Expense Tracker

Innovative Assignment

Programming for Scientific Computing

Kanisha Shah

19BCE253

Stuti Patel

19BCE269

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

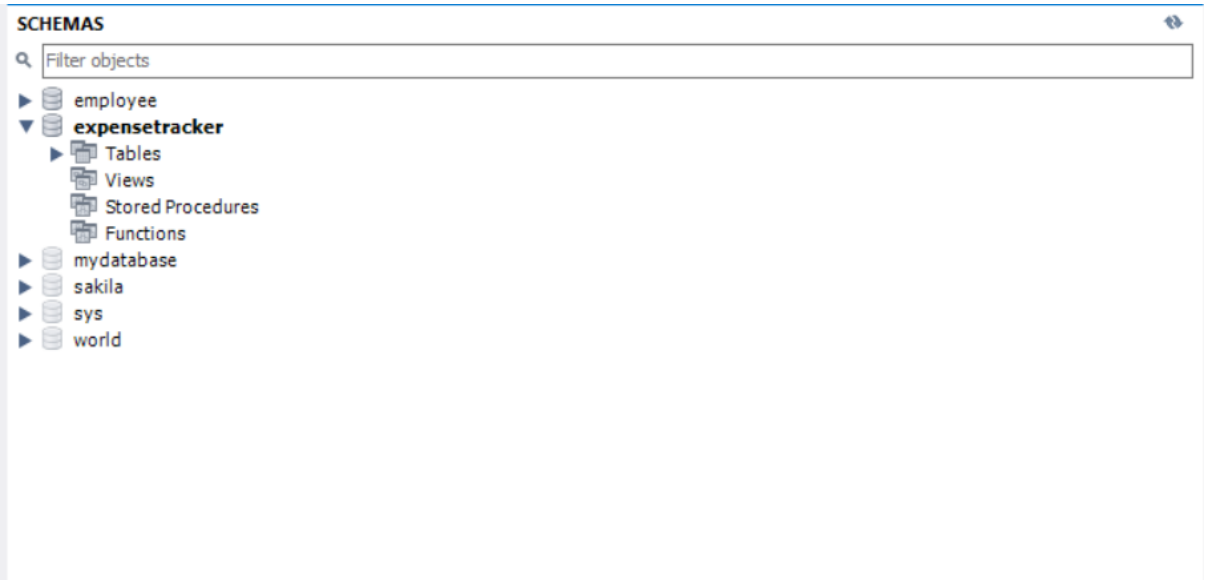
FEATURES OF PROJECT

- 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()
```

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):
    adding = "Insert into Expense (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)


GUI = Tk()
GUI.title("Expense Tracker")
GUI.geometry('700x430')

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

# adding tabs
tab.add(wel, text="Welcome")
tab.add(f1, text=f'{"Expense": ^30s}')
tab.add(f2, text=f'{"Spend Analysis": ^30s}')

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

# background-color
wel.config(bg="salmon")
txt = Label(wel, text="Welcome\n To\n Expense Tracker", font=("Times New
Roman", 36, "bold", "italic"), bg="salmon",
            fg="white")
txt.pack(pady=100)

f1.config(bg="DarkSlateGray1")
f2.config(bg="DarkSlateGray1")

# ----Date------
exp_date = ttk.Label(f1, text='Date:', font=('Times New Roman', 18),
background="DarkSlateGray1")
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='white',
                           font=('Times New Roman', 18))
exp_date_field.grid(row=0, column=1, padx=55, pady=15)

# ----Title------
title = ttk.Label(f1, text='Title:', font=('Times New Roman', 18),
background="DarkSlateGray1")
```

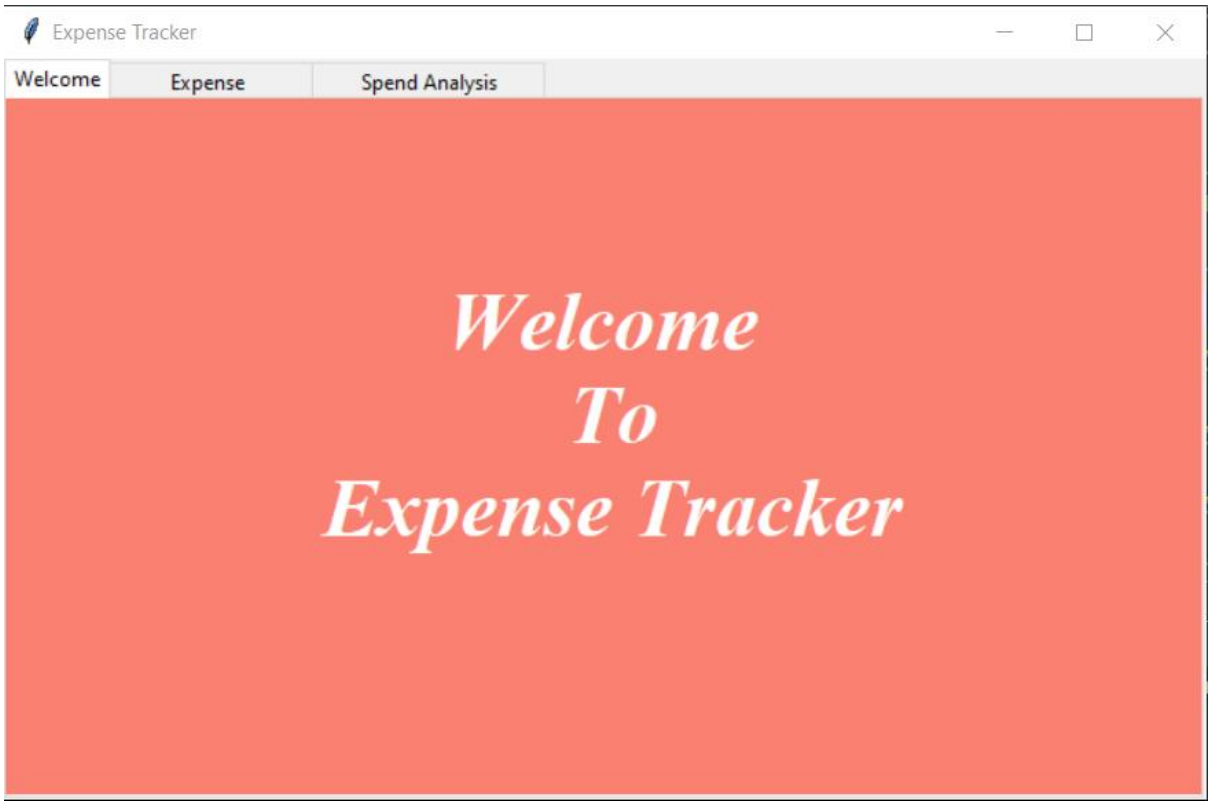
```
title.grid(row=1, column=0, padx=5, pady=15)

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))
title_input.set("Select one")
drop.grid(row=1, column=1, padx=55, pady=15)

# ----Expense------
exp = ttk.Label(f1, text='Expense:', font=('Times New Roman', 18),
background="DarkSlateGray1")
exp.grid(row=2, column=0, padx=55, pady=15)

expense_input = StringVar()

exp_field = ttk.Entry(f1, textvariable=expense_input, font=('Times New
Roman', 18))
exp_field.grid(row=2, column=1, padx=55, pady=15)

# ----Add Button----
bf1Add = ttk.Button(f1, text='Add', command=Addexpense)
bf1Add.grid(row=3, column=1, padx=5, pady=5, ipadx=10, ipady=10)

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
# ----------------------------------------------Spend Analysis--------------
---------------------------------


title = ttk.Label(f2, text='Spend Analysis', font=('Times New Roman', 22),
background="DarkSlateGray1")
title.grid(row=0, column=0, padx=55, pady=15)


def click_weekly():
    mycursor.execute(
        "Select TITLE, sum(Money) TOTAL_EXPENSE from expensetracker.Expense
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 = ttk.Button(f2, text='Weekly', command=click_weekly)
button_weekly.grid(row=2, column=3, padx=25, pady=20, ipadx=10, ipady=10)


def click_monthly():
    mycursor.execute(
        "Select TITLE, sum(Money) TOTAL_EXPENSE from expensetracker.Expense
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 = ttk.Button(f2, text='Monthly', command=click_monthly)
button_monthly.grid(row=3, column=3, padx=25, pady=20, ipadx=10, ipady=10)


def click_yearly():
    mycursor.execute(
        "Select TITLE, sum(Money) TOTAL_EXPENSE from expensetracker.Expense
```

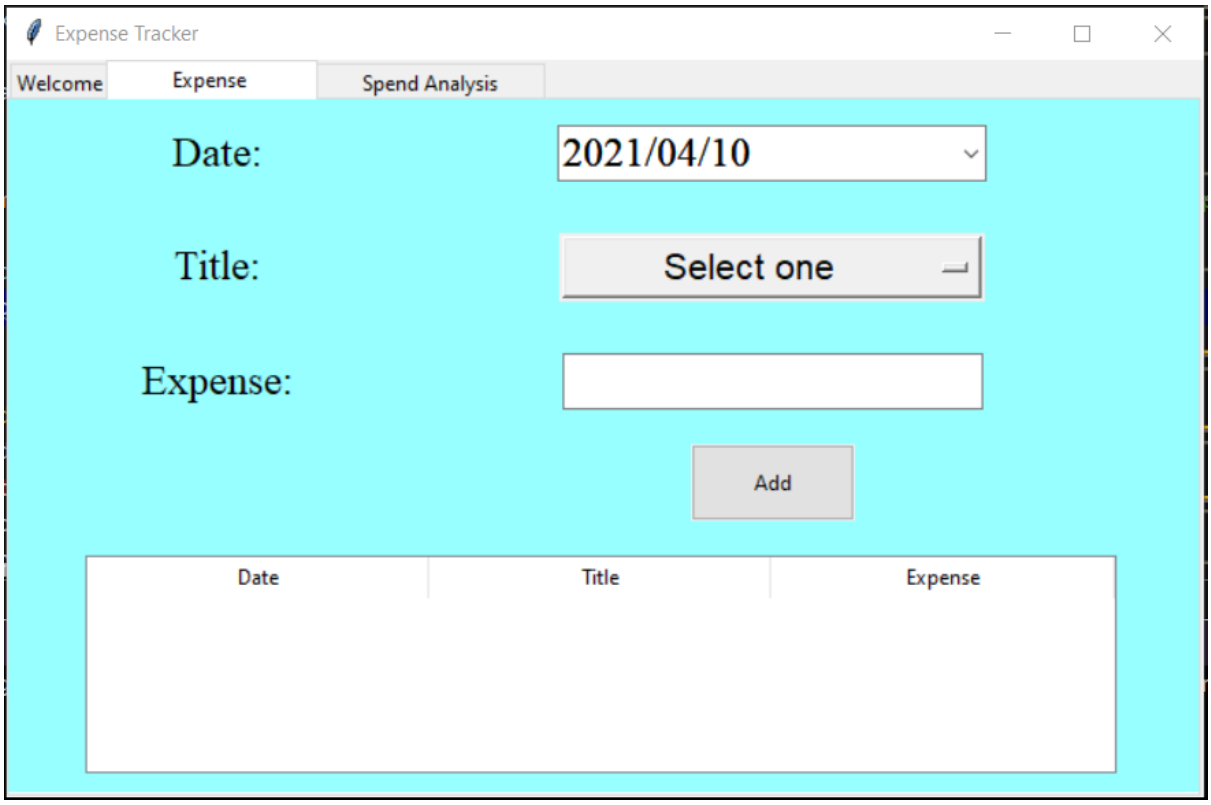
```
where DATE_OF_EXPENSE between curdate() - 365 and curdate() group by
Title");
    myresult = mycursor.fetchall()

    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 = ttk.Button(f2, text='Yearly', command=click_yearly)
button_yearly.grid(row=4, column=3, padx=25, pady=20, ipadx=10, ipady=10)

GUI.mainloop()
```

OUTPUT:

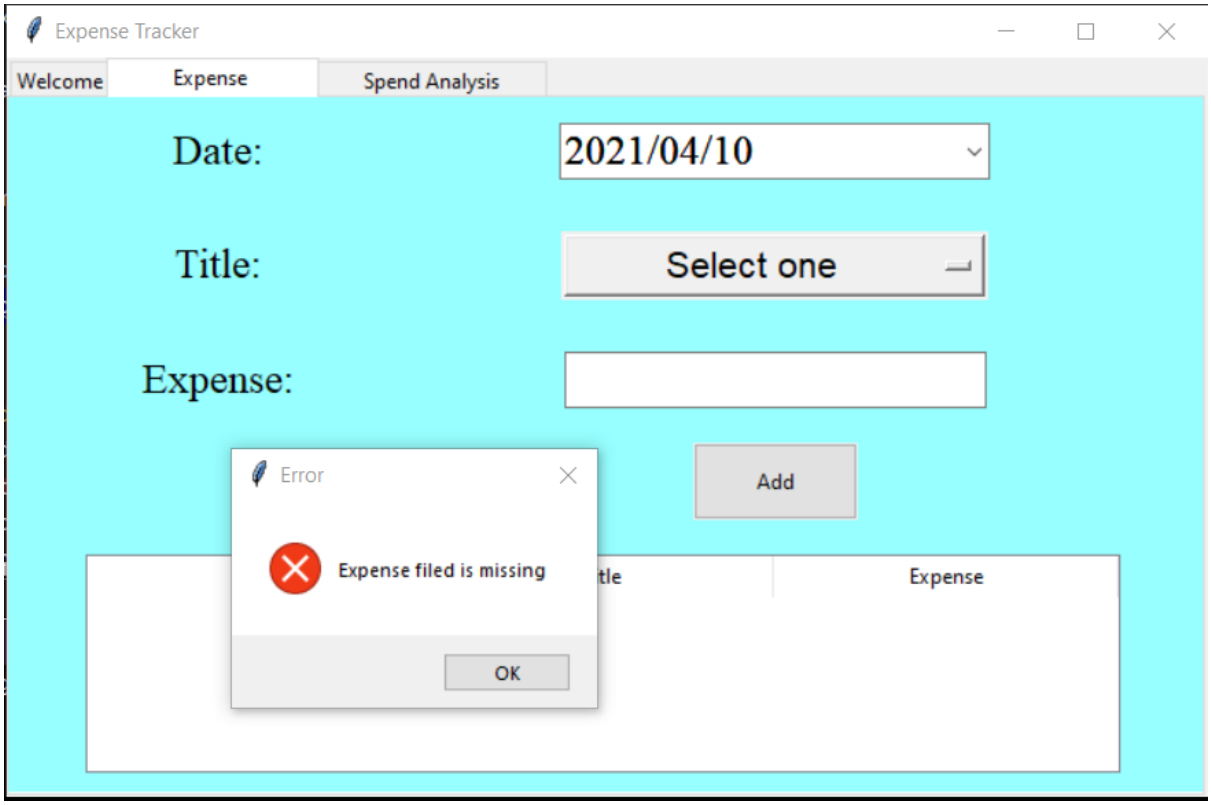
WELCOME SCREEN



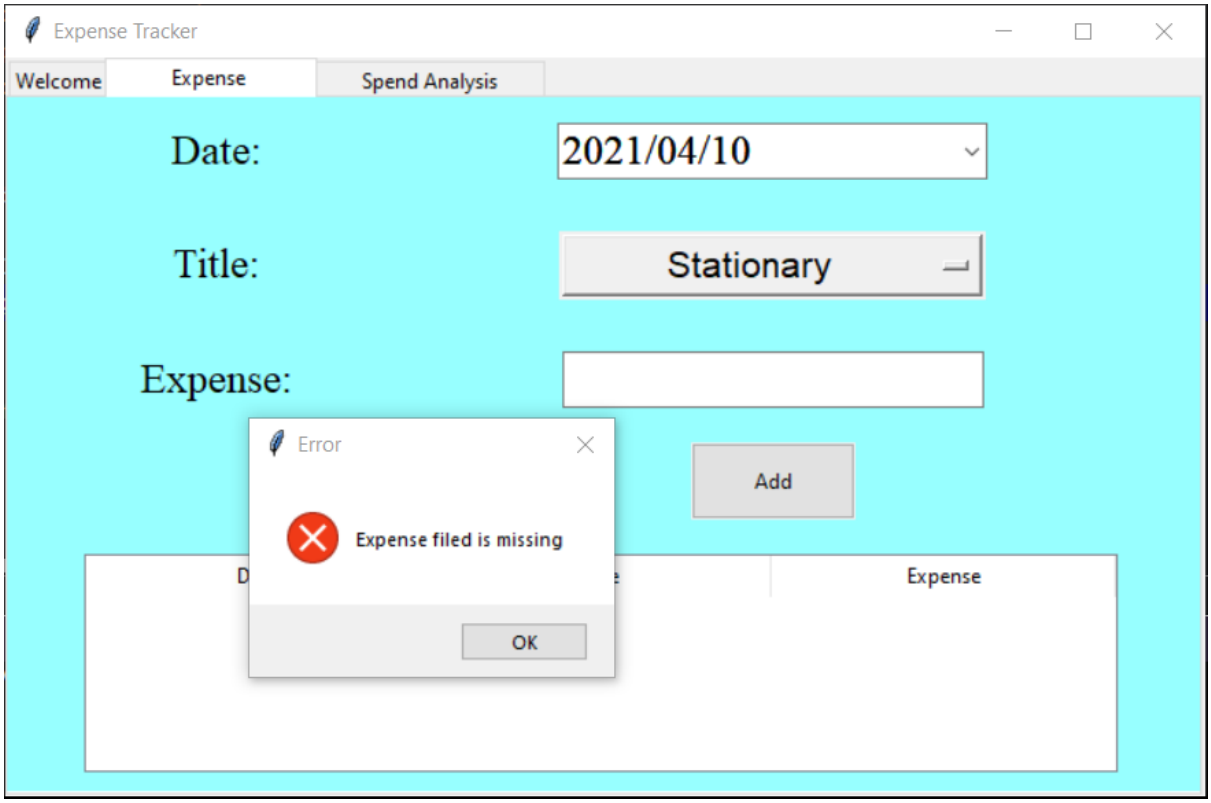
EXPENSE SCREEN



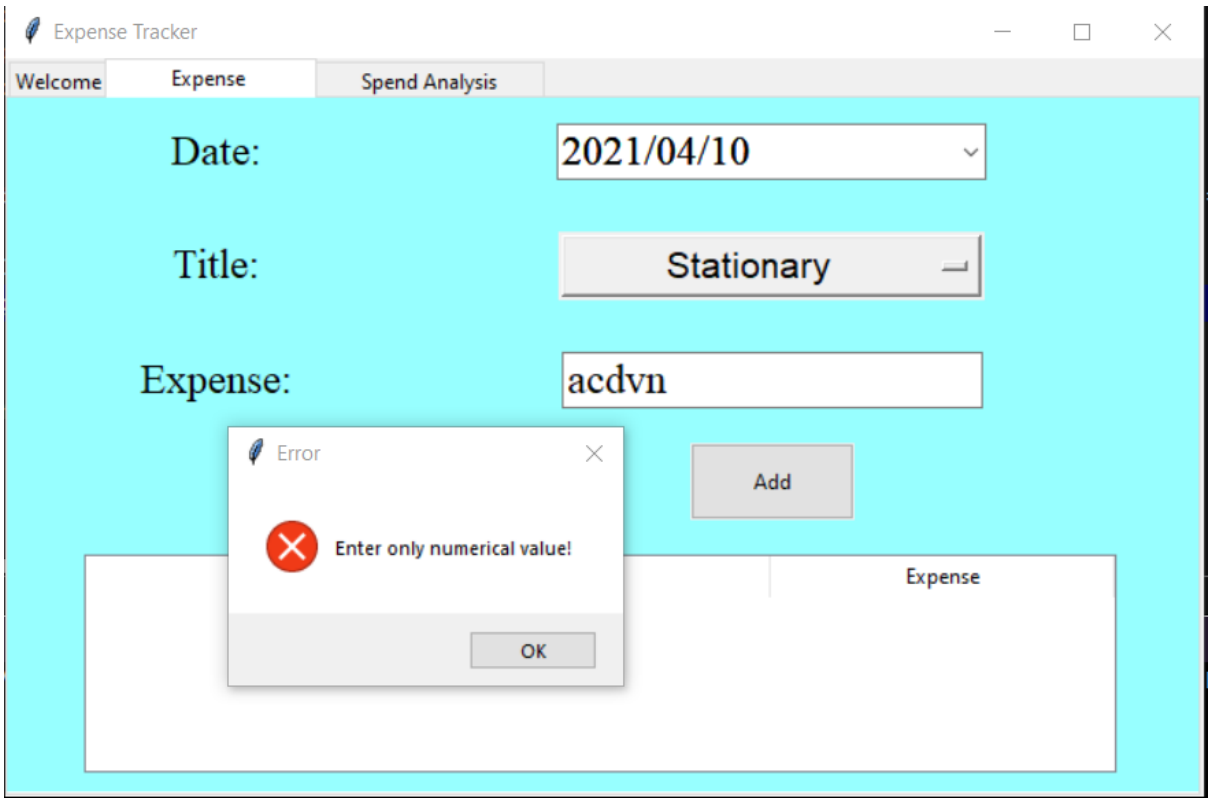
On clicking add when data fields are empty



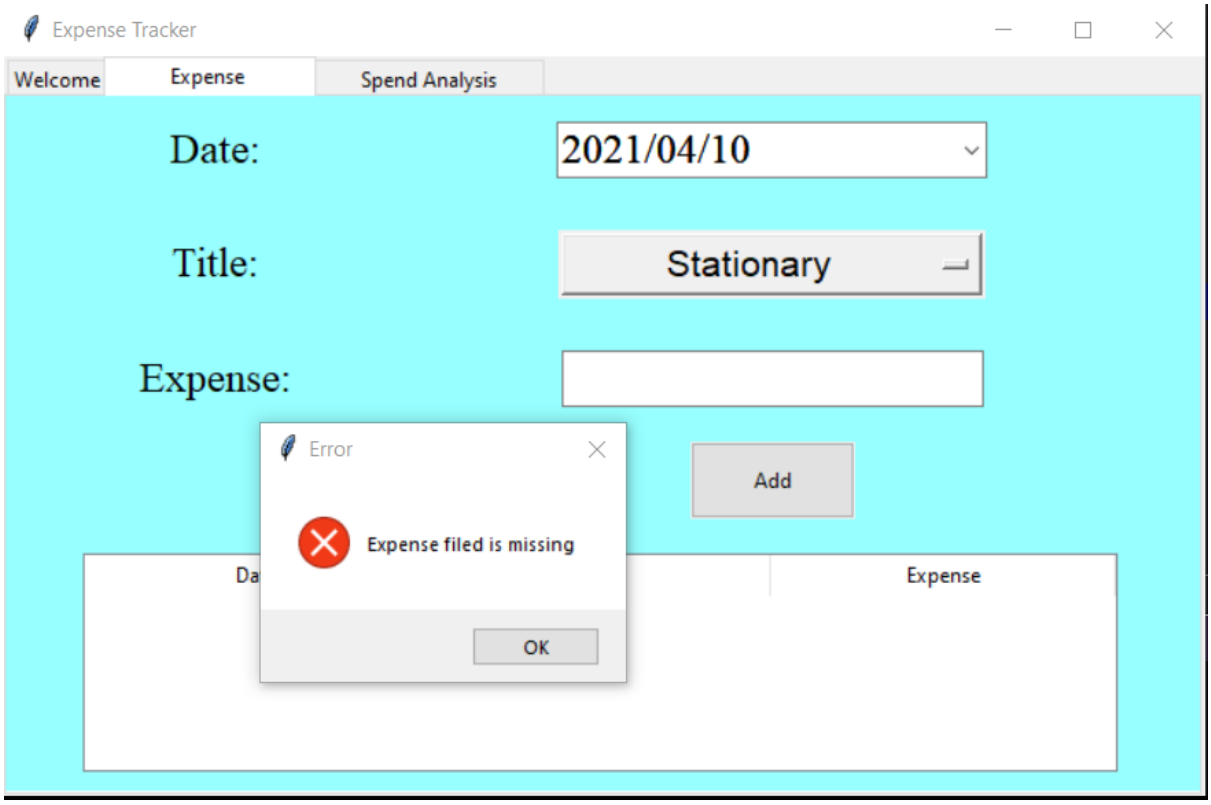
On clicking add when expense field are empty

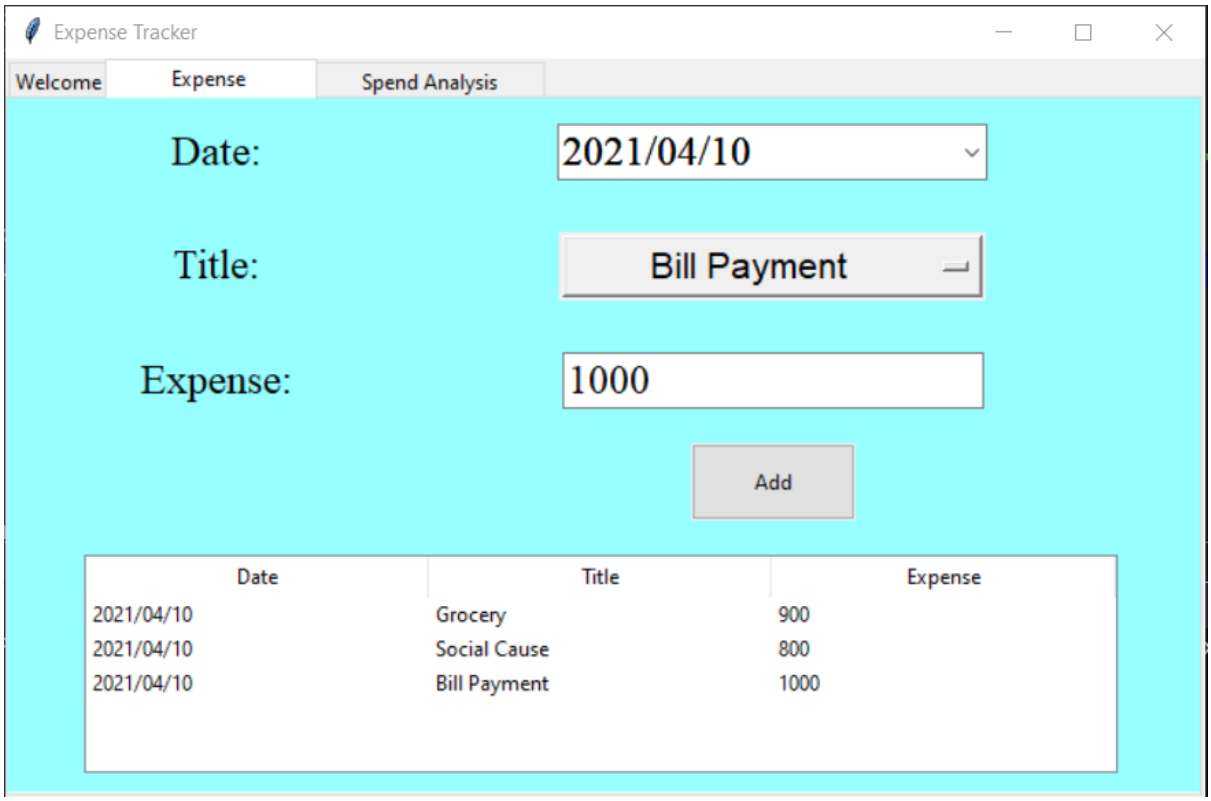


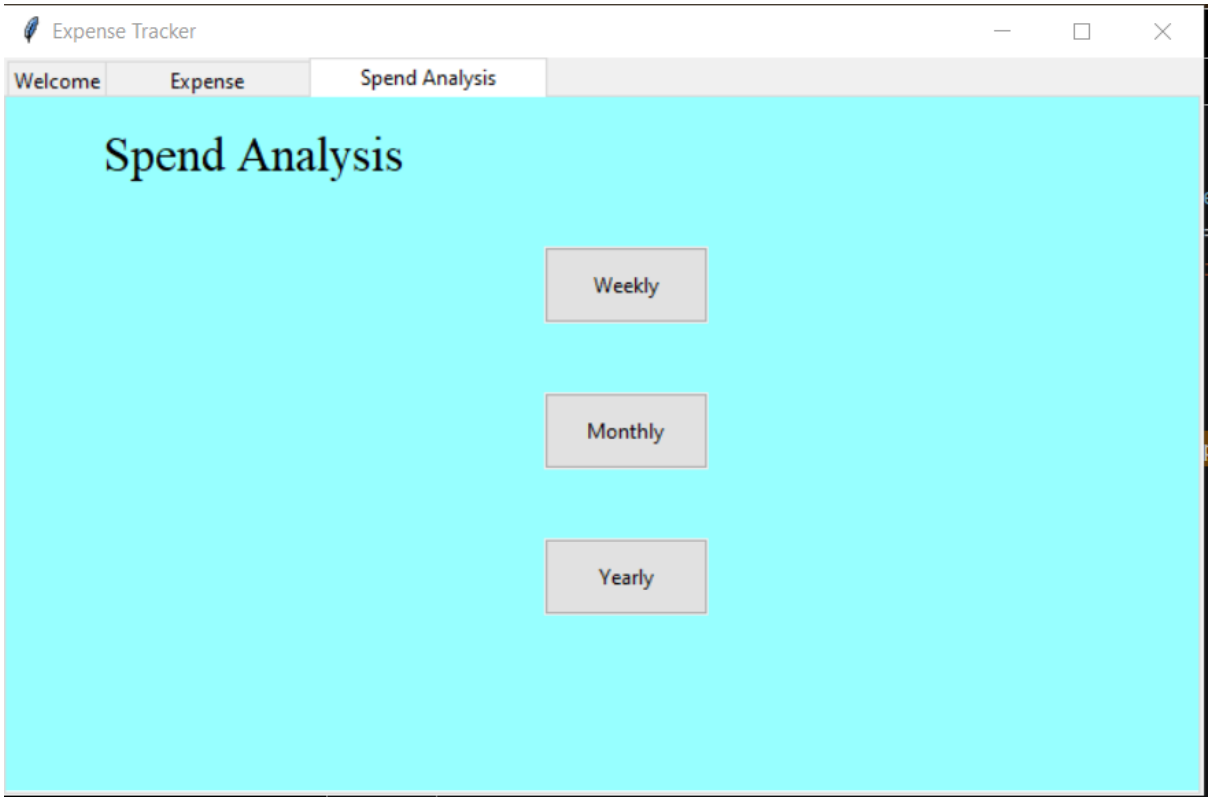
On adding non numerical values



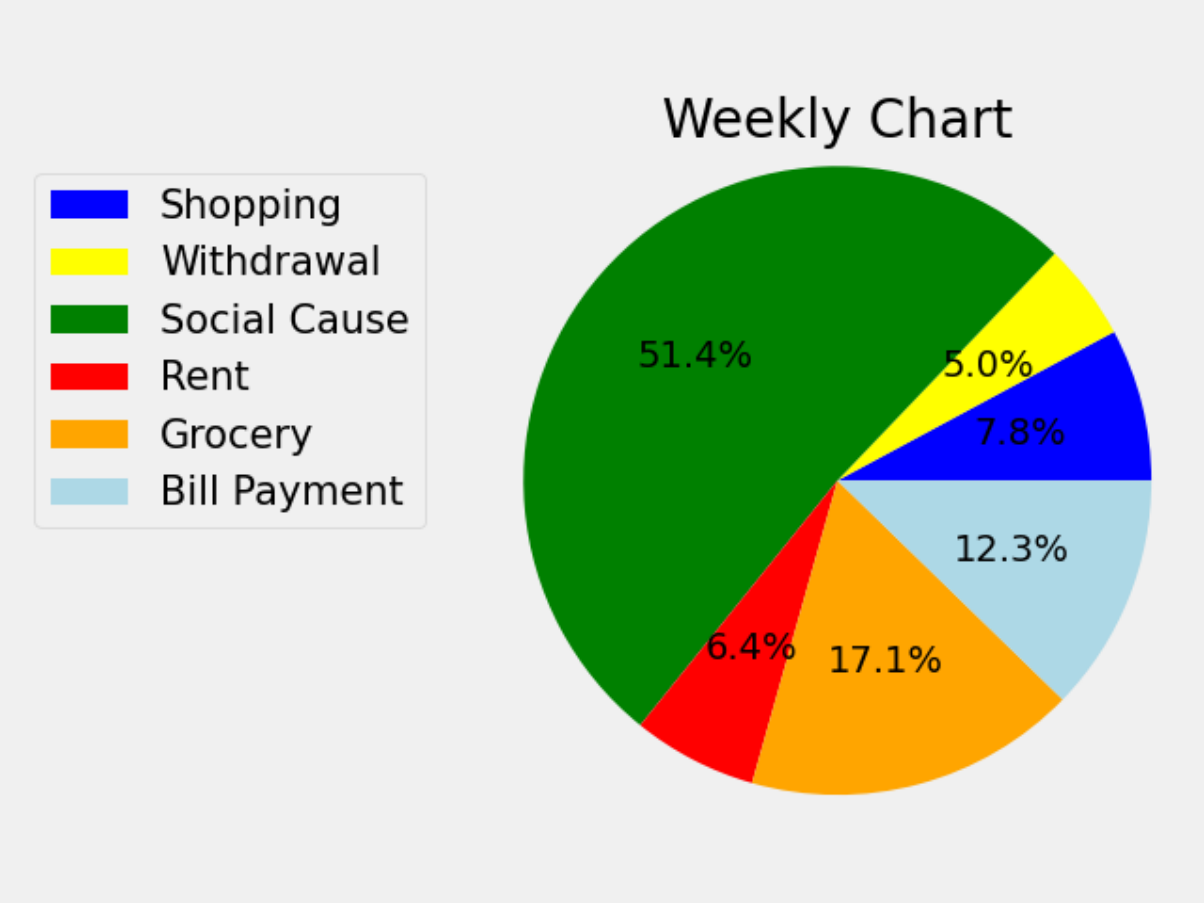
On trying to fill the expense field with space only

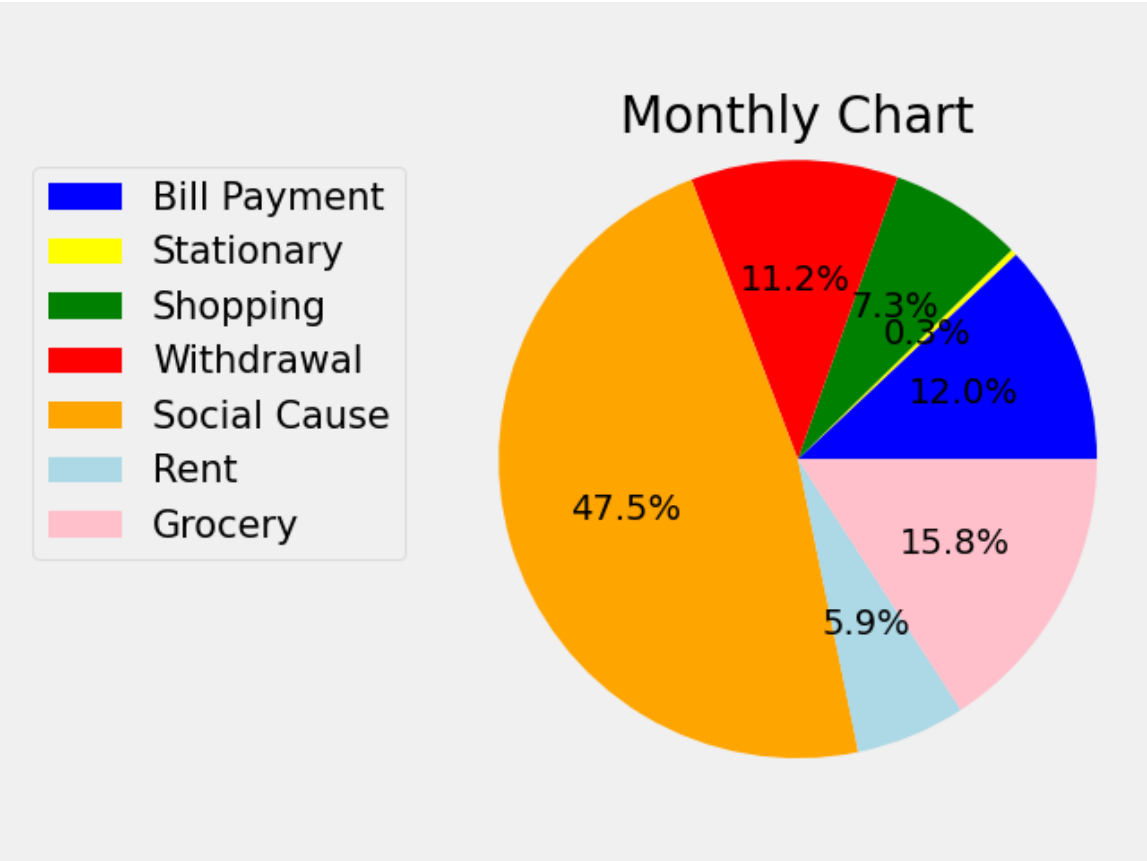


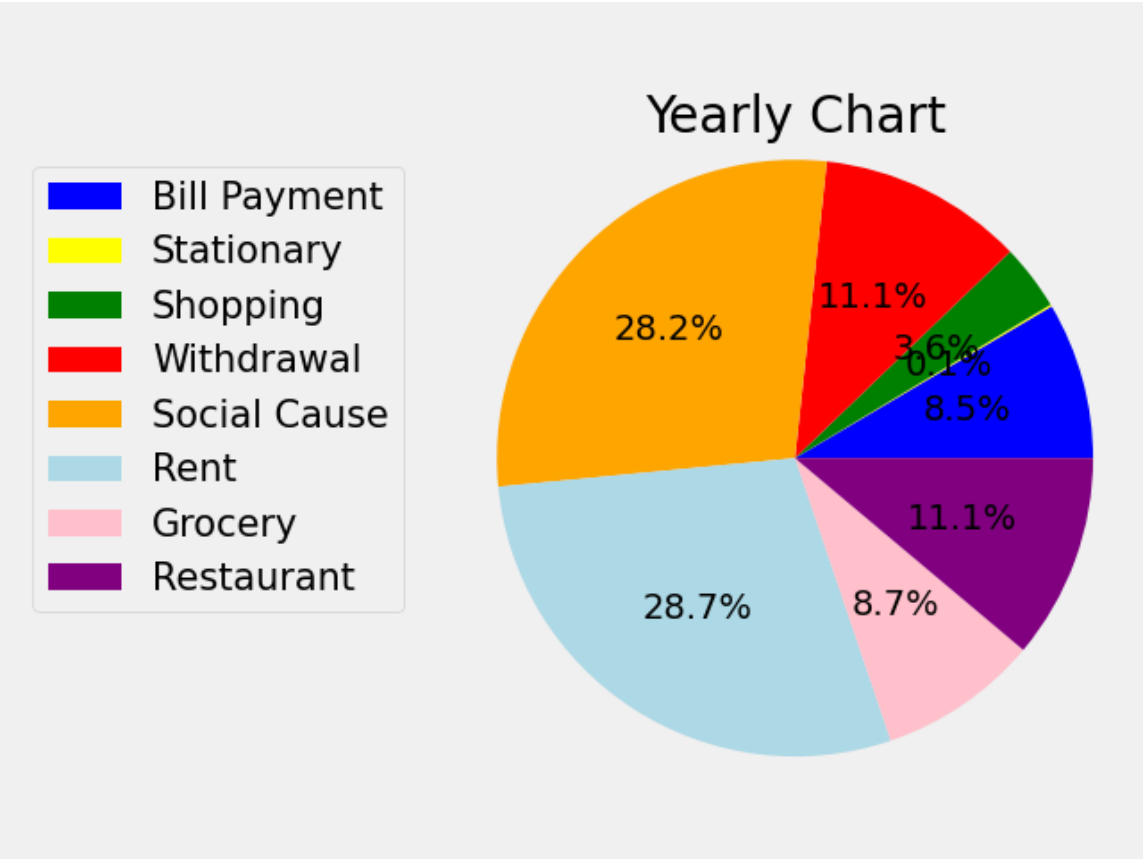




On clicking each button




Monthly Chart
Bill Payment
Stationary
Shopping
Withdrawal
Social Cause
Rent
Grocery
11.2%
7.3%
0.3%
12.0%
47.5%
15.8%
5.9%


Yearly Chart
Bill Payment
Stationary
Shopping
Withdrawal
Social Cause
Rent
Grocery
Restaurant
11.1%
28.2%
3.6%
0.1%
8.5%
11.1%
28.7%
8.7%

```
select * from expensetracker.expense;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DATE_OF_EXPENSE	TITLE	MONEY
2021-04-01	Bill Payment	100
2021-04-02	Stationary	50
2021-04-03	Shopping	600
2021-04-04	Shopping	500
2021-04-07	Withdrawal	700
2021-04-07	Social Cause	900
2021-04-07	Rent	900
2021-04-08	Social Cause	500
2021-04-08	Grocery	900
2021-04-08	Bill Payment	600
2021-04-07	Bill Payment	120
2021-04-06	Grocery	600
2021-03-17	Grocery	100
2021-03-04	Withdrawal	1008
2021-03-02	Social Cause	500
2021-02-23	Bill Payment	1000
2021-02-23	Bill Payment	1000
2021-02-23	Withdrawal	500
2021-02-18	Shopping	500
2021-01-20	Social Cause	5000
2021-01-07	Rent	2000
2021-01-07	Withdrawal	900
2021-01-07	Withdrawal	900
2021-01-09	Grocery	602
2021-01-01	Grocery	800
2021-01-10	Rent	5000

expense 15

```
select * from expensetracker.expense;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DATE_OF_EXPENSE	TITLE	MONEY
2021-04-08	Bill Payment	600
2021-04-07	Bill Payment	120
2021-04-06	Grocery	600
2021-03-17	Grocery	100
2021-03-04	Withdrawal	1008
2021-03-02	Social Cause	500
2021-02-23	Bill Payment	1000
2021-02-23	Bill Payment	1000
2021-02-23	Withdrawal	500
2021-02-18	Shopping	500
2021-01-20	Social Cause	5000
2021-01-07	Rent	2000
2021-01-07	Withdrawal	900
2021-01-07	Withdrawal	900
2021-01-09	Grocery	602
2021-01-01	Grocery	800
2021-01-10	Rent	5000
2021-04-02	Withdrawal	1000
2021-04-10	Grocery	900
2021-04-10	Social Cause	800
2021-04-10	Bill Payment	1000
2021-04-10	Social Cause	5000
2021-01-05	Restaurant	5000
2021-01-05	Rent	5000
2021-04-10	Shopping	1000