FINAL PROJECT

Work on a project- Expense Tracker:

Abstract:

~~The Expe~~nse 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.

1. 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:

1. Click the “show Expenses” button to generate charts.

Python >=3.7Requirements:

Tkinter

Matplotlib

CODE:

import tkinter as tk

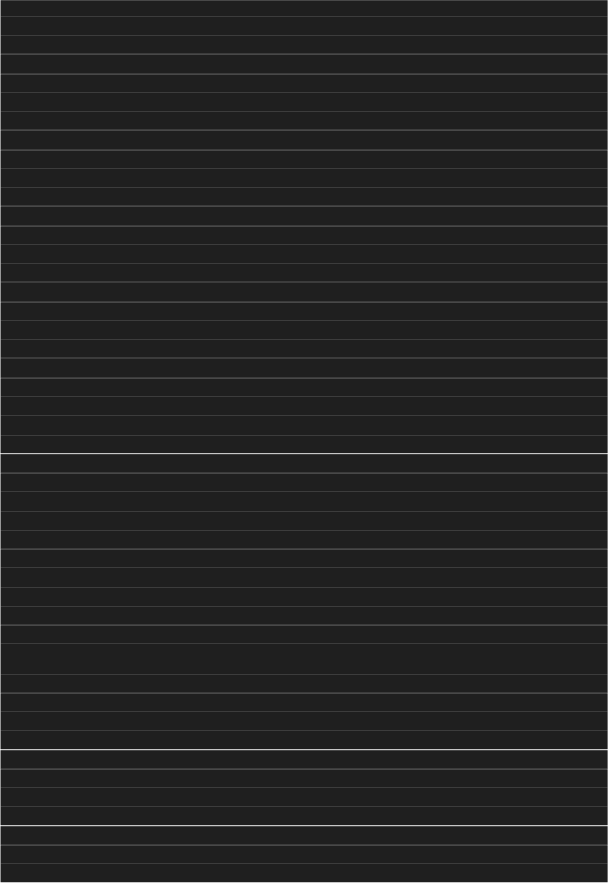
from tkinter import ttk, messagebox, simpledialog import csv

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-MMYYYY):", 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) 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

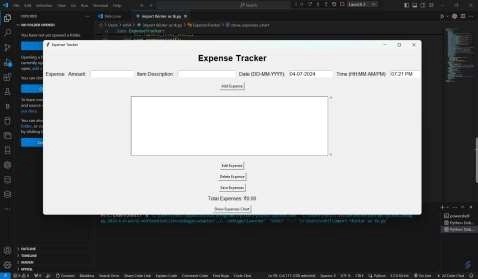
%I:%M %p") for expense in self.expenses]

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()

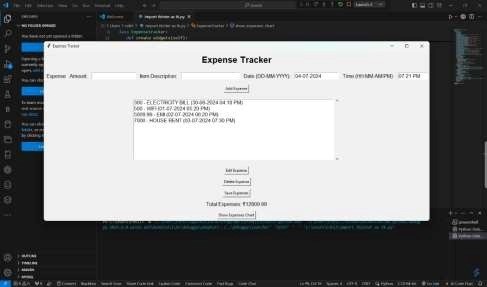
root = tk.Tk()

root.title("Expense Tracker") expense\_tracker = ExpenseTracker(root) root.mainloop()

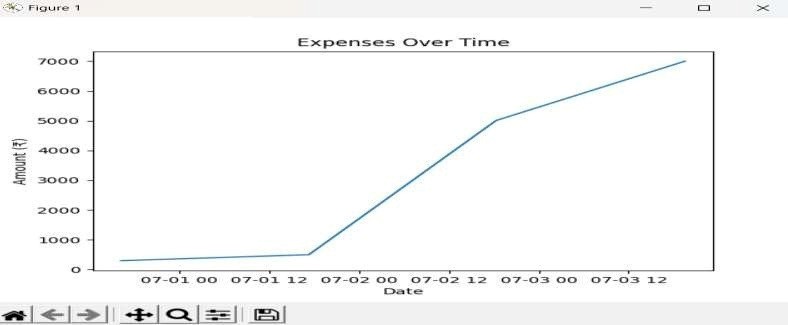
Output-



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.



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,

Lokesh