



**NEW HORIZON  
COLLEGE OF ENGINEERING**

Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC  
Accredited by NAAC with 'A' Grade, Accredited by NBA

**A MINI PROJECT REPORT**

*for*

**Mini Project using Python (20CSE59)**

*on*

**Expense Analyzer**

*Submitted by*

**B.AJAY**

**USN: 1NH20CS039, Sem-Sec: 5-A**

*In partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING**

*in*

**COMPUTER SCIENCE AND ENGINEERING**

**Academic Year: 2022-23**



# NEW HORIZON COLLEGE OF ENGINEERING

Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC  
Accredited by NAAC with 'A' Grade, Accredited by NBA

## CERTIFICATE

This is to certify that the mini project work titled

### Expense Analyzer

submitted in partial fulfillment of the degree of Bachelor of Engineering in  
Computer Science and Engineering by

**B.AJAY**

**USN:1NH20CS039**

*DURING*

*ODD SEMESTER 2022-2023*

*for*

*Course: Mini Project using Python-20CSE59*

Signature of Reviewer

Signature of HOD

**SEMESTER END EXAMINATION**

*Name of the Examiner*

*Signature with date*

1. \_\_\_\_\_

\_\_\_\_\_

2. \_\_\_\_\_

\_\_\_\_\_

## **ABSTRACT**

Money is the most valuable portion of our daily life and without money we will not last one day on the earth. Having an efficient and useful expense tracking system is important when it comes to managing one's money income and expenses.

This project is based on an expense tracking system. This project aims to create an easy, faster and smooth tracking system for the expenses. This project also offers some opportunities that will help the user to sustain all financial activities like digital automated diary.

So, for the better expense tracking system, this project has been developed that will help the users a lot. Most of the people cannot track their expenses and income which will lead them to facing a money crisis, in this case daily expense tracker can help the people to track expense day to day and making life tension free.

So using the daily expense tracker application is important to lead a happy family. Daily expense tracker helps the user to avoid unexpected expenses and bad financial situations. This Project will save time and provide a responsible lifestyle.

The entire program has been developed in Python and uses the Pycharm IDE for running the Python application. The mini-project is completely based on the high-level language. It uses SQLite database and tkinter to provide a simple and easy to understand platform for the users.

## ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be impossible without the mention of the people who made it possible, whose constant guidance and encouragement crowned my efforts with success.

I have great pleasure in expressing gratitude to **Dr. Mohan Manghnani**, Chairman, New Horizon Educational Institutions, for providing necessary infrastructure and creating good environment.

I take this opportunity to express my profound gratitude to **Dr. Manjunatha**, Principal, New Horizon College of Engineering, for his constant support and encouragement.

I would like to thank **Dr. Anandhi R J**, Professor and Dean-Academics, NHCE, for her valuable guidance.

I would also like to thank **Dr. B. Rajalakshmi**, Professor and HOD, Department of Computer Science and Engineering, for her constant support.

I also express my gratitude to Ms. **Sri Vidhya** , Senior Assistant Professor, Department of Computer Science and Engineering, my mini project reviewer, for constantly monitoring the development of the project and setting up precise deadlines. Her valuable suggestions were the motivating factors in completing the work.

**B.Ajay**

**USN: 1NH20CS039**

# CONTENTS

|  |           |
|--|-----------|
| <b>ABSTRACT</b>                            | <b>I</b>  |
| <b>ACKNOWLEDGEMENT</b>                     | <b>II</b> |
| <b>LIST OF FIGURES</b>                     | <b>V</b>  |
| <br>                                       |           |
| <b>1. INTRODUCTION</b>                     |           |
| 1.1. PROBLEM DEFINITION                    | 1         |
| 1.2. OBJECTIVES                            | 1         |
| 1.3. METHODOLOGY TO BE FOLLOWED            | 2         |
| 1.4. EXPECTED OUTCOMES                     | 2         |
| 1.5. HARDWARE AND SOFTWARE REQUIREMENTS    | 3         |
| <br>                                       |           |
| <b>2. FUNDAMENTALS OF JAVA PROGRAMMING</b> |           |
| 2.1. INTRODUCTION                          | 4         |
| 2.2. DATATYPES                             | 4         |
| 2.3. OPERATORS                             | 5         |
| 2.4. LOOPS                                 | 6         |
| 2.5. FUNCTIONS                             | 6         |
| 2.6. CLASSES AND OBJECTS                   | 7         |
| <br>                                       |           |
| <b>3. FUNDAMENTALS OF DBMS</b>             |           |
| 3.1. INTRODUCTION                          | 8         |
| 3.2. CHARACTERISTICS                       | 8         |
| 3.3. ER MODEL                              | 10        |
| 3.4. ENTITY SET AND KEYS                   | 11        |
| 3.5. SQL OVERVIEW                          | 12        |

|                             |           |
|-----------------------------|-----------|
| <b>4. DESIGN</b>            |           |
| 4.1. DESIGN GOALS           | <b>14</b> |
| 4.2. PSEUDOCODE             | <b>14</b> |
| 4.3. FLOWCHART              | <b>15</b> |
| <b>5. IMPLEMENTATION</b>    |           |
| 5.1. MODULE 1 FUNCTIONALITY | <b>16</b> |
| 5.2. MODULE 1 FUNCTIONALITY | <b>17</b> |
| 5.3. MODULE 1 FUNCTIONALITY | <b>17</b> |
| 5.4. MODULE 1 FUNCTIONALITY | <b>18</b> |
| 5.5. MODULE 1 FUNCTIONALITY | <b>19</b> |
| 5.6. MODULE 1 FUNCTIONALITY | <b>20</b> |
| 5.7. MODULE 1 FUNCTIONALITY | <b>21</b> |
| <b>6. RESULTS</b>           | <b>22</b> |
| <b>7. CONCLUSION</b>        | <b>26</b> |
| <b>REFERENCES</b>           | <b>27</b> |

## LIST OF FIGURES

| Figure No | Figure Description     | Page No |
|-----------|------------------------|---------|
| 2.1       | Data Types             | 4       |
| 2.2       | Class & Objects        | 7       |
| 3.1       | DBMS                   | 8       |
| 3.2       | ER Model               | 11      |
| 4.1       | Flowchart              | 15      |
| 5.1       | Login Module           | 16      |
| 5.2       | Main Menu Module       | 17      |
| 5.3       | Add Record Module      | 17      |
| 5.4       | View Records Module    | 18      |
| 5.5       | Delete Record Module   | 19      |
| 5.6       | Update Record Module   | 20      |
| 5.7       | Spending Report Module | 21      |
| 6.1       | Login Page             | 22      |
| 6.2       | Main Menu              | 22      |
| 6.3       | Add Record             | 23      |
| 6.4       | View Records           | 23      |
| 6.5       | Update Record          | 24      |
| 6.6       | Delete Record          | 24      |
| 6.7       | Spending Report        | 25      |

## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1 PROBLEM DEFINITION**

Managing finances is always a challenging task and for this Expense Analyzer plays a key role. At times , Individuals make financial mistakes as they spend carelessly. Moreover, Manually tracking down all the expenses and investments made along with managing the income and budgeting can be a very time consuming and tedious process. This project is aimed to make this process as easy as possible for the user to maintain his/her finances. This project will also help the user to learn about his/her spending patterns and plan their expenses in order to save money and be more responsible financially.

#### **1.2 OBJECTIVES**

- To make expense tracking an easy task.
- To allow the user to store the details of their previous transactions.
- To give the user detailed report about their spending.
- To allow the user to update and delete their previous transaction.



### **1.3 METHODOLOGY TO BE FOLLOWED**

Login module – For the users to enter their username and password to access the application.

Enter details module – For receiving the input of all the details regarding a previous transaction made by their user. The details will be stored in a database.

Update record module – For allowing the user to update certain details of a previously stored transaction in the database.

Delete record module – For allowing the user to delete the details of a previously stored transaction from the database.

Spending report module – To provide the user with a detailed report about his/her spending.

### **1.4 EXPECTED OUTCOMES**

- To get details of a transaction and store them in a database.
- To update and delete the details of a previously stored transaction.
- To provide a detailed report about the user's spending.
- To make the user aware about their spending so that they can spend their income more responsibly.
- To help the users to save money.

## **1.5 HARDWARE AND SOFTWARE REQUIREMENTS**

Hardware Requirements :

- 1) RAM – 3GB OR MORE
- 2 ) Processor – INTEL CORE i3

Software Requirements :

- 1) Pycharm IDE
- 2) Operating system – Windows 11

## CHAPTER 2

# FUNDAMENTALS OF PYTHON PROGRAMMING

### 2.1 Introduction

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently. There are two major Python versions: Python 2 and Python 3. Both are quite different.

### 2.2 Data Types

In programming, data type is an important concept. Variables can store data of different types, and different types can do different things.

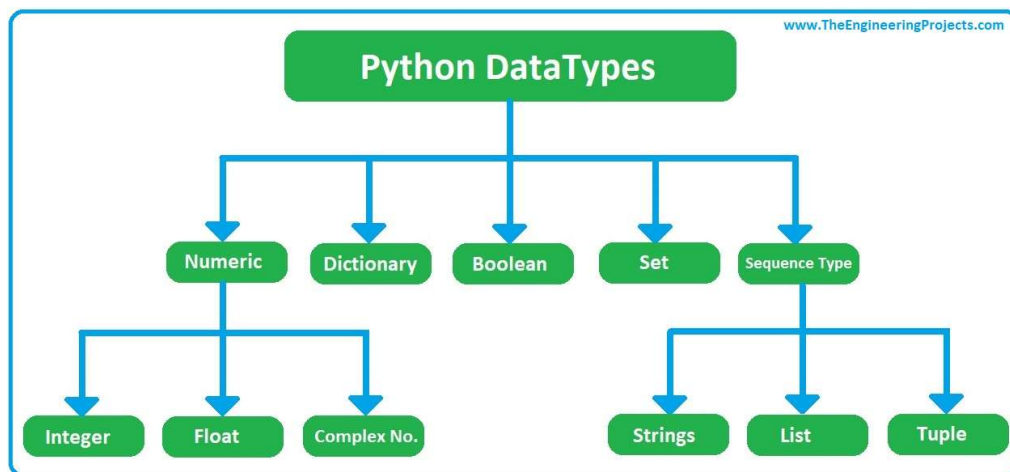


Fig 2.1 Data Types

Python has the following data types built-in by default, in these categories:

Text Type:           str

Numeric Types:    int, float, complex

Sequence Types:  list, tuple, range

Mapping Type:     dict

Set Types:         set, frozenset

Boolean Type:     bool

Binary Types:     bytes, bytearray, memoryview

## 2.3 Operators

Operators are used to perform operations on variables and values.

Python divides the operators in the following groups:

- Arithmetic operators : `+` , `-` , `*` , `/` , `%` , `**` , `//`
- Assignment operators : `=` , `+=` , `-=` , `*=` , `/=`
- Comparison operators : `==` , `!=` , `>` , `<` , `>=` , `<=`
- Logical operators : **and** , **or** , **not**
- Identity operators : **is** , **is not**
- Membership operators : **in** , **not in**
- Bitwise operators : **&** , **|** , **<<** , **>>**

## 2.4 Loops

In computer programming, a loop is a sequence of instructions that is continually repeated until a certain condition is reached.

Python has two primitive loop commands:

- while loops
- for loops

### While loops :

With the while loop we can execute a set of statements as long as a condition is true.

### For loops :

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string) . This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-oriented programming languages . With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

## 2.5 Functions

A function is a block of code which only runs when it is called . You can pass data, known as parameters, into a function . A function can return data as a result . In Python a function is defined using the def keyword . To call a function, use the function name followed by parenthesis.

### Function arguments :

Information can be passed into functions as arguments . Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just

separate them with a comma . The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

## 2.6 Classes and Objects

Python is an object oriented programming language . Almost everything in Python is an object, with its properties and methods . A Class is like an object constructor, or a "blueprint" for creating objects. To create a class, use the keyword class . Now we can use the created class to create objects

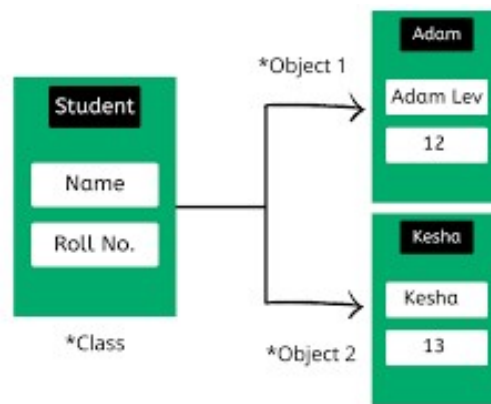


Fig 2.2 Class & Objects

### Init function :

All classes have a function called `__init__()`, which is always executed when the class is being initiated . Use the `__init__()` function to assign values to object properties, or other operations that are necessary to do when the object is being created . The `__init__()` function is called automatically every time the class is being used to create a new object.

## CHAPTER 3

# FUNDAMENTALS OF DBMS

### 3.1 Introduction

Database Management System or DBMS in short refers to the technology of storing and retrieving user's data with utmost efficiency along with appropriate security measures. Database is a collection of related data and data is a collection of facts and figures that can be processed to produce information. Mostly data represents recordable facts. Data aids in producing information, which is based on facts. For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks. A database management system stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.

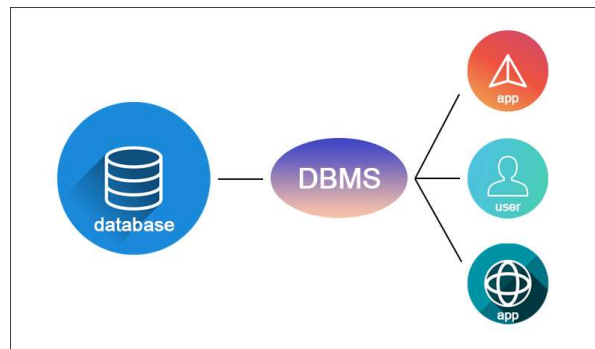


Fig 3.1 DBMS

### 3.2 Characteristics

Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management. A modern DBMS has the following characteristics –

- **Real-world entity** – A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.

- **Relation-based tables** – DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.
- **Less redundancy** – DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.
- **Query Language** – DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.
- **ACID Properties** – DBMS follows the concepts of Atomicity, Consistency, Isolation, and Durability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.
- **Multiple views** – DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.
- **Security** – Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.



### 3.3 ER Model

#### Entity

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity. An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

#### Attributes

Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

Types of Attributes :

- **Simple attribute** – Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
- **Composite attribute** – Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first\_name and last\_name.
- **Derived attribute** – Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average\_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data\_of\_birth.
- **Single-value attribute** – Single-value attributes contain single value. For example – Social\_Security\_Number.

- **Multi-value attribute** – Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email\_address, etc.

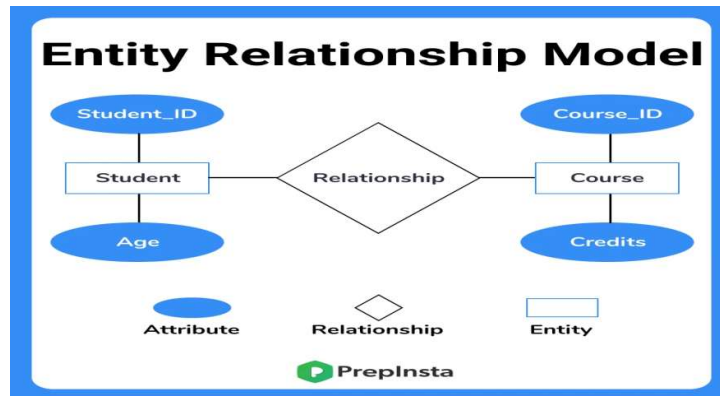


FIG 3.2 ER Model

### Relationship

The association among entities is called a relationship. For example, an employee works\_at a department, a student enrolls in a course. Here, Works\_at and Enrolls are called relationships. The number of participating entities in a relationship defines the degree of the relationship.

- Binary = degree 2
- Ternary = degree 3
- n-ary = degree

### 3.4 Entity Set and Keys

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

- **Super Key** – A set of attributes (one or more) that collectively identifies an entity in an entity set.
- **Candidate Key** – A minimal super key is called a candidate key. An entity set may have more than one candidate key.
- **Primary Key** – A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

### 3.5 SQL Overview

SQL is a programming language for Relational Databases. It is designed over relational algebra and tuple relational calculus. SQL comes as a package with all major distributions of RDBMS.

SQL comprises both data definition and data manipulation languages. Using the data definition properties of SQL, one can design and modify database schema, whereas data manipulation properties allows SQL to store and retrieve data from database.

The instructions that are used to communicate with the database using standard SQL fall into different sub-languages.

- Data Definition Language (DDL)
- Data Manipulation language (DML)
- Data Query Language (DQL)
- Data Control Language (DCL)
- Transaction Control Language (TCL)

#### **DDL Statements :**

DDL statements allow the user to define and manage objects within the database. Tasks might include:

- Create, alter, and drop schemas (users)
- Create, alter, and drop schema objects (table, view, sequence, index, synonym)
- Analyze information on a table, index, or cluster
- Establish auditing options
- Grant and revoke privileges and roles (Considered as DCL statements by some vendors.)

Some examples of key words that start DDL statements include : create , drop , alter , truncate

**DML Statements :**

Manipulate data in a database is done using Data Manipulation Language (DML) statements. Some examples of key words that start DML statements include : insert , delete , update .

**TCL Statements :**

In Oracle DML statements do not automatically save changes. Oracle provides statements that manage changes made by DML statements. The transaction control statements starting keywords are : commit , rollback , savepoint

**DQL Statements :**

DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it. We can define DQL as follows it is a component of SQL statement that allows getting data from the database and imposing order upon it. The transaction control statements starting keywords are : commit , rollback , savepoint

## **CHAPTER 4**

### **DESIGN**

#### **4.1 DESIGN GOALS**

Our design mechanism includes two main things:

1. Reduced complexity in keeping track of previous transactions.
2. A user friendly interface to store and display previous expenses.
3. To provide detailed reports about the user's spending.
4. To make the users aware about their spending pattern and help them in becoming more responsible financially .

#### **4.2 ALGORITHM /PSEUDOCODE**

Step 1 : Start

Step 2 : Ask user which feature of the program they want to use .

Step 3 : Get the necessary transaction details from the user and store them in a database.

Step 4 : Display the previously stored transaction details from the database in the form of a table.

Step 5 : Update the previously stored record as per the user's input.

Step 6 : Delete the particular record selected by the user from the database.

Step 7 : Provide a detailed report about the users spending with the help of the data stored in the database.

Step 8 : End.

### 4.3 FLOWCHART

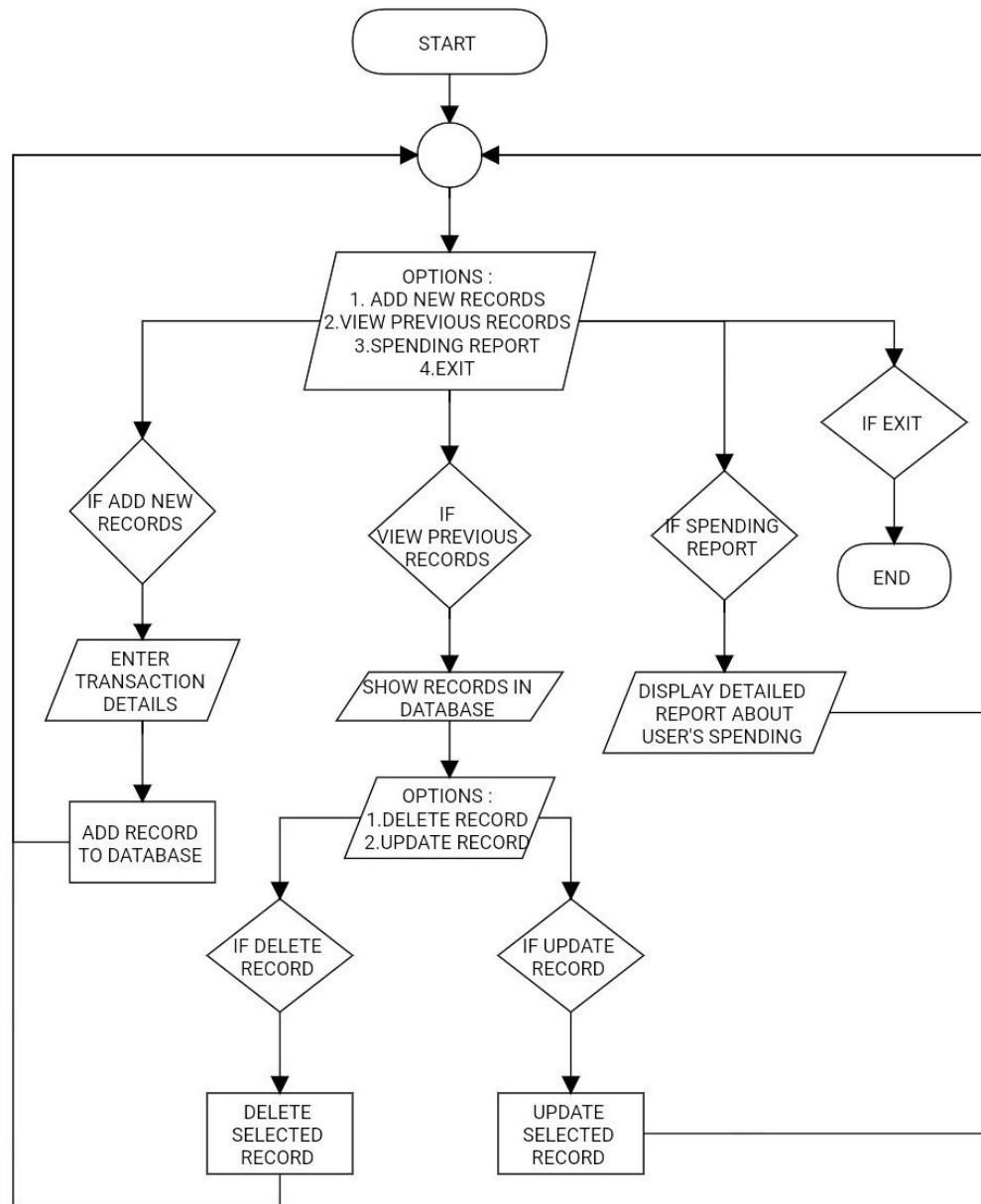


Fig 4.1 flowchart

## CHAPTER 5

### IMPLEMENTATION

#### 5.1 MODULE 1 FUNCTIONALITY

```
#login form
def login():
    username=entry1.get()
    password=entry2.get()
    if(username==" " and password==" "):
        messagebox.showinfo("", "Blank not allowed")
    elif(username=="ajay" and password=="admin"):
        root.iconify()
        openmain()
    else:
        messagebox.showinfo("", "Incorrect Username or Password")

global loginstatus
loginstatus=0
global entry1
global entry2
Label1=Label(root, text="Username", bg="#70D6AA").place(x=20, y=20)
Label2=Label(root, text="Password", bg="#70D6AA").place(x=20, y=90)
entry2 = Entry(root, show="*", width=20, bd=3)
entry2.place(x=110, y=90)
entry1=Entry(root, bd=3)
entry1.place(x=110, y=20)
#entry2=Entry(root, bd=3)
#entry2.place(x=110, y=90)
loginbutton=Button(root, text="Login", command=login, height=2, width=10, bd=3).place(x=80, y=140)

root.mainloop()
```

Fig 5.1 Login module

In this module , The login page is created in which the input is taken from the user for username and password and if it matches then the program allows further execution.

## 5.2 MODULE 2 FUNCTIONALITY

```
#main_menu
def openmain():
    top1 = Toplevel()
    top1.title("Main Menu")
    top1.geometry("800x500")
    top1.configure(bg="#70D6AA")
    addbutton=Button(top1,text="Add New Records",command=addrecord,height=3,width=20,bd=3)
    addbutton.place(x=300,y=100)
    viewbutton = Button(top1, text="View Previous Records", command=viewrecord, height=3, width=20, bd=3)
    viewbutton.place(x=300, y=200)
    reportbutton = Button(top1, text="Spending Report", command=report, height=3, width=20, bd=3)
    reportbutton.place(x=300, y=300)
```

Fig 5.2 Main Menu module

In this module the main menu is created in which there are 3 buttons to access 3 features of the program.

## 5.3 MODULE 3 FUNCTIONALITY

```
#add_record
def addrecord():
    top2 = Toplevel()
    top2.title("Add New Record")
    top2.geometry("800x500")
    top2.configure(bg="#70D6AA")
    Label16 = Label(top2, text="Product Number :", bg="#70D6AA").place(x=20, y=20)
    Label11 = Label(top2, text="Product/Service Name :", bg="#70D6AA").place(x=20, y=100)
    Label12 = Label(top2, text="Category Of Product :", bg="#70D6AA").place(x=20, y=180)
    Label13 = Label(top2, text="Amount Paid :", bg="#70D6AA").place(x=20, y=260)
    Label14 = Label(top2, text="Method Of Payment :", bg="#70D6AA").place(x=20, y=340)
    Label15 = Label(top2, text="Date Of Payment :", bg="#70D6AA").place(x=500, y=280)
    cal = Calendar(top2, selectmode='day', year=2022, month=12, day=20)
    cal.place(x=500, y=20)
    submitbutton=Button(top2,text="Add Record",command=lambda: submit(number_entry,name_entry,category_entry,amount_entry,method_entry,date_entry,top2),
    submitbutton.place(x=350,y=400)
    datebutton=Button(top2,text="Pick Date",command=lambda: grab_date(date_entry,cal),height=2,width=10,bd=3)
    datebutton.place(x=580, y=220)
    number_entry = Entry(top2, bd=3)
    number_entry.place(x=130, y=20)
    name_entry = Entry(top2, bd=3)
    name_entry.place(x=160, y=100)
    category_entry = Entry(top2, bd=3)
    category_entry.place(x=150, y=180)
    amount_entry = Entry(top2, bd=3)
    amount_entry.place(x=110, y=260)
    method_entry = Entry(top2, bd=3)
    method_entry.place(x=150, y=340)
    date_entry = Entry(top2, bd=3)
```

Fig 5.3 Add Record module

In this module , Necessary transaction details are taken from the user and stored into the database.



## 5.4 MODULE 4 FUNCTIONALITY

```
#view_record
def viewrecord():
    top3 = Toplevel()
    top3.title("View Records")
    top3.geometry("1200x500")
    top3.configure(bg="#70D6AA")
    delete_one_button=Button(top3,text="Remove Selected Record",command=lambda: delete_one(tree),height=4,width=20,bd=3)
    delete_one_button.place(x=350, y=400)
    update_one_button = Button(top3, text="Update Selected Record", command=lambda: update_one(tree,top3), height=4,width=20, bd=3)
    update_one_button.place(x=650, y=400)
    conn = sqlite3.connect("expense.db") # create/connect to database
    c = conn.cursor() # create cursor
    Style=ttk.Style()
    Style.theme_use("clam")
    Style.configure("Treeview",background="silver",foreground="black",rowheight=35,fieldbackground="#70D6AA")
    Style.map("Treeview",background=[("selected", "red")])
    tree = ttk.Treeview(top3, column=("Product Number", "Product Name", "Product Category", "Date Of Payment", "Amount Paid", "Method Of Payment"),
    tree.column("#1", anchor=tkinter.CENTER)
    tree.heading("#1", text="Product Number")
    tree.column("#2", anchor=tkinter.CENTER)
    tree.heading("#2", text="Product Name")
    tree.column("#3", anchor=tkinter.CENTER)
    tree.heading("#3", text="Product Category")
    tree.column("#4", anchor=tkinter.CENTER)
    tree.heading("#4", text="Date Of Payment")
    tree.column("#5", anchor=tkinter.CENTER)
    tree.heading("#5", text="Amount Paid")
    tree.column("#6", anchor=tkinter.CENTER)
    tree.heading("#6", text="Method Of Payment")
    c.execute("SELECT * FROM sample4")
    rows=c.fetchall()
    tree.tag_configure('oddrow', background="white")
    tree.tag_configure('evenrow', background="lightblue")
    global count
    count = 0
    for row in rows:
        if count % 2 == 0:
            tree.insert("", tkinter.END, values=row, tags=('evenrow',))
        else:
            tree.insert("", tkinter.END, values=row, tags=('oddrow',))
        count = count + 1
    tree.pack()
    conn.commit() #commit changes
    conn.close() # close connection to database
```

Fig 5.4 View Record module

In this module , Records from the database are selected and displayed on the screen.

Also 2 buttons are provided to allow the user to update and delete records.

## 5.5 MODULE 5 FUNCTIONALITY

```
#To delete a selected record from table
def delete_one(tree):
    x=tree.selection()[0]
    selected=tree.focus()
    temp=tree.item(selected, 'values')
    pnum=temp[0]
    tree.delete(x)
    conn = sqlite3.connect("expense.db") # create/connect to database
    c = conn.cursor() # create cursor
    c.execute("DELETE FROM sample4 WHERE oid="+pnum)
    messagebox.showinfo("", "Record Deleted")
    conn.commit() # commit changes
    conn.close() # close connection to database
```

Fig 5.5 Delete Record module

In this module , The record selected by the user is deleted from the database

## 5.6 MODULE 6 FUNCTIONALITY

```

#To update a selected record
def update_one(tree,top3):
    top3.iconify()
    top4 = Toplevel()
    top4.title("Update Record")
    top4.geometry("900x500")
    top4.configure(bg="#70D6AA")
    Label1 = Label(top4, text="Product/Service Name :", bg="#70D6AA").place(x=20, y=20)
    Label2 = Label(top4, text="Category Of Product :", bg="#70D6AA").place(x=20, y=80)
    Label3 = Label(top4, text="Amount Paid :", bg="#70D6AA").place(x=20, y=140)
    Label4 = Label(top4, text="Method Of Payment :", bg="#70D6AA").place(x=20, y=200)
    Label5 = Label(top4, text="Date Of Payment :", bg="#70D6AA").place(x=20, y=260)
    updatebutton = Button(top4, text="Update Record",command=lambda: update(), height=3, width=20, bd=3)
    updatebutton.place(x=350, y=400)
    name_entry = Entry(top4, bd=3)
    name_entry.place(x=160, y=20)
    category_entry = Entry(top4, bd=3)
    category_entry.place(x=150, y=80)
    amount_entry = Entry(top4, bd=3)
    amount_entry.place(x=110, y=140)
    method_entry = Entry(top4, bd=3)
    method_entry.place(x=150, y=200)
    date_entry = Entry(top4, bd=3)
    date_entry.place(x=150, y=260)
    selected=tree.focus() #Grab record number
    values=tree.item(selected,'values') #Grab record values
    name_entry.insert(0, values[1])
    category_entry.insert(0, values[2])
    amount_entry.insert(0, values[4])
#function to update a selected record
def update():
    top3.destroy()
    conn=sqlite3.connect('expense.db')
    newname = name_entry.get()
    newcategory = category_entry.get()
    newdate = date_entry.get()
    newamount = amount_entry.get()
    newmethod = method_entry.get()
    c=conn.cursor()
    c.execute("UPDATE sample4 SET name=?,category=?,date=?,amount=?,method=? WHERE number=?", (newname,newcategory,newdate,newamount,newmethod,oldnumber))
    conn.commit()
    conn.close()
    messagebox.showinfo("", "Transaction Details Updated")
    tree.update()
    top4.destroy()
    viewrecord()

```

Fig 5.6 Update Record module

In this module , Necessary details about the transaction are taken from the user and those details are updated for the record selected by the user.

## 5.7 MODULE 7 FUNCTIONALITY

```

#spending_report
def report():
    top5 = TopLevel()
    top5.title("Spending Report")
    top5.geometry("1000x720")
    top5.configure(bg="light grey")
    top5.resizable(False,False)
    conn = sqlite3.connect("expense.db") # create/connect to database
    c = conn.cursor() # create cursor
    #=====HEADER=====
    header=Frame(top5,bg='#009df4')
    header.place(x=0,y=0,width=1070,height=60)
    Label1 = Label(top5, text="Ajay", bg="#009df4", font=("",18,"bold"),bd=0,fg="white",activebackground='#32cf8e').place(x=450, y=15)

    #=====BODY=====
    Label2=Label(top5,text="Spending Report",bg="light grey",font=("",18,"bold"),bd=0,fg="#0064d3").place(x=400,y=80)

    #=====BODY FRAME 1 (WHITE BOX)
    bodyframe1=Frame(top5,bg="white")
    bodyframe1.place(x=30,y=120,width=940,height=310)
    #=====BODY FRAME 2 (GREEN BOX)
    bodyframe2 = Frame(top5, bg="green")
    bodyframe2.place(x=30, y=450, width=250, height=100)
    totalamt=Label(top5,text="Total Amount Spent",bg="green",fg="white",font=("",13,"bold")).place(x=65,y=460)
    c.execute("SELECT sum(amount) FROM sample4")
    rows=c.fetchone();
    sum=rows[0];
    sumLabel=Label(top5,text="Rs."+str(sum),bg="green",fg="white",font=("",18,"bold")).place(x=90,y=500)
    c.execute("SELECT sum(amount) FROM sample4 GROUP BY category") # getting money spent on each category
    rows = c.fetchall()
    #converting to list
    amounts=[]
    l = len(rows)
    for i in range(l):
        amt, = rows[i]
        amounts.append(amt)
    #creating and displaying bar graph in window
    fig=Figure(figsize=(5.5,3),dpi=100)
    a=fig.add_subplot(111)
    a.bar(category,amounts)
    a.set_xlabel = ("Category")
    a.set_ylabel = ("Amount spent")
    a.set_title("Money spent of various products")
    canvas=FigureCanvasTkAgg(fig,top5)
    canvas.get_tk_widget().place(x=30,y=130)
    canvas.draw()

    c.execute("SELECT category FROM sample4 GROUP BY category") # getting category names
    rows = c.fetchall()
    # converting to list
    category = []
    l = len(rows)
    for i in range(l):
        cat, = rows[i]
        category.append(cat)

```

Fig 5.7 Spending Report module

In this module , the program analyzes the data of various transactions present in the database and displays a report summarizing the users spending pattern.

## CHAPTER 6

# RESULTS

### 6.1 Snapshots

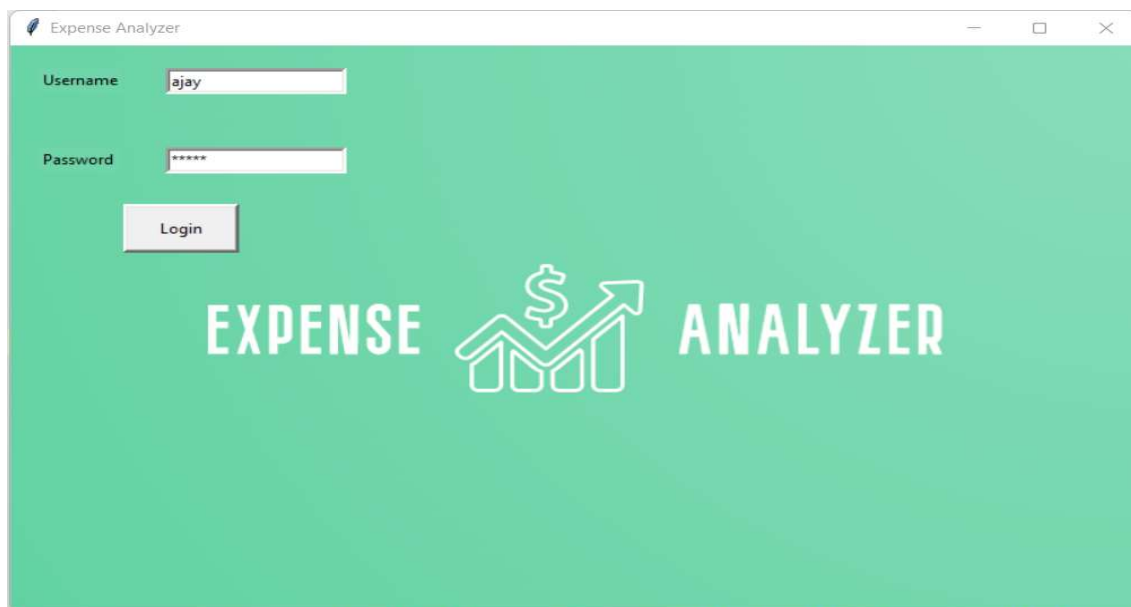


Fig 6.1 Login Page

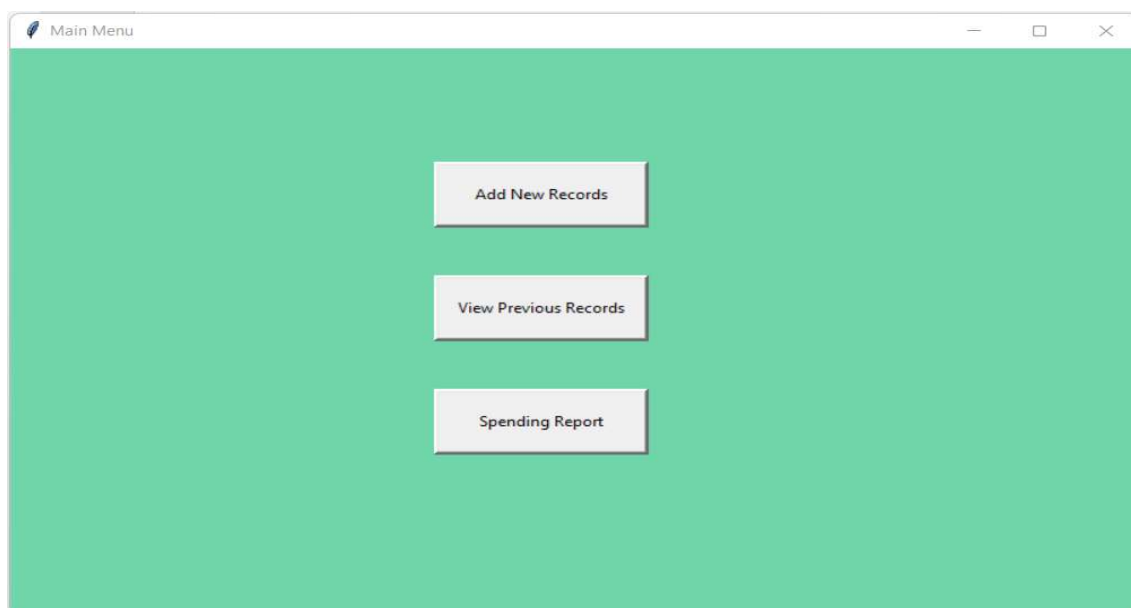
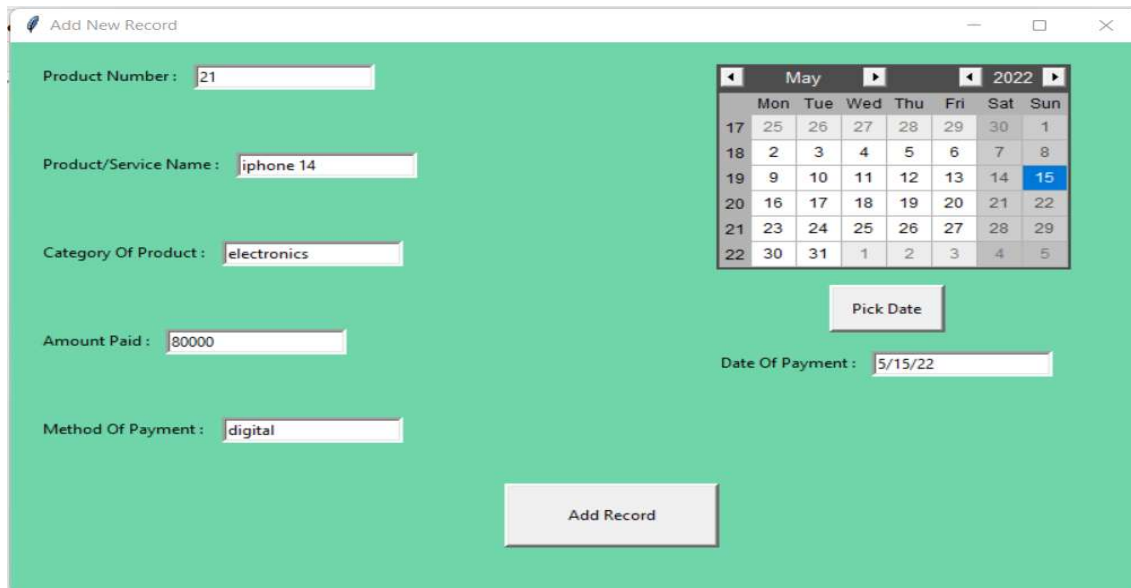


Fig 6.2 Main Menu



The 'Add New Record' form is a web application window with a light green background. It contains several input fields for recording an expense: 'Product Number' (value: 21), 'Product/Service Name' (value: iphone 14), 'Category Of Product' (value: electronics), 'Amount Paid' (value: 80000), and 'Method Of Payment' (value: digital). To the right of these fields is a calendar widget for May 2022, with the date 15 selected. Below the calendar is a 'Pick Date' button. At the bottom center of the form is a large 'Add Record' button.

|    | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|----|-----|-----|-----|-----|-----|-----|-----|
| 17 | 25  | 26  | 27  | 28  | 29  | 30  | 1   |
| 18 | 2   | 3   | 4   | 5   | 6   | 7   | 8   |
| 19 | 9   | 10  | 11  | 12  | 13  | 14  | 15  |
| 20 | 16  | 17  | 18  | 19  | 20  | 21  | 22  |
| 21 | 23  | 24  | 25  | 26  | 27  | 28  | 29  |
| 22 | 30  | 31  | 1   | 2   | 3   | 4   | 5   |

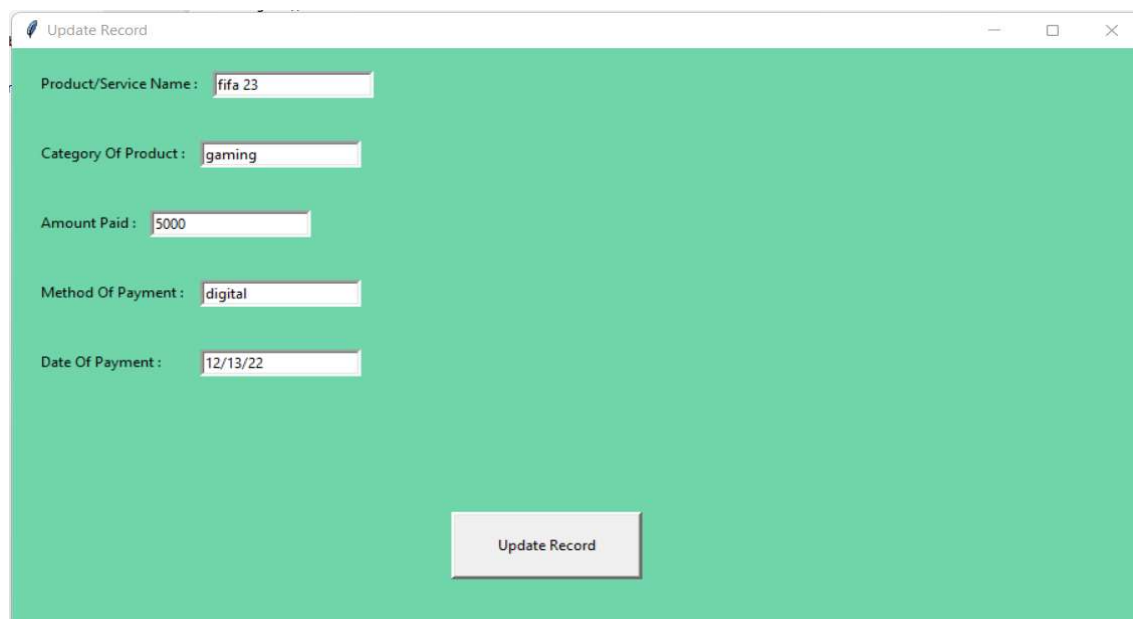
Fig 6.3 Add Record



The 'View Records' window displays a table of expense records. The table has six columns: Product Number, Product Name, Product Category, Date Of Payment, Amount Paid, and Method Of Payment. There are 10 records listed. Below the table are two buttons: 'Remove Selected Record' and 'Update Selected Record'.

| Product Number | Product Name | Product Category | Date Of Payment | Amount Paid | Method Of Payment |
|----------------|--------------|------------------|-----------------|-------------|-------------------|
| 1              | shirt        | clothing         | 12/22/22        | 500         | cash              |
| 2              | fifa 23      | gaming           | 12/13/22        | 5000        | digital           |
| 3              | shoe         | clothing         | 12/8/22         | 4000        | cash              |
| 4              | laptop       | electronics      | 10/24/22        | 6500        | digital           |
| 5              | burger       | food             | 11/17/22        | 250         | cash              |
| 6              | headphones   | electronics      | 10/6/22         | 800         | cash              |
| 7              | pasta        | food             | 11/10/22        | 200         | cash              |
| 8              | smart watch  | electronics      | 9/8/22          | 3000        | digital           |
| 9              | dosa         | food             | 9/15/22         | 100         | cash              |
| 10             | sweater      | clothing         | 8/18/22         | 2000        | cash              |

Fig 6.4 View Records



The 'Update Record' window is a green-themed interface for updating an expense record. It contains five input fields with labels and a single 'Update Record' button at the bottom center.

Product/Service Name :

Category Of Product :

Amount Paid :

Method Of Payment :

Date Of Payment :

Fig 6.5 Update Record



The 'View Records' window displays a table of expense records. The table has six columns: Product Number, Product Name, Product Category, Date Of Payment, Amount Paid, and Method Of Payment. There are 10 records in total. The 6th record, 'headphones', is highlighted in red. Below the table are two buttons: 'Remove Selected Record' and 'Update Selected Record'.

| Product Number | Product Name | Product Category | Date Of Payment | Amount Paid | Method Of Payment |
|----------------|--------------|------------------|-----------------|-------------|-------------------|
| 1              | shirt        | clothing         | 12/22/22        | 500         | cash              |
| 2              | fifa 23      | gaming           | 12/13/22        | 5000        | digital           |
| 3              | shoe         | clothing         | 12/8/22         | 4000        | cash              |
| 4              | laptop       | electronics      | 10/24/22        | 6500        | digital           |
| 5              | burger       | food             | 11/17/22        | 250         | cash              |
| 6              | headphones   | electronics      | 10/6/22         | 800         | cash              |
| 7              | pasta        | food             | 11/10/22        | 200         | cash              |
| 8              | smart watch  | electronics      | 9/8/22          | 3000        | digital           |
| 9              | dosa         | food             | 9/15/22         | 100         | cash              |
| 10             | sweater      | clothing         | 8/18/22         | 2000        | cash              |

Fig 6.6 Delete Record

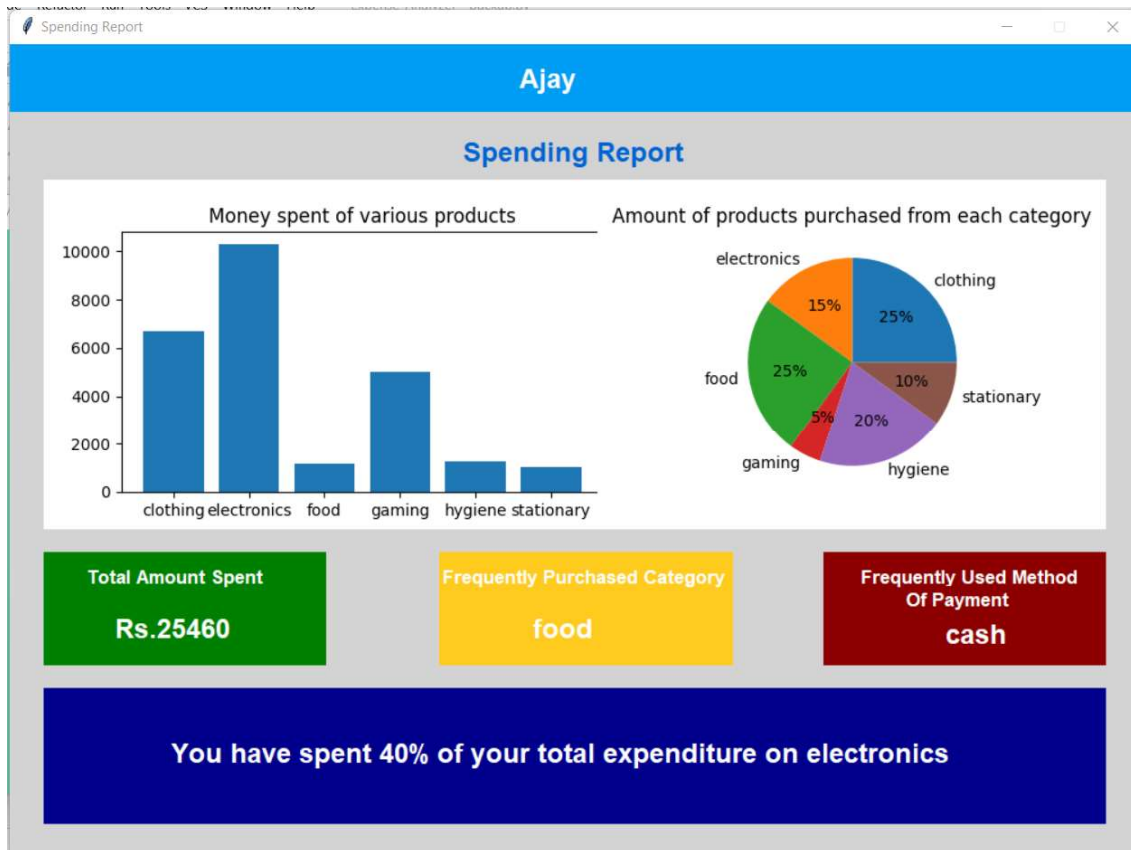


Fig 6.7 Spending Report



## CHAPTER 7

### CONCLUSION

This mini project has successfully accomplished the goals it had set out in the objectives and design sections of this report.

This mini project serves as a companion to the users as it helps in simplifying the process of keeping track of the previously made transactions. It allows the users to enter the details of a transaction and store it in a database. This project also gives the users detailed information about the spending pattern of the user so that the user will be aware when making his/her next transaction and not spend carelessly which will lead to financial problems. It also makes the process of updating or deleting the details of a previously stored transaction details in the database as hassle-free as possible.

This mini project provides a user friendly interface for the users so that they can keep track of their financial transactions and help them in spending their money more efficiently and avoid over spending. It helps the users to become financially responsible.

## REFERENCES

- 1) <https://www.w3schools.com>
- 2) <https://www.programiz.com>
- 3) <https://www.geeksforgeeks.org>
- 4) <https://www.tutorialspoint.com>