**CONTENT:**

* **Introduction and synopsis**
* **System requirement**
* **Source code**
* **Sample output**

**INTRODUCTION AND SYNOPSIS:**

The Bank Account Management System is an application for maintaining a person's account in a bank.This project shows the working of a banking account system and cover the basic functionality of a Bank Account Management System.This is a prototype project which can be improved by using other programming languages like MySQL,PHP and HTML(Django/Flask). To develop a project for solving financial applications of a customer in banking environment in order to nurture the needs of an end banking user by providing various ways to perform banking tasks. Also to enable the user’s work space to have additional functionalities which are not provided under a conventional banking project. The Bank Account Management System undertaken as a project is based on relevant technologies. The main aim of this project is to develop a prototype software for Bank Account Management System. This project has been developed to carry out the processes easily and quickly, which is not possible with the manuals systems, which are overcomed by this project. This project is developed using Python language Creating and managing requirements is a challenge of IT, systems and product development projects or indeed for any activity where you have to manage a contractual relationship. Organization need to effectively define and manage requirements to ensure they are meeting needs of the customer, while proving compliance and staying on the schedule and within budget. The impact of a poorly expressed requirement can bring a business out of compliance or even cause injury or death. Requirements definition and management is an activity that can deliver a high, fast return on investment. The system is designed as an interactive and content management system. The content management system deals with data entry, validation confirm and updating whiles the interactive system deals with system interaction with the administration and users. Thus, above features of this project will save transaction time and therefore increase the efficiency of the system.

**SYSTEM REQUIREMENT:**

**HARDAWARE REQUIREMENT:**

* **Processor –** Intel(R)core(TM)i3-3120M CPU @ 2.50GHZ
* **Manufacture –** Intel
* **Clock–** 2.5GHZ
* **Number Of Cores –** 4
* **Family – 06**
* **Model –** 3A
* **Stepping –** 9
* **RAM –** 4.0GB
* **Video card -** Intel(R)HD Graphics 4000
* **Chipset -** Intel(R)HD Graphics 4000
* **Dedicated Memory –** 32MB
* **Total Memory –** 2.0GB

**SOFTWARE REQUIREMENT:**

* **Operating System –** Windows 10
* **Service Pack –** 0
* **Size –** 32-bit
* **Version –** 10.0.18362
* **Locale –** 0409
* **Code Editor –** Visual Basic Studio Code 2019
* **Interpreter –** Python IDLE 3.8(32-bit)

**SOURCE CODE:**

from tkinter import \*

import os

from PIL import ImageTk, Image

#Main Screen

master = Tk()

master.title('Bank Management')

#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 == "":

        notif.config(fg="red",text="All fields requried \* ")

        return

    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')

    #Labels

    Label(register\_screen, text="Please enter your details below to register", font=('Algerian',12)).grid(row=0,sticky=N,pady=10)

    Label(register\_screen, text="Name", font=('Centaur',12)).grid(row=1,sticky=W)

    Label(register\_screen, text="Age", font=('Centaur',12)).grid(row=2,sticky=W)

    Label(register\_screen, text="Gender", font=('Centaur',12)).grid(row=3,sticky=W)

    Label(register\_screen, text="Password", font=('Centaur',12)).grid(row=4,sticky=W)

    notif = Label(register\_screen, font=('Centaur',12))

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

    #Entries

    Entry(register\_screen,textvariable=temp\_name).grid(row=1,column=0)

    Entry(register\_screen,textvariable=temp\_age).grid(row=2,column=0)

    Entry(register\_screen,textvariable=temp\_gender).grid(row=3,column=0)

    Entry(register\_screen,textvariable=temp\_password,show="\*").grid(row=4,column=0)

    #Buttons

    Button(register\_screen, text="Register", command = finish\_reg, font=('Centaur',12)).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')

                #Labels

                Label(account\_dashboard, text="Account Dashboard", font=('Algerian',12)).grid(row=0,sticky=N,pady=10)

                Label(account\_dashboard, text="Welcome "+name, font=('Algerian',12)).grid(row=1,sticky=N,pady=5)

                #Buttons

                Button(account\_dashboard, text="Personal Details",font=('Centaur',12),width=30,command=personal\_details).grid(row=2,sticky=N,padx=10)

                Button(account\_dashboard, text="Deposit",font=('Centaur',12),width=30,command=deposit).grid(row=3,sticky=N,padx=10)

                Button(account\_dashboard, text="Withdraw",font=('Centaur',12),width=30,command=withdraw).grid(row=4,sticky=N,padx=10)

                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')

    #Label

    Label(deposit\_screen, text="Deposit", font=('Calibri',12)).grid(row=0,sticky=N,pady=10)

    current\_balance\_label = Label(deposit\_screen, text="Current Balance : ₹"+details\_balance, font=('Centaur',12))

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

    Label(deposit\_screen, text="Amount : ", font=('Centaur',12)).grid(row=2,sticky=W)

    deposit\_notif = Label(deposit\_screen,font=('Centaur',12))

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

    #Entry

    Entry(deposit\_screen, textvariable=amount).grid(row=2,column=1)

    #Button

    Button(deposit\_screen,text="Finish",font=('Centaur',12),command=finish\_deposit).grid(row=3,sticky=W,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')

    #Label

    Label(withdraw\_screen, text="Deposit", font=('Centaur',12)).grid(row=0,sticky=N,pady=10)

    current\_balance\_label = Label(withdraw\_screen, text="Current Balance : ₹"+details\_balance, font=('Centaur',12))

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

    Label(withdraw\_screen, text="Amount : ", font=('Centaur',12)).grid(row=2,sticky=W)

    withdraw\_notif = Label(withdraw\_screen,font=('Centaur',12))

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

    #Entry

    Entry(withdraw\_screen, textvariable=withdraw\_amount).grid(row=2,column=1)

    #Button

    Button(withdraw\_screen,text="Finish",font=('Centaur',12),command=finish\_withdraw).grid(row=3,sticky=W,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')

    #Labels

    Label(personal\_details\_screen, text="Personal Details", font=('Centaur',12)).grid(row=0,sticky=N,pady=10)

    Label(personal\_details\_screen, text="Name : "+details\_name, font=('Centaur',12)).grid(row=1,sticky=W)

    Label(personal\_details\_screen, text="Age : "+details\_age, font=('Centaur',12)).grid(row=2,sticky=W)

    Label(personal\_details\_screen, text="Gender : "+details\_gender, font=('Centaur',12)).grid(row=3,sticky=W)

    Label(personal\_details\_screen, text="Balance :₹"+details\_balance, font=('Centaur',12)).grid(row=4,sticky=W)

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')

    #Labels

    Label(login\_screen, text="Login to your account", font=('Centaur',12)).grid(row=0,sticky=N,pady=10)

    Label(login\_screen, text="Username", font=('Centaur',12)).grid(row=1,sticky=W)

    Label(login\_screen, text="Password", font=('Centaur',12)).grid(row=2,sticky=W)

    login\_notif = Label(login\_screen, font=('Centaur',12))

    login\_notif.grid(row=4,sticky=N)

    #Entry

    Entry(login\_screen, textvariable=temp\_login\_name).grid(row=1,column=1,padx=5)

    Entry(login\_screen, textvariable=temp\_login\_password,show="\*").grid(row=2,column=1,padx=5)

    #Button

    Button(login\_screen, text="Login", command=login\_session, width=15,font=('Centaur',12)).grid(row=3,sticky=W,pady=5,padx=5)

#Image import

img = Image.open('#bank.png')

img = img.resize((150,150))

img = ImageTk.PhotoImage(img)

#Labels

Label(master, text = "BANKING", font=('Centaur',14)).grid(row=0,sticky=N,pady=10)

Label(master, text = "-BETA VERSION:)", font=('Centaur',12)).grid(row=1,sticky=N)

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

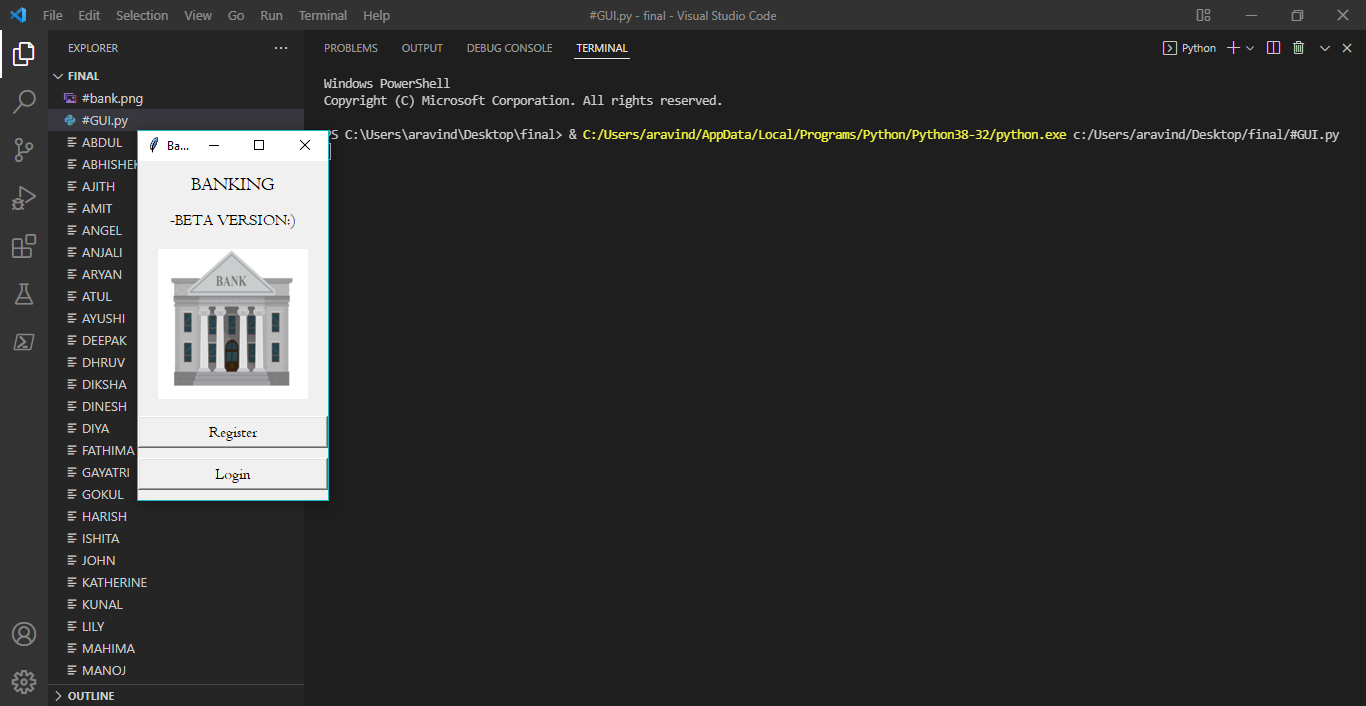
#Buttons

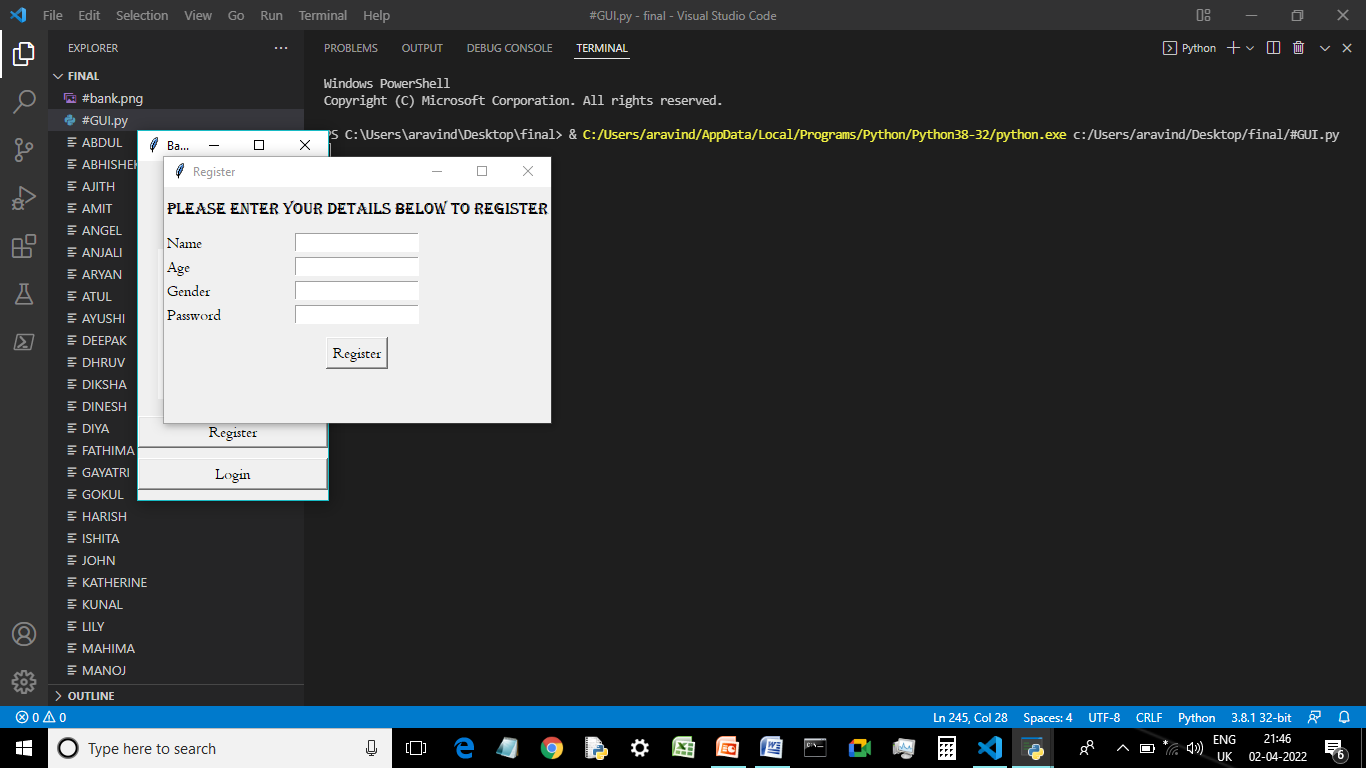
Button(master, text="Register", font=('Centaur',12),width=20,command=register).grid(row=3,sticky=N)

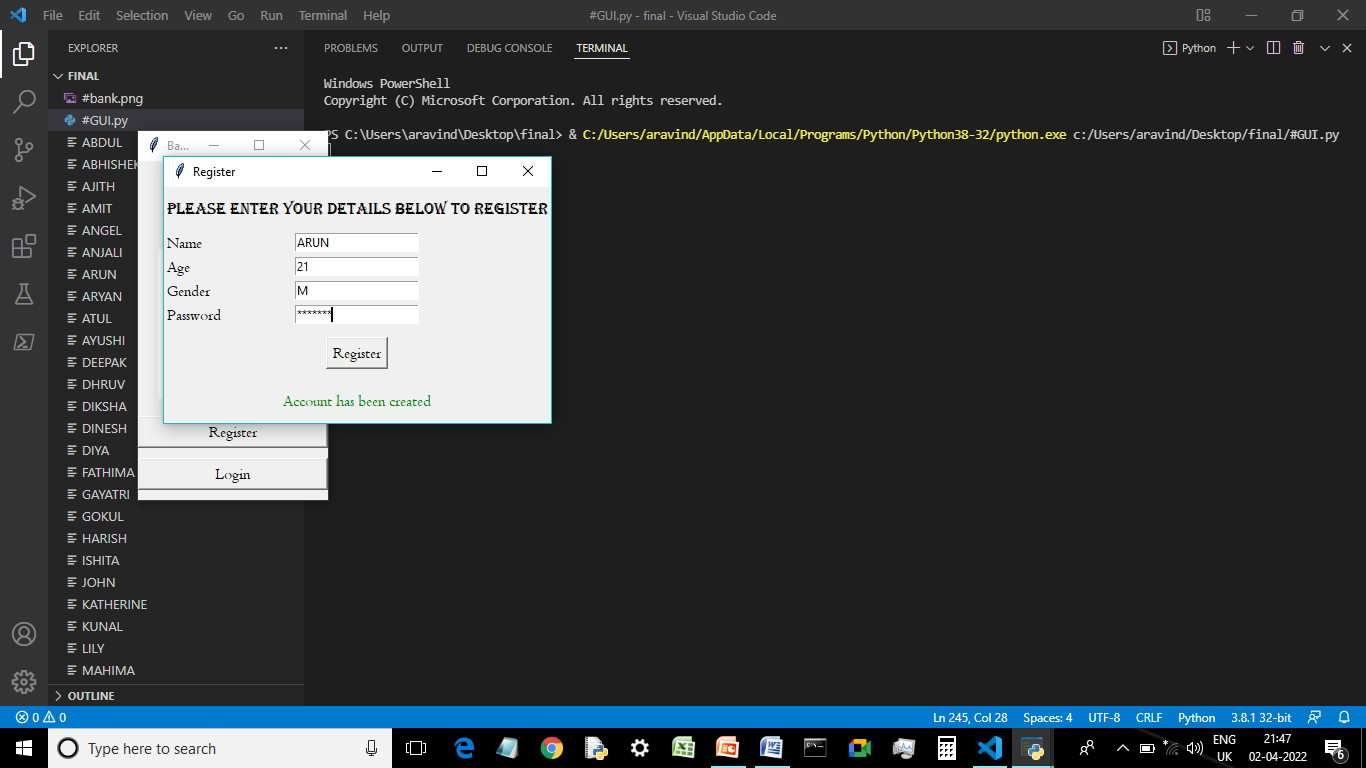
Button(master, text="Login", font=('Centaur',12),width=20,command=login).grid(row=4,sticky=N,pady=10)

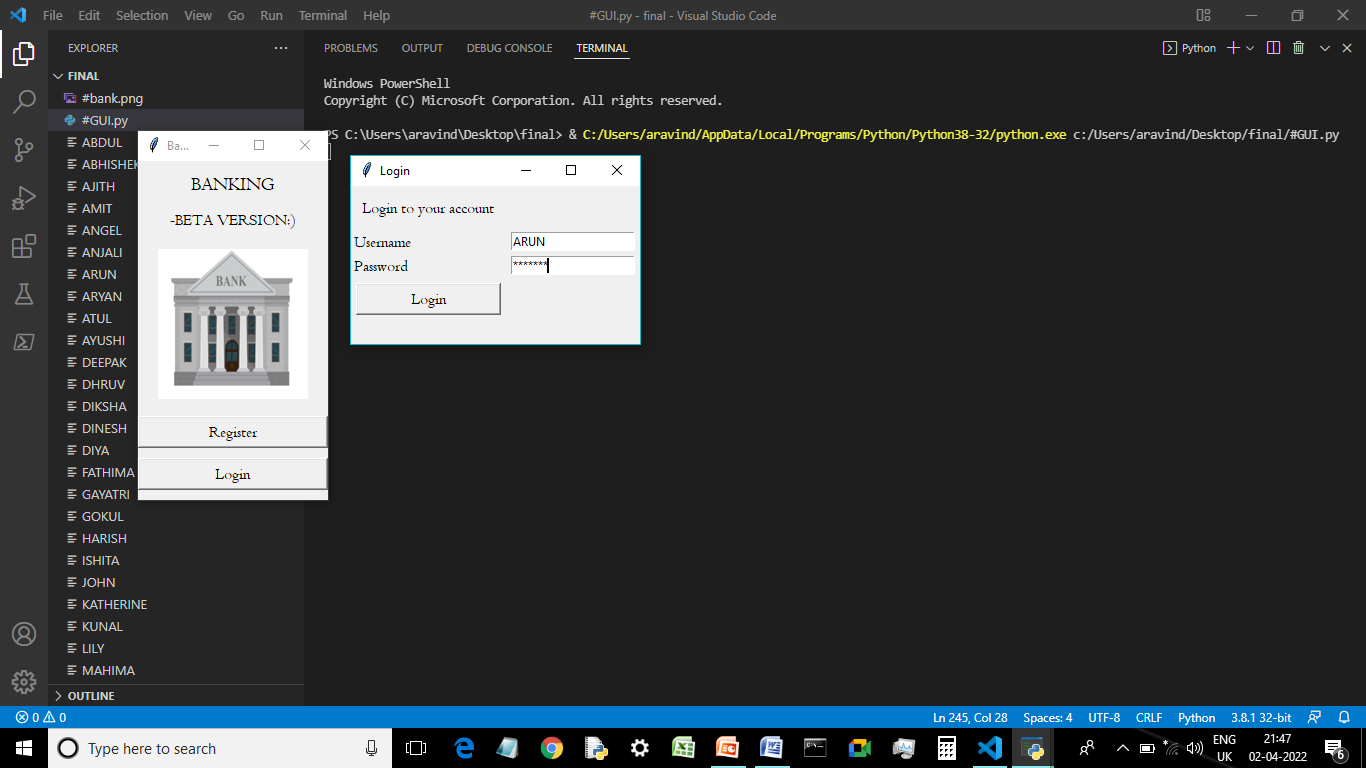
master.mainloop()

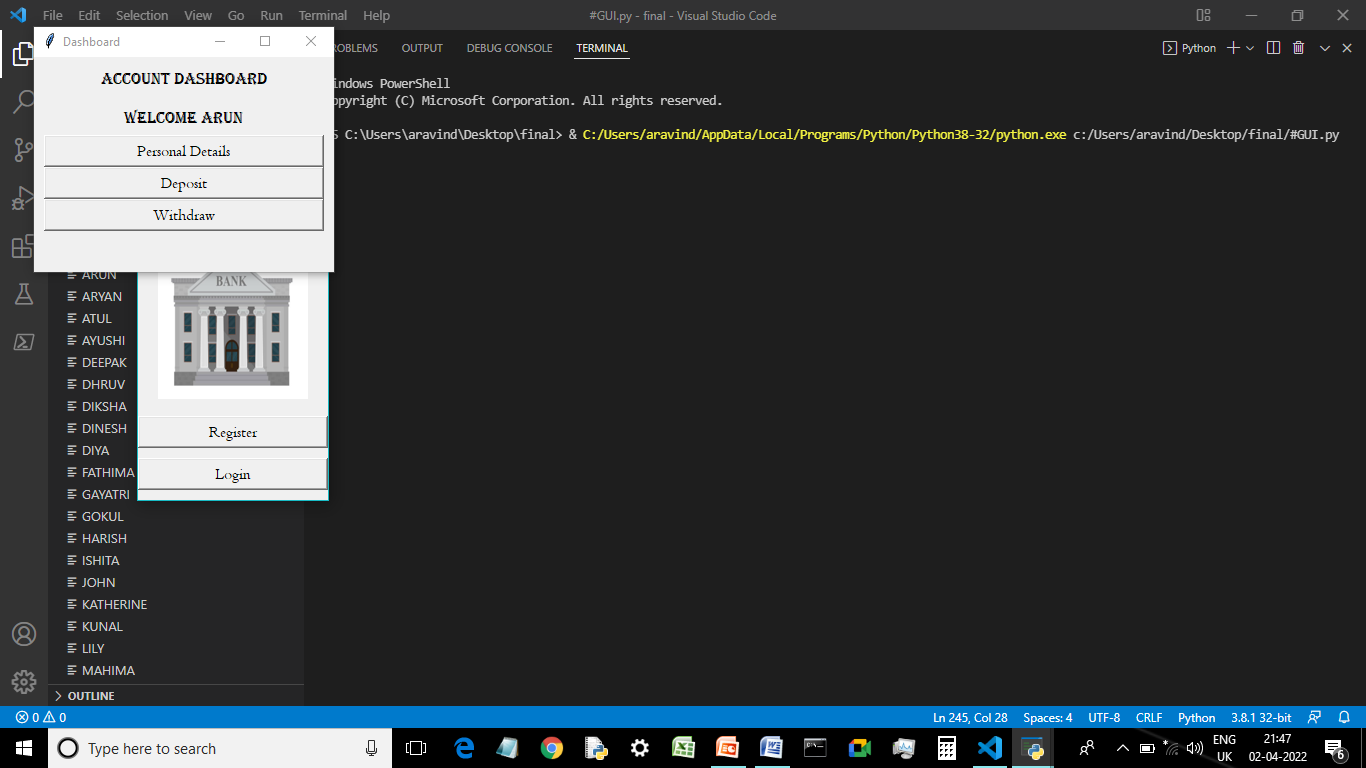
**SAMPLE OUTPUT:**

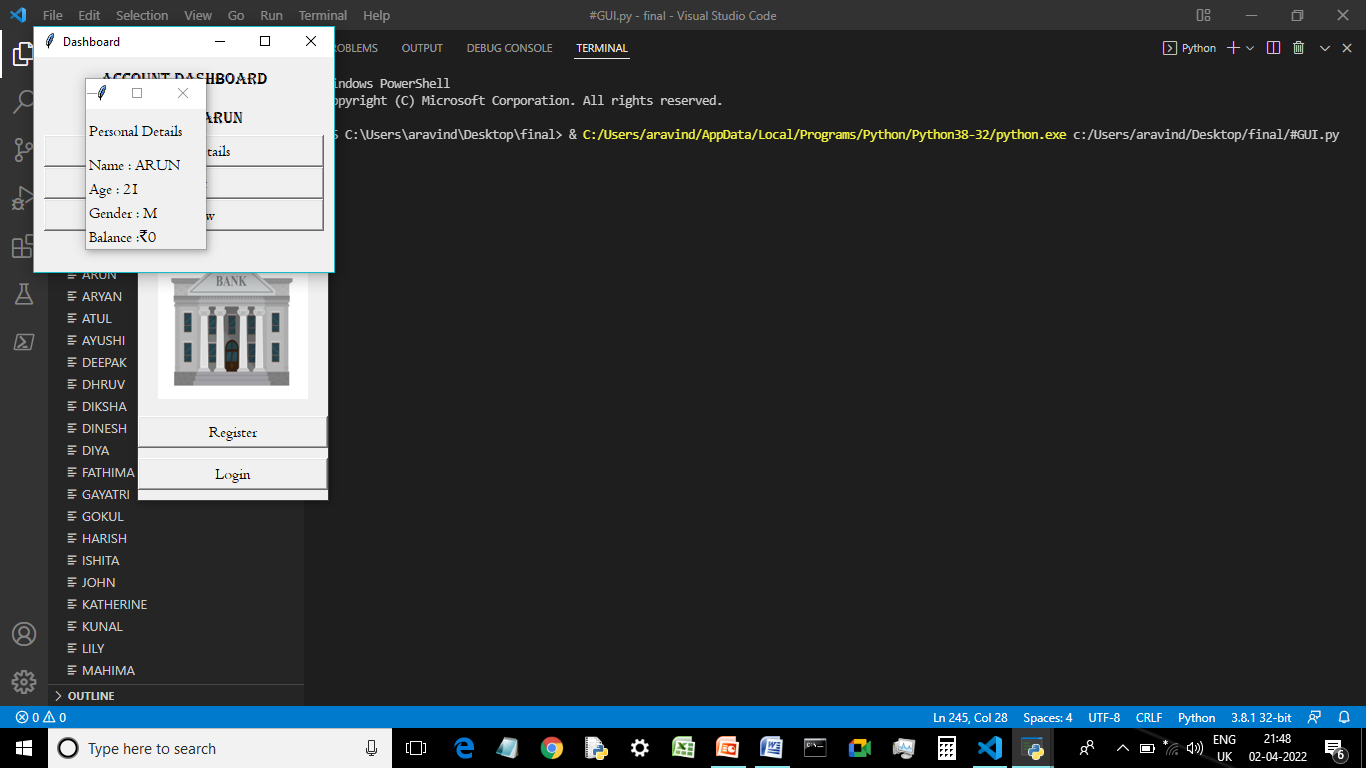
****

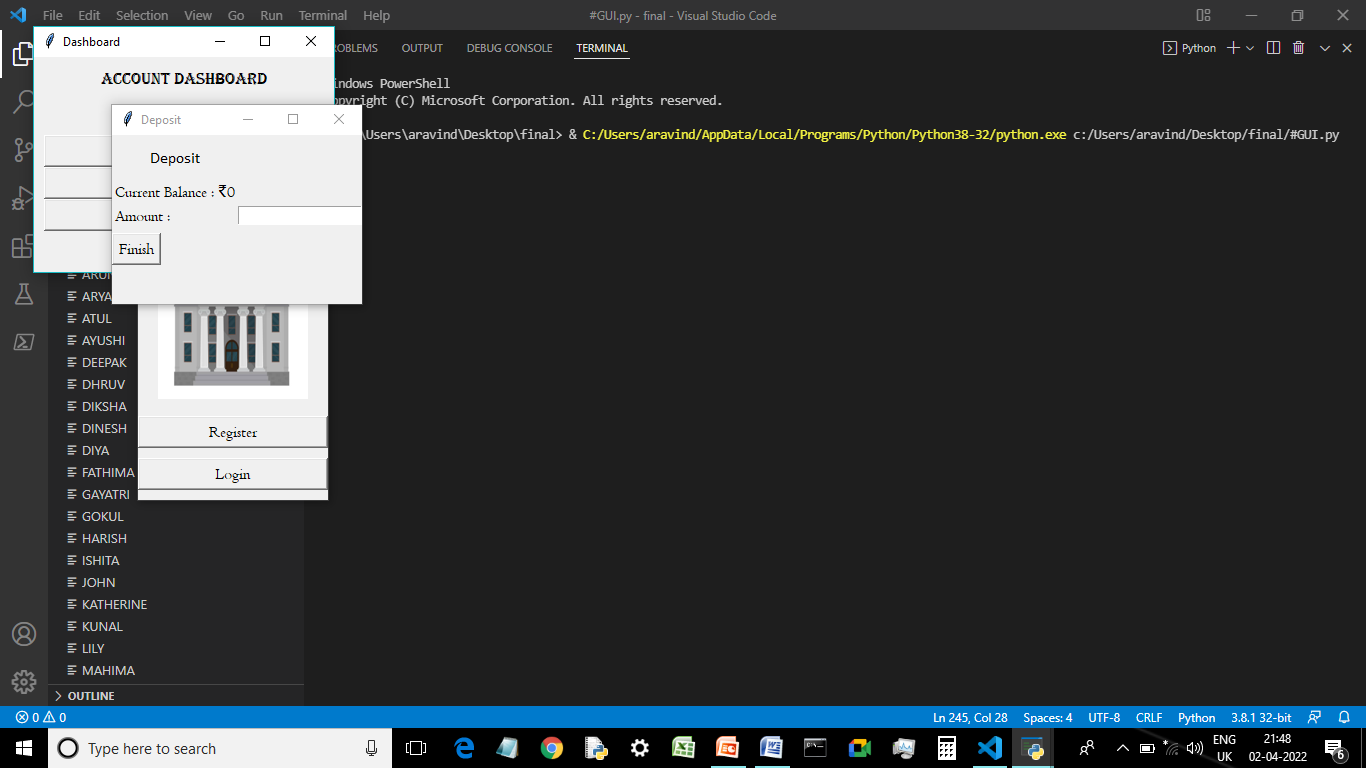
****

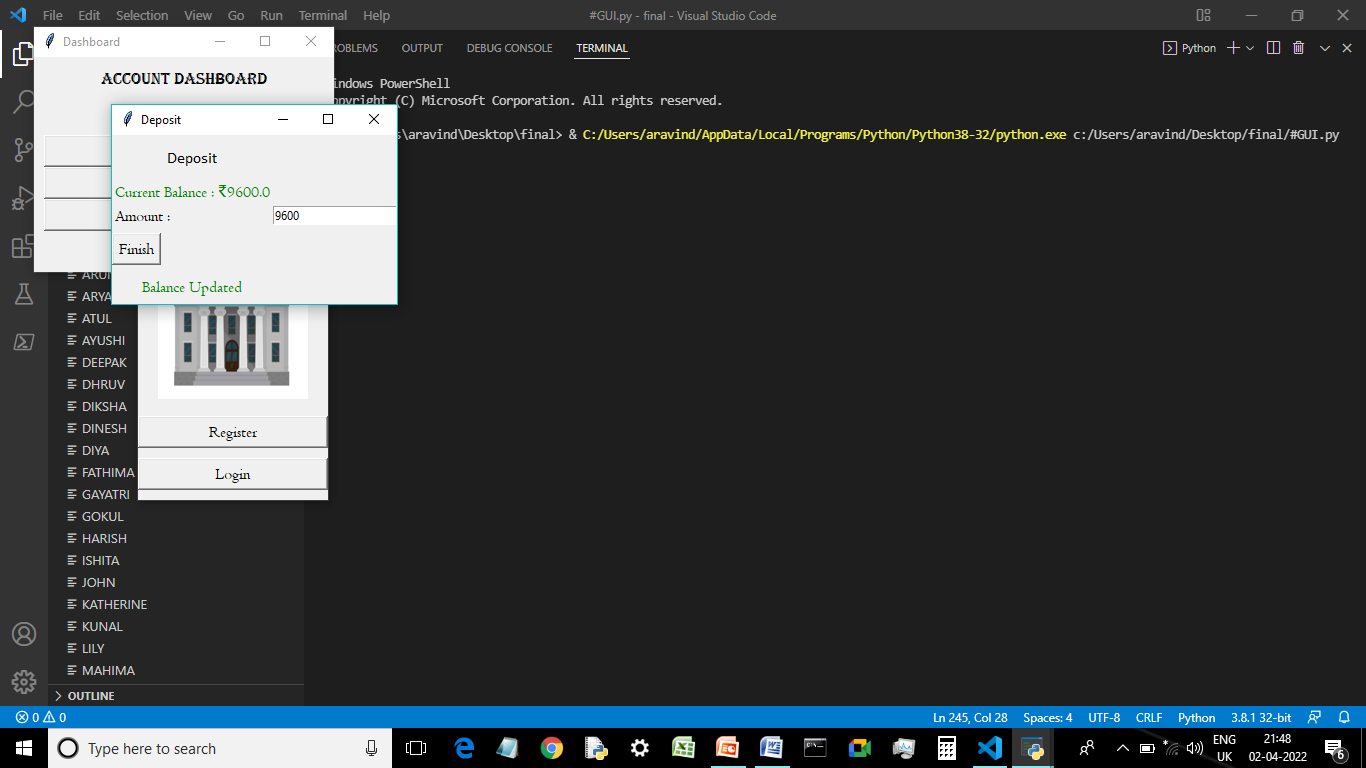
****

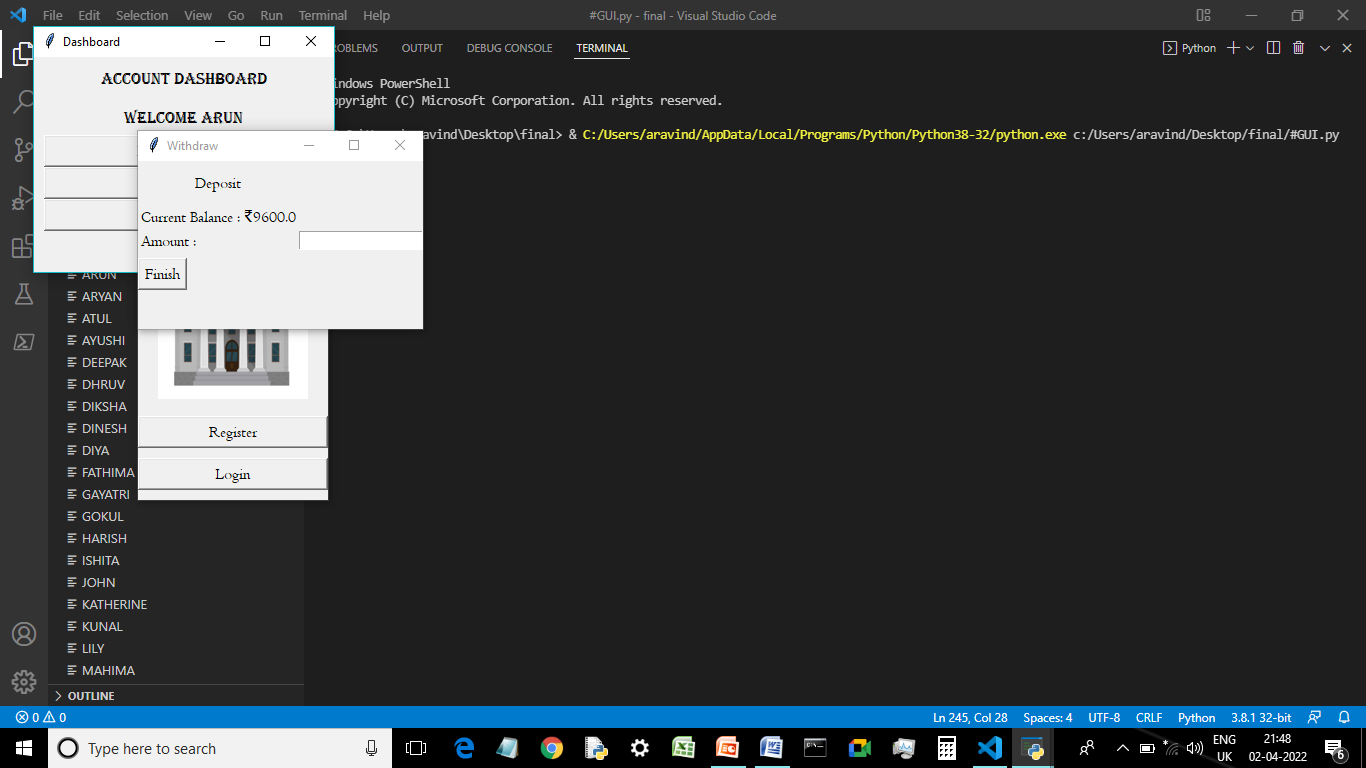
****

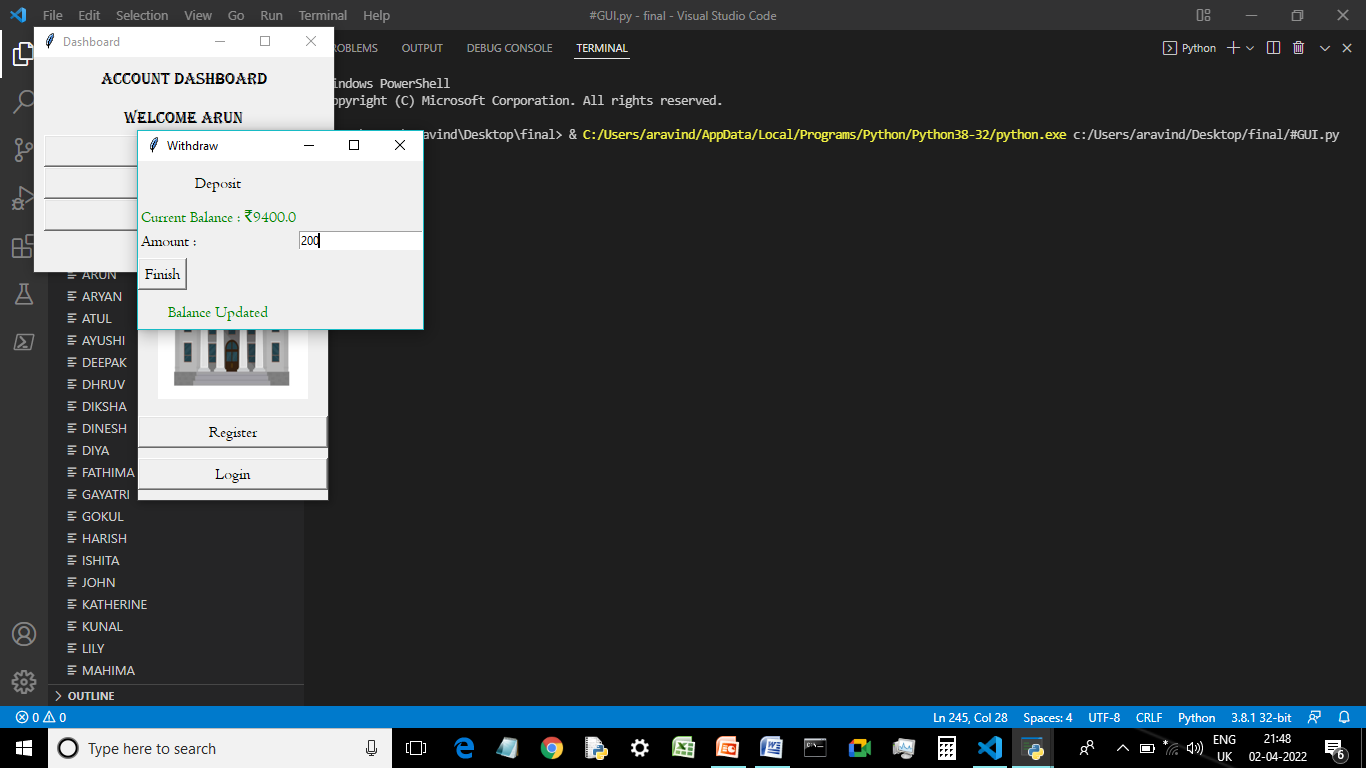
****

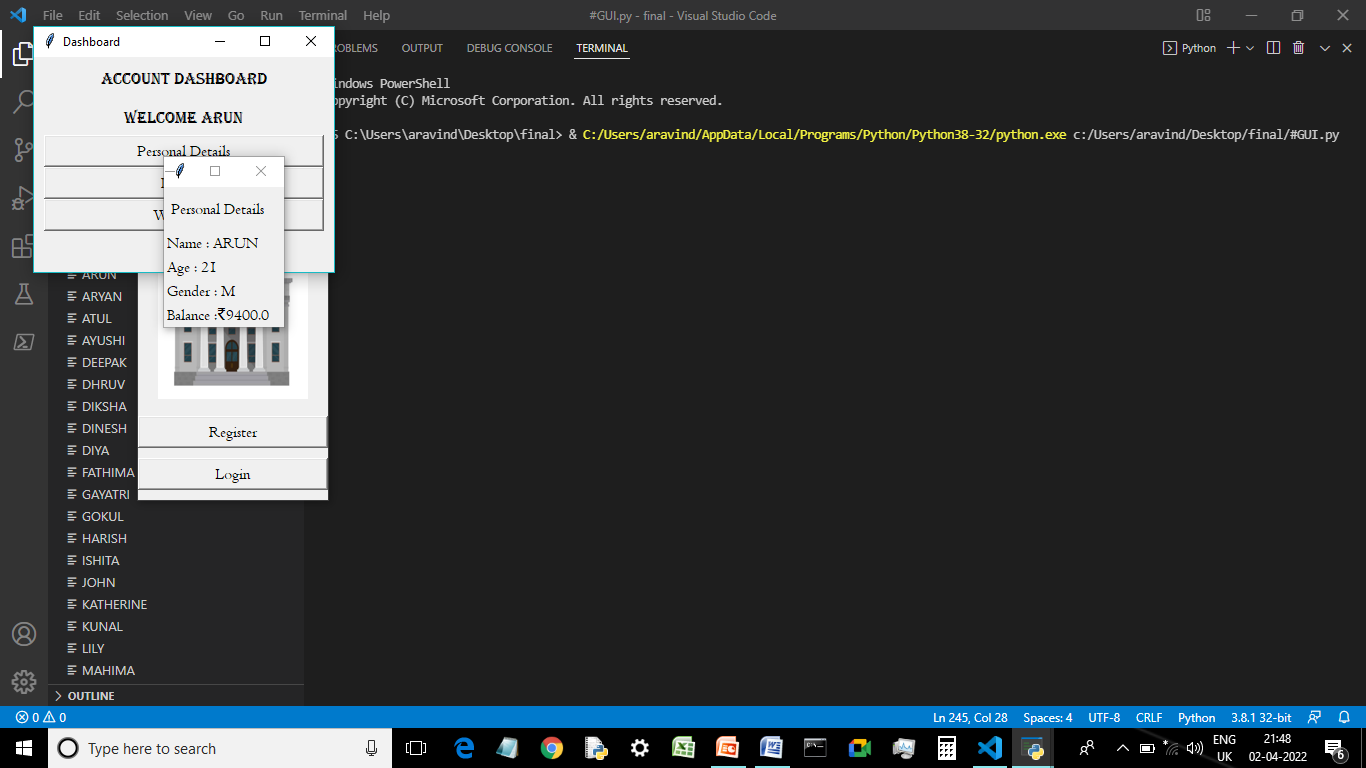
****

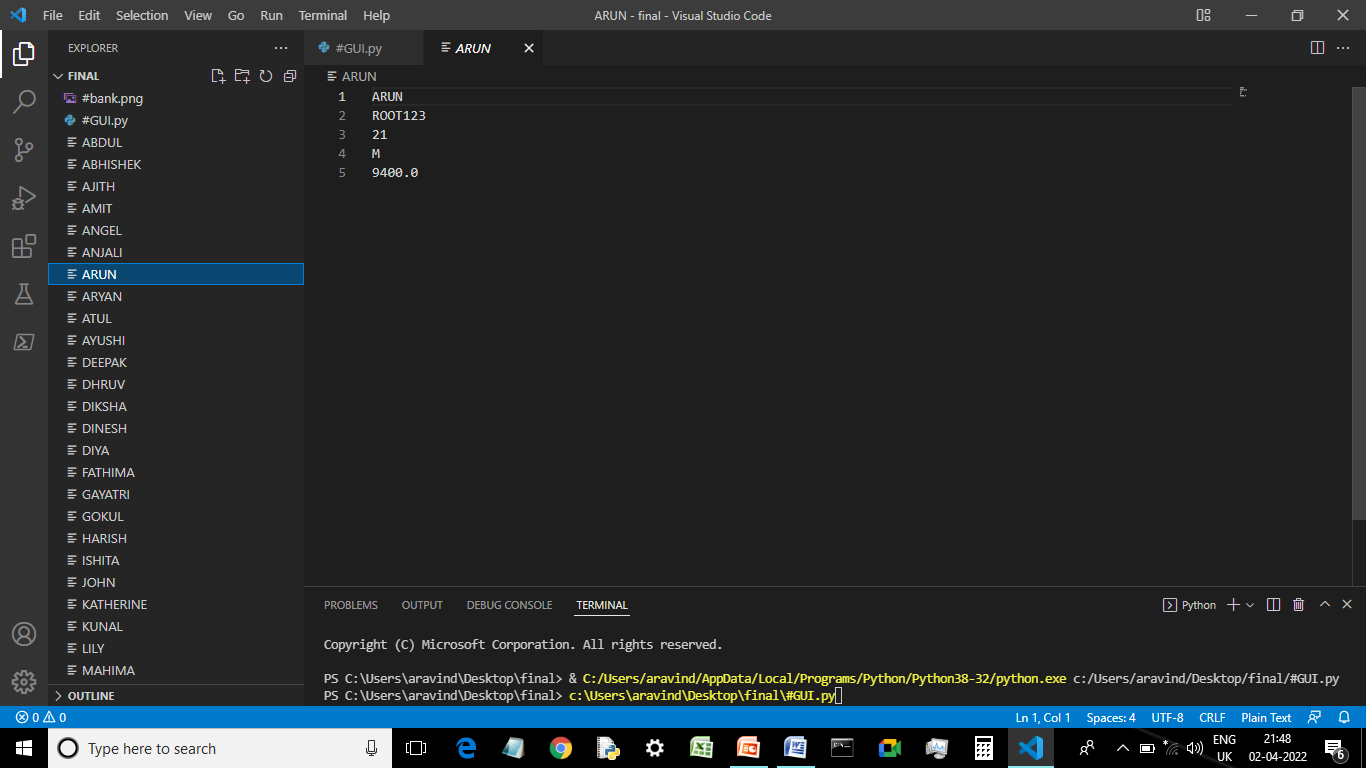
****

****

****

****

****

****