

PROFESSIONAL TRAINING REPORT

at

**Sathyabama Institute of Science and Technology
(DEEMED TO BE UNIVERSITY)**

Submitted in partial fulfillment of the requirements for the award of Bachelor of
Technology Degree in
Information technology

By

VASAN S K

Reg. No. 38120089



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY JEPPIAAR NAGAR,
RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU**

AUGUST 2020



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, Chennai - 600119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the Bonafide work of **VASAN S K (Reg No. 38120089)** who underwent the professional training in **"HOSPITAL MANAGEMENT SYSTEM"** under our supervision from Jan 2020 to June 2020

Internal Guide

M.S Indira K

Head of the Department

Dr. R. Subhashini M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **“VASAN S K”** hereby declare that the professional Training Report on **“HOSPITAL MANAGEMENT SYSTEM”** done by me under the guidance of **M.S INDIRA K** at Sathyabama Institute of Science and technology is submitted in partial fulfillment of the requirements of the award of Bachelor of Technology degree in Information technology.

DATE:21/01/2021

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

We convey our thanks to **Dr. T. SASIKALA M.E., Ph.D., Dean of the Department, SCHOOL OF COMPUTING** for providing us the necessary support and details at the right time during the progressive reviews.

We convey our thanks to **Dr. R. SUBHASHINI., M.E., Ph.D., Head of the Department, Department of INFORMATION TECHNOLOGY** for providing us the necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **M.S INDIRA K** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of **INFORMATION TECHNOLOGY** who were helpful in many ways for the completion of the project.

INTERNSHIP CERTIFICATE



ISO 9001:2015 CERTIFIED

CERTIFICATE OF MERIT

This is to certify that VASAN .S.K (Reg No. 38120089) has successfully completed the internship in PYTHON Application Development in our concern from 21-12-19 to 25-12-19.

During the internship period, the performance of the intern was found to be GOOD.

Program Coordinator



HR Head

COURSE CERTIFICATE



23-Jan-2021

vasan s k

has successfully completed

Programming for Everybody (Getting Started with Python)

an online non-credit course authorized by University of Michigan and offered through Coursera

A handwritten signature in black ink, appearing to read 'Charles', followed by a horizontal line.

Charles Severance
Clinical Professor, School of Information
University of Michigan

COURSE
CERTIFICATE



Verify at coursera.org/verify/7XNBYWTLRX3D

Coursera has confirmed the identity of this individual and their participation in the course.

ABSTRACT

The main purpose of this project was to allow the users to access the HOSPITAL MANAGEMENT application. The project involves PYTHON GRAPHICAL USER INTERFACE and SQLITE3. The libraries involved in this project are TKinter and Pyinstaller. GUI is used to create an interface required to create the interface for the application. SQLite is used to store the appointment details in Database. The general queries used in the project for sqlite database are View all, Delete, Update, Insert TKinter is used to create front end and sqlite is used to create backend. These two are combined and made as an efficient application for Hospital Management. Pyinstaller is used to make the application as a standalone environment to work in any conditions. The application is used for booking appointment and also to update them. The application is also used to displaying the patients one by one. Here we used text to speech. The Hospital Management System can be entered using a username and password, It is accessible either by an administrator or receptionist. Only they can add data into the database. The data can be retrived easily. The data are well protected for personal use and makes the data processing very fast.

TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	ABSTRACT	vi
	1 INTRODUCTION	
	1.1 DOMAIN INTRODUCTION	1
	1.2 GOALS OF PROJECT	2
	1.3 SQLITE DATABASE	2
	1.4 PYTHON FEATURES	
	1.5 PYTHON APPLICATION	3
	2 AIM AND SCOPE OF THE PROJECT	
	2.1 AIM	4
	2.2 SOFTWARE ENVIRONMENT	4
	2.3 HARDWARE ENVIRONMENT	4
	3 SYSTEM SPECIFICATION	
	3.1 BRIEF REQUIREMENTS OF PROJECT	5
	3.2 PYTHON TKINTER GEOMETRY	5
	3.3 TKINTER WIDGETS	5
	3.4 SQLITE DATABASE	9
	4 MODULES	
	4.1 ADDING APPOINTMENTS TO DB	11
	4.2 DISPLAYING APPOINTMENTS	11
	4.3 UPDATING THE DETAILS	12
	4.4 REQUIREMENTS FOR HMS	13
	5 CONCLUSION	14
	REFERENCES	
	APPENDIX	
	A.SOURCE CODE	15
	B.SCREEN SHOTS	22

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
4.1	ARCCHITECTURE OF HOSPITAL MANAGEMENT	11
5.1	INSERTION OF APPOINTMENTS	22
5.2	APPOINTMENTS ADDED TO DATABASE	23
5.3	CREATION OF TABLE IN DATABASE	24
5.4	CREATED TABLE IN DATABASE	25
5.5	DISPLAYING APPOINTMENT	26
5.6	UPDATING AND DELETION OF APPOINTMENT	27

LIST OF TABLES

TABLE NO	TITLE	PAGE NO
3.1	LIST OF WIDGETS	6
3.2	OPTIONS OF BUTTON	7
3.3	OPTIONS OF LABEL	8
3.4	DATABASE TABLE	10

LIST OF ABBRIVATIONS

GUI	GRAPHICAL USER INTERFACE
BG	BACKGROUND COLOR
FG	FOREGROUND COLOR
PADx	PADDING BUTTON IN HORIZONTAL
PADy	PADDING BUTTON IN VERTICAL

CHAPTER 1

INTRODUCTION

1.1 DOMAIN INTRODUCTION:

Python is a simple, general purpose, high level, and object-oriented programming language. Python is an interpreted scripting language also. Guido Van Rossum is known as the founder of Python programming.

Python is a general purpose, dynamic, high level, and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures. Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development. Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development. Python supports multiple programming pattern, including object- oriented, imperative, and functional or procedural programming styles.

1.2 GOALS OF PROJECT

The main objective is to upload Hospital Management in database. A database is a collection of information and is systematically stored in tables in the form of rows and columns. The table in the database has unique name that identifies its contents. The database in turn is further described in detail giving all the fields used with the data types, constraints available, primary key and foreign key. Sqlite Module is used to link python to Database.

1.3 SQLITE DATABASE

- Sqlite3 is a very easy to use database engine.
- It is self-contained, serverless, zero-configuration and transactional.
- It is very fast and lightweight, and the entire database is stored in a single disk file.
- It is used in a lot of applications as internal data storage. the python standard library includes a module called "sqlite3" intended for working with this database.
- This module is a sql interface compliant with the db-api 2.0 specification.

1.4 PYTHON FEATURES:

- Easy to learn
- Interpreter language
- Object oriented language
- Extensible
- Large standard library
- GUI programming support
- Integrated

1.5 PYTHON APPLICATION:

Python is known for its general purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development.

Here, we are specifying applications areas where python can be applied. 1) Web

Application:

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautiful Soup, Feed parser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and develop web based applications. Some important developments are: Python Wiki Engines, Pocoo, Python Blog Software etc.

2) Desktop GUI Applications:

Python provides Tk GUI library to develop user interface in python based application.

Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

3) Software Development:

Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.

4) Scientific and Numeric:

Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.

CHAPTER 2

AIM AND SCOPE OF THE PROJECT

2.1 AIM :

The main objective of the project is to design and develop Hospital Management by GUI using python tkinter module and Sqlite Database.

2.2 SOFTWARE ENVIRONMENT :

The software requirements are the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

Operating System	: Windows XP/2003 and above User
Interface	: GUI
Programming Language	: Python3 with tkinter

2.3 HARDWARE ENVIRONMENT :

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system.

Hardware	: Intel i3
RAM	: 256MB(minimum)

CHAPTER 3 SYSTEM

SPECIFICATION

3.1 BRIEF REQUIREMENTS OF PROJECT :

Python Tkinter:

Developing desktop based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.

1. Import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.
4. Call the main event loop so that the actions can take place on the user's computer screen

3.2 PYTHON TKINTER GEOMETRY:

The Tkinter geometry specifies the method by using which, the widgets are represented on display. The python Tkinter provides the following geometry methods.

1. The pack () method
2. The grid () method
3. The place () method

3.3 TKINTER WIDGETS:

There are various widgets like button, canvas, checkbutton, entry, etc. that are used to build the python GUI applications.

S.No	Widgets	Description
1	Button	The Button is used to add various kinds of buttons to the python application.
2	Checkbutton	The Checkbutton is used to display the CheckButton on the window.

3	Entry	The entry widget is used to display the single-line text field to the user. It is commonly used to accept user values.
4	Frame	It can be defined as a container to which, another widget can be added and organized.
5	Label	A label is a text used to display some message or information about the other widgets.
6	ListBox	The ListBox widget is used to display a list of options to the user.
7	Menu	It is used to add menu items to the user.
8	Message	The Message widget is used to display the message-box to the user.
9	Radiobutton	The Radiobutton is different from a checkbutton. Here, the user is provided with various options and the user can select only one option among them.
10	Scrollbar	It provides the scrollbar to the user so that the user can scroll the window up and down.
11	Text	It is different from Entry because it provides a multi-line text field to the user so that the user can write the text and edit the text inside it.
12	PanedWindow	It is like a container widget that contains horizontal or vertical panes.
13	LabelFrame	A LabelFrame is a container widget that acts as the container.
14	MessageBox	This module is used to display the message-box in the desktop based applications.

Table 3.1 List of Widgets

Python Tkinter button:

We can also associate a method or function with a button which is called when the button is pressed.

Syntax of Button:

W = Button(parent, options)

S.No	Option	Description
1	Activebackground	It represents the background of the button when the mouse hover the button.
2	Activeforeground	It represents the font color of the button when the mouse hover the button.
3	Bd	It represents the border width in pixels.
4	Bg	It represents the background color of the button.
5	Command	It is set to the function call which is scheduled when the function is called.
6	Fg	Foreground color of the button.
7	Font	The font of the button text.
8	Highlightcolor	The color of the highlight when the button has the focus.
9	Padx	Additional padding to the button in the horizontal direction.
10	Pady	Additional padding to the button in the vertical direction.

Table 3.2 Options of Button

3.3.2 ENTRY:

The Entry widget is used to provide the single line text-box to the user to accept a value from the user. We can use the Entry widget to accept the text strings from the user. It can only be used for one line of text from the user. For multiple lines of text, we must use the text widget.

Syntax of Entry:

```
w = Entry (parent, options)
```

3.3.3 PYTHON TKINTER LABEL:

The Label is used to specify the container box where we can place the text or images. This widget is used to provide the message to the user about other widgets used in the python application.

There are the various options which can be specified to configure the text or the part of the text shown in the Label.

Syntax of Label:

```
W=Label(master,options)
```

S.No	Option	Description
1	anchor	It specifies the exact position of the text within the size provided to the widget. The default value is CENTER, which is used to center the text within the specified space.
2	Bg	The background color displayed behind the widget.
3	bitmap	It is used to set the bitmap to the graphical object specified so that, the label can represent the graphics instead of text.
4	Bd	It represents the width of the border. The default is 2 pixels.
5	cursor	The mouse pointer will be changed to the type of the cursor specified, i.e., arrow, dot, etc.
6	Font	The font type of the text written inside the widget.
7	Fg	The foreground color of the text written inside the widget.
9	Image	The image that is to be shown as the label.

Table 3.3 Options of Label

3.3.4 Tkinter MessageBox:

The messagebox module is used to display the message boxes in the python applications. There are the various functions which are used to display the relevant

messages depending upon the application requirements. The

syntax to use the messagebox is given below.

Syntax

1. `messagebox.function_name(title, message [options])`

Parameters

- **Function_name:** It represents an appropriate message box function.
- **Title:** It is a string which is shown as a title of a message box.
- **Message:** It is the string to be displayed as a message on the message box.
- **Options:** There are various options which can be used to configure the message dialog box.

2. parent

The parent option specifies the parent window on top of which, the message box is to be displayed.

There is one of the following functions used to show the appropriate message boxes. All the functions are used with the same syntax but have the specific functionalities.

1. `showinfo()`

The `showinfo()` messagebox is used where we need to show some relevant information to the user.

3.4 SQLITE DATABASE :

Database design is used to manage large bodies of information. In this database we describe all the 4 tables available in the software, which are used to store all the records.

Data types and its description Fields in database table have a data type. Some of the data types used in database table are explained below.

a) Integer:

One optional sign character (+ or -) followed by atleast one digit (0-9). Leading and trailing blanks are ignored. No other character is allowed.

b) Varchar:

It is used to store alpha numeric characters. In this data type we can set the maximum number of characters up to 8000 ranges by default SQL server will set the size to 50 characters large.

c) Date/Time:

Date/Time data type is used for representing data or time.

Fields	Data Type	Relationships
Pid	Varchar(5)	Primary Key
name	Varchar(20)	Not Null
age	Int	Not Null
gender	Varchar(10)	Not null
address	Varchar(50)	Not Null
Phone no	Int	Not Null

Table 3.4 Database Table

CHAPTER 4

MODULES

4.1 ADDING APPOINTMENTS TO DATABASE

- Insert the Attributes of the person in database table and set the Id as the Primary key Attribute and set Datatype of each attribute.
- By Linking Sqlite to Python we can add appointments to the Database which can be further used for retrieval, update and display details of appointments.

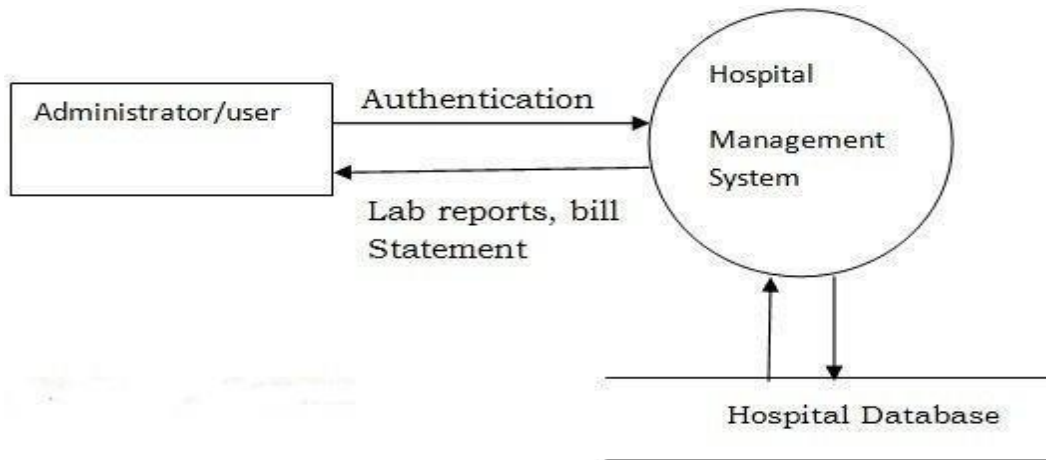


Fig 4.1 ARCHITECTURE OF HOSPITAL MANAGEMENT

Operation performed by Connecting Sqlite to Python:

- creating (CREATE) and deleting (DROP) tables.
- inserting (INSERT) data into the database.
- retrieving data (SELECT) with sqlite.
- updating (UPDATE) and deleting (DELETE) data.

4.2 DISPLAYING APPOINTMENTS

- We can also display the appointments in order of their Id's.
- On Compiling the display.py file we can obtain the appointments stored in the SQLite Database.

Hospital management system will help doctors to devote more time to patients, not documentation.

We all know, that real pro will prefer a clinic, in which:

- All the necessary information about the patient is collected in one place. The document will be able to see the history of the disease and quickly obtain the test results. And the more information the doctor has, the more accurately he can make a diagnosis and prescribe the right treatment.
- A database of document templates is created. It will help the doctor or nurse give a medical protocol to the patient, add information to the patient card. It is useful for doctors as well as for the administration. Duration of an appointment can be shortened without compromising quality that will increase clinic's revenue for sure.
- Doctor can look at the schedule at any time and make changes if necessary.

Hospital authorities must take care of the profitability of the facility. The computer software makes it possible to:

- Analyze doctors' work and evaluate how many hours did each of them work.
- Optimize hospital peak hours, reduce medical equipment downtime.
- Find out which medical directions generate revenue, and which should be closed or modified.
- Evaluate the effectiveness of campaigns to attract visitors.

4.3 UPDATING THE DETAILS

- We can also update the details of appointments by running update.py file
- Searching the name of the patient and then the data is obtained.
- Then we can modify the required Attributes or can entirely delete the data of person from database.

4.4 Requirements for Hospital Management Software

Let's look at the basic requirements for hospital management software, which the developers must take into account:

- Any employee of the clinic must be able to cope with the clinic management system.
- The user interface must be easy, convenient and informative.
- There must be a certain role-based access for each employee according to the powers and responsibilities. Hospital authority must be able to edit the access role and passwords of the employees. It will allow to control all confidential information.

- Hospital software must provide accurate, necessary and relevant information at any time.
- As we are talking about customers' personal data, the maximum level of security and data protection must be provided in the system
- There must be a possibility to improve or extend the clinic automation.

CHAPTER 5

CONCLUSION

Thus the project infer the Hospital management Using Python and Sqlite Database a desktop application which is developed in python platform. It is developed using Python tkinter. Python provides the standard library tkinter for creating the Graphical User Interface for desktop applications. Developing desktop based application with Python tkinter is not a complex task. While Sqlite database is used to store the data in the form of tables by which we can access them when required. We can also perform operations like Insertion of new appointments, deletion of appointments, update appointmets.

APPENDIX

A) SOURCE CODE

```
//appointments from
tkinter import * import
sqlite3
import tkinter.messagebox
conn = sqlite3.connect('database.db') c =
conn.cursor()
ids = []
class Application:
    def __init__(self, master): self.master =
        master
        self.left = Frame(master, width=800, height=720, bg='lightgreen')
        self.left.pack(side=LEFT)
        self.right = Frame(master, width=400, height=720, bg='steelblue')
        self.right.pack(side=RIGHT)
        self.heading = Label(self.left, text="ABC Hospital Appointments", font=('arial 40 bold'), fg='black', bg='lightgreen')
        self.heading.place(x=0, y=0)
        self.name = Label(self.left, text="Patient's Name", font=('arial 18 bold'), fg='black', bg='lightgreen')
        self.name.place(x=0, y=100)
        self.age = Label(self.left, text="Age", font=('arial 18 bold'), fg='black', bg='lightgreen')
        self.age.place(x=0, y=140)
        self.gender = Label(self.left, text="Gender", font=('arial 18 bold'), fg='black',
bg='lightgreen')
        self.gender.place(x=0, y=180)
        self.location = Label(self.left, text="Location", font=('arial 18 bold'), fg='black',
bg='lightgreen')
        self.location.place(x=0, y=220)
        self.time = Label(self.left, text="Appointment Time", font=('arial 18 bold'), fg='black', bg='lightgreen')
        self.time.place(x=0, y=260)
```

```

self.phone = Label(self.left, text="Phone Number", font=('arial 18 bold'), fg='black', bg='lightgreen')
self.phone.place(x=0, y=300) self.name_ent =
Entry(self.left, width=30)
self.name_ent.place(x=250, y=100) self.age_ent =
Entry(self.left, width=30) self.age_ent.place(x=250,
y=140) self.gender_ent = Entry(self.left, width=30)
self.gender_ent.place(x=250, y=180)
self.location_ent = Entry(self.left, width=30)
self.location_ent.place(x=250, y=220) self.time_ent
= Entry(self.left, width=30)
self.time_ent.place(x=250, y=260) self.phone_ent =
Entry(self.left, width=30)
self.phone_ent.place(x=250, y=300)
self.submit = Button(self.left, text="Add Appointment", width=20, height=2, bg='steelblue',
command=self.add_appointment)
self.submit.place(x=300, y=340) def
add_appointment(self):
    self.val1 = self.name_ent.get()
    self.val2 = self.age_ent.get() self.val3
    = self.gender_ent.get() self.val4 =
    self.location_ent.get() self.val5 =
    self.time_ent.get() self.val6 =
    self.phone_ent.get()
    if self.val1 == " or self.val2 == " or self.val3 == " or self.val4 == " or self.val5 == ":
        tkinter.messagebox.showinfo("Warning", "Please Fill Up All Boxes")
    else:
        sql = "INSERT INTO 'appointments' (name, age, gender, location, scheduled_time, phone) VALUES(?,
?, ?, ?, ?, ?)"
        c.execute(sql, (self.val1, self.val2, self.val3, self.val4, self.val5, self.val6))
        conn.commit()
        tkinter.messagebox.showinfo("Success", "Appointment for " +str(self.val1) + " has been created" )

```

```

        self.box.insert(END, 'Appointment fixed for ' + str(self.val1) + ' at ' + str(self.val5)) sql2= "SELECT
        id FROM appointments "
self.result = c.execute(sql2) for
self.row in self.result:
            self.id = self.row[0]
            ids.append(self.id)
        self.new = sorted(ids)
self.final_id = self.new[len(ids)-1]
self.logs = Label(self.right, text="Logs", font=('arial 28 bold'), fg='white', bg='steelblue')
self.logs.place(x=0, y=0)
self.box = Text(self.right, width=50, height=40)
self.box.place(x=20, y=60)
self.box.insert(END, "Total Appointments till now : " + str(self.final_id)) root=Tk()
ob=Application(root)
root.geometry("1200x720+0+0")
root.resizable(False, False)
root.mainloop()
#displaying appointments
from tkinter import *
import sqlite3 import
pytsx3
conn = sqlite3.connect('database.db') c =
conn.cursor()
number = []
patients = []
sql = "SELECT * FROM appointments" res =
c.execute(sql)
for r in res:
    ids = r[0]
    name = r[1]
    number.append(ids)
patients.append(name)
class Application:

```

```

def _init_(self, master):
    self.master = master
    self.x = 0
    self.heading = Label(master, text="Appointments", font=('arial 60 bold'), fg='green')
    self.change = Button(master, text="Next Patient", width=25, height=2, bg='steelblue',
command=self.func)
    self.change.place(x=500, y=600)
    self.n = Label(master, text="", font=('arial 200 bold'))
    self.n.place(x=500, y=100)
    self.pname = Label(master, text="", font=('arial 80 bold'))
    self.pname.place(x=300, y=400)
def func(self):
    self.n.config(text=str(number[self.x]))
    self.pname.config(text=str(patients[self.x]))
    voices = engine.getProperty('voices')
    rate = engine.getProperty('rate')
    engine.setProperty('rate', rate-50)
    engine.say('Patient number ' + str(number[self.x]) + str(patients[self.x]))
    self.x += 1
root = Tk()
b = Application(root)
root.geometry("1366x768+0+0")
root.resizable(False, False)
root.mainloop()
# update the appointments
from tkinter import *
import tkinter.messagebox
import sqlite3
conn = sqlite3.connect('database.db')
c = conn.cursor()
class Application:
    def _init_(self, master):
        self.master = master
        self.heading = Label(master, text="Update Appointments", fg='steelblue', font=('arial 40

```

```

bold'))
    self.name = Label(master, text="Enter Patient's Name", font=('arial 18 bold'))
    self.name.place(x=0, y=60)
    self.namenet = Entry(master, width=30)
    self.namenet.place(x=280, y=62)
    self.search = Button(master, text="Search", width=12, height=1, bg='steelblue',
command=self.search_db)
    def search_db(self):
        self.input = self.namenet.get()
        sql = "SELECT * FROM appointments WHERE name LIKE ?" self.res =
        c.execute(sql, (self.input,))
        for self.row in self.res:
            self.name1 = self.row[1]
            self.gender = self.row[3]
            self.location = self.row[4]
            self.time = self.row[6]
            self.phone = self.row[5]
        self.uname = Label(self.master, text="Patient's Name", font=('arial 18 bold')) self.uage =
        Label(self.master, text="Age", font=('arial 18 bold'))
        self.ugender = Label(self.master, text="Gender", font=('arial 18 bold'))
        self.ugender.place(x=0, y=220)
        self.ulocation = Label(self.master, text="Location", font=('arial 18 bold'))
        self.ulocation.place(x=0, y=260)
        self.untime = Label(self.master, text="Appointment Time", font=('arial 18 bold')) self.uphone =
        Label(self.master, text="Phone Number", font=('arial 18 bold')) self.uphone.place(x=0, y=340)
        self.ent1 = Entry(self.master, width=30)
        self.ent1.place(x=300, y=140)
        self.ent1.insert(END, str(self.name1)) self.ent2
        = Entry(self.master, width=30)
        self.ent2.place(x=300, y=180)
        self.ent2.insert(END, str(self.age)) self.ent3 =
        Entry(self.master, width=30)
        self.ent3.place(x=300, y=220)

```

```

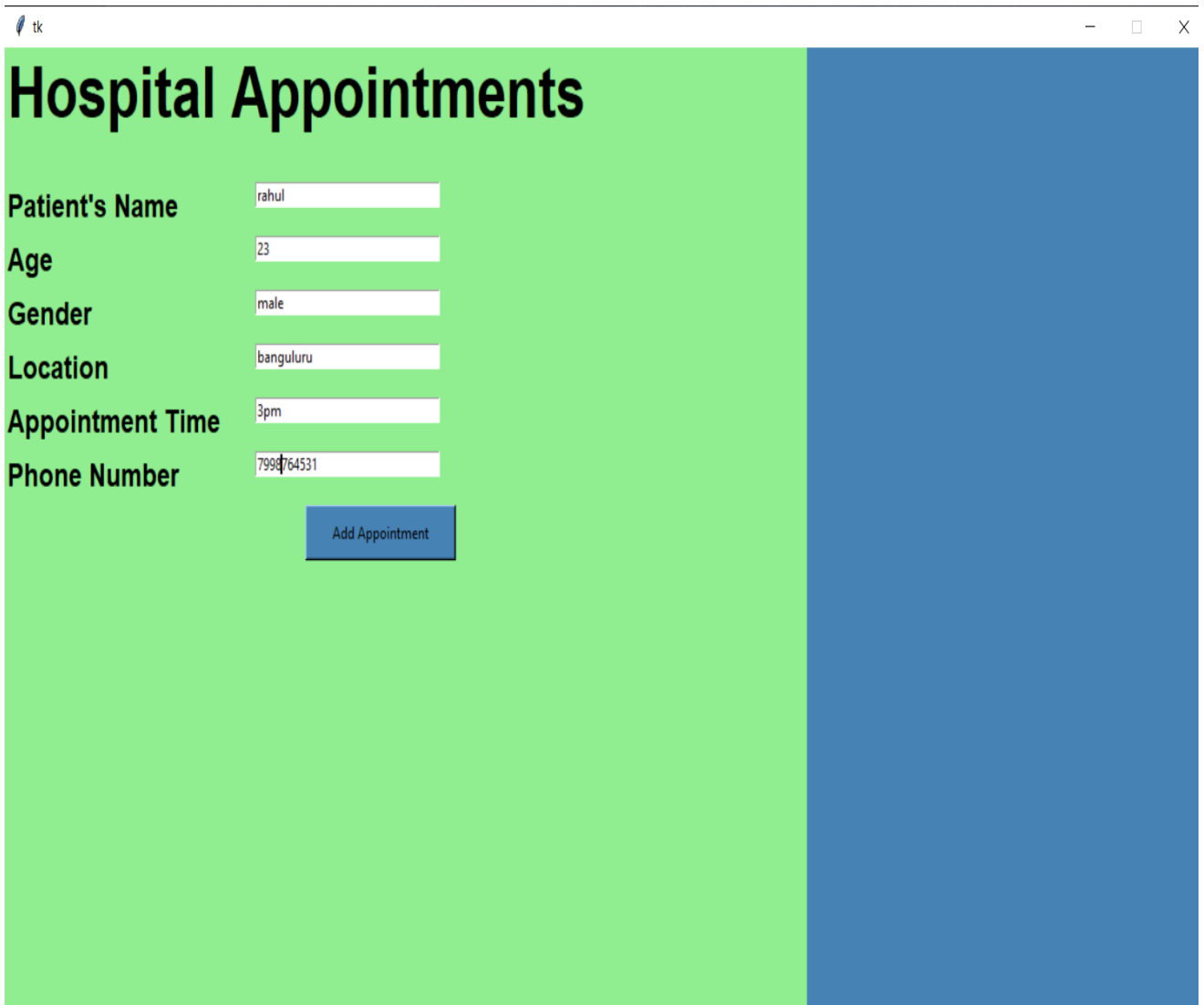
self.ent3.insert(END, str(self.gender)) self.ent4
= Entry(self.master, width=30)
self.ent4.place(x=300, y=260)
self.ent4.insert(END, str(self.location)) self.ent5
= Entry(self.master, width=30)
self.ent5.insert(END, str(self.time)) self.ent6 =
Entry(self.master, width=30)
self.ent6.place(x=300, y=340)
self.ent6.insert(END, str(self.phone))
self.update = Button(self.master, text="Update", width=20, height=2, bg='lightblue', command=self.update_db)
self.update.place(x=400, y=380)
self.delete = Button(self.master, text="Delete", width=20, height=2, bg='red',
command=self.delete_db)
self.delete.place(x=150, y=380) def
update_db(self):
    self.var1 = self.ent1.get()
    self.var2 = self.ent2.get()
    self.var3 = self.ent3.get()
    self.var4 = self.ent4.get()
    self.var5 = self.ent5.get()
    self.var6 = self.ent6.get()
    query = "UPDATE appointments SET name=?, age=?, gender=?, location=?, phone=?, scheduled_time=?
WHERE name LIKE ?"
    c.execute(query, (self.var1, self.var2, self.var3, self.var4, self.var5, self.var6,
self.namenet.get()))
    conn.commit()
    tkinter.messagebox.showinfo("Updated", "Successfully Updated.") def
delete_db(self):
    sql2 = "DELETE FROM appointments WHERE name LIKE ?"
    c.execute(sql2, (self.namenet.get(),))
    conn.commit()
    tkinter.messagebox.showinfo("Success", "Deleted Successfully")
    self.ent1.destroy()

```

```
        self.ent2.destroy()
        self.ent3.destroy()
        self.ent4.destroy()
        self.ent5.destroy()
        self.ent6.destroy()

root = Tk()
b = Application(root)
root.geometry("1200x720+0+0")
root.resizable(False, False)
root.mainloop()
```

B) SCREEN SHOTS



The screenshot shows a Tkinter window with a title bar containing a feather icon, the text 'tk', and standard window controls (minimize, maximize, close). The window has a light green background on the left and a blue background on the right. The title 'Hospital Appointments' is displayed in large black font at the top left. Below the title, there are six labels with corresponding text input fields: 'Patient's Name' (value: rahul), 'Age' (value: 23), 'Gender' (value: male), 'Location' (value: banguluru), 'Appointment Time' (value: 3pm), and 'Phone Number' (value: 7998764531). A blue button labeled 'Add Appointment' is positioned below the input fields.

Field	Value
Patient's Name	rahul
Age	23
Gender	male
Location	banguluru
Appointment Time	3pm
Phone Number	7998764531

Fig 5.1 Insertion of Appointments

In this above Fig 5.1 we can see that the administrator was taking the appointment which contains Patient's name, age, Gender, Location, Appointment time, and finally Phone number. This data can be stored in the SQLite studios as database. It is nothing but fixing an appointment to a Patient. Hospital management system will help doctors to devote more time to patients, not documentation.

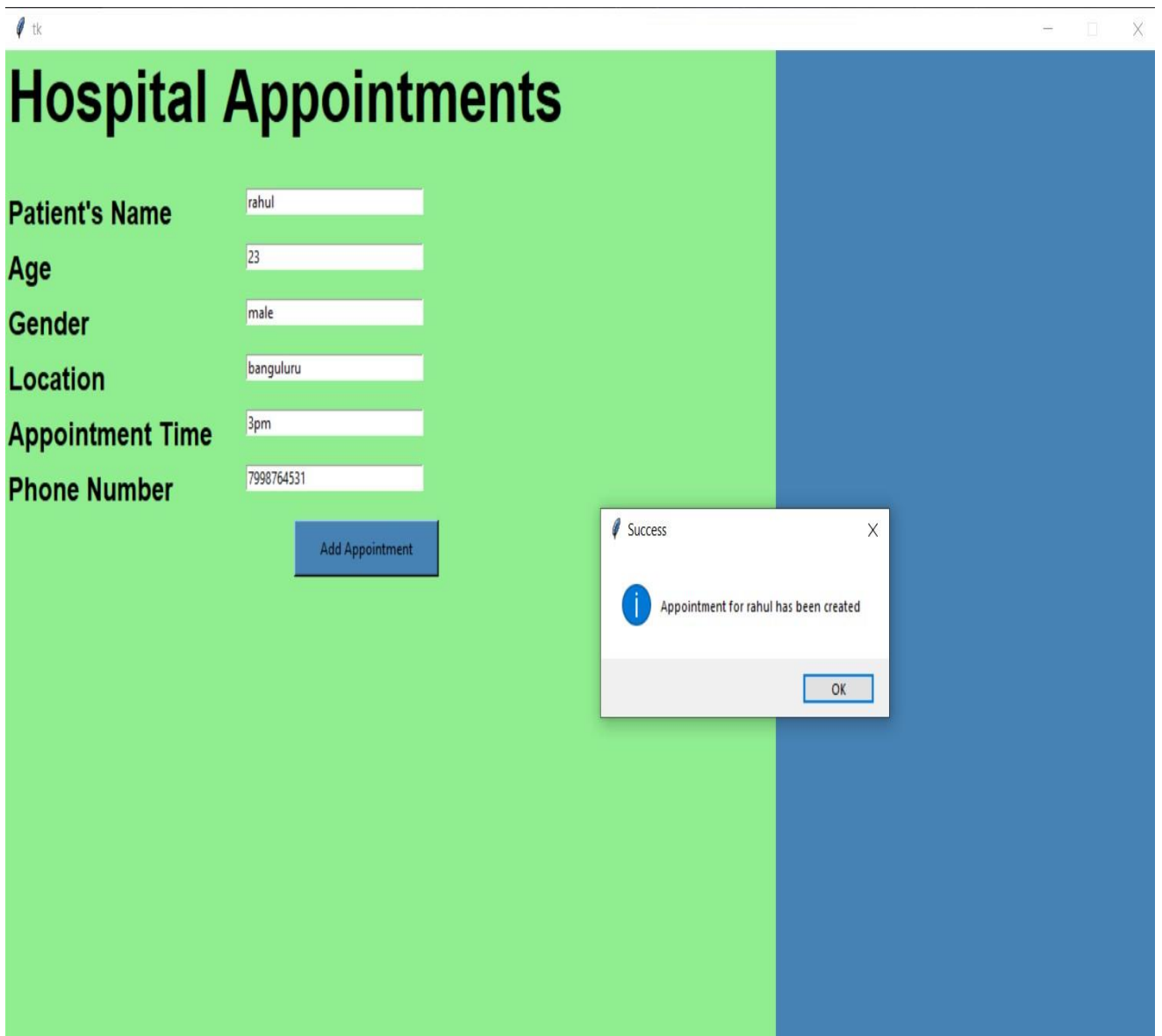


Fig 5.2 Appointments Added to Database

In this Fig 5.2., it show's a dialog box that the appointment has been created. By Linking Sqlite to Python we can add appointments to the Database which can be further used for retrieval, update and display details of appointments.

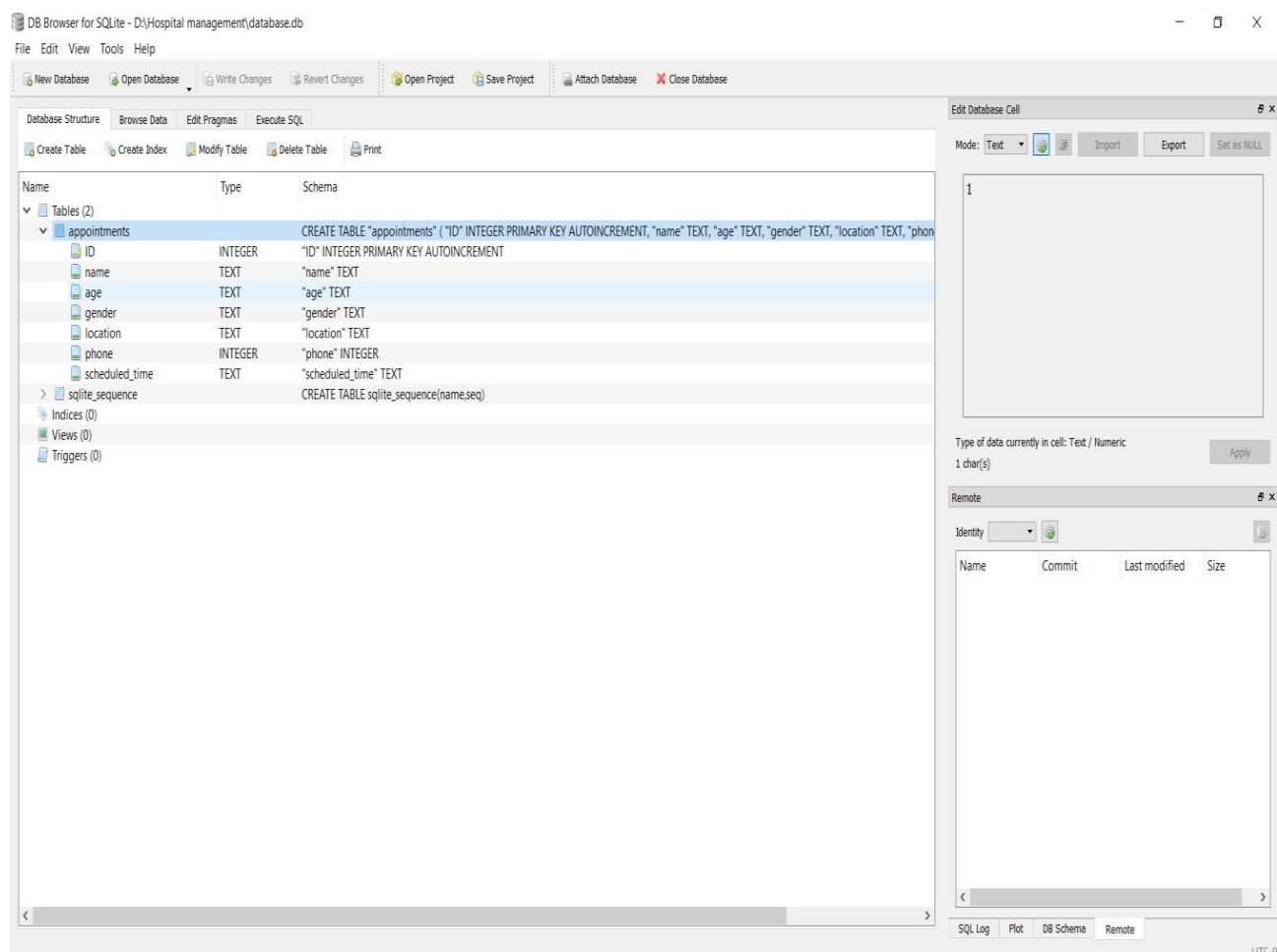


Fig 5.3 Creation of Table in Database

In Fig 5.3., It is used to create a Database. SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single file. SQLite database file are a recommended storage format by the US Library of Congress.

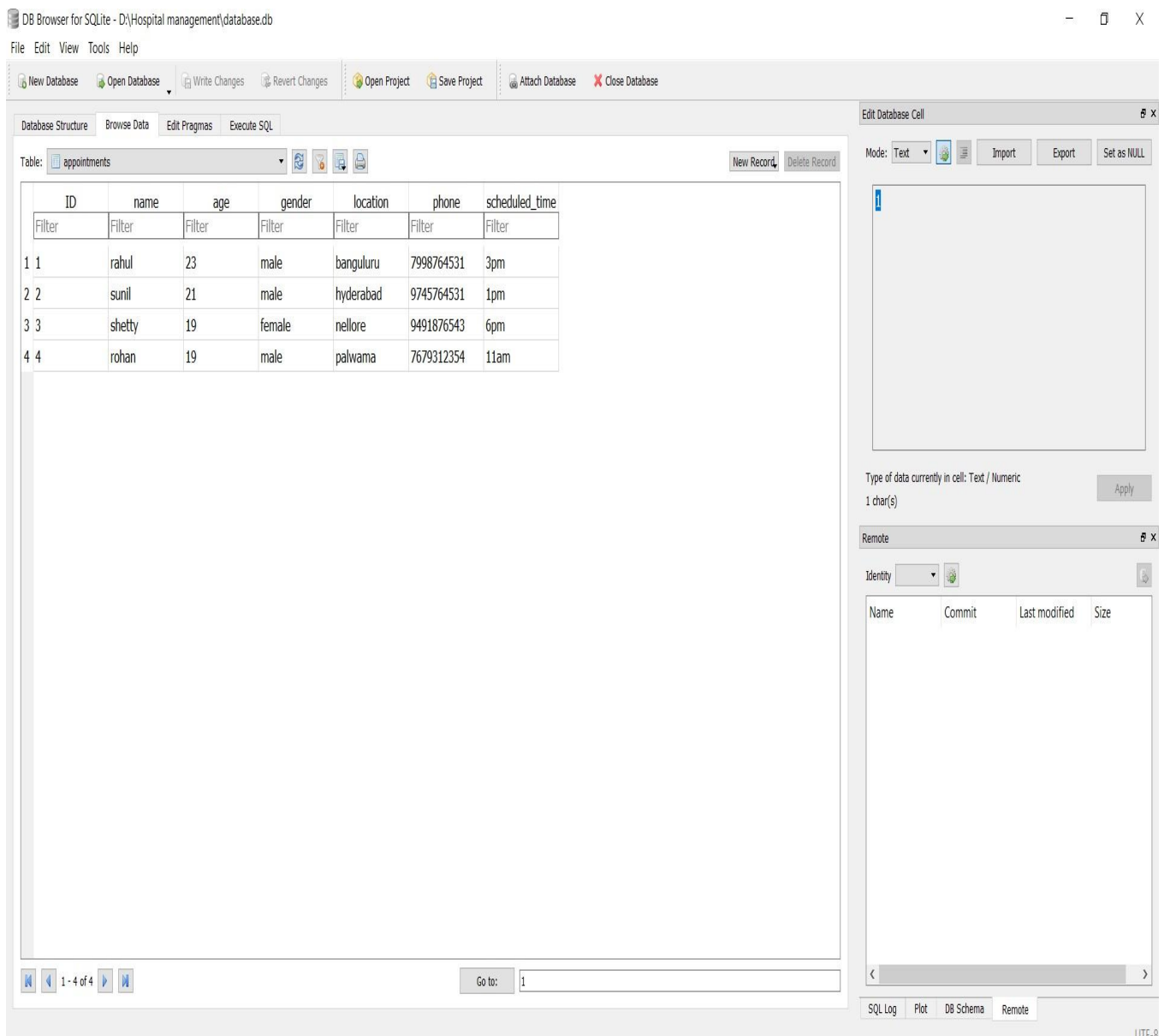


Fig 5.4 Created Table in Database

In the above Fig 5.4 shows that how the data has been stored in the table that has been created in the previous Fig 5.3. This will be helpful to display the content present in the list. In this case SQLite By Linking Sqlite to Python we can add appointments to the Database which can be further used for retrieval, update and display details of appointments.

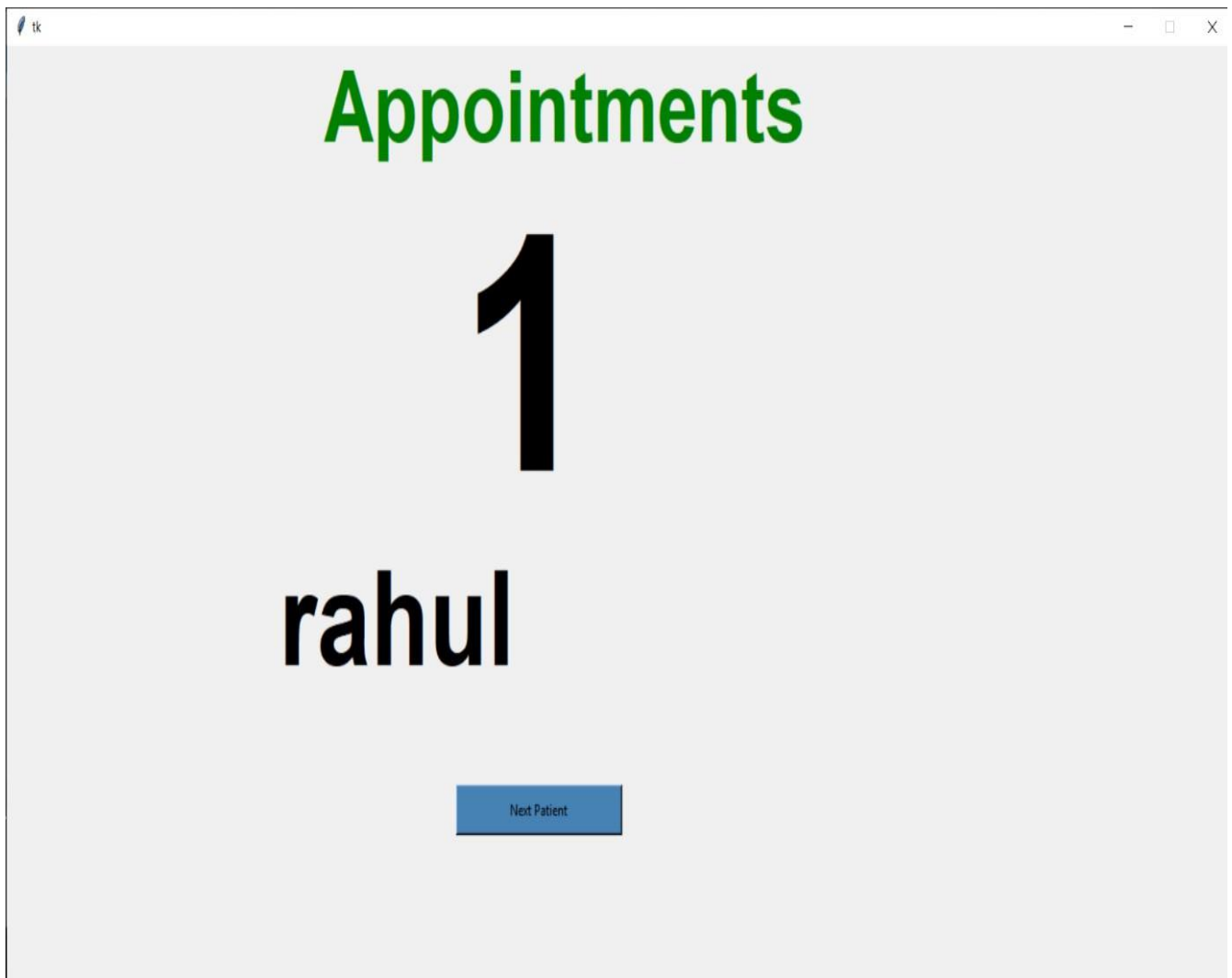


Fig 5.5 Displaying Appointment

In the above Fig 5.5., It shows that how the application was displaying the Patient name and the appointment number that will be easy and quick process to consult the doctor. In this we used the library called pyttsx3 which convert's the text in to speech. In this case application will read the name that shows on the screen.

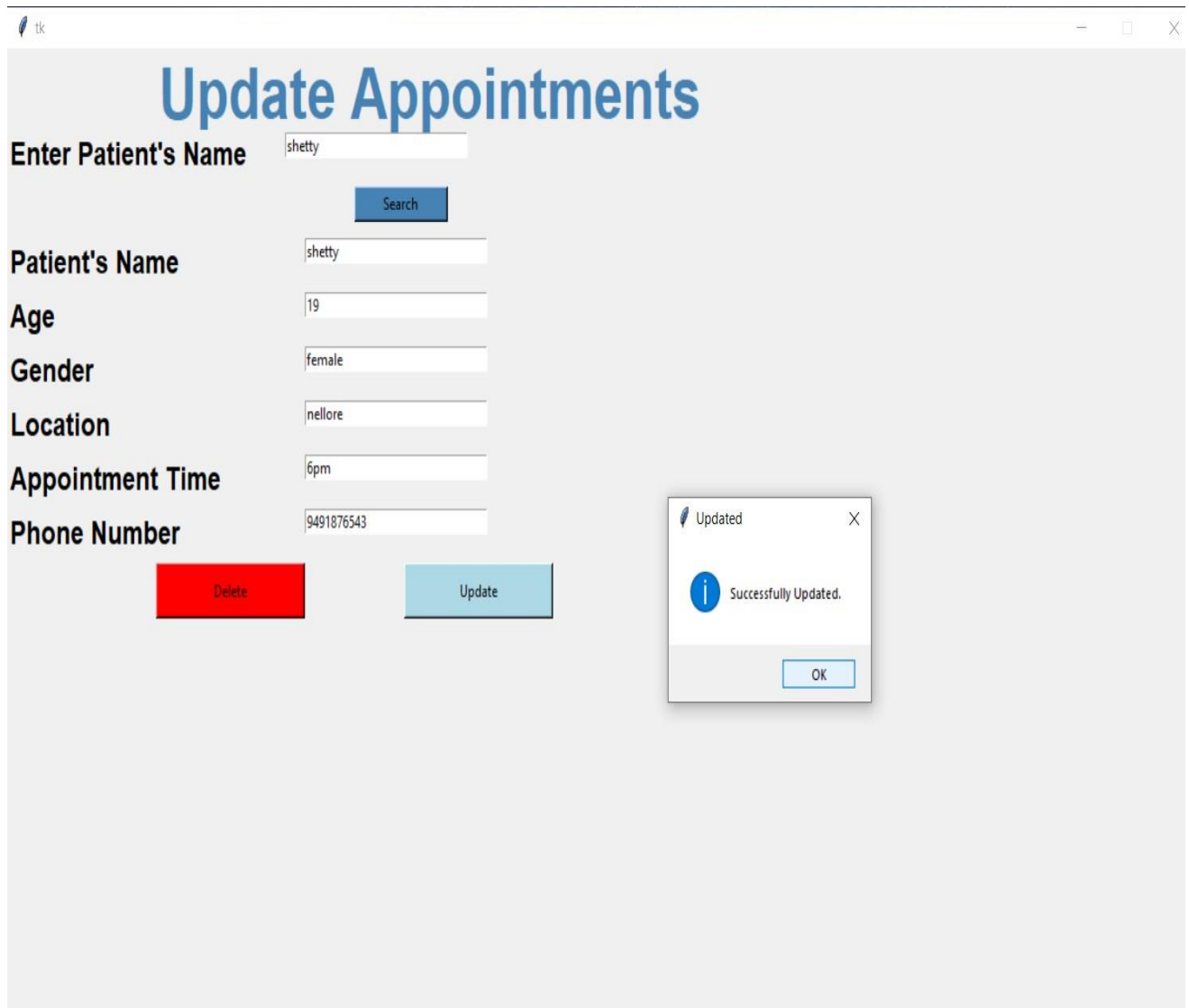


Fig 5.6 Updating and Deletion of Appointment

In the above Fig 5.6., When we given the existing Patient's name in the GUI application then it shows the old data that was given before in the database. Now it will be displayed. In this case we can modify data or we can delete the data even from the SQLite database. It will update use by showing a dialog box on the screen as shown in the above diagram.

REFERENCES

- [1] Python Essential Reference 4th edition, by David M Beazley, major reference Book
Published for Python, 1999.
- [2] Ingram Albert L, Lesley G. Hathorn and Alan Evans Beyond chat on the internet
Computers and Education vol 35 pp21-35, 2000.
- [3] Kitade Keiko L2 Learners' Discourse and SLA Theories in CMC: Coillaborative Interaction in
Internet Chat Computer Assisted Language Learning Vol 13 no 2 pp 143-166 , 2000.
- [4] The Python Standard Library by Example by Doug Hellmann, 2013.
- [5] PYTHON: Python Programming: Programming Guide For Beginners by Os Swift, 2015.
- [6] <https://www.javatpoint.com/python-tkinter>.
- [7] <http://www.ion.illinois.edu/IONresources/conferencing/pedagogic.asp>.