Expense Tracker Project - Complete Documentation

📌Project Overview

The **Expense Tracker** is a mini-project built in Python to help users record, view, update, and delete their expenses. It started as a simple command-line application and was later upgraded with a **Tkinter GUI (Graphical User Interface)** for better usability.

The project teaches **file handling**, **data persistence**, **basic CRUD operations (Create, Read, Update, Delete)**, and **GUI programming with Tkinter**.

🛠️Libraries Used

1. **tabulate**
2. Purpose: To format and display expenses in a neat table in the terminal.
3. Example: `tabulate(data, headers, tablefmt="grid")`
4. **csv**
5. Purpose: To store expenses in a CSV file so data is not lost when the program closes.
6. Example: `csv.reader`, `csv.writer`.
7. **os**
8. Purpose: To check if the expense file exists before loading data.
9. Example: `os.path.exists(filename)`
10. **tkinter** (GUI Library)
11. Purpose: To create buttons, labels, entry fields, and a table view for expenses.
12. Example: `tk.Tk()`, `tk.Button()`, `tk.Entry()`
13. **tkinter.ttk (Treeview widget)**
14. Purpose: To create the table/grid inside the GUI.
15. Example: `ttk.Treeview(root, columns=..., show="headings")`

🧰Key Features Implemented

1. **Add Expense**
2. User enters amount, category, and description.
3. Expense is appended to a CSV file.
4. **View Expenses**
5. Displays all expenses either in terminal (tabulate) or GUI (Treeview).
6. **Update Expense**
7. User selects an expense by index/ID.
8. Updates details and rewrites the CSV file.
9. **Delete Expense**
10. User selects an expense by index/ID.
11. Removes it and updates the CSV file.
12. **Calculate Totals per Category**
13. Groups expenses by category and sums the amounts.
14. **GUI with Tkinter**
15. Buttons for Add, View, Update, Delete.
16. Input fields for expense details.
17. Table view for displaying expenses.

📖Glossary

- **CRUD**: Create, Read, Update, Delete (basic data operations).
- **CSV (Comma-Separated Values)**: Simple file format used to store tabular data.
- **Treeview**: A Tkinter widget to display tabular data inside the GUI.
- **Persistence**: Saving data to a file/database so it remains after the program ends.
- **Widget**: GUI components like buttons, labels, and input fields in Tkinter.

📝Cheat Sheet (Important Syntax)

File Handling (CSV)

```
import csv

# Writing to CSV
with open("expenses.csv", "a", newline="") as f:
    writer = csv.writer(f)
    writer.writerow([id, amount, category, description])

# Reading from CSV
with open("expenses.csv", "r") as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```

Tkinter Basics

```
import tkinter as tk
from tkinter import ttk

root = tk.Tk()  # Create window
root.title("Expense Tracker")

entry = tk.Entry(root)
entry.pack()

button = tk.Button(root, text="Add", command=add_expense)
button.pack()

root.mainloop()  # Run the GUI loop
```

Tkinter Treeview (Table)

```
tree = ttk.Treeview(root, columns=("ID", "Amount", "Category", "Description"),
show="headings")
tree.heading("ID", text="ID")
tree.heading("Amount", text="Amount")
tree.heading("Category", text="Category")
tree.heading("Description", text="Description")
tree.pack()
```

Tabulate (CLI Table)

```
from tabulate import tabulate

data = [[1, 50, "Food", "Pizza"], [2, 20, "Travel", "Bus"]]
print(tabulate(data, headers=["ID", "Amount", "Category", "Description"],
tablefmt="grid"))
```

🔍Explanation of Final GUI Code (Simplified)

- `root = tk.Tk()` → Creates main application window.
- `Entry` widgets → Take user input (Amount, Category, Description).
- `Button` widgets → Perform actions (Add, View, Update, Delete).
- `Treeview` widget → Displays expenses in a table.
- Functions like `add_expense()`, `update_expense()`, `delete_expense()` → Connect logic to button actions.
- CSV file ensures data persistence.

💡 Why These Choices?

- **CSV instead of Database**: Simpler for a beginner project, portable, no extra setup.
- **Tkinter**: Built-in Python GUI library, lightweight, no external installation.
- **tabulate**: Makes terminal outputs look professional and readable.
- **CRUD**: Fundamental operations that prove understanding of data manipulation.

💴How to Explain in an Interview

"The Expense Tracker is a Python project I built to practice file handling, data persistence, and GUI development. I used CSV for storing data since it's lightweight and easy to manage. The project supports CRUD operations – adding, viewing, updating, and deleting expenses. For the terminal version, I used the `tabulate` library to format tables, and later I upgraded it with a Tkinter GUI using Entry, Button, and Treeview widgets. This taught me how to connect user inputs with backend logic, how to manage files consistently, and how to build a simple but functional GUI application."

✅ With this summary, glossary, and cheat sheet, you can: - Recall what you built and why. - Revise important syntax. - Explain your project confidently in interviews. - Rebuild or upgrade it in the future.