Universitatea Națională de Știință și Tehnologie Politehnica din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Expense Tracker

Proiect Python

Nume student:
Trifu Mihai

2023

1. Cerința

Tema aleasă pentru proiectul Python are ca scop categorisirea cheltuielilor, în vederea monitorizării bugetului personal.

2. Tehnologii utilizate

În proiectul de față s-au utilizat concepte de programare orientată pe obiect. Fiind construit în interfața PyCharm, cu ajutorul bibliotecilor externe, programul are ca scop monitorizarea cheltuielilor utilizatorului, facilitând administrarea eficientă a bugetului personal.

3. Descrierea aplicației

3.1. Interfața

Interfața este simplă, sugestivă temei alese și, așadar, prietenoasă cu utilizatorul.

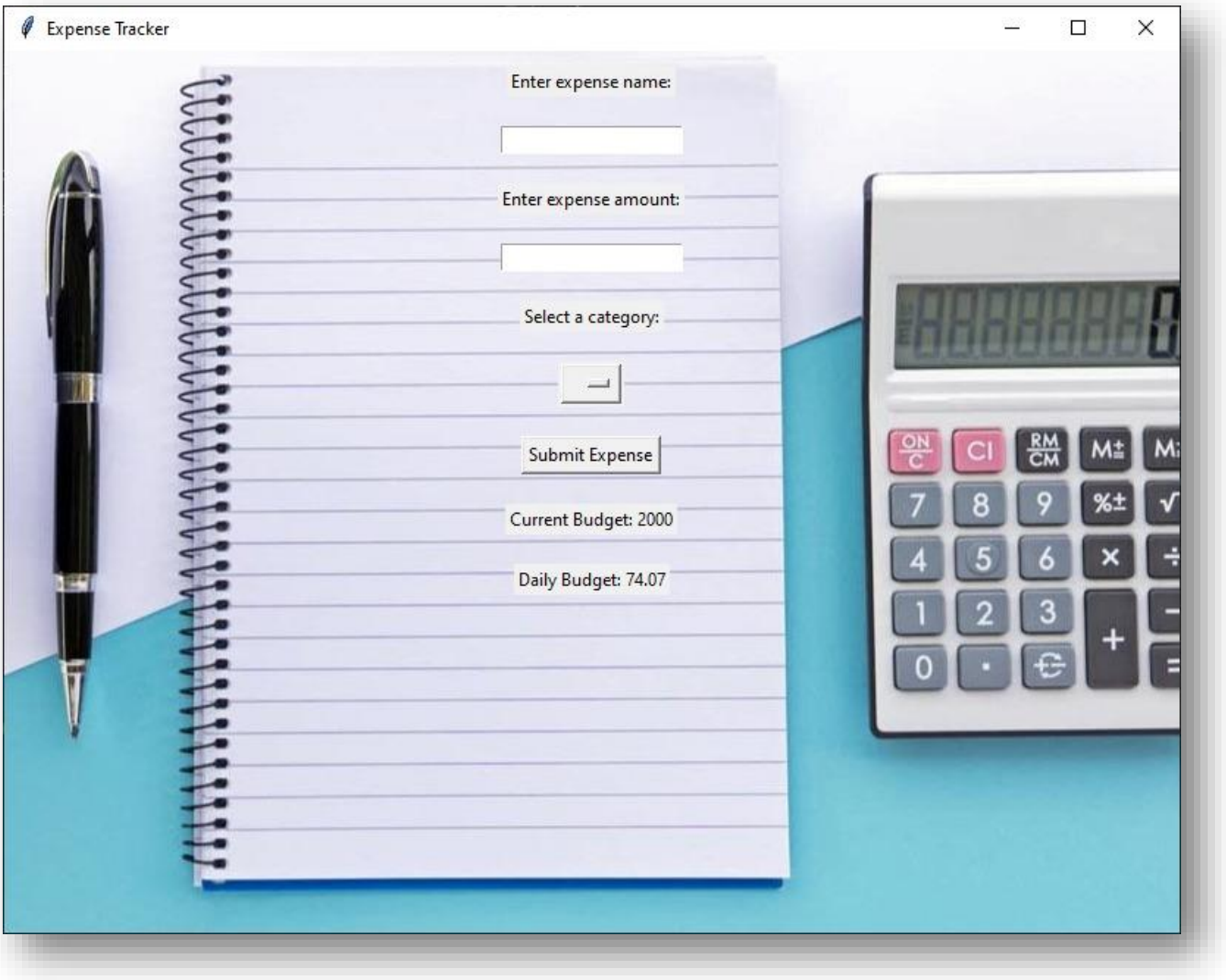

Figura 3.1.1. Pagina principală (introducere cheltuieli)

Pagina principală a interfeței (figura 3.1.1.), permite introducerea de date specifice cheltuielilor utilizatorului: numele produsului achiziționat, suma platită, precum și selectarea categoriei din care aceasta face parte (*mâncare, cheltuieli pentru locuință/ locul de muncă* sau *divertisment*).

3.2. Arhitectură Sistem

1. **Interfața Utilizator (UI):**

- Utilizează biblioteca tkinter pentru crearea interfeței grafice.
- Include etichete (Label), câmpuri de intrare (Entry), meniuri dropdown (OptionMenu), butoane (Button) pentru a interacționa cu utilizatorul.
- Pachetul PIL (Python Imaging Library), care acum se numește Pillow, importă modulele Image și ImageTk.
- Image: Acest modul oferă o clasă cu același nume care este folosită pentru a reprezenta o imagine PIL. Modulul oferă, de asemenea, multe funcții pentru manipularea imaginilor. ImageTk: Acest modul conține suport pentru a crea și modifica obiecte Tkinter BitmapImage și PhotoImage din imagini PIL. Este adesea folosit atunci când doriți să afișați o imagine într-o interfață grafică Tkinter.
- Modulul OS din Python oferă funcții pentru interacțiunea cu sistemul de operare, cum ar fi crearea de fișiere și directoare, gestionarea fișierelor și directoarelor, intrarea, ieșirea, variabilele de mediu, gestionarea proceselor etc.
- Modulul datetime este un modul încorporat în Python care oferă clase pentru manipularea datelor și orelor. Modulul acceptă operaţii aritmetice ale datei și orei, extragerea atributelor, formatarea și oferă informații despre fusul orar.
- Python are un modul Calendar încorporat pentru a lucra cu sarcini legate de dată. Folosind acest modul, putem afișa o anumită lună, precum și întregul calendar al unui an.

```
# Import the necessary libraries
import tkinter as tk  # GUI library
from tkinter import messagebox  # For displaying message boxes
import pandas as pd  # For handling CSV files
import os  # For checking if a file exists
import datetime  # For getting the current date
import calendar  # For getting the number of days in a month
from PIL import Image, ImageTk  # For handling images
```

Figura 3.2.1. Biblioteci

2. **Model de Date:**

- Definește **clasa Expense** pentru a reprezenta obiectele de cheltuială cu atributele: name, amount, și category.

```
# Define the Expense class
class Expense:
    def __init__(self, name, category, amount):  # Initialize an Expense object
        self.name = name  # The name of the expense
        self.category = category  # The category of the expense
        self.amount = amount  # The amount of the expense
```

Figura 3.2.2. Clasa Expense

- Utilizează un fișier CSV (expenses.csv) pentru a stoca datele despre cheltuieli.

3. **Gestionarea Cheltuielilor:**

- Metoda **submit_expense** colectează datele introduse de utilizator și creează un obiect Expense, pe care apoi îl salvează în fișierul CSV.

```
# Submit an expense
def submit_expense(self):
    expense_name = self.name_entry.get()  # Get the name of the expense from the entry
    expense_amount = float(self.amount_entry.get())  # Get the amount of the expense from the entry
    selected_category = self.category_var.get()  # Get the selected category from the dropdown
    new_expense = Expense(name=expense_name, category=selected_category, amount=expense_amount)  # Create a new Expense object
    self.save_expense_to_file(new_expense, self.expense_file_path)  # Save the expense to the file
    self.update_budget()  # Update the budget
    self.update_expense_summary()  # Update the expense summary
    messagebox.showinfo( title: "Success",  message: "Expense saved successfully!")  # Show a success message
```

Figura 3.2.3. Metoda submit_expense

- Metoda **save_expense_to_file** se ocupă de scrierea datelor cheltuielilor în fișierul CSV.

```
# Save an expense to the file
def save_expense_to_file(self, expense: Expense, expense_file_path):
    with open(expense_file_path, "a") as f:  # Open the expense file in append mode
        f.write(f"{expense.name},{expense.amount},{expense.category}\n")  # Write the expense to the file
```

Figura 3.2.4 Metoda save_expense_to_file

4. **Calcularea Bugetului:**

   - Metoda **calculate_initial_budget** calculează bugetul inițial scăzând totalul cheltuielilor dintr-o valoare fixă (în exemplu, 2000).

```
# Calculate the initial budget
def calculate_initial_budget(self):
    try:
        df = pd.read_csv(self.expense_file_path, names=['name', 'amount', 'category'])  # Read the expense file
        total_expense = df['amount'].sum()  # Calculate the total expense
    except pd.errors.EmptyDataError:  # Handle the case when the CSV file is empty
        total_expense = 0
    return 2000 - total_expense  # Return the initial budget
```

Figura 3.2.5. Metoda calculate_initial_budget

   - Metoda **calculate_daily_budget** calculează bugetul zilnic în funcție de bugetul total și zilele rămase în luna curentă.

```
# Calculate the daily budget
def calculate_daily_budget(self):
    now = datetime.datetime.now()  # Get the current date
    days_in_month = calendar.monthrange(now.year, now.month)[1]  # Get the number of days in the current month
    remaining_days = days_in_month - now.day  # Calculate the remaining days in the month
    daily_budget = self.budget / remaining_days if remaining_days > 0 else 0  # Calculate the daily budget
    return round(daily_budget, 2)  # Return the daily budget
```

Figura 3.2.6. Metoda calculate_daily_budget

5. **Actualizare Automată:**

   - La fiecare cheltuială nouă, se actualizează bugetul, eticheta bugetului, eticheta bugetului zilnic și rezumatul cheltuielilor.

6. **Afisarea Informațiilor:**

   - Afisează informații în etichete pentru a prezenta utilizatorului starea actuală a bugetului, bugetul zilnic și rezumatul cheltuielilor.

7. **Manipularea Datelor cu *Pandas*:**

   - Utilizează biblioteca *pandas* pentru manipularea datelor în format tabular, citind și scriind date în fișierul CSV și efectuând operații de grupare pentru rezumatul cheltuielilor.

8. **Controlul Fluxului Principal:**

   - Utilizează o structură **if __name__ == "__main__":** pentru a asigura că programul rulează doar când este rulat ca fișier principal.

   - Inițializează o fereastră principală **(tk.Tk())**, configurează dimensiunile și crează o instanță a aplicației *ExpenseTrackerApp*.

3.3. Prezentare funcționalități

In aceasta pagina, utilizatorul are posibilitatea de a tine evidenta noilor date introduse, afisandu-se totodata bugetul curent ramas, dar si bugetul zilnic, estimativ, ce ar putea folosit de utilizator, in restul zilelor pana la finalul lunii.

1. **Introducerea Cheltuielilor:**
    - Utilizatorul poate introduce următoarele detalii despre o cheltuială:
        - Numele cheltuielii (**Enter expense name:**).
        - Suma cheltuită (**Enter expense amount:**).
        - Categoria cheltuielii (**Select a category:** dintr-un meniu dropdown cu opțiunile: "Food", "Home", "Work", "Fun").

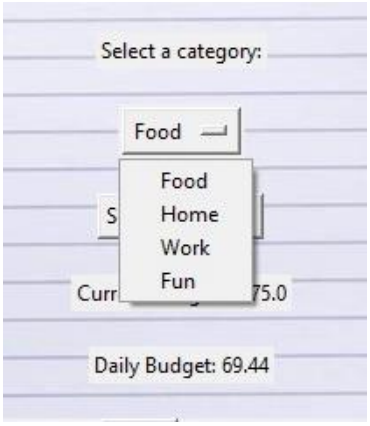

Figura 3.3.1. Meniu de selectare a categoriei

        - Odată intoduse informațiile, ele sunt salvate in aplicație cu ajutorul butonului *submit.*
        - În cazul în care utilizatorul dorește să șteargă o anumită achiziție, el poate folosi butonul *delete* din dreptul fiecarei cheltuieli salvate în baza de date.

2. **Calcularea și Afișarea Bugetului:**
    - Programul calculează și afișează bugetul curent pe baza totalului cheltuielilor efectuate și a unui buget inițial prestabilit.
    - Afișează și bugetul zilnic estimat în funcție de bugetul curent și zilele rămase în luna curentă.

3. **Afisarea Listei Cheltuielilor:**
    - Oferă o listă a produselor/serviciilor achiziționate, în ordinea introducerii.
4. **Salvarea Cheltuielilor:**
    - Cheltuielile introduse de utilizator sunt salvate într-un fișier CSV (**expenses.csv**) pentru a păstra o evidență a acestora.

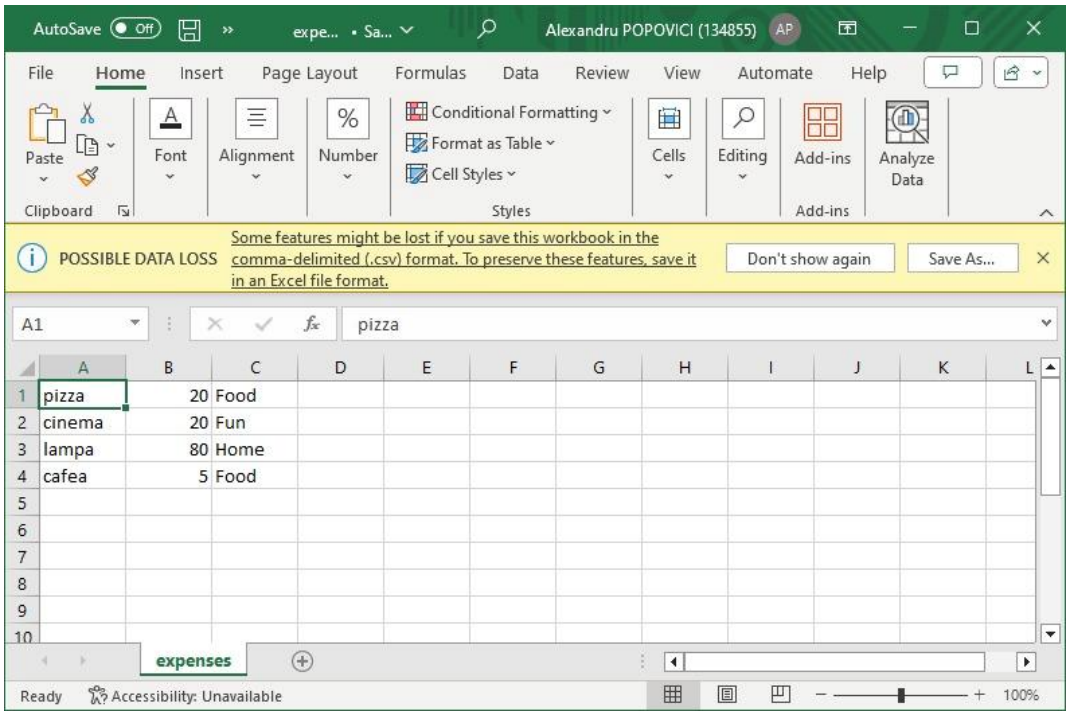

Figura 3.3.2. CSV file

5. **Actualizare Automată a Informațiilor:**

- După ce o cheltuială este înregistrată, programul actualizează automat informațiile afișate privind bugetul.

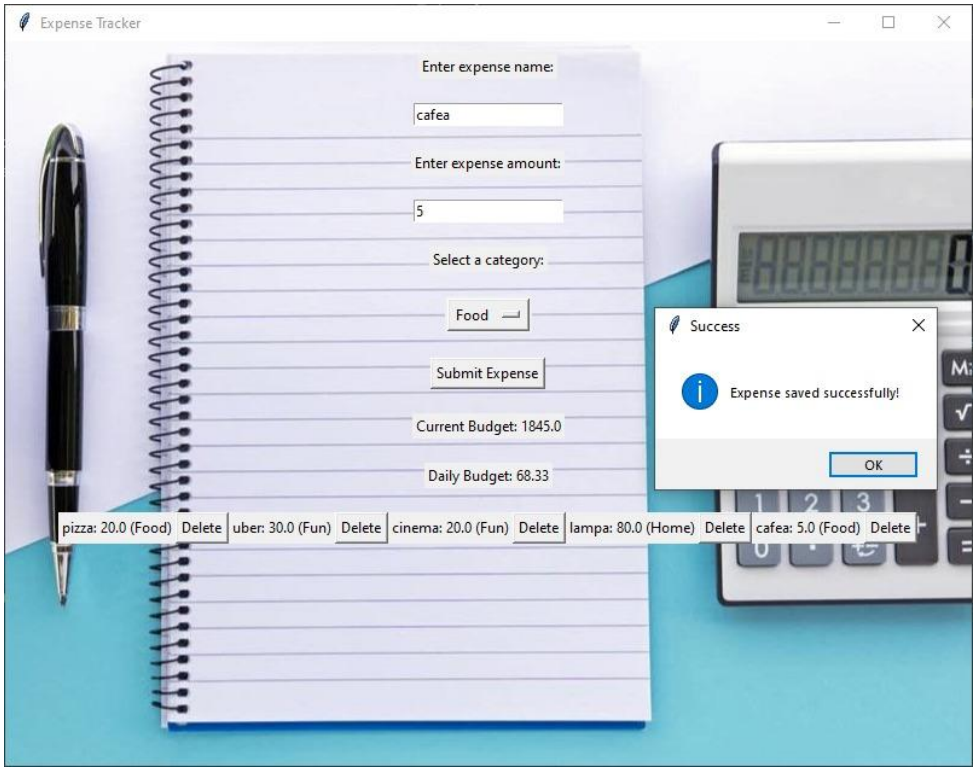

Figura 3.3.3. Mesaj de confirmare a adăugării cheltuielii

- În cazul in care utilizatorul dorește eliminarea unei cheltuieli, se poate folosi butonul *delete* pentru ștergerea articolului din baza de date; scoaterea din listă a unui item se face doar după confirmarea acțiunii de către utilizator (printr-un mesaj de tip pop-up).

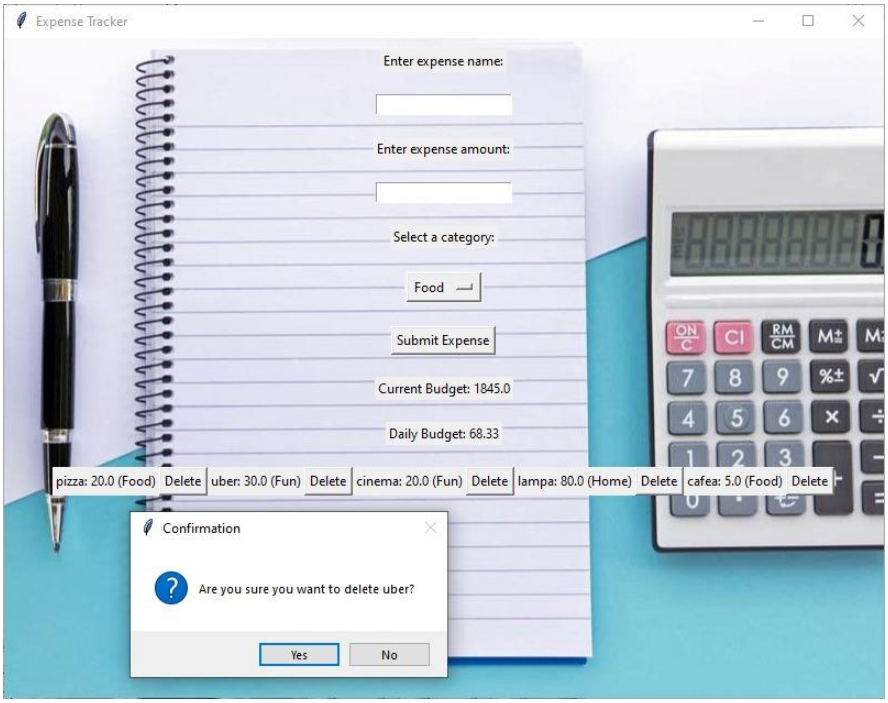

Figura 3.3.4. Mesaj de confirmare a dorinței de ștergere a cheltuielii

- După ștergerea articolului, pe ecran este afișat alt mesaj de tip pop-up, conform imaginii de mai jos:

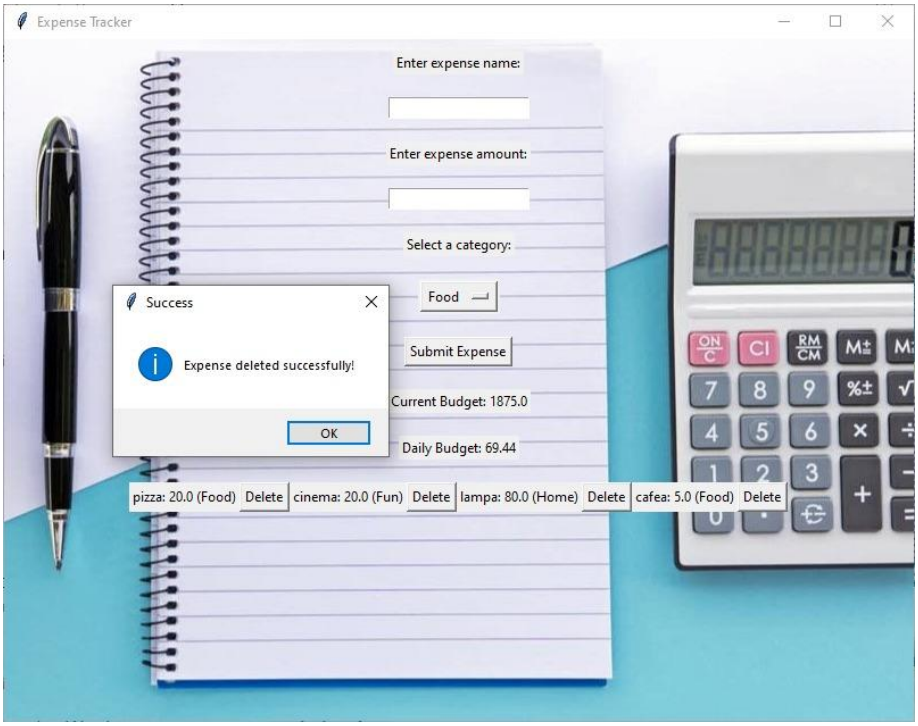

Figura 3.3.5. Mesaj de confirmarea a ștergerii cu succes a cheltuielii

6. **Imagine de Fundal Personalizată:**
    - Adaugă o imagine de fundal personalizată la fereastra principală, pentru a îmbunătăți estetica aplicației.
7. **Feedback Utilizator:**
    - Utilizează ferestre pop-up (**messagebox.showinfo**) pentru a furniza feedback utilizatorului, confirmând succesul salvării cheltuielilor.
8. **Manipularea Datelor cu Pandas:**
    - Se folosește de biblioteca **pandas** pentru manipularea datelor tabulare, citirea și scrierea acestora în fișierul CSV, precum și pentru calculul rezumatului cheltuielilor pe categorii.