Visvesvaraya Technological University

Jnana Sangama, Belagavi – 590018, Karnataka

A Mini Project Report

on

“QUARANTINE CENTER MANAGEMENT SYSTEM”

Submitted in partial fulfillment of the requirement for the DBMS Laboratory with mini project (18CSL58) of V semester

Bachelor of Engineering

In

Computer Science and Engineering

Submitted By

PATEL ARPIT CHIRAGBHAI (1GA18CS104)

Under the Guidance of

Mrs. Reshma S

Assistant Professor,

Dept. of CSE

GLOBAL ACADEMY OF TECHNOLOGY

Department of Computer Science and Engineering

(Accredited by NBA 2019-2022)

Raja Rajeshwari Nagar, Bengaluru – 560 0098

GLOBAL ACADEMY OF TECHNOLOGY

Department of Computer Science and Engineering

(Accredited by NBA 2019-2022)

Raja Rajeshwari Nagar, Bengaluru – 560 098

Certificate

This is to certify that V Semester Mini project entitled **"QUARANTINE CENTER MANAGEMENT SYSTEM"** is a bonafide work carried out by **PATEL ARPIT CHIRAGBHAI (1GA18CS104)**as a partial fulfillment for the award of Bachelor's Degree in Computer Science and Engineering for DBMS Laboratory with Mini Project [18CSL58] as prescribed by **Visvesvaraya Technological University**, **Belagavi** during the year 2020-2021.

Mrs.Reshma S
Assistant Professor,
Dept of CSE,
GAT, Bengaluru.

Dr. Srikanta Murthy K
Professor & Head,
Dept of CSE,
GAT, Bengaluru.

External Exam

Name of the Examiner	Signature with date
1.________________________	________________________
2.________________________	________________________

ABSTRACT

This project intends to include various features related to a quarantine center i.e. information about no of total rooms, vacant rooms, number of patients, patient details etc. The main aim is to help people who are infected with aur are suspected to be infected with the covid-19 virus know about how to proceed further, like they can get the details about the quarantine center, no of rooms, facility in the rooms, doctors associated with us etc . People who wish to stay in a quarantine center can access these details and decide accordingly. Patients have to show a fresh report which states that the person is found positive and after submitting their personal details, they can book a room for themselves according to their condition i.e. if they are asymptomatic or minor fever then they can book simple rooms and if their condition is not good and have high fever, breathing problems etc then they can book rooms with all facilities.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance crowned our efforts with success.

We consider ourselves proud, to be part of **Global Academy of Technology** family, the institution which stood by our way in endeavors.

We express our deep and sincere thanks to our Principal **Dr. N. Rana Pratap Reddy** for his support.

We are grateful to **Dr. Srikanta Murthy K,** Professor and Head, Dept. of Computer Science & Engineering who is source of inspiration and of invaluable help in channelizing my efforts in right direction.

We wish to thank our internal guide **Mrs. Reshma S**, Assistant Professor, Dept of CSE for guiding and correcting various documents with attention and care. She has taken lot of pain to go through the document and make necessary corrections as and when needed.

We would like to thank the faculty members and supporting staff of the Department of CSE, GAT for providing all the support for completing the Project work.

Finally, we are grateful to our parents and friends for their unconditional support and help during our Project work.

PATEL ARPIT CHIRAGBHAI (1GA18CS104)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION ABOUT SQL:

Quarantine center is a place where covid-19 positive patients are kept or rather admitted and are alotted according to their conditions. There they are checked regularly by either permanent or visiting doctors and are governed, kept care by nurses. Along with this, the administration keeps record of the number of patients, patient details, record of rooms and its equipments, doctor details, nurse details, visitor details etc. Doing all this in pen paper mode can be very difficult and can have faults which can serious problems in dealing with such a serious issue.

Here people can do all this digitally which won't have errors. For booking rooms, patients or their relatives won't have to come physically and risk their or other people's lives. Also they can pay the bills online itself to avoid physical contact.If a serious patient slowly recovers and then is shifted to normal room, it will be displayed in the patient profile so the relatives can check their progress. Details of the currently admitted patients and the patients who were admitted in the past are stored for further use. Reviews are also taken by the people which can help to improve the system. Some of the features that it can include in this quarantine center management system are:

- **Patients and rooms database management:** The details of the patients like name, address, room alotted, contact can be stored easily through this application**.**
- **Reports:** The reports that are given by the doctors can also be stored through this application.
- **Employee details**: The details of the doctors, nurses and guards can be stored easily through this application
- **Reviews**: The reviews by the people who have visited and the patients who are released after recovering are collected in a feedback portal.

1.2 INTRODUCTION TO FRONTEND SOFTWARE

Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's cross-platform, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run.

Although Tkinter is considered the de-facto Python GUI framework, it's not without criticism. One notable criticism is that GUIs built with Tkinter look outdated. If you want a shiny, modern interface, then Tkinter may not be what you're looking for.

However, Tkinter is lightweight and relatively painless to use compared to other frameworks. This makes it a compelling choice for building GUI applications in Python, especially for applications where a modern sheen is unnecessary, and the top priority is to build something that's functional and cross-platform quickly.

1.3 PROJECT REPORT OUTLINE

The report is arranged in the following way:

Chapter 1:	Gives the information of the project
Chapter 2:	Gives the Requirement Specification of the project
Chapter 3:	Gives the Objective of the project
Chapter 4:	Gives the Implementation of the project
Chapter 5:	Gives the Front end design of the project
Chapter 6:	Gives the Testing information of the project
Chapter 7:	Gives the Results of the project

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 SOFTWARE REQUIREMENTS

Operating System :	Windows 10
Database :	MySQL
Tools :	Python

2.2 HARDWARE REQUIREMENTS

Processor :	Intel i5 8th Generation
RAM :	8 GB
Hard Disk :	1 TB
Compact Disk :	CD-ROM, CD-R, CD-RW
Input device :	Keyboard, Mouse
Output device :	DESKTOP

CHAPTER 3

OBJECTIVE OF THE PROJECT

The main objective of the Quarantine Center Management System is to manage the Patient information like Patient's name, Patient's Age, Date of Admission etc. This Application is build for the Hospital and hence, only Doctors, Nurses and Receptionist can access the Private information.

The purpose of the project is to reduce manly work and make it easy to store and access the information for the Hospital.

The system includes different functional divisions which are-

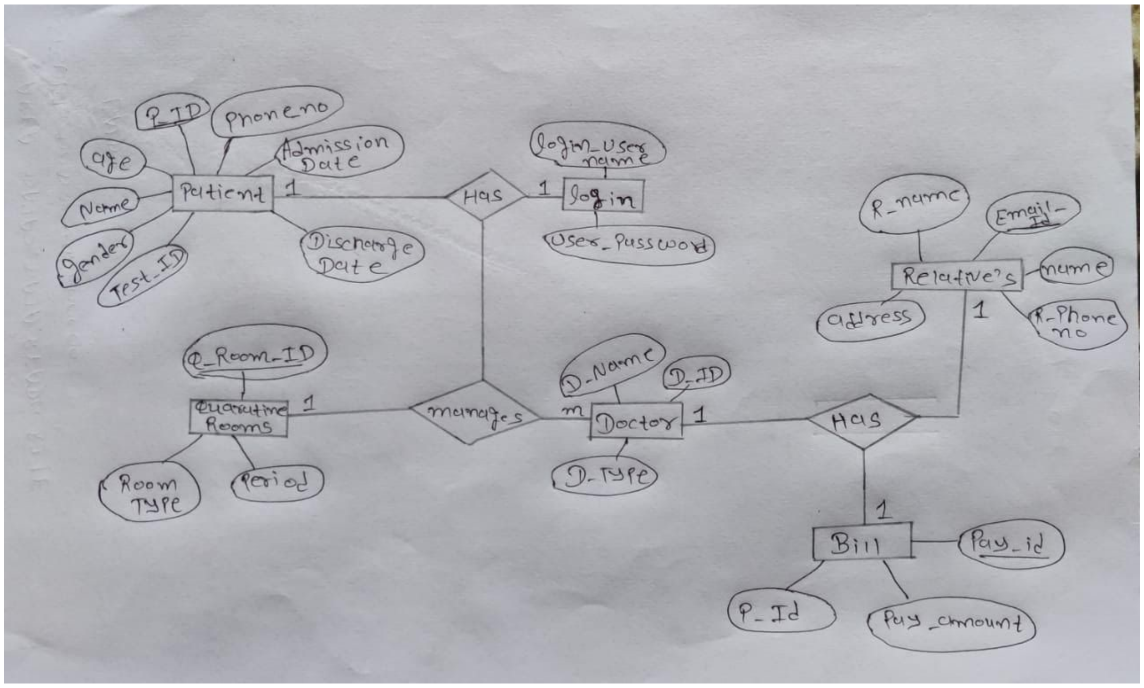
- Insertion of Patient's information
- Accessing and Viewing the information
- Deletion of information
- Searching from the Database
- Update the information
- Accessing the Patient's Information

The project shows information and description of Patients and Relatives thus increasing the efficiency of managing. It deals with monitoring the information for the Hospital. Adding, Deleting, Searching and Updating of records is improved which results in proper resource management for the Hospital.

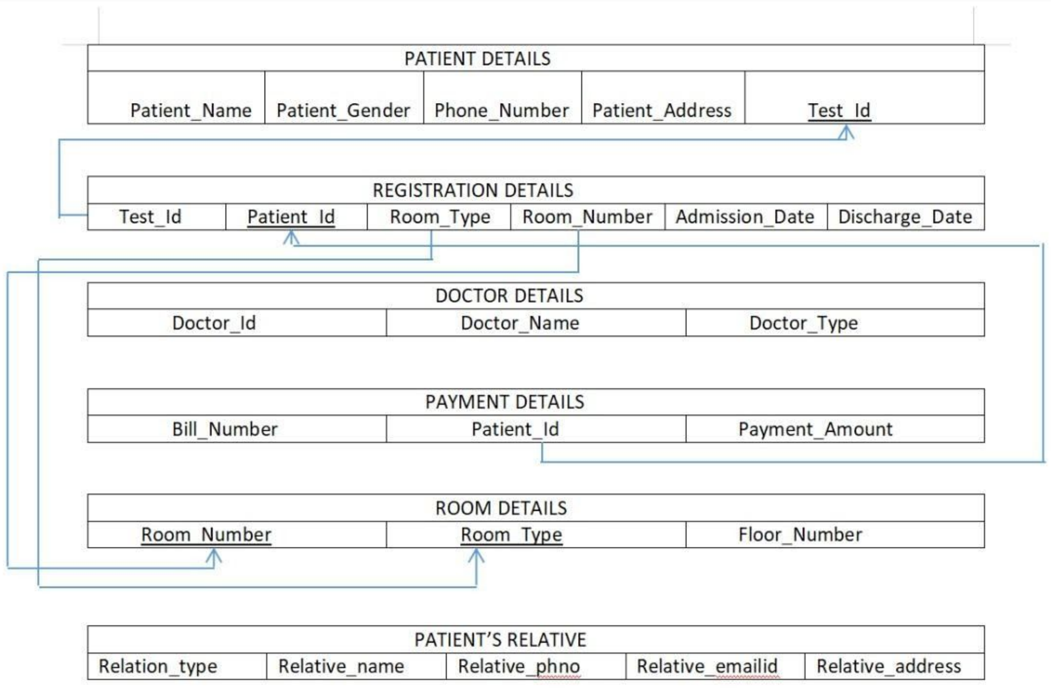
CHAPTER 4

IMPLEMENTATION

4.1 ER DIAGRAM



4.2 MAPPING OF ER DIAGRAM TO SCHEMA DIAGRAM



PATIENT DETAILS				
Patient_Name	Patient_Gender	Phone_Number	Patient_Address	Test_Id

REGISTRATION DETAILS					
Test_Id	Patient_Id	Room_Type	Room_Number	Admission_Date	Discharge_Date

DOCTOR DETAILS		
Doctor_Id	Doctor_Name	Doctor_Type

PAYMENT DETAILS		
Bill_Number	Patient_Id	Payment_Amount

ROOM DETAILS		
Room_Number	Room_Type	Floor_Number

PATIENT'S RELATIVE				
Relation_type	Relative_name	Relative_phno	Relative_emailid	Relative_address

4.3 MAPPING OF THE ER SCHEMA TO RELATIONS

For each regular entity type E in the ER schema, create relation R that includes all simple attributes of E**.**

STEP 1: Mapping of Regular Entities

For each regular entity type E in the ER schema, create relation R that includes all simple attributes of E.

patient_name	**patient_gende r**	**patient_Numb er**	**patient_addre ss**	**Test_id**

Test_id	**patient_id**	**Room_type**	**Room_No**	**Addmission _date**	**discharge_ date**

Doctor_id	**Doctor_name**	**Doctor_Type**

Bill_number	**patient_id**	**Payment_Amount**

Room no.	**Room_type**	**Floor_number**

STEP 2: Mapping of Weak Entity Types

For each weak entity, create a table that includes all of its simple attributes.

Doctor_id	**Doctor_name**	**Doctor_Type**

patient_name	patient_gender	patient_Number	patient_address	Test_id

STEP 3: Mapping of 1-1 Relationship

Identify the relation S that represents the participating entity type at the 1-side of the relationship type.

patient_name	patient_gender	patient_Number	patient_address	Test_id

Include as foreign key in S the primary key of the relations T that represents the other entity type participating in R.

STEP 5: Mapping of M-N Relationship

Create a new relation S to represent R.

Include as foreign key attributes in S the primary key of the relations that represents the participating entity types their combination will form the primary key of S.

Also, include any simple attributes of the M:N relationship type as attributes of S.

Bill_number	patient_id	Payment_Amount

Room no.	Room_type	Floor_number

STEP 6: Mapping of multi-valued attributes

For each multivalued attributes A, create a new relation R. This relation R will include an attribute corresponding to A. plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.

The Primary Key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

STEP 7.: Mapping of N-Ary Relationship Types

For each n-ary relationship type R, where n>2 create a new relationship S to represent R. λ include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.

λ also includes any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S

There are no n-ary relationship types.

4.4 NORMALIZE

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update and anomaly.

FIRST NORMAL FORM (1NF)

A Data is said to be in first normal form if:

1. There are no duplicate rows in the table.
2. Each cell is single valued or atomic

SECOND NORMAL FORM (2NF)

A Database is said to be in second normal form if

1.If the tables are in 1NF and if all non-key attributes of the tables are fully functionally dependent on all the key attributes.

THIRD NORMAL FORM (3NF)

A Database is said to be in second normal form if

1. If all the tables in it are in 2NF and without any transitive dependencies, i.e X>Y, Y>Z, X>Z.

2. According to CODD‟s definition a relation schema R is in 3NF if it satisfies 2NF and all non-prime attributes are transitively dependent on the primary key.

4.5 CREATION OF TABLES

CREATION OF PATIENT TABLE

```
Create Table Patient
(
   Patient_name Char(20)  Primary Key,
   Gender Char(12) NOT NULL,
   PhNo Number(10) NOT NULL,
   Address Varchar(40) NOT NULL,
   Test_id Number(10) NOT NULL);
```

CREATION OF REGISTRATION TABLE

```
Create Table Registration
(
   Test_id References Patient(Test_id) On Delete Cascade,
   Patient_id Number(6) Primary Key,
   Room_Type References Room(Room_Type) On Delete Cascade,
   Room_Number References Room(Room_Number) On Delete Cascade,
   Admission_Date Varchar(12) NOT NULL ,
   Discharge_Date Varchar(12) NOT NULL
);
```

CREATION OF DOCTOR TABLE

```
Create Table Doctor
(
    Doctor_id Varchar(10) PRIMARY KEY,
    Name Char(20) NOT NULL,
    Type Char(12) NOT NULL
);
```

CREATION OF PAYMENT TABLE

```
Create Table Payment
(
    Bill_No Number(12) PRIMARY KEY,
    Patient_id References Registration(Patient_id) On Delete Cascade,
    Amount Varchar(12)
);
```

CREATION OF RELATIVE TABLE

```
Create Table Relative
(
    Type Char(20) NOT NULL,
    Name Char(20) NOT NULL,
    Phone Number(10) NOT NULL,
    EmailId Varchar(24) PRIMARY KEY,
    Address Varchar(40) NOT NULL
);
```

4.6 INSERTION OF TUPLES

INSERTING INTO PATIENT TABLE

```
INSERT INTO Patient VALUES ('Satyam', 'M', 6205620273, 'Sector-4 Qr No-210',
1234567890);

INSERT INTO Patient VALUES ('Arpit', 'M', 7340412567, 'Sector-8 Qr No-255', 2234567890);

INSERT INTO Patient VALUES ('Ajit', 'M', 6206994821, 'Sector-5 Qr No-107', 3234567890);

INSERT INTO Patient VALUES ('Vinita', 'F', 8210244325, 'Sector-5 Qr No-200', 4234567890);

INSERT INTO Patient VALUES ('Akshay', 'M', 6203102997, 'Sector-3 Qr No-353',
5234567890);

SELECT * FROM Patient;
```

INSERTING INTO REGISTRATION TABLE

```
INSERT INTO Registration VALUES (1234567890, 9837463569, 'Normal', 3, '12/10/20',
'27/10/20');

INSERT INTO Registration VALUES (2234567890, 7483947320, 'Normal', 2, '14/10/20',
'29/10/20');

INSERT INTO Registration VALUES (3234567890, 7322947394, 'Equipped', 7, '07/10/20',
'01/10/20');

INSERT INTO Registration VALUES (4234567890, 4347483858, 'Normal', 5, '09/10/20',
'23/10/20');

INSERT INTO Registration VALUES (5234567890, 6647348438, 'Equipped', 6, '08/10/20',
'22/10/20');

SELECT * FROM Registration;
```

INSERTING INTO DOCTOR TABLE

```
INSERT INTO Doctor VALUES (101, 'Siddharth', 'Visiting');

INSERT INTO Doctor VALUES (102, 'Preeti', 'Visiting');

INSERT INTO Doctor VALUES (103, 'Pramod', 'Permanent');

SELECT * FROM Doctor
```

INSERTING INTO PAYMENT TABLE

```
INSERT INTO Payment VALUES (132, 9837463569, 'Rs1200');

INSERT INTO Payment VALUES (126, 7483947320, 'Rs1300');

INSERT INTO Payment VALUES (143, 7322947394, 'Rs7300');

INSERT INTO Payment VALUES (137, 4347483858, 'Rs1200');
INSERT INTO Payment VALUES (129, 6647348438, 'Rs6500');

SELECT * FROM Payment
```

INSERTING INTO RELATIVE TABLE

```
INSERT INTO Relative VALUES ('Father', 'PK Dubey', 8210199836, 'abcd@gmail.com',
'Sector-4 Qr No-210');

INSERT INTO Relative VALUES ('Father', 'CB Patel', 7459873263, 'bcde@gmail.com',
'Sector-5 Qr No-107');

INSERT INTO Relative VALUES ('Father', 'Shakti Das', 8430456210, 'cdef@gmail.com',
'Sector-5 Qr No-107');

INSERT INTO Relative VALUES ('Father', 'VD Pandey', 9108169764, 'defg@gmail.com',
'Sector-5 Qr No-200');

INSERT INTO Relative VALUES ('Father', 'AK Chaoudhary', 9304658077, 'efgh@gmail.com',
'Sector-3 Qr No-353');

SELECT * FROM Relative
```

4.7 CREATION OF TRIGGERS

A trigger is a special kind of a store procedure that executes in response to certain action on the table like insertion, deletion or updation of data.

CREATING A TRIGGER WHICH RAISES AN ERROR WHEN ENTERED INCORRECT EMAIL FORMAT

```
CREATE TRIGGER trigg BEFORE INSERT

ON 'relatives' FOR EACH ROW

INSERT INTO  trigg VALUES(NOW());
```

CHAPTER 5

FRONT END DESIGN

5.1 CONNECTIVITY TO DATABASE

```
 def add_patients(self):
        if self.Test_ID.get() == "" or self.Name_var == "":
            messagebox.showerror("Error", "all fields are required to fill ")
        else:
            con = pymysql.connect(host="localhost", user="root", database="c_patient")
            cur = con.cursor()
            cur.execute("INSERT INTO C_PATIENT VALUES
(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",
                        (self.Patient_var.get(),
                         self.Name_var.get(),
                         self.Test_ID.get(),
                         self.Phone_var.get(),
                         self.Gender_var.get(),
                         self.Doctor_var.get(),
                         self.Room_var.get(),
                         self.Relative_var.get(),
                         self.Age_var.get(),
                         self.txt_address.get('1.0',
                                              END),
                         self.AdmissionDate_var.get(),
                         self.DischargeDate_var.get(),
                         self.Bill_var.get(),
                         self.Death_var.get()
                         ))


            con.commit()
            self.fetch_data()
            self.clear()
            con.close()
            messagebox.showinfo("success", "Recored has been inserted")

    def fetch_data(self):
        con = pymysql.connect(host="localhost", user="root", database="c_patient")
        cur = con.cursor()
        cur.execute("SELECT * FROM C_PATIENT")
        rows = cur.fetchall()
        if len(rows) != 0:
            self.Patient_table.delete(*self.Patient_table.get_children())
            for row in rows:
                self.Patient_table.insert('', END, values=row)
            con.commit()
        con.close()

    def get_cursor(self, ev):
        curosor_row = self.Patient_table.focus()
        contents = self.Patient_table.item(curosor_row)
        row = contents['values']
        self.Patient_var.set(row[0])
        self.Name_var.set(row[1])
        self.Test_ID.set(row[2])
        self.Phone_var.set(row[3])
        self.Gender_var.set(row[4])
        self.Doctor_var.set(row[5])
        self.Room_var.set(row[6])
        self.Relative_var.set(row[7])
        self.Age_var.set(row[8])
        self.txt_address.delete("1.0", END)
        self.txt_address.insert(END, row[9])
        self.AdmissionDate_var.set(row[10])
        self.DischargeDate_var.set(row[11])
        self.Bill_var.set(row[12])
        self.Death_var.set(row[13])

    def clear(self):
        self.Patient_var.set("")
        self.Name_var.set("")
        self.Test_ID.set("")
        self.Phone_var.set("")
        self.Gender_var.set("")
        self.Doctor_var.set("")
        self.Room_var.set("")
        self.Relative_var.set("")
        self.Age_var.set("")
        self.txt_address.delete("1.0", END)
        self.AdmissionDate_var.set("")
        self.DischargeDate_var.set("")
        self.Bill_var.set("")
        self.Death_var.set("")

    def update_data(self):
        con = pymysql.connect(host="localhost", user="root", database="c_patient")
        cur = con.cursor()
        cur.execute(
            "update c_patient set name=%s,
test_id=%s,phone=%s,gender=%s,doctor=%s,room_no=%s,relatives_no=%s,age=%s,"
            "address=%s,admission=%s,discharge=%s, bill_number=%s where patient_id=%s", (
                self.Name_var.get(),
                self.Test_ID.get(),
                self.Phone_var.get(),
                self.Gender_var.get(),
                self.Doctor_var.get(),
                self.Room_var.get(),
                self.Relative_var.get(),
                self.Age_var.get(),
            self.txt_address.get('1.0',
                         END),
            self.AdmissionDate_var.get(),
            self.DischargeDate_var.get(),
            self.Bill_var.get(),
            self.Patient_var.get()))

        con.commit()
        self.fetch_data()
        self.clear()
        con.close()
        messagebox.showinfo("success", "Recored has been updated")

    def delete_data(self):
        con = pymysql.connect(host="localhost", user="root", database="c_patient")
        cur = con.cursor()
        cur.execute("DELETE FROM C_PATIENT WHERE patient_id=%s",
self.Patient_var.get())
        con.commit()
        con.close()
        self.fetch_data()
        self.clear()

    def search_data(self):
        con = pymysql.connect(host="localhost", user="root", database="c_patient")
        cur = con.cursor()
        cur.execute("SELECT * FROM C_PATIENT WHERE " + str(self.Search_by.get()) + "
LIKE"
                                                        " '%" + str(
            self.Search_txt.get()) + "%'")
        rows = cur.fetchall()
        if len(rows) != 0:
            self.Patient_table.delete(*self.Patient_table.get_children())
            for row in rows:
```

```
            self.Patient_table.insert('', END, values=row)
        con.commit()
    con.close()
```

Create Connection

In MySQL, we can use SELECT VERSION() to get the version of MySQL.

```
import pymysql
```

We import the pymysql module.

```
con = pymysql.connect('localhost', 'user7',
    's$cret', 'testdb')
```

We connect to the database with connect. We pass four parameters: the hostname, the MySQL user name, the password, and the database name.

```
with con.cursor() as cur:
```

Using the with keyword, the Python interpreter automatically releases the resources. It also provides error handling. We get a cursor object, which is used to traverse records from the result set.

```
cur.execute('SELECT VERSION()')
```

We call the execute function of the cursor and execute the SQL statement.

```
version = cur.fetchone()
```

The fetchone function fetches the next row of a query result set, returning a single sequence, or None when no more data is available.

```
print(f'Database version: {version[0]}')
```

We print the version of the database.

```
finally:


    con.close()
```

5.2 FRONT END CODE

```
from tkinter import*
from PIL import Image, ImageTk
from tkinter import ttk, messagebox

class patient:
    def __init__(self,root):
        self.root = root
        self.root.title("Login System")
        self.root.geometry("1530x800+0+0")

        #=====BD Image======
        self.bg1 = ImageTk.PhotoImage(file="F:/xj/hq5.jpg")
        bg1 = Label(self.root, image=self.bg1).place(x=0, y=0, relwidth=1, relheight=1)

        #=====Login Frame=====

        Frame_login = Frame(self.root, bg="white")
        Frame_login.place(x=350, y=300, height=340, width=500)

        title = Label(Frame_login,text="Login
Here",font=("Impact",35,"bold"),fg="#d77337",bg="white").place(x=90,y=20)
        desc = Label(Frame_login, text="Accountant Patient login Area", font=("Goudy old
style", 15, "bold"), fg="#d25d17", bg="white").place(x=90,
                                                                     y=90)

        lbl_user = Label(Frame_login, text="Username", font=("Goudy old style", 15,
"bold"),
                fg="gray", bg="white").place(x=90,y=130)
        self.txt_user = Entry(Frame_login,font=("times new roman",15),bg="lightgray")
        self.txt_user.place(x=90,y=160,width=320,height=35)

        lbl_pass = Label(Frame_login, text="Password", font=("Goudy old style", 15,
"bold"),
                  fg="gray", bg="white").place(x=90, y=200)
        self.txt_pass = Entry(Frame_login, font=("times new roman", 15),show="*",
bg="lightgray")
        self.txt_pass.place(x=90, y=230, width=320, height=35)
        login_btn = Button(self.root, text="Login", fg="white", bg="red",
activebackground="red", bd=2,command=self.login_function,cursor="hand2",
                    font=("times new roman", 17)).place(x=510, y=620,width=180,height=40)

    def login_function(self):
        if self.txt_pass.get()=="" or self.txt_user.get()=="":
            messagebox.showerror("Error", "All Fields are required", parent=self.root)
        elif self.txt_pass.get()!="123" or self.txt_user.get()!="123":
            messagebox.showerror("Error", "Invalid Username/Password", parent=self.root)
        else:
            messagebox.showinfo("", "WELCOME",parent=self.root)
            #messagebox.showinfo("Welcome", f"Welcome {self.txt_user.get()}\nYour
Password: {self.txt_pass.get()}", parent=self.root)
            self.root.destroy()
            import art

root = Tk()
obj = patient(root)
root.mainloop()
```

CHAPTER 6

TESTING

This chapter gives the outline of all testing methods that are carried out to get a bug free system. Quality can be achieved by testing the product using different techniques at different phases of the project development. The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components sub assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 TESTING PROCESS

Testing is an integral part of software development. Testing process certifies whether the product that is developed compiles with the standards that it was designed to. Testing process involves building of test cases against which the product has to be tested.

6.2 TESTING OBJECTIVES

The main objectives of testing process are as follows.

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has high probability of finding undiscovered error.
- A successful test is one that uncovers the undiscovered error.

6.3 TEST CASES

The test cases provided here test the most important features of the project.

6.3 TEST CASES

The test cases provided here test the most important features of the project.

6.3.1 Test cases for the project

Table 6.1 ------- Test Case

Sl No	**Test Input**	**Expected Results**	**Observed Results**	**Remarks**
1	Insert a record	New tuple should be inserted.	New record inserted	PASS
2	Search a record	Should display the searched record	Searched record found	PASS
3	Search a record	Should display the searched record	Searched record not found	FAIL
4	Update a record	Update the record	Record Updated	PASS
5	Clear a record	Clear the record	Clear	PASS
6	Delete a record	Delete the record	Record not found	FAIL
7	Delete a record	Delete the record	Selected record deleted	PASS

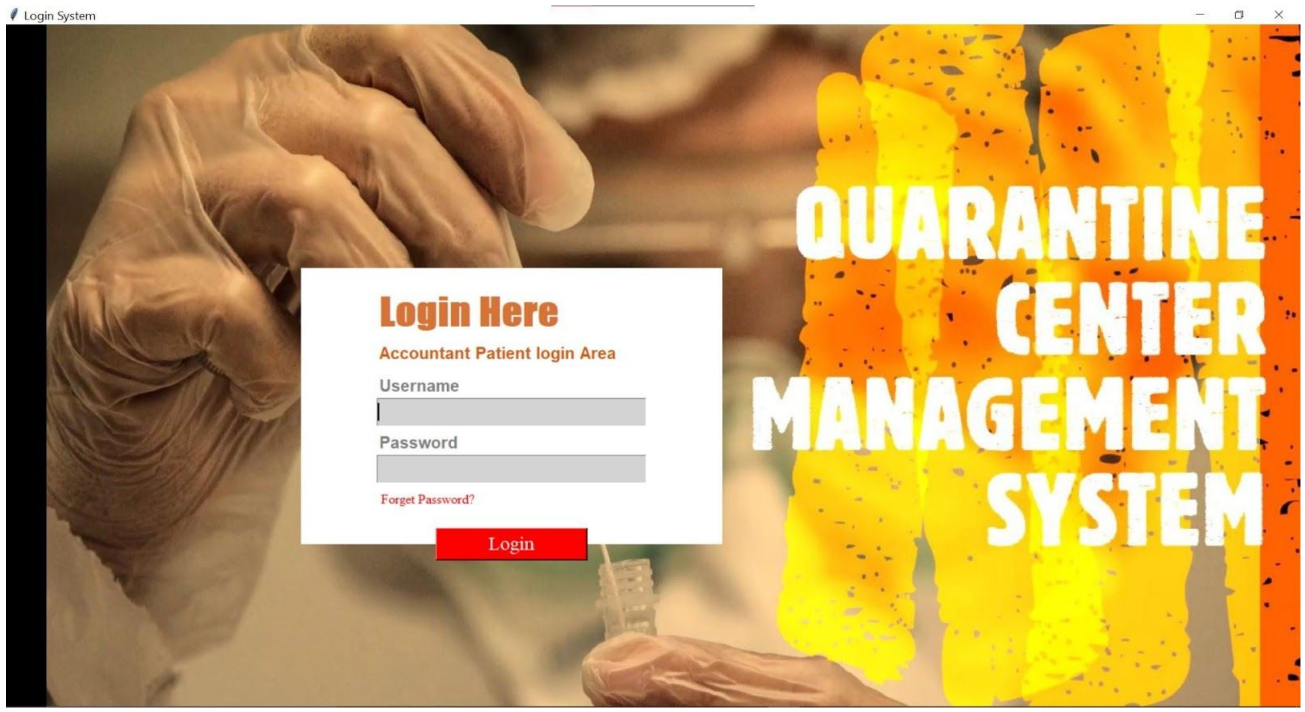
CHAPTER 7

RESULTS

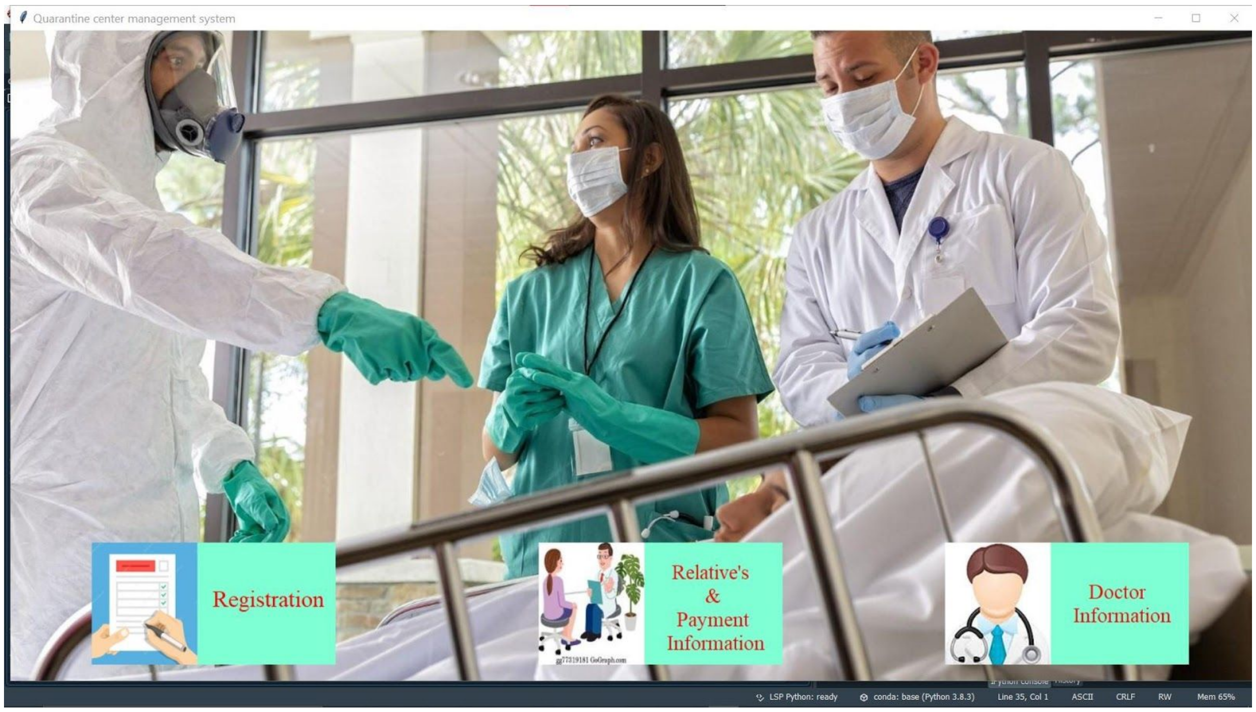
This section describes the screens of the “Criminal Database Management”. The snapshots are shown below for each module.

7.1 SNAPSHOT

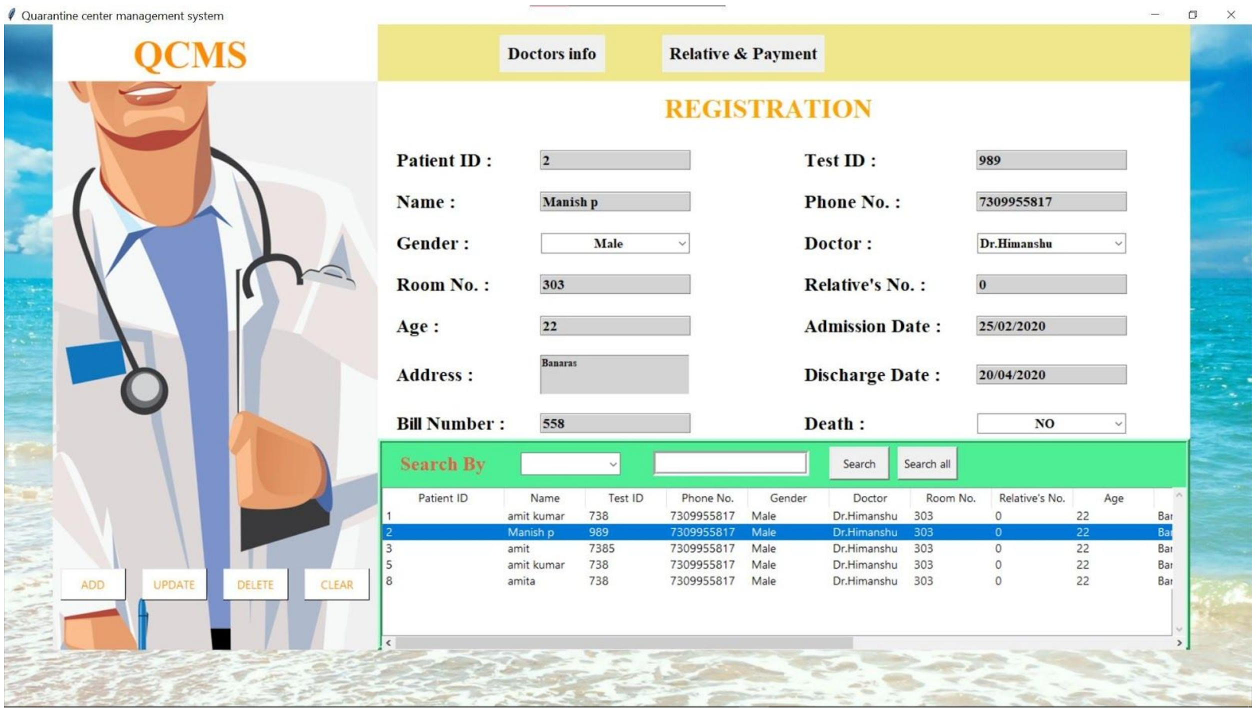
Snapshot1:Login_Page


Login System
QUARANTINE CENTER MANAGEMENT SYSTEM
Login Here
Accountant Patient login Area
Username
Password
Forget Password?
Login

Snapshot2:Home_Page


Quarantine center management system
Registration
Relative's & Payment Information
Doctor Information
LSP Python: ready
conda: base (Python 3.8.3)
Line 35, Col 1
ASCII
CRLF
RW
Mem 65%

Snapshot 3: Registration

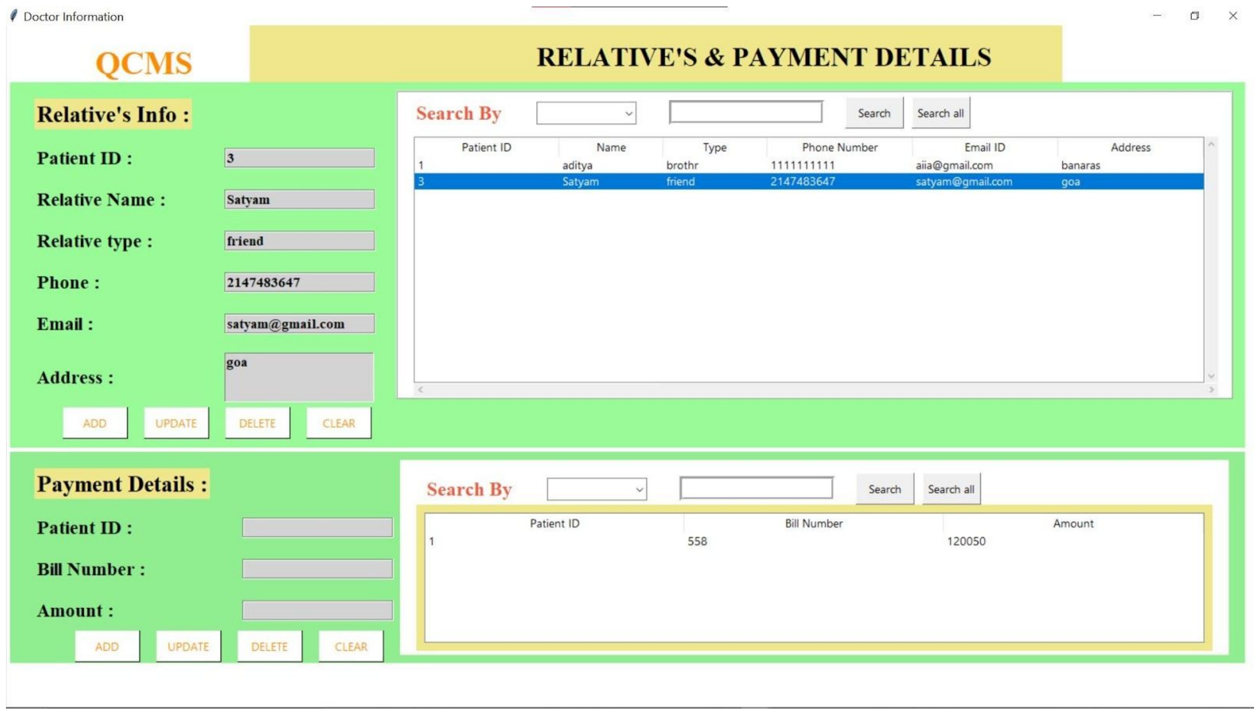


REGISTRATION

Patient ID :	2	Test ID :	989
Name :	Manish p	Phone No. :	7309955817
Gender :	Male	Doctor :	Dr.Himanshu
Room No. :	303	Relative's No. :	0
Age :	22	Admission Date :	25/02/2020
Address :	Banaras	Discharge Date :	20/04/2020
Bill Number :	558	Death :	NO

Patient ID	Name	Test ID	Phone No.	Gender	Doctor	Room No.	Relative's No.	Age	
1	amit kumar	738	7309955817	Male	Dr.Himanshu	303	0	22	Bar
2	Manish p	989	7309955817	Male	Dr.Himanshu	303	0	22	Bar
3	amit	7385	7309955817	Male	Dr.Himanshu	303	0	22	Bar
5	amit kumar	738	7309955817	Male	Dr.Himanshu	303	0	22	Bar
8	amita	738	7309955817	Male	Dr.Himanshu	303	0	22	Bar

Snapshot 4: Relative's and Payment Details


Doctor Information

QCMS

RELATIVE'S & PAYMENT DETAILS

Relative's Info :

Patient ID :	3
Relative Name :	Satyam
Relative type :	friend
Phone :	2147483647
Email :	satyam@gmail.com
Address :	goa

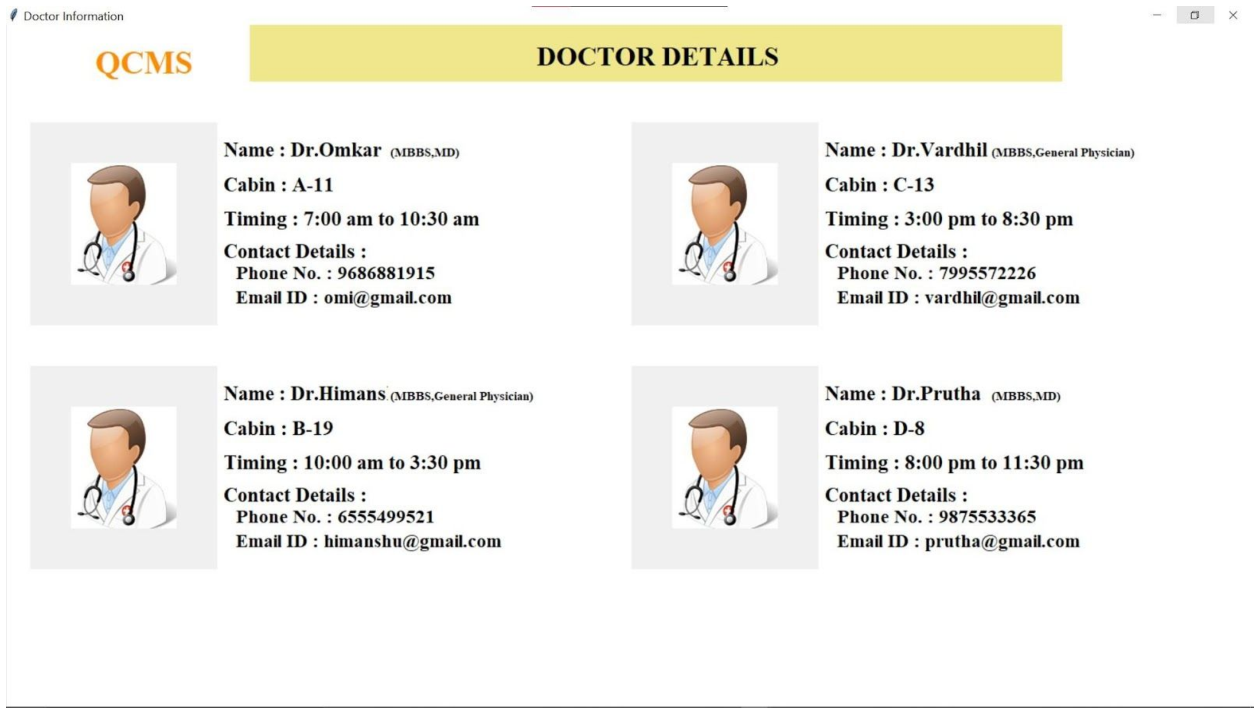
Patient ID	Name	Type	Phone Number	Email ID	Address
1	aditya	brothr	1111111111	aiia@gmail.com	banaras
3	Satyam	friend	2147483647	satyam@gmail.com	goa

Payment Details :

Patient ID :	
Bill Number :	
Amount :	

Patient ID	Bill Number	Amount
1	558	120050

Snapshot 5: Doctor Details


Doctor Information

QCMS

DOCTOR DETAILS

Name : Dr.Omkar (MBBS,MD)
Cabin : A-11
Timing : 7:00 am to 10:30 am
Contact Details :
Phone No. : 9686881915
Email ID : omi@gmail.com

Name : Dr.Vardhil (MBBS,General Physician)
Cabin : C-13
Timing : 3:00 pm to 8:30 pm
Contact Details :
Phone No. : 7995572226
Email ID : vardhil@gmail.com

Name : Dr.Himans (MBBS,General Physician)
Cabin : B-19
Timing : 10:00 am to 3:30 pm
Contact Details :
Phone No. : 6555499521
Email ID : himanshu@gmail.com

Name : Dr.Prutha (MBBS,MD)
Cabin : D-8
Timing : 8:00 pm to 11:30 pm
Contact Details :
Phone No. : 9875533365
Email ID : prutha@gmail.com

CONCLUSION

The development of this Quarantine Center Management Systemt application is great improvement in accessing and holding the data comparing to manual work with so many paper. The computerization of the system speeds up the process.

The Quarantine Center Management System is fast, efficient and reliable, Avoids data redundancy and inconsistency. It contains all the functional features described in objective of the project.

REFERENCES

[1] Stack Overflow
[2] W3 Schools
[3] Codemy.com & Webcode on YouTube
[4] Python 4 Everybody Book