EXPENSE TRACKER

A MINI-PROJECT REPORT

18CSC207J - ADVANCED PROGRAMMING PRACTICE

Submitted by

Ritika PalChaudhuri (RA2111026010496)
Aryan Bali (RA2111026010510)
Monika Maradana (RA2111033010158)

Under the guidance of

Dr. Arun C

Assistant Professor, Department of Computer Science and Engineering

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR - 603203
Chengalpattu District

BONAFIDE CERTIFICATE

Certified that Mini Project report titled “Expense Tracker” is the bona fide work of Ritika PalChaudhuri(RA2111026010496), Aryan Bali(RA2111026010510) and Monika Maradana(RA2111033010158) who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project reportor dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Arun C
GUIDE
Assistant Professor
Department of Computing Technologies

SIGNATURE

Dr. Annie Uthra
HEAD OF THE DEPARTMENT
Professor & Head
Department of Computing Technologies

Index

Abstract

Expense Tracker is a software application that helps users keep track of their expenses. The application provides a user interface to add, view and delete expenses. The application is built using the Python programming language and uses the Tkinter GUI toolkit to create the user interface. The data is stored in an SQLite3 database. This project report will describe the design, implementation and testing of the Expense Tracker. The application has a main window that displays a table with the expenses. The table has columns for the expense description, amount, date and category. The user can add new expenses by clicking on a button that opens a dialog box with fields to enter the expense information. The user can edit an existing expense by selecting it in the table and clicking on a button that opens a dialogue box with the fields pre-populated with the existing values. The user can delete an existing expense by selecting it in the table and clicking on a button.

Modules

The application is implemented using the Python programming language and uses the following libraries:

Tkinter: to create the GUI

SQLite3: to store the data in a database

The application has the following modules:

main.py: the main module that that creates a GUI using the tkinter module to manage daily expenses. It includes functionalities to save, update, and delete records of expenses, and calculate total and remaining expenses. It also interacts with a SQLite database.

mydb.py: a module that defines a Database class that uses SQLite to create and manage a table called "expense_record" which stores items' name, price, and purchase date. It has methods to fetch, insert, remove, and update records, and the class destructor closes the database connection.The

Expense class has the following attributes:

- id: an integer that represents the unique identifier of the expense
- item_name: a string that represents the description of the expense
- item_price: a float that represents the amount of the expense
- purchase_date: a string that represents the date of the expense in the format "YYYY- MM-DD"

The ExpenseDialog has the following attributes:

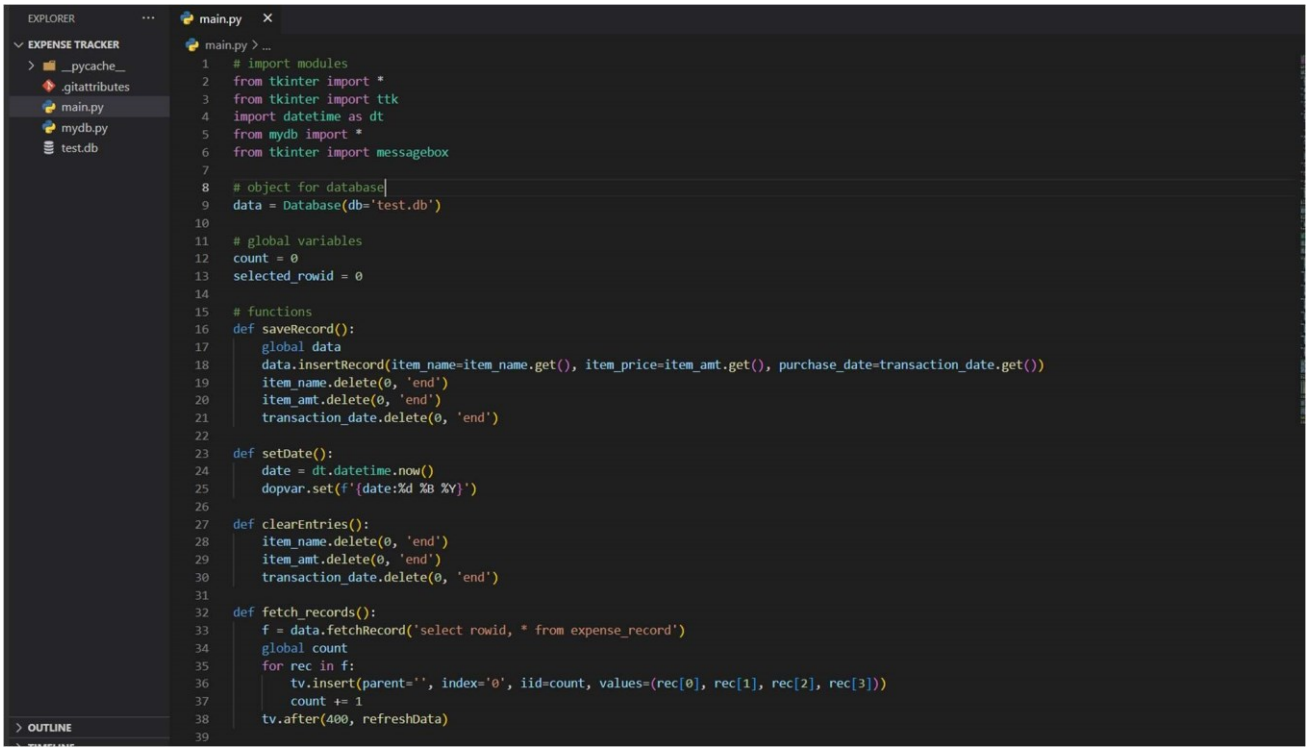
- master: the parent window of the dialogue box

- item_name_var: a Tkinter StringVar that represents the description field in the dialogue box
- item_price_var: a Tkinter DoubleVar that represents the amount field in the dialogue box
- purchase_date_var: a Tkinter StringVar that represents the date field in the dialogue box

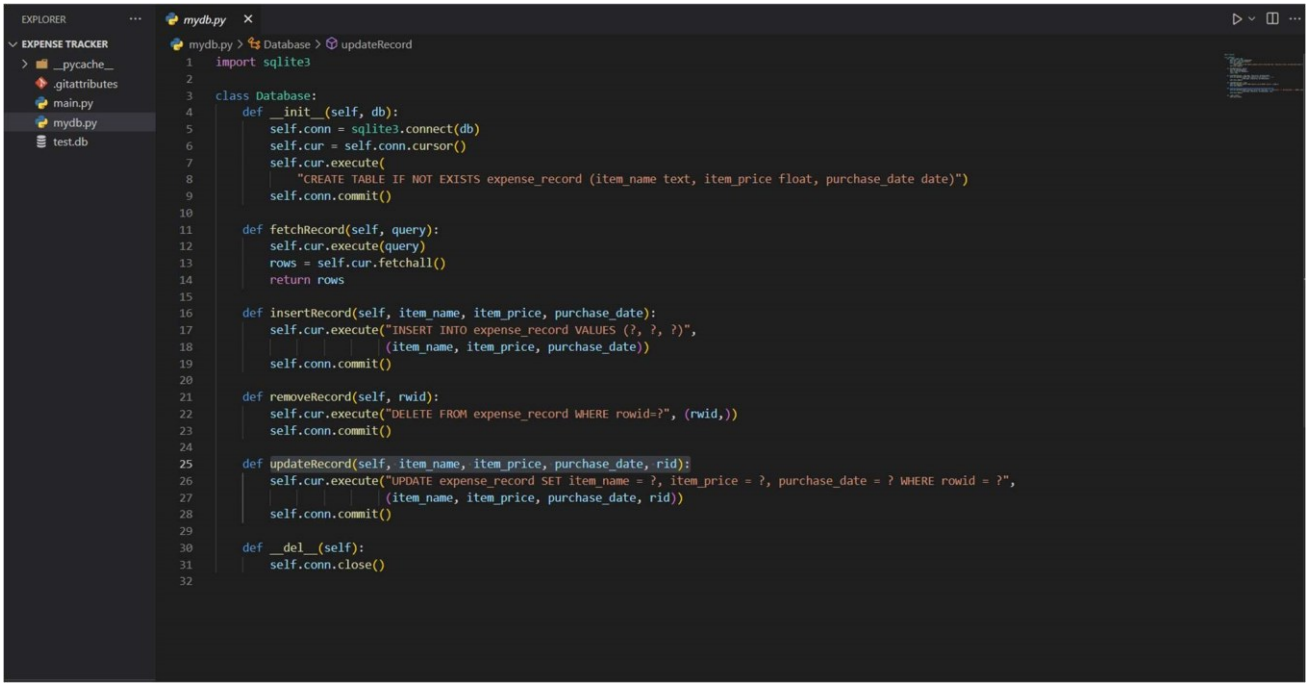
The Database has the following methods:

- _init_(self, db_file): initializes the database connection and creates the expenses table if it doesn't exist
- fetchRecord(self, query): fetches a list of all the expenses in the database
- insertRecord(self, item_name, item_price, purchase_date): inserts a new expense in the database
- removeRecord(self, rwid): deletes an existing expense from the database
- updateRecord(self, item_name, item_price, purchase_date, rid): updates an existing expense in the database

Code

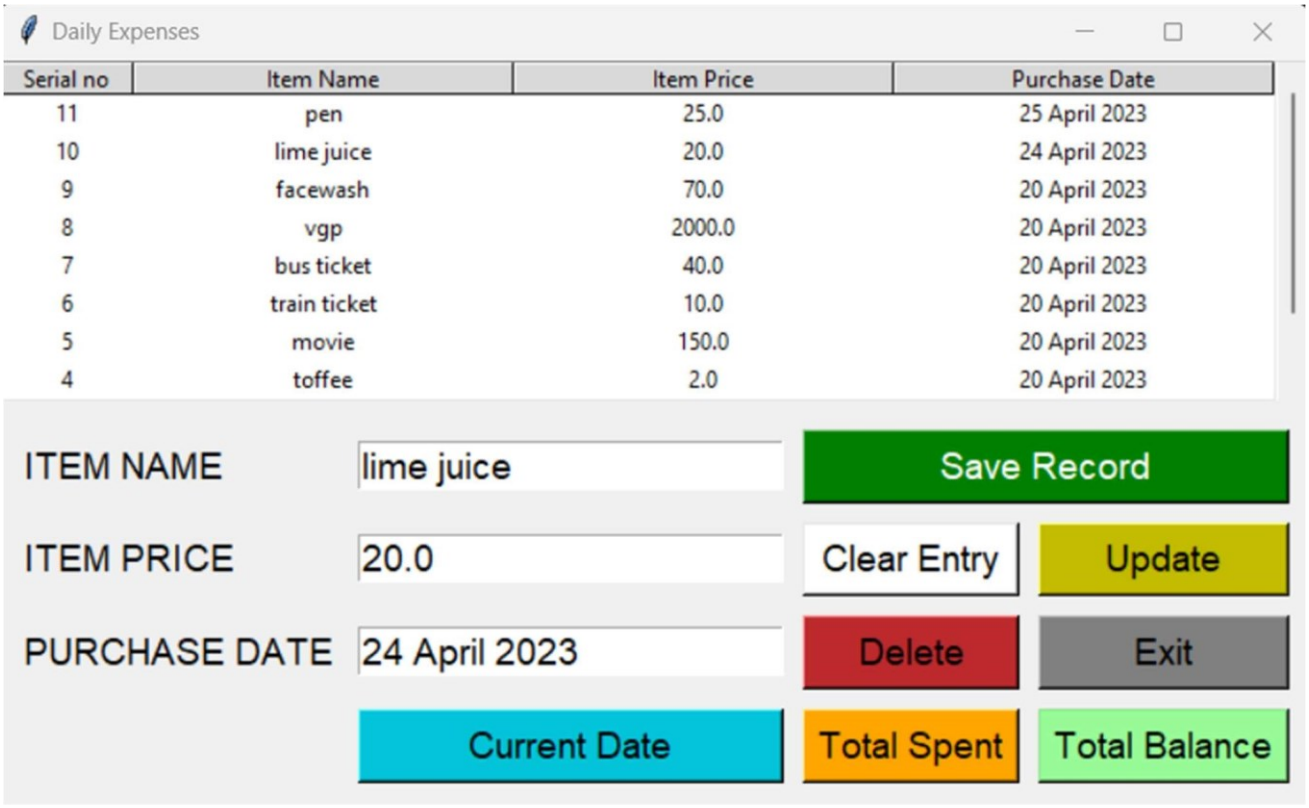


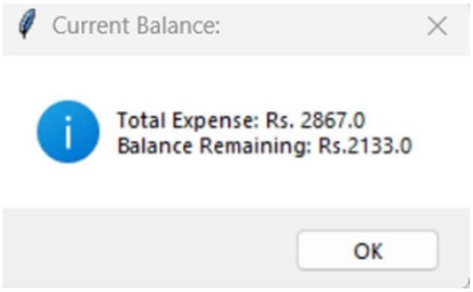
```python
# import modules
from tkinter import *
from tkinter import ttk
import datetime as dt
from mydb import *
from tkinter import messagebox

# object for database
data = Database(db='test.db')

# global variables
count = 0
selected_rowid = 0

# functions
def saveRecord():
    global data
    data.insertRecord(item_name=item_name.get(), item_price=item_amt.get(), purchase_date=transaction_date.get())
    item_name.delete(0, 'end')
    item_amt.delete(0, 'end')
    transaction_date.delete(0, 'end')

def setDate():
    date = dt.datetime.now()
    dopvar.set(f'{date:%d %B %Y}')

def clearEntries():
    item_name.delete(0, 'end')
    item_amt.delete(0, 'end')
    transaction_date.delete(0, 'end')

def fetch_records():
    f = data.fetchRecord('select rowid, * from expense_record')
    global count
    for rec in f:
        tv.insert(parent='', index='0', iid=count, values=(rec[0], rec[1], rec[2], rec[3]))
        count += 1
    tv.after(400, refreshData)
```

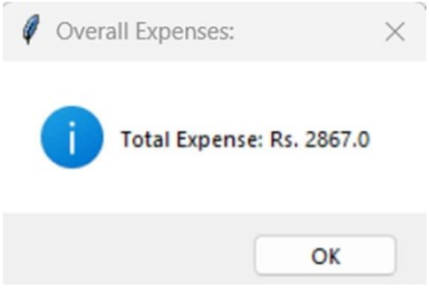


```python
import sqlite3

class Database:
    def __init__(self, db):
        self.conn = sqlite3.connect(db)
        self.cur = self.conn.cursor()
        self.cur.execute(
            "CREATE TABLE IF NOT EXISTS expense_record (item_name text, item_price float, purchase_date date)")
        self.conn.commit()

    def fetchRecord(self, query):
        self.cur.execute(query)
        rows = self.cur.fetchall()
        return rows

    def insertRecord(self, item_name, item_price, purchase_date):
        self.cur.execute("INSERT INTO expense_record VALUES (?, ?, ?)",
                         (item_name, item_price, purchase_date))
        self.conn.commit()

    def removeRecord(self, rwid):
        self.cur.execute("DELETE FROM expense_record WHERE rowid=?", (rwid,))
        self.conn.commit()

    def updateRecord(self, item_name, item_price, purchase_date, rid):
        self.cur.execute("UPDATE expense_record SET item_name = ?, item_price = ?, purchase_date = ? WHERE rowid = ?",
                         (item_name, item_price, purchase_date, rid))
        self.conn.commit()

    def __del__(self):
        self.conn.close()
```

Output


Daily Expenses
Serial no
Item Name
Item Price
Purchase Date
11 pen 25.0 25 April 2023
10 lime juice 20.0 24 April 2023
9 facewash 70.0 20 April 2023
8 vgp 2000.0 20 April 2023
7 bus ticket 40.0 20 April 2023
6 train ticket 10.0 20 April 2023
5 movie 150.0 20 April 2023
4 toffee 2.0 20 April 2023
ITEM NAME
lime juice
ITEM PRICE
20.0
PURCHASE DATE
24 April 2023
Save Record
Clear Entry
Update
Delete
Exit
Current Date
Total Spent
Total Balance


Current Balance:
Total Expense: Rs. 2867.0
Balance Remaining: Rs.2133.0
OK


Overall Expenses:
Total Expense: Rs. 2867.0
OK

Conclusion

The Expense Tracker is a simple yet useful application for users to keep track of their expenses. The application was built using the Python programming language and the Tkinter GUI toolkit. The data is stored in an SQLite3 database. The application provides a user-friendly interface to add, view and delete expenses. The application was tested manually and handled invalid inputs gracefully.