

Shri Vile Parle Kelavani Mandal's
**SHRI BHAGUBHAI MAFATLAL
POLYTECHNIC**

July 24, 2021

Program : **Information Technology**
Course Name : **Programming in Python**
Course Code : **PRP198918**
Semester : **IV**

Title : **Bank Management System**
Student Name: **Durva Haresh Patel**
Student Roll No: **1991036**
SAP ID: **57498190036**

Index

| Sr no. | Topic | Page No. |
|---------------|--------------------------------|-----------------|
| 1 | Abstract | 3 |
| 2 | Problem Statement and features | 3 |
| 3 | Software Requirements | 4 |
| 4 | Hardware Requirements | 5 |
| 5 | Module Implementation | 5 |
| 6 | Mini Project Source Code | 6-10 |
| 7 | Mini Project Result | 11-16 |
| 8 | Conclusion | 17 |
| 9 | References | 17 |

Abstract

My Mini Project titled "Bank Management System" is a software for monitoring and controlling the Transactions in a Bank. This Project is designed coded in Sublime Text Editor and database Management is handled by PHP MyAdmin, a free software tool written in PHP intended to handle the administration of MySQL and MariaDB. This Software mainly focuses on basic operations like to provide User and Admin Login, creating and closing accounts and withdraw and deposit money. My Project is easy to use for both beginners and advanced users. The user interface which is very interactive is designed by using the concept of Tkinter available in python.

Problem Statement and Features:

Develop a Python Mini Project for Bank Management System which offers the following features:

- Admin Login
- Customer Login
- Create Account.
- Delete Account.
- All account holder list
- Deposit and Withdraw Amount and check Balance .

SOFTWARE REQUIREMENTS

1. Python 3.9.3



Fig. 1 Python Logo

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

MySQL is easy to use. It is secure and consist of a solid data security layer ta protects sensitive data from intruders. Client Server Architecture, free to download, it is scalable ,speed and high flexibility.

XAMPP has he ability to serve web pages on he World Wide Wed. A speial tool is provided to password protect the most important parts the package. XAMPP also provides support for creating and manipulating databases in MariaDB and SQLite among others

Sublime Text Editor Sublime Text is a shareware cross-platform source code editor with a Python application programming interface (API). It natively supports many programming languages and markup languages, and functions can be added by users with plugins, typically community-built and maintained under free-software licenses

HARDWARE REQUIREMENTS

- Windows 7 or higher (32bit or 64bit)
- Intel i3 processor (1.30 GHz)
- Minimum 4GB RAM
- 250GB Free Disk Space

Module Implementation

| Sr no. | Module Name | Description | Implementation date |
|--------|--|---|---------------------|
| 1 | Defining the Requirements | Collecting all information needed for implementation of a Bank Management System | 12-06-2021 |
| 2 | Designing on paper | Basic Structure or an outline of the whole project on paper. | 15-06-2021 |
| 3 | Designing the Frames practically. | Design the Frames using Python Tkinter on an Editor without implementation. | 17-06-2021 |
| 4 | Login For Admin or Customer implementation | This Frame enables us to choose our role whether you are a customer Bank Executive | 21-06-2021 |
| 5 | Login Form for Admin implementation | Login Form for admin enables the user to login to the Project as admin and enables the connection to Database.. | 24-06-2021 |
| 6 | Login Form for Customer implementation | Login Form for customer enables the user to login to the Project as customer and enables the connection to Database. | 29-06-2021 |
| 7 | Admin Select implementation | The Admin select Menu Frame contains the all the options used by the admin to access the further Frames or options. | 02-07-2021 |
| 8 | Customer Select implementation | The customer select Menu Frame contains the all the options used by the customer to access the further Frames or options. | 05-07-2021 |
| 9 | Create Account implementation | The Create Account form is used create a new account which can be done b the admin and store the information in the database. | 08-07-2021 |
| 10 | Delete Account implementation | The Delete Account form is used to close or delete a particular account | 10-07-2021 |
| 11 | All Account Holders List implementation | Displays the records of people who have an account in the bank . | 18-07-2021 |
| 12 | Transaction implementation | Transaction form contains deposit, withdraw and . check balance | 21-07-2021 |

Mini Project Source Code

Miniproject

```

1 from tkinter import *
2 from tkinter import messagebox
3 import mysql.connector
4
5 con = mysql.connector.connect(
6     host="localhost",
7     user="root",
8     password="",
9     database="bank"
10 )
11 cur = con.cursor()
12
13 class Main:
14     def __init__(self, root):
15         self.root = root
16         self.root.geometry("450x300")
17         self.root.title("Welcome to SBMP Bank")
18         self.label = Label(root, text="Login Form", font="Times 15 italic")
19         self.label.grid(row=0, column=10, padx=20, pady=10)
20
21         btn1 = Button(root, text="Bank Executive", font="Times 15 bold", bg="LightSkyBlue1", command=self.Admin_Login)
22         btn1.grid(row=5, column=15, padx=10, pady=10)
23
24         btn2 = Button(root, text="Customer", font="Times 15 bold", bg="LightSkyBlue1", command=self.Customer_Login)
25         btn2.grid(row=6, column=15, padx=10, pady=10)
26
27     def Admin_Login(self):
28         self.root.destroy()
29         import AdminLogin
30
31     def Customer_Login(self):
32         self.root.destroy()
33         import CustomerLogin
34
35 root = Tk()
36 obj = Main(root)
37 root.mainloop()
38
39
40
```

Admin Login

```

1 from tkinter import *
2 from tkinter import messagebox
3 import mysql.connector
4 con = mysql.connector.connect(
5     host="localhost",
6     user="root",
7     password="",
8     database="bank"
9 )
10 cur = con.cursor()
11 class AdminLogin:
12     def __init__(self, root):
13         self.root = root
14         self.root.title("Admin Login")
15         self.root.geometry("400x200")
16
17         self.label_label(root, text="Enter Your Credentials:", font="Times 15 italic")
18         self.label.grid(row=0, column=0, sticky=W+E+N+S, padx=20, pady=10)
19
20         self.l1_label(root, text="Admin ID:", font="Times 12")
21         self.l2_label(root, text="Password:", font="Times 12")
22
23         self.l1.grid(row=1, column=0, sticky=W+E+N+S, padx=20, pady=10)
24         self.l2.grid(row=2, column=0, sticky=W+E+N+S, padx=20, pady=10)
25
26         self.e1_entry(root)
27         self.e2_entry(root)
28
29         self.e1.grid(row=1, column=1, sticky=W+E+N+S, padx=20, pady=10)
30         self.e2.grid(row=2, column=1, sticky=W+E+N+S, padx=20, pady=10)
31
32
33         btn1 = Button(root, text="Back", font="Times 10 bold", bg="lightgray", command=self.BackAdmin)
34         btn2 = Button(root, text="Login", font="Times 10 bold", bg="lightgray", command=self.AdminCheck)
35
36         btn1.grid(row=4, column=0, padx=20, pady=10)
37         btn2.grid(row=4, column=1, padx=20, pady=10)
38
39     def AdminCheck(self):
40         if self.e1.get() == "" or self.e2.get() == "":
41             messagebox.showerror("Error", "Please enter all the credentials!!!", parent=self.root)
42         else:
43             try:
44                 cur.execute("select * from adminlogin where AdminUser=? and AdminPass=?", (self.e1.get(), self.e2.get()))
45                 row = cur.fetchone()
46                 if row == None:
47                     messagebox.showerror("Error", "Invalid Username and Password", parent=self.root)
48                 else:
49                     messagebox.showinfo("Welcome", "Login Successfull!!!", parent=self.root)
50                     self.root.destroy()
51                     import AdminPanel
52             except Exception as e:
53                 messagebox.showerror("Error", "Error due to: {str(e)}", parent=self.root)
54
55     def BackAdmin(self):
56         self.root.destroy()
57         import MainProject
58
59 root = Tk()
60 obj = AdminLogin(root)
61 root.mainloop()

```


Customer Login

```

1 from tkinter import *
2 from tkinter import messagebox
3 import mysql.connector
4 con = mysql.connector.connect(
5     host="localhost",
6     user="root",
7     password="",
8     database="bank"
9 )
10 cur = con.cursor()
11 class CustomerLogin:
12     def __init__(self, root):
13         self.root = root
14         self.root.title("Customer Login")
15         self.root.geometry("400x200")
16
17         self.label = Label(root, text="Enter Your Credentials!", font="Times 15 italic")
18         self.label.grid(row=0, column=0, sticky=W+E+N+S, padx=20, pady=10)
19
20         self.l1 = Label(root, text="User ID:", font="Times 12")
21         self.l2 = Label(root, text="Password:", font="Times 12")
22
23         self.l1.grid(row=1, column=0, sticky=W+E+N+S, padx=20, pady=10)
24         self.l2.grid(row=2, column=0, sticky=W+E+N+S, padx=20, pady=10)
25
26         self.e3 = Entry(root)
27         self.e4 = Entry(root)
28
29         self.e3.grid(row=1, column=1, sticky=W+E+N+S, padx=20, pady=10)
30         self.e4.grid(row=2, column=1, sticky=W+E+N+S, padx=20, pady=10)
31
32         btnb1 = Button(root, text="Back", font="Times 10 bold", bg="lightskyblue1", command=self.Backcut)
33         btnb1.grid(row=3, column=0, sticky=W+E+N+S, padx=20, pady=10)
34
35         btnb2 = Button(root, text="Login", font="Times 10 bold", bg="lightskyblue1", command=self.CustomerCheck)
36         btnb2.grid(row=3, column=1, sticky=W+E+N+S, padx=20, pady=10)
37
38         def CustomerCheck(self):
39             if self.e3.get() == "" or self.e4.get() == "":
40                 messagebox.showerror("Error", "Please enter all the credentials!!!", parent=self.root)
41             else:
42                 try:
43                     cur.execute("select * from customer where CustUserNo and CustPassNo", (self.e3.get(), self.e4.get()))
44                     row = cur.fetchone()
45                     if row:
46                         messagebox.showerror("Error", "Invalid Username and Password", parent=self.root)
47                     else:
48                         messagebox.showinfo("Welcome", "Login Successfull!!!", parent=self.root)
49                         self.root.destroy()
50                         import CustomerSelect
51                         except Exception as es:
52                             messagebox.showerror("Error", "Error due to: {str(es)}", parent=self.root)
53
54         def Backcut(self):
55             self.root.destroy()
56             import Miniproject
57
58 root = Tk()
59 obj = CustomerLogin(root)
60 root.mainloop()

```

Admin Select

```

1 from tkinter import *
2 from tkinter import messagebox
3 import mysql.connector
4 con = mysql.connector.connect(
5     host="localhost",
6     user="root",
7     password="",
8     database="bank"
9 )
10 cur = con.cursor()
11 class AdminSelect:
12     def __init__(self, root):
13         self.root = root
14         self.root.title("Admin")
15         self.root.geometry("350x350")
16
17         self.label = Label(root, text="Select Options ", font="Times 15 italic")
18         self.label.grid(row=0, column=0, padx=20, pady=10)
19
20         btnb1 = Button(root, text="Create New Bank Account", font="Times 12", bg="lightskyblue1", command=self.Create_Account)
21         btnb2 = Button(root, text="Close Bank Account", font="Times 12", bg="lightskyblue1", command=self.Close_Account)
22         btnb3 = Button(root, text="All Account Holder List", font="Times 12", bg="lightskyblue1", command=self.ShowData)
23         btnb4 = Button(root, text="Exit", font="Times 12", bg="lightskyblue1", command=self.Exit_AdminSelect)
24
25         btnb1.grid(row=1, column=0, padx=100, pady=10)
26         btnb2.grid(row=2, column=0, padx=100, pady=10)
27         btnb3.grid(row=3, column=0, padx=100, pady=10)
28         btnb4.grid(row=4, column=0, padx=100, pady=10)
29
30         def Create_Account(self):
31             self.root.destroy()
32             import CreateAccount
33
34         def Close_Account(self):
35             self.root.destroy()
36             import CloseAccount
37
38         def ShowData(self):
39             self.root.destroy()
40             import ShowData
41
42         def Exit_AdminSelect(self):
43             self.root.destroy()
44             import AdminLogin
45
46 root = Tk()
47 obj = AdminSelect(root)
48 root.mainloop()

```

Customer Select

The screenshot displays a Python Tkinter application running in a Sublime Text editor. The code defines a `CustomerSelect` class that interacts with a MySQL database and manages a Tkinter window.

```
1 from tkinter import *
2 from tkinter import messagebox
3 import mysql.connector
4
5 con = mysql.connector.connect(
6     host="localhost",
7     user="root",
8     password="",
9     database="bank"
10 )
11 cur = con.cursor()
12
13 class CustomerSelect:
14     def __init__(self, root):
15         self.root = root
16         self.root.title("Customer")
17         self.root.geometry("300x250")
18
19         self.Label_Label(root, text="Select Option:", font="Times 15 italic")
20         self.Label.grid(row=0, column=0, sticky=W, padx=20, pady=10)
21
22         btn1 = Button(root, text="Transaction", font="Times 12", bg="LightSkyBlue1", command=self.Withdraw_Deposit, width=10)
23         btn4 = Button(root, text="Exit", font="Times 12", bg="LightSkyBlue1", command=self.Exit_CustomerSelect, width=10)
24
25
26         btn1.grid(row=4, column=0, padx=100, pady=10)
27         btn4.grid(row=5, column=0, padx=100, pady=10)
28
29     def Withdraw_Deposit(self):
30         self.root.destroy()
31         import WithdrawDeposit
32
33     def Exit_CustomerSelect(self):
34         self.root.destroy()
35         import CustomerSelect
36
37 root = Tk()
38 obj = CustomerSelect(root)
39 root.mainloop()
```

The application window, titled "Customer", has a geometry of 300x250 pixels. It features a label "Select Option:" and two buttons: "Transaction" and "Exit". The "Transaction" button is located at row 4, column 0, and the "Exit" button is at row 5, column 0. Both buttons have a light blue background and a font of Times 12. The "Transaction" button is associated with the `Withdraw_Deposit` method, and the "Exit" button is associated with the `Exit_CustomerSelect` method.

The status bar at the bottom indicates the current position is Line 36, Column 1, and the tab is named "Tab Size 4". The system tray shows the date and time as 24-07-2021, 08:47 PM, and the weather as 28°C Rain showers.

Create Account

```
D:\Python\Miniproject>CreateAccount.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

AdminLoginy * CustomLoginy * AdminSelecty * CustomerSelecty * CreateAccounty * CloseAccounty * Miniprojecty * ShowDatay * WithdrawDeposit *

1 from tkinter import *
2 from tkinter import messagebox
3 import mysql.connector
4
5 con = mysql.connector.connect(
6     host='localhost',
7     user='root',
8     password='',
9     database='bank'
10 )
11 cur = con.cursor()
12
13 class CreateAccount:
14     def __init__(self, root):
15         self.root = root
16         self.root.title("Admin")
17         self.root.geometry("450x600")
18         self.label_label(root, text="Enter Details:", font="Times 15 Italic")
19         self.label.grid(row=0, column=0, sticky=W+E+N+S, padx=20, pady=10)
20
21         self.fname_label(root, text="Enter your Full Name:", font="Times 12")
22         self.age_label(root, text="Enter your Age:", font="Times 12")
23         self.add_label(root, text="Enter your Address:", font="Times 12")
24         self.mobno_label(root, text="Enter your Mobile no.:", font="Times 12")
25         self.occp_label(root, text="Enter your Occupation:", font="Times 12")
26         self.accco_label(root, text="Enter Account Number:", font="Times 12")
27         self.actype_label(root, text="Enter Account Type (Current/Saving):", font="Times 12")
28         self.custlogin_label(root, text="Enter Username:", font="Times 12")
29         self.custpass_label(root, text="Enter Pin:", font="Times 12")
30
31         self.fname_grid(row=1, column=0, sticky=W+E+N+S, padx=20, pady=10)
32         self.age_grid(row=2, column=0, sticky=W+E+N+S, padx=20, pady=10)
33         self.add_grid(row=3, column=0, sticky=W+E+N+S, padx=20, pady=10)
34         self.mobno_grid(row=4, column=0, sticky=W+E+N+S, padx=20, pady=10)
35         self.occp_grid(row=5, column=0, sticky=W+E+N+S, padx=20, pady=10)
36         self.accco_grid(row=6, column=0, sticky=W+E+N+S, padx=20, pady=10)
37         self.actype_grid(row=7, column=0, sticky=W+E+N+S, padx=20, pady=10)
38         self.custlogin_grid(row=8, column=0, sticky=W+E+N+S, padx=20, pady=10)
39         self.custpass_grid(row=9, column=0, sticky=W+E+N+S, padx=20, pady=10)
40
41         self.in_fname_Entry(root)
42         self.in_age_Entry(root)
43         self.in_add_Entry(root)
44         self.in_mobno_Entry(root)
45         self.in_occp_Entry(root)
46         self.in_accco_Entry(root)
47         self.in_actype_Entry(root)
48         self.in_bal_Entry(root)
49         self.in_custlogin_Entry(root)
50         self.in_custpass_Entry(root)
```

```
D:\47\Miniproject\CreateAccount.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

AdminLogin.py CustomerLogin.py AdminSelect.py CustomerSelect.py CreateAccount.py CloseAccount.py Miniproject.py ShowData.py WithdrawDeposit.py

55
56
57 self.in_fname.grid(row=1,column=1,sticky=W,E,N,S,padx=20,pady=10)
58 self.in_age.grid(row=2,column=1,sticky=W,E,N,S,padx=20,pady=10)
59 self.in_add.grid(row=3,column=1,sticky=W,E,N,S,padx=20,pady=10)
60 self.in_mobno.grid(row=4,column=1,sticky=W,E,N,S,padx=20,pady=10)
61 self.in_occp.grid(row=5,column=1,sticky=W,E,N,S,padx=20,pady=10)
62 self.in_accto.grid(row=6,column=1,sticky=W,E,N,S,padx=20,pady=10)
63 self.in_acctype.grid(row=7,column=1,sticky=W,E,N,S,padx=20,pady=10)
64 self.in_bal.grid(row=8,column=1,sticky=W,E,N,S,padx=20,pady=10)
65 self.in_custlogin.grid(row=9,column=1,sticky=W,E,N,S,padx=20,pady=10)
66 self.in_custpass.grid(row=10,column=1,sticky=W,E,N,S,padx=20,pady=10)
67
68 save Button(root,text="Save/Create",font="Times 18 bold",bg="LightSkyBlue1",command=self.Create)
69 exit Button(root,text="Exit",font="Times 18 bold",bg="LightSkyBlue1",command=self.Exit_CreateAccount)
70
71 save.grid(row=11,column=0,padx=20,pady=10)
72 exit.grid(row=12,column=0,padx=20,pady=10)
73
74 def create(self):
75     if self.in_fname.get()==" " or self.in_age.get()==" " or self.in_add.get()==" " or self.in_mobno==" " or self.in_occp==" " or self.in_accto==" " or self.in_acctype==" " or self.in_bal==" ":
76         messagebox.showerror("Error","Error All Fields are required",parent=self.root)
77     else:
78         try:
79             cur.execute("insert into customer (FullName,Age,Address,Mobile,Occupation,AccountNo,AccountType,Balance,Customer,CustPass) values (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",
80                 (self.in_fname.get(),
81                 self.in_age.get(),
82                 self.in_add.get(),
83                 self.in_mobno.get(),
84                 self.in_occp.get(),
85                 self.in_accto.get(),
86                 self.in_acctype.get(),
87                 self.in_bal.get(),
88                 self.in_custlogin.get(),
89                 self.in_custpass.get())
90             )
91             con.commit()
92             con.close()
93             messagebox.showinfo("Success","Account Created")
94         except Exception as es:
95             messagebox.showerror("Error","Error due to: {str(es)}",parent=self.root)
96
97 def Exit_CreateAccount(self):
98     self.root.destroy()
99     import AdminSelect
100
101 root Tk()
102 obj = CreateAccount(root)
103 root.mainloop()
104
```

Close Account

```

1 from tkinter import *
2 from tkinter import messagebox
3 import mysql.connector
4
5 con = mysql.connector.connect(
6     host="localhost",
7     user="root",
8     password="",
9     database="bank"
10 )
11 cur = con.cursor()
12
13 class CloseAccount:
14     def __init__(self, root):
15         self.root = root
16         self.root.title("Close Account")
17         self.root.geometry("400x200")
18
19         self.label = Label(root, text="Enter Details: ", font="Times 15 italic")
20         self.label.grid(row=0, column=0, sticky=W, padx=10, pady=10)
21
22         self.acno_label = Label(root, text="Enter your Account Number: ", font="Times 12")
23         self.acno_label.grid(row=1, column=0, sticky=W, padx=10, pady=10)
24         self.in_acno = Entry(root)
25         self.in_acno.grid(row=2, column=0, sticky=W, padx=10, pady=10)
26
27         btn1 = Button(root, text="Delete", font="Times 10 bold", bg="LightSkyBlue1", command=self.Close)
28         btn2 = Button(root, text="Exit", font="Times 10 bold", bg="LightSkyBlue1", command=self.CloseExit)
29
30         btn1.grid(row=3, column=0, padx=20, pady=10)
31         btn2.grid(row=3, column=1, padx=20, pady=10)
32
33     def Close(self):
34         print(self.in_acno.get())
35         print(self.in_acno.get())
36         if self.in_acno.get() == "":
37             messagebox.showerror("Error", "Error all fields are required")
38         else:
39             try:
40                 cur.execute("DELETE from customer where AccountNo = " + self.in_acno.get())
41                 con.commit()
42                 con.close()
43                 messagebox.showinfo("Closed", "Your Account has been closed")
44                 self.root.destroy()
45                 import AdminSelect
46             except Exception as e:
47                 messagebox.showerror("Error", f"Error due to: {str(e)}", parent=self.root)
48
49     def CloseExit(self):
50         pass
51
52 root = Tk()
53 obj = CloseAccount(root)
54 root.mainloop()

```

Show Data

```

1 from tkinter import *
2 from tkinter import messagebox
3 from tkinter import ttk
4 import mysql.connector
5
6 con = mysql.connector.connect(
7     host="localhost",
8     user="root",
9     password="",
10     database="bank"
11 )
12 cur = con.cursor()
13
14 class ShowData:
15     def __init__(self, root):
16         self.root = root
17         self.root.title("List")
18         self.root.geometry("1100x600")
19         cur.execute("select * from customer")
20         self.tree = ttk.Treeview(self.root)
21         self.tree["show"].heading()
22         self.tree["columns"] = ("id", "FullName", "Age", "Address", "Mobile", "Occupation", "AccountNo", "AccountType", "Balance")
23
24         self.tree.column("id", width=100, minwidth=100, anchor=tk.CENTER)
25         self.tree.column("FullName", width=150, minwidth=150, anchor=tk.CENTER)
26         self.tree.column("Age", width=50, minwidth=50, anchor=tk.CENTER)
27         self.tree.column("Address", width=200, minwidth=200, anchor=tk.CENTER)
28         self.tree.column("Mobile", width=100, minwidth=100, anchor=tk.CENTER)
29         self.tree.column("Occupation", width=150, minwidth=150, anchor=tk.CENTER)
30         self.tree.column("AccountNo", width=100, minwidth=100, anchor=tk.CENTER)
31         self.tree.column("AccountType", width=100, minwidth=100, anchor=tk.CENTER)
32         self.tree.column("Balance", width=100, minwidth=100, anchor=tk.CENTER)
33
34         self.tree.heading("id", text="id", anchor=tk.CENTER)
35         self.tree.heading("FullName", text="Full Name", anchor=tk.CENTER)
36         self.tree.heading("Age", text="Age", anchor=tk.CENTER)
37         self.tree.heading("Address", text="Address", anchor=tk.CENTER)
38         self.tree.heading("Mobile", text="Mobile No.", anchor=tk.CENTER)
39         self.tree.heading("Occupation", text="Occupation", anchor=tk.CENTER)
40         self.tree.heading("AccountNo", text="Account Number", anchor=tk.CENTER)
41         self.tree.heading("AccountType", text="Account Type", anchor=tk.CENTER)
42         self.tree.heading("Balance", text="Balance", anchor=tk.CENTER)
43
44         for i in cur:
45             self.tree.insert("", 1, text="", values=(ro[0], ro[1], ro[2], ro[3], ro[4], ro[5], ro[6], ro[7], ro[8]))
46         self.tree.pack()
47         btn = Button(root, text="Exit", font="Times 15 bold", bg="LightSkyBlue1", command=self.ExitData, width=10)
48         btn.pack()
49
50     def ExitData(self):
51         self.root.destroy()
52         import AdminSelect
53
54 root = Tk()
55 obj = ShowData(root)
56 root.mainloop()

```

Transaction

```
D:\ATM\Miniproject\WithdrawDeposit.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

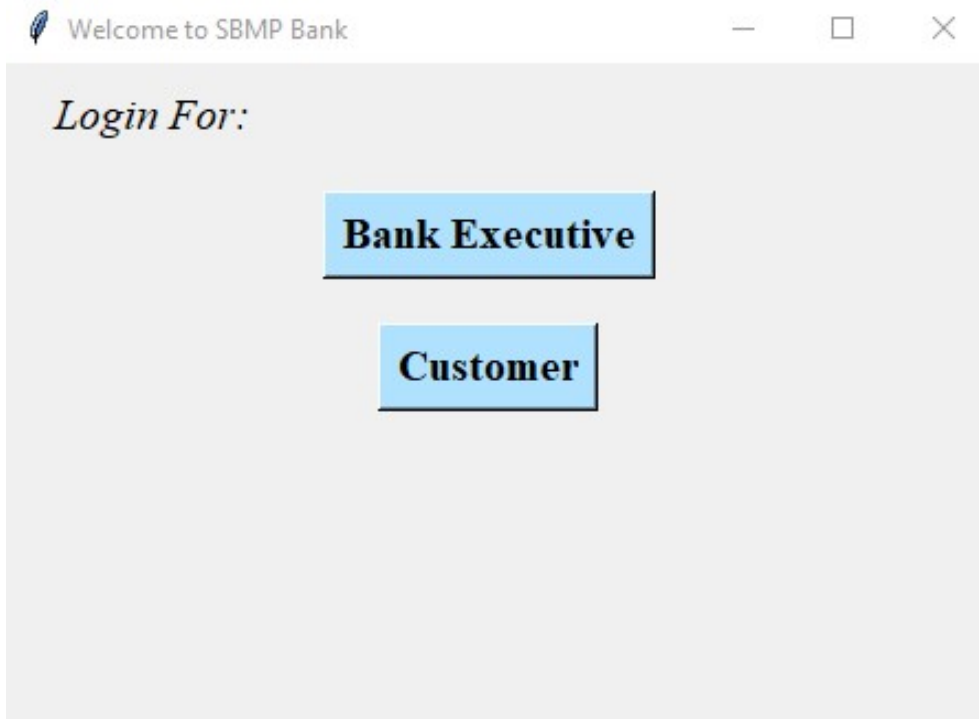
AdminLoginPage.py CustomerLoginPage.py AdminSelect.py CustomerSelect.py CreateAccount.py CloseAccount.py Miniproject.py ShowDelay.py WithdrawDeposit.py

1 from tkinter import *
2 from tkinter import messagebox
3 import mysql.connector
4 class WithdrawDeposit:
5     def __init__(self, root):
6         self.root = root
7         self.root.title("Deposit and Withdraw")
8         self.root.geometry("550x310")
9         self.label = Label(root, text="Perform Transactions:", font="Times 15 italic")
10        self.label.grid(row=0, column=0, sticky=W, padx=20, pady=10)
11        self.depositWithdrawLabel = Label(root, text="Enter Amount to Deposit/Withdraw : ", font="Times 12")
12        self.current_bal = Label(root, text="Current Balance:", font="Times 12")
13        self.depositWithdrawLabel.grid(row=2, column=0, sticky=W, padx=20, pady=10)
14        self.current_bal.grid(row=3, column=0, sticky=W, padx=20, pady=10)
15        self.in_depositWithdrawEntry = Entry(root)
16        self.in_depositWithdrawEntry.grid(row=2, column=1, sticky=W, padx=20, pady=10)
17        self.in_current_balEntry = Entry()
18        self.in_current_bal.grid(row=3, column=1, sticky=W, padx=20, pady=10)
19        btn1 = Button(root, text="Withdraw", font="Times 10 bold", bg="LightSkyBlue1", width=10, command=self.Withdraw)
20        btn2 = Button(root, text="Deposit", font="Times 10 bold", bg="LightSkyBlue1", width=10, command=self.Deposit)
21        btn3 = Button(root, text="Clear", font="Times 10 bold", bg="LightSkyBlue1", width=10, command=self.Clear)
22        btn4 = Button(root, text="Exit", font="Times 10 bold", bg="LightSkyBlue1", width=10)
23        btn1.grid(row=4, column=0, padx=10, pady=10)
24        btn2.grid(row=4, column=1, padx=10, pady=10)
25        btn3.grid(row=4, column=2, padx=10, pady=10)
26        btn4.grid(row=4, column=3, padx=10, pady=10)
27        btn4.grid(row=10, column=2, padx=10, pady=10)
28    def Deposit(self):
29        if self.in_depositWithdraw.get() == "":
30            messagebox.showerror("Error", "Please enter the amount!!", parent=self.root)
31        else:
32            self.dep = float(self.in_depositWithdraw.get())
33            self.result = self.in_current_bal.get()
34            self.in_current_bal.insert(0, str(self.result))
35    def Withdraw(self):
36        self.curr_bal = self.in_current_bal.get()
37        self.withdraw = float(self.in_depositWithdraw.get())
38        if self.curr_bal < self.withdraw:
39            messagebox.showerror("Error", "Not Enough Money to Withdraw", parent=self.root)
40        else:
41            self.result = self.curr_bal - self.withdraw
42            self.in_current_bal.insert(0, str(self.result))
43            self.in_current_bal.config(state=DISABLED)
44    def Clear(self):
45        self.in_depositWithdraw.delete(0, END)
46        self.in_current_bal.delete(0, END)
47
48 root = Tk()
49 obj = WithdrawDeposit(root)
50 root.mainloop()

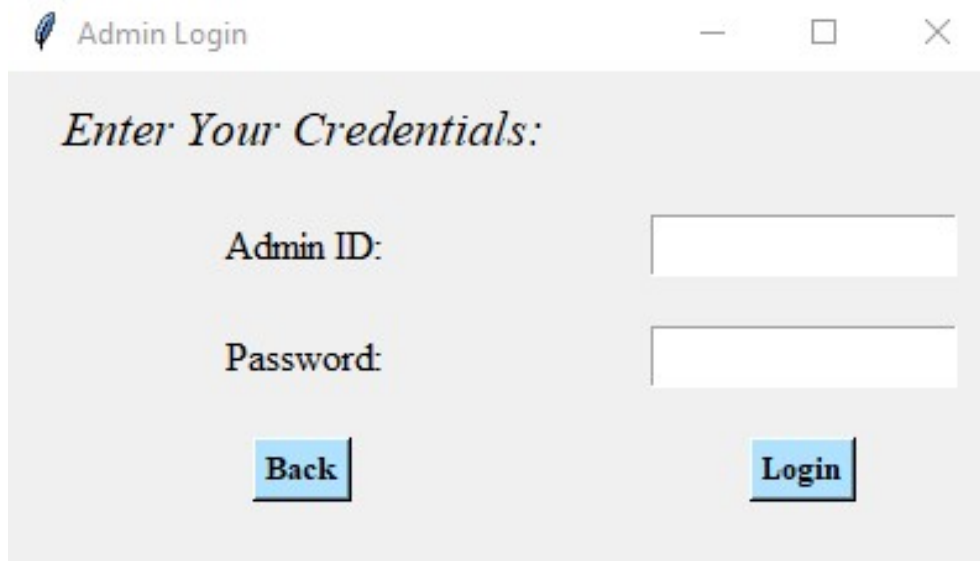
Line 47, Column 5
Tab Size: 4 Python
Type here to search
28°C Rain showers 08:53 PM 24-07-2021
```

Mini Project Result

Select



Admin Login



A screenshot of a web application window titled "Admin Login". The window has a standard OS-style title bar with a feather icon, a minus button, a maximize button, and a close button. The main content area has a light gray background and contains the text *Enter Your Credentials:* in a bold, italicized serif font. Below this text are two labels, "Admin ID:" and "Password:", each followed by a white rectangular input field. At the bottom of the form are two blue buttons with black text: "Back" on the left and "Login" on the right.

Admin Login

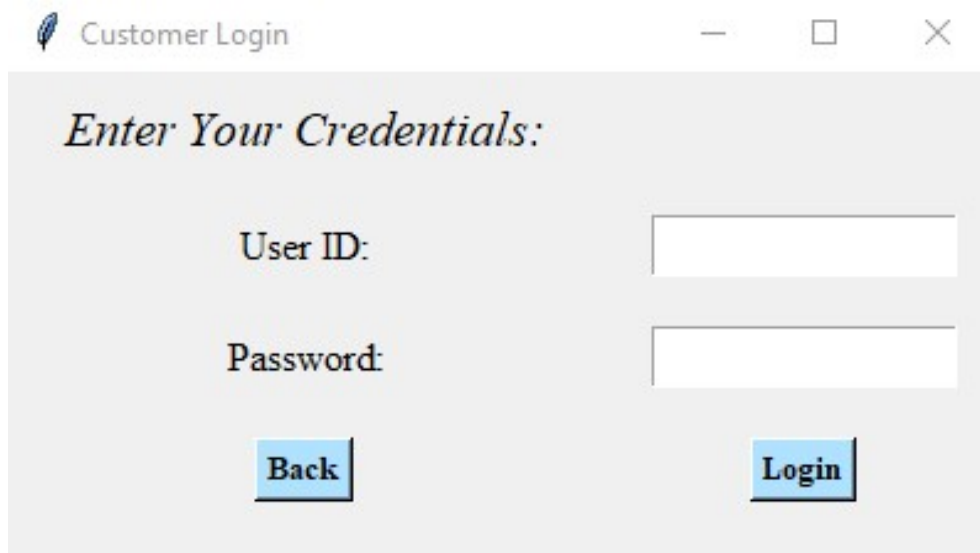
Enter Your Credentials:

Admin ID:

Password:

[Back](#) [Login](#)

Customer Login



A screenshot of a web application window titled "Customer Login". The window has a standard OS-style title bar with a feather icon, a minus button, a maximize button, and a close button. The main content area has a light gray background and contains the text *Enter Your Credentials:* in a bold, italicized serif font. Below this text are two labels, "User ID:" and "Password:", each followed by a white rectangular input field. At the bottom of the form are two blue buttons with black text: "Back" on the left and "Login" on the right.

Customer Login

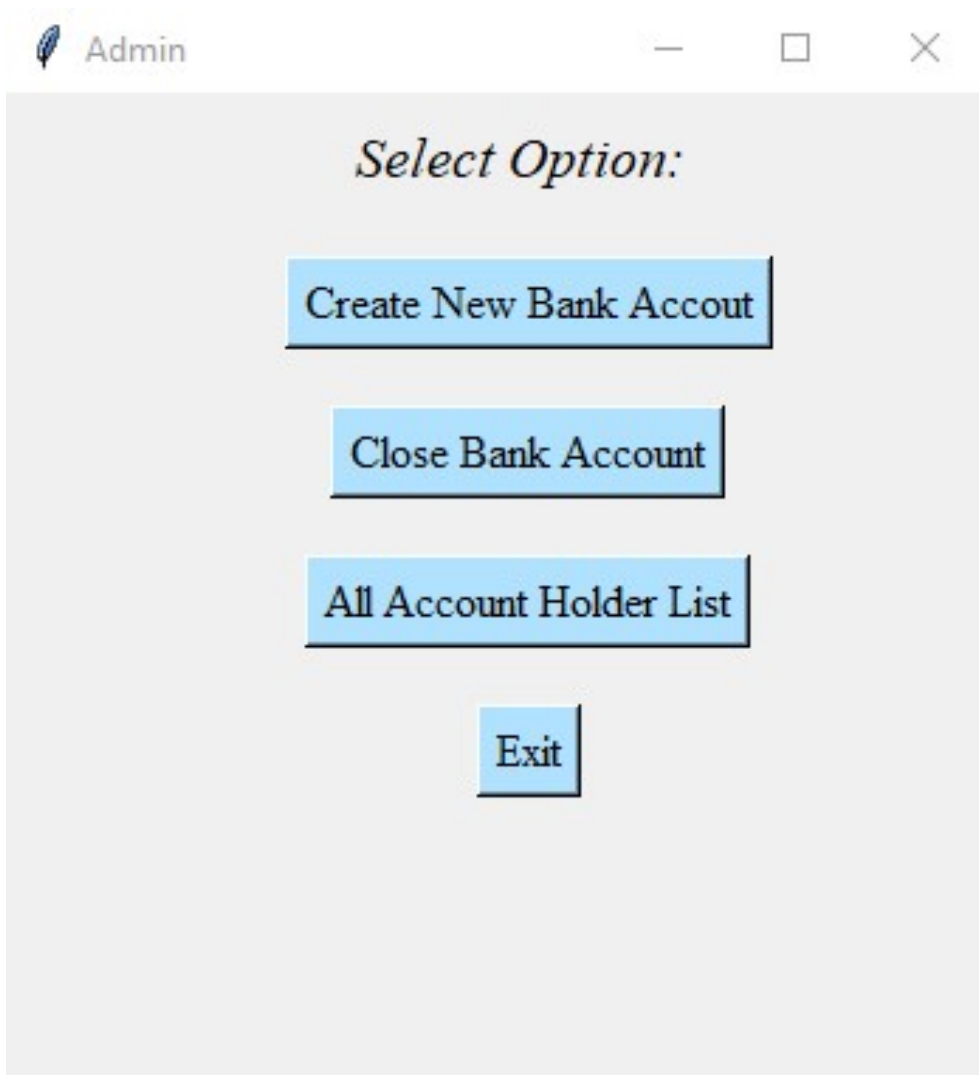
Enter Your Credentials:

User ID:

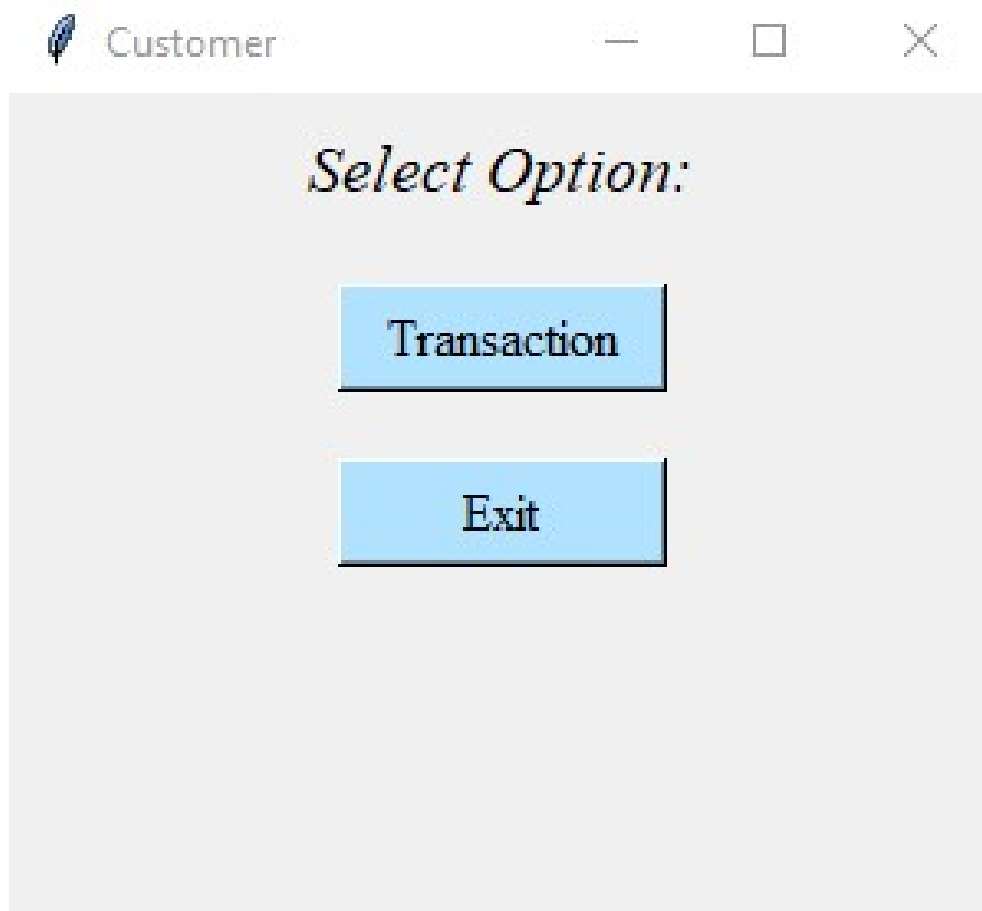
Password:

[Back](#) [Login](#)


Admin Select



Customer Select



Create Account

 Admin—□×

Enter Details:

Enter your Full Name:

Enter your Age:

Enter your Address:

Enter your Mobile no.:

Enter your Occupation:

Enter Account Number:

Enter Account Type (Current/Saving):

Enter Current Balance:


Enter Username:

Enter Pin:

Save/Create

Exit

Close Account

 Close Account — □ ×


Enter Details:

Enter your Account Number:

Delete

Exit


Show Data

 List — □ ×

| ID | Full Name | Age | Address | Mobile No. | Occupation | Account Num | Account Type | Balance |
|----|----------------|-----|------------------------------------|------------|--------------------|-------------|--------------|------------|
| 1 | Joe Woodson | 50 | C/19, Sunshine Appartment, Andher | 9856378292 | Government Servent | 1001 | Savings | 200000.0 |
| 2 | Serena Smith | 31 | A/5, Rose Villa, Goregoan(West) | 9846502711 | Bank Executive | 1002 | Savings | 3000000.0 |
| 3 | Lily Smith | 50 | B/12, Sea View Appartments, Bandra | 9853621937 | Employee | 1003 | Savings | 350000.0 |
| 4 | Sarah Jade | 35 | C/16,Diwan Mension, Bandra(East) | 8794536729 | Bank Executive | 1004 | Savings | 5000000.0 |
| 5 | Charls Winston | 33 | D/13,Amazon,Vile Parle(West) | 8790345279 | Businessmen | 1005 | Current | 50000000.0 |

Exit

Transaction

 Deposit and Withdraw

—

□

×

Perform Transactions:

Enter Amount to Diposit/Withdraw :

Current Balance:

Withdraw

Deposit

Clear

Exit

Conclusion :

Through this Mini Project we learnt about the basics functionalities of Bank Management System. We learnt about the concept designing of Tkinter in python , through which we designed all the frames in the Mini Project. We also learnt about addition of database connectivity in our Mini Project, We use the Login form to connect our project to database. My Mini Project is a simple and easy to understand project which could be understood by a beginner and and also a advanced user.

References :

- <https://projectworlds.in/python-projects-with-source-code/bank-management-system-project-in-python/>
- <https://www.geeksforgeeks.org/python-program-to-create-bankaccount-class-with-deposit-withdraw-function/>
- <https://www.youtube.com/watch?v=sq76a19jAvQt=1185s>