HEALTH CAMP REGISTRATION & REPORTING SYSTEM

DHANASHREE MONDE
ROLL N0 – 2542029

Established – 1961 **Subject: OSDBMS**

SEVA SADAN'S

R. K. TALREJA COLLEGE

ARTS, SCIENCE & COMMERCE

ULHASNAGAR – 421 003

CERTIFICATE

This is to certify that Mr./Ms. DHANASHREE MONDE of S.Y. Information Technology (SYIT) Roll No. 2542029 has satisfactorily completed the Open Source DataBase Management System Mini Project entitled HEALTH CAMP REGISTRATION AND REPORTING 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**

ACKNOWLEDGEMENT

I would like to express my sincere and heartfelt gratitude to my subject teacher, **Asst. Prof. Sahil Shukla**, for his invaluable guidance, constant encouragement, and continuous support throughout the development of my project titled ***"Health Camp Registration & Reporting System using MySQL."*** His clear explanations, practical insights, and willingness to resolve my doubts helped me understand important database concepts such as ER diagrams, table relationships, primary and foreign keys, normalization, and SQL queries in a much better way. His constructive feedback at every stage of the project motivated me to improve my work and complete it successfully with confidence.

I would also like to extend my gratitude to **R.K. Talreja College of Arts, Science and Commerce** for providing excellent academic facilities, technical resources, and a supportive learning environment that made it possible to complete this project effectively as part of my **SYBSC IT** curriculum. The college has always encouraged practical learning, which helped me gain hands-on experience in database development and implementation
.
I am deeply thankful to my parents for their unconditional support, understanding, and motivation throughout this project. Their encouragement gave me the confidence to overcome challenges and stay focused. I also appreciate my friends and classmates for their cooperation, helpful discussions, and moral support, which made the project work more engaging and productive
.
Additionally, I would like to acknowledge the contribution of various online learning platforms, tutorials, and documentation resources that enhanced my knowledge of MySQL and database management systems. These resources helped me strengthen my technical understanding and improve my practical skills
.
Finally, I express my gratitude to everyone who contributed directly or indirectly to the successful completion of this project. Working on this project has been a valuable learning experience that significantly enhanced my analytical thinking, problem-solving ability, and overall technical proficiency in database management.

TABLE OF CONTENTS

CHAPTER 1 - INTRODUCTION

A database system is an organized way to store, manage, and retrieve data efficiently. It allows information to be stored in structured tables made up of rows and columns, where each table represents a specific entity and each record represents a particular instance of that entity. Through relationships between tables, a database system ensures that related data remains connected and consistent. This structured approach reduces duplication, improves accuracy, and makes data easier to manage. A Database Management System (DBMS) provides the tools and interface needed to perform essential operations such as inserting new records, updating existing information, deleting unwanted data, and retrieving specific details using queries. It also ensures data security, backup, and recovery, making the system reliable for long-term use
.

The **Health Camp Registration & Reporting System** addresses the common problems faced during manual record-keeping in health camps. In many health camps, patient registration, doctor assignments, diagnosis details, prescribed medicines, and report preparation are handled using paper forms or basic spreadsheets. This traditional method is time-consuming, difficult to organize, and highly prone to human errors such as duplicate entries, misplaced forms, and incomplete records. As the number of patients increases, managing files and preparing summary reports becomes even more challenging. Searching for a specific patient's record can take a lot of time, and there is always a risk of data loss or damage. By implementing a database-based system, all camp-related information is stored in a structured and centralized manner. This makes data entry faster, record retrieval easier, and report generation more efficient.

It also improves accuracy, reduces paperwork, and ensures that information is securely stored and maintained.

For this project, **MySQL** has been selected as the Database Management System because of its reliability, performance, and wide usage in both academic and real-world applications. MySQL is an open-source relational database management system, meaning it is free to use and does not require licensing fees, which makes it highly suitable for educational and student projects. It supports important relational database features such as primary keys, foreign keys, constraints, and indexing, which help maintain data integrity and establish proper relationships between tables. MySQL is known for its speed, security features, and ability to handle large amounts of data efficiently. Additionally, it has strong community support and extensive documentation, making it easier for beginners to learn and implement. Due to its simplicity, efficiency, and reliability, MySQL is an ideal choice for developing the Health Camp Registration & Reporting System.

CHAPTER 2 – PROBLEM DEFINITION

In many health camps, patient registration and medical records are kept using manual methods, like paper forms or simple spreadsheets. This disorganized system makes it hard to organize, update, and find information efficiently. As the number of patients and records grows, managing data by hand becomes time-consuming and prone to mistakes. Without a proper system, accuracy, security, and overall camp management suffer.

The major issues in the current system include:

- **Data Redundancy:** The same patient or doctor details can be recorded multiple times, causing duplication.
- **Data Inconsistency:** Manual entry can lead to incorrect or incomplete information, which lowers reliability.
- **Lack of Data Security:** Paper records can be lost, damaged, or accessed by unauthorized individuals.
- **Difficulty in Searching Records:** Finding specific patient or camp details requires manual checks, which are slow and inefficient.
- **No Proper Report Generation:** Preparing summaries, like the total number of patients attended or common diseases, requires manual counting, increasing the chance of mistakes.

To solve these issues, a database-driven solution is necessary. A structured system using MySQL can efficiently store and manage data, reduce duplication, maintain consistency, improve security, and generate reports quickly and accurately.

CHAPTER 3 – OBJECTIVE OF THE PROJECT

The main goal of this project is to design and develop a well-organized Health Camp Registration and Reporting System using MySQL to manage all camp-related data in an efficient, structured, and accurate manner. The system aims to replace manual record-keeping methods with a reliable database solution that ensures faster data entry, easy retrieval of information, and smooth report generation. By implementing this system, the project seeks to demonstrate how a relational database can improve overall data management in real-world healthcare scenarios.

The specific goals of the project are:

- To design a structured database system:
  To create well-organized and logically connected tables such as Patient, Doctor, Health Camp, Registration, and Diagnosis. Each table will store specific types of information in a systematic format, with proper relationships established between them. This structure will reduce redundancy, avoid duplication of data, and ensure that related information can be accessed easily through relational links.
- To implement SQL queries for data management:
  To use essential SQL commands such as CREATE (to create tables), INSERT (to add records), UPDATE (to modify existing data), DELETE (to remove records), and SELECT (to retrieve information). These queries will help in performing all necessary database operations effectively. The project will also focus on writing meaningful queries to generate useful outputs such as patient lists, doctor assignments, and diagnosis reports.
- To ensure data integrity with constraints:
  To apply important constraints such as Primary Key, Foreign Key, NOT NULL, and UNIQUE to maintain data accuracy and consistency. Primary keys will uniquely identify each record, foreign keys will maintain relationships between tables, and other constraints will prevent invalid or duplicate data entries. This ensures that the database remains reliable and logically consistent at all times.
- To generate meaningful reports:
  To create summary reports such as total number of patients registered, number of doctors participating, types of diagnoses recorded, and camp-wise statistics. These reports will help demonstrate how databases can be used to analyze and present information in a clear and organized manner.
- To gain hands-on experience with MySQL:
  To develop practical knowledge of database design, table creation, query writing, and overall database management using MySQL. This project will strengthen understanding of relational database concepts and provide real-world exposure to handling structured data in an academic as well as practical environment.

CHAPTER 4 – SCOPE OF THE PROJECT

The Health Camp Registration and Reporting System aims to create a MySQL-based database to manage health camp activities in an organized and efficient way. The system stores patient registration details, doctor information, camp schedules, and medical records. It replaces manual record-keeping with a structured database to improve accuracy, speed, and reliability.

Areas Where the System Can Be Used:

- ✓ Educational Use:
  Colleges and academic institutions can use this project to teach database design, SQL operations, and report generation. It gives students practical experience with real-world database applications.

- ✓ Administrative Use:
  Camp organizers and administrative staff can manage patient registrations, assign doctors, update medical records, and generate reports efficiently using the system.

- ✓ Healthcare Camp Management:
  Small-scale medical camps organized by NGOs, hospitals, or community groups can use the system to keep proper records and monitor camp activities.

Users of the System:

- ✓ Admin:
  The admin has full access to the database. They can add, update, delete, and view all records. The admin oversees the overall system and generates reports.

- ✓ Staff/User:
  Staff members can register patients, view camp details, and update necessary information during camp operations.

- ✓ Students (Academic Users):
  Students can use the system to learn about database concepts and how data is managed in healthcare applications.

Academic Limitations:

The system is mainly designed for academic use and small-scale health camps.

It does not offer advanced options such as online registration portals or multi-user authentication.
The system operates mainly through MySQL queries and lacks a graphical user interface.
It is not meant for large hospitals with complicated medical record systems.

CHAPTER 5 – REQUIREMENT SPECIFICATION

This section outlines the hardware and software needs for developing and running the Health Camp Registration and Reporting System.

5.1 Hardware Requirements

The system does not need high-end hardware and can operate on a basic computer setup.

Computer / Laptop: Any standard desktop or laptop
Minimum RAM: 4 GB RAM (8 GB is recommended for better performance)
Storage: At least 20 GB of free disk space
Processor: Intel i3 or an equivalent processor

5.2 Software Requirements
The following software tools are required for developing and managing the system:

Software	**Purpose**
MySQL Server	Used for database creation and management
MySQL Workbench	Used for writing and executing SQL queries
Operating System (Windows/Linux)	Platform to run MySQL and related tools
SQL	Query language used to perform database operations

CHAPTER 6 – SYSTEM DESIGN

6.1 System Architecture

The Health Camp Registration and Reporting System follows a simple database-centered architecture based entirely on a Database Management System. The system is developed using MySQL, which stores and manages all data in a structured relational format. This architecture focuses only on how users interact with the database, how the MySQL Server works internally, and how SQL queries are processed. It does not involve hardware components, as the design is purely related to DBMS functionality.

In this system, the user—such as an Admin or authorized Staff member—interacts directly with the database through a query interface like MySQL Workbench. The user performs operations such as registering new patients, entering doctor details, recording diagnoses, updating camp information, deleting unnecessary records, and generating reports. These tasks are carried out using SQL commands like INSERT, UPDATE, DELETE, and SELECT. The user writes the required SQL query and executes it through the interface. This method ensures that all data-related operations are performed in a controlled and systematic manner.

The MySQL Server acts as the core component of the system architecture. It stores all the data in well-structured tables such as Patient, Doctor, Health Camp, Registration, and Diagnosis. Each table is connected using relational concepts like Primary Keys and Foreign Keys. The server ensures that these relationships are properly maintained and that no invalid data is entered into the system. It applies constraints such as NOT NULL to prevent empty fields, UNIQUE to avoid duplicate values, and referential integrity rules to maintain valid links between related tables. This guarantees data accuracy, consistency, and reliability throughout the system.
The query execution flow follows a clear and logical sequence. First, the user writes an SQL query and executes it. Second, the query is sent to the MySQL Server. Third, the server checks the query for syntax errors and validates it according to database rules and constraints. After validation, the server processes the query and performs the required action on the relevant tables. Finally, the result—such as a success message or retrieved data—is returned to the user through the interface.

This structured interaction between the user and the MySQL Server ensures efficient data management, secure handling of information, and accurate report generation. By organizing data in relational tables and processing queries systematically, the system improves performance, reduces errors, and provides a dependable solution for managing health camp records.

CHAPTER 7 – DATABASE DESIGN

7.1 Entity Description

The Health Camp Registration and Reporting System consists of five main entities: Health_Camp, Doctor, Patient, Registration, and Diagnosis. Each table stores specific information and is connected through relationships to maintain data integrity and avoid redundancy.

- **Health_Camp Table**
  The Health_Camp table stores information about each health camp organized. It includes details such as camp ID, camp name, location, date of the camp, and organizer name. Each camp is uniquely identified using camp_id. This table helps manage multiple health camps efficiently and keeps track of when and where each camp is conducted.
- **Doctor Table**
  The Doctor table stores details of doctors participating in the health camps. It includes doctor ID, doctor name, specialization, and phone number. Each doctor is uniquely identified by doctor_id. This table ensures that proper medical professionals are assigned and recorded for diagnosis and treatment during the camp.
- **Patient Table**
  The Patient table maintains personal information about patients attending the health camp. It includes patient ID, patient name, age, gender, phone number, and address. Each patient is uniquely identified using patient_id. This table ensures accurate registration and tracking of patient details.
- **Registration Table**
  The Registration table records which patient has registered for which health camp. It contains registration ID, patient ID, camp ID, and registration date. This table creates a relationship between Patient and Health_Camp using foreign keys. It ensures that each registration is properly linked to both the patient and the specific camp.
- **Diagnosis Table**
  The Diagnosis table stores medical examination details of patients. It includes diagnosis ID, patient ID, doctor ID, camp ID, disease details, medicines prescribed, and diagnosis date. This table connects patients, doctors, and health camps together. It helps in maintaining proper treatment records and generating medical reports.

7.2 Table Structure

Below are the detailed structures of each table including attributes and data types.

1. Health_Camp Table

Attribute	Data Type
camp_id	INT (PK) AUTO_INCREMENT
camp_name	VARCHAR(100)
location	VARCHAR(100)
camp_date	DATE
organizer	VARCHAR(100)

2. Doctor Table

Attribute	Data Type
doctor_id	INT (PK) AUTO_INCREMENT
doctor_name	VARCHAR(100)
specialization	VARCHAR(100)
phone	VARCHAR(15)

3. Patient Table

Attribute	Data Type
patient_id	INT (PK) AUTO_INCREMENT
patient_name	VARCHAR(100)
age	INT
gender	VARCHAR(10)
phone	VARCHAR(15)
address	VARCHAR(200)

4. Registration Table

Attribute	Data Type
registration_id	INT (PK) AUTO_INCREMENT
patient_id	INT (FK)
camp_id	INT (FK)
registration_date	DATE

5. Diagnosis Table

Attribute	Data Type
diagnosis_id	INT (PK) AUTO_INCREMENT
patient_id	INT (FK)
doctor_id	INT (FK)
camp_id	INT (FK)
disease_details	VARCHAR(200)
medicines_prescribed	VARCHAR(200)
diagnosis_date	DATE

7.3 Constraints Used

To maintain accuracy, consistency, and proper relationships between tables, the following constraints are used:

PRIMARY KEY

A Primary Key uniquely identifies each record in a table. It does not allow duplicate or NULL values. For example, patient_id in the Patient table and doctor_id in the Doctor table ensure that each record is unique. This helps in identifying each entity without confusion.

FOREIGN KEY

A Foreign Key establishes a relationship between two tables. It links a column in one table to the primary key of another table. For example, patient_id in the Registration table references the Patient table, and camp_id references the Health_Camp table. This ensures referential integrity and prevents invalid data entry.

NOT NULL

The NOT NULL constraint ensures that certain fields cannot be left empty. Important attributes such as patient_name, camp_name, organizer, and diagnosis_date must contain valid values. This prevents incomplete records from being stored in the database.

UNIQUE

The UNIQUE constraint ensures that duplicate values are not allowed in specific columns. For example, phone numbers in the Patient and Doctor tables are unique to avoid duplication of contact details. This improves data reliability and prevents redundancy.

CHAPTER 8 – UML DIAGRAM

Figure 8.1 ER Diagram

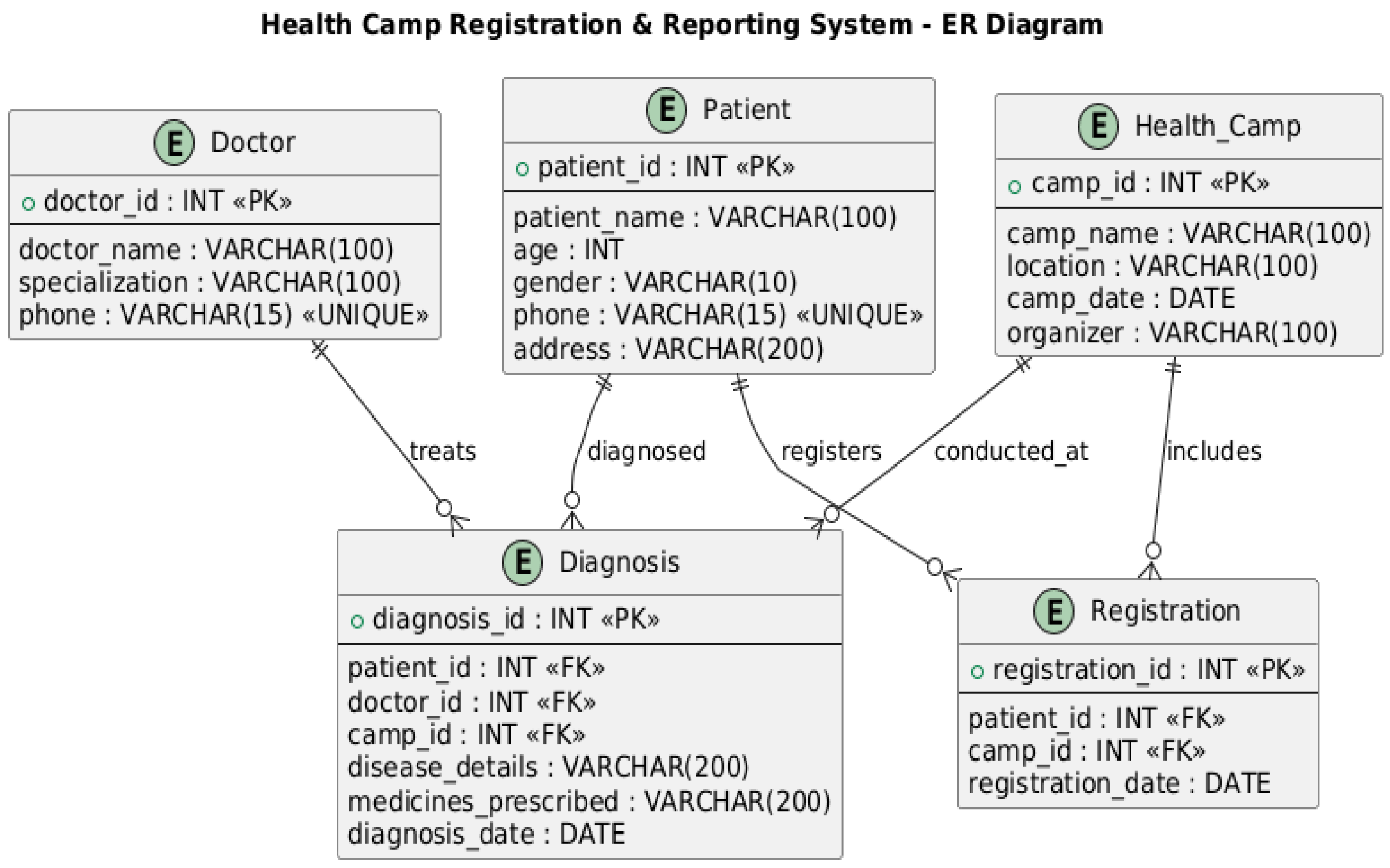


Figure 8.2 Use Case Diagram

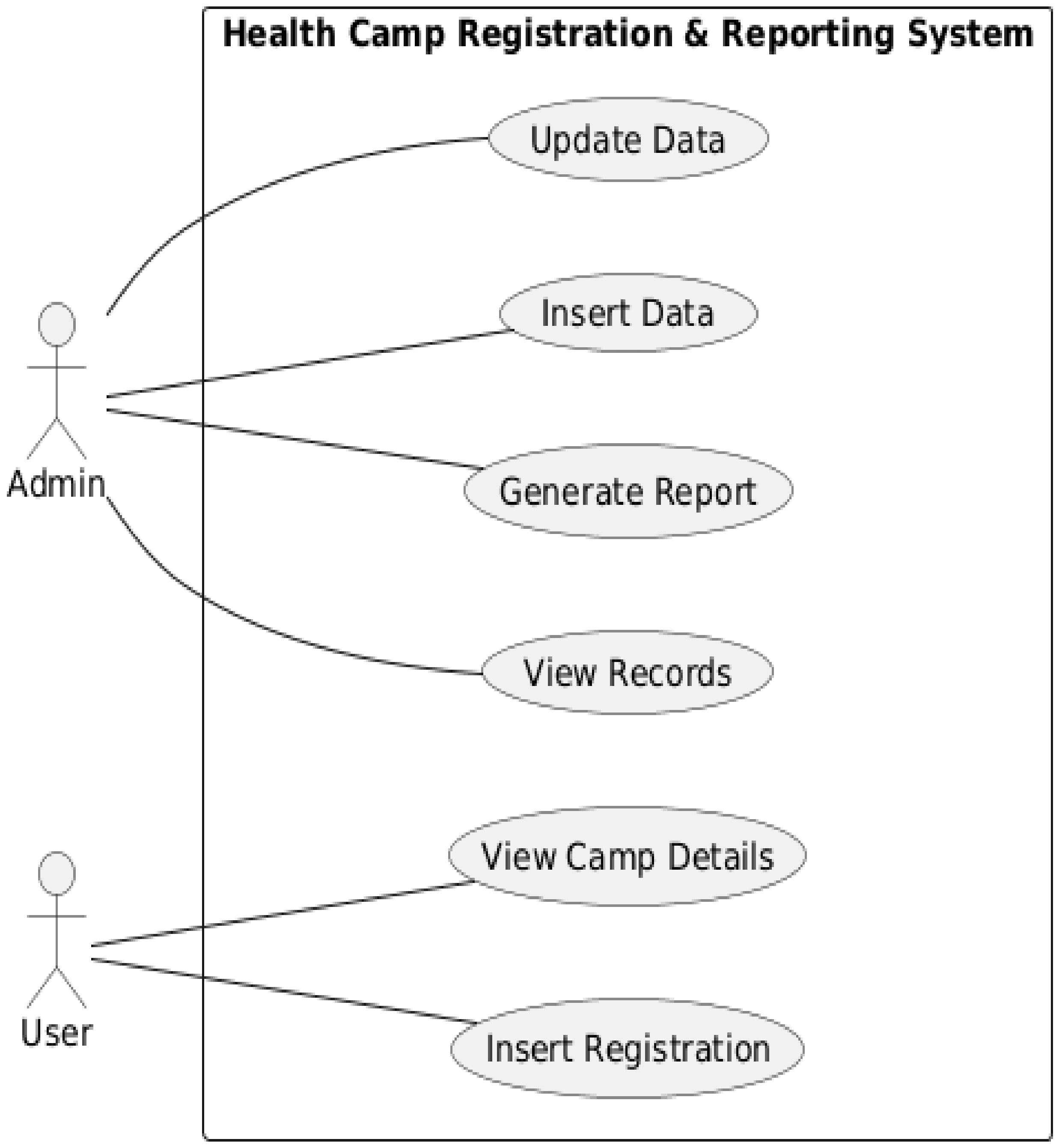


Figure 8.3 Activity Diagram

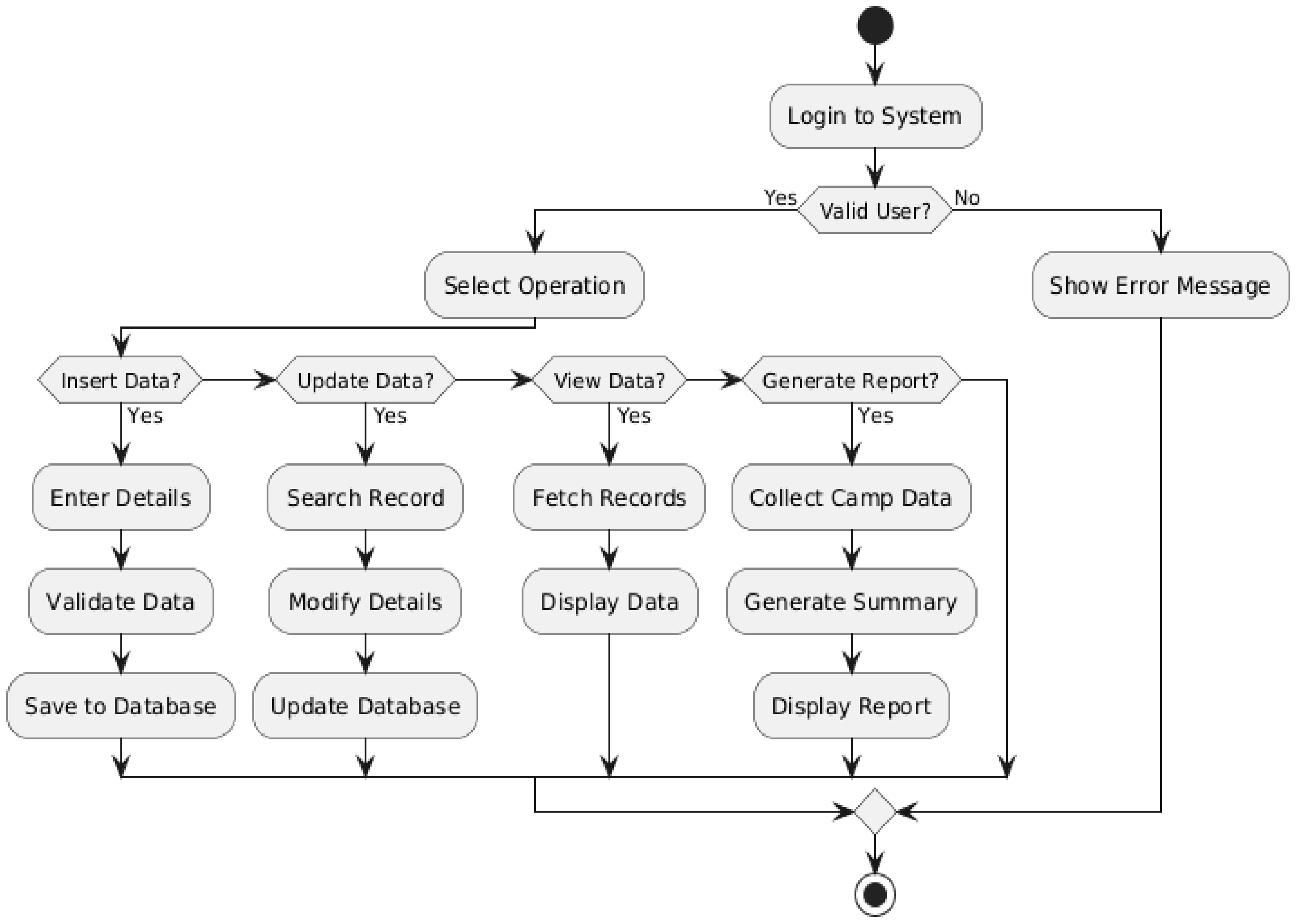
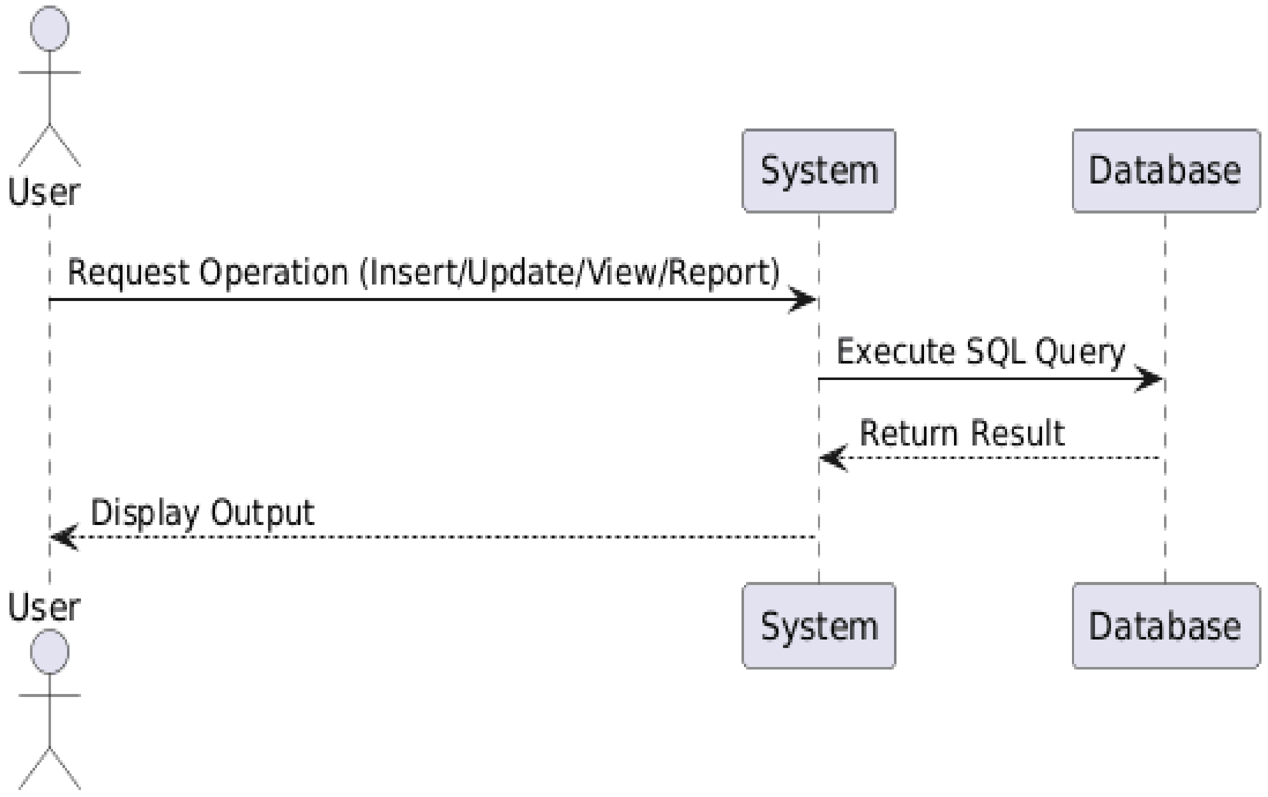


Figure 8.4 Sequence Diagram



CHAPTER 9 – SQL IMPLEMENTATION

9.1 Database Creation

CREATE DATABASE

```
CREATE DATABASE HealthCampDB;
USE HealthCampDB;
```

9.2 Table Creation

CREATE TABLE Queries

1. Health_Camp Table

```
CREATE TABLE Health_Camp (
    camp_id INT PRIMARY KEY AUTO_INCREMENT,
    camp_name VARCHAR(100) NOT NULL,
    location VARCHAR(100) NOT NULL,
    camp_date DATE NOT NULL,
    organizer VARCHAR(100) NOT NULL
);
```

2. Doctor Table

```
CREATE TABLE Doctor (
    doctor_id INT PRIMARY KEY AUTO_INCREMENT,
    doctor_name VARCHAR(100) NOT NULL,
    specialization VARCHAR(100) NOT NULL,
    phone VARCHAR(15) UNIQUE NOT NULL
);
```

3. Patient Table

```
CREATE TABLE Patient (
    patient_id INT PRIMARY KEY AUTO_INCREMENT,
    patient_name VARCHAR(100) NOT NULL,
    age INT NOT NULL,
    gender VARCHAR(10) NOT NULL,
    phone VARCHAR(15) UNIQUE NOT NULL,
    address VARCHAR(200) NOT NULL
);
```

4. Registration Table

```
CREATE TABLE Registration (
    registration_id INT PRIMARY KEY AUTO_INCREMENT,
    patient_id INT NOT NULL,
    camp_id INT NOT NULL,
    registration_date DATE NOT NULL,

    FOREIGN KEY (patient_id) REFERENCES Patient(patient_id),
    FOREIGN KEY (camp_id) REFERENCES Health_Camp(camp_id)
);
```

5. Diagnosis Table

```
CREATE TABLE Diagnosis (
    diagnosis_id INT PRIMARY KEY AUTO_INCREMENT,
    patient_id INT NOT NULL,
    doctor_id INT NOT NULL,
    camp_id INT NOT NULL,
    disease_details VARCHAR(200) NOT NULL,
    medicines_prescribed VARCHAR(200) NOT NULL,
    diagnosis_date DATE NOT NULL,

    FOREIGN KEY (patient_id) REFERENCES Patient(patient_id),
    FOREIGN KEY (doctor_id) REFERENCES Doctor(doctor_id),
    FOREIGN KEY (camp_id) REFERENCES Health_Camp(camp_id)
);
```

9.3 Data Insertion

Insert into Health_Camp

```
INSERT INTO Health_Camp (camp_name, location, camp_date, organizer)
VALUES
('General Health Camp', 'Solapur', '2026-03-10', 'City Hospital'),
('Eye Checkup Camp', 'Pune', '2026-03-15', 'Vision Foundation'),
('Dental Camp', 'Mumbai', '2026-03-20', 'Smile Care NGO');
```

Insert into Doctor

```
INSERT INTO Doctor (doctor_name, specialization, phone)
VALUES
('Dr. Sharma', 'General Physician', '9876543210'),
('Dr. Mehta', 'Ophthalmologist', '9876543211'),
('Dr. Rao', 'Dentist', '9876543212');
```

Insert into Patient

```
INSERT INTO Patient (patient_name, age, gender, phone, address)
VALUES
('Ravi Kumar', 25, 'Male', '9000000001', 'Solapur'),
('Anita Patil', 30, 'Female', '9000000002', 'Pune'),
('Suresh Deshmukh', 40, 'Male', '9000000003', 'Mumbai');
```

Insert into Registration

```
INSERT INTO Registration (patient_id, camp_id, registration_date)
VALUES
(1, 1, '2026-03-10'),
(2, 2, '2026-03-15'),
(3, 3, '2026-03-20');
```

Insert into Diagnosis

```
INSERT INTO Diagnosis (patient_id, doctor_id, camp_id, disease_details,
medicines_prescribed, diagnosis_date)
VALUES
(1, 1, 1, 'Fever and Cold', 'Paracetamol', '2026-03-10'),
(2, 2, 2, 'Weak Eyesight', 'Eye Drops', '2026-03-15'),
(3, 3, 3, 'Tooth Infection', 'Painkillers', '2026-03-20');
```

9.4 Data Retrieval

SELECT

```
SELECT * FROM Patient;
```

WHERE

```
SELECT * FROM Patient
WHERE age > 30;
```

JOIN (Full Detailed Join)

```
SELECT
   p.patient_name,
   d.doctor_name,
   h.camp_name,
   dg.disease_details,
   dg.medicines_prescribed
FROM Diagnosis dg
JOIN Patient p ON dg.patient_id = p.patient_id
JOIN Doctor d ON dg.doctor_id = d.doctor_id
JOIN Health_Camp h ON dg.camp_id = h.camp_id;
```

9.5 Advanced Queries

GROUP BY

```
SELECT h.camp_name, COUNT(r.patient_id) AS total_patients
FROM Health_Camp h
JOIN Registration r ON h.camp_id = r.camp_id
GROUP BY h.camp_name;
```

HAVING

```
SELECT h.camp_name, COUNT(r.patient_id) AS total_patients
FROM Health_Camp h
JOIN Registration r ON h.camp_id = r.camp_id
GROUP BY h.camp_name
HAVING COUNT(r.patient_id) > 1;
```

Subquery

```
SELECT patient_name, age
FROM Patient
WHERE age > (SELECT AVG(age) FROM Patient);
```

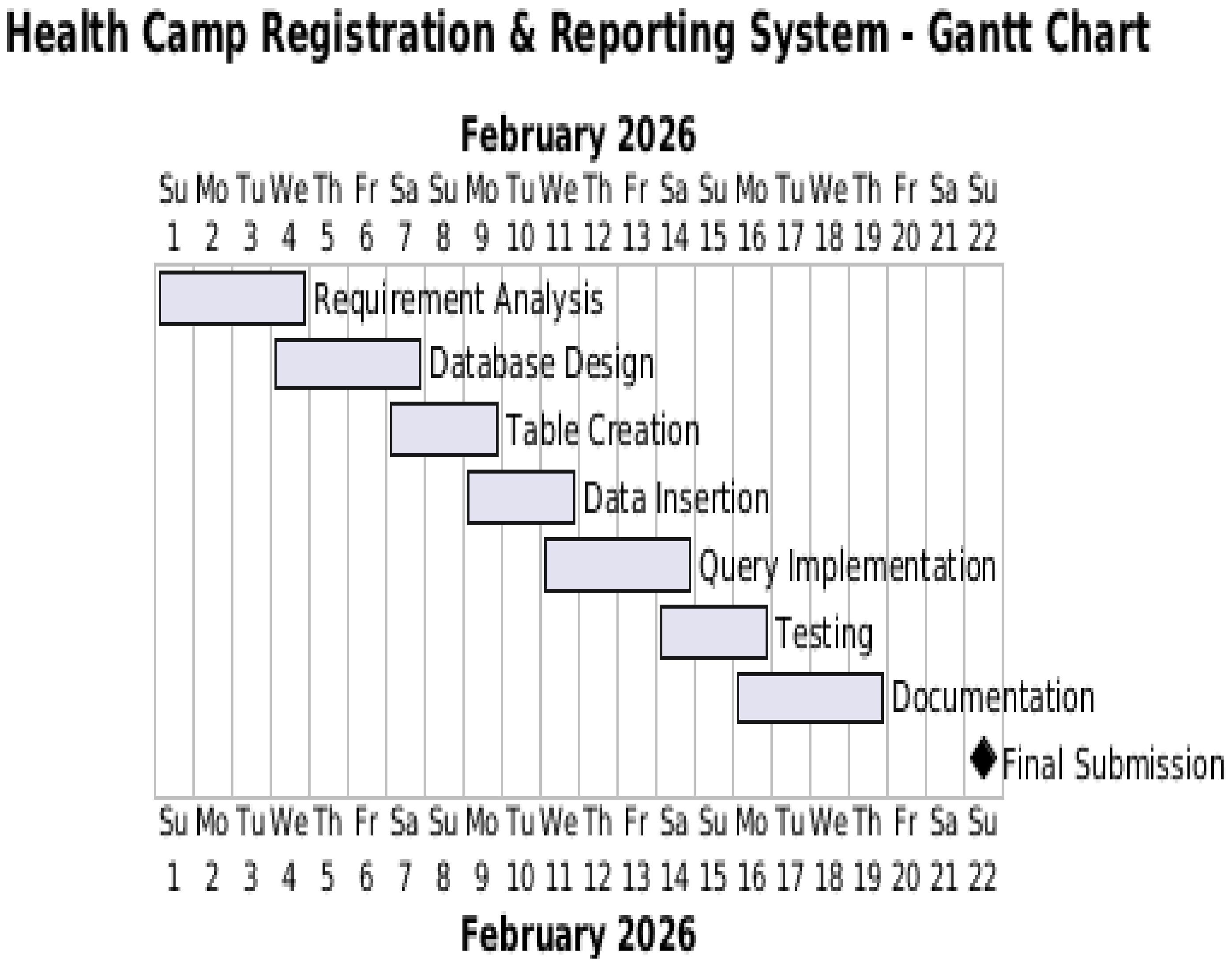
View

```
CREATE VIEW Diagnosis_Report AS
SELECT
    p.patient_name,
    d.doctor_name,
    h.camp_name,
    dg.disease_details,
    dg.medicines_prescribed,
    dg.diagnosis_date
FROM Diagnosis dg
JOIN Patient p ON dg.patient_id = p.patient_id
JOIN Doctor d ON dg.doctor_id = d.doctor_id
JOIN Health_Camp h ON dg.camp_id = h.camp_id;
```

Display View

```
SELECT * FROM Diagnosis_Report;
```

GANTT CHART :


Health Camp Registration & Reporting System - Gantt Chart
February 2026
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
Requirement Analysis
Database Design
Table Creation
Data Insertion
Query Implementation
Testing
Documentation
Final Submission
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
February 2026

CHAPTER 10 - SYSTEM TESTING AND RESULT

10.1 Query Correctness Testing

Query correctness testing ensures that all SQL commands are syntactically correct and perform the intended operations without errors.
The following tests were performed:
1. Database and Table Creation
The CREATE DATABASE and CREATE TABLE commands were executed successfully.
To verify table creation, the following command was used:

```
SHOW TABLES;
```

Sample Output:

```
Health_Camp
Doctor
Patient
Registration
Diagnosis
```

This confirms that all required tables were created successfully.

10.2 Data Validation

Data validation ensures that incorrect or invalid data cannot be inserted into the database.
The following validations were tested:

- **PRIMARY KEY Constraint**
  Attempted to manually insert a duplicate primary key value.
  Result: MySQL displayed a duplicate entry error.
  This confirms that primary keys prevent duplicate records.

- **FOREIGN KEY Constraint**
  Attempted to insert a record in the Registration table with a non-existing camp_id.
  Result: MySQL displayed a foreign key constraint error.
  This confirms referential integrity is maintained.

- **NOT NULL Constraint**
  Attempted to insert a Patient record without patient_name.
  Result: MySQL displayed an error indicating that the field cannot be NULL.
  This confirms that mandatory fields cannot be left empty.

- **UNIQUE Constraint**
  Attempted to insert duplicate phone number in the Patient table.
  Result: MySQL displayed duplicate entry error.
  This confirms that phone numbers must be unique.

10.3 Output Verification

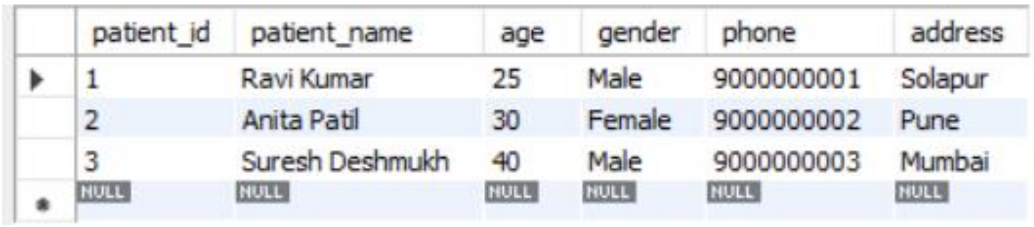
Output verification ensures that query results match the expected data and calculations.

GROUP BY Testing

SELECT h.camp_name, COUNT(r.patient_id) AS total_patients
FROM Health_Camp h
JOIN Registration r ON h.camp_id = r.camp_id
GROUP BY h.camp_name;

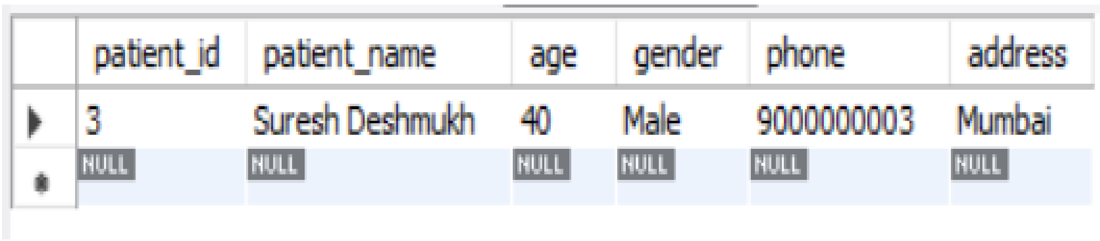
10.4 OUTPUT :

PATIENT 1:



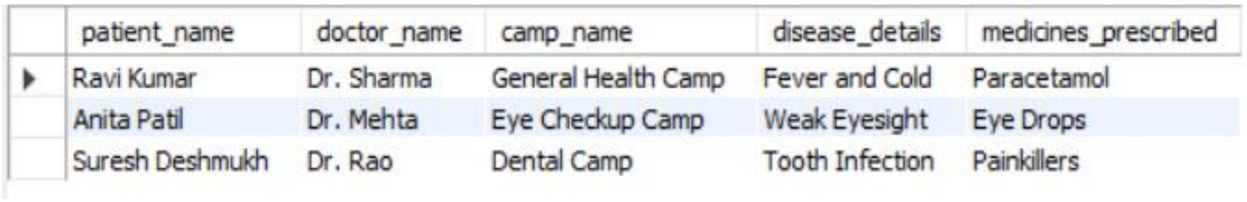
patient_id	patient_name	age	gender	phone	address
1	Ravi Kumar	25	Male	9000000001	Solapur
2	Anita Patil	30	Female	9000000002	Pune
3	Suresh Deshmukh	40	Male	9000000003	Mumbai
NULL	NULL	NULL	NULL	NULL	NULL

PATIENT 2:



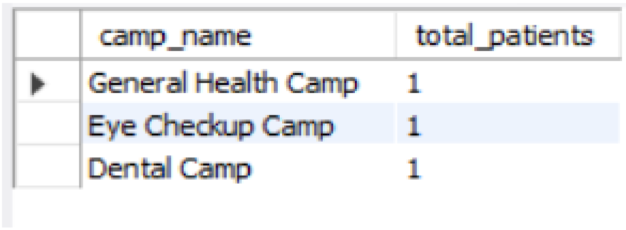
patient_id	patient_name	age	gender	phone	address
3	Suresh Deshmukh	40	Male	9000000003	Mumbai
NULL	NULL	NULL	NULL	NULL	NULL

RESULT 3:



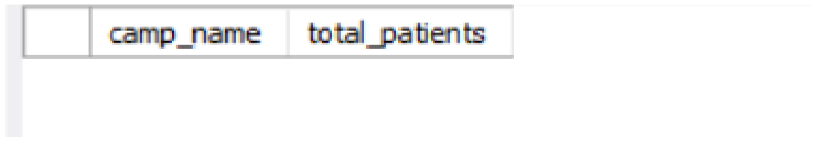
patient_name	doctor_name	camp_name	disease_details	medicines_prescribed
Ravi Kumar	Dr. Sharma	General Health Camp	Fever and Cold	Paracetamol
Anita Patil	Dr. Mehta	Eye Checkup Camp	Weak Eyesight	Eye Drops
Suresh Deshmukh	Dr. Rao	Dental Camp	Tooth Infection	Painkillers

RESULT 4:



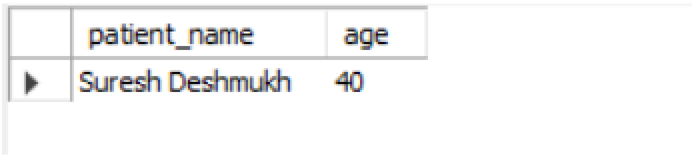
camp_name	total_patients
General Health Camp	1
Eye Checkup Camp	1
Dental Camp	1

RESULT 5:



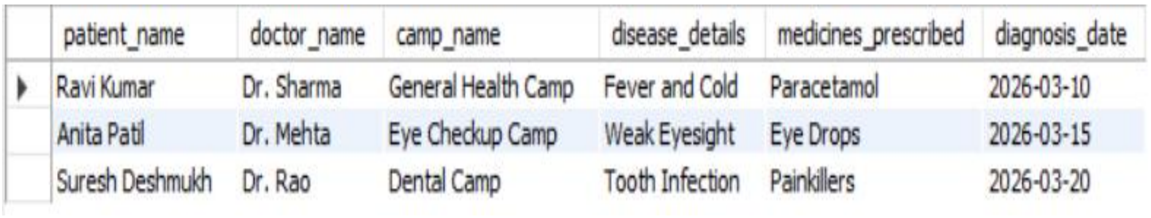
camp_name	total_patients

RESULT 6:



patient_name	age
Suresh Deshmukh	40

DIGNOSIS REPORT 7:



patient_name	doctor_name	camp_name	disease_details	medicines_prescribed	diagnosis_date
Ravi Kumar	Dr. Sharma	General Health Camp	Fever and Cold	Paracetamol	2026-03-10
Anita Patil	Dr. Mehta	Eye Checkup Camp	Weak Eyesight	Eye Drops	2026-03-15
Suresh Deshmukh	Dr. Rao	Dental Camp	Tooth Infection	Painkillers	2026-03-20

CHAPTER 11 -SECURITY,BACKUP,AND RECOVERY

11.1 Security

The security, backup, and recovery features are crucial for protecting patient and camp data from unauthorized access, data loss, and system failure. These features ensure that the Health Camp Registration and Reporting System remains secure and can be restored when needed.

User Privileges

User privileges specify what actions a user can take on the database, like SELECT, INSERT, UPDATE, and DELETE. The database administrator can grant or revoke these privileges to control access.

Example:

```
GRANT SELECT, INSERT ON health_camp_db.* TO 'staff'@'localhost';
```

This command allows the user to view and insert data but not to delete or change existing records.

Access Control

Access control keeps unauthorized users from getting into the database. Only authenticated users with a valid username and password can log in and perform actions.

In MySQL Workbench, you need login credentials to access the database. This ensures that sensitive data like patient details, doctor information, and camp records stay secure.

11.2 Backup

Backup is the process of making a copy of the database to avoid data loss in case of system failure, hardware crash, or accidental deletion.

MySQL offers a tool called mysqldump to create database backups.

Example command:

```
mysqldump -u root -p health_camp_db > health_camp_backup.sql
```

This command creates a backup file named health_camp_backup.sql that contains all tables and data of the system.

Backups help in restoring data whenever necessary.

11.3 Recovery

Recovery involves restoring the database from a backup file after data loss or system failure.

You can restore the backup file using this command:

```
mysql -u root -p health_camp_db < health_camp_backup.sql
```

This command brings back all tables and records from the backup file into the database. Recovery ensures that important health camp data is not lost permanently and can be easily recovered, maintaining system reliability and continuity.

CHAPTER 12 – FUTURE SCOPE AND CONCLUSION

12.1 Future Scope

The Health Camp Registration and Reporting System can be improved by adding new features and technologies to increase efficiency and usability.

Web Integration

The system can be turned into a web application using technologies like PHP, Java, or Python. This change would enable online registration for health camps and remote access to reports. Patients and staff could use a web browser to access the system.

Advanced Reporting

More detailed and dynamic reports can be generated, such as:

- Monthly camp performance reports
- Disease-wise patient statistics
- Doctor-wise consultation reports

Reports can also be exported in PDF or Excel format for official use.

Analytics Features

Data analytics can be added to:

- Examine patient trends
- Identify frequently occurring diseases
- Anticipate future camp needs

Graphical dashboards and charts can help with decision-making and planning.

12.2 Conclusion

The Health Camp Registration and Reporting System was successfully designed and implemented using MySQL. The project achieved the following:

- Designed a structured relational database
- Created tables with proper relationships

- Implemented SQL queries for data insertion and retrieval
- Applied constraints to ensure data integrity
- Generated reports using advanced queries

Through this project, practical knowledge of MySQL was gained, including database creation, table design, constraints, joins, and advanced queries.

This project shows the value of MySQL as a reliable, open-source database management system that ensures data security, consistency, and efficient data handling.

Overall, the system improves accuracy, reduces manual work, and provides an organized way to manage health camp activities.

CHAPTER 13 – REFERENCES

The following references were used during the development of the Health Camp Registration and Reporting System:

- **MySQL Official Documentation**
  Oracle Corporation. *MySQL 8.0 Reference Manual.* Available at:
  https://dev.mysql.com/doc/

- **Prescribed Textbooks**
  Korth, H. F., Silberschatz, A., & Sudarshan, S. *Database System Concepts.* McGraw-Hill Education.

- **Online Learning Sources**
  - W3Schools – SQL and MySQL Tutorials (https://www.w3schools.com/)
  - GeeksforGeeks – Database Management System Tutorials (https://www.geeksforgeeks.org/)
  - TutorialsPoint – MySQL Tutorial (https://www.tutorialspoint.com/mysql/)

These sources helped in understanding database concepts, SQL queries, constraints, and MySQL implementation techniques used in this project.

CHAPTER 14 – GLOSSARY

The following terms are commonly used in this project:

- **DBMS (Database Management System)**
  A software system used to create, manage, and maintain databases. It allows users to store, retrieve, update, and organize data efficiently while ensuring data security and integrity.

- **SQL (Structured Query Language)**
  A standard programming language used to communicate with relational databases. It is used to create tables, insert records, update data, delete records, and retrieve information using queries.

- **Primary Key**
  A field in a table that uniquely identifies each record. It does not allow duplicate or NULL values. For example, patient_id in the Patient table uniquely identifies each patient.

- **Foreign Key**
  A field in one table that refers to the primary key of another table. It establishes a relationship between two tables and maintains referential integrity. For example, patient_id in the Registration table links to the Patient table.

- **MySQL**
  An open-source relational database management system used to store and manage data using SQL. It supports tables, relationships, constraints, indexing, and advanced queries.

- **Table**
  A structured format in a database that stores data in rows and columns. Each table represents a specific entity such as Patient, Doctor, or Health_Camp.

- **Constraint**
  A rule applied to a column in a database table to maintain accuracy and reliability of data. Examples include PRIMARY KEY, FOREIGN KEY, NOT NULL, and UNIQUE.

- **JOIN**
  An SQL operation used to combine data from two or more tables based on a related column. It helps generate combined reports from multiple tables.

- **View**
  A virtual table created using a SELECT query. It does not store data physically but displays data from one or more tables. In this project, Diagnosis_Report is a view.

- **Backup**
  A copy of the database stored separately to prevent data loss due to system failure or accidental deletion.

- **Recovery**
  The process of restoring database data from a backup file after data loss or system failure.