SERVICE BOOKING AND COMPLAINT TRACKING SYSTEM

A PROJECT REPORT
Submitted for the Subject
OPEN SOURCE DATABASE MANAGEMENT SYSTEM (OSDBMS)

Submitted By:
Kalyani Patil

Academic Year:
2025 – 2026

Established – 1961 Subject: OSDBMS

SEVA SADAN'S

R. K. TALREJA COLLEGE

OF

ARTS, SCIENCE & COMMERCE

ULHASNAGAR – 421 003pppop

CERTIFICATE

This is to certify that Mr./Ms. Kalyani Pankaj patil of S.Y. Information Technology (SYIT) Roll No. 2542034 has satisfactorily completed the Open Source DataBase Management System Mini Project entitled Service booking & Complaint Tracking 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**

_____ _____

Acknowlegment

I would like to express my sincere gratitude to my respected subject teacher for providing me with the valuable opportunity to undertake this project titled "Service Booking and Complaint Tracking System" in the subject Database Management System (DBMS). Their continuous guidance, constructive suggestions, and encouragement helped me to understand the practical concepts of database design and implementation.

I am also thankful to the Head of the Department and the faculty members of the Department of Computer Science for their support and for providing the necessary academic environment to successfully complete this project.

I extend my heartfelt thanks to my classmates and friends for their cooperation, discussions, and assistance throughout the development of this project. Their support helped me overcome difficulties during database design, SQL implementation, and documentation preparation.

Finally, I would like to thank my family for their constant motivation, encouragement, and moral support, which played an important role in the successful completion of this project.

This project has enhanced my understanding of MySQL, database constraints, SQL queries, transactions, and overall DBMS concepts.

I am truly grateful to everyone who contributed directly or indirectly to the completion of this work.

INDEX

INTRODUCTION

A database system is an organized collection of related data that is stored, managed, and accessed electronically using specialized software known as a Database Management System (DBMS). The primary purpose of a database system is to store large amounts of information in a structured manner and allow users to retrieve, update, and manage that information efficiently.

Unlike traditional file-based systems, a database system reduces redundancy, improves data consistency, enhances security, and supports multiple users accessing data simultaneously. It ensures that data remains accurate, reliable, and secure while performing various operations such as insertion, deletion, modification, and retrieval.

The Service Booking and Complaint Tracking Database System is designed to address problems commonly faced by service-based organizations. In many businesses, service bookings and complaints are often recorded manually or stored in unorganized spreadsheets. This leads to issues such as data duplication, loss of records, difficulty in tracking service status, delayed complaint resolution, and lack of proper reporting. Without a structured system, managing customer details, service information, booking schedules, and complaint history becomes inefficient and time-consuming.

The proposed database system solves these problems by centralizing all information into a single structured database. It allows easy tracking of customer bookings, monitoring of service status, and systematic handling of complaints. Additionally, it supports generating reports that help management in decision-making and improving service quality.

MySQL is used as the database platform for this project because it is a widely used open-source relational database management system (RDBMS). Being open-source means it is freely available, cost-effective, and supported by a large community of developers. MySQL provides high performance, reliability, and strong data security features. It supports Structured Query Language (SQL), which allows efficient creation, retrieval, and management of database records.

MySQL also supports transactions, constraints, views, and backup and recovery mechanisms, which are essential for maintaining data integrity and consistency. Furthermore, it is compatible with various programming languages such as Python, Java, and PHP, making it suitable for developing integrated database applications.

In conclusion, the Service Booking and Complaint Tracking Database System provides an efficient and structured solution for managing service operations. By using MySQL as the backend database, the system ensures reliability, security, scalability, and cost-effectiveness. The implementation of this database system improves operational efficiency, reduces errors, and enhances overall customer satisfaction.

PROBLEM DEFINITION

In many service-based organizations, service bookings and customer complaints are managed using manual registers, paper files, or basic spreadsheet applications. These traditional methods are not efficient for handling large volumes of data. As the organization grows, the number of customers, services, bookings, and complaints increases significantly. Without a structured system, maintaining accurate records becomes difficult. Manual systems lack proper data validation, relationships between records, and security mechanisms, which leads to operational inefficiencies and increased chances of errors.
The absence of a centralized database creates confusion in tracking service history, monitoring complaint status, and generating reports. Employees may struggle to find previous booking details or complaint records quickly. In addition, updating or modifying information in manual systems often results in duplication or inconsistency of data. These limitations directly affect customer satisfaction and organizational productivity.

Issues in the Existing Manual / Unstructured System

- **Data Redundancy:**
  The same customer details may be entered multiple times for different bookings, increasing storage usage and causing duplication.
- **Data Inconsistency:**
  If customer information is updated in one record but not in others, mismatched or incorrect data may occur.
- **Lack of Data Integrity:**
  Manual systems do not enforce primary keys, foreign keys, or constraints, leading to invalid or incomplete records.
- **Poor Security:**
  Paper files and unprotected spreadsheets can be easily accessed, altered, or deleted without authorization.
- **Difficulty in Tracking and Monitoring:**
  It becomes challenging to track booking status, complaint progress, service dates, and resolution history efficiently.
- **Time-Consuming Report Generation:**
  Preparing monthly or yearly reports requires manual compilation, which increases workload and chances of errors.
- **Risk of Data Loss:**
  Physical records can be damaged due to fire, water, or misplacement. Digital files without backup are also vulnerable to system failure.
- **Limited Scalability:**
  As the organization expands, managing large volumes of data manually becomes impractical and inefficient.

Need for a Database-Driven Solution

There is a need for a database-driven solution to efficiently manage service bookings and customer complaints in a structured and secure manner. A centralized database system helps reduce data redundancy, maintain consistency, and ensure data integrity through constraints and relationships. It improves tracking of bookings and complaint status, enables quick retrieval of information, and supports accurate report generation. Additionally, it enhances data security, prevents data loss through backup mechanisms, and increases overall operational efficiency and customer satisfaction.

OBJECTIVES OF THE PROJECT

The main objective of the Service Booking and Complaint Tracking Database System is to develop an efficient and structured database solution for managing service-related operations in an organized manner. The system aims to simplify the process of storing, retrieving, and managing customer, booking, and complaint data while ensuring accuracy and consistency. The specific objectives of the project are as follows:

- **To design a structured database system:**
  To design and create a well-organized relational database that stores data in multiple related tables such as Customer, Service, Booking, and Complaint. The database structure should follow proper normalization principles to reduce redundancy and improve data efficiency.

- **To implement SQL queries for data management:**
  To use Structured Query Language (SQL) to perform various database operations including inserting new records, updating existing records, deleting unwanted data, retrieving information, creating views, and generating reports for better analysis.

- **To ensure data integrity using constraints:**
  To apply database constraints such as primary keys, foreign keys, unique constraints, and not-null constraints in order to maintain data accuracy, prevent duplication, and enforce valid relationships between tables.

- **To gain hands-on experience with MySQL:**
  To develop practical knowledge and technical skills in MySQL by creating databases, managing tables, executing transactions, implementing backup and recovery techniques, and handling real-time database scenarios.

Overall, the project aims to strengthen understanding of database management concepts and demonstrate how a properly designed database system can improve efficiency and reliability in service-based organizations.

SCOPE OF THE PROJECT

The Service Booking and Complaint Tracking Database System is designed to provide a structured and efficient solution for managing service operations within an organization. The scope of this project includes the design, development, and implementation of a relational database that stores and manages data related to customers, services offered, bookings made, and complaints registered. The system ensures systematic storage of information and enables efficient retrieval, updating, and reporting of data.
The project focuses on developing a centralized database system using MySQL that supports core database functionalities such as data insertion, modification, deletion, querying, constraints enforcement, transactions, views, and backup and recovery mechanisms. The system aims to simplify service management processes and reduce manual workload.

Where the System Can Be Used
The system can be implemented in various environments where service management and complaint handling are required. It is suitable for:

- Appliance repair centers (AC, refrigerator, washing machine, etc.)
- IT service providers (laptop repair, hardware servicing)
- Maintenance service companies
- Technical support departments
- Small and medium-scale enterprises
- Institutional service departments

The system can also be adapted for online service booking platforms with further development.

Users of the System
The system can be accessed and used by different categories of users depending on organizational structure:

1 Administrator (Admin)

- Manages the entire database system.
- Creates and maintains tables.
- Applies constraints and manages transactions.
- Performs backup and recovery operations.
- Monitors data integrity and system security.
- Generates reports for analysis.

2 Staff / Employees

- Enter customer details.
- Create service bookings.
- Update service status (Booked, Completed, Cancelled).
- Register and update complaint records.
- View booking and complaint history.

3 Business User / Manager

- Reviews booking reports.
- Analyzes service performance.
- Tracks complaint resolution rate.
- Makes decisions based on generated reports and views.
- Monitors customer satisfaction trends.

4 Students (Academic Users)

- Use the system to understand relational database design.
- Practice SQL queries and joins.

- Learn about constraints, transactions, and views.
- Gain practical knowledge of MySQL Workbench.

Subdomains of Application

🔷 1. Educational Use

The project can be used as a practical demonstration in academic institutions for subjects related to Database Management Systems (DBMS). It helps students understand database concepts such as normalization, relationships, constraints, SQL operations, and transaction management.

🔷 2. Business Use

Small and medium service-based businesses can implement this system to automate booking management and complaint tracking. It improves operational efficiency, reduces paperwork, and enhances customer service quality.

🔷 3. Administrative Use

Organizations can use the system to maintain centralized records, monitor service activities, and generate structured reports for internal management purposes.

Academic Limitations

Since the project is developed for academic purposes, its scope is limited in the following ways:

- The system does not include advanced user authentication or role-based access control mechanisms.
- It is not designed for handling extremely large-scale enterprise data.
- The project does not include online deployment, cloud integration, or mobile application support.
- Real-time notifications and automated scheduling features are not implemented.
- The graphical user interface is basic and primarily for demonstration.

Despite these limitations, the project successfully demonstrates the practical implementation of a relational database system using MySQL and fulfills academic requirements by covering essential database concepts and operations.

REQUIREMENT SPECIFICATION

The successful implementation of the Service Booking and Complaint Tracking Database System requires appropriate hardware and software resources. Since the project is designed for academic and small to medium-scale organizational use, the system requirements are moderate and easily manageable.

5.1 Hardware Requirements

The hardware requirements specify the minimum physical components necessary to install and run the database system efficiently.

- **Computer / Laptop:**
  A standard desktop computer or laptop is required to install MySQL Server and MySQL Workbench. The system should support multitasking and allow smooth execution of database queries and GUI applications (if implemented).
- **Processor:**
  A minimum Dual-Core processor (Intel i3 or equivalent) is recommended. A higher processor such as Intel i5 or above will improve performance, especially when handling larger datasets or running multiple applications simultaneously.
- **Minimum RAM:**
  At least **4 GB RAM** is required to run MySQL Server and Workbench smoothly. For better performance and faster query execution, **8 GB RAM** is recommended. Higher RAM ensures efficient handling of larger databases and reduces system lag.
- **Storage:**
  A minimum of **10–20 GB free disk space** is required for installing MySQL Server, MySQL Workbench, and storing database files. As the database grows, additional storage may be required to accommodate backup files, logs, and exported data.
- **Input/Output Devices:**
  Standard input devices such as keyboard and mouse are required for query execution and system interaction. A monitor is required for visual interface operations.

5.2 Software Requirements

The software requirements define the programs and platforms needed to develop, execute, and manage the database system.

Software	Purpose
MySQL Server	Acts as the database management system (DBMS). It is used to create databases, define tables, enforce constraints, execute transactions, and manage stored data securely.
MySQL Workbench	Provides a graphical interface for designing databases, writing and executing SQL queries, managing users, performing backup and recovery, and visualizing database schemas.
Operating System (Windows / Linux)	Serves as the platform on which MySQL Server and Workbench operate. The system is compatible with Windows 10/11 and major Linux distributions.
SQL (Structured Query Language)	Used for performing database operations such as CREATE, INSERT, UPDATE, DELETE, SELECT, JOIN, VIEW creation, and transaction management.

SYSTEM DESIGN

6.1 System Architecture (Conceptual)

The Service Booking and Complaint Tracking Database System follows a structured client–database architecture where users interact with the database through a user interface or query environment

1. User Interaction with the Database

Users interact with the database system either through:

- MySQL Workbench (for query execution), or
- A Graphical User Interface (GUI) developed using a programming language such as Python.

The user performs operations such as:

- Adding new customer records
- Creating service bookings
- Updating booking status
- Registering complaints
- Retrieving reports

These operations are translated into SQL queries (such as INSERT, SELECT, UPDATE, DELETE) which are sent to the MySQL Server for execution.

The user does not directly manipulate database files. Instead, all interactions are managed through structured SQL commands.

2. Role of MySQL Server

MySQL Server acts as the core component of the system. It performs the following major functions:

- Stores all database tables (Customer, Service, Booking, Complaint).
- Maintains relationships using primary and foreign keys.
- Enforces constraints to ensure data integrity.
- Executes SQL queries received from the user or application.
- Manages transactions using COMMIT and ROLLBACK.
- Controls data consistency and concurrency.
- Provides backup and recovery mechanisms.
- Creates and manages database views (e.g., Booking_Report).

MySQL Server processes requests and ensures that only valid operations are performed according to defined rules and constraints.

3. Query Execution Flow

The query execution process follows a systematic flow:

1. The user enters an SQL query (e.g., SELECT * FROM Booking).
2. The query is sent to the MySQL Server.
3. The MySQL Query Processor checks:
    - Syntax correctness
    - Table and column existence
    - Permission validation
4. The optimizer determines the best execution plan.
5. The storage engine retrieves or modifies data.
6. The result is returned to the user interface.
7. If it is a transaction-based query:
    - COMMIT saves changes permanently.
    - ROLLBACK cancels changes if an error occurs.

This structured flow ensures efficiency, accuracy, and reliability in data operations.

DATABASE DESIGN

The database design for the Service Booking and Complaint Tracking System is developed using the relational database model in MySQL. The system consists of four main tables: Customer, Service, Booking, and Complaint. These tables are logically connected using primary and foreign key relationships to ensure data consistency, reduce redundancy, and maintain integrity.

7.1 Entity Description

Customer

The Customer table stores basic details of customers who book services. It includes customer ID, name, phone number, email, and address. Each customer is uniquely identified using customer_id. This table is important because bookings and complaints are linked to customers.
The Customer table supports:

- Registration of new customers
- Retrieval of customer contact information
- Linking customers to bookings and complaints

Service

The Service table contains information about services provided by the organization. It includes service ID, service name, description, and charges. Each service is uniquely identified by service_id. This table helps manage service categories and pricing.
This entity helps in:

- Managing different service categories
- Defining service pricing
- Linking services with customer bookings

Booking

The Booking table records service bookings made by customers. It includes booking ID, customer ID, service ID, booking date, service date, and status. This table connects Customer and Service tables through foreign keys and helps track service requests and their progress.
Each booking has:

- A unique booking_id
- customer_id (reference to Customer table)
- service_id (reference to Service table)
- booking date
- service date
- status (Booked, Completed, Cancelled).

Complaint

The Complaint table stores complaints raised by customers regarding their bookings. It contains complaint ID, customer ID, booking ID, complaint date, description, and status. It ensures proper complaint tracking and resolution.
Each complaint contains:

- complaint_id (Primary Key)
- customer_id (Foreign Key)
- booking_id (Foreign Key)
- complaint date
- complaint description
- complaint status (Pending, Resolved)

7.2 Table Structure

7.2.1 : Customer Table:

Attribute	Data Type	Description
customer_id	INT	Unique ID of customer
name	VARCHAR(100)	Customer name
phone	VARCHAR(15)	Customer phone number
email	VARCHAR(100)	Customer email
address	VARCHAR(200)	Customer address

7.2.2 : Service Table:

Attribute	Data Type	Description
service_id	INT	Unique ID of service
service_name	VARCHAR(100)	Name of service
description	VARCHAR(200)	Service details
charges	DECIMAL(10,2)	Service cost

7.2.3 : Booking Table:

Attribute	Data Type	Description
booking_id	INT	Unique booking ID
customer_id	INT	References Customer table
service_id	INT	References Service table
booking_date	DATE	Date of booking
service_date	DATE	Scheduled service date
status	VARCHAR(20)	Booking status

7.2.4 : Complaint Table:

Attribute	Data Type	Description
complaint_id	INT	Unique complaint ID
customer_id	INT	References Customer table
booking_id	INT	References Booking table
complaint_date	DATE	Date of complaint
description	VARCHAR(255)	Complaint details
status	VARCHAR(20)	Complaint status

7.3 Constraints Used

Constraints are rules applied to tables to maintain accuracy, consistency, and integrity of data.

1. PRIMARY KEY

A Primary Key uniquely identifies each record in a table. It ensures that no duplicate or null values are allowed in that column.

Example:

- customer_id in Customer table
- service_id in Service table
- booking_id in Booking table
- complaint_id in Complaint table

2. FOREIGN KEY

A Foreign Key establishes a relationship between two tables. It ensures referential integrity by allowing only valid values that exist in the referenced table.

Example:

- customer_id in Booking references Customer(customer_id)
- service_id in Booking references Service(service_id)
- booking_id in Complaint references Booking(booking_id)

3. NOT NULL

The NOT NULL constraint ensures that a column cannot have empty or null values. It guarantees that important fields must contain data.

Example:

- name in Customer table
- Primary key fields

4. UNIQUE

The UNIQUE constraint ensures that all values in a column are different. It prevents duplicate entries.

Example:

- phone in Customer table

Overall, the database design ensures structured data storage, strong relationships between entities, and enforcement of integrity rules, making the system reliable and efficient.

UML DIAGRAMS

Unified Modeling Language (UML) diagrams are used to visually represent the structure and behavior of the Service Booking & Complaint Tracking Management System. These diagrams help in understanding the database design, system workflow, and interaction between users and the database system. In this project, ER Diagram, Use Case Diagram, Activity Diagram, and Sequence Diagram are used to represent different aspects of the system.

8.1 ER Diagram (Entity Relationship Diagram)

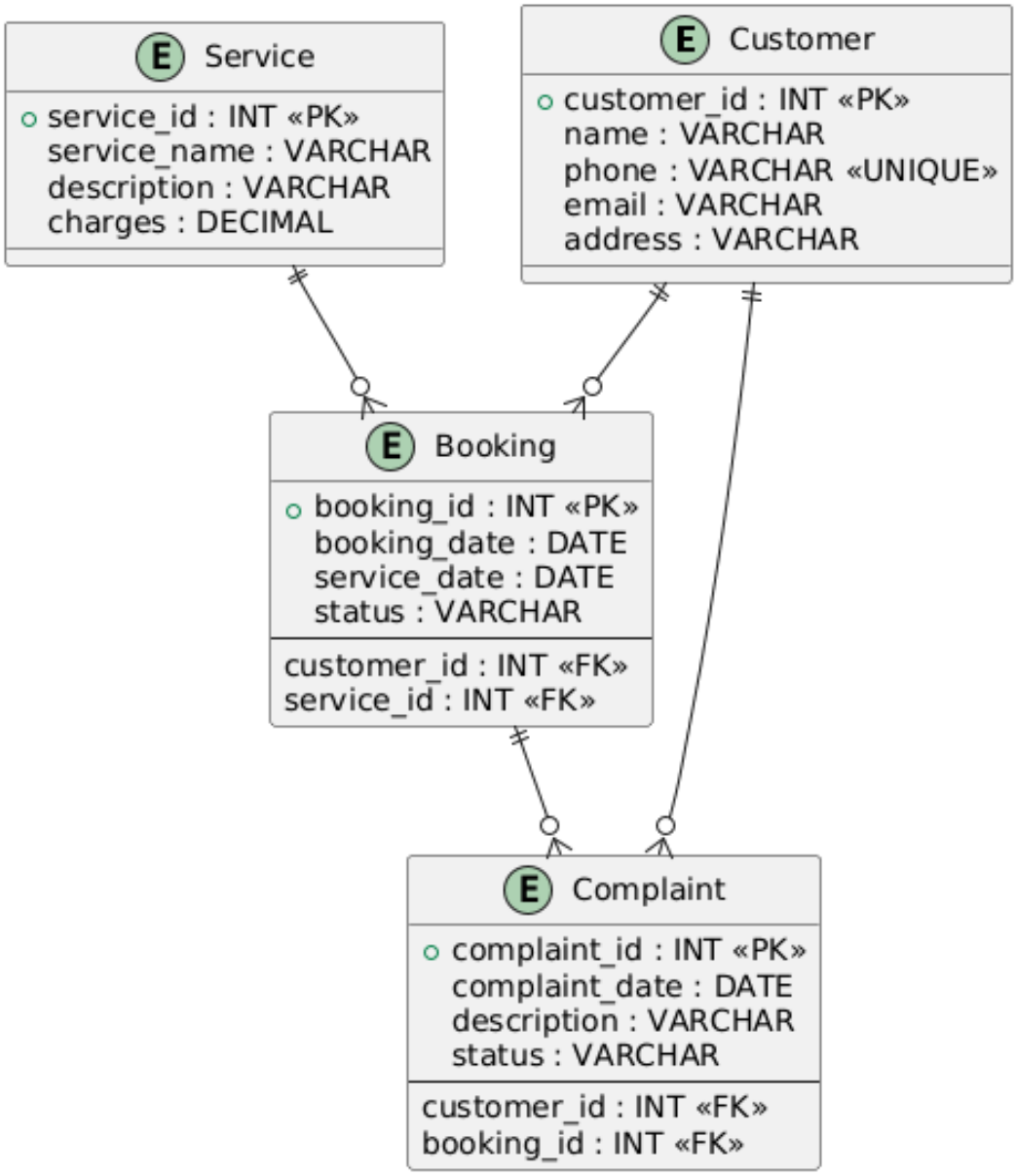

Figure 8.1: ER Diagram

The ER diagram clearly defines how data is interconnected and ensures referential integrity within the database.

8.2 Use Case Diagram

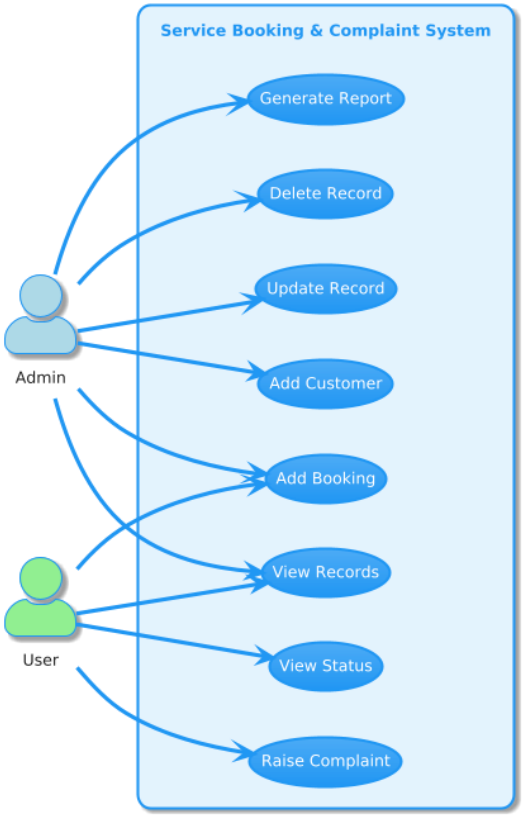

Figure 8.2: Use Case Diagram

Explanation:

The Use Case Diagram shows how users interact with the system. It includes two actors: Admin and User. The Admin can insert, update, delete, and generate reports, while the User can book services, view records, and raise complaints. This diagram explains system functionality from the user's perspective.

8.3 Activity Diagram

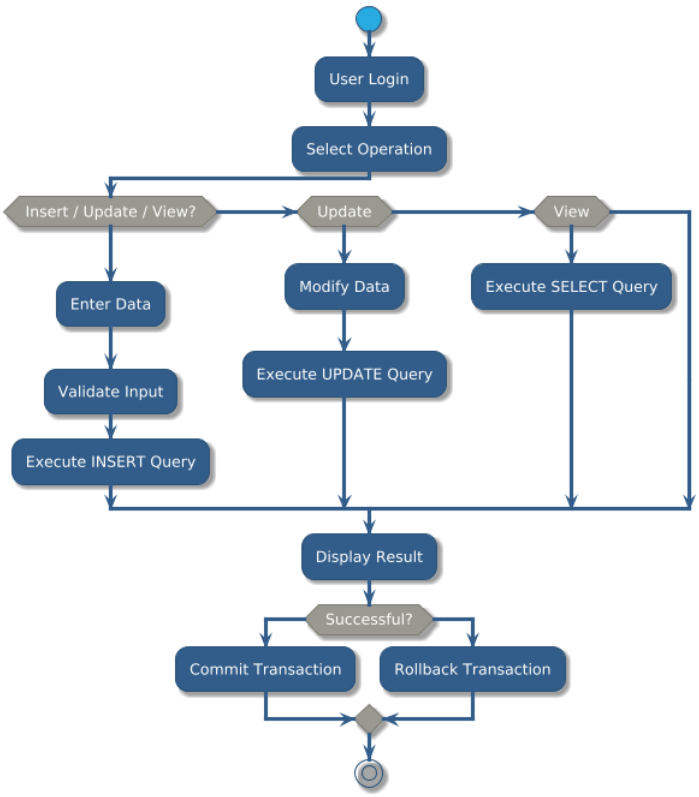

Figure 8.3: Activity Diagram

Explanation:

The Activity Diagram shows the workflow of the system. It starts from user login, selection of operation, execution of SQL query, and finally displaying results. It also shows decision points like commit or rollback. This diagram explains how processes move step by step.

8.4 Sequence Diagram

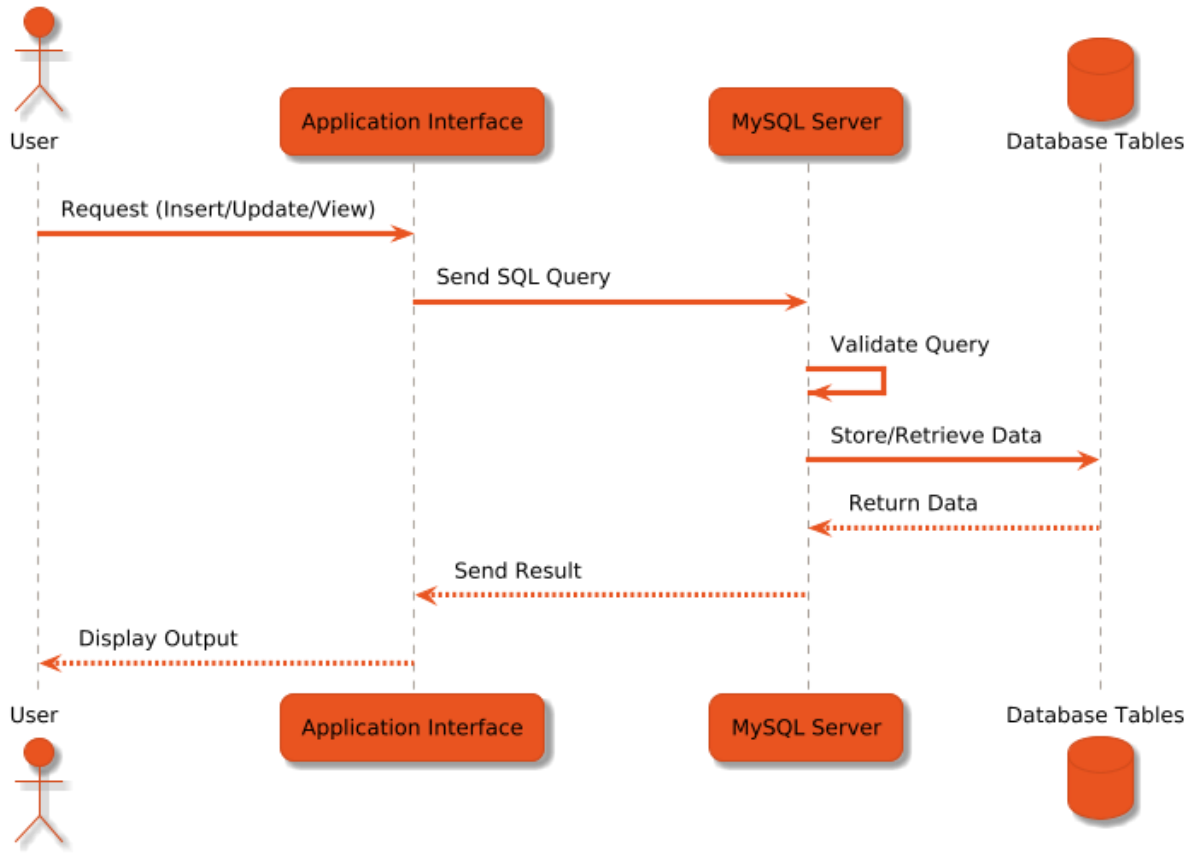

Figure 8.4: Sequence Diagram

Explanation:

The Sequence Diagram shows the interaction between User, Application, and MySQL Server over time. It explains how a request is sent, processed, validated, and how the result is returned. It represents the query execution flow in order.

SQL IMPLEMENTATION

This chapter explains how the Service Booking and Complaint Tracking System is implemented using SQL commands in MySQL. It includes database creation, table creation with constraints, data insertion, retrieval operations, and advanced queries. These SQL operations help in managing structured data efficiently and maintaining data integrity.

9.1 Database Creation

The database is created using the CREATE DATABASE command. After creation, the USE command selects the database for performing further operations.

```
CREATE DATABASE service_booking_db;
USE service_booking_db;
```

The database acts as a container that stores all related tables such as Customer, Service, Booking, and Complaint. This ensures organized data management.

9.2 Table Creation

Tables are created using the CREATE TABLE statement. Each table includes attributes with appropriate data types and constraints such as PRIMARY KEY, FOREIGN KEY, NOT NULL, and UNIQUE.

Customer Table

```
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    phone VARCHAR(15) UNIQUE,
    email VARCHAR(100),
    address VARCHAR(200)
);
```

This table stores customer details. The primary key ensures uniqueness, while NOT NULL ensures mandatory fields are filled.

Service Table

```
CREATE TABLE Service (
    service_id INT PRIMARY KEY,
    service_name VARCHAR(100),
    description VARCHAR(200),
    charges DECIMAL(10,2)
);
```

This table stores details of services offered and their charges.

Booking Table

```
CREATE TABLE Booking (
    booking_id INT PRIMARY KEY,
    customer_id INT,
    service_id INT,
    booking_date DATE,
    service_date DATE,
    status VARCHAR(20),
```

```
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    FOREIGN KEY (service_id) REFERENCES Service(service_id)
);
```
This table links customers and services using foreign keys and tracks booking details.

Complaint Table

```
CREATE TABLE Complaint (
    complaint_id INT PRIMARY KEY,
    customer_id INT,
    booking_id INT,
    complaint_date DATE,
    description VARCHAR(255),
    status VARCHAR(20),
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)
);
```
This table records complaints and maintains relationships with customers and bookings.

9.3 Data Insertion

The INSERT INTO command is used to add records into tables. It allows storing customer details, service information, bookings, and complaints.

1. Insert Data into Customer Table

INSERT INTO Customer VALUES
(1, 'Amit Sharma', '9876543210', 'amit@gmail.com', 'Mumbai'),
(2, 'Neha Patil', '9123456780', 'neha@gmail.com', 'Pune'),
(3, 'Rahul Verma', '9988776655', 'rahul@gmail.com', 'Delhi'),
(4, 'Sneha Joshi', '9090909090', 'sneha@gmail.com', 'Nashik'),
(5, 'Karan Mehta', '9567123456', 'karan@gmail.com', 'Thane');

2. Insert Data into Service Table

```
INSERT INTO Service VALUES
(101, 'AC Repair', 'Air conditioner servicing', 1500),
(102, 'Washing Machine Repair', 'Washing machine maintenance', 1200),
(103, 'Refrigerator Repair', 'Fridge servicing', 1000),
(104, 'Laptop Repair', 'Laptop hardware repair', 2000),
(105, 'Water Purifier Service', 'RO servicing', 800);
```

3. Insert Data into Booking Table

```
INSERT INTO Booking VALUES
(201, 1, 101, '2026-01-10', '2026-01-12', 'Booked'),
(202, 2, 102, '2026-01-11', '2026-01-13', 'Completed'),
(203, 3, 103, '2026-01-12', '2026-01-14', 'Booked'),
(204, 4, 104, '2026-01-13', '2026-01-15', 'Cancelled'),
(205, 5, 105, '2026-01-14', '2026-01-16', 'Booked');
```

4. Insert Data into Complaint Table

```
INSERT INTO Complaint VALUES
(301, 1, 201, '2026-01-13', 'Service delayed', 'Resolved'),
(302, 2, 202, '2026-01-14', 'Technician was late', 'Resolved'),
(303, 3, 203, '2026-01-15', 'Issue not fixed properly', 'Pending'),
(304, 4, 204, '2026-01-16', 'Booking cancelled without notice', 'Resolved'),
```

```
(305, 5, 205, '2026-01-17', 'Service charges too high', 'Pending');
```

9.4 Data Retrieval
Data retrieval is performed using the SELECT statement.

Simple SELECT

```
SELECT * FROM Customer;
```
This retrieves all records from the Customer table.

SELECT with WHERE

```
SELECT * FROM Booking
WHERE status = 'Booked';
```

The WHERE clause filters records based on a condition.

JOIN Query

```
SELECT Customer.name, Service.service_name, Booking.status
FROM Booking
JOIN Customer ON Booking.customer_id = Customer.customer_id
JOIN Service ON Booking.service_id = Service.service_id;
```

JOIN combines data from multiple tables using relationships. It helps generate meaningful reports.

9.5 Advanced Queries
Advanced queries help in performing complex data analysis.

GROUP BY

```
SELECT status, COUNT(*) AS total_complaints
FROM Complaint
GROUP BY status;
```

GROUP BY groups records based on a column and is often used with aggregate functions like COUNT().

HAVING

```
SELECT status, COUNT(*) AS total
FROM Booking
GROUP BY status
HAVING COUNT(*) > 1;
```

HAVING filters grouped results.

Subquery
```
SELECT name
FROM Customer
WHERE customer_id IN (
```

```
 SELECT customer_id FROM Booking
);
```
A subquery is a query inside another query. It helps retrieve dependent data.

View Creation

```
CREATE VIEW Booking_Report AS
SELECT Customer.name, Service.service_name, Booking.service_date,
Booking.status
FROM Booking
JOIN Customer ON Booking.customer_id = Customer.customer_id
JOIN Service ON Booking.service_id = Service.service_id;
```
A view is a virtual table created from a query. It simplifies complex queries and improves readability.
To retrieve data:
```
SELECT * FROM Booking_Report;
```

Transaction

```
START TRANSACTION;

INSERT INTO Booking VALUES
(302, 2, 102, '2026-02-05', '2026-02-07', 'Booked');

INSERT INTO Complaint VALUES
(401, 2, 302, '2026-02-07', 'Service not satisfactory', 'Pending');

COMMIT;
```

- START TRANSACTION begins the transaction.
- SQL statements are executed together.
- COMMIT saves all changes permanently.
- If an error occurs, ROLLBACK can be used to cancel changes.

A transaction ensures data consistency and reliability.

Conclusion

The SQL implementation demonstrates how database operations are performed systematically using MySQL.
It ensures structured data storage, secure relationships between tables, efficient retrieval of records, and advanced data processing using grouping, subqueries, and views.

GANTT CHART :

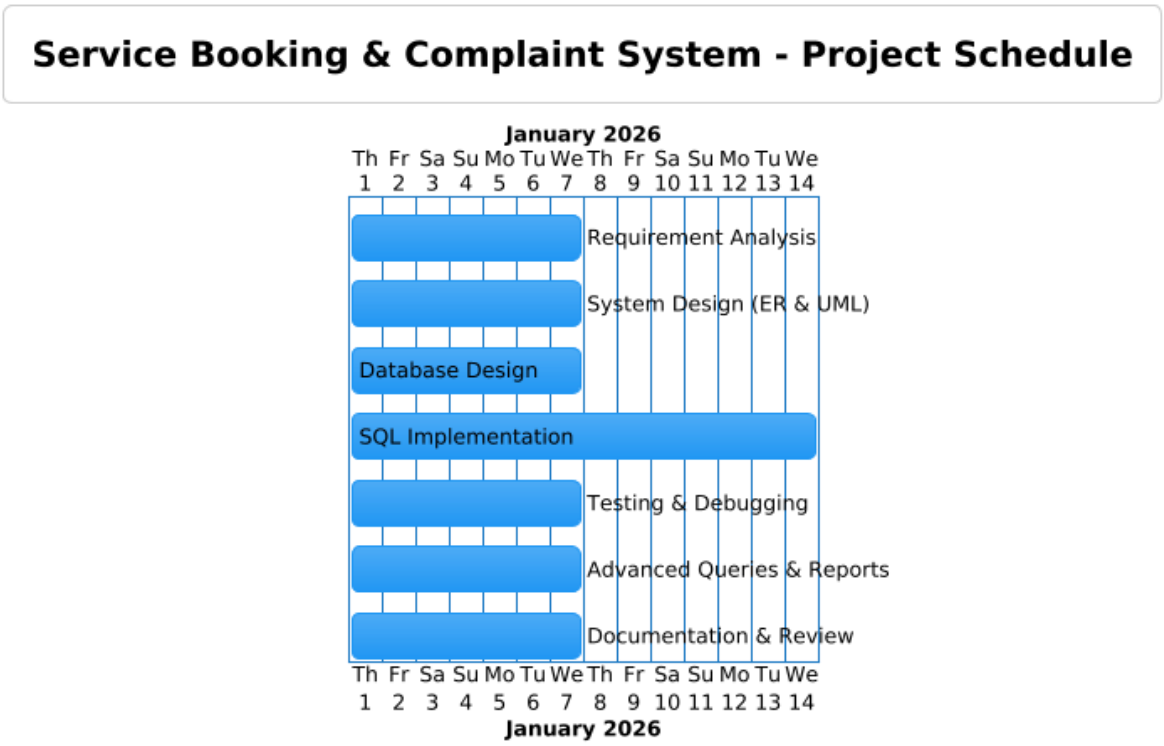

Figure 9.1: Gantt Diagram

SYSTEM TESTING AND RESULT

Testing is an important phase in database development. It ensures that SQL queries work correctly, data is stored properly, and outputs are accurate. In this project, testing was performed at different levels to verify correctness, consistency, and reliability of the database system.

10.1 Query Correctness Testing

Query correctness testing ensures that all SQL queries execute without errors and produce expected results.
The following checks were performed:

- Verification of table creation using DESC table_name;
- Execution of INSERT statements to ensure data is added successfully.
- Testing of SELECT queries to confirm proper data retrieval.
- Validation of JOIN queries to check relationships between tables.
- Testing of UPDATE and DELETE commands to verify record modification.
- Execution of GROUP BY, HAVING, and subqueries to confirm advanced query accuracy.

If any error occurred (such as foreign key violation or duplicate primary key), it was corrected and re-tested.

1. Database and Table Creation Testing

Database and Table Creation Testing is performed to verify that the database and all required tables are created successfully without errors. This testing ensures that the database structure is correctly defined before inserting or retrieving any data.

Actual Result

The database service_booking_db and all tables (Customer, Service, Booking, Complaint) were created successfully.

Customer table:

customer_id	name	phone	email	address
1	Amit Sharma	9876543210	amit@gmail.com	Mumbai
2	Neha Patil	9123456780	neha@gmail.com	Pune
3	Rahul Verma	9988776655	rahul@gmail.com	Delhi
4	Sneha Joshi	9090909090	sneha@gmail.com	Nashik
5	Karan Mehta	9567123456	karan@gmail.com	Thane
NULL	NULL	NULL	NULL	NULL

Service table:

service_id	service_name	description	charges
101	AC Repair	Air conditioner servicing	1500.00
102	Washing Machine Repair	Washing machine maintenance	1200.00
103	Refrigerator Repair	Fridge servicing	1000.00
104	Laptop Repair	Laptop hardware repair	2000.00
105	Water Purifier Service	RO servicing	800.00
NULL	NULL	NULL	NULL

Booking table:

booking_id	customer_id	service_id	booking_date	service_date	status
201	1	101	2026-01-10	2026-01-12	Booked
202	2	102	2026-01-11	2026-01-13	Completed
203	3	103	2026-01-12	2026-01-14	Booked
204	4	104	2026-01-13	2026-01-15	Cancelled
205	5	105	2026-01-14	2026-01-16	Booked
301	1	101	2026-02-01	2026-02-03	Booked
302	2	102	2026-02-05	2026-02-07	Booked
NULL	NULL	NULL	NULL	NULL	NULL

Complaint table:

complaint_id	customer_id	booking_id	complaint_date	description	status
301	1	201	2026-01-13	Service delayed	Resolved
302	2	202	2026-01-14	Technician was late	Resolved
303	3	203	2026-01-15	Issue not fixed properly	Pending
304	4	204	2026-01-16	Booking cancelled without notice	Resolved
305	5	205	2026-01-17	Service charges too high	Pending
401	2	302	2026-02-07	Service not satisfactory	Resolved
NULL	NULL	NULL	NULL	NULL	NULL

2. Join Query

SELECT Customer.name, Service.service_name, Booking.status
FROM Booking

JOIN Customer ON Booking.customer_id = Customer.customer_id

JOIN Service ON Booking.service_id = Service.service_id;

Result outcome:

This SQL query gives a **list of all bookings** showing:

- **Customer name** (Customer.name)
- **Service booked** (Service.service_name)
- **Booking status** (Booking.status)

It combines data from Booking, Customer, and Service tables using their IDs, so each row tells who booked what service and its current status.

name	service_name	status
Amit Sharma	AC Repair	Booked
Amit Sharma	AC Repair	Booked
Neha Patil	Washing Machine Repair	Completed
Neha Patil	Washing Machine Repair	Booked
Rahul Verma	Refrigerator Repair	Booked
Sneha Joshi	Laptop Repair	Cancelled
Karan Mehta	Water Purifier Service	Booked

3. Create a View

```
CREATE VIEW Booking_Details AS
SELECT Customer.name, Service.service_name, Booking.service_date,
Booking.status
FROM Booking
JOIN Customer ON Booking.customer_id = Customer.customer_id
JOIN Service ON Booking.service_id = Service.service_id;
```

Result outcome:

This creates a view named Booking_Details that shows:

- Customer name
- Service booked
- Date of booking
- Booking status

It's like a saved query combining Booking, Customer, and Service tables, so you can easily see all booking details without writing the full query each time.

name	service_name	service_date	status
Amit Sharma	AC Repair	2026-01-12	Booked
Amit Sharma	AC Repair	2026-02-03	Booked
Neha Patil	Washing Machine Repair	2026-01-13	Completed
Neha Patil	Washing Machine Repair	2026-02-07	Booked
Rahul Verma	Refrigerator Repair	2026-01-14	Booked
Sneha Joshi	Laptop Repair	2026-01-15	Cancelled
Karan Mehta	Water Purifier Service	2026-01-16	Booked

4. GROUP BY

```
SELECT status, COUNT(*) AS total_complaints
FROM Complaint
GROUP BY status;
```

Result outcome:

This query groups complaints by their status and counts how many complaints are in each category.

The result shows:

- How many complaints are Pending
- How many complaints are Resolved

Each row represents one status type, and COUNT(*) gives the total number of complaints for that status.

status	total_complaints
Resolved	4
Pending	2

5. HAVING

```
SELECT status, COUNT(*) AS total
FROM Booking
GROUP BY status
HAVING COUNT(*) > 1;
```

Result outcome:

This query groups bookings by their status and counts how many bookings exist for each status. The HAVING COUNT(*) > 1 condition shows only those statuses that have more than one booking.

status	total
Booked	5

6. SUBQUERY

```
SELECT name
FROM Customer
WHERE customer_id IN (
    SELECT customer_id FROM Booking
);
```

Result outcome:

The output shows only customers who have booked a service.
Customers who have not made any booking will not appear in the result.

name
Amit Sharma
Neha Patil
Rahul Verma
Sneha Joshi
Karan Mehta

8. Transaction

```
START TRANSACTION;

INSERT INTO Booking VALUES
(302, 2, 102, '2026-02-05', '2026-02-07', 'Booked');

INSERT INTO Complaint VALUES
```

```
(401, 2, 302, '2026-02-07', 'Service not satisfactory', 'Pending');

COMMIT;
```

Result outcome:
This transaction adds a new booking and its complaint together.
When COMMIT is executed, both records are saved permanently.

It ensures both operations are completed successfully as one unit.

booking_id	customer_id	service_id	booking_date	service_date	status
201	1	101	2026-01-10	2026-01-12	Booked
202	2	102	2026-01-11	2026-01-13	Completed
203	3	103	2026-01-12	2026-01-14	Booked
204	4	104	2026-01-13	2026-01-15	Cancelled
205	5	105	2026-01-14	2026-01-16	Booked
301	1	101	2026-02-01	2026-02-03	Booked
302	2	102	2026-02-05	2026-02-07	Booked
NULL	NULL	NULL	NULL	NULL	NULL

complaint_id	customer_id	booking_id	complaint_date	description	status
301	1	201	2026-01-13	Service delayed	Resolved
302	2	202	2026-01-14	Technician was late	Resolved
303	3	203	2026-01-15	Issue not fixed properly	Pending
304	4	204	2026-01-16	Booking cancelled without notice	Resolved
305	5	205	2026-01-17	Service charges too high	Pending
401	2	302	2026-02-07	Service not satisfactory	Resolved
NULL	NULL	NULL	NULL	NULL	NULL

10.4 Sample Result Explanation :
The Booking_Report shows **which customer booked which service, on what date, and its status**.
Example:

- Amit Sharma → AC Repair → 2026-01-12 → Booked
- Neha Patil → Washing Machine Repair → 2026-01-13 → Completed
- Sneha Joshi → Laptop Repair → 2026-01-15 → Cancelled

Meaning: Each row tells **who booked what service, when, and its current status**.

SECURITY, BACKUP AND RECOVERY

SECURITY

Security in a database ensures that **data is protected** from unauthorized access, modification, or misuse. It includes:

- **User Privileges:** Every user of the database is given specific rights based on their role. For example, an administrator may have full rights to create, modify, or delete data, while a regular staff member may only be allowed to view bookings or update complaint status. This ensures that users can only perform actions that are necessary for their work, minimizing the risk of accidental or malicious changes.
- **Access Control:** Access control defines **who can access the database and what operations they can perform**. It prevents unauthorized users from viewing sensitive information like customer contacts or complaint details. It can include password protection, roles, and restrictions on which tables or records a user can access. This keeps the database safe and ensures compliance with privacy standards.

BACKUP

Backup is the process of **creating a copy of the database** to protect against data loss caused by accidental deletion, hardware failure, or software errors. In MySQL, the **mysqldump** utility is commonly used for this purpose.

- **mysqldump Explanation:** mysqldump creates a **logical backup** of the database by exporting the database structure (tables, columns, constraints) and all data into a file. This file can be stored safely and used later to restore the database if needed. Regular backups ensure that even if the live database is damaged or lost, the information can be recovered.

RECOVERY

Recovery is the process of **restoring a database from a backup** to return it to a previous working state after data loss or corruption.

- **Restore Concept:** The backup file created using tools like mysqldump can be used to recreate the database exactly as it was at the time of the backup. Recovery ensures **business continuity**, allowing the system to continue operating without losing important information such as customer bookings, service details, and complaint records. Recovery strategies may also include restoring only specific tables or data depending on the situation.

Summary:

Security protects the database from unauthorized access, backup ensures a safe copy of all data is available, and recovery allows the database to be restored quickly in case of failure. Together, these processes keep the **Service Booking and Complaint Tracking Database reliable, safe, and operational at all times**.

FUTURE SCOPE AND CONCLUSION

FUTURE SCOPE

The Service Booking and Complaint Tracking Database can be further enhanced with several improvements:

- Web Integration: The database can be connected to a web-based interface, allowing customers to book services online, view their booking history, and lodge complaints directly through a website or mobile application. This increases convenience and accessibility.
- Advanced Reporting: Reports can be generated automatically, providing insights into service trends, booking patterns, and complaint resolution. This helps management make informed decisions and improve service quality.
- Analytics Features: By integrating analytics tools, the database can provide predictions and data-driven insights, such as identifying frequently booked services, peak booking periods, or recurring complaints. This allows the business to optimize resources and enhance customer satisfaction.

CONCLUSION

The project successfully developed a Service Booking and Complaint Tracking Database that manages customer details, service bookings, and complaints efficiently.

- What was achieved: A functional database with tables for customers, services, bookings, and complaints; relationships were established; and transactions, views, and queries were implemented to handle real-world operations.
- Learning outcomes: Understanding of MySQL database design, table creation, relationships, constraints, transactions, views, and queries. Learned how to handle backup, recovery, and security for reliable database management.
- Importance of MySQL: MySQL provides a robust, reliable, and widely used platform for managing structured data. Its support for transactions, security features, and easy integration with applications makes it ideal for business management systems like service booking and complaint tracking.

REFERENCES

• MySQL official documentation, Oracle Corporation.
• Prescribed Database Management System (DBMS) textbooks.
• Online learning sources

• W3Schools SQL Tutorial

• GeeksforGeeks Database Management System

• Tutorialspoint MySQL Tutorial

GLOSSARY

- **DBMS (Database Management System):** Software that allows users to **store, manage, and retrieve data** efficiently. It provides tools for creating, updating, and querying databases.
- **SQL (Structured Query Language):** A programming language used to **interact with a database**. It is used to create tables, insert data, update records, delete data, and retrieve information using queries.
- **Primary Key:** A **unique identifier** for each record in a table. It ensures that no two rows in a table have the same value for this column, maintaining data integrity.
- **Foreign Key:** A column in one table that **links to the primary key of another table**, establishing a relationship between tables. It ensures referential integrity by enforcing valid links between related data.
- **MySQL:** A **popular open-source relational database management system (RDBMS)** that uses SQL for managing and manipulating data. It is widely used for web applications and business databases.
- **View:** A **virtual table** based on a SQL query. It allows users to see specific data from one or more tables without modifying the original tables.
- **Transaction:** A **set of SQL operations executed as a single unit**. Transactions ensure data consistency; they can be **committed** (applied) or **rolled back** (undone) if an error occurs.
- **Access Control:** A security feature that **restricts which users can access or modify data** in the database, ensuring that only authorized users perform certain actions.
- **Complaint Tracking:** A system feature that **records and monitors customer complaints**, including complaint details, booking reference, and resolution status, to improve service quality.

Entities and Attributes

1. **Customer**
    - **Attributes:** customer_id (Primary Key), name, phone, email, address
2. **Service**
    - **Attributes:** service_id (Primary Key), service_name, description, charges
3. **Booking**
    - **Attributes:** booking_id (Primary Key), customer_id (Foreign Key), service_id (Foreign Key), booking_date, service_date, status
4. **Complaint**
    - **Attributes:** complaint_id (Primary Key), customer_id (Foreign Key), booking_id (Foreign Key), complaint_date, description, status