Project for a Python Expense Tracker

We will construct an expense tracker in this python django project that will keep track of our spending. When filling out the registration form, a person must also provide information about his or her income and the amount of money he or she want to save. Some individuals work every day, thus their earnings may be contributed on a regular basis. On a weekly, monthly, and annual basis, details of costs will be shown in the form of a pie chart. To get started with the Expense Tracker project, you'll need to install Django.

Prerequisites for the project

Before beginning this Python Expense Tracker project, you must have a solid understanding of the Django framework, HTML, CSS, javascript, and Python.

Structure of the Project Files

1. Create a project and an app using the Django framework 2. Models.py is the third module. 3. Admin.py is the fourth module. 4. Urls.py is a Python script that generates URLs. 5. Views.py is the sixth Python module.

Source Code for Expense Tracker Python Project

import datetime

import sqlite3

from tkcalendar import DateEntry

from tkinter import \*

import tkinter.messagebox as mb

import tkinter.ttk as ttk

# Connecting to the Database

connector = sqlite3.connect("Expense Tracker.db")

cursor = connector.cursor()

connector.execute(

'CREATE TABLE IF NOT EXISTS ExpenseTracker (ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, Date DATETIME, Payee TEXT, Description TEXT, Amount FLOAT, ModeOfPayment TEXT)'

)

connector.commit()

# Backgrounds and Fonts

dataentery\_frame\_bg = 'Red'

buttons\_frame\_bg = 'Tomato'

hlb\_btn\_bg = 'IndianRed'

lbl\_font = ('Georgia', 13)

entry\_font = 'Times 13 bold'

btn\_font = ('Gill Sans MT', 13)

# Initializing the GUI window

root = Tk()

root.title('PythonGeeks Expense Tracker')

root.geometry('1200x550')

root.resizable(0, 0)

Label(root, text='EXPENSE TRACKER', font=('Noto Sans CJK TC', 15, 'bold'), bg=hlb\_btn\_bg).pack(side=TOP, fill=X)

# StringVar and DoubleVar variables

desc = StringVar()

amnt = DoubleVar()

payee = StringVar()

MoP = StringVar(value='Cash')

# Frames

data\_entry\_frame = Frame(root, bg=dataentery\_frame\_bg)

data\_entry\_frame.place(x=0, y=30, relheight=0.95, relwidth=0.25)

buttons\_frame = Frame(root, bg=buttons\_frame\_bg)

buttons\_frame.place(relx=0.25, rely=0.05, relwidth=0.75, relheight=0.21)

tree\_frame = Frame(root)

tree\_frame.place(relx=0.25, rely=0.26, relwidth=0.75, relheight=0.74)

# Data Entry Frame

Label(data\_entry\_frame, text='Date (M/DD/YY) :', font=lbl\_font, bg=dataentery\_frame\_bg).place(x=10, y=50)

date = DateEntry(data\_entry\_frame, date=datetime.datetime.now().date(), font=entry\_font)

date.place(x=160, y=50)

Label(data\_entry\_frame, text='Payee\t :', font=lbl\_font, bg=dataentery\_frame\_bg).place(x=10, y=230)

Entry(data\_entry\_frame, font=entry\_font, width=31, text=payee).place(x=10, y=260)

Label(data\_entry\_frame, text='Description :', font=lbl\_font, bg=dataentery\_frame\_bg).place(x=10, y=100)

Entry(data\_entry\_frame, font=entry\_font, width=31, text=desc).place(x=10, y=130)

Label(data\_entry\_frame, text='Amount\t :', font=lbl\_font, bg=dataentery\_frame\_bg).place(x=10, y=180)

Entry(data\_entry\_frame, font=entry\_font, width=14, text=amnt).place(x=160, y=180)

Label(data\_entry\_frame, text='Mode of Payment:', font=lbl\_font, bg=dataentery\_frame\_bg).place(x=10, y=310)

dd1 = OptionMenu(data\_entry\_frame, MoP, \*['Cash', 'Cheque', 'Credit Card', 'Debit Card', 'Paytm', 'Google Pay', 'Razorpay'])

dd1.place(x=160, y=305) ; dd1.configure(width=10, font=entry\_font)

Button(data\_entry\_frame, text='Add expense', command=add\_another\_expense, font=btn\_font, width=30,

bg=hlb\_btn\_bg).place(x=10, y=395)

Button(data\_entry\_frame, text='Convert to words before adding', font=btn\_font, width=30, bg=hlb\_btn\_bg).place(x=10,y=450)

# Buttons' Frame

Button(buttons\_frame, text='Delete Expense', font=btn\_font, width=25, bg=hlb\_btn\_bg, command=remove\_expense).place(x=30, y=5)

Button(buttons\_frame, text='Clear Fields in DataEntry Frame', font=btn\_font, width=25, bg=hlb\_btn\_bg,

command=clear\_fields).place(x=335, y=5)

Button(buttons\_frame, text='Delete All Expenses', font=btn\_font, width=25, bg=hlb\_btn\_bg, command=remove\_all\_expenses).place(x=640, y=5)

Button(buttons\_frame, text='View Selected Expense\'s Details', font=btn\_font, width=25, bg=hlb\_btn\_bg,

command=view\_expense\_details).place(x=30, y=65)

Button(buttons\_frame, text='Edit Selected Expense', command=edit\_expense, font=btn\_font, width=25, bg=hlb\_btn\_bg).place(x=335,y=65)

Button(buttons\_frame, text='Convert Expense to a sentence', font=btn\_font, width=25, bg=hlb\_btn\_bg,

command=selected\_expense\_to\_words).place(x=640, y=65)

# Treeview Frame

table = ttk.Treeview(tree\_frame, selectmode=BROWSE, columns=('ID', 'Date', 'Payee', 'Description', 'Amount', 'Mode of Payment'))

X\_Scroller = Scrollbar(table, orient=HORIZONTAL, command=table.xview)

Y\_Scroller = Scrollbar(table, orient=VERTICAL, command=table.yview)

X\_Scroller.pack(side=BOTTOM, fill=X)

Y\_Scroller.pack(side=RIGHT, fill=Y)

table.config(yscrollcommand=Y\_Scroller.set, xscrollcommand=X\_Scroller.set)

table.heading('ID', text='S No.', anchor=CENTER)

table.heading('Date', text='Date', anchor=CENTER)

table.heading('Payee', text='Payee', anchor=CENTER)

table.heading('Description', text='Description', anchor=CENTER)

table.heading('Amount', text='Amount', anchor=CENTER)

table.heading('Mode of Payment', text='Mode of Payment', anchor=CENTER)

table.column('#0', width=0, stretch=NO)

table.column('#1', width=50, stretch=NO)

table.column('#2', width=95, stretch=NO) # Date column

table.column('#3', width=150, stretch=NO) # Payee column

table.column('#4', width=325, stretch=NO) # Title column

table.column('#5', width=135, stretch=NO) # Amount column

table.column('#6', width=125, stretch=NO) # Mode of Payment column

table.place(relx=0, y=0, relheight=1, relwidth=1)

list\_all\_expenses()

# Finalizing the GUI window

root.update()

root.mainloop()

# Functions

def list\_all\_expenses():

global connector, table

table.delete(\*table.get\_children())

all\_data = connector.execute('SELECT \* FROM ExpenseTracker')

data = all\_data.fetchall()

for values in data:

table.insert('', END, values=values)

def view\_expense\_details():

global table

global date, payee, desc, amnt, MoP

if not table.selection():

mb.showerror('No expense selected', 'Please select an expense from the table to view its details')

current\_selected\_expense = table.item(table.focus())

values = current\_selected\_expense['values']

expenditure\_date = datetime.date(int(values[1][:4]), int(values[1][5:7]), int(values[1][8:]))

date.set\_date(expenditure\_date) ; payee.set(values[2]) ; desc.set(values[3]) ; amnt.set(values[4]) ; MoP.set(values[5])

def clear\_fields():

global desc, payee, amnt, MoP, date, table

today\_date = datetime.datetime.now().date()

desc.set('') ; payee.set('') ; amnt.set(0.0) ; MoP.set('Cash'), date.set\_date(today\_date)

table.selection\_remove(\*table.selection())

def remove\_expense():

if not table.selection():

mb.showerror('No record selected!', 'Please select a record to delete!')

return

current\_selected\_expense = table.item(table.focus())

values\_selected = current\_selected\_expense['values']

surety = mb.askyesno('Are you sure?', f'Are you sure that you want to delete the record of {values\_selected[2]}')

if surety:

connector.execute('DELETE FROM ExpenseTracker WHERE ID=%d' % values\_selected[0])

connector.commit()

list\_all\_expenses()

mb.showinfo('Record deleted successfully!', 'The record you wanted to delete has been deleted successfully')

def remove\_all\_expenses():

surety = mb.askyesno('Are you sure?', 'Are you sure that you want to delete all the expense items from the database?', icon='warning')

if surety:

table.delete(\*table.get\_children())

connector.execute('DELETE FROM ExpenseTracker')

connector.commit()

clear\_fields()

list\_all\_expenses()

mb.showinfo('All Expenses deleted', 'All the expenses were successfully deleted')

else:

mb.showinfo('Ok then', 'The task was aborted and no expense was deleted!')

def add\_another\_expense():

global date, payee, desc, amnt, MoP

global connector

if not date.get() or not payee.get() or not desc.get() or not amnt.get() or not MoP.get():

mb.showerror('Fields empty!', "Please fill all the missing fields before pressing the add button!")

else:

connector.execute(

'INSERT INTO ExpenseTracker (Date, Payee, Description, Amount, ModeOfPayment) VALUES (?, ?, ?, ?, ?)',

(date.get\_date(), payee.get(), desc.get(), amnt.get(), MoP.get())

)

connector.commit()

clear\_fields()

list\_all\_expenses()

mb.showinfo('Expense added', 'The expense whose details you just entered has been added to the database')

def edit\_expense():

global table

def edit\_existing\_expense():

global date, amnt, desc, payee, MoP

global connector, table

current\_selected\_expense = table.item(table.focus())

contents = current\_selected\_expense['values']

connector.execute('UPDATE ExpenseTracker SET Date = ?, Payee = ?, Description = ?, Amount = ?, ModeOfPayment = ? WHERE ID = ?',

(date.get\_date(), payee.get(), desc.get(), amnt.get(), MoP.get(), contents[0]))

connector.commit()

clear\_fields()

list\_all\_expenses()

mb.showinfo('Data edited', 'We have updated the data and stored in the database as you wanted')

edit\_btn.destroy()

return

if not table.selection():

mb.showerror('No expense selected!', 'You have not selected any expense in the table for us to edit; please do that!')

return

view\_expense\_details()

edit\_btn = Button(data\_entry\_frame, text='Edit expense', font=btn\_font, width=30,

bg=hlb\_btn\_bg, command=edit\_existing\_expense)

edit\_btn.place(x=10, y=395)

def selected\_expense\_to\_words():

global table

if not table.selection():

mb.showerror('No expense selected!', 'Please select an expense from the table for us to read')

return

current\_selected\_expense = table.item(table.focus())

values = current\_selected\_expense['values']

message = f'Your expense can be read like: \n"You paid {values[4]} to {values[2]} for {values[3]} on {values[1]} via {values[5]}"'

mb.showinfo('Here\'s how to read your expense', message)

def expense\_to\_words\_before\_adding():

global date, desc, amnt, payee, MoP

if not date or not desc or not amnt or not payee or not MoP:

mb.showerror('Incomplete data', 'The data is incomplete, meaning fill all the fields first!')

message = f'Your expense can be read like: \n"You paid {amnt.get()} to {payee.get()} for {desc.get()} on {date.get\_date()} via {MoP.get()}"'

add\_question = mb.askyesno('Read your record like: ', f'{message}\n\nShould I add it to the database?')

if add\_question:

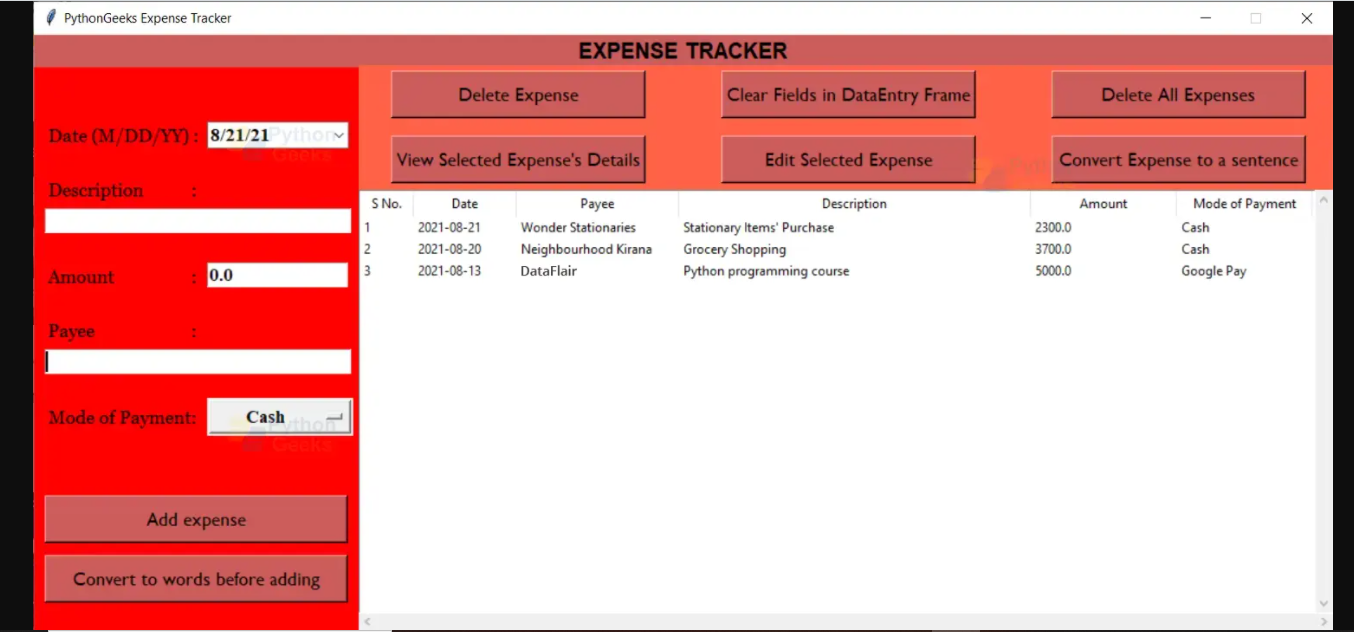
add\_another\_expense()

else:

mb.showinfo('Ok', 'Please take your time to add this record')

Copy

Output



Summary

The expenditure tracker project in Python was completed successfully. While working on this project, we learned a number of topics.