Expense Tracker Application

A simple Python Tkinter-based Expense Tracking Tool

Manoj Yadlapalli
MY71701n@pace.edu
Introduction to Coding – Python
Prof. Shoaib Ahmed
Pace University

Introduction

The **Expense Tracker Application** is a Python-based tool built with Tkinter to help users manage their personal finances. It allows users to add, view, and categorize their expenses, as well as calculate the total spending. Key features include:

- Adding expenses with description, amount, and category
- Viewing all expenses and total amount
- Summarizing expenses by category

This project demonstrates how Tkinter can be used to create simple, user-friendly applications for everyday use.

❖ Purpose: "Helps users monitor and manage their finances easily."

Tutorial/Material Used

- **Python Tkinter Tutorial**:

Real Python Tkinter Tutorial

This tutorial provided a comprehensive introduction to Tkinter, covering the essentials of GUI development in Python.

- **Python Tkinter Documentation**:

Tkinter Documentation

I referred to the official documentation to understand different Tkinter widgets and their functionalities, ensuring the proper use of GUI elements.

- **Python Official Documentation**:

Python Docs

For general Python programming concepts, functions, and error handling techniques, I relied on the official Python documentation.

Customization and Enhancements

•**Expense Categories**:

Added the ability for users to categorize their expenses (e.g., Food, Entertainment, etc.) for better organization and analysis.

•**Error Handling for Input Validation**:

Implemented input checks to ensure all fields are filled and the amount is a valid number, preventing incorrect data entry.

•**Expense Summary by Category**:

Developed a feature to view total expenses grouped by category, helping users identify spending patterns.

•**User-friendly GUI Layout**:

Designed a simple and intuitive interface using Tkinter's layout management, ensuring easy navigation and interaction with the application.

Learning Outcomes

•**Using Tkinter for GUI Development**:

Gained experience in creating graphical user interfaces with Tkinter, including working with widgets like buttons, labels, and entry fields.

•**Handling User Input and Validation**:

Learned how to process and validate user input to ensure data is accurate and properly formatted before storing it.

Learning Outcomes

•**Working with Lists and Dictionaries for Storing Data**:

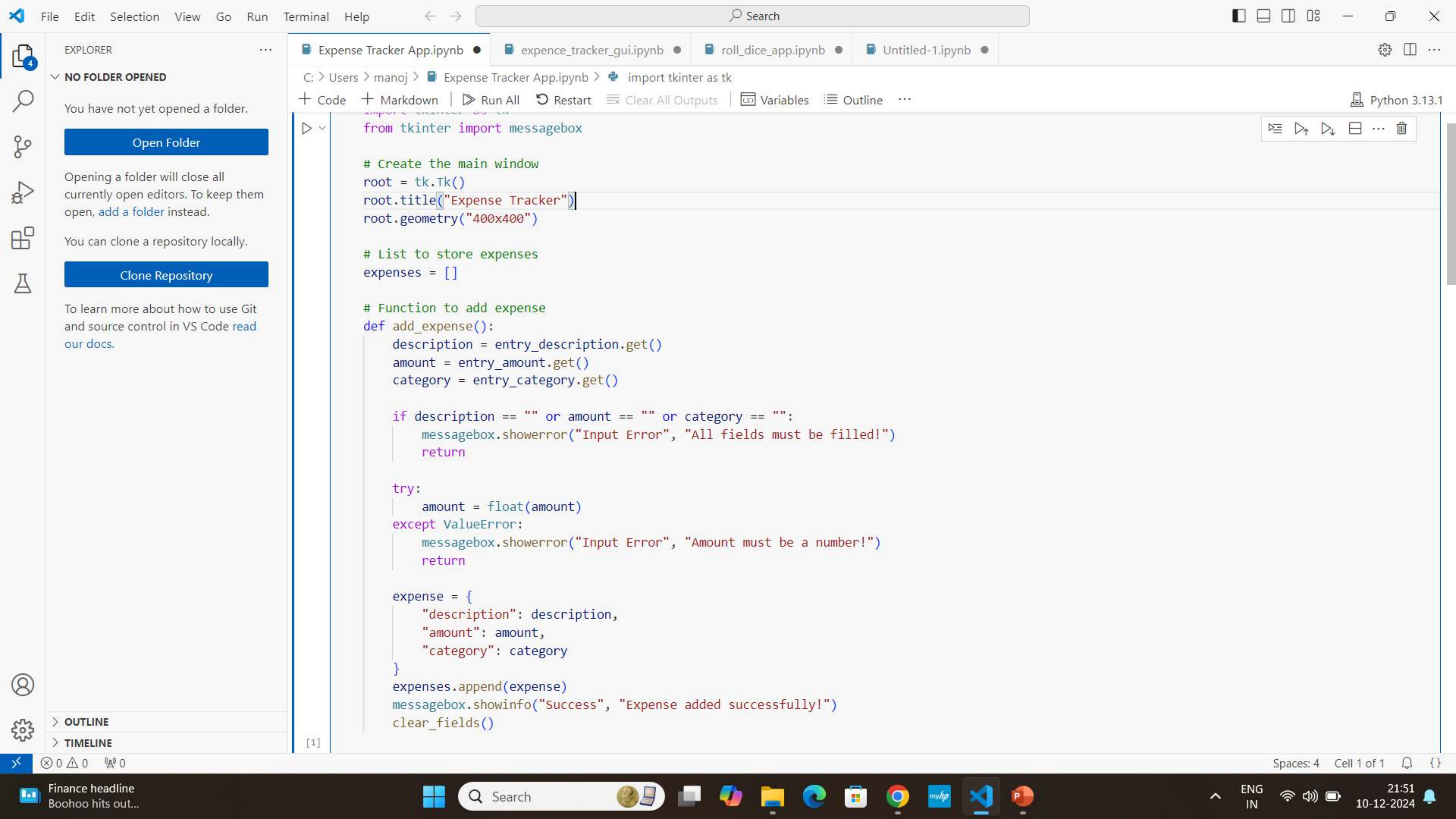
Used lists to store expenses and dictionaries to organize each expense with attributes like description, amount, and category.

•**Displaying Dynamic Content Based on User Input**:

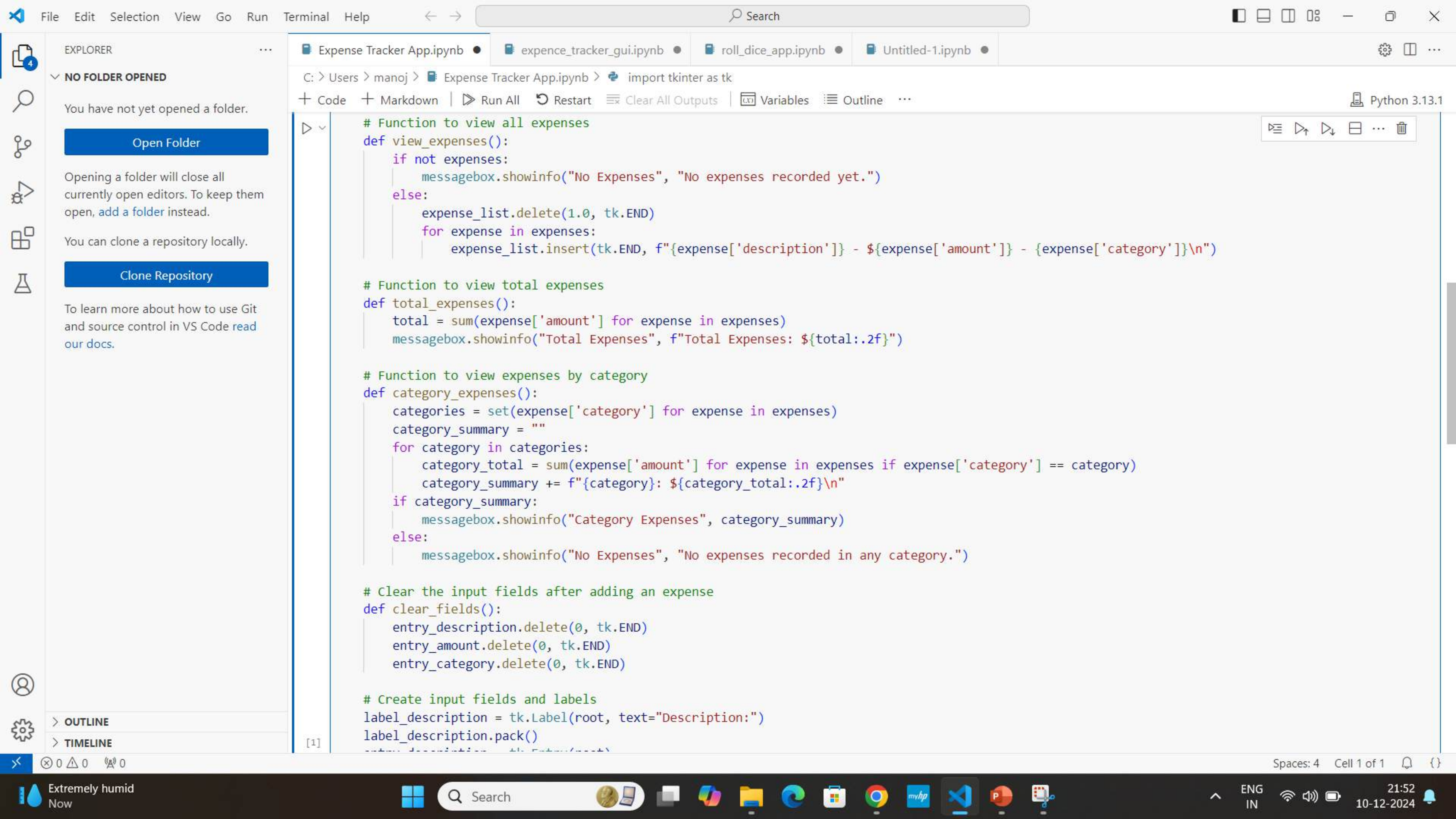
Implemented features that update the interface in real-time based on user actions, such as adding expenses or calculating totals.

•**Implementing Error Handling**:

Developed error handling to ensure the application handles invalid input or missing data without crashing, providing clear feedback to users.


File Edit Selection View Go Run Terminal Help
Search
Expense Tracker App.ipynb
expence_tracker_gui.ipynb
roll_dice_app.ipynb
Untitled-1.ipynb
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
You can clone a repository locally.
Clone Repository
To learn more about how to use Git and source control in VS Code read our docs.
OUTLINE
TIMELINE
C: > Users > manoj > Expense Tracker App.ipynb > import tkinter as tk
Code Markdown Run All Restart Clear All Outputs Variables Outline
Python 3.13.1
[1]
Spaces: 4 Cell 1 of 1
Finance headline
Boohoo hits out...
ENG IN
21:51
10-12-2024

```python
from tkinter import messagebox

# Create the main window
root = tk.Tk()
root.title("Expense Tracker")
root.geometry("400x400")

# List to store expenses
expenses = []

# Function to add expense
def add_expense():
    description = entry_description.get()
    amount = entry_amount.get()
    category = entry_category.get()

    if description == "" or amount == "" or category == "":
        messagebox.showerror("Input Error", "All fields must be filled!")
        return

    try:
        amount = float(amount)
    except ValueError:
        messagebox.showerror("Input Error", "Amount must be a number!")
        return

    expense = {
        "description": description,
        "amount": amount,
        "category": category
    }
    expenses.append(expense)
    messagebox.showinfo("Success", "Expense added successfully!")
    clear_fields()
```



You have not yet opened a folder.

Open Folder

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

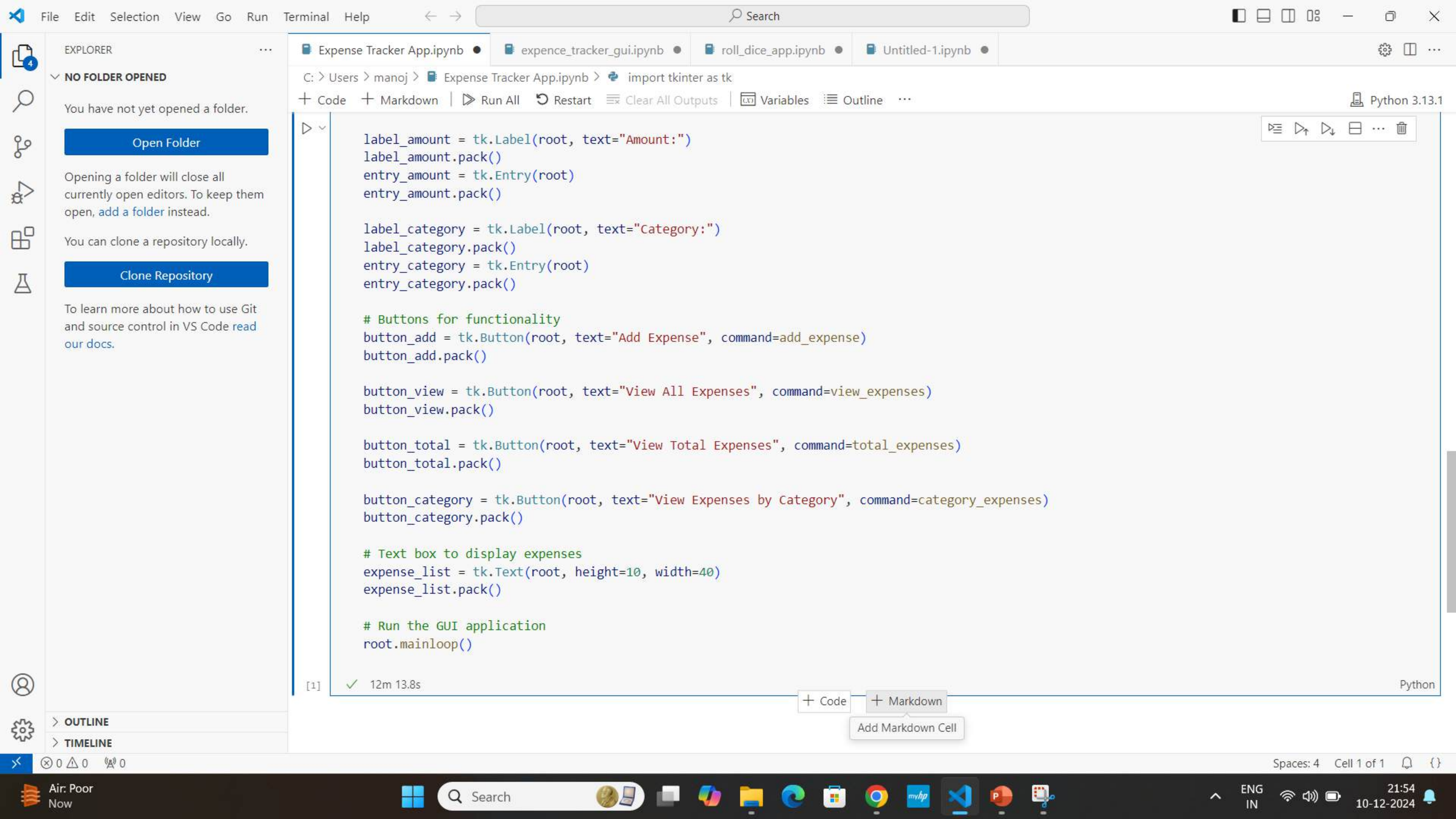
You can clone a repository locally.

Clone Repository

To learn more about how to use Git and source control in VS Code read our docs.

```python
# Function to view all expenses
def view_expenses():
    if not expenses:
        messagebox.showinfo("No Expenses", "No expenses recorded yet.")
    else:
        expense_list.delete(1.0, tk.END)
        for expense in expenses:
            expense_list.insert(tk.END, f"{expense['description']} - ${expense['amount']} - {expense['category']}\n")

# Function to view total expenses
def total_expenses():
    total = sum(expense['amount'] for expense in expenses)
    messagebox.showinfo("Total Expenses", f"Total Expenses: ${total:.2f}")

# Function to view expenses by category
def category_expenses():
    categories = set(expense['category'] for expense in expenses)
    category_summary = ""
    for category in categories:
        category_total = sum(expense['amount'] for expense in expenses if expense['category'] == category)
        category_summary += f"{category}: ${category_total:.2f}\n"
    if category_summary:
        messagebox.showinfo("Category Expenses", category_summary)
    else:
        messagebox.showinfo("No Expenses", "No expenses recorded in any category.")

# Clear the input fields after adding an expense
def clear_fields():
    entry_description.delete(0, tk.END)
    entry_amount.delete(0, tk.END)
    entry_category.delete(0, tk.END)

# Create input fields and labels
label_description = tk.Label(root, text="Description:")
label_description.pack()
```

[1]



You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

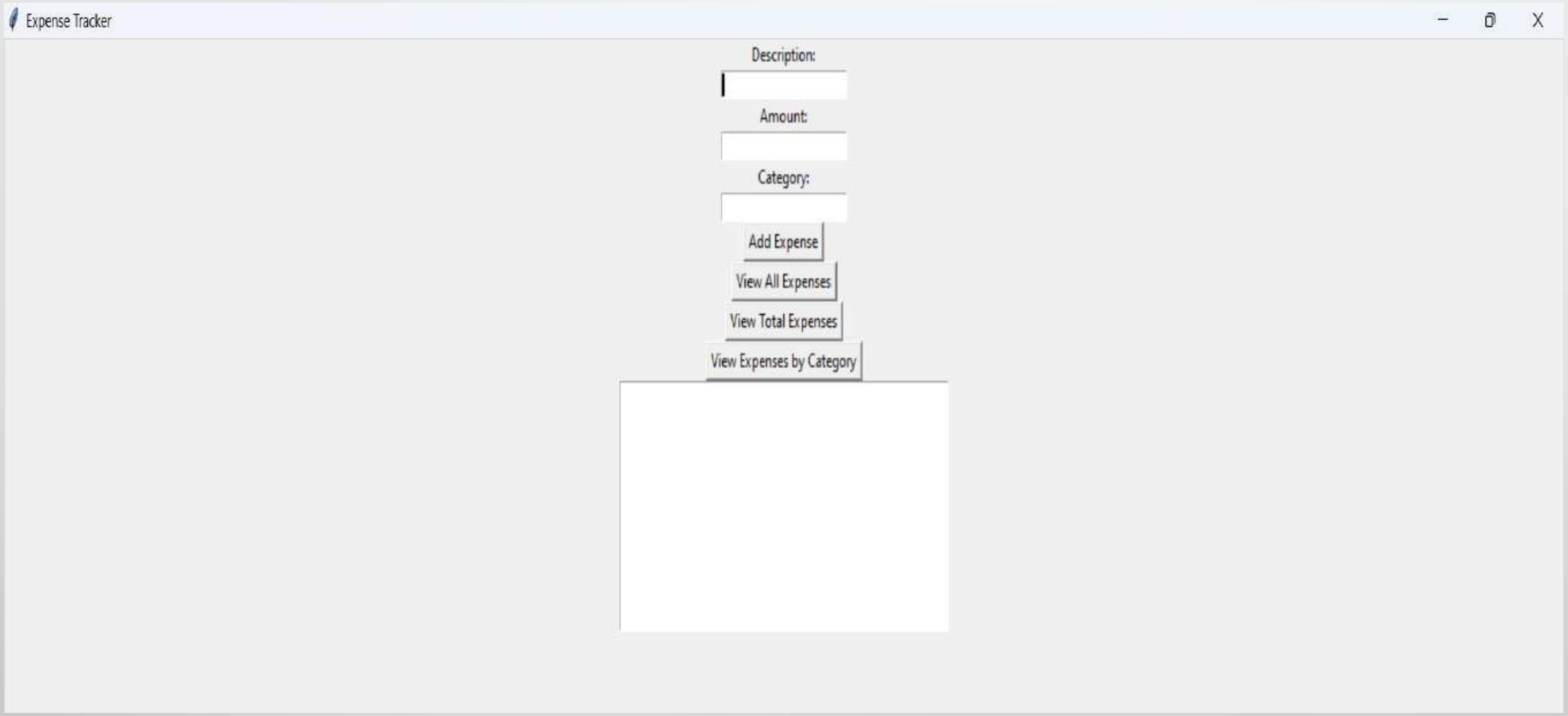
You can clone a repository locally.

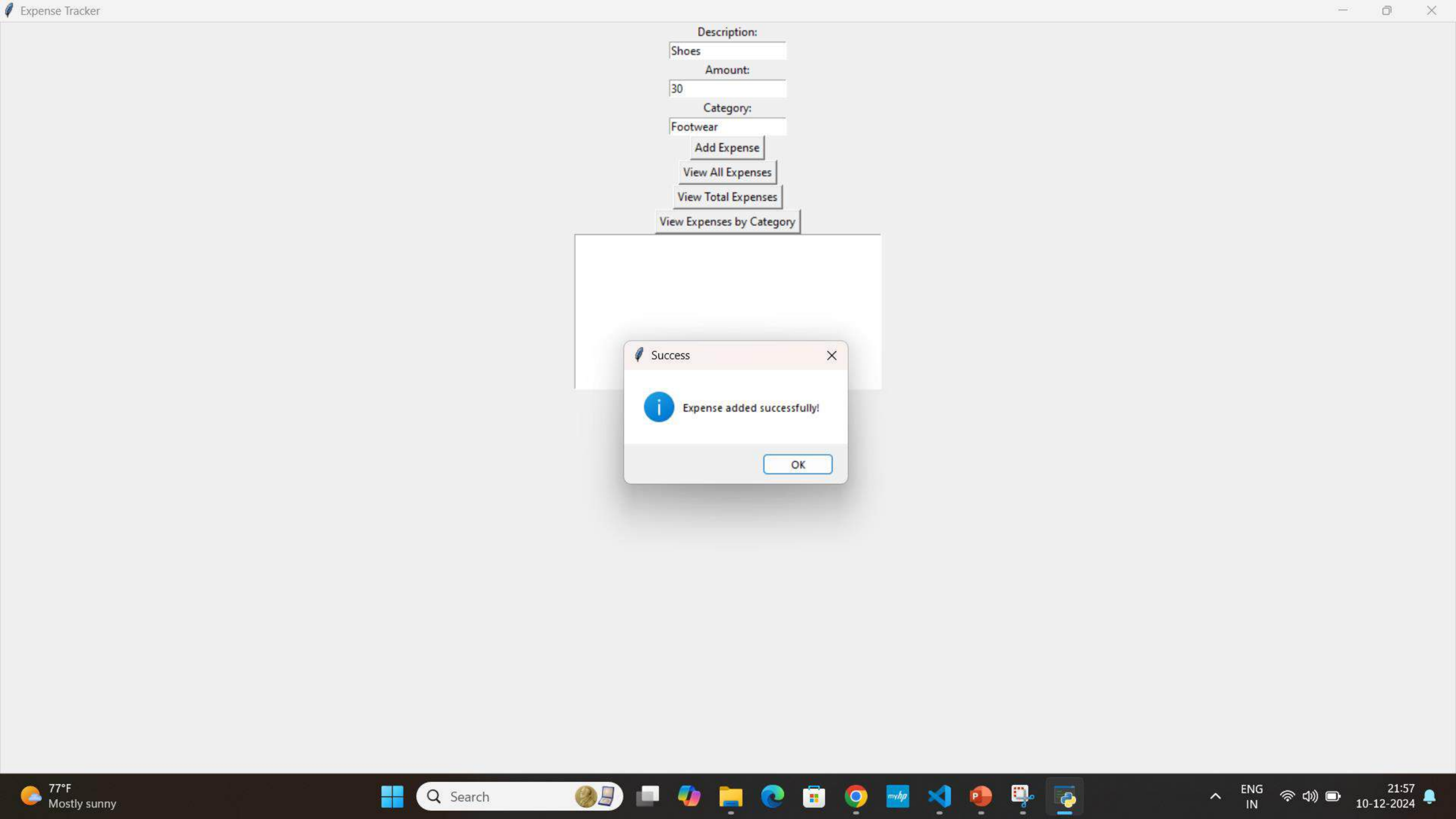
To learn more about how to use Git and source control in VS Code read our docs.

```python
label_amount = tk.Label(root, text="Amount:")
label_amount.pack()
entry_amount = tk.Entry(root)
entry_amount.pack()

label_category = tk.Label(root, text="Category:")
label_category.pack()
entry_category = tk.Entry(root)
entry_category.pack()

# Buttons for functionality
button_add = tk.Button(root, text="Add Expense", command=add_expense)
button_add.pack()

button_view = tk.Button(root, text="View All Expenses", command=view_expenses)
button_view.pack()

button_total = tk.Button(root, text="View Total Expenses", command=total_expenses)
button_total.pack()

button_category = tk.Button(root, text="View Expenses by Category", command=category_expenses)
button_category.pack()

# Text box to display expenses
expense_list = tk.Text(root, height=10, width=40)
expense_list.pack()

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

[1] ✓ 12m 13.8s

RESULT


Expense Tracker
Description:
Amount:
Category:
Add Expense
View All Expenses
View Total Expenses
View Expenses by Category



Expense Tracker

Description:

Shoes

Amount:

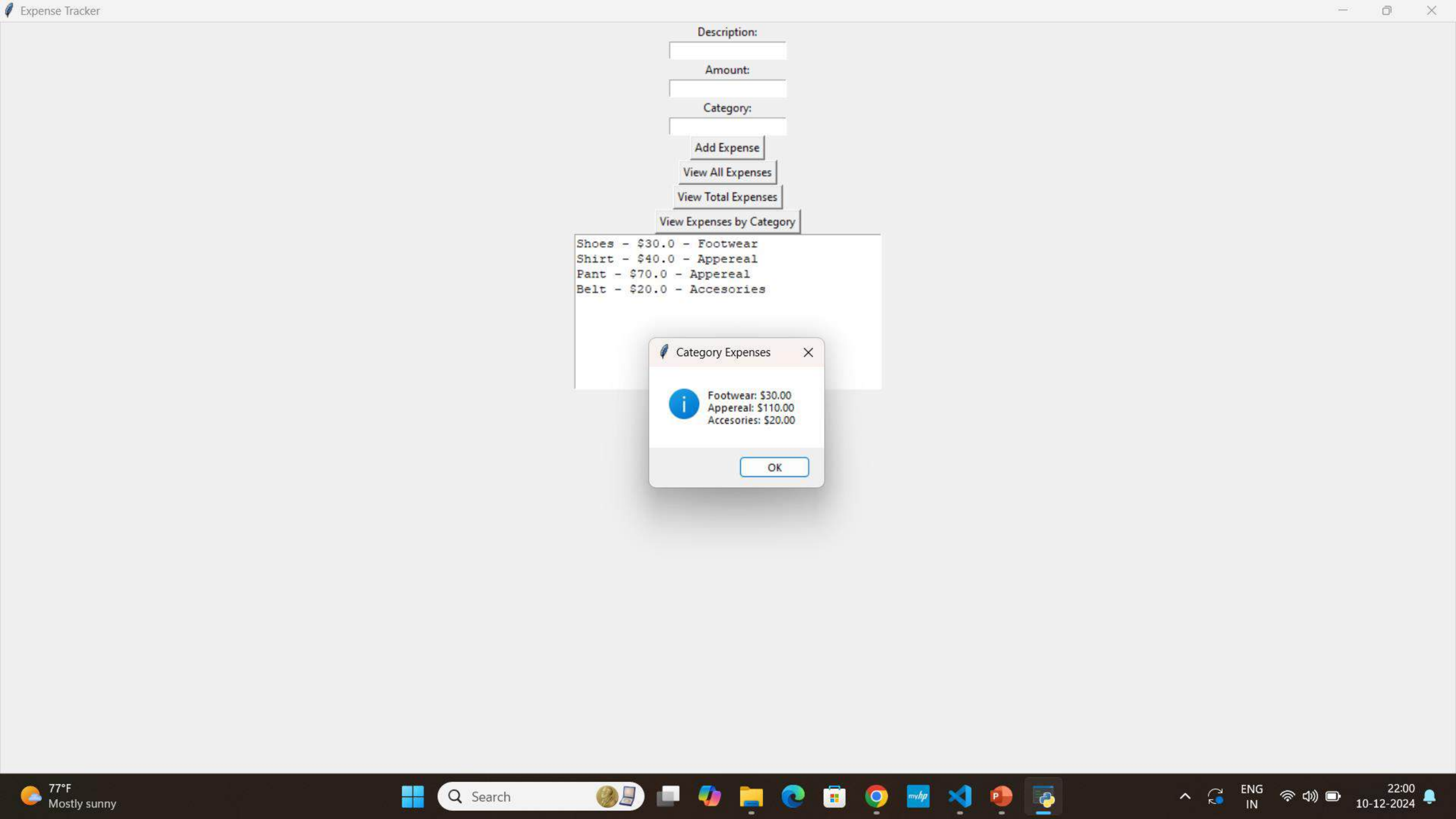
30

Category:

Footwear

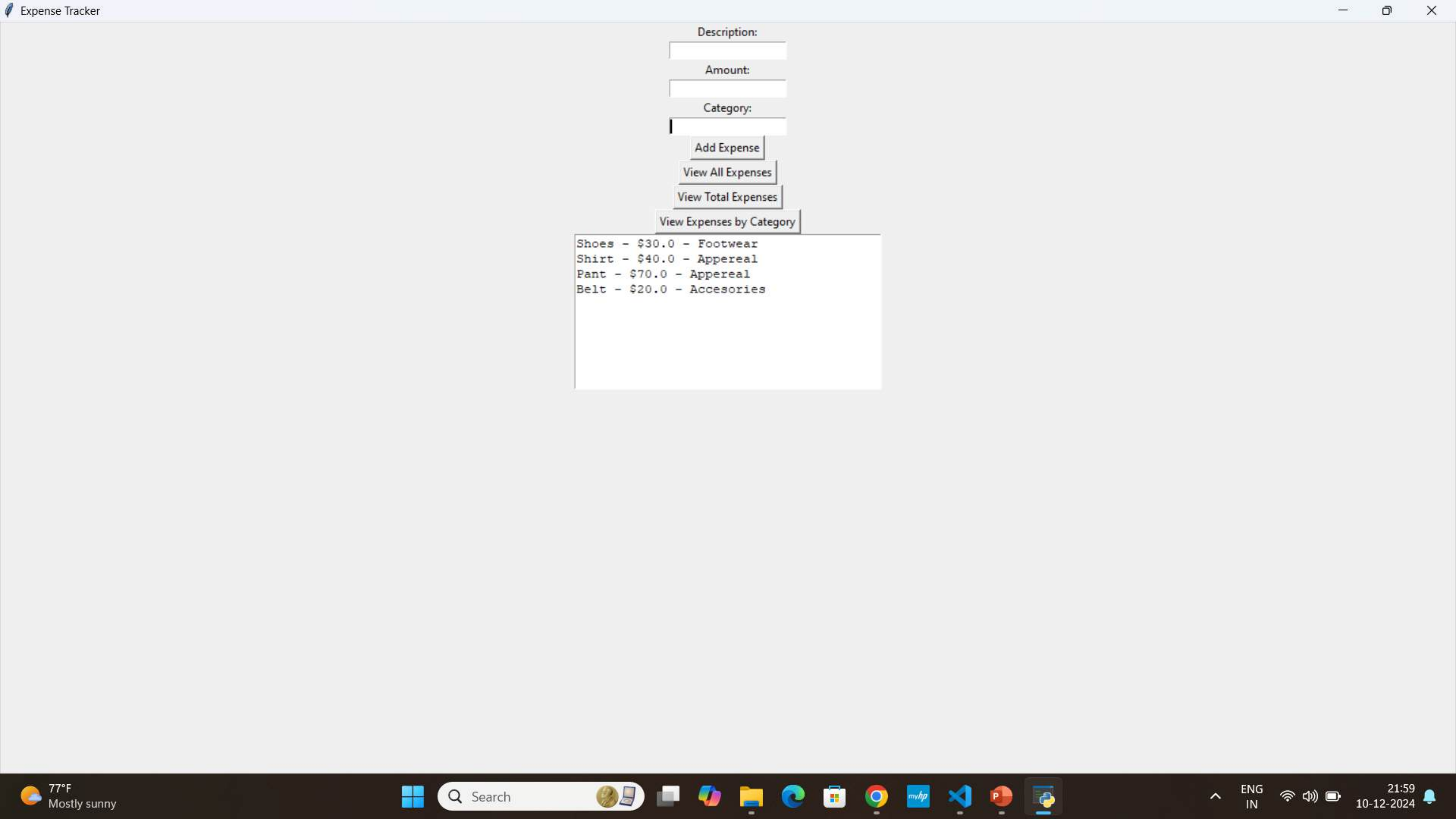
Success

Expense added successfully!

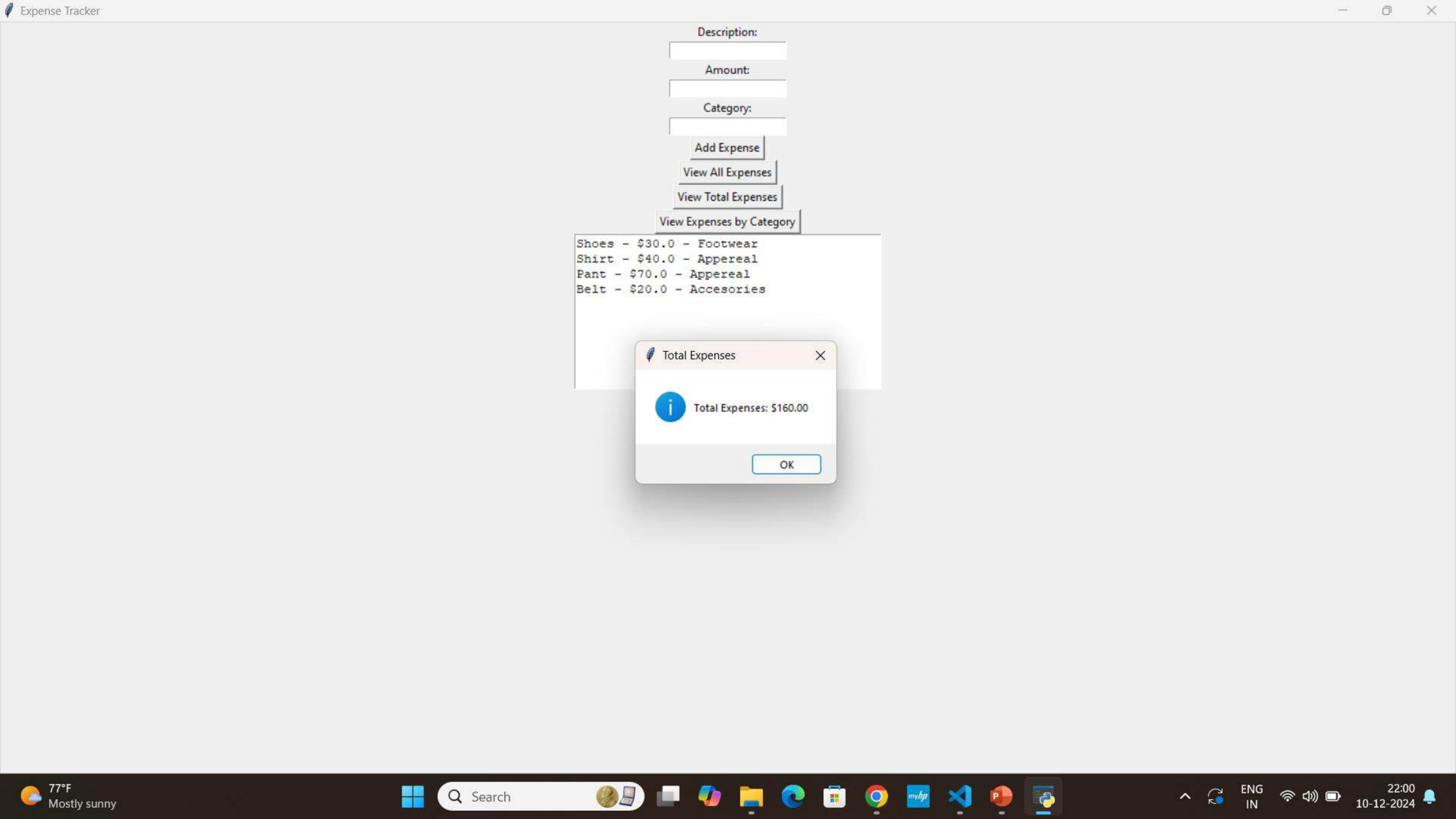


```
Shoes - $30.0 - Footwear
Shirt - $40.0 - Appereal
Pant - $70.0 - Appereal
Belt - $20.0 - Accesories
```

```
Footwear: $30.00
Appereal: $110.00
Accesories: $20.00
```



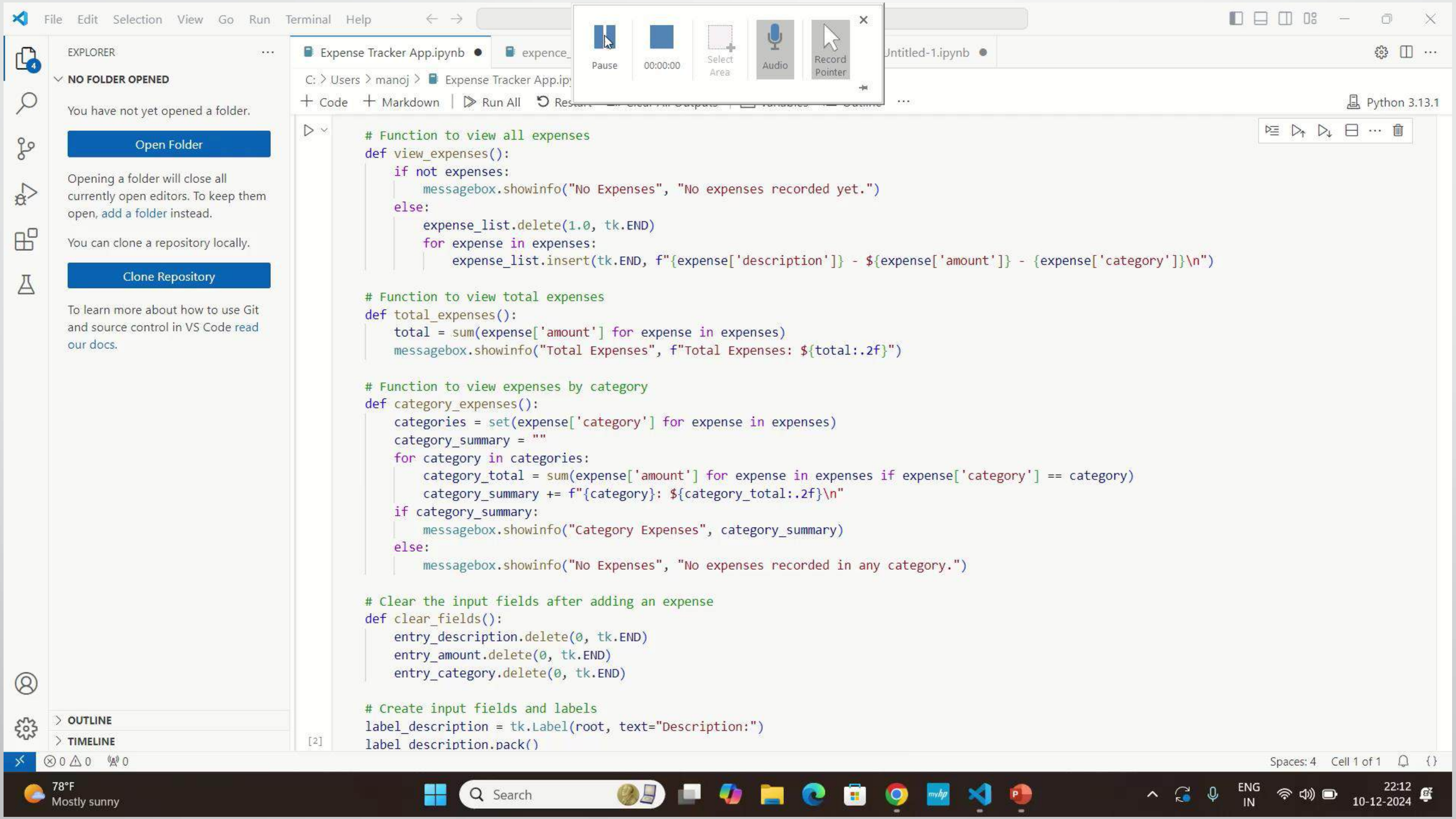
```
Shoes - $30.0 - Footwear
Shirt - $40.0 - Appereal
Pant - $70.0 - Appereal
Belt - $20.0 - Accesories
```



```
Shoes - $30.0 - Footwear
Shirt - $40.0 - Appereal
Pant - $70.0 - Appereal
Belt - $20.0 - Accesories
```

Total Expenses: $160.00

Video Bite of Project



You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

You can clone a repository locally.

To learn more about how to use Git and source control in VS Code read our docs.

```python
# Function to view all expenses
def view_expenses():
    if not expenses:
        messagebox.showinfo("No Expenses", "No expenses recorded yet.")
    else:
        expense_list.delete(1.0, tk.END)
        for expense in expenses:
            expense_list.insert(tk.END, f"{expense['description']} - ${expense['amount']} - {expense['category']}\n")

# Function to view total expenses
def total_expenses():
    total = sum(expense['amount'] for expense in expenses)
    messagebox.showinfo("Total Expenses", f"Total Expenses: ${total:.2f}")

# Function to view expenses by category
def category_expenses():
    categories = set(expense['category'] for expense in expenses)
    category_summary = ""
    for category in categories:
        category_total = sum(expense['amount'] for expense in expenses if expense['category'] == category)
        category_summary += f"{category}: ${category_total:.2f}\n"
    if category_summary:
        messagebox.showinfo("Category Expenses", category_summary)
    else:
        messagebox.showinfo("No Expenses", "No expenses recorded in any category.")

# Clear the input fields after adding an expense
def clear_fields():
    entry_description.delete(0, tk.END)
    entry_amount.delete(0, tk.END)
    entry_category.delete(0, tk.END)

# Create input fields and labels
label_description = tk.Label(root, text="Description:")
label_description.pack()
```

Any Queries

THANK
YOU!