Expense Tracker

Programming for Scientific Computing

Stuti Patel
Kanisha Shah

PACKAGES REQUIRED

pip install tkcalendar

pip install mysql-connector-python

SOFTWARE REQUIREMENTS

MYSQL SERVER

MYSQL WORKBENCH

MYSQL ROUTER

To Download these, use this Link:

https://dev.mysql.com/downloads/file/?id=501541

ORDER OF RUNNING

1. Create Database.py
2. Create Table.py
3. ExpenseTracker.py

FEATURES OF PROJECT

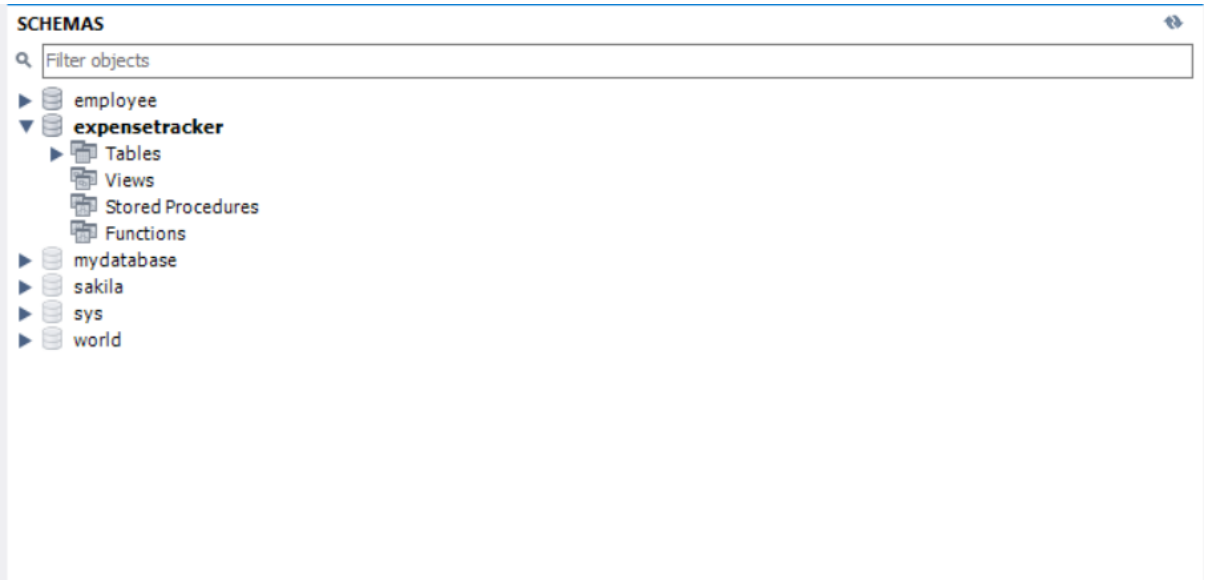
- Welcome page.

- Adding your day-to-day expense
    - To maintain healthy habit one can regularly add their expense to monitor themselves against the misuse of money and learn from their mistakes
    - We have connected the GUI to MYSQL database to store the data for future reference
    - Here, we have validated all the data fields so that wrong data doesn't get added in the database
- Analysing your expense
    - Saving money is the most important lesson in our life. The sooner we learn it the more we save at the end. So these graphs will help you learn this lesson.

INPUT - OUTPUT

CREATE DATABASE

```
import mysql.connector

# It connects you to your Server
myb = mysql.connector.connect(host="localhost", user="root",
passwd="KANISHA*23")

# Returns Object of your Server through which we can modify it
mycursor = myb.cursor()

# It executes the statement
mycursor.execute("CREATE DATABASE ExpenseTracker")
```



CREATE TABLE

```
import mysql.connector

myb = mysql.connector.connect(host="localhost", user="root",
passwd="KANISHA*23", database="ExpenseTracker")

mycursor = myb.cursor()

#Creating table from query
mycursor.execute("CREATE TABLE Expense (DATE_OF_EXPENSE date,TITLE
varchar(20),MONEY int)")

myb.commit()
```

Table: expense

Columns:

DATE_OF_EXPENSE	date
TITLE	varchar(20)
MONEY	int

EXPENSE TRACKER

```
from tkinter import *
from tkinter import ttk
from tkinter import Tk, messagebox
from tkinter.ttk import Notebook
from tkcalendar import DateEntry
import mysql.connector
from matplotlib import pyplot as plt

myb = mysql.connector.connect(host="localhost", user="root",
passwd="KANISHA*23", database="ExpenseTracker")

# Object return points there
mycursor = myb.cursor()


def Add_To_database(a, b, c):
    t = user_input.get().strip()
    print(t)
    adding = "Insert into " + t.lower() + " (DATE_OF_EXPENSE,TITLE,MONEY)
values(%s,%s,%s)"
    entry = (a, b, c)
    mycursor.execute(adding, entry)
    myb.commit()
    print(mycursor.rowcount, "record inserted.")


# validating input fields
def validate():
    a = exp_date_field.get()
    b = title_input.get().strip()
    c = expense_input.get().strip()
    if (len(b) == 0 and len(c) == 0):
        messagebox.showerror("Error", "\tFields can't be empty\nAdd Expense
and proper title for your expense!")
        return False

    elif (len(c) == 0):
        messagebox.showerror("Error", "Expense filed is missing")
        return False

    elif (b == "Select one"):
        messagebox.showerror("Error", "Expense title is missing")
        return False

    val = 0
    try:
        val = float(expense_input.get())
        if (val < 0):
            messagebox.showerror("Error", "Expense can't be negative")
            return False

    except:
        messagebox.showerror("Error", "Enter only numerical value!")
        return False

    return True
```

```
# Adding expense after validating
def Addexpense():
    a = exp_date_field.get()
    b = title_input.get().strip()
    c = expense_input.get().strip()

    if (validate()):
        data = [a, b, c]

        # To show it to user in tree view
        TVExpense.insert('', 'end', values=data)

        Add_To_database(a, b, c)


def nameval():
    c = user_input.get().strip()

    if (len(c) == 0):
        messagebox.showerror("Error", "Username is missing")
        return False
    elif (c.isalpha() != True):
        messagebox.showerror("Error", "Username Can't Contain Numbers")
        return False
    else:
        return True


def already():
    mycursor.execute("SHOW TABLES")
    datab = []

    for x in mycursor:
        s = str(x)[2:-3]
        datab.append(s)
    c = 0
    a = user_input.get().strip()

    for i in datab:
        if a.lower() == i.lower():
            print(i)
            messagebox.showerror("Error", "Username Already Exist")
            return False
        else:
            c = c + 1

    if c == len(datab):
        return True


def removethis():
    wel.destroy()


def remove():
    Name.destroy()


def Not_already():
    mycursor.execute("SHOW TABLES")
```

```
    datab = []

    for x in mycursor:
        s = str(x)[2:-3]
        datab.append(s)
    c = 0
    a = user_input.get().strip()

    for i in datab:
        if a.lower() == i.lower():
            return True
        else:
            c = c + 1

    if c == len(datab):
        messagebox.showerror("Error", "Username Doesn't Exist")
        return False


def login():
    removethis()
    if (nameval()):
        if (Not_already()):
            tab.tab(f1,state='normal')
            tab.tab(f2, state='normal')
            remove()


def signup():
    removethis()
    if (nameval()):
        if (already()):
            t = user_input.get().strip()
            str1 = "Create table " + t + "(DATE_OF_EXPENSE date,TITLE
varchar(20),MONEY int)"
            mycursor.execute(str1)
            myb.commit()
            print(mycursor.rowcount, "record inserted.")
            tab.tab(f1, state='normal')
            tab.tab(f2, state='normal')
            remove()


GUI = Tk()
GUI.title("Expense Tracker")
GUI.geometry('700x430')
# GUI.resizable(0, 0)

# zoomed
# GUI.state('zoomed')

# select page content by clicking on tabs
tab = Notebook(GUI)

# width and height
wel = Frame(tab, width=700, height=430)  # Welcome tab
Name = Frame(tab, width=700, height=430)
f1 = Frame(tab, width=700, height=430)  # Adding daily Expense
f2 = Frame(tab, width=700, height=430)  # Analysis

# adding tabs
```

```
tab.add(wel, text=f'{"Welcome": ^30s}')
tab.add(Name, text=f'{"Login": ^30s}')
tab.add(f1, text=f'{"Expense": ^30s}')
tab.add(f2, text=f'{"Expenditure Analysis": ^30s}')

tab.tab(f1, state='hidden')
tab.tab(f2, state='hidden')


# filling to whole content
tab.pack(fill=BOTH)

# background-color
wel.config(bg="#354a5f")
welcome = Label(wel, text='Expense Tracker', font=('Times New
Roman',36,"bold","italic"), bg="#354a5f", fg="white")
welcome.grid(row=0, column=0,padx=100, pady=100)
# ipadx=100, ipady=100)

next = Button(wel, text='>>', command=removethis,bg="red",fg="white")
next.grid(row=1, column=1, padx=10, pady=70, ipadx=40, ipady=10)

# -----UserName----
Name.config(background="#2b3d4f")
yellow = Label(Name, text="Login / Sign Up ",
bg="#f99406",fg="White",font=('Times New Roman',30,"bold","italic"))
yellow.grid(row=0, column=0,columnspan=10,ipady=10,ipadx=220)

user = Label(Name, text='Username: ', font=('Times New Roman', 24),
bg="#2b3d4f",fg="white")
user.grid(row=3, column=2, padx=55, pady=55)

user_input = StringVar()

user_field = Entry(Name, textvariable=user_input, font=('Times New Roman',
18))
user_field.grid(row=3, column=3, padx=40, pady=55)


# ----Login------
login = Button(Name, text='Login', bg="#1bb6fe",fg="white", font=('Times
New Roman', 18),command=login)
login.grid(row=4, column=2, padx=100, pady=10, ipadx=40, ipady=5)

# ----SigUp------
signup = Button(Name, text='Sign Up',
command=signup,bg="red",fg="white",font=('Times New Roman', 18))
signup.grid(row=4, column=3, padx=0, pady=10, ipadx=30, ipady=5)


f1.config(bg="#2b3d4f")
f2.config(bg="#2b3d4f")

# ----Date------
exp_date = Label(f1, text='Date:', font=('Times New Roman', 18,"bold"),
bg="#2b3d4f",fg="white")
exp_date.grid(row=0, column=0, padx=5, pady=5)

# pip install tkcalendar
exp_date_field = DateEntry(f1, width=19, date_pattern='YYYY/MM/DD',
background='blue',foreground="#2b3d4f",
```

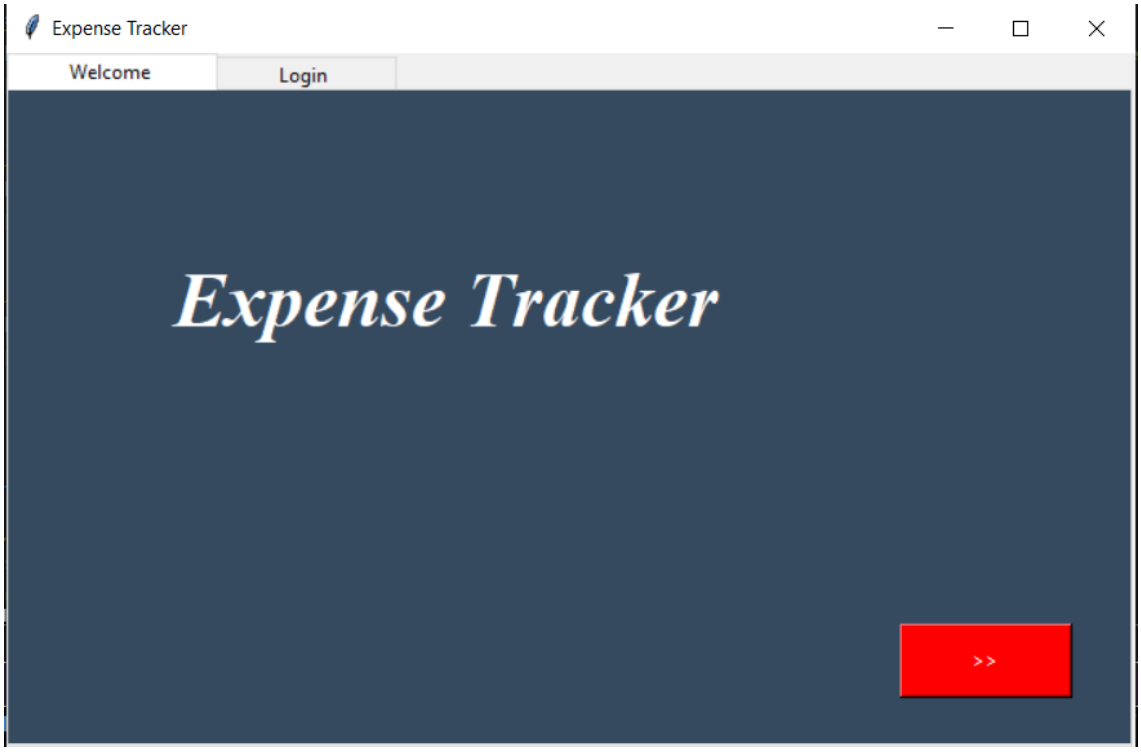
```
                            font=('Times New Roman',
18),bg="#1bb6fe",fg="white")
exp_date_field.grid(row=0, column=1, padx=55, pady=15)

# ----Title------
title = Label(f1, text='Title:', font=('Times New Roman', 18, "bold"),
background="#2b3d4f",fg="white")
title.grid(row=1, column=0, padx=5, pady=15)

title_input = StringVar(GUI)

# Drop down menu
option = [

    "Bill Payment",
    "Stationary",
    "Grocery",
    "Restaurant",
    "Shopping",
    "Withdrawal",
    "Social Cause",
    "Rent"
]

# datatype of menu text
drop = OptionMenu(f1, title_input, *option)
drop.config(width=17, font=('Times Roman', 16),bg="#1bb6fe",fg="white")
title_input.set("Select one")
drop.grid(row=1, column=1, padx=55, pady=15)

# ----Expense------
exp = Label(f1, text='Expense:', font=('Times New Roman', 18, "bold"),
bg="#2b3d4f",fg="white")
exp.grid(row=2, column=0, padx=55, pady=15)

expense_input = StringVar()

exp_field = Entry(f1, textvariable=expense_input, font=('Times New Roman',
18),bg="#1bb6fe",fg="white")
exp_field.grid(row=2, column=1, padx=55, pady=15)

# ----Add Button----
bf1Add = Button(f1, text='Add', command=Addexpense,bg="red", font=('Times
New Roman', 12, "bold"),fg="white")
bf1Add.grid(row=3, column=1, padx=10, pady=10, ipadx=20)

TVList = ['Date', 'Title', 'Expense']
TVExpense = ttk.Treeview(f1, column=TVList, show='headings', height=5)
# for giving column headings
for i in TVList:
    TVExpense.heading(i, text=i.title())
TVExpense.grid(row=4, column=0, padx=45, pady=15, columnspan=3)

# Frame 2
# ----------------------------------------------Expenditure Analysis--------
----------------------------------------


# title = Label(f2, text='Expenditure Analysis', font=('Times New Roman',
34), background="#f99406",fg="white")
# title.grid(row=0, column=0, padx=55, pady=15)
```

```
title = Label(f2, text="Expenditure Analysis ",
bg="#f99406",fg="White",font=('Times New Roman',30,"bold","italic"))
title.grid(row=0, column=0,ipady=10,ipadx=175)


def click_weekly():
    t = user_input.get().strip()
    en = "expensetracker." + t.lower()
    mycursor.execute(
        "Select TITLE, sum(Money) TOTAL_EXPENSE from " + en + " where
DATE_OF_EXPENSE between curdate() - 7 and curdate() group by Title");
    myresult = mycursor.fetchall()

    label = []
    slices = []

    for i in myresult:
        j, k = i
        label.append(j)
        slices.append(k)

    plt.style.use("fivethirtyeight")
    colors = ['Blue', 'Yellow', 'Green', 'Red', 'Orange', 'lightblue',
'pink', 'Purple']
    plt.title("Weekly Chart")
    x, p, texts = plt.pie(slices, colors=colors, radius=1.2,
autopct="%1.1f%%")
    plt.legend(x, label, loc='best', bbox_to_anchor=(-0.1, 1.),
fontsize=15)
    plt.tight_layout()
    plt.show()


# button for knowing the distribution of weekly expense
button_weekly = Button(f2, text='Weekly', command=click_weekly,
bg="#1bb6fe",fg="white",font=('Times New Roman',18))
button_weekly.grid(row=1, column=0, padx=50, pady=30, ipadx=10)


def click_monthly():
    t = user_input.get().strip()
    en = "expensetracker." + t.lower()
    mycursor.execute(
        "Select TITLE, sum(Money) TOTAL_EXPENSE from " + en + " where
DATE_OF_EXPENSE between curdate() - 30 and curdate() group by Title");
    myresult = mycursor.fetchall()  # fetching data from database and then
splitting acc. to need

    label = []
    slices = []
    for i in myresult:
        j, k = i  # As it was stored in tuple of list form
        label.append(j)  # we converted to list
        slices.append(k)
    plt.style.use("fivethirtyeight")  # Style selected
    colors = ['Blue', 'Yellow', 'Green', 'Red', 'Orange', 'lightblue',
'pink', 'Purple']
    x, p, texts = plt.pie(slices, colors=colors, radius=1.2,
autopct="%1.1f%%")  # fixing radius and all
    plt.legend(x, label, loc='best', bbox_to_anchor=(-0.1, 1.),
```

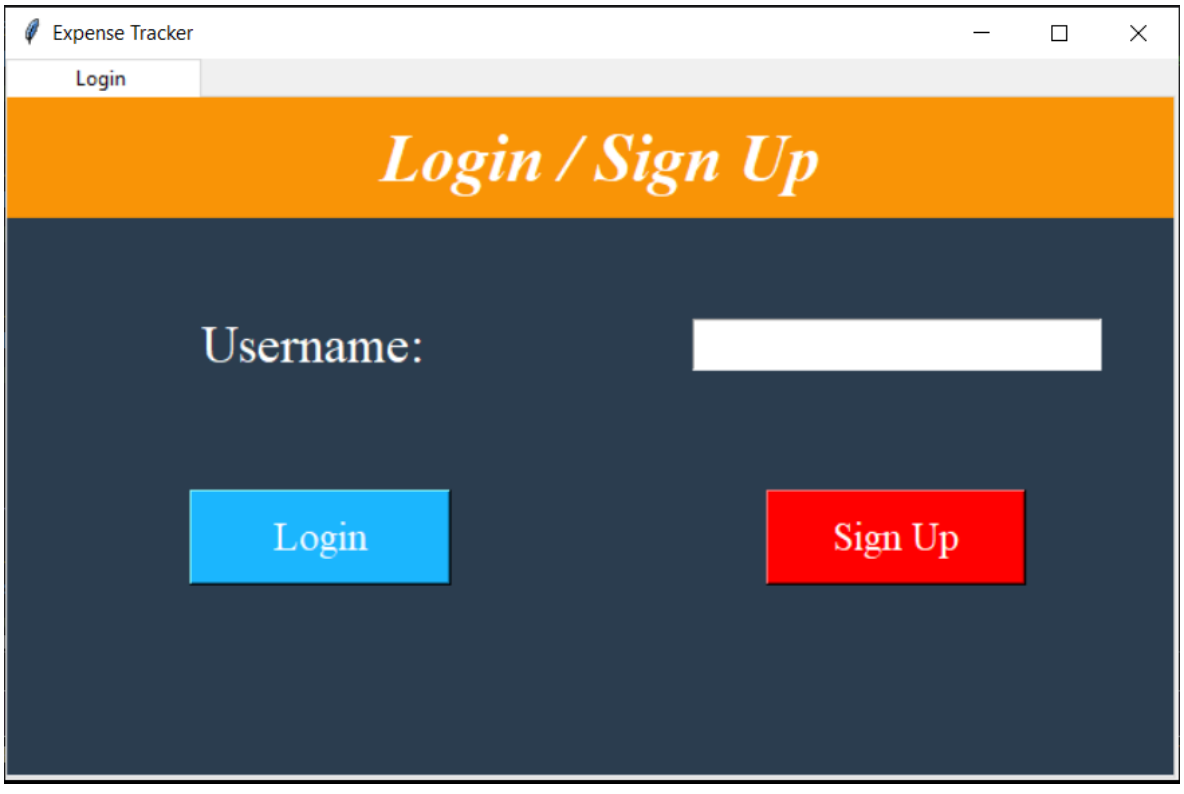
```
fontsize=15)  # Listing the details
    plt.title("Monthly Chart")
    plt.tight_layout()
    plt.show()


# button for knowing the distribution of monthly expense
button_monthly = Button(f2, text='Monthly',
command=click_monthly,bg="#1bb6fe",fg="white",font=('Times New Roman',18))
button_monthly.grid(row=2, column=0, padx=50, pady=30, ipadx=10)


def click_yearly():
    t = user_input.get().strip()
    en = "expensetracker." + t.lower()
    mycursor.execute(
        "Select TITLE, sum(Money) TOTAL_EXPENSE from " + en + " where
DATE_OF_EXPENSE between curdate() - 365 and curdate() group by Title");
    myresult = mycursor.fetchall()

    label = []
    slices = []
    for i in myresult:
        j, k = i
        label.append(j)
        slices.append(k)

    plt.style.use("fivethirtyeight")
    colors = ['Blue', 'Yellow', 'Green', 'Red', 'Orange', 'lightblue',
'pink', 'Purple']
    x, p, texts = plt.pie(slices, colors=colors, radius=1.2,
autopct="%1.1f%%")
    plt.legend(x, label, loc='best', bbox_to_anchor=(-0.1, 1.),
fontsize=15)
    plt.title("Yearly Chart")
    plt.tight_layout()
    plt.show()


# button for knowing the distribution of yearly expense
button_yearly = Button(f2, text='Yearly',
command=click_yearly,bg="#1bb6fe",fg="white",font=('Times New Roman',18))
button_yearly.grid(row=3, column=0, padx=50, pady=30, ipadx=20)




GUI.mainloop()
```

OUTPUT:

Expense Tracker Home Page

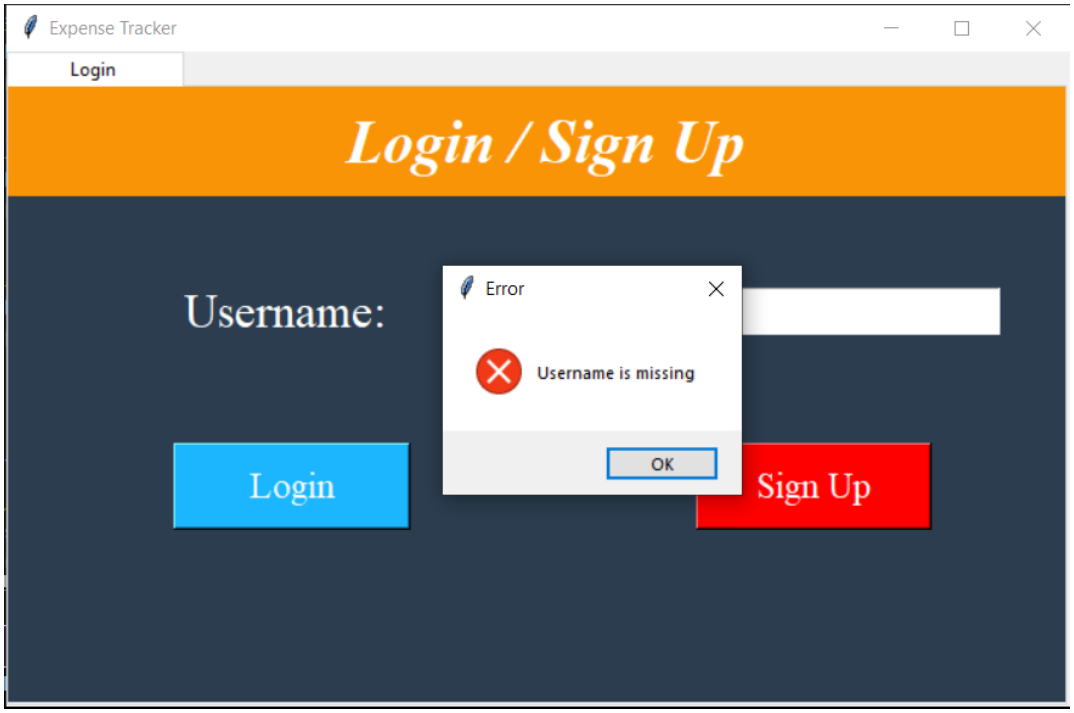


When we click arrow click the welcome page is destroyed and directs us to Login/Sign up page.

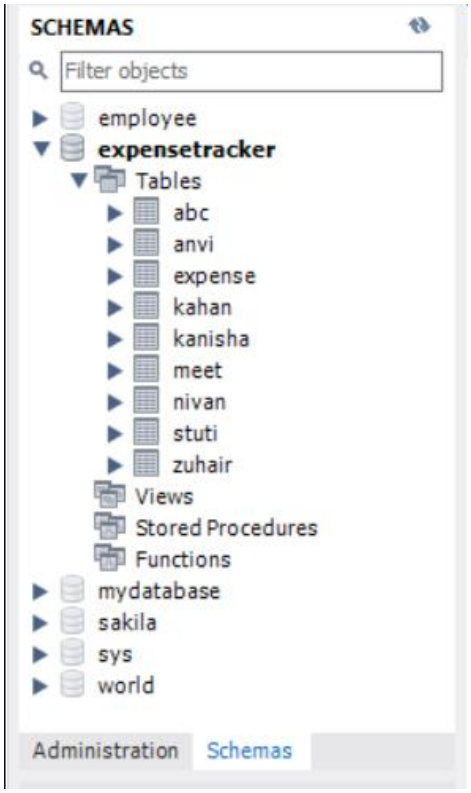


Validations

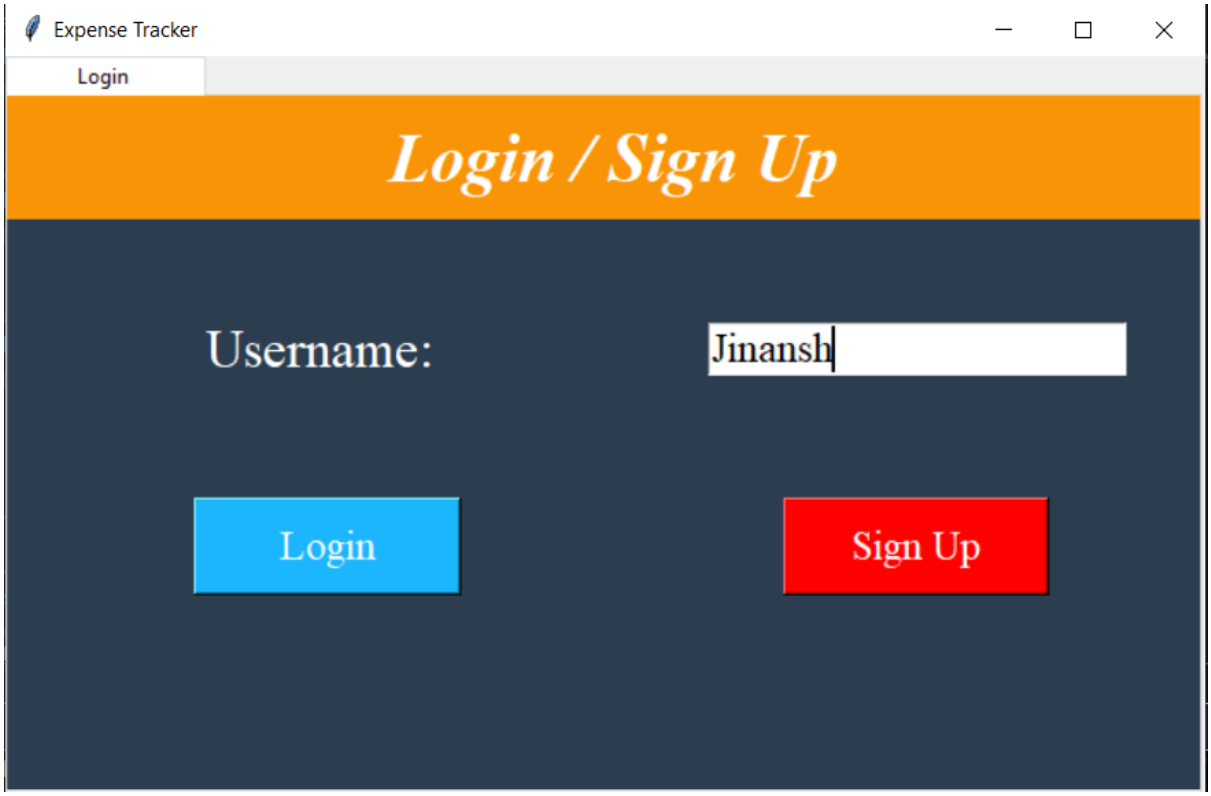
When user clicks login and sign-up button without entering any username it displays error



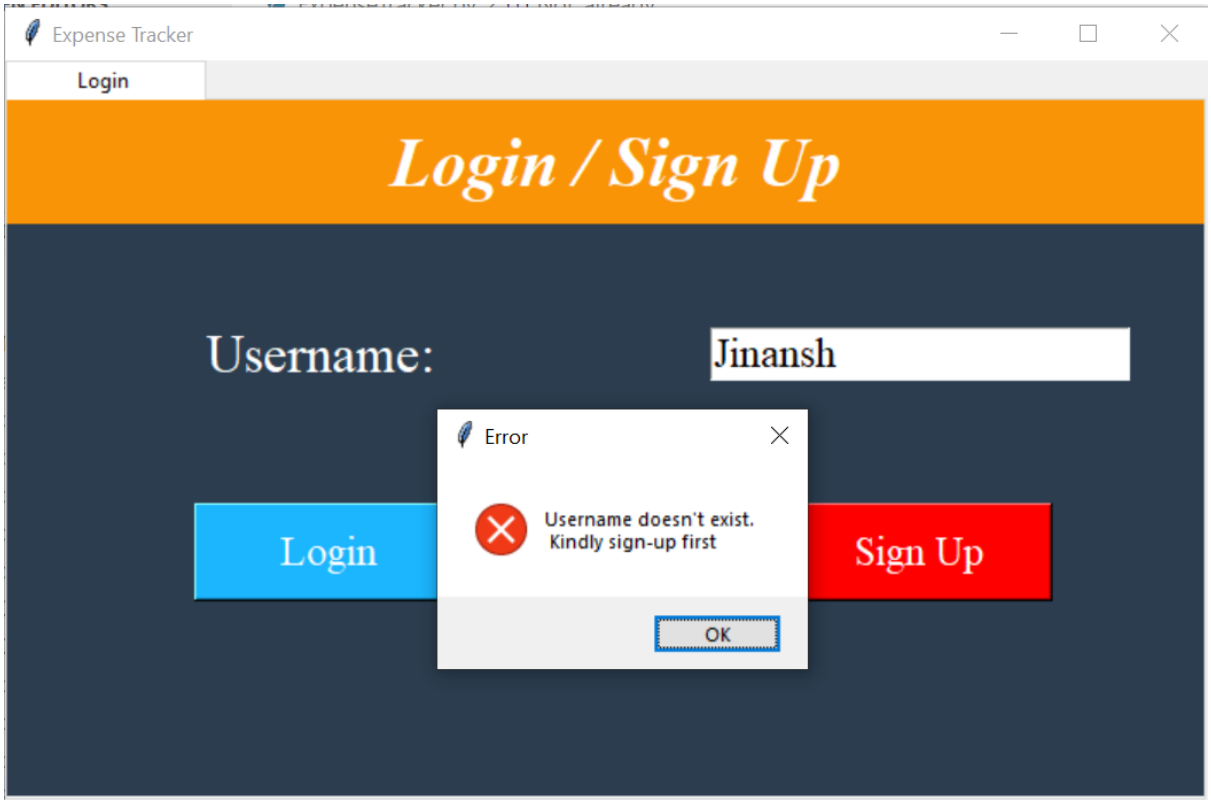
Already existing tables in the database who have signed up previously.



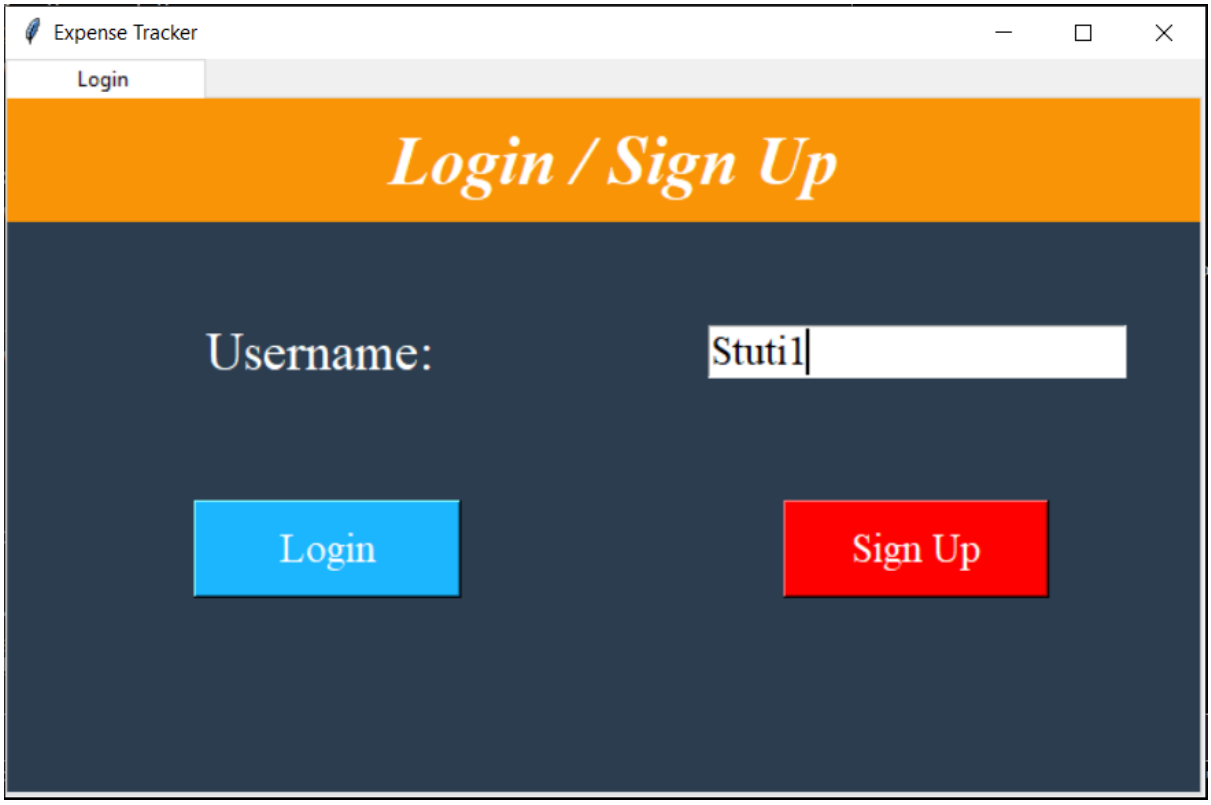
When non existing user tries to login instead of signing up.

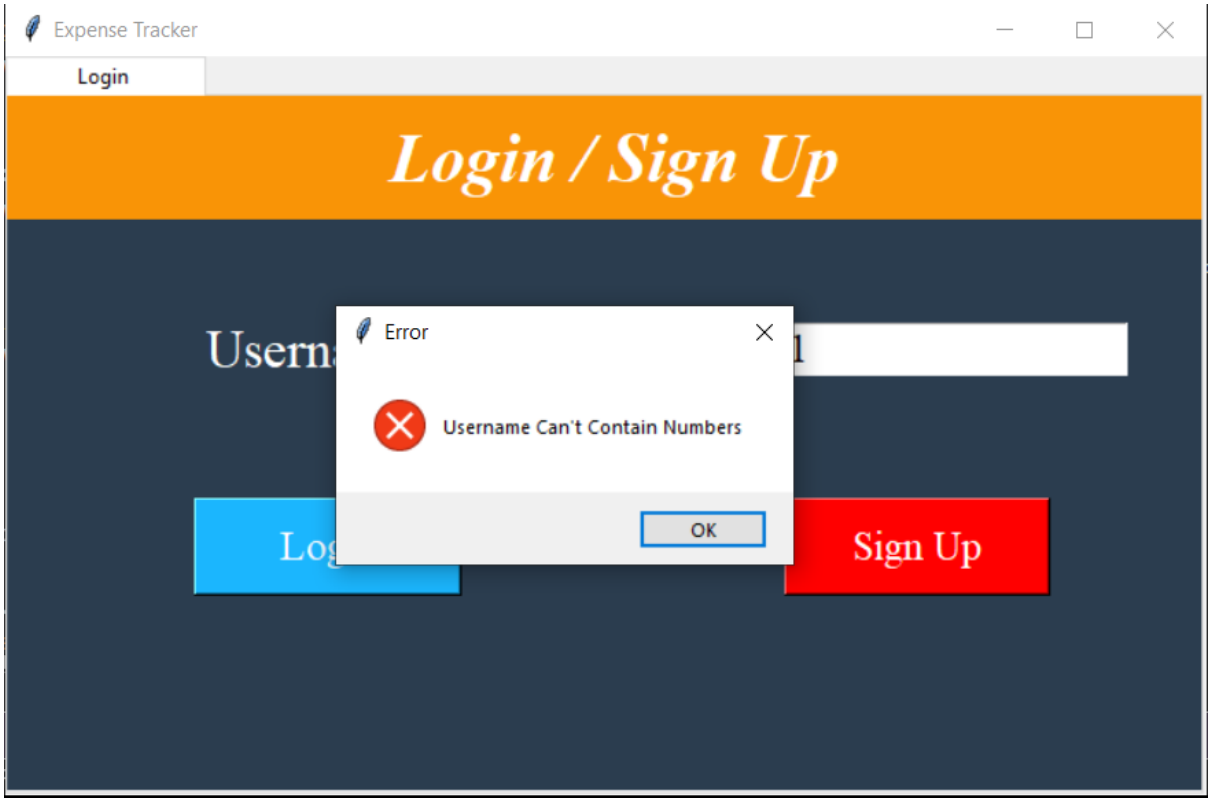


It will show an error saying that username doesn't exist.

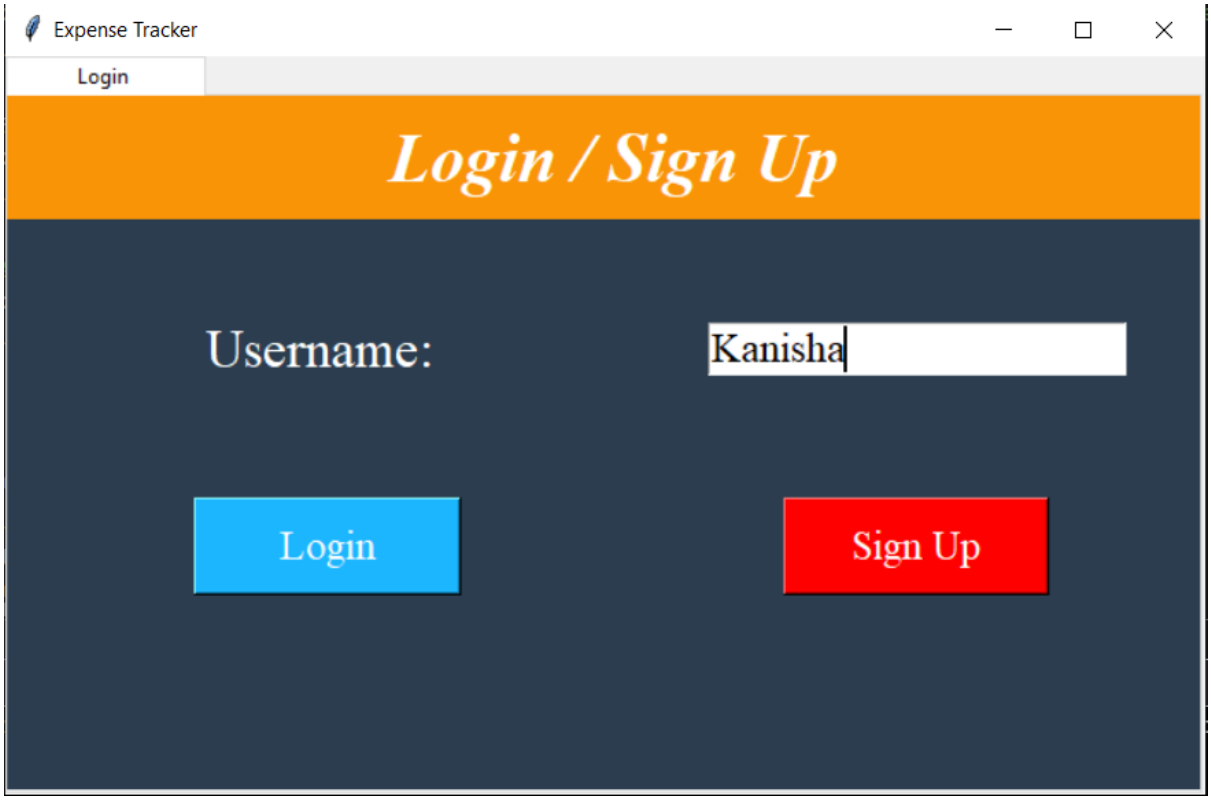


On entering numerical or special in username field to create duplicates isn't permitted. Thus, shows appropriate message.

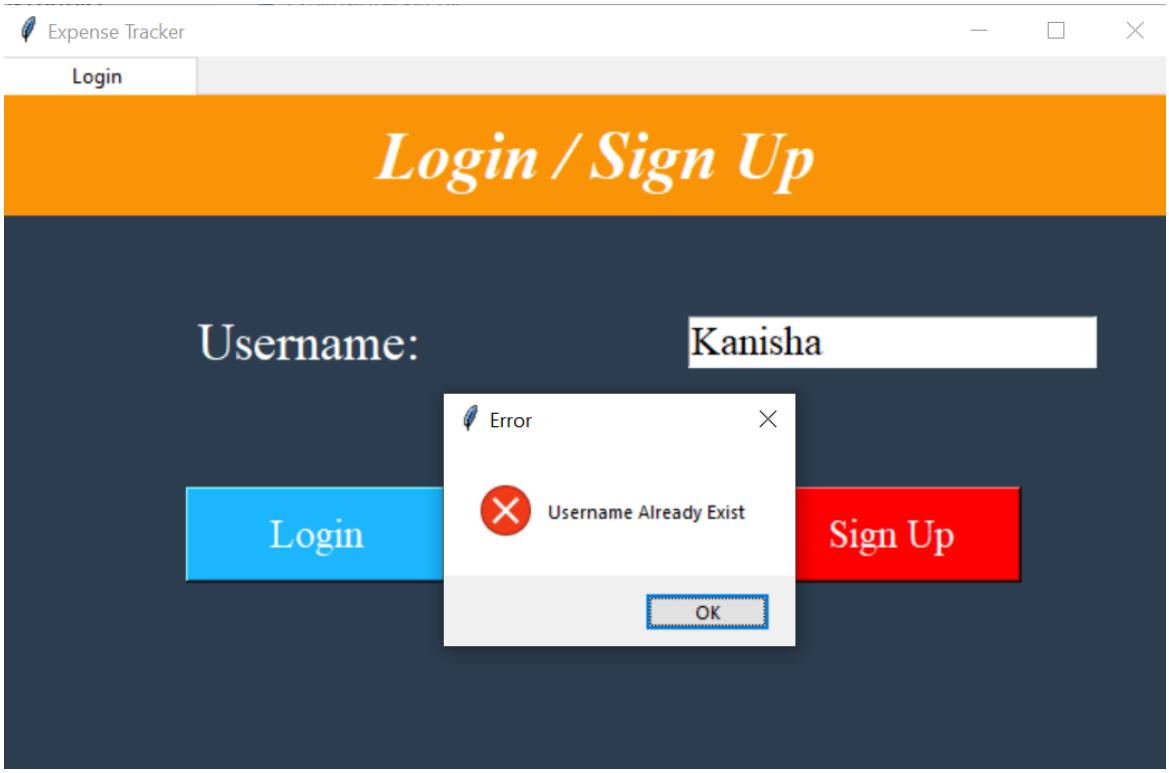




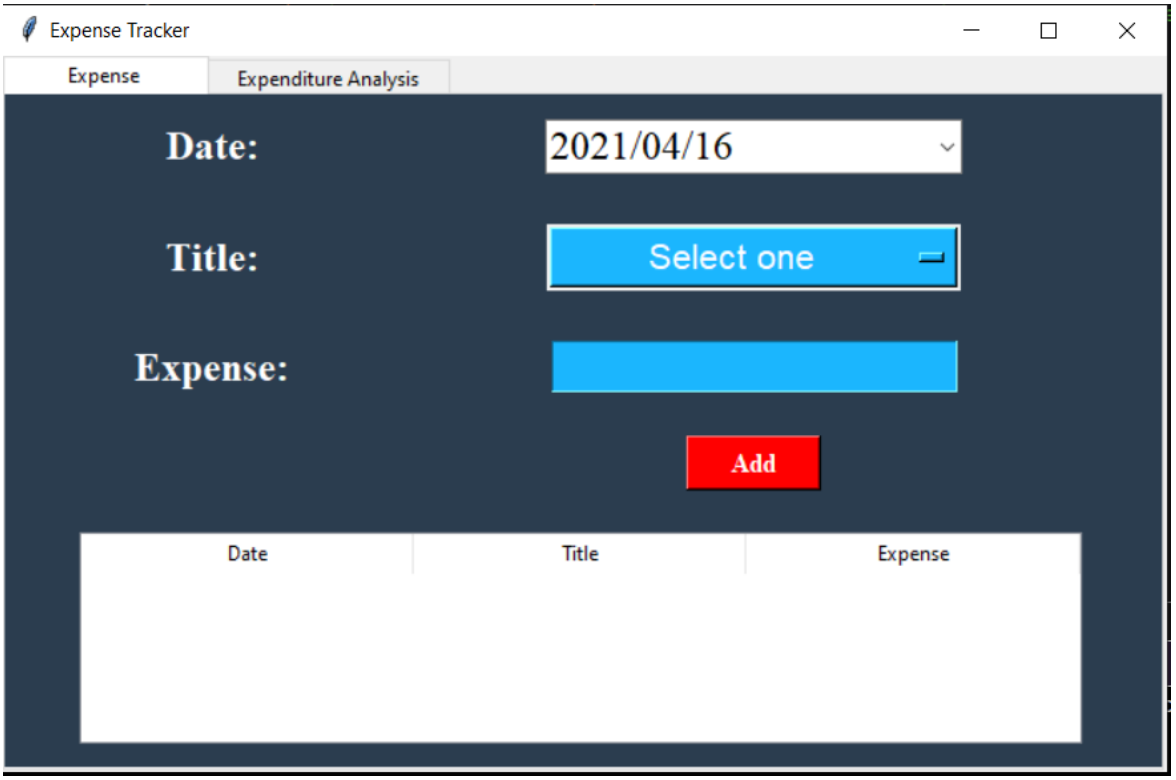
Login



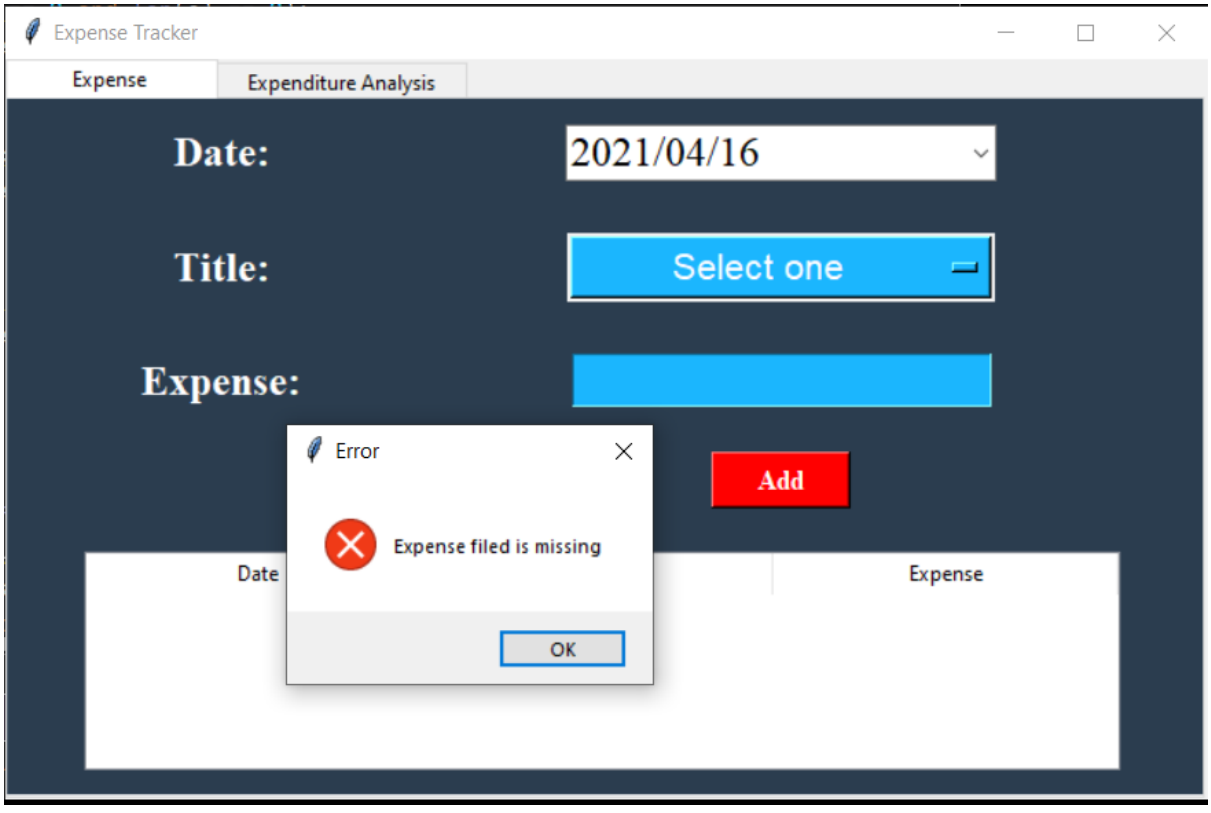
When already existing user tries to sign-up again it will prompt the message.



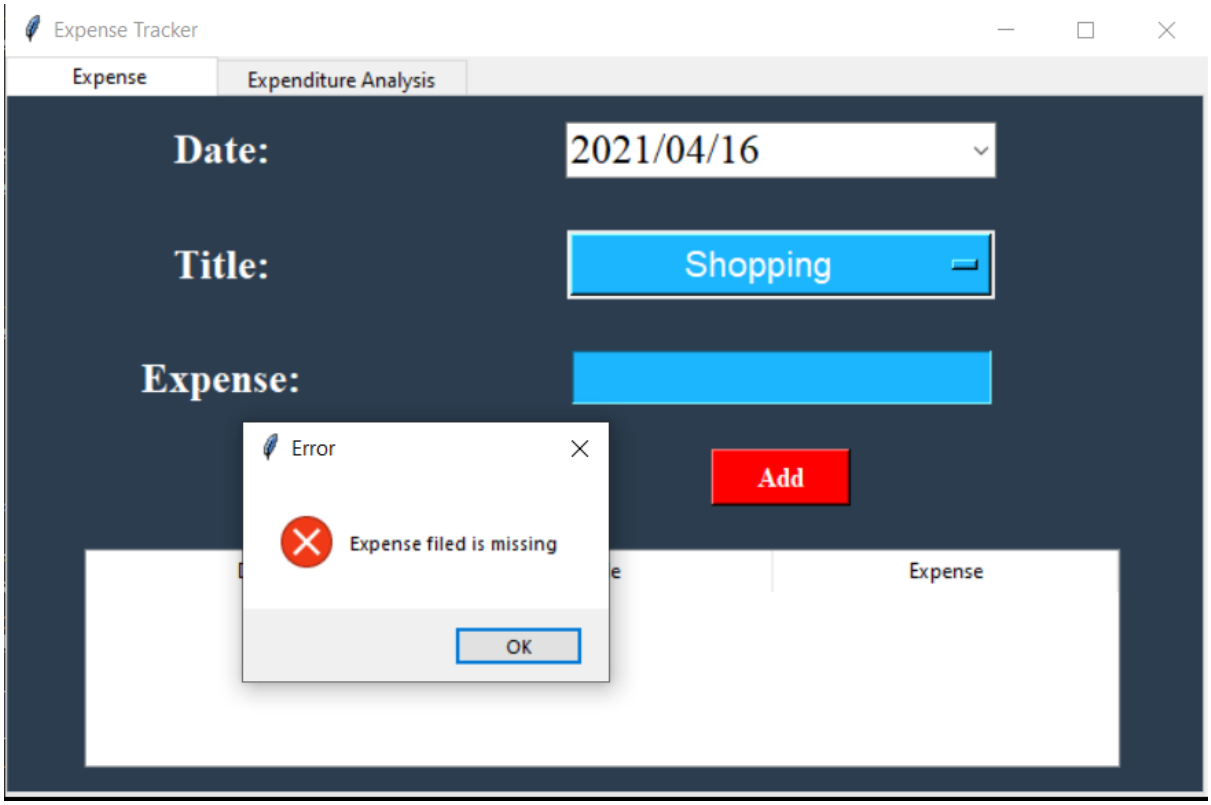
When already existing user clicks Login button it directs them to the main interface (and login page flashes out) where the user can add their daily expense along with the date, they spent it.



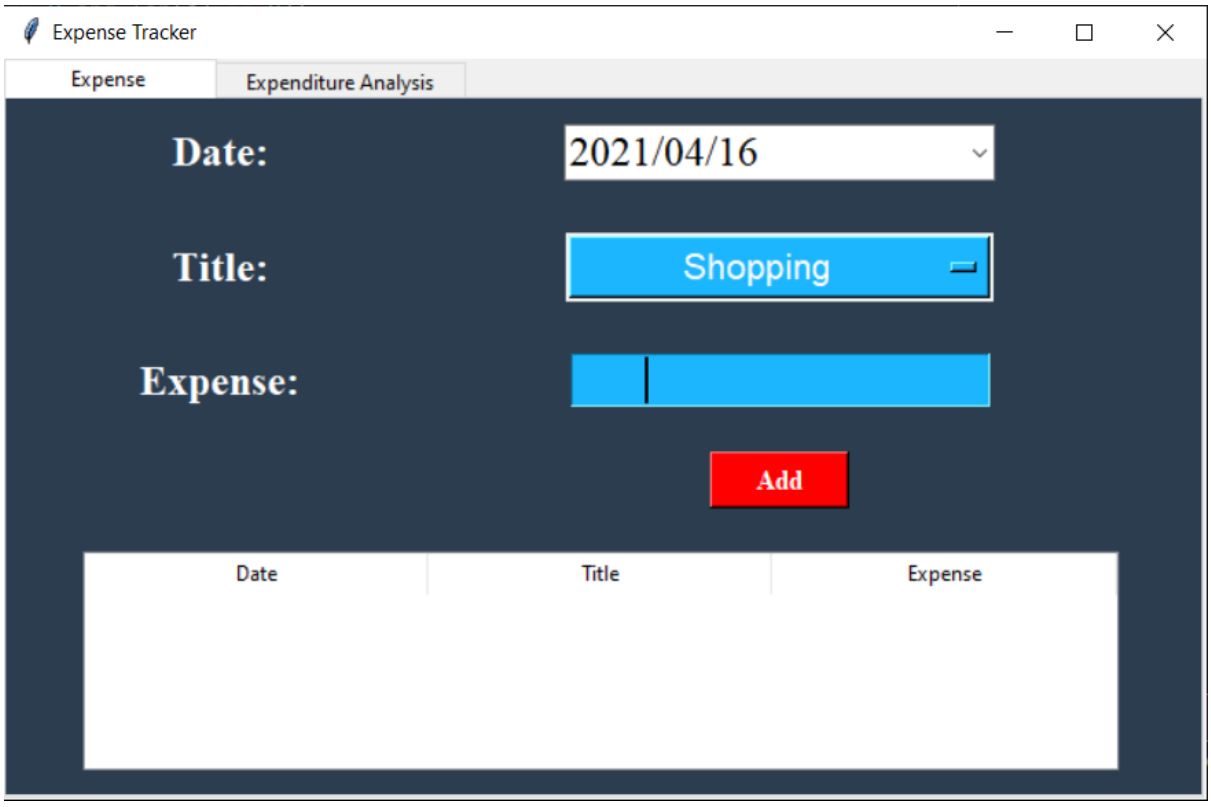
On adding record with empty data fields

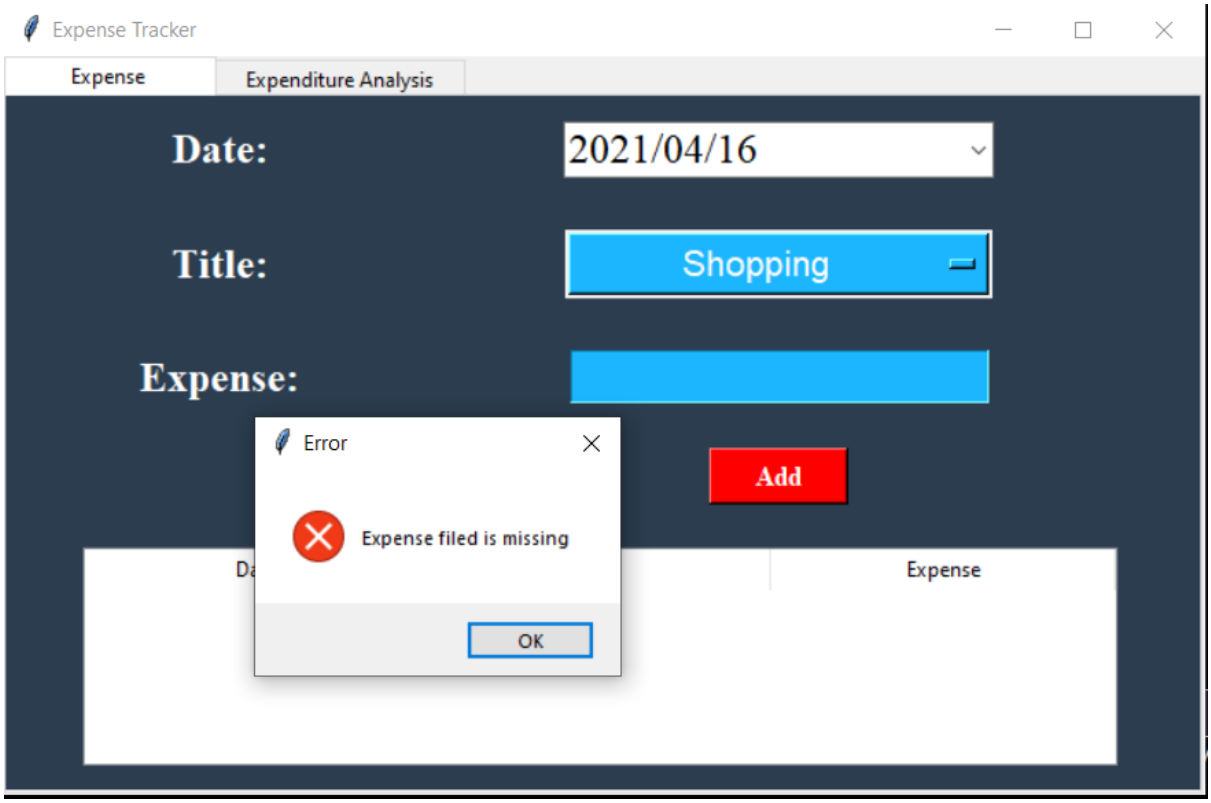


With only expense field empty.

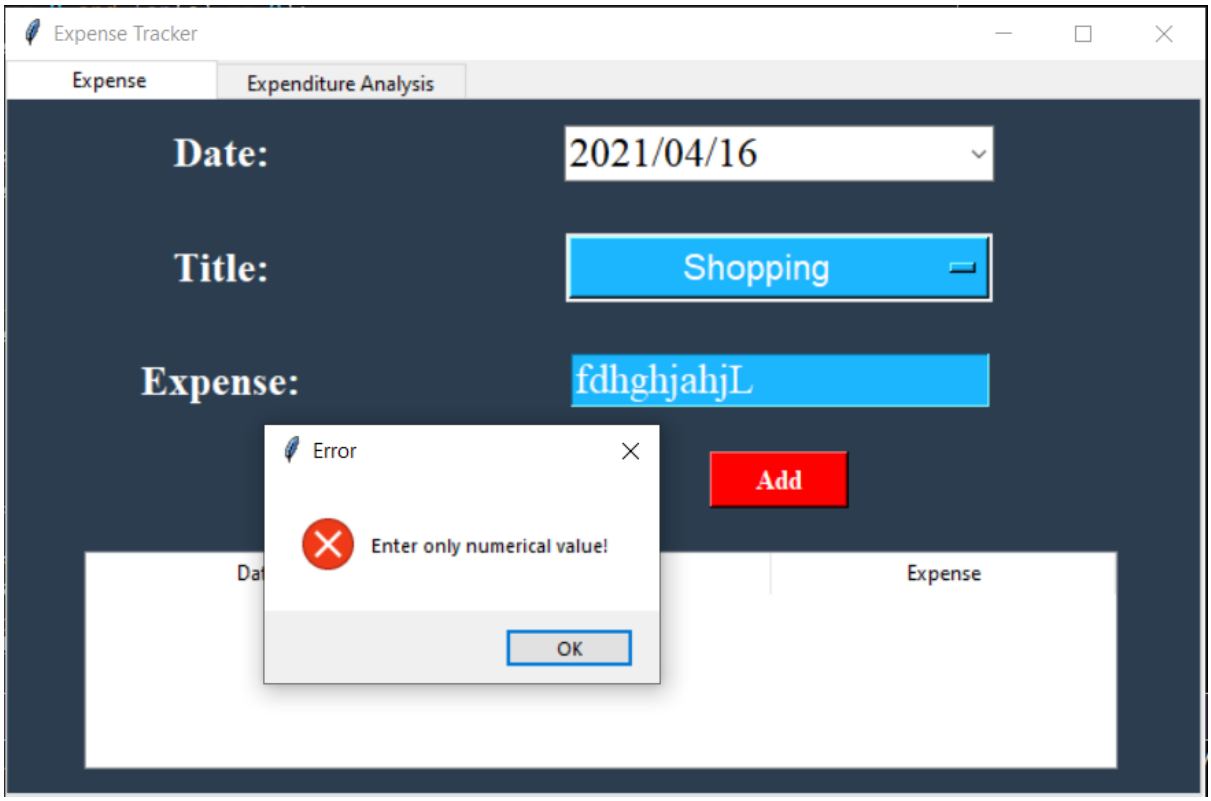


On entering only spaces

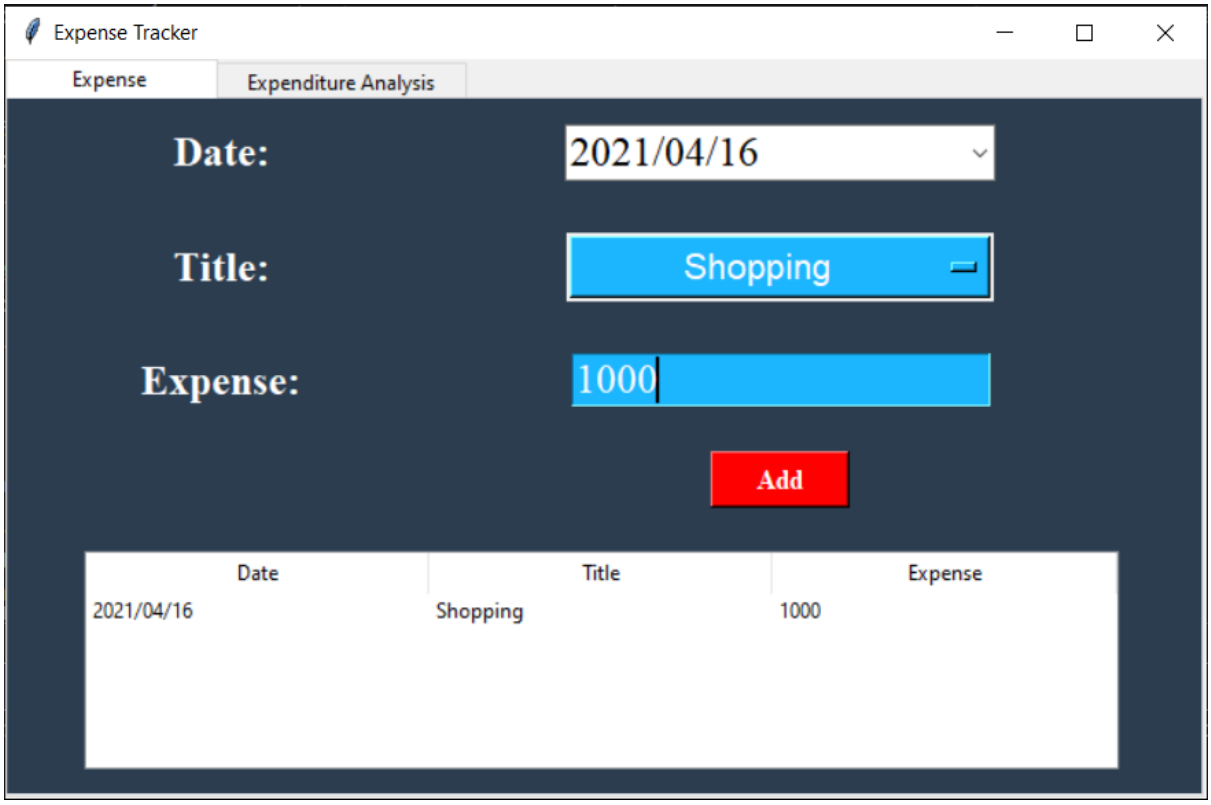




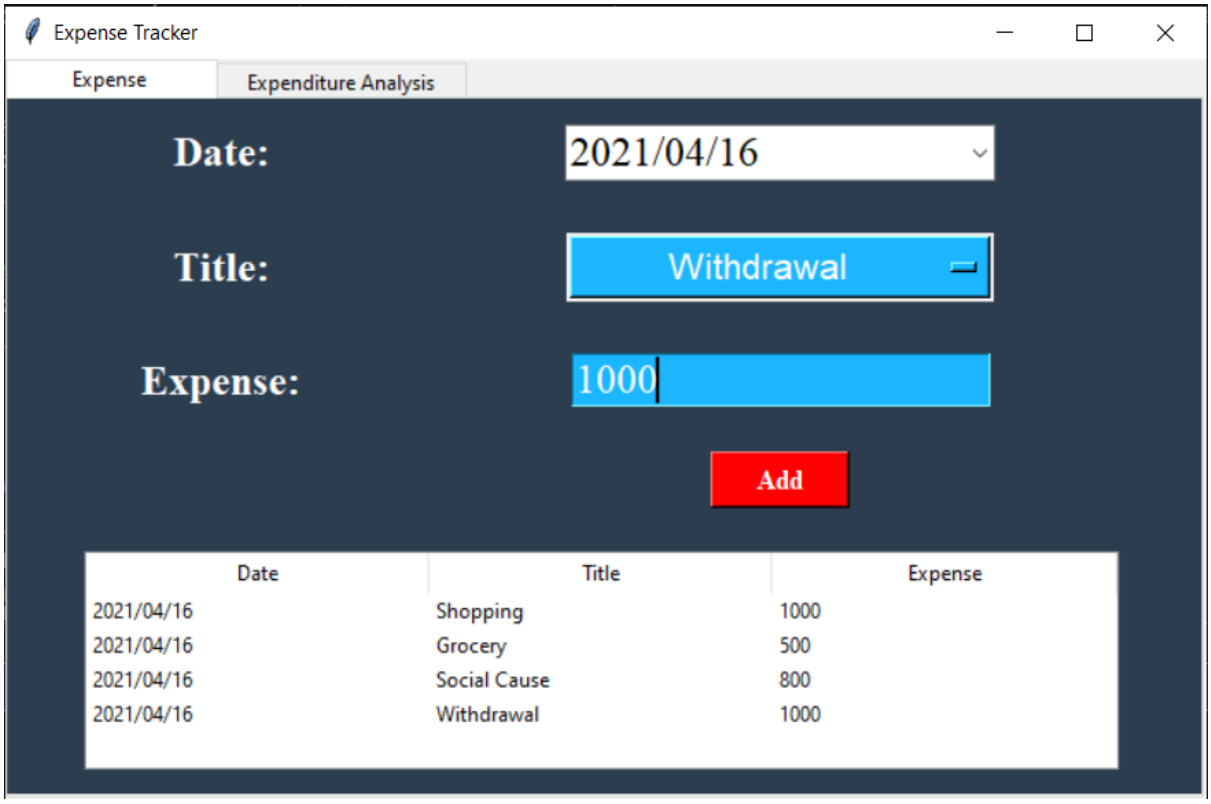
On adding non numeric value in expense field.



On adding all fields correctly

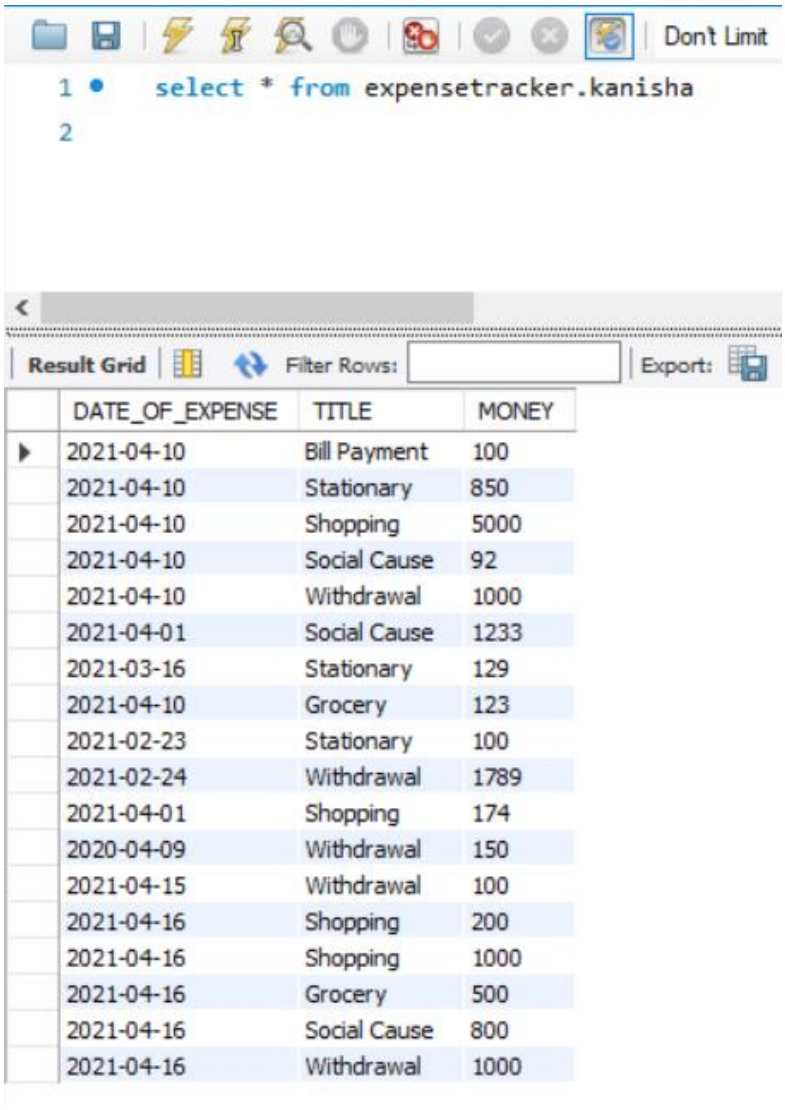


After adding several other records. It will be shown in tabular format to keep track on the data added



Snapshots from MySQL workstation

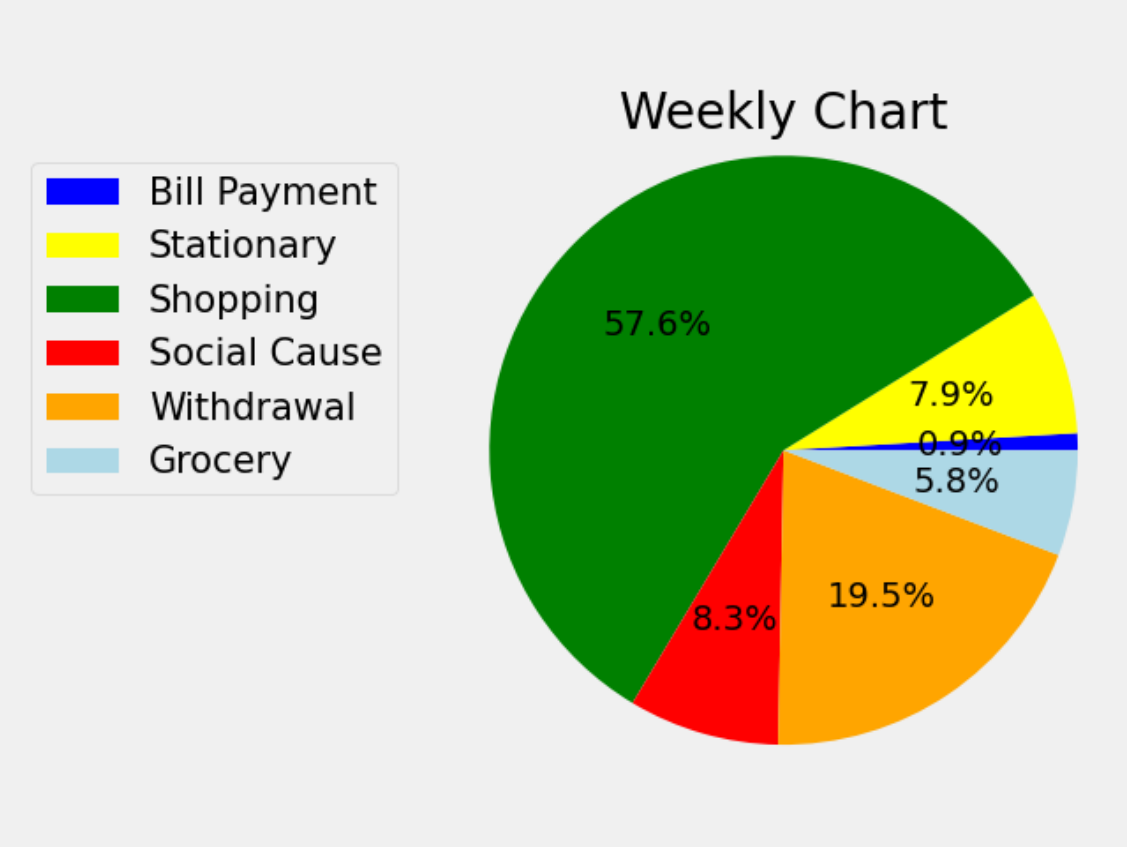
All the previous data will be stored along with the previously stored data.

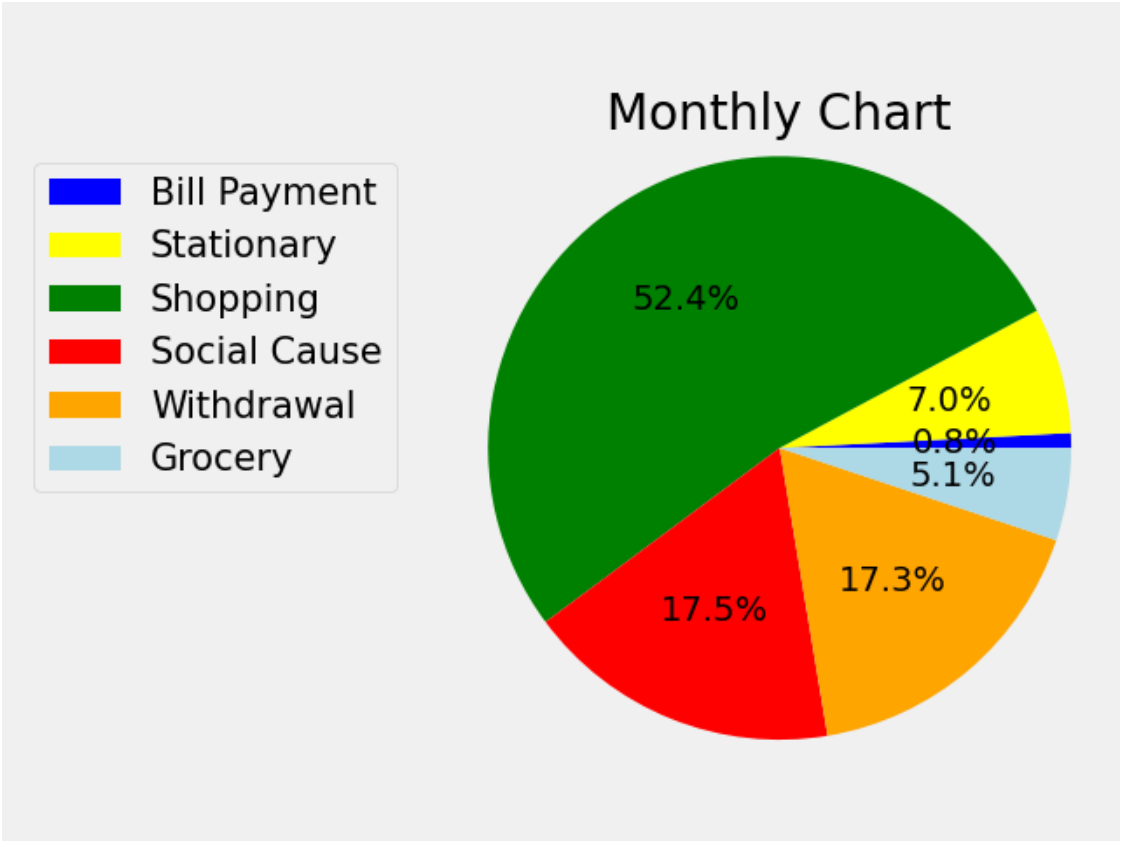


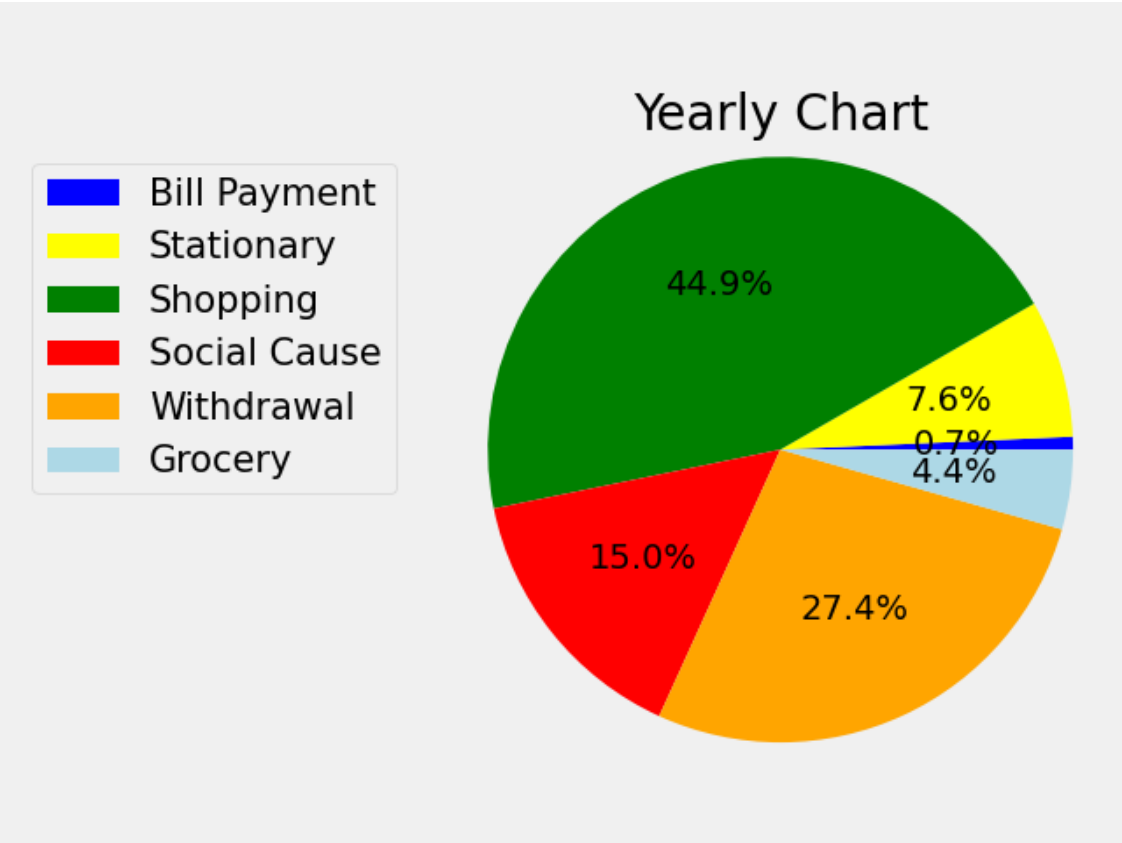
DATE_OF_EXPENSE	TITLE	MONEY
2021-04-10	Bill Payment	100
2021-04-10	Stationary	850
2021-04-10	Shopping	5000
2021-04-10	Social Cause	92
2021-04-10	Withdrawal	1000
2021-04-01	Social Cause	1233
2021-03-16	Stationary	129
2021-04-10	Grocery	123
2021-02-23	Stationary	100
2021-02-24	Withdrawal	1789
2021-04-01	Shopping	174
2020-04-09	Withdrawal	150
2021-04-15	Withdrawal	100
2021-04-16	Shopping	200
2021-04-16	Shopping	1000
2021-04-16	Grocery	500
2021-04-16	Social Cause	800
2021-04-16	Withdrawal	1000

Expenditure Analysis tab

On pressing weekly, monthly and yearly button, it shows your weekly, monthly and yearly analysis of the money spent.

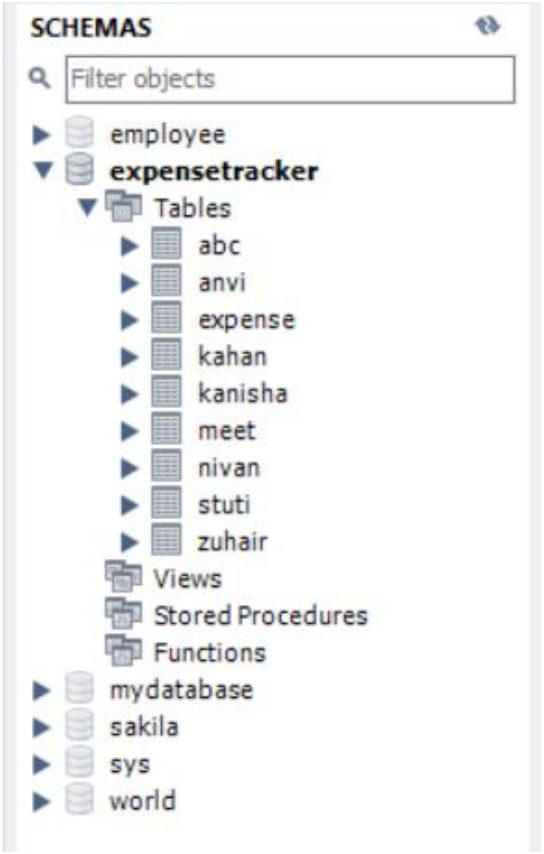


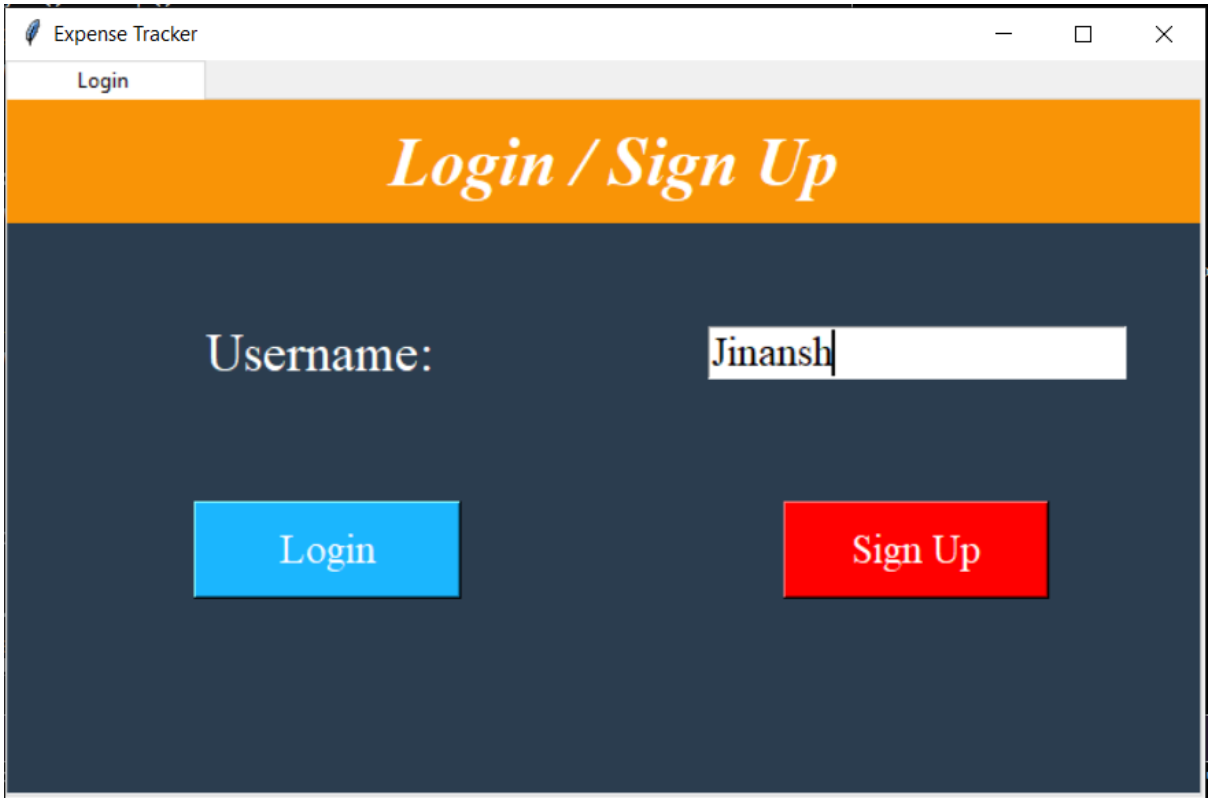

Monthly Chart
Bill Payment
Stationary
Shopping
Social Cause
Withdrawal
Grocery
52.4%
7.0%
0.8%
5.1%
17.3%
17.5%


Yearly Chart
Bill Payment
Stationary
Shopping
Social Cause
Withdrawal
Grocery
44.9%
7.6%
0.7%
4.4%
15.0%
27.4%

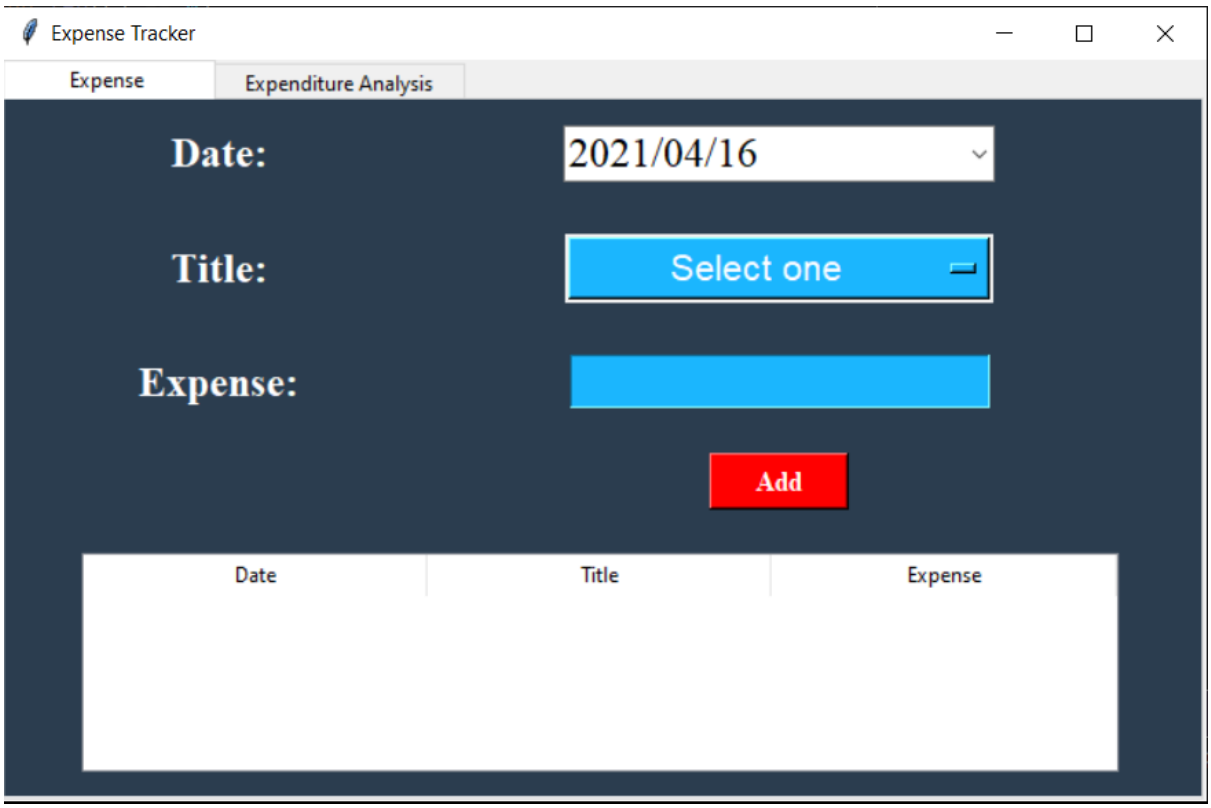
Sign up

Previously existing table in the database

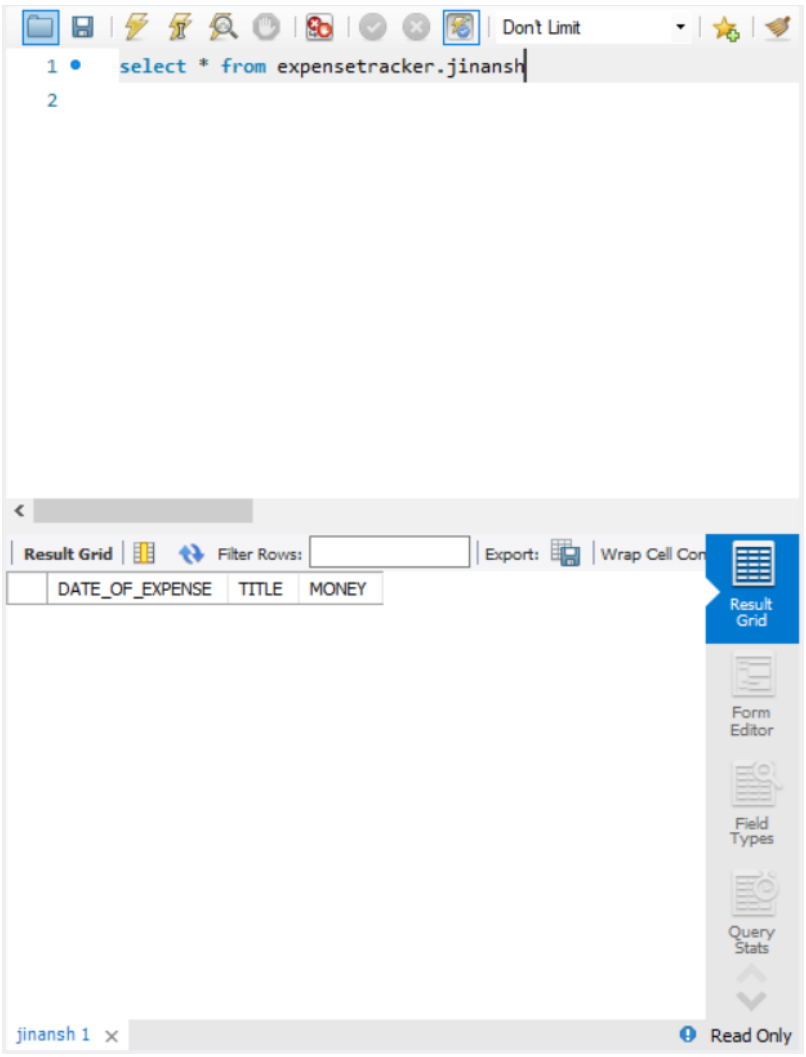




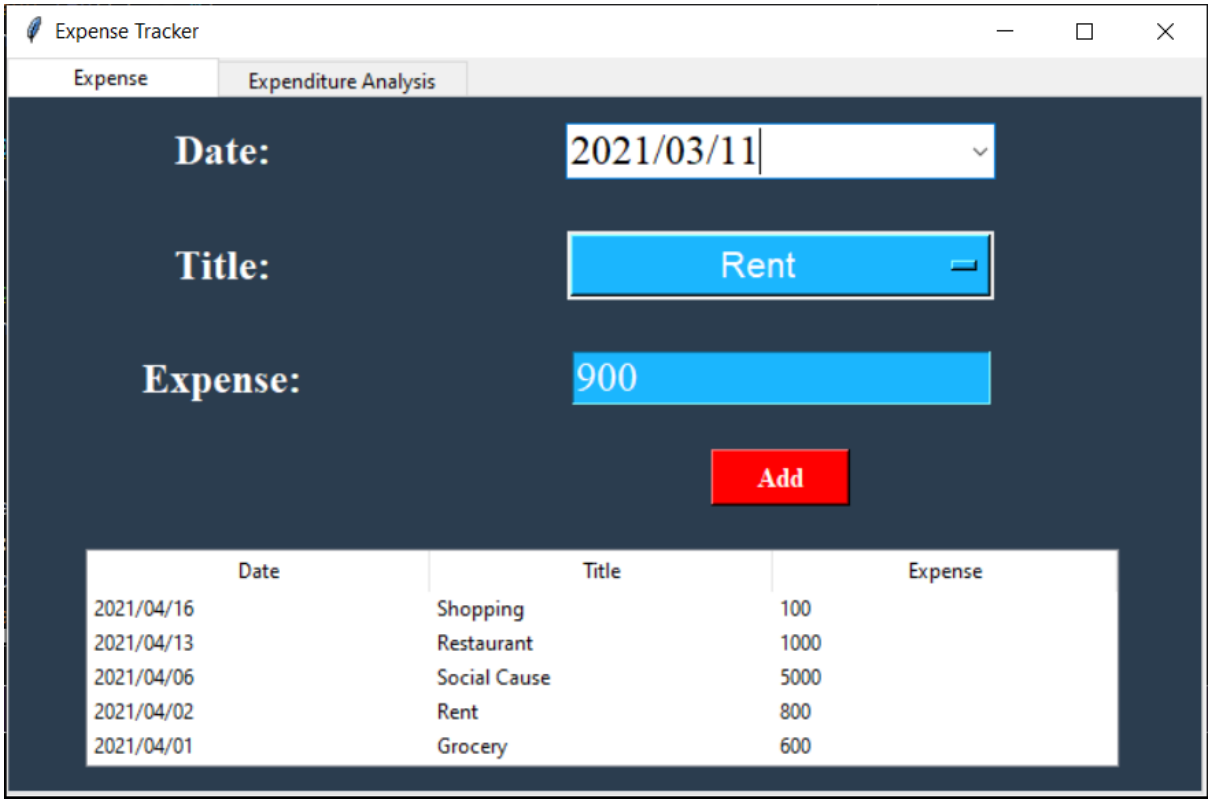
After clicking sign up button user enters the main interface.

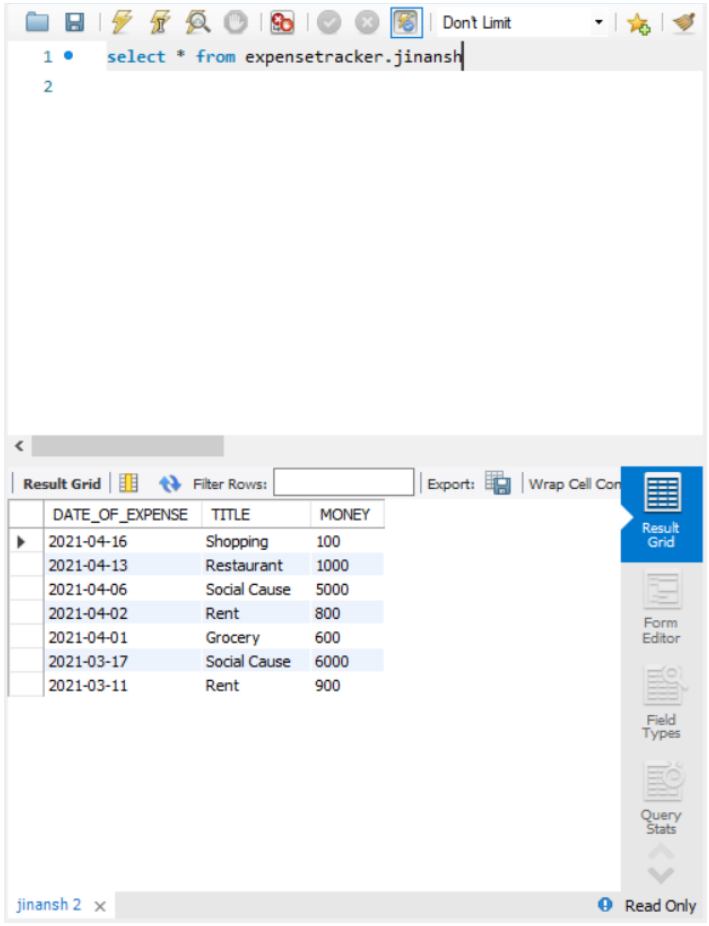


Snapshots from database

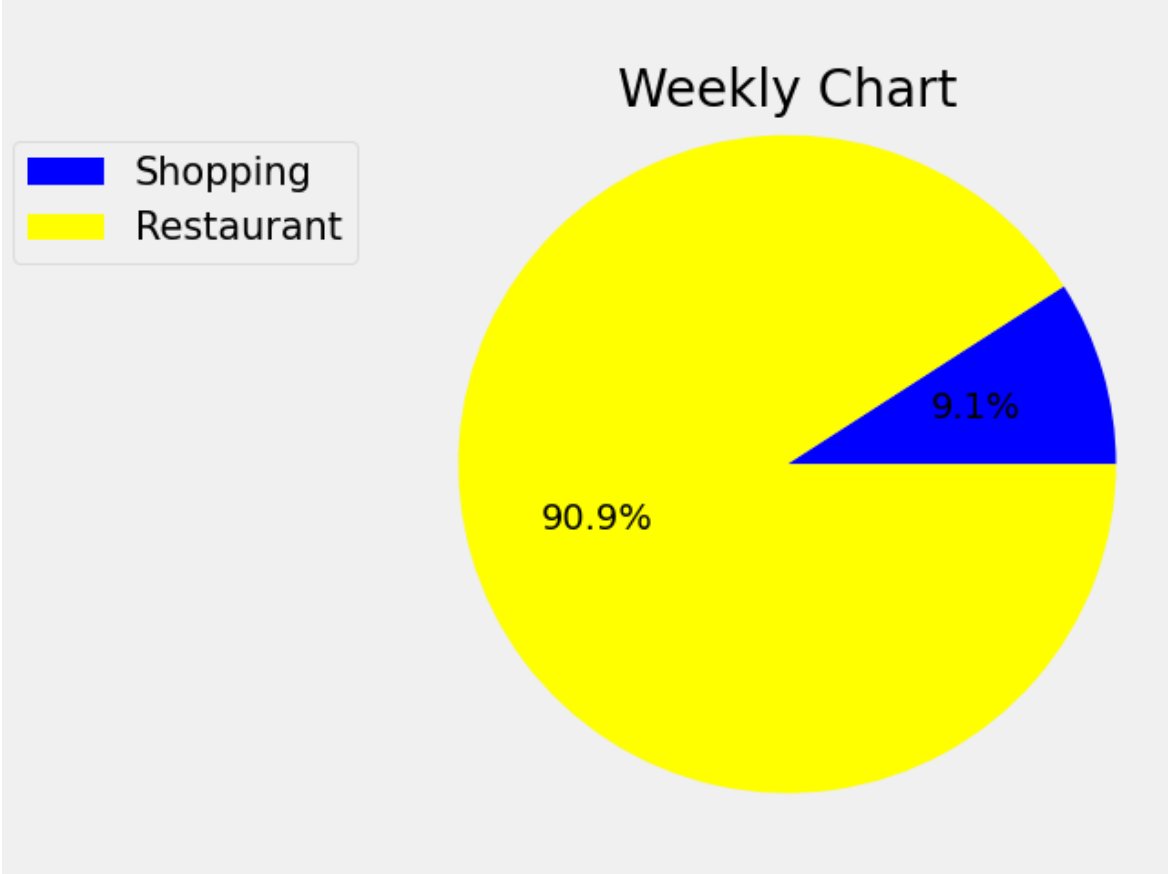


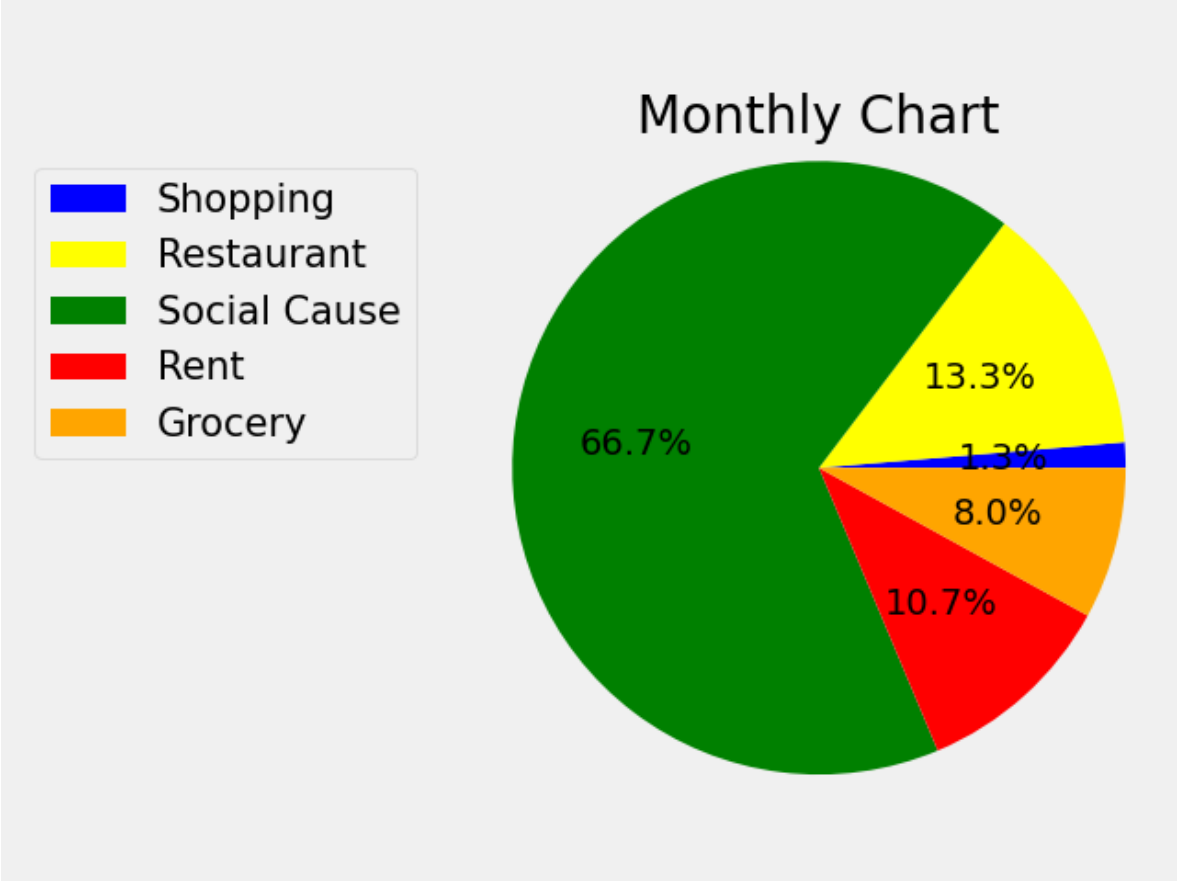
Adding expenditure in the database

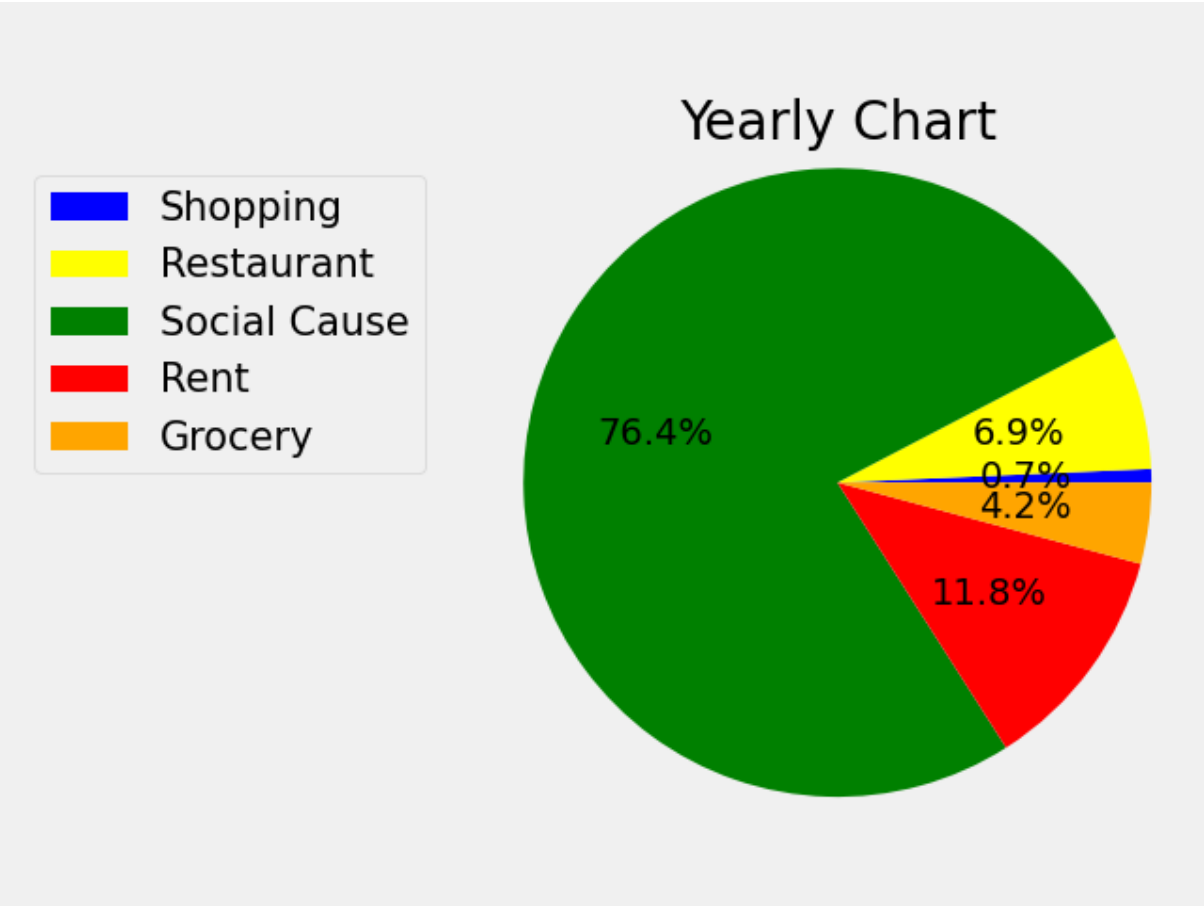




Similar to login we get the desired graphs to analyse our usage of money.






Yearly Chart
Shopping
Restaurant
Social Cause
Rent
Grocery
76.4%
6.9%
0.7%
4.2%
11.8%