AMRITSAR GROUP OF COLLEGES

**Autonomous status conferred by UGC | NAAC-A Grade,**

# Six Weeks Industrial Training Report

**On**

# “PERSONAL EXPENSE MANAGER”

Submitted in the Partial fulfillment of the requirement for the Award of Degree of

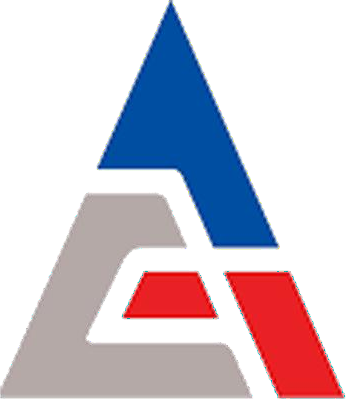
# Bachelor of Technology

in

# COMPUTER SCIENCE & ENGINEERING

Batch

(2020-24)



**Submitted To Submitted by**

Head of Department Gourab Sharma(2000098)

(CSE) Bandana Rajput(2000080)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**ABSTRACT**

Daily Expense Tracker Mobile Application is an application meant for all type of people concerned and cautious about their daily expenses. With the help of this application, users will be able to more careful of their spendings and save more. Also, this application is an easier alternative to keeping track of users’ use of money than the traditional way of writing their expenses in their diary. This application implements least squares method which helps to predict an outcome by finding the best fit line for a set of data. The use of the least squares method will help users in obtaining a successful budget planned with the prediction of the outcome of the budget based on expenses. In conclusion, this application will incorporate the function of generating a monthly expense report and budget outcome predictions

**ACKNOWLEDGEMENT**

This is a humble effort to express our sincere gratitude towards those who have guided and helped us to complete this project.

This project is major milestone during the training period of six weeks. As such this project was a challenge to us and was an opportunity to prove our caliber. We are highly grateful and obliged to each and every one making us help out of problems being faced by us. It would not have been possible to see through the undertaken project without the guidance of VMM instructors.

It was purely on the basis of their experience and knowledge that we able to clear all the theoretical and technical hurdles during the development phases of this project work.

Last but not the least we are very thankful to our Head of Department Dr SANDEEP KUMAR and all Members of Computer Science Deptt. who gave us an opportunity to face real time problems while fulfilling need of an organization by making projects for them

**DECLARATION**

We undersigned solemnly declare that the report on project **PERSONAL EXPENSE MANAGER**is based on our own work carried out during the course of our training.

We assert the statements made and conclusions drawn are an outcome of our research work. We further certify that

1. The work contained in the report is original and has been done by us.
2. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this college or any other of India or abroad.
3. We have followed the guidelines provided by the college in writing the report.
4. Whenever We have used materials (data, theoretical analysis and text) from other sources, We have given due credit to them in the text of the report and giving their details in the references.

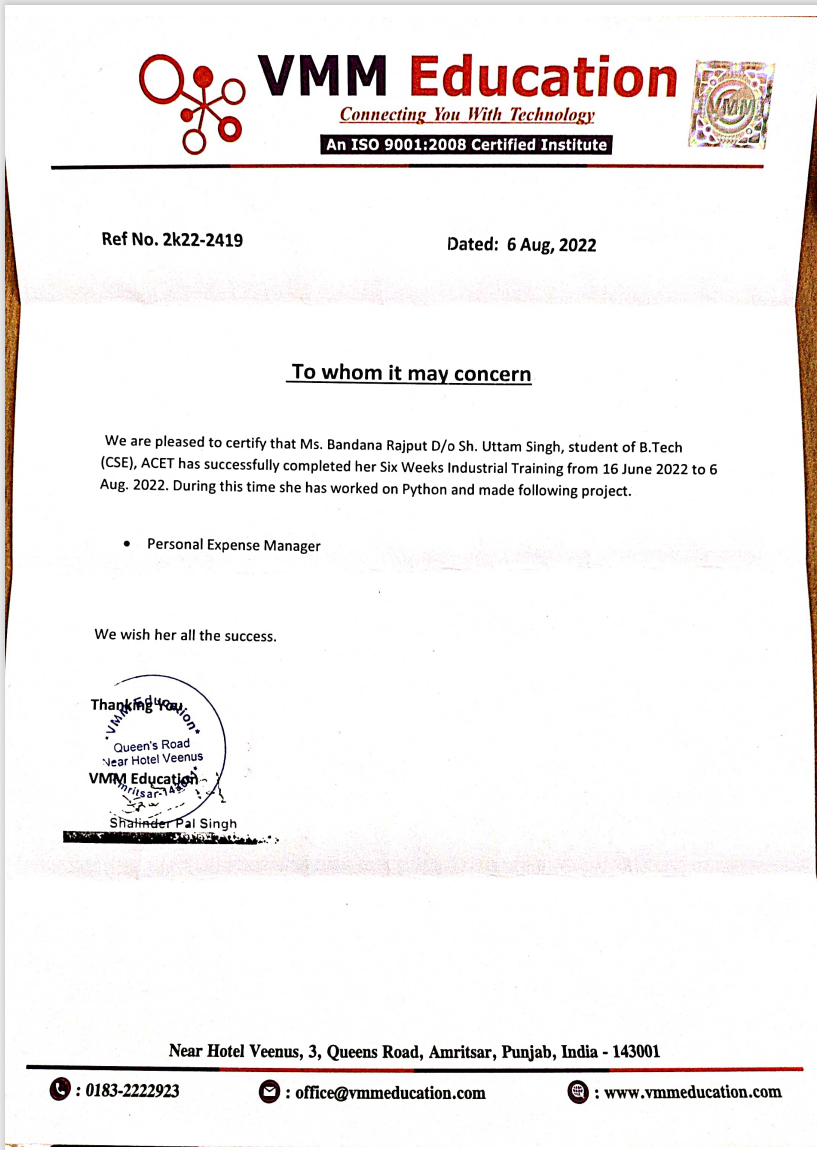
**Gourab Sharma**

**(2000098)**

**Bandana Rajput**

**(2000080)**

****

****

**TABLE OF CONTENTS**

**S.No. Content Page No.**

1 Abstract I

2 Acknowledgement II

3 Declaration III

4 Certificates IV-V

5 List Of Figures VII

6 List Of Tables VIII

7 Introduction to the Project 1-3

8 Hardware and Software Requirements 4

9 Technology Used 5-9

10 Requirement Gathering 10

11 Feasibility Study 11

12 Software Requirement Specifications 12-16

13 Design Phase 17-18

14 Implementation 19-23

15 Testing 24-25

16 Conclusion 26

17 References 27

18 Appendix 28-64

**List Of Figures**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Figure Name** | **PageNo.** |
| 1 | Fig-2.1Usecase Diagram | 10 |
| 2 | Fig-3.1 ER Diagram | 17 |
| 3 | Fig-3.2 DFD-0 | 18 |
| 4 | Fig-3.3 DFD-1 | 18 |

**List Of Tables**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Name** | **PageNo.** |
| 1 | Tab-2.1 LoginPage Test Case | 24 |
| 2 | Tab-2.2 Add Budget Test Case | 25 |

**INTRODUCTION TO THE PROJECT**

This project is based on our learning which we have done on our summer holiday. Our project is PERSONAL EXPENSE MANAGER . This project is made in the team work of Gourab Sharma (2000098) and Bandana Rajput (2000080) .

Expense Manager is a small software which is used to keep the records of all the expense spent by the user . It helps the user to understand and analize his expenses and his/her budget Here user can set a budget for a particular month and spent according to that . User can add his expense, whenever the user added his/her expenses, that amount of expense subtract from the particular month subject and give the user his/her remaining budget . There is the possibility that sometimes user may spent more money as according to his/her budget ,then while adding the extra amount expense, this manager will ask the user from Yes/No messagebox, Do you want to add this expense. If user click on Yes then only that expense will be added otherwise the remaining budget remain same. If extra expense added that cross the budget amount, our expense manager will show negative budget .

This expense manager will open with a page in the user select his role that the user is a admin or a simple user . If user is a simple user then user has to add his / her login information like name and password . If the correct information added then only the user can login successfully otherwise it denied the request. On the other hand, if you are a new user then you must click on sign up and create your account by adding information like name , email , mobile and password . After registering a new account ,it will directly take us to login page .So, now a new user can login by entering his/her credentials . After successfully login , user home page will open which has the option of addig the budget . Here user can add his budget for the current month by adding required information . Now that information show on main page . As our main page contain the three labels like total expense, total Budget, remaining Budget . Here now our Total Budget show the budget of our particular month. As we havenot added the expense, so Total Expense and Total budget columns are empty . As we add the expense, the values changes accordingly . A treeview created by us which show all the expenses added by the user. We have different categories for expenses like food , fuel ,shopping , clothes etc .Profile menu contain the option Change password and Logout . Thus , a user has also the

Option of changing his/her password.

If the main user is a admin then admin can login by entering his credentials. There is no option of registering for a admin as admin must by entered by trusted person. When admin can login successfully , admin homepage will open, which contain some set of rules for admin . On the top of admin homepage , there is Manage admin menu, in which there are two options like add admin , view admin . These option also clear their usage as add admin is used for adding a new admin and view admin is used for viewing all the admins of the particular manager . For adding a new admin we need to add the information like name, email , password and role . A new admin can be added by the choice of old admin. All the admins has the right to view all the admins who are present. Other option is of adding category and viewing category . Atlast profile section contains the options like Logout and Change Password. A admin is a person who performs some operations on user. We added a buuton who help us to perform on users , we added the options like fetch details (for fetching all the detals of a particular user on the box ), show all ( for showing all the users of the expense manager ), delete user ( for deleting a particular user), update user ( for updating a particular user).

I recommend our this project of PERSONAL EXPENSE MANAGER to those person who have able to understand how much they spend unusually which noticing their expenses. If a person thing that he/she can control his expenses by seeing his expenses amount, then person must have to use this expense manager.

In the XAMPP, we have created a database by the name of Expense Manager.In the databade Expense Manager , we have created my tables:--

Various tables Used in the project:--

1 ADMIN:- This table is used to store the admins of the table. A admin is the head in the project.

(A). Username

(B). Email

(C). Password

(D). Role

2.BUDGET:- Budget table is used to add the amount that he/she has the ability to spend in the current month

(A). Id

(B). Amount

(C). Month

(D). Year

(E). Userid

3. CATEGORY:- Category table is used to tell which type of expense is and what is the name and description of that expense

(A). Name

(B). Description

4. EXPENSE:- Expense table is used to add the details of expense that user has spend.Expense has following info with it

(A). Id

(B). Userid

(C). Amount

(D). Date

(E). Category

5. USER:- This table is the main table in the whole project. A user access the table and add the various information like budget,expense. A user use the expense manager .Actually this project is made for user

(A). Id

(B). Name

(C). Email

(D). Mobile

(E). Password

**HARDWARWE & SOFTWARE REQUIREMENTS:**

**HARDWARE REQUIREMENTS:-**

1. Intel i3/i5/i7 processor
2. 4 GB RAM
3. 80 GB + Hard disk space
4. Keyboard
5. Mouse

**SOFTWARE REQUIREMENTS:-**

1. An operating system
2. Xampp
3. IDE for Python(Pycharm)

**TECHNOLOGY USED IN PROJECT:**

**Tkinter:**

The [tkinter](https://docs.python.org/3/library/tkinter.html#module-tkinter) package (“Tk interface”) is the standard Python interface to the Tcl/Tk GUI toolkit. Both Tk and [tkinter](https://docs.python.org/3/library/tkinter.html#module-tkinter) are available on most Unix platforms, including macOS, as well as on Windows systems

Running python -m tkinter from the command line should open a window demonstrating a simple Tk interface, letting you know that [tkinter](https://docs.python.org/3/library/tkinter.html#module-tkinter) is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version.

Tkinter supports a range of Tcl/Tk versions, built either with or without thread support. The official Python binary release bundles Tcl/Tk 8.6 threaded. See the source code for the \_tkinter module for more information about supported versions.

Tkinter is not a thin wrapper, but adds a fair amount of its own logic to make the experience more pythonic. This documentation will concentrate on these additions and changes, and refer to the official Tcl/Tk documentation for details that are unchanged



Fig-1.1 PythonLogo

**MYSQL**

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications.

It is developed, marketed, and supported by MySQL AB, a Swedish company, and written in C programming language and C++ programming language. The official pronunciation of MySQL is not the My Sequel; it is My Ess Que Ell. However, you can pronounce it in your way. Many small and big companies use MySQL. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

#### History:

MySQL was created by a Swedish company, MySQL AB, founded by David Axmark, Allan Larsson and Michael "Monty" Widenius. Original development of MySQL by Widenius and Axmark began in 1994. The first version of MySQL appeared on 23 May 1995. It was initially created for personal usage from MySQL based on the low-level language ISAM, which the creators considered too slow and inflexible. They created a new SQL interface, while keeping the same API as MySQL. By keeping the API consistent with the MySQL system, many developers were able to use MySQL instead of the (proprietarily licensed) MySQL antecedent.

#### Features:

MySQL is offered under two different editions: the open source MySQL Community Server and the proprietary Enterprise Server. MySQL Enterprise Server is differentiated by a series of proprietary extensions which install as server plugins, but otherwise shares the version numbering system and is built from the same code base.

Major features as available in MySQL 5.6:

* A broad subset of ANSI SQL 99, as well as extensions
* Cross-platform support
* Stored procedures, using a procedural language that closely adheres to SQL/PSM[62]
* Triggers
* Cursors
* Updatable views
* Online DDL when using the InnoDB Storage Engine.
* Information schema
* Partitioned tables with pruning of partitions in optimizer
* Shared-nothing clustering through MySQL Cluster
* Multiple storage engines, allowing one to choose the one that is most effective for each table in the application.



Fig-1.2 MysqlLogo

**Pandas:**

Python Pandas is defined as an open-source library that provides high-performance data manipulation in Python. This tutorial is designed for both beginners and professionals.

It is used for data analysis in Python and developed by **Wes McKinney** in 2008. Our Tutorial provides all the basic and advanced concepts of Python Pandas, such as Numpy, Data operation and Time Series

## Key Features of Pandas

* It has a fast and efficient DataFrame object with the default and customized indexing.
* Used for reshaping and pivoting of the data sets.
* Group by data for aggregations and transformations.
* It is used for data alignment and integration of the missing data.
* Provide the functionality of Time Series.

****

Fig-1.3 PandasLogo

# TKcalendar

# tkcalendar is a python module that provides the Calendar and DateEntry widgets for Tkinter. The DateEntry widget is similar to a Combobox, but the drop-down is not a list but a Calendar to select a date. Events can be displayed in the Calendar with custom colors and a tooltip displays the event list for a given day. tkcalendar is compatible with both Python 2 and Python 3. It supports many locale settings (e.g. ‘fr\_FR’, ‘en\_US’, ..) and the colors are customizable.This module shows our project like we are using a specalised applicatiom. We have to choose the dates on calenedar rather than writing , a simple date

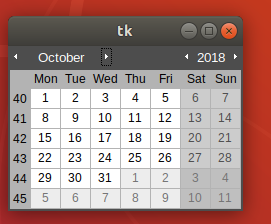
****

Fig-1.4 TkcalenderLogo

**REQUIREMENT GATHERING**

The requirement gathering is process of requirements discovery or generating list of requirements or collecting many requirements as possible by stakeholders. It is also called as requirements elicitation or requirement capture.

The requirements of the built up of an expense and income tracker are gathered from various sources . Reports and articles of previous expense and income tracker are reviewed in order to understand and gather the needed requirements. In this phase, technical and usability of previous systems are evaluated to achieve needed information. We also talk to some persons and ask their view how a expense manager should be and what are the features they want in their expense manager . The gathered requirements from the requirements gathering phase are used in order to identify and to proceed to the designing requirements phase. This phase is concluded with diagrams being made such as context diagram, data flow diagram and activity diagram. The diagrams are made to enlighten the whole system as to how the workflow of the system works

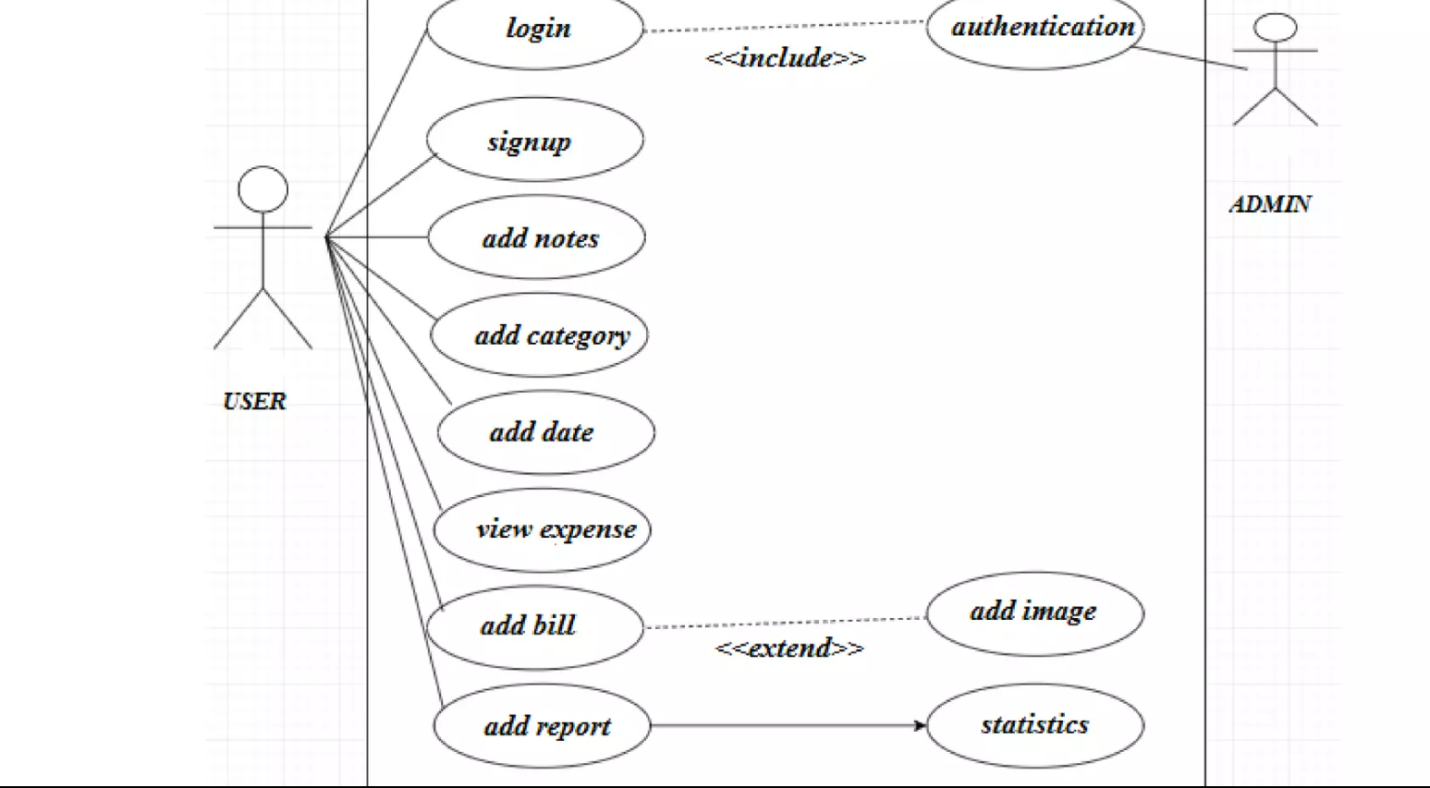


Fig-2.1 Usecase Diagram

**FEASIBILITY STUDY**

After doing the project Expense Manager , study and analysing all the existing or required functionalities of the system, the next task is to do the feasibility study for the project.

A feasibility study is a preliminary study which investigates the information needs of prospective users and determines the resource requirements, determining the cost effectiveness of various alternatives in the designs of the information system, benefits and feasibility of proposed project.

The goal of the feasibility study is to evaluate alternative systems to propose the most feasible and desirable systems for development.

1. **Operational Feasibility**

The operational feasibility is responsible for the operations of mananagement for user as well as admin involved in a project. Our project is operationally feasible because the modules we have considered such as Add and view admin , add and view category and most important add budget and expenses provides a combined platform by using which the related tasks can be handled in a easy manner. Though it doesn’t include the process of deleting account but it is operational to full extent.

1. **Technical Feasibility**

Technical feasibility can be demonstrated if reliable hardware and software capable of meeting needs of proposed system can be developed or acquired by the business in required time. Our project is technically feasible because the required hardware and software needed for our project are made available to us by our lab technician.

1. **Economic Feasibility**

This is a very important aspect to be considered while developing a project. We decided the technology based on minimum possible cost factor.

* All hardware and software cost has to be borne by the organization .
* Overall, we have estimated that the benefits the organization is going to receive from the . proposed system will surely overcome the initial costs and the later on running cost for .. . system.

**SRS – Software Requirement Specification**

**Project Title: PERSONAL EXPENSE MANAGER**

## PURPOSE

The purpose of developing this “Expense Manager” is to help the students to maintain their expenses record and manage their budget accordingly.To help people in analyzing their expense how much budget they have left with for the particular month

## SCOPE

It will integrate the benefits of showing all expenses, remaining budget on same screen. This will provide more flexibility in the modern world structures. It provides a means of collaborator maintaining record of all the spent amount and help them to understand how much money they can spent in the remaining period of month.

## INTRODUCTION

Introduction contain the following sub categories

**Existing System**

The present system is a manual system or a semi-automated system. Manual system involves paper work in the form of maintaining various files and manuals. Maintaining information in the files and manuals is full of pressure and is a tedious process.

A manual system has following disadvantages

* It’s a limited system and fewer users friendly.
* Searching of particular information is critical it takes lot of time.
* In the existing system students need to maintain a file for noticing their expenses.
* In the manual system student need bare the lot of time and cost.
* The existing system needs to save the information in the form of files.
* Limited sharing is possible if the data is in the form of paper or disk drives.
* The manual system gives us less security for saving data, some data may be lost due to mismanagement.
* Wastage a lot of time insearching a item.

### Proposed System

As modern world is going to be advanced , generation of today world want to do every work fastly and easily and infact in less time also. There are obviously many advantages of the expense manager to the users of this application

The system after careful analysis has been identified to be presented with the following modules:

* **User Registration:** A user can register for using the application
* **View admin:**A admin can view all the admin who are present in the particular format
* **Add Admin:** A admin has the right to add another admin.
* **Operation on User:** A admin has the right to perform some operation on user. These operation are update, delete, fetch details and show all the users.
* **Add Category**: An admin also has the right to add new category for the application. The categories which user added are shown to the user on his login page while adding the categories. Various categories included food, clothes,shopping, others
* **View Category**: An admin can view all the categories added by him.
* **Change Password**: Both admin and users has the right to change his/her password whenever they feel the need to do so.
* **Logout**: Logout option is must in every application . This option is available for both user and admin.
* **Add Budget** : This option is for the user. User need to add his budget for the particular month and according to that do his/her expense. User must need to focus that he will not make his/her budget negative
* **View Graph**: A user can view his/her expenses in a graphical manner, so that he can understand how much he/she has spent in what way.
* **Record Storage:** The user information files should be stored in centralized database which can be maintained by the system.
* **Authentication:** Authentication of this application will be provided for only registered members.

## Advantages of Expense Manager

Advantages of the personal expense manager applications for the user of this application are explained below

* **Time and Cost Saving:** It is time and cost saving as no need to buy paper and pen and thus also save our time .
* **Effective Management:** Another advantage is effective management as record is managed in a well mannered form and user can view that record from anywhere
* **Independent of Location:** User can see the record of his/her expenses whenever he/she need that record by just adding the required credentials

# FUNCTIONAL REQUIREMENTS

"Functional requirement describe what a system should do." Functional requirements of our system are explained below.

* Users must have valid User ID and password to login thus creating their individual profiles.
* Administration can update the information of the user
* User can add expenses.
* Administration has the right to view all the user.
* Administration can also delete a particular user whenever the admin want to so..
* User can add his/her budget.
* Users can view their remaining budget
* Administration can add the new admin and view the old admins who are present .
* There can be platform to manage all the expenses at a particular place and all are available at one place.

# NON FUNCTIONAL REQUIREMENTS

Requirement that specifies criteria that can be used to judge the operation of a system are called non functional requirements. Non functional requirements of our system are mentioned below

* + Secure access of confidential data (user’s details).
  + Maximum time availability
  + Better component design to get efficiency at peak time.
  + Flexible service based architecture will be highly desirable for future extension.
  + User registration facility is accessible by administration only.

## SOFTWARE TOOLS

**Database Server:** XAMPP (Mysql)

**Client:** ABC association

**Development Tools:** Pycharm

**Programming Language:** Python

## DEPLOYMENT

**Operating System Server:** Window 10, Linux, UNIX

### HARDWARE SPECIFICATION

**Processor:** Intel Core i5

**RAM:** 4GB

**Hard Disk:** 80 GB

**Keyboard and mouse**

**DESIGN**

**ER Diagram**

The Er diagram for our project is as follow

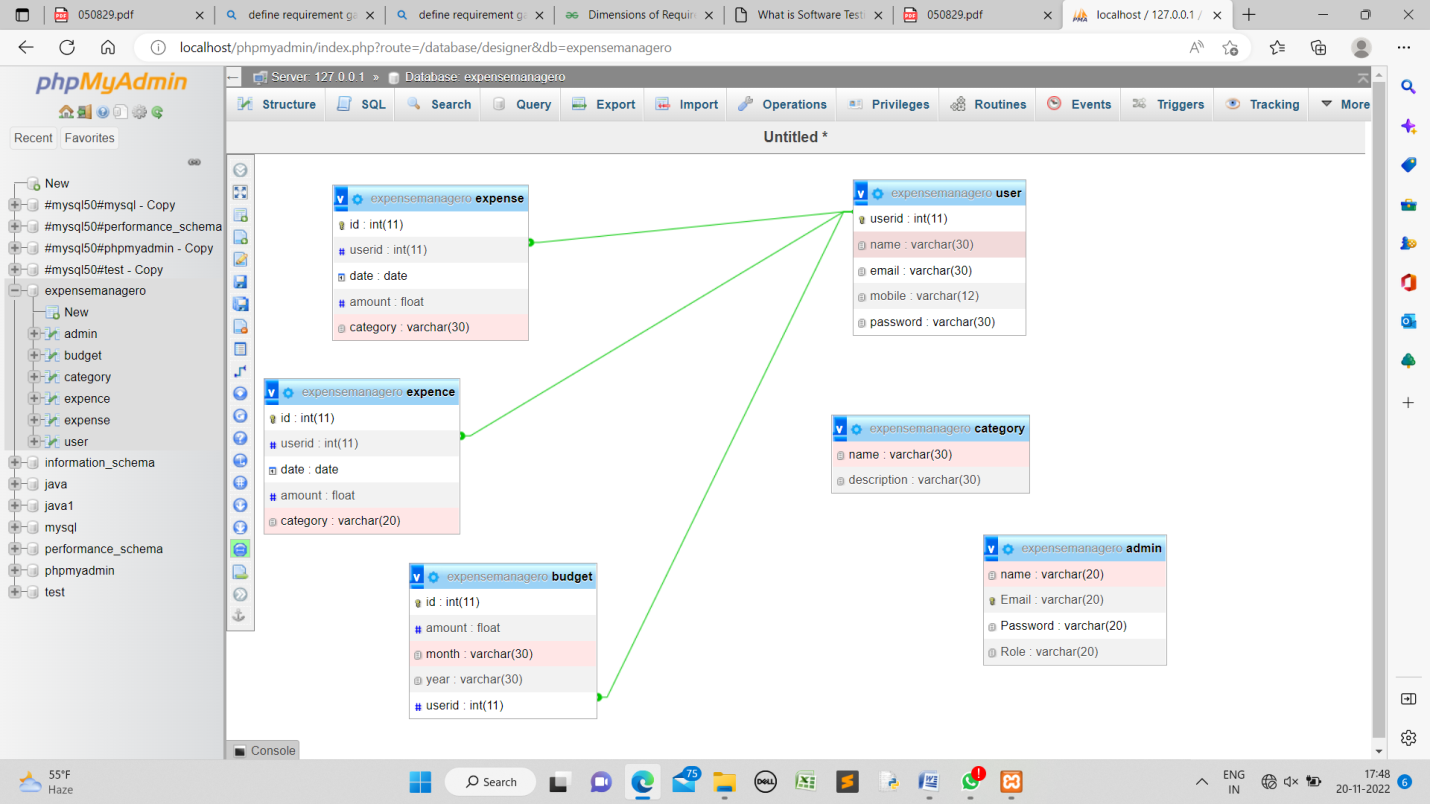
****

Fig-3.1 ER Diagram

**DFD-0**

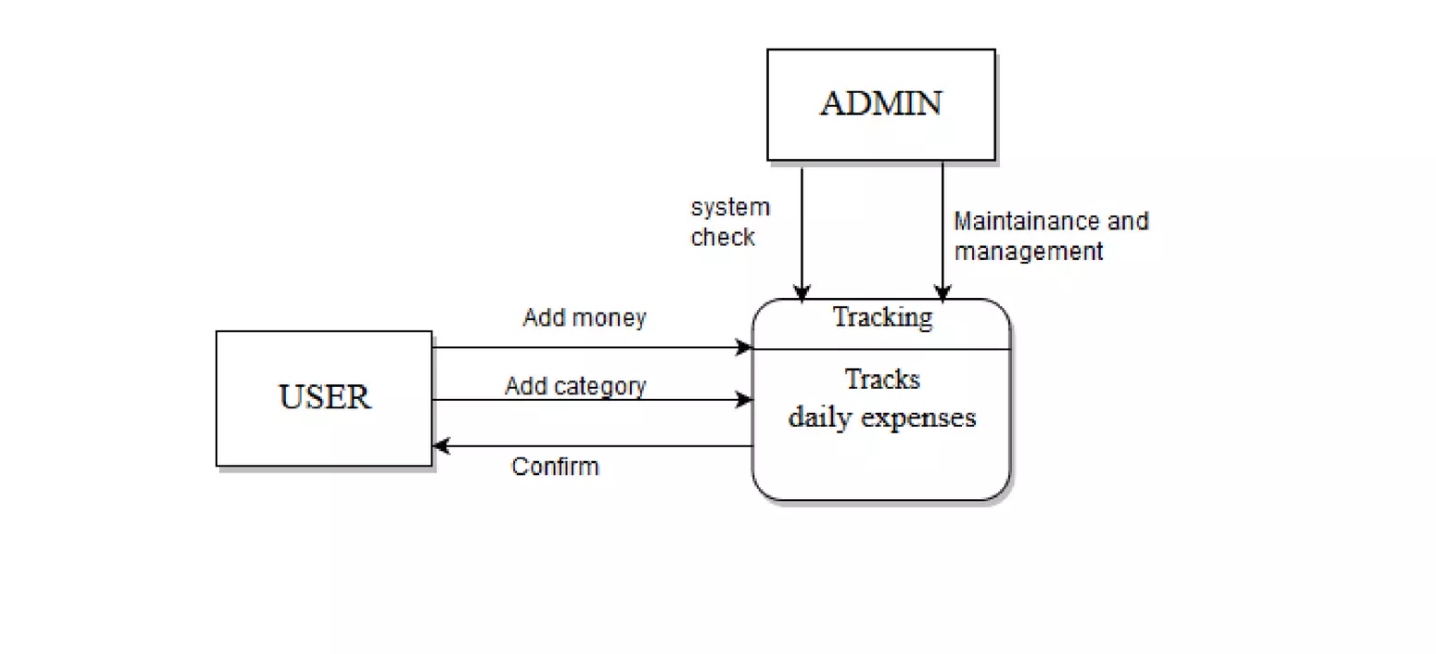
****

Fig-3.2 DFD-0

**DFD-1**

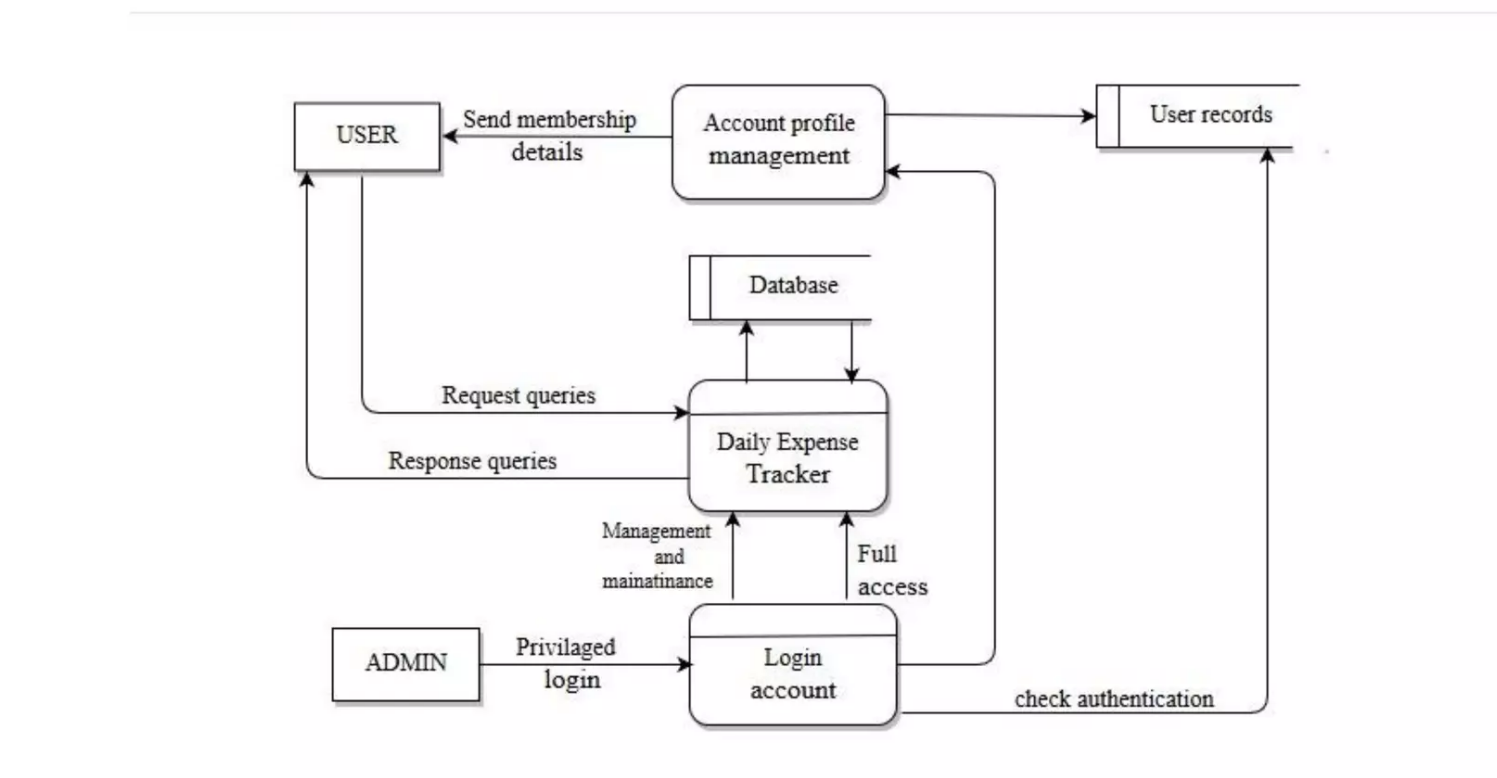
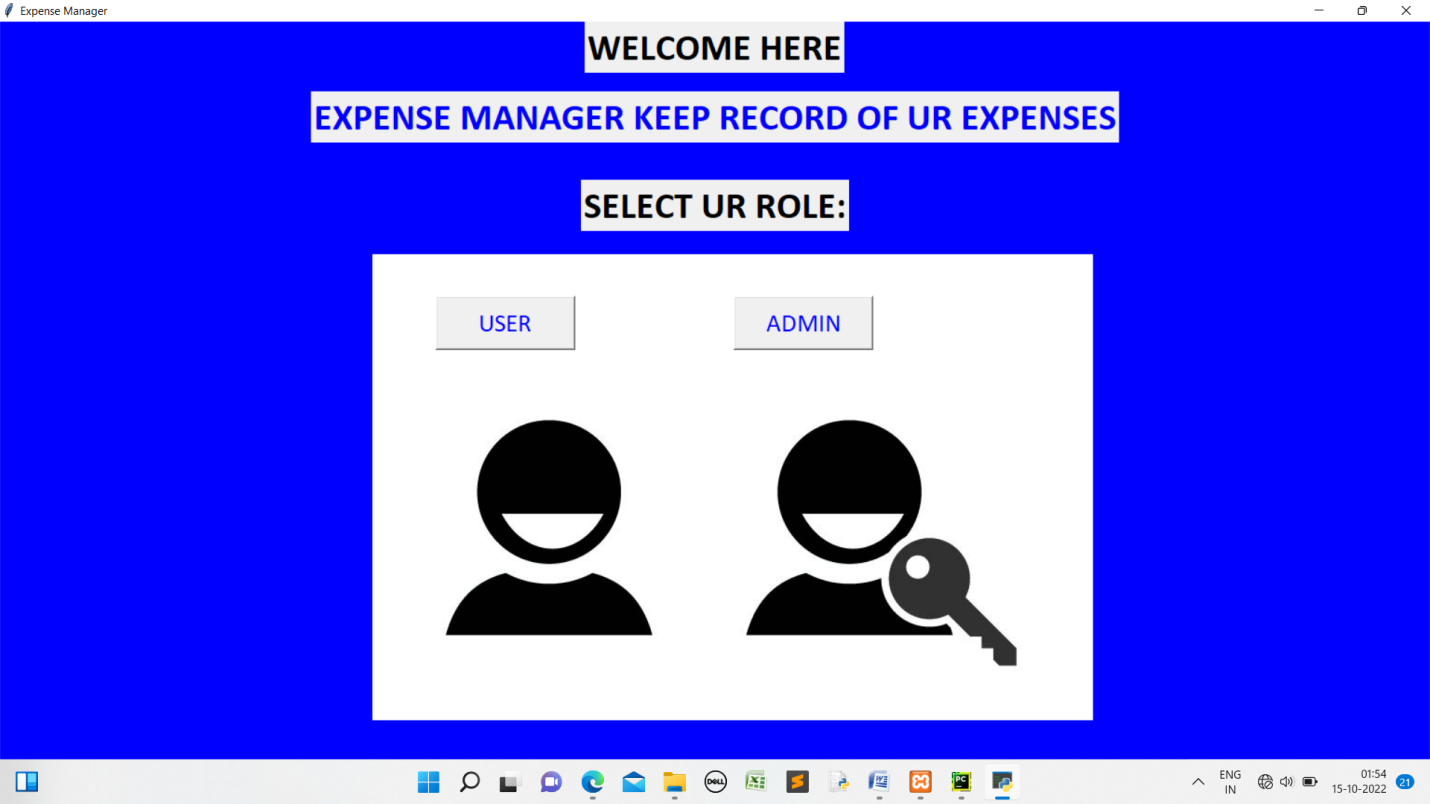
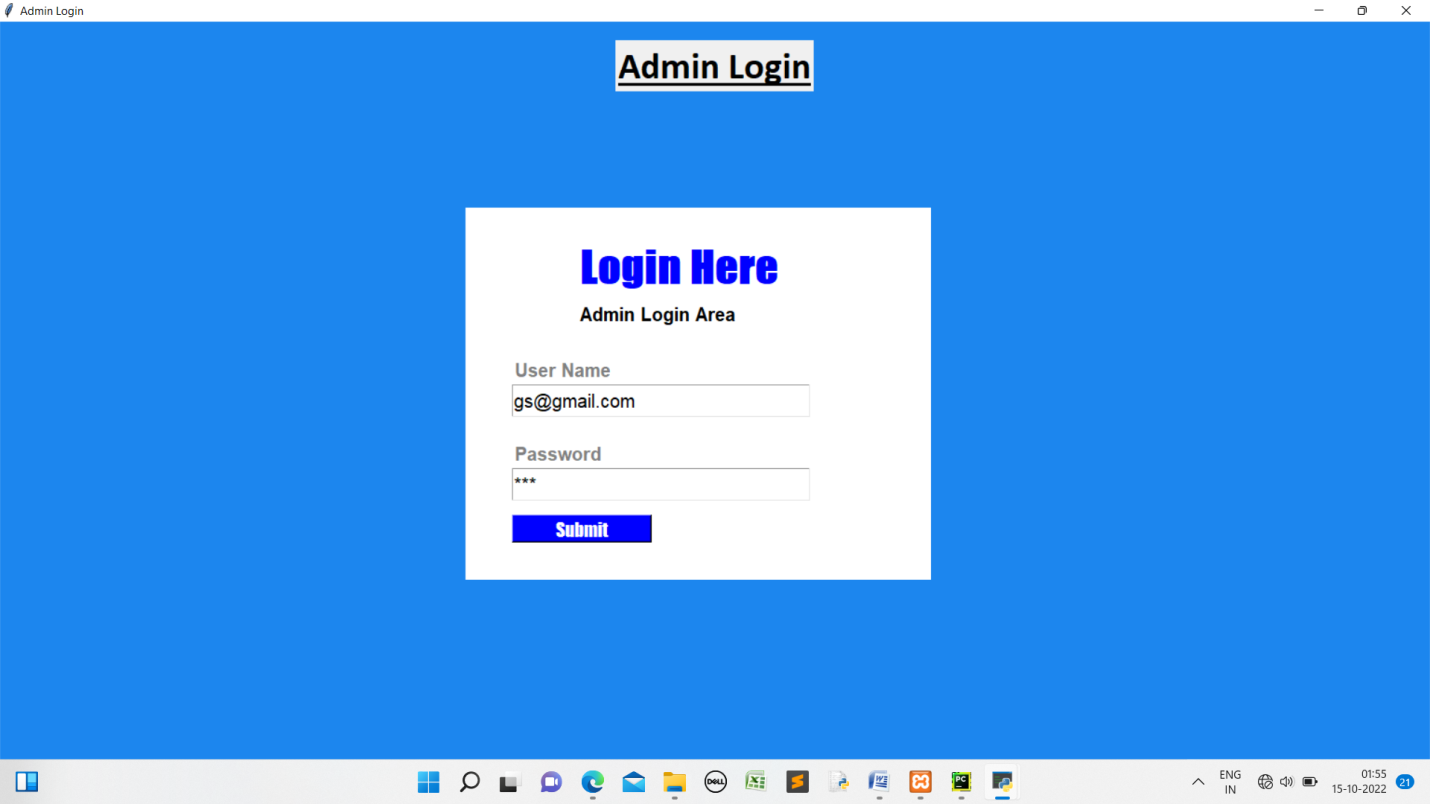


Fig-3.3 DFD-1

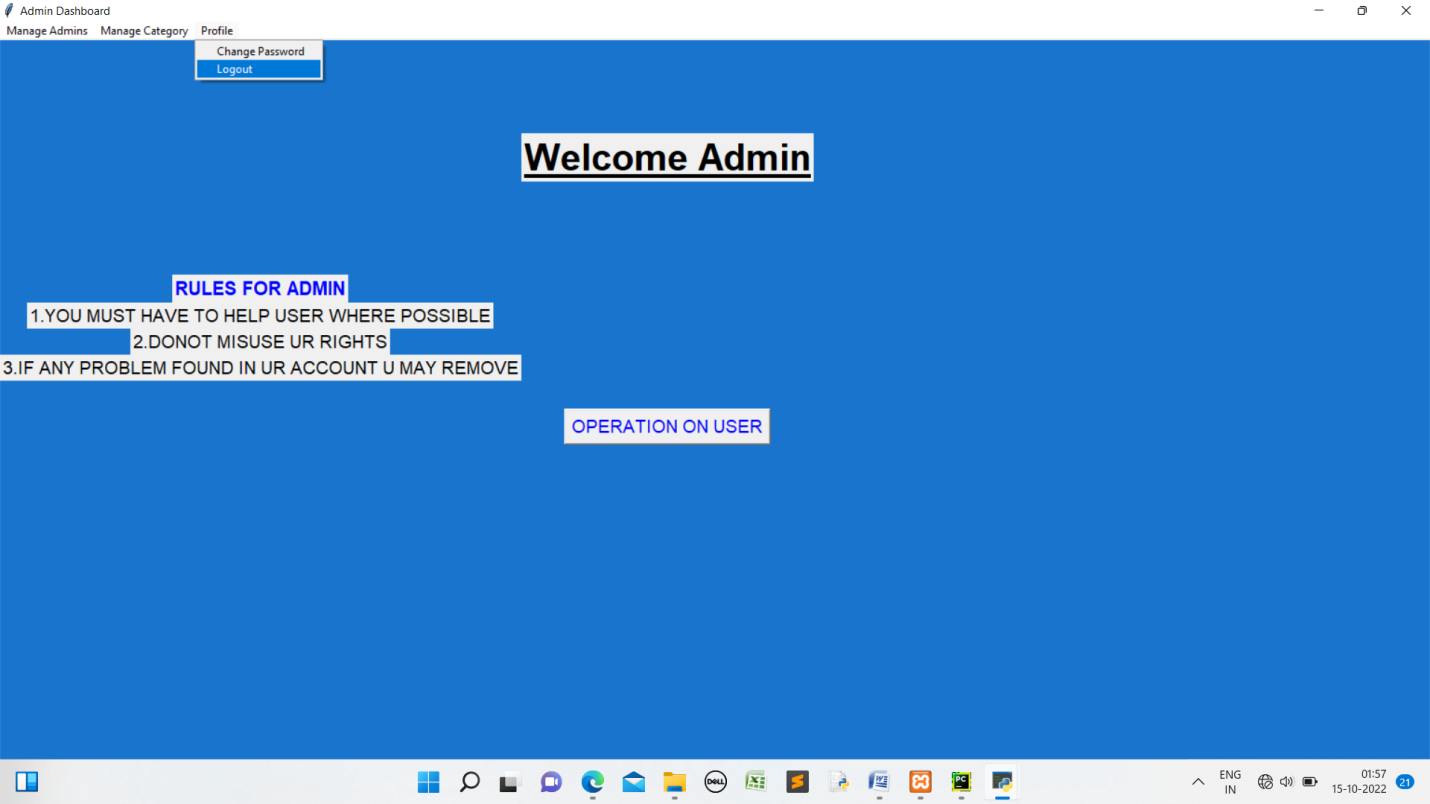
**IMPLEMENTATION**



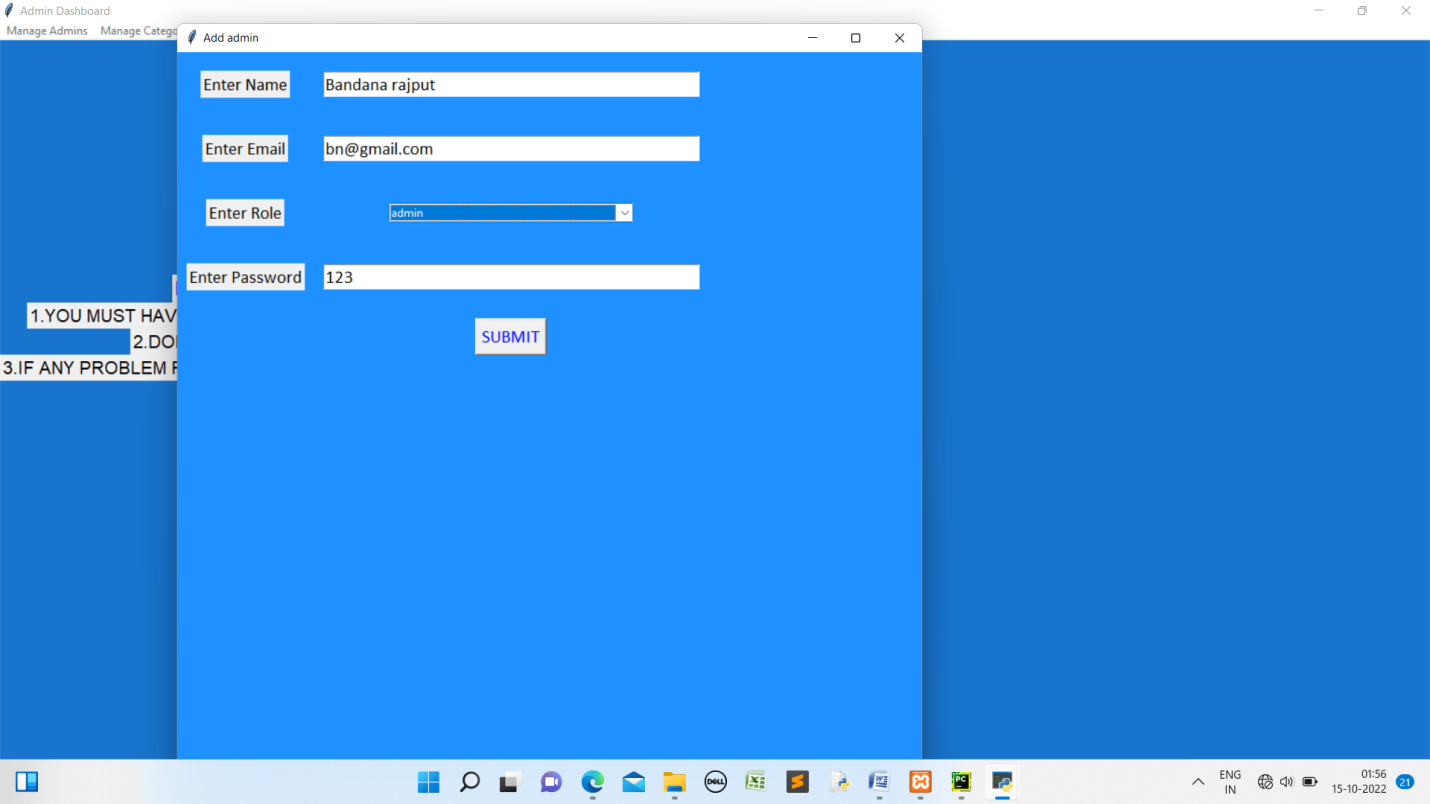
Main Page



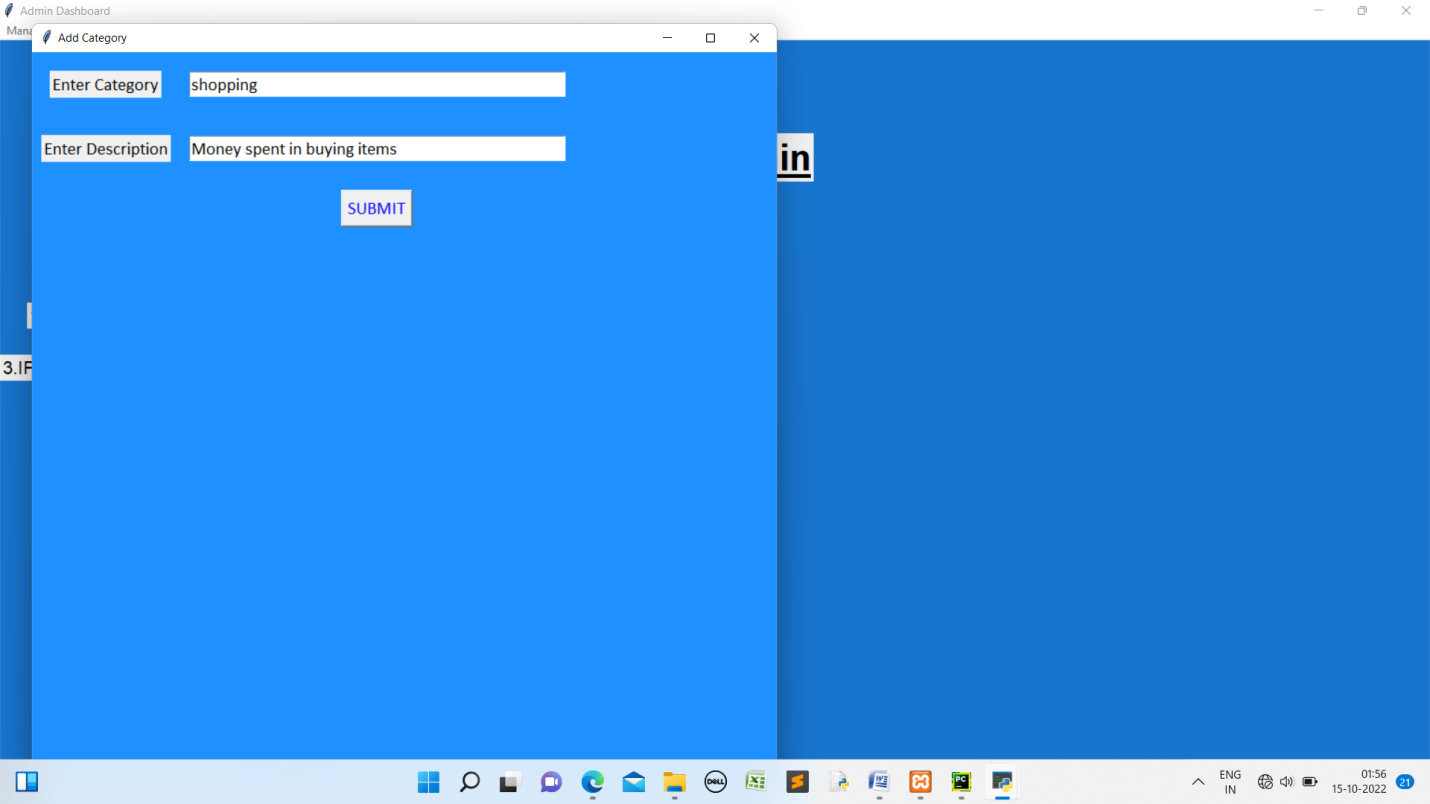
Admin Login Page



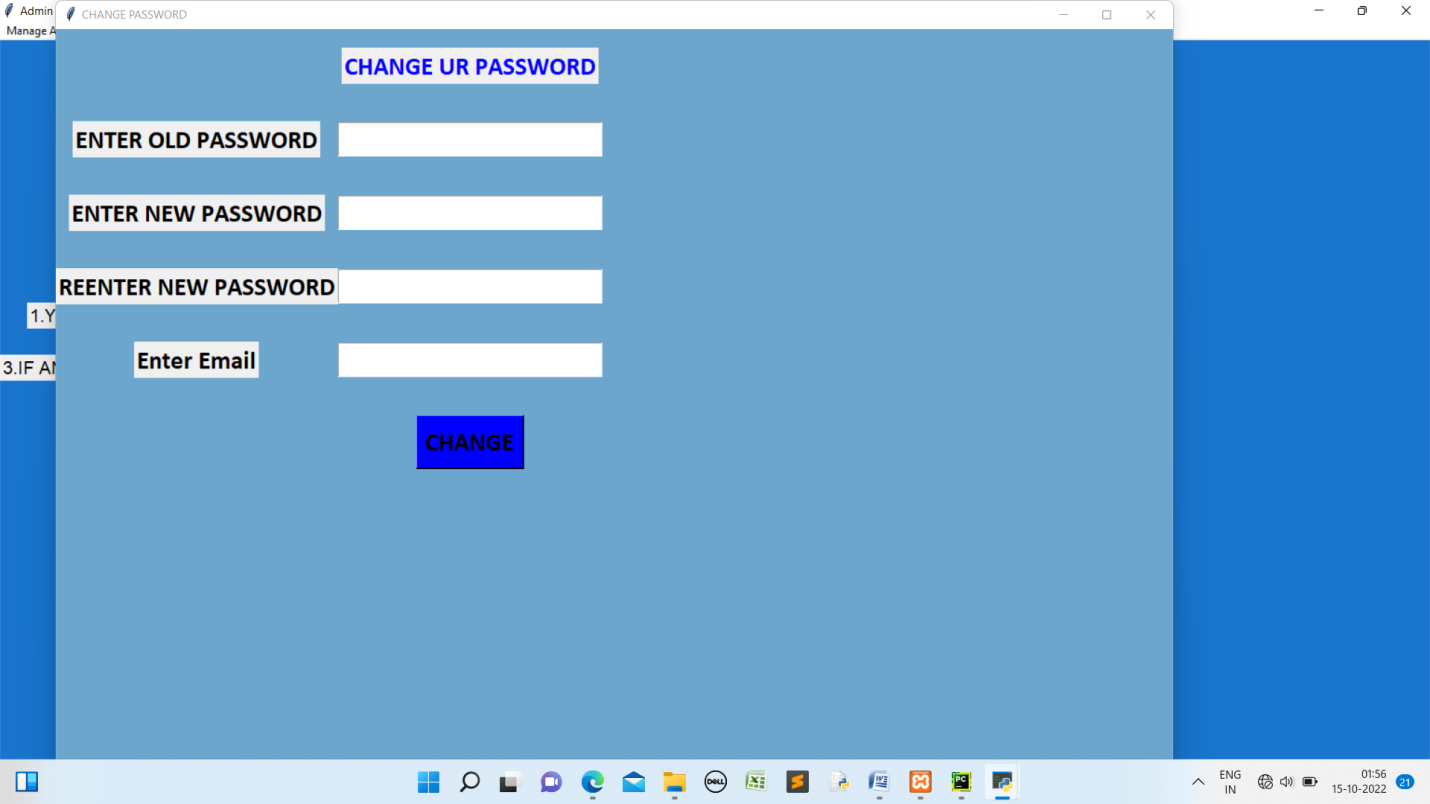
Admin Homepage



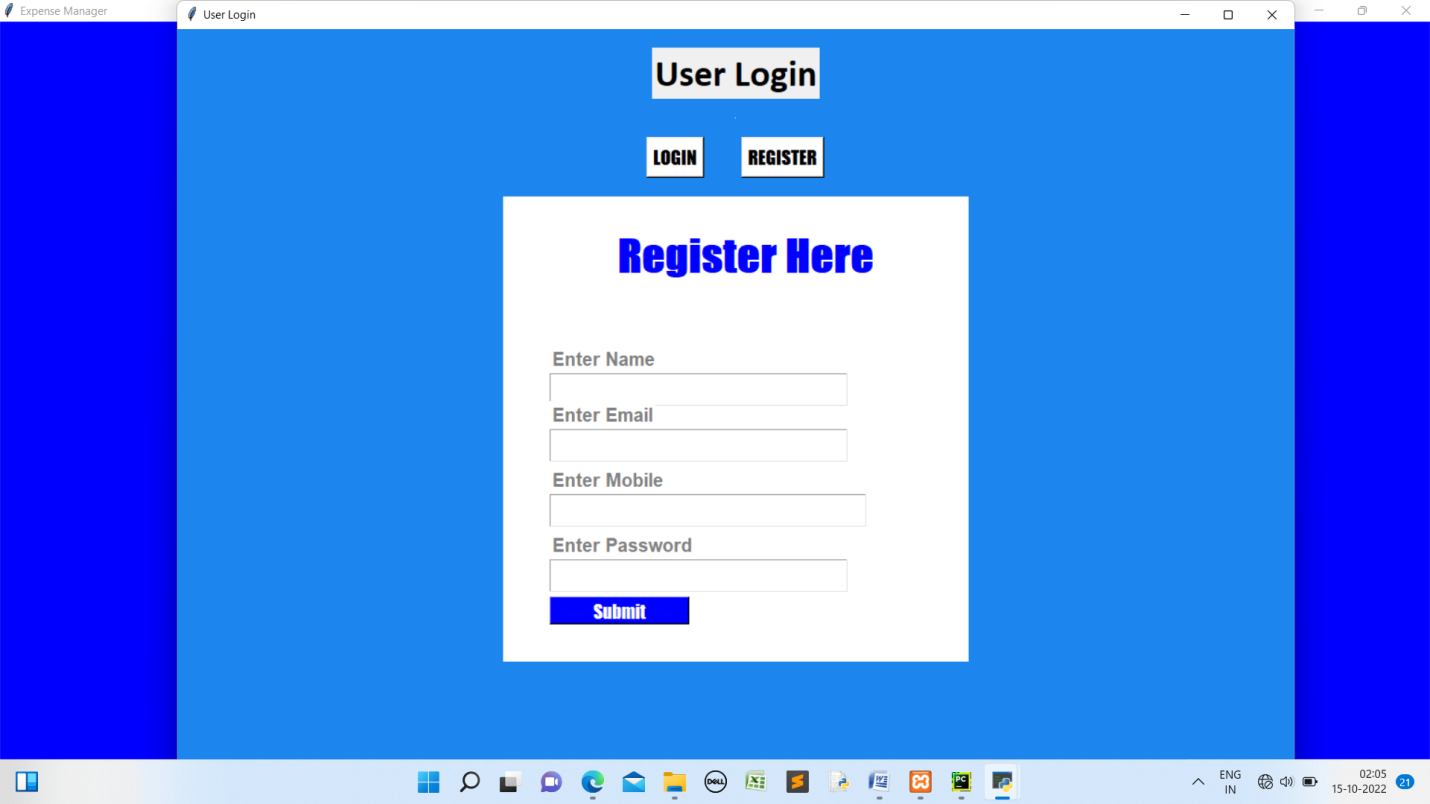
Admin Add admin Page



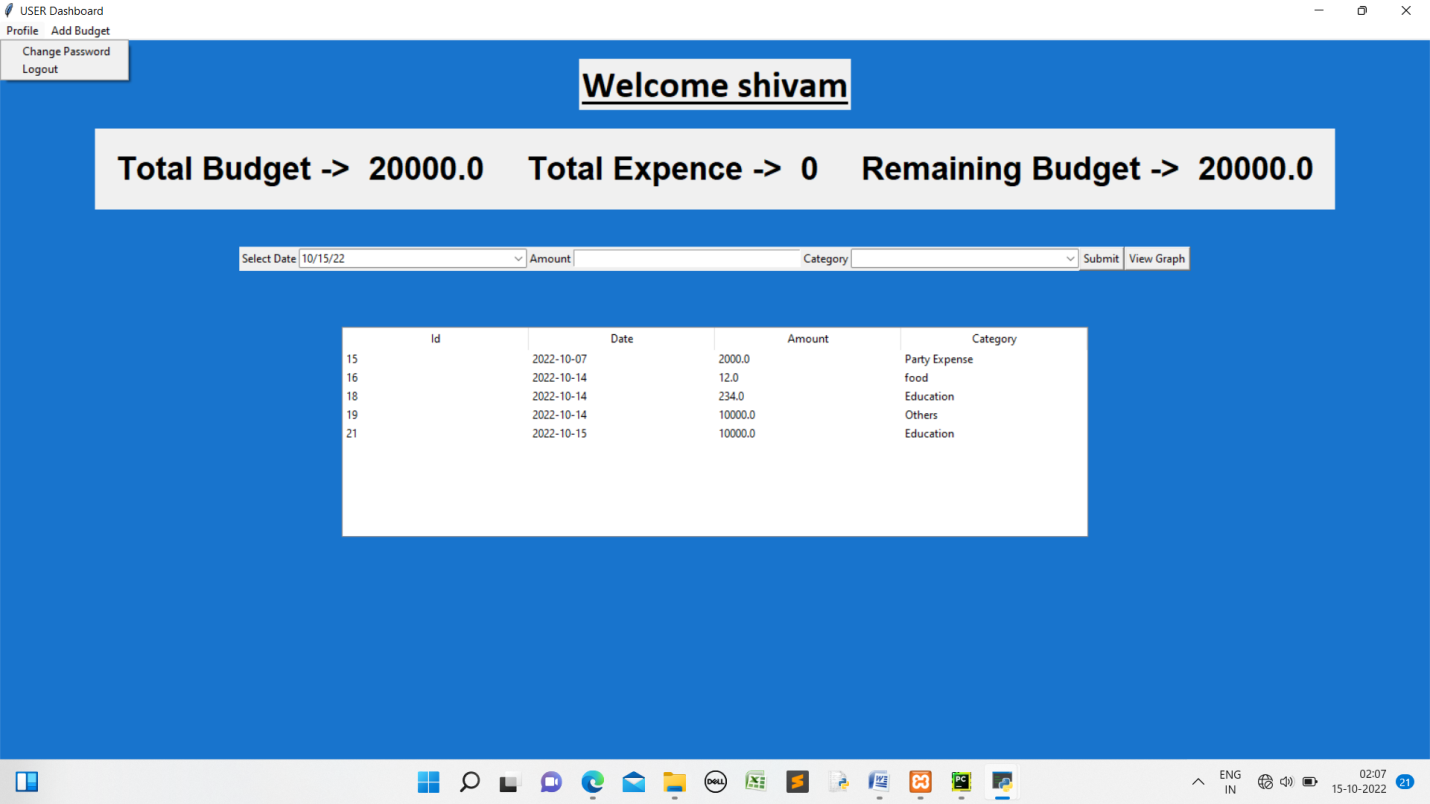
Admin Add category Page



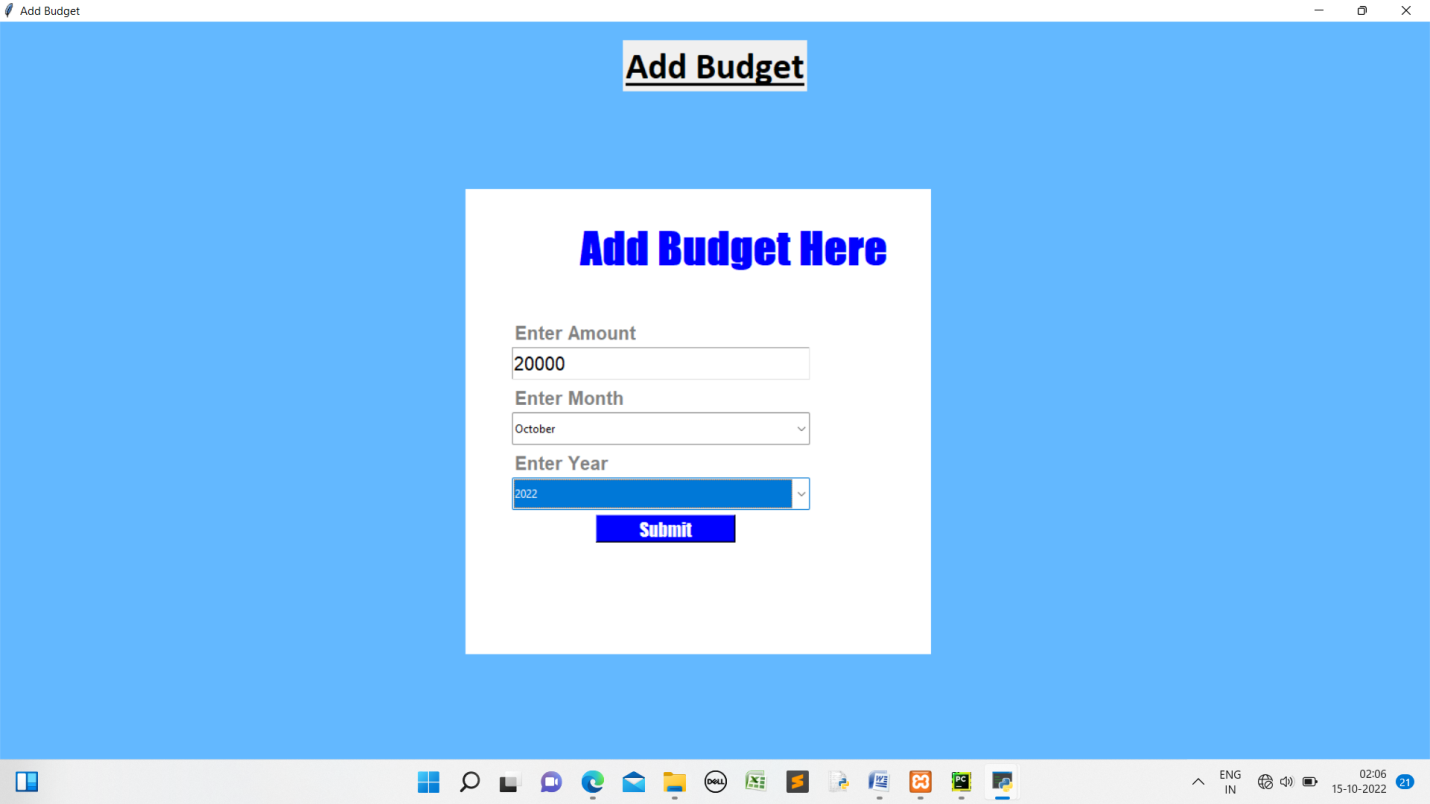
Admin Change Password Page



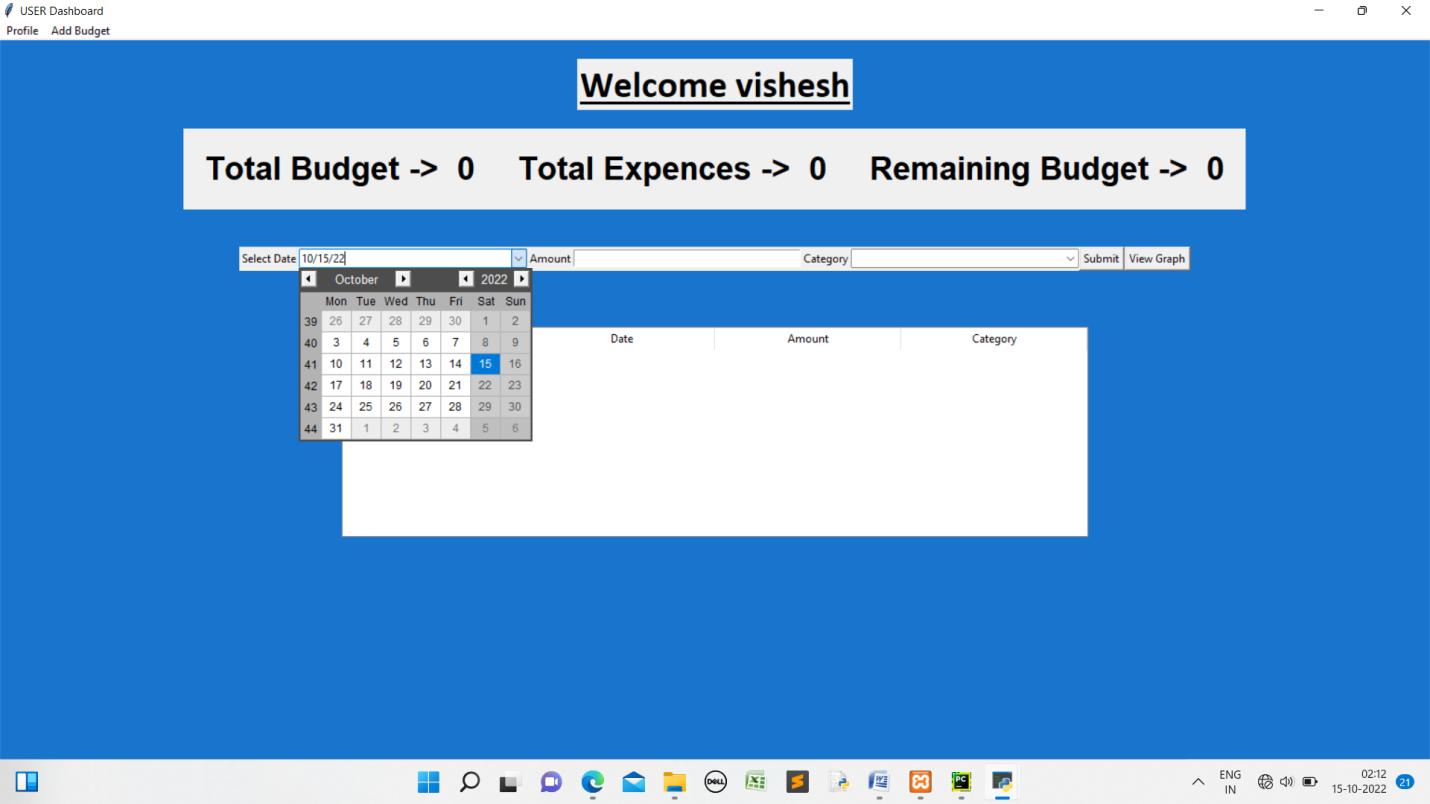
User Register Page



User Home Page



User Add Budget Page

****

User Calender view

**TESTING**

**Testing** is a method to check whether the actual software product matches expected requirements and to ensure that software product is[Defect](https://www.guru99.com/defect-management-process.html)free. **We have done many testes , but it is not possible to write all of them.Therefore we have written only two . These are**

**1.Test case for login page**

**2. Test case for add budget**

Test Case for Login Page

|  |  |  |  |
| --- | --- | --- | --- |
| Test Scenario id | Login-A | Test Case id | Login-1.0 |
| Test Case Description | Login-Positive and Negative Test Case | Test Priority | High |
| Pre-Requisite | A Valid User Account | Post-Requisite | N/A |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| S.No | ACTION | INPUTS | EXPECTED OUTPUT | ACTUAL OUTPUT | TEST BROWSER | TEST RESULT | TEST COMMENTS |
| 1. | Launch Application | Open Application | Expense Manager Home Page | Expense Manager Home Page | Windows Operating System | Pass | [ Gourab Sharma  2/11/2022]:  Launch Successful |
| 2. | Enter correct username and Password and hit Login button | Username  System  Password  \*\*\*\* | Login Success | Login Success | Windows Operating System | Pass | [Gourab Sharma  2/11/2022]:  Login Successful |
| 3. | Enter invalid username and Password and hit Login button | Username  Test  Password  \*\*\* | Entered Username or Password is invalid or create new account | Entered Username or Password is invalid or create new account | Windows Operating System | Pass | [Gourab Sharma  2/11/2022]:  Login UnSuccessful |
| 4. | Enter invalid username and valid Password and hit Login button | Username  Test  Password  \*\*\*\* | Entered Username is invalid or create new account | Entered Username is invalid or create new account | Windows Operating System | Pass | [Gourab Sharma  2/11/2022]:  Login UnSuccessful |
| 5. | Enter valid username and invalid Password and hit Login button | Username  System  Password  \*\*\*\*\* | Entered Password is invalid or click on Forget Password | Entered Password is invalid or click on Forget Password | Windows Operating System | Pass | [Gourab Sharma  2/11/2022]:  Login UnSuccessful |

Tab-2.1 LoginPage Test case

Test Case For Add Budget

|  |  |  |  |
| --- | --- | --- | --- |
| Test Scenario id | Budget-A | Test Case id | Bud-1.0 |
| Test Case Description | Budget-Added and Notadded Test Case | Test Priority | High |
| Pre-Requisite | Already Login | Post-Requisite | N/A |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| S.No | ACTION | INPUTS | EXPECTED OUTPUT | ACTUAL OUTPUT | TEST BROWSER | TEST RESULT | TEST COMMENTS |
| 1. | Open Budget Page | Click on Add Budget | Open Succesfully | Open Successfully | Windows Operating System | Pass | [ Gourab Sharma  2/11/2022]:  Launch Successful |
| 2. | Add amount,  Month and  year | Amount  20000  Month  october  Year 2020 | Added  Successfully | Added Successfully | Windows Operating System | Pass | [Gourab Sharma  2/11/2022]:  Add Successful |
| 3. | Add already added amount,  Month and  year | Amount  20000  Month  october  Year 2020 | Show messagebox Budget already added | Show messagebox Budget already added | Windows Operating System | Pass | [Gourab Sharma  2/11/2022]:  Add  UnSuccessful |
| 4. | Leave one field empty | Amount  Month  october  Year 2020 | Show messagebox Add all fields | Show messagebox Add all fields | Windows Operating System | Pass | [Gourab Sharma  2/11/2022]:  Add  UnSuccessful |

Tab-2.2 Add Budget Test case

**CONCLUSION**

Monitoring your everyday expenses can set aside you cash, yet it can likewise help you set your monetary objectives for what’s to come. On the off chance that you know precisely where your sum is going much of a stretch see where a few reductions and bargains can be made. Expense Tracker project is for keeping our day-to-day expenditures will helps us to keep record of our money daily. The project what we have created is work more proficient than the other income and expense tracker. The project effectively keeps away from the manual figuring for trying not to ascertain the pay and cost each month. It’s a user-friendly application.

Expense Manager as a tool is really helpful for tracking the expenses which we spent on our daily basis work. This project is really a great project but due to time constraint we are not able to make it a well developed project but in future we can add more things in this project and make it more useful for its users. The future scope of this expense manager are as follows:-

1. We can make this available on internet
2. We can add the option of filtering in this
3. We allow users to chat with other
4. Automatically it will keep on sending notifications for our daily expenditure

In nutshell , we can say that this report discussed the methodology used for our industrial training and the hardware and software requirements required. Lastly the diagrams included are content diagram, data flow diagram and entity relationship diagram. These diagrams are included for a better understanding of the flow of the system .

**Biblography & References:--**

**For Tkinter:-**

[https://www.bing.com/search?q=tkinter&qs=n&form=QBRE&sp=-1&pq=tkinter&sc=10-7&sk=&cvid=3B5D0FBD5A6D46848F4BC3176FE18BE1&ghsh=0&ghacc=0&ghpl=#](https://www.bing.com/search?q=tkinter&qs=n&form=QBRE&sp=-1&pq=tkinter&sc=10-7&sk=&cvid=3B5D0FBD5A6D46848F4BC3176FE18BE1&ghsh=0&ghacc=0&ghpl=)

**For MYSQl:-**

[https://www.bing.com/search?q=mysql&qs=n&form=QBRE&sp=-1&pq=msql&sc=10-4&sk=&cvid=9197CEDD62F148C59DC4F6836D318BE4&ghsh=0&ghacc=0&ghpl=#](https://www.bing.com/search?q=mysql&qs=n&form=QBRE&sp=-1&pq=msql&sc=10-4&sk=&cvid=9197CEDD62F148C59DC4F6836D318BE4&ghsh=0&ghacc=0&ghpl=)

**For tkcalendar:-**

[https://www.bing.com/search?q=tkcalendar&qs=n&form=QBRE&sp=-1&pq=tkcalendar&sc=10-10&sk=&cvid=D17F3E1A550E40E7B3E9DC091A5CBA4E&ghsh=0&ghacc=0&ghpl=#](https://www.bing.com/search?q=tkcalendar&qs=n&form=QBRE&sp=-1&pq=tkcalendar&sc=10-10&sk=&cvid=D17F3E1A550E40E7B3E9DC091A5CBA4E&ghsh=0&ghacc=0&ghpl=)

**For Pandas:--**

[https://www.bing.com/search?q=pandas&qs=n&form=QBRE&sp=-1&pq=pa&sc=10-2&sk=&cvid=8F6D2B3E4A6240B08BD0239C6A8DD798&ghsh=0&ghacc=0&ghpl=#](https://www.bing.com/search?q=pandas&qs=n&form=QBRE&sp=-1&pq=pa&sc=10-2&sk=&cvid=8F6D2B3E4A6240B08BD0239C6A8DD798&ghsh=0&ghacc=0&ghpl=)

**Book Provided By VMM Institute**

**SOME LECTURES OF SAURABH SHUKLA**

**APPENDIX**

**Main.py**

from tkinter import \*  
import ADMINLOGIN  
import userloginsignup  
  
class first:  
 def \_\_init\_\_(self):  
 self.one=Tk()  
 self.one.title('Expense Manager')  
 self.one.state('zoomed')  
 self.one.config(bg="blue")  
  
 self.bg = PhotoImage(file=r"C:\Users\GOURAB SHARMA\Downloads\WhatsApp Image 2022-10-01 at 11.47.28 (1).png")  
 self.bg\_image = Label(self.one, bg="white", image=self.bg).place(x=400, y=250)  
  
 self.mainLabel = Label(self.one, text="WELCOME HERE", font=("calibri", 30, 'bold'))  
 self.mainLabel.pack()  
 self.mainLabel1 = Label(self.one, text="EXPENSE MANAGER KEEP RECORD OF UR EXPENSES",fg="blue", font=("calibri", 30, 'bold'))  
 self.mainLabel1.pack(pady=20)  
  
 self.mainlabel2=Label(self.one,text="SELECT UR ROLE:", font=("calibri", 30, 'bold'))  
 self.mainlabel2.pack(pady=20)  
  
 frame\_login = Frame(self.one, bg="white")  
 frame\_login.pack(pady=50,padx=20)  
  
 self.adminloginm = Button(frame\_login, text="ADMIN", font=('calibri', 20),fg="blue",width=10,  
 command=self.adminlogin)  
 self.adminloginm.grid(row=0, column=1,padx=150)  
  
 self.userloginm = Button(frame\_login, text="USER", font=('calibri', 20),fg="blue",width=10,  
 command=self.userlogin)  
 self.userloginm.grid(row=0, column=0,padx=20)  
   
 self.one.mainloop()  
 def adminlogin(self):  
 ADMINLOGIN.main()  
  
  
  
 def userlogin(self):  
 userloginsignup.user()  
  
   
first()

**AdminLogin.py**

from tkinter import \*  
import tkinter.messagebox as msg  
from connect import connect  
import admindashboard  
from PIL import ImageTk,Image  
  
class main:  
 def \_\_init\_\_(self):  
 self.root = Tk()  
 self.root.title("Admin Login")  
 self.root.state('zoomed')  
 self.root.config(bg="DodgerBlue2")  
  
 self.mainLabel = Label(self.root, text="Admin Login", font=('calibri', 30, 'bold', 'underline'))  
 self.mainLabel.pack(pady=20)  
 frame\_login = Frame(self.root, bg="white", width=500, height=400)  
 frame\_login.place(x=500, y=200)  
  
 title = Label(frame\_login, text="Login Here", font=("Impact", 35, "bold"), fg="blue", bg="white").place(x=120,y=30)  
 subtitle = Label(frame\_login, text="Admin Login Area", font=("Gaudy old style", 15, "bold"), fg="black", bg="white").place(x=120,y=100)  
 self.lbl\_username = Label(frame\_login, text="Enter email", font=("Gaudy old style", 15, "bold"), fg="grey", bg="white").place(x=50, y=160)  
 self.txt1 = Entry(frame\_login, font=("Gaudy old style", 15), bg="white")  
 self.txt1.place(x=50, y=190, width=320, height=35)  
 self.lbl\_password = Label(frame\_login, text="Password", font=("Gaudy old style", 15, "bold"), fg="grey", bg="white").place(x=50, y=250)  
 self.txt2 = Entry(frame\_login, show='\*', font=("Gaudy old style", 15), bg="white")  
 self.txt2.place(x=50, y=280, width=320, height=35)  
 button = Button(frame\_login, text="Submit", command=self.checkAdmin, font=("Impact", 15), fg="white", cursor='hand2', bg="blue").place(x=50, y=330, width=150, height=30)  
  
  
 self.root.mainloop()  
  
 def checkAdmin(self):  
 self.email = self.txt1.get()  
 self.password = self.txt2.get()  
 if self.email=="" or self.password=="":  
 msg.showerror("Error","All Fields Are Required",parent=self.root)  
 else:  
  
 conn = connect()  
 cr = conn.cursor()  
 q1 = f"select \* from admin where email='{self.email}' and password='{self.password}'"  
 cr.execute(q1)  
 result = cr.fetchone()  
 if result is None:  
 msg.showerror("Error","Invalid Email/Password",parent=self.root)  
 else:  
 msg.showinfo("","Login Successful")  
 self.root.destroy()  
 admindashboard.dashboard()

**Admindashboard.py**

from tkinter import \*  
import tkinter.messagebox as msg  
from connect import connect  
import tkinter as tk  
import tkinter.ttk as ttk  
import pandas as pd  
import changepassword  
  
class dashboard:  
 def \_\_init\_\_(self):  
 self.root1 = Tk()  
 self.root1.title("Admin Dashboard")  
 self.root1.state('zoomed')  
  
 self.root1.config(bg="Dodgerblue3")  
  
 self.rootMenu = Menu(self.root1)  
 self.root1.config(menu=self.rootMenu)  
  
 self.adminMenu = Menu(self.rootMenu, tearoff=0)  
 self.rootMenu.add\_cascade(label="Manage Admins", menu=self.adminMenu)  
 self.adminMenu.add\_command(label="Add Admin",command=self.newadmin)  
 self.adminMenu.add\_command(label="View Admin",command=self.viewadmin)  
  
 self.catMenu = Menu(self.rootMenu, tearoff=0)  
 self.rootMenu.add\_cascade(label="Manage Category", menu=self.catMenu)  
 self.catMenu.add\_command(label="Add Category",command=self.addcat)  
 self.catMenu.add\_command(label="View Category",command=self.viewcat)  
  
 self.profileMenu = Menu(self.rootMenu, tearoff=0)  
 self.rootMenu.add\_cascade(label="Profile", menu=self.profileMenu)  
 self.profileMenu.add\_command(label="Change Password",command=changepassword.change)  
 self.profileMenu.add\_command(label="Logout", command=lambda :self.root1.destroy())  
  
 self.mainLabel = Label(self.root1, text="Welcome Admin", font=('Gaudy old style', 30, 'bold', 'underline'))  
 self.mainLabel.grid(row=1,column=3,pady=100)  
  
 self.btnin = Button(self.root1, text="OPERATION ON USER", font=('Gaudy old style', 14), fg="blue",command=self.op\_user)  
 self.btnin.grid(row=7, column=3, pady=30)  
  
 self.mainLabelA = Label(self.root1, text="RULES FOR ADMIN", font=('Gaudy old style', 15, 'bold'), fg="blue")  
 self.mainLabelA.grid(row=2, column=2)  
 self.mainLabelB= Label(self.root1, text="1.YOU MUST HAVE TO HELP USER WHERE POSSIBLE", font=('Gaudy old style', 14))  
 self.mainLabelB.grid(row=3, column=2)  
 self.mainLabelC = Label(self.root1, text="2.DONOT MISUSE UR RIGHTS", font=('Gaudy old style', 14))  
 self.mainLabelC.grid(row=4, column=2)  
 self.mainLabelD = Label(self.root1, text="3.IF ANY PROBLEM FOUND IN UR ACCOUNT U MAY REMOVE", font=('Gaudy old style', 14))  
 self.mainLabelD.grid(row=5, column=2)  
  
 def op\_user(self):  
 self.userop = Toplevel()  
  
 self.userop.geometry("1200x1200")  
 self.userop.title("Operation on user")  
 self.userop.config(bg="steelblue1")  
 self.key = StringVar()  
 self.namea=StringVar()  
 self.emaila=StringVar()  
 self.mobilea = StringVar()  
 self.passworda = StringVar()  
  
  
 self.label1 = Label(self.userop, text="Fetch Details of user by adding userid", fg="blue",font=('calibri', 30, 'bold', 'underline'))  
 self.label1.grid(row=2,column=2,pady=20)  
 self.label2 = Label(self.userop ,text="Enter Userid", font=('calibri', 14))  
 self.label2.grid(row=3, column=1, pady=10, padx=10)  
 self.ent2 = Entry(self.userop, width=50 ,textvariable=self.key , font=('calibri', 14))  
 self.ent2.grid(row=3, column=2, pady=10, padx=10)  
 self.btnf = Button(self.userop, text="Fetch Details",fg="blue", font=('calibri', 14),command=self.fetch\_user)  
 self.btnf.grid(row=4,column=2,pady=20)  
 self.label3 = Label(self.userop, text="User Name", font=('calibri', 14))  
 self.label3.grid(row=5, column=1, pady=10, padx=10)  
 self.ent3 = Entry(self.userop, width=50, textvariable=self.namea,font=('calibri', 14))  
 self.ent3.grid(row=5, column=2, pady=10, padx=10)  
 self.label4 = Label(self.userop, text="Email" ,font=('calibri', 14))  
 self.label4.grid(row=6, column=1, pady=10, padx=10)  
 self.ent4 = Entry(self.userop, width=50,textvariable=self.emaila, font=('calibri', 14))  
 self.ent4.grid(row=6, column=2, pady=10, padx=10)  
 self.label5 = Label(self.userop, text="Mobile", font=('calibri', 14))  
 self.label5.grid(row=7, column=1, pady=10, padx=10)  
 self.ent5 = Entry(self.userop, width=50,textvariable=self.mobilea, font=('calibri', 14))  
 self.ent5.grid(row=7, column=2, pady=10, padx=10)  
 self.btnu = Button(self.userop, text="Update User", fg="blue",font=('calibri', 14),command=self.update\_user)  
 self.btnu.grid(row=9, column=2, pady=20)  
 self.btnv = Button(self.userop, text="Show All User",fg="blue", font=('calibri', 14), command=self.show\_user)  
 self.btnv.grid(row=10, column=2, pady=20)  
  
 self.userop.mainloop()  
  
 def update\_user(self):  
 self.id = self.ent2.get()  
 self.name = self.namea.get()  
 self.email = self.emaila.get()  
 self.mobile = self.mobilea.get()  
 self.password = self.passworda.get()  
 if self.name == "" or self.email == "" or self.mobile=="" :  
 msg.showerror("ERROR", "ALL ARE REQUIRED", parent=self.userop)  
 else:  
 self.con = connect()  
 self.cr = self.con.cursor()  
  
 sql = f"update user set name= '{self.name}',email= '{self.email}',mobile = '{self.mobile}' where userid= '{self.id}'"  
 self.cr.execute(sql)  
 msg.showinfo("","User updated scuccessfully",parent=self.userop)  
 self.con.commit()  
  
  
  
 def fetch\_user(self):  
 self.id = self.ent2.get()  
 if self.id == "":  
 msg.showerror("ERROR", "FIELD ID IS REQUIRED", parent=self.userop)  
 else:  
 con = connect()  
 cr = con.cursor()  
 q = f"select userid,name ,email,mobile from user where userid='{self.id}'"  
 cr.execute(q)  
 var = cr.fetchone()  
 if var is None:  
 msg.showerror("", "Invalid User id", parent=self.userop)  
 else:  
 print(var)  
 self.namea.set(var[1])  
 self.emaila.set(var[2])  
 self.mobilea.set(var[3])  
 self.passworda.set(var[4])  
 con.commit()  
 con.close()  
  
  
  
 def delete\_user(self):  
 self.id = self.ent2.get()  
 if self.id=="":  
 msg.showerror("Error","Id Field is Required",parent=self.userop)  
 else:  
 con = connect()  
 cr = con.cursor()  
 cr.execute(f"DELETE from user where userid= '{self.id}'")  
 msg.showinfo("success", "user has been removed", parent=self.userop)  
 con.commit()  
 con.close()  
  
 def newadmin(self):  
 self.admin=Toplevel()  
  
 self.admin.geometry("800x800")  
 self.admin.title("Add admin")  
 self.admin.config(bg="dodger blue")  
 self.lb1 = Label(self.admin, text='Enter Name', font=('calibri', 14))  
 self.lb1.grid(row=0, column=0, pady=20, padx=10)  
 self.txt1 = Entry(self.admin, width=40, font=('calibri', 14))  
 self.txt1.grid(row=0, column=1, pady=20, padx=10)  
  
 self.lb2 = Label(self.admin, text='Enter Email', font=('calibri', 14))  
 self.lb2.grid(row=1, column=0, pady=20, padx=10)  
 self.txt2 = Entry(self.admin, width=40, font=('calibri', 14))  
 self.txt2.grid(row=1, column=1, pady=20, padx=10)  
  
 self.lb3 = Label(self.admin, text='Enter Role', font=('calibri', 14))  
 self.lb3.grid(row=2, column=0, pady=20, padx=10)  
 self.txt3 = ttk.Combobox(self.admin,values=['admin'], width=40, state='readonly')  
 self.txt3.grid(row=2, column=1, pady=20, padx=10)  
  
 self.lb4 = Label(self.admin, text='Enter Password', font=('calibri', 14))  
 self.lb4.grid(row=3, column=0, pady=20, padx=10)  
 self.txt4 = Entry(self.admin, width=40, font=('calibri', 14))  
 self.txt4.grid(row=3, column=1, pady=20, padx=10)  
  
  
 self.btn2 = Button(self.admin, text="SUBMIT",fg="blue", font=('calibri', 14), command=self.insertadmin)  
  
 self.btn2.grid(row=5,column=1,pady=10)  
 self.admin.mainloop()  
  
 def insertadmin(self):  
 name = self.txt1.get()  
 email = self.txt2.get()  
 role = self.txt3.get()  
 password = self.txt4.get()  
  
 if len(name) == 0 or len(email) == 0 or len(role) == 0 or len(password) == 0:  
 msg.showwarning("", "PLease input all details....",parent=self.admin)  
  
 else:  
 con = connect()  
 cur = con.cursor()  
 q = f"select \* from admin where email='{email}'"  
 cur.execute(q)  
 result = cur.fetchone()  
 print(result)  
 if result is None:  
 q1 = f"insert into admin values( '{name}','{email}','{password}','{role}')"  
 cur.execute(q1)  
 con.commit()  
 msg.showinfo('', "Admin added Successfully! Now Admin can Login",parent=self.root1)  
 else:  
 msg.showwarning('', 'Email already exists',parent=self.admin)  
  
  
 def viewadmin(self):  
 self.showadmin=Toplevel()  
 self.showadmin.geometry("800x800")  
 self.showadmin.title("view admin")  
 self.showadmin.config(bg="dodger blue")  
 con = connect()  
 cur = con.cursor()  
 t = Text(self.showadmin, height=200, width=500)  
 t.pack()  
 x = pd.read\_sql\_query("select name,email,role from admin order by email", con)  
 t.insert(tk.END, x)  
 t.config(state='disabled')  
  
 def addcat(self):  
 self.adminc = Toplevel()  
 # self.root.destroy()  
  
 self.adminc.geometry("800x800")  
 self.adminc.title("Add Category")  
 self.adminc.config(bg="dodger blue")  
 self.lb1 = Label(self.adminc, text='Enter Category', font=('calibri', 14))  
 self.lb1.grid(row=0, column=0, pady=20, padx=10)  
 self.txt1 = Entry(self.adminc, width=40, font=('calibri', 14))  
 self.txt1.grid(row=0, column=1, pady=20, padx=10)  
  
 self.lb2 = Label(self.adminc, text='Enter Description', font=('calibri', 14))  
 self.lb2.grid(row=1, column=0, pady=20, padx=10)  
 self.txt2 = Entry(self.adminc, width=40, font=('calibri', 14))  
 self.txt2.grid(row=1, column=1, pady=20, padx=10)  
  
 self.btn2 = Button(self.adminc, text="SUBMIT",fg="blue", font=('calibri', 14),  
 command=self.insertcat)  
  
 self.btn2.grid(row=5, column=1, pady=10)  
 self.admin.mainloop()  
  
  
  
 def insertcat(self):  
 category = self.txt1.get()  
 description = self.txt2.get()  
  
 if len(category) == 0 or len(description) == 0 :  
 msg.showwarning("Error", "PLease input all details....",parent=self.adminc)  
  
 else:  
 con = connect()  
 cur = con.cursor()  
 q = f"select \* from category where name='{category}'"  
 cur.execute(q)  
 result = cur.fetchone()  
 print(result)  
 if result is None:  
 q1 = f"insert into category values( '{category}','{description}')"  
 cur.execute(q1)  
 con.commit()  
 msg.showinfo('Error', "Category added Successfully!",parent=self.adminc)  
 else:  
 msg.showwarning('Error', 'Category/Description already exists',parent=self.adminc)  
  
  
  
 def viewcat(self):  
 self.showview = Toplevel()  
 self.showview.geometry("800x800")  
 self.showview.title("view admin")  
 self.showview.config(bg="dodger blue")  
 con = connect()  
 cur = con.cursor()  
 t = Text(self.showview, height=200, width=500)  
 t.pack()  
 x = pd.read\_sql\_query("select \* from category order by name", con)  
 t.insert(tk.END, x)  
 t.config(state='disabled')  
  
  
  
 def show\_user(self):  
 self.showuser = Toplevel()  
 self.showuser.geometry("800x800")  
 self.showuser.title("view admin")  
 con = connect()  
 cur = con.cursor()  
 t = Text(self.showuser, height=200, width=500)  
 t.pack()  
 x = pd.read\_sql\_query("select userid,name,email,mobile from user order by userid", con)  
 t.insert(tk.END, x)  
 t.config(state='disabled')

**Add-Admin.py**

from tkinter import \*  
import tkinter.messagebox as msg  
from connect import connect  
class add\_admin:  
 def \_\_init\_\_(self):  
 self.root = Tk()  
 self.root.title('ADD ADMIN')  
 self.root.geometry('800x800')  
  
 self.mainLabel = Label(self.root, text="Add New Admin", font=("calibri", 30, 'bold'))  
 self.mainLabel.pack(pady=20)  
 #self.loginFlag = False  
 #self.signupFlag = False  
 self.lb1 = Label(self.signupFrame, text='Enter Name', font=('calibri', 14))  
 self.lb1.grid(row=0, column=0, pady=20, padx=10)  
 self.txt1 = Entry(self.signupFrame, width=40, font=('calibri', 14))  
 self.txt1.grid(row=0, column=1, pady=20, padx=10)  
  
 self.lb2 = Label(self.signupFrame, text='Enter Email', font=('calibri', 14))  
 self.lb2.grid(row=1, column=0, pady=20, padx=10)  
 self.txt2 = Entry(self.signupFrame, width=40, font=('calibri', 14))  
 self.txt2.grid(row=1, column=1, pady=20, padx=10)  
  
 self.lb3 = Label(self.signupFrame, text='Enter Mobile', font=('calibri', 14))  
 self.lb3.grid(row=2, column=0, pady=20, padx=10)  
 self.txt3 = Entry(self.signupFrame, width=40, font=('calibri', 14))  
 self.txt3.grid(row=2, column=1, pady=20, padx=10)  
  
 self.lb4 = Label(self.signupFrame, text='Enter Password', font=('calibri', 14))  
 self.lb4.grid(row=3, column=0, pady=20, padx=10)  
 self.txt4 = Entry(self.signupFrame, width=40, font=('calibri', 14))  
 self.txt4.grid(row=3, column=1, pady=20, padx=10)  
  
  
 self.mainFrame = Frame()  
 self.mainFrame.pack()  
  
 self.btnFrame = Frame()  
  
 self.submitBtn = Button(self.btnFrame, text="SUBMIT", font=('calibri', 14))  
 #command=self.createLogin)  
 self.submitBtn.grid(row=0, column=0, padx=20)  
  
 self.root.mainloop()

**Connect.py**

import pymysql  
def connect():  
 con=pymysql.connect(host="127.0.0.1",  
 user='root',  
 password='',  
 database='expensemanagero')  
 return con

**Changepassword.py**

from tkinter import \*  
import tkinter.messagebox as msg  
from connect import connect  
import userloginsignup  
import admindashboard  
class change:  
 def \_\_init\_\_(self):  
 self.change=Tk()  
 self.change.geometry("1200x1200")  
 self.change.title("CHANGE PASSWORD")  
 self.change.config(bg="SkyBlue3")  
 self.titlechangea = Label(self.change, text="CHANGE UR PASSWORD",fg="blue", font=("calibri", 20, 'bold'))  
 self.titlechangea.grid(row=3, column=1,pady=20)  
 self.titlechange =Label(self.change,text="ENTER OLD PASSWORD", font=("calibri", 20, 'bold'))  
 self.titlechange.grid(row=4,column=0,pady=20)  
 self.titlechangee =Entry(self.change, font=("calibri", 20, 'bold'))  
 self.titlechangee.grid(row=4,column=1,pady=20)  
 self.titlechange1 = Label(self.change, text="ENTER NEW PASSWORD", font=("calibri", 20, 'bold'))  
 self.titlechange1.grid(row=5, column=0,pady=20)  
 self.titlechangee1 = Entry(self.change, font=("calibri", 20, 'bold'))  
 self.titlechangee1.grid(row=5, column=1,pady=20)  
 self.titlechange2 = Label(self.change, text="REENTER NEW PASSWORD", font=("calibri", 20, 'bold'))  
 self.titlechange2.grid(row=6, column=0,pady=20)  
 self.titlechangee2 = Entry(self.change, font=("calibri", 20, 'bold'))  
 self.titlechangee2.grid(row=6, column=1,pady=20)  
 self.mail = Label(self.change, text="Enter Email", font=("calibri", 20, 'bold'))  
 self.mail.grid(row=7, column=0, pady=20)  
 self.maile = Entry(self.change, font=("calibri", 20, 'bold'))  
 self.maile.grid(row=7, column=1, pady=20)  
  
 changebtn=Button(self.change,text="CHANGE",bg="blue", font=("calibri", 20, 'bold'),command=self.changep)  
 changebtn.grid(row=8,column=1,pady=20)  
 #self.password=userloginsignup.self.txt2  
 self.change.mainloop()  
  
 def changep(self):  
 self.old=self.titlechangee.get()  
 self.new=self.titlechangee1.get()  
 self.renter=self.titlechangee2.get()  
 self.email=self.maile.get()  
 if self.old=="" or self.new=="" or self.renter=="" or self.email=="":  
 msg.showwarning("","all fields are required",parent=self.change)  
 #if self.old == self.new:  
 # msg.showwarning("","new and old password cannot be same",parent=self.change)  
 if self.new!=self.renter:  
 msg.showwarning("","new and reenter new password must be same",parent=self.change)  
 else:  
  
 con = connect()  
 cur = con.cursor()  
 q = f"select \* from admin where email='{self.email}' and password='{self.old}' "  
 cur.execute(q)  
 result = cur.fetchone()  
 print(result)  
  
  
 if result is not None:  
  
 q = f"update admin set password='{self.new}' where email='{self.email}'"  
 cur.execute(q)  
 result1 = cur.fetchone()  
 print(result1)  
  
 con.commit()  
 # self.admin.destroy  
 msg.showinfo('', "Password updated Successfully! Now Admin can Login with new password", parent=self.change)  
 else:  
 msg.showwarning("","WRONG details",parent=self.change)

**UserLoginsignup.py**

from tkinter import \*  
import tkinter.messagebox as msg  
from connect import connect  
import userdashboard1  
from PIL import Image,ImageTk  
  
class user:  
 def \_\_init\_\_(self):  
 self.root = Tk()  
 self.root.title('User Login')  
 self.root.geometry('1200x1200')  
 self.root.config(bg="DodgerBlue2")  
  
 self.mainLabel = Label(self.root, text="User Login", font=("calibri", 30, 'bold'))  
 self.mainLabel.pack(pady=20)  
 self.loginFlag = False  
 self.signupFlag = False  
  
 self.conn = connect()  
 self.cr = self.conn.cursor()  
  
 self.mainFrame = Frame(self.root)  
 self.mainFrame.pack()  
  
 self.btnFrame = Frame(self.root,bg="DodgerBlue2")  
 self.btnFrame.pack(pady=20)  
  
 self.loginBtn = Button(self.btnFrame, text="LOGIN", font=("Impact", 15), cursor='hand2', bg="white",  
 command=self.createLogin)  
 self.loginBtn.grid(row=0, column=0, padx=20)  
  
 self.signupBtn = Button(self.btnFrame, text="REGISTER", font=("Impact", 15), cursor='hand2', bg="white",  
 command=self.createSignup)  
 self.signupBtn.grid(row=0, column=1, padx=20)  
 self.btnFrame.pack(pady=20)  
  
 self.createLogin()  
 self.root.mainloop()  
  
  
  
 def createLogin(self):  
 if self.loginFlag is False:  
 self.loginFlag = True  
  
 if self.signupFlag is True:  
 self.signupFlag = False  
 self.signupFrame.destroy()  
 #self.btn2.destroy()  
 #self.mainLabel.config(text="User Login",bg="blue")  
 #self.loginFrame1 = Frame(self.mainFrame)  
 self.frame\_login = Frame(self.root, bg="white", width=500, height=400)  
 self.frame\_login.place(x=350, y=200)  
  
 title = Label(self.frame\_login, text="Login Here", font=("Impact", 35, "bold"), fg="blue", bg="white").place(  
 x=120, y=30)  
 subtitle = Label(self.frame\_login, text="User Login Area", font=("Gaudy old style", 15, "bold"), fg="black",  
 bg="white").place(x=120, y=100)  
 self.lb1 = Label(self.frame\_login, text="Enter Email", font=("Gaudy old style", 15, "bold"), fg="grey",  
 bg="white").place(x=50, y=160)  
 self.txt1 = Entry(self.frame\_login, font=("Gaudy old style", 15), bg="white")  
 self.txt1.place(x=50, y=190, width=320, height=35)  
 self.lb2 = Label(self.frame\_login, text="Enter Password", font=("Gaudy old style", 15, "bold"), fg="grey",  
 bg="white").place(x=50, y=250)  
 self.txt2 = Entry(self.frame\_login, show='\*', font=("Gaudy old style", 15), bg="white")  
 self.txt2.place(x=50, y=280, width=320, height=35)  
 self.btn1 = Button(self.frame\_login, text="Submit", command=self.checkUser, font=("Impact", 15), fg="white",  
 cursor='hand2', bg="blue").place(x=50, y=330, width=150, height=30)  
  
  
  
 def createSignup(self):  
  
 if self.signupFlag is False:  
 if self.loginFlag is True:  
 self.loginFlag = False  
 self.signupFlag = True  
 self.frame\_login.destroy()  
 #self.btn1.destroy()  
  
 self.signupFrame = Frame(self.root, bg="white", width=500, height=500)  
 self.signupFrame.place(x=350, y=180)  
  
 title = Label(self.signupFrame, text="Register Here", font=("Impact", 35, "bold"), fg="blue",bg="white").place(x=120, y=30)  
  
 self.lb1 = Label(self.signupFrame, text="Enter Name", font=("Gaudy old style", 15, "bold"), fg="grey",bg="white").place(x=50, y=160)  
 self.txt1 = Entry(self.signupFrame, font=("Gaudy old style", 15), bg="white")  
 self.txt1.place(x=50, y=190, width=320, height=35)  
 self.lb2 = Label(self.signupFrame, text="Enter Email", font=("Gaudy old style", 15, "bold"), fg="grey",  
 bg="white").place(x=50, y=220)  
 self.txt2 = Entry(self.signupFrame, font=("Gaudy old style", 15), bg="white")  
 self.txt2.place(x=50, y=250, width=320, height=35)  
 self.lb3 = Label(self.signupFrame, text="Enter Mobile", font=("Gaudy old style", 15, "bold"), fg="grey", bg="white").place(x=50, y=290)  
 self.txt3 = Entry(self.signupFrame, font=("Gaudy old style", 15), bg="white")  
 self.txt3.place(x=50, y=320, width=340, height=35)  
 self.lb4 = Label(self.signupFrame, text="Enter Password", font=("Gaudy old style", 15, "bold"), fg="grey",bg="white").place(x=50, y=360)  
 self.txt4 = Entry(self.signupFrame, show='\*', font=("Gaudy old style", 15), bg="white")  
 self.txt4.place(x=50, y=390, width=320, height=35)  
 self.btn2 = Button(self.signupFrame, text="Submit", command=self.insertUser, font=("Impact", 15), fg="white",cursor='hand2', bg="blue").place(x=50, y=430, width=150, height=30)  
 """  
 #self.mainLabel.config(text="Register new User")  
 self.signupFrame = Frame(self.mainFrame)  
  
 self.lb1 = Label(self.signupFrame, text='Enter Name', font=('calibri', 14))  
 self.lb1.grid(row=0, column=0, pady=20, padx=10)  
 self.txt1 = Entry(self.signupFrame, width=40, font=('calibri', 14))  
 self.txt1.grid(row=0, column=1, pady=20, padx=10)  
  
 self.lb2 = Label(self.signupFrame, text='Enter Email', font=('calibri', 14))  
 self.lb2.grid(row=1, column=0, pady=20, padx=10)  
 self.txt2 = Entry(self.signupFrame, width=40, font=('calibri', 14))  
 self.txt2.grid(row=1, column=1, pady=20, padx=10)  
  
 self.lb3 = Label(self.signupFrame, text='Enter Mobile', font=('calibri', 14))  
 self.lb3.grid(row=2, column=0, pady=20, padx=10)  
 self.txt3 = Entry(self.signupFrame, width=40, font=('calibri', 14))  
 self.txt3.grid(row=2, column=1, pady=20, padx=10)  
  
 self.lb4 = Label(self.signupFrame, text='Enter Password', font=('calibri', 14))  
 self.lb4.grid(row=3, column=0, pady=20, padx=10)  
 self.txt4 = Entry(self.signupFrame, width=40, font=('calibri', 14))  
 self.txt4.grid(row=3, column=1, pady=20, padx=10)  
  
 self.signupFrame.pack()  
 self.btn2 = Button(self.mainFrame, text="SUBMIT", font=('calibri', 14),  
 command=self.insertUser)  
 self.btn2.pack(pady=10)  
 """  
  
 def checkUser(self):  
 email = self.txt1.get()  
 password = self.txt2.get()  
 if email=="" or password=="":  
 msg.showerror("Error","All Fields are requird",parent=self.root)  
 else:  
  
 q = f"select \* from user where email='{email}' and password='{password}'"  
 self.cr.execute(q)  
 result = self.cr.fetchone()  
 if result is None:  
 msg.showerror("","Invalid Email/Password",parent=self.root)  
 else:  
 msg.showinfo("", "Login Successful")  
 self.root.destroy()  
 userdashboard1.dashboard(result)  
  
  
 def insertUser(self):  
 name = self.txt1.get()  
 email = self.txt2.get()  
 mobile = self.txt3.get()  
 password = self.txt4.get()  
  
 if len(name) == 0 or len(email) == 0 or len(mobile) == 0 or len(password) == 0:  
 msg.showwarning("","PLease input all details....",parent=self.mainFrame)  
 else:  
 q = f"select \* from user where email='{email}' or mobile='{mobile}'"  
 self.cr.execute(q)  
 result = self.cr.fetchone()  
 if result is None:  
 q1 = f"insert into user values(NULL, '{name}','{email}','{mobile}','{password}')"  
 self.cr.execute(q1)  
 self.conn.commit()  
 msg.showinfo('',"User Registered Successfully! Please Login Now",parent=self.mainFrame)  
 self.createLogin()  
 else:  
 msg.showwarning('','Email/Mobile already exists',parent=self.mainFrame)

**Userdashboard1.py**

from tkinter import \*  
import tkinter.messagebox as msg  
from connect import connect  
import addbudget1  
import tkcalendar  
import tkinter.ttk as ttk  
import datetime  
import changepassuser  
import viewgraph  
  
  
class dashboard:  
 def \_\_init\_\_(self, user):  
 self.root = Tk()  
 self.root.title("USER Dashboard")  
 self.root.state('zoomed')  
 self.root.config(bg="Dodgerblue3")  
  
 self.rootMenu = Menu(self.root)  
 self.root.config(menu=self.rootMenu)  
  
 self.profileMenu = Menu(self.rootMenu, tearoff=0)  
 self.rootMenu.add\_cascade(label="Profile", menu=self.profileMenu)  
 self.profileMenu.add\_command(label="Change Password",command=changepassuser.change)  
 self.profileMenu.add\_command(label="Logout", command=lambda: self.root.destroy())  
  
 self.rootMenu.add\_command(label="Add Budget", command=lambda: addbudget1.main(userid=user[0]))  
  
 self.conn = connect()  
 self.cr = self.conn.cursor()  
  
 self.mainLabel = Label(self.root, text=f"Welcome {user[1]}", font=('calibri', 30, 'bold', 'underline'))  
 self.mainLabel.pack(pady=20)  
  
 self.f1 = Frame(self.root)  
 self.f1.pack()  
  
 self.lb11 = Label(self.f1, text=f"Total Budget -> {0}", font=('arial', 26, 'bold'))  
 self.lb11.grid(row=0, column=0, pady=20, padx=20)  
  
 self.lb12 = Label(self.f1, text=f"Total Expences -> {0}", font=('arial', 26, 'bold'))  
 self.lb12.grid(row=0, column=1, pady=20, padx=20)  
  
 self.lb13 = Label(self.f1, text=f"Remaining Budget -> {0}", font=('arial', 26, 'bold'))  
 self.lb13.grid(row=0, column=2, pady=20, padx=20)  
  
 self.frame = Frame(self.root)  
 self.frame.pack(pady=40)  
  
 self.lb1 = Label(self.frame, text="Select Date")  
 self.lb1.grid(row=0, column=0)  
  
 self.txt1 = tkcalendar.DateEntry(self.frame, width=37)  
 self.txt1.grid(row=0, column=1)  
  
 self.lb2 = Label(self.frame, text="Amount")  
 self.lb2.grid(row=0, column=2)  
  
 self.txt2 = Entry(self.frame, width=40)  
 self.txt2.grid(row=0, column=3)  
  
 cat = self.getCategories()  
 self.lb3 = Label(self.frame, text="Category")  
 self.lb3.grid(row=0, column=4)  
  
 self.txt3 = ttk.Combobox(self.frame, width=37, state='readonly', values=cat)  
 self.txt3.grid(row=0, column=5)  
  
 self.userid = user[0]  
  
 self.btn = Button(self.frame, text="Submit", command=self.addExpence)  
 self.btn.grid(row=0, column=6)  
 self.btn = Button(self.frame, text="View Graph", command=self.view)  
 self.btn.grid(row=0, column=7)  
 col = ["id", "date", 'amount', 'category']  
 self.tv = ttk.Treeview(self.root, columns=col)  
 self.tv.pack(pady=20)  
 for i in col:  
 self.tv.heading(i, text=i.capitalize())  
 self.tv['show'] = 'headings'  
 self.getValues()  
 self.getBudget()  
 self.root.mainloop()  
  
 def view(self):  
 a = self.userid  
 viewgraph.main(a)  
  
 def getValues(self):  
 q = f"select id,date,amount,category from expense where userid='{self.userid}'"  
 self.cr.execute(q)  
 result = self.cr.fetchall()  
  
 for j in self.tv.get\_children():  
 self.tv.delete(j)  
  
 for i in range(0, len(result)):  
 self.tv.insert("",index=i,values=list(result[i]))  
  
  
 def addExpence(self):  
 self.date = self.changeDateFormat(self.txt1.get())  
 self.amount = self.txt2.get()  
 self.category = self.txt3.get()  
 option = True  
 if float(self.amount) > self.remainingBudget:  
 option = msg.askyesno("", "Your Budget has been Exceeded! Still want to add new Expence?",parent=self.root)  
 if len(self.date) == 0 or len(self.amount) == 0 or len(self.category) == 0:  
 msg.showwarning("", "PLease input all fields", parent=self.root)  
 else:  
 if option is True:  
 q = f"insert into expense values(NULL, '{self.userid}', '{self.date}','{self.amount}','{self.category}')"  
 self.cr.execute(q)  
 self.conn.commit()  
 msg.showinfo("", "Expence Added Successfully",parent=self.root)  
 self.getBudget()  
 self.getValues()  
 else:  
 msg.showinfo("", "Your Budget Remains Same",parent=self.root)  
 self.txt2.delete(0, "end")  
 self.txt3.set('')  
  
 def getBudget(self):  
 month = datetime.datetime.now().month  
 year = datetime.datetime.now().year  
 d = {1: "January", 2: "Febuary", 3: "March", 4: 'April', 5: "May", 6: 'June',  
 7: 'July', 8: 'August', 9: 'September', 10: 'October', 11: 'November',  
 12: 'December'}  
 self.conn = connect()  
 self.cr = self.conn.cursor()  
  
 q = f"select amount from budget where userid='{self.userid}' and month ='{d[month]}'" \  
 f"and year='{year}'"  
 self.cr.execute(q)  
 result = self.cr.fetchone()  
 print(result)  
 self.totalBudget = result[0]  
 newMonth = ""  
 if len(str(month)) == 1:  
 newMonth = f"0{month}"  
 q1 = f"select date,amount from expense where userid='{self.userid}'"  
 self.cr.execute(q1)  
 alldata = self.cr.fetchall()  
  
 print(alldata)  
 self.totalExpence = 0  
 for i in alldata:  
 self.totalExpence = self.totalExpence +i[1]  
  
 print(self.totalExpence)  
  
 self.remainingBudget = self.totalBudget - self.totalExpence  
 self.lb11.config(text=f"Total Budget -> {self.totalBudget}")  
 self.lb12.config(text=f"Total Expence -> {self.totalExpence}")  
 self.lb13.config(text=f"Remaining Budget -> {self.remainingBudget}")  
  
  
 def getCategories(self):  
 q = "select \* from category"  
 self.cr.execute(q)  
 result = self.cr.fetchall()  
 # print(result)  
 data = []  
 for i in result:  
 data.append(i[0])  
 return data  
  
  
 def changeDateFormat(self, date):  
 date\_lst = str(date).split('/')  
 new\_date\_st = ""  
 new\_date\_st = new\_date\_st + "20" + date\_lst[2] + "-"  
  
 if len(date\_lst[0]) == 1:  
 new\_date\_st = new\_date\_st + "0" + date\_lst[0] + "-"  
 else:  
 new\_date\_st = new\_date\_st + date\_lst[0] + "-"  
  
 if len(date\_lst[1]) == 1:  
 new\_date\_st = new\_date\_st + "0" + date\_lst[1]  
 else:  
 new\_date\_st = new\_date\_st + date\_lst[1] + "-"  
 # print(new\_date\_st)  
 return new\_date\_st

**AddBudget.py**

from tkinter import \*  
import tkinter.messagebox as msg  
import tkinter.ttk as ttk  
from connect import connect  
  
class main:  
 def \_\_init\_\_(self, userid):  
 self.root = Tk()  
 self.root.title("Add Budget")  
 self.root.state('zoomed')  
 self.root.config(bg="SteelBlue1")  
  
 self.mainLabel = Label(self.root, text="Add Budget", font=('calibri', 30, 'bold', 'underline'))  
 self.mainLabel.pack(pady=20)  
 self.userid = userid  
 self.Frame1 = Frame(self.root, bg="white", width=500, height=500)  
 self.Frame1.place(x=500, y=180)  
  
 title = Label(self.Frame1, text="Add Budget Here", font=("Impact", 35, "bold"), fg="blue", bg="white").place(  
 x=120, y=30)  
  
 self.lb1 = Label(self.Frame1, text="Enter Amount", font=("Gaudy old style", 15, "bold"), fg="grey",  
 bg="white").place(x=50, y=160)  
 self.txt1 = Entry(self.Frame1, font=("Gaudy old style", 15), bg="white")  
 self.txt1.place(x=50, y=190, width=320, height=35)  
 self.lb2 = Label(self.Frame1, text="Enter Month", font=("Gaudy old style", 15, "bold"), fg="grey", bg="white").place(x=50, y=220)  
 months = ['January', 'Febuary', 'March', "April", "May", "June", "July", "August", "September", "October", "November", "December"]  
 self.txt2 = ttk.Combobox(self.Frame1, values=months, state='readonly', width=50)  
 self.txt2.place(x=50, y=250, width=320, height=35)  
 self.lb3 = Label(self.Frame1, text="Enter Year", font=("Gaudy old style", 15, "bold"), fg="grey", bg="white").place(x=50, y=290)  
 year = ['2020', '2021', '2022', '2023', '2024', '2025', '2026', '2027', '2028', '2029', '2030']  
 self.txt3 = ttk.Combobox(self.Frame1, values=year, state='readonly', width=50)  
 self.txt3.place(x=50, y=320, width=320, height=35)  
 self.btnb = Button(self.Frame1, text="Submit", command=self.addBudget, font=("Impact", 15), fg="white", cursor='hand2', bg="blue").place(x=50, y=380, width=150, height=30)  
 self.root.mainloop()  
  
 def addBudget(self):  
 self.amount = self.txt1.get()  
 self.month = self.txt2.get()  
 self.year = self.txt3.get()  
  
 if len(self.amount) == 0 or len(self.month) == 0 or len(self.year) == 0:  
 msg.showwarning("","Please Input all Fields",parent=self.root)  
 else:  
 conn = connect()  
 cr = conn.cursor()  
 q1 = f"select \* from budget where userid='{self.userid}' and month='{self.month}' and year='{self.year}'"  
 cr.execute(q1)  
 result = cr.fetchone()  
 if result is None:  
 q = f"insert into budget values(NULL,'{self.amount}','{self.month}','{self.year}','{self.userid}')"  
 cr.execute(q)  
 conn.commit()  
 msg.showinfo("","Budget addded successfully",parent=self.root)  
 else:  
 msg.showwarning("","You have already added a budget for this month",parent=self.root)

**Viewgraph.py**

import datetime  
from tkinter import \*  
import tkinter.ttk as ttk  
from connect import connect  
from matplotlib.figure import Figure  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk  
  
  
class main:  
 def \_\_init\_\_(self, userid):  
 self.root = Tk()  
 self.root.title("Graph view")  
 self.root.geometry('800x800')  
 self.root.config(bg="SteelBlue1")  
 self.userid = userid  
 self.mainLabel = Label(self.root, text="View Expense Graphs", font=('calibri', 30, 'bold', 'underline'))  
 self.mainLabel.pack(pady=20)  
  
 self.comb = ttk.Combobox(self.root, values=("Monthly Expenses",),  
 state='readonly', font=('arial', 12))  
 self.comb.pack(pady=10)  
 self.comb.set('Monthly Expenses')  
 self.comb.bind("<<ComboboxSelected>>", self.plotChart)  
  
 self.frame = Frame(self.root)  
  
  
 self.frame.pack()  
 self.plotMonthlyChart()  
  
 self.root.mainloop()  
  
 def plotChart(self, event):  
 self.choice = self.comb.get()  
 if self.choice == 'Monthly Expenses':  
 self.plotMonthlyChart()  
 else:  
 pass  
  
 def plotMonthlyChart(self):  
 month = str(datetime.datetime.now().month)  
 year = datetime.datetime.now().year  
 month = str(month)  
 if len(month) == 1:  
 month = f"0{month}"  
 self.conn = connect()  
 self.cr = self.conn.cursor()  
 q1 = f"select date,amount, category from expense where userid='{self.userid}'"  
 self.cr.execute(q1)  
 alldata = self.cr.fetchall()  
  
 print(alldata)  
 self.categories = {}  
 for i in alldata:  
 if str(i[0])[0:7] == f"{year}-{month}":  
 if i[2] in self.categories:  
 self.categories[i[2]] += i[1]  
 else:  
 self.categories[i[2]] = i[1]  
  
 f = Figure(figsize=(5, 5), dpi=100)  
 a = f.add\_subplot(111)  
 a.pie(x=list(self.categories.values()), labels=list(self.categories.keys()))  
  
 canvas = FigureCanvasTkAgg(figure=f, master=self.frame)  
 canvas.get\_tk\_widget().pack()  
  
 def plotGraph(self):  
 f = Figure(figsize=(5,5), dpi=100)  
 a = f.add\_subplot(111)  
 a.plot([1,2,34,5],[3,4,5,6])  
  
 canvas = FigureCanvasTkAgg(figure=f, master=self.frame)  
 canvas.get\_tk\_widget().pack()

**Changepassuser.py**

from userdashboard1 import \*  
class change:  
 def \_\_init\_\_(self):  
 self.change=Tk()  
 self.change.geometry("1200x1200")  
 self.change.title("CHANGE PASSWORD")  
 self.change.config(bg="SkyBlue3")  
 self.titlechangea = Label(self.change, text="CHANGE UR PASSWORD", font=("calibri", 20, 'bold'))  
 self.titlechangea.grid(row=3, column=1,pady=20)  
 self.titlechange =Label(self.change,text="ENTER OLD PASSWORD", font=("calibri", 20, 'bold'))  
 self.titlechange.grid(row=4,column=0,pady=20)  
 self.titlechangee =Entry(self.change, font=("calibri", 20, 'bold'))  
 self.titlechangee.grid(row=4,column=1,pady=20)  
 self.titlechange1 = Label(self.change, text="ENTER NEW PASSWORD", font=("calibri", 20, 'bold'))  
 self.titlechange1.grid(row=5, column=0,pady=20)  
 self.titlechangee1 = Entry(self.change, font=("calibri", 20, 'bold'))  
 self.titlechangee1.grid(row=5, column=1,pady=20)  
 self.titlechange2 = Label(self.change, text="REENTER NEW PASSWORD", font=("calibri", 20, 'bold'))  
 self.titlechange2.grid(row=6, column=0,pady=20)  
 self.titlechangee2 = Entry(self.change, font=("calibri", 20, 'bold'))  
 self.titlechangee2.grid(row=6, column=1,pady=20)  
 self.mail = Label(self.change, text="ENTER EMAIL", font=("calibri", 20, 'bold'))  
 self.mail.grid(row=7, column=0, pady=20)  
 self.maile = Entry(self.change, font=("calibri", 20, 'bold'))  
 self.maile.grid(row=7, column=1, pady=20)  
  
 changebtn=Button(self.change,text="CHANGE", font=("calibri", 20,'bold'), command = self.changep)  
 changebtn.grid(row=8,column=1,pady=20)  
 #self.password=userloginsignup.self.txt2  
 self.change.mainloop()  
  
  
 def changep(self):  
 self.old=self.titlechangee.get()  
 self.new=self.titlechangee1.get()  
 self.renter=self.titlechangee2.get()  
 self.email=self.maile.get()  
  
 if self.old=="" or self.new=="" or self.renter=="" or self.email=="":  
 msg.showwarning("","all fields are required",parent=self.change)  
 #if self.old == self.new:  
 # msg.showwarning("","new and old password cannot be same",parent=self.change)  
 if self.new!=self.renter:  
 msg.showwarning("","new and reenter new password must be same",parent=self.change)  
 else:  
  
 con = connect()  
 cur = con.cursor()  
 q = f"select \* from user where email='{self.email}' and password='{self.old}' "  
 cur.execute(q)  
 result = cur.fetchone()  
 print(result)  
 if result is not None:  
 q = f"update user set password='{self.new}' where email='{self.email}'"  
 cur.execute(q)  
 result1 = cur.fetchone()  
 print(result1)  
  
 con.commit()  
 # self.admin.destroy  
 msg.showinfo('', "Password updated Successfully! Now user can Login with new password", parent=self.change)  
 else:  
 msg.showwarning("","WRONG details",parent=self.change)