

Mini Project Report

of

Database Systems Lab (CSE 2262)

TITLE

EMERGENCY WARD MANAGEMENT SYSTEM

SUBMITTED

BY

STUDENT NAME REG. NO ROLL NO SECTION

1. NAMRATA RAJ 220905550 61 C2

1. PRAKHAR ARYA 220905458 49 C2

Department of Computer Science and Engineering

**Manipal Institute of Technology, Manipal.**

**April 2024**



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manipal

17/04/2024

CERTIFICATE

This is to certify that the project titled Emergency ward Management System is a record of the bonafide work done by Namrata Raj (Reg. No. 220905550) & Prakhar Arya (Reg. No. 220905458) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in COMPUTER SCIENCE & ENGINEERING of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2023-2024.

Name and Signature of Examiners:

TABLE OF CONTENTS

ABSTRACT

**CHAPTER 1: INTRODUCTION**

**CHAPTER 2: PROBLEM STATEMENT & OBJECTIVES**

**CHAPTER 3: METHODOLOGY**

**CHAPTER 4: RESULTS & SNAPSHOTS**

**CHAPTER 5: CONCLUSION**

**CHATER 6: LIMITATIONS & FUTURE WORK**

**CHAPTER 7: REFERENCES**

ABSTRACT

This system is designed to deal with day-to-day operations and management of hospital emergency ward activities to for efficient hospital management, focusing on patient admissions and surgeries. Noteworthy one-to-many relationships include patients with staff, reflecting personalized care.

The database streamlines processes, optimizing coordination among doctors, nurses, and patients for an effective emergency ward experience.

**CHAPTER 1: INTRODUCTION**

The management of emergency wards is a critical aspect of healthcare delivery, as it involves the timely and efficient treatment of patients in need of urgent medical attention. To streamline this process and ensure the best possible care for patients, emergency ward management systems have been developed to support healthcare professionals in their decision-making and treatment processes.

These systems apply the principles of database management to organize and store patient information, such as medical history, vital signs, and test results, in a secure and accessible manner. By utilizing database management system rules, emergency ward management systems can effectively track and update patient records in real-time, enabling healthcare providers to make informed decisions quickly and accurately.

Furthermore, database management systems help emergency wards optimize resources, track inventory levels, and manage scheduling of staff and equipment. This ensures that the emergency ward operates efficiently and effectively, ultimately leading to improved patient outcomes and satisfaction.

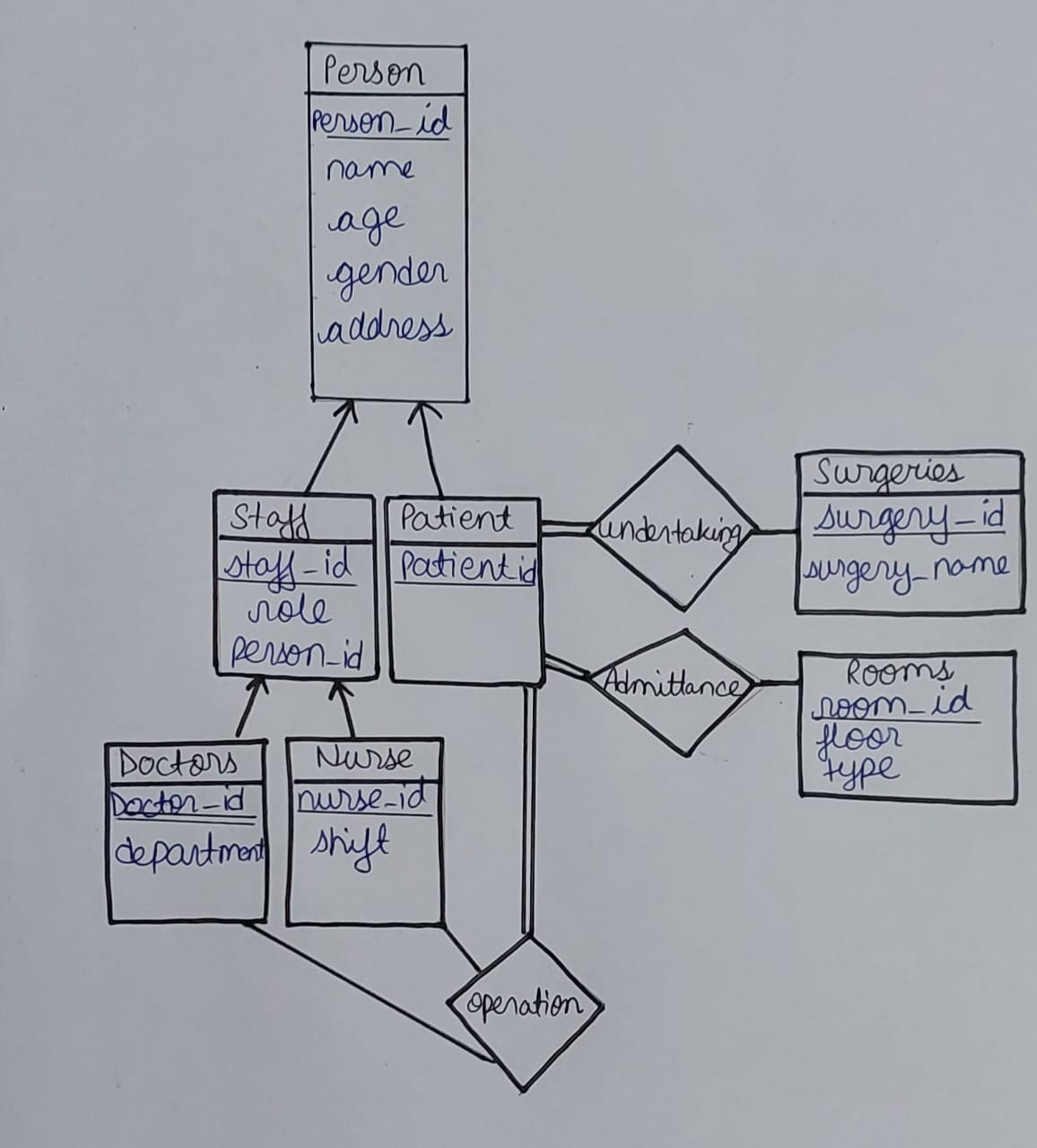
In this we will explore the role of database management systems in emergency ward management, discussing their benefits and challenges, and examining how healthcare organizations can maximize their potential to enhance patient care in emergency situations.

**CHAPTER 2: PROBLEM STATEMENT & OBJECTIVES**

Design and maintain a Database named "Emergency Ward" for a hospital to efficiently track and manage patient admissions for surgeries. The goal is to streamline the admittance process, allocate empty rooms, and assign appropriate doctors. The database begins with the highest-level entity set called "Person," which is further divided into "Staff" and "Patients". The "Staff" set includes "Doctor" and "Nurse," each containing basic information such as name, age, gender, and address. The "Doctor" table is linked to “Operation” table. Additionally, a ternary relationship called "Operation" involves “Staff" including both “Doctor” as well as “Nurse” and "Patients." Patients exhibit total participation in this relationship and other tables, including "Rooms" in a relationship “Admittance*”* for room allocation and another table named "Surgeries" that includes the names of different procedures (e.g. endoscopy) where the relationship is called “Undertaking”.

**CHAPTER 3: METHODOLOGY**

*ER Diagram:*

**

Person Table:

2NF: Yes

3NF: Yes, by separating address into its own table.

Staff Table:

2NF: Yes

3NF: Yes

Doctor Table:

2NF: Yes

3NF: Yes

Nurse Table:

2NF: Yes

3NF: Yes

Patient Table:

2NF: Yes

3NF: Yes

Rooms Table:

2NF: Yes

3NF: Yes

Admittance Table:

2NF: Yes

3NF: No, The Admittance table includes room\_id, which likely relates to room attributes such as floor and type stored in the Rooms table. These attributes depend on room\_id, then there exists a transitive dependency

Surgeries Table:

2NF: Yes

3NF: Yes, he Surgeries table contains information about surgeries, including surgery\_name which constant for the same number.

Operation Table:

2NF: Yes

3NF: Yes, Operation table links patients, doctors, and surgeries and the details are implied by the already present ids.

*Database Schema:*

Person (person\_id, name, age, gender, address)

Staff (staff\_id, role, person\_id)

Patient (patient\_id, person\_id)

Doctor (doctor\_id, department, staff\_id)

Nurse (nurse\_id, staff\_id,shift)

Rooms (room\_id, floor, type)

Admittance(admittance\_id, patient\_id, room\_id, admission\_date, discharge\_date)

Surgeries (surgery\_id, surgery\_name)

Operation(operation\_id,patient\_id,doctor\_id,surgery\_id,operation\_date)

*DDL Commands to create table with necessary constraints:*

*drop table person;*

*drop table staff;*

*drop table doctor;*

*drop table nurse;*

*drop table patient;*

*drop table rooms;*

*drop table admittance;*

*drop table surgeries;*

*drop table operation;*

CREATE TABLE Person (

person\_id INT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

age INT,

gender VARCHAR(10),

address VARCHAR(255)

);

CREATE TABLE Staff (

staff\_id INT PRIMARY KEY,

role VARCHAR(20),

person\_id INT REFERENCES Person(person\_id)

);

CREATE TABLE Doctor (

doctor\_id INT PRIMARY KEY,

department VARCHAR(50),

staff\_id INT REFERENCES Staff(staff\_id)

);

CREATE TABLE Nurse (

nurse\_id INT PRIMARY KEY,

staff\_id INT REFERENCES Staff(staff\_id),

shift VARCHAR(10)

);

CREATE TABLE Patient (

patient\_id INT PRIMARY KEY,

person\_id INT REFERENCES Person(person\_id)

);

CREATE TABLE Rooms (

room\_id INT PRIMARY KEY,

floor INT,

type VARCHAR(20)

);

CREATE TABLE Admittance (

admittance\_id INT PRIMARY KEY,

patient\_id INT REFERENCES Patient(patient\_id),

room\_id INT REFERENCES Rooms(room\_id),

admission\_date DATE,

discharge\_date DATE

);

CREATE TABLE Surgeries (

surgery\_id INT PRIMARY KEY,

surgery\_name VARCHAR(50)

);

CREATE TABLE Operation (

operation\_id INT PRIMARY KEY,

patient\_id INT REFERENCES Patient(patient\_id),

doctor\_id INT REFERENCES Doctor(doctor\_id),

surgery\_id INT REFERENCES Surgeries(surgery\_id),

operation\_date DATE

);

*Basic and Complex SQL Queries:*

1. Retrieve the names and ages of all patients admitted to the hospital.

SELECT p.name, p.age

FROM Patient pa, Person p

where pa.person\_id = p.person\_id;

2. Get the names and roles of all staff members (doctors and nurses).

SELECT p.name, s.role

FROM Staff s,Person p

where s.person\_id = p.person\_id;

3. Find the total count of patients admitted to each room type.

SELECT r.type, COUNT(a.patient\_id) AS patient\_count

FROM Admittance a,Rooms r

where a.room\_id = r.room\_id

GROUP BY r.type;

4. Retrieve the names and department of all doctors in the hospital.

SELECT p.name, d.department

FROM Doctor d

JOIN Staff s ON d.staff\_id = s.staff\_id

JOIN Person p ON s.person\_id = p.person\_id;

5. Find the names of nurses who work night shifts.

SELECT p.name

FROM Nurse n

JOIN Staff s ON n.staff\_id = s.staff\_id

JOIN Person p ON s.person\_id = p.person\_id

WHERE n.shift = 'Night';

6. Get the names of patients who have undergone a surgery.

SELECT p.name

FROM Operation o

JOIN Patient pa ON o.patient\_id = pa.patient\_id

JOIN Person p ON pa.person\_id = p.person\_id;

7. Get the total number of operations performed on each surgery type.

SELECT s.surgery\_name, COUNT(o.operation\_id) AS total\_operations

FROM Operation o

JOIN Surgeries s ON o.surgery\_id = s.surgery\_id

GROUP BY s.surgery\_name;

8. Retrieve the names of doctors who have performed surgeries.

SELECT p.name

FROM Operation o

JOIN Doctor d ON o.doctor\_id = d.doctor\_id

JOIN Staff s ON d.staff\_id = s.staff\_id

JOIN Person p ON s.person\_id = p.person\_id;

9. Find the names of nurses who are assigned to patients admitted to standard rooms.

SELECT p.name

FROM Nurse n

JOIN Staff s ON n.staff\_id = s.staff\_id

JOIN Person p ON s.person\_id = p.person\_id

JOIN Admittance a ON n.nurse\_id = a.patient\_id

JOIN Rooms r ON a.room\_id = r.room\_id

WHERE r.type = 'Standard';

10. Retrieve the names of patients who have not undergone any surgery.

SELECT p.name

FROM Patient pa

JOIN Person p ON pa.person\_id = p.person\_id

LEFT JOIN Operation o ON pa.patient\_id = o.patient\_id

WHERE o.operation\_id IS NULL;

*PL /SQL Queries*

1. Function to Get Surgery Name

CREATE OR REPLACE FUNCTION get\_surgery\_name(surgery\_id IN INT) RETURN VARCHAR AS

surgery\_name VARCHAR(50);

BEGIN

SELECT surgery\_name INTO surgery\_name FROM Surgeries WHERE surgery\_id = surgery\_id;

RETURN surgery\_name;

END;

/

2. Get names and department of all doctors in the hospital

CREATE OR REPLACE FUNCTION get\_doctor\_department(doctor\_id IN INT) RETURN VARCHAR AS

department VARCHAR(50);

BEGIN

SELECT department INTO department FROM Doctor WHERE doctor\_id = doctor\_id;

RETURN department;

END;

/

3.Function to Calculate Length of Stay for a Patient

CREATE OR REPLACE FUNCTION calculate\_length\_of\_stay(admittance\_id IN INT) RETURN NUMBER AS

length\_of\_stay NUMBER;

BEGIN

SELECT TRUNC(discharge\_date - admission\_date) INTO length\_of\_stay FROM Admittance WHERE admittance\_id = admittance\_id;

RETURN length\_of\_stay;

END;

/

4Trigger to add details in a log table

CREATE OR REPLACE TRIGGER log\_room\_changes

BEFORE UPDATE OF room\_id ON Admittance

FOR EACH ROW

BEGIN

INSERT INTO Room\_Log (admittance\_id, old\_room\_id, new\_room\_id, change\_date)

VALUES (:OLD.admittance\_id, :OLD.room\_id, :NEW.room\_id, SYSDATE);

END;

/

5.PL/SQL Trigger to Prevent Duplicate Surgeries

CREATE OR REPLACE TRIGGER prevent\_duplicate\_surgeries

BEFORE INSERT ON Operation

FOR EACH ROW

DECLARE

surgery\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO surgery\_count

FROM Operation

WHERE patient\_id = :NEW.patient\_id

AND surgery\_id = :NEW.surgery\_id;

IF surgery\_count > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Patient cannot undergo the same surgery twice.');

END IF;

END;

/

*Java code for functional design (DB connectivity)*

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or

// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.

import database.DatabaseConnection;

import gui.MainFrame;

import javax.swing.\*;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class Main {

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

try {

// Establish database connection

Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "system", "Prakhar13#");

if (con != null) {

System.out.println("Successful connection");

// Create database connection object

DatabaseConnection dbConnection = new DatabaseConnection(con);

// Create and show the main frame within the EDT

SwingUtilities.invokeLater(() -> {

MainFrame mainFrame = new MainFrame(dbConnection);

mainFrame.setVisible(true);

});

} else {

System.out.println("Error in connection");

JOptionPane.showMessageDialog(null, "Error connecting to database", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(null, "Error connecting to database: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

});

}

}

*Java code for functional design (SQL & PL/SQL Procedure/Function call /data access)*

private void executeQuery(String query) {

try {

ResultSet resultSet = null;

CallableStatement stmt = null;

PreparedStatement pstmt = null;

ResultSet rs;

switch (query) {

case "1. Retrieve the names and ages of all patients admitted to the hospital.":

resultSet = dbConnection.executeQuery("SELECT p.name, p.age " +

"FROM Patient pa, Person p " +

"WHERE pa.person\_id = p.person\_id");

break;

case "2. Get the names and roles of all staff members (doctors and nurses).":

resultSet = dbConnection.executeQuery("SELECT p.name, s.role " +

"FROM Staff s, Person p " +

"WHERE s.person\_id = p.person\_id");

break;

case "3. Find the total count of patients admitted to each room type.":

resultSet = dbConnection.executeQuery("SELECT r.type, COUNT(a.patient\_id) AS patient\_count " +

"FROM Admittance a,Rooms r where a.room\_id = r.room\_id " +

"GROUP BY r.type");

break;

case "4. Retrieve the names and department of all doctors in the hospital.":

resultSet = dbConnection.executeQuery("SELECT p.name, d.department " +

"FROM Doctor d " +

"JOIN Staff s ON d.staff\_id = s.staff\_id " +

"JOIN Person p ON s.person\_id = p.person\_id");

break;

case "5. Find the names of nurses who work night shifts.":

resultSet = dbConnection.executeQuery("SELECT p.name " +

"FROM Nurse n " +

"JOIN Staff s ON n.staff\_id = s.staff\_id " +

"JOIN Person p ON s.person\_id = p.person\_id " +

"WHERE n.shift = 'Night'");

break;

case "6. Get the names of patients who have undergone a surgery.":

resultSet = dbConnection.executeQuery("SELECT p.name " +

"FROM Operation o " +

"JOIN Patient pa ON o.patient\_id = pa.patient\_id " +

"JOIN Person p ON pa.person\_id = p.person\_id");

break;

case "7. Get the total number of operations performed on each surgery type.":

resultSet = dbConnection.executeQuery("SELECT s.surgery\_name, COUNT(o.operation\_id) AS total\_operations " +

"FROM Operation o " +

"JOIN Surgeries s ON o.surgery\_id = s.surgery\_id " +

"GROUP BY s.surgery\_name");

break;

case "8. Retrieve the names of doctors who have performed surgeries.":

resultSet = dbConnection.executeQuery("SELECT distinct(p.name) " +

"FROM Operation o " +

"JOIN Doctor d ON o.doctor\_id = d.doctor\_id " +

"JOIN Staff s ON d.staff\_id = s.staff\_id " +

"JOIN Person p ON s.person\_id = p.person\_id");

break;

case "9. Find the names of nurses who are assigned to patients admitted to standard rooms.":

resultSet = dbConnection.executeQuery("SELECT p.name " +

"FROM Nurse n " +

"JOIN Staff s ON n.staff\_id = s.staff\_id " +

"JOIN Person p ON s.person\_id = p.person\_id " +

"JOIN Admittance a ON n.nurse\_id = a.patient\_id " +

"JOIN Rooms r ON a.room\_id = r.room\_id " +

"WHERE r.type = 'Standard'");

break;

case "10. Retrieve the names of patients who have not undergone any surgery.":

resultSet = dbConnection.executeQuery("SELECT p.name " +

"FROM Patient pa " +

"JOIN Person p ON pa.person\_id = p.person\_id " +

"LEFT JOIN Operation o ON pa.patient\_id = o.patient\_id " +

"WHERE o.operation\_id IS NULL");

break;

case "1. Function to Get Surgery Name. ":

// Fetch surgeryId

String fetchSurgeryIdQuery = "SELECT surgery\_id FROM Surgeries";

pstmt = dbConnection.prepareStatement(fetchSurgeryIdQuery);

rs = pstmt.executeQuery();

while(rs.next()) {

int surgeryId = rs.getInt("surgery\_id");

stmt = dbConnection.prepareCall("{? = call get\_surgery\_name(?)}");

stmt.registerOutParameter(1, Types.VARCHAR);

stmt.setInt(2, surgeryId); // surgeryId needs to be fetched from the database or user input

stmt.execute();

String surgeryName = stmt.getString(1);

// Do something with surgeryName

}

break;

case "2. Get names and department of all doctors in the hospital. ":

// Fetch doctorId

String fetchDoctorIdQuery = "SELECT doctor\_id FROM Doctor";

pstmt = dbConnection.prepareStatement(fetchDoctorIdQuery);

rs = pstmt.executeQuery();

while(rs.next()) {

int doctorId = rs.getInt("doctor\_id");

// Assign to stmt here

stmt = dbConnection.prepareCall("{? = call get\_doctor\_department(?)}");

stmt.registerOutParameter(1, Types.VARCHAR);

stmt.setInt(2, doctorId); // doctorId needs to be fetched from the database or user input

stmt.execute();

String doctorDepartment = stmt.getString(1);

// Do something with doctorDepartment

}

break;

case "3. Calculate length of stay for a patient. ":

// Fetch admittanceId

String fetchAdmittanceIdQuery = "SELECT admittance\_id FROM Admittance";

pstmt = dbConnection.prepareStatement(fetchAdmittanceIdQuery);

rs = pstmt.executeQuery();

while(rs.next()) {

int admittanceId = rs.getInt("admittance\_id");

// Assign to stmt here

stmt = dbConnection.prepareCall("{? = call calculate\_length\_of\_stay(?)}");

stmt.registerOutParameter(1, Types.INTEGER);

// admittanceId needs to be fetched from the database or user input

stmt.setInt(2, admittanceId);

stmt.execute();

int lengthOfStay = stmt.getInt(1);

// Do something with lengthOfStay

}

break;

default:

resultArea.setText("Selected query not implemented yet.");

return;

}

if (resultSet != null) {

// Display the results in the text area

displayResultSet(resultSet);

} else {

resultArea.setText("ResultSet is null.");

}

} catch (SQLException ex) {

ex.printStackTrace();

resultArea.setText("Error executing query: " + ex.getMessage());

}

}

**CHAPTER 4: RESULTS & SNAPSHOTS**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**CHAPTER 5: CONCLUSION**

In conclusion, the development and implementation of an emergency ward management system that utilizes a database management system is crucial for ensuring the efficient and effective operation of hospital emergency wards. The system streamlines processes, optimizes resource utilization, and enhances coordination among healthcare providers, ultimately leading to improved patient outcomes and satisfaction.

By incorporating one-to-many relationships, such as patients with staff, the system can personalize care and ensure that each patient receives the necessary attention and treatment. Additionally, the real-time tracking and updating of patient records enable healthcare providers to make informed decisions quickly and accurately, further enhancing the quality of care delivered.

While there are challenges and complexities associated with designing and maintaining such a system, the benefits far outweigh these obstacles. Healthcare organizations can leverage the potential of database management systems to maximize patient care in emergency situations, improving overall hospital management and operations. Overall, the integration of database management systems in emergency ward management is essential for modern healthcare delivery and should be a priority for hospitals seeking to enhance their emergency care services.

**CHATER 6: LIMITATIONS & FUTURE WORK**

*LIMITATIONS:*

One potential limitation of this database system is the potential for data entry errors or inconsistencies, which could lead to inaccurate patient information being stored in the system. Additionally, as with any technology, there may be issues with system compatibility or technical difficulties that could impact the functionality of the database. Furthermore, the system may be limited in its ability to handle a high volume of patients or complex medical cases, leading to potential delays or errors in patient care.

*FUTURE WORK:*

In the future, this database system could be expanded to include additional features such as automated reminders for staff tasks, integrated billing and payment processing, and real-time monitoring of patient vitals. Additionally, incorporating artificial intelligence algorithms could help to improve decision-making processes and optimize resource allocation in the emergency ward. Furthermore, implementing secure access controls and encryption methods could enhance data security and privacy. Overall, continuous improvement and adaptation of the database system could further enhance the efficiency and effectiveness of emergency ward management.

**CHAPTER 7: REFERENCES**

* DBC Tutorial (Oracle): Official tutorial from Oracle covering JDBC basics and advanced topics.
* JDBC API Documentation: Official documentation of the JDBC API.
* Java Swing Tutorial (Oracle): Official tutorial from Oracle covering Swing basics and advanced GUI components. Link
* Swing API Documentation: Official documentation of the Swing API.