**I. SYSTEM SPECIFICATION AT A GLANCE**

**AIM**

The objective of our project is to computerize the telephone directory. Front end is developed using Python 3.7.7 software and back end is with MySQL 8.0. Thus, the entire project is based on Python-MySQL connectivity. The various modules included are,

1. Login
2. Add a record to the directory
3. Displaying all records
4. Search for a record using name
5. Search using city
6. Search using start alphabet
7. Modify a record
8. Delete a record
9. Logout

**MODULES**

**Login**

This is to provide a secured environment for the system. Software allows only the person with the correct username and password to enter into the software.

**Adding a record to the directory**

In this module, software provides the interface to enter the details required to add the details of a person to directory. This includes directory number, person’s name, city, telephone number, address and occupation.

**Displaying all records**

Module lists all details of every person using tree view widget in Python’s tkinter module.

**Search for a record using name**

Here name of the person is entered as input and all records are searched and it displays the record with the name entered as input

**Search using city**

This module allows the administrator to search the details based on city. This displays the details of all persons of a city.

**Search using start alphabet**

Here an alphabet is entered as input and the records with name starting with alphabet is displayed.

**Modify a record**

Here we enter directory number as input and search and display that record. The administrator can modify all details except directory number and save.

**Delete a record**

Here we enter directory number as input and search for that record and remove that record from our database.

**Logout**

This module allows the administrator to come out from the software whenever he is moving out of his seat. So that unauthorized people cannot handle the things.

**DATABASE**

This project includes one database in MySQL named ‘Telephone’, having table “” which stores details like directory number (dino), person’s name (pname), person’s city (pcity), telephone number (telephone), address of the person (address) and occupation of the person (occupation) and “” which stores

**II. REQUIREMENT ANALYSIS**

**HARDWARE REQUIREMENTS**

* Processor :  Intel® CoreTM i3 CPU
* CPU speed :  2.67 GHz
* RAM           :  4GB
* Hard disk memory :  80GB
* Cache                               :  512 KB
* Monitor
* Keyboard

**SOFTWARE REQUIREMENTS**

* Operating System           : Microsoft Windows 10
* Software                          : Python 3.8.5 and MySQL Server 8.0.21

**III. SYSTEM SPECIFICATIONS**

**DATA DESIGN**

**MySQL Database: Telephone**

**Table: DIRECTORY**

**Description of data:**

DNO: Directory number

NAME: Name of the person

PHONE\_NO: Telephone number of the person

JOB: Occupation of the person

**Table: ADDRESS**

**Description of data:**

AID: Address Id of the person

HNO: House number where the person

AREA: Locality in which the person lives

CITY: City of the person

PINCODE: Postal Address code where the person lives

DNO: Directory number

**IV. PYTHON OVERVIEW**

The **Python** **programming language** is an object-oriented language, which means that it can model real-world entities. It is also dynamically-typed because it carries out type-checking at runtime. It does so to make sure that the type of construct matches what we expect it to be. The distinctive feature of Python is that it is an interpreted language. The Python **IDLE** (Integrated Development Environment) executes instructions one line at a time.

### **What makes Python so powerful?**

Python provides the **PyPI** (Python Package Index). It is a repository of third-party [**Python modules**](https://data-flair.training/blogs/python-modules/) and you can install it using a program called **pip**.

## **Python History**

The Python programming language was conceived in the late 1980s and was named after the *“*BBC TV show Monty Python’s Flying Circus”*.* Guido van Rossum started implementing Python at CWI in the Netherlands in December of 1989.

**Python Constructs**

### **i. Functions**

A [**function in Python**](https://data-flair.training/blogs/python-function/) is a collection of statements grouped under a name. You can use it whenever you want to execute all those statements at a time. You can call it wherever you want and as many times as you want in a program. A function may return a value.

### **ii. Classes**

Python is an object-oriented language. It supports classes and objects. A class is an abstract data type. In other words, it is a blueprint for an object of a certain kind. It holds no values. An object is a real-world entity and an instance of a class.

### **iii. Modules**

A Python module is a collection of related classes and functions. We have modules for mathematical calculations, string manipulations, web programming, and many more. User can define his/her own modules, apart from built-in modules.

### **iv. Packages**

[**Python package**](https://data-flair.training/blogs/python-packages/) is a collection of related modules. You can either import a package or create your own.

### **v. List**

**A list** is a collection of values. Declared in the CSV (Comma-Separated Values) format and delimit using square brackets:

arity = [1,2,3]

A list may also contain elements of different types, and the indexing begins at 0. You can also slice lists; slicing is a way of retrieving some values from it.

### **vi. Tuple**

A [**tuple**](https://data-flair.training/blogs/python-tuples-syntax-examples/)is like a list, but it is immutable (you cannot change its values).

pizza = (‘base’, ‘sauce’, ‘cheese’, ‘mushroom’)

### **vii. Dictionary**

A [**dictionary**](https://data-flair.training/blogs/python-dictionaries/) is a collection of key-value pairs. Declare it using curly braces, and commas to separate key-value pairs. Also, separate values from keys using a colon (:).

student = {‘Name’: ‘Abc’, ‘Age’: 21}

### **viii. Comments and Docstrings**

Declare comments using an octothorpe (#). However, Python does support multiline comments using docstring. Also, docstrings are documentation strings that help explain the code.  
#This is a comment  
“““  
This is a docstring  
”””  
Python has a lot of other constructs. These include control structures, functions, exceptions, etc.

## **Features of Python**

The Python programming language is one of the richest languages.

**i. Easy**

Python is very easy to learn and understand; using this Python tutorial, any beginner can understand the basics of Python.

**ii. Interpreted**

It is interpreted (executed) line by line. This makes it easy to test and debug.

**iii. Object-Oriented**

The Python programming language supports classes and objects. We discussed these above.

**iv. Free and Open Source**

The language and its source code are available to the public for free; there is no need to buy a costly license.

**v. Portable**

Since it is open-source, you can run Python on Windows, Mac, Linux or any other platform. Your programs will work without needing to the changed for every machine.

**vi. GUI Programming**

You can use it to develop a GUI (Graphical User Interface). One way to do this is through **Tkinter**.

**vii. Large Library**

Python provides you with a large standard library. You can use it to implement a variety of functions without needing to reinvent the wheel every time. Just pick the code you need and continue. This lets you focus on other important tasks.

**Interface Python with SQL**

The Python programming language has powerful features for database programming. Python supports various databases like MySQL, Oracle, PostgreSQL, etc. Python also supports Data Definition Language (DDL), Data Manipulation Language (DML) and Data Query Statements. For database programming, Python DB-API is a widely used module that provides a database application programming interface. It’s a standard for database interfaces. Python Database API supports wide range of database servers like MySQL, Oracle, PostgreSQL, etc.

**Python-MySQL Connectivity**

While designing real-life applications, certain situations arise pertaining to storing some important and necessary information by the user. Usually, the data inputted by the user along with the generated output is displayed but not stored because all program execution takes place inside the RAM which is a temporary memory and as soon as we close the form, its contents (form input and generated output) get erased. They cannot be retrieved since they are not saved on a hard disk (or any secondary storage device). Thus, when the application is executed the second time, it requires a new set of inputs from the user. This limitation can be overcome by sending output generated and saving the input fetched from the user in a database created at the back-end of the application. The input is fetched from the user using Python Interface. This is termed as the **Front-End Interface (Python)** of the application. An application usually stores a lot of data in the form of a database which is not directly accessible to the user. This database is used by the application to give suitable response to the user. This database is called **Back-End Database (MySQL).**

**MySQL-CONNECTOR**

So, we are integrating MySQL with Python interface for executing any database applications. To establish connectivity between Python and MySQL, we have to install MySQL-connector using pip command on the command prompt; MySQL-connector is an interface for connecting to MySQL database server from Python. It implements the Python Database API.

**Python-MySQL Database Access**

MySQL-connector is the Python interface to work with MySQL databases. It must be imported in Python to work with any MySQL databases. All the SQL commands are implemented through Python Interface.

**V. MODULES**

Modules in our project along with their associated functions are as follows:

|  |  |  |
| --- | --- | --- |
| **Module Name** | **Purpose** | **Functions** |
| mysql.connnector | To establish connection between Python and MySQL | connect( )  cursor( )  commit( )  execute( )  fetchone( )  fetchall( )  close( ) |
| tkinter | to create the GUI | Tk( )  title( )  geometry( )  configure( )  deiconify( )  Label( )  place( )  Entry( )  mainloop( )  Button( )  get( )  pack()  frame( )  scrollbar( )  destroy( ) |
| Tkinter’s  tkmessagebox | To create intractable message boxes | showerror( )  showinfo( ) |
| User defined functions | Deals with the SQL based operations | checking()  insertingdir()  insertingadd()  inserteddir()  insertedadd()  display()  searchname()  searchcity()  searchname1()  show1()  show2()  show3() |

**VI. TABLE DESIGN AND STRUCTURE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sno.** | **Field Name** | **Type/**  **Constraints** | **Purpose** |
| 1 | DNO | INT (5)(PRIMARY KEY, NOT NULL) | It stores the directory id |
| 2 | NAME | VARCHAR(30) | It stores the name of the person |
| 3 | PHONE\_NO | VARCHAR (10) | It stores the telephone number |
| 4 | JOB | VARCHAR (30) | It stores the Occupation of the person |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sno.** | **Field Name** | **Type/**  **Constraints** | **Purpose** |
| 1 | AID | INT (5)(PRIMARY KEY, NOT NULL) | It stores the address id |
| 2 | HNO | VARCHAR(4) | It stores the house number |
| 3 | AREA | VARCHAR (30) | It stores the area |
| 4 | CITY | VARCHAR (30) | It stores the city |
| 5 | PINCODE | INT (6) | It stores the pincode |
| 6 | DNO | INT (5)(FOREIGN KEY) | It stores the directory id |

**VII. PROCEDURE/FUNCTION DESCRIPTION**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Function** | **Purpose** |
| 1 | checking() | Checks the username and password |
| 2 | insertingdir() | For entering values for directory table |
| 3 | insertingadd() | For entering values for address table |
| 4 | inserteddir() | For inserting values in directory table |
| 5 | insertedadd() | For inserting values in address table |
| 6 | display() | For displaying all the records |
| 7 | searchname() | For entering name to search |
| 8 | searchcity() | For entering city to search |
| 9 | searchname1() | For entering first letter of name to search |
| 10 | show1() | For searching and displaying the records based on name |
| 11 | show2() | For displaying the records based on city |
| 12 | show3() | For searching records with starting alphabet of name |
|  |  |  |

**VIII. SOURCE CODE**

**Main.py**

# Import Statements-----------------------------------------------------------------------------------------------

from tkinter import \*

from tkinter import messagebox

from tkinter import ttk

#import pyglet

import mysql.connector as ms

from PIL import Image,ImageTk

# Definitions----------------------------------------------------------------------------------------------------------

# Function to check Username and Password

def checking():

global i,t

un=username.get()

pw=password.get()

if un=='A' and pw=='A':

login\_window.destroy()

t=1

elif un=='Username' and pw=='Password':

messagebox.showerror(title='Geo', message='Default fields')

password\_entry.delete(0,END)

username\_entry.delete(0,END)

elif un=='' or pw=='':

messagebox.showerror(title='Geo', message='Please enter both fields')

password\_entry.delete(0, END)

else:

i+=1

if i<3:

x='incorrect password/username, '+str(i)+' of 3 tries'

messagebox.showerror(title='Geo', message=x)

password\_entry.delete(0, END)

password\_entry.focus()

else:

messagebox.showerror(title='Geo', message='Too many incorrect attempts....The program will now close')

login\_window.destroy()

# Function to enter value in Directory Table

def insertingdir():

global dno,name,phone,job

insertion\_window=Toplevel(main\_window)

insertion\_window.geometry('720x470')#

insertion\_window.grid\_propagate(0)

insertion\_window.resizable(False,False)

main\_window.withdraw()

dno=StringVar()

name=StringVar()

phone=StringVar()

job=StringVar()

Label(insertion\_window,bg='#bc8dbb',height=1000,width=1000).pack()

x='INSERT INTO DIRECTORY VALUES(%s,%s,%s,%s)'

lbl\_title = Label(insertion\_window, text = "INSERTING RECORDS", font=('Papyrus', 30,'bold', ),bg='PINK2', fg='GREY10').place(x=125,y=10)

dno\_entry=Entry(insertion\_window,font=('abel',15),border=1,width=25,bg='#e0a8c5',textvariable=dno)

dno\_entry.place(x=81,y=80)

Label(insertion\_window,text='DNO:',font=('BOLD',17),bg='#e0a8c5').place(x=5,y=80)

name\_entry=Entry(insertion\_window,font=('abel',15),border=1,width=25,bg='#e0a8c5',textvariable=name)

name\_entry.place(x=82,y=120)

Label(insertion\_window,text='NAME:',font=('BOLD',17),bg='#e0a8c5').place(x=0,y=120)

phone\_entry=Entry(insertion\_window,font=('abel',15),border=1,width=25,bg='#e0a8c5',textvariable=phone)

phone\_entry.place(x=445,y=80)

Label(insertion\_window,text='PHONE:',font=('BOLD',17),bg='#e0a8c5').place(x=350,y=80)

job\_entry=Entry(insertion\_window,font=('abel',15),border=1,width=25,bg='#e0a8c5',textvariable=job)

job\_entry.place(x=445,y=120)

Label(insertion\_window,text='JOB:',font=('BOLD',17),bg='#e0a8c5').place(x=360,y=120)

Button(insertion\_window,text='SUBMIT',bg='#e0a8c5',width=30,height=2,command=lambda:[inserteddir(),dno\_entry.delete(0,END),name\_entry.delete(0,END), phone\_entry.delete(0,END),job\_entry.delete(0,END)]).place(x=450,y=180)

Button(insertion\_window,text='CLOSE',bg='#e0a8c5',width=30,height=2,command=lambda:[insertion\_window.destroy(),main\_window.deiconify()]).place(x=10,y=280)

# Function to enter values in Address Table

def insertingadd()

global aid,hno,area,city,pincode,dno

insertion\_window=Toplevel(main\_window)

insertion\_window.geometry('300x333')

aid=StringVar()

hno=StringVar()

area=StringVar()

city=StringVar()

pincode=StringVar()

dno=StringVar()

main\_window.withdraw()

Label(insertion\_window,bg='#bc8dbb',height=1000,width=1000).pack()

x='INSERT INTO DIRECTORY VALUES(%s,%s,%s,%s)'

aid\_entry=Entry(insertion\_window,font=('abel',15),border=0,width=20,bg='#e0a8c5',textvariable=aid)

aid\_entry.place(x=32,y=10)

Label(insertion\_window,text='AID:',font=('abel',13),bg='#e0a8c5').place(x=0,y=10)

hno\_entry=Entry(insertion\_window,font=('abel',15),border=0,width=19,bg='#e0a8c5',textvariable=hno)

hno\_entry.place(x=37,y=50)

Label(insertion\_window,text='HNo:',font=('abel',13),bg='#e0a8c5').place(x=0,y=50)

area\_entry=Entry(insertion\_window,font=('abel',15),border=0,width=19,bg='#e0a8c5',textvariable=area)

area\_entry.place(x=40,y=90)

Label(insertion\_window,text='Area:',font=('abel',13),bg='#e0a8c5').place(x=0,y=90)

city\_entry=Entry(insertion\_window,font=('abel',15),border=0,width=20,bg='#e0a8c5',textvariable=city)

city\_entry.place(x=37,y=140)

Label(insertion\_window,text='City:',font=('abel',13),bg='#e0a8c5').place(x=0,y=140)

pincode\_entry=Entry(insertion\_window,font=('abel',15),border=0,width=20,bg='#e0a8c5',textvariable=pincode)

pincode\_entry.place(x=68,y=180)

Label(insertion\_window,text='Pin code:',font=('abel',13),bg='#e0a8c5').place(x=0,y=180)

dno\_entry=Entry(insertion\_window,font=('abel',15),border=0,width=20,bg='#e0a8c5',textvariable=dno)

dno\_entry.place(x=37,y=220)

Label(insertion\_window,text='DNo:',font=('abel',13),bg='#e0a8c5').place(x=0,y=220)

Button(insertion\_window,text='Submit',bg='#e0a8c5',width=20,command=lambda:[insertedadd(),aid\_entry.delete(0,END),hno\_entry.delete(0,END),area\_entry.delete(0,END),city\_entry.delete(0,END),pincode\_entry.delete(0,END),dno\_entry.delete(0,END)]).place(x=0,y=280)

Button(insertion\_window,text='Close',bg='#e0a8c5',width=20,command=lambda:[insertion\_window.destroy(),main\_window.deiconify()]).place(x=150,y=280)

# Function to Insert values into Directory Table

def inserteddir():

global dno,name,phone,job,dirdno

c.execute('SELECT DNO FROM DIRECTORY')

dirdno=[]

r=c.fetchall()

for i in r:

dirdno.append(i)

che=0

x='INSERT INTO DIRECTORY VALUES(%s,%s,%s,%s)'

dirno=dno.get()

nam=name.get()

phoneno=phone.get()

jobb=job.get()

saf=0

for i in nam:

if i.isalpha()==False:

saf=1

if saf==1:

messagebox.showerror(title='Geo',message='Name can’t have special charecters or numbers')

saf=0

for i in dirno:

if i.isnumeric()==False:

saf=1

if saf==1:

messagebox.showerror(title='Geo',message='Dno can only have numbers')

saf=0

for i in jobb:

if i.isalpha()==False:

saf=1

if saf==1:

messagebox.showerror(title='Geo',message='Job can’t have numbers or special charecters')

saf=0

for i in phoneno:

if i.isnumeric()==False:

saf=1

if saf==1:

messagebox.showerror(title='Geo',message='Phone number can only have numbers')

if dirno in dirdno:

messagebox.showerror(title='Geo',message='duplicate key')

if len(phoneno)!=10:

messagebox.showerror(title='Geo',message='invalid phone no, must be 10 digits')

che=1

if len(dirno)>5:

messagebox.showerror(title='Geo',message='invalid dno must be 5 or less digits')

che=1

if che!=1:

v=(dirno,nam,phoneno,jobb)

c.execute(x,v)

con.commit()

dirdno.append(dirno)

messagebox.showinfo(title='geo', message='inserted')

# Function to Insert values into Address Table

def insertedadd():

global aid,hno,area,city,pincode,dno

che=0

x='INSERT INTO ADDRESS VALUES(%s,%s,%s,%s,%s,%s)'

ai=aid.get()

hn=hno.get()

are=area.get()

cit=city.get()

pin=pincode.get()

dn=dno.get()

saf=0

for i in nam:

if i.isalpha()==False:

saf=1

if saf==1:

messagebox.showerror(title='Geo',message='can’t have special charecters')

saf=0

for i in nam:

if i.isalpha()==False:

saf=1

if saf==1:

messagebox.showerror(title='Geo',message='Name can’t have special charecters or numbers')

saf=0

for i in dirno:

if i.isnumeric()==False:

saf=1

if saf==1:

messagebox.showerror(title='Geo',message='Dno can only have numbers')

saf=0

for i in jobb:

if i.isalpha()==False:

saf=1

if saf==1:

messagebox.showerror(title='Geo',message='Job can’t have numbers or special charecters')

saf=0

for i in phoneno:

if i.isnumeric()==False:

saf=1

if saf==1:

messagebox.showerror(title='Geo',message='Phone number can only have numbers')

if dirno in dirdno:

messagebox.showerror(title='Geo',message='duplicate key')

if len(phoneno)!=10:

messagebox.showerror(title='Geo',message='invalid phone no, must be 10 digits')

che=1

if len(dirno)>5:

messagebox.showerror(title='Geo',message='invalid dno must be 5 or less digits')

che=1

if che!=1:

v=(dirno,nam,phoneno,jobb)

c.execute(x,v)

con.commit()

dirdno.append(dirno)

messagebox.showinfo(title='geo', message='inserted')

# Function to Display Directory Table

def display():

display\_window=Toplevel(main\_window)

display\_window.geometry('835x300')

main\_window.withdraw()

x='SELECT \* FROM DIRECTORY'

c.execute(x)

tree=ttk.Treeview(display\_window,selectmode='browse')

tree.grid(row=1,column=1,padx=20,pady=20)

tree['columns']=('DNo','Name','Phone','Job')

tree['show']='headings'

tree.column("DNo",width=200,anchor='c')

tree.column("Name",width=200,anchor='c')

tree.column("Phone",width=200,anchor='c')

tree.column("Job",width=200,anchor='c')

tree.heading("DNo",text='DNo')

tree.heading("Name",text='Name')

tree.heading("Phone",text='Phone')

tree.heading("Job",text='Job')

r=c.fetchall()

for i in r:

tree.insert('','end',iid=i[0],text=i[0],values=(i[0],i[1],i[2],i[3]))

Button(display\_window,text='close',font=('abel',10),border=0,command=lambda:[display\_window.destroy(),main\_window.deiconify()]).place(x=0,y=250)

# Function to Enter Name to Search

def searchname():

global name

search\_window=Toplevel(main\_window)

search\_window.geometry('300x250')

search\_window.resizable(False,False)

main\_window.withdraw()

name=StringVar()

Label(search\_window,bg='#bc8dbb',height=1000,width=1000).pack()

Label(search\_window,text='Name',bg='#e0a8c5',font=('abel',13)).place(x=0,y=30)

Entry(search\_window,bg='#e0a8c5',border=0,font=('abel',15),textvariable=name).place(x=44,y=30)

Button(search\_window,text='Enter',bg='#e0a8c5',width=20,command=lambda:[show1(),search\_window.withdraw()]).place(x=0,y=100)

Button(search\_window,text='Close',bg='#e0a8c5',width=20,command=lambda:[search\_window.destroy(),main\_window.deiconify()]).place(x=150,y=100)

# Function to Enter City to Search

def searchcity():

global city

search\_window=Toplevel(main\_window)

search\_window.geometry('300x250')

search\_window.resizable(False,False)

main\_window.withdraw()

city=StringVar()

Label(search\_window,bg='#bc8dbb',height=1000,width=1000).pack()

Label(search\_window,text='City',bg='#e0a8c5',font=('abel',13)).place(x=0,y=30)

Entry(search\_window,bg='#e0a8c5',border=0,font=('abel',15),textvariable=city).place(x=44,y=30)

Button(search\_window,text='Enter',bg='#e0a8c5',width=20,command=lambda:[show2(),search\_window.withdraw()]).place(x=0,y=100)

Button(search\_window,text='Close',bg='#e0a8c5',width=20,command=lambda:[search\_window.destroy(),main\_window.deiconify()]).place(x=150,y=100)

# Function to Enter First Letter of Name to Search

def searchname1():

global name

search\_window=Toplevel(main\_window)

search\_window.geometry('300x250')

search\_window.resizable(False,False)

main\_window.withdraw()

name=StringVar()

Label(search\_window,bg='#bc8dbb',height=1000,width=1000).pack()

Label(search\_window,text='Name',bg='#e0a8c5',font=('abel',13)).place(x=0,y=30)

ent=Entry(search\_window,bg='#e0a8c5',border=0,font=('abel',15),textvariable=name)

ent.place(x=44,y=30)

Button(search\_window,text='Enter',bg='#e0a8c5',width=20,command=lambda:[show3(),search\_window.withdraw()]).place(x=0,y=100)

Button(search\_window,text='Close',bg='#e0a8c5',width=20,command=lambda:[search\_window.destroy(),main\_window.deiconify()]).place(x=40,y=100)

# Function to Search by Name

def show1():

global name

show\_window=Toplevel(main\_window)

show\_window.geometry('835x50')

nam=name.get()

x='SELECT \* FROM DIRECTORY WHERE NAME=%s'

v=(nam,)

c.execute(x,v)

tree=ttk.Treeview(show\_window,selectmode='browse')

tree.pack()

tree['columns']=('DNo','Name','Phone','Job')

tree['show']='headings'

tree.column("DNo",width=200,anchor='c')

tree.column("Name",width=200,anchor='c')

tree.column("Phone",width=200,anchor='c')

tree.column("Job",width=200,anchor='c')

tree.heading("DNo",text='DNo')

tree.heading("Name",text='Name')

tree.heading("Phone",text='Phone')

tree.heading("Job",text='Job')

r=c.fetchall()

for i in r:

tree.insert('','end',iid=i[0],text=i[0],values=(i[0],i[1],i[2],i[3]))

Button(show\_window,text='close',font=('abel',20),border=0,command=lambda:[show\_window.destroy(),main\_window.deiconify()]).place(x=0,y=70)

# Function to Search by City

def show2():

global city

show\_window=Toplevel(main\_window)

show\_window.geometry('835x50')

cit=city.get()

x='SELECT \* FROM DIRECTORY D,ADDRESS A WHERE A.CITY=%s AND A.DNO=D.DNO '

v=(cit,)

c.execute(x,v)

tree=ttk.Treeview(show\_window,selectmode='browse')

tree.grid(row=1,column=1,padx=20,pady=20)

tree['columns']=('DNo','Name','Phone','Job')

tree['show']='headings'

tree.column("DNo",width=200,anchor='c')

tree.column("Name",width=200,anchor='c')

tree.column("Phone",width=200,anchor='c')

tree.column("Job",width=200,anchor='c')

tree.heading("DNo",text='DNo')

tree.heading("Name",text='Name')

tree.heading("Phone",text='Phone')

tree.heading("Job",text='Job')

r=c.fetchall()

for i in r:

tree.insert('','end',iid=i[0],text=i[0],values=(i[0],i[1],i[2],i[3]))

Button(show\_window,text='close',font=('abel',20),border=0,command=lambda:[show\_window.destroy(),main\_window.deiconify()]).place(x=0,y=70)

# Function to Search by First Letter of Name

def show3():

global name

nam=name.get()

x='SELECT \* FROM DIRECTORY WHERE NAME LIKE "%s%"'

show\_window=Toplevel(main\_window)

show\_window.geometry('835x50')

v=(nam,)

c.execute(x,v)

tree=ttk.Treeview(show\_window,selectmode='browse')

tree.grid(row=1,column=1,padx=20,pady=20)

tree['columns']=('DNo','Name','Phone','Job')

tree['show']='headings'

tree.column("DNo",width=200,anchor='c')

tree.column("Name",width=200,anchor='c')

tree.column("Phone",width=200,anchor='c')

tree.column("Job",width=200,anchor='c')

tree.heading("DNo",text='DNo')

tree.heading("Name",text='Name')

tree.heading("Phone",text='Phone')

tree.heading("Job",text='Job')

r=c.fetchall()

for i in r:

tree.insert('','end',iid=i[0],text=i[0],values=(i[0],i[1],i[2],i[3]))

Button(show\_window,text='close',font=('abel',20),border=0,command=lambda:[show\_window.destroy(),main\_window.deiconify()]).place(x=0,y=70)

# Login Page-----------------------------------------------------------------------------------------------------------

pyglet.font.add\_file('FontsFree-Net-Abel-Regular.ttf')

t,i=0,0

login\_window=Tk(className=" Geo Enterprises")

login\_window.geometry("1079x720")

login\_window.resizable(False,False)

# Vars

username=StringVar()

password=StringVar()

# Labels

loginimage=PhotoImage(file="1079x720.png")

Label(login\_window,image=loginimage).pack()

Label(login\_window,width=50,height=30,bg='white').place(x=350,y=200)

Label(login\_window,text='Login',bg='white',font=('abel',40)).place(x=465,y=220)

# Username Entry

username\_entry=Entry(login\_window,font=('abel',15),border=0,width=20,bg='white',textvariable=username)

username\_entry.place(x=390,y=350)

username\_entry.insert(0,'Username')

username\_entry.focus()

Frame(login\_window,width=290,height=2,bg='black').place(x=380,y=375)

# Password Entry

password\_entry=Entry(login\_window,font=('abel',15),border=0,width=20,bg='white',textvariable=password,show='\*')

password\_entry.place(x=390,y=450)

password\_entry.insert(0,'Password')

Frame(login\_window,width=290,height=2,bg='black').place(x=380,y=475)

# Button To Login

Button(login\_window,width=20,text='Login',font='abel',border=0,bg='light grey',command=checking,cursor="hand2").place(x=450,y=575)

login\_window.mainloop()

if t!=1:

exit()

# Sql-------------------------------------------------------------------------------------------------------

con=ms.connect(host='localhost',user='root',password='MMM123',database='Directory')

if con.is\_connected()==False:

print('not connected')

exit()

c=con.cursor()

x='CREATE TABLE IF NOT EXISTS DIRECTORY(DNO INT(5) NOT NULL PRIMARY KEY,NAME VARCHAR(30),PHONE\_NO CHAR(10),JOB VARCHAR(30))'

c.execute(x)

x='CREATE TABLE IF NOT EXISTS ADDRESS(AID INT(5) NOT NULL PRIMARY KEY,HNO VARCHAR(4),AREA VARCHAR(30),CITY VARCHAR(30),PINCODE INT(6),DNO INT(5),CONSTRAINT FK FOREIGN KEY (DNO) REFERENCES DIRECTORY(DNO) ON DELETE CASCADE ON UPDATE CASCADE )'

c.execute(x)

# Creating Window---------------------------------------------------------------------------------------------------

main\_window=Tk(className='Geo enterprises')

#width=main\_window.winfo\_screenwidth()

#height=main\_window.winfo\_screenheight()

#m=str(width)+'x'+str(height)

#main\_window.geometry("1500x940")#

#main\_window.resizable(False,False)

IMG1="geo.png"

def canvases(images,w,h):

photo=Image.open(images)

photo1=photo.resize((w,h),Image.Resampling.LANCZOS)

photo2=ImageTk.PhotoImage(photo1)

canvas = Canvas(main\_window, width='%d'%w, height='%d'%h)

canvas.grid(row = 0, column = 0)

canvas.grid\_propagate(0)

canvas.create\_image(0, 0, anchor = NW, image=photo2)

canvas.image=photo2

return canvas

w = main\_window.winfo\_screenwidth()

h = main\_window.winfo\_screenheight()

canvas=canvases(IMG1,w,h)

# Labels

backimage=PhotoImage(file="geo.png")

Label(canvas,image=backimage).pack()

# Options

#lbl\_title = Label(canvas, text = "GEO ENTERPRISES", font=('Papyrus', 50,'bold', ),bg='PINK2', fg='GREY5')

#lbl\_title.place(x=370,y=40)

Button(canvas,width=45,height=3,font=('abel',15),text='INSERTING RECORDS(Directory)',border=1,cursor='hand2',command=insertingdir,bg='#b782f1').place(x=200,y=160)

Button(canvas,width=45,height=3,font=('abel',15),text='INSERTING RECORDS(Address)',border=1,cursor='hand2',bg='#b782f1',command=insertingadd).place(x=200,y=285)

Button(canvas,width=45,height=3,font=('abel',15),text='DISPLAY',border=1,cursor='hand2',bg='#b782f1',command=display).place(x=200,y=405)

Button(canvas,width=45,height=3,font=('abel',15),text='SEARCH BY NAME',border=1,cursor='hand2',bg='#b782f1',command=searchname).place(x=200,y=520)

Button(canvas,width=45,height=3,font=('abel',15),text='SEARCH BY CITY',border=1,cursor='hand2',bg='#9772b1',command=searchcity).place(x=670,y=160)

Button(canvas,width=45,height=3,font=('abel',15),text='SEARCH BY 1ST LETTER OF NAME',border=1,cursor='hand2',bg='#9772b1',command=searchname1).place(x=670,y=285)

Button(canvas,width=45,height=3,font=('abel',15),text='UPDATE',border=1,cursor='hand2',bg='#9772b1').place(x=670,y=405)

Button(canvas,width=45,height=3,font=('abel',15),text='DELETE',border=1,cursor='hand2',bg='#9772b1').place(x=670,y=520)

main\_window.mainloop()

**IX. OUTPUT SCREENS**

**Login screen**

**Home screen**

**Adding a record**

**Display all records**

**Searching for a number using name**

**Search using city**

**Search using the start alphabet of name**

**Modifying a record**

**Displaying all records**

**Deleting a record**

**Displaying all records**

**X. FUTURE ENHANCEMETS**

In the future designs, this program can be modified to not only store details but also the images of the people. Option for changing login details can be included.

**XI. BIBLIOGRAPHY**

* Computer Science with Python by Sumita Arora
* Tutorialspoint.com
* RealPython.com
* Python and Tkinter Programming by John
* Grayson
* Codemy.com
* Stackoverflow.com