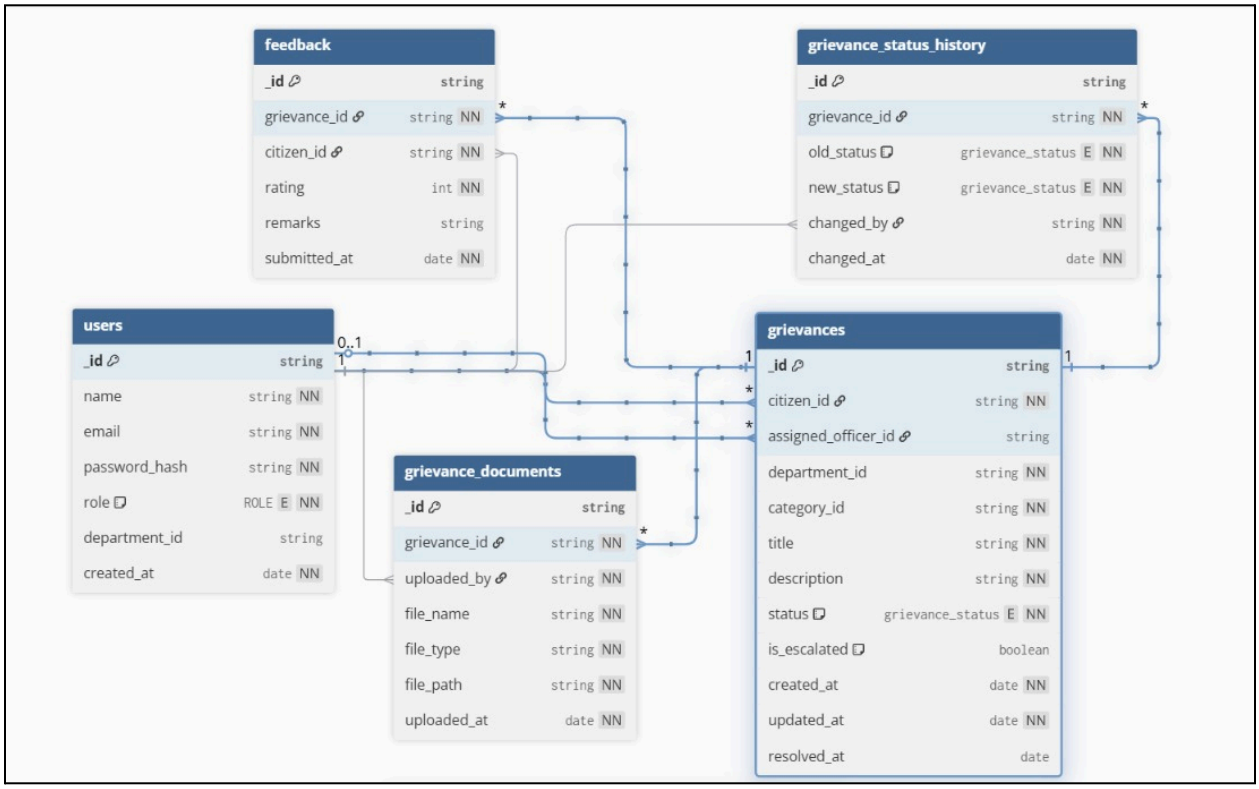
DB Design

E-Governance Grievance Redressal System

https://dbdiagram.io/d/E-Gov-Grievance-Design-69526a7a39fa3db27bbb0903



Enum ROLE {
	ADMIN
	OFFICER
	SUPERVISOR
	CITIZEN
}

Enum grievance_status {
	SUBMITTED
	ASSIGNED
	IN_REVIEW
	ESCALATED
	RESOLVED
	CLOSED
	REOPENED

}

1. Core Data Model Overview

The system consists of the following core collections (as shown in the diagram):

1. `users`
2. `grievances`
3. `grievance_status_history`
4. `grievance_documents`
5. `feedback`

Each collection is described in detail below.

2. Users Collection (`users`)

Purpose

Stores **identity and authorization information** for all system users.

Supported Roles

- `ADMIN`
- `SUPERVISOR`
- `OFFICER`
- `CITIZEN`

Fields Description

Field	Type	Description
`_id`	string	MongoDB ObjectId
`name`	string	User’s full name
`email`	string	Unique login identifier
`password_hash`	string	Encrypted password
`role`	enum	RBAC role
`department_id`	string	Assigned department (Officers only)

`created_at`	date	Account creation timestamp

Business Rules

- Email must be unique
- Passwords are never stored in plain text
- `department_id` is mandatory only for Officers
- Role assignment is restricted to Admins

3. Grievances Collection (`grievances`) – CORE ENTITY

Purpose

Represents a **single grievance** raised by a citizen and tracks its lifecycle.

Fields Description

Field	Type	Description
`_id`	string	Unique grievance identifier
`citizen_id`	string	User who raised the grievance
`department_id`	string	Department handling the grievance
`assigned_officer_id`	string	Officer currently assigned to the grievance (assigned by Supervisor/Admin.)
`category_id`	string	Category of grievance
`title`	string	Short grievance title
`description`	string	Detailed grievance description
`status`	enum	Current grievance status
`is_escalated`	boolean	Escalation indicator
`created_at`	date	Submission time
`updated_at`	date	Last modification time

`resolved_at`	date	Resolution timestamp

Grievance Status Lifecycle

SUBMITTED → ASSIGNED → IN_REVIEW → RESOLVED → CLOSED

↓

ESCALATED

↓

REOPENED

Business Rules

- Only Citizens can create grievances
- Initial status is always `SUBMITTED`
- `resolved_at` is populated only when status = `RESOLVED`
- Grievances are never deleted (immutability for audit)

4. Grievance Status History (`grievance_status_history`)

Purpose

Maintains a **complete audit trail** of all status changes for a grievance.

Fields Description

Field	Type	Description
`_id`	string	Unique history record
`grievance_id`	string	Related grievance
`old_status`	enum	Previous status
`new_status`	enum	Updated status

`changed_by`	string	User who performed the action
`changed_at`	date	Timestamp of change

Business Rules

- Every status change must create a history record
- History records are immutable
- Used for accountability and transparency

5. Grievance Documents (`grievance_documents`)

Purpose

Stores **metadata of documents** uploaded during grievance submission.

Actual files are stored externally (filesystem / cloud storage).

Fields Description

Field	Type	Description
`_id`	string	Document identifier
`grievance_id`	string	Associated grievance
`uploaded_by`	string	User who uploaded
`file_name`	string	Original file name
`file_type`	string	MIME type
`file_path`	string	Storage location
`uploaded_at`	date	Upload timestamp

Business Rules

- Document upload is optional
- Multiple documents per grievance allowed
- Documents are immutable after upload
- Files are accessed only via secured APIs

6. Feedback Collection (`feedback`)

Purpose

Captures **citizen feedback after grievance resolution**.

Fields Description

Field	Type	Description
`_id`	string	Feedback identifier
`grievance_id`	string	Related grievance
`citizen_id`	string	Feedback provider
`rating`	int	Rating (1–5)
`remarks`	string	Optional comments
`submitted_at`	date	Submission time

Business Rules

- Feedback allowed only after `RESOLVED` or `CLOSED`
- One feedback per grievance
- Feedback is immutable
- Used for analytics and performance reports

7. Relationships (Logical, Application-Level)

User → Grievances	One citizen can raise many grievances
Grievance → Documents	One grievance can have multiple documents
Grievance → History	One grievance has many status history records
Grievance → Feedback	One grievance can have only one feedback
User → History	Status changes are user-driven

MongoDB does **not enforce foreign keys** — all relations are validated at application level.

8. SLA & Escalation Design (Derived Data)

- SLA hours are stored in JSON at **category level**
- SLA is evaluated dynamically:
- now −(minus) created_at > slaHours
- No scheduled job is used
- Escalation occurs only when user initiates it
- Escalation sets:
    - `is_escalated = true`
    - `status = ESCALATED`
    - History entry created

9. Why This Design Is Correct

- Clean separation of concerns
- MongoDB-optimized (no joins, reference-based)
- Strong auditability
- Scalable document handling
- Simple SLA logic without schedulers
- Easy to explain in viva and interviews

10. Conclusion

This database design provides a **robust, scalable, and transparent foundation** for an E-Governance Grievance Redressal System.
It balances **real-world architectural best practices** with **academic simplicity**, ensuring clarity, maintainability, and extensibility.