Established – 1961 Subject: OS DBMS

SEVA SADAN'S

R. K. TALREJA COLLEGE

OF

ARTS, SCIENCE & COMMERCE

ULHASNAGAR – 421 003

CERTIFICATE

This is to certify that Mr. Neev Sabban of S.Y. Information Technology Roll No. 2542039 has satisfactorily completed the Open Source DataBase Management System Mini Project on Citizen Grievance Redressal Database during the academic year 2025 – 2026, as a part of the practical requirement. The project work is found to be satisfactory and is approved for submission.

PROF. INCHARGE	HEAD OF DEPT
Prof. Sahil Shukla	

INDEX

1. INTRODUCTION

A Citizen Grievance Redressal System (CGRS) is a fundamental component of e-Governance, providing citizens with a structured and formal channel to lodge complaints against government and municipal services. Government bodies are responsible for offering a reliable mechanism for citizens to report grievances related to Public Services (such as Public Education, Public Safety, and Public Health) and Civic Services (such as Water Supply, Electricity, Road Maintenance, Sanitation, Public Transportation, Healthcare, and Municipal Administration).

As urbanisation increases and population continues to grow, the volume of grievances filed by citizens rises substantially. Manually processing these grievances, or using unorganised systems such as registers and spreadsheets, leads to data loss, duplication of records, inconsistent status updates, and delayed resolutions. A properly designed Database Management System (DBMS) is therefore essential to store, manage, and retrieve grievance-related data in an efficient and reliable manner.

This project has been designed and implemented using MySQL to develop a structured, normalised Citizen Grievance Redressal Database. The disorganised grievance data has been transformed into a relational database by applying normalisation rules — Unnormalized Form, First Normal Form (1NF), and a fully normalised relational structure — to eliminate data redundancy, preserve data integrity, prevent anomalies, and improve overall database performance.

MySQL is selected as the database platform because it is open-source, reliable, widely used in academic and government applications, and supports SQL queries, constraints, transaction control, indexing, and security mechanisms. This project provides practical exposure to DBMS concepts and demonstrates how proper database design improves storage efficiency, data accuracy, and query performance in real-world e-Governance applications.

2. PROBLEM DEFINITION

Many government bodies and municipal corporations continue to maintain citizen grievance records using manual registers or unstructured spreadsheets. These traditional approaches result in several critical problems that reduce the efficiency and transparency of the grievance redressal process:

- Data Redundancy – Citizen names, contact details, and department information are repeatedly stored in multiple records, consuming unnecessary storage space.
- Data Inconsistency – When grievance status is updated in one record but not in others, the data becomes inconsistent and unreliable.
- Update Anomaly – Changing a department name or officer assignment requires updates across multiple rows, increasing the risk of errors.
- Insertion Anomaly – A new grievance cannot be registered without having complete citizen and department information available simultaneously.
- Deletion Anomaly – Deleting a grievance record may unintentionally remove important citizen or department information from the system.
- Multi-Valued Attributes – Storing multiple grievances for a single citizen in one field (comma-separated) prevents proper querying and analysis.
- Slow Query Performance – Large, unstructured tables make it difficult to generate status-wise or department-wise reports efficiently.
- Lack of Audit Trail – Without proper status history tracking, it is impossible to know when a grievance was last updated and by whom.

These issues highlight the urgent need for a structured, normalised relational database that properly organises citizen, grievance, department, officer, and status data into separate related tables — ensuring consistency, transparency, and efficiency in grievance redressal.

3. OBJECTIVES OF THE PROJECT

The main objectives of this project are:

- To design and implement a structured Citizen Grievance Redressal Database using MySQL.
- To apply database normalisation rules (Unnormalized Form to 1NF to a fully normalised relational structure) to organise grievance data properly.
- To eliminate data redundancy and prevent update, insertion, and deletion anomalies.
- To create multiple related tables (Citizens, Departments, Grievances, Officers, StatusHistory) connected using primary keys and foreign keys.
- To implement SQL queries for data manipulation including INSERT, UPDATE, DELETE, and SELECT operations.
- To generate department-wise and status-wise grievance reports using GROUP BY and JOIN operations.
- To enforce data integrity through constraints such as PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, and DEFAULT.
- To implement role-based database security using GRANT and REVOKE commands.
- To set up backup and recovery mechanisms using mysqldump to prevent data loss.
- To gain practical experience with Database Management System (DBMS) concepts applied in a real-world e-Governance scenario.

4. SCOPE OF THE PROJECT

The scope of the Citizen Grievance Redressal Database project includes the following areas:

- Conversion of unnormalised data into structured tables – The project transforms a single flat grievance table containing repeated and multi-valued data into multiple related, normalised tables.
- Application of normalisation rules – The data passes through Unnormalised Form (UNF), First Normal Form (1NF), and a fully normalised relational structure to ensure data integrity and eliminate redundancy.
- Creation of entities and relationships – Tables such as Citizens, Departments, Grievances, Officers, and StatusHistory are created and connected using primary and foreign keys to model real-world grievance workflows.
- Implementation of SQL operations – SQL commands including CREATE, INSERT, UPDATE, DELETE, and SELECT are used to define and manage the database. Advanced operations like JOIN queries and GROUP BY reports are also implemented.
- Use of constraints – PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, and DEFAULT constraints are applied to maintain data accuracy and referential integrity.
- Database security – Role-based access control is implemented using GRANT and REVOKE commands to restrict unauthorized access.
- Backup and recovery – Regular backups using mysqldump and table-level backup strategies are implemented to safeguard data.

This project is limited to backend database design using MySQL and is intended primarily for academic learning. It does not include a frontend user interface. The database is designed to handle small to medium datasets (100 to 1,000 records) and is structured to be scalable for future enhancements.

5. REQUIREMENT SPECIFICATION

a. Hardware Requirements

Component	Specification
Output Devices	Monitor / Laptop
Input Devices	Keyboard, Mouse
Processor	Intel i3 or above
RAM	Minimum 4 GB
Hard Disk	Minimum 20 GB

b. Software Requirements

Software	Purpose
MySQL Server 8.0	Database creation and management
MySQL Workbench	Query execution and schema design interface
Windows OS	Operating platform
SQL	Query language for data operations

6. SYSTEM DESIGN

6.1 System Architecture

The Citizen Grievance Redressal Database follows a simple database-centred architecture. The user interacts directly with the database using SQL queries executed through MySQL Workbench. There is no frontend interface; the entire system operates at the backend level.

The core component of the system is MySQL Server, which is responsible for storing, managing, and processing all citizen, grievance, department, officer, and status history data. The server ensures that data is stored in normalised tables and maintains relationships using primary keys and foreign keys.

When a user executes an SQL command such as INSERT, UPDATE, DELETE, or SELECT, the query is sent to the MySQL Server. The server validates the query syntax, checks constraints, verifies relationships between tables, and processes the request. If the query violates any constraint (for example, a duplicate primary key or an invalid foreign key reference), the server rejects the operation to maintain data integrity.

The architecture ensures:

- Proper organisation of grievance data into multiple related tables
- Elimination of redundancy through normalisation
- Enforcement of data integrity using constraints
- Secure access control through user privileges
- Reliable transaction execution using COMMIT and ROLLBACK

6.2 Data Flow

The data flow in the system is straightforward. The citizen's grievance information flows through the following stages:

1. Citizen data is entered into the Citizens table with a unique citizen_id.
2. Department and Officer data are pre-populated in the Departments and Officers tables.
3. A grievance is submitted by linking citizen_id and dept_id in the Grievances table with a description, category, and submit_date.
4. Officers update the status of grievances, and each update is recorded in the StatusHistory table with remarks and update_date.
5. Administrators can query the database to generate department-wise and status-wise reports using JOIN and GROUP BY operations.

6.3 System Architecture Flow

Step	Component	Role
1	User	Enters SQL query in MySQL Workbench
2	MySQL Workbench	Sends query to MySQL Server
3	MySQL Server	Validates syntax and checks constraints
4	GrievanceDB	Executes query on the database tables
5	MySQL Server	Returns result to Workbench
6	User	Views result in tabular form

7. DATABASE DESIGN

Each entity represents a separate real-world object involved in the citizen grievance redressal process. The database is divided into multiple related tables to maintain a normalised structure and eliminate data redundancy.

7.1 Entity Description

- Citizens – Stores details of citizens who file grievances. Each citizen is uniquely identified by citizen_id and contains their name, contact number, and address.
- Departments – Contains information about government departments responsible for resolving specific types of grievances. Each department is identified by dept_id and includes dept_name and location.
- Grievances – The core table that records each complaint filed by a citizen against a specific department. It stores grievance_id, references to citizen_id and dept_id, a description, category, submit_date, and current status.
- Officers – Stores information about government officers assigned to handle grievances within specific departments. Each officer is linked to a department via dept_id.
- StatusHistory – An audit table that tracks every status update made to a grievance. It records the officer who made the update, the new status, date of update, and remarks.

7.2 Table Structure

1. Citizens Table

Attribute	Data Type	Constraint
citizen_id	INT	PRIMARY KEY
citizen_name	VARCHAR(100)	NOT NULL
contact	VARCHAR(15)	NOT NULL, UNIQUE
address	VARCHAR(200)	NOT NULL

2. Departments Table

Attribute	Data Type	Constraint
dept_id	INT	PRIMARY KEY
dept_name	VARCHAR(100)	NOT NULL
location	VARCHAR(100)	NOT NULL

3. Grievances Table

Attribute	Data Type	Constraint
grievance_id	INT	PRIMARY KEY, AUTO_INCREMENT
citizen_id	INT	FOREIGN KEY → Citizens
dept_id	INT	FOREIGN KEY → Departments
description	VARCHAR(500)	NOT NULL
category	VARCHAR(50)	NOT NULL
submit_date	DATE	NOT NULL
status	VARCHAR(30)	DEFAULT 'Pending'

4. Officers Table

Attribute	Data Type	Constraint
officer_id	INT	PRIMARY KEY, AUTO_INCREMENT
officer_name	VARCHAR(100)	NOT NULL
dept_id	INT	FOREIGN KEY → Departments

5. StatusHistory Table

Attribute	Data Type	Constraint
history_id	INT	PRIMARY KEY, AUTO_INCREMENT
grievance_id	INT	FOREIGN KEY → Grievances
officer_id	INT	FOREIGN KEY → Officers
status	VARCHAR(30)	NOT NULL
update_date	DATE	NOT NULL
remarks	VARCHAR(300)	NOT NULL

7.3 Constraints Used

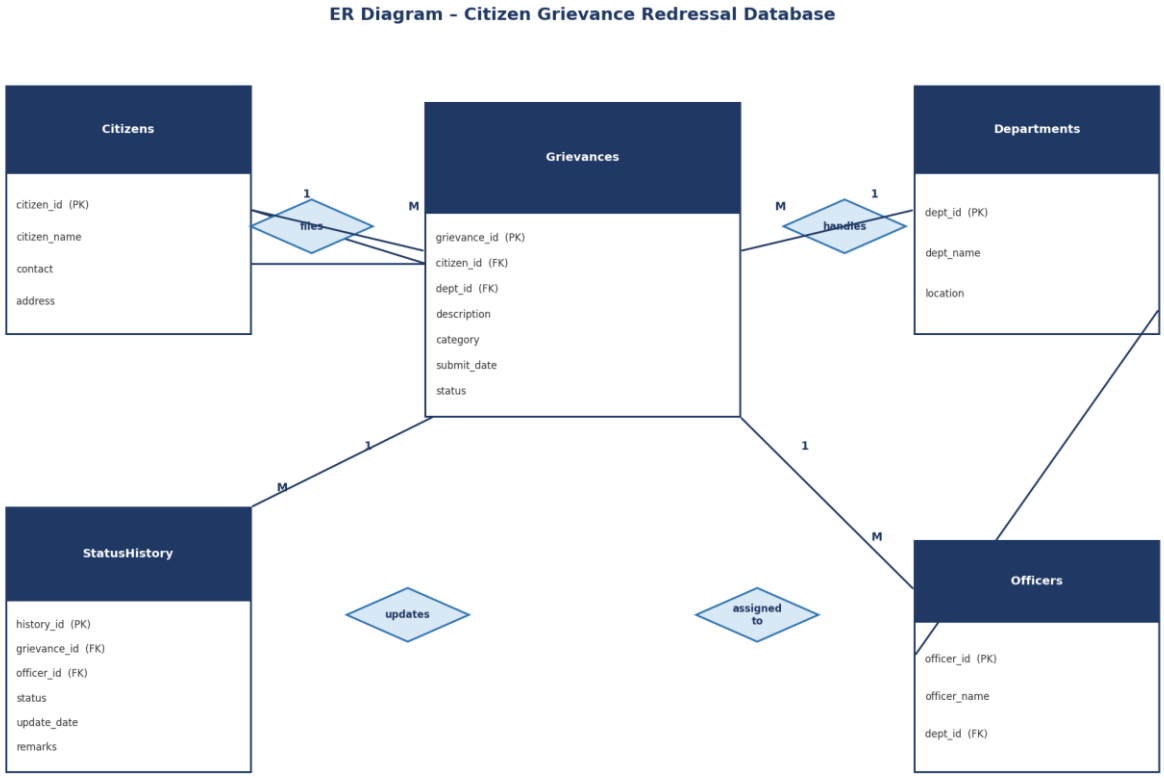
In this project, various constraints are applied in MySQL to maintain accuracy, consistency, and proper relationships between tables.

- PRIMARY KEY – Uniquely identifies each record in a table. citizen_id, dept_id, grievance_id, officer_id, and history_id serve as primary keys, ensuring no duplicate records exist in any table.
- FOREIGN KEY – Establishes relationships between tables. For example, citizen_id and dept_id in the Grievances table reference the Citizens and Departments tables respectively, ensuring referential integrity and preventing orphan records.
- NOT NULL – Ensures that important fields such as citizen_name, description, category, submit_date, and remarks cannot be left empty, guaranteeing essential information is always stored.
- UNIQUE – Prevents duplicate values in the contact field of the Citizens table, ensuring no two citizens share the same mobile number.
- DEFAULT – The status field in the Grievances table is assigned a default value of 'Pending' when a new grievance is filed, reducing the need for manual status entry.
- AUTO_INCREMENT – Applied to grievance_id, officer_id, and history_id to automatically generate sequential unique identifiers without manual input.

8. UML DIAGRAMS

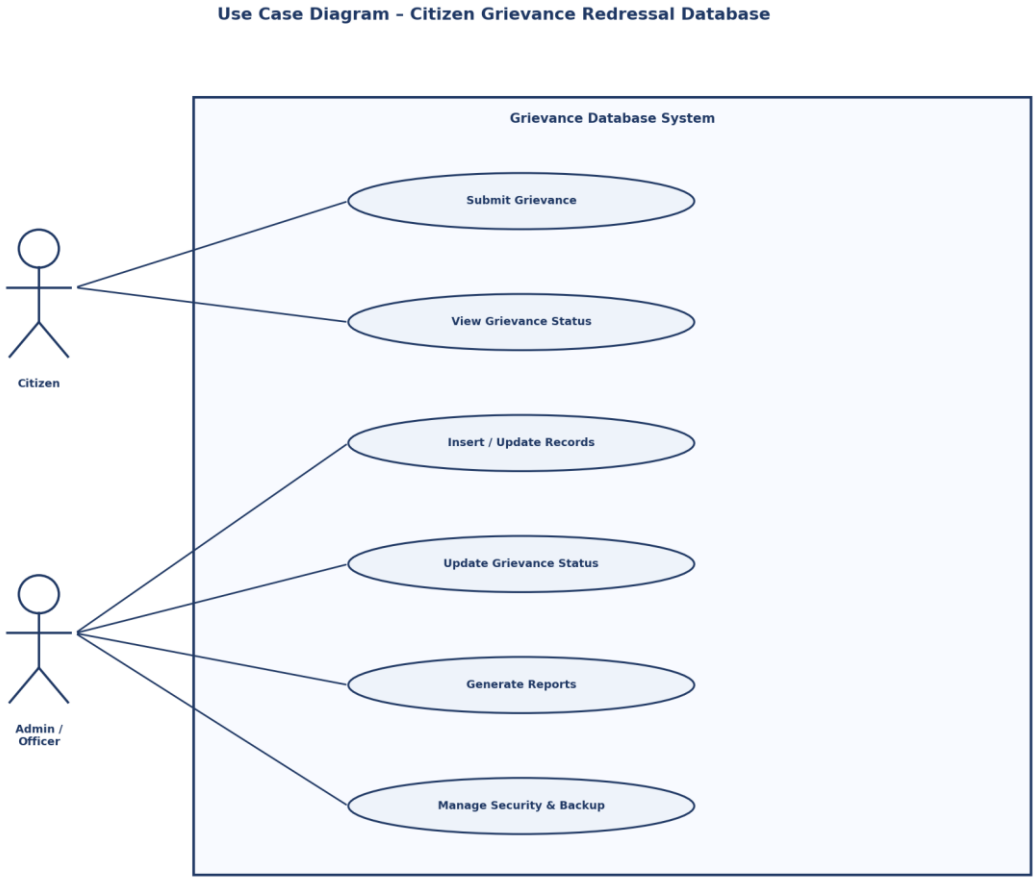
8.1 ER Diagram

The ER Diagram shows the entities involved in the Citizen Grievance Redressal System and their relationships. Each entity is represented as a box with its attributes listed inside. Diamonds represent relationships between entities, and cardinality labels (1 and M) indicate one-to-many relationships.



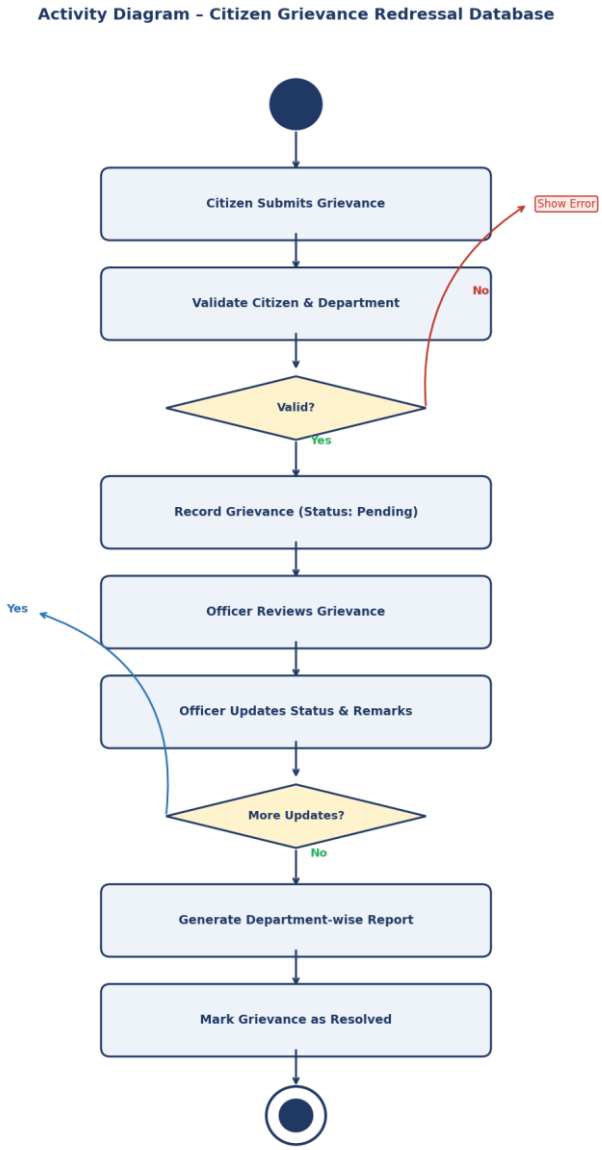
8.2 Use Case Diagram

The Use Case Diagram illustrates the interactions between actors (Citizen and Admin/Officer) and the Citizen Grievance Redressal Database System. Stick figures represent the actors, ovals represent use cases, and the rectangle represents the system boundary.



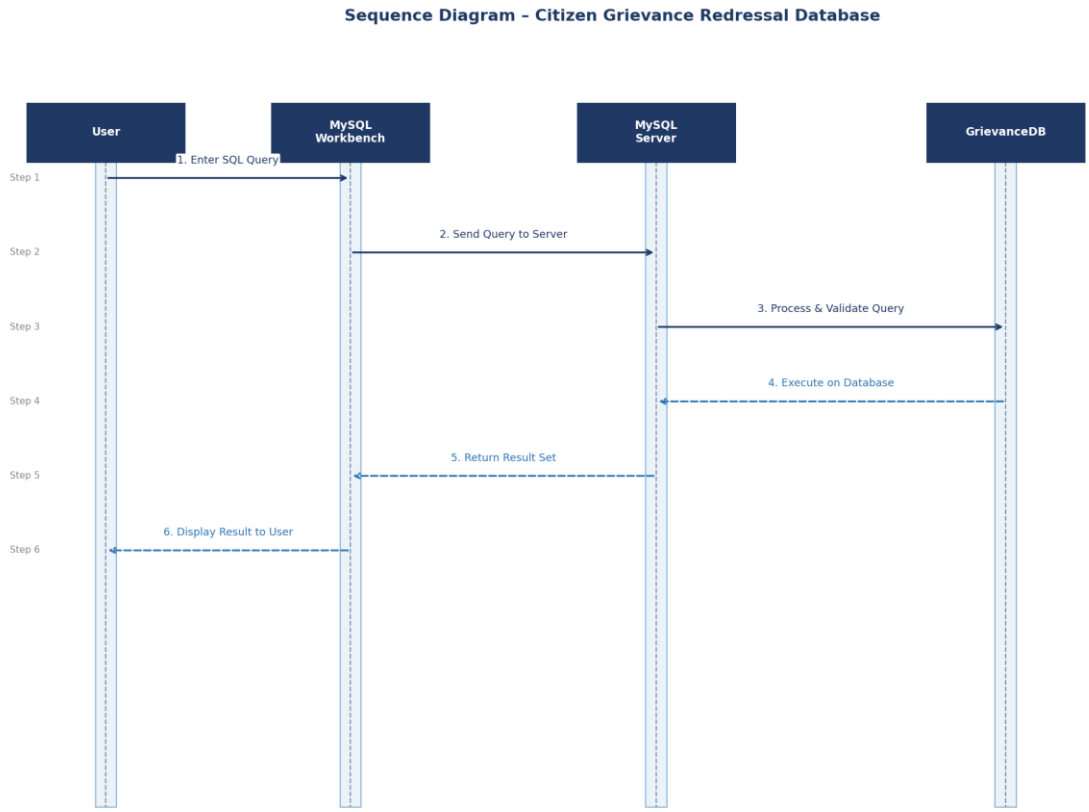
8.3 Activity Diagram

The Activity Diagram shows the complete workflow of the Citizen Grievance Redressal process. A filled circle represents the start, rounded rectangles represent activities, diamonds represent decision points, and a bull's-eye represents the end state. Arrows show the flow direction.



8.4 Sequence Diagram

The Sequence Diagram shows the step-by-step query execution sequence between the User, MySQL Workbench, MySQL Server, and GrievanceDB. Boxes at the top represent components, dashed vertical lines are lifelines, and horizontal arrows show messages passing between components.



9. SQL IMPLEMENTATION

The SQL implementation of the Citizen Grievance Redressal Database was performed using MySQL. The implementation demonstrates how unnormalised grievance data is converted into a proper normalised relational database structure by applying normalisation rules.

9.1 Database Creation

```
CREATE DATABASE GrievanceDB;
USE GrievanceDB;
```

9.2 Unnormalised Table

Initially all grievance-related information was stored in a single table containing repeated and multi-valued data. This structure causes data redundancy, update anomalies, and multi-valued attributes stored in one column.

```
CREATE TABLE grievance_unnormalized (
  citizen_id      INT,
  citizen_name    VARCHAR(100),
  contact         VARCHAR(15),
  department_name VARCHAR(100),
  officer_name    VARCHAR(100),
  grievances      VARCHAR(500)
);

INSERT INTO grievance_unnormalized VALUES
(1,'Ramesh','9876543210','Water Dept','Sharma','No water supply, Pipe burst'),
(1,'Ramesh','9876543210','Road Dept','Verma','Pothole on main road'),
(2,'Sunil','9123456780','Water Dept','Sharma','Dirty water issue');
```

9.3 Conversion to First Normal Form (1NF)

To achieve 1NF, repeating groups and multi-valued attributes were removed. Each grievance is now stored as a separate atomic row:

```
CREATE TABLE grievance_1nf (
  citizen_id      INT,
  citizen_name    VARCHAR(100),
  department_name VARCHAR(100),
  officer_name    VARCHAR(100),
  grievance       VARCHAR(300)
);

INSERT INTO grievance_1nf VALUES
(1,'Ramesh','Water Dept','Sharma','No water supply'),
(1,'Ramesh','Water Dept','Sharma','Pipe burst'),
(1,'Ramesh','Road Dept','Verma','Pothole on main road'),
(2,'Sunil','Water Dept','Sharma','Dirty water issue');
```

9.4 Final Normalised Structure

Partial and transitive dependencies are removed by creating separate related tables connected through foreign keys:

```
CREATE TABLE citizens (
  citizen_id   INT NOT NULL PRIMARY KEY,
  citizen_name VARCHAR(100) NOT NULL,
  contact      VARCHAR(15)  NOT NULL UNIQUE,
  address      VARCHAR(200) NOT NULL
);

CREATE TABLE departments (
  dept_id   INT NOT NULL PRIMARY KEY,
  dept_name VARCHAR(100) NOT NULL,
  location  VARCHAR(100) NOT NULL
);

CREATE TABLE officers (
  officer_id   INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  officer_name VARCHAR(100) NOT NULL,
  dept_id      INT,
  FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);
```

```
CREATE TABLE grievances (
  grievance_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  citizen_id   INT,
  dept_id      INT,
  description  VARCHAR(500) NOT NULL,
  category     VARCHAR(50)  NOT NULL,
  submit_date  DATE         NOT NULL,
  status       VARCHAR(30)  DEFAULT 'Pending',
  FOREIGN KEY (citizen_id) REFERENCES citizens(citizen_id),
  FOREIGN KEY (dept_id)    REFERENCES departments(dept_id)
);

CREATE TABLE statushistory (
  history_id   INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  grievance_id INT,
  officer_id   INT,
  status       VARCHAR(30)  NOT NULL,
  update_date  DATE         NOT NULL,
  remarks      VARCHAR(300) NOT NULL,
  FOREIGN KEY (grievance_id) REFERENCES grievances(grievance_id),
  FOREIGN KEY (officer_id)   REFERENCES officers(officer_id)
);
```

9.5 Data Retrieval Using JOIN

```
-- View all grievances with citizen name, department and status
SELECT c.citizen_name, d.dept_name, g.description,
       g.category, g.submit_date, g.status
FROM grievances g
JOIN citizens    c ON g.citizen_id = c.citizen_id
JOIN departments d ON g.dept_id    = d.dept_id;

-- Department-wise grievance count report
SELECT d.dept_name, COUNT(g.grievance_id) AS total_grievances
FROM grievances g
JOIN departments d ON g.dept_id = d.dept_id
GROUP BY d.dept_name;

-- Status-wise grievance count
SELECT status, COUNT(*) AS total FROM grievances GROUP BY status;
```

9.6 Transaction Management

```
START TRANSACTION;
INSERT INTO citizens VALUES (5,'Priya','9000111222','Thane');
INSERT INTO grievances(citizen_id,dept_id,description,category,submit_date)
  VALUES (5,2,'Road broken near school','Road','2026-02-01');
COMMIT;

-- Rollback on error
START TRANSACTION;
UPDATE grievances SET status='Resolved' WHERE grievance_id=10;
ROLLBACK;
```

Transaction control ensures data consistency when multiple related operations must all succeed or all fail together.

10. SYSTEM TESTING AND RESULT

10.1 Database Creation Test

Definition: Verifies whether the GrievanceDB database is created successfully and is usable.

Expected Result: Database created and selected without errors.

```
mysql> CREATE DATABASE GrievanceDB;
Query OK, 1 row affected (0.055 sec)

mysql> USE GrievanceDB;
Database changed
```

10.2 Unnormalised Table Testing

Definition: Tests creation of the unnormalised table containing redundant and multi-valued data.

Expected Result: Table created successfully; data inserted shows redundancy.

```
mysql> DESC grievance_unnormalized;
+----------------+--------------+------+
| Field          | Type         | Null |
+----------------+--------------+------+
| citizen_id     | int          | YES  |
| citizen_name   | varchar(100) | YES  |
| contact        | varchar(15)  | YES  |
| department_name| varchar(100) | YES  |
| officer_name   | varchar(100) | YES  |
| grievances     | varchar(500) | YES  |
+----------------+--------------+------+
6 rows in set (0.068 sec)
```

10.3 1NF Testing

Definition: Tests that multi-valued grievances column is split into atomic single-value rows.

Expected Result: Each grievance appears in a separate row.

```
mysql> SELECT * FROM grievance_1nf;
+------------+--------------+-----------------+--------------+-------------------+
| citizen_id | citizen_name | department_name | officer_name | grievance         |
+------------+--------------+-----------------+--------------+-------------------+
|          1 | Ramesh       | Water Dept      | Sharma       | No water supply   |
|          1 | Ramesh       | Water Dept      | Sharma       | Pipe burst        |
|          1 | Ramesh       | Road Dept       | Verma        | Pothole on road   |
|          2 | Sunil        | Water Dept      | Sharma       | Dirty water issue |
+------------+--------------+-----------------+--------------+-------------------+
4 rows in set (0.008 sec)
```

10.4 Primary Key Constraint Testing

Definition: Verifies that duplicate primary keys are rejected by the database.

Expected Result: Second INSERT fails due to duplicate citizen_id.

```
mysql> INSERT INTO citizens VALUES (1,'Ramesh','9876543210','Mumbai');
Query OK, 1 row affected (0.048 sec)

mysql> INSERT INTO citizens VALUES (1,'Test','9999999999','Delhi');
ERROR 1062 (23000): Duplicate entry '1' for key 'citizens.PRIMARY'
```

10.5 Foreign Key Constraint Testing

Definition: Ensures grievances cannot be filed against a non-existent department.

Expected Result: INSERT fails because dept_id 99 does not exist.

```
mysql> INSERT INTO grievances(citizen_id,dept_id,description,category,submit_date)
       VALUES (1,99,'Test grievance','Test','2026-01-01');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key
constraint fails (grievances, FOREIGN KEY (dept_id) REFERENCES departments)
```

10.6 UNIQUE Constraint Testing

Definition: Verifies that duplicate contact numbers are not allowed in the Citizens table.

Expected Result: Second INSERT fails due to duplicate contact number.

```
mysql> INSERT INTO citizens VALUES (5,'Rohan','9876543210','Nagpur');
ERROR 1062 (23000): Duplicate entry '9876543210' for key 'citizens.contact'
```

10.7 Status Update Testing

Definition: Tests updating grievance status and inserting a corresponding status history record.

Expected Result: Status updated and StatusHistory record inserted successfully.

```
mysql> UPDATE grievances SET status='In Progress' WHERE grievance_id=1;
Query OK, 1 row affected (0.032 sec)

mysql> INSERT INTO statushistory(grievance_id,officer_id,status,update_date,remarks)
       VALUES (1,1,'In Progress','2026-01-13','Officer assigned, work started');
Query OK, 1 row affected (0.018 sec)
```

10.8 JOIN Query Testing

Definition: Tests whether normalised tables correctly retrieve combined data using JOIN.

Expected Result: Correct combined data displayed without redundancy.

```
mysql> SELECT c.citizen_name, d.dept_name, g.description, g.status
       FROM grievances g
       JOIN citizens    c ON g.citizen_id = c.citizen_id
       JOIN departments d ON g.dept_id    = d.dept_id;
+--------------+------------------------+----------------------------+-------------+
| citizen_name | dept_name              | description                | status      |
+--------------+------------------------+----------------------------+-------------+
| Ramesh       | Water Department       | No water supply in area    | Pending     |
| Ramesh       | Road Department        | Pothole on main road       | In Progress |
| Sunil        | Water Department       | Dirty water issue          | Resolved    |
| Anjali       | Electricity Department | Power cut for 5 hours      | Pending     |
+--------------+------------------------+----------------------------+-------------+
4 rows in set (0.007 sec)
```

10.9 GROUP BY Report Testing

Definition: Tests department-wise grievance count report using GROUP BY and aggregate functions.

Expected Result: Total grievances per department displayed correctly.

```
mysql> SELECT d.dept_name, COUNT(g.grievance_id) AS total_grievances
       FROM grievances g
       JOIN departments d ON g.dept_id = d.dept_id
       GROUP BY d.dept_name;
+------------------------+-----------------+
| dept_name              | total_grievances |
+------------------------+-----------------+
| Water Department       |               2 |
```

```
| Road Department        |                1 |
| Electricity Department |                1 |
+------------------------+------------------+
3 rows in set (0.005 sec)
```

10.10 Final Result

After applying normalisation and implementing the complete database structure, the following results were observed:

- Data redundancy removed – Citizen and department data is stored once and referenced via foreign keys.
- Update anomaly eliminated – Changing a department name only requires one update in the Departments table.
- Referential integrity maintained – Foreign key constraints prevent invalid citizen or department references.
- Audit trail established – StatusHistory records every grievance update with officer details and remarks.
- Structured relational database achieved – All five normalised tables exist and function correctly.

```
mysql> SHOW TABLES;
+---------------------------+
| Tables_in_grievancedb     |
+---------------------------+
| citizens                  |
| departments               |
| grievance_1nf             |
| grievance_unnormalized    |
| grievances                |
| officers                  |
| statushistory             |
+---------------------------+
7 rows in set (0.183 sec)
```

11. SECURITY, BACKUP AND RECOVERY

11.1 Database Security

Database security ensures that only authorised users can access and modify grievance data. Access control is implemented using MySQL's built-in user management and privilege system.

User Authentication

```
CREATE USER 'grievance_admin'@'localhost' IDENTIFIED BY 'Admin@2026';
CREATE USER 'grievance_viewer'@'localhost' IDENTIFIED BY 'View@2026';
```

Granting Privileges

```
GRANT ALL PRIVILEGES ON GrievanceDB.* TO 'grievance_admin'@'localhost';
GRANT SELECT ON GrievanceDB.* TO 'grievance_viewer'@'localhost';
FLUSH PRIVILEGES;
```

Revoking Privileges

```
REVOKE INSERT, UPDATE, DELETE ON GrievanceDB.* FROM 'grievance_viewer'@'localhost';
```

This ensures only authorised users can modify grievance tables, viewer users can only read data, and data integrity is maintained through controlled access.

11.2 Backup Strategy

Full Database Backup

```
mysqldump -u root -p GrievanceDB > grievance_backup.sql
```

Table-Level Backup

```
CREATE TABLE citizens_backup     AS SELECT * FROM citizens;
CREATE TABLE grievances_backup   AS SELECT * FROM grievances;
CREATE TABLE statushistory_backup AS SELECT * FROM statushistory;
```

11.3 Recovery Process

Restore Full Database

```
mysql -u root -p GrievanceDB < grievance_backup.sql
```

Restore From Backup Table

```
INSERT INTO citizens SELECT * FROM citizens_backup
WHERE citizen_id NOT IN (SELECT citizen_id FROM citizens);
```

11.4 Transaction Control for Recovery

```
START TRANSACTION;
DELETE FROM grievances WHERE status = 'Resolved';
ROLLBACK;   -- Restores all deleted records if error occurs
COMMIT;     -- Confirms the operation if everything is correct
```

After implementing security, backup, and recovery: database access is restricted to authorised users, all data changes are transactional and recoverable, backup copies are maintained, and the normalised structure remains consistent and secure.

12. FUTURE SCOPE AND CONCLUSION

12.1 Future Scope

- Development of a web or mobile application frontend to allow citizens to submit and track grievances online without needing direct database access.
- Implementation of an SMS and Email notification system to automatically alert citizens when their grievance status is updated.
- Integration of stored procedures and triggers to automate routine operations such as auto-assigning officers and sending alerts on overdue grievances.
- Development of analytical dashboards for government authorities to monitor department performance, average resolution time, and grievance trends.
- Extension to higher normal forms such as BCNF and 4NF for even stricter data integrity.
- Cloud deployment on platforms such as AWS RDS or Google Cloud SQL for scalability and remote access.

12.2 Conclusion

The Citizen Grievance Redressal Database project successfully demonstrates the practical implementation of Database Management System (DBMS) concepts and normalisation techniques in a real-world e-Governance scenario. By transforming unnormalised grievance data into a structured, relational database using MySQL, the project has achieved its core objectives of eliminating redundancy, maintaining data integrity, and improving query performance.

The five-table normalised structure — Citizens, Departments, Grievances, Officers, and StatusHistory — efficiently models the entire grievance lifecycle from submission to resolution. The use of primary keys, foreign keys, constraints, JOIN queries, and GROUP BY reports demonstrates how a well-designed database can improve transparency and accountability in public grievance management systems.

This project strengthens the understanding of structured database design and its critical importance in building reliable, scalable, and secure e-Governance applications.

13. REFERENCES

6. MySQL Official Documentation – Oracle Corporation. https://dev.mysql.com/doc/
7. Database Management System – Textbooks prescribed by Mumbai University for S.Y. Information Technology.
8. E-Governance and Public Administration Study Materials – Ministry of Electronics and Information Technology (MeitY), Government of India.
9. Fundamentals of Database Systems – Ramez Elmasri and Shamkant B. Navathe.
10. Online Resources – W3Schools SQL Tutorial (https://www.w3schools.com/sql/) and TutorialsPoint DBMS Guide.

14. GLOSSARY

Term	Definition
DBMS	Database Management System – Software used to store, manage, and retrieve data efficiently in the form of relational databases.
SQL	Structured Query Language – Standard language used to create, insert, update, delete, and retrieve data from a database.
Normalisation	The process of organising a database to reduce redundancy and improve data integrity by applying normal form rules.
Primary Key	A unique identifier for each record in a table that does not allow duplicate or NULL values.
Foreign Key	A field in a table that references the primary key of another table to maintain referential integrity.
1NF	First Normal Form – Ensures all column values are atomic with no multi-valued attributes or repeating groups.
Grievance	A formal complaint filed by a citizen against a government department regarding a civic service issue.
StatusHistory	An audit table that records every status change of a grievance including the updating officer and remarks.
mysqldump	A MySQL utility used to create a complete backup of a database in the form of SQL statements.
GRANT/REVOKE	SQL commands used to assign or remove access privileges from database users.

15. APPENDIX / SQL CODE

The following is the complete SQL code executed using MySQL to implement the Citizen Grievance Redressal Database.

```
-- ============================================================
-- CITIZEN GRIEVANCE REDRESSAL DATABASE
-- Student  : Neev Sabban
-- Roll No  : 2542039
-- College  : R.K. Talreja College, Ulhasnagar
-- Subject  : Open Source Database Management System
-- Year     : 2025 - 2026
-- ============================================================

CREATE DATABASE GrievanceDB;
USE GrievanceDB;

-- UNNORMALIZED TABLE
CREATE TABLE grievance_unnormalized (
    citizen_id      INT,
    citizen_name    VARCHAR(100),
    contact         VARCHAR(15),
    department_name VARCHAR(100),
    officer_name    VARCHAR(100),
    grievances      VARCHAR(500)
);

INSERT INTO grievance_unnormalized VALUES
(1, 'Ramesh', '9876543210', 'Water Dept', 'Sharma', 'No water supply,
Pipe burst'),
(1, 'Ramesh', '9876543210', 'Road Dept',  'Verma',  'Pothole on main
road'),
(2, 'Sunil',  '9123456780', 'Water Dept', 'Sharma', 'Dirty water
issue');

-- 1NF TABLE
CREATE TABLE grievance_1nf (
    citizen_id      INT,
    citizen_name    VARCHAR(100),
    department_name VARCHAR(100),
    officer_name    VARCHAR(100),
    grievance       VARCHAR(300)
);

INSERT INTO grievance_1nf VALUES
(1, 'Ramesh', 'Water Dept', 'Sharma', 'No water supply'),
(1, 'Ramesh', 'Water Dept', 'Sharma', 'Pipe burst'),
(1, 'Ramesh', 'Road Dept',  'Verma',  'Pothole on main road'),
(2, 'Sunil',  'Water Dept', 'Sharma', 'Dirty water issue');

-- NORMALIZED TABLES
CREATE TABLE citizens (
    citizen_id   INT          NOT NULL,
    citizen_name VARCHAR(100) NOT NULL,
    contact      VARCHAR(15)  NOT NULL,
    address      VARCHAR(200) NOT NULL,
    PRIMARY KEY (citizen_id),
    UNIQUE KEY (contact)
);
```

```
INSERT INTO citizens VALUES
(1, 'Ramesh', '9876543210', 'Mumbai'),
(2, 'Sunil',  '9123456780', 'Pune'),
(3, 'Anjali', '9001122334', 'Nashik'),
(4, 'Rohan',  '9777888999', 'Nagpur');

CREATE TABLE departments (
    dept_id   INT          NOT NULL,
    dept_name VARCHAR(100) NOT NULL,
    location  VARCHAR(100) NOT NULL,
    PRIMARY KEY (dept_id)
);

INSERT INTO departments VALUES
(1, 'Water Department',       'Mumbai'),
(2, 'Road Department',        'Mumbai'),
(3, 'Electricity Department', 'Pune');

CREATE TABLE officers (
    officer_id   INT          NOT NULL AUTO_INCREMENT,
    officer_name VARCHAR(100) NOT NULL,
    dept_id      INT,
    PRIMARY KEY (officer_id),
    FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);

INSERT INTO officers VALUES (1,'Sharma',1),(2,'Verma',2),(3,'Patil',3);

CREATE TABLE grievances (
    grievance_id INT          NOT NULL AUTO_INCREMENT,
    citizen_id   INT,
    dept_id      INT,
    description  VARCHAR(500) NOT NULL,
    category     VARCHAR(50)  NOT NULL,
    submit_date  DATE         NOT NULL,
    status       VARCHAR(30)  DEFAULT 'Pending',
    PRIMARY KEY (grievance_id),
    FOREIGN KEY (citizen_id) REFERENCES citizens(citizen_id),
    FOREIGN KEY (dept_id)    REFERENCES departments(dept_id)
);

INSERT INTO grievances VALUES
(1,1,1,'No water supply in area','Water','2026-01-10','Pending'),
(2,1,2,'Pothole on main road','Road','2026-01-12','In Progress'),
(3,2,1,'Dirty water issue','Water','2026-01-15','Resolved'),
(4,3,3,'Power cut for 5 hours daily','Electricity','2026-01-
20','Pending');

CREATE TABLE statushistory (
    history_id   INT          NOT NULL AUTO_INCREMENT,
    grievance_id INT,
    officer_id   INT,
    status       VARCHAR(30)  NOT NULL,
    update_date  DATE         NOT NULL,
    remarks      VARCHAR(300) NOT NULL,
    PRIMARY KEY (history_id),
    FOREIGN KEY (grievance_id) REFERENCES grievances(grievance_id),
    FOREIGN KEY (officer_id)   REFERENCES officers(officer_id)
);
```

```
INSERT INTO statushistory VALUES
(1,1,1,'Pending','2026-01-10','Grievance received'),
(2,2,2,'In Progress','2026-01-13','Team assigned for road repair'),
(3,3,1,'Resolved','2026-01-18','Water pipe cleaned and fixed'),
(4,4,3,'Pending','2026-01-20','Under review');

-- QUERIES
SELECT c.citizen_name, d.dept_name, g.description, g.status
FROM grievances g
JOIN citizens    c ON g.citizen_id = c.citizen_id
JOIN departments d ON g.dept_id    = d.dept_id;

SELECT d.dept_name, COUNT(g.grievance_id) AS total_grievances
FROM grievances g
JOIN departments d ON g.dept_id = d.dept_id
GROUP BY d.dept_name;

SELECT status, COUNT(*) AS total FROM grievances GROUP BY status;

SHOW TABLES;
```