Python Project Using TKinter

On

Courier Management System

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Gautam Sunuwar

Registration number: 12111066

Roll No:K21DPB40

Section:K21DP

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

About Courier Management System

A **Courier Management System In Python** has a create account form and login page, under the creating account module includes such as username, password, registration number as (reg no) , gender, mobile number and email id, and after the user create their account, the user can login now to the system and monitor their items.

we are going to show about how to create a management system to track packages or couriers that are delivered to a specific destination. How does the consignment is tracked and what is the current status of the package? All these things we are going to be included in our project that is Courier Management System Project In Python .

The Package delivery Management System Project In Python is developed using Python Programming, This Project is created using Graphical User Interface (GUI) Tkinter and connected to the database using SQLlite3. This project was really helpful and challenging for us which assisted us to learn Python Programming Language smoothing and patiently.

Project Overview: Courier Management System Python Project

Project Name:	**Courier Management System Project in Python**
Abstract:	**A GUI-based program in python that basically includes the use of the Tkinter and Sqlite3 database for execution.**
Language/Technologies Used:	**Python,Tkinter**
IDE	**Visual Studio Code**
Database	**Sqlite3**
Python version (Recommended):	**Above 3.7 (3.9 used)**
Type/Category:	**2nd year 3rd semester project using python.**

Features and benefits of Courier/Package Management System

The basic task to be performed on this Project are:

1. Create an account if you are a new user and log in if you are already registered.
2. Add all the details of the user who wants to track the courier or package delivered to the destination.
3. Check the current status of the package.

Code flow: Module wise description

1. Importing the libraries

```

from tkinter import *
from tkinter import messagebox as ms
from tkinter import ttk
import sqlite3
import random
from tkinter import Button

```

Explanation:

The import function includes these modules in the project

These modules are used for the following purposes:

1. Tkinter – To create the GUI.
2. SQLite3 – To connect the program to the database and store information in it.
3. Tkinter.messagebox – To show a display box, displaying some information or an error or warning
4. Tkinter.ttk – To create the tree where all the information will be displayed.
5. random– It is an in-built module of Python which is used to generate random numbers.

2.Creating the database and table for the courier management system

```
with sqlite3.connect('GautamPranay.db') as db:
    c = db.cursor()
try:
    c.execute('CREATE TABLE IF NOT EXISTS user (username TEXT NOT NULL ,password TEX
except:
    pass
db.commit()
db.close()
```

Explanation:

Here we have created a database "**GautamPranay5"** and performed the database connection with sqlite3. This is stored in a variable db. After that, the cursor object c is created for executing the queries. In the try block, we are using the execute method in which we are creating a table user if it doesn't exist else it will do nothing i.e **pass. The commit()** is used to save the changes.

3.Declaring the variables required for the project

```
class main:
    def __init__(self, master):

        self.master = master

        self.username = StringVar()
        self.password = StringVar()
        self.n_username = StringVar()
        self.n_password = StringVar()
        self.n_reg = StringVar()
        self.n_mobile = StringVar()
        self.mobile11 = StringVar()
        self.widgets()
```

Explanation:

In this code block, we have declared the main class, where we have defined a init method in which the variables of string type have been declared. The self parameter is a reference to the current instance of the class and is used to access variables that belong to the class.

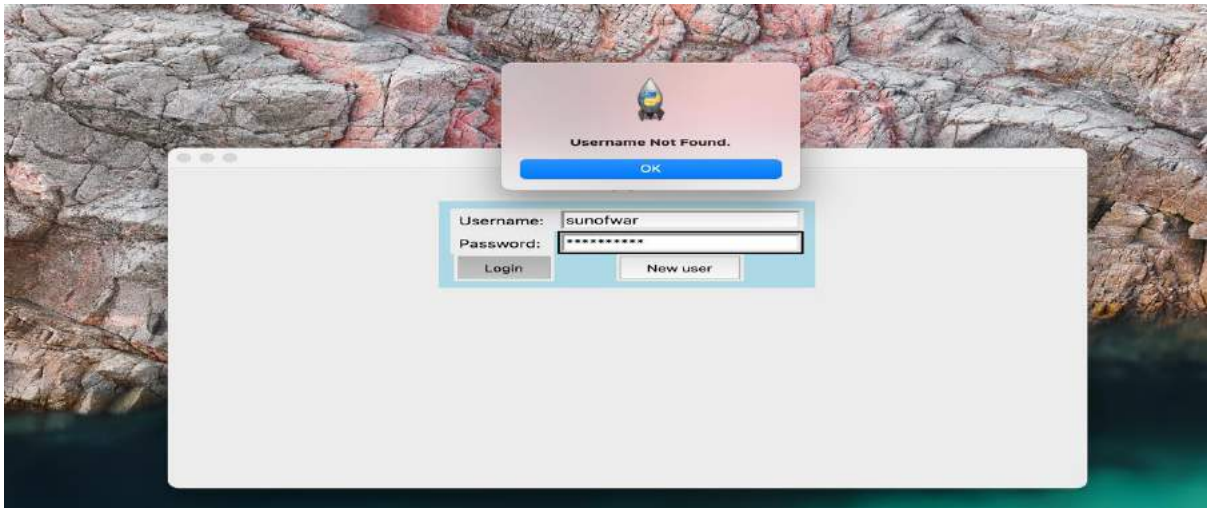
4.Code for creating Login function

```python

    def login(self):

        with sqlite3.connect('GautamPranay.db') as db:
            c = db.cursor()

        # Find user If there is any take proper action
        find_user = ('SELECT * FROM user WHERE username = ? and password = ?')
        c.execute(find_user, [(self.username.get()), (self.password.get())])
        result = c.fetchall()

        if result:
            self.track()
        else:
            ms.showerror('Oops!', 'Username Not Found.')

```

Explanation:

Here, in the function def login(self): We are working on our database created in sqlite3 i.e record123.db . The find_user is the variable that stored the value on the execution of the query. The query is used to check whether the user with the username and password we provide is already present in the database. If it is present it will fetch all the data with **fetchall()** method and store the output in the variable result. If the result is present then it will track else it will show an error massage "'Oops!', 'Username Not Found."

Output 1:

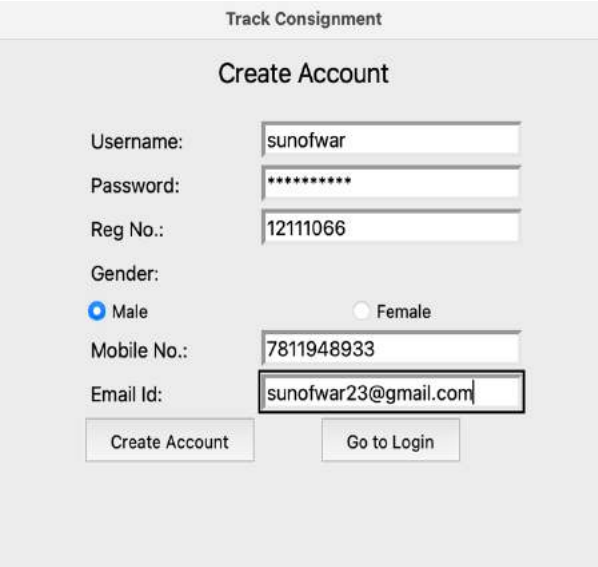


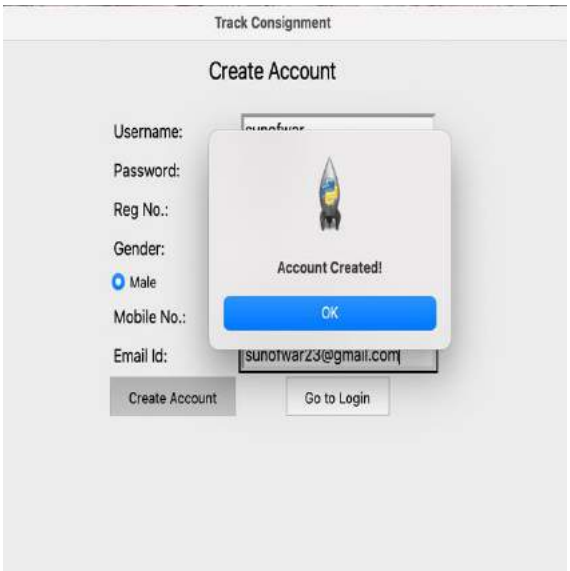
5. Develop a code for creating a new user account

```python

    def new_user(self):

        with sqlite3.connect('GautamPranay.db') as db:
            c = db.cursor()
        if self.n_username.get() != ' ' and self.n_password.get() != ' ' and self.n_mobile.get() != ' ':
            find_user = ('SELECT * FROM user WHERE username = ?')
            c.execute(find_user, [(self.n_username.get())])

            if c.fetchall():
                ms.showerror('Error!', 'Username Taken Try a Diffrent One.')
            else:
                insert = 'INSERT INTO user(username,password,mobile) VALUES(?,?,?)'
                c.execute(insert, [(self.n_username.get()),
                           (self.n_password.get()), (self.n_mobile.get())])
                db.commit()

                ms.showinfo('Success!', 'Account Created!')
                self.log()
        else:
            ms.showerror('Error!', 'Please Enter the details.')

```

Explanation:
def new_user(self): In this code block, we are creating a new user if not created in our database record123.db. For that, we will create a new account that stores the information of the user such as name, password, registration number, gender, mobile no, and email where the name, password, and mobile number are used to track the package. In the if the block it will check whether the user with the username is present it will be stored in the variable find_user. If the user is already present it will print the message 'Error!', 'Username is already Taken. 'Else create a new account by clicking on "New User". Store the values. Once it's done it will show a message" 'Success!', 'Account Created!'. else it will show a massage "Error!', 'Please Enter the details.'

Output 2:





6. Code to track the courier/package

```python

    def consignment(self):

        try:
            with sqlite3.connect('GautamPranay.db') as db:
                c = db.cursor()

            # Find user If there is any take proper action
            find_user = ('SELECT * FROM user WHERE mobile= ?')
            c.execute(find_user, [(self.mobile11.get())])
            result = c.fetchall()

            if result:
                self.track()
                self.crff.pack_forget()
                self.head['text'] = self.username.get() + \
                    '\n Your Product Details'
                self.consi.pack()
            else:
                ms.showerror('Oops!', 'Mobile Number Not Found.')
        except:
            ms.showerror('Oops!', 'Mobile Number Not Found.')

```

Explanation:

def consignment(self): In this code block, we are making use of the database record123.db.Here we will check for the mobile number and consignment number while tracking, if the mobile number is present in the database all the details related to the mobile number are fetched in the result. If the result is true it will show the product details else will show an error message as "'Oops!', '**Mobile Number Not Found**.'

Output 3:

7. Code to view the tracking information

```python

    def track1(self):
        self.consi.pack_forget()
        self.head['text'] = self.username.get() + '\n Track your Product'
        self.crff.pack()

    def log(self):
        self.username.set('')
        self.password.set('')
        self.crf.pack_forget()
        self.head['text'] = 'Login'
        self.logf.pack()

    def cr(self):
        self.n_username.set('')
        self.n_password.set('')
        self.logf.pack_forget()
        self.head['text'] = 'Create Account'
        self.crf.pack()

    def track(self):
        self.logf.pack_forget()
        self.head['text'] = self.username.get() + '\n Track your Product'

        self.crff.pack()
```

Explanation:

def track(self): It checks for the tracking of the package. The “text” variable takes the username and Tracks the product.

def log(self): The log function is used for the login page takes the username and password and the “text” variable takes the “Login” text for logging to the page.

def create(self): It is used for creating the account.

8. Module containing frames, text, labels, and buttons of the courier management system in python.

```python

    def widgets(self):
        self.head = Label(self.master, text='LOGIN', font=('', 20), pady=10)
        self.head.pack()

        self.logf = Frame(self.master, padx=10, pady=10)

        self.logf.configure(background='lightblue')
        #PhotoImage(self.logf,file = 'lpu_logo.png')
        Label(self.logf, text='Username: ', font=(
            '', 15), pady=5, padx=5).grid(sticky=W)
        Entry(self.logf, textvariable=self.username,
              bd=3, font=('', 15)).grid(row=0, column=1)
        Label(self.logf, text='Password: ', font=(
            '', 15), pady=5, padx=5).grid(sticky=W)
        Entry(self.logf, textvariable=self.password, bd=3,
              font=('', 15), show='*').grid(row=1, column=1)
        Button(self.logf, text=' Login ', background='lightgrey', bd=2, font=(
            '', 13), padx=6, pady=6, command=self.login).grid(row=8, column=0)
        Button(self.logf, text=' New user ', background='lightgrey', bd=2, font=(
            '', 13), padx=6, pady=6, command=self.cr).grid(row=8, column=1)

```

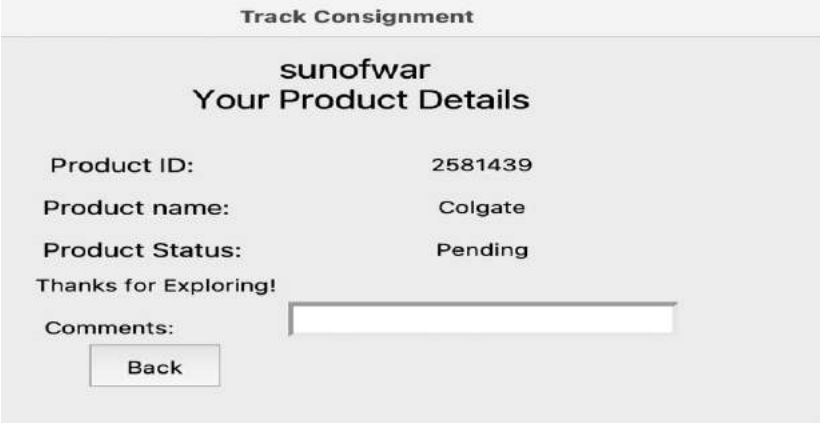
```python

        self.logf.pack()

        self.crf = Frame(self.master, padx=10, pady=10)
        Label(self.crf, text='Username: ', font=(
            '', 15), pady=5, padx=5).grid(sticky=W)
        Entry(self.crf, textvariable=self.n_username,
              bd=3, font=('', 15)).grid(row=0, column=1)

        Label(self.crf, text='Password: ', font=(
            '', 15), pady=5, padx=5).grid(sticky=W)
        Entry(self.crf, textvariable=self.n_password, bd=3,
              font=('', 15), show='*').grid(row=1, column=1)

        Label(self.crf, text='Reg No.: ', font=(
            '', 15), pady=5, padx=5).grid(sticky=W)
        Entry(self.crf, textvariable=self.n_reg, bd=3,
              font=('', 15)).grid(row=2, column=1)
        Label(self.crf, text='Gender: ', font=(
            '', 15), pady=5, padx=5).grid(sticky=W)
        var = IntVar()
        R1 = Radiobutton(self.crf, text="Male", variable=var,
                         value=1).grid(sticky=W)

        R2 = Radiobutton(self.crf, text="Female", variable=var,
                         value=2).grid(row=4, column=1)
        Label(self.crf, text='Mobile No.: ', font=(
            '', 15), pady=5, padx=5).grid(sticky=W)
        Entry(self.crf, textvariable=self.n_mobile,
              bd=3, font=('', 15)).grid(row=5, column=1)

        Label(self.crf, text='Email Id: ', font=(
            '', 15), pady=5, padx=5).grid(sticky=W)
        Entry(self.crf, bd=3, font=('', 15)).grid(row=6, column=1)

        Button(self.crf, text='Create Account', background='lightgrey', bd=2, font=(
            '', 13), padx=6, pady=6, command=self.new_user).grid(row=11, column=0)
        Button(self.crf, text='Go to Login', background='lightgrey', bd=2, font=(
            '', 13), padx=6, pady=6, command=self.log).grid(row=11, column=1)
```

```python

        self.crff = Frame(self.master, padx=10, pady=10)

        Label(self.crff, text='Consignment No: ', font=(
            '', 15), pady=5, padx=5).grid(sticky=W)
        Entry(self.crff, bd=3, font=('', 15)).grid(row=0, column=1)
        Label(self.crff, text='Mobile no:', font=(
            '', 15), pady=5, padx=5).grid(sticky=W)
        Entry(self.crff, bd=3, textvariable=self.mobile11,
              font=('', 15)).grid(row=1, column=1)
        Button(self.crff, text='Track', background='lightgrey', bd=2, font=(
            '', 13), padx=6, pady=6, command=self.consignment).grid(row=4, column=0)

        self.consi = Frame(self.master, padx=10, pady=10)

        Label(self.consi, text=' Product ID:', font=(
            '', 15), pady=5, padx=5).grid(sticky=W)
        Label(self.consi, text=random.randint(565154, 99994216),
              font=('', 13), pady=5, padx=5).grid(row=0, column=1)
        L = ['Bag', 'Colgate', 'shoe', 'Redme 2', 'Jeans',
             'Parrot', 'Mac', 'Ipad', 'Pen', 'Book', 'shirt']
        f = random.randint(0, 10)
        Label(self.consi, text='Product name: ', font=(
            '', 15), pady=5, padx=5).grid(sticky=W)
        Label(self.consi, text=L[f], font=('', 13),
              pady=5, padx=5).grid(row=1, column=1)
        Label(self.consi, text='Product Status: ', font=(
            '', 15), pady=5, padx=5).grid(sticky=W)
        Label(self.consi, text='Pending', font=('', 13),
              pady=5, padx=5).grid(row=2, column=1)
        Label(self.consi, font=('', 13),
              text='Thanks for Exploring!').grid(row=4, column=0)

        Label(self.consi, text='Comments:', font=('', 13)).grid(
            row=5, column=0, padx=5, sticky='sw')
        Entry(self.consi, bd=3, font=('', 15)).grid(row=5, column=1)

        Button(self.consi, text='Back', background='lightgrey', bd=2, font=(
            '', 13), padx=6, pady=6, command=self.track1).grid(row=6, column=0)

```

Explanation:

Here we have provided all the widgets that are required for the form creation of login, tracking, and delivery. We are using the randint function of the random module for randomly selecting the product numbers and we have defined the 10 products which the system selects randomly while executing the program.

Output 4:



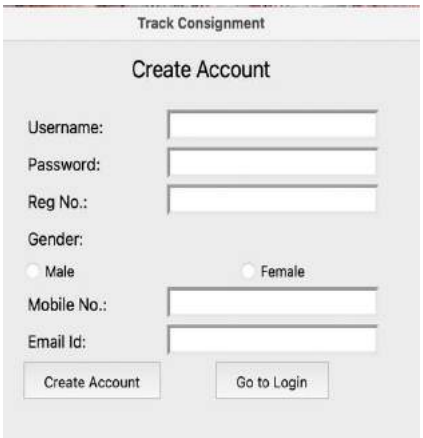
9. Creating a display window

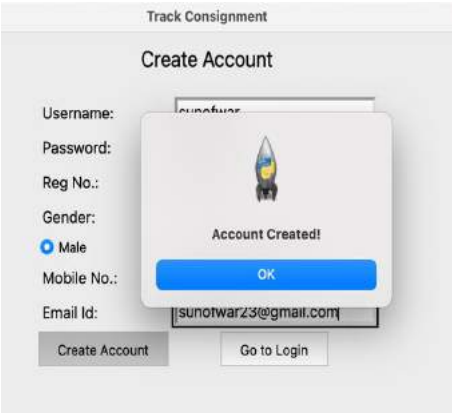
```
if __name__ == '__main__':

    root = Tk()
    root.title('Track Consignment')
    root.geometry('800x450+300+300')
    main(root)

    root.mainloop()
```

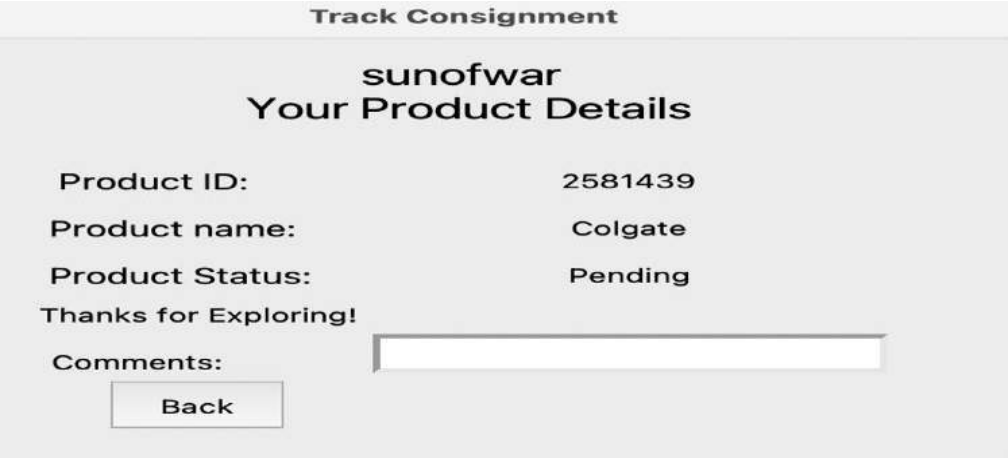
Explanation:

Here we have defined the main function, where The root window is created. The root window is the main application window in our programs. We have defined the title of the window as “Track consignment”. The window size is defined using the geometry function. The root.mainloop() is simply a method in the main window that executes what we wish to execute in an application.

Output Screen shot







Summary

In this article, we have learned to develop the **package or courier management system using python** programming language and using the GUI Tkinter module connecting to the SQLite3 database and performing the functions of the product tracking.