EXPENSE TRACKER

A MINI-PROJECT REPORT

18CSC207J - ADVANCED PROGRAMMING PRACTICE

Submitted by

DIVYANSHU YADAV RA2111003010693

Under the guidance of

MALAR SELVI G

Assistant Professor, Department of Computer Science and Engineering

in partial fulfilment for the award of the degreeof

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Deemed to be University u/s 3 of UGC Act, 1956

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

MAY 2023

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

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

BONAFIDE CERTIFICATE

Register No **RA2111003010693**

Certified to be the bonafide work done by **DIVYANSHU YADAV** of II Year/ IV Sem B.Tech Degree course in **ADVANCED PROGRAMMING PRACTICE 18CSC207J** in **SRM INSTITUTE OF SCIENCE & TECHNOLOGY,** Kattankulathur during the academic year 2022-2023

DATE:

SIGNATURE	**SIGNATURE**
LAB INCHARGE	**HEAD OF THE DEPARTMENT**
G. Malarselvi	**Dr M. Pushpalatha**
Assistant Professor	Professor and Head,
Department of Computing Technologies	Department of Computing Technologies
SRM Institute of Science and Technology	SRM Institute of Science and Technology

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 creates the main window and starts the event loop

expense.py: a module that defines the Expense class that represents an expense

expense_dialog.py: a module that defines the ExpenseDialog class that creates the dialogue box to add or edit an expense

database.py: a module that defines the Database class that handles the communication with the database

The Expense class has the following attributes:

id: an integer that represents the unique identifier of the expense

description: a string that represents the description of the expense

amount: a float that represents the amount of the expense

date: a string that represents the date of the expense in the format "YYYY-MM-DD"

category: a string that represents the category of the expense

The ExpenseDialog class has the following attributes:

master: the parent window of the dialogue box expense: an instance of the Expense class that represents the expense to edit, or None if adding a new expense

description_var: a Tkinter StringVar that represents the description field in the dialogue box

amount_var: a Tkinter DoubleVar that represents the amount field in the dialogue box

date_var: a Tkinter StringVar that represents the date field in the dialogue box

category_var: a Tkinter StringVar that represents the category field in the dialogue box

The Database class has the following methods:

init(self, db_file): initializes the database connection and creates the expenses table if it doesn't exist

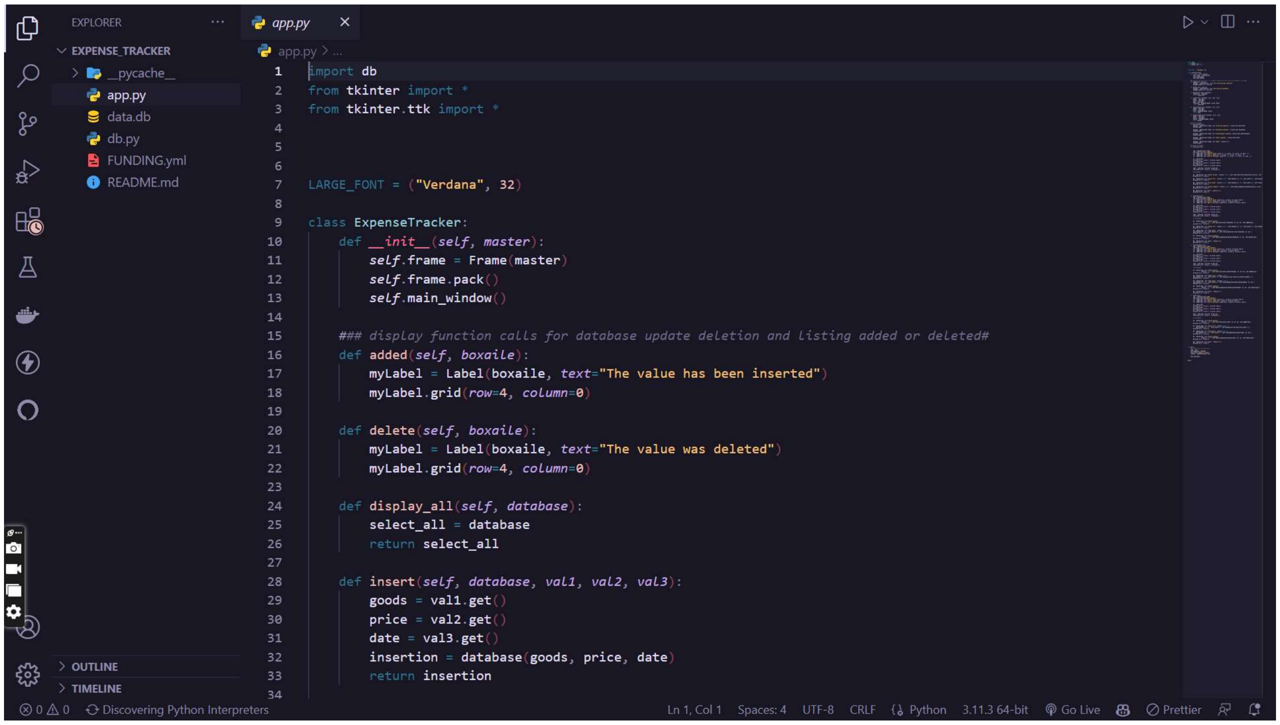
add_expense(self, expense): inserts a new expense in the database

get_expenses(self): returns a list of all the expenses in the database

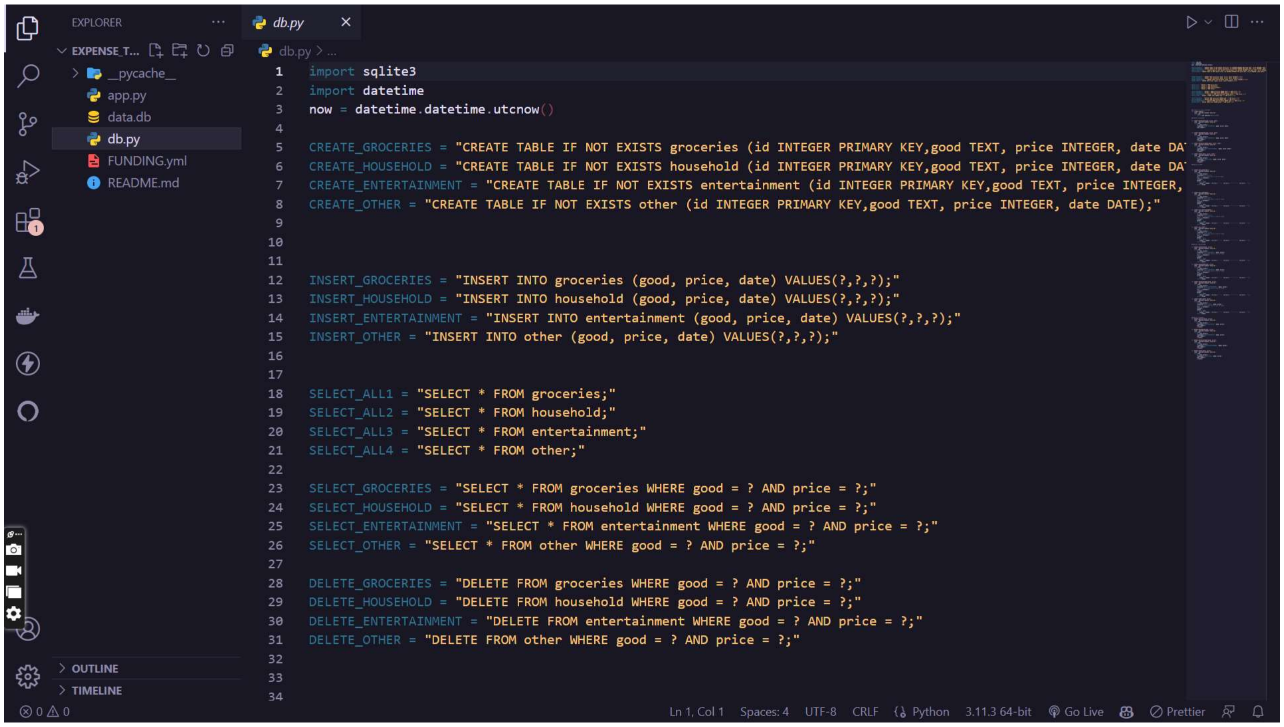
update_expense(self, expense): updates an existing expense in the database

delete_expense(self, expense): deletes an existing expense from the database

Code

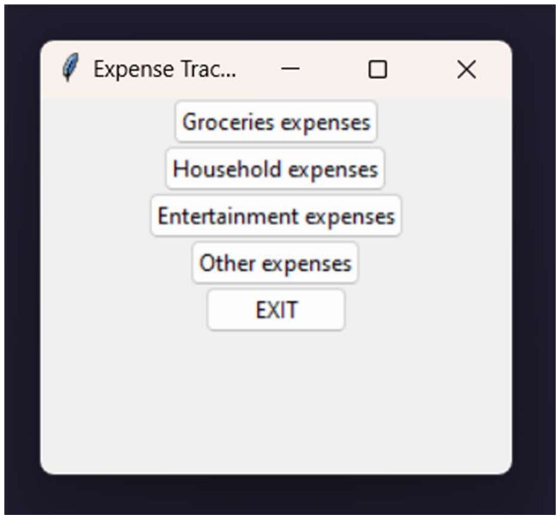

EXPLORER
EXPENSE_TRACKER
__pycache__
app.py
data.db
db.py
FUNDING.yml
README.md
OUTLINE
TIMELINE
app.py
Discovering Python Interpreters
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit Go Live Prettier

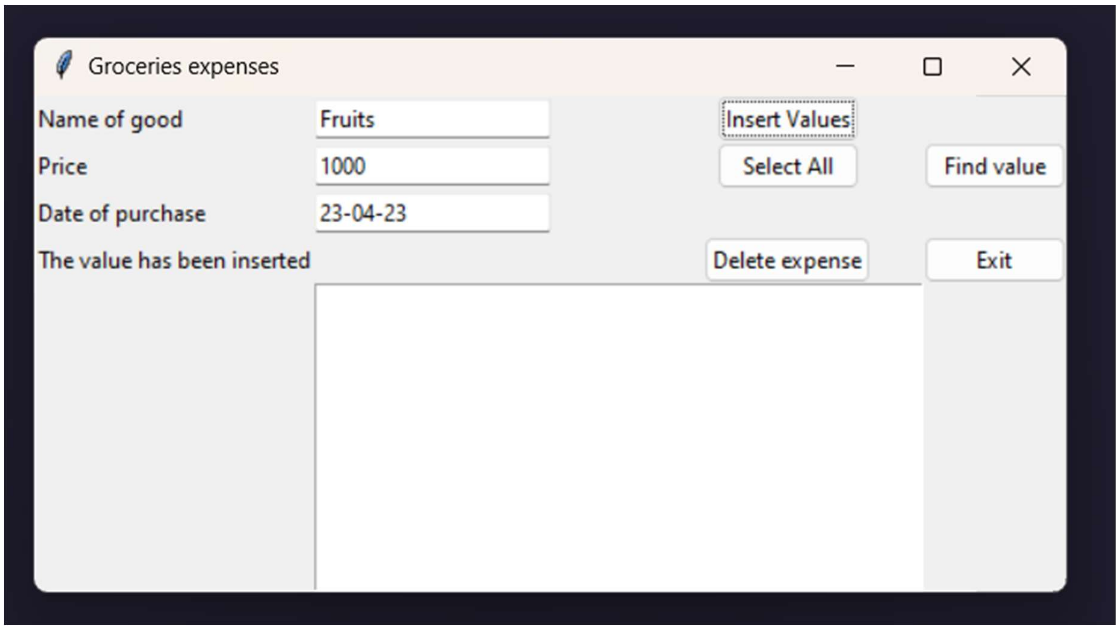
```python
import db
from tkinter import *
from tkinter.ttk import *



LARGE_FONT = ("Verdana", 32)

class ExpenseTracker:
    def __init__(self, master):
        self.frame = Frame(master)
        self.frame.pack()
        self.main_window()

    ### display function calls for database update deletion and listing added or deleted#
    def added(self, boxaile):
        myLabel = Label(boxaile, text="The value has been inserted")
        myLabel.grid(row=4, column=0)

    def delete(self, boxaile):
        myLabel = Label(boxaile, text="The value was deleted")
        myLabel.grid(row=4, column=0)

    def display_all(self, database):
        select_all = database
        return select_all

    def insert(self, database, val1, val2, val3):
        goods = val1.get()
        price = val2.get()
        date = val3.get()
        insertion = database(goods, price, date)
        return insertion

```



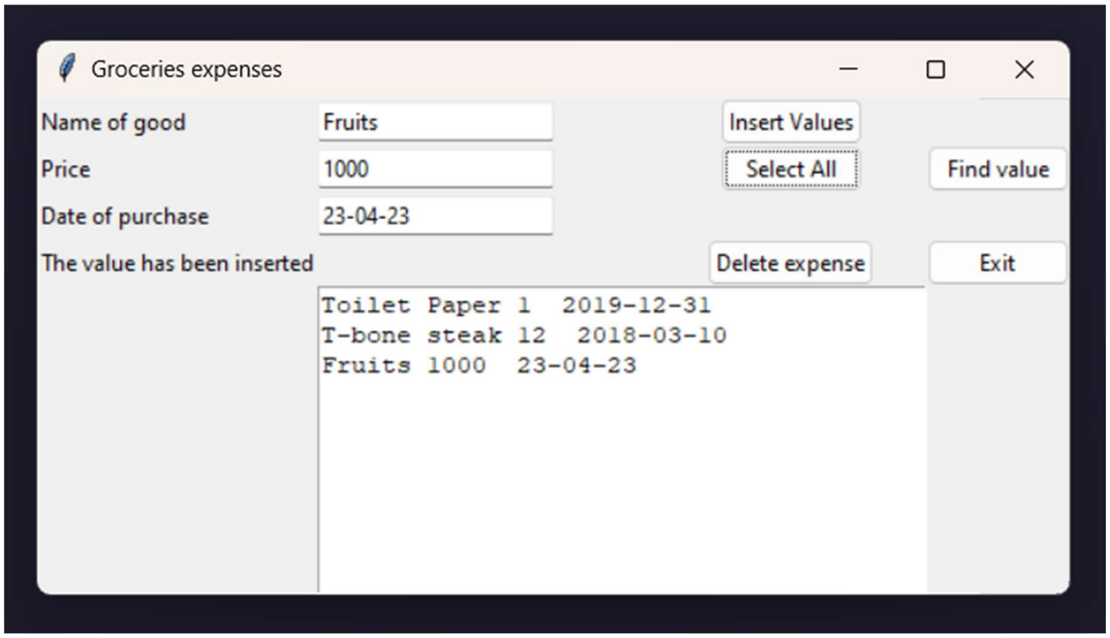
```python
import sqlite3
import datetime
now = datetime.datetime.utcnow()

CREATE_GROCERIES = "CREATE TABLE IF NOT EXISTS groceries (id INTEGER PRIMARY KEY,good TEXT, price INTEGER, date DA
CREATE_HOUSEHOLD = "CREATE TABLE IF NOT EXISTS household (id INTEGER PRIMARY KEY,good TEXT, price INTEGER, date DA
CREATE_ENTERTAINMENT = "CREATE TABLE IF NOT EXISTS entertainment (id INTEGER PRIMARY KEY,good TEXT, price INTEGER,
CREATE_OTHER = "CREATE TABLE IF NOT EXISTS other (id INTEGER PRIMARY KEY,good TEXT, price INTEGER, date DATE);"



INSERT_GROCERIES = "INSERT INTO groceries (good, price, date) VALUES(?,?,?);"
INSERT_HOUSEHOLD = "INSERT INTO household (good, price, date) VALUES(?,?,?);"
INSERT_ENTERTAINMENT = "INSERT INTO entertainment (good, price, date) VALUES(?,?,?);"
INSERT_OTHER = "INSERT INTO other (good, price, date) VALUES(?,?,?);"


SELECT_ALL1 = "SELECT * FROM groceries;"
SELECT_ALL2 = "SELECT * FROM household;"
SELECT_ALL3 = "SELECT * FROM entertainment;"
SELECT_ALL4 = "SELECT * FROM other;"

SELECT_GROCERIES = "SELECT * FROM groceries WHERE good = ? AND price = ?;"
SELECT_HOUSEHOLD = "SELECT * FROM household WHERE good = ? AND price = ?;"
SELECT_ENTERTAINMENT = "SELECT * FROM entertainment WHERE good = ? AND price = ?;"
SELECT_OTHER = "SELECT * FROM other WHERE good = ? AND price = ?;"

DELETE_GROCERIES = "DELETE FROM groceries WHERE good = ? AND price = ?;"
DELETE_HOUSEHOLD = "DELETE FROM household WHERE good = ? AND price = ?;"
DELETE_ENTERTAINMENT = "DELETE FROM entertainment WHERE good = ? AND price = ?;"
DELETE_OTHER = "DELETE FROM other WHERE good = ? AND price = ?;"



```

Output (Snapshot)





Name of good Fruits
Price 1000
Date of purchase 23-04-23
The value has been inserted



Name of good: Fruits
Price: 1000
Date of purchase: 23-04-23

The value has been inserted

```
Toilet Paper 1  2019-12-31
T-bone steak 12  2018-03-10
Fruits 1000  23-04-23
```

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.