E-Governance Grievance Redressal System

Item	Details
Project Name	E-Governance Grievance Redressal System
Author	Yati
Date	2025-2026
Status	Working

Purpose of Document
This document describes the architecture, design decisions, APIs, data models, workflows, and non-functional aspects of the E-Governance Grievance Redressal System.
Target audience:

- Developers
- Evaluators / Reviewers
- Interview panels
- Maintenance & enhancement teams

1. Introduction

The E-Governance Grievance Redressal System enables citizens to lodge grievances related to public services, upload supporting documents, track their resolution, and escalate delays in a transparent and accountable manner.
The system is designed using microservices architecture, ensuring scalability, security, and clear separation of responsibilities.
Static reference data (Departments, Categories, and SLA definitions) is maintained as JSON configuration inside the Grievance Service to reduce unnecessary services, databases, and operational complexity.

Business Problem Statement

- Citizens face difficulty tracking grievances across departments
- Manual grievance handling causes:
    - Delays
    - Lack of accountability
    - Poor transparency
- No centralized platform for grievance lifecycle tracking
- Escalations are often missed due to lack of monitoring
- Feedback and performance insights are not structured

Business Objective

- Provide a centralized digital platform for grievance management
- Ensure accountability through department assignment
- Enable real-time grievance tracking (PNR-like system)

- Enforce strict grievance lifecycle
- Improve citizen trust and transparency
- Support scalability using microservices

2. Architectural Overview

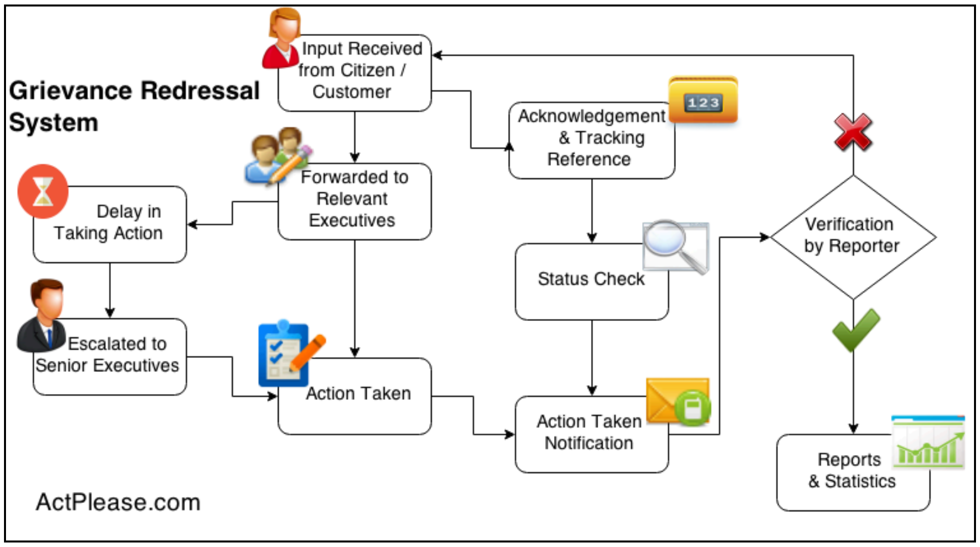
Key Principles

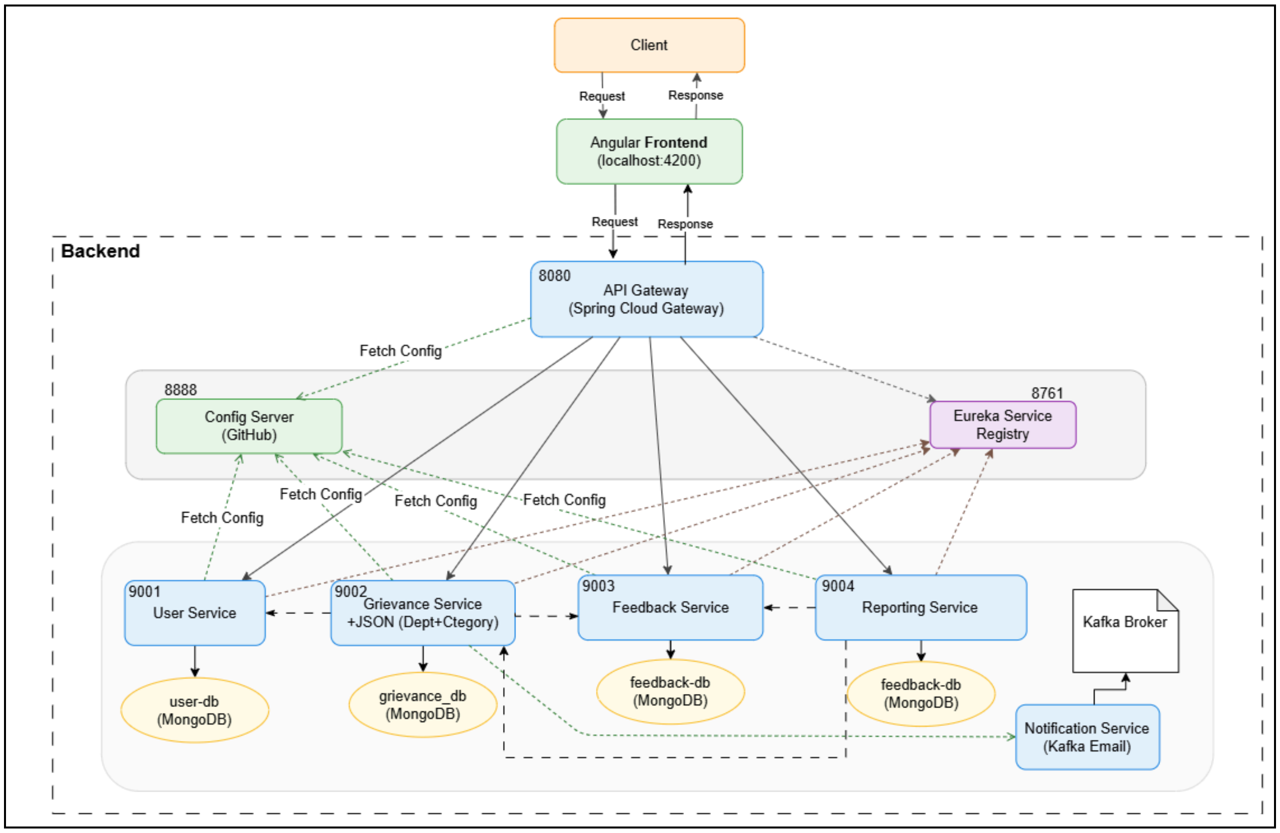
- Single entry point (API Gateway)
- JWT-based authentication
- Role-Based Access Control (RBAC)
- Database per service
- Stateless notifications
- Read-only analytics
- On-demand SLA evaluation (no schedulers)

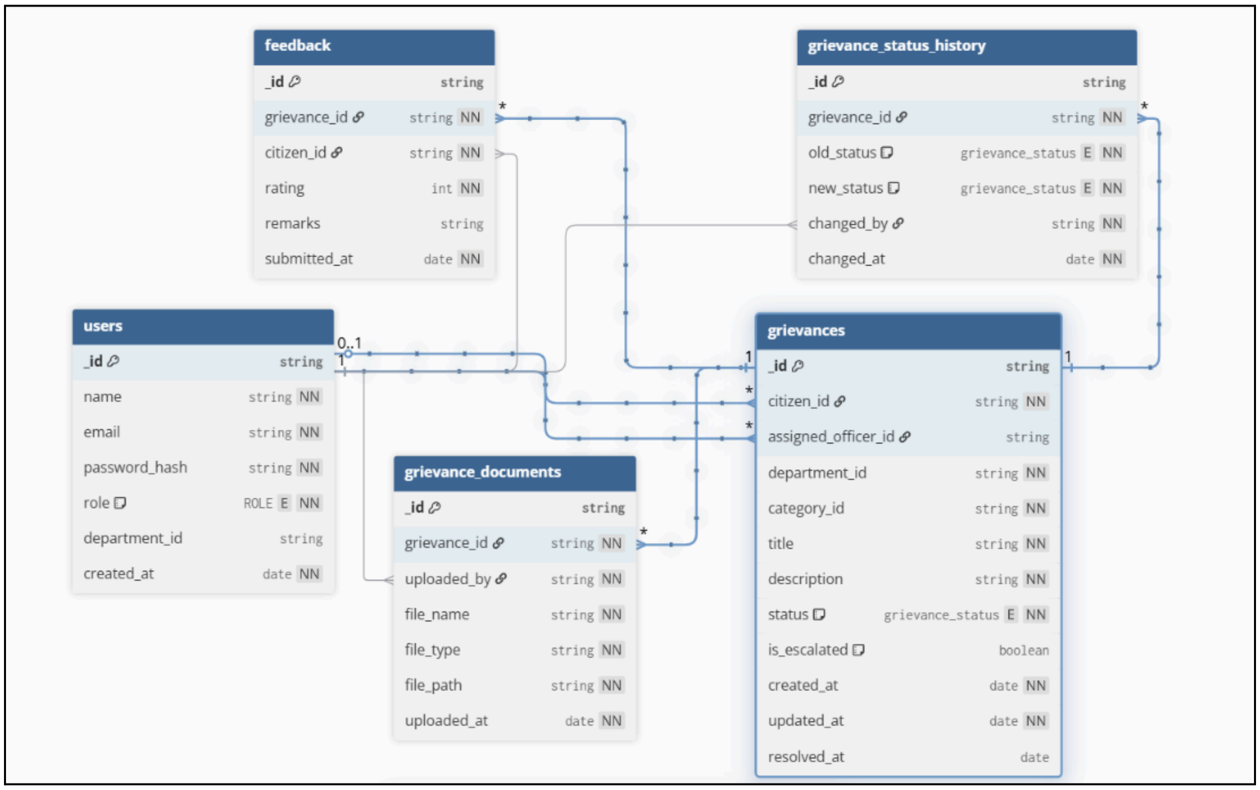
Business Rules:

1. Only authorized roles can perform actions permitted to their role (RBAC enforced).
2. Each user must register with a unique email address.
3. Only citizens are allowed to create grievances.
4. Every grievance is uniquely identified by a system-generated grievanceId.
5. Grievances must follow a fixed lifecycle and cannot skip status transitions.
6. SLA for a grievance is determined by its category and stored in JSON.
7. A grievance can be escalated only if its SLA time is breached.
8. Each grievance can be escalated and reopened only once.
9. Feedback can be submitted only after a grievance is resolved or closed.
10. All grievance status changes must be recorded and cannot be deleted.

Flow:


Grievance Redressal System
Input Received from Citizen / Customer
Acknowledgement & Tracking Reference
123
Forwarded to Relevant Executives
Delay in Taking Action
Escalated to Senior Executives
Action Taken
Status Check
Action Taken Notification
Verification by Reporter
Reports & Statistics
ActPlease.com





feedback

_id	string	
grievance_id	string	NN
citizen_id	string	NN
rating	int	NN
remarks	string	
submitted_at	date	NN

grievance_status_history

_id	string	
grievance_id	string	NN
old_status	grievance_status E	NN
new_status	grievance_status E	NN
changed_by	string	NN
changed_at	date	NN

users

_id	string	
name	string	NN
email	string	NN
password_hash	string	NN
role	ROLE E	NN
department_id	string	
created_at	date	NN

grievance_documents

_id	string	
grievance_id	string	NN
uploaded_by	string	NN
file_name	string	NN
file_type	string	NN
file_path	string	NN
uploaded_at	date	NN

grievances

_id	string	
citizen_id	string	NN
assigned_officer_id	string	
department_id	string	NN
category_id	string	NN
title	string	NN
description	string	NN
status	grievance_status E	NN
is_escalated	boolean	
created_at	date	NN
updated_at	date	NN
resolved_at	date	

Links:

Architecture Diagram:
https://drive.google.com/file/d/1zZa5FkS3n25UV9JUi7VA2zK3-gULcYcp/view?usp=sharing
Databse Design Diagram:
https://dbdiagram.io/d/E-Gov-Grievance-Design-69526a7a39fa3db27bbb0903

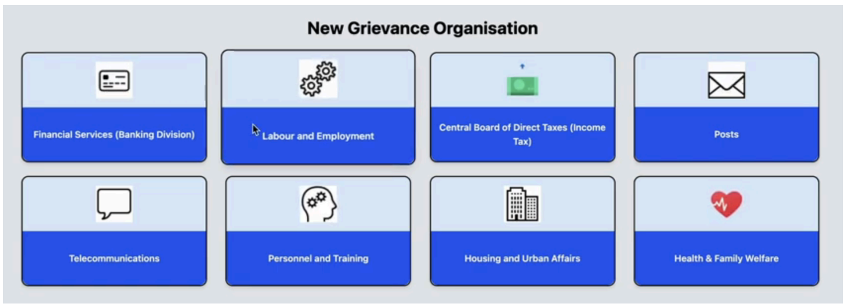
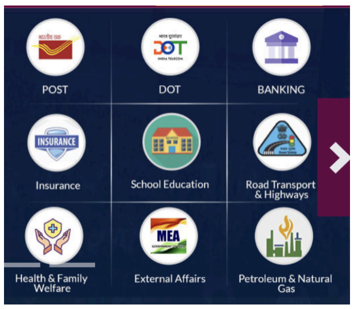
3. Roles & Responsibilities

Role	Responsibilities
Citizen	Register/Login , Lodge grievances, upload documents, view status, escalate delays, close/reopen grievances, give feedback
Officer	Review, process, and resolve grievances
Supervisor	Assign grievances, monitor escalations, view reports
Admin	Manage users and roles

Mandatory Pages

System Admin	Department Officer	Supervisory Officer	Citizen
Login (backend)	Login (initial mail)	Login (initial mail)	Login / Register
			Lodge Grievance
			My Grievances
Grievance Details	Grievance Details	Grievance Details	Grievance Details
	Reports (By Assigned Category)	Reports (By Category)	
Admin Panel			
			Citizen Dashboard
Officer Dashboard	Officer Dashboard (By Assigned Category)	Officer Dashboard (By Category)	

Possible Department Examples: - Json document


New Grievance Organisation
Financial Services (Banking Division)
Labour and Employment
Central Board of Direct Taxes (Income Tax)
Posts
Telecommunications
Personnel and Training
Housing and Urban Affairs
Health & Family Welfare


4. Infrastructure Services

4.1 API Gateway

Business Rules

- Every request must pass through the Gateway
- JWT must be present for protected endpoints
- Role validation happens before routing
- Unauthorized requests are rejected at gateway level

4.2 Config Server

Business Rules

- All services load configuration at startup
- No service stores hardcoded database credentials

4.3 Eureka Registry

Business Rules

- Services must self-register
- Gateway must resolve services dynamically

5. Business Services

5.1 User & Authentication Service

Purpose

Manages user identity, authentication, and authorization.

Core Business Rules

Registration Rules

- Email must be unique
  Password must be encrypted before storage
- Public registration defaults to role CITIZEN
- Only Admin can create Officers, Supervisors, and Admins

Authentication Rules

- Login fails if credentials are invalid
- JWT must contain:
    - userId
    - role
- JWT expiry must be enforced

Authorization Rules

- Users can view/update only their own profile
- Admin can view/update any user
- Only Admin can change roles

- Only Admin can assign departments to Officers and Supervisors

Method	Endpoint	Purpose	Request Payload	Success Response	Success Code	Error Codes
POST	`/auth/register`	Register a new user (Citizen by default)	`{ "name": "Amit", "email": "amit@gmail.com", "password": "Test@123" }`	`"userId"`	**201 CREATED**	400 (validation), 409 (email exists)
POST	`/auth/login`	Authenticate user and issue JWT	`{ "email": "amit@gmail.com", "password": "Test@123" }`	`{ "token": "jwt-token", "userId": "userId", "role": "CITIZEN" }`	**200 OK**	401 (invalid credentials)
GET	`/users/{userId}`	Get user profile	—	`{ "id": "userId", "name": "Amit", "email": "amit@gmail.com", "role": "CITIZEN", "departmentId": null, "createdAt": "2025-01-10T10:00:00" }`	**200 OK**	404 (not found), 403 (forbidden)
PUT	`/users/{userId}`	Update user profile	`{ "name": "Amit Kumar" }`	`"UPDATED"`	**200 OK**	400 (invalid), 403 (forbidden), 404 (not found)
PUT	`/users/{userId}/role/{role}`	Change user role - by Admin	—	`"ROLE_UPDATED"`	**200 OK**	400 (invalid role), 403 (forbidden), 404 (not found)
PUT	`/users/{userId}/department/{departmentId}`	Assign department to officer - by admin	—	`"DEPARTMENT_ASSIGNED"`	**200 OK**	400 (invalid dept), 403 (forbidden), 404 (not found)
GET	`/users/role/{role}`	Get users by role	—	`[ { "id": "userId", "name": "Officer A", "email": "officer@gov.in", "departmentId": "deptId" } ]`	**200 OK**	400 (invalid role)

5.2 Grievance Service (CORE SERVICE)

Purpose

Handles grievance lifecycle from creation to closure, including document uploads, SLA checks, and escalation.

5.2.1 JSON-Based Master Data Rules

Departments, categories, and SLA values are stored in JSON.

```
{
  "categoryId": 101,
  "name": "Water Leakage",
  "slaHours": 48
}
```

Rules

- Departments, categories, and SLA values are read-only
- Data is loaded at application startup
- Every grievance must reference:
  - a valid departmentId
  - a valid categoryId
- SLA is defined at category level
- Invalid IDs result in request rejection

5.2.2 Grievance Creation Rules (With Document Upload)

- Only CITIZEN can create grievances
- Title and description are mandatory
- Initial status must be SUBMITTED
- grievanceId is auto-generated
- Supporting documents are optional
- Uploaded documents are immutable after submission
- File type and size validation is mandatory
- Status history entry is mandatory
- Notification must be triggered

5.2.3 Grievance Access Rules

- Citizen can view only their own grievances
- Officer can view grievances assigned to them
- Supervisor & Admin can view all grievances

5.2.4 Grievance Lifecycle Rules

Action	**Allowed Role**	**Rule**
Assign	Supervisor/Admin	Only if status = SUBMITTED

Review	Officer	Only if assigned
Resolve	Officer	Only if IN_REVIEW
Close	Officer	Only if RESOLVED
Reopen	Citizen	Only within allowed time
Escalate	Citizen	Only if SLA breached

5.2.5 SLA & Escalation Rules

- SLA duration is calculated using category-level slaHours
- SLA breach is evaluated on-demand when:
  - grievance is viewed
  - citizen logs in
  - citizen clicks “Escalate”
- No scheduled job or cron is used
- Escalation is allowed only if:
  - grievance is not RESOLVED or CLOSED
  - SLA time has elapsed
- Escalation can happen only once
- Status becomes ESCALATED
- On escalation, grievance becomes visible to Supervisors for intervention.
- Escalation must be recorded in history
- Notification must be sent to Supervisor

5.2.6 Reopen Rules

- Grievance can be reopened only once
- Reopen allowed only if status = RESOLVED
- Reopen must be within time window (e.g., 7 days)
- Status becomes REOPENED
- History entry is mandatory

5.2.7 Audit Rules

- Every status change must be recorded
- History entries are immutable
- Audit data cannot be deleted

Json Document - example - For Department and category
departments-categories-sla.json :-

```
{
  "departments": [
    {
      "departmentId": "D001",
      "name": "Water Supply Department",
      "categories": [
        {
          "categoryId": "C101",
          "name": "Water Leakage",
          "slaHours": 48,
          "description": "Leakage in pipelines, taps, or water mains"
        },
        {
          "categoryId": "C102",
          "name": "Low Water Pressure",
          "slaHours": 72,
          "description": "Insufficient water pressure in households"
        },
        {
          "categoryId": "C103",
          "name": "No Water Supply",
          "slaHours": 24,
          "description": "Complete disruption of water supply"
        }
      ]
    },
    {
      "departmentId": "D002",
      "name": "Electricity Department",
      "categories": [
        {
          "categoryId": "C201",
          "name": "Power Outage",
          "slaHours": 12,
          "description": "Unexpected electricity outage"
        },
        {
          "categoryId": "C202",
          "name": "Voltage Fluctuation",
          "slaHours": 48,
          "description": "Frequent voltage ups and downs"
        },
        {
          "categoryId": "C203",
          "name": "Billing Issue",
          "slaHours": 96,
          "description": "Incorrect electricity bill or meter reading"
        }
      ]
    },
    {
      "departmentId": "D003",
      "name": "Municipal Corporation",
      "categories": [
        {
          "categoryId": "C301",
          "name": "Garbage Collection",
          "slaHours": 24,
          "description": "Garbage not collected from locality"
        },
        {
          "categoryId": "C302",
          "name": "Road Damage",
          "slaHours": 120,
          "description": "Potholes or damaged roads"
        },
        {
          "categoryId": "C303",
          "name": "Street Light Issue",
          "slaHours": 48,
          "description": "Non-functional street lights"
        }
      ]
    }
  ]
}
```

Method	Endpoint	Purpose	Request Payload	Success Response	Success Code	Error Codes
GET	**`/reference/departments`**	Get all departments (JSON)	—	`[{"departmentId:"1","name": "Water Dept" } ]`	**200 OK**	500
GET	**`/reference/departments/{departmentId}/categories`**	Get categories + SLA	—	`[{"categoryId":"101","name":"Leakage","slaHours":48}]`	**200 OK**	404

POST	`/grievances`	Create grievance (optional upload document)	`multipart/form-data:title, description, departmentId, categoryId, document`	`"grievanceId"`	**201 CREATED**	400, 404
GET	`/grievances/{grievanceId}`	Get grievance details	—	`{ grievance object }`	**200 OK**	404, 403
GET	`/grievances/citizen/{citizenId}`	Get grievances of a citizen	—	`[ { grievance summary } ]`	**200 OK**	404
GET	`/grievances/department/{departmentId}`	Get department grievances	—	`[ { grievance summary } ]`	**200 OK**	404
PUT	`/grievances/{grievanceId}/assign/{officerId}`	Assign grievance	—	`"ASSIGNED"`	**200 OK**	400, 403, 404
PUT	`/grievances/{grievanceId}/review`	Mark in review	—	`"IN_REVIEW"`	**200 OK**	400, 403
PUT	`/grievances/{grievanceId}/resolve`	Resolve grievance	`{ "remarks": "Resolved" }`	`"RESOLVED"`	**200 OK**	400, 403
PUT	`/grievances/{grievanceId}/close`	Close grievance	—	`"CLOSED"`	**200 OK**	400, 403
PUT	`/grievances/{grievanceId}/reopen`	Reopen grievance	—	`"REOPENED"`	**200 OK**	400, 403
PUT	`/grievances/{grievanceId}/escalate`	Escalate grievance (SLA breach)	—	`"ESCALATED"`	**200 OK**	400, 403
GET	`/grievances/{grievanceId}/history`	Get status history	—	`[ { history record } ]`	**200 OK**	404
GET	`/grievances/{grievanceId}/documents`	List uploaded documents	—	`[ { document metadata } ]`	**200 OK**	404
GET	`/grievances/{grievanceId}/documents/{documentId}`	Download document	—	`file stream`	**200 OK**	404, 403

5.3 Feedback Service

Purpose
Capture citizen feedback post-resolution.

Business Rules
- Feedback allowed only if grievance is CLOSED
- Only grievance owner can submit feedback
- Only one feedback per grievance
- Rating must be between 1 and 5
- Feedback cannot be edited or deleted
- Feedback is read-only once submitted
- Feedback data is used only for analytics

Method	Endpoint	Purpose	Request Payload	Success Response	Success Code	Error Codes
POST	**`/feedback/grievance/{grievanceId}`**	Submit feedback	`{"rating”: 4,"remarks": "Resolved properly" }`	`"feedbackId"`	**201 CREATED**	400 (invalid rating), 403 (not owner), 404 (grievance not found), 409 (already submitted)
GET	**`/feedback/grievance/{grievanceId}`**	Get feedback of grievance	—	`{ "feedbackId": "id", "rating": 4, "remarks": "Resolved properly", "submittedAt": "2025-01-12T18:00:00" }`	**200 OK**	404
GET	**`/feedback/{grievanceId}`**	Get feedback submitted by user	—	`[ { "grievanceId": "id","citizenId": "id", "rating": 4 ,"createdAt": "2025-12-31T13:39:06.210Z" } ]`	**200 OK**	404
GET	`/feedback/department/{departmentId}`	Get department feedback (analytics)	—	`[ { "grievanceId": "id", "rating": 5 } ]`	**200 OK**	404

5.4 Notification Service
Purpose
Send email notifications asynchronously.

Business Rules

- Notification service stores no data
- It cannot be called directly by frontend
- Emails are triggered only by internal services
- Failure to send email must not block business flow
- Retry mechanism may be applied (optional)

5.5 Reporting & Analytics Service

Purpose

Provide dashboards and KPIs.

Business Rules

1. Reporting APIs are read-only
2. No transactional updates allowed
3. Only Supervisor & Admin can access reports
4. Aggregation must not impact grievance service
5. Reports use:
   - grievance status
   - timestamps
   - feedback ratings
   - escalation counts

Method	Endpoint	Purpose	Request Payload	Success Response	Success Code	Error Codes
GET	`/reports/grievances/status/{status}`	Get grievances by status	—	`{"status":"IN_REVIEW", "count": 12 }`	**200 OK**	400 (invalid status)
GET	`/reports/grievances/department/{departmentId}`	Get department grievance summary	—	`{ "departmentId": "1", "total": 120 }`	**200 OK**	404
GET	`/reports/avg-resolution-time`	Get average resolution time	—	`{ "averageHours": 36 }`	**200 OK**	404
GET	`/reports/avg-resolution-time/department/{departmentId}`	Get dept-wise avg resolution time	—	`{ "departmentId": "1", "averageHours": 28 }`	**200 OK**	404
GET	`/reports/department-performance`	Department performance metrics	—	`[ { "departmentId": "1", "avgRating": 4 } ]`	**200 OK**	404
GET	`/reports/user/{userId}`	**User grievance summary**	—	`{ "userId": "userId", "totalGrievances": 5 }`	**200 OK**	404

	GET
	`/reports/grievances/sla-breaches`
	
	`SLA violated grievances`					

6. End-to-End Business Flow (With Rules)

Grievance Submission

1. Citizen logs in
2. Submits grievance with optional document upload
3. JSON validation occurs
4. Grievance saved with status SUBMITTED
5. History entry created
6. Email notification sent

Grievance Resolution

1. Supervisor assigns grievance
2. Officer reviews
3. Officer resolves
4. History entries created
5. Citizen notified
6. Feedback allowed

Grievance Escalation

1. Citizen views grievance
2. SLA evaluated dynamically
3. Citizen escalates grievance
4. Status updated to ESCALATED
5. Supervisor notified

7. Security Rules Summary

- Passwords are never stored in plain text
- JWT required for all protected endpoints
- Role checks enforced at gateway & service layer
- Services trust JWT, not user input

8. Design Trade-Offs & Justification

Decision	Justification
JSON master data + SLA	Static, simple, no extra DB
grievanceId only	No duplicate identifiers
On-demand escalation	No scheduler complexity
Document metadata only	Clean DB, scalable
Stateless notifications	Lightweight & scalable
Separate feedback service	Clean lifecycle separation
Separate reporting service	Performance isolation

9. Future Enhancements

- Multi-level SLA escalation
- SMS notifications
- Admin UI for JSON management
- Cloud document storage (S3)
- Public grievance tracking

10. Conclusion

This design:

- Meets all functional requirements
- Enforces clear business rules
- Ensures security, auditability, and transparency
- Supports document uploads and SLA-based escalation
- Balances enterprise architecture with academic simplicity