iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** III **Month of publication:** March 2022

DOI: https://doi.org/10.22214/ijraset.2022.40653

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue III Mar 2022- Available at www.ijraset.com

Orderzy: Retail Management System

Sanyam Aware[1], Rashi Agarwal[2], Prof. Gurunath Waghale[3]
[1, 2]*Department of Computer Science & Engineering, MIT School of Engineering, Loni-Kalbhor, Pune*
[3]*Guide*

Abstract: ***Retail management refers to the process which helps the customers to purchase their desired resources from any sort of store service. Online Retail Management System allows the customers to have a good experience and leave the store with a smile. It helps them to shop without any difficulty. It saves the time of customers since they will be able to buy their desired items in one stretch. One can design a 'Retail Management' application to help a store gain more customers by increasing their shopping experience to a greater extent. Through this project, the customer will be able to easily login to the application after filling a short and easy registration form, and will be issued with a unique password each time he/she logs in, which is also one of our USP, so in this way no one would have to remember a fixed password, they won't have to go through long processes to retrieve or create a new password. Your password will be a click away. Browsing through the application is amazingly easy as we have provided different categories on the main screen which on a click show the products available in that category. We at Orderzy, provide our very own payment method through a wallet that can store an amount you would like to use for future purchases.***

I. INTRODUCTION

A. *Problem Definition*

Python Project on Retail Management System using SQL

The Retail Store Management System is a system designed for managing i.e., for ordering, arranging and selling goods using E-commerce portal. In this project we have worked on how a user can purchase on a commercial website like Orderzy with different ranges of products and prices. To store the data, we have used SQL along with Tkinter for GUI.

B. *What our program basically is…*

Electronic commerce, or eCommerce, refers to the purchasing and selling of goods or services via electronic means, such as the Internet or mobile phone applications. It may also refer to the process of creating, marketing, servicing and paying for services and goods. Businesses, governments and the public can participate in eCommerce transactions.

II. IMPLEMENTATION

A. *Language Used*

1) *Python:* Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

B. *Technology Used*

1) *IDE(PYCHARM)*

a) *PyCharm:* It is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. PyCharm is cross-platform, with Windows, macOS and Linux versions, but IDE may also be used to develop applications in other programming languages via plug-ins, including Ada, ABAP, C, C++, C#, Clojure, COBOL, D, Erlang, Fortran, Groovy, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, R, Ruby (including Ruby on Rails framework), Rust, Scala, and Scheme.

b) *Python for Artificial Intelligence:* We use Python because Python programs can be close to pseudo-code. It is designed for humans to read. Python is reasonably efficient. Efficiency is usually not a problem for small examples. If your Python code is not efficient enough, a general procedure to improve it is to find out what is taking most of the time, and implement just that part more efficiently in some lower-level language. Most of these lower-level languages interoperate with Python nicely. This will result in much less programming and more efficient code (because you will have more time to optimize) than writing everything in a low-level language. You will not have to do that for the code here if you are using it for course projects.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue III Mar 2022- Available at www.ijraset.com

c) *MySQL:* MySQL is the most popular Open-Source Relational Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. This tutorial will give you a quick start to MySQL and make you comfortable with MySQL programming.

d) *Operating System:* Microsoft Windows, commonly referred to as Windows, is a group of several proprietary graphical operating system families, all of which are developed and marketed by Microsoft. Active Microsoft Windows families include Windows NT and Windows IoT; these may encompass subfamilies, e.g. Windows Server or Windows Embedded Compact.

2) *Python TKINTER(GUI):* 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 GUI applications. Creating a GUI using tkinter is an easy task. Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

III. PYTHON CODE & OUTPUT

A. *Python Code for Registration Page*

```
import ttkthemes
from tkinter import *
from tkinter import ttk
from tkinter.ttk import Combobox
from tkinter import messagebox
import pymysql


                ######################    FUNCTIONS    ######################

def clear():
    first_name_entry.delete(0,END)
    surname_entry.delete(0, END)
    emailid_entry.delete(0,END)
    contact_num_entry.delete(0,END)
    day.current(0)
    month.current(0)
    year.current(0)
    address_entry_1.delete(0,END)
    address_entry_2.delete(0, END)
    city_entry.delete(0,END)
    pincode_entry.delete(0,END)


def sign_up():
    if first_name_entry.get()=='' or surname_entry.get()=='' or emailid_entry.get()=='\
        or contact_num_entry.get()=='' or day.get()=='DAY' or month.get()=='MONTH'\
        or year.get()=='YEAR' or address_entry_1.get()=='' or address_entry_2.get()=='' or city_entry.get()=='\
        or pincode_entry.get()=='':
        messagebox.showerror('Error', 'All fields are required')
```

```
    else:
        try:
            app_database_object = pymysql.connect(host='localhost', user='root', password='root', database='user_credentials')
            app_database = app_database_object.cursor()
            app_database.execute('select * from user_credentials where user_email_id=%s', emailid_entry.get())
            row_1 = app_database.fetchone()
            app_database.execute('select * from user_credentials where user_contact_num=%s', contact_num_entry.get())
            row_2 = app_database.fetchone()
            if row_1 != None:
                messagebox.showerror('Error', 'Customer email already exists')

            elif row_2 !=None:
                messagebox.showerror('Error', 'Contact number already exists')

            else:
                default_wallet_balance = 300000
                dob = (year.get() + "-" + month.get()+ "-" + day.get())
                address = (address_entry_1.get() + " " + address_entry_2.get())
                val = (first_name_entry.get(), surname_entry.get(), emailid_entry.get(), contact_num_entry.get(), dob, address,
city_entry.get(), pincode_entry.get())
                app_database.execute('insert into user_credentials(first_name, last_name, user_email_id, user_contact_num, user_dob,
user_address, user_city, user_area_pincode) values(%s,%s,%s,%s,%s,%s,%s,%s)', val)
                app_database.execute('insert into login_credentials(user_email,wallet_balance) values(%s,%s)', (emailid_entry.get(),
default_wallet_balance))
                app_database_object.commit()
                app_database_object.close()

                messagebox.showinfo('Success', 'Successfully signed up')
                clear()


        except Exception as e:
                messagebox.showerror('Error', f'Error due to {e}')

        main_window.destroy()
        import login_page




                    ############################    GUI    ############################

main_window = ttkthemes.ThemedTk()
main_window.get_themes()
main_window.set_theme('breeze')

main_window.geometry('550x595+350+30')
main_window.title('SIGN UP')
main_window.resizable(0,0)
```

```
credentials_frame = Frame(main_window, width=510, height=580)
credentials_frame.place(x=10, y=0)

title_label = Label(credentials_frame, text='Sign Up', font=('arial', 22, 'bold'), fg='midnightblue')
title_label.place(x=0, y=15)

sub_title_label = Label(credentials_frame, text="It's Quick & Easy.", font=('arial', 10, 'bold'), fg='midnightblue')
sub_title_label.place(x=0, y=50)

line_label = Label(credentials_frame, text="----------------------------------------", font=('arial', 10, 'bold'), fg='midnightblue')
line_label.place(x=0, y=70)

first_name_label = Label(credentials_frame, text="First name", font=('times new roman', 14, 'bold'), fg='midnightblue')
first_name_label.place(x=0, y=100)

first_name_entry = ttk.Entry(credentials_frame, font=("times new roman", 12), width=25)
first_name_entry.place(x=0, y=128)

surname_label = Label(credentials_frame, text="Last name", font=('times new roman', 14, 'bold'), fg='midnightblue')
surname_label.place(x=293, y=100)

surname_entry = ttk.Entry(credentials_frame, font=("times new roman", 12), width=25)
surname_entry.place(x=293, y=128)

emailid_label = Label(credentials_frame, text="Email ID", font=('times new roman', 14, 'bold'), fg='midnightblue')
emailid_label.place(x=0, y=170)

emailid_entry = ttk.Entry(credentials_frame, font=("times new roman", 12), width=25)
emailid_entry.place(x=0, y=198)

contact_num_label = Label(credentials_frame, text="Contact Number", font=('times new roman', 14, 'bold'), fg='midnightblue')
contact_num_label.place(x=293, y=170)

contact_num_entry = ttk.Entry(credentials_frame, font=("times new roman", 12), width=25)
contact_num_entry.place(x=293, y=198)

DOB_label = Label(credentials_frame, text="Date Of Birth", font=('times new roman', 14, 'bold'), fg='midnightblue')
DOB_label.place(x=0, y=240)

day = ttk.Combobox(credentials_frame, font=('times new roman', 12), state='readonly')
day['values'] = ('DAY','01','02','03','04','05','06','07','08','09','10',
                 '11','12','13','14','15','16','17','18','19','20',
                 '21','22','23','24','25','26','27','28','29','30','31')
day.place(x=0, y=268)
day.current(0)

month = ttk.Combobox(credentials_frame, font=('times new roman', 12), state='readonly')
month['values'] = ('MONTH','01','02','03','04','05','06','07','08','09','10','11','12')
```

```
month.place(x=165, y=268)
month.current(0)

year = ttk.Combobox(credentials_frame, font=('times new roman', 12), state='readonly')
year['values'] = ('YEAR', '1990', '1991', '1992', '1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000',
                  '2001', '2002', '2003', '2004', '2005', '2006', '2007', '2008', '2009', '2010',
                  '2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018', '2019', '2020')
year.place(x=320, y=268)
year.current(0)

address_label = Label(credentials_frame, text="Your Address", font=('times new roman', 14, 'bold'), fg='midnightblue')
address_label.place(x=0, y=308)

address_entry_1 = ttk.Entry(credentials_frame, font=("times new roman", 12), width=62)
address_entry_1.place(x=0, y=338)

address_entry_2 = ttk.Entry(credentials_frame, font=("times new roman", 12), width=62)
address_entry_2.place(x=0, y=372)

city_label = Label(credentials_frame, text="City", font=('times new roman', 14, 'bold'), fg='midnightblue')
city_label.place(x=0, y=408)

city_entry = ttk.Entry(credentials_frame, font=("times new roman", 12), width=25)
city_entry.place(x=0, y=437)

pincode_label = Label(credentials_frame, text="Pincode", font=('times new roman', 14, 'bold'), fg='midnightblue')
pincode_label.place(x=294, y=408)

pincode_entry = ttk.Entry(credentials_frame, font=("times new roman", 12), width=25)
pincode_entry.place(x=294, y=436)

T_and_C_label = Label(credentials_frame, text="By signing up, you agree with our Terms, Conditions and Privacy policies.",
font=('times new roman', 10), fg='midnightblue')
T_and_C_label.place(x=0, y=478)

T_and_C_label = Label(credentials_frame, text="You may receive email notifications from us and can opt out at any time.",
font=('times new roman', 10), fg='midnightblue')
T_and_C_label.place(x=0, y=498)

signup_button_img = ttk.Button(credentials_frame, text="Sign Up", cursor='hand2', command=sign_up)
signup_button_img.place(x=214, y=540)



main_window.mainloop()
```

B. Python Code for Login Page

```
import ttkthemes
from tkinter import *
from tkinter import ttk
import random
import smtplib
from tkinter import messagebox
import pymysql



                    ######################    FUNCTIONS    ######################

def generate_otp():
    password_generation_list = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q',
                                'R', 'S',
                                'T', 'U', 'V', 'W', 'X', 'Y', 'Z', '0', '1', '2', '3',
                                '4', '5',
                                '6', '7', '8', '9' 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',
                                'k', 'l', 'm',
                                'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
    char = random.choices(password_generation_list, k=8)

    otp = ""
    for item in char:
        otp += item


    if username_entry.get() == '':
        messagebox.showerror('Error', 'Your registered email id is required.')

    else:
        login_database_object = pymysql.connect(host='localhost', user='root', password='root',
                                    database='user_credentials')
        login_database = login_database_object.cursor()
        login_database.execute('select * from user_credentials where user_email_id=%s', username_entry.get())
        row = login_database.fetchone()
        if row != None:
            val = (otp, username_entry.get())
            login_database.execute('select * from login_credentials where user_email=%s', username_entry.get())
            row = login_database.fetchone()
            if row != None:
                login_database.execute('update login_credentials set one_time_password=%s where user_email=%s', val)
            else:
                login_database.execute('insert into login_credentials(one_time_password, user_email) values(%s,%s)', val)
            login_database_object.commit()
            login_database_object.close()

        else:
            messagebox.showerror('Error', 'Invalid email-id')
```

```
    try:
        receiver_id = username_entry.get()

        server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
        server.login("automated.otp@gmail.com", "retail_system")
        server.sendmail("automated.otp@gmail.com", receiver_id, otp)

        server.quit()
        messagebox.showinfo('Success', 'Your one time password was sent to your registered email id')
    except Exception as e:
        messagebox.showerror('Error', f'Error due to {e}')



def login():
    if otp_entry.get() == '':
        messagebox.showerror('Error', "The field can't be left empty.")

    else:
        otp_verification_object = pymysql.connect(host='localhost', user='root', password='root',
                                  database='user_credentials')
        otp_verification = otp_verification_object.cursor()
        otp_verification.execute('select * from login_credentials where one_time_password=%s', otp_entry.get())
        row = otp_verification.fetchone()
        if row != None:
            logged_in = 'YES'
            val = (logged_in, username_entry.get())
            messagebox.showinfo('Logged In', 'You have successfully logged in')
            otp_verification.execute('update login_credentials set login_status=%s where user_email=%s', val)
            otp_verification_object.commit()
            otp_verification_object.close()
            root_window.destroy()
            import main_system


        else:
            messagebox.showerror('Error', 'Incorrect OTP, please re-verify the password')




def redirect_to_signup():
    root_window.destroy()
    import registration_page


                ############################    GUI    ############################
```

```
root_window = ttkthemes.ThemedTk()
root_window.get_themes()
root_window.set_theme('breeze')

root_window.geometry('1220x645+20+0')
root_window.title('LOGIN')
root_window.resizable(0, 0)

bg_img = PhotoImage(file='login_bg.png')
bg_img_label = Label(root_window, image=bg_img)
bg_img_label.place(x=0, y=0)

login_frame = Frame(root_window, width=580, height=570, bg='#363d4f', bd=10)
login_frame.place(x=80, y=50)

heading_label = Label(login_frame, text='Orderzy', font=('Freestyle Script', 62, 'bold'), bg='#363d4f', fg='#f9d4cb')
heading_label.place(x=100, y=10)

username_label = Label(login_frame, text='Email ID', font=('Bodoni MT Condensed', 18, 'italic'), bg='#363d4f', fg='#f9d4cb')
username_label.place(x=0, y=150)
username_entry = ttk.Entry(login_frame, font=("Times new roman", 12), width=40)
username_entry.place(x=0, y=195)

otp_generation_button = ttk.Button(login_frame, text='Generate OTP', cursor='hand2', command=generate_otp)
otp_generation_button.place(x=390, y=197)

otp_label = Label(login_frame, text='One-Time-Password', font=('Bodoni MT Condensed', 18, 'italic'), bg='#363d4f', fg='#f9d4cb')
otp_label.place(x=0, y=260)
otp_entry = ttk.Entry(login_frame, font=("Times new roman", 12), width=40)
otp_entry.place(x=0, y=303)

login_button = ttk.Button(login_frame, text='Log In', cursor='hand2', command=login)
login_button.place(x=390, y=306)

or_label = Label(login_frame, text='----------  OR  ----------', font=('arial', 18, 'italic'), bg='#363d4f', fg='#f9d4cb')
or_label.place(x=0, y=400)

create_new_frame = Frame(root_window, width=500, height=80, bg='#363d4f', bd=10)
create_new_frame.place(x=60, y=500)

create_new_label = Label(create_new_frame, text="Don't have an account?", font=("arial", 14, 'italic'), bg='#363d4f', fg='#f9d4cb')
create_new_label.place(x=15, y=0)

signup_pg_import_button = Button(create_new_frame, text='Sign Up', font=('arial', 12, 'italic'), bd=0, cursor='hand2', bg='#363d4f',
fg='#f9d4cb'
                                 , activebackground='#363d4f', activeforeground='#f9d4cb', command=redirect_to_signup)
signup_pg_import_button.place(x=13, y=25)

root_window.mainloop()
```

C. Python Code for Account Settings Page

```
import ttkthemes
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
import pymysql

                        ####################    FUNCTIONS    ####################
def verify_email():
    def update_email():
        if enter_new_email_entry.get() == '':
            messagebox.showerror('Error', "Field can't remain empty")
        elif enter_new_email_entry.get() == current_email_entry.get():
            messagebox.showerror('Error', 'You have entered you current email address')
        else:
            val = (enter_new_email_entry.get(), current_email_entry.get())
            login_database_object = pymysql.connect(host='localhost', user='root', password='root',
                                                    database='user_credentials')
            login_database = login_database_object.cursor()
            login_database.execute('update user_credentials set user_email_id=%s where user_email_id=%s', val)
            login_database.execute('update login_credentials set user_email=%s where user_email=%s', val)
            login_database_object.commit()
            login_database_object.close()

            messagebox.showinfo('Success', 'Your email address was successfully changed.')

    if current_email_entry.get() == '':
        messagebox.showerror('Error', 'Please enter your current email address for verification.')

    else:
        login_database_object = pymysql.connect(host='localhost', user='root', password='root',
                                                database='user_credentials')
        login_database = login_database_object.cursor()
        login_database.execute('select * from user_credentials where user_email_id=%s', current_email_entry.get())
        row = login_database.fetchone()
        if row != None:
            enter_new_email_label = Label(email_updation_frame, text='Enter New Email Id', font=('arial', 14),
                                          bg='#363d4f', fg='#f9d4cb')
            enter_new_email_label.place(x=10, y=170)
            enter_new_email_entry = Entry(email_updation_frame, width=30)
            enter_new_email_entry.place(x=190, y=173)

            update_email_button = ttk.Button(email_updation_frame, text='Update Email ID', cursor='hand2',
                                             command=update_email)
            update_email_button.place(x=288, y=207)

        else:
            messagebox.showerror('Error', 'The email-id entered is not registered')
```

```
def edit_email():
    address_updation_frame.grid_remove()
    contact_updation_frame.grid_remove()
    email_updation_frame.grid()



def verify_contact():
    def update_contact():
        if enter_new_contact_entry.get() == '':
            messagebox.showerror('Error', "Field can't remain empty")
        elif enter_new_contact_entry.get() == current_contact_entry.get():
            messagebox.showerror('Error', 'You have entered you current contact number')
        else:
            val = (enter_new_contact_entry.get(), current_contact_entry.get())
            login_database_object = pymysql.connect(host='localhost', user='root', password='root',
                                                    database='user_credentials')
            login_database = login_database_object.cursor()
            login_database.execute('update user_credentials set user_contact_num=%s where user_contact_num=%s', val)
            login_database_object.commit()
            login_database_object.close()

            messagebox.showinfo('Success', 'Your contact number was successfully changed.')

    if current_contact_entry.get() == '':
        messagebox.showerror('Error', "Field can't remain empty")
    else:
        login_database_object = pymysql.connect(host='localhost', user='root', password='root',
                                                database='user_credentials')
        login_database = login_database_object.cursor()
        login_database.execute('select * from user_credentials where user_contact_num=%s', current_contact_entry.get())
        row = login_database.fetchone()
        if row != None:
            enter_new_contact_label = Label(contact_updation_frame, text='Enter New Contact Number', font=('arial', 14),
                                            bg='#363d4f', fg='#f9d4cb')
            enter_new_contact_label.place(x=10, y=170)
            enter_new_contact_entry = Entry(contact_updation_frame, width=30)
            enter_new_contact_entry.place(x=260, y=173)

            update_contact_button = ttk.Button(contact_updation_frame, text='Update Contact Number', cursor='hand2',
                                               command=update_contact)
            update_contact_button.place(x=313, y=207)
        else:
            messagebox.showerror('Error', "This contact number is not registered")

def edit_contact():
    address_updation_frame.grid_remove()
    email_updation_frame.grid_remove()
    contact_updation_frame.grid()
```

```
def verify_address():
    def update_pincode():
        if change_pincode_entry.get() == '':
            messagebox.showerror('Error', "Field can't remain empty")
        else:
            val = (change_pincode_entry.get(), email_address_entry.get())
            login_database_object = pymysql.connect(host='localhost', user='root', password='root',
                                    database='user_credentials')
            login_database = login_database_object.cursor()
            login_database.execute('update user_credentials set user_area_pincode=%s where user_email_id=%s', val)
            login_database_object.commit()
            login_database_object.close()

            messagebox.showinfo('Success', 'Your city was successfully updated.')
    def update_city():
        if change_city_entry.get() == '':
            messagebox.showerror('Error', "Field can't remain empty")
        else:
            val = (change_city_entry.get(), email_address_entry.get())
            login_database_object = pymysql.connect(host='localhost', user='root', password='root',
                                    database='user_credentials')
            login_database = login_database_object.cursor()
            login_database.execute('update user_credentials set user_city=%s where user_email_id=%s', val)
            login_database_object.commit()
            login_database_object.close()

            messagebox.showinfo('Success', 'Your city was successfully updated.')
    def update_address():
        if change_address_entry.get() == '':
            messagebox.showerror('Error', "Field can't remain empty")
        else:
            val = (change_address_entry.get(), email_address_entry.get())
            login_database_object = pymysql.connect(host='localhost', user='root', password='root',
                                    database='user_credentials')
            login_database = login_database_object.cursor()
            login_database.execute('update user_credentials set user_address=%s where user_email_id=%s', val)
            login_database_object.commit()
            login_database_object.close()

            messagebox.showinfo('Success', 'Your address was successfully updated.')
    if email_address_entry.get() == '':
        messagebox.showerror('Error', "Field can't remain empty")
    else:
        login_database_object = pymysql.connect(host='localhost', user='root', password='root',
                                database='user_credentials')
        login_database = login_database_object.cursor()
        login_database.execute('select * from user_credentials where user_email_id=%s', email_address_entry.get())
```

```
    row = login_database.fetchone()
    if row != None:
        your_address_label = Label(address_updation_frame, text="Your Address", font=('arial', 14),
                                   bg='#363d4f', fg='#f9d4cb')
        your_address_label.place(x=10, y=185)
        login_database.execute('SELECT user_address FROM user_credentials where user_email_id=%s limit
0,5',email_address_entry.get())
        i = 0
        for entry in login_database:
            for j in range(len(entry)):
                change_address_entry = ttk.Entry(address_updation_frame, width=40)
                change_address_entry.place(x=195, y=183)
                change_address_entry.insert(END, entry[j])
            i = i + 1
        update_address_button = ttk.Button(address_updation_frame, text='Update Address', cursor='hand2',
                                           command=update_address)
        update_address_button.place(x=373, y=225)

        your_city_label = Label(address_updation_frame, text="Your City", font=('arial', 14),
                                bg='#363d4f', fg='#f9d4cb')
        your_city_label.place(x=10, y=275)
        login_database.execute('SELECT user_city FROM user_credentials where user_email_id=%s limit 0,5',
                               email_address_entry.get())
        i = 0
        for entry in login_database:
            for j in range(len(entry)):
                change_city_entry = ttk.Entry(address_updation_frame, width=40)
                change_city_entry.place(x=195, y=273)
                change_city_entry.insert(END, entry[j])
            i = i + 1
        update_city_button = ttk.Button(address_updation_frame, text='Update City', cursor='hand2',
                                        command=update_city)
        update_city_button.place(x=395, y=315)

        your_pincode_label = Label(address_updation_frame, text="Your Area Pincode", font=('arial', 14),
                                   bg='#363d4f', fg='#f9d4cb')
        your_pincode_label.place(x=10, y=365)
        login_database.execute('SELECT user_area_pincode FROM user_credentials where user_email_id=%s limit 0,5',
                               email_address_entry.get())
        i = 0
        for entry in login_database:
            for j in range(len(entry)):
                change_pincode_entry = ttk.Entry(address_updation_frame, width=40)
                change_pincode_entry.place(x=195, y=363)
                change_pincode_entry.insert(END, entry[j])
            i = i + 1
        update_pin_button = ttk.Button(address_updation_frame, text='Update Area Pincode', cursor='hand2',
                                       command=update_pincode)
        update_pin_button.place(x=342, y=405)
```


ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue III Mar 2022- Available at www.ijraset.com

```
        else:
            messagebox.showerror('Error', "Email address not found")

def edit_address():
    contact_updation_frame.grid_remove()
    email_updation_frame.grid_remove()
    address_updation_frame.grid()



def back():
    try:
        root_window.destroy()
        import main_system
    except Exception as e:
        messagebox.showerror('Error', f'Error due to {e}')


def log_out():
    logged_out = 'NO'
    logged_in = 'YES'
    val = (logged_out, logged_in)
    logout_object = pymysql.connect(host='localhost', user='root', password='root',
                                    database='user_credentials')
    logout = logout_object.cursor()
    logout.execute('update login_credentials set login_status=%s where login_status=%s', val)
    logout_object.commit()
    logout_object.close()
    root_window.destroy()
    import login_page


                ####################    GUI    ####################

root_window = ttkthemes.ThemedTk()
root_window.get_themes()
root_window.set_theme('breeze')

root_window.geometry('1220x645+20+0')
root_window.title('ACCOUNT SETTINGS')
root_window.config(bg='#242b3d')
root_window.resizable(0, 0)

heading_label = Label(root_window, text="MY PROFILE", font=('Berlin Sans FB', 44), bg='#242b3d', fg='#f9d4cb')
heading_label.place(x=20, y=5)

settings_frame = Frame(root_window, width=1200, height=520, bg='#363d4f')
settings_frame.place(x=10, y=100)
```

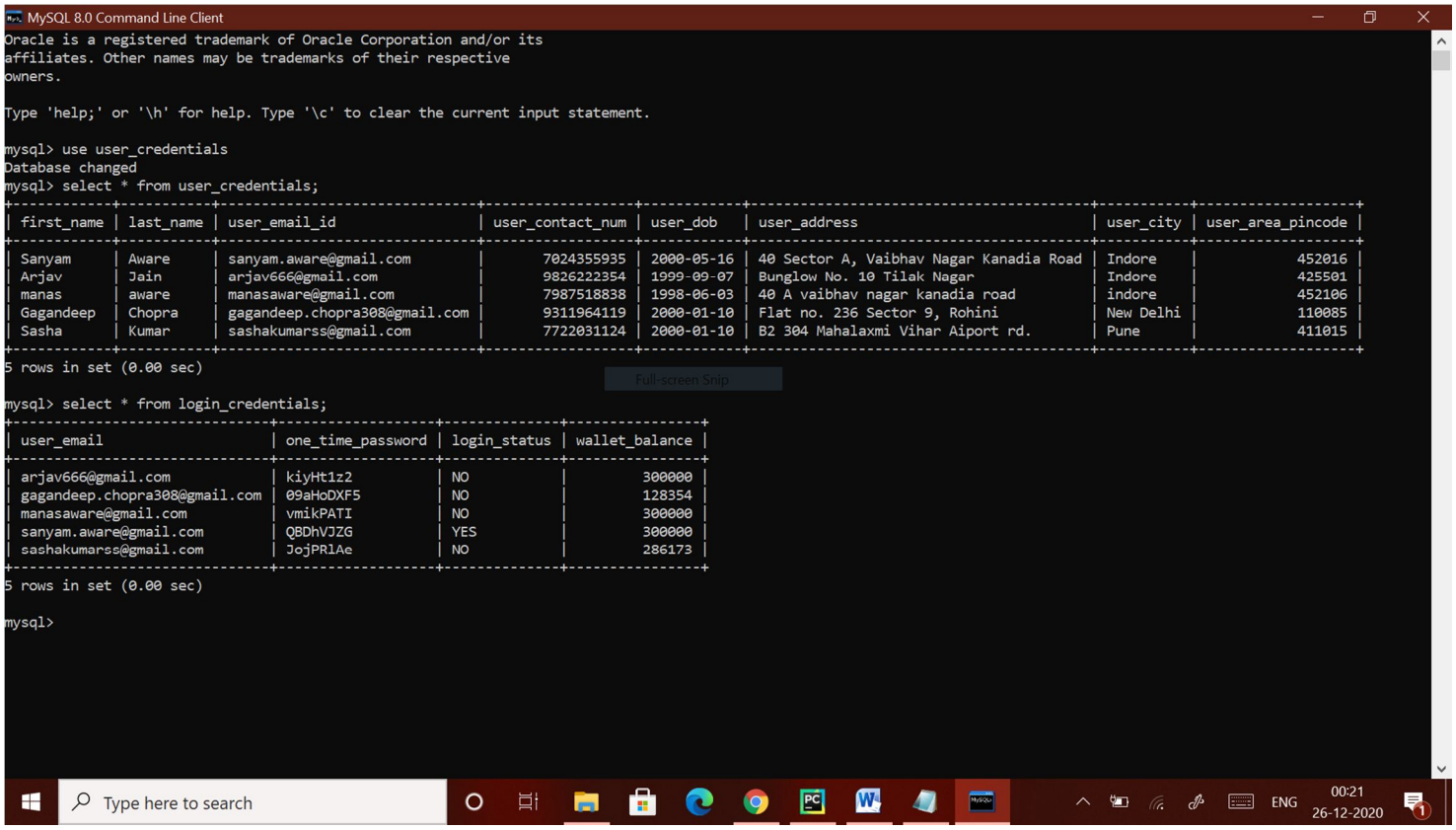

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue III Mar 2022- Available at www.ijraset.com

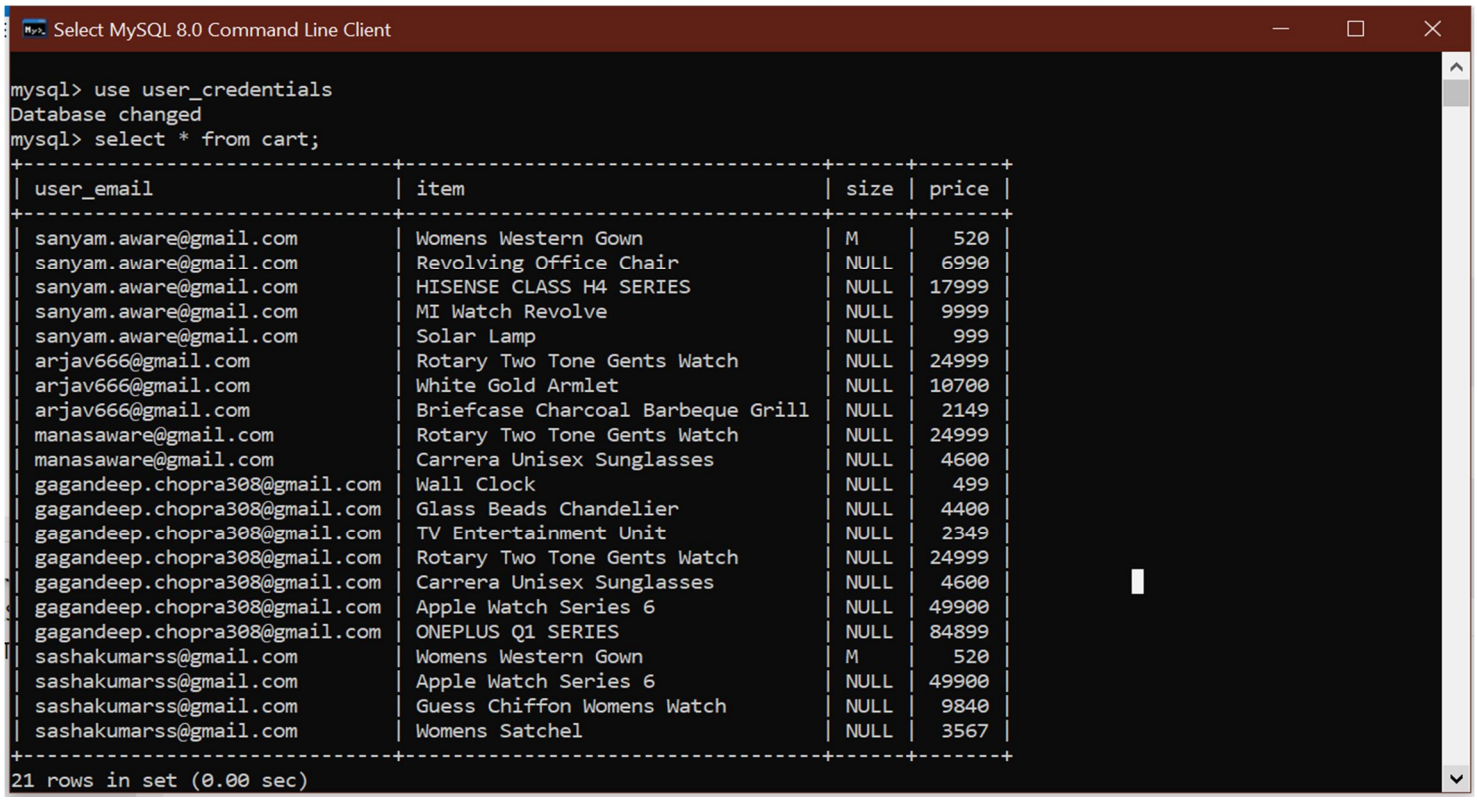
```
edit_profile_label = Label(settings_frame, text="EDIT MY PROFILE", font=('Copperplate Gothic Bold', 24, 'underline'),
cursor='hand2', bg='#363d4f', fg='#f9d4cb', bd=0, activebackground='#363d4f',
                    activeforeground='#f9d4cb')
edit_profile_label.place(x=10, y=10)


#FRAME WHERE EMAIL CREDENTIALS WILL BE UPDATED
email_updation_frame = Frame(settings_frame, width=580, height=520, bg='#363d4f')
email_updation_frame.grid(column=0, row=0, padx=(610,0), pady=0)
email_updation_frame.grid_remove()

#LABEL FOR CURRENT EMAIL ID
current_email_label = Label(email_updation_frame, text='Current Email ID', font=('arial', 14), bg='#363d4f', fg='#f9d4cb')
current_email_label.place(x=10, y=60)

#ENTRY FIELD FOR ENTERING CURRENT EMAIL ID
current_email_entry = Entry(email_updation_frame, width=30)
current_email_entry.place(x=190, y=63)

#BUTTON TO VERIFY CURRENT EMAIL ID
verify_email_button = ttk.Button(email_updation_frame, text='Verify Email ID', cursor='hand2', command=verify_email)
verify_email_button.place(x=295, y=100)

#BUTTON FOR ENABLING THE OPTION TO UPDATE EMAIL ID
edit_email_button = Button(settings_frame, text="UPDATE EMAIL ADDRESS", font=('Segoe Print', 12), cursor='hand2',
bg='#363d4f', fg='#f9d4cb', bd=0, activebackground='#363d4f',
                    activeforeground='#f9d4cb', command=edit_email)
edit_email_button.place(x=10, y=55)


#FRAME WHERE CONTACT NUMBER WILL BE UPDATED
contact_updation_frame = Frame(settings_frame, width=580, height=520, bg='#363d4f')
contact_updation_frame.grid(column=0, row=0, padx=(610,0), pady=0)
contact_updation_frame.grid_remove()

#LABEL FOR CURRENT CONTACT NUMBER
current_contact_label = Label(contact_updation_frame, text='Current Contact Number', font=('arial', 14), bg='#363d4f',
fg='#f9d4cb')
current_contact_label.place(x=10, y=60)

#ENTRY FIELD FOR ENTERING CURRENT CONTACT NUMBER
current_contact_entry = Entry(contact_updation_frame, width=30)
current_contact_entry.place(x=260, y=63)

#BUTTON TO VERIFY CURRENT CONTACT NUMBER
verify_contact_button = ttk.Button(contact_updation_frame, text='Verify Contact Number', cursor='hand2',
command=verify_contact)
verify_contact_button.place(x=320, y=100)
```



```
#BUTTON FOR ENABLING THE OPTION TO UPDATE CONTACT NUMBER
edit_contact_button = Button(settings_frame, text="UPDATE CONTACT NUMBER", font=('Segoe Print', 12), cursor='hand2',
bg='#363d4f', fg='#f9d4cb', bd=0, activebackground='#363d4f',
                activeforeground='#f9d4cb', command=edit_contact)
edit_contact_button.place(x=10, y=100)


#FRAME WHERE ADDRESS WILL BE UPDATED
address_updation_frame = Frame(settings_frame, width=580, height=520, bg='#363d4f')
address_updation_frame.grid(column=0, row=0, padx=(610,0), pady=0)
address_updation_frame.grid_remove()

#LABEL FOR CURRENT ADDRESS
email_address_label = Label(address_updation_frame, text='Email Address', font=('arial', 14), bg='#363d4f', fg='#f9d4cb')
email_address_label.place(x=10, y=60)

#ENTRY FIELD FOR ENTERING CURRENT ADDRESS
email_address_entry = Entry(address_updation_frame, width=30)
email_address_entry.place(x=274, y=63)

#BUTTON TO FIND ADDRESS USING EMAIL ID
find_address_button = ttk.Button(address_updation_frame, text='Search Address', cursor='hand2', command=verify_address)
find_address_button.place(x=373, y=100)

#BUTTON FOR ENABLING THE OPTION TO UPDATE ADDRESS
edit_address_button = Button(settings_frame, text="UPDATE YOUR ADDRESS", font=('Segoe Print', 12), cursor='hand2',
bg='#363d4f', fg='#f9d4cb', bd=0, activebackground='#363d4f',
                activeforeground='#f9d4cb', command=edit_address)
edit_address_button.place(x=10, y=145)


#BUTTON FOR RETURNING BACK TO THE MAIN PAGE
back_button = Button(settings_frame, text='BACK', font=('Agency FB', 16, 'bold'), cursor='hand2', bg='#363d4f', fg='#f9d4cb',
bd=0, activebackground='#363d4f',
                activeforeground='#f9d4cb', command=back)
back_button.place(x=10, y=423)


#BUTTON FOR LOGGING OUT
logout_button = Button(settings_frame, text='Log Out', font=('Agency FB', 18, 'bold'), cursor='hand2', bg='#363d4f', fg='#f9d4cb',
bd=0, activebackground='#363d4f',
                activeforeground='#f9d4cb', command=log_out)
logout_button.place(x=10, y=460)


root_window.mainloop()
```

IV. MySQL TABLES





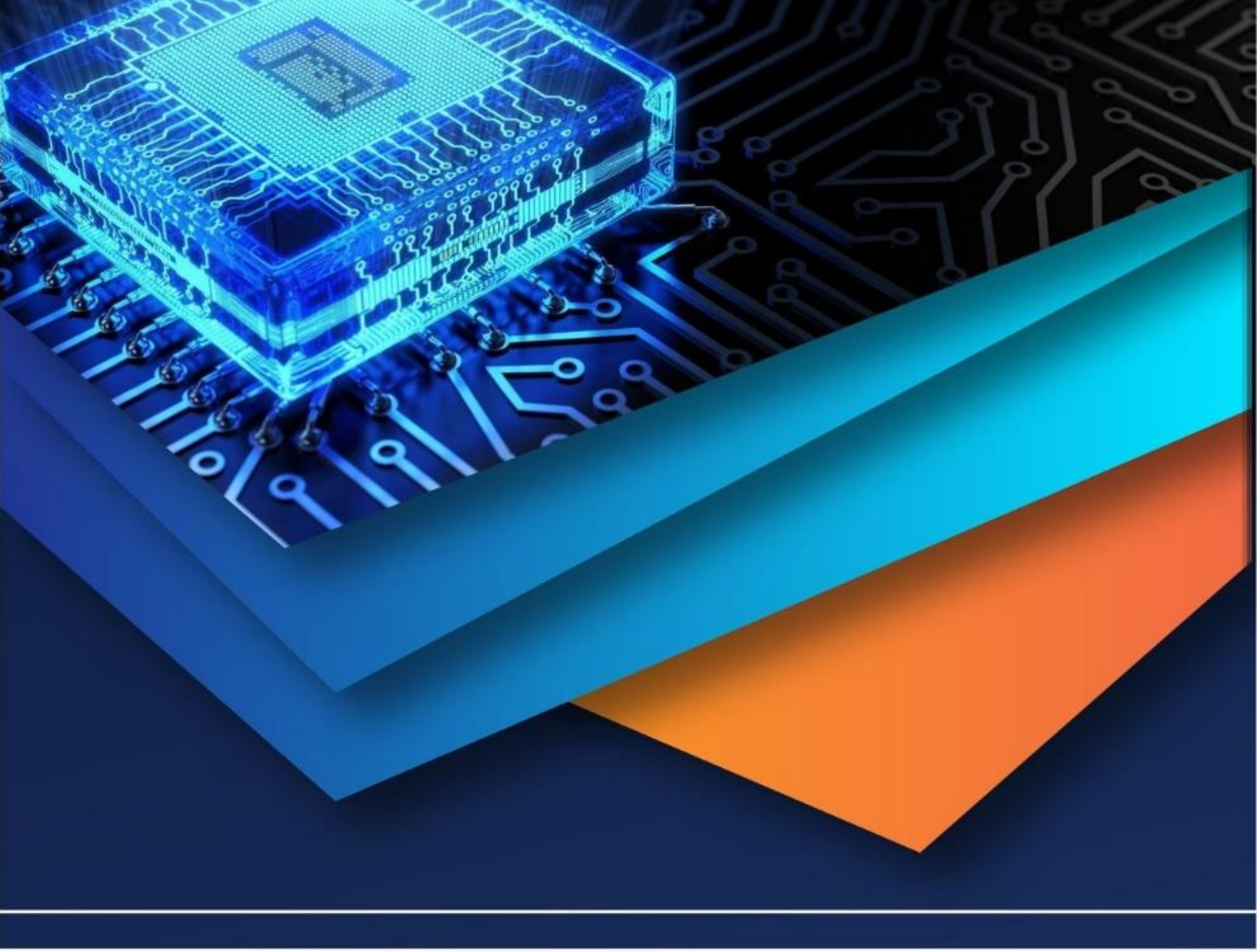
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue III Mar 2022- Available at www.ijraset.com

V. CONCLUSION & FUTURE ENHANCEMENTS

In this Python project, we have successfully implemented a prototype of an e-commerce application with the current accuracy being 80%. All the features as seen have been tried multiple times to bring out the best possible solution, the future enhancements would include a mobile application, a website and other algorithms to increase the productivity and usability of the application.

REFERENCES

[1] https://patents.google.com/patent/US5510979A/en
[2] https://www.sciencedirect.com/science/article/abs/pii/S096969891830643X
[3] https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=retail+technology+for+stores&btnG=
[4] https://www.tutorialspoint.com/
[5] https://stackoverflow.com/
[6] https://www.w3schools.com/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY