E GOVERNMENT READDRESSAL SYSTEM

1. Purpose of the document

This document describes the system architecture, design decisions, component structure, APIs, data models, and non-functional aspects of the E-Government Grievance Redressal System

It is intended for:

- Developers
- Reviewers
- Interview discussions
- Maintenance & enhancement planning

---

2. System Overview

2.1 Business Objective:

Provide a **scalable, secure, and maintainable E-Governance platform** to:

- Enable citizens to register and track grievances digitally
- Ensure accountability by assigning grievances to relevant departments
- Monitor grievance lifecycle and resolution timelines
- Improve transparency through centralized tracking and notifications
- Support future growth using a microservices-based architecture

2.2 High-Level Features:

- Citizen registration and secure login using JWT authentication
- Online grievance lodging with description and document/image upload
- Centralized grievance tracking with unique reference ID (PNR-like)
- Grievance lifecycle management: *Submitted* → *Dept_Review* → *Assigned* → *In_Progress* → *Work_Done* → *Resolved* → *Reopened* → *Escalated* → *Closed*
- Automatic or manual department assignment for accountability
- Role-based dashboards and access control (Citizen / Officer / Supervisor / Admin/ Case Worker)
- Status change notifications via notification service
- Escalation of grievances if not resolved within defined time limits (SLAs)
- Feedback submission by citizens after grievance resolution
- Reports and analytics for department-wise and category-wise performance

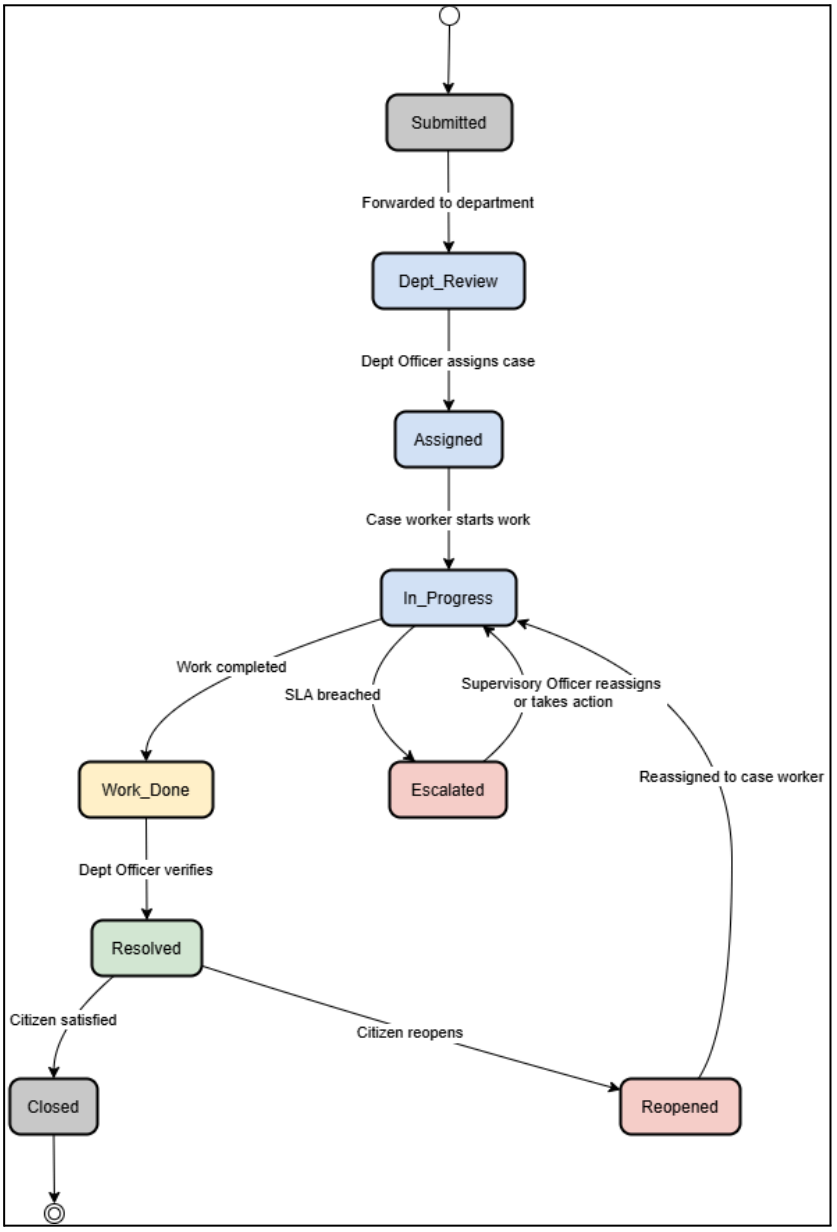
2.3 Roles & Access Control:

- **Citizen** – Register, submit grievances, track status, and provide feedback
- **Department Officer** – View and resolve assigned grievances
- **Supervisory Officer** – Monitor grievance status and escalations
- **System Admin** – Manage users, departments, roles, and grievance categories
- **Case Worker** - Multiple per department, handled by DO.

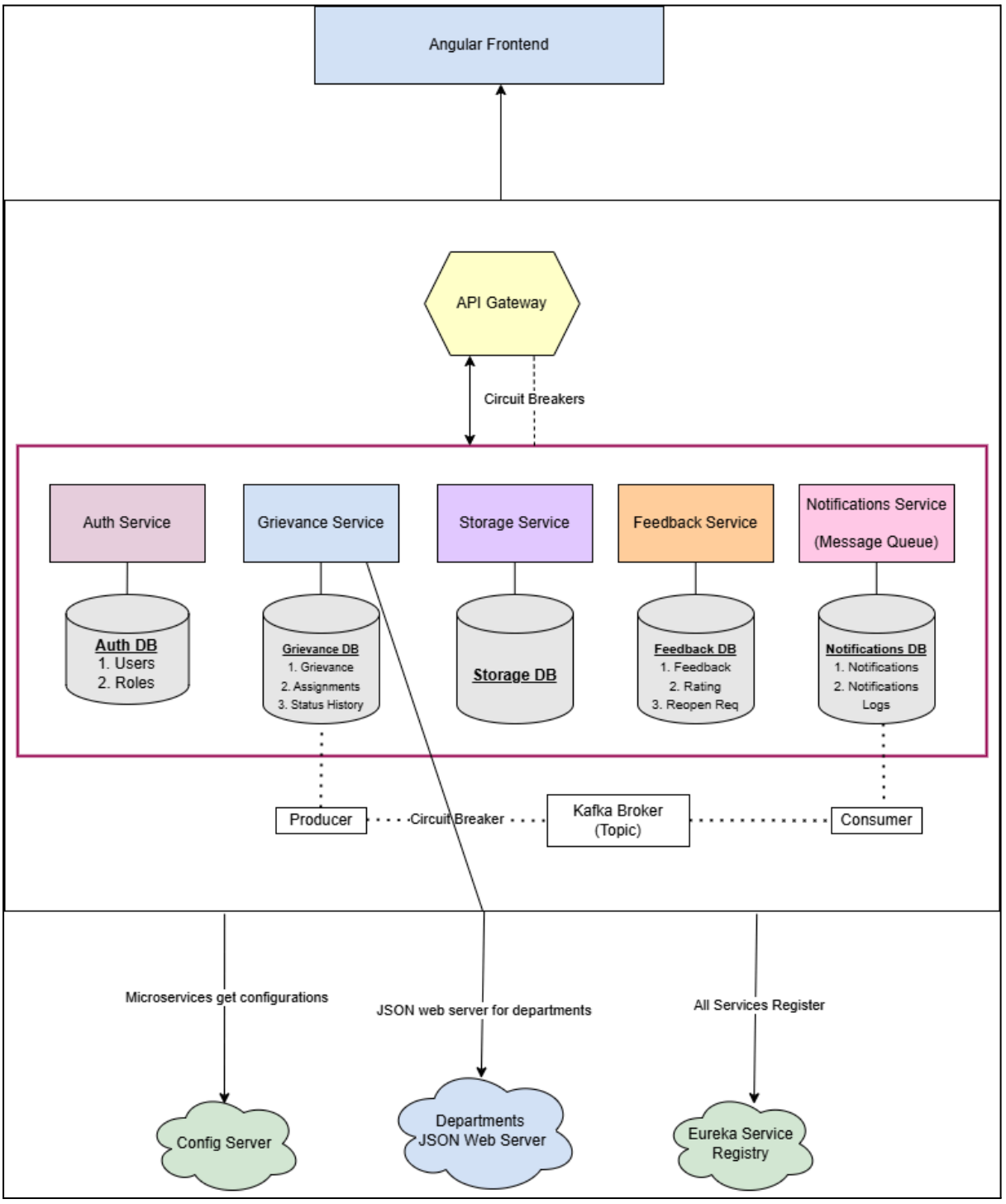
2.4 Business Rules:

- Only authenticated users with valid roles can access system features, enforced through JWT-based role authorization.
- Citizens are the only users permitted to lodge grievances, and each grievance must be associated with exactly one category and department.
- Every grievance is assigned a unique system-generated grievance ID and timestamp at the time of submission.

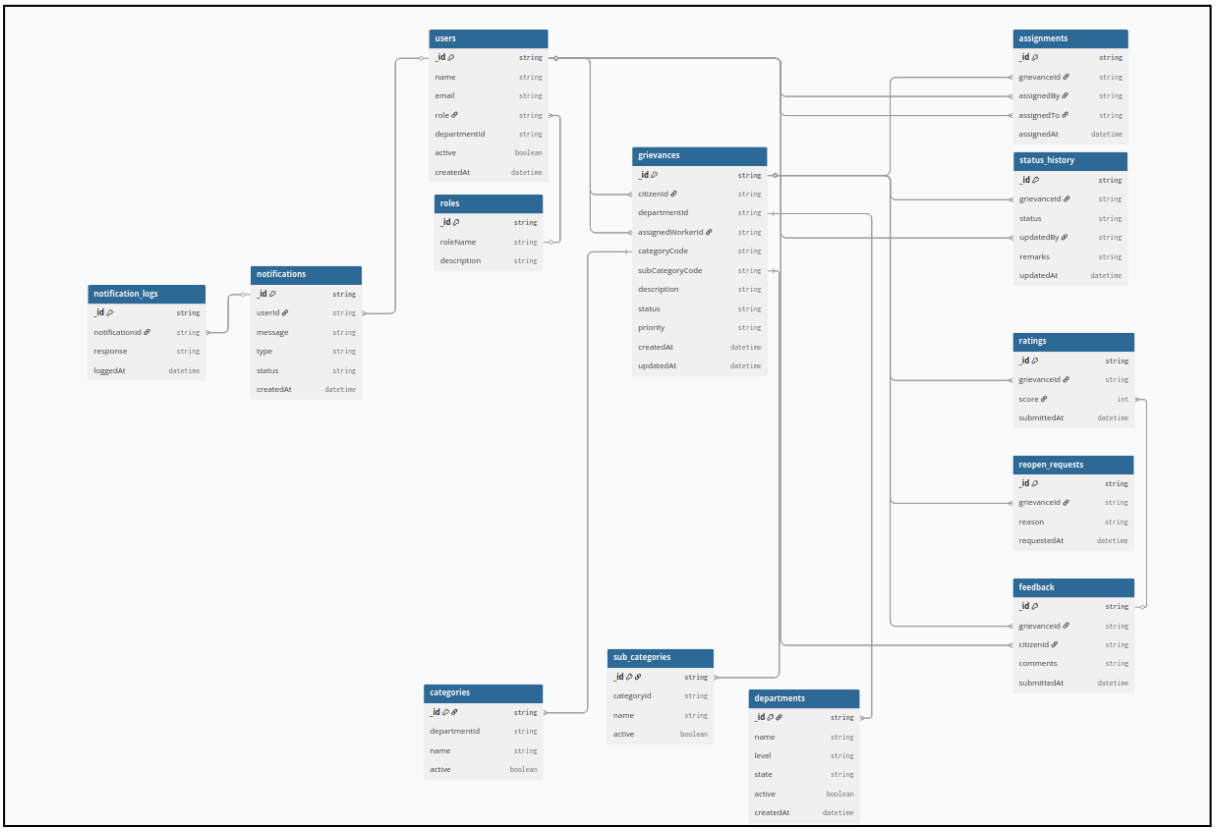
- Grievances follow a strictly enforced lifecycle:
  Submitted → Assigned → In Review → Resolved → Closed,
  with no status skipping allowed.
- Grievances are automatically routed to the appropriate department based on category-to-department mapping.
- Only authorized department officers can update grievance status and submit resolution remarks.
  A grievance is eligible for escalation if it remains unresolved beyond a predefined resolution timeframe.
- Citizens may submit feedback and ratings only after a grievance has been resolved, with limited scope for reopening.
- Administrative and supervisory users have read-only access to analytics and performance reports derived from grievance data.
  All grievance actions, status changes, and resolutions are logged with timestamps to ensure transparency, traceability, and auditability.



3. Architecture Diagram



4. DB Schema for each microservice



Auth Service

Users - Collection

Attribute	Type	Description
`_id`	String	User ID
`name`	String	Full name
`email`	String	Login email
`role`	Enum	CITIZEN / DO / CASE_WORKER / SO / ADMIN
`departmentId`	String	Nullable (for DO / CW)
`active`	Boolean	Enabled/disabled
`createdAt`	DateTime	Created timestamp

Roles - Collection

Attribute	Type	Description
_id	String	Role ID
roleName	String	Role name
description	String	Role description

Grievance Service

grievances

Attribute	Type	Description
_id	String	Grievance ID
citizenId	String	Citizen user ID
departmentId	String	Assigned department
assignedWorkerId	String	Case Worker ID
categoryCode	String	WATER, ELECTRICITY
subCategoryCode	String	SUPPLY_DELAY, CORRUPTION
description	String	Complaint text
status	Enum	GrievanceStatus
priority	String	LOW / MEDIUM / HIGH
createdAt	DateTime	Creation time
updatedAt	DateTime	Last update

assignments

Attribute	Type	Description
_id	String	Assignment ID
grievanceId	String	Grievance reference

assignedBy	String	DO ID
assignedTo	String	Case Worker ID
assignedAt	DateTime	Assignment time

status history

Attribute	Type	Description
_id	String	History record ID
grievanceId	String	Grievance reference
status	Enum	Status value
updatedBy	String	User ID
remarks	String	Optional note
updatedAt	DateTime	Timestamp

Feedback service

feedback

Attribute	Type	Description
_id	String	Feedback ID
grievanceId	String	Grievance reference
citizenId	String	Citizen ID
comments	String	Feedback text
submittedAt	DateTime	Timestamp

ratings

Attribute	Type	Description
_id	String	Rating ID
grievanceId	String	Grievance reference

score	Integer	1–5
submittedAt	DateTime	Timestamp

reopen req

Attribute	Type	Description
_id	String	Request ID
grievanceId	String	Grievance reference
reason	String	Reopen reason
requestedAt	DateTime	Timestamp

Notification service **Use kafka messenger broker this service will be the consumer**

notifs

Attribute	Type	Description
_id	String	Notification ID
userId	String	Recipient
message	String	Notification text
type	Enum	EMAIL / SMS / IN_APP
status	String	SENT / FAILED
createdAt	DateTime	Timestamp

notif_logs

Attribute	Type	Description
_id	String	Log ID
notificationId	String	Parent notification
response	String	Gateway response
loggedAt	DateTime	Timestamp

5. API Design

Grievance Service

/api/grievances

	API Name	HTTP Method	Endpoint	Request Body / Params	Success Response	Error Responses	Description
1	Create Grievance	POST	`/api/grievances/create`	Body: Grievance JSON	`201 CREATED` + Grievance	`400` Validation failed `500` Server error	Citizen submits a new grievance
2	Get Grievance by ID	GET	`/api/grievances/{id}`	grievanceId	`200 OK` + Grievance	`404` Grievance not found	Fetch a specific grievance
3	Get All Grievances	GET	`/api/grievances` or `/api/grievances/getAll`	—	`200 OK` + List of grievances	`500` Server error	Fetch all grievances
4	Assign Grievance	PATCH	`/api/grievances/{id}/assign`	Query:`assignedByassignedTo`	`200 OK` + Updated grievance	`404` Not found`500` Server error	Dept Officer assigns grievance to Case Worker
5	Update Grievance Status	PATCH	`/api/grievances/{id}/status`	Query:`statusupdatedByremarks` (optional)	`200 OK` + Updated grievance	`404` Not found `400` Invalid state `500` Server error	Update grievance workflow status
6	Get Status History	GET	`/api/grievances/{id}/history`	grievanceId	`200 OK` + Status history list	`404` Not found	Get full lifecycle history of grievance

Department Service

/api/departments

API	Method	Endpoint	Request	Success	Errors	Description
Create department	POST	`/create`	Body: department	201	400	Add department
Get all departments	GET	`/getAll`	—	200	—	List departments
Get dept by ID	GET	`/{id}`	Path: id	200	404	Department details
Get categories	GET	`/{id}/categories`	Path: deptId	200	404	Categories under dept
Get subcategories	GET	`/categories/{code}/subcategories`	Path: categoryCode	200	404	Sub-categories
Resolve dept by category	GET	`/resolve`	Param: categoryCode	200	404	Used by grievance service

Feedback Service

/api/feedback

API	Method	Endpoint	Request	Success	Errors	Description
Submit feedback	POST	`/`	Body: feedback	201	400	Citizen rating
Get feedback	GET	`/grievance/{id}`	Path: grievanceId	200	404	View feedback

Analytics	GET	`/stats`	—	200	—	Reports (future)

Notification Service

/api/notifications

API	Method	Endpoint	Request	Success	Errors	Description
Send notification	POST	`/send`	Body: notification	202	500	Send SMS/Email
Notify on status	POST	`/status-update`	Body: grievanceId, status	202	—	Status change alert

Storage Service

/api/storage

API	Method	Endpoint	Request	Success	Errors	Description
Upload file	POST	`/upload`	Multipart file	201	500	Upload attachment
Get file	GET	`/{fileId}`	Path: fileId	200	404	Download
Delete file	DELETE	`/{fileId}`	Path: fileId	204	404	Remove file

6. Tech Stack

Backend

- Java 21
- Spring Boot 3.2.x
- Spring Web
- Spring Data MongoDB (Reactive)
- Spring Validation

- MongoDB
- Spring Security with JWT
- Swagger / OpenAPI
- Maven

Frontend

- Angular (latest version)
- TypeScript
- Angular Material / Bootstrap
- Reactive Forms
- HttpClient & Interceptors

Database

- MongoDB (one DB per microservice)

DevOps

- Docker
- Docker Compose
- Jenkins
- SonarQube