**Integrated Hospital Management System ( IHMS)**

**Team members**:

Batch A1:

Rohan Thomas ( 170905068, Roll no 10 )

Akshatha Kamath ( 170905072, Roll no 11)

**Abstract**

We intend to build a hospital management system. Each patient has a unique ID, as well as a full description including his/her personal details such as address, blood group, phone number, past illness as well as the disease and current treatment going on. The database also comprises of details of the doctors, their qualifications and the patients they are attending to. The software’s capabilities include adding new data into the system, maintaining and updating the doctor and blood bank data base, as well as providing information regarding bed availability in the hospital. The user can also search for availability of particular doctors from a particular medical department, as well as view the list of scheduled appointments

**Problem statement**

To design and implement an Integrated system for managing patient, doctor and other hospital records. The user is exposed to an easy to use GUI. The application is very intuitive and assists the user to structure his/her hospital appointments and view various other information. Its core functionalities involve storing, updating and retrieving information pertaining to doctors, appointments, blood bank through user-friendly menu driven modules. The data requirements involve the patient records, doctor and department records, blood bank details, room booking details as well as the appointment details. This develops and is superior to the already existing systems in that it allows the desired level of transparency to the user and allows better control to the doctors. We aim to achieve better health care as well as an increasingly transparent hospital management to ensure healthcare for all.

**Schema**

Patient ( pid, pat\_name, dob, gender, email, address, blood\_group)

Doctor(did, dno, doc\_name, address, email, phone, gender)

Appointment(pid, did, ap\_date, ap\_time)

Room(rno, type,floor)

Dependent(pid, dep\_name, gender, relation)

Blood\_bank(pid, bd\_day, blood\_group, quantity)

Booking(rno, pid, in\_date, out\_date)

Department(dno, dname, dept\_head)

**DDL Commands to Create Tables**

drop table donations;

drop table booking;

drop table blood\_bank;

drop table room;

drop table appointment;

drop table patient;

drop table doctor;

drop table department;

create table donations(donate number(5) primary key);

create table department(

dno number(3) primary key,

dept\_name varchar(20) not null,

dept\_head number(4) not null);

create table patient(

pid number(4) primary key,

pat\_name varchar(20) not null,

dob date not null,

gender varchar(1) not null,

email varchar(20) not null,

phno number(10) not null,

address varchar(20) not null,

blood\_group varchar(5) not null,

check (gender in ('M', 'F', 'm', 'f','O','o'))

);

create table doctor(

did number(4) primary key,

dno number(3) references department,

doc\_name varchar(20) not null,

address varchar(15) not null,

email varchar(20) not null,

gender varchar(1) not null,

phone number(10) not null,

check (gender in ('M', 'F', 'm', 'f','O','o'))

);

create table appointment(

pid number(4) references patient,

did number(4) references doctor,

ap\_date date not null,

primary key(pid,did,ap\_date));

create table room(

rno number(4) primary key,

type varchar(10) not null,

floor number(2) not null

);

create table blood\_bank(

pid number(4) references patient,

bd\_day timestamp not null,

blood\_group varchar(3) not null,

amount number(4) not null,

primary key(pid,bd\_day)

);

create table booking(

pid number(4) references patient,

rno number(4) references room,

in\_date date not null,

out\_date date not null,

primary key(pid,rno));

**DB-Connectivity and Access**

**try**{

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.*getConnection*( "jdbc:oracle:thin:@localhost:1521:xe","system","awesome12");

Statement stmt=con.createStatement();

ResultSet rs = stmt.executeQuery(query);

con.close();

}

**catch**(SQLException e){

JOptionPane.*showMessageDialog*(**this**,e.getMessage().toString());

}

**catch**(Exception e)

{

JOptionPane.*showMessageDialog*(**this**,"Error");

}

**List of Queries Used**

**Triggers**

We have used triggers to calculate the number of blood donations that take place every time a patient donates blood. Since a patient may withdraw blood, the check for only donation is taken into consideration.

The following is the code snippet which has to be executed in sql:

create or replace trigger don\_count

before insert on blood\_bank

for each row

begin

if :New.amount>0 then

update donations set donate=donate+1;

end if;

end;

/

**Procedures**

We have used a pl/sql procedure with in and out parameters in order to stimulate donation or withdrawal from the blood bank. Donations are free to take place but for a withdrawal, the current amount in the blood bank must be more than what the patient requires. Should that not be the case, the insert query is revoked.

The following is the code snippet which has to be executed in sql:

create or replace procedure insert\_bank(chk out number,pid1 blood\_bank.pid%type,quan1 blood\_bank.amount%type) is

tot number;

bg blood\_bank.blood\_group%type;

insufficientBlood Exception;

begin

select blood\_group into bg from patient where pid=pid1;

savepoint s1;

insert into blood\_bank values(pid1,systimestamp,bg,quan1);

select sum(amount) into tot from blood\_bank where blood\_group=bg;

if tot<0 then

chk:=0;

raise insufficientBlood;

end if;

chk:=1;

exception

when insufficientBlood then

rollback to s1;

end;

/

