

# EXPENSE TRACKER REPORT

PYTHON PROJECT

## VITyarthi PROJECT

NAME : KAASHISH ADLAK

REG NO.: 25MIB10007

# 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

```

+-----+ | ExpenseTracker | +-----+ | - ID:
Integer | | - Date: Date | | - Payee: String | | - Description: String | | - Amount:
Float | | - ModeOfPayment: String | +-----+ | + addExpense() | | +
deleteExpense() | | + updateExpense() | | + fetchAll() | | + fetchOne() |
+-----+ +-----+ | Database |
+-----+ | + connect() | | + executeQuery() | | + commit() |
+-----+ +-----+ | GUI |
+-----+ | + renderForm() | | + renderTable() | | + clearFields() | |
+ showMessage() | +-----+

```

## DFD Level 0

```

+-----+ | USER | +-----+ | v
+-----+ | Expense Tracker | +-----+ | v
+-----+ | Database (SQLite) | +-----+

```

## DFD Level 1

```

+-----+ | USER | +-----+ |
+-----+ | | | | v v v v +-----+
+-----+ +-----+ +-----+ | Add Exp | | Edit Exp | | Delete Exp | |
View Expense | +-----+ +-----+ +-----+ +-----+ | | | |
+-----+ Write/Read +-----+ | v +-----+
SQLite Database | +-----+

```

## ERD Diagram

```

+-----+ | ExpenseTracker |
+-----+ | ID (PK) INTEGER | | Date DATETIME |
| Payee TEXT | | Description TEXT | | Amount FLOAT | | ModeOfPayment TEXT |
+-----+

```

## GUI Layout Diagram

```

+-----+ | EXPENSE TRACKER (Title
Bar) | +-----+ | DATA ENTRY PANEL
| TABLE PANEL | | +-----+ | +-----+ | Date:
[DateEntry] | +-----+ | | Payee: [Textbox] | | ID | Date | Payee
| ... | | | Description: [Textbox] | | +-----+ | | Amount:
[Textbox] | | Data Rows... | | | Mode: [Dropdown] | +-----+ | |
[Add Button] | | | [Convert to Words] | |
+-----+ | BUTTONS PANEL (Delete,
Edit, Clear, etc.) | +-----+

```



# 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)

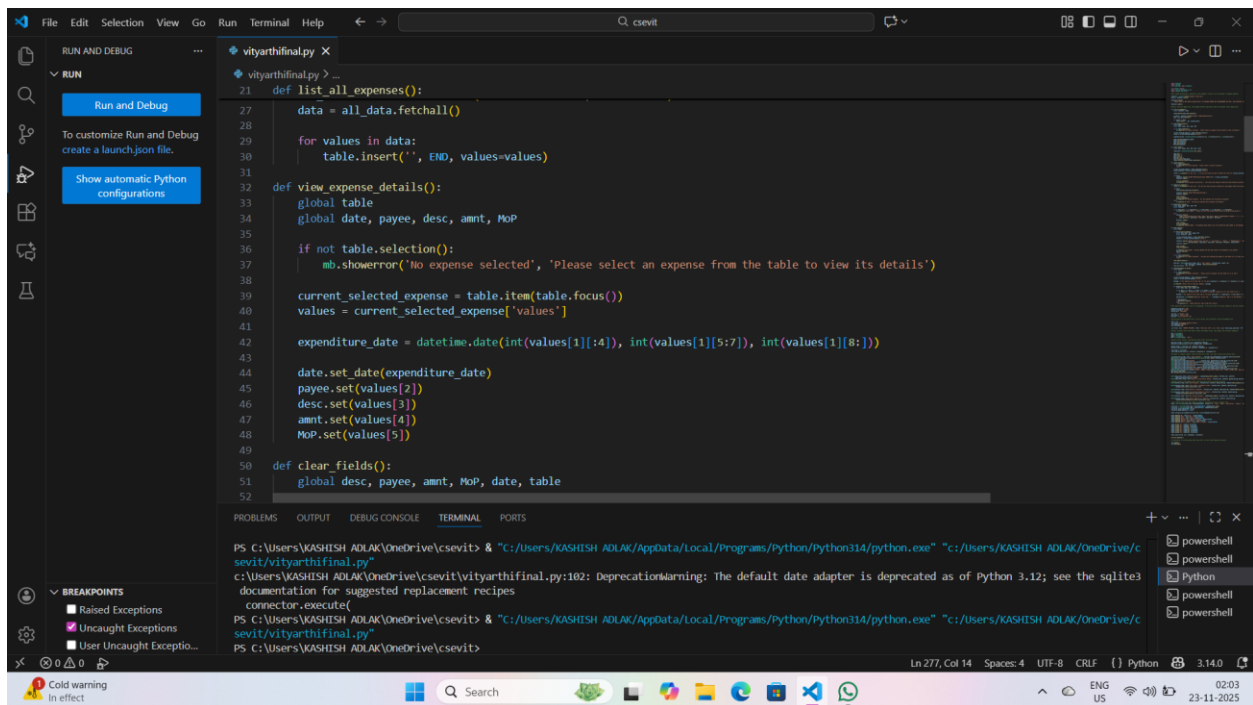
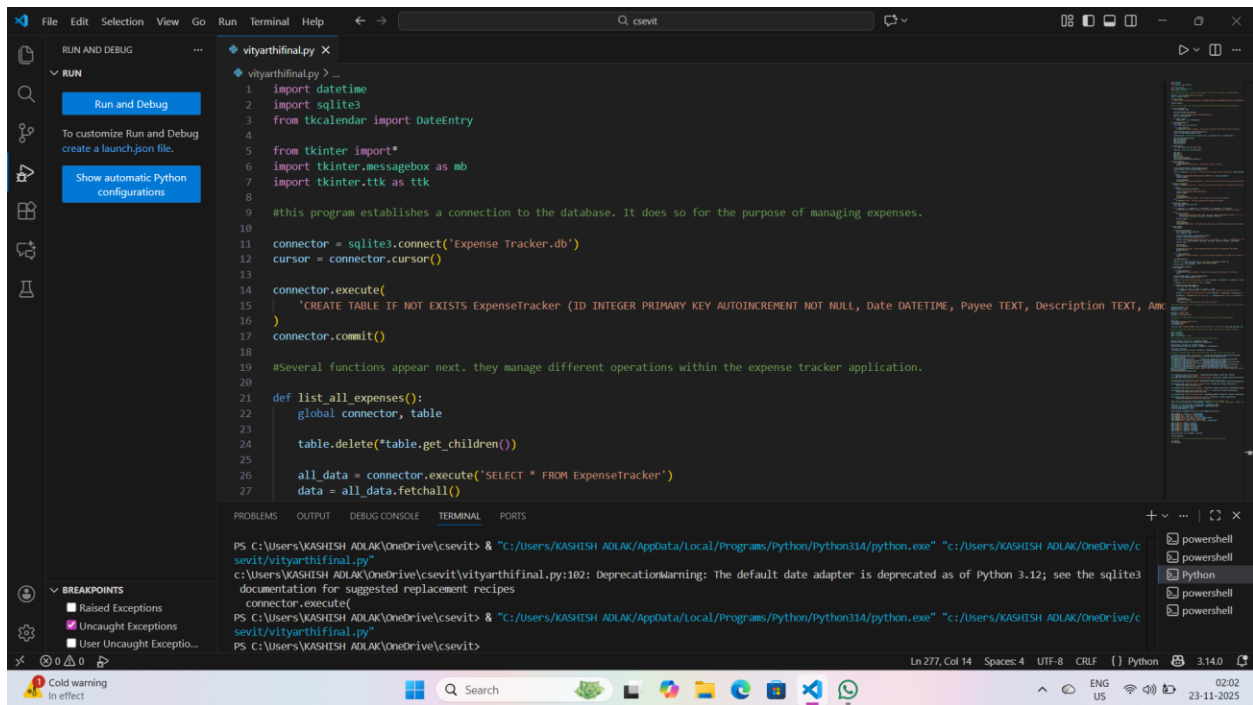
Buttons Frame (top center)

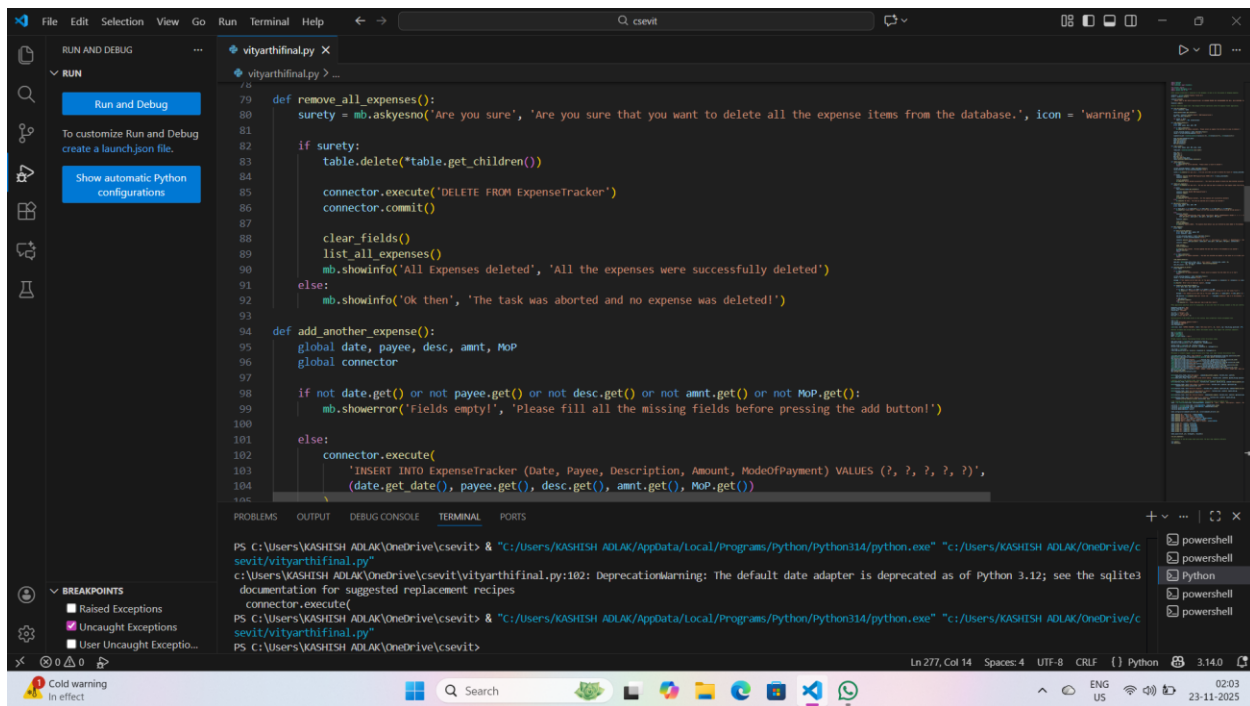
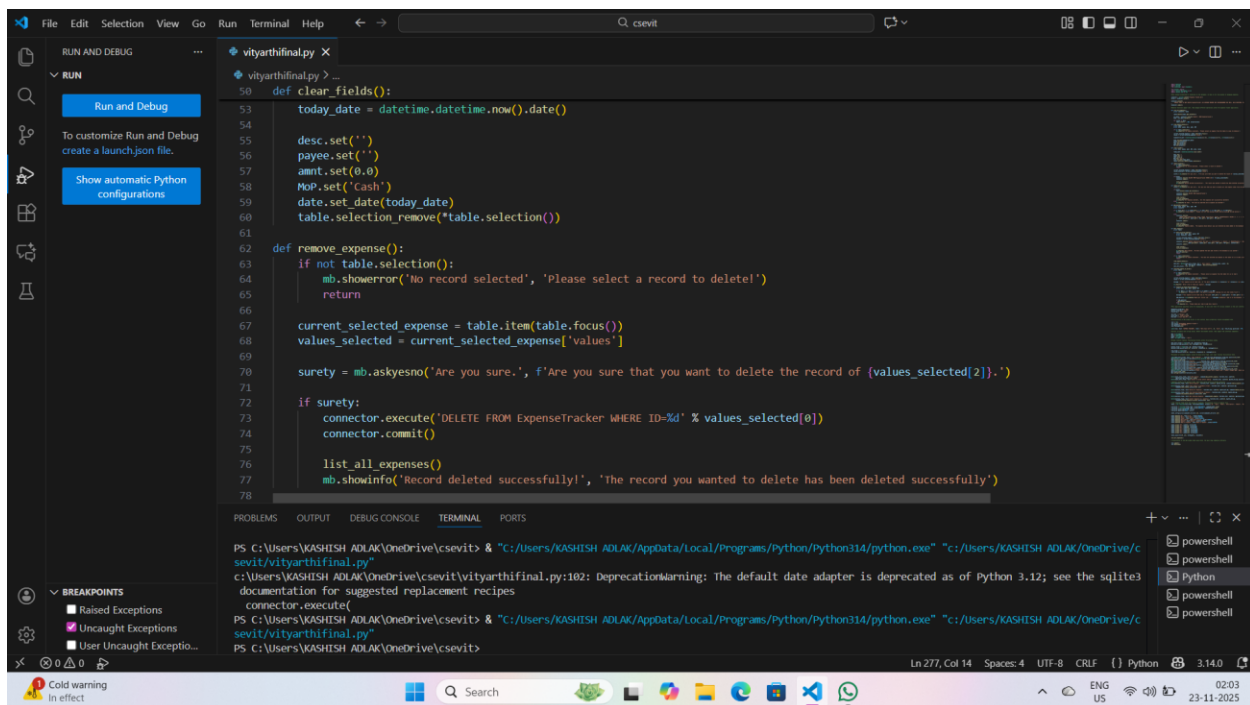
TreeView Frame (right side listing the expenses)

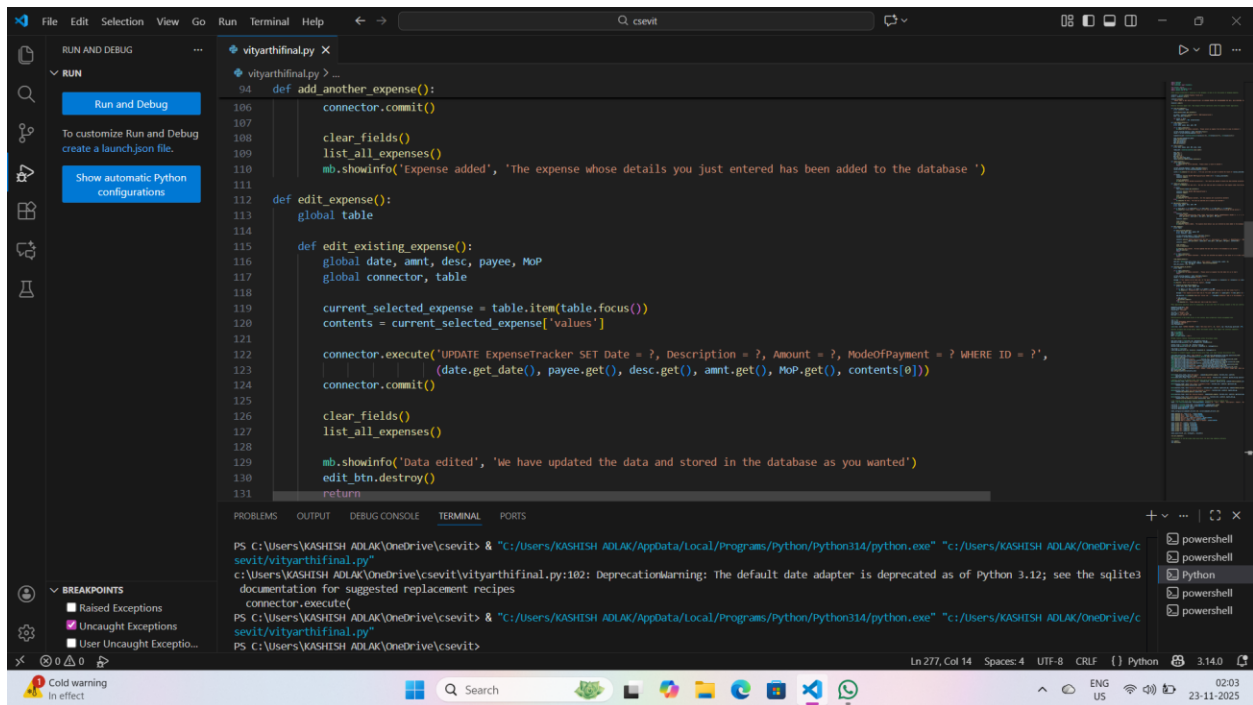
## 2.4 Application Flow

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.

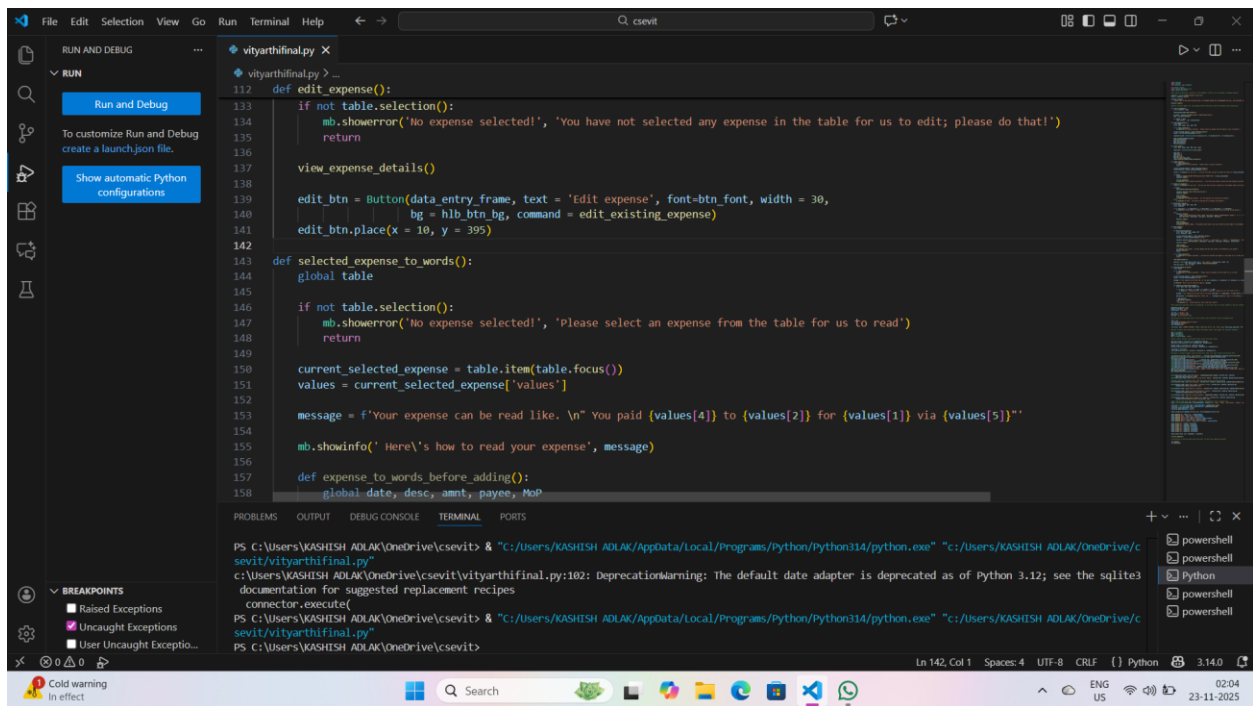
## RESULT



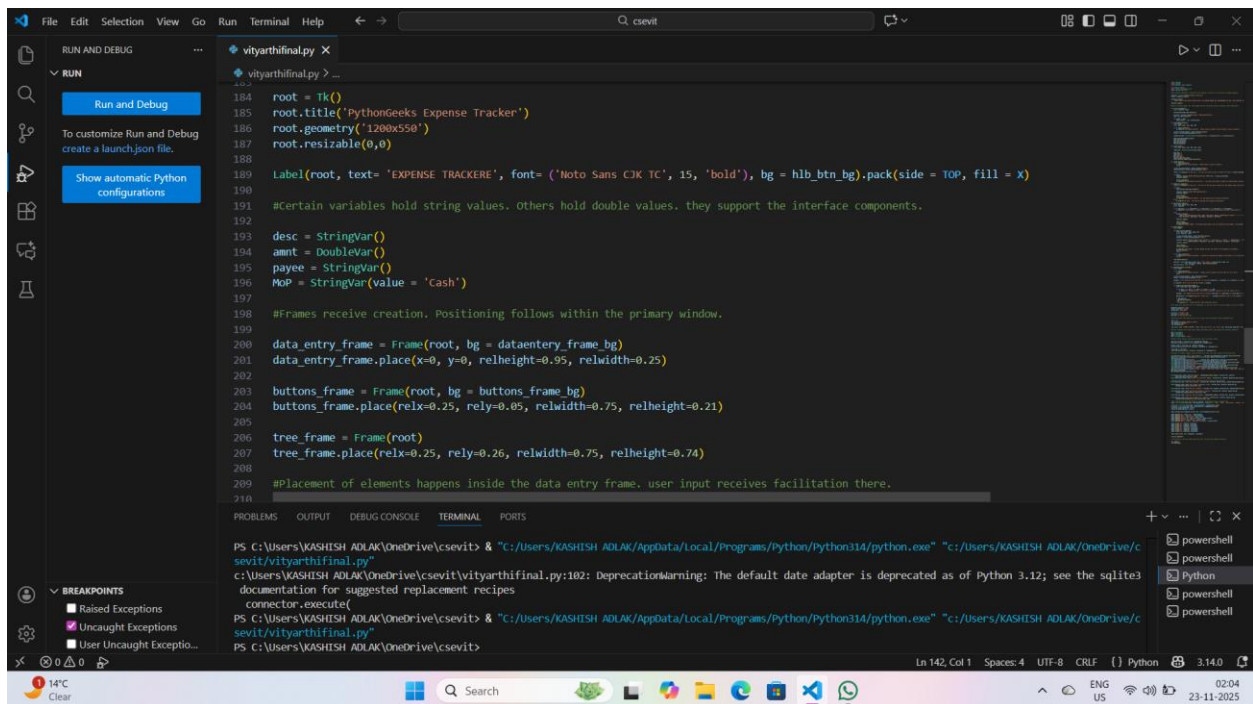
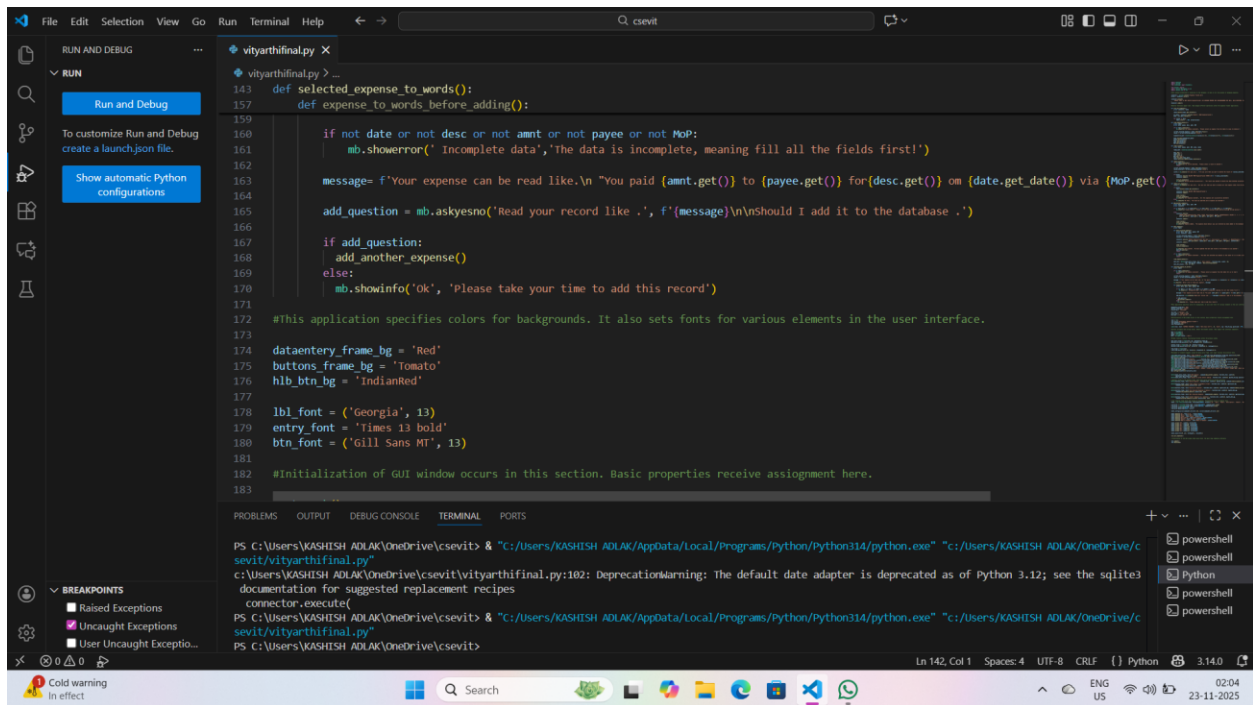


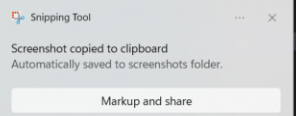


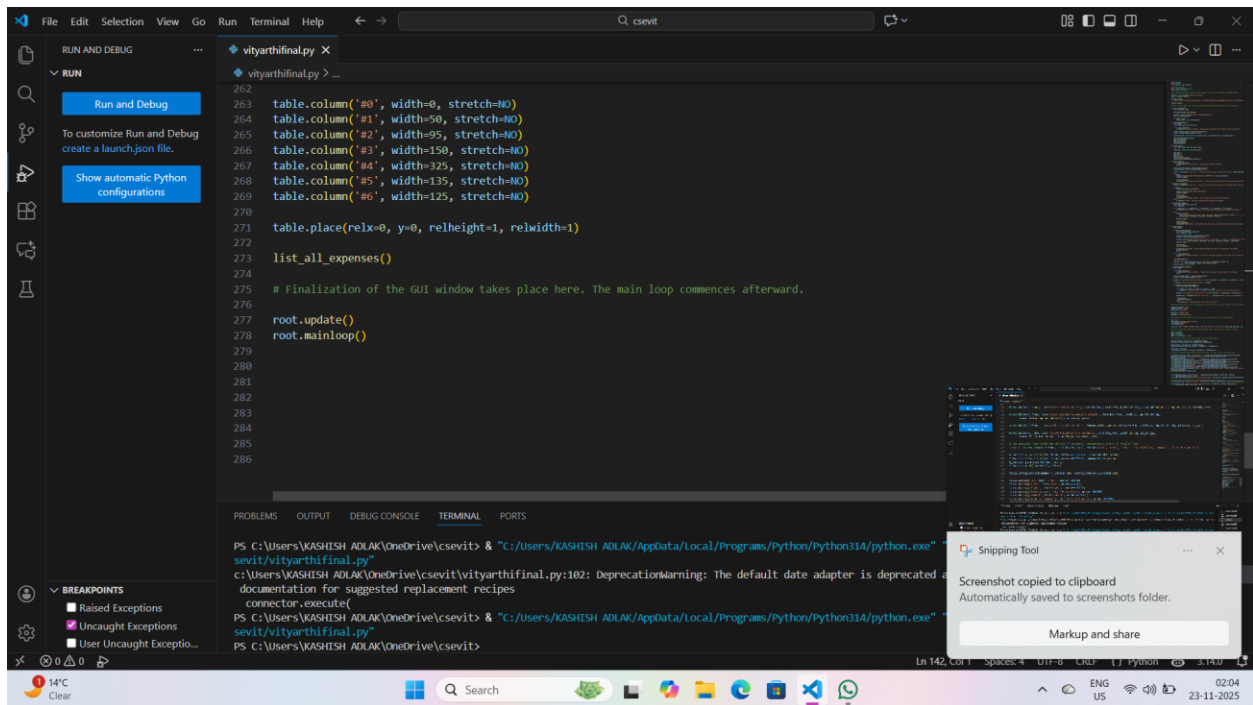
```
File Edit Selection View Go Run Terminal Help
vityarthifinal.py X
vityarthifinal.py > ...
94 def add_another_expense():
106     connector.commit()
107
108     clear_fields()
109     list_all_expenses()
110     mb.showinfo('expense added', 'The expense whose details you just entered has been added 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         mb.showinfo('Data edited', 'We have updated the data and stored in the database as you wanted')
129         edit_btn.destroy()
130     return
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



```
File Edit Selection View Go Run Terminal Help
vityarthifinal.py X
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
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```







## OUTPUT

[https://youtu.be/dzEkYdu8B\\_c?si=2iyzs4YyTzu](https://youtu.be/dzEkYdu8B_c?si=2iyzs4YyTzu)  
vHITA

### 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.