EXPENSE TRACKER REPORT

PYTHON PROJECT

VITyarthi PROJECT

NAME : ADITI SINGH

REG NO.: 25BCE10868

1. Introduction

Managing day-to-day expenses is an essential task for individuals and businesses alike. Most people struggle to track their spending habits due to manual methods such as notebooks or basic spreadsheets. To overcome these challenges, a computer-based solution offers a more efficient, organized, and error-free approach.

This project presents a Python-based Expense Tracker Application built using Tkinter for the graphical user interface, SQLite as the database for storing expense records, and tkcalendar to facilitate date selection. The application enables users to add, update, view, and delete expenses in a user-friendly manner while maintaining a persistent database.

2. Problem Statement

Many individuals lack a systematic way to track their daily expenditures. Manual tracking methods are time-consuming, prone to errors, and often lack proper organization and retrieval features. Without proper tracking, users cannot analyze their spending patterns or maintain financial discipline.

The main problem addressed by this project is the absence of an easy-to-use, computerized system that allows users to record, edit, and manage expenses efficiently, ensuring accuracy and quick access to financial records.

3. Functional Requirements

Functional requirements describe what the system should do.

3.1 Add Expense

The user must be able to enter details such as date, payee, description, amount, and mode of payment.

The application should store this data in an SQLite database.

3.2 View Expense

The system must display all stored expenses in a tabular Tree View.

The user must be able to select an expense to view its detailed information.

3.3 Edit Expense

The user must be able to update previously stored expense information.

Updated records must be saved in the database.

3.4 Delete Expense

The user must be able to delete a selected expense.

The system must also offer an option to delete all expenses at once.

3.5 Clear Fields

The input form should reset to default values when the "Clear Fields" action is triggered.

3.6 Convert Expense to Words

The system allows users to convert a selected expense into a readable sentence.

This helps users understand the record linguistically.

3.7 Date Selection

The calendar widget must allow easy date picking to avoid format errors.

4. Non-Functional Requirements

Non-functional requirements define the system’s quality attributes.

4.1 Usability

The interface should be simple and intuitive.

Font style, button size, and layout should support easy navigation.

4.2 Performance

The system should quickly load, retrieve, and update expense data.

Operations like database read/write must be fast.

4.3 Reliability

Data stored in SQLite must be persistent.

The system should not crash during common operations such as adding or editing entries.

4.4 Maintainability

The source code should be modular, with separate functions handling specific tasks.

Clear naming conventions and comments should help future developers modify the system.

4.5 Portability

As the system uses Python and SQLite, it can run on multiple operating systems (Windows, macOS, Linux).

4.6 Security

Data integrity must be preserved by preventing invalid or incomplete fields when saving.

The system prompts users before deleting records to avoid accidental loss.

5. System Architecture

The system follows a simple three-layer architecture:

5.1 Presentation Layer (User Interface Layer)

Built using Tkinter.

Includes widgets such as Labels, Entry fields, Buttons, OptionMenu, DateEntry, and TreeView.

Supports user interaction for adding, viewing, editing, and deleting expenses.

5.2 Application Logic Layer

Consists of Python functions handling operations:

add_another_expense() – Adds new expenses

edit_expense() – Updates selected records

remove_expense() – Deletes a record

remove_all_expenses() – Clears all data

list_all_expenses() – Loads all data from database

view_expense_details() – Displays details of selected entry

Ensures validation, message prompts, and linking UI with backend.

5.3 Data Layer (Database Layer)

Uses SQLite database named Expense Tracker.db.

Table: ExpenseTracker

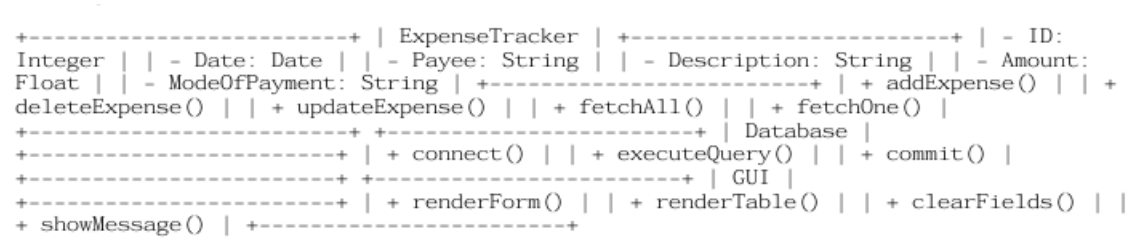
Columns: ID, Date, Payee, Description, Amount, ModeOfPayment

Stores all expenses securely and persistently.

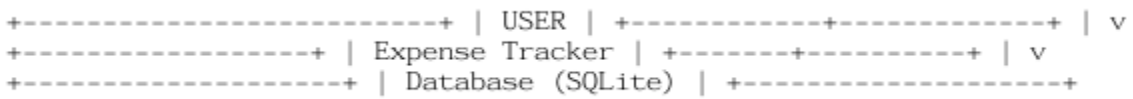
DESIGN DIAGRAMs

Use Case Diagrams

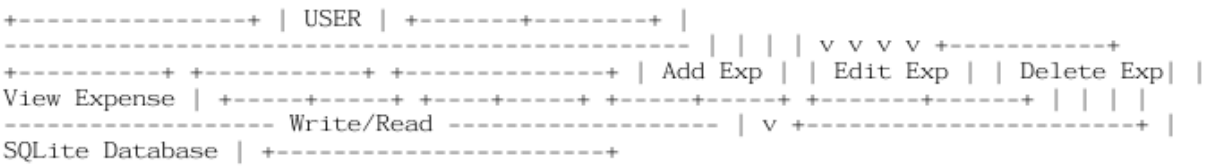
Work Flow Diagram



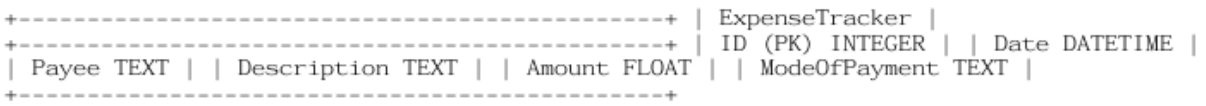
DFD Level 0



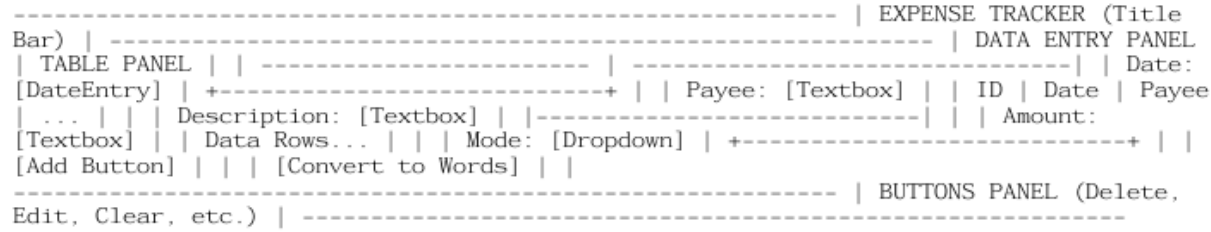
DFD Level 1



ERD Diagram



GUI Layout Diagram



1. Design Decisions

The following design choices were made to ensure that the Expense Tracker is simple, efficient, and user-friendly:

1.1 Choice of Programming Language

Python was chosen because:

It offers strong GUI support through Tkinter.

It integrates easily with SQLite for backend storage.

It is simple to learn and widely used in software development.

1.2 GUI Framework (Tkinter)

Tkinter was selected as the GUI framework because:

It comes built-in with Python (no additional installation needed).

It supports widgets like buttons, forms, tables, dropdowns, and message boxes.

It is ideal for desktop applications with moderate complexity.

1.3 Database (SQLite)

SQLite was preferred due to:

Lightweight storage in a single .db file.

No server installation required.

Fast read/write operations for small applications.

1.4 TkCalendar Widget

The DateEntry widget from tkcalendar was used to:

Avoid incorrect date formatting.

Provide a calendar-style selection.

1.5 Table Display (TreeView)

TreeView from Tkinter's ttk module was used for:

Structured representation of expenses.

Horizontal and vertical scrolling.

Easy selection of specific records.

1.6 Modular Function Design

Functions like add_another_expense(), edit_expense(), and remove_expense() were separated to:

Improve readability.

Make the code maintainable.

Allow reuse and easy debugging.

2. Implementation Details

Below is a summary of how the system is built internally:

2.1 Database Initialization

A table named ExpenseTracker is created with fields:

ID (Primary Key)

Date

Payee

Description

Amount

Mode of Payment

This ensures all expenses are stored permanently.

2.2 Main Functional Modules

• Add Expense

Takes form input values.

Validates empty fields.

Inserts values into database.

• View Expense

Loads all values from database into TreeView.

Allows selection to view details.

• Edit Expense

Retrieves selected record.

Updates fields in the form.

Saves modified data back to the database.

• Delete Expense

Deletes the selected item or all items.

Confirmation message ensures safety.

• Clear Fields

Resets the input form to default values.

• Convert Expense to Words

Converts numeric and textual details into a readable sentence.

2.3 GUI Layout

The window is divided into:

Data Entry Frame (left side)

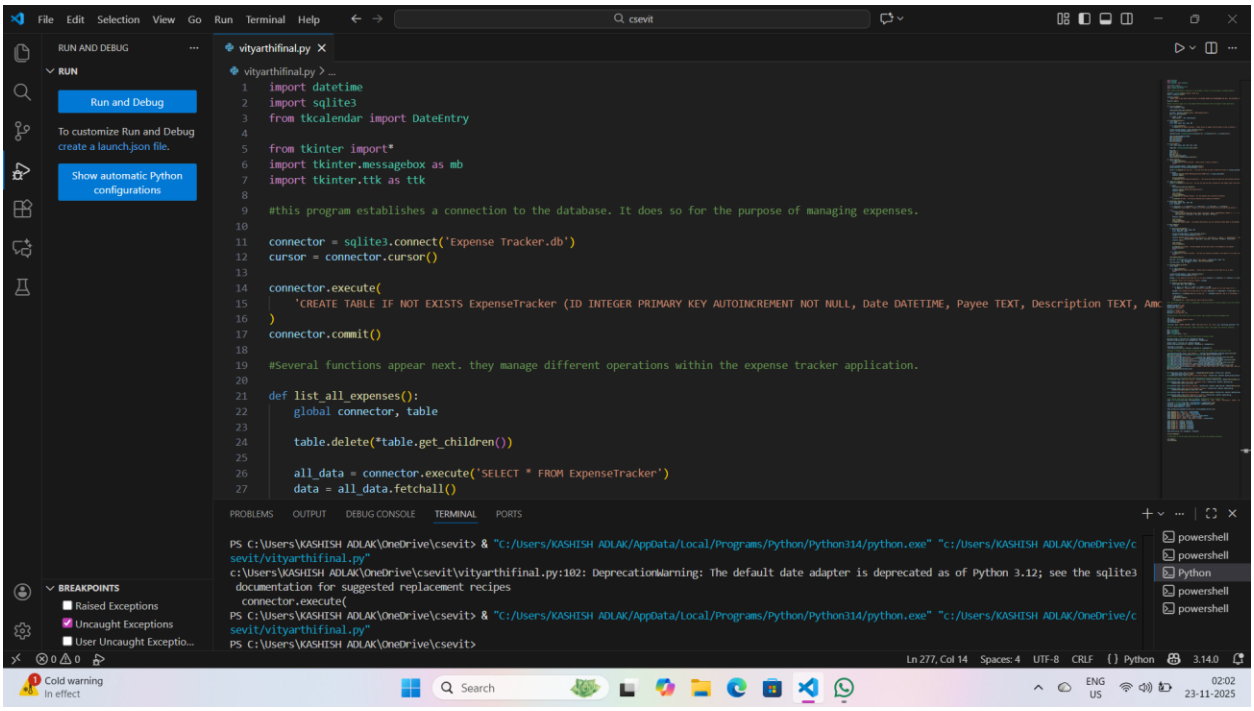
Buttons Frame (top center)

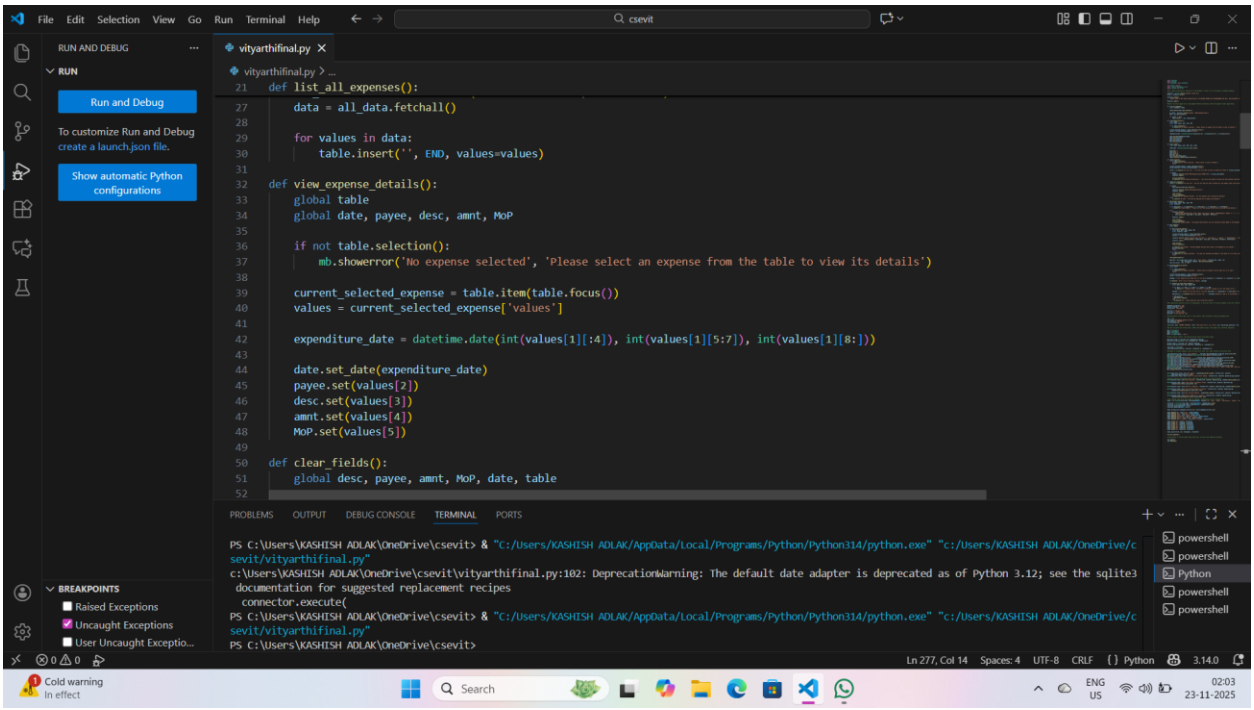
TreeView Frame (right side listing the expenses)

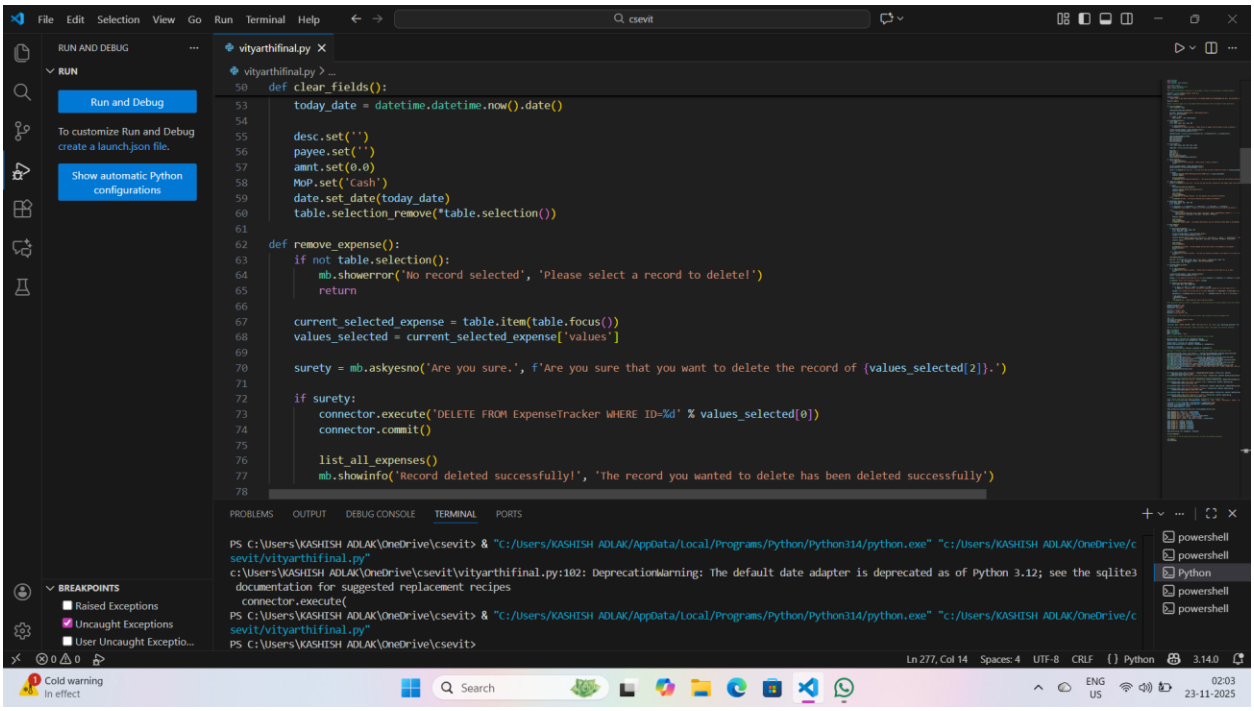
2.4 Application Flow

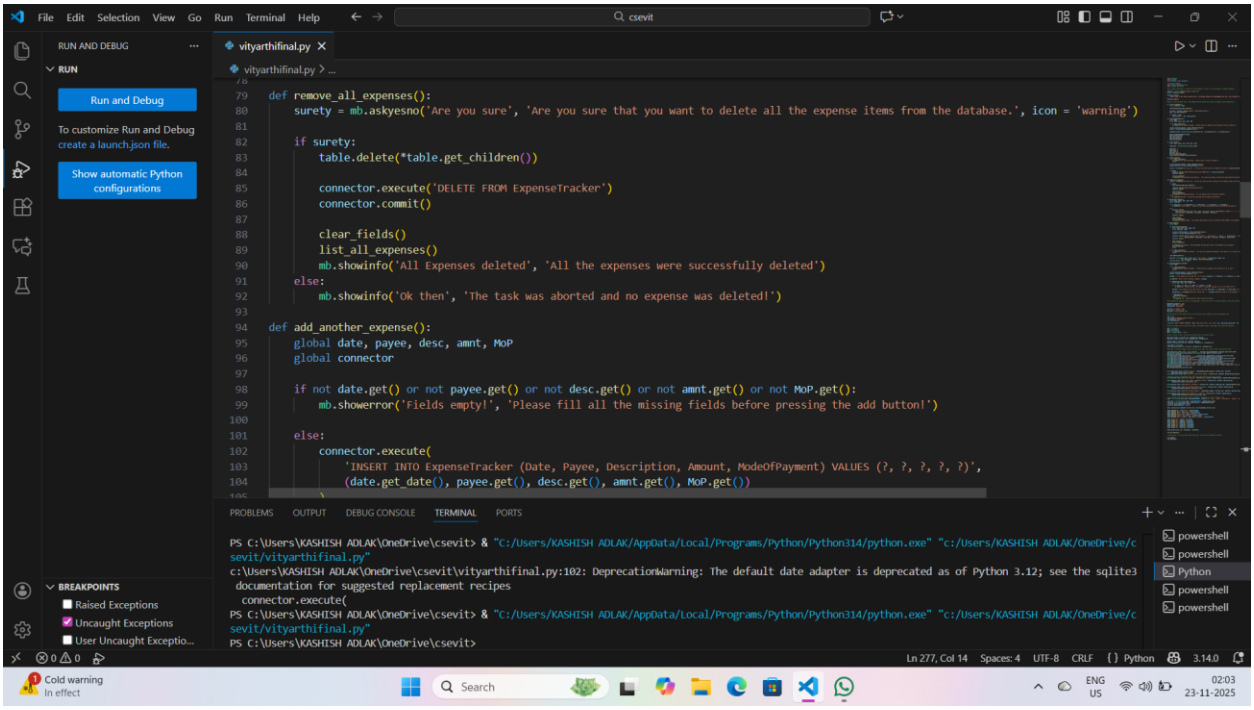
1. User inputs expense details.
2. Data gets stored in the SQLite database.
3. Data is fetched and displayed in the TreeView.
4. User can edit, delete, or view details of each entry.

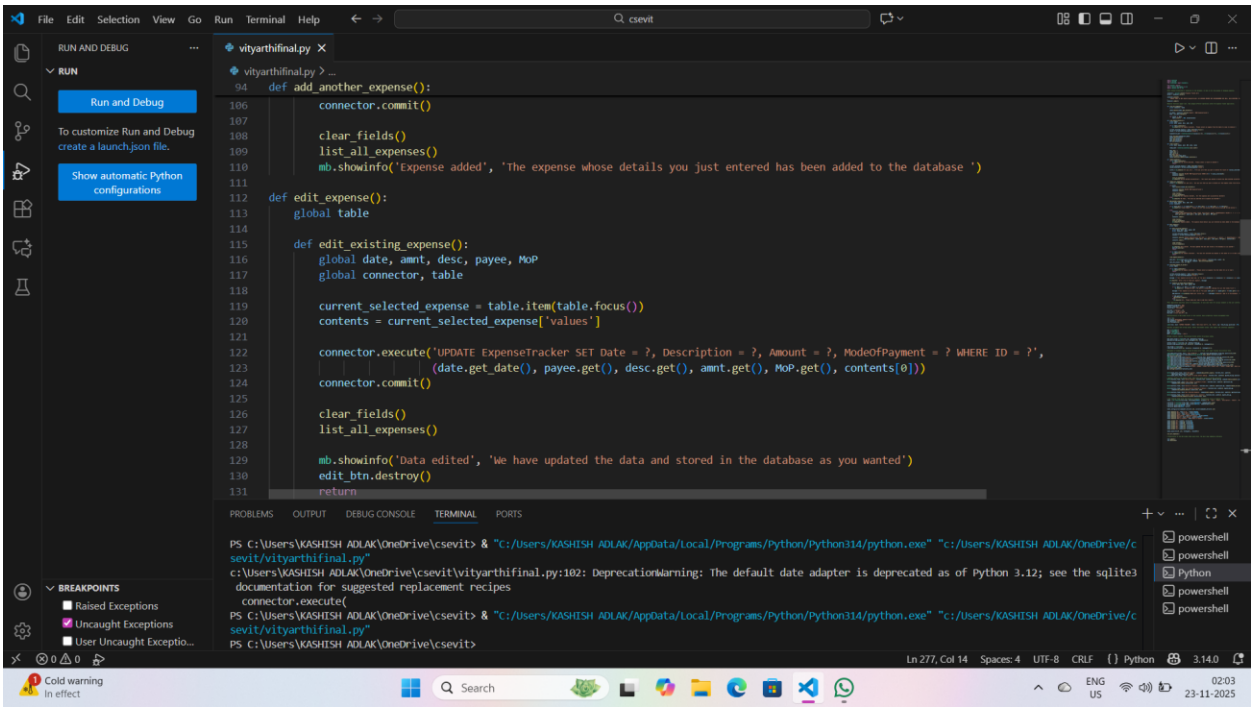
RESULT

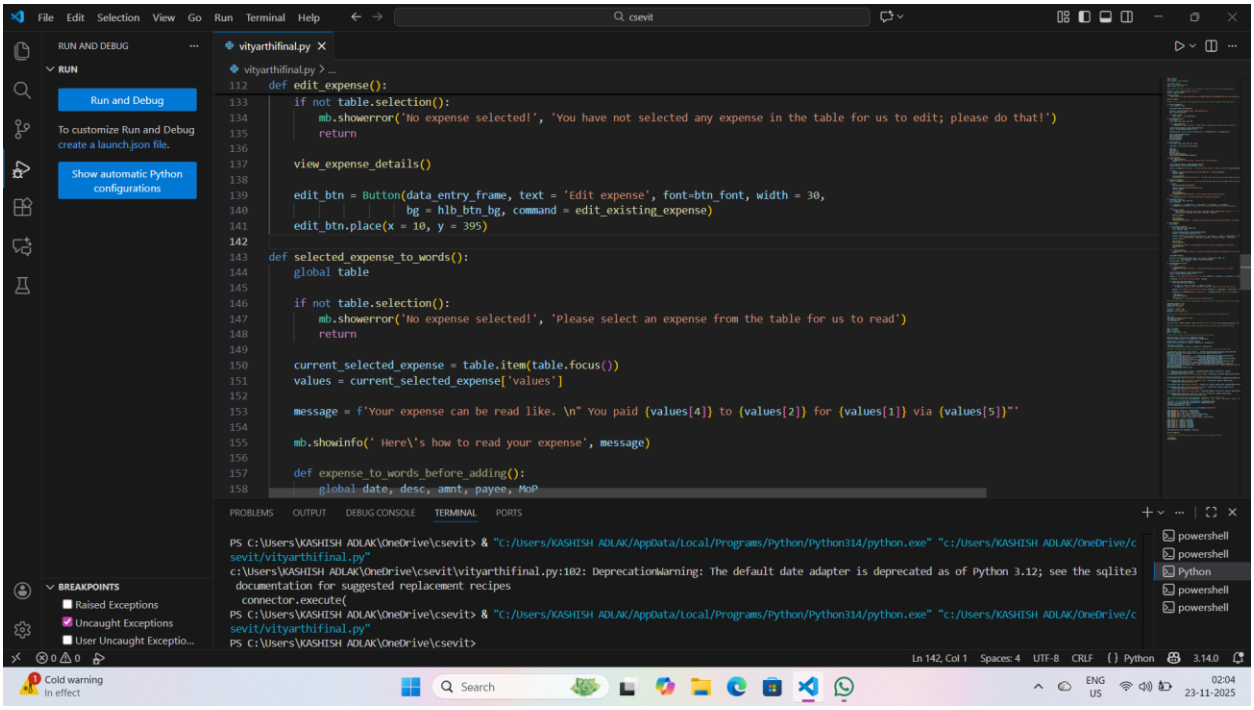

File Edit Selection View Go Run Terminal Help
csevit
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show automatic Python configurations
vityarthifinal.py
vityarthifinal.py > ...
import datetime
import sqlite3
from tkcalendar import DateEntry
from tkinter import*
import tkinter.messagebox as mb
import tkinter.ttk as ttk
#this program establishes a connection to the database. It does so for the purpose of managing expenses.
connector = sqlite3.connect('Expense Tracker.db')
cursor = connector.cursor()
connector.execute(
'CREATE TABLE IF NOT EXISTS ExpenseTracker (ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, Date DATETIME, Payee TEXT, Description TEXT, Amo
)
connector.commit()
#Several functions appear next. they manage different operations within the expense tracker application.
def list_all_expenses():
global connector, table
table.delete(*table.get_children())
all_data = connector.execute('SELECT * FROM ExpenseTracker')
data = all_data.fetchall()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
c:\Users\KASHISH ADLAK\OneDrive\csevit\vityarthifinal.py:102: DeprecationWarning: The default date adapter is deprecated as of Python 3.12; see the sqlite3 documentation for suggested replacement recipes
connector.execute(
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
PS C:\Users\KASHISH ADLAK\OneDrive\csevit>
powershell
powershell
Python
powershell
powershell
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptio...
Ln 277, Col 14 Spaces: 4 UTF-8 CRLF { } Python 3.14.0
Cold warning
In effect
Search
ENG
US
02:02
23-11-2025

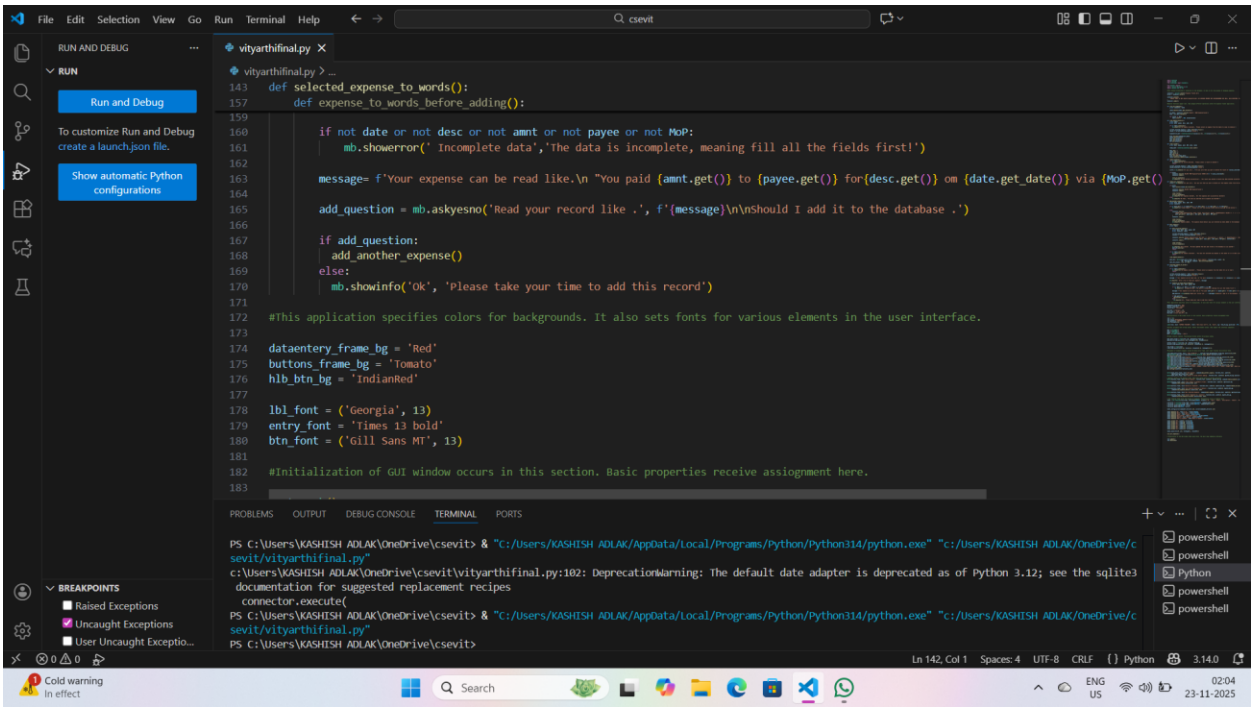

File Edit Selection View Go Run Terminal Help
csevit
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show automatic Python configurations
vityarthifinal.py
vityarthifinal.py > ...
def list_all_expenses():
data = all_data.fetchall()
for values in data:
table.insert('', END, values=values)
def view_expense_details():
global table
global date, payee, desc, amnt, MoP
if not table.selection():
mb.showerror('No expense selected', 'Please select an expense from the table to view its details')
current_selected_expense = table.item(table.focus())
values = current_selected_expense['values']
expenditure_date = datetime.date(int(values[1][:4]), int(values[1][5:7]), int(values[1][8:]))
date.set_date(expenditure_date)
payee.set(values[2])
desc.set(values[3])
amnt.set(values[4])
MoP.set(values[5])
def clear_fields():
global desc, payee, amnt, MoP, date, table
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
c:\Users\KASHISH ADLAK\OneDrive\csevit\vityarthifinal.py:102: DeprecationWarning: The default date adapter is deprecated as of Python 3.12; see the sqlite3 documentation for suggested replacement recipes
connector.execute(
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
PS C:\Users\KASHISH ADLAK\OneDrive\csevit>
powershell
powershell
Python
powershell
powershell
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptio...
Ln 277, Col 14 Spaces: 4 UTF-8 CRLF { } Python 3.14.0
Cold warning
In effect
Search
ENG
US
02:03
23-11-2025

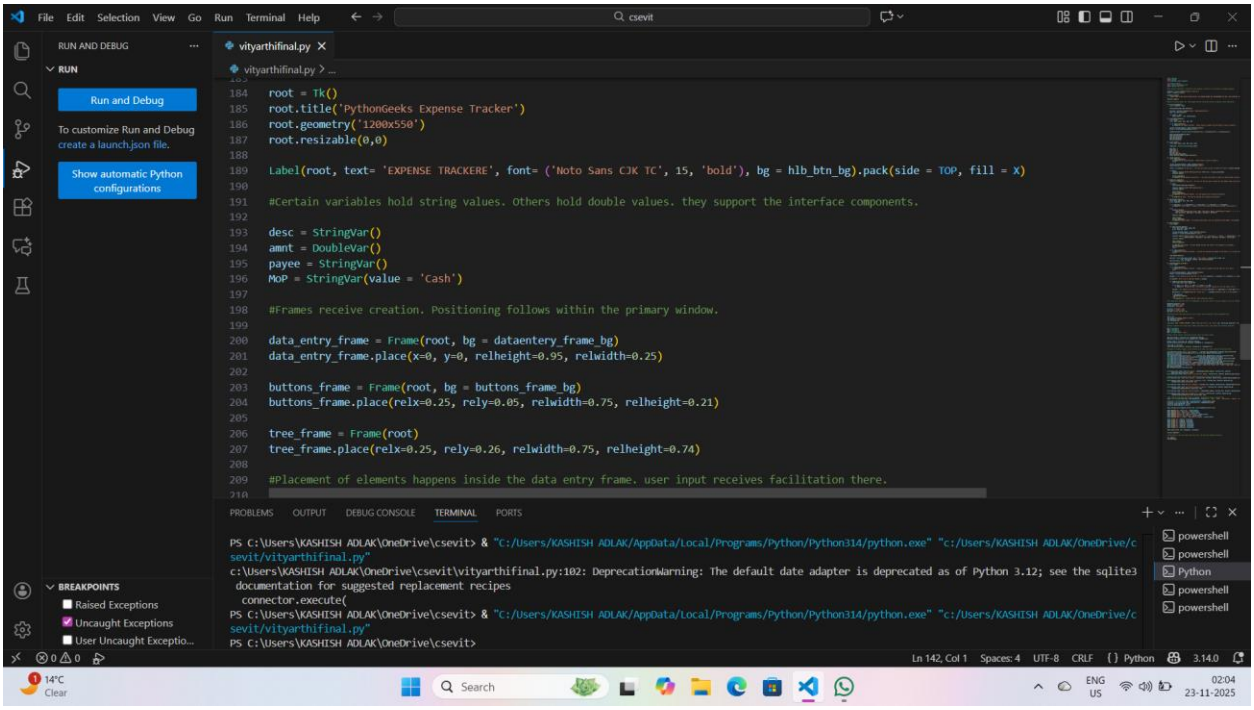

File Edit Selection View Go Run Terminal Help
csevit
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show automatic Python configurations
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptio...
vityarthifinal.py
vityarthifinal.py > ...
50 def clear_fields():
53 today_date = datetime.datetime.now().date()
54
55 desc.set('')
56 payee.set('')
57 amnt.set(0.0)
58 MoP.set('Cash')
59 date.set_date(today_date)
60 table.selection_remove(*table.selection())
61
62 def remove_expense():
63 if not table.selection():
64 mb.showerror('No record selected', 'Please select a record to delete!')
65 return
66
67 current_selected_expense = table.item(table.focus())
68 values_selected = current_selected_expense['values']
69
70 surety = mb.askyesno('Are you sure.', f'Are you sure that you want to delete the record of {values_selected[2]}.')
71
72 if surety:
73 connector.execute('DELETE FROM ExpenseTracker WHERE ID=%d' % values_selected[0])
74 connector.commit()
75
76 list_all_expenses()
77 mb.showinfo('Record deleted successfully!', 'The record you wanted to delete has been deleted successfully')
78
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
c:\Users\KASHISH ADLAK\OneDrive\csevit\vityarthifinal.py:102: DeprecationWarning: The default date adapter is deprecated as of Python 3.12; see the sqlite3 documentation for suggested replacement recipes
connector.execute(
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
PS C:\Users\KASHISH ADLAK\OneDrive\csevit>
powershell
powershell
Python
powershell
powershell
Ln 277, Col 14 Spaces: 4 UTF-8 CRLF { } Python 3.14.0
Cold warning In effect
Search
ENG US
02:03 23-11-2025

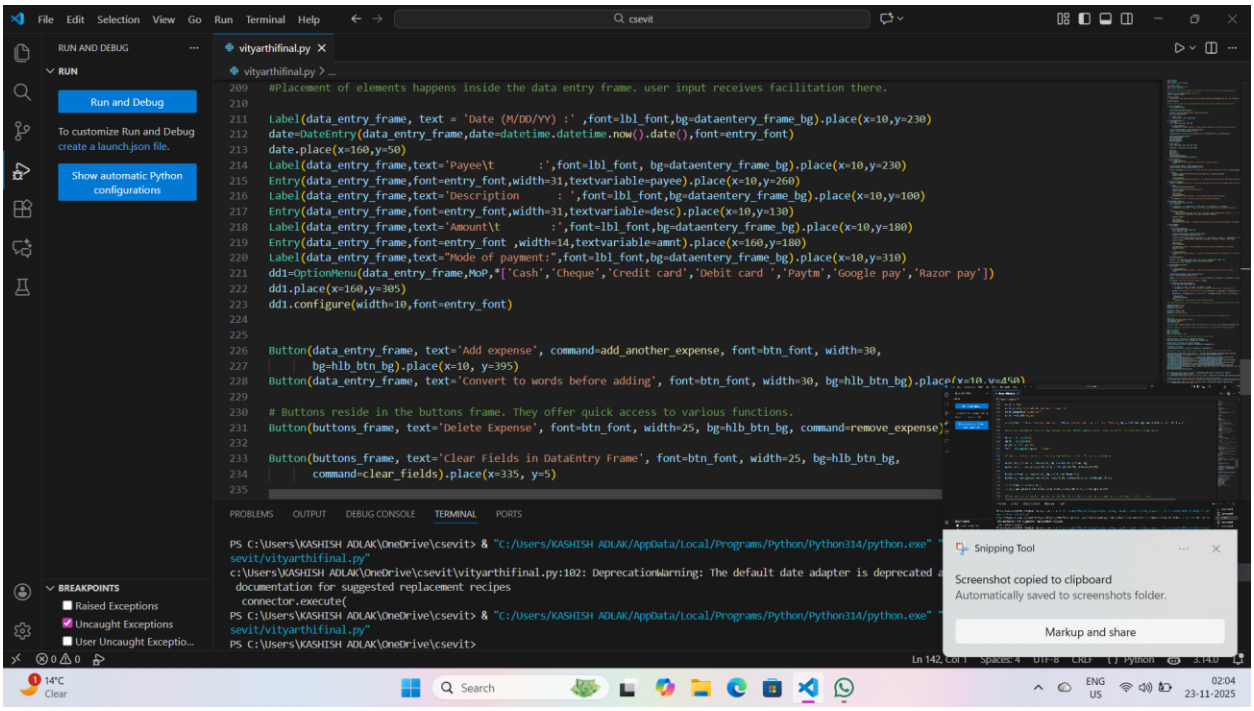

File Edit Selection View Go Run Terminal Help
csevit
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show automatic Python configurations
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptio...
vityarthifinal.py
vityarthifinal.py > ...
79 def remove_all_expenses():
80 surety = mb.askyesno('Are you sure', 'Are you sure that you want to delete all the expense items from the database.', icon = 'warning')
81
82 if surety:
83 table.delete(*table.get_children())
84
85 connector.execute('DELETE FROM ExpenseTracker')
86 connector.commit()
87
88 clear_fields()
89 list_all_expenses()
90 mb.showinfo('All Expenses deleted', 'All the expenses were successfully deleted')
91 else:
92 mb.showinfo('Ok then', 'The task was aborted and no expense was deleted!')
93
94 def add_another_expense():
95 global date, payee, desc, amnt, MoP
96 global connector
97
98 if not date.get() or not payee.get() or not desc.get() or not amnt.get() or not MoP.get():
99 mb.showerror('Fields empty!', 'Please fill all the missing fields before pressing the add button!')
100
101 else:
102 connector.execute(
103 'INSERT INTO ExpenseTracker (Date, Payee, Description, Amount, ModeOfPayment) VALUES (?, ?, ?, ?, ?)',
104 (date.get_date(), payee.get(), desc.get(), amnt.get(), MoP.get())
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
c:\Users\KASHISH ADLAK\OneDrive\csevit\vityarthifinal.py:102: DeprecationWarning: The default date adapter is deprecated as of Python 3.12; see the sqlite3 documentation for suggested replacement recipes
connector.execute(
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
PS C:\Users\KASHISH ADLAK\OneDrive\csevit>
powershell
powershell
Python
powershell
powershell
Ln 277, Col 14 Spaces: 4 UTF-8 CRLF { } Python 3.14.0
Cold warning In effect
Search
ENG US
02:03 23-11-2025

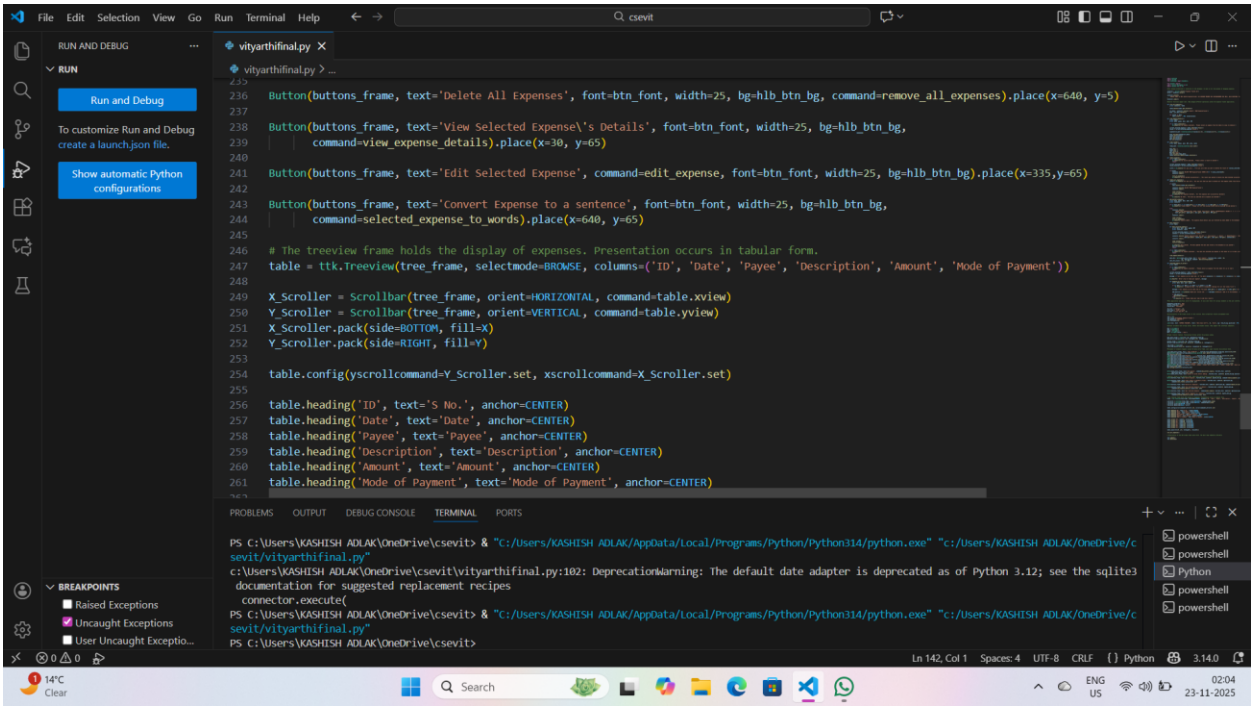

File Edit Selection View Go Run Terminal Help
csevit
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show automatic Python configurations
vityarthifinal.py
vityarthifinal.py > ...
94 def add_another_expense():
106 connector.commit()
107
108 clear_fields()
109 list_all_expenses()
110 mb.showinfo('Expense added', 'The expense whose details you just entered has been added to the database ')
111
112 def edit_expense():
113 global table
114
115 def edit_existing_expense():
116 global date, amnt, desc, payee, MoP
117 global connector, table
118
119 current_selected_expense = table.item(table.focus())
120 contents = current_selected_expense['values']
121
122 connector.execute('UPDATE ExpenseTracker SET Date = ?, Description = ?, Amount = ?, ModeOfPayment = ? WHERE ID = ?',
123 (date.get_date(), payee.get(), desc.get(), amnt.get(), MoP.get(), contents[0]))
124 connector.commit()
125
126 clear_fields()
127 list_all_expenses()
128
129 mb.showinfo('Data edited', 'We have updated the data and stored in the database as you wanted')
130 edit_btn.destroy()
131 return
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
c:\Users\KASHISH ADLAK\OneDrive\csevit\vityarthifinal.py:102: DeprecationWarning: The default date adapter is deprecated as of Python 3.12; see the sqlite3 documentation for suggested replacement recipes
connector.execute(
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
PS C:\Users\KASHISH ADLAK\OneDrive\csevit>
powershell
powershell
Python
powershell
powershell
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptio...
Ln 277, Col 14 Spaces: 4 UTF-8 CRLF { } Python 3.14.0
Cold warning
In effect
Search
ENG US
02:03
23-11-2025

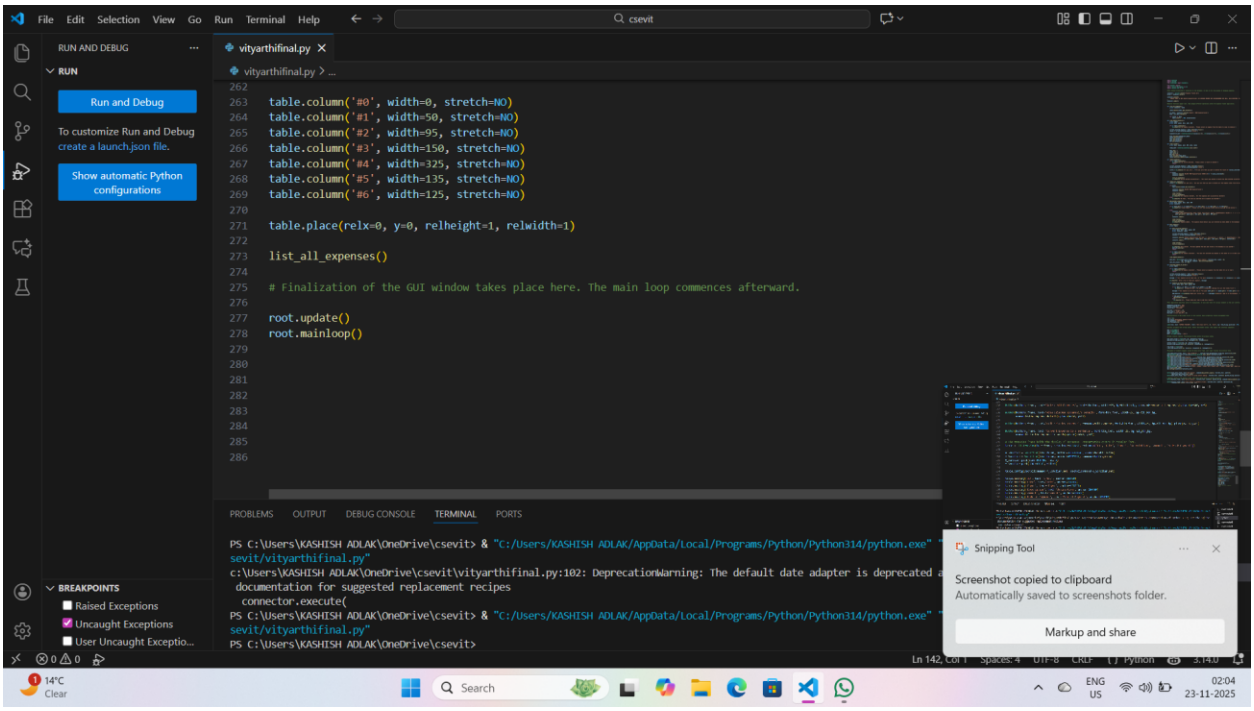

File Edit Selection View Go Run Terminal Help
csevit
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show automatic Python configurations
vityarthifinal.py
vityarthifinal.py > ...
112 def edit_expense():
133 if not table.selection():
134 mb.showerror('No expense selected!', 'You have not selected any expense in the table for us to edit; please do that!')
135 return
136
137 view_expense_details()
138
139 edit_btn = Button(data_entry_frame, text = 'Edit expense', font=btn_font, width = 30,
140 bg = hlb_btn_bg, command = edit_existing_expense)
141 edit_btn.place(x = 10, y = 395)
142
143 def selected_expense_to_words():
144 global table
145
146 if not table.selection():
147 mb.showerror('No expense selected!', 'Please select an expense from the table for us to read')
148 return
149
150 current_selected_expense = table.item(table.focus())
151 values = current_selected_expense['values']
152
153 message = f'Your expense can be read like. \n" You paid {values[4]} to {values[2]} for {values[1]} via {values[5]}"'
154
155 mb.showinfo(' Here\'s how to read your expense', message)
156
157 def expense_to_words_before_adding():
158 global date, desc, amnt, payee, MoP
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
c:\Users\KASHISH ADLAK\OneDrive\csevit\vityarthifinal.py:102: DeprecationWarning: The default date adapter is deprecated as of Python 3.12; see the sqlite3 documentation for suggested replacement recipes
connector.execute(
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
PS C:\Users\KASHISH ADLAK\OneDrive\csevit>
powershell
powershell
Python
powershell
powershell
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptio...
Ln 142, Col 1 Spaces: 4 UTF-8 CRLF { } Python 3.14.0
Cold warning
In effect
Search
ENG US
02:04
23-11-2025


File Edit Selection View Go Run Terminal Help
csevit
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show automatic Python configurations
vityarthifinal.py
vityarthifinal.py > ...
def selected_expense_to_words():
def expense_to_words_before_adding():
if not date or not desc or not amnt or not payee or not MoP:
mb.showerror(' Incomplete data','The data is incomplete, meaning fill all the fields first!')
message= f'Your expense can be read like.\n "You paid {amnt.get()} to {payee.get()} for{desc.get()} om {date.get_date()} via {MoP.get()
add_question = mb.askyesno('Read your record like .', f'{message}\n\nShould I add it to the database .')
if add_question:
add_another_expense()
else:
mb.showinfo('Ok', 'Please take your time to add this record')
#This application specifies colors for backgrounds. It also sets fonts for various elements in the user interface.
dataentery_frame_bg = 'Red'
buttons_frame_bg = 'Tomato'
hlb_btn_bg = 'IndianRed'
lbl_font = ('Georgia', 13)
entry_font = 'Times 13 bold'
btn_font = ('Gill Sans MT', 13)
#Initialization of GUI window occurs in this section. Basic properties receive assiognment here.
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
c:\Users\KASHISH ADLAK\OneDrive\csevit\vityarthifinal.py:102: DeprecationWarning: The default date adapter is deprecated as of Python 3.12; see the sqlite3 documentation for suggested replacement recipes
connector.execute(
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
PS C:\Users\KASHISH ADLAK\OneDrive\csevit>
powershell
powershell
Python
powershell
powershell
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptio...
Ln 142, Col 1 Spaces: 4 UTF-8 CRLF { } Python 3.14.0
Cold warning
In effect
Search
ENG US
02:04
23-11-2025


File Edit Selection View Go Run Terminal Help
csevit
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show automatic Python configurations
vityarthifinal.py
vityarthifinal.py > ...
root = Tk()
root.title('PythonGeeks Expense Tracker')
root.geometry('1200x550')
root.resizable(0,0)
Label(root, text= 'EXPENSE TRACKERE', font= ('Noto Sans CJK TC', 15, 'bold'), bg = hlb_btn_bg).pack(side = TOP, fill = X)
#Certain variables hold string values. Others hold double values. they support the interface components.
desc = StringVar()
amnt = DoubleVar()
payee = StringVar()
MoP = StringVar(value = 'Cash')
#Frames receive creation. Positioning follows within the primary window.
data_entry_frame = Frame(root, bg = dataentery_frame_bg)
data_entry_frame.place(x=0, y=0, relheight=0.95, relwidth=0.25)
buttons_frame = Frame(root, bg = buttons_frame_bg)
buttons_frame.place(relx=0.25, rely=0.05, relwidth=0.75, relheight=0.21)
tree_frame = Frame(root)
tree_frame.place(relx=0.25, rely=0.26, relwidth=0.75, relheight=0.74)
#Placement of elements happens inside the data entry frame. user input receives facilitation there.
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
c:\Users\KASHISH ADLAK\OneDrive\csevit\vityarthifinal.py:102: DeprecationWarning: The default date adapter is deprecated as of Python 3.12; see the sqlite3 documentation for suggested replacement recipes
connector.execute(
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/csevit/vityarthifinal.py"
PS C:\Users\KASHISH ADLAK\OneDrive\csevit>
powershell
powershell
Python
powershell
powershell
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptio...
Ln 142, Col 1 Spaces: 4 UTF-8 CRLF { } Python 3.14.0
14°C
Clear
Search
ENG US
02:04
23-11-2025


#Placement of elements happens inside the data entry frame. user input receives facilitation there.
Label(data_entry_frame, text = 'Date (M/DD/YY) :' ,font=lbl_font,bg=dataentery_frame_bg).place(x=10,y=230)
date=DateEntry(data_entry_frame,date=datetime.datetime.now().date(),font=entry_font)
date.place(x=160,y=50)
Label(data_entry_frame,text='Payee\t :',font=lbl_font, bg=dataentery_frame_bg).place(x=10,y=230)
Entry(data_entry_frame,font=entry_font,width=31,textvariable=payee).place(x=10,y=260)
Label(data_entry_frame,text='Description : ',font=lbl_font,bg=dataentery_frame_bg).place(x=10,y=100)
Entry(data_entry_frame,font=entry_font,width=31,textvariable=desc).place(x=10,y=130)
Label(data_entry_frame,text='Amount\t :',font=lbl_font,bg=dataentery_frame_bg).place(x=10,y=180)
Entry(data_entry_frame,font=entry_font ,width=14,textvariable=amnt).place(x=160,y=180)
Label(data_entry_frame,text="Mode of payment:",font=lbl_font,bg=dataentery_frame_bg).place(x=10,y=310)
dd1=OptionMenu(data_entry_frame,MoP,*['Cash','Cheque','Credit card','Debit card ','Paytm','Google pay','Razor pay'])
dd1.place(x=160,y=305)
dd1.configure(width=10,font=entry_font)
Button(data_entry_frame, text='Add expense', command=add_another_expense, font=btn_font, width=30,
bg=hlb_btn_bg).place(x=10, y=395)
Button(data_entry_frame, text='Convert to words before adding', font=btn_font, width=30, bg=hlb_btn_bg).place(x=10,y=450)
# Buttons reside in the buttons frame. They offer quick access to various functions.
Button(buttons_frame, text='Delete Expense', font=btn_font, width=25, bg=hlb_btn_bg, command=remove_expense).
Button(buttons_frame, text='Clear Fields in DataEntry Frame', font=btn_font, width=25, bg=hlb_btn_bg,
command=clear_fields).place(x=335, y=5)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe"
sevit/vityarthifinal.py"
c:\Users\KASHISH ADLAK\OneDrive\csevit\vityarthifinal.py:102: DeprecationWarning: The default date adapter is deprecated
documentation for suggested replacement recipes
connector.execute(
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe"
sevit/vityarthifinal.py"
PS C:\Users\KASHISH ADLAK\OneDrive\csevit>
Snipping Tool
Screenshot copied to clipboard
Automatically saved to screenshots folder.
Markup and share
Ln 142, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.14.0
14°C Clear
Search
ENG US
02:04 23-11-2025


Button(buttons_frame, text='Delete All Expenses', font=btn_font, width=25, bg=hlb_btn_bg, command=remove_all_expenses).place(x=640, y=5)
Button(buttons_frame, text='View Selected Expense\'s Details', font=btn_font, width=25, bg=hlb_btn_bg,
command=view_expense_details).place(x=30, y=65)
Button(buttons_frame, text='Edit Selected Expense', command=edit_expense, font=btn_font, width=25, bg=hlb_btn_bg).place(x=335,y=65)
Button(buttons_frame, text='Convert Expense to a sentence', font=btn_font, width=25, bg=hlb_btn_bg,
command=selected_expense_to_words).place(x=640, y=65)
# The treeview frame holds the display of expenses. Presentation occurs in tabular form.
table = ttk.Treeview(tree_frame, selectmode=BROWSE, columns=('ID', 'Date', 'Payee', 'Description', 'Amount', 'Mode of Payment'))
X_Scroller = Scrollbar(tree_frame, orient=HORIZONTAL, command=table.xview)
Y_Scroller = Scrollbar(tree_frame, orient=VERTICAL, command=table.yview)
X_Scroller.pack(side=BOTTOM, fill=X)
Y_Scroller.pack(side=RIGHT, fill=Y)
table.config(yscrollcommand=Y_Scroller.set, xscrollcommand=X_Scroller.set)
table.heading('ID', text='S No.', anchor=CENTER)
table.heading('Date', text='Date', anchor=CENTER)
table.heading('Payee', text='Payee', anchor=CENTER)
table.heading('Description', text='Description', anchor=CENTER)
table.heading('Amount', text='Amount', anchor=CENTER)
table.heading('Mode of Payment', text='Mode of Payment', anchor=CENTER)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/c
sevit/vityarthifinal.py"
c:\Users\KASHISH ADLAK\OneDrive\csevit\vityarthifinal.py:102: DeprecationWarning: The default date adapter is deprecated as of Python 3.12; see the sqlite3
documentation for suggested replacement recipes
connector.execute(
PS C:\Users\KASHISH ADLAK\OneDrive\csevit> & "C:/Users/KASHISH ADLAK/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/KASHISH ADLAK/OneDrive/c
sevit/vityarthifinal.py"
PS C:\Users\KASHISH ADLAK\OneDrive\csevit>
powershell
powershell
Python
powershell
powershell
Ln 142, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.14.0
14°C Clear
Search
ENG US
02:04 23-11-2025



OUTPUT

https://youtu.be/dzEkYdu8B_c?si=2iyzs4YyTzuvHITA

3. Testing Approach

Several testing techniques were followed:

3.1 Manual Testing

GUI applications require manual interaction. The following operations were tested:

Adding new expenses.

Editing existing entries.

Deleting expenses one by one and all at once.

Viewing selected record details.

Ensuring scrollbars work properly.

Checking date selection from the calendar.

3.2 Functional Testing

Each function was tested separately:

Input validation

Correct insertion into DB

Accurate fetching of data

Update and delete operations

3.3 Boundary Testing

Empty fields

Very large entries in description

Zero or negative amount values

Invalid mode of payment selection

3.4 Database Testing

Verified:

Data integrity

Auto-incrementing ID

Correct deletion and refresh in the TreeView

4. Challenges Faced

4.1 Handling Tkinter Layout

Managing multiple frames and widget positions required careful alignment to avoid overlapping.

4.2 Date Formatting Issues

Dates needed conversion between:

TkCalendar format

SQLite storage format

Python datetime object

4.3 Updating TreeView

Refreshing the list required:

Clearing previous entries

Re-fetching updated data

4.4 Editing an Existing Record

Identifying the selected row's ID and updating it correctly required proper mapping between:

TreeView selection

Database entry

4.5 Validations

Preventing empty field submissions and handling invalid input took multiple error checks.

5. Language and Key Takeaways

5.1 Programming Language Used

Python 3 with libraries:

Tkinter (GUI)

Tkcalendar (Date picker)

SQLite3 (Database)

ttk module (TreeView table)

5.2 Key Takeaways

Learned how to build a complete GUI application.

Understood database integration with front-end elements.

Improved program structuring using functions.

Learned how to handle form inputs and data validation.

Gained experience with CRUD operations (Create, Read, Update, Delete).

Enhanced understanding of event-driven programming in Tkinter.

6. Future Enhancements

The application can be improved by adding:

6.1 Expense Categories

Food, Travel, Rent, Shopping, Bills, etc.

6.2 Monthly and Yearly Reports

Automatically calculate:

Total spending

Category-wise breakdown

6.3 Graphs & Charts

Use matplotlib to display:

Pie charts

Bar graphs

Spending trends

6.4 Export Options

Allow user to export data to:

Excel

CSV

PDF

6.5 Authentication System

Add login/logout for user-specific expense tracking.

6.6 Cloud Backup

Sync data with Google Drive or cloud database.

6.7 Mobile App Version

Convert the project into a mobile application using Kivy or Flutter.

7. References

- Python Documentation – https://docs.python.org
- Tkinter Official Guide – https://tkdocs.com
- SQLite Documentation – https://sqlite.org/docs.html

- tkcalendar Library – https://github.com/j4321/tkcalendar
- StackOverflow discussions on Tkinter and TreeView widgets.