FINAL PROJECT

Work on a project-

Expense Tracker:

Abstract:

The Expense Tracker is a simple Python application designed to help users manage their expenses efficiently. It provides a user-friendly interface using Tkinter for data input and visualization capabilities through Matplotlib. Let's dive into the key features:

1. **Expense Recording**: Users can add their expenses, including the date, description ,time, and amount.
2. **Expense Visualization**:The application generates visualizations to help users understand their spending habits:

   Line Chart: Tracks expenses against the budget for each category.
3. **Usage Instructions**:

   Adding Expense:

   Fill in the required details (date, time, description, amount).

   Click the "Add Expense" button.

   Deleting Expense:

   Select an expense from the list.

   Click the "Delete Expense" button to remove it.

   show Expenses:

   Click the "show Expenses" button to generate charts.
4. **Requirements**:

   Python >=3.7

   Tkinter

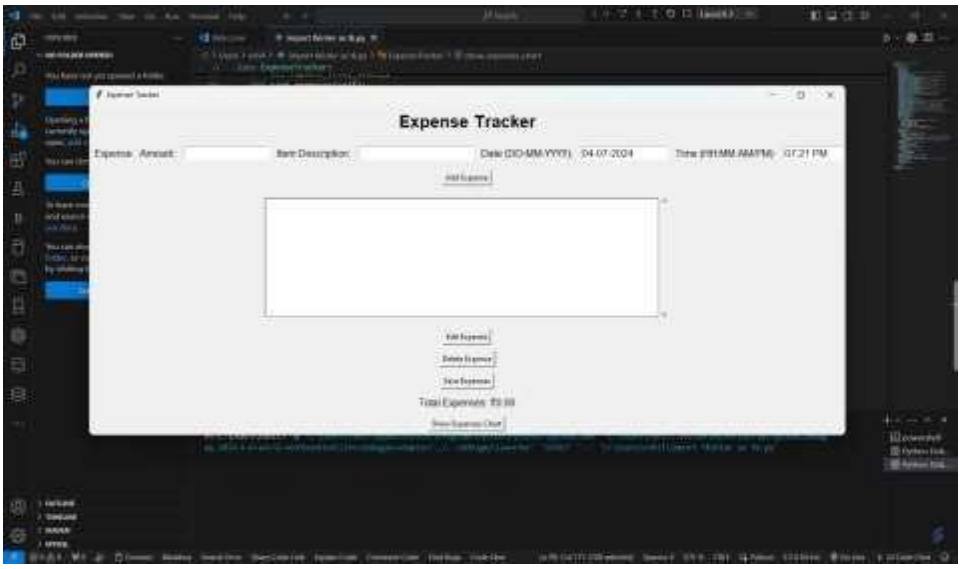
   Matplotlib

CODE:

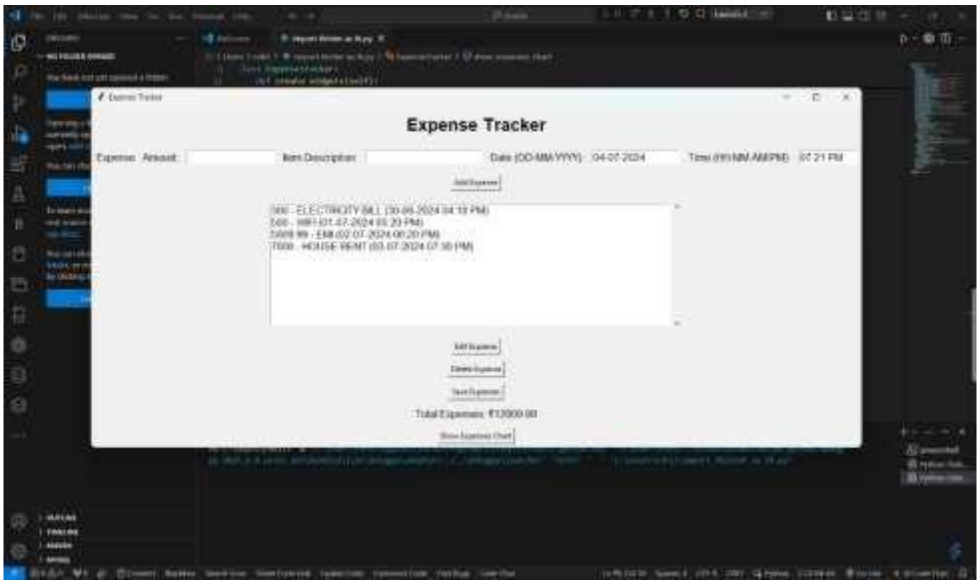
```
import tkinter as tk
from tkinter import ttk, messagebox, simpledialog
import csv
import matplotlib.pyplot as plt
from datetime import datetime
class ExpenseTracker:
    def __init__(self, root):
        self.root = root
        self.expenses = []
        self.create_widgets()
    def create_widgets(self):
        self.label = tk.Label(self.root, text="Expense Tracker",
font=("Helvetica", 20, "bold"))
        self.label.pack(pady=10)
        self.frame_input = tk.Frame(self.root)
        self.frame_input.pack(pady=10)
        self.expense_label = tk.Label(self.frame_input, text="Expense
Amount:", font=("Helvetica", 12))
        self.expense_label.grid(row=0, column=0, padx=5)
        self.expense_entry = tk.Entry(self.frame_input, font=("Helvetica",
12), width=15)
        self.expense_entry.grid(row=0, column=1, padx=5)
        self.item_label = tk.Label(self.frame_input, text="Item Description:",
font=("Helvetica", 12))
        self.item_label.grid(row=0, column=2, padx=5)
        self.item_entry = tk.Entry(self.frame_input, font=("Helvetica", 12),
width=20)
        self.item_entry.grid(row=0, column=3, padx=5)
        self.date_label = tk.Label(self.frame_input, text="Date (DD-MM-
YYYY):", font=("Helvetica", 12))
        self.date_label.grid(row=0, column=4, padx=5)
        self.date_entry = tk.Entry(self.frame_input, font=("Helvetica", 12),
width=15)
        self.date_entry.grid(row=0, column=5, padx=5)
        self.date_entry.insert(0,  datetime.now().strftime("%d-%m-%Y"))

        self.time_label = tk.Label(self.frame_input, text="Time (HH:MM
AM/PM):", font=("Helvetica", 12))
        self.time_label.grid(row=0, column=6, padx=5)
        self.time_entry = tk.Entry(self.frame_input, font=("Helvetica", 12),
width=10)
        self.time_entry.grid(row=0, column=7, padx=5)
        self.time_entry.insert(0, datetime.now().strftime("%I:%M %p"))
        self.add_button = tk.Button(self.root, text="Add Expense",
command=self.add_expense)
        self.add_button.pack(pady=5)
        self.frame_list = tk.Frame(self.root)
```

```python
        self.frame_list.pack(pady=10)
        self.scrollbar = tk.Scrollbar(self.frame_list)
        self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
        self.expense_listbox = tk.Listbox(self.frame_list, font=("Helvetica",
12), width=70, yscrollcommand=self.scrollbar.set)
        self.expense_listbox.pack(pady=5)
        self.scrollbar.config(command=self.expense_listbox.yview)
        self.edit_button = tk.Button(self.root, text="Edit Expense",
command=self.edit_expense)
        self.edit_button.pack(pady=5)
        self.delete_button = tk.Button(self.root, text="Delete Expense",
command=self.delete_expense)
        self.delete_button.pack(pady=5)
        self.save_button = tk.Button(self.root, text="Save Expenses",
command=self.save_expenses)
        self.save_button.pack(pady=5)
        self.total_label = tk.Label(self.root, text="Total Expenses:",
font=("Helvetica", 12))
        self.total_label.pack(pady=5)
        self.show_chart_button = tk.Button(self.root, text="Show Expenses
Chart", command=self.show_expenses_chart)
        self.show_chart_button.pack(pady=5)
        self.update_total_label()
    def add_expense(self):
        expense = self.expense_entry.get()
        item = self.item_entry.get()
        date = self.date_entry.get()
        time = self.time_entry.get()
        if expense and date:
            self.expenses.append((expense, item, date, time))
            self.expense_listbox.insert(tk.END, f"{expense} - {item} ({date}
{time})")
            self.expense_entry.delete(0, tk.END)
            self.item_entry.delete(0, tk.END)
            self.date_entry.delete(0, tk.END)
            self.time_entry.delete(0, tk.END)
            self.date_entry.insert(0, datetime.now().strftime("%d-%m-%Y"))

            self.time_entry.insert(0, datetime.now().strftime("%I:%M %p"))
        else:
            messagebox.showwarning("Warning", "Expense and Date cannot be
empty.")
        self.update_total_label()
    def edit_expense(self):
        selected_index = self.expense_listbox.curselection()
        if selected_index:
            selected_index = selected_index[0]
            selected_expense = self.expenses[selected_index]
            new_expense = simpledialog.askstring("Edit Expense", "Enter new expense amount:", initialvalue=selected_expense[0])
            if new_expense:
                self.expenses[selected_index] = (new_expense, selected_expense[1], selected_expense[2], selected_expense[3])
                self.expense_listbox.delete(selected_index)
                self.expense_listbox.insert(selected_index, f"{new_expense} - {selected_expense[1]} ({selected_expense[2]} {selected_expense[3]})")
        self.update_total_label()
    def delete_expense(self):
        selected_index = self.expense_listbox.curselection()
        if selected_index:
            selected_index = selected_index[0]
            del self.expenses[selected_index]
            self.expense_listbox.delete(selected_index)
        self.update_total_label()
    def save_expenses(self):
        with open("expenses.csv", "w", newline="") as file:
            writer = csv.writer(file)
            writer.writerow(["Expense", "Item", "Date", "Time"])
            for expense in self.expenses:
                writer.writerow(expense)
    def update_total_label(self):
        total = sum(float(expense[0]) for expense in self.expenses)
        self.total_label.config(text=f"Total Expenses: ₹{total:.2f}")
    def show_expenses_chart(self):
       dates = [datetime.strptime(expense[2] + " " + expense[3], "%d-%m-%Y %I:%M %p") for expense in self.expenses]

        amounts = [float(expense[0]) for expense in self.expenses]
        plt.plot(dates, amounts)
        plt.xlabel("Date")
        plt.ylabel("Amount (₹)")
        plt.title("Expenses Over Time")
        plt.show()
root = tk.Tk()
root.title("Expense Tracker")
expense_tracker = ExpenseTracker(root)
root.mainloop()
```

Output-



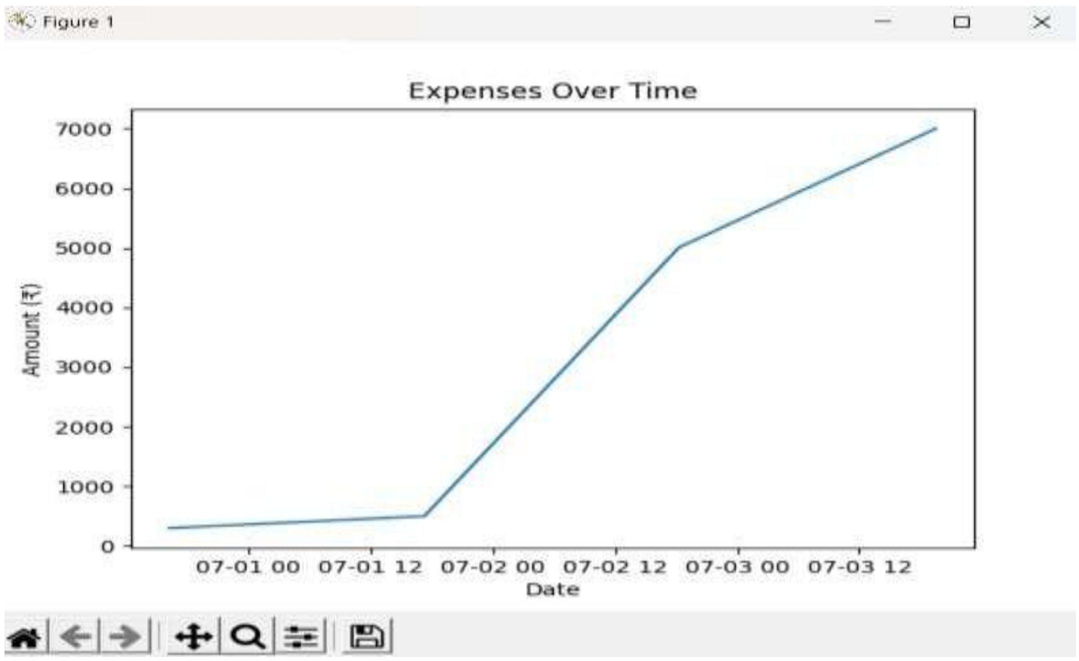
Interface of the expense tracker by entering amount,description and editing date & time. Simply click on the ‘add expense’. Then it will be added to the section.



```
500 - ELECTRICITY BILL (30-06-2024 04:10 PM)
500 - WIFI (01-07-2024 05:20 PM)
5000.00 - EMI (02-07-2024 08:20 PM)
7000 - HOUSE RENT (03-07-2024 07:30 PM)
```

Total Expenses: ₹13000.00

After adding the expenses click on ‘save expenses’. It will show the total expenses in INR . Later click on the ‘show expenses chart’.



Now we can see the total expenses in the line chart by showing the axis X as ‘date’ and the axis Y as ‘Amount’. This shows a data visualization of the expenses tracker.

Submitted by,

Manmohan kumar