

**S.B.O.A. SCHOOL AND JUNIOR COLLEGE**

**Anna Nagar Western Extension, Chennai - 600 101**

**PROJECT REPORT**

**2021 – 2022**

**Name :**

**Standard :              Sec:**

**Reg. No :**

**Title of the Project :**

**BONAFIDE CERTIFICATE**

**Certified to be the bonafide project work done by ……………………………………………………………… of Std………………………… … in the……………………………………………………………… laboratory of S.B.O.A.  School & Junior College, Chennai - 600 101 during the year 2021 - 2022.**

**Date : Teacher-in-Charge**

**Submitted for the ……………………………………………………………………**

**examination held in the year 2020 – 2021 at S.B.O.A. School and Junior**

**College.  Chennai - 600 101.**

**External  Examiner Internal  Examiner**

**ACKNOWLEDGMENT**

I would like to state that this project is my original work and would like to thank all those people who have wholeheartedly extended their cooperation and guidance for making it possible to complete this project on time.

My sincere gratitude to Our School Management for providing us the best infrastructure and all the required resources. My special thanks to school Principal **Mrs. Sharadha Ramamurthy** and Vice Principal **Mrs. Mahalakshmi** for their unconditional support. Many many thanks to my Computer Science teacher **Mrs. K. Bhavani** for her valuable guidance and support. I would also like to thank my family members and friends for their cooperation in completing this project within stipulated time.



EXPENSE TRACKER

Computer Science Project





**Index**

Title Page no.

* Objective 6
* Modules and files used 7
* Scope 8
* Source Code 10
* Output Screen 36
* Bibliography 43

**OBJECTIVE**

 **Log Expenses**: Users can enter various types of expenses, including categories (e.g., food, entertainment, transportation), amounts, dates, and notes.

 **Organize Expenses by Category**: The program allows users to categorize their expenses, making it easier to review spending patterns over time.

 **Monitor Total Spending**: The app calculates and displays the total amount spent, helping users track their overall financial situation.

 **Data Storage and Management**: The app stores all data in a CSV file, providing a record of all past expenses. Users can add, edit, and delete expenses.

 **User Authentication**: Each user has their own unique account, allowing for personal tracking of expenses and safeguarding user data.

 **Expense Analysis**: The app provides a visual representation (pie chart) of expenses, making it easy for users to analyse their spending by category.

 **Simple and Accessible Interface**: The app ensures ease of use through an intuitive and clean design, making expense tracking convenient for everyday users.

**Modules and Files Used**

1. **MODULES**

 **CSV MODULE**: Handles reading from and writing to the CSV file that stores user expense data, such as expenses and user credentials.

 **SYS MODULE**: Used to accept the user ID passed from login.py to main.py, enabling user-specific functionality.

 **TKINTER MODULE (and its submodules messagebox, ttk)**: Provides the graphical user interface (GUI) for the app, including windows, input fields, buttons, and dialog boxes.

 **DATETIME MODULE**: Helps manage dates by automatically recording the current date for expenses or validating user input for date fields.

 **MATPLOTLIB MODULE:** Used to generate and display pie charts that visually represent expense distribution by category.

 **PIL MODULE (Image, ImageTk)**: Handles image processing and rendering for displaying images (e.g., icons, backgrounds) in the app.

 **SUBPROCESS**: Enables the app to reopen login.py (or other scripts) when the user logs out by running Python scripts as separate processes.

1. **DATAFILES**

* **CSV FILES:**

MULTIPLE NUMBER OF CSV FILES USED AND ACCESSSED USING PYTHON OBJECTS

**SCOPE**

This project is carried out in Python IDLE version 3.9 (64-bit). The user will first be asked to create their own account with a User ID and a password. The User Id allows the user to access the following functionalities:

Option 1: (Add Expense)

To record a new expense by providing details such as date, category, amount, and note. The expense is assigned a unique Id and is automatically saved in the expenses.csv file.

Option 2: (Edit Expense)

To update existing expense details. The user selects the expense to be edited by entering its Id. This ensures easy management and correction of expense records, while ensuring changes are securely updated in the CSV file.

Option 3: (Delete Expense)

To remove an existing expense. The user specifies the serial number of the expense to be deleted. This operation is verified by checking the User ID, ensuring only the owner can delete their data.

Option 4: (View Expenses)

To display all existing expenses, including columns for serial number, date, category, amount, and notes. The records are filtered by the current User ID, ensuring privacy. The expenses are displayed in sequential order to maintain clarity.

Option 5: (Analyse Expenses)

To visualize expenses using a category-wise pie chart. The chart displays each category with percentage contributions, along with a color-coded legend below the chart for ease of understanding. This helps users analyze their spending patterns.

Option 6: (Logout)

To securely log out of the current session. This closes the expense tracker application and redirects the user to the login screen for enhanced security and session management.

**Login.py :**

SOURCE CODE

import tkinter as tk

from tkinter import messagebox

import csv

import os

from PIL import Image, ImageTk

# File for storing login credentials

CREDENTIALS\_FILE = 'login\_credentials.csv'

# Ensure credentials file exists

if not os.path.exists(CREDENTIALS\_FILE):

    with open(CREDENTIALS\_FILE, 'w', *newline*='') as csvfile:

        writer = csv.DictWriter(csvfile, *fieldnames*=['userid', 'fullname', 'password'])

        writer.writeheader()

# Function to verify login credentials

*def* login():

    userid = userid\_entry.get().strip()

    password = password\_entry.get().strip()

    if not userid or not password:

        messagebox.showerror("Login Failed", "User ID and Password cannot be empty!")

        return

    with open(CREDENTIALS\_FILE, 'r') as csvfile:

        reader = csv.DictReader(csvfile)

        for row in reader:

            if row['userid'] == userid and row['password'] == password:

                messagebox.showinfo("Login Success", *f*"Welcome, {row['fullname']}!")

                root.destroy()  # Close login window

                open\_main\_window(userid)  # Open main window

                return

    messagebox.showerror("Login Failed", "Invalid User ID or Password!")

# Function to create a new account

*def* create\_account():

*def* save\_account():

        new\_userid = userid\_entry\_new.get().strip()

        fullname = fullname\_entry\_new.get().strip()

        new\_password = password\_entry\_new.get().strip()

        if not new\_userid or not fullname or not new\_password:

            messagebox.showerror("Error", "All fields are required!")

            return

        # Check if User ID already exists

        with open(CREDENTIALS\_FILE, 'r') as csvfile:

            reader = csv.DictReader(csvfile)

            for row in reader:

                if row['userid'] == new\_userid:

                    messagebox.showerror("Error", "User ID already exists!")

                    return

        # Save the new account

        with open(CREDENTIALS\_FILE, 'a', *newline*='') as csvfile:

            writer = csv.DictWriter(csvfile, *fieldnames*=['userid', 'fullname', 'password'])

            writer.writerow({'userid': new\_userid, 'fullname': fullname, 'password': new\_password})

        messagebox.showinfo("Success", "Account created successfully!")

        create\_window.destroy()

    # Open a new window for account creation

    create\_window = tk.Toplevel(root)

    create\_window.title("Create New Account")

    create\_window.geometry("300x250")

    tk.Label(create\_window, *text*="User ID:").pack(*pady*=5)

    userid\_entry\_new = tk.Entry(create\_window, *width*=30)

    userid\_entry\_new.pack(*pady*=5)

    tk.Label(create\_window, *text*="Full Name:").pack(*pady*=5)

    fullname\_entry\_new = tk.Entry(create\_window, *width*=30)

    fullname\_entry\_new.pack(*pady*=5)

    tk.Label(create\_window, *text*="Password:").pack(*pady*=5)

    password\_entry\_new = tk.Entry(create\_window, *show*="\*", *width*=30)

    password\_entry\_new.pack(*pady*=5)

    tk.Button(create\_window, *text*="Create Account", *command*=save\_account, *width*=20).pack(*pady*=10)

# Function to open the main window

*def* open\_main\_window(*userid*):

    import subprocess

    subprocess.Popen(['python', 'Main.py', *userid*])

# Setup the login UI

root = tk.Tk()

root.title("Login Page")

root.geometry("300x320")

image\_path = 'cg.png'

image = Image.open(image\_path)

window\_width = 360

window\_height = 550

image.thumbnail((window\_width, window\_height))  # Resize keeping aspect ratio

image = image.resize((window\_width, window\_height))

tk\_image = ImageTk.PhotoImage(image)

background\_label = tk.Label(root, *image*=tk\_image)

background\_label.image = tk\_image  # Keep a reference to prevent garbage collection

background\_label.place(*x*=0, *y*=0, *relwidth*=1, *relheight*=1)  # Fill the window

tk.Label(root, *text*="Expense Tracker Login", *font*=("Arial", 14),*bd*=1, *relief*="solid").pack(*pady*=30)

tk.Label(root, *text*="User ID:").pack(*side*='top',*anchor*='nw',*padx*= 59)

userid\_entry = tk.Entry(root, *width*=30)

userid\_entry.pack(*pady*=5)

tk.Label(root, *text*="Password:").pack(*side*='top',*anchor*='nw',*padx*= 59)

password\_entry = tk.Entry(root, *show*="\*", *width*=30)

password\_entry.pack(*pady*=5)

tk.Button(root, *text*="Login", *command*=login, *width*=15,*bd*=3, *relief*="raised").pack(*pady*=20)

tk.Button(root, *text*="Create New Account", *command*=create\_account, *width*=20,*bd*=3, *relief*="raised").pack(*pady*=5)

root.mainloop()

**Main.py :**

import csv

import sys  # To accept user ID from Login.py

import tkinter as tk

from tkinter import messagebox, ttk

from datetime import date

import matplotlib.pyplot as plt

from matplotlib.figure import Figure

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

from PIL import Image, ImageTk

import subprocess

import tkinter.simpledialog as simpledialog

# CSV file setup

CSV\_FILE = 'expenses.csv'

FIELDS = ['id','date', 'category', 'amount', 'note', 'userid']

# Get the User ID passed from Login.py

if len(sys.argv) > 1:

    CURRENT\_USER = sys.argv[1]

else:

    CURRENT\_USER = None

def add\_expense():

    """Add new expense to CSV file"""

    if not CURRENT\_USER:

        messagebox.showerror("Error", "User ID is missing!")

        return

    date\_str = date\_entry.get().strip()

    category = category\_var.get()

    if category == 'Select Category':

        category = 'Other'

    if not date\_str:

        date\_str = date.today().strftime("%Y-%m-%d")

    note = note\_entry.get("1.0", "end-1c")

    amount = amount\_entry.get()

    if not amount.isdigit():

        messagebox.showerror("Invalid Input", "Amount must be a number!")

        return

    # Generate a new unique ID

    try:

        with open(CSV\_FILE, 'r') as csvfile:

            reader = csv.DictReader(csvfile)

            rows = list(reader)

            Id = int(rows[-1]['id']) + 1 if rows else 1

            formatted\_id = str(Id).zfill(3)

    except FileNotFoundError:

        formatted\_id = '001'

    # Write the new expense to the CSV file

    try:

        with open(CSV\_FILE, 'a', newline='') as csvfile:

            writer = csv.DictWriter(csvfile, fieldnames=['id', 'date', 'category', 'amount', 'note', 'userid'])

            if csvfile.tell() == 0:

                writer.writeheader()

            writer.writerow({

                'id': formatted\_id,

                'date': date\_str,

                'category': category,

                'amount': amount,

                'note': note,

                'userid': CURRENT\_USER

            })

        messagebox.showinfo("Success", "Expense added successfully")

    except Exception as e:

        messagebox.showerror("Error", f"Failed to add expense: {e}")

    calculate\_total()

    clear\_fields()

    csvfile.close()

def view\_expenses():

    """Display expenses for the logged-in user with dynamic Serial Numbers"""

    expense\_window = tk.Toplevel(root)

    expense\_window.title("Your Expenses")

    try:

        with open(CSV\_FILE, 'r') as csvfile:

            reader = csv.DictReader(csvfile)

            rows = [row for row in reader if row['userid'] == CURRENT\_USER]  # Filter by logged-in user ID

    except FileNotFoundError:

        rows = []

    # Setup the Treeview

    tree = ttk.Treeview(expense\_window, selectmode='browse')

    # Define columns

    tree['columns'] = ('S.No', 'ID', 'Date', 'Category', 'Amount', 'Note')

    # Format columns

    tree.column("#0", width=0, stretch=tk.NO)  # Hide the default first column

    tree.column("S.No", anchor=tk.W, width=50)

    tree.column("ID", anchor=tk.W, width=50)

    tree.column("Date", anchor=tk.W, width=120)

    tree.column("Category", anchor=tk.W, width=100)

    tree.column("Amount", anchor=tk.W, width=80)

    tree.column("Note", anchor=tk.W, width=250)

    # Create column headings

    tree.heading("#0", text='', anchor=tk.W)

    tree.heading("S.No", text='S.No', anchor=tk.W)

    tree.heading("ID", text='ID', anchor=tk.W)

    tree.heading("Date", text='Date', anchor=tk.W)

    tree.heading("Category", text='Category', anchor=tk.W)

    tree.heading("Amount", text='Amount', anchor=tk.W)

    tree.heading("Note", text='Note', anchor=tk.W)

    # Add data to the Treeview with dynamic S.No

    sno = 1

    for row in rows:

        tree.insert('', 'end', values=(sno, row['id'], row['date'], row['category'], row['amount'], row['note']))

        sno += 1

    # Pack the Treeview

    tree.pack(fill=tk.BOTH, expand=1, padx=10, pady=10)

    csvfile.close()

def delete\_expense():

    """Delete an expense based on the entered ID and user ID."""

    if not CURRENT\_USER:

        messagebox.showerror("Error", "User ID is missing!")

        return

    # Prompt the user to enter the ID of the expense to delete

    expense\_id = simpledialog.askstring("Delete Expense", "Enter the ID of the expense to delete:")

    if not expense\_id:

        messagebox.showerror("Error", "Expense ID is required!")

        return

    try:

        with open(CSV\_FILE, 'r') as csvfile:

            reader = csv.DictReader(csvfile)

            rows = list(reader)

        # Filter out the expense to delete

        updated\_rows = [row for row in rows if not (row['id'] == expense\_id and row['userid'] == CURRENT\_USER)]

        # Check if the expense with the given ID was found and deleted

        if len(rows) == len(updated\_rows):

            messagebox.showinfo("Not Found", "No expense found with the provided ID for the current user!")

            return

    except FileNotFoundError:

        messagebox.showerror("Error", "No expenses recorded yet!")

        return

    # Write the updated rows back to the CSV file

    try:

        with open(CSV\_FILE, 'w', newline='') as csvfile:

            writer = csv.DictWriter(csvfile, fieldnames=['id', 'date', 'category', 'amount', 'note', 'userid'])

            writer.writeheader()

            writer.writerows(updated\_rows)

        messagebox.showinfo("Success", "Expense deleted successfully!")

    except Exception as e:

        messagebox.showerror("Error", f"Failed to delete expense: {e}")

def edit\_expense():

    """Edit an expense based on the unique ID"""

    from tkinter import simpledialog

    if not CURRENT\_USER:

        messagebox.showerror("Error", "User ID is missing!")

        return

    # Ask the user for the ID of the expense to edit

    expense\_id = simpledialog.askstring("Edit Expense", "Enter the ID of the expense to edit:")

    if not expense\_id:

        return

    try:

        with open(CSV\_FILE, 'r') as csvfile:

            reader = csv.DictReader(csvfile)

            rows = list(reader)

        expense\_found = False

        for row in rows:

            if int(row['id']) == int(expense\_id) and row['userid'] == CURRENT\_USER:

                expense\_found = True

                # Prompt the user for new details

                new\_date = simpledialog.askstring("Edit Expense", "Enter new date (YYYY-MM-DD):", initialvalue=row['date'])

                new\_category = simpledialog.askstring("Edit Expense", "Enter new category:", initialvalue=row['category'])

                new\_amount = simpledialog.askfloat("Edit Expense", "Enter new amount:", initialvalue=row['amount'])

                new\_note = simpledialog.askstring("Edit Expense", "Enter new note:", initialvalue=row['note'])

                # Update the row

                row['date'] = new\_date or row['date']

                row['category'] = new\_category or row['category']

                row['amount'] = new\_amount or row['amount']

                row['note'] = new\_note or row['note']

                break

        if not expense\_found:

            messagebox.showerror("Error", "Expense not found or you don't have permission to edit it!")

            return

        # Save updated data back to the file

        with open(CSV\_FILE, 'w', newline='') as csvfile:

            writer = csv.DictWriter(csvfile, fieldnames=['id', 'date', 'category', 'amount', 'note', 'userid'])

            writer.writeheader()

            writer.writerows(rows)

        messagebox.showinfo("Success", "Expense updated successfully!")

    except FileNotFoundError:

        messagebox.showerror("Error", "Expense file not found!")

def calculate\_total():

    """Calculate total spend from CSV file for a specific user"""

    try:

        with open(CSV\_FILE, 'r') as csvfile:

            reader = csv.DictReader(csvfile)

            # Filter rows by the current user's ID and sum the 'amount' column

            total\_spend = sum(float(row['amount']) for row in reader if row['userid'] == CURRENT\_USER)

    except FileNotFoundError:

        total\_spend = 0

    total\_label.config(text=f"Total Spend: ₹{total\_spend:.2f}")

def clear\_fields():

    """Clear input fields"""

    category\_var.set("Select Category")

    amount\_entry.delete(0, tk.END)

    note\_entry.delete("1.0", tk.END)

    date\_entry.delete("0", tk.END)

def show\_category\_analysis():

    """Display pie chart for the current user's expenses"""

    if not CURRENT\_USER:

        messagebox.showerror("Error", "User ID is missing!")

        return

    try:

        with open(CSV\_FILE, 'r') as csvfile:

            reader = csv.DictReader(csvfile)

            rows = [row for row in reader if row['userid'] == CURRENT\_USER]

    except FileNotFoundError:

        rows = []

    category\_expenses = {}

    for row in rows:

        category = row['category']

        amount = float(row['amount'])

        category\_expenses[category] = category\_expenses.get(category, 0) + amount

    if not category\_expenses:

        messagebox.showinfo("Info", "No data to display!")

        return

    categories = list(category\_expenses.keys())

    amounts = list(category\_expenses.values())

    # Create a new window for the analysis

    analysis\_window = tk.Toplevel(root)

    analysis\_window.title("Category Analysis")

    analysis\_window.geometry('750x550')

    # Create a figure for the pie chart

    fig = Figure(figsize=(6, 6), dpi=100)

    ax = fig.add\_subplot(111)

    # Create the pie chart

    wedges, texts , autotexts= ax.pie(amounts, colors=plt.cm.tab20.colors[:len(categories)], startangle=90,autopct='%1.1f%%')

    ax.axis('equal')  # Equal aspect ratio ensures the pie is a circle

    ax.set\_title('Expense Distribution by Category')

    for autotext in autotexts:

        autotext.set\_color('black')

        autotext.set\_fontsize(9)

    # Add a legend

    ax.legend(wedges, categories, title="Categories", loc="center left", bbox\_to\_anchor=(-0.14,0, 0, 1))

    # Add the pie chart to the tkinter window

    canvas = FigureCanvasTkAgg(fig, master=analysis\_window)

    canvas.draw()

    canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

def logout():

    """Close the main window and reopen the login page."""

    root.destroy()

    subprocess.Popen(['python', 'Login.py'])

# Main UI

root = tk.Tk()

root.title("Expense Tracker")

# Load the background image

image\_path = 'cg3.png'

image = Image.open(image\_path)

# Resize the image if necessary to fit the window size

# Comment this section out if you want the image to retain its original size

window\_width = 550

window\_height = 450

image.thumbnail((window\_width, window\_height))  # Resize keeping aspect ratio

image = image.resize((window\_width, window\_height))  # Resize to exact fit (may distort image)

# Convert the image to a format tkinter can use

tk\_image = ImageTk.PhotoImage(image)

# Create a Label with the image

background\_label = tk.Label(root, image=tk\_image)

background\_label.image = tk\_image  # Keep a reference to prevent garbage collection

background\_label.place(x=0, y=0, relwidth=1, relheight=1)  # Fill the window

# Input Frames

input\_frame = tk.Frame(root,bg='#e2e9ec',width=500, height=300,bd=3, relief="raised")

input\_frame.pack(padx=30, pady=30,fill=tk.Y)

input\_frame.pack\_propagate(False)

tk.Label(input\_frame, text="Category:",bg='#D3D3D3').grid(column=0, row=0,pady=8,padx=10)

category\_var = tk.StringVar(input\_frame)

category\_var.set("Select Category")

category\_option = tk.OptionMenu(input\_frame, category\_var, "Food", "Transport", "Entertainment", "Other")

category\_option.grid(column=1, row=0)

tk.Label(input\_frame, text="Amount:",bg='#D3D3D3').grid(column=0, row=1,pady=12)

amount\_entry = tk.Entry(input\_frame)

amount\_entry.grid(column=1, row=1,pady=12,padx=5)

tk.Label(input\_frame, text="Date (YYYY-MM-DD):",bg='#D3D3D3').grid(column=0, row=2,padx= 10)

date\_entry = tk.Entry(input\_frame)

date\_entry.grid(column=1, row=2)

tk.Label(input\_frame, text="Note:",bg='#D3D3D3').grid(column=0, row=4,pady=1)

note\_entry = tk.Text(input\_frame, width=20, height=5)

note\_entry.grid(column=1, row=4,pady=10,padx=10)

# Buttons

button\_frame = tk.Frame(root,bg='#e2e9ec',bd=3, relief="raised")

button\_frame.pack(padx=10, pady=10)

add\_button = tk.Button(button\_frame, text="Add Expense", command=add\_expense,bd=3)

add\_button.pack(side=tk.LEFT, padx=5)

view\_button = tk.Button(button\_frame, text="View Expenses", command=view\_expenses,bd=3)

view\_button.pack(side=tk.LEFT, padx=5)

total\_button = tk.Button(button\_frame, text="Edit Expense", command=edit\_expense,bd=3)

total\_button.pack(side=tk.LEFT, padx=5)

clear\_button = tk.Button(button\_frame, text="Delete Expense", command=delete\_expense,bd=3)

clear\_button.pack(side=tk.LEFT, padx=5)

analysis\_button = tk.Button(button\_frame, text="Category Analysis", command=show\_category\_analysis,bd=3)

analysis\_button.pack(side=tk.LEFT, padx=5)

# Logout Button

tk.Button(root, text="Logout", command=logout, bg="#ffcccc", bd=3).pack(pady=10)

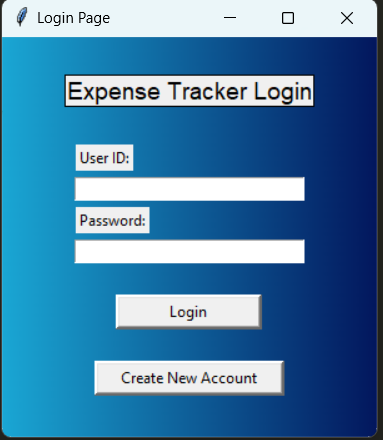
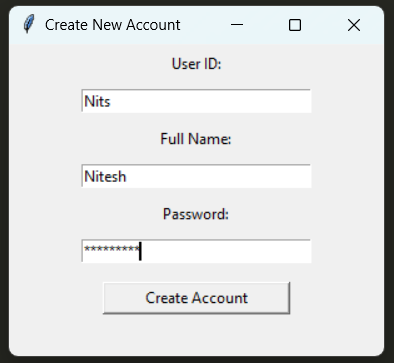
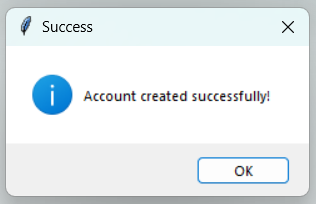
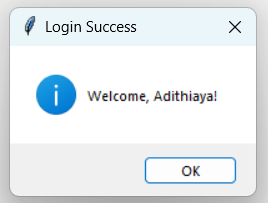
# Total Spend Label

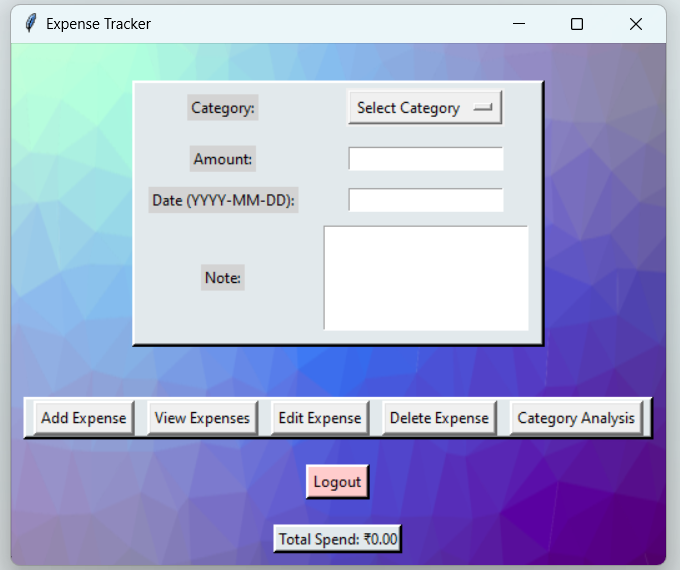
total\_label = tk.Label(root, text="Total Spend: ₹0.00",bg='#e2e9ec',bd=3, relief="raised")

total\_label.pack(pady=10)

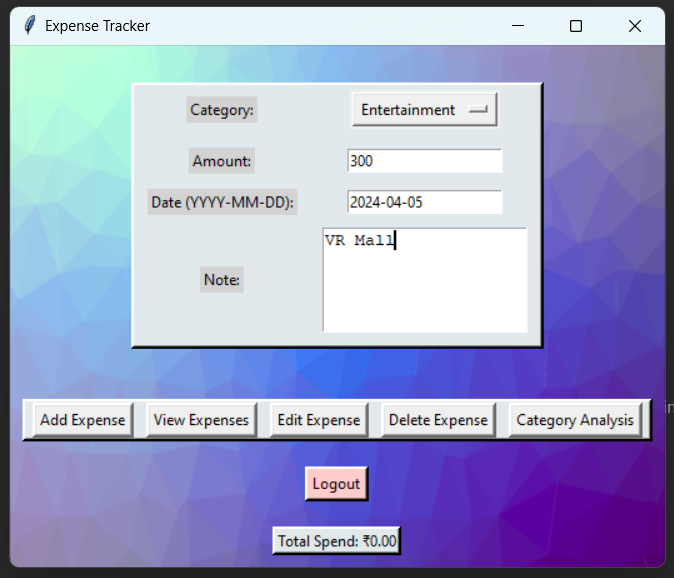
root.mainloop()

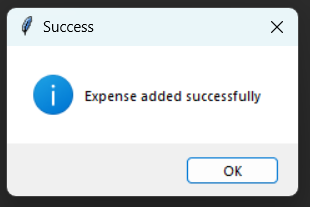
OUTPUT SCREEN

* This is the window, i.e., the Graphical User Interface (GUI) which gets opened on executing the source code.
* Let us now click on the Create New Account button to create an account and start the program.
* By Clicking Create Acoount Button, we now created a new user account.
* Now Click Ok on the dialog box.
* After entering correct login credentials, we shall click on the Login Button to sign in to main screen.
* Click Ok.
* This takes us to main screen:

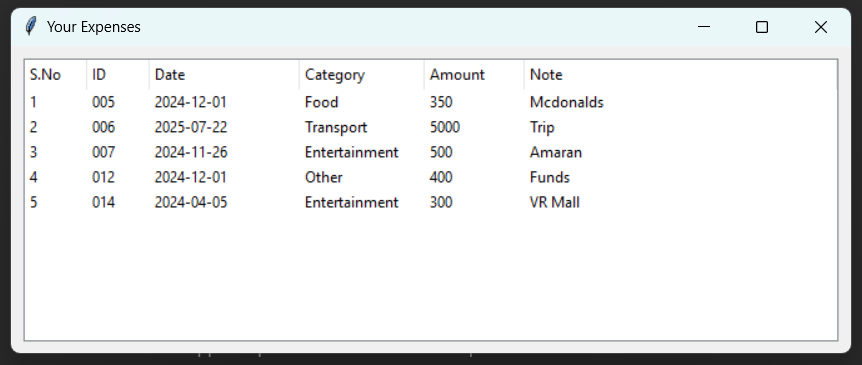


* Enter the necessary details and click Add Expense Button

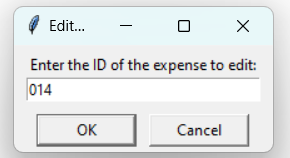
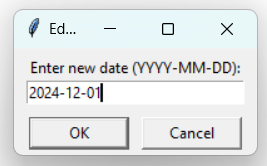


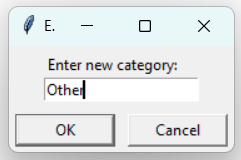
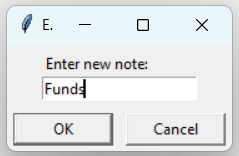
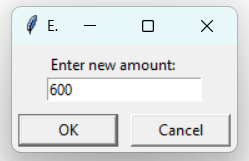


* After Clicking ok(notice the Total spend label changes) , click View expenses Button.
* This window shows all the expenses made by the user

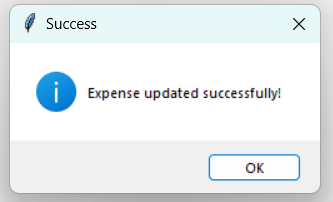


* Now click Edit expense button and fill all the details

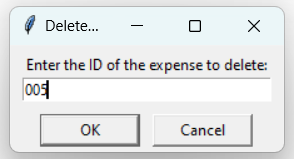


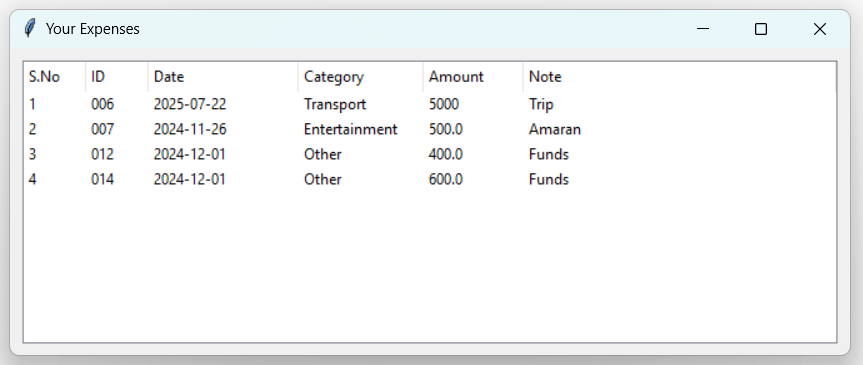


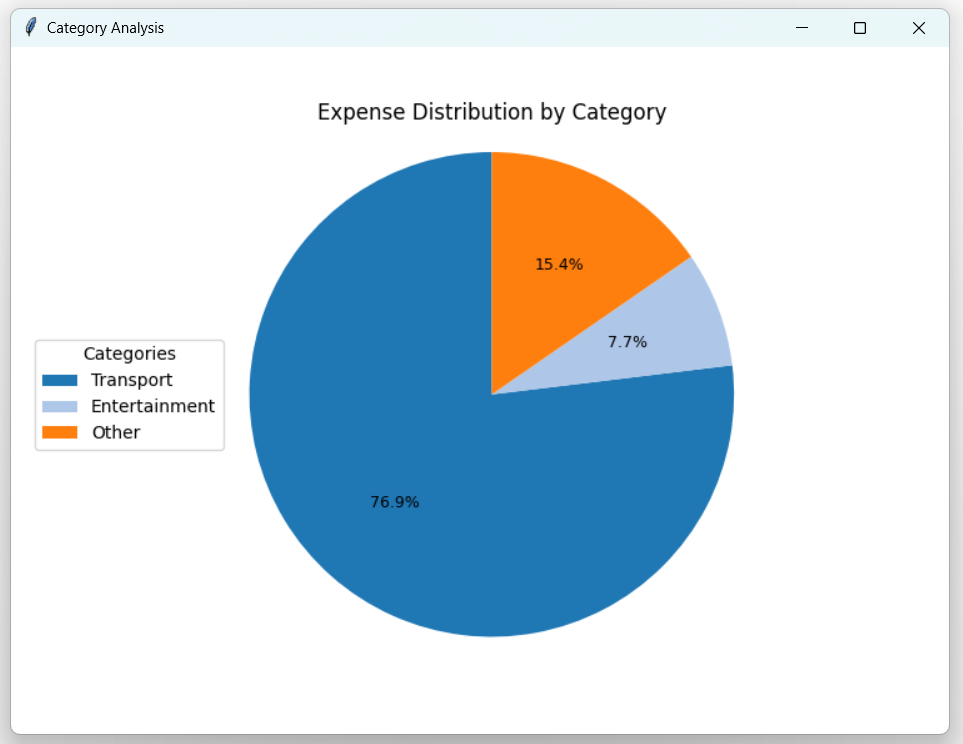
* This dialog box appears, click ok(this updates the expense)



* Next click Delete expense button and enter the Id of that expense which should be deleted. The expense is deleted from the csv file.



* Click view expense to View the changes made.
* To view the expense analysis based on the category, click Category analysis button



* Finally, clicking Logout button brings us back to Login page.

Bibliography

* <https://www.geeksforgeeks.org/computer-science-projects/>
* <https://www.youtube.com/watch?v=8exB6Ly3nx0>
* [https://www.youtube.com/](https://www.youtube.com/watch?v=VMP1oQOxfM0)
* <https://www.javatpoint.com/python-tkinter>
* <https://www.geeksforgeeks.org/python-tkinter-tutorial>
* <https://realpython.com/python-gui-tkinter/>