Task 2 - A Report on Exp\_Tracker.py, a comprehensive Expense Tracking software solution.

David Rogers – 1193292

University of the Sunshine Coast

ICT703 – Programming

Prepared For: Andrew Lang

17 January 2025

Contents

[Introduction 3](#_Toc187827878)

[Problem Description 4](#_Toc187827879)

[Overview 4](#_Toc187827880)

[Required Features 4](#_Toc187827881)

[Software – Expense Tracker 5](#_Toc187827882)

[Features 5](#_Toc187827883)

[Limitations 5](#_Toc187827884)

[System Instructions 5](#_Toc187827885)

[Database 8](#_Toc187827886)

[Structure 8](#_Toc187827887)

[Relationships 8](#_Toc187827888)

[Azure hosting 8](#_Toc187827889)

[Runtime Dependencies 9](#_Toc187827890)

[Standard Python Modules used in Expense Tracker 9](#_Toc187827891)

[External Modules used by Expense Tracker 9](#_Toc187827892)

[Pseudocode 11](#_Toc187827893)

[Initial 11](#_Toc187827894)

[Final 15](#_Toc187827895)

[Test Cases 20](#_Toc187827896)

[Function addTrans() 20](#_Toc187827897)

[Function addCat() 26](#_Toc187827898)

[Function updateByTranID(tranID) 31](#_Toc187827899)

[Function deleteByTranID(tranID) 37](#_Toc187827900)

[Conclusion 41](#_Toc187827901)

[References 42](#_Toc187827902)

[Appendix 43](#_Toc187827903)

[Supporting Queries 43](#_Toc187827904)

## Introduction

The Expense Tracker program is a Python-based software solution designed to help individuals track their expense transactions against a pre-set budget. It also provides a range of reports to track their spending and gauge their financial position over time. This report provides a detailed overview of the program, including a description of its functionality and components, a breakdown of its pseudocode, and a test plan to help ensure its reliability and effectiveness.

The primary objective of the Expense Tracker program is to allow users to input, categorise, and view their expense transactions over time. It offers features such as adding new transaction records, updating and deleting transactions, viewing a summary of existing transactions, as well as categorising expenses into different types (e.g., Home, Entertainment, Vehicle). Additionally, the program is designed to provide an intuitive interface, making it accessible to both novice and experienced users.

This report provides a description of the Expense Tracker program, outlining its key features and how it operates. It also includes information on the Microsoft SQL Azure-based database that supports the program, as well as the various run-time dependencies and instructions for their installation.

The report then presents both the initial and final versions of the program’s pseudocode. This serves as a high-level blueprint of the program’s structure and logic. Lastly, a test plan for a sample of the program’s functions is provided. Its aim is to ensure that these functions perform as intended, with a focus on validating their usability and accuracy.

## Problem Description

### Overview

The requirements for the Expense Tracker software are described as follows,

*An expense tracker is a python program designed to help users manage and monitor their expenses. By inputting data about their spending, users can categorise expenses, track their financial habits, and make informed decisions to better manage their budgets in the future.*

### Required Features

* Allow User login with a secure password to the Expense Tracker system -
  + Users can create a new account with their name, a secure password and a Budget amount that will be monitored throughout their interactions with the system.
  + Multiple users can access the system, each with their own secure Login, Budget, and Expense Transactions.
* Account Management -
  + Allows users to add new Expense Transactions. These will include details such as Expense Category, Date and Time, Amount, and a Description.
  + Allows Users to search through Expense Transaction records by Category, Date or Time.
  + Once the required Expense Transaction is found, the system should allow users to Update the details of the Transaction or Delete the Transaction.
  + Allows users to add new Expense Categories and to Update or Delete Categories if required.
  + Allows users to monitor their total Expense Transaction Amount in relation to their pre-set Budget as new Transactions are Added or Transactions are Updated.
* Reporting -
  + Provide the ability for users to run a range of comprehensive Reports on the state of their Expenses based on Category, Date or Time.
  + Offer the user the ability to export these reports to an external file.

## Software – Expense Tracker

### Features

The Expense Tracker software meets the requirements in the following ways -

* It allows multiple users to securely log into the system and maintains separate expense records for each user.
* It allows users to Add, Update and Delete their Expense Transactions.
* It allows users to Add, Update and Delete Categories, where possible (refer Limitations).
* It provides the user with the ability to set a budget limit. It also monitors the total expenses in relation to that budget limit with each successive Expense Transaction. Users are given a warning when they reach 90% of the budget allowance as well as when they exceed it.
* It provides the user with detailed reports to monitor their spending habits in relation to an expense category, as well as reports based on the time and date range of target expenses.

### Limitations

The Expense Tracker software is dedicated to monitoring a user’s incurred expenses. It does not, therefore, provide a way to monitor the user’s income or revenue streams compared to those expenses.

Additionally, it should be noted that once a Category has been assigned to an Expense transaction for a user, it cannot be deleted via the software unless all transactions (for all users) associated with that Category are first deleted. It is suggested in the software that users contact the administrator so all users can be notified of the pending change, and individual users can decide if their relevant expenses can be deleted.

The program also allows all users to change the name of a Category. This change will be applicable to all users of the program.

### System Instructions

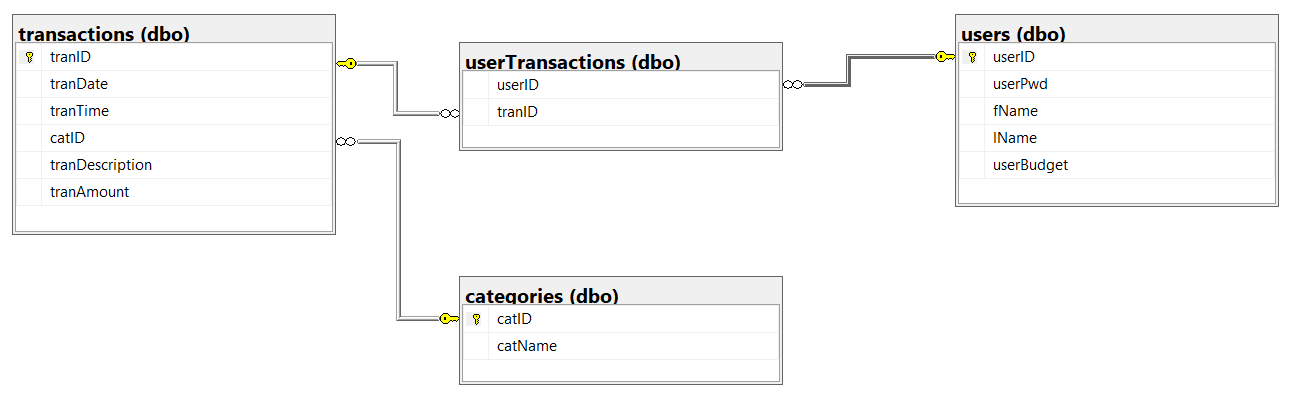
Expense Tracker is a menu-driven program. It is designed to be intuitive for beginners and seasoned users alike. Menus are traversed by pressing the corresponding letter (in brackets) for the desired action. A sample process flow is as follows -

* Banner Screen
  + The user may (L)OGIN with their existing user ID and password; or
  + The user may decide to (C)REATE a new account, providing their details and a budget limit that will be tracked automatically by the program as they enter and change their Expense Transactions.
* Main Menu
  + From here the users can access the -
    - (T)RANSACTIONS menu
    - (C)ATEGORIES menu
    - (B)UDGET menu
    - (R)EPORTS menu
    - (Q)UIT the program
* Transactions Menu
  + From here the user can -
    - (A)DD a new Expense Transaction; or
    - (S)EARCH for existing transactions by their Category, Time, or Date.
      * Once an existing transaction is found, the user can choose to -
        + (U)PDATE the transactions details; or
        + (D)ELETE the transaction.
* Categories Menu
  + From here the user can -
    - (A)DD a new expense Category; or
    - (U)PDATE a Category name; or
    - (D)ELETE a Category.
  + It should be noted that the program will only allow a Category to be deleted if it is no longer associated with any transactions for any user.
  + Updating a Category name will update that name for all users.
* Budget Menu
  + From here the user can -
    - (C)HECK the current budget amount, as well as their current expenditure in relation to that amount; or
    - (U)PDATE their budget amount to reflect a change in circumstances.
* Reports Menu
  + From here the user can run various detailed reports and save these to an external file for later review. The available reports are -
    - (1) All Transactions – a detailed report of all the users current Expense Transactions, as well as a review of their total expenditure compared to their budget limit.
    - (2) Transactions By Category – the user is given the opportunity to choose a particular Category, and the system will provide a detailed report of all transactions the user has in that category. It provides a total of those expenses as well as a review of their total expenditure compared to their budget limit.
    - (3) Transactions By Date – the user is given the opportunity to enter a date range and the system will provide a detailed report of all transactions that occurred between those dates. It provides a total of those expenses as well as a review of their total expenditure compared to their budget limit.
    - (4) Transactions By Time – the user is given the opportunity to enter a time range on a particular date and the system will provide a detailed report of all transactions that occurred between those times on that date. It provides a total of those expenses as well as a review of their total expenditure compared to their budget limit.
* Quit
  + Allows the user to (Q)uit the program when finished.

## Database

### Structure

The Exp\_Tracker SQL Server database has the following structure. It is comprised of 4 tables – *transactions*, *userTransactions*, *categories* and *users*.



### Relationships

The Exp\_Tracker database uses the following relationships to associate its data -

* A one-to-many relationship between the *categories* table and the *transactions* table, allowing for one Category to be used in many Transactions.
* A many-to-many relationship between the *transactions* table and the *users* table via the *userTransactions* table. This allows for many Users to be involved with many different Transactions.

### Azure hosting

The Exp\_Tracker database is housed in the Microsoft Azure cloud. This allows for portability of the Expense Tracker software. It can be run anywhere that has a Python interpreter and an Internet connection.

## Runtime Dependencies

### Standard Python Modules used in Expense Tracker

* **msvcrt**:
  + **Purpose**: This module provides access to some of the functions in the Microsoft Visual C Runtime Library.
  + **Usage**: It is used in Expense Tracker for detecting key presses in a Windows environment to allow the program to pause and restart via user interaction.
* **sys**:
  + **Purpose**: Provides access to system-specific parameters and functions.
  + **Usage**: In Expense Tracker it is used to redirect the standard output from terminal printing to file printing.
* **os**:
  + **Purpose**: Provides a way of interacting with the Windows operating system.
  + **Usage**: Used in Expense Tracker to clear the terminal screen.
* **time**:
  + **Purpose**: This module provides time-related functions.
  + **Usage**: It is used in Expense Tracker to create a sleep delay in a database connection retry loop.
* **datetime**:
  + **Purpose**: The datetime module supplies classes for manipulating dates and times.
  + **Usage**: In Expense Tracker it is used to correct the formatting of date values, both to Australian format and to a format acceptable to the Microsoft SQL database.

### External Modules used by Expense Tracker

* **pyodbc**:
  + **Purpose**: Used to allow Python to connect to a database via Open Database Connectivity (ODBC).
  + **Usage**: In the case of Expense Tracker, *pyodbc* has been used to connect to the Exp\_Tracker SQL database and then get and set data records.
* **art**:
  + **Purpose**: Used to print ASCII art text to the terminal.
  + **Usage**: In Expense Tracker, *art* is used to print an ASCII art logo for the start of the program.
* **getpass**:
  + **Purpose**: This module allows secure prompting of the user for a password without echoing it to the terminal.
  + **Usage**: The Expense Tracker program uses this to allow secure entry of the user’s password.
* **re**:
  + **Purpose**: This is the regular expression module in Python. It allows for advanced pattern matching and manipulation of strings.
  + **Usage**: In the case of the Expense Tracker program, it is used to ensure Currency amounts always have two decimal places as well as validating input for a user-entered filename and a person’s name.
* **tabulate**:
  + **Purpose**: Used to pretty-print tabular data in various formats.
  + **Usage**: This module is used in Expense Tracker to build and display various formatted reports of Expense Transactions.

**How to Install External Modules:**

For external modules like pyodbc, getpass, tabulate, and re, these can be installed using pip (Python's package manager).

*pip install pyodbc getpass tabulate re*

## Pseudocode

### Initial

This is the initial Pseudocode written for Task 2, ICT703.

Initialise Global Variables

userID = 0

Loop until valid answer received -

    Ask the user if the wish to (L)OGIN, (C)REATE a new user, or (Q)UIT

    If they want to CREATE a new user

        Issue them with a new USERID

        Create the database information related to a new user (fName, lName, budget)

    Otherwise if they want to LOGIN then ask them for their USERID and PASSWORD

        Loop until valid uName and pWord received.

        if uName and passWord are valid

            Continue

        otherwise

            Loop to login screen

        Otherwise

            Quit

Loop until valid answer received -

    Provide user with a menu (T)RANSACTIONS, (C)ATEGORIES, (B)UDGET, (R)EPORTS, (Q)UIT

        If 'T' then open the transactions menu

            Loop until valid answer received -

            Provide user with submenu (A)DD or (S)EARCH transactions

                If 'A' then

 Get transaction information from user - fName, lName, Date, Time, catName, Description, Amount

                    Convert fName, lName to userID

                    Convert catName to catID

                    Add transaction to transactions table in database

                If 'S' then

                    Loop until valid answer received -

Provide user with a sub-submenu - Search by (C)ATEGORY or (D)ATE or (T)IME or (R)ETURN to previous menu

                        if 'C' then

                            Provide user with list of current catName

                            Get catName to search on from user

                            Check if catName is valid

                            Select all transactions with that catName from database

Present report of transactions found with transID, date, time, fName, lName, catName, description, amount

                        otherwise if 'D' then

                            Get date to search on from user

                            Check if date is valid

                            Select all transactions with that date from database

Present report of transactions found with transID, date, time, fName, lName, catName, description, amount

                        otherwise if 'T' then

                            Get time to search on from user

                            Check if time is valid

                            Select all transactions with that time from database

Present report of transactions found with transID, date, time, fName, lName, catName, description, amount

                        otherwise if 'R' then

                            break out of loop

Provide the user with a sub-sub-submenu - (U)PDATE or (D)ELETE transaction or (R)ETURN to previous menu

                            if 'U' then

                                get transID from user

                                search the database for that transID and present it to the user

                                if valid transID loop through each field in the record found

                                    present contents of each field

                                    ask user if they wish to update the field

                                    create a new list with old and updated field details

                                    update the transID in database with new list of elements

                            if 'D' then

                                get transID from user

                                search the database for that transID and present it to the user

                                if valid transID

                                    delete record with transID from the database

                            if 'R' then

                                break from loop

        otherwise if 'C' then

            get a list of current catNames from database

            present user with the list of catNames

            provide user with a sub-menu - (U)PDATE, (A)DD, (D)ELETE a Category, or (R)ETURN

                if 'U' then

                    ask user for catID of catName they wish to update

                    check if catID is valid

                    ask user for new catName and update the database with it

                if 'A' then

                    get the last catID in the database and increment by 1

                    ask the user for the new catName

                    insert the new catID and catName into the database

                if 'D' then

                    ask user for catID they wish to delete

                    check database for any transactions with that catID

                        if none,

                            delete catID, catName from the database

                        otherwise

                            return error to user

                if 'R' then

                    break out of loop

        otherwise if 'B' then

            present the user with their current budget limit from database

            ask user if they wish to (U)PDATE budget or (R)ETURN

                if 'U' then

                    ask user for new budget limit

                    update database with new budget limit

                if 'R' then

                    break out of loop

        otherwise if 'R' then

            present the user with a list of available reports

            ask user which report they would like to run

            print requested report for user

        otherwise if 'Q' then

            break out of loop

            end program

### Final

This is the final version of the pseudocode for the program Exp\_Tracker.

*Initialise Global Variables*

userID = “”

*Body of Pseudocode*

Clear the screen

Display Expense Tracker Logo

Loop until valid answer received -

Ask the user if the wish to (L)OGIN, (C)REATE a new user, or (Q)UIT

If they want to CREATE a new user

Issue them with a new USERID

Ask them for their First Name, Last Name and Budget

Create the database information related to a new user (First Name, Last Name, budget)

Otherwise if they want to LOGIN then ask them for their USERID and PASSWORD

Loop until valid User Name and Password received

if User Name and Password are valid

Continue to Main Program

otherwise

Loop to login screen

Otherwise if they want to QUIT

Quit the program

*Main Program*

Clear the screen

Loop until valid answer received -

Provide user with a menu (T)RANSACTIONS, (C)ATEGORIES, (B)UDGET, (R)EPORTS, (Q)UIT

If 'T' then open the transactions menu

Clear the screen

Loop until valid answer received -

Provide user with submenu (A)DD or (S)EARCH transactions

If 'A' then with the current UserID

Present the user with a list of valid Categories or ask to create a new Category if none exist.

Get transaction information from user - Date, Time, Category ID, Description, Amount

Add transaction to transactions table in database

Add the userID and tranID to the usertransactions table in the database

If 'S' then

Clear the Screen

Loop until valid answer received

Provide user with a sub-submenu - Search by (C)ATEGORY or (D)ATE or (T)IME or (R)ETURN to previous menu

if 'C' then with the current userID

Provide user with list of current categories

Get category ID to search for from user

Check if category name is valid

Select all transactions with that category ID from database

Present report of transactions found with transID, date, time, catName, description, amount

Ask user if they wish to amend one of the transactions

If Yes - Goto Update/Delete Menu

If No – Return to Previous Menu

otherwise if 'D' then with current userID

Get date to search on from user

Check if date is valid

Select all transactions with that date from database for current user

Present report of transactions found with transID, date, time, catName, description, amount

Ask user if they wish to amend one of the transactions

If Yes - Goto Update/Delete Menu

If No – Return to Previous Menu

otherwise if 'T' then with the current UserID

Get time to search on from user

Check if time is valid

Select all transactions with that time for the current user from database

Present report of transactions found with transID, date, time, category name, description, amount

Ask user if they wish to amend one of the transactions

If Yes - Goto Update/Delete Menu

If No – Return to Previous Menu

otherwise if 'R' then

Return to previous Menu

UPDATE/DELETE MENU

Provide the user with a sub-sub-submenu - (U)PDATE or (D)ELETE transaction or (R)ETURN to previous menu

if 'U' then

Get transID from user

Search the database for that transID and present transID, date, time, category name, description, amount to the user

Provide the user with a list of available fields

Ask user if they wish to update a field

Get the field they wish to update and the update details

Update the transID in database with new details.

Return to previous menu

if 'D' then

Get transID from user

search the database for that transID and present transID, date, time, category name, description, amount to the user

Warn user that DELETE cannot be undone

Delete record with that transID from transactions table in the database.

Delete record with tranID and current userID from userTransactions table in the database.

Return to previous menu

if 'R' then

Return to previous menu

Otherwise if 'C' then

Clear the screen

Get a list of current categories from database

Present user with the list of categories

Provide user with a sub-menu - (U)PDATE, (A)DD, (D)ELETE a Category, or (R)ETURN

If 'U' then

Ask user for category ID of category they wish to update

Check if category ID is valid

Ask user for new category name and update the database with it

If 'A' then

Issue a new category ID

Ask the user for the new catName

Insert the new category ID and category name into the database

If 'D' then

Ask user for category ID they wish to delete

Check database for any transactions with that category ID

If none,

Delete category ID, category name from the database

Otherwise

Advise user that program Administrator must check with all users regarding their transactions for that category and delete manually.

If 'R' then

Return to previous menu

Otherwise if 'B' then

Clear the screen

Present the user with their current budget limit from database

Ask user if they wish to (U)PDATE budget or (R)ETURN

If 'U' then

Ask user for new budget limit

Update database with new budget limit

If 'R' then

Return to previous menu

Otherwise if 'R' then

Clear the screen

Present the user with a list of available reports

(1) Report on your current transactions

        (2) Report on your transactions by category

        (3) Report on your transactions between two dates

        (4) Report on your transactions between two times of day on a date

        (R)ETURN to previous menu

Ask user which report they would like to run

If number 1-4

Print requested report for user

Offer the user ability to save to external file

If (Y)es then save report to external file

If (N)o return to Reports Menu

Otherwise if ‘R’ then

Return to previous menu

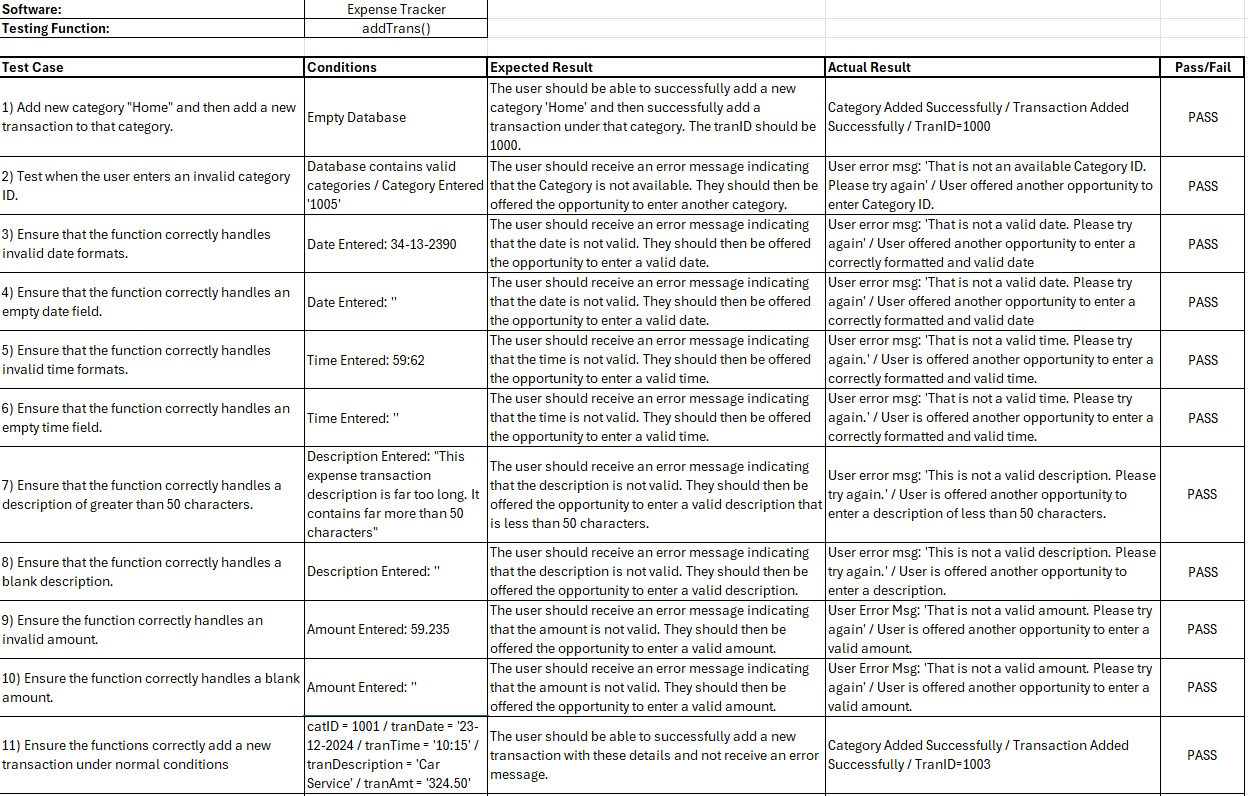
Otherwise if 'Q' then

Break out of loop

End program

## Test Cases

### Function addTrans()

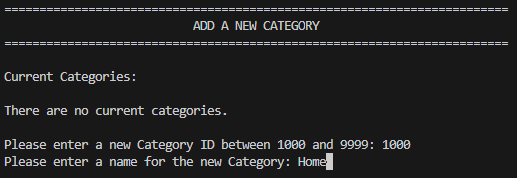
****

#### Functional Testing of addTrans()

1. Add new category "Home" and then add a new transaction to that category.

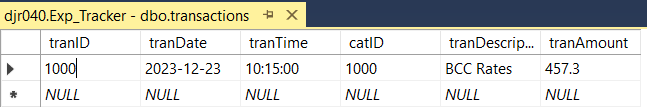
Steps:

* + Delete all records from the Exp\_Tracker database.
  + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (A)DD.
  + “There are no current categories.”.



* + Add new Category ID – 1000.
  + Add new Category Name – Home.
  + Category is now available – Enter the Category ID – 1000.
  + Enter details for new Transaction.

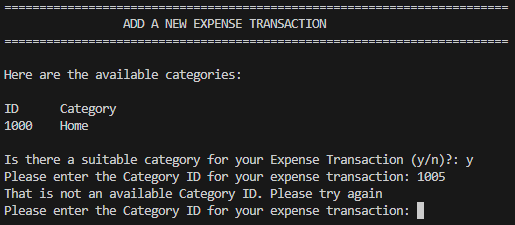




1. Test when the user enters an invalid category ID.

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (A)DD.
  + Enter an invalid Category ID (catID = 1005).



1. Ensure that the function correctly handles invalid date formats.

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (A)DD.
  + Enter a valid Category ID (catID=1000).
  + Enter an invalid Date (tranDate = 34-13-2390).



1. Ensure that the function correctly handles an empty date field.

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (A)DD.
  + Enter a valid Category ID (catID=1000).
  + Enter a blank Date (tranDate = ‘’).



1. Ensure that the function correctly handles invalid time formats.

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (A)DD.
  + Enter a valid Category ID (catID=1000).
  + Enter a valid Date (tranDate = 23-12-2023).
  + Enter an invalid Time (tranTime = 59:62).



1. Ensure that the function correctly handles an empty time field.

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (A)DD.
  + Enter a valid Category ID (catID=1000).
  + Enter a valid Date (tranDate = 23-12-2023).
  + Enter an invalid Time (tranTime = ‘’).



1. Ensure that the function correctly handles a description of greater than 50 characters.

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (A)DD.
  + Enter a valid Category ID (catID=1000).
  + Enter a valid Date (tranDate = 23-12-2023)..
  + Enter a valid Time (tranTime = 10:15).
  + Enter a Description of invalid length (tranDescription = “This expense transaction description is far too long. It contains far more than 50 characters”).



1. Ensure that the function correctly handles a blank description.

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (A)DD.
  + Enter a valid Category ID (catID=1000).
  + Enter a valid Date (tranDate = 23-12-2023).
  + Enter a valid Time (tranTime = 10:15).
  + Enter a blank Description (tranDescription = ‘’).



1. Ensure the function correctly handles an invalid amount.

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (A)DD.
  + Enter a valid Category ID (catID=1000).
  + Enter a valid Date (tranDate = 23-12-2023).
  + Enter a valid Time (tranTime = 10:15).
  + Enter a valid Description (tranDescription = ‘Insurance’).
  + Enter an invalid Amount (tranAmt = 59.235).



1. Ensure the function correctly handles a blank amount.

Steps:

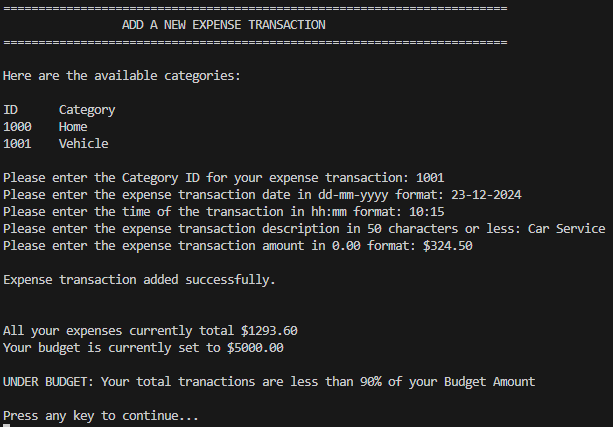
* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (A)DD.
  + Enter a valid Category ID (catID=1000).
  + Enter a valid Date (tranDate = 23-12-2023).
  + Enter a valid Time (tranTime = 10:15).
  + Enter a valid Description (tranDescription = ‘Insurance’).
  + Enter a blank Amount (tranAmt = ‘’).



1. Ensure the functions correctly add a new transaction under normal conditions.

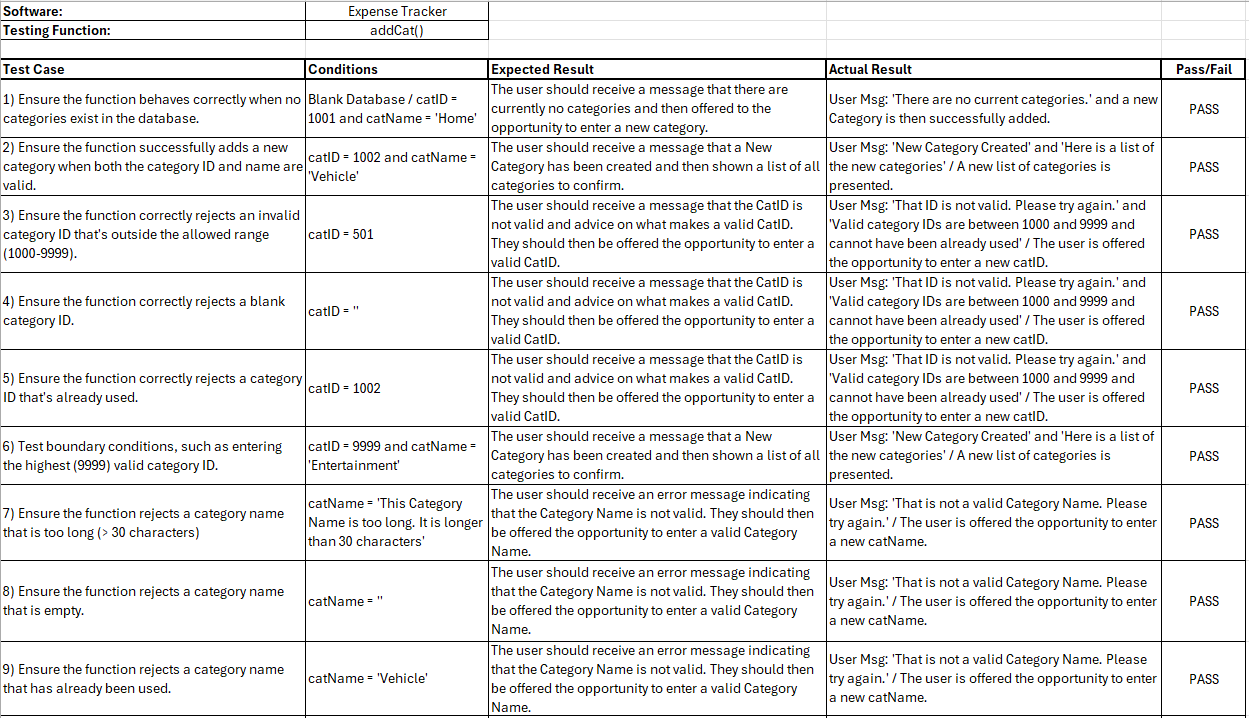
Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (A)DD.
  + Enter a valid Category ID (catID=1001).
  + Enter a valid Date (tranDate = 23-12-2023).
  + Enter a valid Time (tranTime = 10:15).
  + Enter a valid Description (tranDescription = ‘Car Service’).
  + Enter a valid Amount (tranAmt = 324.50).





### Function addCat()

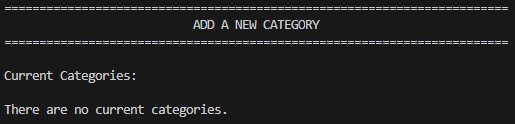


#### Functional Testing of addCat()

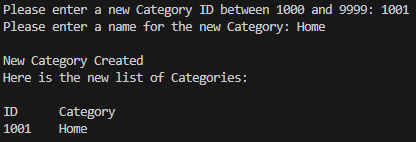
1. Ensure the function behaves correctly when no categories exist in the database.

Steps:

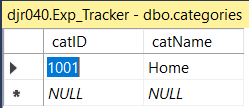
* + Delete all records from the Exp\_Tracker database.
  + Log into Expense Tracker Software.
  + Choose (C)ATEGORIES then (A)DD.



* + Add a new Category ID (catID = 1001).
  + Add a new Category Name (catName = ‘Home’).



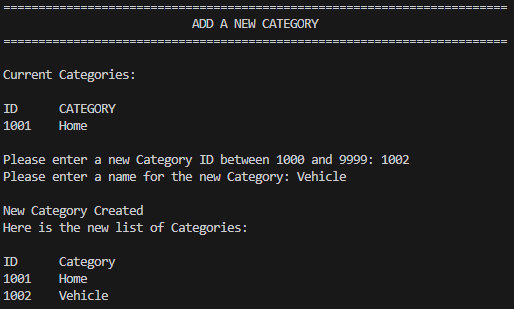
* + New Category is now available and in the database.

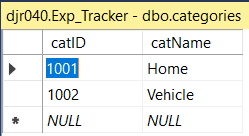


1. Ensure the function successfully adds a new category when both the Category ID and Name are valid.

Steps:

* + Delete all records from the Exp\_Tracker database.
  + Log into Expense Tracker Software.
  + Choose (C)ATEGORIES then (A)DD.
  + Add a new Category ID (catID = 1002).
  + Add a new Category Name (catName = ‘Vehicle’).

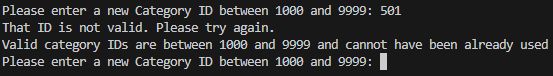




1. Ensure the function correctly rejects an invalid Category ID that is outside the allowed range (1000-9999).

Steps:

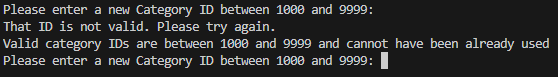
* + Log into Expense Tracker Software.
  + Choose (C)ATEGORIES then (A)DD.
  + Add an invalid Category ID (catID = 501).



1. Ensure the function correctly rejects a blank Category ID.

Steps:

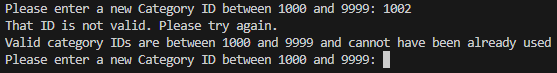
* + Log into Expense Tracker Software.
  + Choose (C)ATEGORIES then (A)DD.
  + Add a blank Category ID (catID = ‘’).



1. Ensure the function correctly rejects a Category ID that is already used.

Steps:

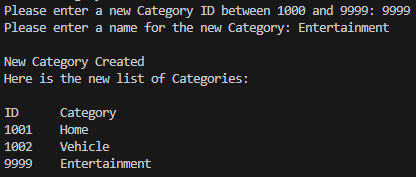
* + Log into Expense Tracker Software.
  + Choose (C)ATEGORIES then (A)DD.
  + Add a Category ID that is already in use (catID = 1002).



1. Test boundary conditions, such as entering the highest (9999) valid Category ID.

Steps:

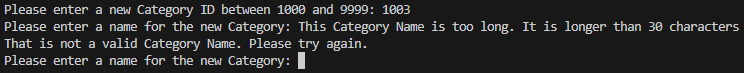
* + Log into Expense Tracker Software.
  + Choose (C)ATEGORIES then (A)DD.
  + Add the highest possible Category ID (catID = 9999).



1. Ensure the function rejects a Category Name that is too long (> 30 characters)

Steps:

* + Log into Expense Tracker Software.
  + Choose (C)ATEGORIES then (A)DD.
  + Add a Category ID (catID = 1003).
  + Add an invalid Category Name (catName = “This Category Name is too long. It is longer than 30 characters”).



1. Ensure the function rejects a Category Name that is empty.

Steps:

* + Log into Expense Tracker Software.
  + Choose (C)ATEGORIES then (A)DD.
  + Add a Category ID (catID = 1003).
  + Add a blank Category Name (catName = ‘’).



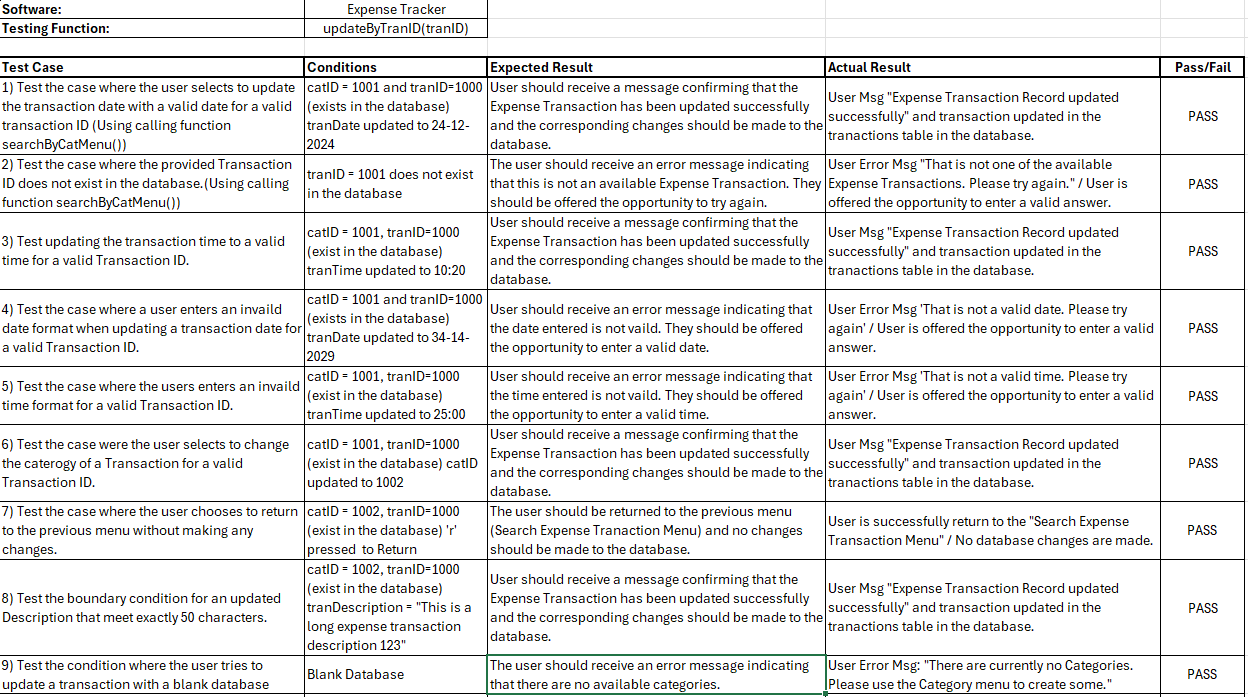
1. Ensure the function rejects a Category Name that has already been used.

Steps:

* + Log into Expense Tracker Software.
  + Choose (C)ATEGORIES then (A)DD.
  + Add a Category ID (catID = 1003).
  + Add a Category Name that is already in use (catName = ‘Vehicle’).



### Function updateByTranID(tranID)

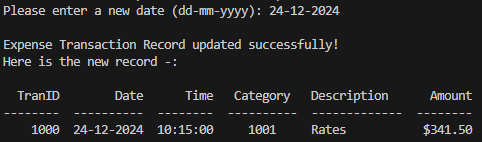


#### Functional testing of updateByTranID(tranID)

1. Test the case where the user selects to update the Transaction Date with a valid Date for a valid Transaction ID (Using calling function searchByCatMenu()).

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (S)EARCH then (C)ATEGORY.
  + Select a Category from the list to search under (catID = 1001).
  + Select a TranID from the list of available transactions (tranID = 1000).
  + Choose (U)PDATE to update the Transaction.
  + Choose (1) to update the Date.
  + Enter the valid Date to update (tranDate = 24-12-2024).





1. Test the case where the provided Transaction ID does not exist in the database (Using calling function searchByCatMenu()).

Steps:

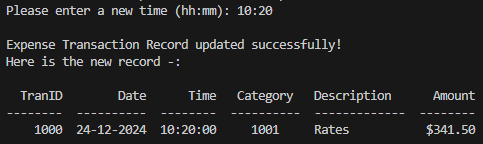
* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (S)EARCH then (C)ATEGORY.
  + Select a Category from the list to search under.
  + Select an invalid Transaction ID.



1. Test updating the Transaction Time to a valid Time for a valid Transaction ID.

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (S)EARCH then (C)ATEGORY.
  + Select a Category from the list to search under (catID = 1001).
  + Select a TranID from the list of available transactions (tranID = 1000).
  + Choose (U)PDATE to update the Transaction.
  + Choose (2) to update the Time.
  + Enter the valid Time to update (tranTime = ’10:20’).

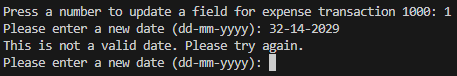




1. Test the case where a user enters an invalid date format when updating a Transaction Date for a valid Transaction ID.

Steps:

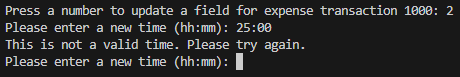
* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (S)EARCH then (C)ATEGORY.
  + Select a Category from the list to search under (catID = 1001).
  + Select a TranID from the list of available transactions (tranID = 1000).
  + Choose (U)PDATE to update the Transaction.
  + Choose (1) to update the Date.
  + Enter an invalid Date (tranDate = 32-14-2029).



1. Test the case where the user enters an invalid time format for a valid Transaction ID.

Steps:

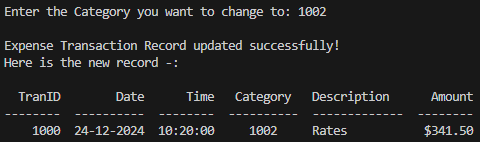
* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (S)EARCH then (C)ATEGORY.
  + Select a Category from the list to search under (catID = 1001).
  + Select a TranID from the list of available transactions (tranID = 1000).
  + Choose (U)PDATE to update the Transaction.
  + Choose (2) to update the Time.
  + Enter an invalid Time (tranTime = 25:00).



1. Test the case were the user selects to change the Category of a Transaction for a valid Transaction ID.

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (S)EARCH then (C)ATEGORY.
  + Select a Category from the list to search under (catID = 1001).
  + Select a TranID from the list of available transactions (tranID = 1000).
  + Choose (U)PDATE to update the Transaction.
  + Choose (3) to update the Category.
  + Enter a valid Category ID (catID = 1002).

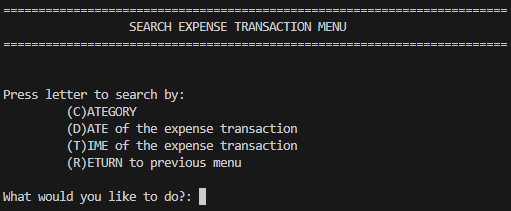




1. Test the case where the user chooses to return to the previous menu without making any changes.

Steps:

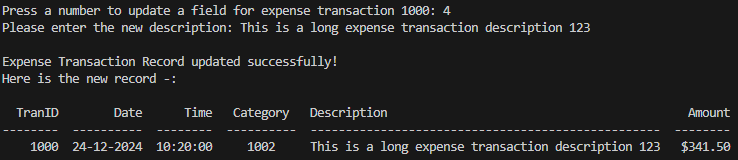
* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (S)EARCH then (C)ATEGORY.
  + Select a Category from the list to search under (catID = 1002).
  + Select a TranID from the list of available transactions (tranID = 1000).
  + Choose (U)PDATE to update the Transaction.
  + Choose (R) to return to the previous menu.



1. Test the boundary condition for an updated Description that meets exactly 50 characters.

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (S)EARCH then (C)ATEGORY.
  + Select a Category from the list to search under (catID = 1002).
  + Select a TranID from the list of available transactions (tranID = 1000).
  + Choose (U)PDATE to update the Transaction.
  + Choose (4) to update the Description.
  + Enter a valid (but long) Description (tranDescription = ‘This is a long expense transaction description 123’.

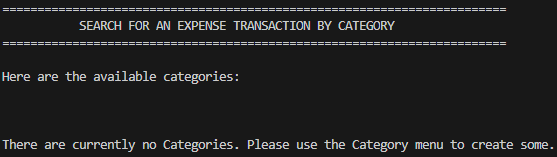




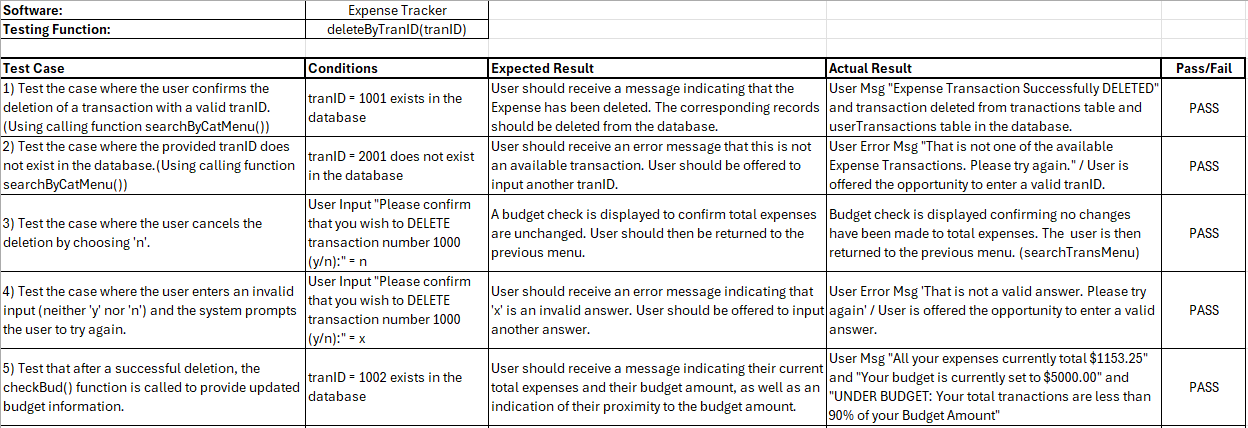
1. Test the condition where the user tries to update a Transaction with a blank database

Steps:

* + Delete all transactions from database.
  + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (S)EARCH then (C)ATEGORY



### Function deleteByTranID(tranID)

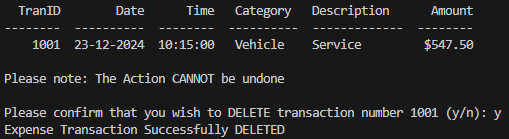


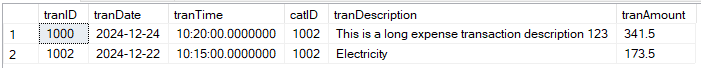
#### Functional Testing of deleteByTranID(tranID)

1. Test the case where the user confirms the deletion of a Transaction with a valid tranID. (Using calling function searchByCatMenu()).

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (S)EARCH then (C)ATEGORY.
  + Select a Category from the list to search under (catID = 1001).
  + Select a TranID from the list of available transactions (tranID = 1001).
  + Choose (D)ELETE to delete the Transaction.
  + Choose ‘y’ to confirm the deletion.

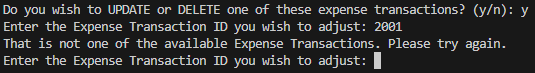




1. Test the case where the provided tranID does not exist in the database (Using calling function searchByCatMenu()).

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (S)EARCH then (C)ATEGORY.
  + Select a Category from the list to search under (catID = 1002).
  + Enter an invalid Transaction ID (tranID = 2001).



1. Test the case where the user cancels the deletion by choosing 'n'.

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (S)EARCH then (C)ATEGORY.
  + Select a Category from the list to search under (catID = 1002).
  + Select a TranID from the list of available transactions (tranID = 1000).
  + Choose (D)ELETE to delete the Transaction.
  + Choose selection ‘n’.

A screen shot of a computer

Description automatically generated

1. Test the case where the user enters an invalid input (neither 'y' nor 'n') and the system prompts the user to try again.

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (S)EARCH then (C)ATEGORY.
  + Select a Category from the list to search under (catID = 1002).
  + Select a TranID from the list of available transactions (tranID = 1000).
  + Choose (D)ELETE to delete the Transaction.
  + Choose invalid selection ‘x’.

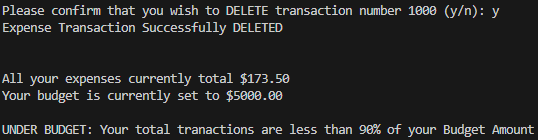
A black background with white text

Description automatically generated

1. Test that after a successful deletion, the checkBud() function is called to provide updated budget information.

Steps:

* + Log into Expense Tracker Software.
  + Choose (T)RANSACTIONS then (S)EARCH then (C)ATEGORY.
  + Select a Category from the list to search under (catID = 1002).
  + Select a TranID from the list of available transactions (tranID = 1000).
  + Choose (D)ELETE to delete the Transaction.
  + Choose ‘y’ to confirm the deletion.
  + Budget information is presented to the user.



## Conclusion

In conclusion, the Expense Tracker software offers a solution for individuals seeking to manage their personal finances. Through its intuitive interface, users can track their expenses, categorise transactions, and gain insight into their financial standing.

The detailed pseudocode and test plan presented in this report provide a clear understanding of the program's architecture and validation processes, ensuring its reliability.

While there are some limitations to its functionality, such as a lack of revenue tracking and category deletion bounded by existing transactions, overall, the Expense Tracker program is well-equipped to support users in making informed decisions about their spending.

## References

I acknowledge the use of ChatGPT in this assessment to complete the following –

* Provide a method to programmatically clear the terminal on a Windows PC.
* Provide the Regular Expression pattern for the function *isValidName.*
* Provide a method to check if a date is in a valid format in the function *isValidDate.*
* Provide a method to check if a time is in a valid format in the function *isValidTime.*
* Provide the Regular Expression pattern for the function *hasTwoDecimalPlaces.*
* Provide a method for connecting to a SQL Database, extracting data based on a given SQL query and then closing the connection, including error checking.
* Provide a method for connecting to a SQL Database, setting data based on a given SQL query and then closing the connection, including error checking.
* Provide a method to pause the program until a key is pressed for the function *pause.*
* Provide a method to check if a given filename is valid on the Windows PC for the function *isValidFilename.*
* Provide a method to convert date from one format to another, e.g. YYYY-MM-DD to DD-MM-YYYY, for the functions *fixDate* and *convertDate.*
* Provide a method to convert an amount to a currency with only two decimal places for the function *fixAmt.*
* Provide information on using the tabulate module for various functions.
* Provide a method for extracting a budget amount from a list of tuples for function *getBud*.
* Provide a method to check if a Directory in a Windows file system currently exists (and create it if it does not) for function *saveToFile.*
* Provide a method for redirecting the Standard Out to a file instead of the terminal for function *saveToFile.*
* Provide an example of a text-based menu-driven program to assist with creating the menu system.
* Create a T-SQL query to delete all database records but maintain the structure and referential integrity of the database.
* Provide a list of test cases for the functions *addTrans*, *addCat*, *updateByTranID* and *deleteByTranID*.

## Appendix

### Supporting Queries

* Delete all transactions from the database but maintain structure and relationships – for testing purposes.
* T-SQL Query Used on Exp\_Tracker SQL database as follows,

-- Step 1: Disable all foreign key constraints in the database

DECLARE @sql NVARCHAR(MAX) = '';

-- Disable foreign key constraints

SELECT @sql = @sql + 'ALTER TABLE [' + s.name + '].[' + t.name + '] NOCHECK CONSTRAINT ALL; '

FROM sys.tables t

INNER JOIN sys.schemas s ON t.schema\_id = s.schema\_id

WHERE t.is\_ms\_shipped = 0; -- Exclude system tables

EXEC sp\_executesql @sql;

-- Step 2: Delete all records from each table in the database

SET @sql = '';

-- Generate DELETE statements for each table

SELECT @sql = @sql + 'DELETE FROM [' + s.name + '].[' + t.name + ']; '

FROM sys.tables t

INNER JOIN sys.schemas s ON t.schema\_id = s.schema\_id

WHERE t.is\_ms\_shipped = 0; -- Exclude system tables

EXEC sp\_executesql @sql;

-- Step 3: Re-enable all foreign key constraints in the database

SET @sql = '';

-- Re-enable foreign key constraints

SELECT @sql = @sql + 'ALTER TABLE [' + s.name + '].[' + t.name + '] WITH CHECK CHECK CONSTRAINT ALL; '

FROM sys.tables t

INNER JOIN sys.schemas s ON t.schema\_id = s.schema\_id

WHERE t.is\_ms\_shipped = 0; -- Exclude system tables

EXEC sp\_executesql @sql;