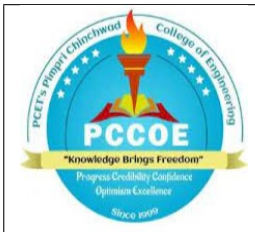

Pimpri Chinchwad Education Trust's
Pimpri Chinchwad College of Engineering

Python Curricular Activity

PRN.: 123M1H041 **Batch : 2**
Name of Student: Darshan Pathak **Submission Date: Mar.26.2024**
Assigment 4: Curriculum Activity Name: GUI Based applications
BranchHub Inventory Management App

Q. Design and Develop Graphical User Interface using MySql and TkInter Module of

Python Program:

```
import random
from time import time
from tkinter import *
import customtkinter as ct
from datetime import datetime
import mysql.connector
import pytz
from PIL import Image, ImageTk
from tkinter.ttk import Treeview
from tkinter import ttk
from tkinter import messagebox

# Main application window
root = ct.CTk()
root.geometry("856x645")
root.title("BranchHub - Inventory Management App")

# Set appearance mode and color theme
ct.set_appearance_mode("dark")
ct.set_default_color_theme('dark-blue')

# Global constants for database connection
DB_HOST = 'localhost'
DB_USER = 'root'
DB_PASSWORD = 'root'
DB_NAME = 'python_ca'

# Function to establish a database connection and create a cursor
def connect_to_database():
    return mysql.connector.connect(
        host=DB_HOST,
        user=DB_USER,
        password=DB_PASSWORD,
        database=DB_NAME
    )

```

```
def execute_query(query, params=None):
    try:
        with connect_to_database() as db:
            cursor = db.cursor()
            cursor.execute(query, params)
            return cursor.fetchall()
    except mysql.connector.Error as e:
        messagebox.showerror("Database Error", "An error occurred while accessing the database.")
        return None

def execute_non_query(query, params=None):
    try:
        with connect_to_database() as db:
            cursor = db.cursor()
            cursor.execute(query, params)
            db.commit()
            return True
    except mysql.connector.Error as e:
        messagebox.showerror("Database Error", "An error occurred while accessing the database.")
        return False

#Login form---
def login_db():
    try:
        password_en = int(en_password.get())
        username = (en_branch_name.get())

    except:
        messagebox.showerror("Error", "Please enter both username and password.")
        return

    myquery=("select * from branch where branch_name='%s' and branch_password='%s'"%(username,password_en))
    res=execute_query(myquery)

    if len(res)==0:
        messagebox.showerror("Error", "Please enter correct details.")
        return

    branch_id=res[0][0]
    branch_name=res[0][1]
    branch_pass=res[0][4]
    branch_mg = res[0][5]

    if username == branch_name and password_en == branch_pass:
        ct.set_appearance_mode("dark")
        ct.set_default_color_theme('dark-blue')

        # Create a new window for branch interface
        log_window=Toplevel(root)
        log_window.configure(bg="black")
        log_window.columnconfigure(1, minsize=200)
        tabview = ct.CTkTabview(log_window,segmented_button_fg_color='#272b33',segmented_button_selected_color='#4149d1',segmented_button_selected_hover_color='green',width=856,height=645)

        tabview.add("Buy Managment")
        tabview.add('Sell Managment')
        tabview.add('Add New Employee')
        tabview.add('Add New Client')

        tabview.pack(padx=10, pady=10)

        def tell_choice(choice):
```

```
            return choice

        c_id=ct.CTkEntry(tabview.tab("Buy Managment"),width=200)
        c_id2=ct.CTkEntry(tabview.tab("Sell Managment"),width=200)
        cl_name = ct.CTkEntry(tabview.tab("Add New Client"),width=200)

        im_name = ct.CTkEntry(tabview.tab("Add New Employee"),width=500)
        im_last = ct.CTkEntry(tabview.tab("Add New Employee"),width=500)
        im_sex = ct.CTkEntry(tabview.tab("Add New Employee"),width=500)
        im_call = ct.CTkEntry(tabview.tab("Add New Employee"),width=500)
        im_birth = ct.CTkEntry(tabview.tab("Add New Employee"),width=500)
        im_salary=ct.CTkEntry(tabview.tab("Add New Employee"),width=500)

        im_name.grid(row=0,column=1,padx=20)
        im_last.grid(row=1,column=1,padx=20)
        im_sex.grid(row=2,column=1,padx=20)
        im_call.grid(row=3,column=1,padx=20)
        im_birth.grid(row=4,column=1,padx=20)
        im_salary.grid(row=5,column=1,padx=20)

        im_namel=ct.CTkLabel(tabview.tab("Add New Employee"),text="Enter Employee
    Name",width=200)
        im_lastl=ct.CTkLabel(tabview.tab("Add New Employee"),text="Enter Employee
    Last Name",width=200)
        im_sexl=ct.CTkLabel(tabview.tab("Add New Employee"),text="Enter Sex(f or m)
    ",width=200)
        im_calll=ct.CTkLabel(tabview.tab("Add New Employee"),text="Enter Employee
    Call Number",width=200)
        im_birthl=ct.CTkLabel(tabview.tab("Add New Employee"),text="Enter Employee
    BirthDay",width=200)
        im_salaryl=ct.CTkLabel(tabview.tab("Add New Employee"),text="Enter Salary",
    width=200)

        im_namel.grid(row=0, column=0)
        im_lastl.grid(row=1, column=0)
        im_sexl.grid(row=2, column=0)
        im_calll.grid(row=3, column=0)
        im_birthl.grid(row=4, column=0)
        im_salaryl.grid(row=5, column=0)

        c_id.grid(row=0,column=1,padx=20)
        c_id2.grid(row=0,column=1,padx=20)
        cl_name.grid(row=0,column=1,padx=20)

        amount=ct.CTkEntry(tabview.tab("Buy Managment"),width=500)
        amount3=ct.CTkEntry(tabview.tab("Sell Managment"),width=500)
        cl_ln=ct.CTkEntry(tabview.tab("Add New Client"),width=500)
        cl_ln.grid(row=1,column=1,padx=20)
        amount3.grid(row=1,column=1,padx=20)
        amount.grid(row=1,column=1,padx=20)

        price=ct.CTkEntry(tabview.tab("Buy Managment"),width=500)
        price3=ct.CTkEntry(tabview.tab("Sell Managment"),width=500)
        cl_call=ct.CTkEntry(tabview.tab("Add New Client"),width=500)
        cl_call.grid(row=2,column=1,padx=20)
        price3.grid(row=2,column=1,padx=20)
        price.grid(row=2,column=1,padx=20)

        row_mat=ct.CTkOptionMenu(master=tabview.tab("Buy Managment"),
        values=["400-Cocoa Beans Grade A", "401-Cocoa Beans Grade B", "402-Vanilla
    Beans Grade A", "403-Vanilla Beans Grade B", "404-Strawberries Grade A", "405-
    Strawberries Grade B", "406-Blueberries Grade A", "407-Blueberries Grade B"],
        command=tell_choice)

        products=ct.CTkOptionMenu(master=tabview.tab("Sell Managment"),
```

```
        values = ["300-Chocolate Bar", "301-Vanilla Ice Cream", "302-Strawberry Yogurt", "303-Blueberry Muffin", "304-Caramel Popcorn", "305-Pineapple Juice", "306-Orange Marmalade"]
        )
        cl_addres=ct.CTkEntry(master=tabview.tab("Add New Client"),width=500)
        cl_addres.grid(row=3,column=1)

        products.grid(row=3,column=1)
        row_mat.grid(row=3,column=1)

        cl_id = ct.CTkLabel(tabview.tab("Buy Managment"),text="Enter Client ID")
        cl_id.grid(row=0,column=0)
        cl2_id = ct.CTkLabel(tabview.tab("Sell Managment"),text="Enter Client ID")
        cl2_id.grid(row=0,column=0)
        cl_namela=ct.CTkLabel(tabview.tab("Add New Client"),text="Enter Client Name")
        cl_namela.grid(row=0,column=0)

        amount2 = ct.CTkLabel(tabview.tab("Buy Managment"),text="Buy Amount")
        amount3_l=ct.CTkLabel(tabview.tab("Sell Managment"),text="Enter Amount")
        amount3_l.grid(row=1,column=0)
        amount2.grid(row=1,column=0)
        cl_lastna=ct.CTkLabel(tabview.tab("Add New Client"),text="Enter Client Last Name")
        cl_lastna.grid(row=1,column=0)

        price2 = ct.CTkLabel(tabview.tab("Buy Managment"),text="Price ")
        price4 = ct.CTkLabel(tabview.tab("Sell Managment"),text="Price")
        price4.grid(row=2,column=0)
        price2.grid(row=2,column=0)
        cl_callla=ct.CTkLabel(tabview.tab("Add New Client"),text="Enter Client Call Number")
        cl_callla.grid(row=2,column=0)

        typee2=ct.CTkLabel(tabview.tab("Sell Managment"),text="Type Of Prudoct")
        typee = ct.CTkLabel(tabview.tab("Buy Managment"),text="Type of Product ")
        typee.grid(row=3,column=0)
        typee2.grid(row=3,column=0)
        cl_addresla=ct.CTkLabel(tabview.tab("Add New Client"),text="Enter Client Address")
        cl_addresla.grid(row=3,column=0)


        #Create Submit Button
        def sub_for_buy():
            date =datetime.now(pytz.timezone('Asia/Kolkata')).strftime("%Y-%m-%d")
            time = datetime.now(pytz.timezone('Asia/Kolkata')).strftime("%H:%M:%S")
            buy_id = random.randint(1,999999)
            row=int(row_mat.get()[0:3])
            sub_for_buy_query="insert into buy values(%s ,%s ,%s, %s,%s,%s,'%s','%s')"%(buy_id,branch_id,c_id.get(),row,amount.get(),price.get(),date,time)

            if execute_non_query(sub_for_buy_query):
                messagebox.showinfo("Success", "Information Successfully Added To Database!")
            else:
                messagebox.showerror("Error", "Failed to add information to the database.")

        def show_table(table):
            # Execute the SQL query to fetch client records
            myquery = f"SELECT * FROM {table}"
            records = execute_query(myquery)

            # Create a Treeview widget to display the records
            tree = Treeview(log_window, height=10)
```

```
212
213             # Define columns
214             if table == 'buy':
215                 tree["columns"] = ("buy_id", "branch_id", "client_id", "Call rm_id"
    , "buy_amount", 'buy_price', 'buy_date', 'buy_time')
216             elif table == 'sell':
217                 tree["columns"] = ("sell_id", "branch_id", "client_id", "product_id
    ", "sell_amount", 'sell_price', 'date_', 'time_')
218             elif table == 'client':
219                 tree["columns"] = ("client_name", "client_last", "client_call", "
    client_address")
220             elif table == 'product':
221                 tree["columns"] = ("product_id", "product_name", "product_amount")
222
223             # Insert column headings
224             for column in tree["columns"]:
225                 tree.heading(column, text=column, command=lambda c=column:
    sort_treeview(tree, c, False))
226
227             # Customizing header colors
228             style = ttk.Style()
229             style.theme_use("clam")  # You can change the theme to match your
    application
230             style.configure("Treeview.Heading", background="blue", foreground="
    white", font=('Helvetica', 10, 'bold'))
231
232             # Adjusting column width and alignment
233             for column in tree["columns"]:
234                 tree.heading(column, text=column.title(), anchor='w')  # Title case
     and left alignment
235                 tree.column(column, width=50, anchor='w')  # Adjust width and left
    alignment
236
237             # Insert data rows with background and foreground colors
238             for i, record in enumerate(records):
239                 if i % 2 == 0:
240                     bg_color = "black"
241                     fg_color = "white"
242                 else:
243                     bg_color = "gray"
244                     fg_color = "black"
245                 tree.insert("", "end", values=record, tags=(f'row{i}',))
246                 tree.tag_configure(f'row{i}', background=bg_color, foreground=
    fg_color)
247
248             # Pack the Treeview widget to the entire log_window
249             tree.pack(fill='both', expand=True, padx=10, pady=10)
250
251             # Function to sort treeview columns
252         def sort_treeview(tv, col, reverse):
253             l = [(tv.set(k, col), k) for k in tv.get_children('')]
254             l.sort(reverse=reverse)
255             # rearrange items in sorted positions
256             for index, (val, k) in enumerate(l):
257                 tv.move(k, '', index)
258             # reverse sort next time
259             tv.heading(col, command=lambda: sort_treeview(tv, col, not reverse))
260
261
262         def sub_for_sell():
263             date =datetime.now(pytz.timezone('Asia/Kolkata')).strftime("%Y-%m-%d")
264             time = datetime.now(pytz.timezone('Asia/Kolkata')).strftime("%H:%M:%S")
265             sell_id = random.randint(1,999999)
266             prd=int(products.get()[0:3])
267             sub_for_sell_query="insert into sell values(%s ,%s ,%s, %s,%s,%s,'%s
    ','%s')"%(sell_id,branch_id,c_id2.get(),prd,amount3.get(),price3.get(),date,time
```

```
    )
            if execute_non_query(sub_for_sell_query):
                messagebox.showinfo("Success", "Information Successfully Added To Database!")
            else:
                messagebox.showerror("Error", "Failed to add information to the database.")

        def add_client():
            client_id = random.randint(1,999999)

            add_client_query =("insert into client values('%s', '%s', '%s', '%s','%s')"%(client_id , cl_name.get(),cl_ln.get(),cl_call.get(),cl_addres.get()))

            if execute_non_query(add_client_query):
                messagebox.showinfo("Success", "Information Successfully Added To Database!")
            else:
                messagebox.showerror("Error", "Failed to add information to the database.")

        def add_employee():
            Employe_id = random.randint(1,999999)
            add_employee_query =("insert into Employee values('%s', '%s', '%s', '%s','%s','%s','%s','%s')"%(Employe_id , im_name.get(),im_last.get(),im_sex.get(),im_call.get(),im_birth.get(),branch_id,im_salary.get()))

            if execute_non_query(add_employee_query):
                messagebox.showinfo("Success", "Information Successfully Added To Database!")
            else:
                messagebox.showerror("Error", "Failed to add information to the database.")


        submit_btn = ct.CTkButton(tabview.tab("Buy Managment"),text="Add To Database",command=sub_for_buy,fg_color='#5c2751')
        submit_btn.grid(row=6,column=1,columnspan=5,padx=10,pady=10,ipadx=250)

        submit_btn2 = ct.CTkButton(tabview.tab("Sell Managment"),text="Add To Database",command=sub_for_sell,fg_color='#5c2751')
        submit_btn2.grid(row=6,column=1,columnspan=2,ipadx=150,padx=10,pady=10)

        client_btn = ct.CTkButton(tabview.tab("Buy Managment"), text="Show Me Available Clients", command=lambda: show_table("buy"))
        client_btn.grid(row=10,column=1,ipadx=150 , padx=10,pady=10,columnspan=2)

        client_btn2=ct.CTkButton(tabview.tab("Sell Managment"),text="Show Me Available Clients",command=show_table("sell"))
        client_btn2.grid(row=10,column=1,ipadx=150,padx=10,pady=10,columnspan=2)

        add_client_btn = ct.CTkButton(tabview.tab("Add New Client"),text="Add This Client",command=add_client,fg_color='#5c2751')
        add_client_btn.grid(row=10,column=1,ipadx=150,padx=10,pady=10,columnspan=2)

        add_emp = ct.CTkButton(tabview.tab("Add New Employee"),text="Add This Employee",fg_color='#47661d',command=add_employee)
        add_emp.grid(row=10,column=1,ipadx=150,padx=10,pady=10,columnspan=2)



def display_data(tab, table_name):
    # Fetch data from the specified table
    query = (f"SELECT * FROM {table_name}")
    table_data = execute_query(query, params=None)

```

```
    # Display data in a table
    table = ct.Treeview(tab)
    for i, row in enumerate(table_data):
        table.insert("", 'end', text=str(i + 1), values=row)
    table.pack(fill=ct.BOTH, expand=True)

# LOGIN PAGE UI
img = Image.open("logo2.png")
img = ImageTk.PhotoImage(img.resize((500, 500)))

tabview = ct.CTkTabview(root, segmented_button_fg_color='#272b33',
                        segmented_button_selected_color='#4149d1',
                        segmented_button_selected_hover_color='green', width=856,
    height=400)
tabview.pack(padx=10, pady=10)
tabview.add("Log in")

# Entry fields for username and password
en_password = ct.CTkEntry(placeholder_text="Password", master=tabview.tab("Log in")
    , text_color='white',width=200)
en_branch_name = ct.CTkEntry(placeholder_text="Branch Name", master=tabview.tab("
    Log in"), text_color='white',width=200)
en_password.pack(side='bottom')
en_branch_name.pack(side='bottom')

# Login button
photo_image = PhotoImage(file='login.png')
button_login = ct.CTkButton(text="Login", master=tabview.tab("Log in"),
    corner_radius=10, command=login_db,
                            hover_color='#243a9c', image=photo_image, compound='
    right',width=200)
button_login.pack(padx=20, pady=20, side='bottom')

# Logo image
img = Image.open("logo2.png")
img = ImageTk.PhotoImage(img.resize((500, 500), Image.Resampling.LANCZOS))
label_logo = ct.CTkLabel(tabview.tab("Log in"), image=img, text="",width=300)
label_logo.pack(side='top')

# Textbox for displaying welcome message
textbox = ct.CTkTextbox(root, state="normal")
textbox.pack(side='bottom')

# Get the current date and time in the Asia/Kolkata timezone
current_time = datetime.now(pytz.timezone('Asia/Kolkata'))

# Format the date and time string
date_string = current_time.strftime("Date: %d %B %Y\n")
time_string = current_time.strftime("Time: %H:%M:%S")

# Insert the formatted date and time string into the textbox
textbox.insert("1.0", "Welcome to BranchHub \n\n" + date_string + time_string + "\n
    ")

# Disable the textbox
textbox.configure(state="disabled")

root.mainloop()
```

SQL Program:

```
use python_ca;
create table employee(
    emp_id int primary key ,
    emp_name varchar(20),
    emp_last varchar(20),
    emp_sex varchar(1),
    emp_call varchar(11),
    emp_birth_day date,
    branch_id int,
    salary int
);
create table branch(
    branch_id int primary key ,
    branch_name varchar(20),
    branch_call varchar(11),
    branch_address varchar(30),
    branch_password int,
    mng_id int ,
    foreign key (mng_id) references employee(emp_id) on delete set null
);
create table client(
    client_id int primary key ,
    client_name varchar(20),
    client_last varchar(20),
    client_call varchar(11),
    client_address varchar(30)

);
create table raw_material(
    rm_id int primary key ,
    rm_name varchar(20),
    rm_type varchar(20)
);

create table product(
    product_id int primary key ,
    product_name varchar(20),
    product_amount int
);
alter table product change product_amount Product_price int;

create table sell(
    sell_id int primary key,
    branch_id int,
    client_id int,
    product_id int,
    sell_amount int,
    sell_price int,
    date_ date,
    time_ time,
    foreign key (branch_id) references branch(branch_id) on delete set null ,
    foreign key (client_id) references client(client_id) on delete set null ,
    foreign key (product_id) references product(product_id) on delete set null
);
create table buy(
    buy_id int primary key, #change
    branch_id int,
    client_id int,
    rm_id int,
    buy_amount int,
    buy_price int,
    buy_date date,
    buy_time time,
    foreign key (branch_id) references branch(branch_id) on delete set null ,
    foreign key (client_id) references client(client_id) on delete set null ,
```

```
    foreign key (rm_id) references raw_material(rm_id) on delete set null
);

create table rm_for_branch(
    rm_id int ,
    branch_id int,
    amount int,
    primary key (rm_id,branch_id),
    foreign key (rm_id) references raw_material(rm_id) on delete cascade ,
    foreign key (branch_id) references branch(branch_id) on delete cascade

);

create table pr_for_branch(
    pr_id int,
    branch_id int,
    amount int,
    primary key (pr_id,branch_id),
    foreign key (pr_id) references product(product_id) on delete cascade ,
    foreign key (branch_id) references branch(branch_id) on delete cascade
);
```

User Interface

Login Page:

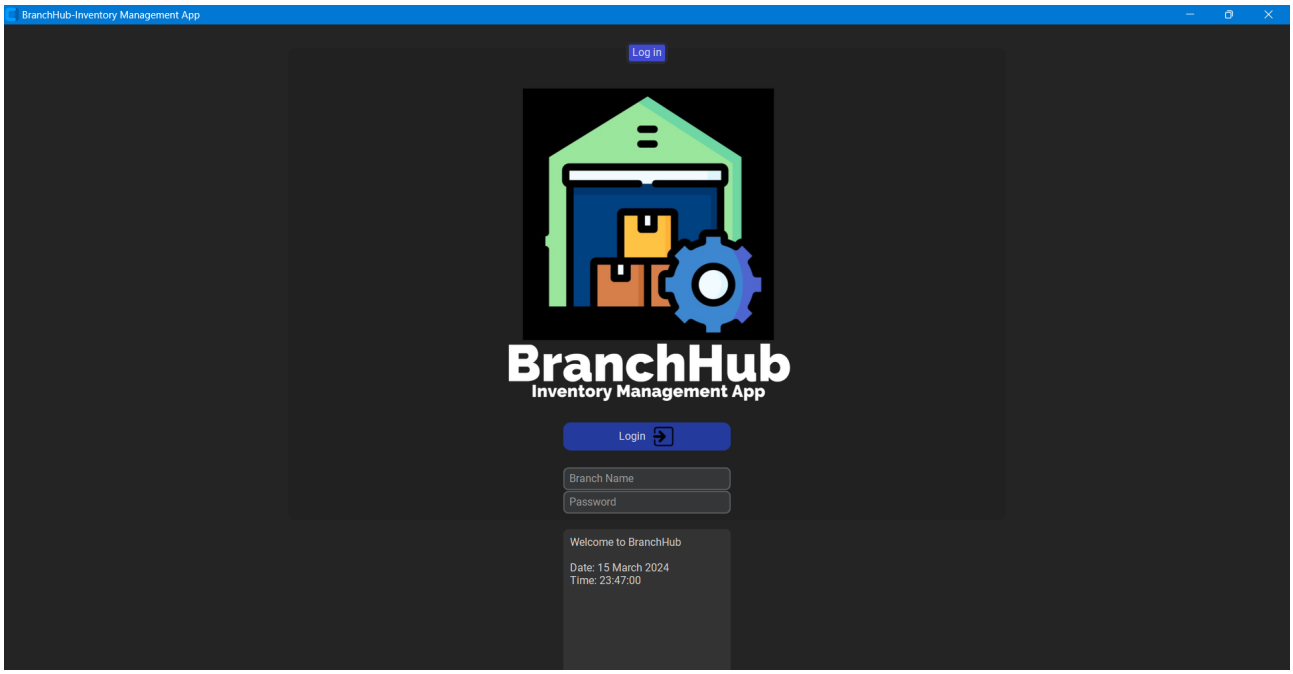

Figure 1: Login$_{Page}$

Login Error:

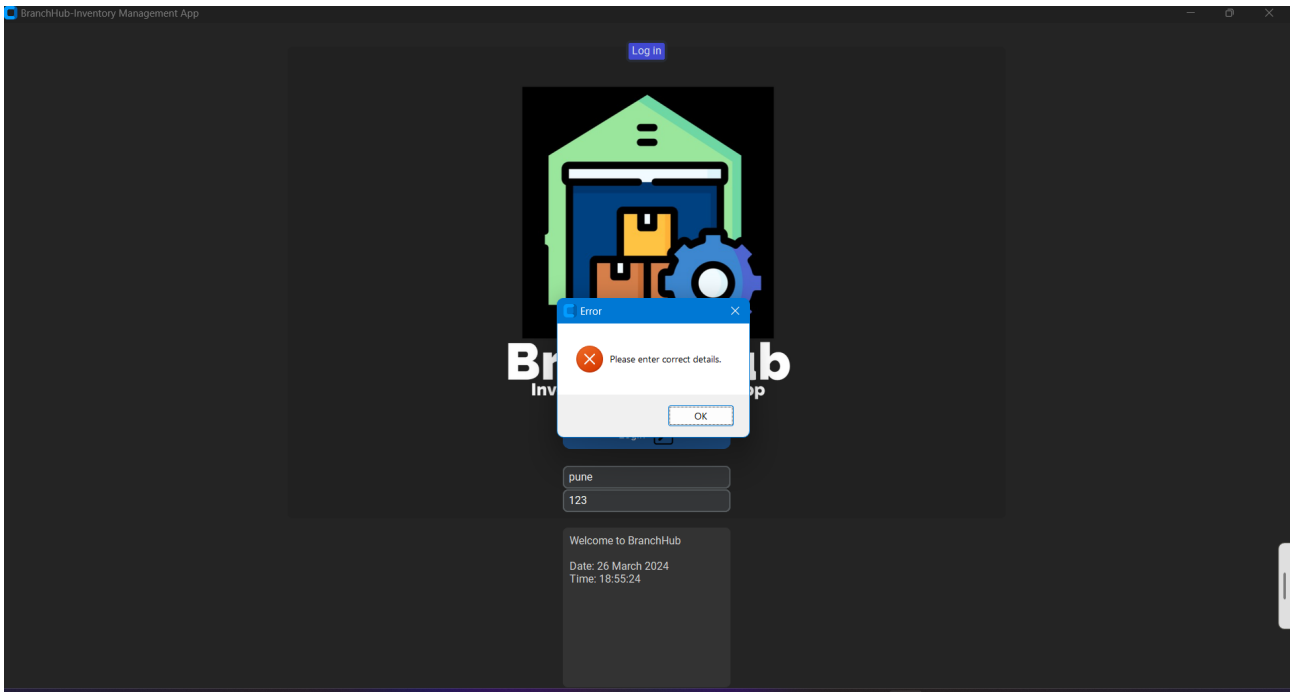

Figure 2: Login$_{Error}$

Main Page:

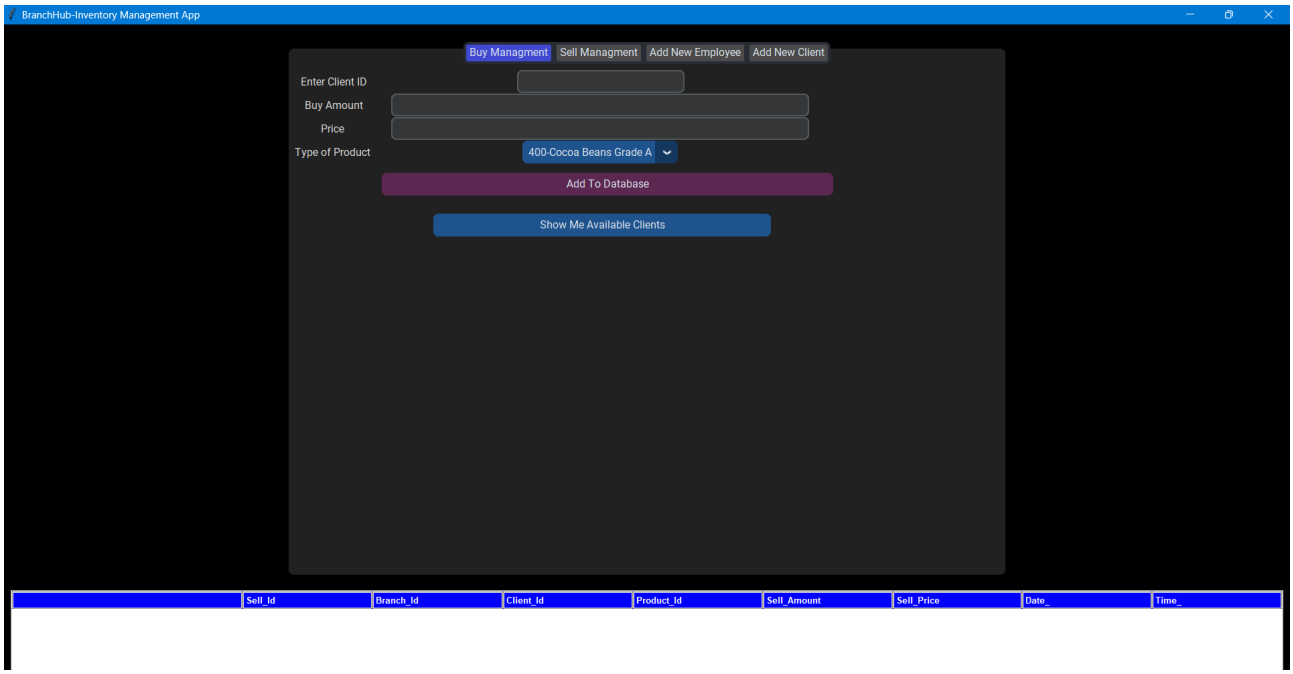

Figure 3: Main$_{P}age$

Update Employee:

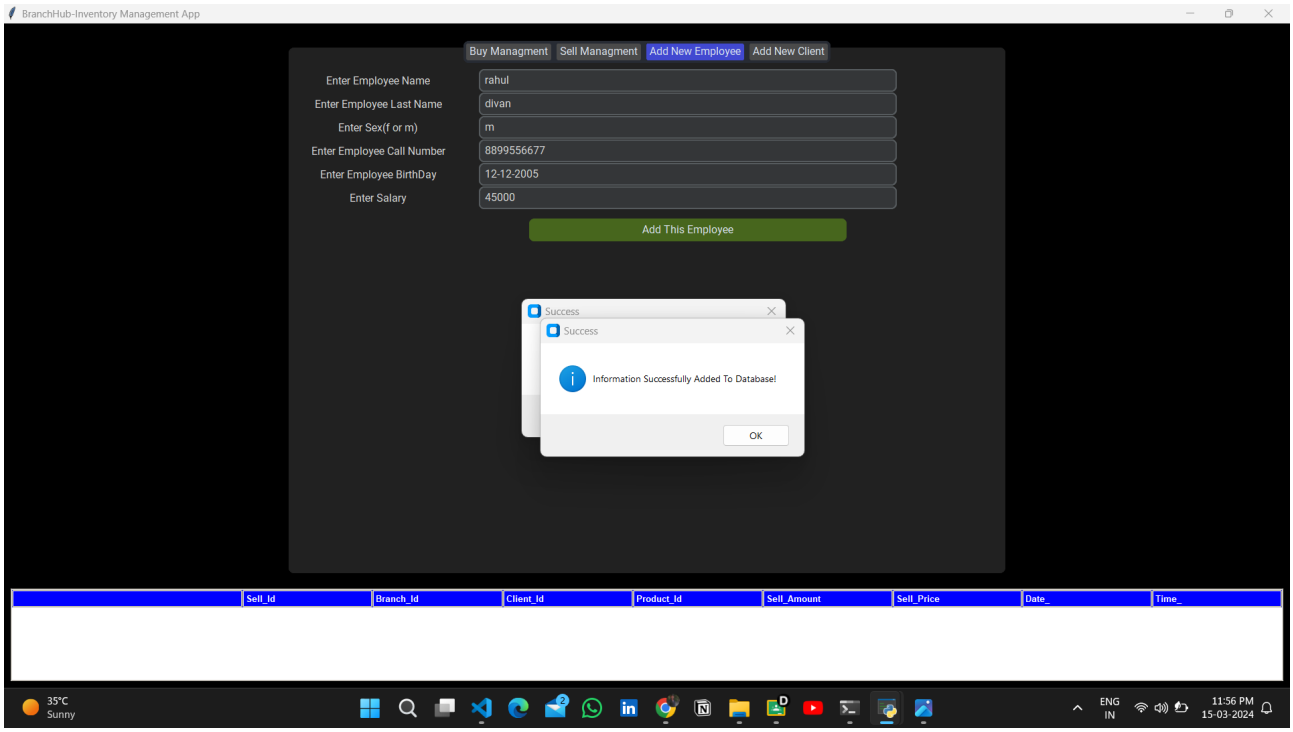

Figure 4: Update$_{E}mployee$

Update Management:

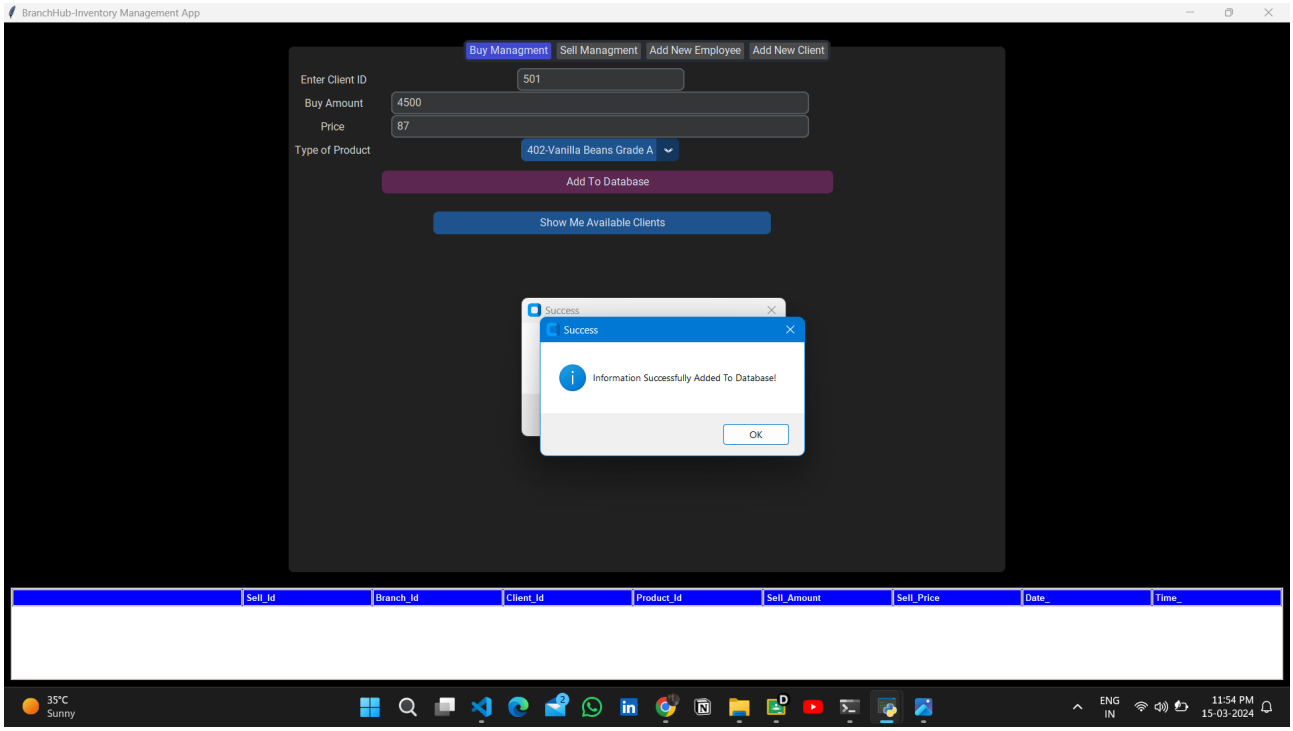

Figure 5: Update$_{M}anagement$