J.Sriganesh 205002095
Swetha Subramanian 205002111
Thanseer Hishak 205002113

DBMS PROJECT REPORT

Problem Statement: **Hospital Management System.**

Develop a database management system to manage a hospital. Assign unique IDs to the patients and store the relevant information under the same. You'll have to add the patient's name, personal details, contact number, disease name, and the treatment the patient is going through. You'll also have to mention under which hospital department the patient is (such as cardiac, gastro, etc.).

After that, you should add information about the hospital's doctors. A doctor can treat multiple patients, and he/she would have a unique ID as well. Doctors would also be classified in different departments.

Requirement Analysis:

Based on the given problem statement there are two types of requirement:

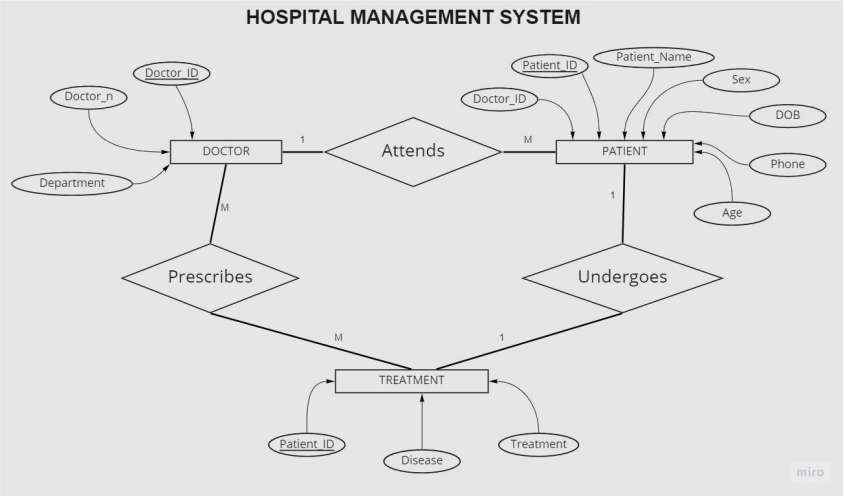
1) User-Input requirement: It is the data that has to be collected/Input from user to perform queries and return the required output.

2) Prerequisite Requirement: It is the data that has to be already stored in database in order to perform the required queries and return required output.

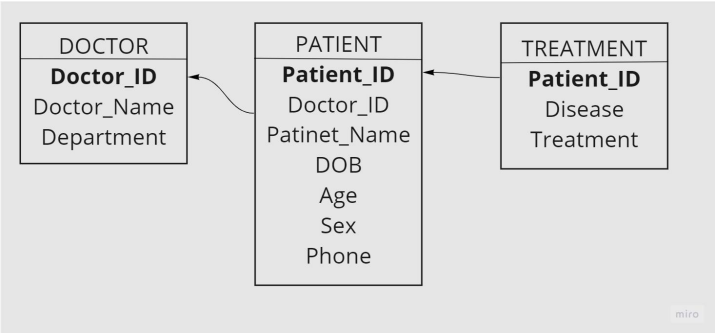
Data to be collected or stored:

Patient Details like Name,Age,DOB,Sex,Contact Information,etc.
Doctor Details like Name,Specialization.
Medicines or Treatment given.
Name of Disease.

ER DIAGRAM:



DATABASE SCHEMA:



DATABASE NORMALISATION:

Before Normalization:

1.Patients_Table:

Patient ID	Patient Name	DOB	AGE	Sex	Ph No:	Doctor ID	Department	Disease	Treatment

2.Doctors_Table:

Doctor ID	Doctor Name	Department

The Tables are already in 1st Normal Form since there is only a single value in each field.

To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency.
Here in 1st table
Example:

Patient ID	Doctor ID	Department
1000	100	Cardiology
1001	101	General medicine
1002	103	Ent
1003	103	Ent

Here,
Department cannot alone decide the value of Patient ID;
Department together with Doctor ID cannot decide the value of Patient ID;
Department together with Patient ID cannot decide the value of Doctor ID;
Hence,
Department would be a non-prime attribute, as it does not belong to the one only candidate key {Patient ID,Doctor ID} ;
But, Doctor ID -> Department, i.e., Department is dependent on Doctor ID, which is a proper subset of the candidate key. Non-prime attribute Department is dependent on a proper subset of the candidate key, which is a partial dependency and so this relation is not in 2NF.

To convert the above relation to 2NF,
we need to split the table into two tables such as :
Table 1: Patient ID, Doctor ID
Table 2: Doctor ID, Department

<u>So Tables in 2NF will be:</u>

Patients Table						
Patient ID	Patient Name	DOB	AGE	Sex	Phone No:	Doctor ID

Doctors Table		
Doctor ID	Doctor Name	Department

Treatment Table		
Patient ID	Disease Name	Treatment

Now the relation is in 2NF form with 3 tables.

Additionally we need a Password Manager that gives separate access to each User.

We have 3 Users as follows:
1. Admin
2. Doctor
3. Receptionist

- We need to assign each with proper rights for their roles.Thus Admin can view all Accounts.
- While Doctor can diagnose the patient and add disease and treatment details into the Treatment Table.
- The Receptionist should be able to add patients and delete them with patient details.They must also be able to check if the patient is admitted/got an appointment.

Since Password table is not related to the actual database we have a separate table called PassData as follows:

Passdata		
UserName	Password	Designation

While has the data prerecorded or Stored beforehand so that it checks if the user details are correct and gives the appropriate role permissions.

The important SQL commands / Tkinter commands used:

MODULES USED:

```
from os import pardir
from tkinter import *
import tkinter as tk
from tkinter import ttk
from subprocess import PIPE, call
from tkinter import messagebox
from tkinter.messagebox import askyesno
import pyodbc
import tkinter
```

Tkinter,ttk,message for GUI design.
Subprocess to link Windows.
Pyodbc to establish connection to SQL server.

BUTTON COMMANDS:

```
#TO CLEAR THE ENTRY BOXES
def btn_clear():
    entry01.delete(0,END)
    entry11.delete(0,END)
    entry21.delete(0,END)
    entry31.delete(0,END)
    entry41.delete(0,END)
    entry51.delete(0,END)
    entry61.delete(0,END)

#TO REFRESH
def btn_refresh():
    root.destroy()
    caller = Callpy("C:/Users/thans/Desktop/TRIAL/RECEPTIONIST.py")
    caller.CallFile()

#TO UPDATE VALUES
def btn_update():
    answer = askyesno(title='confirmation',message='Are you sure that you want to Update?')
    if answer:
        ppidd = input711.get()
        pname = input611.get()
        pdob = input511.get()
        page = input411.get()
        psex = input311.get()
        pphno = input211.get()
        pdid = input111.get()
        cursor = conn.cursor()
        cursor.execute("UPDATE Patients_Table SET Patient_Name = (?),DOB = (?),
       Age = (?),Sex = (?),Phone_Number = (?),Doctor_ID = (?)
       WHERE Patient_ID = (?)",(pname,pdob,page,psex,pphno,pdid,ppidd))
        conn.commit()
        conn.close()
        root.destroy()
        caller = Callpy("C:/Users/thans/Desktop/TRIAL/RECEPTIONIST.py")
        caller.CallFile()
```

```
#TO INSERT VALUES INTO THE ENTRY BOXES
def btn_select():
    selected = tree3.focus()
    values = tree3.item(selected, 'values')
    entry61.insert(0,values[0])
    entry51.insert(0,values[1])
    entry41.insert(0,values[2])
    entry31.insert(0,values[3])
    entry21.insert(0,values[4])
    entry11.insert(0,values[5])
    entry01.insert(0,values[6])
```

#TO INSERT INTO THE TABLE

```
def btn_insert():
    answer = askyesno(title='confirmation',message='Are you sure that you want to ADD?')
    if answer:
        Pname1 = input6.get()
        dob1 = input5.get()
        age1 = input4.get()
        sex1 = input3.get()
        phno1 = input2.get()
        DID1 = input1.get()
        cursor = conn.cursor()
        cursor.execute(
            "Insert  into  Patients_Table(Patient_Name,DOB,Age,Sex,Phone_Number,Doctor_ID)
values (?,?,?,?,?,?);", (Pname1, dob1, age1,sex1,phno1,DID1))
        conn.commit()
        entry0.delete(0,END)
        entry1.delete(0,END)
        entry2.delete(0,END)
        cursor.close()
        root.destroy()
        caller = Callpy("C:/Users/thans/Desktop/TRIAL/RECEPTIONIST.py")
        caller.CallFile()
```

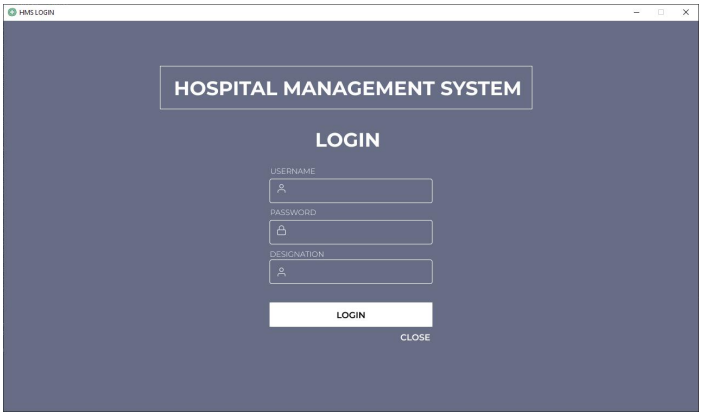
```
#TO DELETE DATA
def btn_delete():
    answer  =  askyesno(title='confirmation',message='Are  you  sure  that  you  want  to
Delete?\nChanges Made Are permanent!')
    if answer:
        x = tree4.selection()[0]
        selected =tree4.focus()
        arr = tree4.item(selected,'values')
        cursor = conn.cursor()
        cursor.execute("DELETE FROM Patients_Table where Patient_ID = (?);",(arr[0]))
        conn.commit()
        tree4.delete(x)
        root.destroy()
        caller = Callpy("C:/Users/thans/Desktop/TRIAL/RECEPTIONIST.py")
        caller.CallFile()
```

```
#TO DISPLAY WITH PATIENT NAME
def btn_view():
    pid141 = input141.get()
    x = conn.cursor()
    for i in x.execute("select * from Patients_Table where Patient_Name = (?)",(pid141)):
        if i[0] == None:
            messagebox.showinfo("ERROR","Invalid Input or Patient Doesn't exist")
        tree5.insert(parent='',index='end',iid=
i ,text="",values=(i[0],i[1],i[2],i[3],i[4],i[5],i[6]))
    conn.commit()
    x.close()
```

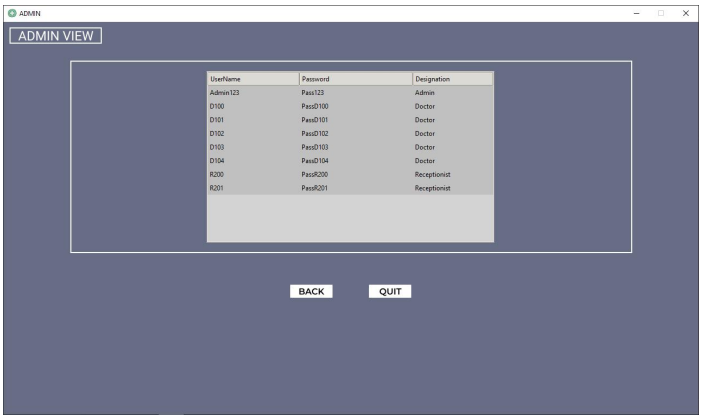
```
#TREE VIEW FOR DOCTOR:
#Table1
tree1 = ttk.Treeview(window)
tree1['columns']                                                                                =
("Doctor_ID","Doctor_Name","Department","Patient_ID","Disease","Treatment")

#column format
tree1.column("#0",width = 0)
tree1.column("Doctor_ID",anchor=W,width = 40,minwidth=40)
tree1.column("Doctor_Name",anchor=W,width = 100,minwidth=100)
tree1.column("Department",anchor=W,width = 100,minwidth=100)
tree1.column("Patient_ID",anchor=CENTER,width = 40,minwidth=40)
tree1.column("Disease",anchor=CENTER,width = 120,minwidth=120)
tree1.column("Treatment",anchor=CENTER,width = 120,minwidth=120)

#headings
tree1.heading("#0",text="",anchor=W)
tree1.heading("Doctor_ID",text="Doctor_ID",anchor=W)
tree1.heading("Doctor_Name",text="Doctor_Name",anchor=W)
tree1.heading("Department",text="Department",anchor=W)
tree1.heading("Patient_ID",text="Patient_ID",anchor=CENTER)
tree1.heading("Disease",text="Disease",anchor=CENTER)
tree1.heading("Treatment",text="Treatment",anchor=CENTER)
tree1.place(x=532,y=65,width=697,height=378)

#Parent
x = conn.cursor()
count=0
for i in x.execute("select * from DoctorTable"):
    tree1.insert(parent='',index='end',iid= count ,text="",values=(i[0],i[1],i[2]))
    count +=1
conn.commit()
x.close()

#Child

l1=l2=l3=l4=l5=[]

#Doctor 1
y1 = conn.cursor()
y1.execute("select  *  from  TreatmentTable  where  Patient_ID  in(select  Patient_ID  from
Patients_Table where Doctor_ID =100)")
for i in y1:
    l1.append(i)
if len(l1)>0:
    count1 = 0
    itr = 100
    for j in range(len(l1)):

tree1.insert(parent='0',index='end',iid=itr,text='',values=("","","",l1[j][0],l1[j][1],l1[
j][2]))
        itr +=1
        count1 +=1

y1.close()
```

```
#CHECK USER LOGIN
def submit():
    username = input1.get()
    password = input2.get()
    designation = input3.get()

    c = conn.cursor()
    c.execute("Select password from PassData where username = (?)",(username))
    for i in c:
        if(i[0] == password):
            d = conn.cursor()
            d.execute("Select designation from PassData where username = (?)",(username))
            for j in d:
                if(j[0] == designation):
                    if designation == "Admin":
                        window.destroy()
                        caller = Callpy("C:/Users/thans/Desktop/TRIAL/ADMIN.py")
                        caller.CallFile()
                    elif designation == "Doctor":
                        window.destroy()
                        caller = Callpy("C:/Users/thans/Desktop/TRIAL/DOCTOR.py")
                        caller.CallFile()
                    elif designation == "Receptionist":
                        window.destroy()
                        caller = Callpy("C:/Users/thans/Desktop/TRIAL/RECEPTIONIST.py")
                        caller.CallFile()
                else:
                    messagebox.showwarning("Warning","Invalid designation")
        else:
            messagebox.showwarning("Warning","Invalid password or username")

    conn.commit()
```

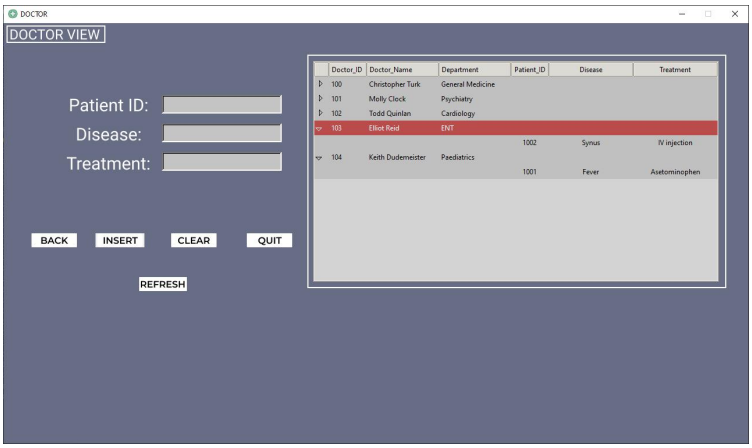
LOGIN WINDOW:



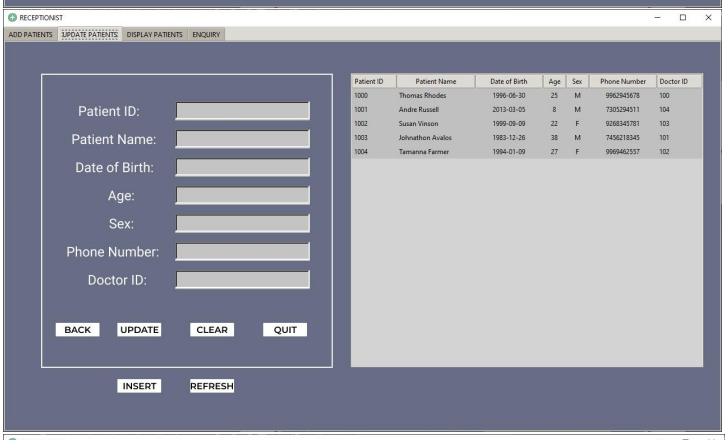
ADMIN LOGIN:

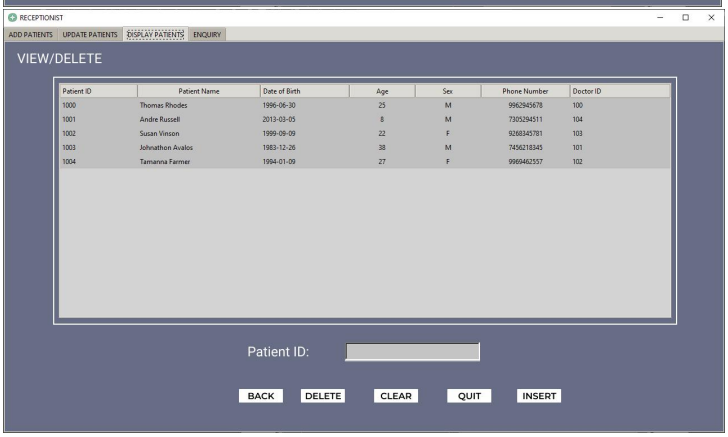


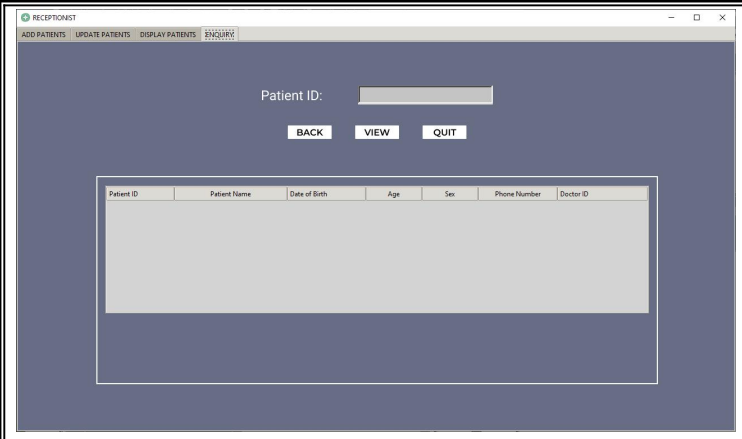
DOCTOR LOGIN:



RECEPTIONIST LOGIN:






RECEPTIONIST
ADD PATIENTS
UPDATE PATIENTS
DISPLAY PATIENTS
ENQUIRY
Patient ID:
BACK
VIEW
QUIT
Patient ID
Patient Name
Date of Birth
Age
Sex
Phone Number
Doctor ID