Expense Tracker

Python Programming Internship Project

Submitted By: P Charan teja
Submission Date: 25/02/2025

1. Project Overview

The Expense Tracker project is designed to reinforce your understanding of Python programming concepts and enhance your skills in building practical applications. In this project, you will be developing an Expense Tracker application that allows users to manage their expenses efficiently. This real-world application will involve handling data, user input, and implementing key

2. Project Objective

To accurately outline the scope of work required for a project, it is crucial to first identify its objectives. Pinpointing what the project hopes to accomplish will assist in determining its inclusions and limitations.

User Input and Data Management: Develop a system that allows users to input their daily expenses.
Data Storage: Implement a mechanism to store and manage the entered expense data.
Expense Categories: Categorize expenses into different categories for better organization.
Data Analysis: Provide users with insights into their spending patterns, such as monthly summaries and category-wise expenditure.
User-Friendly Interface: Create a user-friendly interface for a seamless user experience.
Error Handling: Implement error handling to ensure the application can handle unexpected.

3. Project Structure

1. User Input: Users can input their daily expenses, including the amount and a brief description.
2. Data Storage: Expenses will be stored in a file (e.g., expenses.txt).
3. Expense Categories: Users can categorize expenses (e.g., Food, Transportation, Entertainment).
4. Data Analysis: Users can view monthly summaries and category-wise expenditure.
5. User Interface: A simple command-line interface will be provided.
6. Error Handling: The application will handle invalid inputs gracefully.
7. Documentation: The code will be well-documented.

4. Python Source Code

python

```
import datetime
import os

# File to store expenses
EXPENSE_FILE = "expenses.txt"

# Expense categories
CATEGORIES = ["Food", "Transportation", "Entertainment", "Utilities", "Other"]

def load_expenses():
"""Load expenses from the file."""
expenses = []
if os.path.exists(EXPENSE_FILE):
with open(EXPENSE_FILE, "r") as file:
for line in file:
date, category, amount, description = line.strip().split("|")
expenses.append({
"date": date,
"category": category,
"amount": float(amount),
"description": description
})
return expenses
```

```
def save_expenses(expenses):
"""Save expenses to the file."""
with open(EXPENSE_FILE, "w") as file:
for expense in expenses:
file.write(f"{expense['date']}|{expense['category']}|{expense['amount']}|{expense['descripti
on']}\n")


def add_expense():
"""Add a new expense."""
print("\n--- Add New Expense ---")
date = input("Enter the date (YYYY-MM-DD): ")
try:
datetime.datetime.strptime(date, "%Y-%m-%d")
except ValueError:
print("Invalid date format. Please use YYYY-MM-DD.")
return

print("Select a category:")
for i, category in enumerate(CATEGORIES, 1):
print(f"{i}. {category}")
category_choice = int(input("Enter the category number: "))
if category_choice < 1 or category_choice > len(CATEGORIES):
print("Invalid category choice.")
return
```

```
category = CATEGORIES[category_choice - 1]

amount = float(input("Enter the amount spent: "))
description = input("Enter a brief description: ")

expenses = load_expenses()
expenses.append({
"date": date,
"category": category,
"amount": amount,
"description": description
})
save_expenses(expenses)
print("Expense added successfully!")

def view_monthly_summary():
"""View monthly summary of expenses."""
print("\n--- Monthly Summary ---")
month = input("Enter the month (YYYY-MM): ")
expenses = load_expenses()
monthly_expenses = [e for e in expenses if e["date"].startswith(month)]
if not monthly_expenses:
print("No expenses found for this month.")
return
```

```
total = sum(e["amount"] for e in monthly_expenses)
print(f"\nTotal expenses for {month}: ${total:.2f}")

print("\nCategory-wise expenditure:")
category_totals = {}
for e in monthly_expenses:
category_totals[e["category"]] = category_totals.get(e["category"], 0) + e["amount"]
for category, amount in category_totals.items():
print(f"{category}: ${amount:.2f}")

def view_all_expenses():
"""View all expenses."""
print("\n--- All Expenses ---")
expenses = load_expenses()
if not expenses:
print("No expenses found.")
return

for expense in expenses:
print(f"{expense['date']}
| {expense['category']} | $
{expense['amount']:.2f} |
{expense['description']}")

def main():
```

```
"""Main function to run the Expense Tracker."""
while True:
print("\n--- Expense Tracker ---")
print("1. Add Expense")
print("2. View Monthly Summary")
print("3. View All Expenses")
print("4. Exit")
choice = input("Enter your choice: ")

if choice == "1":
add_expense()
elif choice == "2":
view_monthly_summary()
elif choice == "3":
view_all_expenses()
elif choice == "4":
print("Exiting the Expense Tracker. Goodbye!")
break
else:
print("Invalid choice. Please try again.")

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

Explanation of the Code

1. Data Storage:

- Expenses are stored in a text file (expenses.txt) in the format: date|category|amount|description.
- The load_expenses() function reads expenses from the file, and the save_expenses() function writes expenses to the file.

2. User Input:
- The add_expense() function allows users to input a new expense, including the date, category, amount, and description.
- The date is validated using datetime.datetime.strptime() to ensure it is in the correct format (YYYY-MM-DD).

3. Expense Categories:
- Users can select a category from a predefined list (CATEGORIES).

4. Data Analysis:
- The view_monthly_summary() function calculates the total expenses for a given month and provides a category-wise breakdown.
- The view_all_expenses() function displays all expenses stored in the file.

5. Error Handling:
- The application handles invalid date formats and category choices gracefully.

6. User Interface:
- A simple command-line interface is provided, allowing users to add expenses, view summaries, and exit the application.

How to Run the Code

1. Save the code in a file named expense_tracker.py.
2. Run the file using Python:
```
bash
python expense_tracker.py
```

3. Follow the on-screen instructions to add expenses, view summaries, and analyze spending patterns.

Sample Output

```
--- Expense Tracker ---
```

```
1. Add Expense

2. View Monthly Summary

3. View All Expenses

4. Exit

Enter your choice: 1

--- Add New Expense ---

Enter the date (YYYY-MM-DD): 2023-10-15

Select a category:

1. Food

2. Transportation

3. Entertainment

4. Utilities

5. Other

Enter the category number: 1

Enter the amount spent: 25.50

Enter a brief description: Lunch at Cafe

Expense added successfully!

--- Expense Tracker ---

1. Add Expense

2. View Monthly Summary

3. View All Expenses

4. Exit

Enter your choice: 2


--- Monthly Summary ---
```

```
Enter the month (YYYY-MM): 2023-10


Total expenses for 2023-10: $25.50


Category-wise expenditure:

Food: $25.50
```

Requirements and Features

User Input: Allow users to input their daily expenses, including the amount spent and a brief description.

Data Storage: Use appropriate data structures or file handling techniques to store and retrieve expense data.

Expense Categories: Implement the ability for users to categorize their expenses (e.g., food, transportation, entertainment).

Data Analysis: Provide users with the option to view summaries of their monthly expenses and category-wise expenditure.

User Interface: Create a simple and intuitive user interface to interact with the Expense Tracker.

Error Handling: Include error handling mechanisms to address potential issues during user interaction.

Documentation: Provide clear documentation for your code, explaining the logic behind key functions and overall program structure

Conclusion

This implementation provides a fully functional *Expense Tracker* application that meets all the project requirements. You can further enhance it by adding features like graphical user interfaces (GUI) using libraries like Tkinter or PyQt.