**LIBRARIES USED**

**AND THEIR PURPOSE**

**LIBRARIES USED**

**AND THEIR PURPOSE**

1. **CSV**:- To keep track of data in an organized way
2. **Datetime**:- To get the date and time

**LIST OF FIGURES**

**LIST OF FIGURES**

* **Figure 1 :-** Menu Screen
* **Figure 2.1 :-** Entry - 1
* **Figure 2.2 :-** Database Change - 1
* **Figure 3.1 :- E**ntry - 2
* **Figure 3.2 :-** Database Change - 2
* **Figure 4.1 :-** Entry Update - 1
* **Figure 4.2 :-** Enry Update Database Change
* **Figure 5 :-** Tracking Expense Screen and Output
* **Figure 6 :-** Saving file Screen

**REQUIREMENTS**

**SOFTWARE REQUIREMENTS**

* Python 3.7 or older installed on the device
* IDE to run the program

**HARDWARE REQUIREMENTS**

* Modern Operating System:
  + Windows 7 or 10
  + Mac OS X 10.11 or higher, 64-bit
  + Linux: RHEL 6/7, 64-bit (almost all libraries also work in Ubuntu)
* x86 64-bit CPU (Intel / AMD architecture). ARM CPUs are not supported.
* 4 GB RAM
* 5 GB free disk space

**ALGORITHM**

**ALGORITHM**

**Step 1: Check File Existence**

**1.1** Try opening the file "data\_file.csv".

**1.2** If the file doesn't exist:

**1.2.1** Create a new file named "data\_file.csv".

**1.2.2** Write the header row: ["S.No.", "Expense", "Amount", "Type", "Date", "Time", "Remarks"] to the new file.

**Step 2: Main Menu Loop**

**2.1** Display a menu with these options:

- Add a new entry (option 1)

- Update an entry (option 2)

- Track expenditure (option 3)

- Save data to file (option 4)

- Exit (option 5)

**2.2** Ask the user to enter their choice (1-5).

**2.3** If the user chooses:

- Option 1: Proceed to Step 3 (Add New Entry).

- Option 2: Proceed to Step 4 (Update Entry).

- Option 3: Proceed to Step 5 (Track Expenditure).

- Option 4: Proceed to Step 6 (Save Data to File).

- Option 5: Exit the program.

2.4. Go back to Step 2.2 to display the menu again until the user chooses to exit.

**Step 3: Add New Entry (Option 1)**

**3.1** Ask the user for:

- Expense category

- Amount (as a number)

- Type (1 for Asset, 2 for Liability)

- Date (optional, if not provided use current date)

- Time (optional, if not provided use current time)

- Remarks (optional)

**3.2** Find the next available serial number (S.No.) by checking the last entry in the data file.

**3.3** Create a new entry list with the information from 3.1: [S.No., expense category, amount, type, date, time, remarks].

**3.4** Open the data file in append mode.

**3.5** Write the new entry list from 3.3 to the data file.

**3.6** Close the data file.

**3.7** Inform the user that the entry has been added successfully.

**Step 4: Update Entry (Option 2)**

**4.1** Ask the user to enter the serial number of the expense to update.

**4.2** Find the corresponding entry in the data file.

**4.3** Ask the user which field they want to update (expense category, amount, type, date, time, or remarks).

**4.4** Allow the user to modify the chosen field.

**4.5** Open the data file in write mode.

**4.6** Read all entries from the data file except the one being updated.

**4.7** Create a new list to hold the updated data.

**4.8** For each entry in the data file:

- If it's not the entry being updated, add it to the new list without changes.

- If it's the entry being updated, add it to the new list with the modified field from 4.4.

**4.9** Write the updated entry list from 4.8 back to the data file.

**4.10** Close the data file.

**4.11** Inform the user that the entry has been updated successfully.

**Step 5: Track Expenditure (Option 3)**

**5.1** Show a sub-menu with these options:

- Get monthly expenditure (option 1)

- Get yearly expenditure (option 2)

- Get total expenditure (option 3)

- Get expenditure on a particular category (option 4)

**5.2** Ask the user to choose their desired option (1-4).

**5.3** Based on the chosen option:

- Option 1 (Monthly):

- Ask the user for the month (numerical MM format).

- Find entries in the data file matching the specified month.

- Calculate and display the total expenditure for the month.

- Option 2 (Yearly):

- Ask the user for the year (numerical YYYY format).

- Find entries in the data file matching the specified year.

- Calculate and display the total expenditure for the year.

- Option 3 (Total):

- Read all entries from the data file except the header.

- Calculate and display the total expenditure across all entries.

Option 4 (Category):

- Ask the user for the category name.

- Find entries in the data file matching the specified category.

- Calculate and display the total expenditure for that category.

**Step 6: Save Data to File (Option 4**)

**6.1** Open the data file in write mode.

**6.2** Read all entries from the program's memory.

**6.3** Write all entries to the data file (overwriting the existing data).

**6.4** Close the data file.

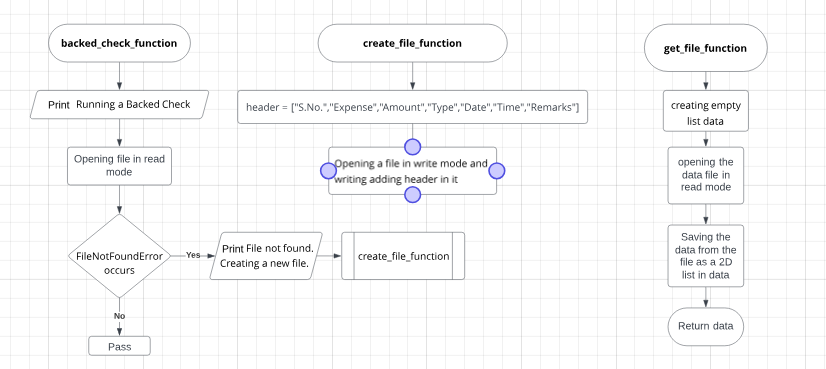
**6.5** Inform the user that the data has been saved successfully.

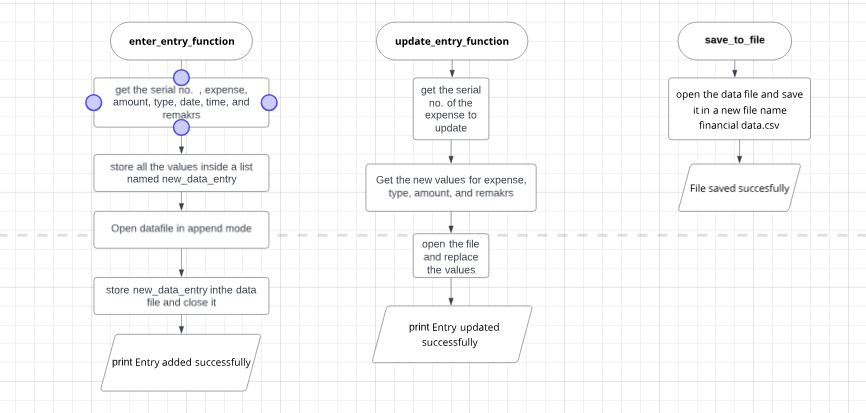
**Step 7: Exit (Option 5)**

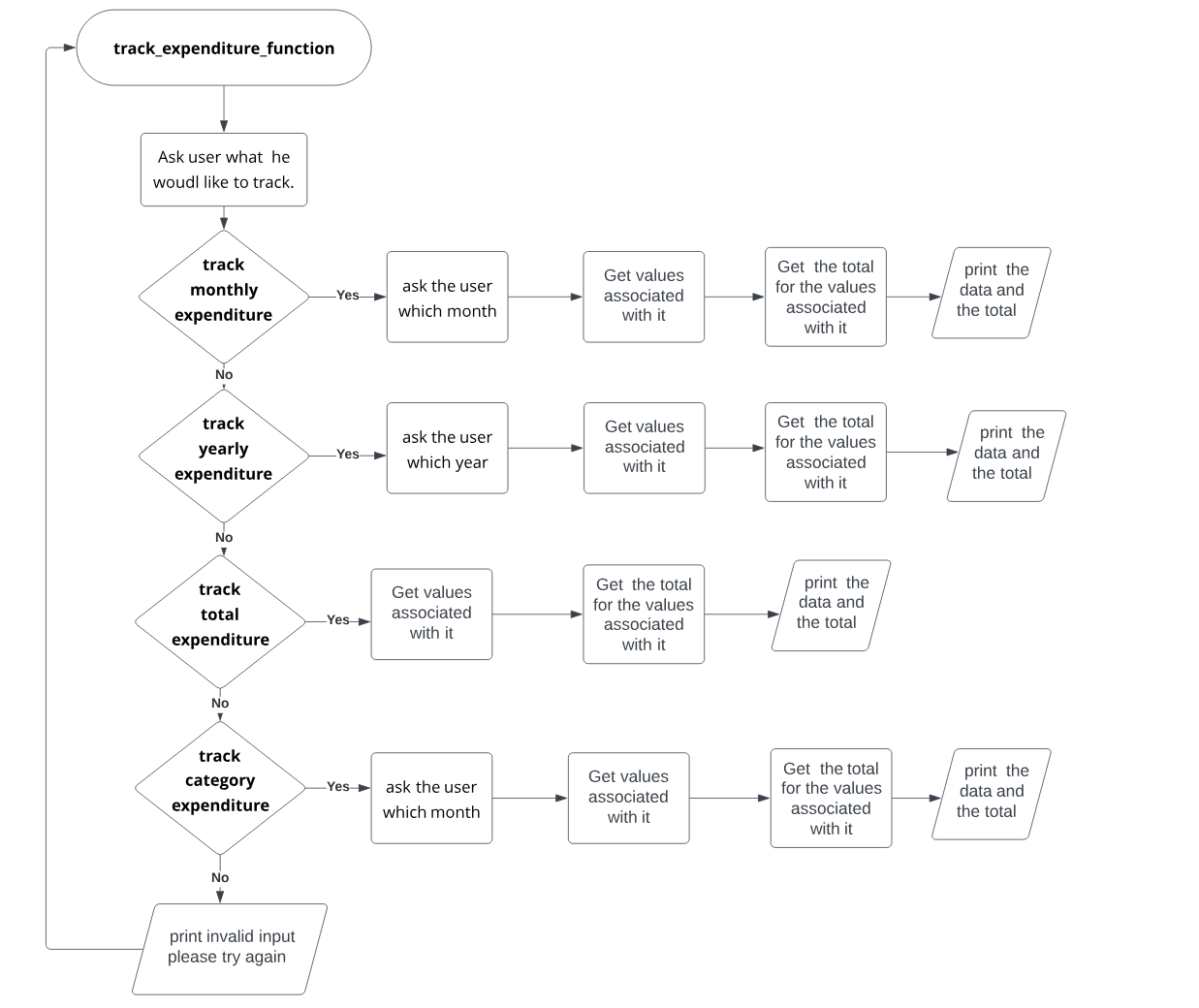
**7.1** End the program.

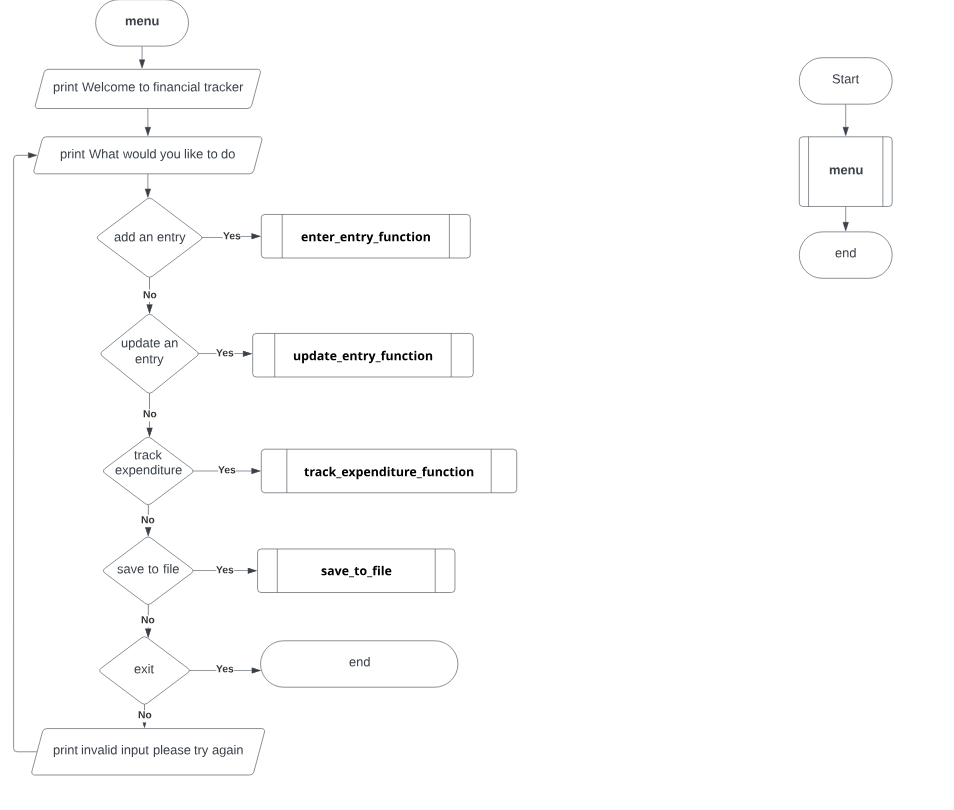
**FLOWCHART**

**FLOWCHART**









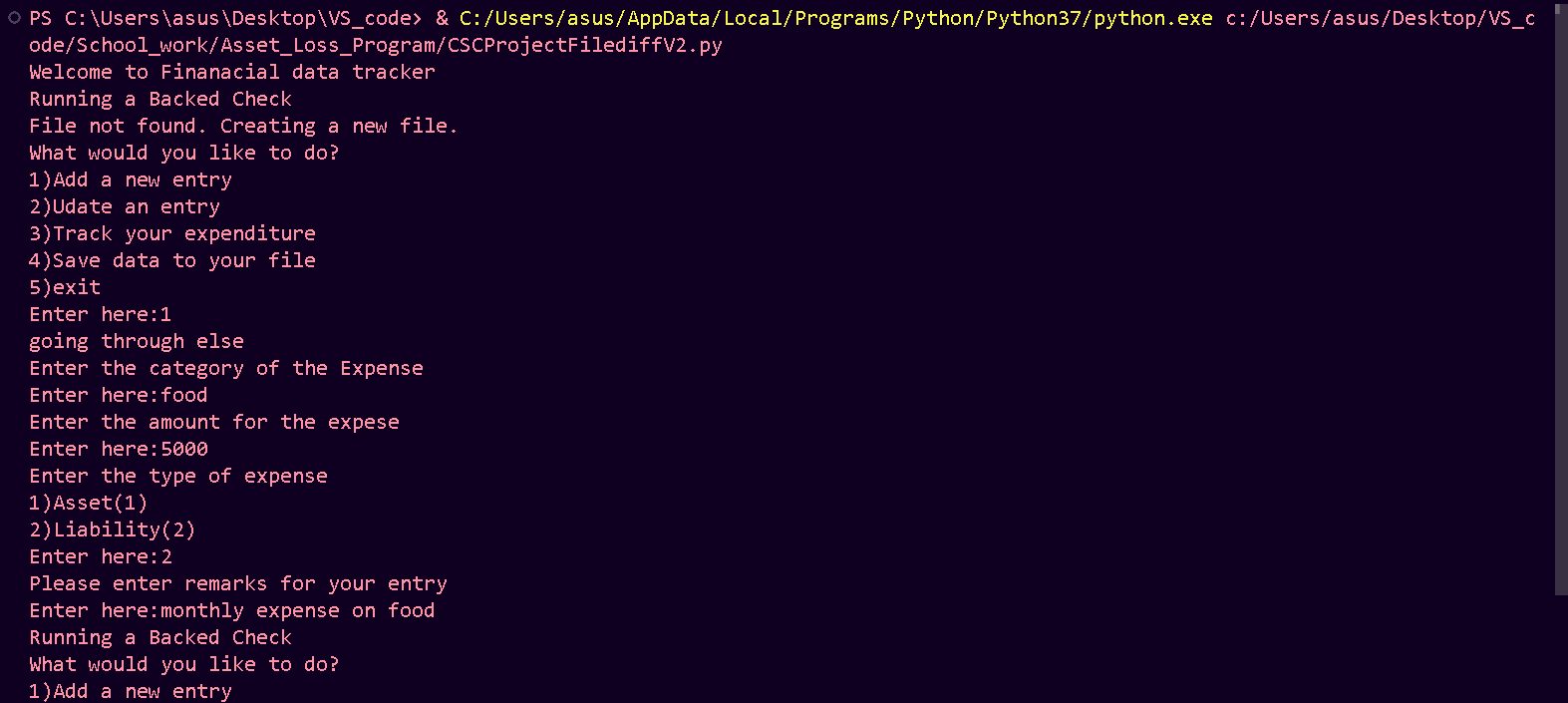
**INTERFACE**

**INTERFACE**

**Menu Screen**

**Figure 1**

Entry - 1

**Figure 2.1**

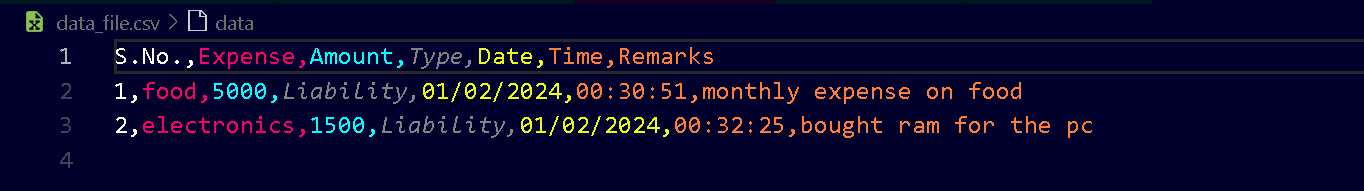
**Database Change - 1**

**Figure 2.2**

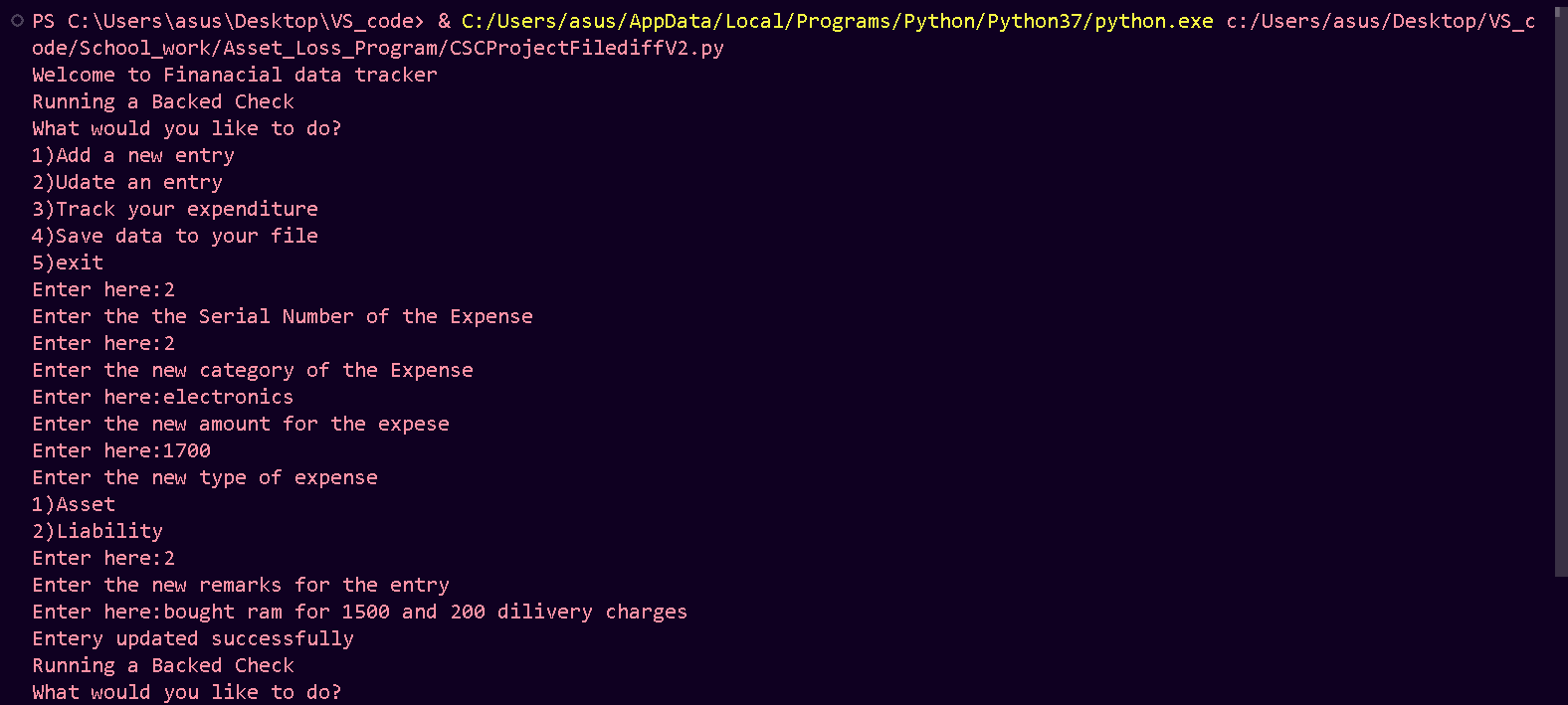
**Entry - 2**

**Figure 3.1**

**Database Change - 2**

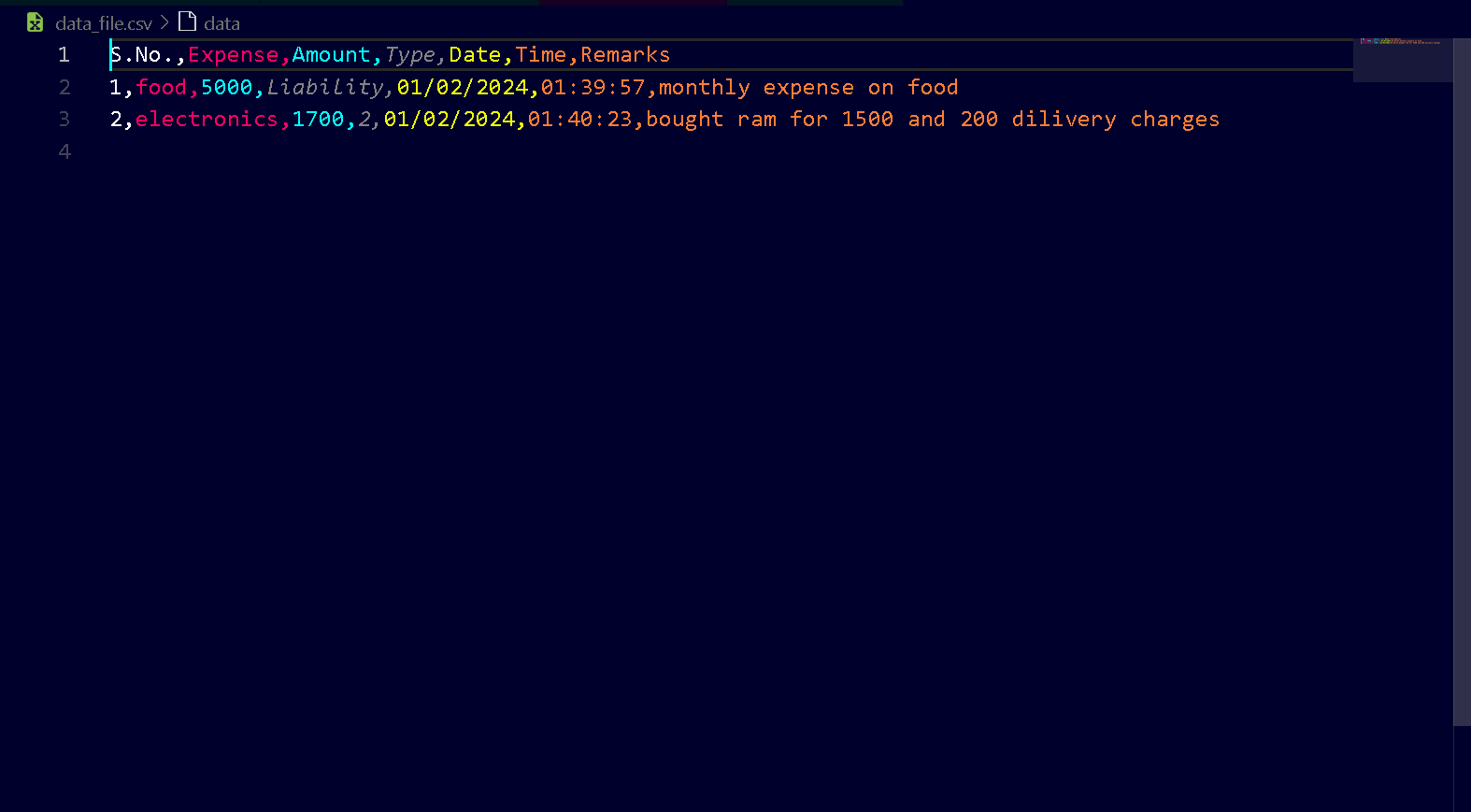
**Figure 3.2**

**Entry Update - 1**



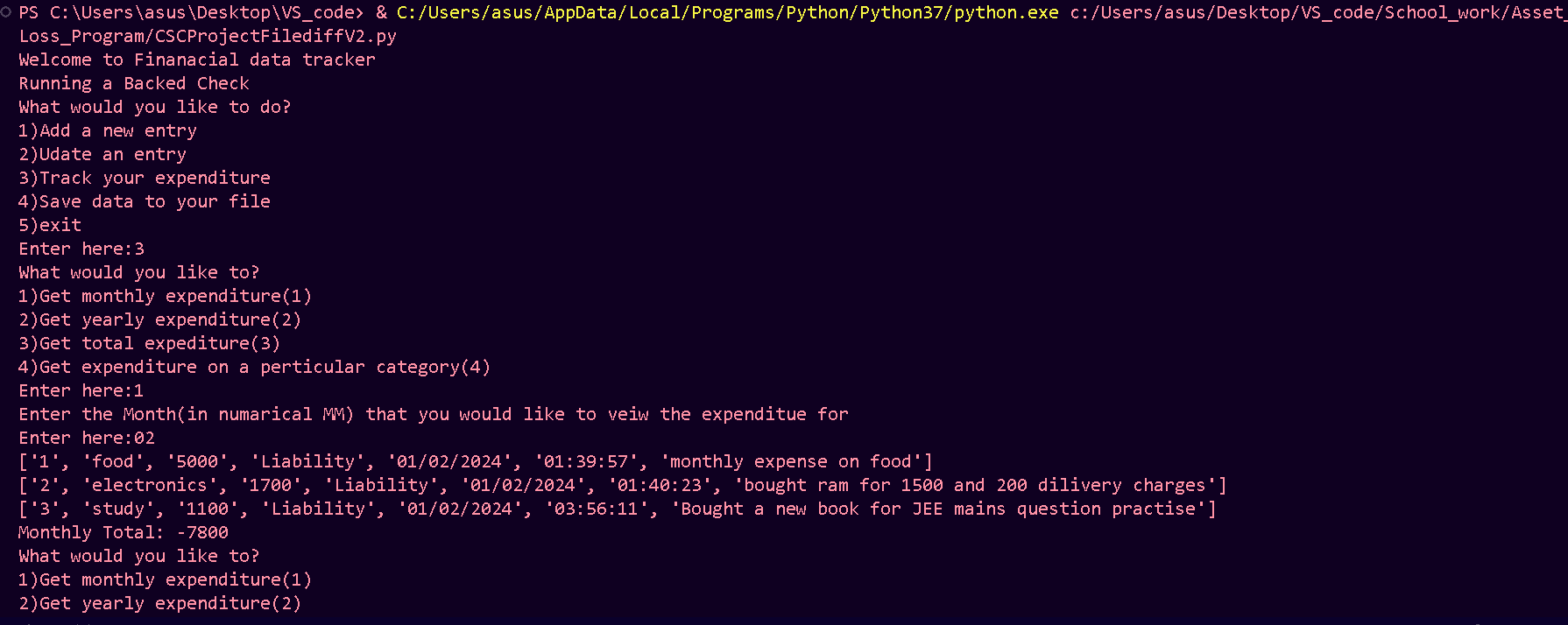
**Figure 4.1**

**Entry Update Database Change**

**Figure 4.2**

**T**

**racking Expense Screen and Output**

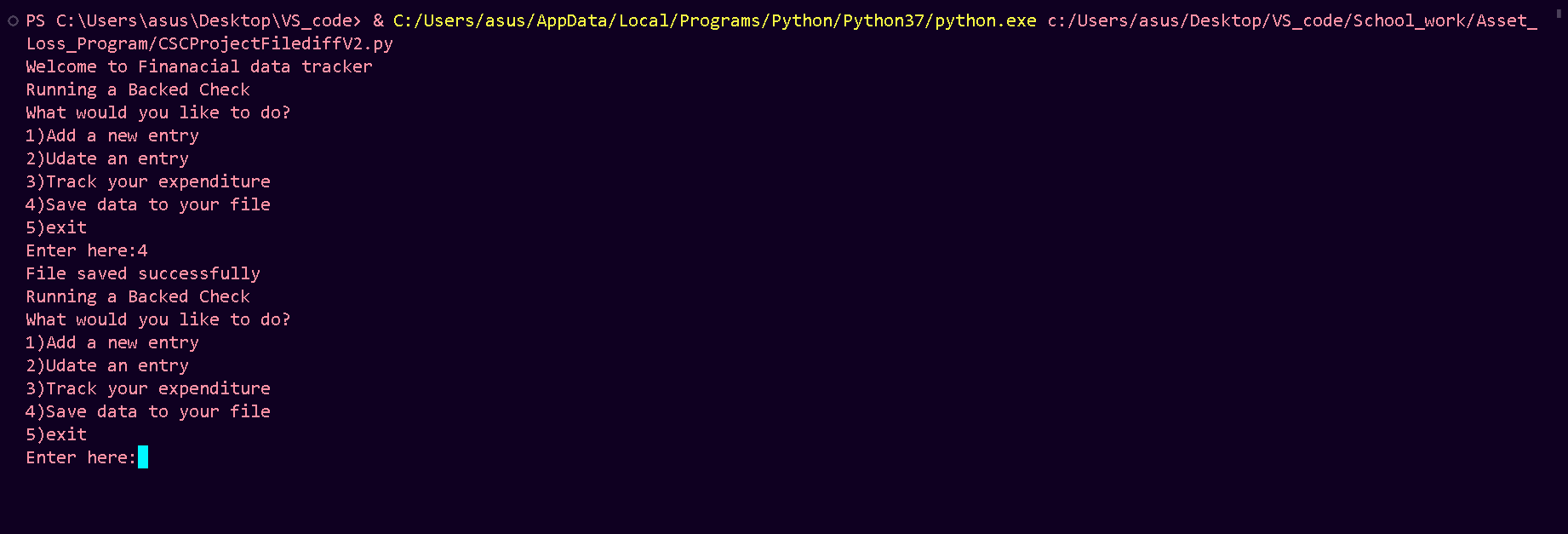
****

**Figure 5**

**[www.cbseportal.com](http://www.cbseportal.com)**

**[www.cbseportal.com](http://www.cbseportal.com)**

**Saving file Screen**

**[](http://www.cbseportal.com)**

**[Figure 6m](http://www.cbseportal.com)**

**Coding**

**Coding**

import csv

from datetime import datetime

def create\_file\_function():

header = ["S.No.","Expense","Amount","Type","Date","Time","Remarks"]

with open('data\_file.csv', mode='w', newline='') as file:

writer = csv.writer(file)

writer.writerow(header)

file.close()

def backed\_check\_function():

print("Running a Backed Check")

try:

with open('data\_file.csv', mode='r',newline='') as file:

pass

except FileNotFoundError:

print("File not found. Creating a new file.")

create\_file\_function()

def get\_file\_function():

data = []

with open('data\_file.csv', mode='r',newline='') as file:

reader = csv.reader(file)

for i in reader:

data.append(i)

return data

def enter\_entry\_function():

temp\_data = get\_file\_function()

length = len(temp\_data)

if length > 1:

sno = length

else:

sno = 1

expense = input('Enter the category of the Expense\nEnter here:')

amount = int(input('Enter the amount for the expese\nEnter here:'))

type = 0

while True:

n = input('Enter the type of expense\n1)Asset(1)\n2)Liability(2)\nEnter here:')

if n == "1":

type = "Asset"

break

elif n == "2":

type = "Liability"

break

else:

print("Invalid input please try again")

continue

now = datetime.now()

date = now.strftime("%d/%m/%Y")

time = now.strftime("%H:%M:%S")

remarks = input("Please enter remarks for your entry\nEnter here:")

new\_data\_entry = [sno, expense, amount, type , date, time , remarks]

with open('data\_file.csv', 'a' , newline='') as file:

writer = csv.writer(file)

writer.writerow(new\_data\_entry)

file.close()

print("Entry added successfully")

def update\_entry\_function():

temp\_data = get\_file\_function()

get\_sno = int(input("Enter the the Serial Number of the Expense\nEnter here:"))

get\_expense = input('Enter the new category of the Expense\nEnter here:')

get\_amount = int(input('Enter the new amount for the expese\nEnter here:'))

get\_type = 0

while True:

n = input('Enter the new type of expense\n1)Asset(1)\n2)Liability(2)\nEnter here:')

if n == "1":

get\_type = "Asset"

break

elif n == "2":

get\_type = "Liability"

break

else:

print("Invalid input please try again")

continue

get\_date = temp\_data[get\_sno][4]

get\_time = temp\_data[get\_sno][5]

get\_remarks = input('Enter the new remarks for the entry\nEnter here:')

temp\_data[get\_sno] = [get\_sno, get\_expense, get\_amount, get\_type ,get\_date, get\_time , get\_remarks]

with open('data\_file.csv', mode='w', newline='') as file:

writer = csv.writer(file)

writer.writerows(temp\_data)

file.close()

print("Entery updated successfully")

def track\_expenditure\_function():

while True:

tef\_ans = int(input("What would you like to?\n1)Get monthly expenditure(1)\n2)Get yearly expenditure(2)\n3)Get total expediture(3)\n4)Get expenditure on a perticular category(4)\nEnter here:"))

if tef\_ans == 1:

temp\_data = get\_file\_function()

temp\_data.pop(0)

temp\_ans = input("Enter the Month(in numarical MM) that you would like to veiw the expenditue for\nEnter here:")

temp\_month\_data = []

temp\_month\_data\_total = 0

for i in temp\_data:

month = i[4].split('/')

if month[1] == temp\_ans:

temp\_month\_data.append(i)

if i[3] == "Asset":

temp\_month\_data\_total += int(i[2])

elif i[3] == "Liability":

temp\_month\_data\_total -= int(i[2])

else:

print("Invalid entries in file please recheak the file or create a new one")

exit()

for i in temp\_month\_data:

print(i)

print("Monthly Total:" , temp\_month\_data\_total)

elif tef\_ans == 2:

temp\_data = get\_file\_function()

temp\_data.pop(0)

temp\_ans = input("Enter the Year(in numarical YYYY) that you would like to veiw the expenditue for\nEnter here:")

temp\_year\_data = []

temp\_year\_data\_total = []

for i in temp\_data:

year = i[5].split('/')

if year[2] == temp\_ans:

temp\_year\_data.append(i)

if i[3] == "Asset":

temp\_year\_data\_total += int(i[2])

elif i[3] == "Liability":

temp\_year\_data\_total -= int(i[2])

else:

print("Invalid entries in file please recheak the file or create a new file")

exit()

for i in temp\_year\_data:

print(i)

print("Yearly Total:" , temp\_year\_data\_total)

elif tef\_ans == 3:

temp\_data = get\_file\_function()

temp\_data.pop(0)

temp\_total\_expenditure\_data\_total = 0

for i in temp\_data:

if i[3] == "Asset":

temp\_total\_expenditure\_data\_total += int(i[2])

elif i[3] == "Liability":

print(type(i[2]))

temp\_total\_expenditure\_data\_total -= int(i[2])

else:

print("Invalid entries in file please recheak the file or create a new file")

for i in temp\_data:

print(i)

print("Total:" , temp\_total\_expenditure\_data\_total)

elif tef\_ans == 4:

temp\_data = get\_file\_function()

temp\_data.pop(0)

t = input("Enter the category of the expense you would like to view: ")

temp\_category\_data = []

temp\_category\_data\_total = 0

for i in temp\_data:

if i[1]==t:

temp\_category\_data.append(i)

if i[3] == "Asset":

temp\_category\_data\_total += int(i[2])

elif i[3] == "Liability":

temp\_category\_data\_total -= int(i[2])

else:

print("Invalid entries in file please recheak the file or create a new file")

for i in temp\_category\_data:

print(i)

print("Category Total:" , temp\_category\_data\_total)

else:

print('invalid input please try again')

continue

def save\_to\_file():

temp\_data = get\_file\_function()

with open("Finanacial Data.csv", "w",newline='') as file:

writer = csv.writer(file)

writer.writerows(temp\_data)

file.close()

print("File saved successfully")

def menu():

print('Welcome to Finanacial data tracker')

while True:

backed\_check\_function()

print('What would you like to do?')

menu\_answer= int(input('1)Add a new entry\n2)Udate an entry\n3)Track your expenditure\n4)Save data to your file\n5)exit\nEnter here:'))

if menu\_answer == 1:

enter\_entry\_function()

elif menu\_answer == 2:

update\_entry\_function()

elif menu\_answer == 3:

track\_expenditure\_function()

elif menu\_answer == 4:

save\_to\_file()

elif menu\_answer == 5:

exit()

else:

print('Invalid input please try again')

continue

menu()

**LIMITATIONS**

**LIMITATIONS**

* Requires prior knowledge to operate
* Requires resources to be downloaded and saved on the device
* The accuracy of your data depends entirely on manual entry, making it susceptible to human error and forgetting to log transactions.
* Integration with bank accounts or credit cards for automatic transaction import may not be available for all institutions or require manual setup and maintenance.
* Categorizing expenses might be subjective and inconsistent, leading to inaccurate insights and skewed reports.

**FUTURE SCOPE**

**FUTURE SCOPE**

* Automatic transaction import with advanced categorization and data enrichment using artificial intelligence (AI) for improved accuracy and efficiency.
* AI-powered features like automatic bill payments, expense reminders, and personalized budgeting suggestions based on spending patterns.
* Granular control over data sharing and privacy settings to empower users and build trust.
* Fingerprint or facial recognition for secure and convenient access to the program.
* Integration with educational materials and tools to promote financial literacy and responsible money management.
* Integration with goal-setting tools and investment tracking to support achieving long-term financial objectives.

**BIBLIOGRAPHY**

* **<https://www.python.org>**
* **<https://www.geeksforgeeks.org>**
* **<https://www.programiz.com>**
* **<https://www.google.co.in>**
* **<https://scholar.google.com>**