

**Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and
Technology**

(Deemed to be University Estd. u/s 3 of UGC Act, 1956)

School of Computing

B.Tech. – Computer Science and Engineering



VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SDG 4: Quality Education

Course Code : 10211CS207

Course Name : Database Management Systems

Slot No :

DBMS PROJECT REPORT

Title: Hospital Patient Record System (HPRS)

Submitted by:

VTUNO	REGISTER NUMBER	STUDENT NAME
VTU28629	24UECS0828	POLINA KIRAN TEJA

Under the guidance of:

Dr V Senthil Kumar

Type your text

INDEX	PAGE
-------	------

1. Introduction.....	3
2. Problem Statement.....	4
3. Objectives.....	5
4. System Requirements	6
5. System Analysis and Design.....	7
6. ER Diagram (Conceptual Design)	8
7. Schema Design (Oracle).....	9
8.Normalization.....	11
9. Implementation (SQL Queries)	12
10. Input and Output.....	13
11. Integration with MongoDB (NoSQL).....	14
12. Results and Discussion.....	15
13. Conclusion	16

1. Introduction

In the modern healthcare environment, hospitals generate and manage vast amounts of patient-related data every day. Traditionally, most of this information—such as patient details, medical history, and treatment records—was stored manually, leading to inefficiencies, data redundancy, and difficulties in retrieving vital information quickly. As healthcare organizations grow, the need for a reliable, efficient, and secure digital database management system becomes increasingly essential.

The Hospital Patient Record System (HPRS) is designed to manage hospital data efficiently by organizing patient details, diagnoses, and treatment records in a structured and centralized manner. This system enhances patient care by providing instant access to patient information, tracking medical treatments, and ensuring accurate record-keeping. It enables healthcare providers to make informed decisions, reduces manual workload, and ensures consistency in patient records.

The system also emphasizes data integrity and security. Every patient and treatment record is stored with appropriate relational constraints, ensuring data accuracy and preventing duplication. The system enforces logical rules—such as ensuring valid age ranges and positive treatment fees—to maintain reliable data quality. Additionally, integration with MongoDB (NoSQL) supports the storage of unstructured medical data such as digital prescriptions, diagnostic images, and doctor's notes, making HPRS a hybrid solution suitable for real-world hospital management.

2. Problem Statement

Hospitals handle large volumes of patient data daily—ranging from personal details to medical histories, diagnoses, and treatment records. Managing such information manually or through disconnected systems can result in several challenges such as data redundancy, lack of integration, data security concerns, and limited analytics. The Hospital Patient Record System aims to overcome these challenges by implementing a centralized database that maintains accurate, secure, and easily retrievable patient and treatment records. The system ensures referential integrity, enforces constraints, and supports both structured (SQL) and unstructured (NoSQL) data management for comprehensive hospital information processing.

3. Objectives

To design a relational database for storing patient and treatment data.

- To ensure data integrity using primary key, foreign key, and check constraints.
- To facilitate secure storage and retrieval of patient medical records.
- To integrate MongoDB for handling unstructured hospital data such as digital prescriptions, images, and notes.
- To demonstrate CRUD operations (Create, Read, Update, Delete) and relational joins in Oracle.
- To enhance hospital efficiency by enabling fast data access and report generation.

4. System Requirements

Hardware Requirements:

- Processor: Intel i5 or higher
- RAM: 8 GB or above
- Hard Disk: 250 GB minimum

Software Requirements:

- Operating System: Windows 10 / Linux
- Database: Oracle 12c or higher
- NoSQL: MongoDB 6.0
- Tools: Oracle SQL Developer, MongoDB Compass
- Front-End (Optional): Java / Python / Web Interface

5. System Analysis and Design

The Hospital Patient Record System identifies two major entities: Patient and Treatment.

Relationships:

- One patient can have multiple treatments.
- Each treatment belongs to exactly one patient.

Constraints:

- Fees > 0
- Age BETWEEN 1 AND 120

6. ER Diagram (Conceptual Design)

Entities:

Patient (Patient_ID, Name, Age, Gender, Contact, Address)

Treatment (Treat_ID, Patient_ID, Doctor_Name, Diagnosis, Date, Fees)

Relationship: Patient \leftrightarrow Treatment \rightarrow One-to-Many

Foreign Key: Patient_ID in Treatment references

Patient(Patient_ID).

7. Schema Design (Oracle)

```
CREATE TABLE Patient (
    Patient_ID NUMBER PRIMARY KEY,
    Name VARCHAR2(50),
    Age NUMBER CHECK (Age BETWEEN 1 AND 120),
    Gender VARCHAR2(10),
    Contact VARCHAR2(15),
    Address VARCHAR2(100)
);
```

```
SQL> DESC Patient;
      Name          Null?       Type
-----  -----  -----
PATIENT_ID      NOT NULL  NUMBER
NAME            VARCHAR2(50)
AGE             NUMBER
GENDER          VARCHAR2(10)
CONTACT         VARCHAR2(15)
ADDRESS          VARCHAR2(100)
```

```
CREATE TABLE Treatment (
    Treat_ID NUMBER PRIMARY KEY,
    Patient_ID NUMBER REFERENCES Patient(Patient_ID),
```

```
Doctor_Name VARCHAR2(50),  
Diagnosis VARCHAR2(100),  
Date DATE,  
Fees NUMBER CHECK (Fees > 0)  
);
```

```
SQL> DESC Treatment;  
Name          Null?    Type  
-----  -----  -----  
TREAT_ID      NOT NULL NUMBER  
PATIENT_ID           NUMBER  
DOCTOR_NAME        VARCHAR2(50)  
DIAGNOSIS         VARCHAR2(100)  
DATE             DATE  
FEES             NUMBER
```

8. Normalization

The database follows normalization up to Third Normal Form (3NF):

1NF: All fields contain atomic values (no repeating groups).

2NF: Non-key attributes depend entirely on the primary key.

3NF: No transitive dependency exists.

This ensures minimal redundancy, consistent data, and efficient updates.

9. Implementation (SQL Queries)

```
INSERT INTO Patient VALUES (1, 'Ravi Kumar', 45, 'Male',  
'9876543210', 'Chennai');
```

```
INSERT INTO Patient VALUES (2, 'Anita Sharma', 32, 'Female',  
'9823456789', 'Bangalore');
```

```
INSERT INTO Treatment VALUES (101, 1, 'Dr. Mehta', 'Diabetes  
Checkup', TO_DATE('2025-10-10', 'YYYY-MM-DD'), 1200);
```

```
INSERT INTO Treatment VALUES (102, 1, 'Dr. Mehta', 'Blood  
Test', TO_DATE('2025-10-12', 'YYYY-MM-DD'), 800);
```

```
INSERT INTO Treatment VALUES (103, 2, 'Dr. Nair', 'Fever  
Consultation', TO_DATE('2025-10-15', 'YYYY-MM-DD'), 600);
```

```
SQL > INSERT INTO Patient VALUES (1, 'Ravi Kumar', 45, 'Male', '9876543210',  
'Chennai');  
1 row(s) inserted.
```

```
SQL > INSERT INTO Patient VALUES (2, 'Anita Sharma', 32, 'Female', '9823456789',  
'Bangalore');  
1 row(s) inserted.
```

```
SQL > INSERT INTO Treatment VALUES (101, 1, 'Dr. Mehta', 'Diabetes Checkup',  
TO_DATE('2025-10-10', 'YYYY-MM-DD'), 1200);  
1 row(s) inserted.
```

```
SQL > INSERT INTO Treatment VALUES (102, 1, 'Dr. Mehta', 'Blood Test',  
TO_DATE('2025-10-12', 'YYYY-MM-DD'), 800);  
1 row(s) inserted.
```

```
SQL > INSERT INTO Treatment VALUES (103, 2, 'Dr. Nair', 'Fever Consultation',  
TO_DATE('2025-10-15', 'YYYY-MM-DD'), 600);  
1 row(s) inserted.
```

10. Input and Output

Sample Input:

```
INSERT INTO Patient VALUES (3, 'Manoj Das', 27, 'Male',  
'9004567812', 'Delhi');
```

```
INSERT INTO Treatment VALUES (104, 3, 'Dr. Patel', 'Skin Allergy',  
TO_DATE('2025-10-18','YYYY-MM-DD'), 950);
```

Sample Output:

Name	Age	Doctor_Name	Diagnosis	Fees
Ravi Kumar	45	Dr. Mehta	Diabetes Checkup	1200
Ravi Kumar	45	Dr. Mehta	Blood Test	800
Anita Sharma	32	Dr. Nair	Fever Consultation	600
Manoj Das	27	Dr. Patel	Skin Allergy	950

11. Integration with MongoDB (NoSQL)

Database Name: HospitalDB

patients Collection Example:

```
{  
    "_id": 1,  
    "patient_id": "P001",  
    "name": "Ravi Kumar",  
    "age": 45,  
    "gender": "Male",  
    "contact": "9876543210",  
    "address": "Chennai",  
    "medical_history": ["Diabetes", "Hypertension"] }
```

Treatments Collection Example:

```
{  
    "_id": 101,  
    "treat_id": "T101",  
    "patient_id": "P001",  
    "doctor_name": "Dr. Mehta",  
    "diagnosis": "Diabetes Checkup",  
    "date": "2025-10-10",  
    "fees": 1200  
}
```

MongoDB Queries:

```
db.patients.insertOne({...});  
  
db.treatments.find({}, {doctor_name:1, diagnosis:1, _id:0});  
  
db.patients.updateOne({patient_id:'P001'}, {$push:{medical_history:'Cholesterol'}});  
  
db.treatments.find({fees:{$gt:1000}}, {patient_id:1, doctor_name:1, fees:1, _id:0});
```

```
> db.patients.insertOne({
...   "_id": 1,
...   "patient_id": "P001",
...   "name": "Ravi Kumar",
...   "age": 45,
...   "gender": "Male",
...   "contact": "9876543210",
...   "address": "Chennai",
...   "medical_history": ["Diabetes", "Hypertension"]
... });
{
  "acknowledged": true,
  "insertedId": 1
}
```

```
> db.treatments.find({}, {doctor_name:1, diagnosis:1, _id:0});
{ "doctor_name" : "Dr. Mehta", "diagnosis" : "Diabetes Checkup" }
{ "doctor_name" : "Dr. Mehta", "diagnosis" : "Blood Test" }
{ "doctor_name" : "Dr. Nair", "diagnosis" : "Fever Consultation" }
{ "doctor_name" : "Dr. Patel", "diagnosis" : "Skin Allergy" }
```

```
> db.patients.updateOne(
...   {patient_id:'P001'},
...   {$push:{medical_history:'Cholesterol'}}
... );
{
  "acknowledged" : true,
  "matchedCount" : 1,
  "modifiedCount" : 1
}

> db.treatments.find(
...   {fees:{$gt:1000}},
...   {patient_id:1, doctor_name:1, fees:1, _id:0}
... );
{ "patient_id" : "P001", "doctor_name" : "Dr. Mehta", "fees" : 1200 }
```

12. Results and Discussion

The Hospital Patient Record System successfully manages patient and treatment data in a structured and secure manner. The relational design enforces data integrity, and MongoDB integration enhances flexibility by storing unstructured data such as digital prescriptions and notes. The system improves efficiency, reduces redundancy, and provides reliable access to hospital information.

13. Conclusion

The Hospital Patient Record System (HPRS) provides an efficient, secure, and scalable digital framework for hospital management. It ensures reliable data storage, enforces relational constraints, and integrates NoSQL for handling complex healthcare data. Overall, it enhances hospital operations and supports quality patient care.

14. References

- 1. Oracle Database Documentation – Oracle Corporation**
- 2. MongoDB Official Documentation**
- 3. Silberschatz, Korth, Sudarshan – Database System Concepts**
- 4. Raghu Ramakrishnan – Database Management Systems**
- 5. Elmasri & Navathe – Fundamentals of Database Systems**