**FREEDOM INTERNATIONAL SCHOOL**

# 33, Sector IV, HSR Layout, Bengaluru, Karnataka 560102

SCHOOL CODE: 45175 AFFILIATION NUMBER: 830183

**COMPUTER SCIENCE PROJECT**

**ON**

**Banking Software**

**DONE BY**

Adit Prabhu

Dheeraj Rao

Sri Vallabha B S

Class XI A

**Under the guidance of**

**Ms. SUMITA TYAGI**

**PGT- COMPUTER SCIENCE**

**Vice Principal Principal**

**Ms. Clara David Ms. Sneha Rai**

**Freedom International School Freedom International School**

**Bangalore Bangalore**



**FREEDOM INTERNATIONAL SCHOOL**

# 33, Sector IV, HSR Layout, Bengaluru, Karnataka 560102

SCHOOL CODE: 45175 AFFILIATION NUMBER:830183

**CERTIFICATE**

This is to certify that the Computer Science Project Report entitled

**Project Name,**

was carried out by **Adit Prabhu, Dheeraj Rao and Sri Vallabha BS** of Class **XI**,

Roll No. **2, 12, 32**, a student of

FREEDOM INTERNATIONAL SCHOOL in partial fulfilment of the Computer Science Practical Examinations prescribed by the CBSE

during the Academic Year 2022-2023.

I certify that this project has been done by him/her with his/her own effort with the guidance of the teacher.

**Signature of the Teacher In-Charge Signature of the Principal**

**Ms. Sumita Tyagi Ms. Sneha Rai**

**Name of the Examiners Signature with date**

1. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
2. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ACKNOWLEDGEMENT**

We would like to express our special thanks and gratitude to our teacher and project guide Ms. Sumita Tyagi who gave us the opportunity to work on this project. A lot of research was involved which helped us in learning more about the topic and discovering many new things. This has been an important learning experience.

Our sincere thanks to Ms. Sneha Rai, our principal, for her coordination in extending every possible support for the completion of this project.

We also thank our parents for their motivation and support.

Last but not least, we would like to thank all those who had helped directly or indirectly towards the completion of this project.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Serial No.** | **Contents** | **Page No.** |
| 1. | Project Description | 7 |
| 2. | Requirements | 8 |
| 3. | Source code | 9-20 |
| 4. | Output Screenshots | 21-25 |
| 5. | Future Scope | 26 |
| 6. | References | 27 |

**PROJECT DESCRIPTION**

Our Project is to create a Banking Management software. The main purpose of our software is to simulate net/online banking facilities provided by banks.

Our software allows our users to create an account with a username and password. It also ensures that the password chosen by the user is strong.

After money is deposited to the users account, they are able to withdraw money from the account or transfer money to other user users. This process is made more secure by verifying the password each time a withdrawal or transfer is made.

The users can also see the history of the latest transaction completed by them including the type of transaction, date, amount and net balance at that time. The users also receive a notification when money is transferred to them.

All the data of the users is stored in a sqlite3 database. This data includes their username, password, balance and transaction history.

The software is very user-friendly and easy on the eyes.

**PROJECT REQUIREMENTS**

1. tkinter
2. ttk
3. sqlite3
4. time
5. PIL

**SOURCE CODE**

from tkinter import \*

from tkinter import messagebox

from tkinter import ttk

import sqlite3

import time

from PIL import ImageTk, Image

user\_list=[]

trans\_hist\_limit=10

notif\_limit=10

def init():

conn=sqlite3.connect("Database.db")

cur=conn.cursor()

cur.execute("""CREATE TABLE IF NOT EXISTS Users(Name text, Username text, Password text, Balance int, Trans\_Hist text, Notifications text);""")

conn.commit()

conn.close()

init\_login('start')

def init\_login(From):

global tb\_username

global tb\_password

global login\_win

if From=='signup':

signup\_win.destroy()

elif From=='main':

user\_list.clear()

main\_win.destroy()

login\_win=Tk()

login\_win.geometry('1500x750')

login\_win.title("Login page")

Logo = ImageTk.PhotoImage(Image.open('Bank Logo.jpg'))

login\_win.iconphoto(False, Logo)

Bg = ImageTk.PhotoImage(Image.open('Bank Background.jpg').resize((1500,750)))

lbl\_bg=Label(login\_win, image=Bg)

lbl\_bg.pack()

lbl\_login=Label(login\_win,text="Login",justify='center',font = ('Arial ' , 25, 'bold'), fg='#0f0',bg='#054175').place(relx=0.5,rely=0.35,anchor='c')

lbl\_username=Label(login\_win,text="Username:", font = ('Arial' , 15), fg='#ffffff', bg='#054175').place(relx=0.42,rely=0.4 , anchor='c')

tb\_username=Entry(login\_win ,font = ('Arial' , 15))

tb\_username.place(relx=0.54,rely=0.4,anchor ='c')

lbl\_password=Label(login\_win,text="Password:",font = ('Arial' , 15), fg='#ffffff', bg='#054175').place(relx=0.42,rely=0.45 , anchor='c')

tb\_password=Entry(login\_win,show='\*',font = ('Arial' , 15))

tb\_password.place(relx=0.54,rely=0.45,anchor ='c')

btn\_login=Button(text="Login",command=login,font = ('Arial' , 15))

btn\_login.place(relx=0.5 , rely=0.51,anchor = 'c')

login\_win.bind('<Return>' , lambda event:login())

lbl\_signup=Label(login\_win,text="Don't have an account?",font = ('Calibri' , 15,'bold'), fg='#ffbf00', bg='#055d8b').place(relx=0.5 , rely = 0.56 , anchor = 'c')

btn\_signup=Button(text="Signup",font = ('Arial' , 15),command=lambda: init\_signup('login'))

btn\_signup.place(relx=0.5 , rely = 0.61, anchor = 'c')

login\_win.mainloop()

def init\_signup(From):

global signup\_win

global tb\_name

global tb\_username

global tb\_password

global tb\_reenterpass

global tb\_initdep

if From=='login':

login\_win.destroy()

elif From=='main':

main\_win.destroy()

signup\_win = Tk()

signup\_win.title("Sign-up Page")

signup\_win.geometry('1500x750')

Logo = ImageTk.PhotoImage(Image.open('Bank Logo.jpg'))

signup\_win.iconphoto(False, Logo)

Bg = ImageTk.PhotoImage(Image.open('Bank Background.jpg').resize((1500,750)))

lbl\_bg=Label(signup\_win, image=Bg)

lbl\_bg.pack()

frame\_ui = Frame(signup\_win, bg='#055a86')

frame\_ui.place(relx=0.375,rely=0.25)

lbl\_createacnt=Label(frame\_ui,text= 'Create an Account', font = ('Arial' , 25, 'bold'), fg='#0f0', bg='#055a86').grid(row=0,column=0,columnspan=2)

lbl\_name=Label(frame\_ui,text='Enter Name:', font = ('Arial ' , 15), bg='#055a86', fg='#ffffff').grid(row=1,column=0,pady=5)

tb\_name=Entry(frame\_ui, width=10, font = ('Arial' , 15))

tb\_name.grid(row=1, column=1, pady=5, padx=5)

lbl\_username=Label(frame\_ui, text= 'Enter Username:', font = ('Arial' , 15), bg='#055a86', fg='#ffffff').grid(row=2,column=0,pady=5)

tb\_username=Entry(frame\_ui, width=10, font = ('Arial' , 15))

tb\_username.grid(row=2, column=1, pady=5, padx=5)

lbl\_password=Label(frame\_ui, text= 'Enter Password:', font = ('Arial' , 15), bg='#055a86', fg='#ffffff').grid(row=3,column=0,pady=5)

tb\_password=Entry(frame\_ui,width=10,show='\*', font = ('Arial' , 15))

tb\_password.grid(row=3, column=1, pady=5, padx=5)

lbl\_reenterpass=Label(frame\_ui,text='Re-enter password:', font = ('Arial' , 15), bg='#055a86', fg='#ffffff').grid(row=4,column=0,pady=5)

tb\_reenterpass=Entry(frame\_ui,width=10,show='\*', font = ('Arial' , 15))

tb\_reenterpass.grid(row=4, column=1, pady=5, padx=5)

lbl\_initdep=Label(frame\_ui,text='Enter Initial Deposit:', font = ('Arial' , 15), bg='#055a86', fg='#ffffff').grid(row=5,column=0,pady=5)

tb\_initdep=Entry(frame\_ui,width=10, font = ('Arial' , 15))

tb\_initdep.grid(row=5, column=1, pady=5, padx=5)

btn\_signup=Button(frame\_ui, text='Sign Up',command=signup, font = ('Arial' , 15))

btn\_signup.grid(row=6, column=0, columnspan=2, pady=10)

signup\_win.bind('<Return>' , lambda event:signup())

lbl\_login=Label(frame\_ui, text="Already have an account?",font = ('Calibri' , 15,'bold'), fg='#ffbf00', bg='#055a86').grid(row=7,column=0,columnspan=2,pady=15)

btn\_login=Button(frame\_ui,text="Go back to login page", font = ('Arial' , 15), command=lambda: init\_login('signup'))

btn\_login.grid(row=8,column=0,columnspan=2,pady=5)

signup\_win.mainloop()

def init\_main(From):

global main\_win

global user\_list

global frame\_content

global frame\_centering

global lbl\_balance

if From=='login':

login\_win.destroy()

elif From=='signup':

signup\_win.destroy()

conn=sqlite3.connect("Database.db")

cur=conn.cursor()

main\_win=Tk()

main\_win.title("Main page")

main\_win.geometry('1250x750')

Logo = ImageTk.PhotoImage(Image.open('Bank Logo.jpg'))

main\_win.iconphoto(False, Logo)

Bg = ImageTk.PhotoImage(Image.open('Bank Background.jpg').resize((1500,750)))

lbl\_bg=Label(main\_win, image=Bg)

lbl\_bg.place(relx=0,rely=0, anchor='nw')

lbl\_name=Label(main\_win,text=("Name : "+user\_list[1]), font = ('Arial' , 15), bg='#010f4c', fg='#ffffff')

lbl\_name.grid(row=0,column=0, padx=2, pady=2)

lbl\_username=Label(main\_win,text=("Username : "+user\_list[2]), font = ('Arial' , 15), bg='#010f4c', fg='#ffffff')

lbl\_username.grid(row=1,column=0, padx=2, pady=2)

lbl\_balance=Label(main\_win,text=("Balance : "+str(user\_list[4])), font = ('Arial' , 15), bg='#010f4c', fg='#ffffff')

lbl\_balance.grid(row=2,column=0, padx=2, pady=2)

btn\_logout=Button(main\_win,text="Logout", font = ('Arial' , 15),command=lambda: init\_login('main'))

btn\_logout.grid(row=3, column=0, pady=10)

frame\_btn=Frame(main\_win,width=50,height=100)

frame\_btn.grid(row=4, column=0, sticky=N, padx=10, pady=20)

btn\_deposit=Button(frame\_btn,text='Deposit', width=15, font = ('Arial' , 15), command=lambda: display\_tab('deposit') )

btn\_deposit.grid(row=0,column=0)

btn\_withdrawal=Button(frame\_btn,text='Withdraw', width=15, font = ('Arial' , 15), command=lambda: display\_tab('withdraw') )

btn\_withdrawal.grid(row=1,column=0)

btn\_transfer=Button(frame\_btn,text='Transfer', width=15, font = ('Arial' , 15), command=lambda: display\_tab('transfer') )

btn\_transfer.grid(row=2,column=0)

btn\_history=Button(frame\_btn,text='Transaction History', width=15, font = ('Arial' , 15), command=lambda: display\_tab('transaction history') )

btn\_history.grid(row=3,column=0)

frame\_content=Frame(main\_win,width=750,height=500,pady=10,padx=0,bg='#7fd2df')

frame\_content.grid(row=4,column=1)

frame\_content.grid\_propagate(False)

frame\_centering=Frame(frame\_content, bg='#7fd2df')

frame\_centering.place(relx=0.5,rely=0,anchor='n')

display\_tab('deposit')

for i in eval(user\_list[6]):

messagebox.showinfo("Money Transferred", i)

user\_list[6]='[]'

cur.execute("UPDATE Users SET Notifications = '[]' WHERE rowid = ?", (user\_list[0],))

conn.commit()

main\_win.mainloop()

conn.commit()

conn.close()

def display\_tab(tab):

for widgets in frame\_centering.winfo\_children():

widgets.destroy()

if tab=='transfer':

global tb\_user

global tb\_transfer

lbl\_transfer=Label(frame\_centering, text="Transfer", font = ('Arial' , 20, 'bold'), bg='#7fd2df', fg='#000000', pady=20).grid(row=0,column=0,columnspan=2)

lbl\_user=Label(frame\_centering,text='Enter username of recepient :', font = ('Arial' , 15), bg='#7fd2df', fg='#000000').grid(row=1,column=0)

tb\_user=Entry(frame\_centering, font = ('Arial' , 15))

tb\_user.grid(row=1,column=1)

lbl\_amt=Label(frame\_centering,text='Enter amount to be transferred :', font = ('Arial' , 15), bg='#7fd2df', fg='#000000').grid(row=2,column=0)

tb\_transfer=Entry(frame\_centering, font = ('Arial' , 15))

tb\_transfer.grid(row=2,column=1)

btn\_transfer = Button(frame\_centering , text = 'Transfer ', font = ('Arial',15), command = lambda: transfer(tb\_transfer.get(), tb\_user.get()))

btn\_transfer.grid(row=3, column = 0 , columnspan = 2, pady=10)

main\_win.bind('<Return>' , lambda event:transfer(tb\_transfer.get(), tb\_user.get()))

elif tab=='withdraw':

global tb\_withdraw

lbl\_withdraw=Label(frame\_centering, text="Withdraw", font = ('Arial' , 20, 'bold'), bg='#7fd2df', fg='#000000', pady=20).grid(row=0,column=0,columnspan=2)

lbl\_amount= Label(frame\_centering , text = 'Enter amount to be withdrawn : ', font = ('Arial',15), bg='#7fd2df', fg='#000000').grid(row=1,column=0)

tb\_amount = Entry(frame\_centering, font = ('Arial' , 15))

tb\_amount.grid(row=1,column=1)

btn\_withdraw = Button(frame\_centering , text = 'Withdraw', font = ('Arial' , 15) , command = lambda: withdraw(tb\_amount.get()))

btn\_withdraw.grid(row=2, column=0, columnspan = 2, pady=10)

main\_win.bind('<Return>' , lambda event:withdraw(tb\_amount.get()))

elif tab=='deposit':

global tb\_deposit

lbl\_deposit=Label(frame\_centering, text="Deposit", font = ('Arial' , 20, 'bold'), bg='#7fd2df', fg='#000000', pady=20).grid(row=0,column=0,columnspan=2)

lbl\_amount= Label(frame\_centering , text = 'Enter amount to be deposited : ', font = ('Arial' , 15), bg='#7fd2df', fg='#000000').grid(row=1 , column=0)

tb\_amount = Entry(frame\_centering, font = ('Arial' , 15))

tb\_amount.grid(row=1,column=1)

btn\_deposit = Button(frame\_centering , text = 'Deposit', font = ('Arial' , 15), command = lambda: deposit(tb\_amount.get()))

btn\_deposit.grid(row=2, column=0, columnspan = 2, pady=10)

main\_win.bind('<Return>' , lambda event:deposit(tb\_amount.get()))

elif tab=='transaction history':

lbl\_TransHist=Label(frame\_centering, text="Transaction History", font = ('Arial' , 20, 'bold'), bg='#7fd2df', fg='#000000', pady=20).grid(row=0,column=0)

style = ttk.Style()

style.configure("Style1.Treeview", font=('Arial', 15))

style.configure("Style1.Treeview.Heading", font=('Arial',15,'bold'))

table = ttk.Treeview(frame\_centering, style="Style1.Treeview")

table['columns'] = ("Date", "Transaction", "Change in balance", "Total balance")

table.column("#0", width=0, stretch=NO)

table.column("Date", anchor=CENTER, width=125)

table.column("Transaction", anchor=W, width=300)

table.column("Change in balance", anchor=CENTER, width=150)

table.column("Total balance", anchor=CENTER, width=125)

table.heading("#0", text='', anchor=CENTER)

table.heading("Date", text="Date", anchor=CENTER)

table.heading("Transaction", text="Transaction", anchor=W)

table.heading("Change in balance", text="Change in Bal.", anchor=CENTER)

table.heading("Total balance", text="Total Bal.", anchor=CENTER)

update\_info()

for i in range(len(user\_list[5])):

table.insert(parent='', index=0, iid = len(user\_list[5])-i, text='', values = user\_list[5][i])

table.grid(row=1,column=0)

main\_win.bind('<Return>', lambda event:display\_tab('transaction history'))

return()

def login():

global user\_list

global tb\_username

global tb\_password

conn=sqlite3.connect("Database.db")

cur=conn.cursor()

if tb\_username.get()=='' or tb\_password.get()=='':

messagebox.showwarning("All fields are required","Please enter all required fields")

else:

cur.execute("SELECT rowid,\* FROM Users WHERE USERNAME=?",(tb\_username.get(),))

temp\_list=cur.fetchone()

if temp\_list==None:

messagebox.showwarning("Invalid username","Username does not exist")

elif tb\_password.get()==temp\_list[3]:

user\_list=list(temp\_list)

user\_list[5]=eval(user\_list[5])

init\_main('login')

else:

messagebox.showwarning("Wrong username or password","Username and password don't match")

conn.commit()

conn.close()

return()

def signup():

global user\_list

conn=sqlite3.connect("Database.db")

cur=conn.cursor()

if tb\_name.get()=='' or tb\_username.get()=='' or tb\_password.get()=='' or tb\_reenterpass.get()=='' or tb\_initdep.get()=='':

messagebox.showwarning("All fields are required","Please enter all required fields")

elif len(tb\_password.get())<8:

messagebox.showwarning("Password too short","Password must be at least 8 characters long")

elif tb\_password.get()!=tb\_reenterpass.get():

messagebox.showwarning("Password does not match","Password entered in password textbox does not match password entered in reenter password textbox")

elif not (tb\_initdep.get().isdigit()):

messagebox.showwarning("Invalid initial deposit","The initial deposit must be a number")

else:

cur.execute("SELECT rowid,\* FROM Users WHERE Username=?",(tb\_username.get(),))

temp\_list=cur.fetchone()

if temp\_list!=None:

messagebox.showwarning("Username already taken","Please choose another username")

else:

cur.execute("INSERT INTO Users VALUES (?,?,?,?,?,?)",(tb\_name.get(),tb\_username.get(),tb\_password.get(),int(tb\_initdep.get()),'[]','[]'))

conn.commit()

cur.execute("SELECT rowid,\* FROM Users WHERE Username=?",(tb\_username.get(),))

conn.commit()

user\_list=list(cur.fetchone())

user\_list[5]=eval(user\_list[5])

init\_main('signup')

conn.commit()

conn.close()

return()

def deposit(amount):

conn=sqlite3.connect("Database.db")

cur=conn.cursor()

if amount=='':

messagebox.showwarning("Invalid deposit amount","Please enter amount to deposit")

elif not amount.isdigit():

messagebox.showwarning("Invalid deposit amount","Amount to deposit should be a positive number rounded off to nearest rupee")

elif int(amount)==0:

messagebox.showwarning("Invalid deposit amount","Amount to deposit should be a positive number rounded off to nearest rupee")

else:

update\_info()

balance=user\_list[4]

cur.execute("UPDATE Users SET Balance = ? WHERE rowid = ?",(balance+int(amount),user\_list[0]))

conn.commit()

trans\_hist=list(user\_list[5])

if len(trans\_hist)>=trans\_hist\_limit:

trans\_hist.pop(0)

trans\_hist.append([time.strftime("%d/%m/%Y"), "Deposited Rs."+amount, '+'+amount, balance+int(amount)])

cur.execute("UPDATE Users SET Trans\_Hist = ? WHERE rowid = ?",(str(trans\_hist),user\_list[0]))

messagebox.showinfo("Transaction succesfull","Transaction completed successfully")

update\_info()

conn.commit()

conn.close()

return()

def withdraw(amount):

conn=sqlite3.connect("Database.db")

cur=conn.cursor()

if amount=='':

messagebox.showwarning("Invalid withdraw amount","Please enter amount to withdraw")

elif not amount.isdigit():

messagebox.showwarning("Invalid deposit amount","Amount to deposit should be a positive number rounded off to nearest rupee")

elif int(amount)==0:

messagebox.showwarning("Invalid deposit amount","Amount to deposit should be a positive number rounded off to nearest rupee")

else:

update\_info()

balance=int(user\_list[4])

if int(amount)>balance:

messagebox.showwarning("Error","You cannot withdraw more than you have in your account")

else:

auth\_win=Toplevel(main\_win)

auth\_win.title("Authentication")

Logo = ImageTk.PhotoImage(Image.open('Bank Logo.jpg'))

auth\_win.iconphoto(False, Logo)

def confirm\_pass():

if tb\_pass.get()==user\_list[3]:

auth\_win.destroy()

cur.execute("UPDATE Users SET Balance = ? WHERE rowid = ?",(balance-int(amount),user\_list[0]))

conn.commit()

trans\_hist=list(user\_list[5])

if len(trans\_hist)>=trans\_hist\_limit:

trans\_hist.pop(0)

trans\_hist.append([time.strftime("%d/%m/%Y"), "Withdrawed Rs."+amount , '-'+amount, balance-int(amount)])

cur.execute("UPDATE Users SET Trans\_Hist = ? WHERE rowid = ?",(str(trans\_hist),user\_list[0]))

conn.commit()

messagebox.showinfo("Transaction succesful","Amount withdrawn successfully")

update\_info()

else:

auth\_win.destroy()

messagebox.showwarning("Authentication failed","Wrong password, Please try again.")

lbl\_auth=Label(auth\_win,text="Please confirm your password to continue")

lbl\_auth.grid(row=0,column=0,columnspan=2)

lbl\_pass=Label(auth\_win,text="Password:")

lbl\_pass.grid(row=1,column=0)

tb\_pass=Entry(auth\_win,show='\*')

tb\_pass.grid(row=1,column=1)

btn\_auth=Button(auth\_win,text="Confirm",command=confirm\_pass)

btn\_auth.grid(row=2,column=0,columnspan=2)

auth\_win.bind('<Return>', lambda event:confirm\_pass())

auth\_win.mainloop()

conn.commit()

conn.close()

return()

def transfer(amount,user):

conn=sqlite3.connect("Database.db")

cur=conn.cursor()

if amount=='':

messagebox.showwarning("Invalid transfer amount","Please enter amount to transfer")

elif not amount.isdigit():

messagebox.showwarning("Invalid deposit amount","Amount to deposit should be a positive number rounded off to nearest rupee")

elif int(amount)==0:

messagebox.showwarning("Invalid deposit amount","Amount to deposit should be a positive number rounded off to nearest rupee")

elif user==user\_list[2]:

messagebox.showwarning("Invalid Username","You cannot transfer money to yourself")

else:

update\_info()

balance=int(user\_list[4])

cur.execute("SELECT rowid, Balance, Trans\_Hist, Notifications FROM Users WHERE Username=?",(user,))

conn.commit()

transferee\_list=cur.fetchone()

if float(amount) > balance:

messagebox.showwarning("Error","You cannot transfer more than you have in your account")

elif transferee\_list==None:

messagebox.showwarning("Invalid Username","No account with username "+user+" exists")

else:

auth\_win=Toplevel(main\_win)

auth\_win.title("Authentication")

Logo = ImageTk.PhotoImage(Image.open('Bank Logo.jpg'))

auth\_win.iconphoto(False, Logo)

def confirm\_pass():

if tb\_pass.get()==user\_list[3]:

auth\_win.destroy()

cur.execute("UPDATE Users SET Balance=? WHERE rowid=?",(balance-int(amount),user\_list[0]))

conn.commit()

cur.execute("UPDATE Users SET Balance=? WHERE rowid=?",(transferee\_list[1]+int(amount),transferee\_list[0]))

conn.commit()

trans\_hist=list(user\_list[5])

if len(trans\_hist)>=trans\_hist\_limit:

trans\_hist.pop(0)

trans\_hist.append([time.strftime("%d/%m/%Y"), "Transferred Rs."+amount+" to "+user, '-'+amount, balance-int(amount)])

cur.execute("UPDATE Users SET Trans\_Hist = ? WHERE rowid = ?",(str(trans\_hist),user\_list[0]))

conn.commit()

trans\_hist=eval(transferee\_list[2])

if len(trans\_hist)>=trans\_hist\_limit:

trans\_hist.pop(0)

trans\_hist.append([time.strftime("%d/%m/%Y"), "Received Rs."+amount+" from "+user\_list[2], '+'+amount, transferee\_list[1]+int(amount)])

cur.execute("UPDATE Users SET Trans\_Hist = ? WHERE rowid = ?",(str(trans\_hist),transferee\_list[0]))

conn.commit()

notifications=eval(transferee\_list[3])

if len(notifications)>=notif\_limit:

notifications.pop(0)

notifications.append("Received Rs."+amount+" from "+user\_list[2])

cur.execute("UPDATE Users SET Notifications = ? WHERE rowid = ?", (str(notifications), transferee\_list[0]))

conn.commit()

messagebox.showinfo("Transaction successful","Transaction completed successfully")

update\_info()

return()

else:

auth\_win.destroy()

messagebox.showwarning("Authentication failed","Wrong password, Please try again.")

return()

lbl\_auth=Label(auth\_win,text="Please confirm your password to continue")

lbl\_auth.grid(row=0,column=0,columnspan=2)

lbl\_pass=Label(auth\_win,text="Password:")

lbl\_pass.grid(row=1,column=0)

tb\_pass=Entry(auth\_win,show='\*')

tb\_pass.grid(row=1,column=1)

btn\_auth=Button(auth\_win,text="Confirm",command=confirm\_pass)

btn\_auth.grid(row=2,column=0,columnspan=2)

auth\_win.bind('<Return>', lambda event:confirm\_pass())

auth\_win.mainloop()

conn.commit()

conn.close()

return()

def update\_info():

global user\_list

conn=sqlite3.connect("Database.db")

cur=conn.cursor()

cur.execute("SELECT rowid, \* FROM Users WHERE rowid = ?",(user\_list[0],))

user\_list=list(cur.fetchone())

user\_list[5]=eval(user\_list[5])

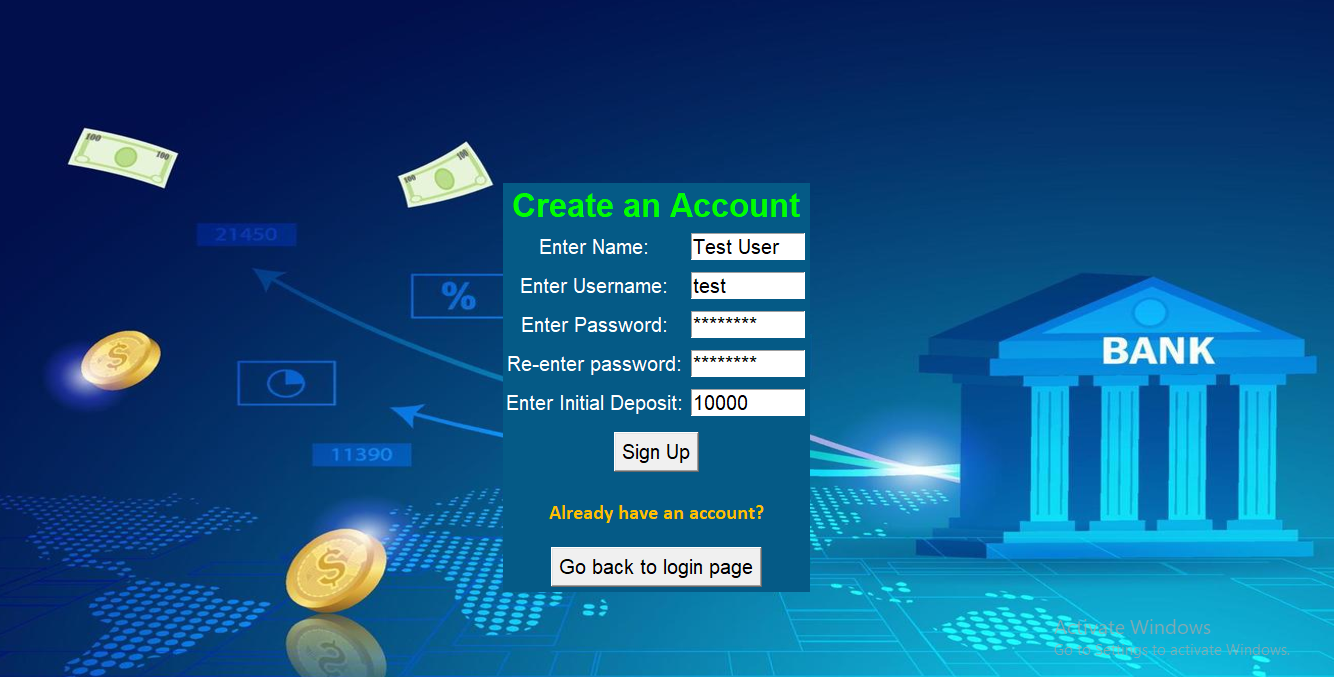
lbl\_balance.config(text=("Balance : "+str(user\_list[4])))

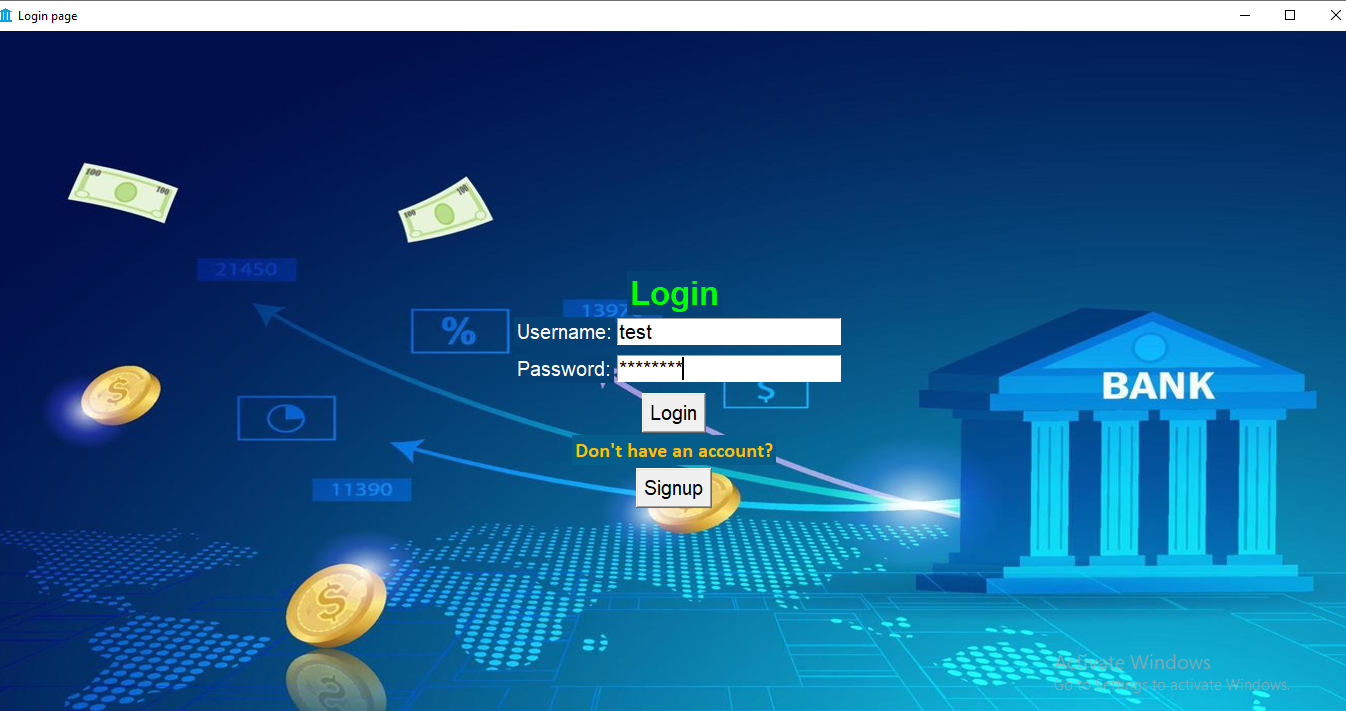
conn.commit()

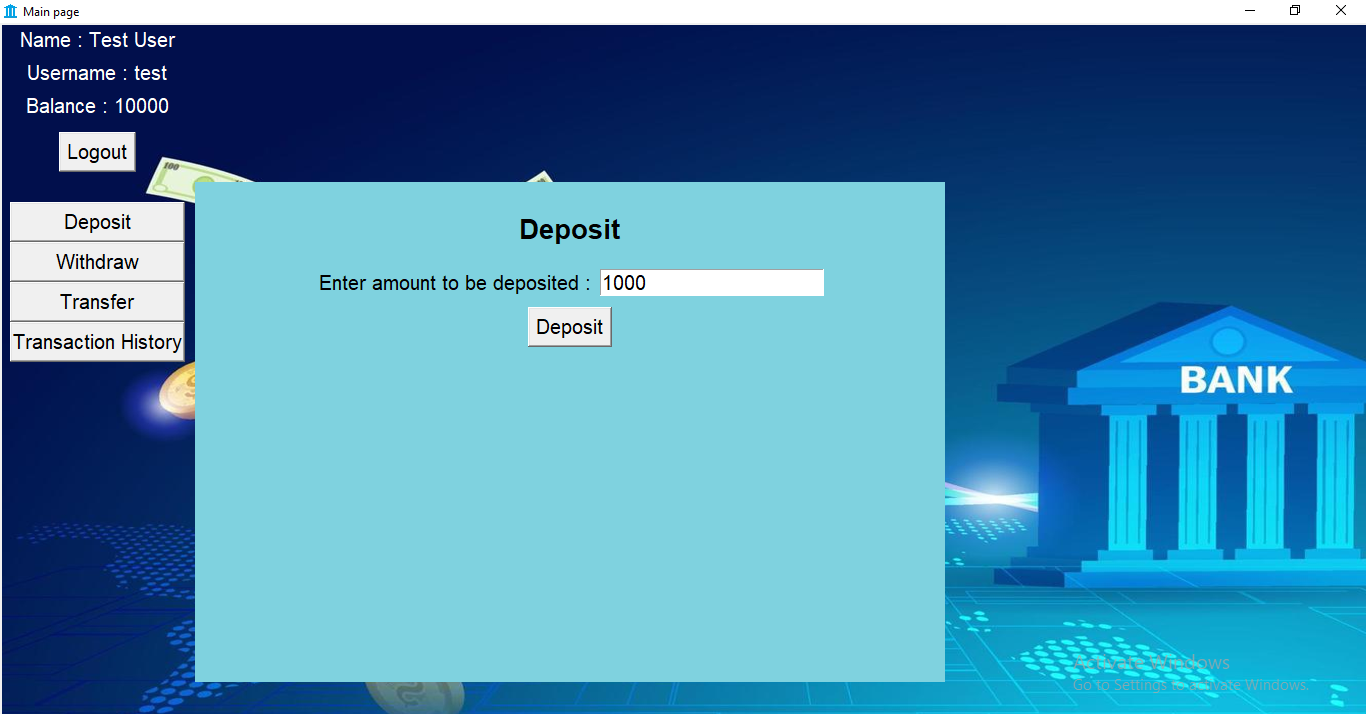
conn.close()

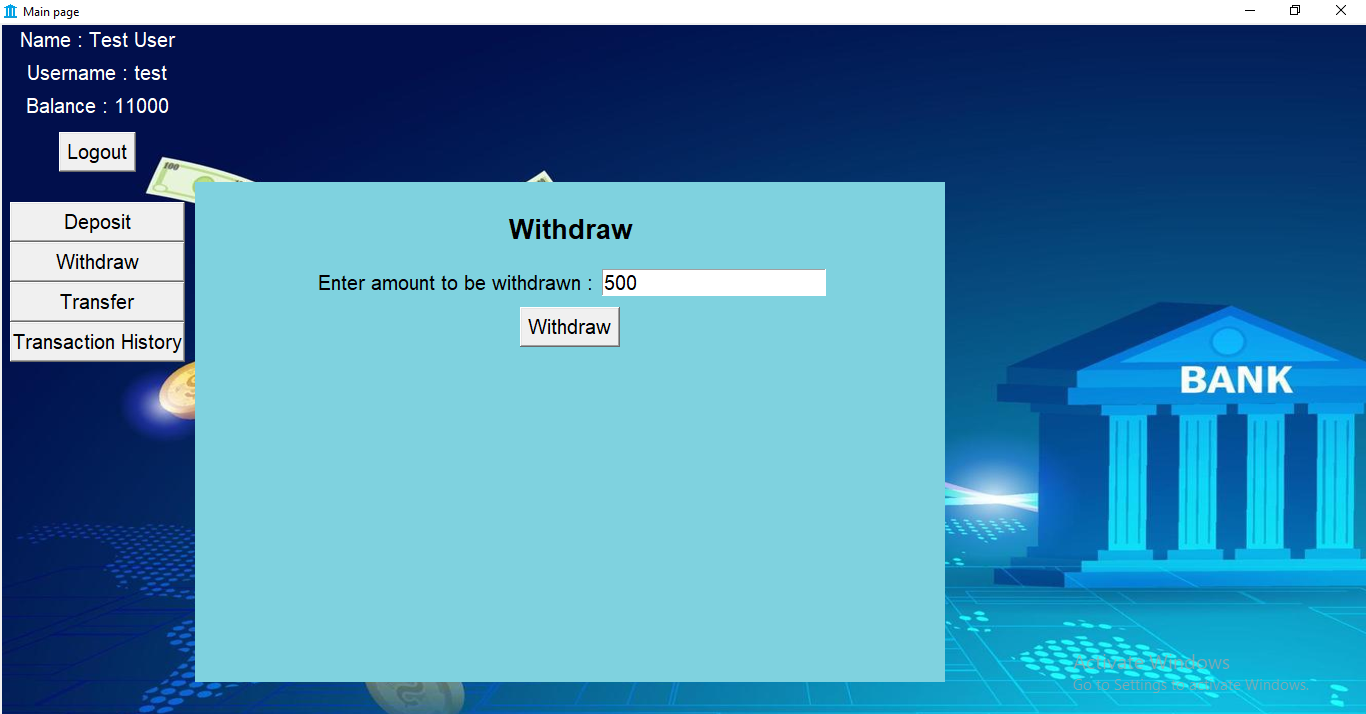
init()

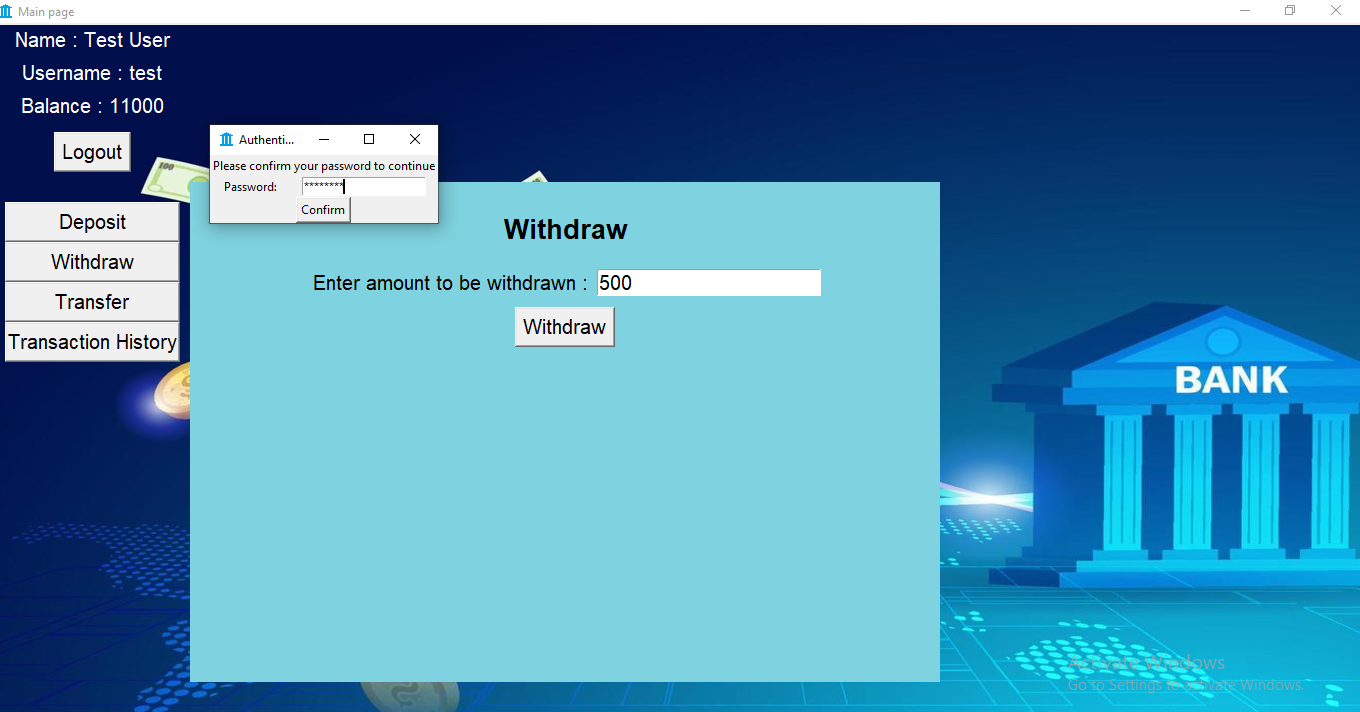
**OUTPUT SCREENSHOTS**

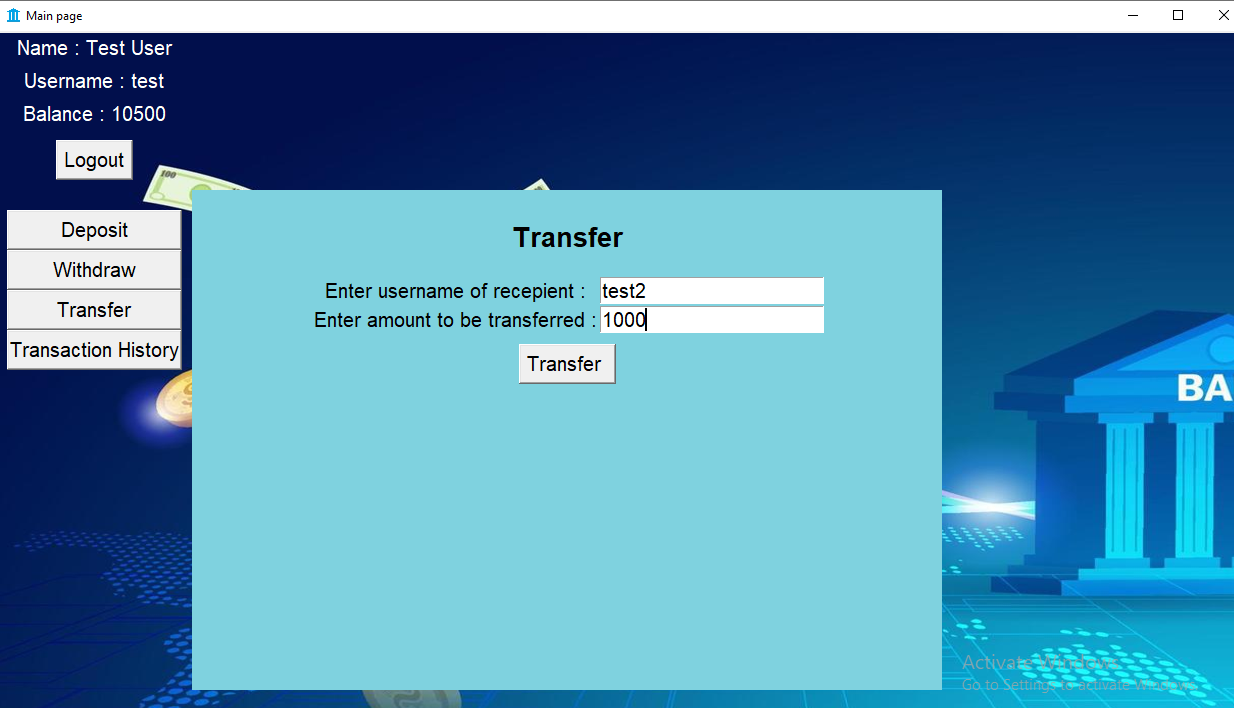


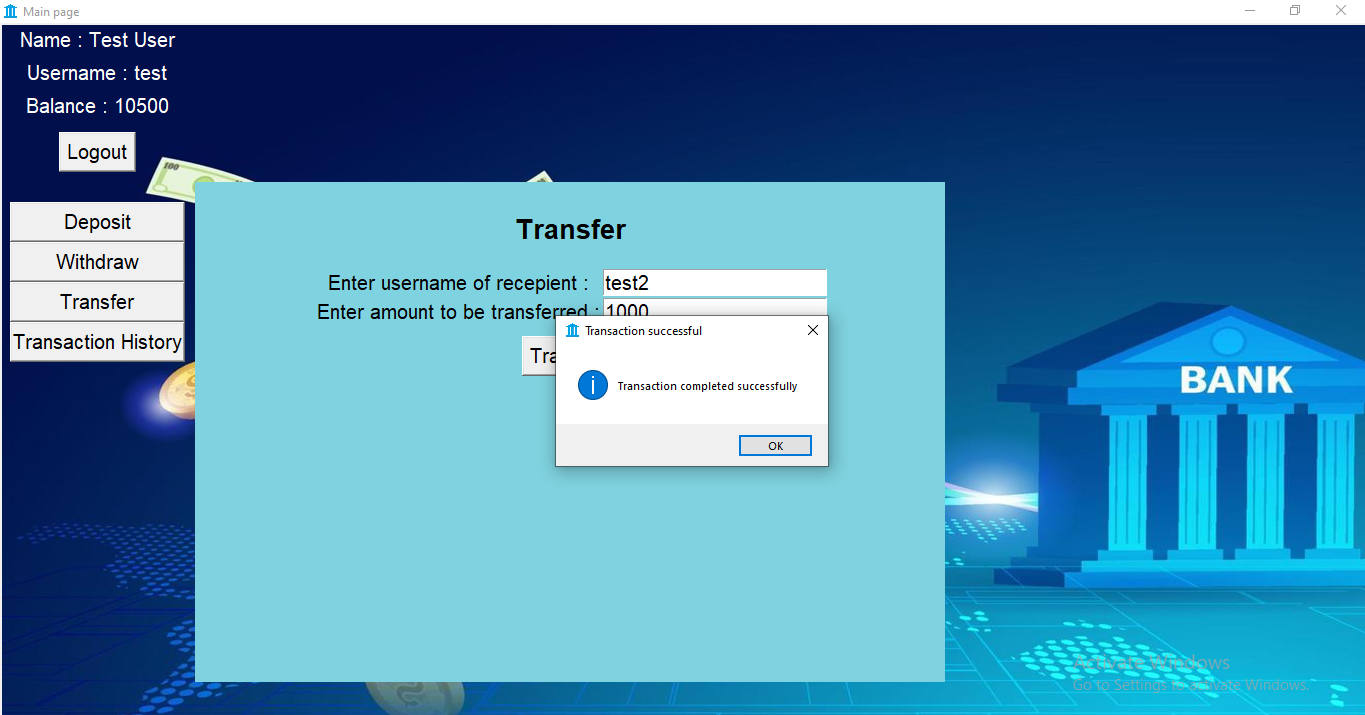


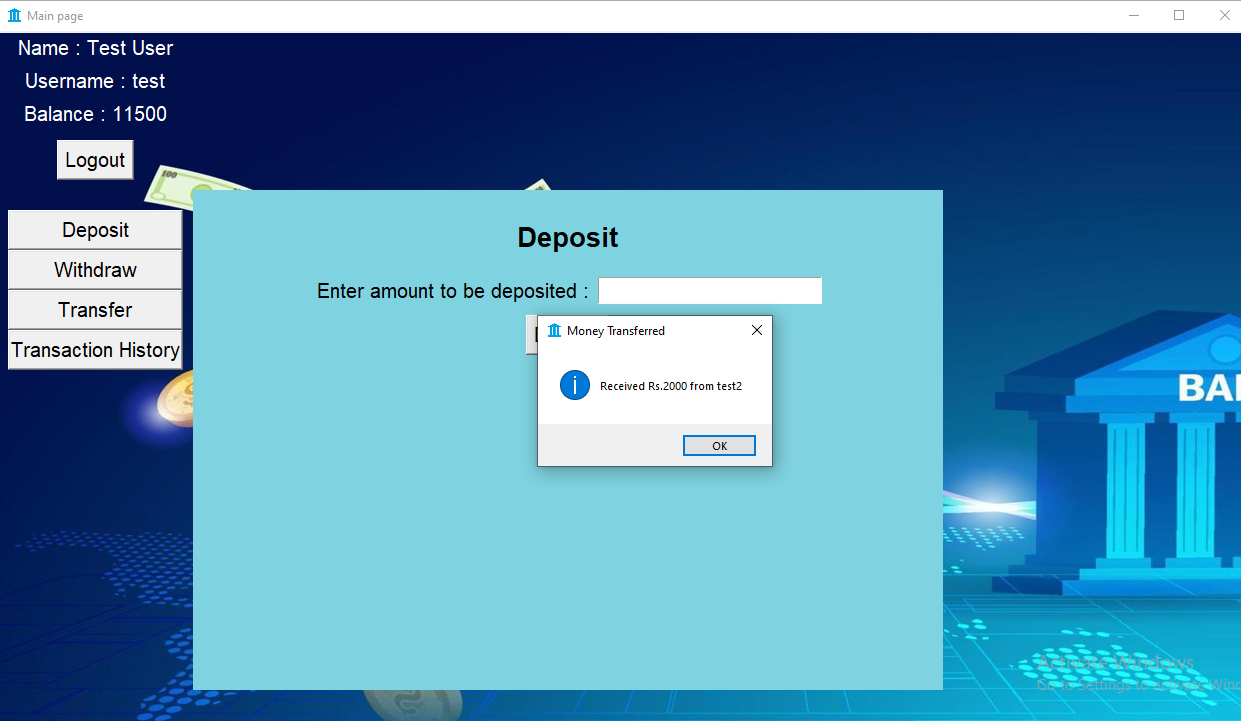


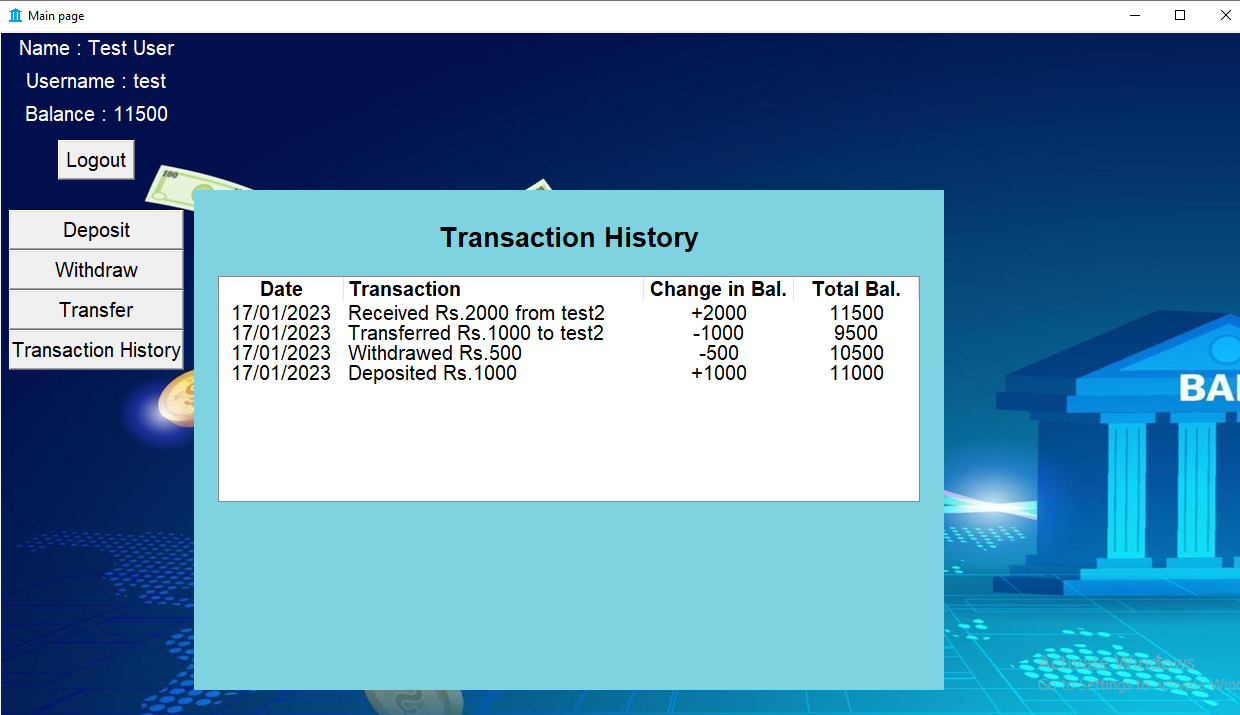












**FUTURE SCOPE**

The following enhancements can be easily added, to make this software even more useful:

1. Increase security of software.
2. Provide interest on deposited amount.
3. The ability to apply for loans.
4. Add Two-factor authentication with OTP through e-mail.
5. Creating a central server where database can be stored securely.
6. Increase the efficiency of code.

**REFERENCES**

1. <https://www.youtube.com/>
2. <https://www.geeksforgeeks.org/>
3. <https://stackoverflow.com/>
4. <https://www.tutorialspoint.com/index.htm>