Singapore

FINANCE TRACKER

Computer Science Project

By: Chebrolu Naga Venkata Naha Gupta
Class: 12B

Singapore

COMPUTER SCIENCE PROJECT
CBSE Class - XII
Academic Year 2024-2025

Name: Chebrolu Naga Venkata Neha Gupta
Reg. No:

External Examiner
( )

Teacher in Charge

CERTIFICATE

This is to certify that Chebrolu Naga Venkata Naha Gupta of class XII has prepared the report on the project titled “Personal Finance Tracker” in accordance with the guidelines given by Central Board of Secondary Education. The report is found worthy of acceptance as final project report for Computer Science of class XII during the academic year 2024-2025.

Teacher: Ritu Arora

ACKNOWLEDGEMENT

It’s my humble pleasure to thank my Computer Science Teacher Ms. Ritu Arora, our management, Secretary and Principal who gave me the opportunity to do this wonderful project on the topic “Personal Finance Tracker” and for helping and guiding me throughout the completion of my project. This project has helped me in doing a lot of research enabling me to enhance my scientific skills.

Class XII

CONTENTS

INTRODUCTION

About Python

Python is an open-source easy to understand programming language with two popular versions: Python 2.7 and Python 3. Python recently ended support for it's 2.7 version which has been in use for many years by millions of developers. Python was programmed in ABC and modula-3 by Guido van Rossum in the year 1991. He named the language after his favourite TV show, Monty Python's Flying Circus.

The language has become popular due to it's simple syntax and wide range of libraries. Many consider Python to have replaced PHP in the open-source web framework application acronym LAMP (Linux, Apache, MySQL, PHP/Python/Pearl). Python has also picked up popularity in the fields of data science and data analysis and has also replaced MATLAB in several aspects.

Python, being a high-level interpreted language, has also become popular for artificial intelligence and machine learning in the past decade. There have been several libraries such as scipy-kit, keras, tensorflow, numpy, nlp, etc., developed for the purpose of machine learning in Python.

Tech-giants and Silicon Valley companies including Google, Facebook, Amazon, and Netflix use Python for several tasks. Google uses the Django Web framework (used in this project) to create a secured browsing experience for Billions.

Python is actively used by almost every device in one form or the other. On installation, Python comes with a default Programming IDE, named the IDLE. The IDLE is easy to use and recommended for beginners and even professionals for short programming tasks.

By default, a Python installation includes libraries like math, json, pickle, time, etc. However, the user can install additional libraries (modules) too using the 'pip install' command on the command line.

Features of Python

1. Python is a high-level language that uses an interpreter in contrast to a compiler used by traditional programming languages. This makes Python beginner friendly and accessible.
2. Python is an open-source language that is available on a wide range of Operating Systems including Windows, Linux, MacOS, UNIX, etc.
3. Python has a wide range of libraries with varying features that facilitate efficient application development.
4. Being an open source language, Python is frequently updated and maintained.
5. Python provides security to data and applications with its libraries.
6. Due to its high popularity, a wide range of Python documentation and developers are available for help.

About MySQL

MySQL is an open-source relational database management system developed by Oracle Corporation. MySQL is written in C and C++ and was founded by MySQL AB, a Swedish software company later acquired by Sun Microsystems (which was in turn acquired by Oracle Corporation).

MySQL was co-founded by Michael Widenius in 1995. The name MySQL is a combination of 'My', the name of Michael's daughter, and 'SQL', which stands for Structured Query Language. The dolphin in the MySQL Logo is named Sakila.

MySQL is a widely used database and stands for the 'M' in the acronym LAMP. Most Cloud providers including Amazon Web Services (AWS), Microsoft Azure, Oracle Cloud Infrastructure, Google Cloud, etc., provide database services using MySQL software. Most of the world's data is stored in MySQL due to its simplicity and security.

MySQL still supports two of its versions: MySQL 5.7 and MySQL 8.0. Although initially MySQL was designed for a command line interface, there are several GUI interfaces such as MySQL workbench available now.

Features of MySQL

1.MySQL is popular among databases for its simplicity. Users interact with the database software using SQL commands which are like simple English sentences and not case sensitive.
2.MySQL software is secure. Due to its high security, it is also used by governments and tech-giants across the world.
3.MySQL is available on several Operating Systems including Linux, Windows, UNIX and MacOS.
4.MySQL is localized, i.e., it provides support in several languages.

ABOUT THE PROJECT

1. Purpose:
    - A Python-based expense tracker application that uses Tkinter for GUI, MySQL for database management, and Pandas/Matplotlib/Seaborn for data visualization.

2. Modules Used:
    - tkinter: For creating the graphical user interface.
    - mysql.connector: For connecting and interacting with the MySQL database.
    - pandas: For data manipulation and analysis.
    - matplotlib & seaborn: For creating visualizations of the expenses

3. Core Functions:
    - connect(): Establishes a connection with the MySQL database.
    - add_category(con, name): Adds a new expense category to the database.
    - add_transaction(con, date, description, amount, category_id): Adds a new transaction (date, description, amount, and category) to the database.
    - get_transactions(con): Retrieves all transaction data, joining transactions and categories tables.
    - visualize_transactions(df): Generates two types of visualizations:
        - Monthly expense trends.
        - Expense distribution by category

4. Database Details:
    - Categories Table: Stores the names of expense categories.
    - Transactions Table: Contains details about each transaction, linked to a category using category_id.

5. GUI Features:
    - Add new categories using a simple input form.
    - Add new transactions by providing date, description, amount, and category ID.
    - View and analyze transactions with charts:
        - Monthly trends.
        - Expense distribution by category.

6. Key Elements of the GUI:
    - Input Fields: For adding categories and transactions.

- Buttons:
  - To submit category or transaction data.
  - To visualize all transaction data.

- Error Handling: Uses messagebox to show success or error alerts.

7. How It Works:

- The main function connects to the database and initializes the GUI.
- User interacts with the GUI to add or view expense data.
- Data is fetched, processed, and visualized with Pandas and Seaborn.

8. Usage:

- MySQL Database: Requires an existing database with appropriate tables (categories and transactions).

- Run the Script: Execute the program to launch the GUI and interact with the expense tracker.

This project integrates data management, visualization, and user interface development for a practical expense-tracking application.

SOURCE CODE

```
import tkinter as tk
from tkinter import messagebox
import mysql.connector
from mysql.connector import Error
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# Function to connect to MySQL database
def connect():
    try:
        con = mysql.connector.connect(
            host='localhost',
            database='prj',
            user='root',
            password='MySQL123!'
        )
        if con.is_connected():
            print('Connected to MySQL database')
            return con
    except Error as e:
        print(e)
    return None


# Function to add a category
def add_category(con, name):
    try:
        cursor = con.cursor()
        sql = "INSERT INTO categories (name) VALUES (%s)"
        cursor.execute(sql, (name,))
        con.commit()
        messagebox.showinfo("Success", "Category added successfully!")
    except Error as e:
        messagebox.showerror("Error", f"Failed to add category: {e}")


# Function to add a transaction
def add_transaction(con, date, description, amount, category_id):
    try:
        cursor = con.cursor()
```

```
        sql = """INSERT INTO transactions
                (date, description, amount, category_id)
                VALUES (%s, %s, %s, %s)"""
        cursor.execute(sql, (date, description, amount, category_id))
        con.commit()
        messagebox.showinfo("Success", "Transaction added successfully!")
    except Error as e:
        messagebox.showerror("Error", f"Failed to add transaction: {e}")


# Function to fetch all transactions
def get_transactions(con):
    try:
        query = """
        SELECT t.id, t.date, t.description, t.amount, c.name as category
        FROM transactions t
        JOIN categories c ON t.category_id = c.id
        """
        df = pd.read_sql(query, con)
        return df

    except Error as e:
        messagebox.showerror("Error", f"Failed to fetch data: {e}")
        return None


# Function to visualize transactions
def visualize_transactions(df):
    df['date'] = pd.to_datetime(df['date'])
    df['month'] = df['date'].dt.to_period('M')
    monthly_expenses = df.groupby('month')['amount'].sum().reset_index()

    plt.figure(figsize=(10, 6))
    sns.barplot(x='month', y='amount', data=monthly_expenses)
    plt.title('Total Expenses Per Month')
    plt.xlabel('Month')
    plt.ylabel('Total Amount')
    plt.xticks(rotation=45)
    plt.show()

    plt.figure(figsize=(10, 6))
    sns.barplot(x='category', y='amount', data=df)
    plt.title('Expenses by Category')
    plt.xlabel('Category')
```

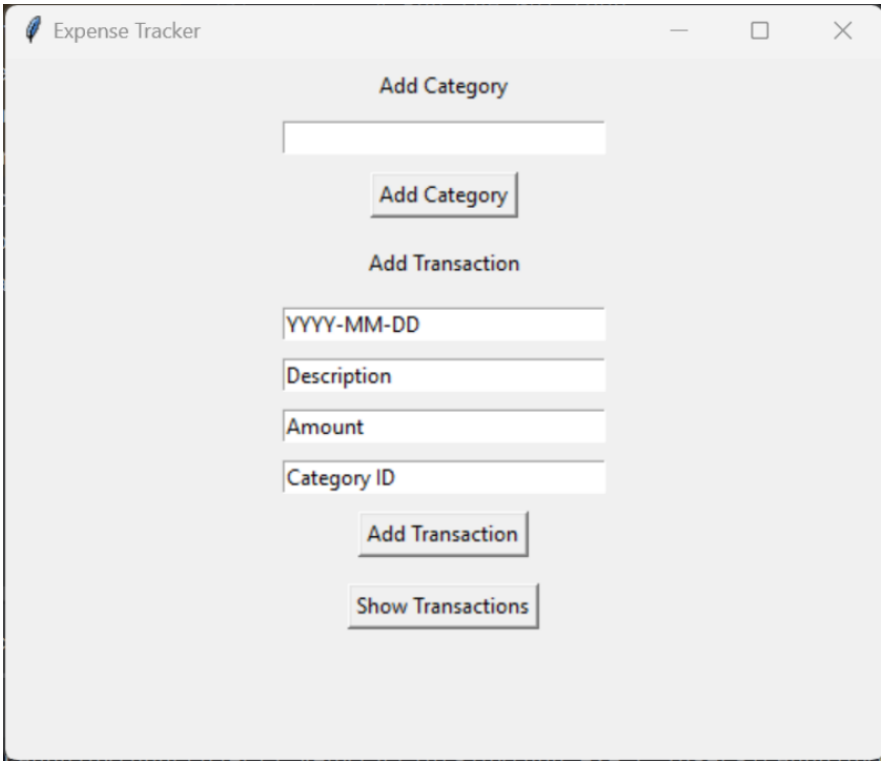
```
    plt.ylabel('Amount')
    plt.xticks(rotation=45)
    plt.show()


# GUI Setup
def setup_gui(con):
    root = tk.Tk()
    root.title("Expense Tracker")
    root.geometry("500x400")

    # Add Category Section
    tk.Label(root, text="Add Category").pack(pady=5)
    category_entry = tk.Entry(root, width=30)
    category_entry.pack(pady=5)

    def add_category_gui():
        category = category_entry.get()

        if category:
            add_category(con, category)
        else:
            messagebox.showwarning("Input Error", "Please enter a category name.")

    tk.Button(root, text="Add Category", command=add_category_gui).pack(pady=5)

    # Add Transaction Section
    tk.Label(root, text="Add Transaction").pack(pady=10)

    date_entry = tk.Entry(root, width=30)
    date_entry.insert(0, "YYYY-MM-DD")
    date_entry.pack(pady=5)

    description_entry = tk.Entry(root, width=30)
    description_entry.insert(0, "Description")
    description_entry.pack(pady=5)

    amount_entry = tk.Entry(root, width=30)
    amount_entry.insert(0, "Amount")
    amount_entry.pack(pady=5)

    category_id_entry = tk.Entry(root, width=30)
    category_id_entry.insert(0, "Category ID")
    category_id_entry.pack(pady=5)
```

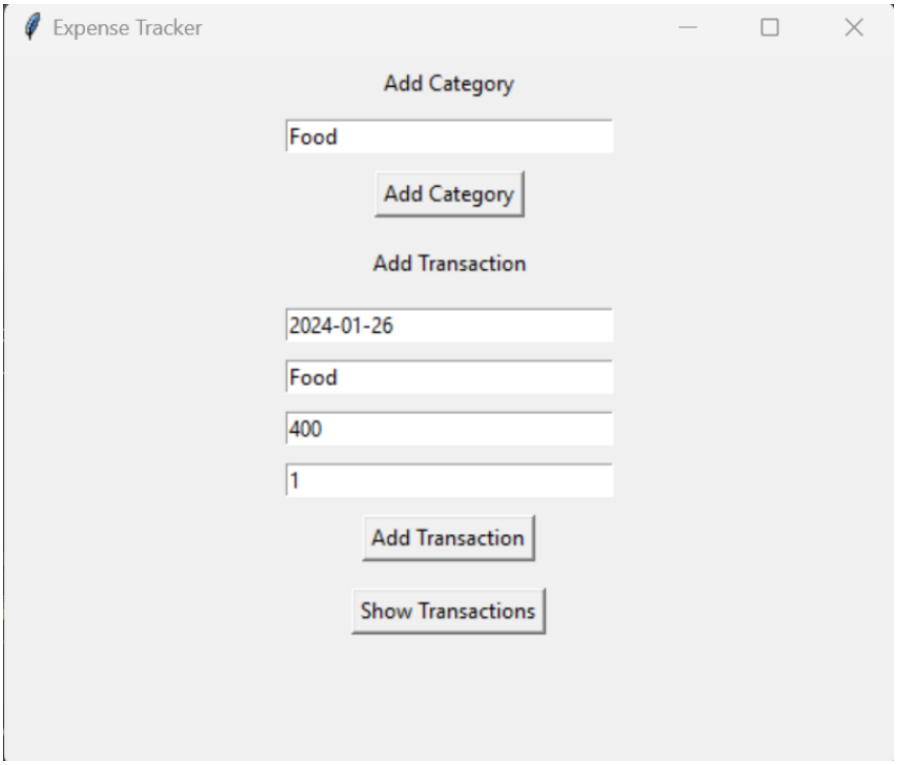
```
    def add_transaction_gui():
        date = date_entry.get()
        description = description_entry.get()
        amount = amount_entry.get()
        category_id = category_id_entry.get()

        if date and description and amount and category_id:
            try:
                amount = float(amount)
                category_id = int(category_id)
                add_transaction(con, date, description, amount, category_id)

            except ValueError:
                messagebox.showwarning("Input Error", "Please enter valid data types.")
        else:
            messagebox.showwarning("Input Error", "Please fill in all fields.")

    tk.Button(root, text="Add Transaction", command=add_transaction_gui).pack(pady=5)

    # Display and Visualize Transactions
    def show_transactions():
        df = get_transactions(con)

        if df is not None:
            print(df)
            visualize_transactions(df)

    tk.Button(root, text="Show Transactions", command=show_transactions).pack(pady=10)

    # Run the GUI loop
    root.mainloop()


# Main Function
def main():
    con = connect()

    if con:
        setup_gui(con)
        con.close()

if __name__ == "__main__":
    main()
```

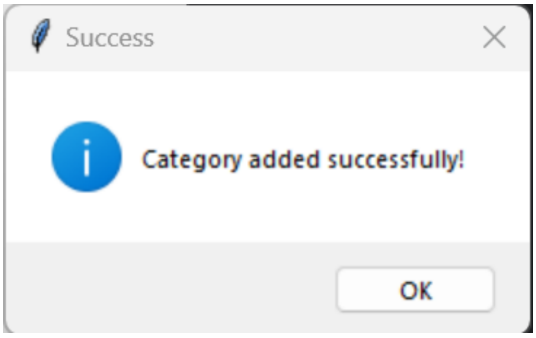
SQL:

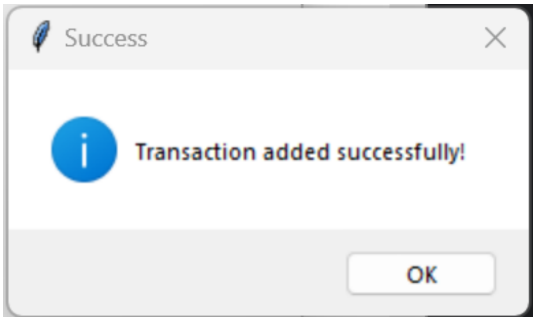
```
CREATE TABLE categories (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100)
);
CREATE TABLE transactions (
   id INT AUTO_INCREMENT PRIMARY KEY,
   date DATE,
   description VARCHAR(255),
   amount DECIMAL(10, 2),
   category_id INT,
   FOREIGN KEY (category_id) REFERENCES categories(id)
);
```

PROJECT


Expense Tracker
Add Category
Add Category
Add Transaction
YYYY-MM-DD
Description
Amount
Category ID
Add Transaction
Show Transactions


Expense Tracker
Add Category
Food
Add Category
Add Transaction
2024-01-26
Food
400
1
Add Transaction
Show Transactions


Success
Category added successfully!
OK


Success
Transaction added successfully!
OK

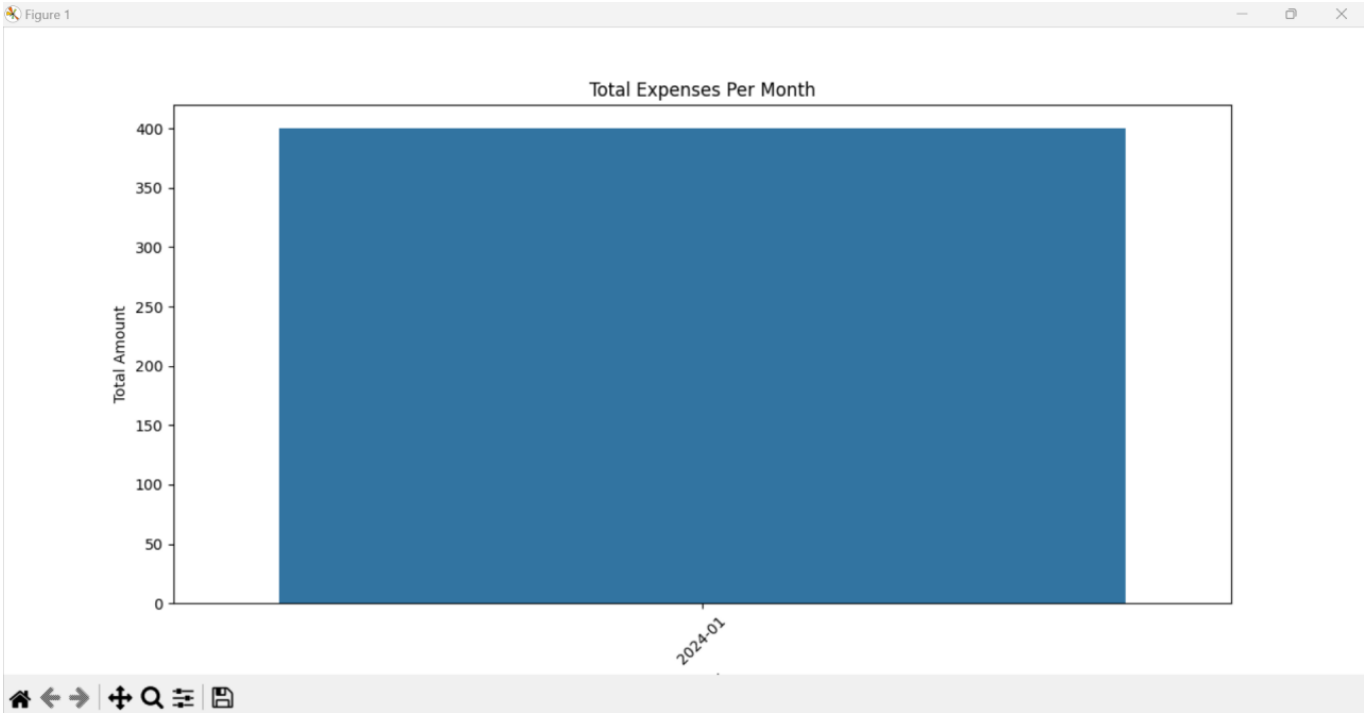
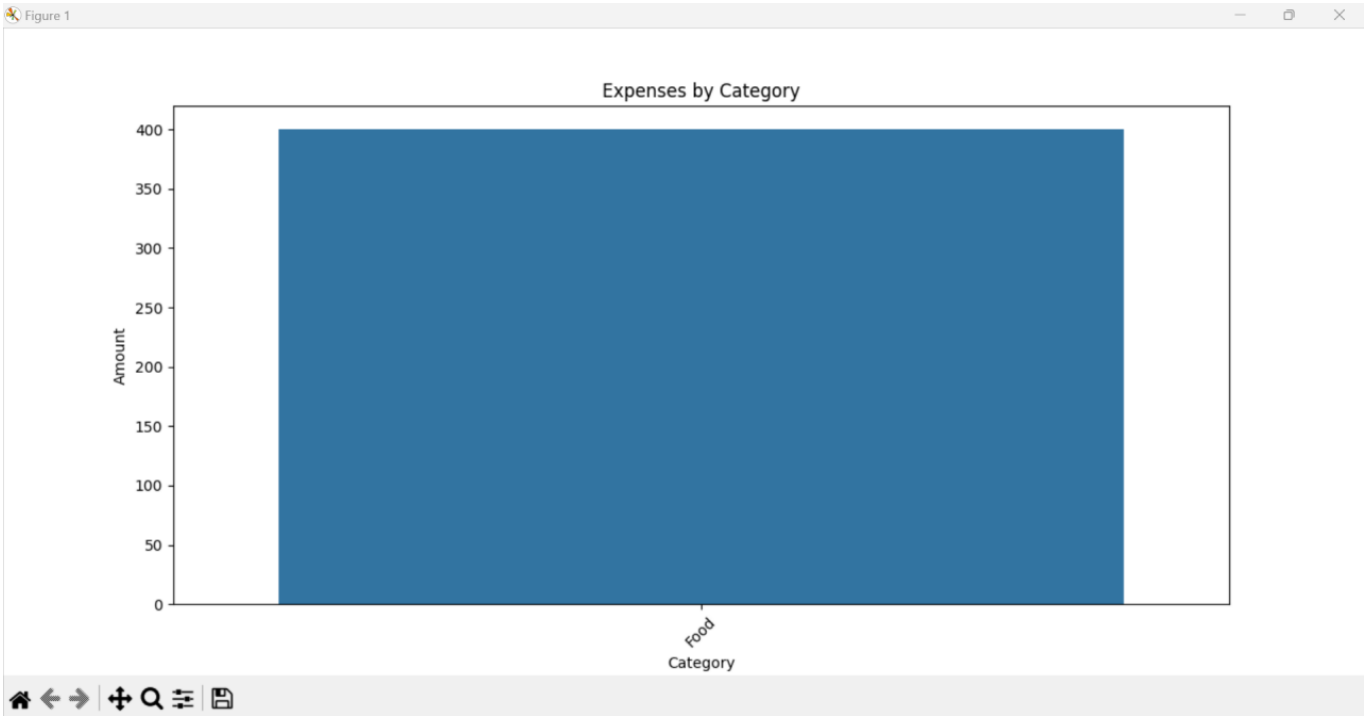
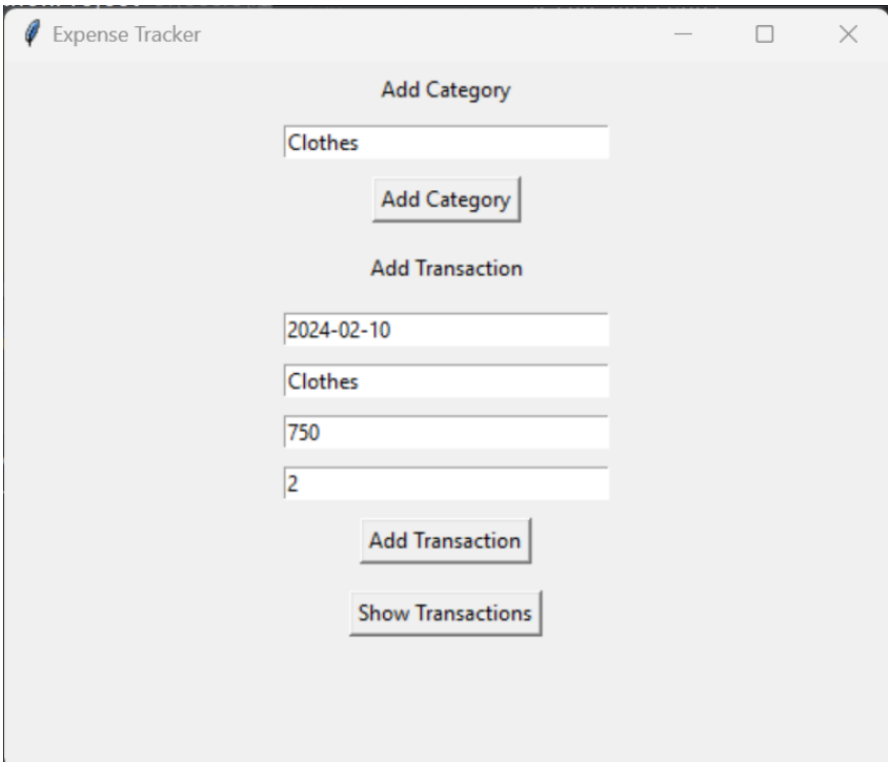
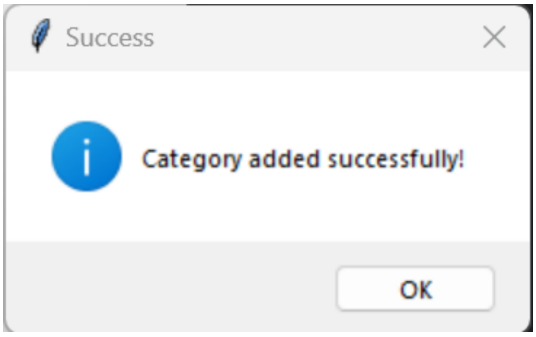


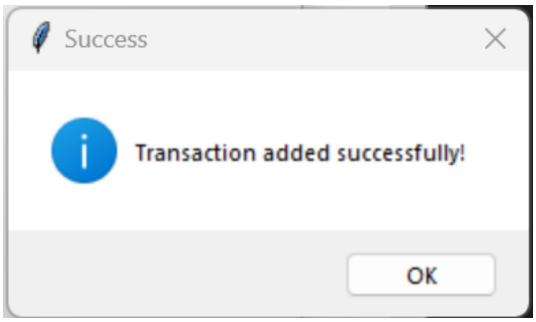
Figure 1
Total Expenses Per Month
Total Amount
400
350
300
250
200
150
100
50
0
2024-01




```
C:\Users\raghu\pythonProject\.venv\Scripts\python.exe C:\Users\raghu\pythonProject\project.py
Connected to MySQL database
C:\Users\raghu\pythonProject\project.py:57: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database str
  df = pd.read_sql(query, con)
   id        date description  amount category
0   1  2024-01-26        Food   400.0     Food
```




Success
Category added successfully!
OK


Success
Transaction added successfully!
OK

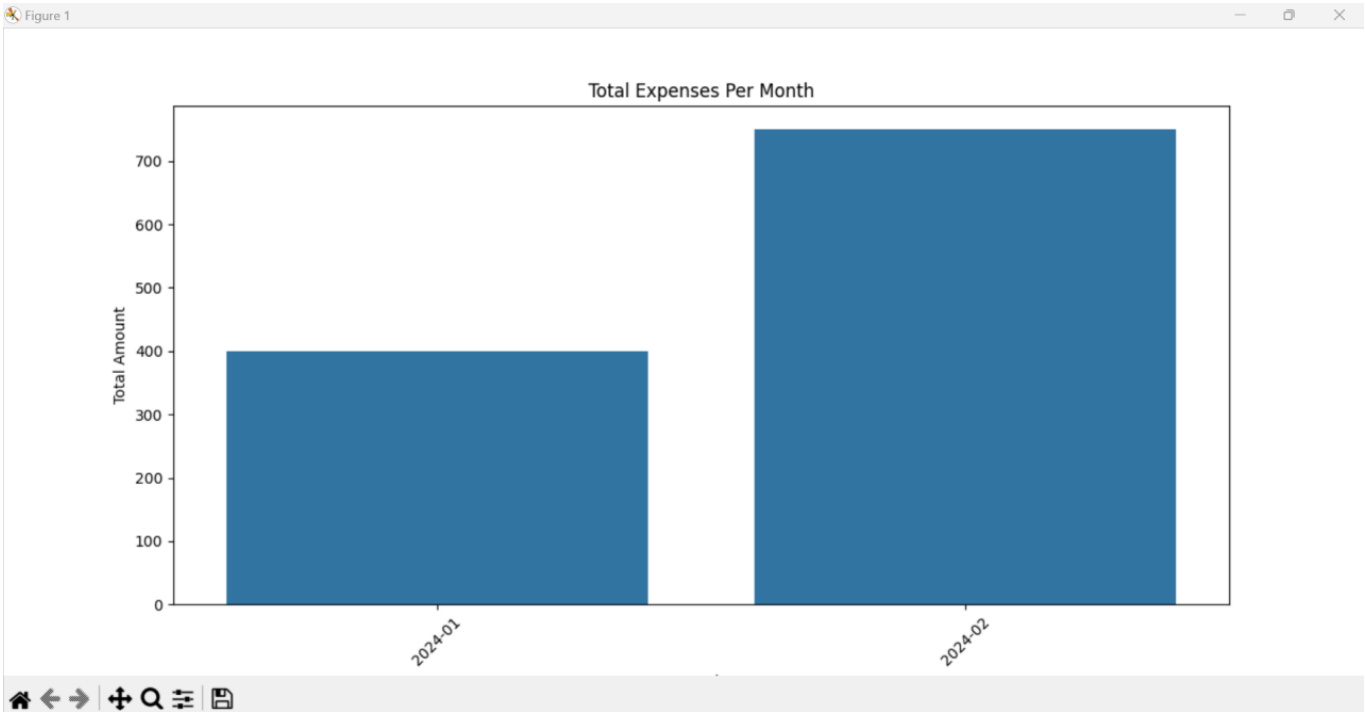
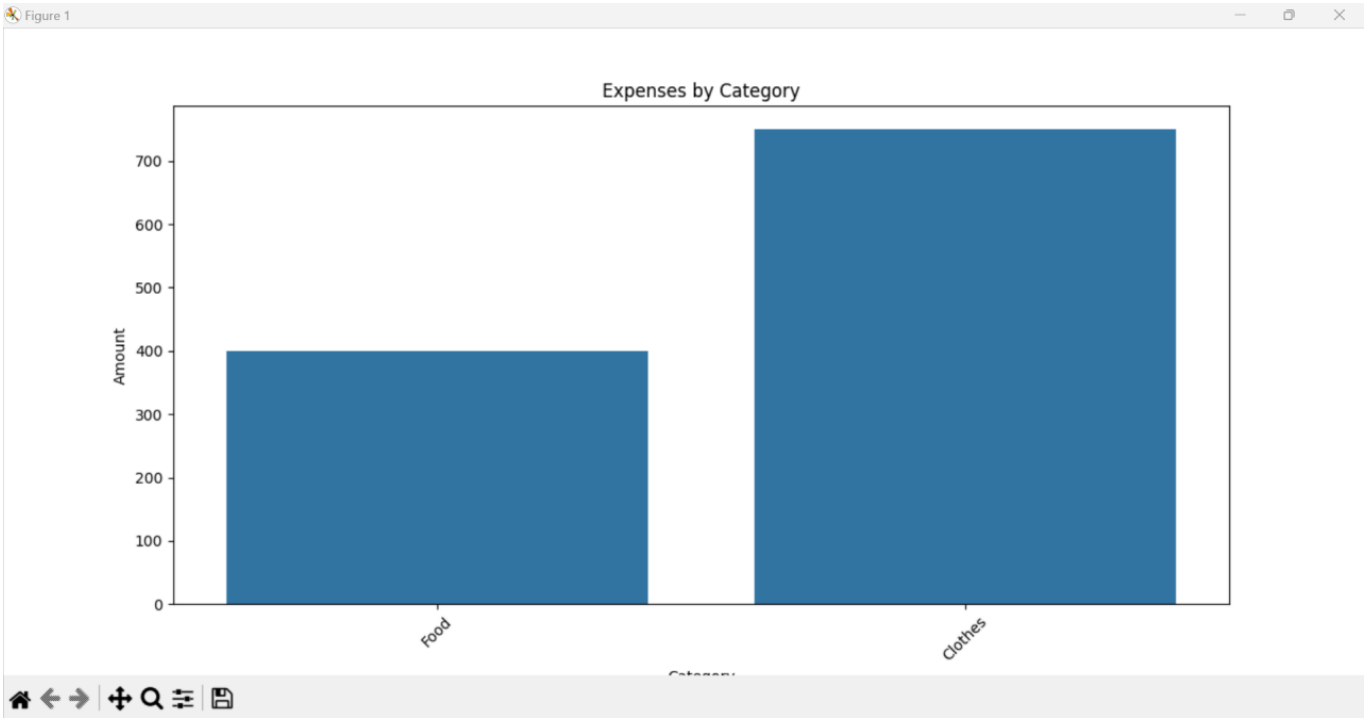
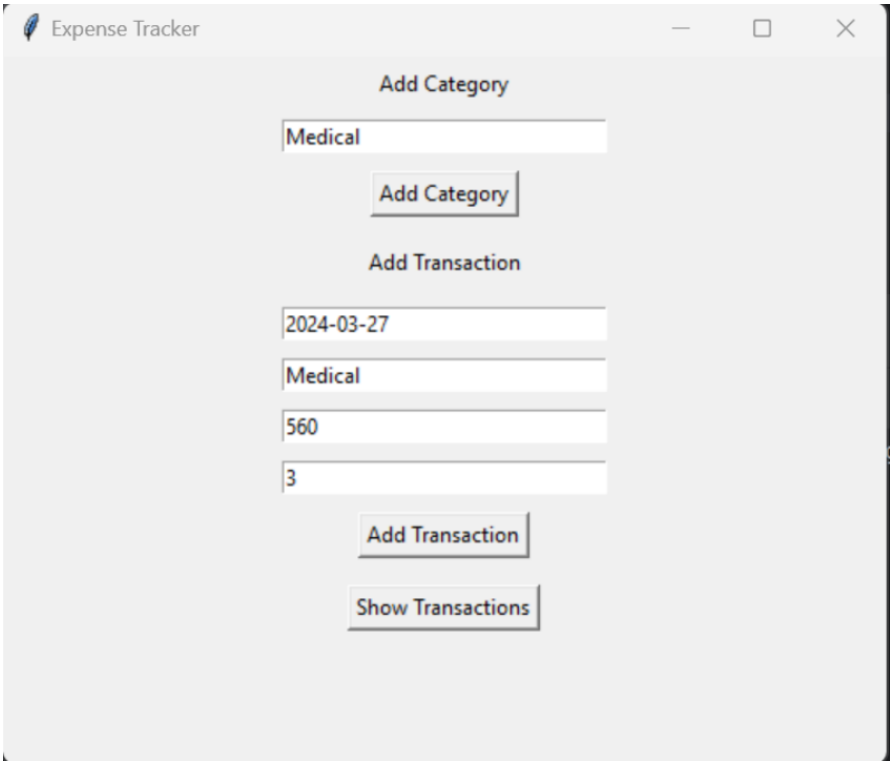
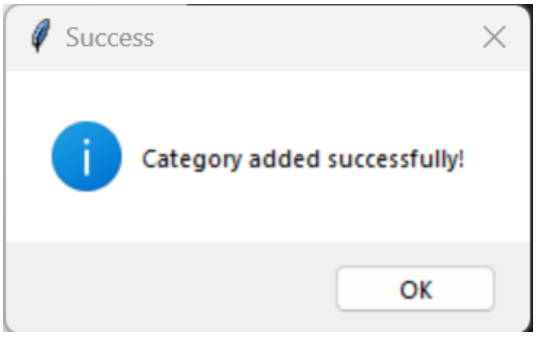


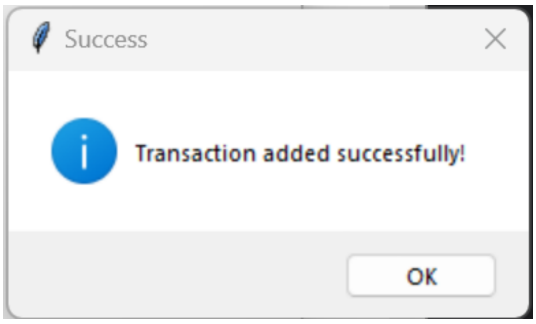
Figure 1
Total Expenses Per Month
Total Amount
0
100
200
300
400
500
600
700
2024-01
2024-02




```
C:\Users\raghu\pythonProject\.venv\Scripts\python.exe C:\Users\raghu\pythonProject\project.py
Connected to MySQL database
C:\Users\raghu\pythonProject\project.py:57: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string U
  df = pd.read_sql(query, con)
   id        date description  amount category
0   1  2024-01-26        Food   400.0     Food
1   3  2024-02-10     Clothes   750.0  Clothes
```




Success
Category added successfully!
OK


Success
Transaction added successfully!
OK

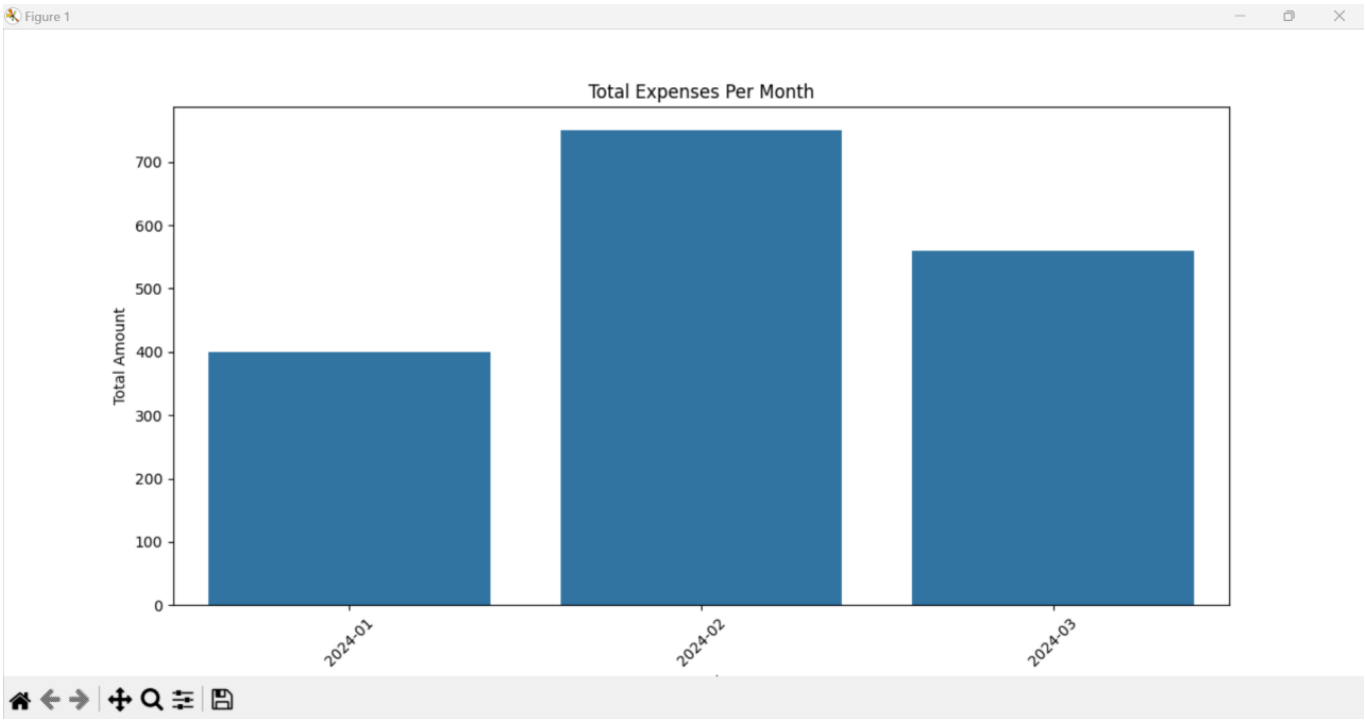
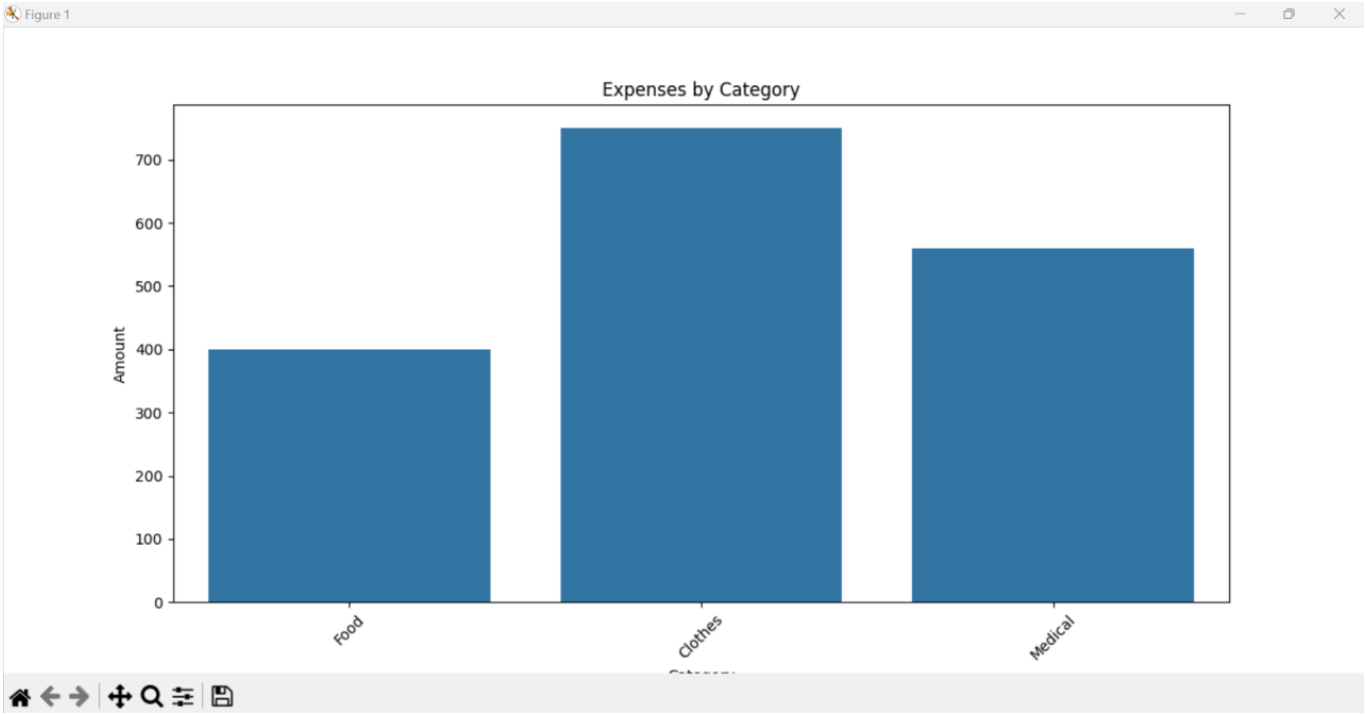


Figure 1
Total Expenses Per Month
Total Amount
0
100
200
300
400
500
600
700
2024-01
2024-02
2024-03




```
C:\Users\raghu\pythonProject\.venv\Scripts\python.exe C:\Users\raghu\pythonProject\project.py
Connected to MySQL database
C:\Users\raghu\pythonProject\project.py:57: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string
  df = pd.read_sql(query, con)
   id        date description  amount category
0   1  2024-01-26        Food   400.0     Food
1   3  2024-02-10     Clothes   750.0  Clothes
2   4  2024-03-27     Medical   560.0  Medical
```

FUTURE ENHANCEMENTS

User Authentication and Permissions (Security)
- Feature: Implement user authentication (login/signup) with role-based access control (admin, user).
- Enhancement: Add password hashing, session management, and multi-factor authentication (MFA).
- Technology: Use libraries like Flask, Django, or integrate third-party services like OAuth for secure authentication.

Real-Time Data Updates
- Feature: Implement real-time data updating for transactions and account balance tracking.
- Enhancement: Use web sockets to update the UI in real-time or periodically fetch data from external APIs (e.g., exchange rates or financial market data).
- Technology: Socket.IO or WebSockets for real-time communication, periodic fetching with schedule or APScheduler.

Mobile App or Web Interface
- Feature: Create a mobile or web-based version of your finance tracker.
- Enhancement: Develop a mobile app using frameworks like Kivy or React Native, or build a web interface using Flask/Django.
- Technology: Flask, Django, Kivy, or React Native

Transaction Reminders and Notifications
- Feature: Add reminders for bill payments, subscriptions, or other financial deadlines.
- Enhancement: Implement email or SMS notifications for upcoming payments or recurring bills.
- Technology: Twilio (for SMS), smtplib (for email), APScheduler (for scheduling reminders).

Backup and Data Recovery
- Feature: Add features for automatic backups and data recovery.
- Enhancement: Regular backups of data to cloud storage or local servers to ensure data integrity.
- Technology: Cloud storage solutions like Google Drive, AWS S3, or simple SQLite backups.

BIBLIOGRAPHY

https://pandas.pydata.org/
https://matplotlib.org/
https://seaborn.pydata.org/
https://docs.python.org/3/library/tk.html