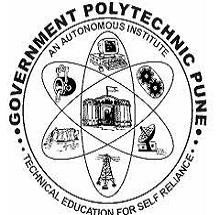
# Government Polytechnic, Pune-16

**(An Autonomous Institute of Government of Maharashtra)**



**A**

**MICROPROJECT REPORT**

**ON**

**“Bank Management System”**

## SUBMITED BY:

## Dhanashri Ghadage-1907020

**Sonali Gade - 1907016**

**Prerna Divekar-1907014**

**Under the Guidance of**

**Smt Kiran Gaikwad**

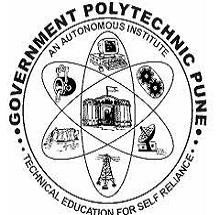
**DEPARTMENT OF INFORMATION TECHNOLOGY**

**(Academic Year: 2020-21)**

**Government Polytechnic, Pune-16**

**(An Autonomous Institute of Government of Maharashtra)**

**Department Information Technology**



**CERTIFICATE**

This is to certify that **Dhanashri Arjun Ghadage (1907020),Sonali Gade(1907016),Prerna Divekar(1907014)** of Third Year Diploma in Information Technology has successfully completed Micro project titled **“Bank Management System”** as part of his diploma curriculum of Course Operating System in year 2020-21.

|  |  |  |
| --- | --- | --- |
| **Project Guide**  **(Mrs. Kiran Gaikwad)** | **H.O.D**  **(Mrs. M. U. Kokate**) | **Principal**  **(Dr. V. S. Bandal)** |

**ACKNOWLEDGEMENT**

It is my proud privilege and duty to acknowledge the kind of help and guidance received from several people in the preparation of this report. It would not have been possible to prepare this report in this form without their valuable help, cooperation and guidance.

Firstandforemost,IwishtorecordmysinceregratitudetotheManagementofthiscollegeand to our Respected Principal Dr. V. S. Bandal sir, for his constant support and encouragement in the preparation of this report and for the availability of library and laboratory facilities needed to prepare this report.

My sincere thanks to Mrs, M. U. Kokate ma’am, Head of department, Information Technology, Government Polytechnic, Pune for her valuable suggestions and guidance throughout the preparation of this report.

I express my sincere gratitude to my Guide, Mrs.Kiran Gaikwad ma’am,for guiding us in investigations of this seminar and in carrying out experimental work. Our numerous discussions were extremely helpful. I hold her in esteem for guidance, encouragement and inspiration received from her.

Last but not the least I wish to thank my parents for financing my studies and helping me throughout my life for achieving perfection and excellence. Their personal help in making this report and seminar worth presentation is gratefully acknowledged.

**Dhanashri Ghadage**

**(1907020)**

**Sonali Gade**

**(1907016)**

**Prerna Divekar**

**(1907014)**

**Index**

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **Contents** | **Pg.No.** |
| **1.** | **Abstract** | 5 |
| **2.** | **Introduction** | 6 |
| **3.** | **Description** | 8-10 |
| **4.** | **Code** | 11-13 |
| **5.** | **Implementation** | 14-25 |
| **6.** | **Outcome Achieves** | 26 |
| **7.** | **Conclusion** | 27 |
| **8.** | **References** | 28 |

**Abstract**

The content of an abstract must answer or address the needs of every issue happen in bank management system project. These issues could be lacking of security in manual managing of bank accounts or to address the efficiency of banking transactions by making the management system an online project. For example is:

To create a project for resolving a customer’s financial applications in a banking environment in order to meet the needs of an end banking user by giving multiple ways to accomplish banking chores. Also, to provide additional features to the user’s work space that aren’t available in a traditional banking project.

This project abstract for Bank Management System is important because you can foresee the function of the system upon creating it.

**Introduction**

The “Bank Account Management System” project is a model Internet Banking Site. This site enables the customers to perform the basic banking transactions by sitting at their office or at homes through PC or laptop. The system provides the access to the customer to create an account, deposit/withdraw the cash from his account, also to view reports of all accounts present. The customers can access the banks website for viewing their Account details and perform the transactions on account as per their requirements. With Internet Banking, the brick and mortar structure of the traditional banking gets converted into a click and portal

model, thereby giving a concept of virtual banking a real shape. Thus today's banking is no longer confined to branches. E-banking facilitates banking transactions by customers round

the clock globally. The primary aim of this “Bank Account Management System” is to provide an improved design methodology, which envisages the future expansion, and modification, which is necessary for a core sector like banking. This necessitates the design to be expandable and

modifiable and so a modular approach is used in developing the application software. Anybody who is an Account holder in this bank can become a member of Bank Account Management System. He has to fill a form with his personal details and Account Number. Bank is the place where customers feel the sense of safety for their property. In the bank,customers deposit and withdraw their money. Transaction of money also is a part where customer takes shelter of the bank. Now to keep the belief and trust of customers, there is the

positive need for management of the bank, which can handle all this with comfort and ease. Smooth and efficient management affects the satisfaction of the customers and staff

members, indirectly. And of course, it encourages management committee in taking some needed decision for future enhancement of the bank. Now a day’s, managing a bank is tedious job up to certain limit. So software that reduces the work is essential. Also today’s world is a genuine computer world and is getting faster and faster day-by-day. Thus, considering above necessities, the software for bank management has became necessary which would be useful in managing the bank more efficiently. All transactions are carried out online by transferring from accounts in the same Bank or international bank. The software is meant to overcome the drawbacks of the manual system.

**Description**

# Python GUI – tkinter

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.  
**To create a tkinter app:**

1. **Importing the module – tkinter**
2. **Create the main window (container)**
3. **Add any number of widgets to the main window**
4. **Apply the event Trigger on the widgets.**

There are two main methods used which the user needs to remember while creating the Python application with GUI.

**Tk(screenName=None,  baseName=None,  className=’Tk’, useTk=1):**

To create a main window, tkinter offers a method ‘Tk(screenName=None,  baseName=None,  className=’Tk’,  useTk =1)’. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

m=tkinter.Tk()

where m is the name of the main window object

**2.mainloop():** There is a method known by the name mainloop() is used when your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

m.mainloop()

tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes class.

**pack() method:**It organizes the widgets in blocks before placing in the parent widget.

**grid() method:**It organizes the widgets in grid (table-like structure) before placing in the parent widget.

**place() method:**It organizes the widgets by placing them on specific positions directed by the programmer.

There are a number of widgets which you can put in your tkinter application. Some of the major widgets are explained below:

**1.Button**:To add a button in your application, this widget is used.  
The general syntax is:

w=Button(master, option=value)

master is the parameter used to represent the parent window.  
There are number of options which are used to change the format of the Buttons. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* **activebackground**: to set the background color when button is under the cursor.
* **activeforeground**: to set the foreground color when button is under the cursor.
* **bg**: to set he normal background color.
* **command**: to call a function.
* **font**: to set the font on the button label.
* **image**: to set the image on the button.
* **width**: to set the width of the button.
* **height**: to set the height of the button.

**2.Canvas:**It is used to draw pictures and other complex layout like graphics, text and widgets.  
The general syntax is:

w = Canvas(master, option=value)

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* **bd**: to set the border width in pixels.
* **bg**: to set the normal background color.
* **cursor**: to set the cursor used in the canvas.
* **highlightcolor**: to set the color shown in the focus highlight.
* **width**: to set the width of the widget.
* **height**: to set the height of the widget.

|  |
| --- |
| **from** tkinter **import** **\*** |

**3.Entry:**It is used to input the single line text entry from the user.. For multi-line text input, Text widget is used.  
The general syntax is:

w=Entry(master, option=value)

master is the parameter used to represent the parent window.  
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* **bd**: to set the border width in pixels.
* **bg**: to set the normal background color.
* **cursor**: to set the cursor used.
* **command**: to call a function.
* **highlightcolor**: to set the color shown in the focus highlight.
* **width**: to set the width of the button.
* **height**: to set the height of the button.

**4.Frame:** It acts as a container to hold the widgets. It is used for grouping and organizing the widgets. The general syntax is:

w = Frame(master, option=value)

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* **highlightcolor**: To set the color of the focus highlight when widget has to be focused.
* **bd**: to set the border width in pixels.
* **bg**: to set the normal background color.
* **cursor**: to set the cursor used.
* **width**: to set the width of the widget.
* **height**: to set the height of the widget.

# Python: Pillow (a fork of PIL)

Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language. It incorporates lightweight image processing tools that aids in editing, creating and saving images. Support for Python Imaging Library got discontinued in 2011, but a project named pillow forked the original PIL project and added Python3.x support to it. Pillow was announced as a replacement for PIL for future usage. Pillow supports a large number of image file formats including BMP, PNG, JPEG, and TIFF. The library encourages adding support for newer formats in the library by creating new file decoders.  
This module is not preloaded with Python. So to install it execute the following command in the command-line:

**from PIL import Image**

**img = Image.open(r"img")**

**Code**

#imports

from tkinter import \*

import os

from PIL import ImageTk, Image

from tkinter import messagebox

from tkinter import ttk

#Main Screen

master = Tk()

master.title('Banking App'.center(420))

master.geometry("1000x1000+0+0")

master.configure(background = "black")

#Functions

def finish\_reg():

    name = temp\_name.get()

    age = temp\_age.get()

    gender = temp\_gender.get()

    password = temp\_password.get()

    all\_accounts = os.listdir()

    if name == "" or age == "" or gender == "" or password == "":

        return messagebox.showerror("Error","All Fields Required")

    for name\_check in all\_accounts:

        if name == name\_check:

            notif.config(fg="red",text="Account already exists")

            return

        else:

            new\_file = open(name,"w")

            new\_file.write(name+'\n')

            new\_file.write(password+'\n')

            new\_file.write(age+'\n')

            new\_file.write(gender+'\n')

            new\_file.write('0')

            new\_file.close()

            notif.config(fg="green", text="Account has been created")

def register():

    #Vars

    global temp\_name

    global temp\_age

    global temp\_gender

    global temp\_password

    global notif

    temp\_name = StringVar()

    temp\_age = StringVar()

    temp\_gender = StringVar()

    temp\_password = StringVar()

    #Register Screen

    register\_screen = Toplevel(master)

    register\_screen.title('Register')

    register\_screen.geometry("600x500+0+0")

    #Labels

    Label(register\_screen, text="Enter Your details to Register", font=('times new roman',35)).grid(row=0,sticky=N,pady=10)

    Label(register\_screen, text="Name",fg='red', font=('times new roman',25)).grid(row=1,sticky=W)

    Label(register\_screen, text="Age",fg='red', font=('times new roman',25)).grid(row=2,sticky=W)

    Label(register\_screen, text="Gender", fg='red',font=('times new roman',25)).grid(row=3,sticky=W)

    Label(register\_screen, text="Password", fg='red',font=('times new roman',25)).grid(row=4,sticky=W)

    notif = Label(register\_screen, font=('times new roman',25))

    notif.grid(row=6,sticky=N,pady=10)

    #Entries

    Entry(register\_screen,textvariable=temp\_name,bd=5,fg='blue',font=('times new roman',20,"bold")).grid(row=1,column=0,padx=20,pady=10)

    Entry(register\_screen,textvariable=temp\_age,bd=5,fg='blue',font=('times new roman',20,"bold")).grid(row=2,column=0,padx=20,pady=10)

    #Entry(register\_screen,textvariable=temp\_gender,bd=5,font=('times new roman',20,"bold")).grid(row=3,column=0,padx=20,pady=10)

    gender=ttk.Combobox(register\_screen,textvariable=temp\_gender,font=('times new roman',20,"bold"),width=20,state="readonly")

    gender['values']=("Male","Female","Others")

    gender.grid(row=3,column=0,padx=20,pady=10,)

    Entry(register\_screen,textvariable=temp\_password,bd=5,show="\*",fg='blue',font=('times new roman',20,"bold")).grid(row=4,column=0,padx=20,pady=10)

    #Buttons

    Button(register\_screen, text="Register", command = finish\_reg,fg='blue', font=('times new roman',25)).grid(row=5,sticky=N,pady=10)

def login\_session():

    global login\_name

    all\_accounts = os.listdir()

    login\_name = temp\_login\_name.get()

    login\_password = temp\_login\_password.get()

    for name in all\_accounts:

        if name == login\_name:

            file = open(name,"r")

            file\_data = file.read()

            file\_data = file\_data.split('\n')

            password  = file\_data[1]

            #Account Dashboard

            if login\_password == password:

                login\_screen.destroy()

                account\_dashboard = Toplevel(master)

                account\_dashboard.title('Dashboard')

                account\_dashboard.geometry("500x700+0+0")

                account\_dashboard.configure(background = "black")

                #Labels

                Label(account\_dashboard, text="Account Dashboard", fg="#ff00bf",font=('times new roman',40)).grid(row=0,sticky=N,pady=10)

                Label(account\_dashboard, text="Welcome "+name,fg="#ff00bf", font=('times new roman',40)).grid(row=1,sticky=N,pady=5)

                #Buttons

                Button(account\_dashboard, text="Personal Details",fg="blue",font=('times new roman',30),width=20,command=personal\_details).grid(row=2,sticky=N,padx=10,pady=20)

                Button(account\_dashboard, text="Deposit",fg="blue",font=('times new roman',30),width=20,command=deposit).grid(row=3,sticky=N,padx=10,pady=20)

                Button(account\_dashboard, text="Withdraw",fg="blue",font=('times new roman',30),width=20,command=withdraw).grid(row=4,sticky=N,padx=10,pady=20)

                Label(account\_dashboard).grid(row=5,sticky=N,pady=10)

                return

            else:

                login\_notif.config(fg="red", text="Password incorrect!!")

                return

    login\_notif.config(fg="red", text="No account found !!")

def deposit():

    #Vars

    global amount

    global deposit\_notif

    global current\_balance\_label

    amount = StringVar()

    file   = open(login\_name, "r")

    file\_data = file.read()

    user\_details = file\_data.split('\n')

    details\_balance = user\_details[4]

    #Deposit Screen

    deposit\_screen = Toplevel(master)

    deposit\_screen.title('Deposit')

    deposit\_screen.geometry("500x500+0+0")

    #Label

    Label(deposit\_screen, text="Deposit",fg="#ff00bf", font=('times new roman',40)).grid(row=0,sticky=N,pady=10)

    current\_balance\_label = Label(deposit\_screen,fg="red", text="Current Balance : $"+details\_balance, font=('times new roman',30))

    current\_balance\_label.grid(row=1,sticky=W)

    Label(deposit\_screen, text="Amount : ",fg="blue", font=('times new roman',30)).grid(row=2,sticky=W)

    deposit\_notif = Label(deposit\_screen,fg="blue",font=('times new roman',10))

    deposit\_notif.grid(row=4, sticky=N,pady=5)

    #Entry

    Entry(deposit\_screen, textvariable=amount,bd=5,fg="blue",font=('times new roman',20,'bold')).grid(row=2,sticky=E,padx=20,pady=10)

    #Button

    Button(deposit\_screen,text="Finish",fg="blue",font=('times new roman',30),command=finish\_deposit).grid(row=5,sticky=N,pady=5

def finish\_deposit():

    if amount.get() == "":

        deposit\_notif.config(text='Amount is required!',fg="red")

        return

    if float(amount.get()) <=0:

        deposit\_notif.config(text='Negative currency is not accepted', fg='red')

        return

    file = open(login\_name, 'r+')

    file\_data = file.read()

    details = file\_data.split('\n')

    current\_balance = details[4]

    updated\_balance = current\_balance

    updated\_balance = float(updated\_balance) + float(amount.get())

    file\_data       = file\_data.replace(current\_balance, str(updated\_balance))

    file.seek(0)

    file.truncate(0)

    file.write(file\_data)

    file.close()

    current\_balance\_label.config(text="Current Balance : $"+str(updated\_balance),fg="green")

    deposit\_notif.config(text='Balance Updated', fg='green')

def withdraw():

     #Vars

    global withdraw\_amount

    global withdraw\_notif

    global current\_balance\_label

    withdraw\_amount = StringVar()

    file   = open(login\_name, "r")

    file\_data = file.read()

    user\_details = file\_data.split('\n')

    details\_balance = user\_details[4]

    #Deposit Screen

    withdraw\_screen = Toplevel(master)

    withdraw\_screen.title('Withdraw')

    withdraw\_screen.geometry("500x500+0+0")

    #Label

    Label(withdraw\_screen, text="Withdraw", fg="#ff00bf",font=('times new roman',40)).grid(row=0,sticky=N,pady=10)

    current\_balance\_label = Label(withdraw\_screen,fg="red", text="Current Balance : $"+details\_balance, font=('times new roman',30))

    current\_balance\_label.grid(row=1,sticky=W)

    Label(withdraw\_screen, text="Amount : ",fg="blue", font=('times new roman',30)).grid(row=2,sticky=W)

    withdraw\_notif = Label(withdraw\_screen,font=('times new roman',10))

    withdraw\_notif.grid(row=4, sticky=N,pady=5)

    #Entry

    Entry(withdraw\_screen,fg="blue",bd=5,font=('times new roman',20,'bold'), textvariable=withdraw\_amount).grid(row=2,padx=20,pady=10,sticky=E)

    #Button

    Button(withdraw\_screen,fg="blue",bd=5,font=('times new roman',30),text="Finish",command=finish\_withdraw).grid(row=5,sticky=N,pady=5)

def finish\_withdraw():

    if withdraw\_amount.get() == "":

        withdraw\_notif.config(text='Amount is required!',fg="red")

        return

    if float(withdraw\_amount.get()) <=0:

        withdraw\_notif.config(text='Negative currency is not accepted', fg='red')

        return

    file = open(login\_name, 'r+')

    file\_data = file.read()

    details = file\_data.split('\n')

    current\_balance = details[4]

    if float(withdraw\_amount.get()) >float(current\_balance):

        withdraw\_notif.config(text='Insufficient Funds!', fg='red')

        return

    updated\_balance = current\_balance

    updated\_balance = float(updated\_balance) - float(withdraw\_amount.get())

    file\_data       = file\_data.replace(current\_balance, str(updated\_balance))

    file.seek(0)

    file.truncate(0)

    file.write(file\_data)

    file.close()

    current\_balance\_label.config(text="Current Balance : $"+str(updated\_balance),fg="green")

    withdraw\_notif.config(text='Balance Updated', fg='green')

def personal\_details():

    #Vars

    file = open(login\_name, 'r')

    file\_data = file.read()

    user\_details = file\_data.split('\n')

    details\_name = user\_details[0]

    details\_age = user\_details[2]

    details\_gender = user\_details[3]

    details\_balance = user\_details[4]

    #Personal details screen

    personal\_details\_screen = Toplevel(master)

    personal\_details\_screen.title('Personal Details')

    personal\_details\_screen.geometry("400x500+0+0")

    #Labels

    Label(personal\_details\_screen, text="Personal Details",fg="green", font=('times new roman',40)).grid(row=0,sticky=N,pady=10,padx=20)

    Label(personal\_details\_screen, text="Name : "+details\_name,fg="red", font=('times new roman',20)).grid(row=1,sticky=W,pady=10,padx=20)

    Label(personal\_details\_screen, text="Age : "+details\_age,fg="red", font=('times new roman',20)).grid(row=2,sticky=W,pady=10,padx=20)

    Label(personal\_details\_screen, text="Gender : "+details\_gender,fg="red", font=('times new roman',20)).grid(row=3,sticky=W,pady=10,padx=20)

    Label(personal\_details\_screen, text="Balance :$"+details\_balance, fg="red",font=('times new roman',20)).grid(row=4,sticky=W,pady=10,padx=20)

def login():

    #Vars

    global temp\_login\_name

    global temp\_login\_password

    global login\_notif

    global login\_screen

    temp\_login\_name = StringVar()

    temp\_login\_password = StringVar()

    #Login Screen

    login\_screen = Toplevel(master)

    login\_screen.title('Login')

    login\_screen.geometry("650x500+0+0")

    #Labels

    Label(login\_screen, text=" Enter your details below to Login ", font=('times new roman',35)).grid(row=0,sticky=W)

    Label(login\_screen, text="Username",fg='red', font=('times new roman',25)).grid(row=1,sticky=W)

    Label(login\_screen, text="Password", fg='red',font=('times new roman',25)).grid(row=2,sticky=W)

    login\_notif = Label(login\_screen, font=('times new roman',12))

    login\_notif.grid(row=4,sticky=W)

    #Entry

    Entry(login\_screen, textvariable=temp\_login\_name,bd=5,fg='blue',font=('times new roman',20,'bold')).grid(row=1,padx=20,pady=10,sticky=N)

    Entry(login\_screen, textvariable=temp\_login\_password,show="\*",bd=5,fg='blue',font=('times new roman',20,'bold')).grid(row=2,padx=20,pady=10,sticky=N)

    #Button

    Button(login\_screen, text="Login", command=login\_session,fg="blue",font=('times new roman',25)).grid(row=5,sticky=N,pady=20,padx=10)

#Image import

img = Image.open('1..jpg')

img = img.resize((1000,400))

img = ImageTk.PhotoImage(img)

#Labels

Label(master, text = "Online Banking ", font=('times new roman',40)).grid(row=0,sticky=N,pady=10)

Label(master, text = "The Most Secure Bank You've Probably Used", font=('times new roman',40)).grid(row=1,sticky=N)

Label(master, image=img).grid(row=2,sticky=N,pady=15)

#Buttons

Button(master, text="Register", font=('times new roman',30),width=25,command=register).grid(row=3,sticky=N)

Button(master, text="Login", font=('times new roman',30),width=25,command=login).grid(row=4,sticky=N,pady=10)

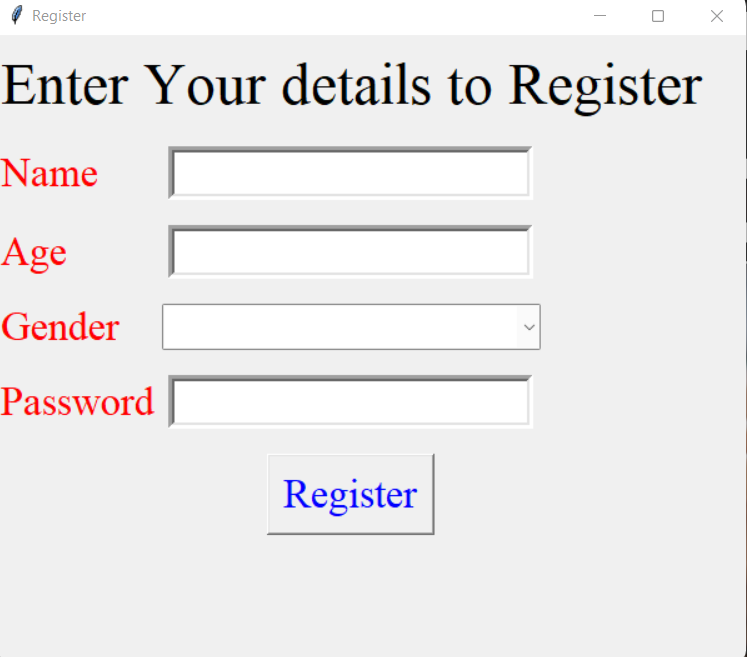
master.mainloop()

**Implementation**

**Home Page**

****

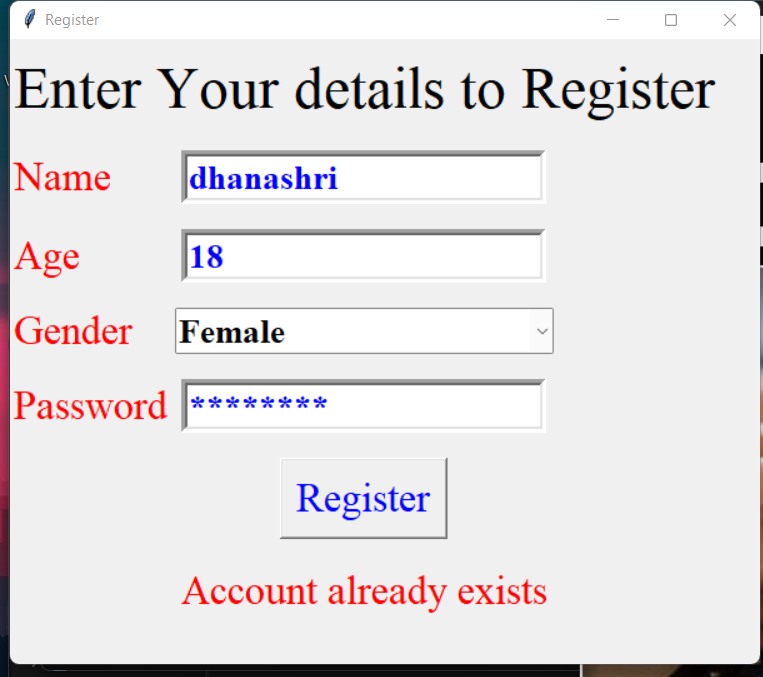
**Registration Form**

****

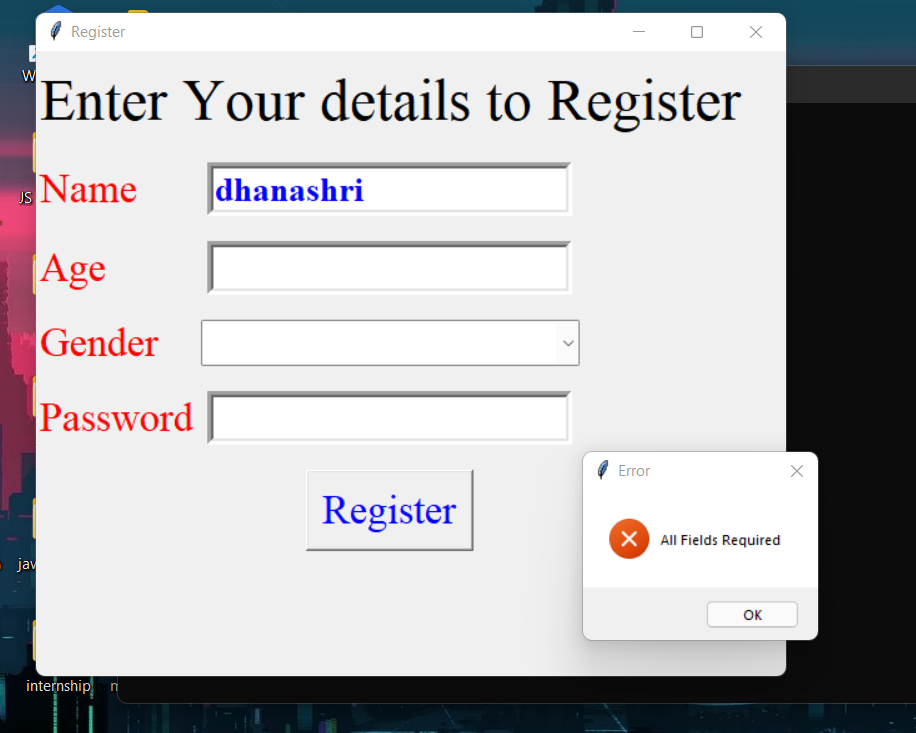
**When all fields are correct**

****

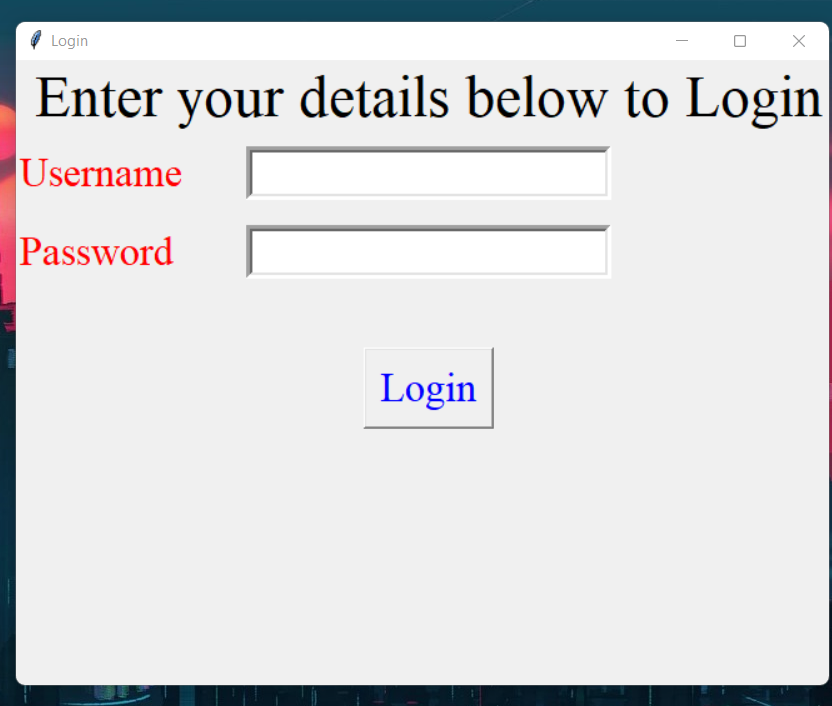
**When given details are already exist in System**

****

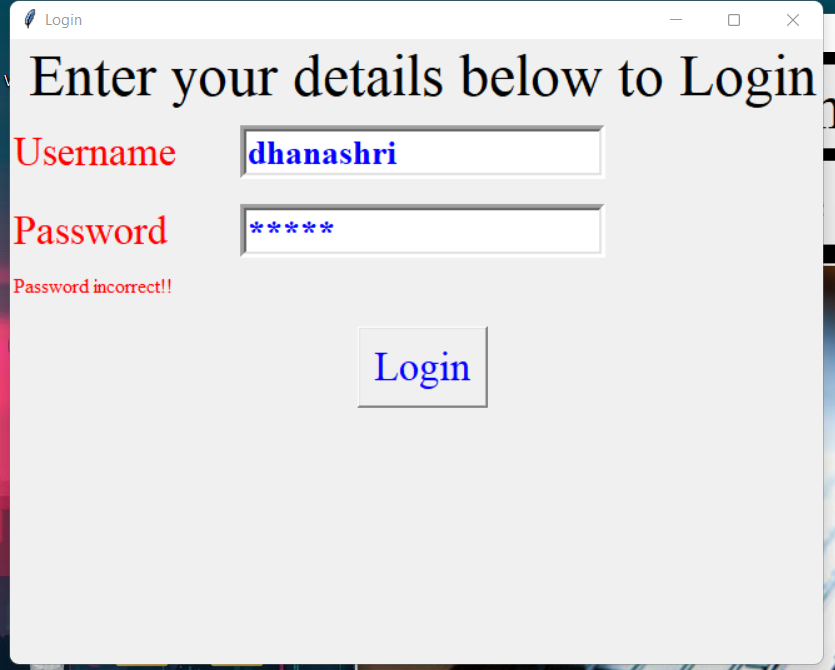
**When any of the fields remain empty**

****

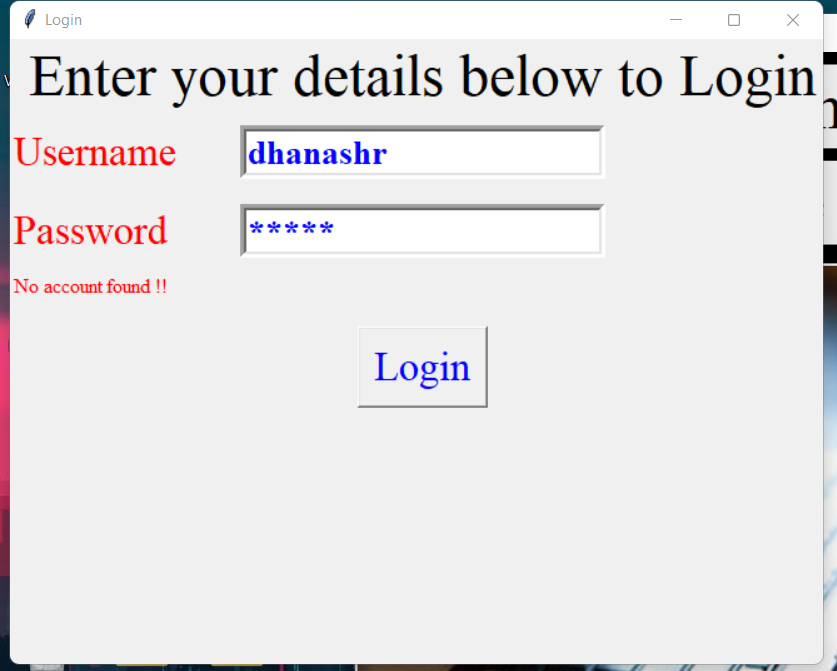
**Login Form**

****

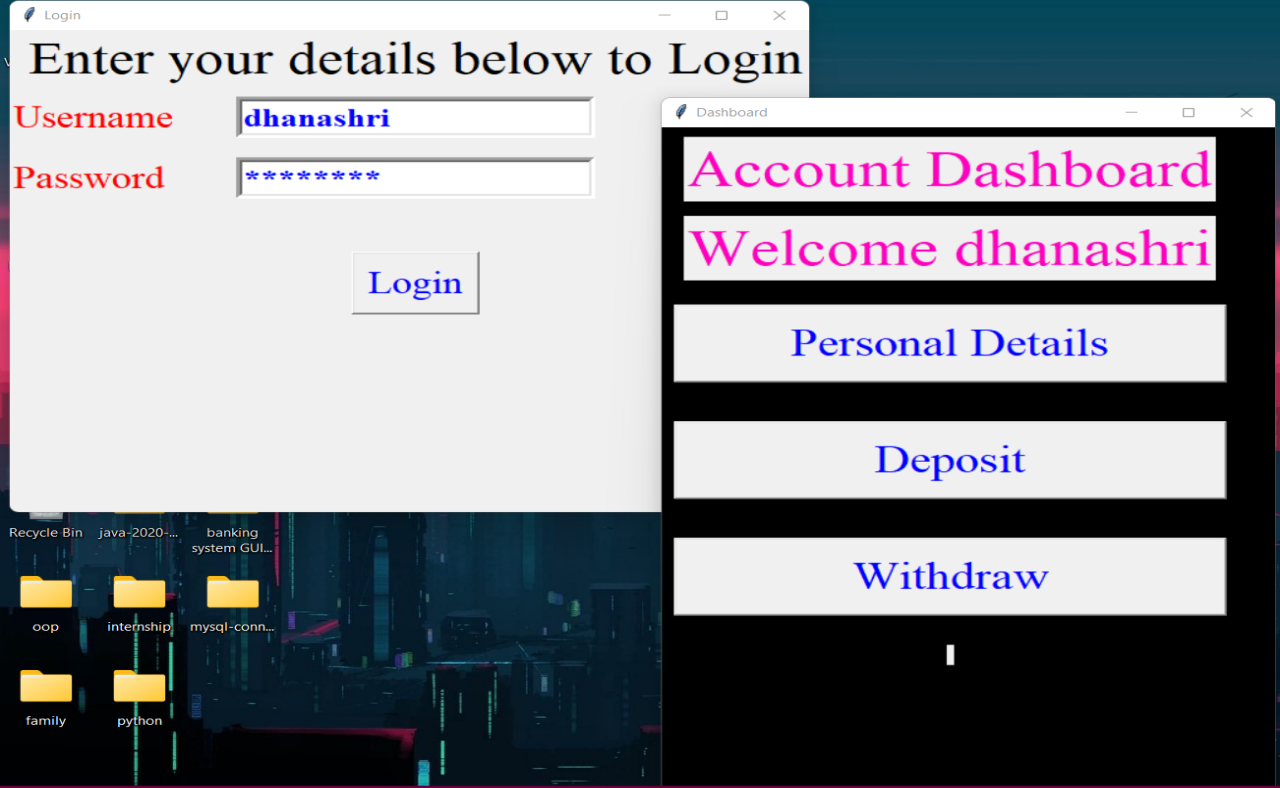
**When password is incorrect for entered username**

****

**When username and password are not registered in the system**

****

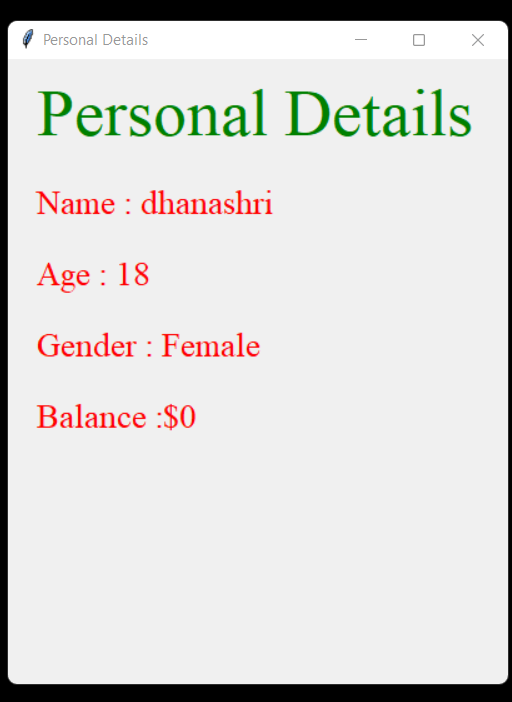
**When username and password both are correct**

****

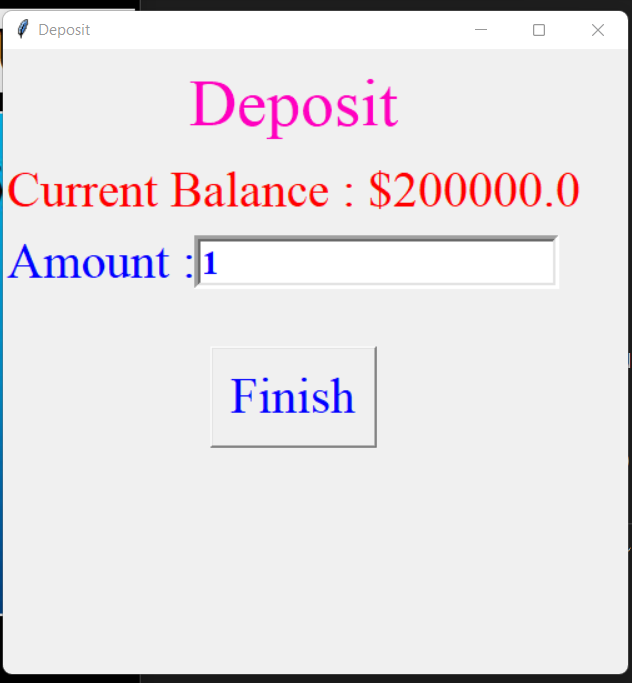
**Dashboard**

****

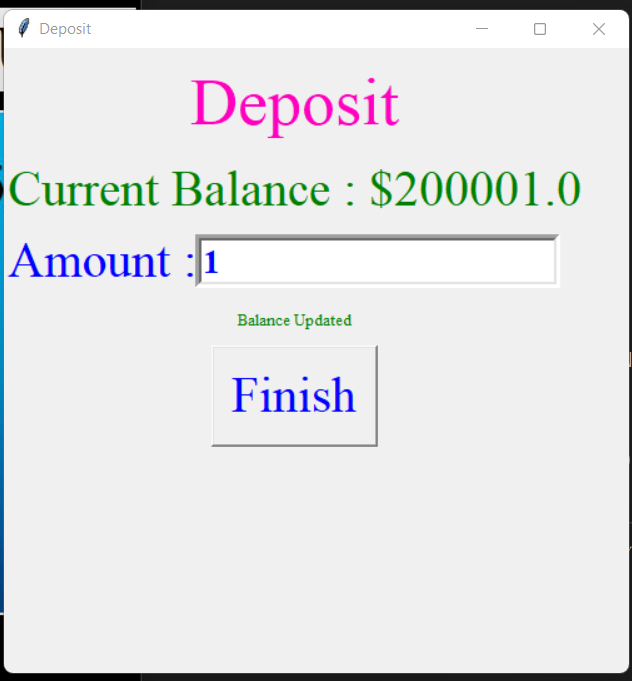
**Personal Details**

****

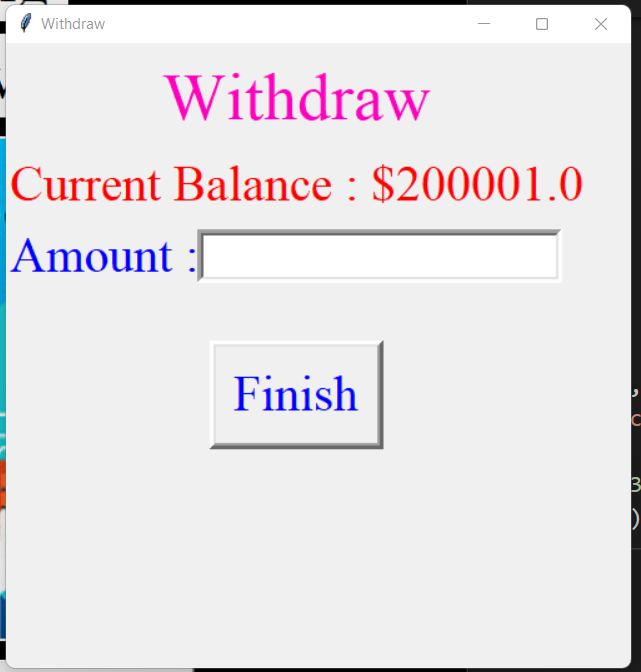
**Deposit**

****

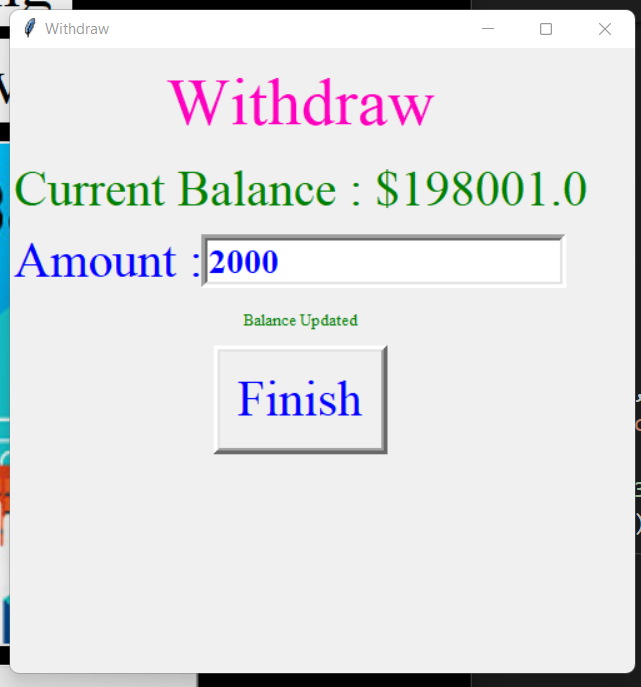
**When press finish button**

****

**Withdraw**

****

**When we press on finish button**

****

**Course Outcome Achieve**

1. Execute programs using operators and control flow statements.
2. Perform Operations on Python Data structures.
3. Develop applications using Functions, Modules and Packages.
4. Develop applications using object oriented concepts in python.
5. Write Python code for File Handling.
6. Working together as a group ,teamwork.
7. Logic building.

**Conclusion**

Hence we successfully creates the project on Bank Management System in Python Programming Language.

**References**

1. <https://www.geeksforgeeks.org/python-pillow-a-fork-of-pil/>
2. <https://www.geeksforgeeks.org/python-gui-tkinter/>
3. [https://docs.python.org/3/library/tkinter.ttk.html/](https://www.geeksforgeeks.org/introduction-linux-shell-shell-scripting/)