

Industrial Internship Report on

"Password Manager"

Prepared by

[Syeda Taneen Falak]

Executive Summary

This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT).

This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks' time.

My project was about the Password Manager.

This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

TABLE OF CONTENT

1 Preface

2 Introduction

2.1 Basic Introduction

2.2 Objective and Scope

2.3 Tools and Techniques

***Python**

***Python GUI Framework**

***Tinkter**

***MySQL**

3. System Analysis

3.1 Preliminary Analysis & Information Gathering

3.2 Input /Output:

4. Screenshots

5.Coding

1.PREFACE:

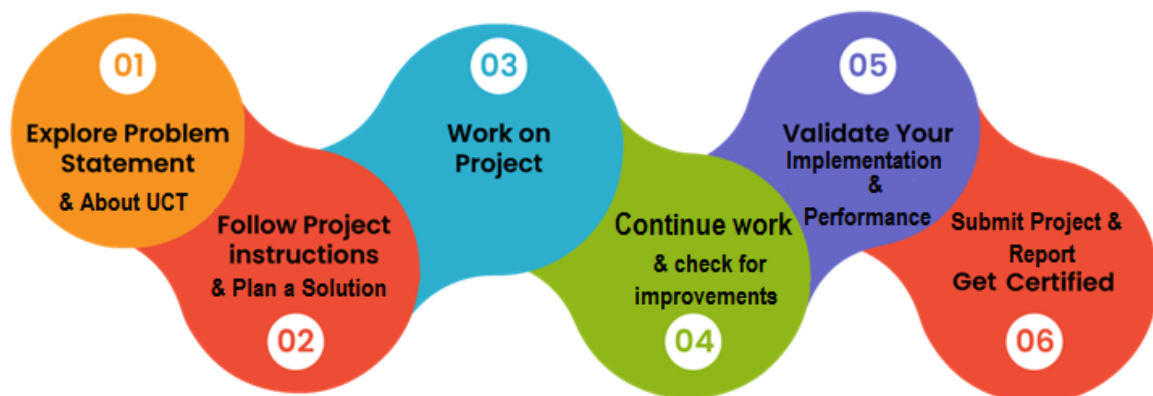
This includes the summary of the whole 6 week's work. In this 6 week's I worked on my project which was given by the Upskill Campus, my project topic is "The Password Manager"

In this project I learned a lot of this related to python and upskill campus was a great guide to develop our skill and gets hands on practice on the programming language called Python.

I also learned GUI(Graphical user Interface) and I implemented it in my project, along with this I worked on storing the details in the database and used the application(XAMPP Control Panel) to store the applications data such as email Id, Name, Passwords etc.

In my project "The Password Manager" it includes the register page, login page which are interconnected to each other and the data which u are filling like passwords and all are stored in the database and for the database I used XAMPP Control Panel .

It was a great experience to work with the Upskill Campus and thank u providing such a great opportunity. The Program was planned in such a beautiful way the below figure represents it



Thank u to the guides in the upskill campus group for supporting and making this project a success.

2. INTRODUCTION

2.1 Basic Introduction:

Password Manager is an GUI based application that allows users to store and generate random passwords. It is created in Python using Tkinter and Mysql database to manage passwords for several application. A password manager is a program that houses all your passwords, as well as other information, in one convenient location with one master password. The benefits of using Password Manager are:

- A Password Manager will do the work of creating the complicated passwords you need to help protect your online accounts.
- You need to remember only the password manager's password. That single password will give you access to all of your others.

Not only do password managers help securely house your passwords, but they can also generate passwords that are unique and complex, which makes them more difficult to crack or guess.

It also simplifies your life by making account access easier for you and more difficult for hackers. You don't have to memorize any passwords except for the password to your password manager.

That means you can actually follow unpleasant but useful security advice. Like ever reusing a password and always using long, strong and complex passwords.

2.2 OBJECTIVE AND SCOPE:

In this project, we will built an application which will store the User's password, as well as other information, in one convenient location with one master password. Python Features and methods are used to implement in this project.

Today, people have a large number of passwords for social media sites, work logins, shopping pages, online banking and much more. While it is important to use strong passwords and to use different passwords on each site, it can be a difficult task to remember all them. With a password manager, you simply enter the GUI app, provide the master password you set for the password manager in that software, then log in to the GUI app and the username and password will be stored for you. However, the primary purpose of this project to make Password Manager application user-friendly so that any individual can interact with the system.

The main scope and deliverables of the project would be to:

- Understand and prepare detailed user requirement and specifications.
- Prepare high level and detailed design specifications of the system.
- Prepare Test Plan & Test cases.
- Develop the system and coding.
- Perform unit testing, integration testing and system testing.
- Demonstrate a bug free application after suitable modification, if needed.
- Develop a GUI application using Python Tkinter.

2.3 Tools And Technologies:

Python:-

Python is an interpreted high-level and general-purpose programming language. Its language constructs and object-oriented approach aim to help programming write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports structured, Object-oriented and functional programming.

Python was created in the late 1980's and first released in 1991, by Guido Van Rossum at Centrum Wiskunde & Informatica in the Netherlands as a successor to ABC programming language.

Python GUI Framework:

A Graphical User Interface (GUI) is the first thing your user sees and interacts with when he opens your application or website. Having a good GUI goes a long way in increasing your platform's reputation and user count.

A user interface usually includes a host of visual elements like icons, buttons, graphics, displayed text, and several other forms of input, like checkbox, text input boxes, and such.

Tkinter :

Often referred to as the go-to GUI toolkit by a majority of Python developers, **Tkinter** was created to equip modern developers with a standard interface to the Tk GUI toolkit with its Python bindings. The name Tkinter comes from Tk interface. Tkinter was written by Fredrik Lundh and it is [free software](#) released under a Python license.

(a)Window:

This term has different meanings in different contexts, but in general it refers to a rectangular area somewhere on the user's display screen.

(b)Widget:

The generic term for any of the building blocks that make up an application in a graphical user interface.

1.Core widgets:

Containers: frame, label frame, top level, paned window. Buttons: button, radio button, check button (checkbox), and menu button. Text widgets: label, message, text. Entry widgets :scale, scrollbar, list box, slider, spin box, entry (single line),option menu, text (multiline), and canvas (vector and pixel graphics).

2.Tkinter provides three modules that allow pop-up dialogs to be displayed: tk.messagebox (confirmation, information, warning and error dialogs)

(c)Themed Tk:-

This allows Tk widgets to be easily themed to look like the native desktop environment in which the application is running, thereby addressing a long-standing criticism of Tk (and hence of Tkinter). Some widgets are exclusive to ttk, such as the combo box, progress bar and tree view widgets.

(d)Frame:-

In Tkinter, the Frame widget is the basic unit of organization for complex layouts. A frame is a rectangular area that can contain other widgets.

(e)Pack:-

Pack it into position so it becomes visible. Developers also have the option to use ‘.grid()’(row=int , column=int to define rows and columns to position the widget, defaults to 0) and ‘.place() ’ (relx=int or decimal ,rely=int or decimal , define coordinates in the frame, or window) .Tkinter is included with standard [Linux](#), [Microsoft Windows](#) and Mac OS X installs of Python.

MySQL:

My SQL is an open-source relational database management system(RDBMS). language for relational database management systems. It is used for storing, manipulating and retrieving data in databases.

Its name is a combination of "My", the name of co-founder [Michael Widenius](#)'s daughter, and "SQL", the abbreviation for [Structured Query Language](#). A [relational database](#) organizes data into one or more data tables in which data types may be related to each other these relations help structure the data

SQL is used to create, modify and extract data from the relational database, as well as control user access to the database.

3. SYSTEM ANALYSIS

3.1 Preliminary Analysis & Information Gathering :

System analysis is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvements on the system. System analysis is a problem-solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is viewed as a whole, the inputs are identified and the system is subjected to close study to identify the problem areas. The solutions are given as a proposal. The proposal is reviewed on user request and suitable changes are made. This loop ends as soon as the user is satisfied with the proposal.

3.2 Input/Output:

Input Design:

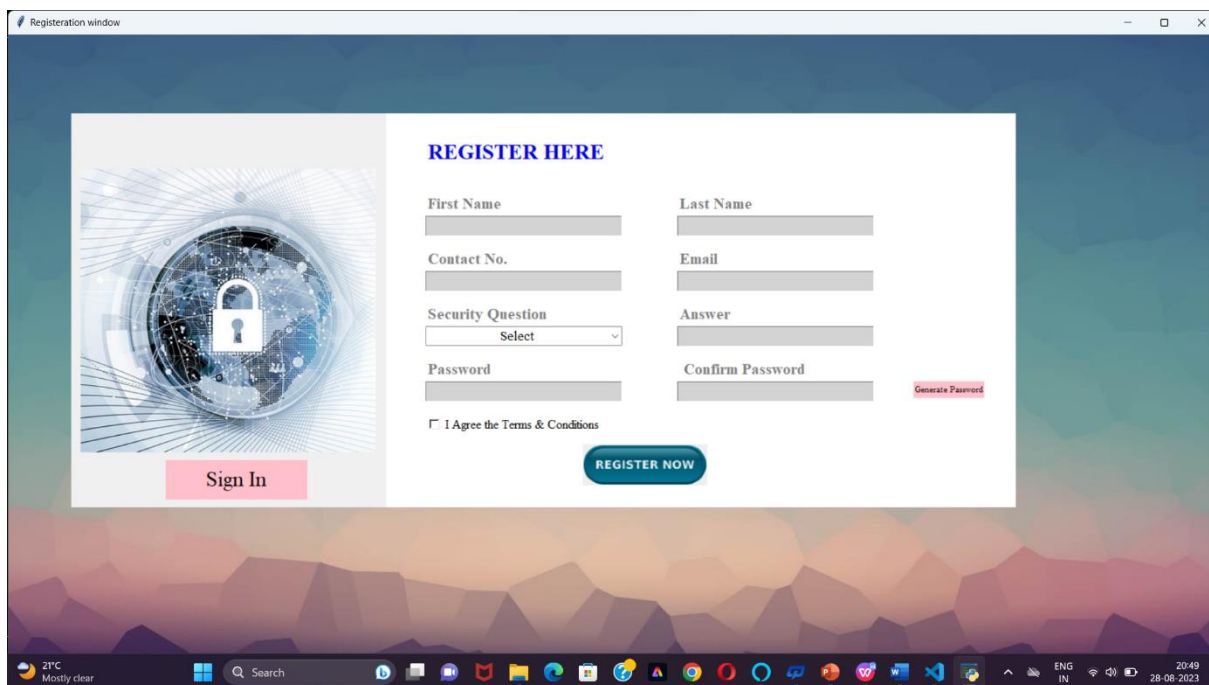
Input design is the link that ties the information system into the world of its users. The input design involves determining the inputs, validating the data, minimizing the data entry and provides a multi-user facility. Inaccurate inputs are the most common cause of errors in data processing.

Errors entered by the data entry operators can be controlled by input design. The user-originated inputs are converted to a computer-based format in the input design.

Input data are collected and organized into groups of similar data. Once identified, the appropriate input media are selected for processing. All the input data are validated and if any data violates any conditions, the user is warned

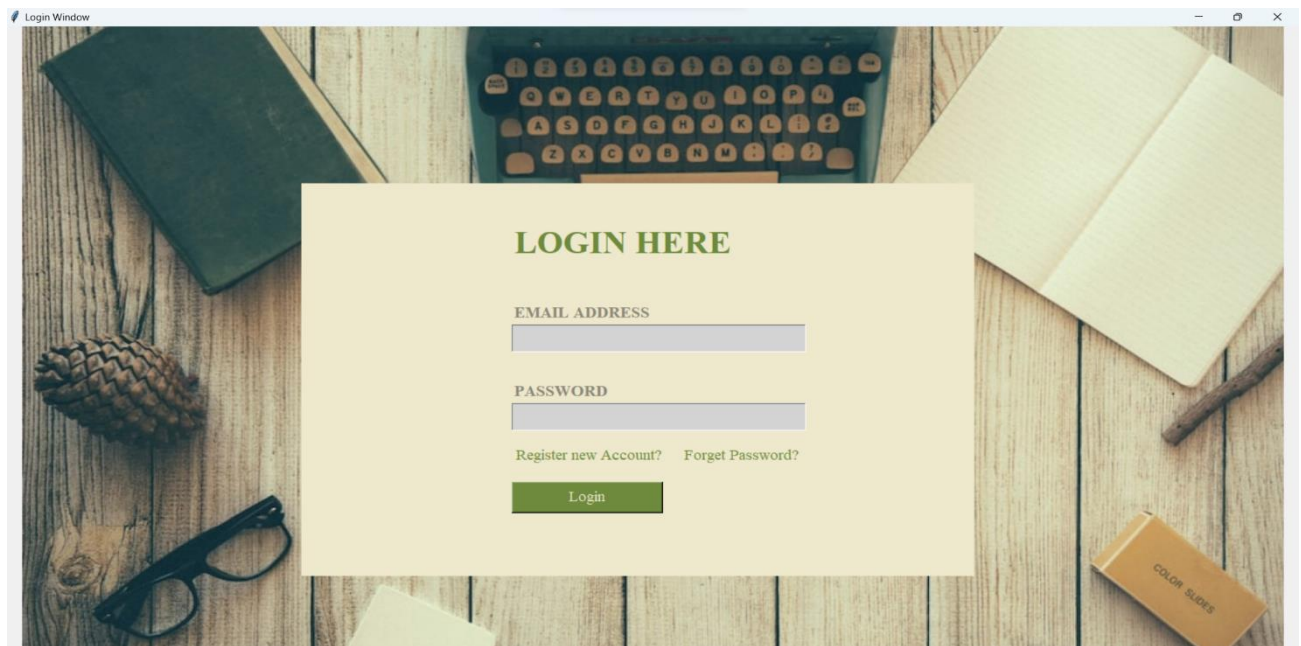
by a message. If the data satisfies all the conditions, it is transferred to the appropriate tables in the database. In this project the student details are to be entered at the time of registration. A page is designed for this purpose which is user friendly and easy to use. The design is done such that users get appropriate messages when exceptions occur.

3.Screenshots:-

A screenshot of a web browser window titled "Registration window". The page has a dark blue header and a light blue background with a geometric pattern. On the left, there is a circular graphic with a padlock and the text "Sign In" below it. On the right, there is a "REGISTER HERE" section with a form. The form has two columns: the left column contains "First Name", "Contact No.", "Security Question" (with a dropdown menu showing "Select"), and "Password"; the right column contains "Last Name", "Email", "Answer", and "Confirm Password". There is a "Generate Password" link next to the "Confirm Password" field. At the bottom of the form, there is a checkbox labeled "I Agree the Terms & Conditions" and a "REGISTER NOW" button. The Windows taskbar is visible at the bottom, showing the date and time as 20:49 on 28-08-2023.

Register Page

Login Window



The login window features a background image of a wooden desk with a typewriter, a pinecone, glasses, and a notebook. A central yellow box contains the login form.

LOGIN HERE

EMAIL ADDRESS

PASSWORD

[Register new Account?](#) [Forget Password?](#)

phpMyAdmin

Recent

Favorites

New

employee

employee1

New

employee

employee2

information_schema

mysql

performance_schema

phpmyadmin

test

Server: 127.0.0.1 * Database: employee1 * Table: employee

Browse

Structure

SQL

Search

Insert

Export

Import

Privileges

Operations

Tracking

Triggers

Showing rows 0 - 2 (3 total, Query took 0.0008 seconds.)

SELECT * FROM `employee`

Profiling

[Edit inline]

[Edit]

[Explain SQL]

[Create PHP code]

[Refresh]

Show all

Number of rows: 25

Filter rows: Search this table

Sort by key: None

Extra options

Id

f_name

l_name

contact

email

question

answer

password

Edit

Copy

Delete

1

Syeda Taneen

Falak

8961832156

taneensyed4@gmail.com

Your Birth Place

Banglore

56789

Edit

Copy

Delete

13

Taneen

Falak

2345678976

taneenfalak@gmail.com

Your Birth Place

Manglore

Hallo@1234

Edit

Copy

Delete

14

Taneen

Falak

4567890123

taneen123@gmail.com

Your Birth Place

Manglore

HELLO

Check all

With selected

Edit

Copy

Delete

Export

Show all

Number of rows: 25

Filter rows: Search this table

Sort by key: None

Query results operations

Print

Copy to clipboard

Export

Display chart

Create view

Bookmark this SQL query

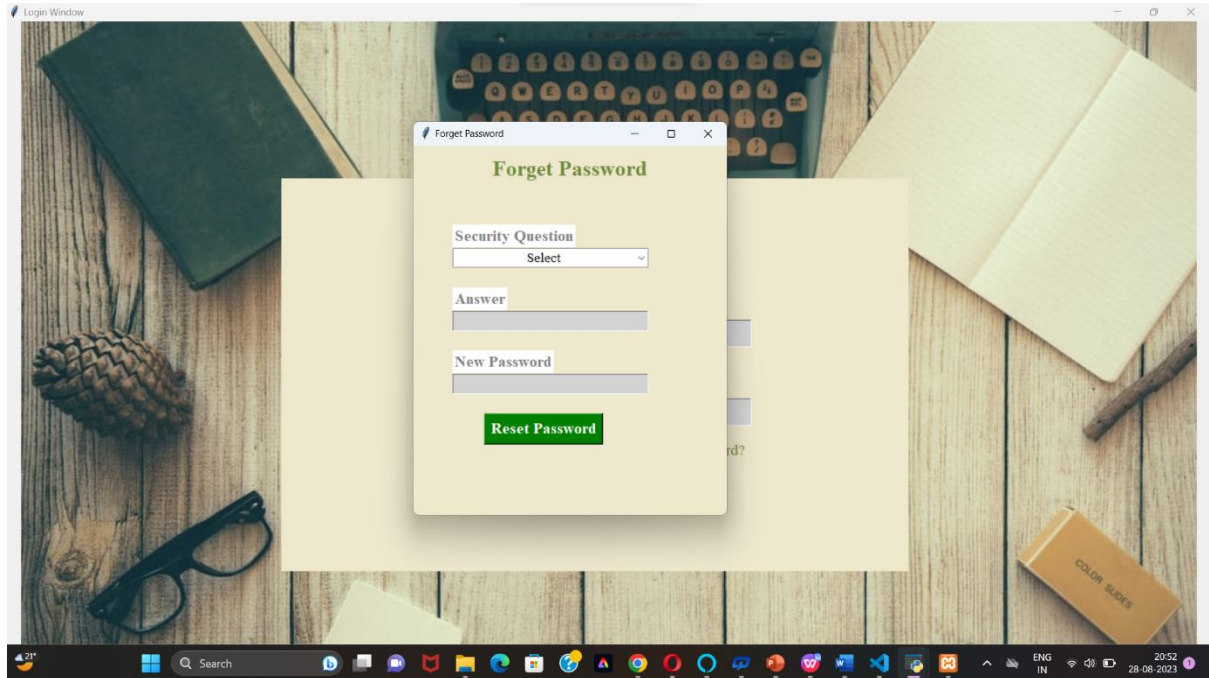
Label:

Let every user access this bookmark

Bookmark this SQL query

Console

Database



Code:-

Register Page

```
from tkinter import*
from tkinter import ttk,messagebox
from PIL import Image,ImageTk #pip install pillow
import pymysql #pip install pymysql
import random

class Register:
    def __init__(self,root):
        self.root=root
        self.root.title("Registration window")
        self.root.geometry("1550x1250+0+0")
        self.root.config(bg="grey")
        #===Bg Image===
```

```

self.bg=ImageTk.PhotoImage(file="images/Bg.jpg")
bg=Label(self.root,image=self.bg).place(x=0,y=0,relwidth=1,relheight=1
)

#===LEFT Image===
self.left=ImageTk.PhotoImage(file="images/Bg manager.jpg")
left=Label(self.root,image=self.left).place(x=80,y=100,width=400,height
t=500)

#====Registration Frame====
frame1=Frame(self.root,bg="white")
frame1.place(x=480,y=100,width=800,height=500)

title=Label(frame1,text="REGISTER HERE", font=("times new
roman",20,"bold"),bg="white",fg="Blue").place(x=50,y=30)

#-----Row1

f_name=Label(frame1,text="First Name", font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=50,y=100)
self.txt_fname=Entry(frame1,font=("times new
roman",15),bg="lightgray")
self.txt_fname.place(x=50,y=130,width=250)

l_name=Label(frame1,text="Last Name", font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=370,y=100)
self.txt_lname=Entry(frame1,font=("times new
roman",15),bg="lightgray")
self.txt_lname.place(x=370,y=130,width=250)

#-----Row2

contact=Label(frame1,text="Contact No.", font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=50,y=170)
self.txt_contact=Entry(frame1,font=("times new
roman",15),bg="lightgray")
self.txt_contact.place(x=50,y=200,width=250)

email=Label(frame1,text="Email", font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=370,y=170)
self.txt_email=Entry(frame1,font=("times new
roman",15),bg="lightgray")
self.txt_email.place(x=370,y=200,width=250)

#-----Row3

```

```

        question=Label(frame1,text="Security Question", font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=50,y=240)

        self.cmb_quest=ttk.Combobox(frame1,font=("times new
roman",13),state='readonly',justify=CENTER)
        self.cmb_quest['values']=("Select","Your First Pet Name","Your Birth
Place","Your Best Friend Name")
        self.cmb_quest.place(x=50,y=270,width=250)
        self.cmb_quest.current(0)

        answer=Label(frame1,text="Answer", font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=370,y=240)
        self.txt_answer=Entry(frame1,font=("times new
roman",15),bg="lightgray")
        self.txt_answer.place(x=370,y=270,width=250)

#-----Row2

def generateFunction(entry1: Entry,entry2: Entry):

    smallLetters="abcdefghijklmnopqrstuvwxyz"
    capitalLetters="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    numbers="0123456789"
    specialCharacters=' '!@#$%^&*_-+=:;"/?><,. '''
    password_list=[]
    for i in range(10):
        a = random.randint(1,4)
        if a==1:
            password_list.append(smallLetters[random.randint(0,len(sma
llLetters)-1)])
        elif a==2:
            password_list.append(capitalLetters[random.randint(0,len(c
apitalLetters)-1)])
        elif a==3:
            password_list.append(numbers[random.randint(0,len(numbers)
-1)])
        else:
            password_list.append(specialCharacters[random.randint(0,le
n(specialCharacters)-1)])

    global password_string
    password_string = "".join(password_list)

```

```

entry1.delete(0,END)
entry1.insert(0,password_string)
entry2.delete(0,END)
entry2.insert(0,password_string)

password=Label(frame1,text="Password", font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=50,y=310)
self.txt_password=Entry(frame1,font=("times new
roman",15),bg="lightgray")
self.txt_password.place(x=50,y=340,width=250)

cpassword=Label(frame1,text=" Confirm Password", font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=370,y=310)
self.txt_cpassword=Entry(frame1,font=("times new
roman",15),bg="lightgray")
self.txt_cpassword.place(x=370,y=340,width=250)

Button(self.root,text="Generate Password",command=lambda:
generateFunction(self.txt_password,self.txt_cpassword ),font=("Times new
roman",8), bd=0,cursor="hand2",background="Pink").place(x=1150,y=440,
width=90)

#-----Terms-----
self.var_chk=IntVar()
chk=Checkbutton(frame1,text="I Agree the Terms & Conditions"
,variable=self.var_chk,onvalue=1,offvalue=0, bg="white" ,font=("times new
roman", 12) ).place(x=50,y=380)

self.btn_img=ImageTk.PhotoImage(file="images/Button.png")
btn_register=Button(frame1,image=self.btn_img,
bd=0,cursor="hand2",command=self.register_data).place(x=250,y=420)

btn_login=Button(self.root,text="Sign
In",command=self.login_window,font=("Times new roman",20),
bd=0,cursor="hand2",background="Pink").place(x=200,y=540, width=180)

def login_window(self):
    self.root.destroy()
    import login

def clear(self):
    self.txt_fname.delete(0,END)

```



```

self.txt_lname.delete(0,END)
self.txt_contact.delete(0,END)
self.txt_email.delete(0,END)
self.txt_answer.delete(0,END)
self.txt_password.delete(0,END)
self.txt_cpassword.delete(0,END)
self.cmb_quest.current(0)

def register_data(self):
    if self.txt_fname.get()==" " or self.txt_contact.get()==" " or
self.txt_email.get()==" " or self.cmb_quest.get()=="Select" or
self.txt_answer.get()==" " or self.txt_password.get()==" " or
self.txt_cpassword.get()==" ":
        messagebox.showerror("Error","All Fields Are
Required",parent=self.root)
    elif self.txt_password.get()!= self.txt_cpassword.get():
        messagebox.showerror("Error","Password and confirm password should
be same",parent=self.root)
    elif self.var_chk.get()==0:
        messagebox.showerror("Error","Please Agree our Terms and
Conditions",parent=self.root)
    else:
        try:
            con=pymysql.connect(host="localhost",user="root",password="",d
atabase="employee1")
            cur=con.cursor()
            cur.execute("select * from employee where email=%s",
self.txt_email.get())
            row=cur.fetchone()
            print(row)
            if row!=None:
                messagebox.showerror("Error","User already Exists,Please
try with another email",parent=self.root)
            else:
                cur.execute("insert into
employee(f_name,l_name,contact,email,question,answer,password)
values(%s,%s,%s,%s,%s,%s,%s)",
                    ( self.txt_fname.get(),
                      self.txt_lname.get(),
                      self.txt_contact.get(),
                      self.txt_email.get(),
                      self.cmb_quest.get(),
                      self.txt_answer.get(),
                      self.txt_password.get()

```

```

        ) )
        con.commit()
        con.close()
        messagebox.showerror("Error", "Register
Successfull",parent=self.root)
        self.clear()
    except Exception as es:
        messagebox.showerror("Error",f"Error due to:
{str(es)}",parent=self.root)

root=Tk()
obj=Register(root)
root.mainloop()

```

Login Page:-

```

import tkinter as tk
from PIL import Image, ImageTk
import pymysql
from tkinter import CENTER, END, Button, Entry, Label, Tk, Toplevel,
messagebox,ttk
class Register:
    def __init__(self,root):
        self.root=root
        self.root.title("Login Window")
        self.root.geometry("400x400")

        #===Bg Image===
        self.bg=ImageTk.PhotoImage(file="images/BG login.jpg")
        bg=tk.Label(self.root,image=self.bg).place(x=0,y=0,relwidth=1,relheight=1)

        #===Full Window Size===
        self.root.state('zoomed')

```

```
#=====Frames=====
login_frame1=tk.Frame(self.root,bg="cornsilk2")
login_frame1.place(x=350,y=200,width=800,height=500)

title=tk.Label(login_frame1,text="LOGIN HERE", font=("times new
roman",30,"bold"),bg="cornsilk2",fg="darkolivegreen4").place(x=250,y=50)

email=tk.Label(login_frame1,text="EMAIL ADDRESS", font=("times new
roman",15,"bold"),bg="cornsilk2",fg="cornsilk4").place(x=250,y=150)
self.txt_email=tk.Entry(login_frame1, font=("times new
roman",15),bg="lightgray")
self.txt_email.place(x=250,y=180,width=350,height=35)

pass_=tk.Label(login_frame1,text="PASSWORD", font=("times new
roman",15,"bold"),bg="cornsilk2",fg="cornsilk4").place(x=250,y=250)
self.txt_pass_=tk.Entry(login_frame1, font=("times new
roman",15),bg="lightgray")
self.txt_pass_.place(x=250,y=280,width=350,height=35)

btn_reg=tk.Button(login_frame1,cursor="hand2",command=self.register_wi
ndow,text="Register new Account?",font=("times new
roman",14),bg="cornsilk2",bd=0,fg="darkolivegreen4").place(x=250,y=330)

btn_forget=tk.Button(login_frame1,cursor="hand2",command=self.forget_p
assword_window,text="Forget Password?",font=("times new
roman",14),bg="cornsilk2",bd=0,fg="darkolivegreen4").place(x=450,y=330)

btn_login=tk.Button(login_frame1,text="Login",command=self.login,font=
("times new
roman",14),fg="cornsilk2",bg="darkolivegreen4",cursor="hand2").place(x=250,y=3
80,width=180,height=40)

def reset(self):
    self.cmb_quest.current(0)
    self.txt_new_pass.delete(0,END)
    self.txt_answer.delete(0,END)
    self.txt_pass_.delete(0,END)
    self.txt_email.delete(0,END)

def forget_password(self):
    if self.cmb_quest.get()=="Select" or self.txt_answer.get()==" " or
self.txt_new_pass.get()=="":

```



```

        messagebox.showerror("Error","All fields are
required",parent=self.root2)
    else:
        try:
            con=pymysql.connect(host="localhost",user="root",password="",d
atabase="employee1")
            cur=con.cursor()
            cur.execute("select * from employee where email=%s and
question=%s and answer=%s
",(self.txt_email.get(),self.cmb_quest.get(),self.txt_answer.get()))
            row=cur.fetchone()
            print(row)
            if row==None:

                messagebox.showerror("Error","Please Select the Correct
Security Question / Enter Answer",parent=self.root2)
            else:
                cur.execute("update employee set password=%s where
email=%s ",(self.txt_new_pass.get(),self.txt_email.get()))
                con.commit()
                con.close()
                messagebox.showinfo("Success","Your password has been
reset,Please login with new password",parent=self.root2)
                self.reset()
                self.root2.destroy()
        except Exception as es:
            messagebox.showerror("Error",f"Error due to:
{str(es)}",parent=self.root)

    def forget_password_window(self):
        if self.txt_email.get()=="":
            messagebox.showerror("Error","Please enter the email Address to
reset your password",parent=self.root)
        else:
            try:
                con=pymysql.connect(host="localhost",user="root",password="",d
atabase="employee1")
                cur=con.cursor()
                cur.execute("select * from employee where email=%s
",(self.txt_email.get()))
                row=cur.fetchone()
                print(row)
                if row==None:

```

```

        messagebox.showerror("Error","Please enter the valid email
Address to reset your password",parent=self.root)
    else:

        con.close()
        self.root2=Toplevel()
        self.root2.title("Forget Password")
        self.root2.geometry("400x470+510+150")
        self.root2.config(bg="cornsilk2")
        self.root2.focus_force()
        self.root2.grab_set()

        t=Label(self.root2,text="Forget Password",font=("times new
roman",20,"bold"),bg="cornsilk2",fg="darkolivegreen4").place(x=0,y=10,relwidth
=1)

        #-----Forget Password

        question=Label(self.root2,text="Security Question",
font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=50,y=100)

        self.cmb_quest=ttk.Combobox(self.root2,font=("times new
roman",13),state='readonly',justify=CENTER)
        self.cmb_quest['values']=("Select","Your First Pet
Name","Your Birth Place","Your Best Friend Name")
        self.cmb_quest.place(x=50,y=130,width=250)
        self.cmb_quest.current(0)

        answer=Label(self.root2,text="Answer", font=("times new
roman",15,"bold"),bg="white",fg="gray").place(x=50,y=180)
        self.txt_answer=Entry(self.root2,font=("times new
roman",15),bg="lightgray")
        self.txt_answer.place(x=50,y=210,width=250)

        new_password=Label(self.root2,text="New Password",
font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=50,y=260)
        self.txt_new_pass=Entry(self.root2,font=("times new
roman",15),bg="lightgray")
        self.txt_new_pass.place(x=50,y=290,width=250)

        btn_change_password=Button(self.root2,text="Reset
Password",command=self.forget_password,bg="green",fg="white",font=("times new
roman",15,"bold")).place(x=90,y=340)

    except Exception as es:

```

```

        messagebox.showerror("Error",f"Error due to:
{str(es)}",parent=self.root)

def register_window(self):
    self.root.destroy()
    import register

def login(self):
    if self.txt_email.get()=="6" or self.txt_pass_.get()=="":
        messagebox.showerror("Error","All fields are
required",parent=self.root)
    else:
        try:
            con=pymysql.connect(host="localhost",user="root",password="",d
atabase="employee1")
            cur=con.cursor()
            cur.execute("select * from employee where email=%s and
password=%s",(self.txt_email.get(),self.txt_pass_.get()))
            row=cur.fetchone()
            print(row)
            if row==None:
                messagebox.showerror("Error","Invalid USERNAME &
PASSWORD",parent=self.root)

            else:
                messagebox.showerror("Success","Welcome",parent=self.root)
                con.close()
        except Exception as es:
            messagebox.showerror("Error",f"Error due to:
{str(es)}",parent=self.root)

root=tk.Tk()
obj=Register(root)
root.mainloop()

```