E-Governance Grievance Redressal System

DOCUMENT CONTROL

Item	Details
Project Name	E-Governance Grievance Redressal System
Author	Aryan
Date	(Date)
Status	Working

PURPOSE OF THE DOCUMENT

This document describes the **system architecture, design decisions, component structure, APIs, data models, and non-functional aspects** of the Smart Order Management System.

It is intended for:

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

SYSTEM OVERVIEW

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

Business Rules

1. Only authenticated users with valid roles can access system features, enforced through JWT-based role authorization.

2. Citizens are the only users permitted to lodge grievances, and each grievance must be associated with exactly one category and department.

3. Every grievance is assigned a unique system-generated grievance ID and timestamp at the time of submission.

4. Grievances follow a strictly enforced lifecycle: Submitted → Assigned → In Review → Resolved → Closed, with no status skipping allowed.

5. Grievances are automatically routed to the appropriate department based on category-to-department mapping.

6. Only authorized department officers can update grievance status and submit resolution remarks.

7. A grievance is eligible for escalation if it remains unresolved beyond a predefined resolution timeframe.

8. Citizens may submit feedback and ratings only after a grievance has been resolved, with limited scope for reopening.

9. Administrative and supervisory users have read-only access to analytics and performance reports derived from grievance data.

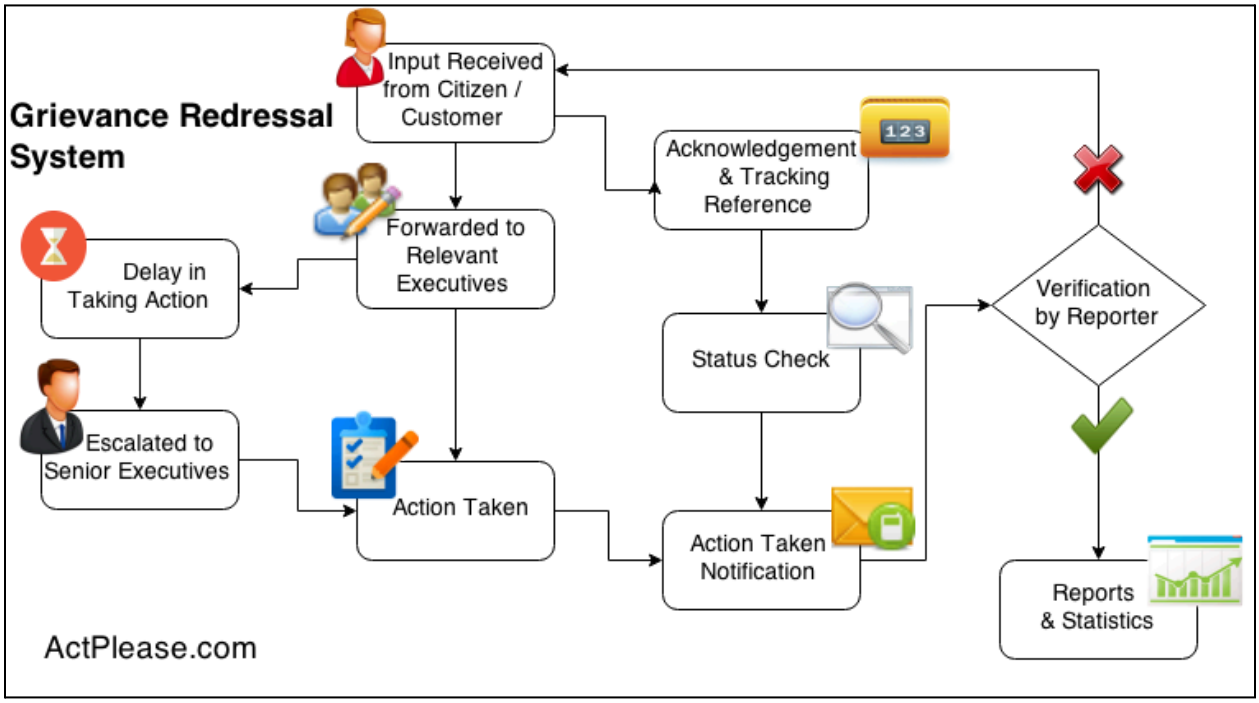
10. All grievance actions, status changes, and resolutions are logged with timestamps to ensure transparency, traceability, and auditability.

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 → Assigned → In Review → Resolved → Closed/Withdrawn*
- Automatic or manual department assignment for accountability
- Role-based dashboards and access control (Citizen / Officer / Supervisor / Admin)

- Status change notifications via notification service
- Escalation of grievances if not resolved within defined time limits
- Feedback submission by citizens after grievance resolution
- Reports and analytics for department-wise and category-wise performance

FLOW:



PAGES/FEATURES

Login page
Register Page
Home Page
Issues register page (Document / image upload) AI for category assigning on description
View status (**Submitted → Assigned → In Review → Resolved → Closed/Withdraw)**
Issue page (open + closed - tabs)
Profile page (adhaar card, name)
Feedback page (after resolution - customer can give feedback)
Mail system - notification service
Escalations - time period mai resolve nhi hua to escalate to higher authorities
Contact Us Page

Mandatory Pages

1. Different frontend for different roles (for efficient scaling)

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 - Citizen
Officer Dashboard	Officer Dashboard (By Assigned Category)	Officer Dashboard (By Category)	

Core Components/ Pages

- Login / Register - User
- Citizen Dashboard - Citizen
- Lodge Grievance
- My Grievances
- Grievance Details
- Officer Dashboard
- Reports
- Admin Panel

Optional:

FAQ

Possible Department Examples:

ROLES

1. **System Admin** (CM/PM): Manage departments, grievance categories, users, and roles

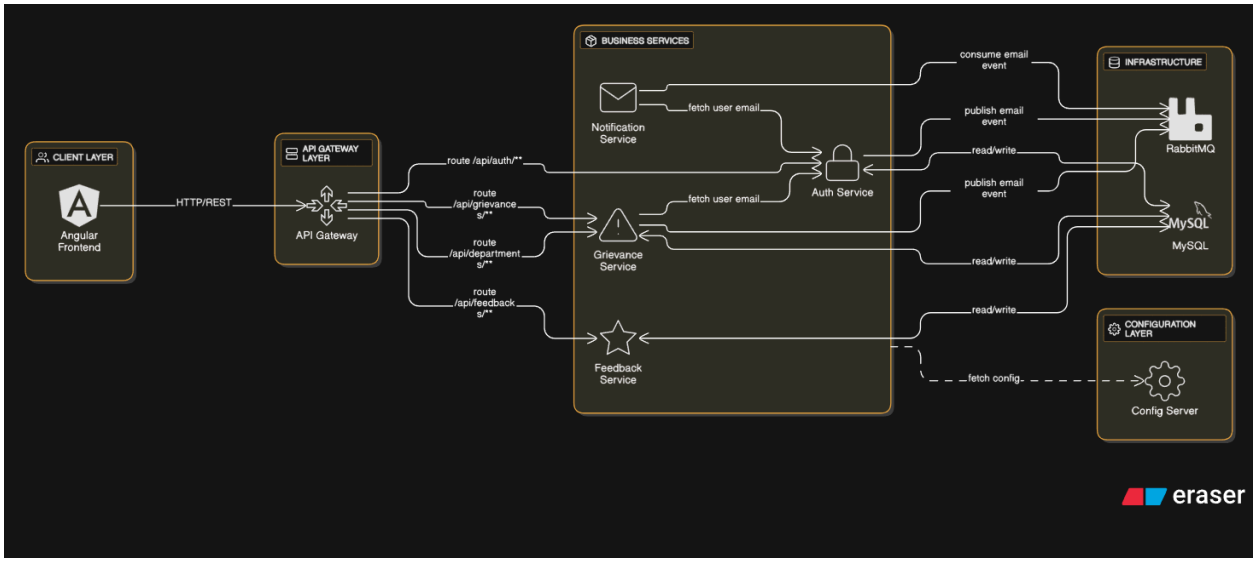
2. Department Officer: View and resolve assigned grievances

3. Supervisory Officer (Department Head): Monitor grievance status and escalations

4. Citizen: Register, submit grievances, and track status

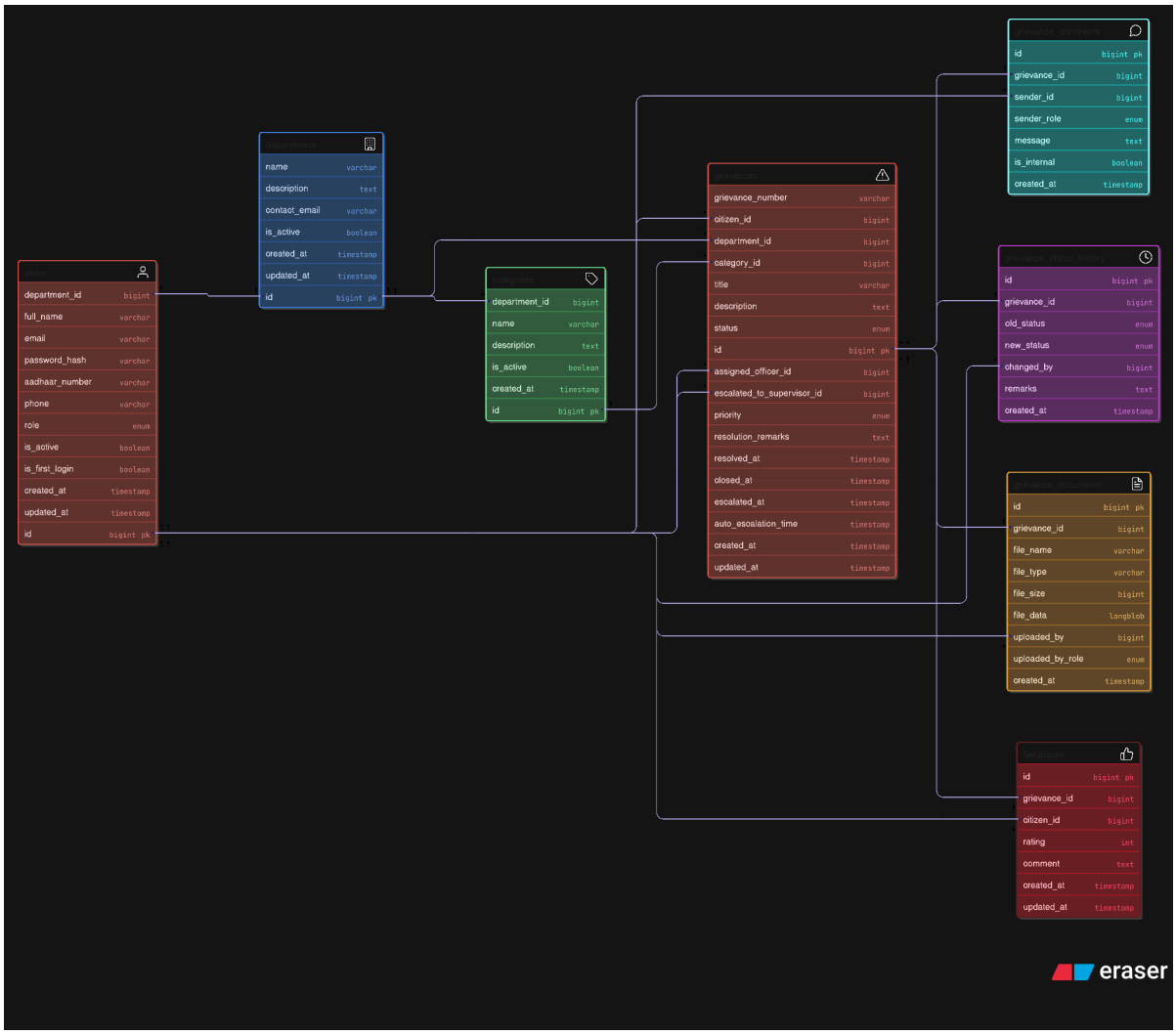
Architecture Diagram:

https://app.eraser.io/workspace/wpgaOZ2nPP4XFqlyJ6XC



ER Diagram:

https://app.eraser.io/workspace/jl9A0y4lZgxXF4p017Vr



MicroServices

1. API GATEWAY
2. Security Service *(id,full_name, aadhaar_number, email, password_hash, role, phone)*
3. Config Server
4. Grievance Service *(id, grievance_number, citizen_id, department_id,title, description, status, assigned_officer_id)*
5. Notification service *(id, user_id, subject, message)*
6. Department Service *(id, name, description)*
7. Feedback Service *(id, grievance_id, citizen_id, comment)*
8. Storage Service *(id, grievance_id, file_name, file_data)*

Optional: Workflow/Resolution Service

Endpoints

1. Security Service /User Service

Authentication & Security Service APIs

HTTP Method	Endpoint	Description	Access Role	Notes
POST	`/auth/register`	Register a new user	Citizen / Admin	Department account creation via request header
POST	`/auth/login`	Authenticate user and issue JWT	All Users	Returns JWT token
POST	`/auth/logout`	Logout user	Authenticated User	Invalidates session/token
GET	`/auth/validate`	Validate JWT token	Internal / Gateway	Used by API Gateway
POST	`/auth/change-password`	Change account password	Authenticated User	Requires old password

POST	`/auth/forgot-password`	Initiate password reset	All Users	Sends reset link/OTP

User Service APIs

HTTP Method	Endpoint	Description	Access Role	Notes
GET	`/users`	Fetch all users	Admin	Supports pagination
GET	`/users/{userId}`	Fetch user details by ID	Admin, Self	Restricted access
PUT	`/users/{userId}`	Update user profile	Admin, Self	Profile information only
DELETE	`/users/{userId}`	Delete / deactivate user	Admin	Soft delete recommended

---

Role & Department Mapping APIs

HTTP Method	Endpoint	Description	Access Role	Notes
GET	`/users/by-role/{role}`	Get users by role	Admin	Example: CITIZEN, OFFICER
GET	`/users/by-department/{departmentId}`	Get users by department	Admin	Used for officer assignment
PUT	`/users/{userId}/role`	Assign or update user role	Admin	Role-based access enforced
PUT	`/users/{userId}/department`	Assign user to department	Admin	Mandatory for officers

2. **Department Service -**

Method	Endpoint	Role	Description
GET	/departments	Public	Get All Departments (for Dropdowns/Filters).
POST	/departments	Admin	Create a new Department.
GET	/departments/{id}	Public	Get Single Department details.
PUT	/departments/{id}	Admin	Update Department details.
DELETE	/departments/{id}	Admin	Soft delete a department.
POST	/departments/{id}/officers	Supervisor	Assign an existing User to this Department (as Officer).
GET	/departments/{id}/officers	Supervisor	List officers specifically for this department.

GET	/departments/{id}/categories	Public	Get categories (e.g., Water -> "Pipe Leak", "Bad Quality").
POST	/departments/{id}/categories	Admin	Add a new grievance category.
GET	/departments/categories	Public	Flat list of all categories (for Search).

3. Grievance Service

Handles the lifecycle: Submitted → Assigned → In Review → Resolved.

HTTP Method	Endpoint	Description	Access Role	Notes
POST	`/grievances`	Lodge a new grievance	Citizen	Generates a unique grievance ID (PNR-like)
GET	`/grievances/my-grievances`	List grievances of logged-in citizen	Citizen	Supports pagination
GET	`/grievances`	List all grievances	Admin, Supervisor	Supports filters like `?status=PENDING&deptId=2`
GET	`/grievances/{id}`	Get full details of a grievance	Citizen, Officer, Admin,	Access restricted to authorized users

			Supervisor	
PATCH	`/grievances/{id}/assign/{officerId}`	Assign grievance to an officer	Supervisor	Can be auto-assigned or manual
PATCH	`/grievances/{id}/status`	Update grievance status	Officer	Includes resolution remarks
POST	`/grievances/{id}/comments`	Add comment to grievance	Citizen, Officer	Supports clarification & discussion
GET	`/grievances/{id}/comments`	Fetch grievance comment history	Citizen, Officer	Used for chat-style UI

Status Update Request Body Example

```
{
  "status": "RESOLVED",
  "remarks": "Fixed the pothole."
}
```

Comment Request Body Example

```
{
  "senderId": "user_001",
  "role": "OFFICER",
  "message": "Please provide the exact street name.",
  "updateStatusTo": "CLARIFICATION_REQUIRED"
}
```

To track progress: - with detailed info

No need to box your item

Keep the item in its original manufacturer packaging.

No need to print a label

Carrier will bring a label at the time of pickup.

Tracking Id: 6275

Return pickup location

Priyanshu Kumar Saw,
Tower A, Green Boulevard, NOIDA, UTTAR PRADESH, 201301
Phone Number: 870

Scheduled Pickup

Saturday, Oct 12, 2024, 07:00 am - 10:00 pm

Important information

Return the item in the original condition and brand packaging, to avoid return getting rejected. Click here for more details.

4. Feedback Service

(loop or only once - decide?)

Method	Endpoint	Role	Description
POST	/feedbacks	Citizen	Submit feedback. **Check:** If grievanceId already has feedback, throw Error. Body: { "grievanceId": "123", "rating": 5, "comment": "Fast service!" }

GET	/feedbacks/grievance/{id}	Any	Check if feedback exists for a ticket.
GET	/feedbacks/average	Admin	Get average rating per department. (for Admin Dashboard)

5. API Gateway

1. /api/auth/** → Security Service
2. /api/users/** → User Service
3. /api/grievances/** → Grievance Service
4. /api/departments/** → Department Service
5. /api/feedbacks/** → Feedback Service

6. DepartmentService

Method	Endpoint	Role	Description
GET	/	Public/Auth	**Get All Departments.** Used to populate the "Select Department" dropdown on the frontend or filters on dashboards.
GET	/{id}	Public/Auth	**Get Single Department.** Returns details like name, description, and contact info.
POST	/	ADMIN	**Create Department.** Add a new department to the system.

PUT	/{id}	ADMIN	**Update Department.** Edit name or contact email.
DELETE	/{id}	ADMIN	**Delete Department.** (Soft delete recommended so old grievances don't break).
GET	/{id}/categories	Public/Auth	**Get Categories by Dept.** If user selects "Water Dept", call this to show ["Pipe Leak", "Low Pressure"].
POST	/{id}/categories	ADMIN	**Add Category.** Add a new issue type to a department.
GET	/categories	Public/Auth	**Get All Categories.** Flat list of all possible issues (useful for search).

7.Notification Service

Method	**Endpoint**	**Role**	**Description**
POST	**/notifications/send**	**Internal**	**Manually trigger email/SMS.**
GET	**/notifications/my**	**Any**	**View notification history in the app.**

Notification When?

i. User register
ii.password change
iii. Reset password

iv. status of grievance changes
Optional: **Send Reminder**

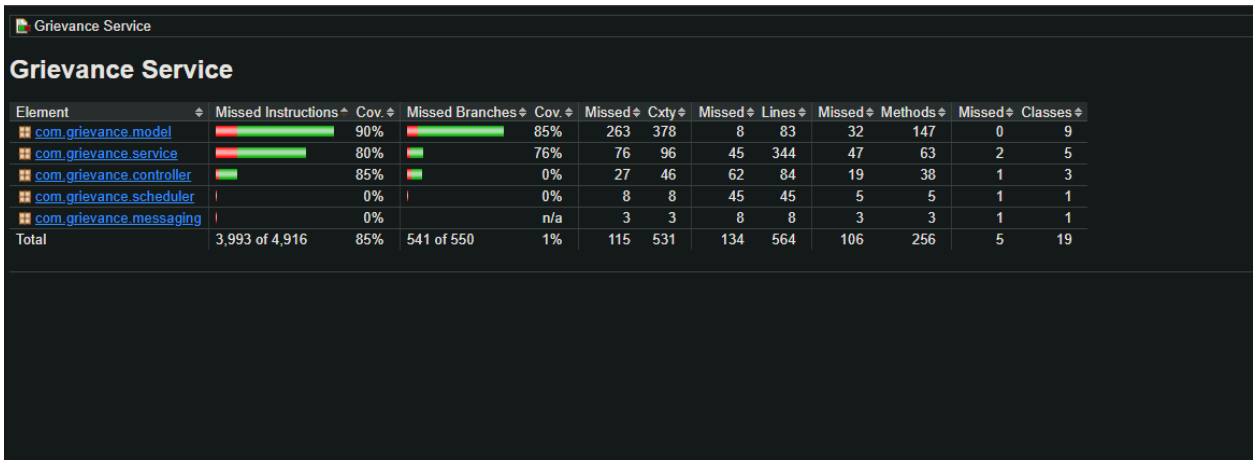
Additional Features

1. Login/Get grievance detail pe trigger escalation if not done any status change in specified duration of time.
2. Spam detection for grievance using AI
3. Rewrite description using AI.

Non functional Requirements:

- RESTful API design
- Centralized exception handling
- Secure handling of citizen data
- Scalable and maintainable codebase

Jacoco Code coverage



DATA DESIGN (LLD)

Service Responsibilities

Service	Database	Responsibilities
Auth Service	`grievance_auth`	User management, Authentication, Authorization, JWT generation

Grievance Service	`grievance_main`	Grievance CRUD, Status management, Assignment, Documents, Comments
Notification Service	`grievance_notifications`	Email notifications via SMTP
Feedback Service	`grievance_feedback`	Post-resolution feedback, Ratings

---

3. Database Design

3.1 Auth Service Database (`grievance_auth`)

3.1.1 Users Table

sql

```
CREATE TABLE users (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    full_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    aadhaar_number VARCHAR(12) UNIQUE,
    phone VARCHAR(15),
    role ENUM('CITIZEN', 'DEPT_OFFICER', 'SUPERVISOR', 'ADMIN') NOT NULL,
    department_id BIGINT,
    is_active BOOLEAN DEFAULT TRUE,
    is_first_login BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,

    INDEX idx_email (email),

    INDEX idx_role (role),

    INDEX idx_department (department_id)

);
```

Constraints:

- Email must be unique and valid format
- Aadhaar must be 12 digits (only for CITIZEN)
- Password must meet complexity requirements (min 8 chars)
- Officers must have department_id assigned

3.2 Grievance Service Database (`grievance_main`)

3.2.1 Departments Table

sql

```
CREATE TABLE departments (

    id BIGINT PRIMARY KEY AUTO_INCREMENT,

    name VARCHAR(100) UNIQUE NOT NULL,

    description TEXT,

    contact_email VARCHAR(100),

    is_active BOOLEAN DEFAULT TRUE,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

);
```

3.2.2 Categories Table

sql

```
CREATE TABLE categories (

    id BIGINT PRIMARY KEY AUTO_INCREMENT,

    department_id BIGINT NOT NULL,
```

```
    name VARCHAR(100) NOT NULL,
    description TEXT,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (department_id) REFERENCES departments(id),
    UNIQUE KEY uk_dept_category (department_id, name)
);
```

3.2.3 Grievances Table

sql

```
CREATE TABLE grievances (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    grievance_number VARCHAR(20) UNIQUE NOT NULL,
    citizen_id BIGINT NOT NULL,
    department_id BIGINT NOT NULL,
    category_id BIGINT NOT NULL,
    title VARCHAR(200) NOT NULL,
    description TEXT NOT NULL,
    status ENUM('SUBMITTED', 'ASSIGNED', 'IN_REVIEW', 'RESOLVED',
            'CLOSED', 'ESCALATED') DEFAULT 'SUBMITTED',
    assigned_officer_id BIGINT,
    escalated_to_supervisor_id BIGINT,
    priority ENUM('LOW', 'MEDIUM', 'HIGH', 'CRITICAL') DEFAULT 'MEDIUM',
    resolution_remarks TEXT,
    resolved_at TIMESTAMP NULL,
```

```
    closed_at TIMESTAMP NULL,
    escalated_at TIMESTAMP NULL,
    auto_escalation_time TIMESTAMP NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (department_id) REFERENCES departments(id),
    FOREIGN KEY (category_id) REFERENCES categories(id),
    INDEX idx_citizen (citizen_id),
    INDEX idx_status (status),
    INDEX idx_officer (assigned_officer_id),
    INDEX idx_department (department_id),
    INDEX idx_grievance_number (grievance_number)
);
```

3.2.4 Grievance Status History Table

sql

```
CREATE TABLE grievance_status_history (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    grievance_id BIGINT NOT NULL,
    old_status ENUM('SUBMITTED', 'ASSIGNED', 'IN_REVIEW', 'RESOLVED',
                'CLOSED', 'ESCALATED'),
    new_status ENUM('SUBMITTED', 'ASSIGNED', 'IN_REVIEW', 'RESOLVED',
                'CLOSED', 'ESCALATED') NOT NULL,
    changed_by BIGINT NOT NULL,
    remarks TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    FOREIGN KEY (grievance_id) REFERENCES grievances(id) ON DELETE CASCADE,
    INDEX idx_grievance (grievance_id)
);
```

3.2.5 Grievance Documents Table

sql

```
CREATE TABLE grievance_documents (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    grievance_id BIGINT NOT NULL,
    file_name VARCHAR(255) NOT NULL,
    file_type VARCHAR(50),
    file_size BIGINT,
    file_data LONGBLOB NOT NULL,
    uploaded_by BIGINT NOT NULL,
    uploaded_by_role ENUM('CITIZEN', 'OFFICER', 'SUPERVISOR') NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (grievance_id) REFERENCES grievances(id) ON DELETE CASCADE,
    INDEX idx_grievance (grievance_id)
);
```

3.2.6 Grievance Comments Table

sql

```
CREATE TABLE grievance_comments (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    grievance_id BIGINT NOT NULL,
    sender_id BIGINT NOT NULL,
    sender_role ENUM('CITIZEN', 'DEPT_OFFICER', 'SUPERVISOR', 'ADMIN') NOT NULL,
    message TEXT NOT NULL,
```

```
    is_internal BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (grievance_id) REFERENCES grievances(id) ON DELETE CASCADE,
    INDEX idx_grievance (grievance_id)
);
```

3.3 Feedback Service Database (`grievance_feedback`)

3.3.1 Feedbacks Table

sql

```
CREATE TABLE feedbacks (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    grievance_id BIGINT UNIQUE NOT NULL,
    citizen_id BIGINT NOT NULL,
    rating INT NOT NULL CHECK (rating BETWEEN 1 AND 5),
    comment TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_grievance (grievance_id),
    INDEX idx_citizen (citizen_id),
    INDEX idx_rating (rating)
);
```

---

4. API Design

4.1 Authentication APIs (Base: `/api/auth`)

4.1.1 Register Citizen

```
POST /api/auth/register
Content-Type: application/json
```

Request Body:

```
{
  "fullName": "string",
  "email": "string",
  "password": "string",
  "aadhaarNumber": "string (12 digits)",
  "phone": "string"
}
```

Response (201 Created):

```
{
  "success": true,
  "message": "Registration successful",
  "data": {
    "userId": 1,
    "email": "user@example.com"
  }
}
```

Validations:

- Email: valid format, unique
- Password: min 8 chars, 1 uppercase, 1 lowercase, 1 digit
- Aadhaar: exactly 12 digits, unique
- Phone: 10 digits

4.1.2 Login

POST /api/auth/login

Content-Type: application/json

Request Body:

```
{
  "email": "string",
  "password": "string"
}
```

Response (200 OK):

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "email": "user@example.com",
  "role": "CITIZEN",
  "isFirstLogin": false,
  "message": "Login successful"
}
```

JWT Claims:

```
{
  "sub": "user@example.com",
  "userId": 1,
  "role": "CITIZEN",
  "departmentId": null,
  "iat": 1640000000,
  "exp": 1640086400
```

}

4.1.3 Change Password

POST /api/auth/change-password

Authorization: Bearer <token>

Content-Type: application/json

Request Body:

```
{
  "oldPassword": "string",
  "newPassword": "string"
}
```

Response (200 OK):

```
{
  "success": true,
  "message": "Password changed successfully",
  "data": null
}
```

4.2 User Management APIs (Base: `/api/users`)

4.2.1 Create Officer (Admin Only)

POST /api/users/officers

Authorization: Bearer <admin-token>

Content-Type: application/json

Request Body:

{

```
  "fullName": "string",
  "email": "string",
  "role": "DEPT_OFFICER" | "SUPERVISOR",
  "departmentId": 1,
  "phone": "string"
}
```

Response (201 Created):

```
{
  "success": true,
  "message": "Officer created successfully",
  "data": {
    "userId": 10,
    "email": "officer@dept.gov.in",
    "temporaryPassword": "TempPass123@"
  }
}
```

Business Logic:

1. Generate 12-character random password
2. Set isFirstLogin = true
3. Send credentials via email (RabbitMQ)
4. Return temp password to admin

4.2.2 Get All Users (Paginated)

GET /api/users?page=0&size=20

Authorization: Bearer <admin-token>

Response (200 OK):

```
{
  "success": true,
  "message": "Users fetched successfully",
  "data": {
    "content": [...],
    "pageable": {...},
    "totalPages": 5,
    "totalElements": 100
  }
}
```

4.3 Grievance APIs (Base: `/api/grievances`)

4.3.1 Lodge Grievance

POST /api/grievances

Authorization: Bearer <citizen-token>

X-User-Id: 1

Content-Type: application/json

Request Body:

```
{
  "title": "string (max 200)",
  "description": "string (required)",
  "departmentId": 1,
  "categoryId": 5
}
```

Response (201 Created):


```
{
  "success": true,
  "message": "Grievance lodged successfully",
  "data": {
    "id": 1,
    "grievanceNumber": "GRV-2024-000001",
    "status": "SUBMITTED",
    "citizenId": 1,
    "departmentId": 1,
    "categoryId": 5,
    "createdAt": "2024-01-07T10:00:00"
  }
}
```


Business Logic Flow:

1. Validate department and category exist

2. Generate unique grievance number: GRV-{YEAR}-{6-digit-sequence}

3. Create grievance with status = SUBMITTED

4. Record status history

5. Auto-assign to officer with least load

6. Set auto_escalation_time = now + 72 hours

7. Send notification to citizen (RabbitMQ)

8. Send notification to assigned officer (RabbitMQ)

4.3.2 Update Status (Officer/Supervisor)

PATCH /api/grievances/{id}/status

Authorization: Bearer <officer-token>

X-User-Id: 10

Content-Type: application/json

Request Body:

```
{
  "status": "IN_REVIEW" | "RESOLVED",
  "remarks": "string"
}
```

Response (200 OK):

```
{
  "success": true,
  "message": "Status updated successfully",
  "data": null
}
```

Status Transition Rules:

SUBMITTED → ASSIGNED (auto, system)

ASSIGNED → IN_REVIEW (officer)

IN_REVIEW → RESOLVED (officer, requires remarks)

RESOLVED → CLOSED (citizen, via feedback)

RESOLVED → ESCALATED (citizen, if unsatisfied)

ESCALATED → RESOLVED (supervisor)

4.3.3 Escalate Grievance (Citizen)

POST /api/grievances/{id}/escalate

Authorization: Bearer <citizen-token>

X-User-Id: 1

Response (200 OK):

```
{
  "success": true,
  "message": "Grievance escalated successfully",
  "data": null
}
```

Business Rules:

- Only RESOLVED grievances can be escalated
- Only the owner citizen can escalate
- Status changes to ESCALATED
- Notification sent to supervisor

4.3.4 Get Officer Dashboard

GET /api/grievances/dashboard/officer?officerId=10

Authorization: Bearer <officer-token>

Response (200 OK):

```
{
  "success": true,
  "data": {
    "openIssues": 12,
    "assignedToMe": 5,
```

```
    "assignedToMeIds": [1, 5, 8, 12, 20],
    "inReview": 3,
    "resolved": 25,
    "closed": 60
  }
}
```


4.4 Document APIs

4.4.1 Upload Document

```
POST /api/grievances/{id}/documents
Authorization: Bearer <token>
X-User-Id: 1
Content-Type: application/json

Request Body:
{
  "fileName": "complaint_photo.jpg",
  "fileType": "image/jpeg",
  "fileDataBase64": "base64-encoded-string"
}

Response (201 Created):
{
  "success": true,
  "message": "Document uploaded successfully",
  "data": {
    "id": 1,
```

```
  "fileName": "complaint_photo.jpg",
  "fileSize": 245678,
  "uploadedBy": 1,
  "createdAt": "2024-01-07T10:00:00"
 }
}
```


Implementation:

- Decode Base64 to byte[]
- Store in LONGBLOB column
- Max file size: 5MB
- Allowed types: jpg, png, pdf

4.5 Feedback APIs (Base: `/api/feedbacks`)

4.5.1 Submit Feedback

POST /api/feedbacks

Authorization: Bearer <citizen-token>

X-User-Id: 1

Content-Type: application/json

Request Body:


```
{
 "grievanceId": 1,
 "rating": 4,
 "comment": "Good service"
}
```

Response (201 Created):


```
{
  "success": true,
  "message": "Feedback submitted successfully",
  "data": {
    "id": 1,
    "grievanceId": 1,
    "rating": 4
  }
}
```


Business Rules:

- Grievance must be in RESOLVED status
- Only one feedback per grievance
- Rating: 1-5 (integer)
- After feedback, grievance status → CLOSED
