Established – 1961 Subject: OSDBMS

SEVA SADAN'S

R. K. TALREJA COLLEGE

OF

ARTS, SCIENCE & COMMERCE ULHASNAGAR – 421 003

CERTIFICATE

This is to certify that Mr Arpit Markandeyprasad Mishra of S.Y. Information Technology (SYIT) Roll No. 2542026 has satisfactorily completed the Open Source DataBase Management System Mini Project entitled Doctor Appointment Scheduling Database system . 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**

INDEX

CHAPTER 1: INTRODUCTION

The rapid advancement of information technology has significantly transformed various sectors, including healthcare. One of the most critical challenges faced by healthcare institutions is the efficient management of patient appointments. In many hospitals and clinics, appointment scheduling is still handled manually or through semi-automated systems, which often results in inefficiency, human error, and poor utilization of medical resources. As the number of patients increases, managing appointments without a structured database system becomes increasingly complex.

The **Doctor Appointment Scheduling Database System** is designed to address these challenges by providing a centralized, database-driven solution for managing doctor schedules, patient records, and appointment bookings. The system ensures that appointments are scheduled accurately, conflicts are avoided, and information is stored in a structured and secure manner. By automating appointment scheduling, the system reduces administrative workload and improves overall operational efficiency. This project focuses on the design and implementation of a relational database using DBMS principles. The system stores critical information such as doctor details, patient profiles, availability schedules, and appointment records. The use of a database management system ensures data consistency, integrity, and reliability. Concepts such as normalization, constraints, transactions, and relationships are applied to ensure optimal database performance.

The Doctor Appointment Scheduling Database System not only improves efficiency but also enhances the patient experience by reducing waiting time and ensuring better coordination between doctors and patients. The project serves as a practical demonstration of how DBMS concepts can be applied to solve real-world problems in the healthcare domain.

CHAPTER 2: PROBLEM DEFINITION

In traditional healthcare environments, appointment scheduling is commonly performed using manual registers, spreadsheets, or basic standalone software tools. While these methods may appear sufficient for small clinics with limited patient flow, they become increasingly inefficient as the number of patients and doctors grows. Manual systems lack automation and real-time synchronization, making it difficult to maintain accurate and up-to-date appointment records.

One of the primary challenges in traditional scheduling systems is the absence of real-time availability tracking. When multiple staff members handle appointment bookings simultaneously, there is no centralized mechanism to verify whether a specific time slot has already been allocated. This significantly increases the chances of double booking and scheduling conflicts. Such errors not only inconvenience patients but also disrupt the doctor's workflow.

Patients often experience long waiting times due to improper slot allocation and poor time management. In many cases, appointments are scheduled without considering the doctor's maximum patient capacity per day. As a result, some days become overloaded while others remain underutilized. This imbalance affects operational efficiency and reduces overall patient satisfaction.

Additionally, managing cancellations, rescheduling, and emergency appointments becomes complicated in manual systems. Without an integrated database, updating one record does not automatically reflect in other related records. This leads to inconsistencies and confusion among administrative staff.

Major Problems Identified

1. Lack of real-time appointment tracking

2. High risk of double booking and scheduling conflicts

3. No centralized database system
4. Difficulty in managing cancellations and rescheduling
5. Inefficient time slot allocation
6. Uneven workload distribution among doctors
7. Long patient waiting times
8. Data redundancy and duplication
9. Data inconsistency across multiple records
10. Absence of automated validation and constraint enforcement
11. Poor data security and lack of access control

CHAPTER 3: OBJECTIVES OF THE PROJECT

The primary objective of this project is to design and implement a database system that automates doctor appointment scheduling.

Secondary objectives include:

- Eliminating double booking of appointments
- Maintaining accurate patient and doctor records
- Ensuring data integrity using constraints
- Implementing transactions for safe appointment booking
- Demonstrating practical application of DBMS concepts

CHAPTER 4: SCOPE OF THE PROJECT

The scope of the Doctor Appointment Scheduling System defines the boundaries and limitations of the project.

This project focuses on building a database system to store and manage appointment data efficiently. It is designed mainly for academic and learning purposes.

4.1 Application Areas

The system can be used in the following areas:

- Educational institutions for teaching database concepts
- Local clinics for better accessibility
- Training programs related to data management
- Academic mini-projects

4.2 Users of the System

The main users of the system are:

- Administrator – Manages database and all records.
- Data Entry Operator – Records transactions and data.
- Students – Use the system for learning DBMS concepts.
- Local Clinic Receptionist – Can use it to maintain Appointment records.

4.4 Future Expandability

The system can be extended in future to include:

 Integration of a complete Hospital Management System including billing and payment modules.

 Addition of Electronic Medical Records (EMR) to store patient medical history, prescriptions, and diagnoses.

 Development of a web or mobile application interface for online appointment booking and management.

 Implementation of automated SMS and email notifications for appointment reminders.

 Addition of advanced reporting and analytics features for workload and performance analysis.

CHAPTER 5: REQUIREMENT SPECIFICATION

This chapter describes the hardware and software requirements needed to implement the Stock Transaction Recording System.

5.1 Hardware Requirements :

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

5.2 Software Requirements :

Software	Purpose
MySQL Server	Database management
MySQL Workbench	SQL query execution and database design
Operating System	Windows / Linux
SQL	Query language

MS Word	Documentation preparation

CHAPTER 6: SYSTEM DESIGN

System design describes how the database system works conceptually.

6.1 System Architecture (Conceptual):

The system consists of the following components:

- User
- MySQL Workbench (Interface)
- MySQL Server (Database Engine)

Working:

1. The user enters SQL queries using MySQL Workbench.
2. The queries are sent to the MySQL Server.
3. The server processes the queries.
4. Results are returned to the user.
5. Data is stored in relational tables.

6.2 Data Flow Description

Receptionist enters Patient or Appointment data.

Data is validated by constraints.

Data is stored in tables.

Reports are generated using SELECT queries.

6.3Process Flow (Textual)

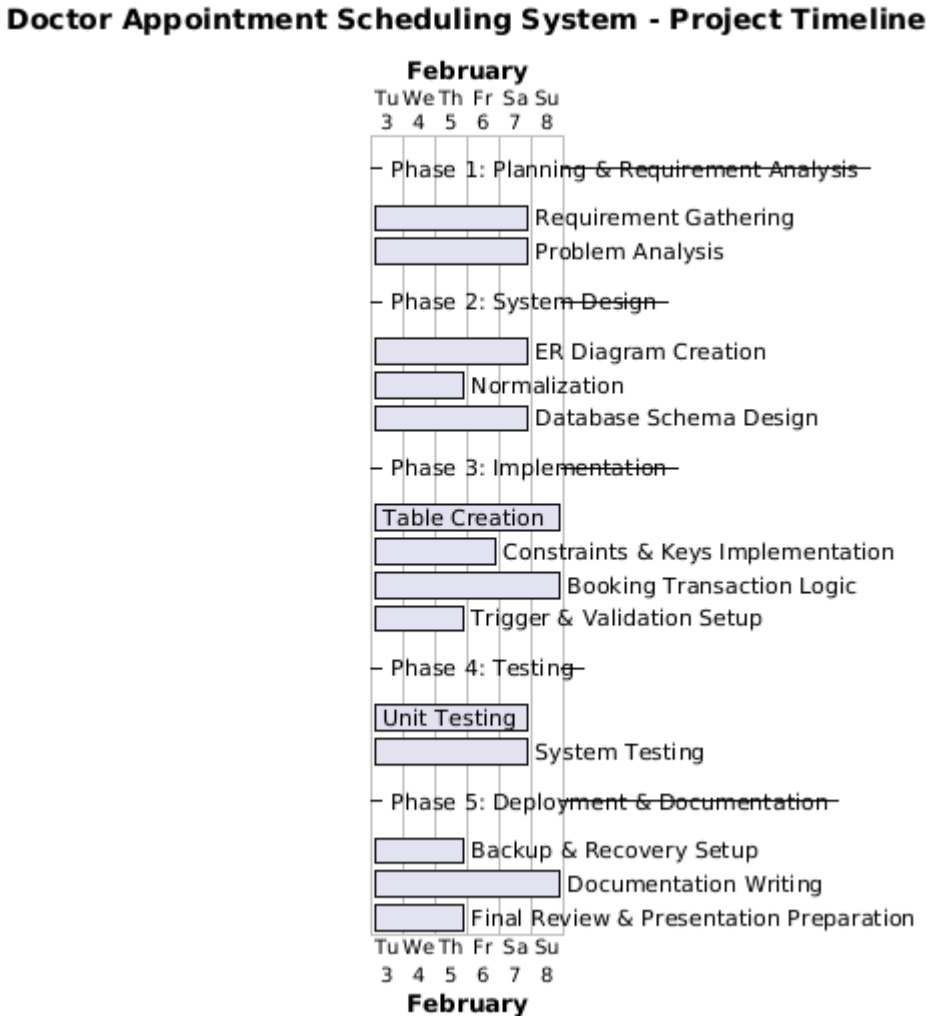
User → SQL Query → MySQL Server → Database Tables → Result Output

PROJECT SCHEDULE (GANTT CHART)

A Gantt Chart is used to represent the project timeline and activities.

Table 6.1: Project Schedule (Gantt Chart) Activity

Table 6.1: Project Schedule (Gantt Chart)



CHAPTER 7: DATABASE DESIGN

Database design is one of the most important phases of this project. It defines how data is structured, stored, and related to each other. A well designed database reduces redundancy and improves data integrity.

The Stock Transaction Recording System uses a relational database model. The database is designed using MySQL and consists of four main tables: User, Stock, Transaction, and Portfolio.

7.1 Entity Description

1. Doctors Entity

Attributes:

a. doctor_id (Primary Key)
b. doctor_name
c. specialization
d. phone_number
e. email
f. joining_date

2. Patients Entity Attributes:

a. patient_id(primary key)

b. patient_name

c. phone

d. email

e. age

f. gender

3. Doctor_Availability Entity

a. availability_id (Primary Key)

b. doctor_id (Foreign Key)

c. doctor_name

d. available_date

d. start_time

e. end_time

f. max_patients

4.. Appointments Entity

Attributes:

a. appointment_id (Primary Key)
b. doctor_id (Foreign Key)
c. patient_id (Foreign Key)
d. appointment_date
e. appointment_time
f. status (Booked / Cancelled / Completed)

7.2 Table Structure

Table 7.1: Doctors Table

Field Name	Data Type	Description
doctor_id	INT	Unique ID of Doctor
doctor_name	VARCHAR(100)	Name of Doctor
Specialization	VARCHAR(100)	Doctor's Specialization
Phone	VARCHAR(20)	Contact number
Email	VARCHAR(100)	Email of Doctor
Joining_date	TIMESTAMP	Joining date of Doctor

Table 7.2: Patients Table

Field Name	Data Type	Description
patient_id	INT	Unique ID of Patient
patient_name	VARCHAR(100)	Name of Patient
Phone	VARCHAR(15)	Contact Number

Email	VARCHAR(100)	Email of Patient
Age	INT	Age of Patient
Gender	VARCHAR(20)	Gender of Patient

Table 7.3: Doctor_availability Table

Field Name	Data Type	Description
avaliability_id	INT	Unique ID For Doctor Available
doctor_id	INT	References doctors(doctor_id)
doctor_name	INT	Name of Doctor
Available_date	DATE	Date on which doctor is avaiilable
Start_time	TIME	Starting duty time
End_time	TIME	Ending duty time
Max_patients	INT	Max. number of patients in a single day

Table 7.4: Appointments Table

Field Name	Data Type	Description

appointment_id	INT	Unique ID of Appointment
doctor_id	INT	References doctors(doctor_id)
patient_id	INT	References patients(patient_id)
appointment_date	DATE	Date of Appointment
appointment_time	TIMESTAMP	Appointment time
Status	Enum('Booked','Cancelled', 'Completed')	Appointment Status

7.4 Constraints Used Constraints are used to maintain data accuracy and consistency.

Constraints Used in the Database

The system uses several integrity constraints to ensure reliability and prevent conflicts.

1 Primary Key Constraint

- Ensures each record is uniquely identifiable.
- Used in all tables (doctor_id, patient_id, appointment_id, etc.)

2 Foreign Key Constraint

- Maintains referential integrity between tables.
- Example:
    - appointment.doctor_id → references doctor.doctor_id
    - appointment.patient_id → references patient.patient_id

3 NOT NULL Constraint

- Prevents empty critical fields.
- Example:
    - doctor_name
    - appointment_date

4 UNIQUE Constraint

- Prevents duplicate entries.
- Example:
    - phone in doctors
    - email in doctors/patients (optional)

5 CHECK Constraint (Logical Validation)

- Ensures valid data entry.
- Example:
    - max_patients_per_day > 0
    - appointment_date >= CURRENT_DATE

6 ENUM Constraint

- Restricts role and status to predefined values.
- Example:
    - status: Booked, Cancelled, Completed

7 Transaction Control

- Used to prevent double booking.
- Ensures atomic booking process.

7.5 Transaction for Appointment booking Operation

In this system, every appointment booking or cancellation activity is treated as a database transaction.

When the admin registers a new appointment:

A new record is inserted into the Appointments table. The Appointments table is updated by adding the appointment details.

Although, The system has to check if the doctor is available on the day of appointment, or the time or if the no. of max patients hasn't exceeded.

If this is the case, then no values are added.

7.6 Data Consistency

Data consistency means that the appointments updated in the Appointments table must always match the availability of doctor and patient.

This is ensured by:

- Using constraints

- Updating Appointments only after valid transactions

- Preventing booking a doctor on the date he/she is not available on, the end time is exceeded or the doctor has achieved max no. of patients in a day. In such cases, the records are not updated and the transaction is cancelled.

7.7 Rollback Example

If an error occurs during appointment updation(Like doctor unavailability or slots full), the transaction is rolled back.
Rollback cancels all changes made during the operation. This ensures that:

- No duplicate appointment entry is done

- Appointment table remain consistent

- Database integrity is preserved

Thus, either all steps succeed or none are applied.

CHAPTER 8: UML DIAGRAMS

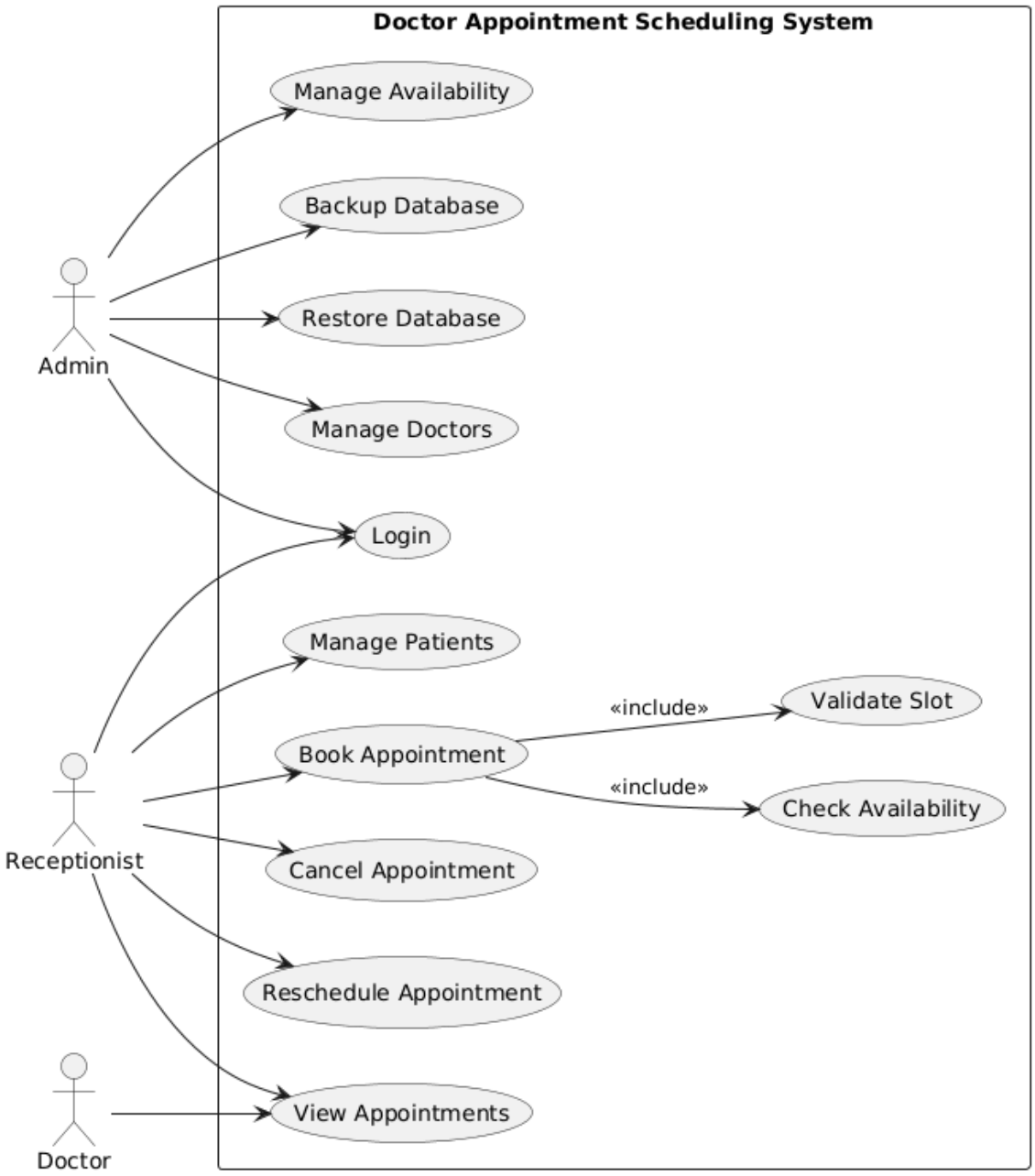
Unified Modeling Language (UML) diagrams are used to represent the structure and behavior of the system in a graphical form. These diagrams help in understanding how different components of the system interact with each other and how data flows through the system.

For the Stock Transaction Record System, the following UML diagrams are used:

- ER Diagram
- Use Case Diagram
- Sequence Diagram
- Activity Diagram

These diagrams provide a clear visualization of the database structure, user interactions, and transaction flow.

- USE CASE DIAGRAM:



- **The Use Case Diagram represents the interaction between the Admin, Receptionist, Doctor and the system. It shows the different operations that they can perform on the system.**

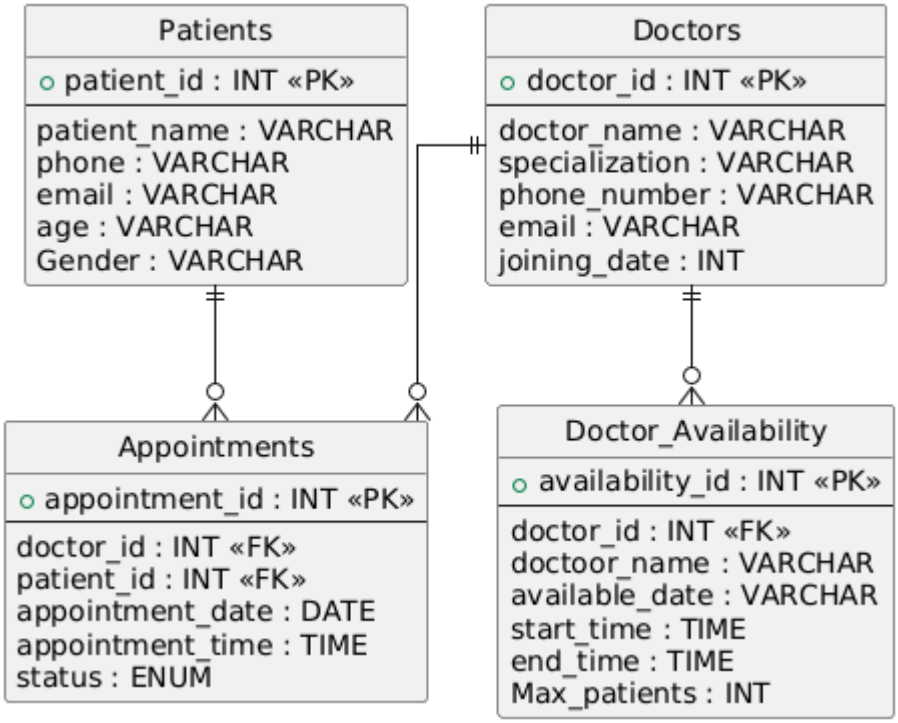
- *Actor:* ***Admin***
- *Use Cases:*
- *Manage Database*
- *Backup Database*
- *Restore Database*
- *Manage Doctors*
- *Log into the system*

- *Actor:* ***Receptionist***
- *Use Cases:*
- *Manage Patients*
- *Book Appointments(Includes slot validation and availability checking)*
- *Cancel appointment*

- *Actor:* ***Doctor***
- *Use Cases:*
- *View Appointments*
- ***<u>This diagram indicates that the Admin and Receptionist can both Log into the system but have different operations to do while the doctor can only access the appointments.</u>***

ER Relationship :

DOCTOR APPOINTMENT SCHEDULING



The ER (Entity Relationship) Diagram represents the database structure of the system and the relationships between different entities.

Entities:

- doctor – Stores information about the doctor's information.
- patients – Stores details of patient's information.
- Doctor_Availability – Stores Availability record of doctors.
- Appointments – Stores the appointment data.

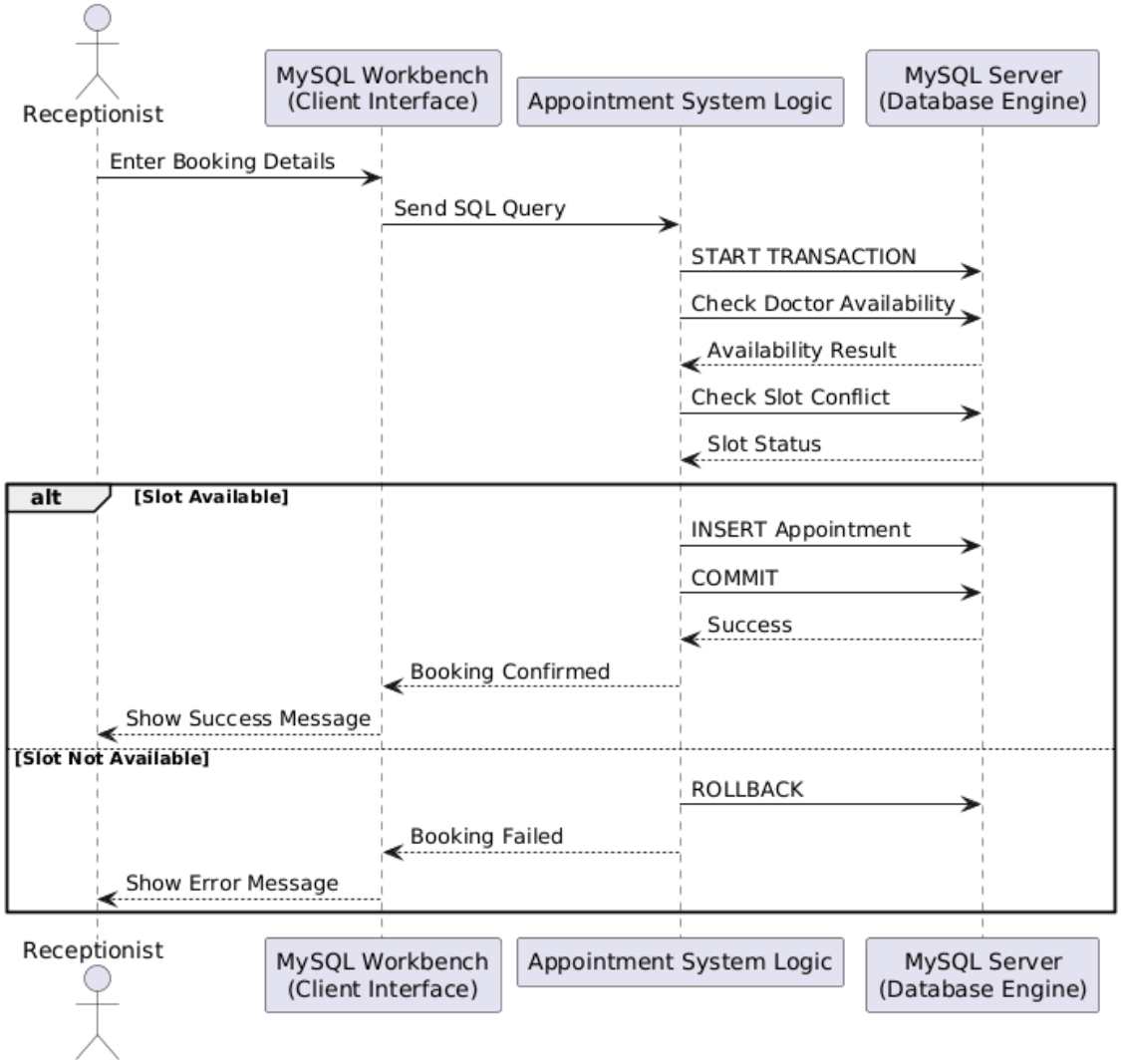
Relationships:

- One doctor can have many appointments.
- One patient can book multiple appointments.

- One doctor can have multiple availability records.

The ER diagram shows how the primary keys and foreign keys connect the tables and ensure data integrity in the database.

SEQUENCE DIAGRAM :



Sequence Diagram shows how objects interact in a sequence of time.

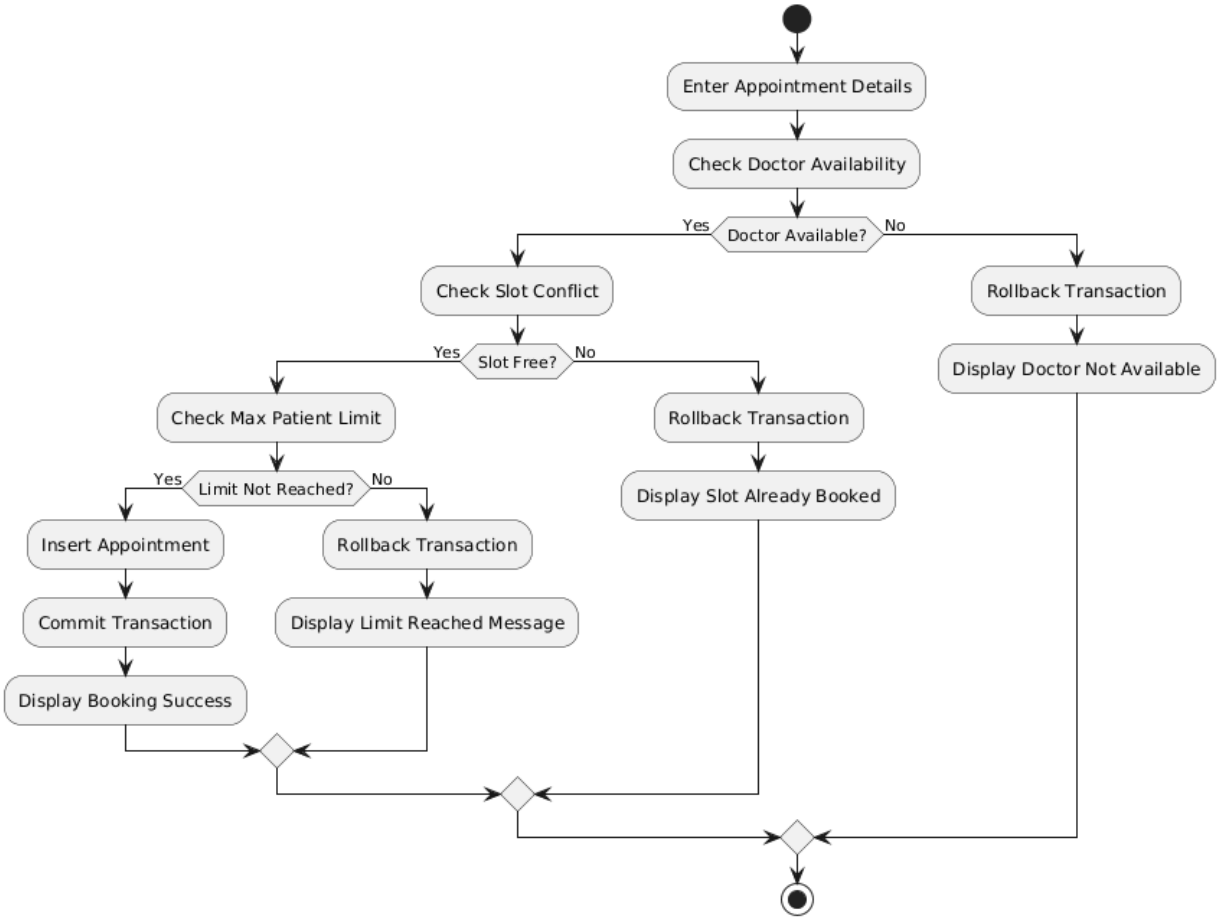
Objects:

- Receptionist
- MySQL Workbench
- MySQL Server
- Database Tables

Interaction Flow:

1. User sends SQL query
2. MySQL Workbench forwards query
3. MySQL Server processes query
4. Database tables are accessed
5. Result is returned
6. Output is shown to receptionist

- **ACTIVITY DIAGRAM**



An Activity Diagram represents the flow of activities in the system from start to end.

Steps in Activity Flow:

Start

User enters SQL query

System validates data

If data is valid → Insert/Update records

If data is invalid → Show error

Update Appointments

Store transaction record

Display result

End

CHAPTER 9: SQL IMPLEMENTATION

This chapter describes the SQL commands used to implement the database for the Stock Transaction Record System. SQL is used to create the database, define tables, insert records, retrieve data, and manage transactions. Transaction control commands such as COMMIT and ROLLBACK are used to maintain data consistency during buy and sell operations.

9.1 DATABASE CREATION

```
CREATE DATABASE doctor_appointment_db;

USE doctor_appointment_db;
```

Explanation:

The above commands create a database named doctor_appointment_db and select it for further operations.

9.2 TABLE CREATION

9.2.1 Doctor Table

```
CREATE TABLE doctors (

    doctor_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    doctor_name VARCHAR(100) NOT NULL,
    specialization VARCHAR(100) NOT NULL,
    phone VARCHAR(15) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Explanation:

This table stores Doctor's information. The doctor_id uniquely identifies each doctor.

9.2.2 Patients Table

```
CREATE TABLE patients (
    patient_id INT AUTO_INCREMENT PRIMARY KEY,
    patient_name VARCHAR(100) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    email VARCHAR(100),
    dob DATE,
    CONSTRAINT uq_patient_contact UNIQUE (phone, email)
);
```

Explanation:

This table stores Patients details such as patient's name, contact details, etc.

9.2.3 doctor_availability Table

```
CREATE TABLE doctor_availability (
    availability_id INT AUTO_INCREMENT PRIMARY KEY,
    doctor_id INT NOT NULL,
    available_date DATE NOT NULL,
    start_time TIME NOT NULL,
    end_time TIME NOT NULL,
    max_patients INT NOT NULL,

    CONSTRAINT fk_av_doctor
        FOREIGN KEY (doctor_id) REFERENCES doctors(doctor_id)
        ON DELETE CASCADE,

    CONSTRAINT chk_time_valid
        CHECK (start_time < end_time),
```

```
    CONSTRAINT uq_doctor_slot
        UNIQUE (doctor_id, available_date, start_time, end_time)
);
```

Explanation:

This table stores all the records of the doctors available dates for appointment.

The Uses of contraints is for checking invalid time ranges and checking duplicate available slots.

9.2.4 Appointments Table

```
CREATE TABLE appointments (
    appointment_id INT AUTO_INCREMENT PRIMARY KEY,
    doctor_id INT NOT NULL,
    patient_id INT NOT NULL,
    appointment_date DATE NOT NULL,
    appointment_time TIME NOT NULL,
    status ENUM('BOOKED', 'CANCELLED', 'COMPLETED') DEFAULT 'BOOKED',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    CONSTRAINT fk_app_doctor
        FOREIGN KEY (doctor_id) REFERENCES doctors(doctor_id),

    CONSTRAINT fk_app_patient
        FOREIGN KEY (patient_id) REFERENCES patients(patient_id),

    CONSTRAINT uq_doctor_time
        UNIQUE (doctor_id, appointment_date, appointment_time)
);
```

Explanation:

This table stores the Appointment updates.

The UNIQUE constraint ensures that a same doctor cannot be booked at the exact same date and exact same time.

And also ensuring that same doctor can book multiple doctors

9.3 DATA INSERTION

Insert into Doctor Table

```
INSERT INTO doctors

(doctor_id, doctor_name, specialization, phone, email, joining_date)

VALUES

(1, 'Dr. Sudhir Sharma', 'General Physician', '772564895',
'SSharma@gmail.com', '2026-02-07 12:02:02'),

(2, 'Dr. Roger Dsouza', 'Orthopedic doctor', '9891758265',
'dsouza@gmail.com', '2022-10-01 07:32:00'),

(3, 'Dr. Sheetal Pal', 'Cardiologist', '713984049', 'SheetalP@gmail.com',
'2020-05-15 10:00:00'),

(4, 'Dr. Rajdev Tiwari', 'General Physician', '98934554129',
'PrajotTiwari@gmail.com', '2020-05-29 07:30:06'),

(5, 'Dr. Manoj Mhatre', 'General Physician', '7209180857',
'DMhatre@gmail.com', '2018-05-10 07:30:00'),

(6, 'Surgeon Pankaj Mishra', 'Neuro-Surgeon', '9960854240',
'HM@gmail.com', '2018-11-10 11:48:00'),

(7, 'Dr. Aman Khan', 'ENT Specialist', '9345678123',
'aman.khan@gmail.com', '2022-08-25 09:00:00'),

(8, 'Dr. payal jain', 'Gynaecologist', '9699925479', 'Payal2234@gmail.com',
'2021-06-12 09:00:00'),

(9, 'Dr. Anjali Mehta', 'Dermatologist', '9123456789',
'anjali.mehta@gmail.com', '2020-03-20 10:00:00'),

(10, 'Dr. Sneha Patil', 'General Physician', '9012345678',
'sneha.patil@gmail.com', '2023-01-12 08:15:00');
```

Insert into Patients Table

```
INSERT INTO patients
(patient_id, patient_name, phone, email, age, gender)
VALUES
(1, 'Amit Verma', '8888888888', NULL, 21, 'Male'),
(2, 'Arpit Mishra', '9892795409', 'amishra@gmail.com', 19, 'Male'),
(3, 'Irfan Maniyar', '9899542639', 'IrfanM@gmail.com', 20, 'Male'),
(4, 'Kevin Mariam', '7710964279', 'KevM@gmail.com', 19, 'Male'),
(5, 'Harshit Mishra', '7208780667', 'Harshit2234@gmail.com', 24, 'Male'),
(6, 'Sudha Prajapati', '8936542511', 'SP@gmail.com', 31, 'Female'),
(7, 'Anita Shah', '9692145879', 'Anita785@gmail.com', 29, 'Female');
```

Insert into doctor_availability Table

```
INSERT INTO doctor_availability
(availability_id, doctor_id, doctor_name, available_date, start_time,
end_time, max_patients)
VALUES
(101, 1, 'Dr. Sudhir Sharma', '2026-02-23', '09:00:00', '14:00:00', 8),
(102, 2, 'Dr. Roger Dsouza', '2026-02-23', '09:00:00', '12:00:00', 10),
(103, 3, 'Dr. Sheetal Pal', '2026-02-28', '09:00:00', '12:00:00', 10),
(104, 4, 'Dr. Rajdev Tiwari', '2026-02-23', '09:00:00', '14:00:00', 8),
(105, 5, 'Dr. Manoj Mhatre', '2026-03-02', '11:00:00', '17:00:00', 10),
(106, 6, 'Surgeon Pankaj Mishra', '2026-03-04', '07:30:00', '12:00:00', 15),
(107, 7, 'Dr. Aman Khan', '2026-03-05', '09:00:00', '14:00:00', 10),
(108, 8, 'Dr. Payal Jain', '2026-03-05', '11:00:00', '19:00:00', 20),
(109, 9, 'Dr. Anjali Mehta', '2026-03-09', '07:30:00', '13:00:00', 15),
```

(110, 10, 'Dr. Sneha Patil', '2026-03-12', '07:30:00', '13:00:00', 15);

9.4 DATA RETRIEVAL QUERIES

View all doctors

SELECT * FROM doctors;

View all patients

SELECT * FROM patients;

Check available doctors

SELECT * FROM doctor_availability;

9.5 TRANSACTION MANAGEMENT (APPOINTMENT BOOKING)

The BOOKING operation must insert a transaction record and update the Appointments table in a single transaction.

```
START TRANSACTION;


INSERT INTO appointments

(doctor_id, patient_id, appointment_date, appointment_time)

SELECT

   1,  -- doctor_id

   1,  -- patient_id

   '2026-03-10',

   '10:00:00'

FROM doctor_availability da

WHERE da.doctor_id = 1

AND da.available_date = '2026-03-10'

AND '10:00:00' BETWEEN da.start_time AND da.end_time

AND (

   SELECT COUNT(*)
```

```
    FROM appointments a
    WHERE a.doctor_id = 1
    AND a.appointment_date = '2026-03-10'
) < da.max_patients;
COMMIT;
```

Explanation:

If the doctor is available on the date, the time is in the range of duty and unique and if the slots for the day are left, only then the transaction is committed and changes are saved permanently.

9.6 TRANSACTION MANAGEMENT (UPDATING/CANCELLING APPOINTMENT)

Before Updating the Appointment on a next date, the system needs to verify if the doctor is available on the same date.

```
START TRANSACTION;

UPDATE appointments a
JOIN doctor_availability da
```

```
ON da.doctor_id = a.doctor_id
SET
   a.appointment_date = '2026-03-10',
   a.appointment_time = '11:00:00'
WHERE a.appointment_id = 1
AND da.available_date = '2026-03-10'
AND '11:00:00' BETWEEN da.start_time AND da.end_time
AND (
   SELECT COUNT(*)
   FROM appointments
   WHERE doctor_id = a.doctor_id
   AND appointment_date = '2026-03-10'
) < da.max_patients;

FOR CANCELLATION;

UPDATE appointments
SET status = 'Cancelled'
WHERE appointment_id = 1;
```

9.7 ADVANCED QUERIES

Show Total Appointments by Doctor (GROUP BY + JOIN)

```
SELECT
   d.doctor_name,
   COUNT(a.appointment_id) AS total_appointments
FROM doctors d
LEFT JOIN appointments a
   ON d.doctor_id = a.doctor_id
GROUP BY d.doctor_id
ORDER BY total_appointments DESC;
```

Show Available Slots Per Doctor (SUBQUERY)

```
SELECT
   da.doctor_id,
   da.available_date,
   da.max_patients - (
      SELECT COUNT(*)
      FROM appointments a
      WHERE a.doctor_id = da.doctor_id
```

```
        AND a.appointment_date = da.available_date
    ) AS slots_remaining
FROM doctor_availability da;
```

9.8 VIEW CREATION

```
CREATE VIEW doctor_schedule AS

SELECT

    d.doctor_name,

    a.appointment_date,

    a.appointment_time,

    p.patient_name

FROM appointments a

JOIN doctors d ON a.doctor_id = d.doctor_id

JOIN patients p ON a.patient_id = p.patient_id;
```

Thus, the Doctor Appointment Scheduling System ensures data integrity and consistency using transaction management techniques.

CHAPTER 10: SYSTEM TESTING AND RESULT

System testing is performed to verify whether the developed database system works correctly and produces accurate results. It ensures that all SQL queries, constraints, and transactions operate as expected.

The Doctor Appointment Scheduling System is tested using different SQL queries for insertion, retrieval, updating, and deletion of data. Sample data is used to validate the working of the system.

10.1 Testing Strategy

The following testing methods are used:

- Query execution testing
- Data validation testing
- Constraint testing
- Transaction testing
- Result verification

Each module of the system is tested independently and then tested as an integrated system.

10.2 Test Cases

Table 10.1: Test Cases:

Test Case ID	Operation	Input	Expected Result	Status

TC01	Insert Doctor	Valid Doctor data	Record inserted successfully	Pass
TC02	Insert Patient	Valid Patient data	patient record inserted	Pass
TC03	Insert Doctor Avaliability	Valid availability record	Avaliability record updated	Pass
TC04	Booking Transaction	Doctor -> Avaliable, Slots -> Avaliable	Transaction Successful	Pass
TC05	View Appointment Report	SELECT query	Correct appointment list displayed	Pass

10.3 Sample Output (Result Tables):

10.3.1 Database Creation and Use Database.

```
mysql> create database doctor_appointment_db;
ERROR 1007 (HY000): Can't create database 'doctor_appointment_db'; database exists
mysql> use doctor_appointment_db;
Database changed
mysql>
```

10.3.2 Tables Creation (doctors, patients, doctor_avaliability, Appointments)

10.3.3 A. doctors Table:

```
mysql> CREATE TABLE doctors (
    ->     doctor_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     doctor_name VARCHAR(100) NOT NULL,
    ->     specialization VARCHAR(100) NOT NULL,
    ->     phone VARCHAR(15) UNIQUE NOT NULL,
    ->     email VARCHAR(100) UNIQUE,
    ->     joining_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
```

B. patients Table:

```
mysql> CREATE TABLE patients (
    ->     patient_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     patient_name VARCHAR(100) NOT NULL,
    ->     phone VARCHAR(15) NOT NULL,
    ->     email VARCHAR(100),
    ->     age INT,
    ->     Gender VARCHAR(20),
    ->     CONSTRAINT uq_patient_contact UNIQUE (phone,email)
    -> );
```

C. Doctor Availability Table(Buy/Sell Operation):

```
mysql> CREATE TABLE doctor_availability (
    ->     availability_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     doctor_id INT NOT NULL,
    ->     available_date DATE NOT NULL,
    ->     start_time TIME NOT NULL,
    ->     end_time TIME NOT NULL,
    ->     max_patients INT,
    ->
    -> CONSTRAINT fk_av_doctor
    ->         FOREIGN KEY (doctor_id) REFERENCES doctors(doctor_id)
    ->         ON DELETE CASCADE,
    ->
    ->     CONSTRAINT chk_time_valid
    ->         CHECK (start_time < end_time),
    ->
    ->     CONSTRAINT uq_doctor_slot
    ->         UNIQUE (doctor_id, available_date, start_time, end_time)
    -> );
```

D. Appointments Table:

```
mysql> CREATE TABLE appointments (
    ->     appointment_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     doctor_id INT NOT NULL,
    ->     patient_id INT NOT NULL,
    ->     appointment_date DATE NOT NULL,
    ->     appointment_time TIME NOT NULL,
    ->     status ENUM('BOOKED', 'CANCELLED', 'COMPLETED') DEFAULT 'BOOKED',
    ->     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ->
    ->     CONSTRAINT fk_app_doctor
    ->         FOREIGN KEY (doctor_id) REFERENCES doctors(doctor_id),
    ->
    ->     CONSTRAINT fk_app_patient
    ->         FOREIGN KEY (patient_id) REFERENCES patients(patient_id),
    ->
    ->     CONSTRAINT uq_doctor_time
    ->         UNIQUE (doctor_id, appointment_date, appointment_time)
    -> );
```

10.3.3 Table Value Insertion and Display:

A. Insert into doctors table:

```
mysql> INSERT INTO doctors
    -> (doctor_id, doctor_name, specialization, phone, email, joining_date)
    -> VALUES
    -> (1, 'Dr. Sudhir Sharma', 'General Physician', '772564895', 'SSharma@gmail.com', '2026-02-07 12:02:02'),
    -> (2, 'Dr. Roger Dsouza', 'Orthopedic doctor', '9891758265', 'dsouza@gmail.com', '2022-10-01 07:32:00'),
    -> (3, 'Dr. Sheetal Pal', 'Cardiologist', '713984049', 'SheetalP@gmail.com', '2020-05-15 10:00:00'),
    -> (4, 'Dr. Rajdev Tiwari', 'General Physician', '98934554129', 'PrajotTiwari@gmail.com', '2020-05-29 07:30:06'),
    -> (5, 'Dr. Manoj Mhatre', 'General Physician', '7209180857', 'DMhatre@gmail.com', '2018-05-10 07:30:00'),
    -> (6, 'Surgeon Pankaj Mishra', 'Neuro-Surgeon', '9960854240', 'HM@gmail.com', '2018-11-10 11:48:00'),
    -> (7, 'Dr. Aman Khan', 'ENT Specialist', '9345678123', 'aman.khan@gmail.com', '2022-08-25 09:00:00'),
    -> (8, 'Dr. payal jain', 'Gynaecologist', '9699925479', 'Payal2234@gmail.com', '2021-06-12 09:00:00'),
    -> (9, 'Dr. Anjali Mehta', 'Dermatologist', '9123456789', 'anjali.mehta@gmail.com', '2020-03-20 10:00:00'),
    -> (10, 'Dr. Sneha Patil', 'General Physician', '9012345678', 'sneha.patil@gmail.com', '2023-01-12 08:15:00');
```

- Display doctors Table:

```
mysql> select * from doctors;
+-----------+----------------------+-------------------+-------------+-----------------------+---------------------+
| doctor_id | doctor_name          | specialization    | phone       | email                 | joining_date        |
+-----------+----------------------+-------------------+-------------+-----------------------+---------------------+
|         1 | Dr. Sudhir Sharma    | General Physician | 772564895   | SSharma@gmail.com     | 2026-02-07 12:02:02 |
|         2 | Dr. Roger Dsouza     | Orthopedic doctor | 9891758265  | dsouza@gmail.com      | 2022-10-01 07:32:00 |
|         3 | Dr. Sheetal Pal      | Cardiologist      | 713984049   | SheetalP@gmail.com    | 2020-05-15 10:00:00 |
|         4 | Dr. Rajdev Tiwari    | General Physician | 98934554129 | PrajotTiwari@gmail.com | 2020-05-29 07:30:06 |
|         5 | Dr. Manoj Mhatre     | General Physician | 7209180857  | DMhatre@gmail.com     | 2018-05-10 07:30:00 |
|         6 | Surgeon Pankaj Mishra | Neuro-Surgeon     | 9960854240  | HM@gmail.com          | 2018-11-10 11:48:00 |
|         7 | Dr. Aman Khan        | ENT Specialist    | 9345678123  | aman.khan@gmail.com   | 2022-08-25 09:00:00 |
|         8 | Dr. payal jain       | Gynaecologist     | 9699925479  | Payal2234@gmail.com   | 2021-06-12 09:00:00 |
|         9 | Dr. Anjali Mehta     | Dermatologist     | 9123456789  | anjali.mehta@gmail.com | 2020-03-20 10:00:00 |
|        10 | Dr. Sneha Patil      | General Physician | 9012345678  | sneha.patil@gmail.com | 2023-01-12 08:15:00 |
+-----------+----------------------+-------------------+-------------+-----------------------+---------------------+
10 rows in set (0.01 sec)
```

B. Insert into patients Table:

```
mysql> INSERT INTO patients
    -> (patient_id, patient_name, phone, email, age, gender)
    -> VALUES
    -> (1, 'Amit Verma', '8888888888', NULL, 21, 'Male'),
    -> (2, 'Arpit Mishra', '9892795409', 'amishra@gmail.com', 19, 'Male'),
    -> (3, 'Irfan Maniyar', '9899542639', 'IrfanM@gmail.com', 20, 'Male'),
    -> (4, 'Kevin Mariam', '7710964279', 'KevM@gmail.com', 19, 'Male'),
    -> (5, 'Harshit Mishra', '7208780667', 'Harshit2234@gmail.com', 24, 'Male'),
    -> (6, 'Sudha Prajapati', '8936542511', 'SP@gmail.com', 31, 'Female'),
    -> (7, 'Anita Shah', '9692145879', 'Anita785@gmail.com', 29, 'Female');
```

- Display Patients Table:

```
mysql> select * from patients;
+------------+-----------------+------------+-----------------------+------+--------+
| patient_id | patient_name    | phone      | email                 | age  | Gender |
+------------+-----------------+------------+-----------------------+------+--------+
|          1 | Amit Verma      | 8888888888 | NULL                  |   21 | Male   |
|          2 | Arpit Mishra    | 9892795409 | amishra@gmail.com     |   19 | Male   |
|          3 | Irfan Maniyar   | 9899542639 | IrfanM@gmail.com      |   20 | Male   |
|          4 | Kevin Mariam    | 7710964279 | KevM@gmail.com        |   19 | Male   |
|          5 | Harshit Mishra  | 7208780667 | Harshit2234@gmail.com |   24 | Male   |
|          6 | Sudha Prajapati | 8936542511 | SP@gmail.com          |   31 | Female |
|          7 | Anita Shah      | 9692145879 | Anita785@gmail.com    |   29 | Female |
+------------+-----------------+------------+-----------------------+------+--------+
7 rows in set (0.01 sec)
```

C. Insert Into doctor_avaliability:

```
-> INSERT INTO doctor_availability
-> (availability_id, doctor_id, doctor_name, available_date, start_time, end_time, max_patients)
-> VALUES
-> (101, 1, 'Dr. Sudhir Sharma', '2026-02-23', '09:00:00', '14:00:00', 8),
-> (102, 2, 'Dr. Roger Dsouza', '2026-02-23', '09:00:00', '12:00:00', 10),
-> (103, 3, 'Dr. Sheetal Pal', '2026-02-28', '09:00:00', '12:00:00', 10),
-> (104, 4, 'Dr. Rajdev Tiwari', '2026-02-23', '09:00:00', '14:00:00', 8),
-> (105, 5, 'Dr. Manoj Mhatre', '2026-03-02', '11:00:00', '17:00:00', 10),
-> (106, 6, 'Surgeon Pankaj Mishra', '2026-03-04', '07:30:00', '12:00:00', 15),
-> (107, 7, 'Dr. Aman Khan', '2026-03-05', '09:00:00', '14:00:00', 10),
-> (108, 8, 'Dr. Payal Jain', '2026-03-05', '11:00:00', '19:00:00', 20),
-> (109, 9, 'Dr. Anjali Mehta', '2026-03-09', '07:30:00', '13:00:00', 15),
-> (110, 10, 'Dr. Sneha Patil', '2026-03-12', '07:30:00', '13:00:00', 15);
```

- Display doctor_availability table:

```
mysql> select * from doctor_availability;
+-----------------+-----------+-----------------------+----------------+------------+----------+--------------+
| availability_id | doctor_id | doctor_name           | available_date | start_time | end_time | max_patients |
+-----------------+-----------+-----------------------+----------------+------------+----------+--------------+
|             101 |         1 | Dr. Sudhir Sharma     | 2026-02-23     | 09:00:00   | 14:00:00 |            8 |
|             102 |         2 | Dr. Roger Dsouza      | 2026-02-23     | 09:00:00   | 12:00:00 |           10 |
|             103 |         3 | Dr. Sheetal Pal       | 2026-02-28     | 09:00:00   | 12:00:00 |           10 |
|             104 |         4 | Dr. Rajdev Tiwari     | 2026-02-23     | 09:00:00   | 14:00:00 |            8 |
|             105 |         5 | Dr. Manoj Mhatre      | 2026-03-02     | 11:00:00   | 17:00:00 |           10 |
|             106 |         6 | Surgeon Pankaj Mishra | 2026-03-04     | 07:30:00   | 12:00:00 |           15 |
|             107 |         7 | Dr. Aman Khan         | 2026-03-05     | 09:00:00   | 14:00:00 |           10 |
|             108 |         8 | Dr. Payal Jain        | 2026-03-05     | 11:00:00   | 19:00:00 |           20 |
|             109 |         9 | Dr. Anjali Mehta      | 2026-03-09     | 07:30:00   | 13:00:00 |           15 |
|             110 |        10 | Dr. Sneha Patil       | 2026-03-12     | 07:30:00   | 13:00:00 |           15 |
+-----------------+-----------+-----------------------+----------------+------------+----------+--------------+
10 rows in set (0.02 sec)
```

C. Insert into Appointments Table(BOOKING TRANSACTION):

- Appointment Booking Transaction:

```
mysql> start transaction;
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO appointments
    -> (doctor_id, patient_id, appointment_date, appointment_time)
    -> SELECT
    ->     1,  -- doctor_id
    ->     1,  -- patient_id
    ->     '2026-02-10',
    ->     '10:00:00'
    -> FROM doctor_availability da
    -> WHERE da.doctor_id = 1
    -> AND da.available_date = '2026-02-10'
    -> AND '10:00:00' BETWEEN da.start_time AND da.end_time
    -> AND (
    ->     SELECT COUNT(*)
    ->     FROM appointments a
    ->     WHERE a.doctor_id = 1
    ->     AND a.appointment_date = '2026-02-10'
    -> ) < da.max_patients;
```

- Second Entry:

```
mysql> INSERT INTO appointments
    -> (doctor_id, patient_id, appointment_date, appointment_time)
    -> SELECT
    ->     3,  -- doctor_id
    ->     6,  -- patient_id
    ->     '2026-02-28',
    ->     '10:00:00'
    -> FROM doctor_availability da
    -> WHERE da.doctor_id = 3
    -> AND da.available_date = '2026-02-28'
    -> AND '10:00:00' BETWEEN da.start_time AND da.end_time
    -> AND (
    ->     SELECT COUNT(*)
    ->     FROM appointments a
    ->     WHERE a.doctor_id = 3
    ->     AND a.appointment_date = '2026-02-28'
    -> ) < da.max_patients;
```

- Appointments after entry:

```
mysql> select * from appointments;
+----------------+-----------+------------+------------------+------------------+--------+---------------------+
| appointment_id | doctor_id | patient_id | appointment_date | appointment_time | status | created_at          |
+----------------+-----------+------------+------------------+------------------+--------+---------------------+
|              3 |         1 |          1 | 2026-02-10       | 10:00:00         | Booked | 2026-02-07 12:08:36 |
|              4 |         3 |          6 | 2026-02-28       | 10:00:00         | Booked | 2026-02-20 10:25:13 |
+----------------+-----------+------------+------------------+------------------+--------+---------------------+
2 rows in set (0.01 sec)
```

D. Insert into Appointments Table(CANCELLATION OPERATION):

- Start Cancellation Transaction:

```
mysql> update appointments
    -> set status = 'Cancelled'
    -> where appointment_id = 4;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

- Appointments After Cancellation:

```
mysql> select * from appointments;
+----------------+-----------+------------+------------------+------------------+-----------+---------------------+
| appointment_id | doctor_id | patient_id | appointment_date | appointment_time | status    | created_at          |
+----------------+-----------+------------+------------------+------------------+-----------+---------------------+
|              3 |         1 |          1 | 2026-02-10       | 10:00:00         | Booked    | 2026-02-07 12:08:36 |
|              4 |         3 |          6 | 2026-02-28       | 10:00:00         | Cancelled | 2026-02-20 10:25:13 |
+----------------+-----------+------------+------------------+------------------+-----------+---------------------+
2 rows in set (0.00 sec)
```

- Rollback:

```
mysql>
mysql> ROLLBACK;
Query OK, 0 rows affected (0.006 sec)
```

10.3.4 Aggregate Query (GROUP BY + HAVING)

- GROUP BY + JOIN:

```
mysql> SELECT
    ->     d.doctor_name,
    ->     COUNT(a.appointment_id) AS total_appointments
    -> FROM doctors d
    -> LEFT JOIN appointments a
    ->     ON d.doctor_id = a.doctor_id
    -> GROUP BY d.doctor_id
    -> ORDER BY total_appointments DESC;
+----------------------+--------------------+
| doctor_name          | total_appointments |
+----------------------+--------------------+
| Dr. Sudhir Sharma    |                  1 |
| Dr. Sheetal Pal      |                  1 |
| Dr. Roger Dsouza     |                  0 |
| Dr. Rajdev Tiwari    |                  0 |
| Dr. Manoj Mhatre     |                  0 |
| Surgeon Pankaj Mishra |                  0 |
| Dr. Aman Khan        |                  0 |
| Dr. payal jain       |                  0 |
| Dr. Anjali Mehta     |                  0 |
| Dr. Sneha Patil      |                  0 |
+----------------------+--------------------+
```

- SUBQUERY:

```
mysql> SELECT
    ->     da.doctor_id,
    ->     da.available_date,
    ->     da.max_patients - (
    ->         SELECT COUNT(*)
    ->         FROM appointments a
    ->         WHERE a.doctor_id = da.doctor_id
    ->         AND a.appointment_date = da.available_date
    ->     ) AS slots_remaining
    -> FROM doctor_availability da;
+-----------+----------------+-----------------+
| doctor_id | available_date | slots_remaining |
+-----------+----------------+-----------------+
|         1 | 2026-02-23     |               8 |
|         2 | 2026-02-23     |              10 |
|         3 | 2026-02-28     |               9 |
|         4 | 2026-02-23     |               8 |
|         5 | 2026-03-02     |              10 |
|         6 | 2026-03-04     |              15 |
|         7 | 2026-03-05     |              10 |
|         8 | 2026-03-05     |              20 |
|         9 | 2026-03-09     |              15 |
|        10 | 2026-03-12     |              15 |
+-----------+----------------+-----------------+
```

10.3.5 VIEW REPORT:

CREATE VIEW:

```
mysql> CREATE VIEW doctor_schedule AS
    -> SELECT
    ->     d.doctor_name,
    ->     a.appointment_date,
    ->     a.appointment_time,
    ->     p.patient_name
    -> FROM appointments a
    -> JOIN doctors d ON a.doctor_id = d.doctor_id
    -> JOIN patients p ON a.patient_id = p.patient_id;
Query OK, 0 rows affected (0.05 sec)
```

VIEW OUTPUT:

```
mysql> select * from doctor_schedule;
+-------------------+------------------+------------------+-----------------+
| doctor_name       | appointment_date | appointment_time | patient_name    |
+-------------------+------------------+------------------+-----------------+
| Dr. Sudhir Sharma | 2026-02-10       | 10:00:00         | Amit Verma      |
| Dr. Sheetal Pal   | 2026-02-28       | 10:00:00         | Sudha Prajapati |
+-------------------+------------------+------------------+-----------------+
```

10.4 Result Analysis

The results show that:

- Data is inserted successfully into all tables.
- Foreign key relationships are maintained.
- Appointment Status is updated after each transaction.
- Invalid Appointments are not allowed.

The system performs accurately and maintains data consistency.

CHAPTER 11: SECURITY, BACKUP AND RECOVERY

Database security and data protection are essential to prevent unauthorized access and data loss. This system uses MySQL's built-in security and backup features to ensure safe storage of stock transaction records.

11.1 Security

Security is maintained using the following methods:

- User authentication through MySQL login
- Role-based access control
- Granting limited privileges to users
- Use of passwords for database access

Example of User Privilege Control:

CREATE USER 'dbuser'@'localhost' IDENTIFIED BY 'password123';
GRANT SELECT, INSERT, UPDATE ON doctor_appointment_db.* TO 'dbuser'@'localhost';

This ensures that only authorized users can access and modify the database.

11.2 Backup

Backup is the process of creating a copy of the database to prevent data loss due to system failure or accidental deletion.

MySQL provides a tool called mysqldump for database backup.

Backup Command:

Mysqldump -u root -p doctor_appointment_db > doctor_backup.sql

This command creates a backup file named doctor_backup.sql.

11.3 Recovery

Recovery is the process of restoring data from a backup file.

Restore Command:

Mysql -u root -p doctor_appointment_db < doctor_backup.sql

This restores the database to its previous state.

CHAPTER 12: FUTURE SCOPE AND CONCLUSION

Future Scope

The Doctor Appointment Scheduling Database system has been designed as a structured and scalable solution for managing healthcare appointments efficiently. Although the current implementation successfully handles doctor availability, patient records, appointment booking, and conflict prevention, there are several areas where the system can be further expanded and enhanced.

1. **Web-Based Interface Integration**
   The database can be integrated with a web or mobile application interface to allow patients to book appointments online. This would eliminate manual booking and make the system accessible remotely.

2. **Real-Time Slot Allocation System**
   A dynamic slot generation mechanism can be added where appointment slots are automatically divided based on consultation time (e.g., 15 or 30 minutes per patient).

3. **Online Payment Integration**
   Payment gateway integration can be implemented to allow patients to pay consultation fees while booking appointments.

4. **SMS and Email Notifications**
   Automatic reminders and confirmations can be sent to patients via SMS or email to reduce no-shows.

12.2 Conclusion

The Doctor Appointment Scheduling Database system provides a reliable, structured, and centralized solution to manage healthcare appointments efficiently. It eliminates the limitations of manual scheduling systems by

ensuring data consistency, preventing appointment conflicts, and maintaining integrity through constraints and transaction control.
The implementation uses relational database concepts such as primary keys, foreign keys, constraints, transactions, and triggers to ensure accuracy and reliability. Advanced SQL queries and validation checks prevent overbooking and ensure that appointments are scheduled only when doctors are available.
The system improves operational efficiency by organizing doctor schedules, patient records, and appointment data in a structured manner. It reduces redundancy, enhances security through access control mechanisms, and ensures data safety with proper backup and recovery strategies.
Overall, this project demonstrates the practical application of Database Management System (DBMS) concepts in solving real-world healthcare scheduling problems. With further enhancements and integration into a full-stack application, the system has strong potential to be implemented in real healthcare environments.

CHAPTER 13: REFERENCES

1. MySQL Official Documentation https://dev.mysql.com/doc/

2. Korth, H. F., Silberschatz, A., Sudarshan, S.
Database System Concepts, McGraw-Hill.

3. Elmasri, R., Navathe, S.
Fundamentals of Database Systems, Pearson Education.

4. Online SQL learning resources
https://www.w3schools.com/sql/

CHAPTER 14: GLOSSARY

Term.	Description
1. DBMS	Database Management System
2. SQL.	Structured Query Language
3. MySQL	Open-source relational database
4. Primary Key	Unique identifier for a table
5. Foreign Key	Field linking two tables
6. Transaction.	A single logical unit of work
7. Commit.	Saves changes permanently
8. Rollback	Cancels changes
9. ER Diagram	Entity Relationship Diagram
10. Gantt Chart	Project scheduling chart