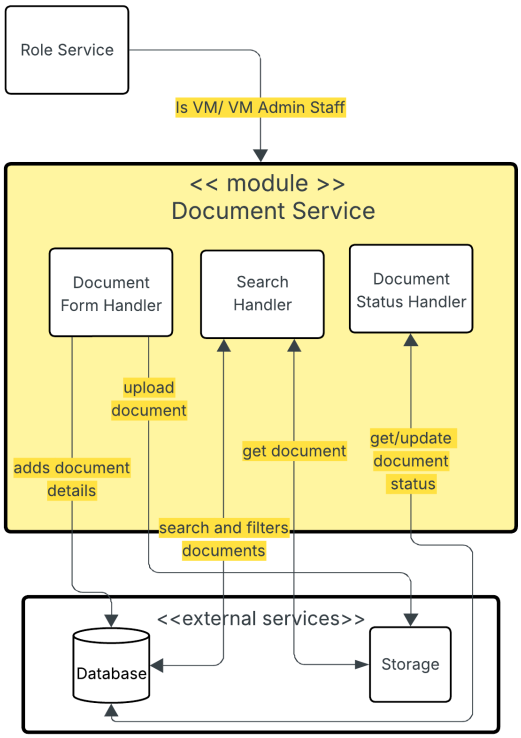
Blackbox	Intent/Responsibility
User Service	Handles authentication and provides user information to other services.
Admin Panel	Provides the administrative user interface for managing user roles.

Role Service	Handles role definitions, permissions, and authorization checks. Ensures correct access control for all internal services and the Admin Panel.
Document Service	Stores, retrieves, and manages legislative documents. Ensures secure access and proper lifecycle handling of documents.
Session Service	Handles building of session agenda and management of session schedules, minutes, and journals.
Booking Service	Handles scheduling logic, availability checking, and booking status management.
Visitor Service	Manages visitor and beneficiary profiles tracked in the system.
Inquiry Service	Provides an access point for staff to process inquiries submitted by citizens.

5.3 Building Block View: Level 3

5.3.1 Blackbox of Document Service

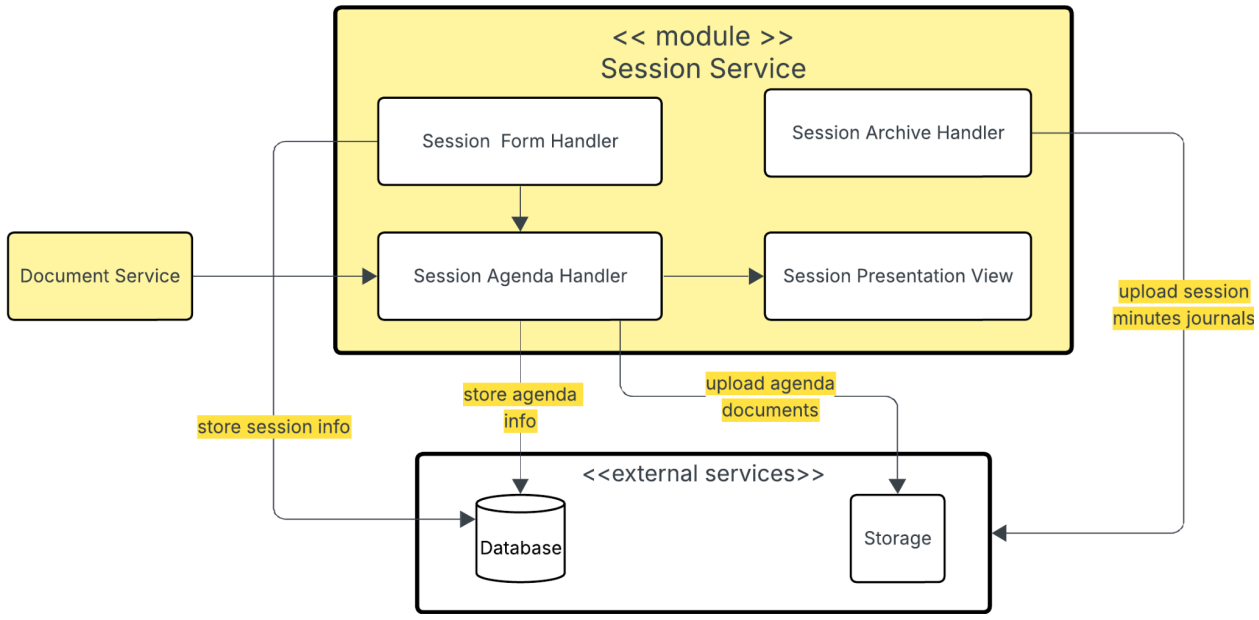




Contained Blackboxes

Blackbox	Intent/Responsibility
Search Handler	Allows the staff user to look for specific documents by document ID or name.
Document Form Handler	Allows the staff to log documents by uploading and filling in document information.
Document Status Handler	Allows the staff to see the current status of the document, its history of status changes, and update its status based on the proper workflow.

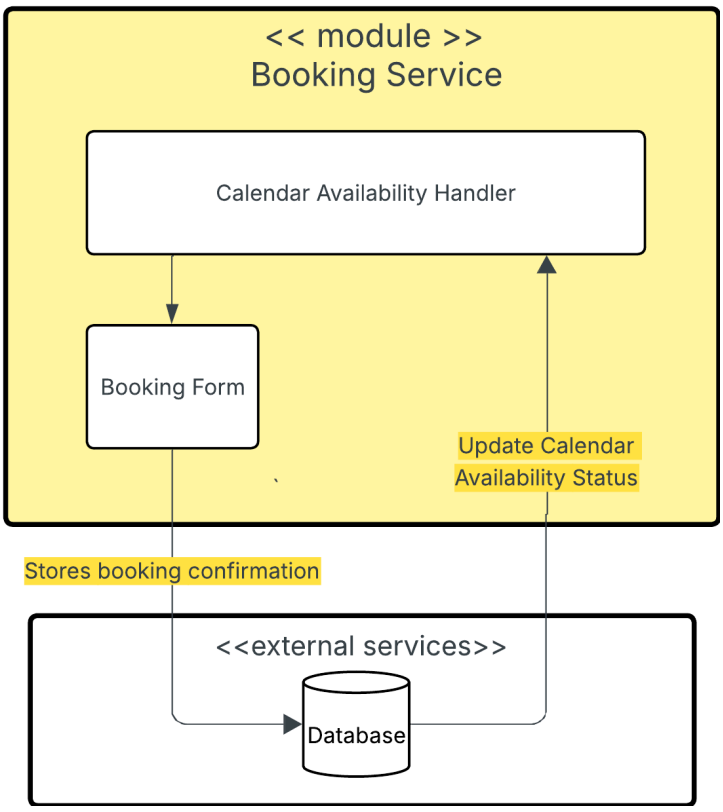
5.3.2 Blackbox of Session Service



Contained Blackboxes

Blackbox	Intent/Responsibility
Session Form Handler	Allows staff to add new session schedules along with necessary information.
Session Agenda Handler	Provides an easy way for staff to view and build session agendas by
Session Presentation Handler	Allows staff to present a session with intuitive controls for navigating to other sections of the agenda.
Session Archive Handler	Allows staff to access past sessions, view, and upload minutes and journal documents.

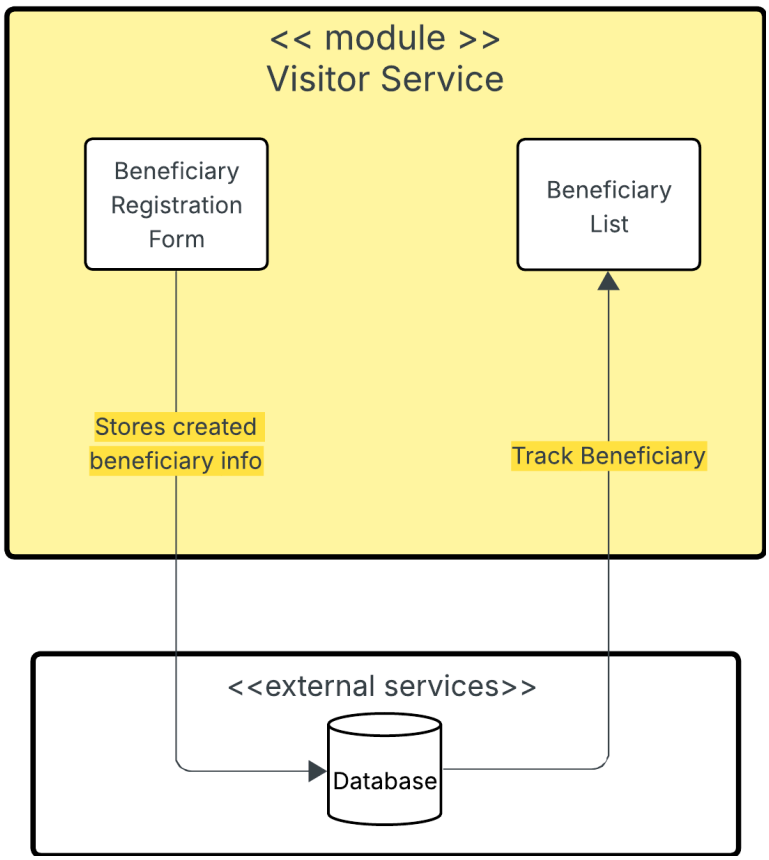
5.3.3 Blackbox of Booking Service



Contained Blackboxes

Blackbox	Intent/Responsibility
Calendar Availability Handler	Allows councilors to check available time slots for booking the conference room and ensures no scheduling conflict occurs by providing real-time availability data.
Booking Form	Provides an interface for councilors to input booking details and submit a conference room booking request.

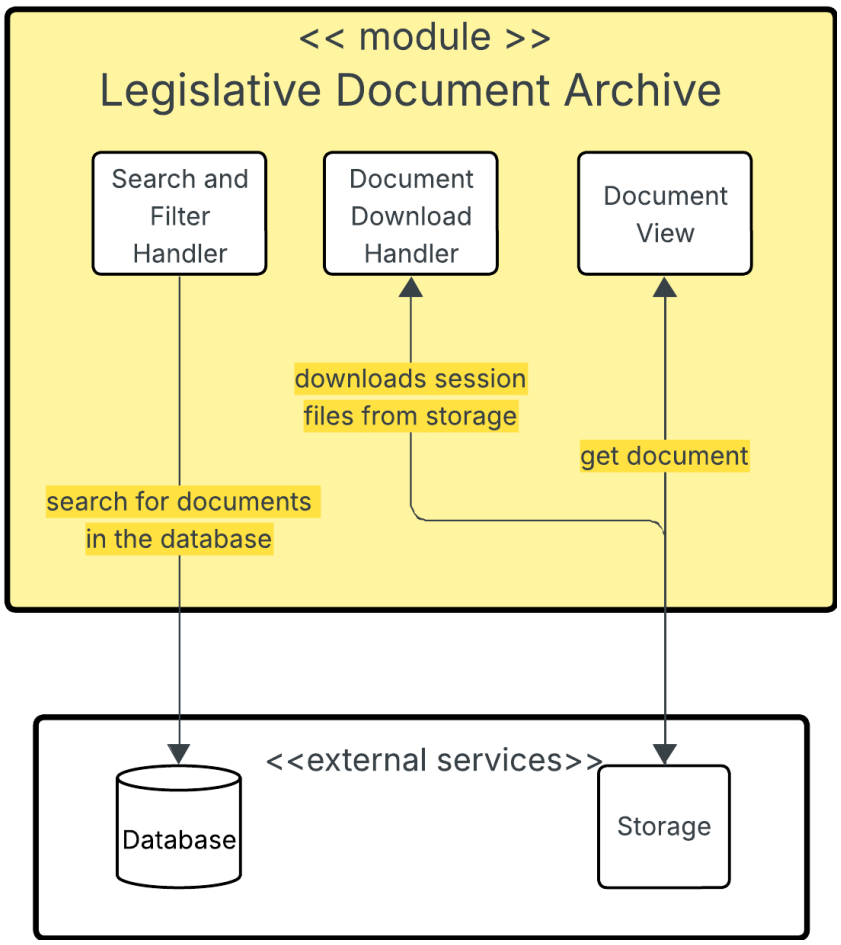
5.3.4 Blackbox of Visitor Service



Contained Blackboxes

Blackbox	Intent/Responsibility
Beneficiary Registration Form	Records and validates all necessary information about the beneficiary.
Beneficiary List	An archive of beneficiaries to properly monitor benefits handed out.

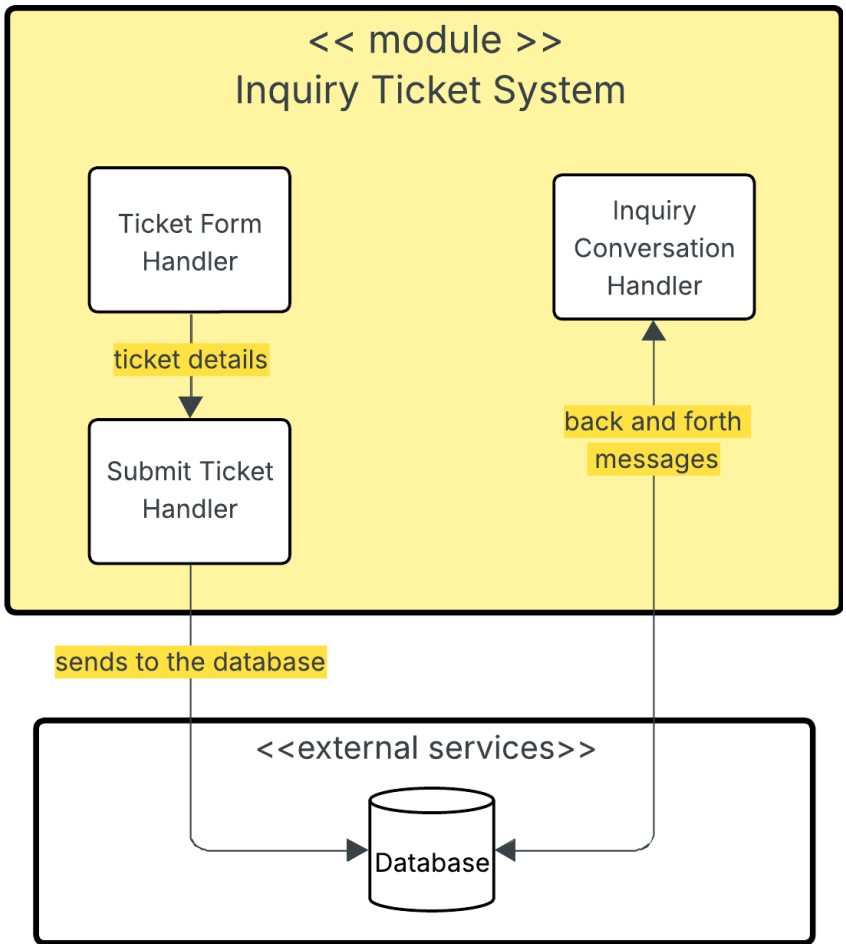
5.3.5 Blackbox of Legislative Document Archive



Contained Blackboxes

Blackbox	Intent/Responsibility
Search and Filter Handler	Helps users find specific legislative documents through searching and narrowing down results.
Document View	Displays the selected legislative document for users to read.
Document Download Handler	Allows users to download a copy of the legislative document.

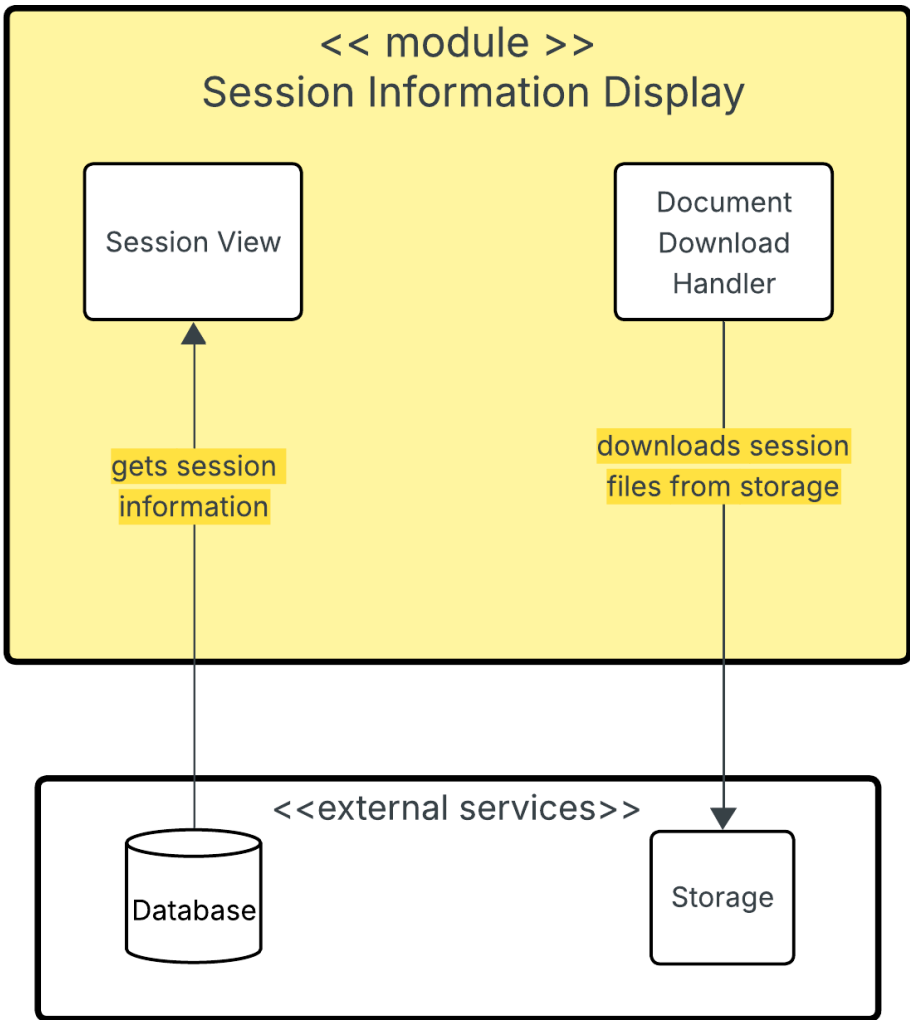
5.3.6 Blackbox of Inquiries Ticket System



Contained Blackboxes

Blackbox	Intent/Responsibility
Ticket Form Handler	Collects the information that a resident provides when submitting an inquiry.
Submit Ticket Handler	Saves the inquiry to the system and sends it to the appropriate office staff.
Inquiry Conversation Handler	Handles the back-and-forth messages between the resident and the office staff for an inquiry.

5.3.7 Blackbox of Session Information Display



Contained Blackboxes

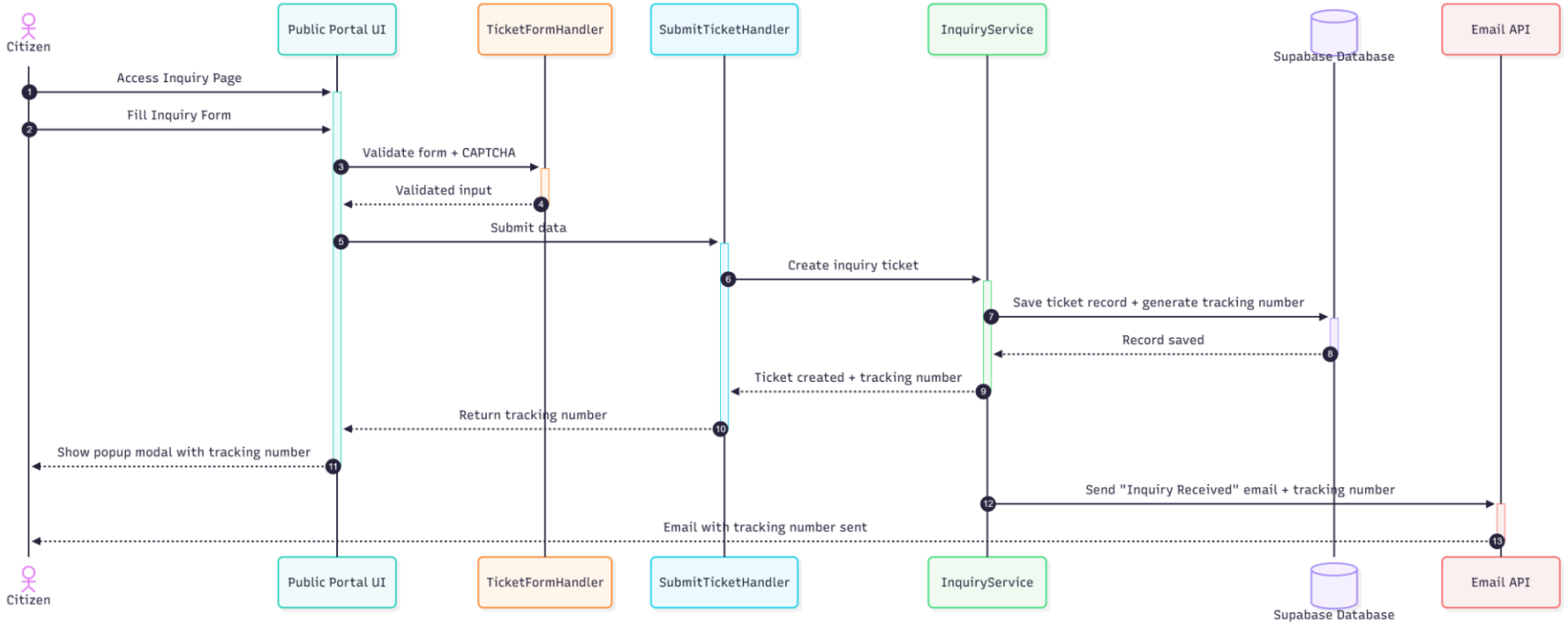
Blackbox	Intent/Responsibility
Session View	Displays the details and information about the selected session.
Document Download Handler	Allows users to download the files and documents related to a session.

6. Runtime View

This chapter contains use case scenarios and their interactions with the building blocks of the application at runtime. It denotes the use case and its chronological interaction, along with its description within the building blocks of the system.

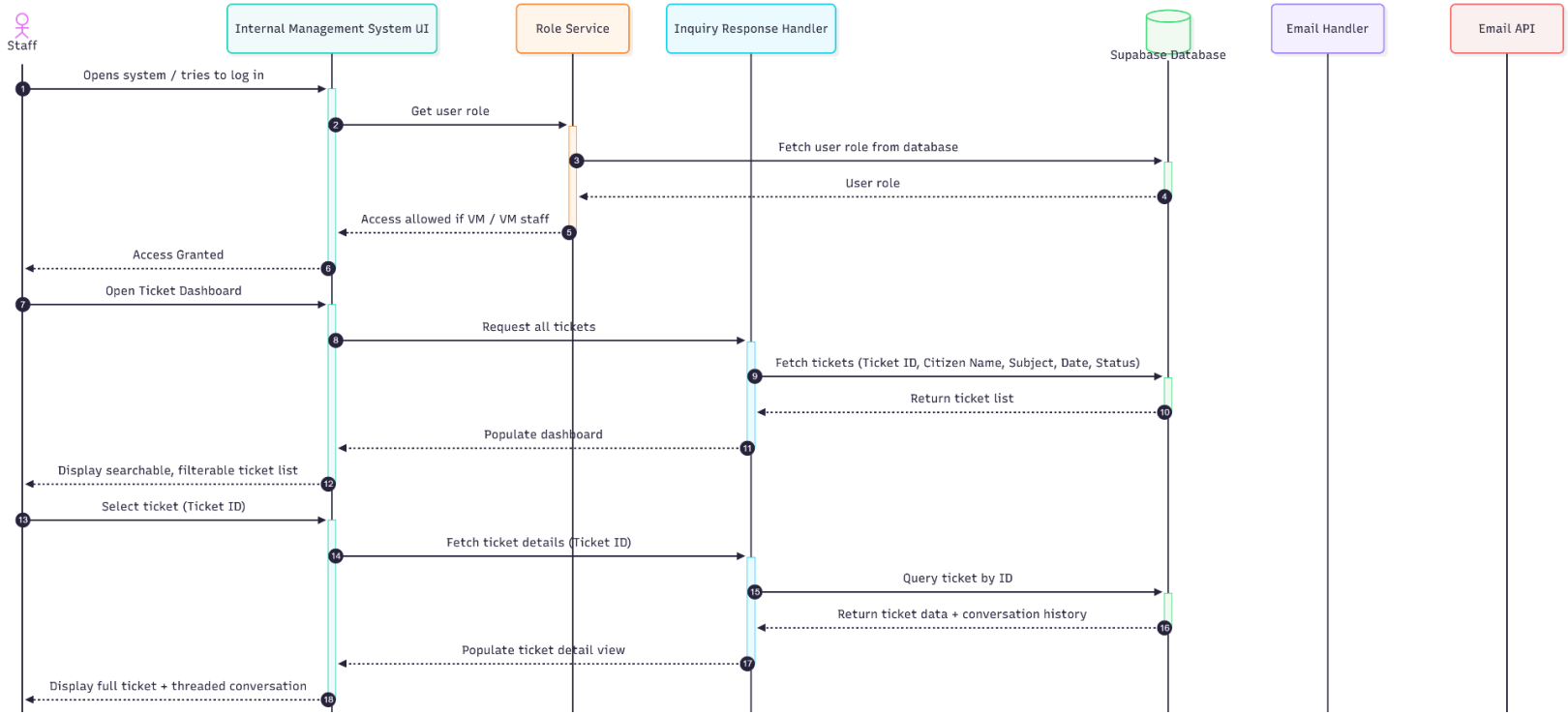
6.1 Runtime Scenario 1: Citizen Submits an Inquiry (Public Portal)

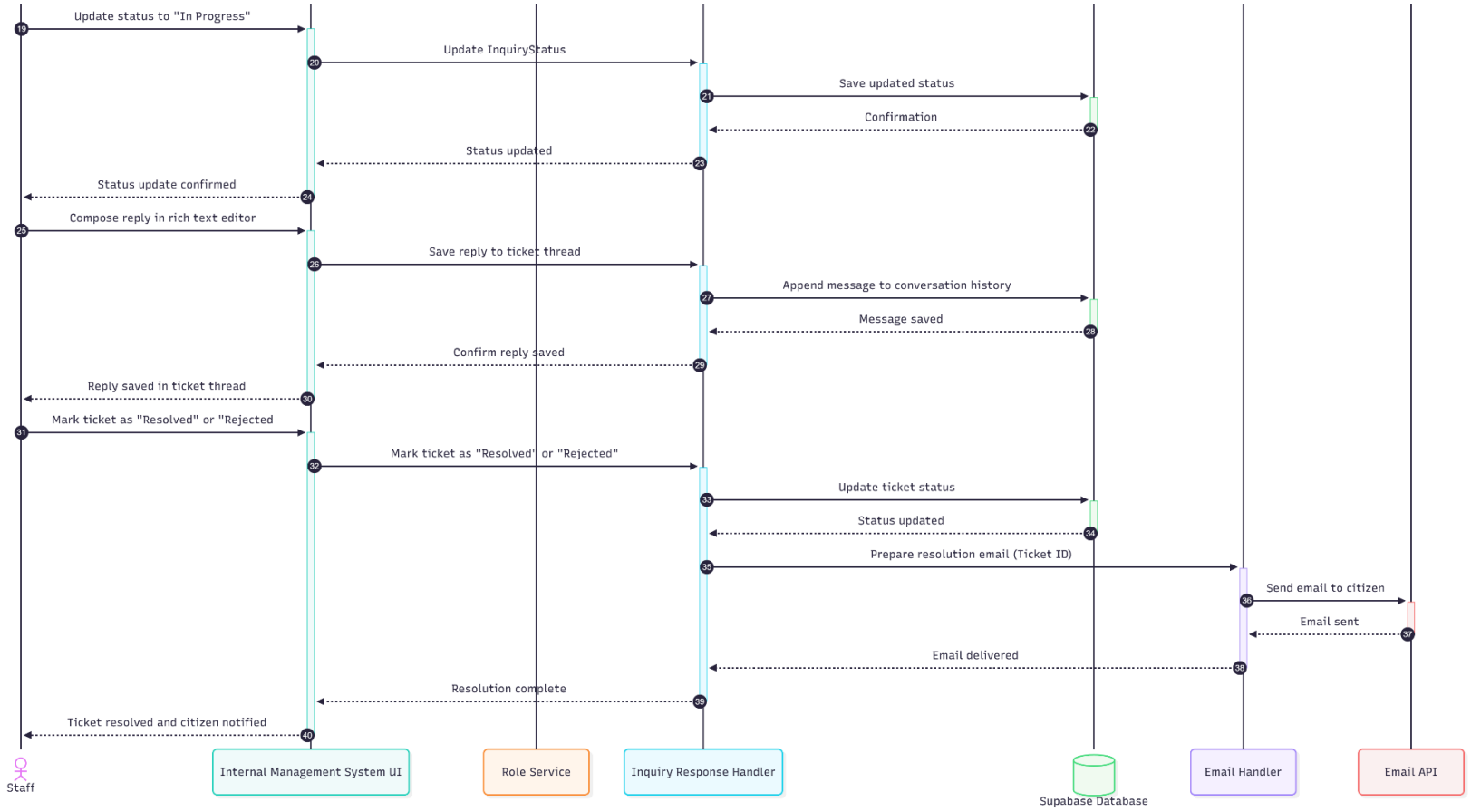
This runtime scenario explains what happens after a citizen submits an inquiry through the portal. The system checks the form, saves the inquiry to the database, and generates a tracking number. The citizen sees this tracking number in a pop-up and also receives it through email. After that, the inquiry appears in the Internal Management System so staff can begin processing it.



6.2 Runtime Scenario 2: Staff Processes an Inquiry Ticket (Internal Workflow)

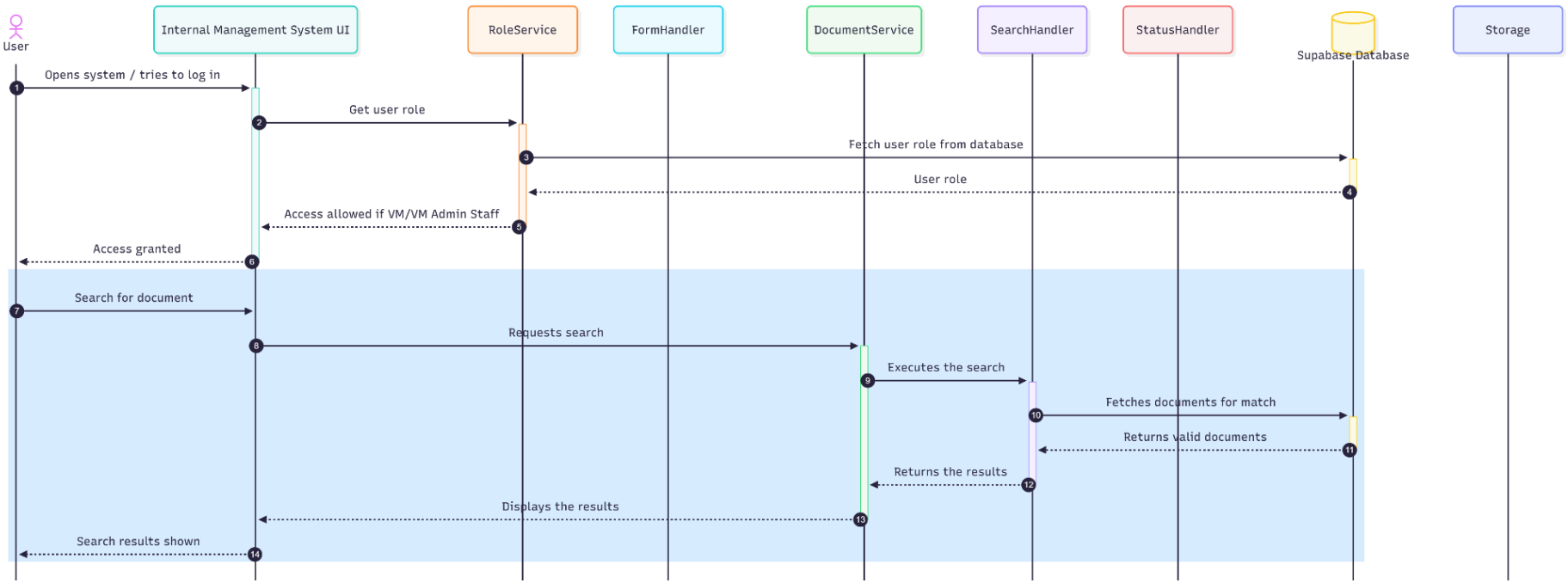
This runtime scenario aims to describe how staff receive, review, update, and resolve or reject the inquiry ticket that has been submitted into the system by the citizen. The sequence of interaction is initiated after the inquiry already exists in the Supabase database.



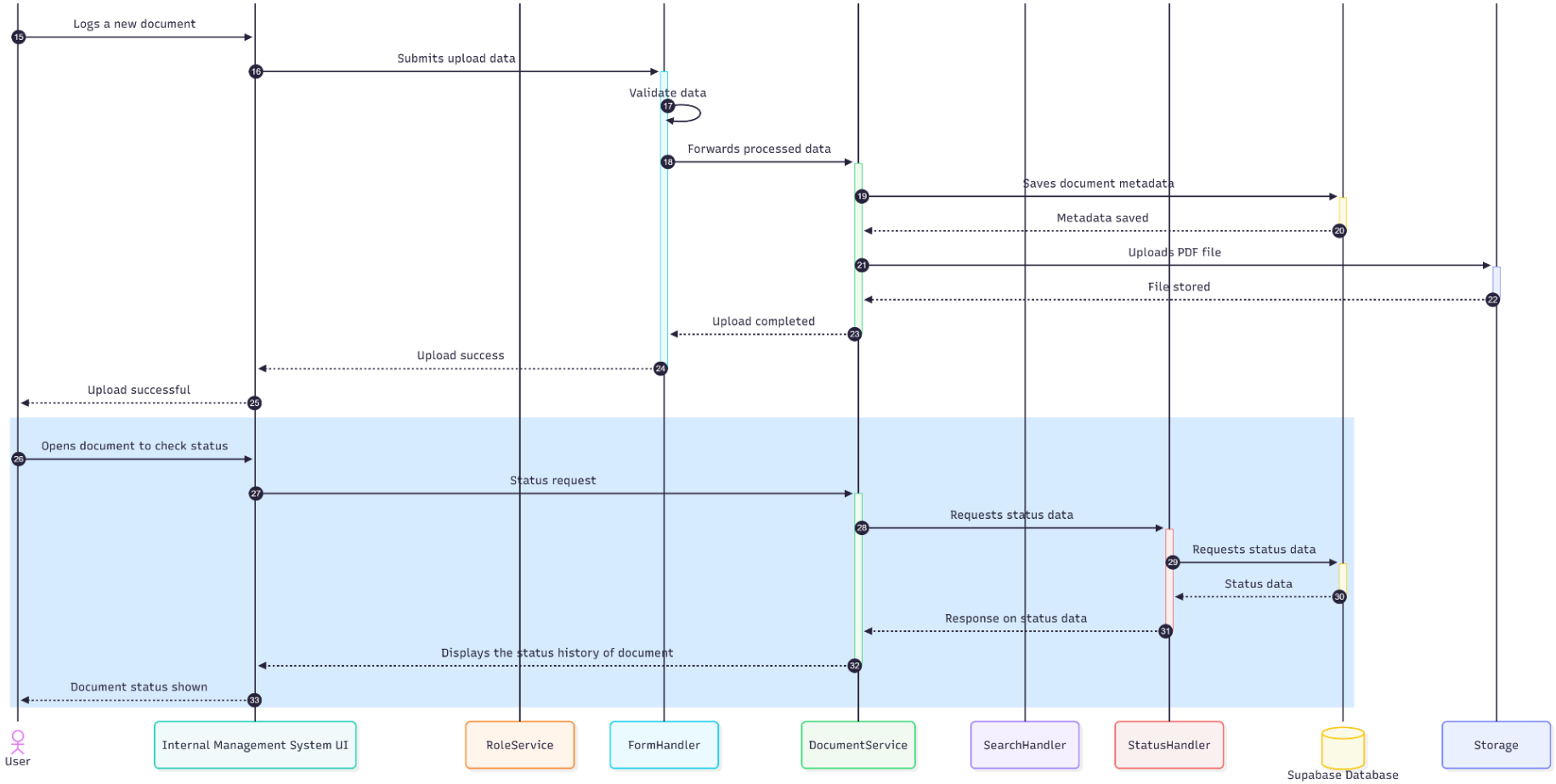

Update status to "In Progress"
19
Update InquiryStatus
20
Save updated status
21
Confirmation
22
Status updated
23
Status update confirmed
24
Compose reply in rich text editor
25
Save reply to ticket thread
26
Append message to conversation history
27
Message saved
28
Confirm reply saved
29
Reply saved in ticket thread
30
Mark ticket as "Resolved" or "Rejected"
31
Mark ticket as "Resolved" or "Rejected"
32
Update ticket status
33
Status updated
34
Prepare resolution email (Ticket ID)
35
Send email to citizen
36
Email sent
37
Email delivered
38
Resolution complete
39
Ticket resolved and citizen notified
40
Staff
Internal Management System UI
Role Service
Inquiry Response Handler
Supabase Database
Email Handler
Email API

6.3 Runtime Scenario 3: Document Tracking

This runtime scenario depicts how staff interact with the Internal Management System to search documents, upload new files, and check document status. It depicts the flow of requests between the user interface, backend services, and the Supabase database, demonstrating how each operation is completed from beginning to end.



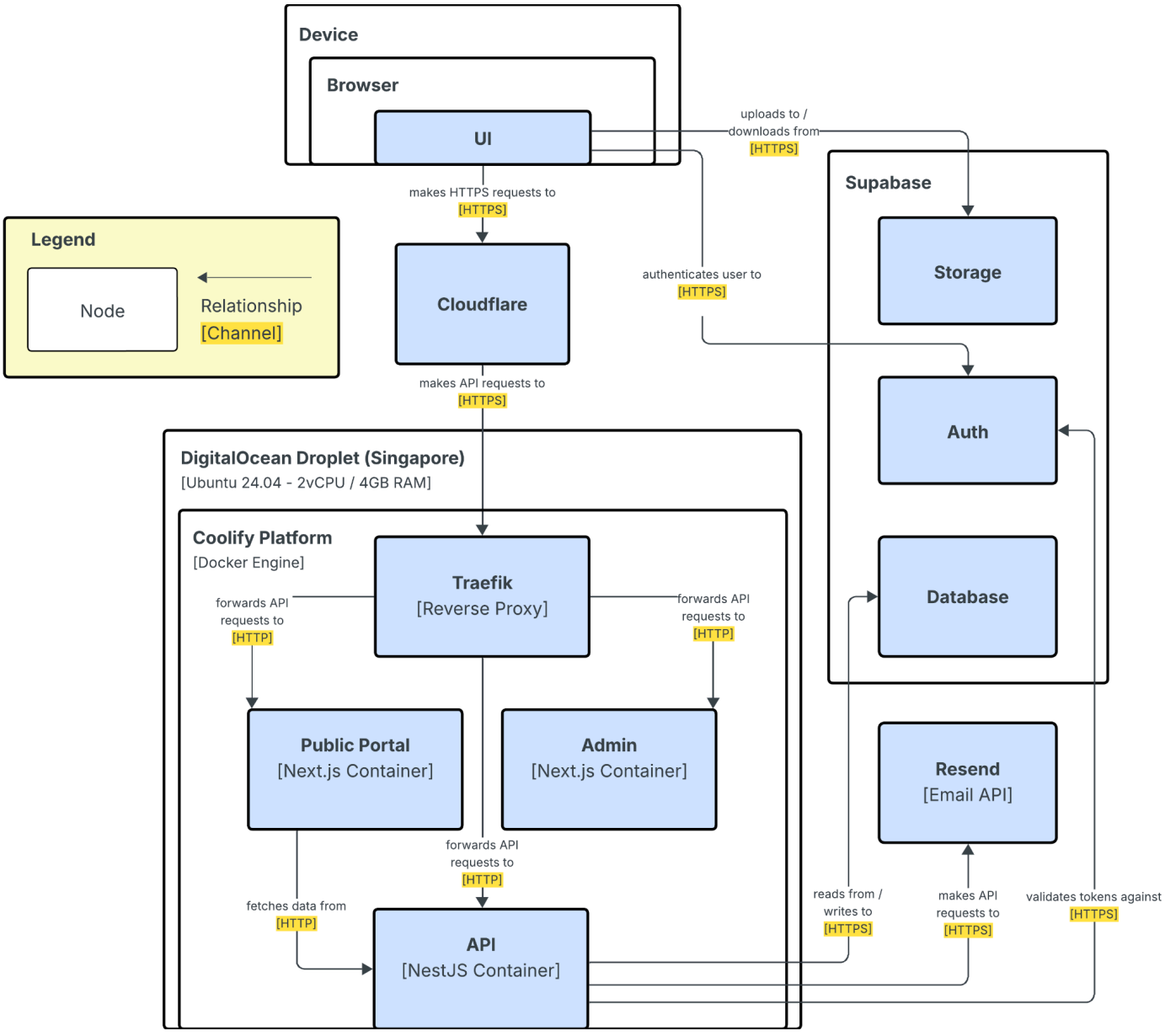
PEAK SOLUTIONS



7. Deployment View

This chapter contains the technical infrastructure of the system when it is distributed and accessed by different users and maps the software building blocks to those elements.

7.1 Overview of the Deployment View



The DALI Portal is deployed with a modern, containerized cloud architecture for cost-efficiency, security, and scalability. The system is distributed across three primary cloud providers to leverage the specific strengths of each:

1. **Security Edge** (Cloudflare): All incoming traffic is routed through Cloudflare's global edge network. This layer provides DDoS protection, IP-based rate limiting, and Web Application Firewall (WAF) capabilities.

2. **Compute Runtime** (DigitalOcean): The core application is hosted on a Virtual Private Server (VPS) located in Singapore to minimize latency for users based in Iloilo. The server utilizes Coolify to orchestrate a Dockerized environment. A Traefik reverse proxy manages internal routing, directing traffic to the Next.js (Frontend) and NestJS (Backend) containers:
    a. dali.iloilocity.gov.ph routes to the Public Portal container
    b. admin.dali.iloilocity.gov.ph routes to the Admin / IMS container
    c. api.dali.iloilocity.gov.ph routes to the API container

3. **Managed Data Infrastructure** (Supabase): To reduce the load on the VPS and ensure data durability, stateful components are offloaded to Supabase. This includes the PostgreSQL database, File Storage for legislative document PDFs, and Authentication services. The client-side application communicates directly with Supabase for file uploads, bypassing the backend bottleneck.

7.1.1 Elements of the Deployment View

Name	Purpose
Device	The physical device (Laptop, Smartphone, Tablet) used by Citizens or Staff to access the system.
Browser	The runtime environment that executes the Client-Side (UI) Bundle. It handles the UI rendering, hydration, and direct interaction with Supabase.
Cloudflare	Acts as the secure entry point. It handles DNS resolution, blocks bot traffic, and mitigates DDoS attacks before forwarding legitimate requests to the VPS.
DigitalOcean Droplet	A Linux (Ubuntu) Virtual Private Server located in Singapore (2 vCPU / 4GB RAM). It hosts the Docker engine and runtime environment.

Coolify Platform	Manages the Docker containers, ensuring they restart automatically if they crash. It also handles build pipelines and deployment from GitHub.
Traefik	A reverse proxy that receives traffic from Cloudflare and routes it to the correct container based on the domain name.
Public Portal	Runs the Next.js application for citizens.
Admin / IMS	Runs the Next.js application for internal staff.
API	Runs the NestJS backend.
Supabase Storage	Handles file uploads such as documents, images, and attachments required by the DALI Portal.
Supabase Auth	Manages user identities, password hashing, and JWT token generation. It provides the API for login/logout operations.
Supabase Database	A managed PostgreSQL database. It stores all structured data (Users, Documents, Logs) and enforces Row-Level Security (RLS).
Resend	A transactional email service used by the API to send emails to users.

8. Crosscutting Concepts

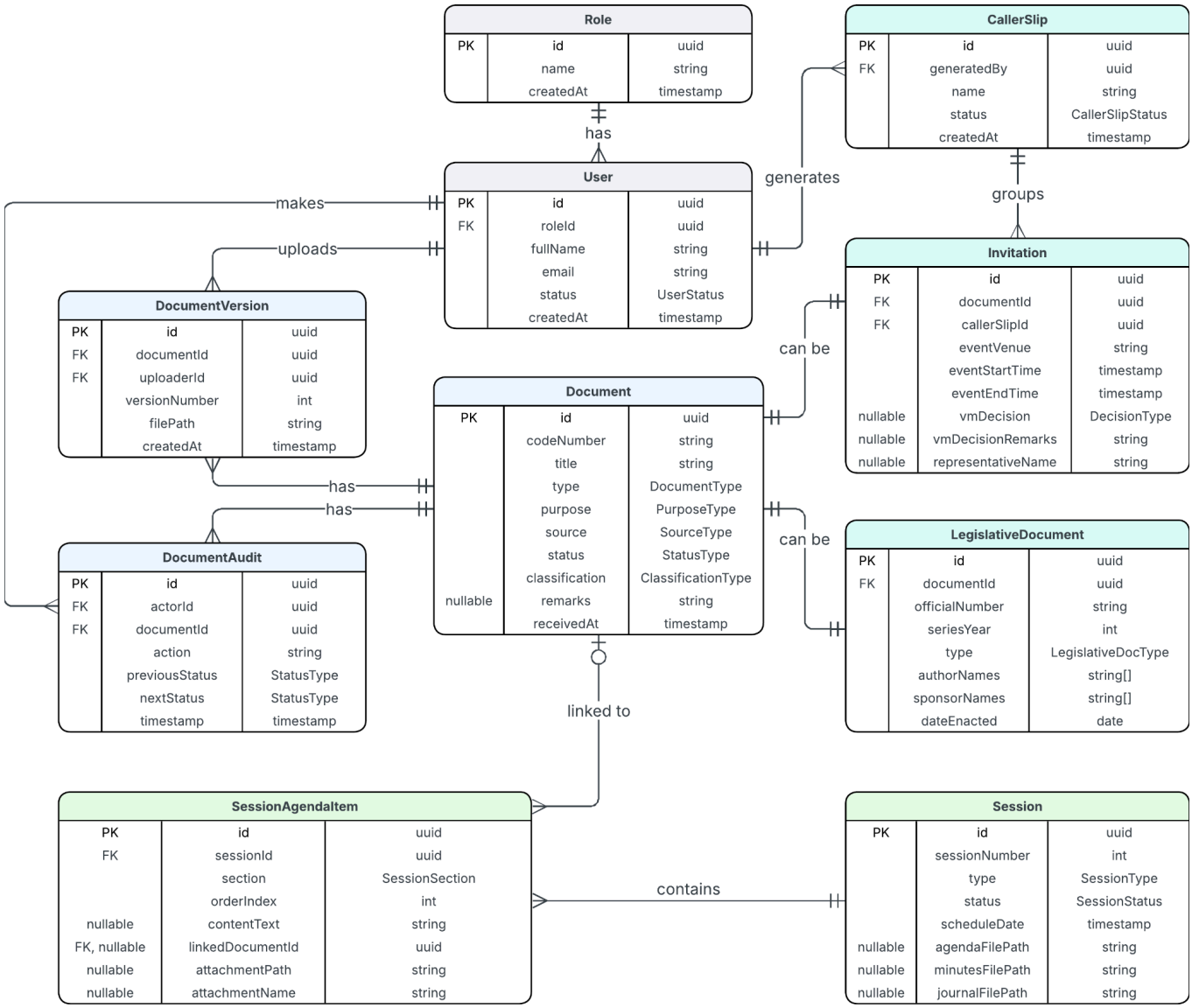
This chapter describes concepts that are relevant to multiple parts (cross-cutting) of the system. These concepts are often related to the building blocks detailed in Chapter 5.

8.1 Domain Concepts

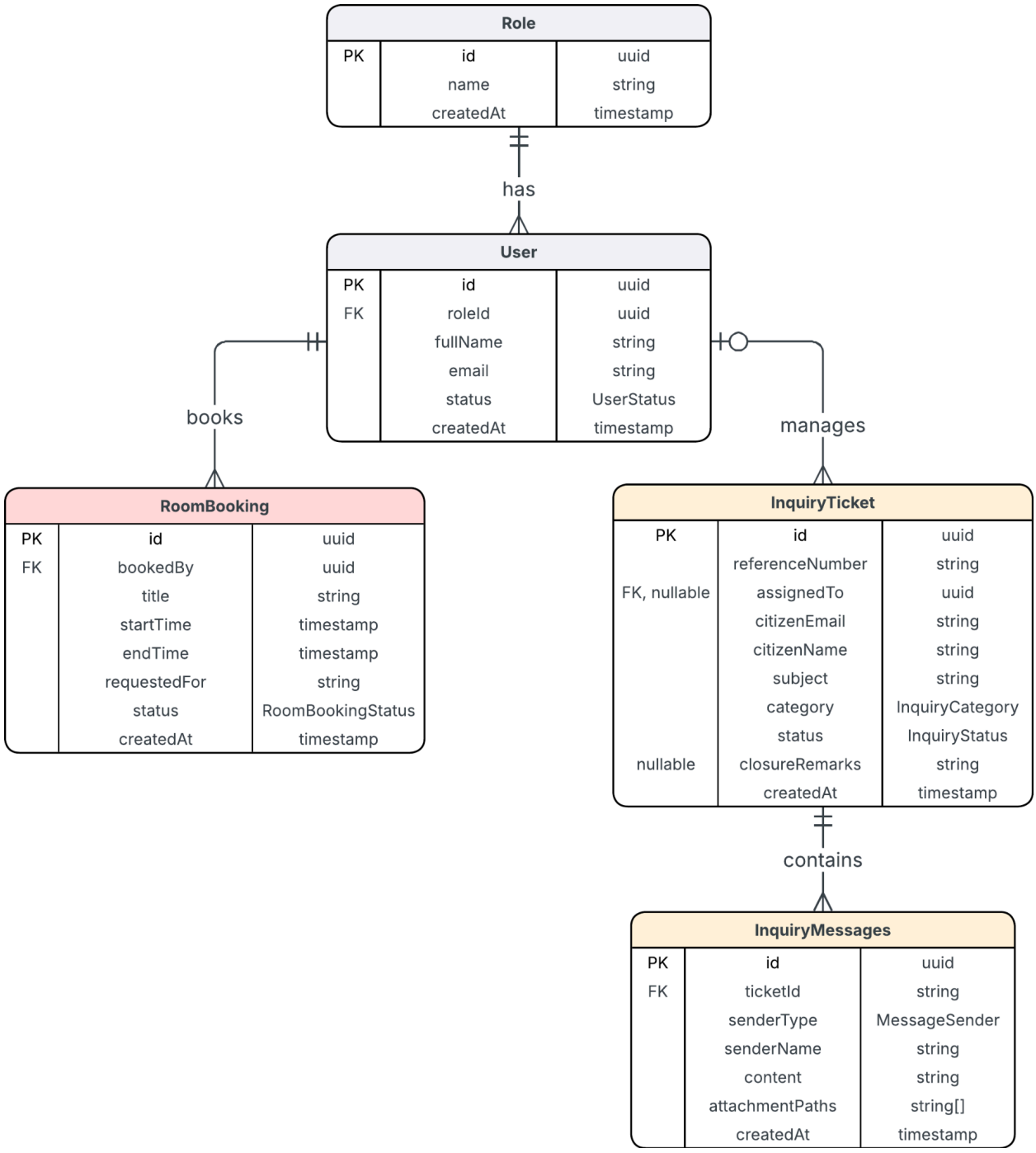
8.1.1 Domain Models

These models represent the core business entities and their relationships. The following are separate views for the **Entity-Relationship Diagram (ERD)** defined by the system.

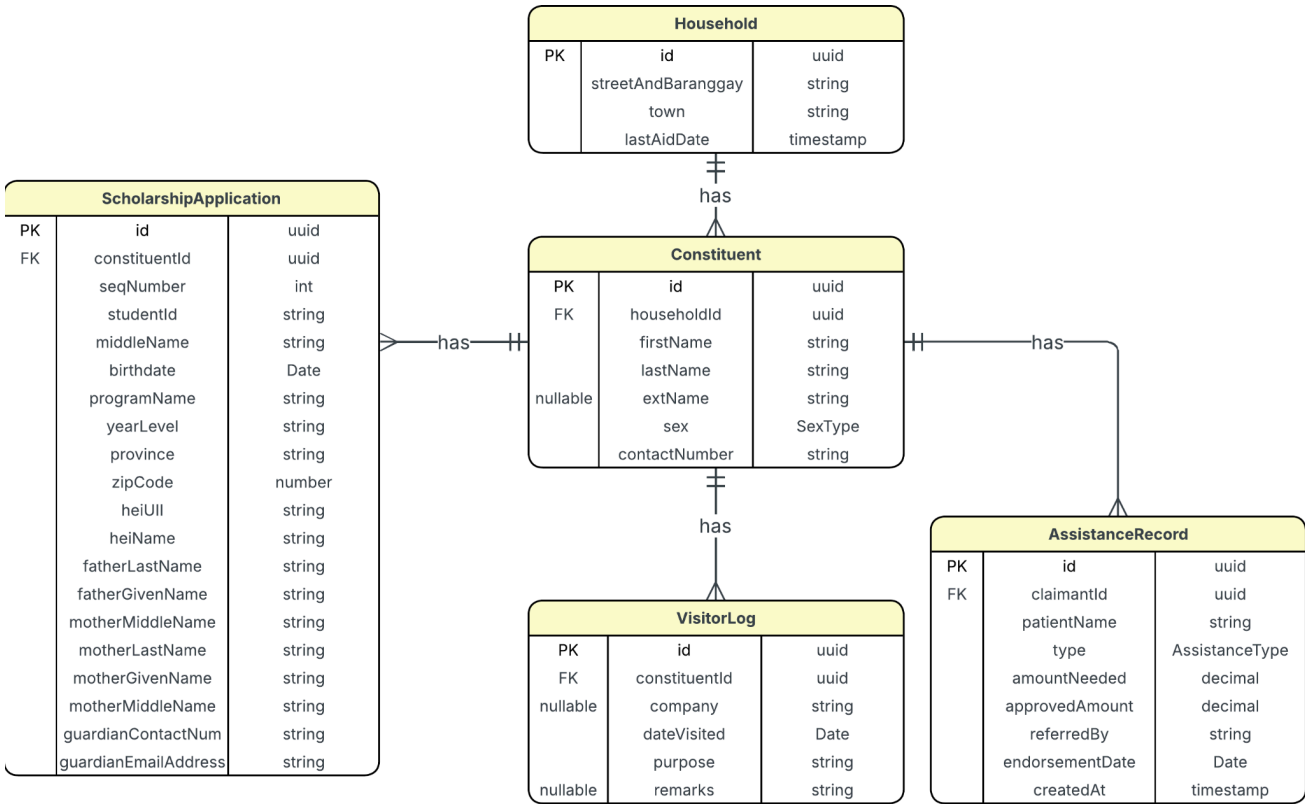
Documents and Sessions (Part 1 of 3)



Inquiries and Bookings (Part 2 of 3)



Visitors and Beneficiaries (Part 3 of 3)

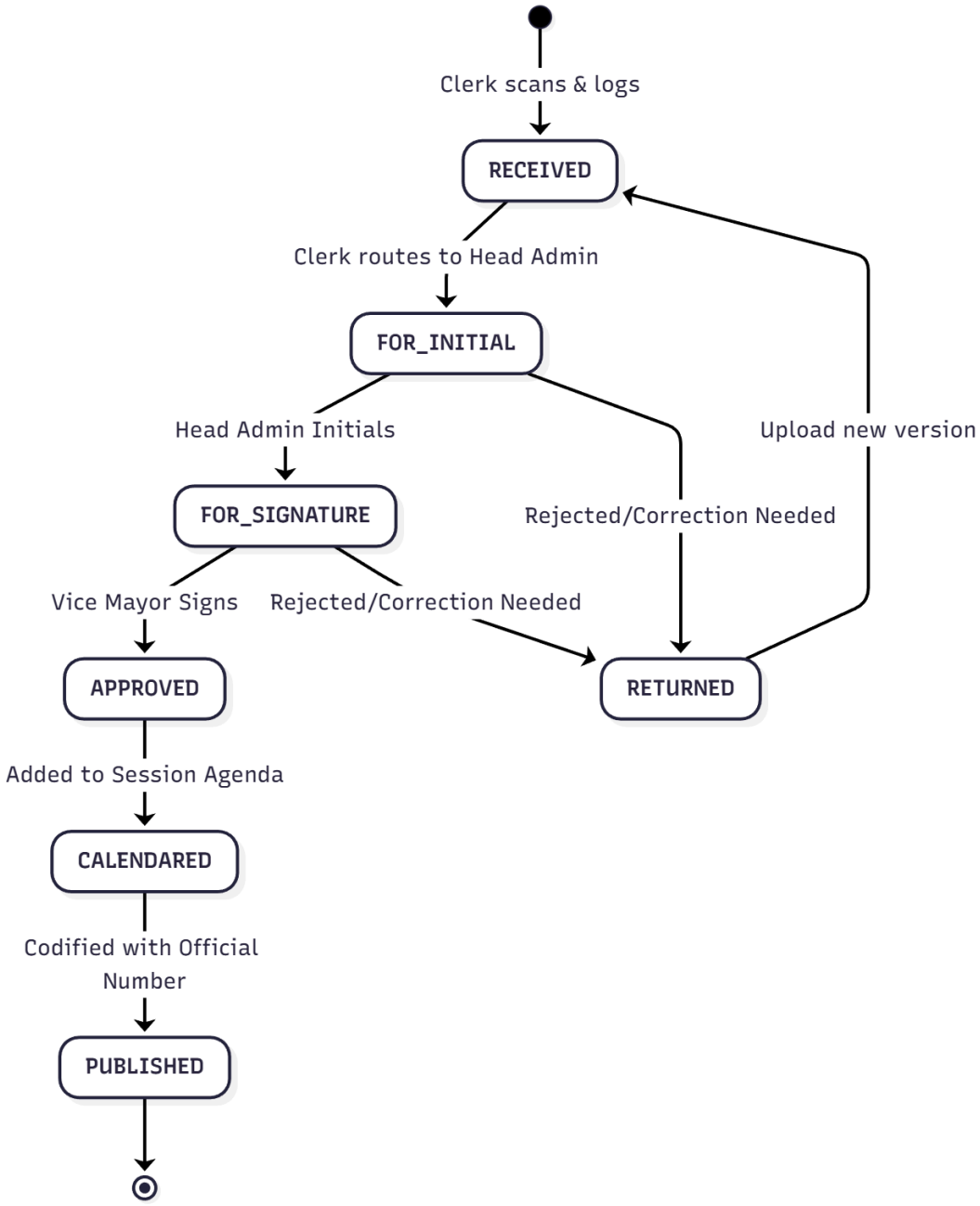


8.1.2 Domain Processes

This section defines the processes that control the lifecycle of key entities. The system enforces these transitions rigidly to ensure that a record does not skip steps.

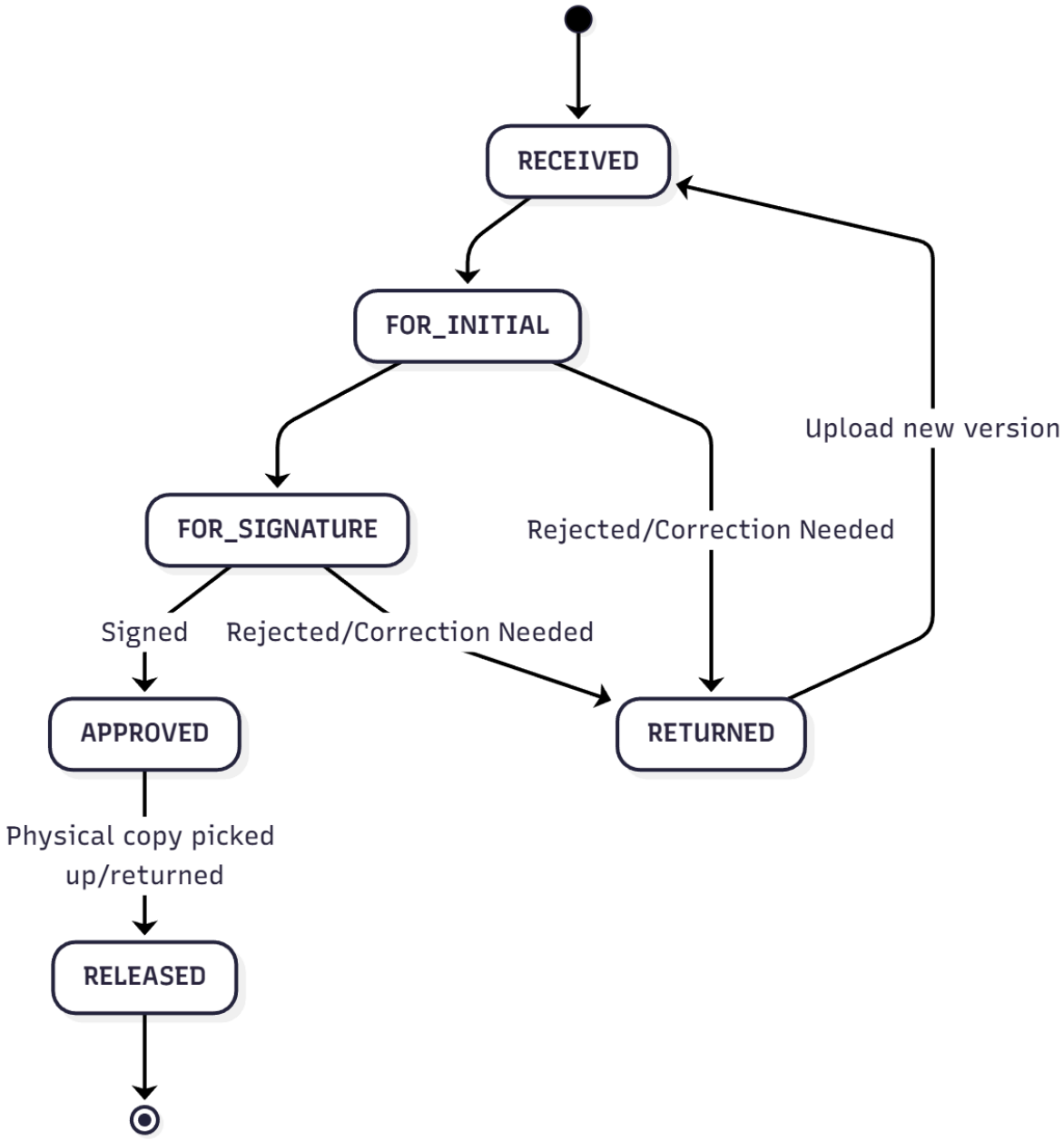
1. **Legislative Document Flow**

Applicable to: Proposed Ordinances, Proposed Resolutions, Committee Reports.



2. Administrative Document Flow

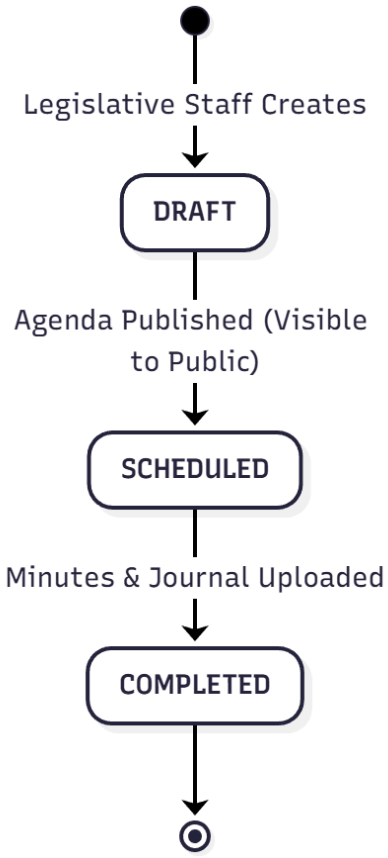
Applicable to: Payroll, Contracts, Memos, Endorsements.



3. Invitation & Caller Slip Flow

1. Invitations are logged as `RECEIVED`
2. Admin Staff groups pending invitations into a Caller's Slip.
3. The Vice Mayor reviews the Slip and assigns a decision to each invitation:
   - `ATTEND`
   - `DECLINE`
   - `ASSIGN_REPRESENTATIVE`

4. Session Status Flow

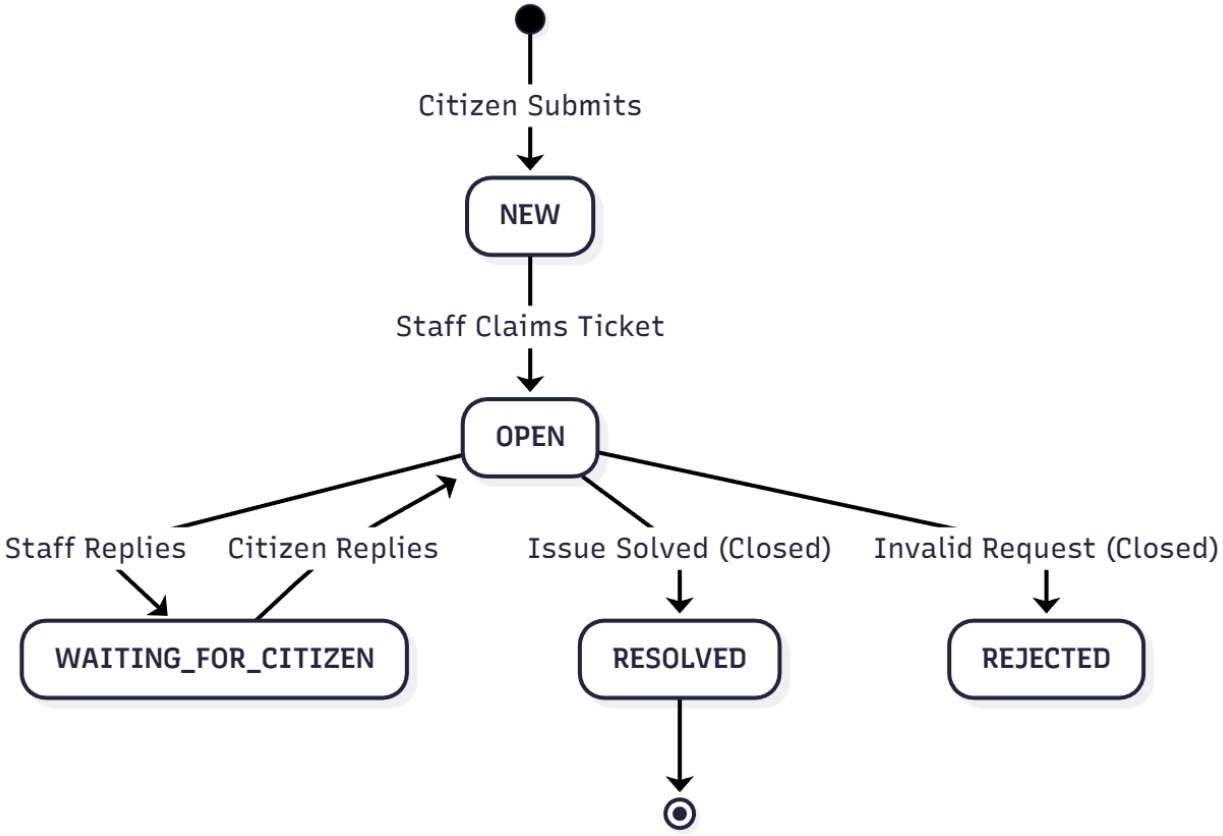


5. Conference Room Booking Flow

Direct Path (VICE_MAYOR, ADMIN_HEAD, ADMIN_STAFF, OVM_STAFF):
Request submitted `CONFIRMED`

Approval Path (COUNCILOR):
Request Submitted `PENDING` Head Admin/VM Reviews `CONFIRMED` or `REJECTED`.

6. Inquiry Ticket Flow



8.2 Security and Role-Based Access Control (RBAC)

8.2.1 Authentication

By delegating identity management to Supabase Auth, we reduce vulnerabilities associated with custom authentication.

- **Credential Storage**: Bcrypt, an industry standard, is used to hash passwords. Credentials are not kept in plain text.
- **Session Management**: JSON Web Tokens (JWTs) are used in authenticated sessions.
    - The JWT acts as the user's "Passport," carrying the immutable auth.uid()
    - Tokens have a limited duration and are automatically refreshed by the client's SDK.

8.2.2 Role-Based Access Control (RBAC)

The system follows a strict role-based access control (RBAC) model. Crucially, this is enforced using PostgreSQL Row-Level Security (RLS), which ensures that even if the API layer is breached, unauthorized queries are rejected.

Role Definition

Roles are mapped to the Business Elements (BE) defined in the requirements. Permissions are split between **View** (Select) and **Write** (Insert/Update/Delete).

Role (Code)	Business Element	Read Access (View)	Write Access (Edit/Create)
PUBLIC	BE-1 Public	Public Ordinances & Resolutions	Create Inquiries
		Session Schedules	Cannot edit submitted records
		Own Inquiries & Requests	
IT_ADMIN	BE-2 IT Admin	All System Data (Users, Config)	Manage User Roles (profiles)
		EXCLUDES Legislative Documents	System Configurations
			Full CRUD Access, except for document handling.
VICE_MAYOR	BE-3 Vice Mayor	All Internal Documents	Approve Documents (Status → Approved)
		Inquiry Tickets	Update Inquiry Ticket Status (Respond)
		Approval Queues	

OVM_STAFF	BE-4 VM Office Staff	All Citizen Inquiries	Update Inquiry Ticket Status (Respond)
		Visitor & Beneficiary Records	Update Beneficiary Records
		Caller Slips	
ADMIN_STAFF	BE-5 VM Admin Office Staff	Internal Documents	Create/Log Documents and caller slips
		Room Bookings	Request Initial Status
			Approve and Make Bookings
ADMIN_HEAD	BE-6 VM Admin Office Staff Head	All Internal Documents	Initial Documents (Status → Initialed)
			All other access is the same as BE-5
COUNCILOR	BE-7 City Councilor	Session Minutes & Documents	Request Room Bookings
		Approved Ordinances	
		Conference Room Availability	
LEGISLATIVE_STAFF	BE-8 Legislative Staff	Session files and information	Manage Sessions (Create/Edit)
			Build Agendas
			Upload Minutes & Journals

8.2.3 Abuse Prevention (Anti-Bot)

Since the portal accepts public inputs (Business Element BE1), safeguards against automation are strictly enforced within the application logic:

- **CAPTCHA Challenges**: Public submission endpoints (Public Inquiry Forms) require a CAPTCHA token validation to distinguish human users from automated scripts.
- **Input Sanitization**: Inputs are validated on the client-side for UX and strictly sanitized on the server-side to prevent injection attacks.

8.3 Deployment and Infrastructure

This section describes the cross-cutting concepts related to how the DALI Portal is deployed, hosted, and maintained across its cloud infrastructure. It covers the distribution of services, containerization, orchestration, and routing. These concepts influence the system's **Reliability**, **Security**, and **Maintainability**, ensuring stable operation for both public users and internal staff.

8.3.1 Cloud Based Architecture

The DALI portal is deployed using a distributed cloud setup composed of Cloudflare, DigitalOcean Droplet, Supabase, and Resend. This separation of responsibilities ensures that no single system becomes a bottleneck and contributes to higher reliability, security, and performance.

Cloud Service	Role/Responsibility
Cloudflare (Security Edge)	Provides WAF, DDoS protection, bot filtering, DNS, and HTTPS termination
DigitalOcean Droplet (Compute Runtime)	Hosts the application container using Coolify for deployment orchestration
Supabase (Database, Auth, Storage)	Offers managed PostgreSQL, secure file storage, authentication, and automated backups.
Resend (Email Service)	Send transactional emails

Overall benefit	Reduces single points of failure and increases system reliability.

8.3.2 Containerization and Orchestration

The DALI Portal runs inside Docker containers managed through Coolify. This modern approach ensures consistent environments, stable deployments, and simplified maintenance across development and production. Each major component runs in its own container, isolating failures and enabling independent scaling.

Container	Technology Framework	Role/Responsibility
Public Portal	Next.js	Frontend for citizens to access legislative documents and submit inquiries.
Internal Management System (IMS)	Next.js	Frontend for internal staff to manage documents, sessions, tickets, bookings, and beneficiary records.
Backend API	NestJS	Handles business logic, API endpoints, communication with Supabase (DB, Auth, Storage), and triggers emails via Resend.

Coolify Feature	Benefit
Docker Containers	A standardized environment ensures consistent behavior on all stages (dev → prod).

Coolify Deployment Orchestration	Handles automated builds, health checks, and container restarts.
Container Isolation	Ensures that failures in one component (e.g., API) do not impact others.
SSL management	Automatically provisions and renews SSL/TLS certificates for deployed services, ensuring secure HTTPS communication without manual configuration.
Overall benefit	Enhances maintainability and operational stability.

8.3.3 Reverse Proxy and Routing

Traefik acts as the system's routing layer, forwarding incoming requests from Cloudflare to the correct Docker container based on hostname. It also manages SSL certificates.

Traefik Function	Description
Routing	Directs traffic to Public Portal, IMS, or API containers based on domain.
Overall Benefit	Provides automated and secure request routing and uninterrupted service availability.

8.3.4 Network Security Layer

Multiple security layers protect the DALI system. These include Cloudflare protections, encrypted communication, private backend networking, and strict authorization policies at the database level.

Security Layer	Purpose
Cloudflare WAF & DDoS Protection	Blocks malicious traffic before it reaches the VPS.

End-to-End HTTPS Encryption	Protects all communication between client and server.
Private Networking	Restricts internal communication between the API and database to non-public networks.
Supabase Row-Level Security (RLS)	Enforces fine-grained access control, ensuring users only access data they are authorized to view.
JWT-Based Access Control	Validates identity and permissions for each request using signed JSON Web Tokens.
Overall Benefit	Strengthens system security, ensures compliance with the Data Privacy Act, and protects sensitive government data and citizen data.

8.3.5 Email Service Integration

The system uses Resend as its transactional email provider. It is a cloud-based service responsible for delivering automated emails.

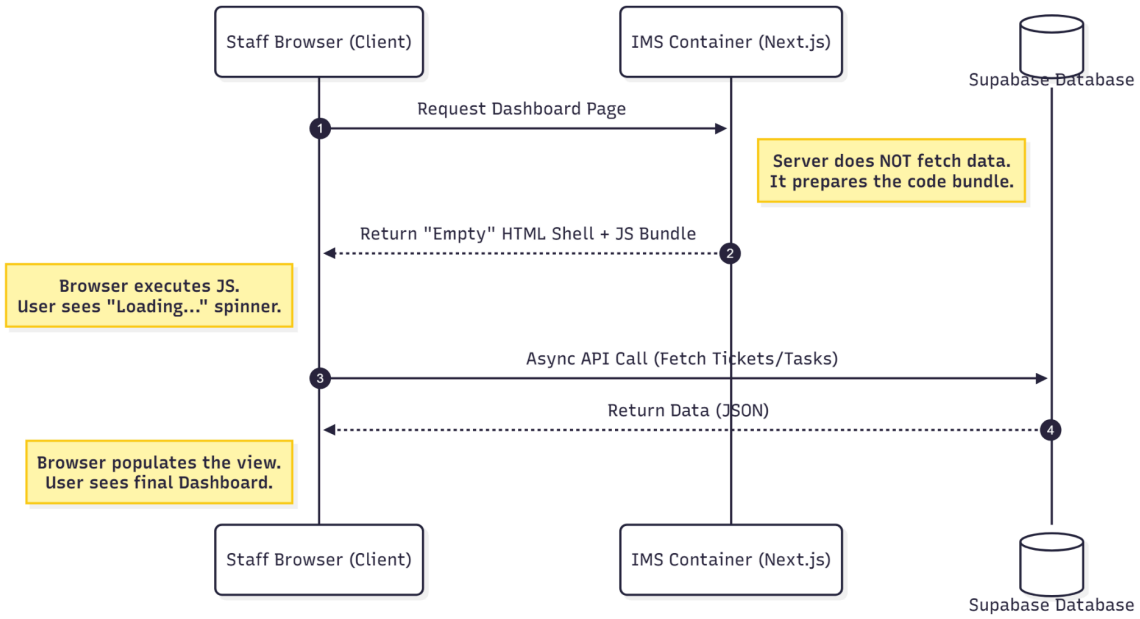
Resend Function	Description
Transactional Email Delivery	Sends reliable and timely email notifications triggered by system events.
High Deliverability	Ensures emails reach inboxes by using domain authentication and reputable sending infrastructure.
API Integration	The API integrates seamlessly with the NestJS backend for templated, automated messaging.
Overall benefit	Provides a reliable and developer-friendly email delivery service, assuring system communication reach without delay.

8.4 Frontend Design Patterns

This section describes the architectural patterns used to render the user interface and manage data retrieval. The DALI Portal utilizes a Hybrid Rendering Strategy, employing specific patterns best suited for the distinct requirements of the public citizens versus the internal staff.

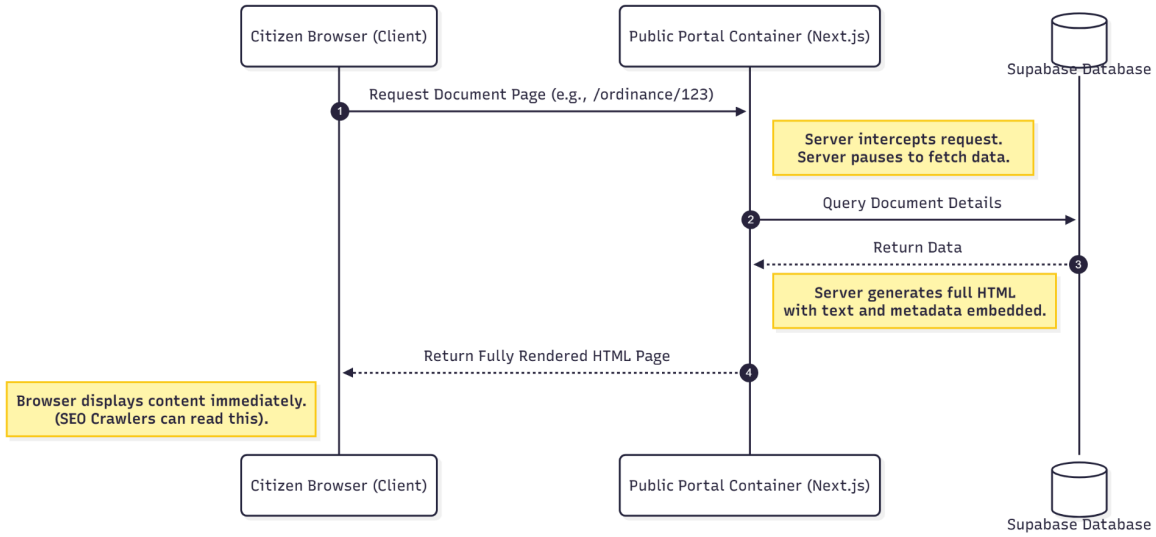
8.4.1 Client Side Rendering

Client-Side Rendering is a technique where the rendering of the content happens in the user's browser rather than on the web server. The server sends a minimal HTML container and JavaScript bundles. The browser then executes the script to fetch data and update the interface dynamically.



8.4.2 Server Side Rendering

Server-Side Rendering is a technique where the full HTML page is generated on the server before being sent to the client. When a request is made, the server executes the logic, fetches the necessary data, populates the template, and delivers a complete page.



8.5 User Interface / User Experience

The user experience (UX) describes the overall feeling and efficiency a user achieves while interacting with the system. For the Sangguniang Panlungsod ng Iloilo portal, the UX is constructed to be trustworthy, highly efficient, and transparent, ensuring public access to legislative data is frictionless.

8.5.1 Theme

Branding and Color Scheme

The prominent, deep reddish-brown color scheme seen in the header is a direct result of a client mandate rooted in their institutional branding. This is intended to enforce consistency, authority, and immediate recognition across all public platforms. The prominent placement of the Iloilo City Council's official seal and name at the top left of the navigation bar immediately establishes the website's authority and institutional identity. This branding is crucial for user experience as it builds instant trust and recognition that the site is legitimate and official.

Tailwind CSS

Usage of libraries such as Tailwind CSS enhances user experience by enforcing a consistent design for spacing and styling, and its streamlined production output ensures faster page load times, which is crucial for maximizing user engagement.

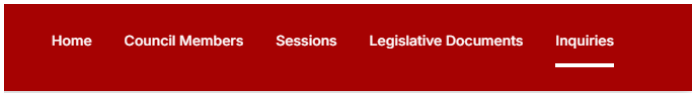
8.5.2 Design System

Primary Navigation

This uses a standard, familiar horizontal navigation pattern, making it easy for users to scan and predict where they need to go. The top horizontal menu is the most important guide for the user experience because it tells people exactly where they can go on the site. The labels are straightforward.

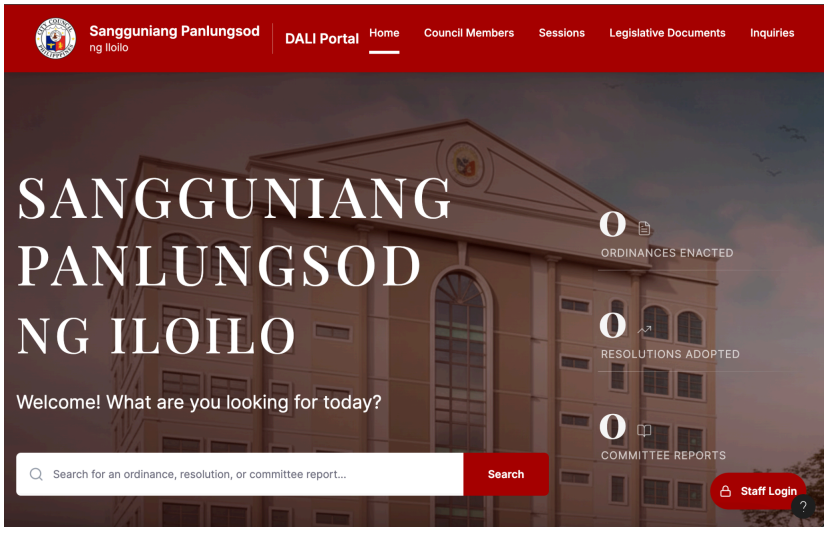
Active State Indicator

The line under the clicked word in the menu is a helpful signal that instantly shows the user which page they are on. This simple visual cue prevents confusion, making it much easier for people to know where they are and quickly navigate the website.



Direct Search Access

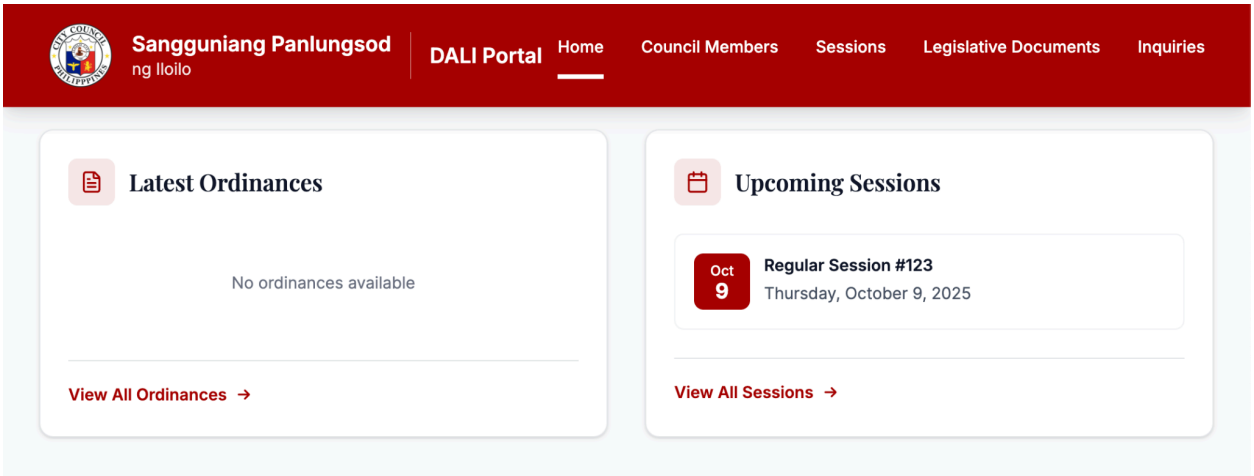
The decision to place a prominent search bar on the front page is a powerful design choice that prioritizes findability, enabling people to immediately locate specific documents or topics. This direct access significantly reduces the effort required to get answers, accelerating their journey to desired content and improving their satisfaction.



Home Page Dashboard

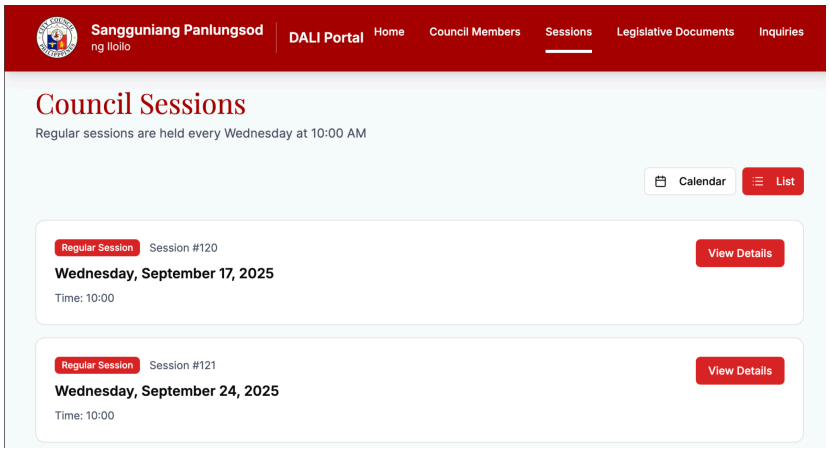
The Home page is intentionally designed as a dashboard, displaying summary cards for Latest Ordinances and Upcoming Sessions immediately upon scrolling. This layout is a

key usability feature because it places the most critical, time-sensitive information directly in the user's view.



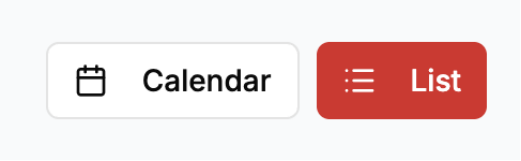
Sessions Recency

Listing the Council Sessions by event date, starting with the most current or nearest session, is key to effective information architecture for time-sensitive data. This approach prioritizes the user's need for up-to-date scheduling, maximizing efficiency and ensuring they quickly access the latest, most relevant information without needing to scroll through history.



Flexible View Options

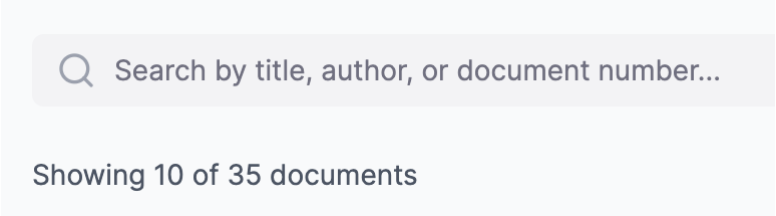
The ability to switch between List and Calendar views is essential because it lets users choose the visual format that best matches their task and thinking style. By offering both options, the system ensures that every user can consume



the session schedule data most efficiently and comfortably, significantly boosting usability and overall satisfaction.

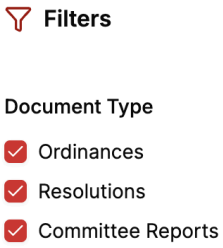
Prominent Search Bar

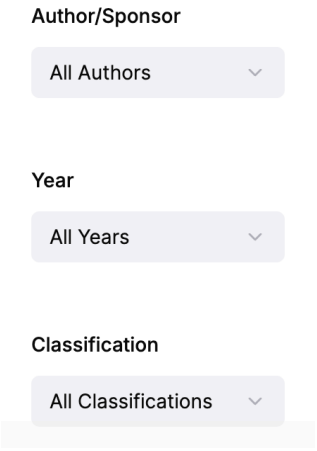
A large, central search bar allows users to perform free-text searches (by title, author, or document number), directly supporting the core task of findability.



Check Box Filters (Document Type):

Users can quickly narrow down the document list by type (Ordinances, Resolutions, Committee Reports). This multi-select filtering is crucial for efficiency, as users often know the category of the document they are looking for.



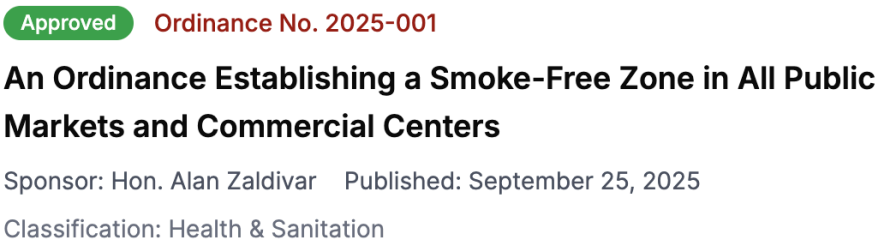


Dropdown Filters (Author/Sponsor, Year):

These controls allow for structured, hierarchical filtering. This addresses the need to browse by specific criteria, which is vital in a large, official document database.

At-a-Glance Detail

Each document entry is concise, displaying the title, document number (e.g., *Ordinance No. 2025-001*), sponsor/author, and classification. This provides enough detail for users to judge relevance without having to click "View Details."

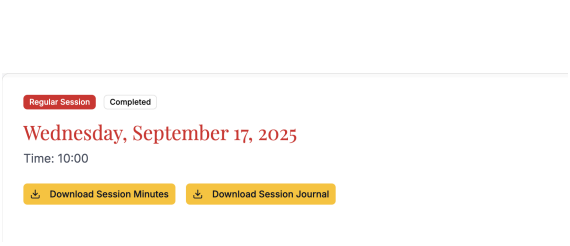


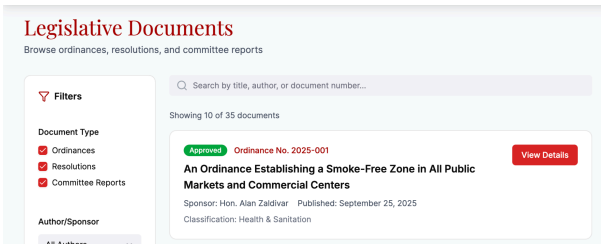
High Contrast Action Buttons (View Details)

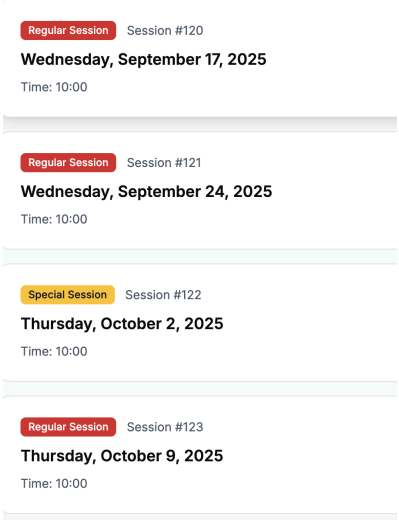
The "View Details" for both the sessions and legislative documents is placed inside a bright red button that stands out clearly on the page. This is important because the button acts as a big, obvious target, making it easy for the person to know exactly where to click to get the full information about the document.

High Contrast Action Buttons (Download)

The download buttons are designed to be immediately actionable and clear, with distinct yellow/gold color, which contrasts strongly with the white background and the red tags. This high contrast makes them stand out as the primary action points on the page, ensuring users quickly find the document links.







Session Type Badges

The use of distinct color-coded "badges" for session types is a powerful visual cue that greatly enhances the clarity and rapid scanning of information. The consistent use of red for "Regular Session" and yellow for "Special Session" immediately emphasizes the difference between them. This color coding lets users quickly recognize the status or type of meeting without needing to read the text, prioritizing key information and improving the speed at which they can process the list.

9. Architectural Decisions

This chapter contains the decisions that are important to the architecture of the system including rationales.

9.1 Next.js for Public Portal and Internal System

Section	Description
Title	ADR 1: Next.js for Public Portal and Internal System
Status	Accepted
Context	The DALI Portal requires two distinct interfaces: a Public Portal that needs high performance and Search Engine Optimization (SEO) for citizens, and an Internal Management System that requires rich interactivity for staff. Using different frameworks for each would fragment the codebase and increase the learning curve for the development team.
Decision	**Next.js 16 (App Router)** will be used for developing both the public and internal frontend applications. This unified

	framework allows the team to leverage **Server-Side Rendering (SSR)** for the public portal's SEO needs and **Client-Side Rendering (CSR)** for the internal dashboard's interactivity, while sharing common UI components.
Consequences	While this unifies the tech stack, it introduces complexity regarding the "Client vs. Server Component" boundary. Developers must be disciplined in marking components with `"use client"` only when necessary to avoid bloating the client-side bundle size.

9.2 NestJS for Backend API

Section	Description
Title	ADR 2: NestJS for Backend API
Status	Accepted
Context	The system handles complex business logic, such as validating document workflows, detecting booking conflicts, and generating caller slips. Implementing this logic directly within Next.js API routes or a standard Express.js server can lead to disorganized code that is difficult to test and maintain as the system grows.
Decision	**NestJS** will be used as the dedicated backend API framework. Its opinionated structure (Controllers, Services, Modules) and strict TypeScript integration enforce a clear separation of concerns, ensuring that business logic is isolated from the **HTTP** transport layer.
Consequences	NestJS requires more boilerplate code compared to simpler frameworks. However, this trade-off results in a more robust, testable, and scalable backend architecture suitable for government workflows.

9.3 Supabase for Authentication, Database, and Storage Provider

Section	Description
Title	ADR 3: Supabase for Authentication, Database and Storage
Status	Accepted
Context	The DALI Portal needs a secure, relational database to manage document lifecycles and citizen inquiries, an authentication system to handle strict **Role-Based Access Control (RBAC)** for government employees, and an object storage solution that can store multiple PDF legislative documents. Creating these three independent services from scratch would take a long time and raise the risk of security issues.
Decision	Supabase will be used to handle user authentication, database management and media storage.
Consequences	System availability is partially dependent on Supabase uptime

9.4 Resend for Transactional Email

Section	Description
Title	ADR 4: Resend for Transactional Email
Status	Accepted
Context	The DALI Portal allows citizens to submit concerns or inquiries regarding city services. Once these inquiries are reviewed and processed by internal staff, citizens need to be notified about the final decision, whether their inquiry has been resolved or rejected. This ensures transparency and provides closure to the inquiry process. Sending these notifications manually is inefficient, error-prone, and may delay citizen updates.
Decision	The system will use Resend as its transactional email provider

	to send automated notifications only after an inquiry has been resolved or rejected. Integration with the NestJS backend will trigger an email containing the final remarks or instructions from staff.
Consequences	Citizens receive a reliable notification about the outcome of their inquiry, ensuring transparency and trust. The system reduces manual workload and minimizes the potential for human error in sending notifications. Notifications are dependent on Resend's uptime, so any downtime could delay delivery. The inquiry thread is locked after closure, and the notification is sent only at the conclusion of the inquiry, not upon initial submission.

9.5 Virtual Private Server(VPS) over Serverless hosting

Section	Description
Title	ADR 5: Virtual Private Server (VPS) over Serverless hosting
Status	Accepted
Context	The DALI Portal handles sensitive government data, including citizen information and legislative documents. Serverless hosting platforms often abstract away network and infrastructure details, limiting direct control over security configurations, persistent storage, and custom networking.
Decision	A VPS will be used instead of serverless hosting. This allows the team to configure private networking, firewall rules, and reverse proxies (Traefik) to enforce a secure architecture. Persistent storage and container orchestration can also be handled more predictably.
Consequences	The system provides greater control over security, networking, and server configuration, but the team must manage server maintenance, patching, scaling, and monitoring themselves, resulting in slightly higher operational overhead compared to serverless alternatives.

9.6 Using Digital Ocean Droplet for VPS

Section	Description
Title	ADR 6: Using Digital Ocean Droplet for VPS
Status	Accepted
Context	A VPS is required for hosting the DALI system's containers and services. Among available VPS providers, the choice must balance performance, cost, scalability, and integration with container orchestration tools like Coolify.
Decision	DigitalOcean Droplets will be used as the VPS provider. DigitalOcean offers reliable virtual machines with predictable pricing, straightforward networking, and native support for private networks, which aligns with the security and operational requirements of the DALI system.
Consequences	Using DigitalOcean simplifies provisioning and scaling of VPS instances for staging and production environments. The provider's uptime SLA ensures high availability. However, the team remains responsible for managing OS updates, backups, and monitoring of the Droplets.

9.7 Coolify as VPS Management Platform

Section	Description
Title	ADR 7: Coolify as VPS Management Platform
Status	Accepted
Context	Managing multiple Docker containers, SSL certificates, automated builds, and health checks manually on a VPS introduces operational complexity. A container management and orchestration platform can simplify deployment, monitoring, and scaling.
Decision	Coolify will be used as the VPS management platform. Coolify handles container builds, restarts, health checks, TLS

	certificate provisioning, and environment isolation. It provides a developer-friendly interface to manage multiple services efficiently.
Consequences	Using Coolify reduces operational overhead by automating builds, restarts, and certificate management. It ensures container isolation, improving system stability and fault tolerance. However, the system depends on Coolify features and updates, and any limitations in the platform may affect deployment workflows.

9.8 Server Side Rendering for Public Portal

Section	Description
Title	ADR 8: Server Side Rendering for Public Portal
Status	Accepted
Context	The Public Portal allows citizens to view Ordinances and Resolutions. For transparency, these documents must be discoverable via search engines like Google. Additionally, many citizens access the site using mobile data with potentially unstable connections, where waiting for client-side JavaScript to load before seeing content results in a poor user experience.
Decision	The team will strictly utilize **Server-Side Rendering (SSR)** for the Public Portal pages. The Next.js server will fetch document data from the database and generate the complete HTML on the server before sending it to the user's browser.
Consequences	This ensures "First Contentful Paint" is immediate, and search engine crawlers can index legislative text effectively. But, this increases the CPU load on the hosting server, as it must generate a new HTML page for every request instead of serving a static file.

9.9 Client Side Rendering for Internal Management System

Section	Description
Title	ADR 9: Client Side Rendering for Internal Management System
Status	Accepted
Context	The IMS acts as a daily workstation for staff. It requires high interactivity, such as sorting complex tables, opening modals, and modifying records. Since this system is behind a login wall, SEO is not a requirement. Reloading the entire page for every interaction would be slow and disruptive to the staff's workflow.
Decision	The team will utilize Client-Side Rendering (CSR) for the IMS. By using the `"use client"` directive in Next.js, the browser will handle the UI rendering and state management, fetching data asynchronously via APIs only when needed.
Consequences	The application behaves like a responsive desktop application with instant transitions between views, improving staff productivity. But, The initial load time is slightly longer as the browser must download the JavaScript bundles before the dashboard becomes interactive.

9.10 Cloudflare as Security Edge Layer

Section	Description
Title	ADR 10: Cloudflare as Security Edge Layer
Status	Accepted
Context	The DALI Portal handles sensitive citizen data and must be protected against DDoS attacks, malicious traffic, and other web-based threats. Directly exposing the VPS to the public internet increases security risks.
Decision	Cloudflare will act as the security edge layer for the system, providing DDoS protection, Web Application Firewall (WAF),

	SSL termination, and traffic filtering. All incoming requests will first pass through Cloudflare before reaching the VPS.
Consequences	Cloudflare reduces the risk of malicious traffic reaching the VPS and improves overall system resilience. SSL/TLS management is simplified, as certificates can be managed at the edge. However, reliance on Cloudflare introduces partial dependency on a third-party service for availability and security enforcement.

9.11 Zustand as State Management for Internal Management System

Section	Description
Title	ADR 11: Zustand as State Management for Internal Management System
Status	Accepted
Context	The IMS requires efficient state management to handle dynamic data such as legislative document lists, session schedules, inquiry tickets, bookings, and other user interactions. Proper state management ensures that updates to documents, tickets, or session data are reflected consistently across all components without unnecessary re-renders or stale information.
Decision	The IMS frontend will use Zustand for state management. Zustand provides a lightweight, reactive, and scalable solution, allowing components to subscribe to relevant slices of state without unnecessary re-renders.
Consequences	State management is simplified, resulting in better performance and maintainability. Developers can easily share and update state across components. However, misuse of global state could lead to difficult-to-debug side effects if not managed carefully.

9.12 Client-Side Uploading of Files to Supabase Storage

Section	Description
Title	ADR 12: Client-Side Uploading of Files to Supabase Storage
Status	Accepted
Context	Legislative documents, inquiry attachments, and other media files can be large, and uploading them through the VPS backend could strain server RAM and bandwidth.
Decision	Files will be uploaded directly from the client to Supabase storage using signed URLs or client SDKs. The backend will only handle metadata and access control, while the file transfer bypasses the VPS to save resources.
Consequences	Client-side uploads reduce server memory usage and improve scalability. Upload speed is improved for large files, as the transfer does not need to route through the backend. However, proper validation and security checks must still be implemented to prevent unauthorized uploads or malicious files.

9.13 Shadcn for UI components

Section	Description
Title	ADR 13: Shadcn for UI components
Status	Accepted
Context	Both the Public Portal and IMS require consistent, reusable, and accessible UI components to improve development speed and maintain visual consistency across pages.
Decision	The system will use Shadcn for UI components. Shadcn provides pre-built, accessible, and themable components compatible with Next.js and Tailwind CSS, allowing rapid development of forms, buttons, modals, and data tables.

Consequences	UI development is accelerated, and design consistency is maintained across applications. Components are accessible by default, improving usability and compliance with accessibility standards. However, updates to the library could require refactoring if breaking changes are introduced.

12. Glossary

This chapter defines important terms or jargons used in the system that might be unfamiliar to some stakeholders. Terms here are listed and defined in much more understandable words.

Term	Definition
Agenda	A structured list or outline of topics, discussions, and items to be addressed during a legislative session.
API	Application Program Interface. A set of rules and tools that allows different software applications to communicate with each other, enabling data exchange and functionality integration.
arc42	A template for documenting software and system architecture. This document follows the arc42 structure.
Bcrypt	A password-hashing function used by Supabase Auth to ensure user passwords are not stored in plain text.
Blackbox	An architectural view where the internal workings of a component are hidden, focusing only on its inputs, outputs, and responsibilities.
Caller's Slip	A summary document (digest) generated for the Vice Mayor containing a list of incoming invitations and requests for decision-making.
CAPTCHA	A challenge-response security measure used within the application to reliably determine whether the user interacting with a form or function is a human or an automated program (bot).
Cloudflare	The service acts as the Security Edge for the system. It provides DNS resolution, DDoS protection, and Web Application Firewall (WAF) features.
Coolify	A self-hostable platform used on the DigitalOcean VPS to orchestrate and manage the Docker containers and deployment pipelines.

CPU	The Central Processing Unit - responsible for interpreting and executing the instructions contained in computer programs.
CRM	Customer Relationship Management. A system or software used to manage interactions, communications, and data related to clients or citizens, often improving service delivery and record-keeping.
CSR	A web application rendering technique where the server sends a minimal HTML shell and all of the application's core logic and rendering is performed directly by the browser using JavaScript.
DALI	Digital Access for Legislative Information. The name of the portal being developed.
Data Privacy Act of 2012	The Data Privacy Act of 2012 (DPA), officially known as Republic Act No. 10173, is the primary law in the Philippines that protects individual personal data in the digital age. It establishes a comprehensive legal framework to regulate the collection, handling, and disposal of personal information in both the government and private sectors.
DDoS	Distributed Denial of Service. A malicious attempt to disrupt normal traffic of the server. Cloudflare is used to mitigate this.
DigitalOcean	A cloud infrastructure provider that offers cloud services to deploy, manage, and scale applications. It provides virtual machines, databases, storage, and networking solutions for developers.
DigitalOcean Droplet	The virtual private server (VPS) instance (Linux/Ubuntu) located in Singapore that hosts the application runtime.
Docker	An open-source platform that allows programmers to combine an application and all of its dependencies, libraries, and configuration files into a single, standardized unit known as a container.

Entity Relationship DIagram (ERD)	A visual representation of the system's database structure that illustrates how data entities relate to one another, used to design the underlying database schema.
Express.js	A minimal and flexible Node.js web application framework that provides tools for building APIs, web servers, and server-side applications efficiently.
HTML	The standard markup language used to create the structure and content of documents displayed in a web browser. It defines the text, headings, links, images, and other elements using a system of tags.
HTTP	Hypertext Transfer Protocol. The underlying protocol used by the World Wide Web that defines how messages are formatted and transmitted between the client (browser) and the server.
HTTPS	HyperText Transfer Protocol Secure - an extension of the Hypertext Transfer Protocol (HTTP) used for secure communication over a computer network. HTTPS ensures authenticated and encrypted communication between a client (typically a web browser) and a server.
Hydration	The process where the client-side JavaScript converts the static HTML served by the server (Next.js) into an interactive application in the browser.
IMS	Internal Management System. The secure, password-protected side of the application used by the Vice Mayor, Councilors, and Staff.
Initialed	The status of a document after it has been reviewed and approved by the Head Admin Staff, making it eligible for the Vice Mayor's signature.
Inquiry Ticket	A record created in the system when a citizen submits a form via the Public Portal. It tracks the status of the request (e.g., New, In Progress, Resolved).

Journal	A narrative record of the proceedings of a legislative session.
JWT	JSON Web Token. A compact, URL-safe means of representing claims to be transferred between two parties. Used here for secure, authenticated user sessions.
Metadata	Information that describes and provides context about other data, making it easier to find, understand, organize, or manage.
Minutes	The official written record of the discussions and decisions made during a legislative session.
NestJS	The framework used for the Backend API container. It handles business logic that requires server-side processing.
Next.js	The React framework used for both the Public Portal and Internal Management System frontends, supporting Server-Side Rendering (SSR).
Ordinance	A piece of legislation enacted by a municipal authority. These are stored as PDFs in the system.
OS	The operating system manages computer hardware and software resources and provides common services for computer programs.
OVM	Office of the Vice Mayor.
PostgreSQL	A powerful, open-source Object-Relational Database Management System (ORDBMS) known for its robustness, reliability, data integrity, and feature set.
RAM	Random Access Memory - a form of computer data storage that is highly fast and directly accessible to the CPU.
RBAC	Role-Based Access Control. A method of restricting network access based on the roles of individual users within the enterprise (e.g., IT Admin, Staff, Vice Mayor).
Resend	The third-party transactional email API used to send

	automated notifications (e.g., ticket tracking numbers, booking confirmations).
Resolution	A formal expression of opinion, will, or intent voted on by the City Council.
Responsive Web Design	A web design approach that ensures websites and applications adapt smoothly to different screen sizes and devices, providing an optimal viewing experience on desktops, tablets, and mobile phones.
RLS	Row-Level Security. A security feature in PostgreSQL (Supabase) that restricts data access for specific rows in a table based on the user's authenticated role.
SEO	The practice of optimizing web content to improve its visibility and ranking on search engines like Google, increasing organic traffic.
Session	A formal meeting or period during which the Vice Mayor presides over legislative activities, including discussions, approvals, and other official functions.
SLA	A formal contract between a service provider and a client that defines the expected level of service, performance metrics, and responsibilities.
Supabase	A Backend-as-a-Service (BaaS) provider used for the PostgreSQL Database, Authentication, and File Storage.
SDK	Software Development Kit. A collection of tools, libraries, documentation, and sample code that helps developers build applications for a specific platform or service.
SRP	Single Responsibility Principle. A class or module should have only one reason to change, promoting maintainable code.
SSL	Secure Sockets Layer. A cryptographic protocol that encrypts data transmitted over the Internet. While mostly replaced by TLS, the term SSL is still commonly used.

SSR	Server-Side Rendering. A technique in web development where web pages are rendered on the server and sent fully formed to the client browser, improving performance and SEO.
Tailwind CSS	A utility-first CSS framework that allows developers to build custom user interfaces quickly using pre-defined classes in HTML.
Ticket Reference Number	A unique identifier assigned to a support request or inquiry ticket, allowing users and administrators to track its status.
TLS	Transport Layer Security. The modern, secure protocol that encrypts communications over a network, effectively replacing SSL.
Traefik	A modern HTTP reverse proxy and load balancer used to route internal traffic to the correct Docker containers based on the subdomain.
UI	User Interface. Everything the user interacts with or sees on the screen.
URL	Uniform Resource Locator. The fundamental address system used to reference web pages, images, files, and services across the Internet.
VM	Vice Mayor. Refers to the Vice Mayor of Iloilo City.
VPS	Virtual Private Server. A virtual machine sold as a service by an Internet hosting service.
WAF	Web Application Firewall. A security shield provided by Cloudflare that filters and monitors HTTP traffic between the web application and the Internet.
Whitebox	An architectural view that shows the internal structure and logic of a specific system component.
Zustand	A lightweight state management library for React applications, used to manage and share application state outside of component hierarchies.