PERSONAL EXPENSE TRACKER

PROJECT REPORT

(TEAM ID:PNT2022TMID03133)

Submitted by

NETHESVAR RATNAM (19EUEC090)

NETHEESH (19EUEC089)

NETHRA J (19EUEC091)

NISHIKANTH S(19EUEC092)

in partial fulfillment of the requirements for the award of the

degreeof

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY COIMBATORE

(An Autonomous Institution)

ANNA UNIVERSITY:CHENNAI

TABLE OF CONTENTS

1.INTRODUCTION:

1.1 PROJECT REVIEW:

When it comes to tracking expenses, you can make your system as simple as collecting receipts and organizing them once a month.

You might get a little more information from other expense tracking systems (listing them in a spreadsheet, using money management software or even choosing an online application), but all methods have one thing in common: you have to get in the habit of thinking about your expenses.

It’s very easy to misplace a receipt or forget about any cash you spent. You may even think that a cup of coffee or a trip to the vending machine isn’t worth tracking — although those little expenses can add up amazingly fast.

There are all sorts of opportunities to throw a kick into your plan to track expenses. You have to get in the habit of doing so, to reduce those lapses, and make sure that the data you’re basing financial decisions on is solid.

This project will request the clients to add their expenses and in view of their costs, wallet status will be refreshed which will be noticeable to the client.

- The user interacts with the application.
- Application will ask users to add their expenses and based on their expenses and based on their expenses wallet balance will be updated which will be visible to the user.
- Also, users can get an analysis of their expenditure in graphical forms.
- They have an opinion to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert.
- Setting up Application Environment
- Create Flask Project
- Work with IBM Cloud CLI, Docker CLI, Sendgrid
- Implementation of Web Application

- Create UI to Interact with the application
- Connect IBM DB2 with ython Integration of Sendgrid Service with Python
- Deployment if Cloud Application
- Containerize the application
- Upload Image in IBM Container directory
- Deploy on Kubernetes Cluster

1.2 PURPOSE:

- Help the people to track their expenses.
- Alert users when they exceed the limit of their budget.
- A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about financial management.

2. LITERATURE SURVEY:

2.1 EXISTING PROMBLEM:

- Lack of visual analytics for visual data.
- Lack of support for splitting up group expenses.
- Most of the applications are used only for personal use.
- Most of the applications does not incorporate shared group expenses.
- Efforts has to be made to include each and every transactions into the input field.

2.2 REFERENCES:

S.NO	TITLE	AUTHOR	YEAR	ABSTRACT	TECHNOLOGY
1.	EXPENSE MANAGER APPLICATION	Velmurugan A, Albert Mayan J, Niranjana P and Richard Francis	2020	This application is used to keep record of user personal expenses, his/her contribution in group expenditures, top investment options, view of the current stock market and grap the best ongoing offer in the market. It eliminate the sticky notes, spredsheets confussion and data handling inconsistency problems.	Android studio, Kotlin, java, SQLite, Android OS, Figma designing tool.
2.	EXPENSE TRACKER	Nidhi Jitendra Jadhav, Rutuja Vijay Chakor , Trupti Mahesh Gunjal , Damayanti. D. Pawar	2022	This system takes the user's income and divides it into daily expense allowances. If you exceed that day's expense, it will be deducted from your income and replaced with a new daily expense allowance. If the amount is smaller, it will be saved. At the end of the month, the daily spending tracking system will provide a report that shows the incomeexpenditure curve.	Mobile application, Using Database layer which holds all of the data and financial information, supported by User Interface.
3.	Daily Expense Tracker	Tamia Ruvimbo Masendu , Aanajey Mani Tripath	2022	Daily Expense Tracker is a gadget that being developed to help customers in budget planning. It offers end customers to file their earnings and costs within the finances that have been planned beforehand.	This application is a GUI (Graphics User Interface) based application.Technology used Java (Apache NetBeans IDE 13) and my MySQL Workbench.
4.	Expense Tracker Application	Velmurugan. R , Mrs. P. Usha	2021	This application allows the user to maintain a computerized diary. which will keep a track of Expenses of a user on a day to-day basis. This application keeps a record of your expenses and also will give you a category wise distribution of your expenses. It will generate report at the	Java, Xml, MySQL

				end of month to show Expense via a graphical representation.	

5.	EXPENDITURE MANAGEMENT SYSTEM	Dr. V. Geetha, G. Nikhitha, H. Sri Lasya, Dr. C.K.Gomathy	2022	This application uses Weekly Budget Planner to track their expenses. Automated message Alert is generated when they cross their budget. UPI linkup to track their online transactions. Weekly and Monthly Analysis are generated in the form of pie. chart. App Authentication for security of the user. Income. Expenses, and Wish List are the three data entry choices available to the user	JavaScript, JSX, React and Mangodb.
6.	A Case Study of Tracking Expenses by Commodity at Widget Farmers Cooperative	Dan Underwood	2011	Growing pressure from competition, shrinking markets, and poor economic conditions are making many agribusinesses look for ways to maximize profits and remain healthy. Widget Farmers Coop (WFC) is a large retail agricultural supply cooperative with 12 locations in two states. It has over 40 million dollars in annual sales each year since its creation in 2004. WFC management would like to track expenses and identify areas of the business that are profitable and capitalize on them, as well as identify areas that are not profitable and realign or eliminate them. Using Excel, the WFC regional accounting team designed Cost Allocation Tool 1 (CAT 1), a spreadsheet to allocate expenses based on product category both by site and as a whole for WFC.	Excel, designed Cost Allocation Tool 1 (CAT 1)
7.	Expense Tracker : A Smart Approach to Track Everyday Expense	Hrithik Gupta, Anant Prakash Singh, Navneet Kumar and J. Angelin Blessy	2020	Expense Tracker is a day-to-day expense management system designed to easily and efficiently track the daily expenses of unpaid and unpaid staff through a computerized system that eliminates the need for manual paper tasks that systematically maintains records and easily accesses data stored by the user.	Java (Apache Netbins 11.3) and MySQL Workbench 8.0 CE

8.	Expense Tracker	ATIYA KAZI1 , PRAPHULLA S. KHERADE2 , RAJ S. VILANKAR3 , PARAG M. SAWANT	2021	We are building an android application named as "Expense Tracker". As the name suggests, this project is an android app which is used to track the daily expenses of the user. It is like digital record keeping which keeps the records of expenses done by a user. The application keeps the track of the Income and Expenses both of user on a day-to-day basis. This application takes the income of a user and manage its daily expenses so that the user can save money	SQL lite queries and saw in advanced cell.
9.	A Review on Budget Estimator Android Application	Namita Jagtap1, Priyanka Joshi2, Aditya Kamble3	2019	In existing, we need to maintain the excel sheets, csv etc. files for the user daily and monthly expenses. In existing, there is no as such complete solution to keep a track of its daily expenditure easily. to do so a person as to keep a log in a diary or in a computer, also all the calculations needs to be done by the user which may sometimes results in errors leading to losses.	Firebase Authentication
10.	EXPENSE TRACKER MOBILE APPLICATION	Angad Manchanda	2012	Modern life offers a plethora of options of services and goods for consumers. As a result, people's expenses have gone up dramatically, e.g., compared to a decade ago, and the cost of living has been increasing day by day. Thus it becomes essential to keep a check on expenses in order to live a good life with a proper budget set up.	HTML, CSS and javascript code.

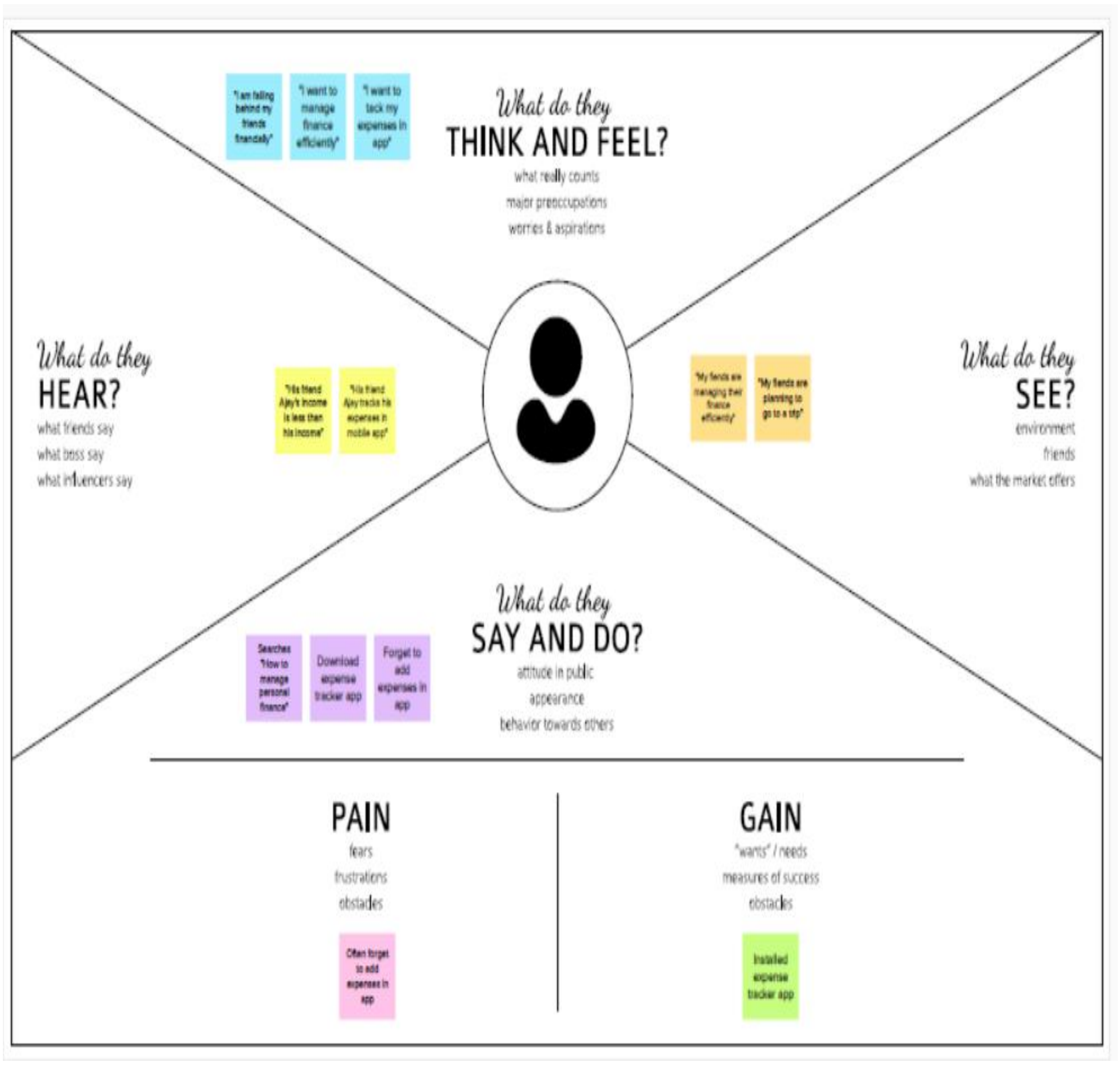
2.3 PROBLEM STATEMENT:

Many organizations have their own system to record their income and expenses, which they feel is the main key point of their business progress. It is good habit for a person to record daily expenses and earning but due to unawareness and lack of proper applications to suit their privacy, lacking decision making capacity people are using traditional note keeping methods to do so. Due to lack of a complete tracking system, there is a 2 constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.

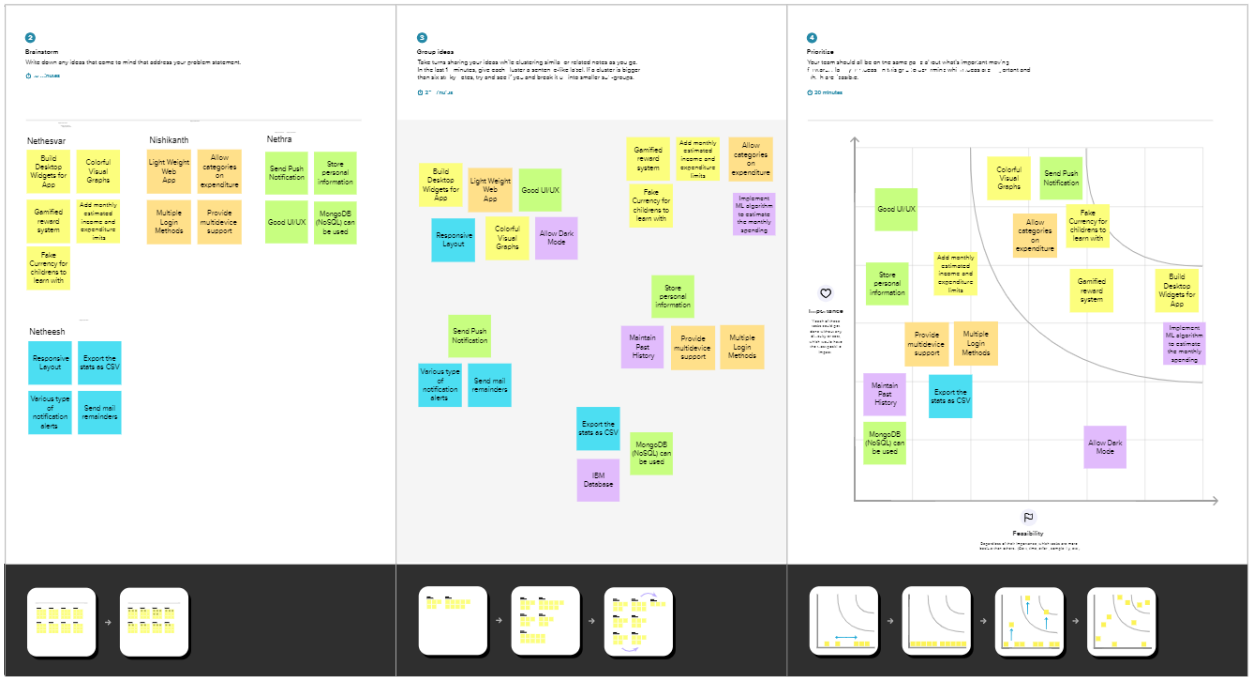
Who does the problem affect?	People getting regular wages.
What is the issue?	The paper based expense tracker system does not provide the user portability , existing system only used on paper based records so unable to update anywhere expenses done and unable to update the location of the expense detailsdisruptive that the proposed system.
When does the issue occurs?	When the digits could not be recognized correctly. When the transactions are not successful. When the elder people unable to understand the smaller handwritten digits. When the paper based expense tracker records are subjected to fire accident, flood, etc.
Where is the issue occurring?	The issue occurs when the person is unable to track his income and expenditure.
Why is it important that we fix the problem?	By solving this issue those people getting regular wages can track their expenses and avoid unwanted expenses.

3. IDEATION AND PROPOSED SOLUTION:

3.1 EMPATHY MAP CANVAS:


"I am falling behind my friends financially"
"I want to manage finance efficiently"
"I want to tack my expenses in app"
What do they
THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
What do they
HEAR?
what friends say
what boss say
what influencers say
"His friend Ajay's income is less than his income"
"His friend Ajay tracks his expenses in mobile app"
"My fiends are managing their finance efficiently"
"My fiends are planning to go to a trip"
What do they
SEE?
environment
friends
what the market offers
Searches "How to manage personal finance"
Download expense tracker app
Forget to add expenses in app
What do they
SAY AND DO?
attitude in public
appearance
behavior towards others
PAIN
fears
frustrations
obstacles
Often forget to add expenses in app
GAIN
"wants" / needs
measures of success
obstacles
Installed expense tracker app

3.2 IDEATION & BRAINSTORMING:

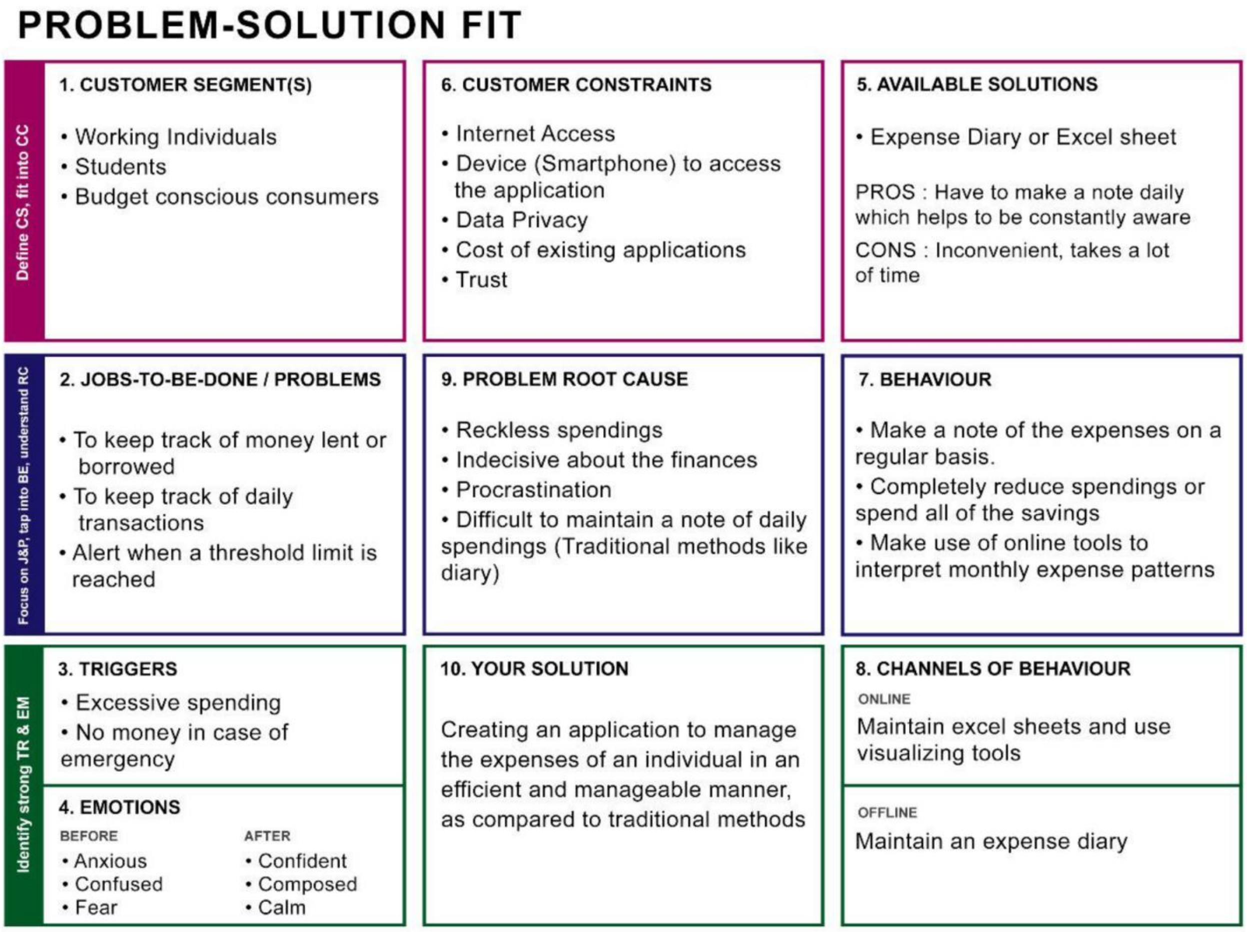

Brainstorm
Write down any ideas that come to mind that address your problem statement.
Nethesvar
Build Desktop Widgets for App
Colorful Visual Graphs
Gamified reward system
Add monthly estimated income and expenditure limits
Fake Currency for childrens to learn with
Nishikanth
Light Weight Web App
Allow categories on expenditure
Multiple Login Methods
Provide multidevice support
Nethra
Send Push Notification
Store personal information
Good UI/UX
MongoDB (NoSQL) can be used
Netheesh
Responsive Layout
Export the stats as CSV
Various type of notification alerts
Send mail remainders
Group ideas
Allow Dark Mode
Implement ML algorithm to estimate the monthly spending
Maintain Past History
IBM Database
Prioritize
Importance
Feasibility
20 minutes

3.3 PROPOSED SOLUTION:

S.No.	Parameter	Description
1.	Problem Statement (Problem to besolved)	In a Traditional Paper based expense tracking system, it is difficult to track our monthly expenses manually. Some of the records may get lost in case of fire, floods, etc. We are trying to solve this problem in a more efficient way.
2.	Idea / Solution description	This expense tracker is a computerised application which keeps track of all your finances and helps in accounting andbudgeting.
3.	Novelty / Uniqueness	The User gets notified once their expensetouches 50% 75% 90% & 100% of their limits. Display the costs on a monthly and weekly basis in a pie chart.

4.	Social Impact / CustomerSatisfaction	This Application is able to generate reportsof their spendings. It can create awarenessamong common people about finance.It makes users financially responsible and satisfy them without letting them to debt.
5.	Business Model (Revenue Model)	As this project is intended purely foreducational purposes, we keep this application free of cost.
6.	Scalability of the Solution	This Application can handle large numbersof users and data with high performance and security. This application can be used for both large scale and small scalepurposes.

3.4 PROPOSED SOLUTION FIT:



PROBLEM-SOLUTION FIT

Define CS, fit into CC

1. CUSTOMER SEGMENT(S)

- Working Individuals
- Students
- Budget conscious consumers

6. CUSTOMER CONSTRAINTS

- Internet Access
- Device (Smartphone) to access the application
- Data Privacy
- Cost of existing applications
- Trust

5. AVAILABLE SOLUTIONS

- Expense Diary or Excel sheet

PROS : Have to make a note daily which helps to be constantly aware

CONS : Inconvenient, takes a lot of time

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS

- To keep track of money lent or borrowed
- To keep track of daily transactions
- Alert when a threshold limit is reached

9. PROBLEM ROOT CAUSE

- Reckless spendings
- Indecisive about the finances
- Procrastination
- Difficult to maintain a note of daily spendings (Traditional methods like diary)

7. BEHAVIOUR

- Make a note of the expenses on a regular basis.
- Completely reduce spendings or spend all of the savings
- Make use of online tools to interpret monthly expense patterns

Identify strong TR & EM

3. TRIGGERS

- Excessive spending
- No money in case of emergency

4. EMOTIONS

BEFORE	AFTER
• Anxious	• Confident
• Confused	• Composed
• Fear	• Calm

10. YOUR SOLUTION

Creating an application to manage the expenses of an individual in an efficient and manageable manner, as compared to traditional methods

8. CHANNELS OF BEHAVIOUR

ONLINE

Maintain excel sheets and use visualizing tools

OFFLINE

Maintain an expense diary

4. REQUIREMENT ANALYSIS:

4.1 FUNCTIONAL REQUIREMENT:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through Email Account
FR-2	User Confirmation	Confirmation via Email
FR-3	Calendar	Personal expense tracker application must allowuser to add the data to their expenses.
FR-4	Graphical Representation	This application should graphically represent theexpense in the form of report.
FR-5	Report Generation	Graphical representation of report must begenerated.
FR-6	Category	This application shall allow users to add categoriesof their expenses.

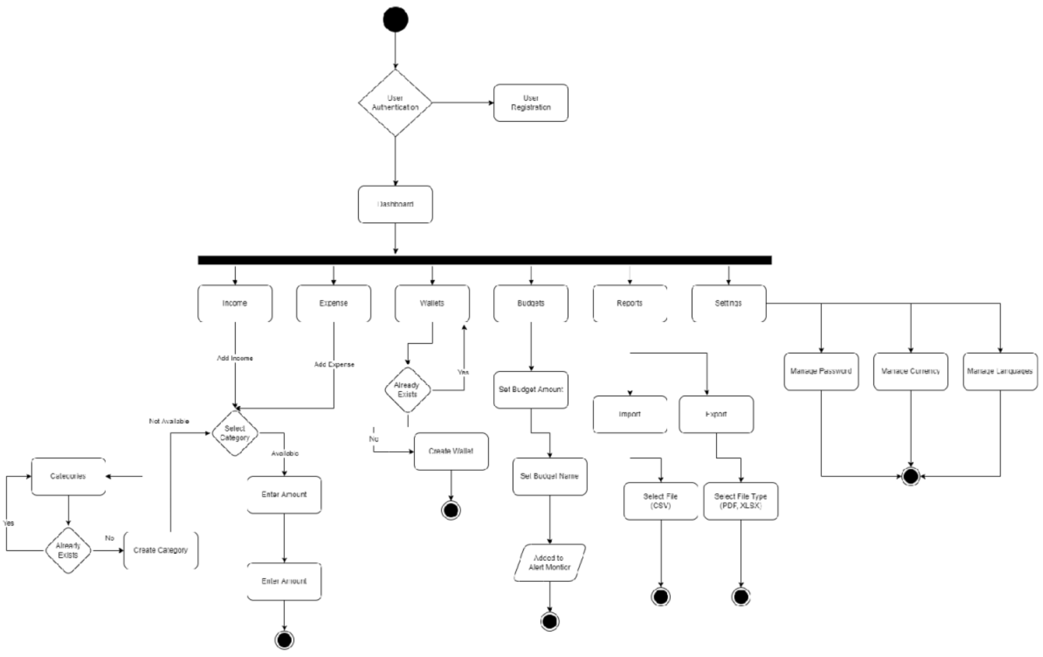
4.2 NONFUNCTIONAL REQUIREMENT:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Helps to keep an accurate record and track oftheir income and expenses easily.

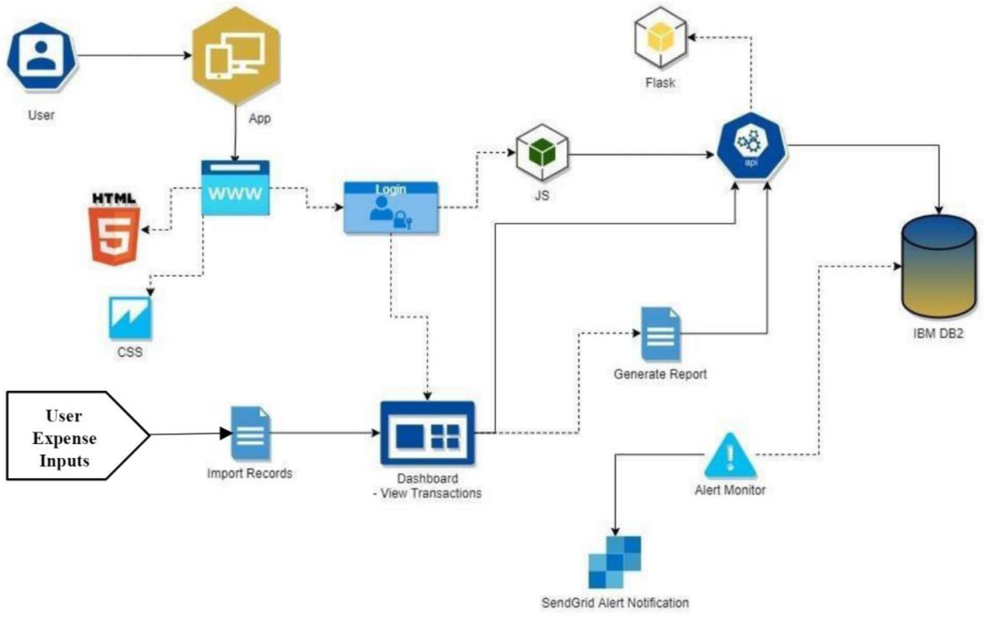
NFR-2	**Security**	We save the password in the encrypted form so itwill add more secure to the application user.
NFR-3	**Reliability**	Each data record is stored on a well-built efficientdatabase schema. There is no risk of data loss.
NFR-4	**Performance**	Expense kinds include categories and an option.The system's throughput is boosted because to the lightweight database support.
NFR-5	**Availability**	User can able to access the application with thehelp of the internet throw the web browser.
NFR-6	**Scalability**	The ability to appropriately handle increasingdemands.

5. PROJECT DESIGNING:

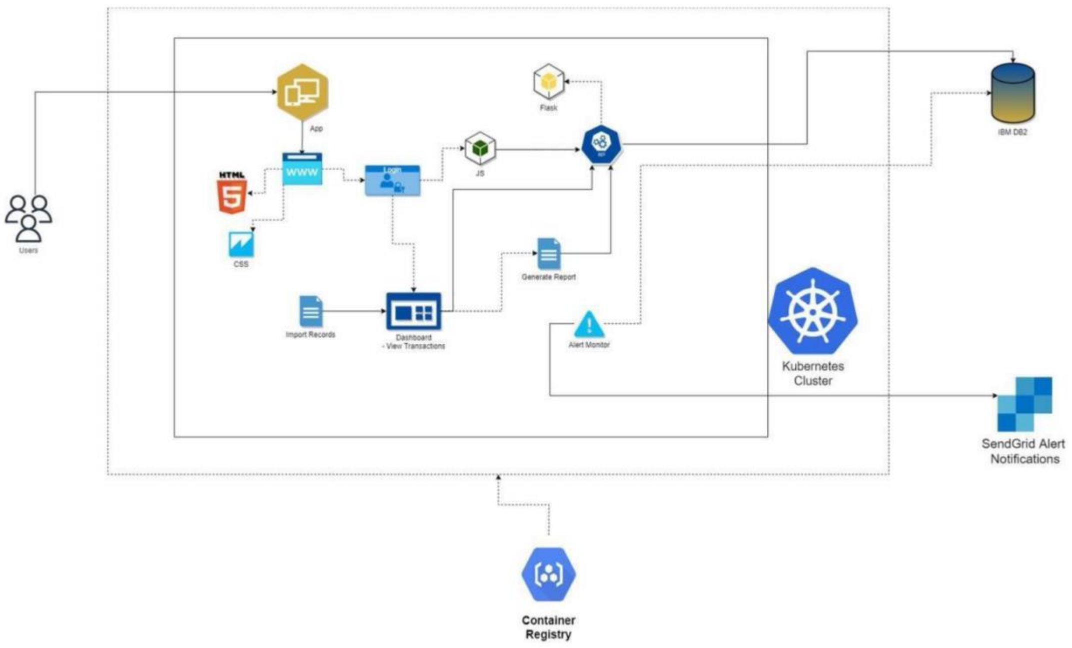
5.1 DATAFLOW DIAGRAM



5.2 SOLUTION AND TECHNICAL ARCHITECTURE:



Technology Architecture:



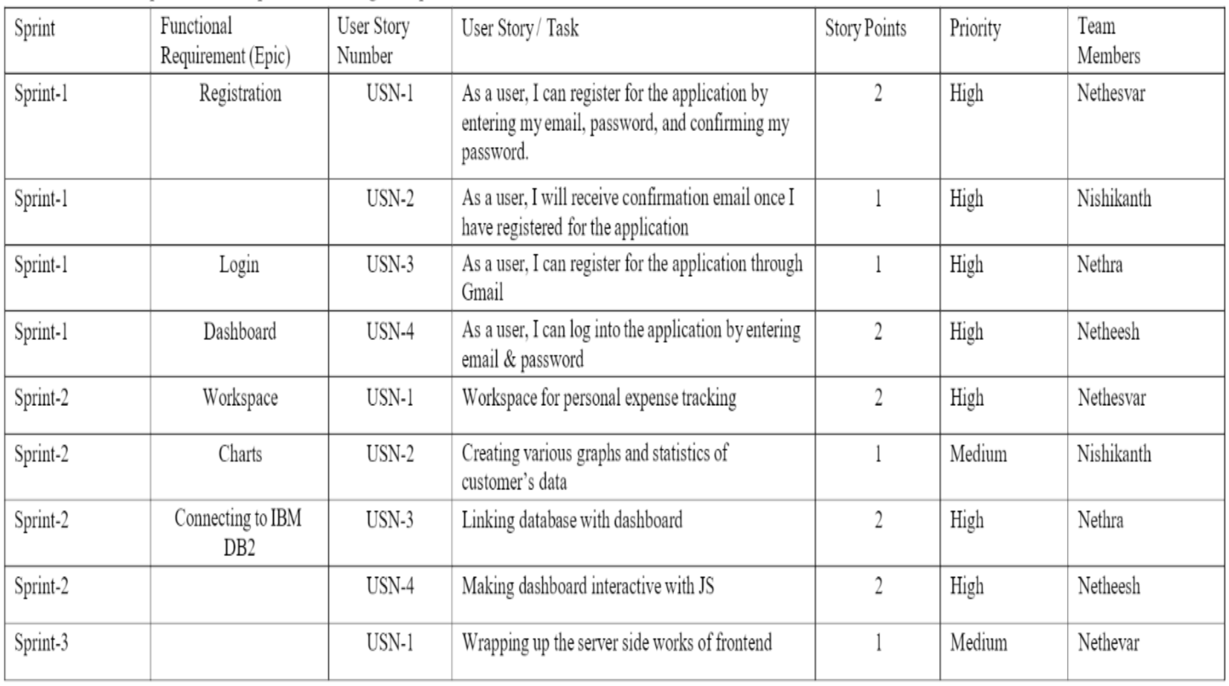
5.3 USER STORIES:

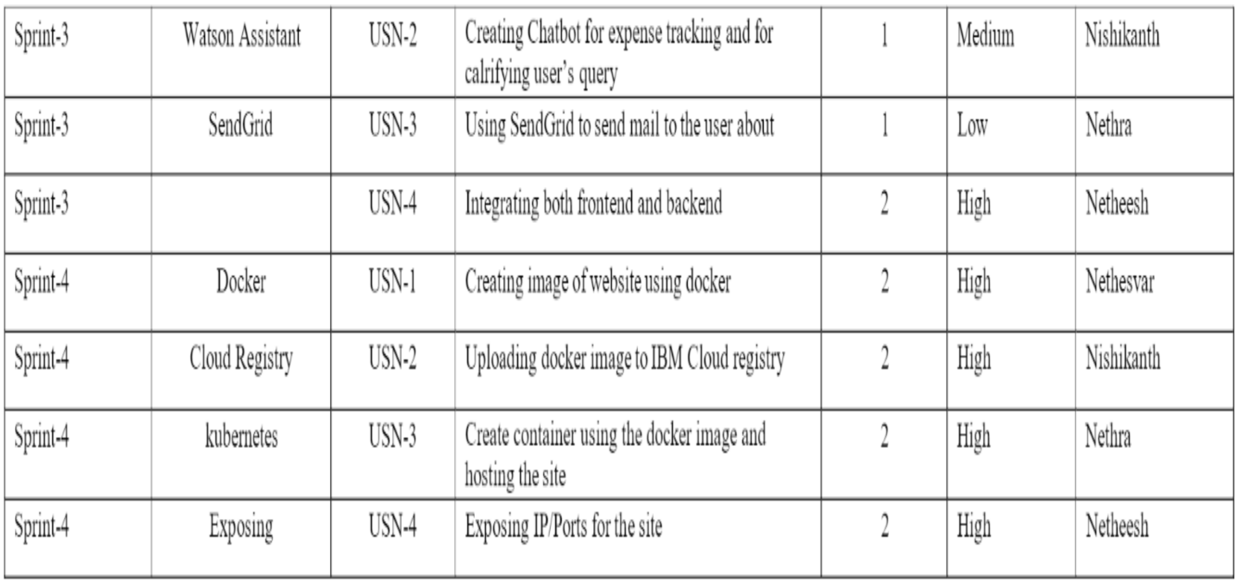
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirmingmy password.	I can access my account /dashboard	High	Sprint-1
	Login	USN-2	As a user, I can register for the applicationthrough Gmail	I can access my account/dashboard.	High	Sprint-1
	Dashboard	USN-3	As a user, I can log into the application byentering email & password	I get all the info needed inmy dashboard.	Low	Sprint-2
	Order creation	USN-4	As a customer, I can place my order withthe detailed description of my query	I can ask my query	Medium	Sprint-2
	Forgot password	USN-5	As a customer, I can reset my password bythis option in case I forgot my old password.	I get access to my accountagain	High	Sprint-3
	Order details	USN-6	As a Customer, I can see the current stats oforder.	I get better understanding	Medium	Sprint-4
Agent (Web user)	Login	USN-1	As an agent I can login to the application byentering correct email and password	I can access my account/dashboard	Medium	Sprint-3
	Dashboard	USN-2	As an agent I can see the order detailsassigned to me by admin	I can see the tickets to which I could answer	High	Sprint-3
	Address column	USN-3	As an agent I get to have conversation withthe customer and clear his/her doubts	I can clarify issues	High	Sprint-3
	Forgot password	USN-4	As an agent I can reset my password by this option In case I forgot my old password	I get to access to my account again	Medium	Sprint-4
Admin user (mobile user and web user)	Login	USN-1	As a admin, I can login to the application byentering email and password	I can access my account/dashboard	High	Sprint-1
	Dashboard	USN-2	As an admin I can see all the orders raised inthe entire system and lot more	I can assign agents by seeing those order.	High	Sprint-1
	Agent creation	USN-3	As an admin I can create an agent for clarifying the	I can create agents	High	Sprint-2

			customers queries			
	Assignment agent	USN-4	As an admin I can assign an agent for eachorder created by the customer.	Enable agent to clarify thequeries.	High	Sprint-1
	Forgot password	USN-5	As an admin I can reset my password by thisoption in case I forgot my old password.	I get access to account	High	Sprint-1

6. PROJECT PLANNING AND SCHEDULING:

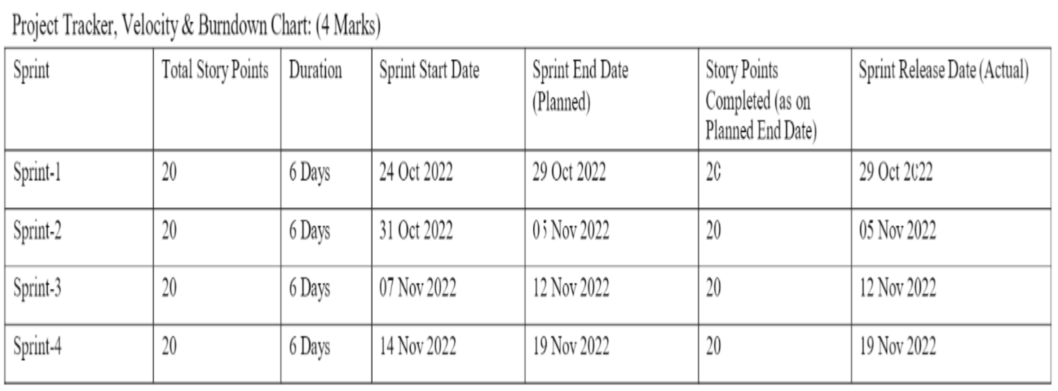
6.1 SPRINT PLANNING AND ESTIMATION:





Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Nethesvar
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Nishikanth
Sprint-1	Login	USN-3	As a user, I can register for the application through Gmail	1	High	Nethra
Sprint-1	Dashboard	USN-4	As a user, I can log into the application by entering email & password	2	High	Netheesh
Sprint-2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Nethesvar
Sprint-2	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Nishikanth
Sprint-2	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Nethra
Sprint-2		USN-4	Making dashboard interactive with JS	2	High	Netheesh
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Nethevar
Sprint-3	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for calrifying user's query	1	Medium	Nishikanth
Sprint-3	SendGrid	USN-3	Using SendGrid to send mail to the user about	1	Low	Nethra
Sprint-3		USN-4	Integrating both frontend and backend	2	High	Netheesh
Sprint-4	Docker	USN-1	Creating image of website using docker	2	High	Nethesvar
Sprint-4	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Nishikanth
Sprint-4	kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Nethra
Sprint-4	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Netheesh

6.2 Sprint Delivery Plan:



Project Tracker, Velocity & Burndown Chart: (4 Marks)

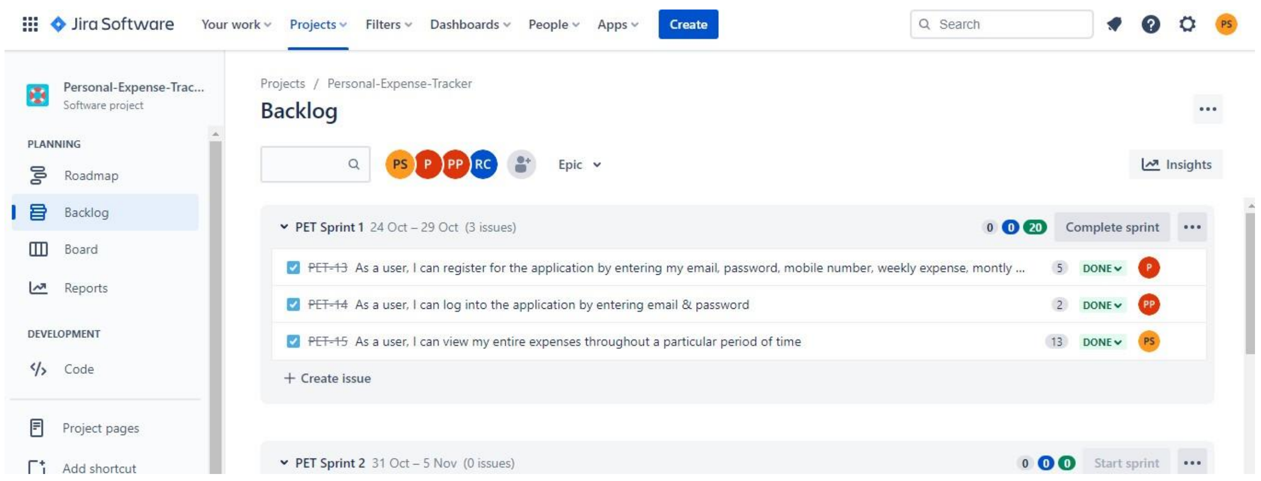
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:

Imagine we have a 10-day sprint duration and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV=Sprint\ Duration\ /\ Velocity=20/6=3.33$$

6.3 Reports from JIRA



PET Sprint 1 24 Oct – 29 Oct (3 issues) 0 0 20 Complete sprint

~~PET-13~~ As a user, I can register for the application by entering my email, password, mobile number, weekly expense, montly ...	5	DONE	P
~~PET-14~~ As a user, I can log into the application by entering email & password	2	DONE	PP
~~PET-15~~ As a user, I can view my entire expenses throughout a particular period of time	13	DONE	PS

+ Create issue

PET Sprint 2 31 Oct – 5 Nov (0 issues) 0 0 0 Start sprint

CHAPTER 7

CODING AND SOLUTIONING

base_template.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <!-- Required meta tags -->

  <meta charset="utf-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=device-width, initial-scale=1">


  <!-- Bootstrap CSS -->

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">


  <!-- bootstrap for the cards -->

  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">


  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>


  {% block title %}

    <title>Base Template</title>

  {% endblock title %}

</head>

<body>

  <div class="container-fluid">

    <div class="row flex-nowrap">

      <div class="col-auto col-md-3 col-xl-2 px-sm-2 px-0" style="background-color: #B2D3C2">

        <div class="d-flex flex-column align-items-center align-items-sm-start px-3 pt-2 min-vh-100" style="color:black">

          <p class="d-flex align-items-center pb-3 mb-md-0 me-md-auto text-white text-decoration-none">

            <span class="fs-5 d-none d-sm-inline" style="color:black; font-weight: bold;">Personal Expense Tracker</span>

            <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/piggybank.png"
style="width:50px;height: 50px;">

          </p>

          <ul class="nav nav-pills flex-column mb-sm-auto mb-0 align-items-center align-items-sm-start" id="menu">

            <li class="nav-item mt-2" style="background-color: {{'#00AD83' if highlight == 'dashboard'}}; height: 50px; width: 150px;
border-radius: 5px;">
```

```
                <a href="dashboard" class="nav-link align-middle px-0" style="color:black;">
                    <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/house-outline.svg"
style="width:20px;height:20px;margin-left: 5px;">
                    <span class="ms-1 d-none d-sm-inline">Home</span>
                </a>
            </li>


            <li class="nav-item mt-2" style="background-color: {{'#00AD83' if highlight == 'addexpense'}};">
                <a href="addexpense" class="nav-link px-0 align-middle" style="color:black;">
                    <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/pay-15.png"
style="width:20px;height:20px;margin-left: 5px;">
                    <span class="ms-1 d-none d-sm-inline">Add Expense</span>
                </a>
            </li>


            <li class="nav-item mt-2" style="background-color: {{'#00AD83' if highlight == 'recurringexpense'}};">
                <a href="recurringexpense" class="nav-link px-0 align-middle" style="color:black;">
                    <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/pay-15.png"
style="width:20px;height:20px;margin-left: 5px;">
                    <span class="ms-1 d-none d-sm-inline">Initiate a recurring expense</span>
                </a>
            </li>


            <!-- <li class="nav-item mt-2" style="background-color: {{'#00AD83' if highlight == 'modifyexpense'}};">
                <a href="modifyexpense" class="nav-link px-0 align-middle" style="color:black;">
                    <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/edit_icon.svg"
style="width:20px;height:20px;margin-left: 5px;">
                    <span class="ms-1 d-none d-sm-inline">Modify Expense</span>
                </a>
            </li> -->


            <li class="nav-item mt-2" style="background-color: {{'#00AD83' if highlight == 'viewrecurring'}};">
                <a href="viewrecurring" class="nav-link px-0 align-middle" style="color:black;">
                    <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/graph.png"
style="width:20px;height:20px;margin-left: 5px;">
                    <span class="ms-1 d-none d-sm-inline">View recurring expenses</span>
                </a>
            </li>


            <li class="nav-item mt-2" style="background-color: {{'#00AD83' if highlight == 'analysis'}};">
                <a href="analysis" class="nav-link px-0 align-middle" style="color:black;">
                    <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/graph.png"
style="width:20px;height:20px;margin-left: 5px;">
```

```
                                <span class="ms-1 d-none d-sm-inline">View Analysis</span>
                            </a>
                        </li>

                        <li class="nav-item mt-2" style="background-color: {{'#00AD83' if highlight == 'rewards'}};">
                            <a href="rewards" class="nav-link px-0 align-middle" style="color:black;">
                                <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/reward.png"
style="width:20px;height:20px;margin-left: 5px;">
                                <span class="ms-1 d-none d-sm-inline">Rewards & Goals</span>
                            </a>
                        </li>

                        <li class="nav-item mt-2" style="background-color: {{'#00AD83' if highlight == 'addcategory'}};">
                            <a href="addcategory" class="nav-link px-0 align-middle" style="color:black;">
                                <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/category_add.webp"
style="width:20px;height:20px;margin-left: 5px;">
                                <span class="ms-1 d-none d-sm-inline">Create category</span>
                            </a>
                        </li>

                        <li class="nav-item mt-2" style="background-color: {{'#00AD83' if highlight == 'setmonthlylimit'}};">
                            <a href="setmonthlylimit" class="nav-link px-0 align-middle" style="color:black;">
                                <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/category_add.webp"
style="width:20px;height:20px;margin-left: 5px;">
                                <span class="ms-1 d-none d-sm-inline">Set Monthly Limit</span>
                            </a>
                        </li>
                    </ul>

                    <ul class="nav nav-pills flex-column mb-sm-auto mb-0 align-items-center align-items-sm-end" id="menu">
                        <li class="nav-item mt-2">
                            <a href="logout" class="nav-link px-0 align-middle" style="color:black;">
                                <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/log-out.png"
style="width:20px;height:20px;margin-left: 5px;">
                                <span class="ms-1 d-none d-sm-inline">Log Out</span>
                            </a>
                        </li>
                    </ul>

                </div>
            </div>
            {% block content %}
```

```
                <h1>This needs to be overriden</h1>
            {% endblock content %}
        </div>
    </div>

    {% block script %}
    <script></script>
    {% endblock script %}
</body>
</html>
```

addcategory.html

```
{% extends 'base_template.html' %}

{% block title %}
<title>Add Category</title>
{% endblock title %}

{% set highlight = 'addcategory' %}

{% block content %}
<div class="col py-3" style="background-color:#00AD83">
    <h3 style="color:white; text-align: center;">Add category</h3>
    <div class="container mt-3" style="width: 600px;">
        <div class="card shadow-lg bg-white rounded">
            <form action="/addcategory" method="POST">
                <div class="card-header" style="text-align: center;">
                    <span style="display:inline-flex"><h4>New Category</h4><img src="https://s3.jp-tok.cloud-object-
storage.appdomain.cloud/personalexpensetrackercapd/category_add.webp" style=" margin-left:10px; width:30px; height:30px"></span>
                    <span style="display:inline-flex"><h5>Include a category called 'recurring' if you want to use recurring expenses</h5></span>

                </div>
                <div class="card-body">
                        <div class="mb-3">
                        <label for="category" class="form-label">Category Name: </label>
                        <input type="text" class="form-control" name="category" id="category"></input>
                        </div>
                        <div class="mb-3">
                        <label for="description" class="form-label">Description of Category: </label>
                        <input type="text" class="form-control" name="description" id="description"></input>
                        </div>

                </div>

                <div class="card-footer text-muted" style="text-align:center">

                    <button type="submit" style="background-color:#00AD83; border-color:#00AD83; border-radius:5px;">Add
category</button>

                </div>

            </form>

        </div>

    </div>

</div>

{% endblock content %}
```

addexpense.html

```
{% extends 'base_template.html' %}


{% block title %}

<title>Add Expense</title>

{% endblock title %}


{% set highlight = 'addexpense' %}


{% block content %}

<div class="col py-3" style="background-color:#00AD83">

    <h3 style="color:white; text-align: center;">Add expense</h3>

    <div class="container mt-3" style="width: 600px;">

        <div class="card shadow-lg bg-white rounded">

            <form action="/addexpense" method="POST">

                <div class="card-header" style="text-align: center;">

                    <span style="display:inline-flex"><h4>Expense Made</h4><img src="https://s3.jp-tok.cloud-object-
storage.appdomain.cloud/personalexpensetrackercapd/pay-15.png" style=" margin-left:10px; width:30px; height:30px"></span>

                </div>

                <div class="card-body">

                    <div class="mb-3">

                        <label for="amountspent" class="form-label">Amount Spent: (Rs) </label>

                        <input type="number" class="form-control" name="amountspent" id="amountspent" placeholder="100.00" required>

                    </div>

                    <div class="mb-3">

                        <label for="expensecategory" class="form-label">Expense Category: </label>

                        <select name="category" id="category" class="form-control" placeholder="Select a category" required>

                            <option value="">Select a category</option>
                            {% for cat in categories %}
                                <option value="{{ cat[0] }}">{{ cat[1] }}</option>
                            {% endfor %}
                        </select>
                    </div>
                    <div class="mb-3">
                        <label for="date" class="form-label">Date of Expense: </label>
                        <input type="date" class="form-control" name="date" id="date" required></input>
                    </div>
                    <div class="mb-3">
                        <label for="description" class="form-label">Description of Expense: </label>
                        <input type="text" class="form-control" name="description" id="description"></input>
                    </div>
                    <div class="mb-3">
                        <label for="group" class="form-label">Group(if needed): </label>
                        <div title="New group" style="float:right" value="Create group" onclick="addGroup()">ADD GROUP</div>
                        <br/>

                        <select name="group" id="group" class="form-control">
                            <option value="">Select existing group</option>
                            {% for group in groups %}
                                <option value="{{ group[0] }}">{{ group[1] }}</option>
                            {% endfor %}
                        </select>
                    </div>
                </div>
                <div class="card-footer text-muted" style="text-align:center">
                    <button type="submit" value="submit" style="background-color:#00AD83; border-color:#00AD83; border-
radius:5px;">Submit Expense</button>
                </div>
            </form>
        </div>
    </div>
</div>
{% endblock content %}

{% block script %}
<script>
    function addGroup(e) {
        // e.preventDefault();
        group = window.prompt('Enter group name: ')
```

```
        console.log('PROMPT WINDOW SHOWN'+group);

        const formData = new FormData();
        formData.append("groupname", group);

        const xhttp = new XMLHttpRequest();
        xhttp.onload = function() {
            if (this.readyState == 4 && this.status == 200) {
                var groupid= JSON.parse(this.responseText);
                console.log(groupid);
                // create option using DOM
                const newOption = document.createElement('option');
                const optionText = document.createTextNode(groupid['groupname']);
                newOption.appendChild(optionText);
                newOption.setAttribute('value',groupid['groupID']);
                const selectDropdown = document.getElementById('group');
                selectDropdown.appendChild(newOption);
                console.log('GROUPID :'+ groupid['groupID']);
            }
        }
        xhttp.open("POST", "http://localhost:5000/addgroup");
        xhttp.send(formData);
    }
    document.querySelector('#date').valueAsDate = new Date();
</script>
{% endblock script %}
```

addgoal.html

```
{% extends 'base_template.html' %}

{% block title %}
<title>Add Goal and Reward</title>
{% endblock title %}

{% block content %}
<div class="col py-3" style="background-color:#00AD83">
    <h3 style="color:white; text-align: center;">Add Goal and Reward</h3>
    <div class="container mt-3" style="width: 600px;">
        <div class="card shadow-lg bg-white rounded">
            <form action="/addgoal" method="POST">
```

```
                <div class="card-header" style="text-align: center;">
                    <span style="display:inline-flex"><h4>Goal & Reward</h4><img src="https://s3.jp-tok.cloud-object-
storage.appdomain.cloud/personalexpensetrackercapd/goal-icon.webp" style=" margin-left:10px; width:30px; height:30px"></span>
                </div>
                <div class="card-body">
                    <div class="mb-3">
                        <label for="amountspent" class="form-label">Goal Wallet Balance: (Rs) </label>
                        <input type="number" class="form-control" name="goal_amount" id="goal_amount" placeholder="100.00" required>
                    </div>

                    <div class="mb-3">
                        <label for="date" class="form-label">Date of Validity: </label>
                        <input type="date" class="form-control" name="date" id="date" required></input>
                    </div>

                    <div class="mb-3">
                        <label for="description" class="form-label">Reward: </label>
                        <input type="text" class="form-control" name="reward" id="reward"></input>
                    </div>
                </div>
                <div class="card-footer text-muted" style="text-align:center">
                    <button type="submit" value="submit" style="background-color:#00AD83; border-color:#00AD83; border-
radius:5px;">Create Goal & Reward</button>
                </div>
            </form>
        </div>
    </div>
</div>
{% endblock content %}
```

analysis.html

```
{% extends 'base_template.html' %}

{% block title %}
<title>Analysis</title>
{% endblock title %}

{% set highlight = 'analysis' %}

{% block content %}
```

```
<div class="col-auto px-0 col-lg-10 col-md-6 col-sm-4">
  <div class="card min-vh-100" style="background-color: #00ad83">
    <h4 class="card-header">Analysis of my expenses</h4>
    <div class="card-body">
      <div class="row flex-nowrap">
        <div class="col col-lg-5 col-md-3 px-4" style="background-color: #00ad83">
          <img
            id="picture"
            src="data:image/jpeg;base64,{{ img_data1 }}"
          />
        </div>
        <div class="col col-lg-5 col-md-3 px-4" style="background-color: #00ad83">
          <img
            id="picture"
            src="data:image/jpeg;base64,{{ img_data2 }}"
          />
        </div>
      </div>
    </div>
  </div>
</div>
{% endblock content %}

{% block script %}
<script type="text/javascript">
  function generate_graph1() {}
</script>
{% endblock script %}
```

dashboard.html

```
{% extends 'base_template.html' %}

{% block title %}
<title>Dashboard</title>
{% endblock title %}

{% set highlight = 'dashboard' %}

{% block content %}
```

```
<div class="col py-3" style="background-color:#00AD83">

   <h4 style="color:red;">{{ msg }}</h4>

   <h3 style="color:black; text-align: center;">Welcome Back! {{ email }}</h3>

   <div class="d-flex justify-content-end">

      <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/wallet_money.webp"

         style="height: 30px; width:30px">

      <h4 style="margin-left:10px;">Wallet Balance: <span><h5 style="display:inline"><i>{{wallet}}</i></h5></span></h4>

      <a href="updatebalance"><img src="https://s3.jp-tok.cloud-object-
storage.appdomain.cloud/personalexpensetrackercapd/edit_pencil.png"

         style="margin-left:5px; height: 30px; width:30px"></a>

   </div>

   <h3>Here are your expenses:</h3>

   <div class="card-deck">

      {% for expense in expenses %}

      <div class="card shadow-lg bg-white rounded" style="margin: 20px;width:20rem; height:20rem;">

         <div class="card-header" style="text-align: center;">

            <h4>Expense {{loop.index}}</h4>

         </div>

         <div class="card-body">

            <h6 class="card-text">

               Amount Spent:

               <span style="color:#00AD83"> Rs {{expense['EXPENSE_AMOUNT']}}</span>

               <br><br>

               Description:

               <span style="color:#00AD83">{{expense['DESCRIPTION']}}</span>

               <br><br>

               Category:

               <span style="color:#00AD83">{{expense['CATEGORY_NAME']}}</span>

            </h6>

            <a href="/modifyexpense?expenseid={{expense['EXPENSEID']}}">Modify</a>

         </div>

         <div class="card-footer text-muted" style="text-align:center">

            <h6>Date on which Expense was made: <span style="color:#00AD83">{{expense['DATE']}}</span></h6>

         </div>

      </div>

      {% endfor %}

   </div>


</div>

{% endblock content %}
```

login.html

```
<!doctype html>

<html lang="en">

 <head>

  <!-- Required meta tags -->

  <meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">



  <!-- Bootstrap CSS -->

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">



  <title>Login</title>

 </head>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>



  <body style="background-color:#B2D3C2">

      <div class="container mt-3">

          <h1 style="color: black; text-align: center;">

              Personal Expense Tracker <img src="https://s3.jp-tok.cloud-object-
storage.appdomain.cloud/personalexpensetrackercapd/piggybank.png" style="width:50px;height: 50px;">

          </h1>

          <div class="container mt-5" style="width: 600px;">

            <h4>{{ msg }}</h4>

                <div class="card shadow-lg bg-white rounded">

                    <div class="card-header" style="text-align: center;">

                      <h4>Login</h4>

                    </div>

                    <div class="card-body">

                      <form action="/login" method="POST">

                        <div class="mb-3">

                            <label for="email" class="form-label">Email: </label>

                            <input type="email" class="form-control" name="email" id="email" placeholder="abc@gmail.com">

                          </div>

                          <div class="mb-3">

                            <label for="passowrd" class="form-label">Password: </label>

                            <input type="password" class="form-control" name="password" id="password"></input>

                          </div>

                          <button type="submit" style="background-color:#00AD83; border-color:#00AD83; border-radius:5px;">Login</button>
                    </form>
                  </div>
                  <div class="card-footer text-muted" style="text-align:center">
                    New user? <span><a href="/">Register Here</a></span>
                  </div>
                </div>
          </div>
       </div>
  </body>
</html>
```

modifyexpense.html

```
{% extends 'base_template.html' %}

{% block title %}
<title>Modify Expense</title>
{% endblock title %}

{% block content %}
<div class="col py-3" style="background-color:#00AD83">
    <h3 style="color:white; text-align: center;">Modify expense</h3>
    <div class="container mt-3" style="width: 600px;">
        <div class="card shadow-lg bg-white rounded">
            <form action="/modifyexpense" method="POST">
                <div class="card-header" style="text-align: center;">
                    <span style="display:inline-flex">
                        <h4>Expense Made</h4>
                        <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/pay-15.png" style=" margin-
left:10px; width:30px; height:30px">
                    </span>
                </div>
                <div class="card-body">
                    <div class="mb-3">
                        <label for="amountspent" class="form-label">Amount Spent: (Rs) </label>
                        <input type="number" class="form-control" name="amountspent" id="amountspent" placeholder="100.00"
value="{{expense['EXPENSE_AMOUNT']}}" required>
                    </div>

                    <div class="mb-3">
                        <label for="expensecategory" class="form-label">Expense Category: </label>
                        <select name="category" id="category" class="form-control" placeholder="Select a category">
                            <option value="">Select a category</option>
                            {% for category in categories %}
                                <option value="{{ category[0] }}" {{'selected' if expense['CATEGORYID'] == category[0]}}>{{ category[1]
}}</option>
                            {% endfor %}
                        </select>
                    </div>

                    <div class="mb-3">
                        <label for="date" class="form-label">Date of Expense: </label>
                        <input type="date" class="form-control" name="date" id="date" value="{{expense['DATE']}}" required></input>
                    </div>

                    <div class="mb-3">
                        <label for="description" class="form-label">Description of Expense: </label>
                        <input type="text" class="form-control" name="description" id="description"
value="{{expense['DESCRIPTION']}}"></input>
                    </div>

                    <div class="mb-3">
                        <label for="group" class="form-label">Group(if needed): </label>
                        <div title="New group" style="float:right" value="Create group" onclick="addGroup()">ADD GROUP</div><br/>

                        <select name="group" id="group" class="form-control">
                            <option value="">Select existing group</option>
                            {% for group in groups %}
                                <option value="{{ group[0] }}" {{'selected' if expense.get('GROUPID') and expense.get('GROUPID') ==
group[0]}}>{{ group[1] }}</option>
                            {% endfor %}
                        </select>
                    </div>

                    <input type="hidden" name="expenseid" value="{{expense['EXPENSEID']}}" />
                    <input type="hidden" name="oldamountspent" value="{{expense['EXPENSE_AMOUNT']}}" />
                </div>
                <div class="card-footer text-muted" style="text-align:center">
                    <button type="submit" value="submit" style="background-color:#00AD83; border-color:#00AD83; border-
radius:5px;">Submit Expense</button>
                </div>
            </form>
        </div>
    </div>
</div>
{% endblock content %}

{% block script %}
<script>
    function addGroup(e) {
        // e.preventDefault();
        group = window.prompt('Enter group name: ')
        console.log('PROMPT WINDOW SHOWN'+group);

        const formData = new FormData();
        formData.append("groupname", group);

        const xhttp = new XMLHttpRequest();
        xhttp.onload = function() {
            if (this.readyState == 4 && this.status == 200) {
                var groupid= JSON.parse(this.responseText);
                console.log(groupid);
                // create option using DOM
                const newOption = document.createElement('option');
                const optionText = document.createTextNode(groupid['groupname']);
                newOption.appendChild(optionText);
                newOption.setAttribute('value',groupid['groupID']);
                const selectDropdown = document.getElementById('group');
                selectDropdown.appendChild(newOption);
                console.log('GROUPID :'+ groupid['groupID']);
            }
        }
        xhttp.open("POST", "http://localhost:5000/addgroup");
        xhttp.send(formData);
    }
</script>
{% endblock script %}
```

recurringexpense.html

```
{% extends 'base_template.html' %}
{% block title %}
<title>Recurring Expense</title>
{% endblock title %}

{% set highlight = 'recurringexpense' %}

{% block content %}
<div class="col py-3" style="background-color:#00AD83">
  <h3 style="color:white; text-align: center;">Add Recurring Expense</h3>
  <div class="container mt-3" style="width: 600px;">
    <div class="card shadow-lg bg-white rounded">
      <form action="/recurringexpense" method="POST">
        <div class="card-header" style="text-align: center;">
          <span style="display:inline-flex"><h4>Expense Made</h4><img src="https://s3.jp-tok.cloud-object-
storage.appdomain.cloud/personalexpensetrackercapd/pay-15.png" style=" margin-left:10px; width:30px; height:30px"></span>
        </div>
        <div class="card-body">
          <div class="mb-3">
            <label for="amountspent" class="form-label">Amount Spent: (Rs) </label>
            <input type="number" class="form-control" name="amountspent" id="amountspent" placeholder="100.00" required>
            </div>
            <div class="mb-3">
            <label for="expensecategory" class="form-label">Expense Category: </label>
            <select name="category" id="category" class="form-control" placeholder="Select a category">
              <option value="">Select a category</option>
              {% for cat in categories %}
                <option value="{{ cat[0] }}">{{ cat[1] }}</option>
              {% endfor %}
            </select>
            </div>
            <div class="mb-3">
            <label for="date" class="form-label">Date of Expense: </label>
            <input type="date" class="form-control" name="date" id="date" required></input>
            </div>
            <div class="mb-3">
            <label for="description" class="form-label">Description of Expense: </label>
            <input type="text" class="form-control" name="description" id="description"></input>
            </div>
            <!-- <div class="mb-3">
            <label for="duration" class="form-label">Number of autorenewals (in months) </label>
```

```
                    <input type="text" class="form-control" name="autorenewals" id="autorenewals"></input>

                    </div> -->

                    <!-- <div class="mb-3"> -->

                    <!-- <label for="group" class="form-label">Group(if needed): </label> -->

                    <!-- <div title="New group" style="float:right" value="Create group" onclick="addGroup()">ADD GROUP</div><br/>


                    <select name="group" id="group" class="form-control">

                        <option value="">Select existing group</option>

                        {% for group in groups %}

                            <option value="{{ group[0] }}">{{ group[1] }}</option>

                        {% endfor %}

                    </select>

                    </div> -->

                <!-- </div> -->

                <div class="card-footer text-muted" style="text-align:center">

                    <button type="submit" value="submit" style="background-color:#00AD83; border-color:#00AD83; border-
radius:5px;">Submit Expense</button>

                </div>

            </form>

        </div>

    </div>

</div>

{% endblock content %}


{% block script %}

    <script>

        function addGroup(e) {

            // e.preventDefault();

            group = window.prompt('Enter group name: ')

            console.log('PROMPT WINDOW SHOWN'+group);


            const formData = new FormData();

            formData.append("groupname", group);


            const xhttp = new XMLHttpRequest();

            xhttp.onload = function() {

                if (this.readyState == 4 && this.status == 200) {

                    var groupid= JSON.parse(this.responseText);

                    console.log(groupid);

                    // create option using DOM

                    const newOption = document.createElement('option');
                    const optionText = document.createTextNode(groupid['groupname']);
                    newOption.appendChild(optionText);
                    newOption.setAttribute('value',groupid['groupID']);
                    const selectDropdown = document.getElementById('group');
                    selectDropdown.appendChild(newOption);
                    console.log('GROUPID :'+ groupid['groupID']);
                  }
              }
              xhttp.open("POST", "http://localhost:5000/addgroup");
              xhttp.send(formData);
          }
          document.querySelector('#date').valueAsDate = new Date();
      </script>
{% endblock script %}
```

registration.html

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

    <title>Registration</title>
  </head>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

    <body style="background-color:#B2D3C2">
        <div class="container mt-3">
            <h1 style="color: black; text-align: center;">
                Personal Expense Tracker <img src="https://s3.jp-tok.cloud-object-
storage.appdomain.cloud/personalexpensetrackercapd/piggybank.png" style="width:50px;height: 45px;">
            </h1>
            <div class="container mt-2" style="width: 600px;">
                <div class="card shadow-lg bg-white rounded">
            <div class="card-header" style="text-align: center;">
              <h4>Registration Form</h4>
            </div>
            <div class="card-body">
              <form action="/" method="POST">
                <div class="mb-3">
                  <label for="email" class="form-label">Email: </label>
                  <input type="email" class="form-control" name="email" id="email" placeholder="abc@gmail.com">
                </div>
                <div class="mb-3">
                  <label for="passowrd" class="form-label">Password: </label>
                  <input type="password" class="form-control" name="password" id="password"></input>
                  <p style="color: gray;" class="mt-3">Please make sure that the password meets the following requirements:</p>
                  <ol style="color: gray;"><li>Minimum of 8 characters</li><li>Contains an upper case and a special character</li></ol>
                </div>
                <div class="mb-3">
                  <label for="confirmpassword" class="form-label">Confrim Password: </label>
                  <input type="password" class="form-control" name="confirmpassword" id="confirmpassword"
placeholder="********">
                </div>
                <div class="mb-3">
                  <label for="wallet" class="form-label">Initial Wallet Amount (Rs): </label>
                  <input type="number" class="form-control" name="wallet" id="wallet" placeholder="10000.00">
                </div>
                <button type="submit" style="background-color:#00AD83; border-color:#00AD83; border-
radius:5px;">Register</button>
              </form>
            </div>
            <div class="card-footer text-muted" style="text-align:center">
              Already an existing user? <span><a href="login">Login Here</a></span>
            </div>
          </div>
      </div>
    </div>
  </body>
</html>
```

rewards.html

```
{% extends 'base_template.html' %}
{% block title %}
<title>Goals and Rewards</title>
{% endblock title %}

{% set highlight = 'rewards' %}

{% block content %}

<div class="col py-3" style="background-color:#00AD83">

    <h3>Here are your current rewards and goals:</h3>
    <div class="card-deck">
        <!-- {% set count = 1 %} -->
        {% for goal in goals %}

        <div class="card shadow-lg bg-white rounded" style="margin: 20px;width:20rem; height:20rem;">
            <div class="card-header" style="text-align: center;">
                <h4>Goal and Reward {{loop.index}}</h4>
            </div>
            <div class="card-body">
                <h6 class="card-text">Amount Set: <span style="color:#00AD83"> Rs {{goal[0]}}</span>
                <br><br>Reward: <span style="color:#00AD83">{{goal[2]}}</span></h6>
            </div>
            <div class="card-footer text-muted" style="text-align:center">
                <h6><br><br>Date of Validity: <span style="color:#00AD83">{{goal[1]}}</span></h6>
            </div>
        </div>
        {% endfor %}
    </div>

    <div style="text-align: center; margin-top: 5px">
        <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/goal-icon.webp"
            style="margin-left:5px; height:30px; width:30px;">
        <a href="addgoal" style="color:black; text-decoration:none;"><h4 style="display: inline">Add Goal and Reward</h4></a>
    </div>



</div>
{% endblock content %}
```

setmonthlylimit.html

```
{% extends 'base_template.html' %}


{% block title %}
<title>Set Monthly Limit</title>
{% endblock title %}


{% set highlight = 'setmonthlylimit' %}


{% block content %}
<div class="col py-3" style="background-color:#00AD83">
    <h3 style="color:white; text-align: center;">Set Monthly Limit</h3>
    <div class="container mt-3" style="width: 600px;">
        <div class="card shadow-lg bg-white rounded">
            <form action="/setmonthlylimit" method="POST">
                <div class="card-header" style="text-align: center;">
                    <span style="display:inline-flex">
                        <h4>Monthly Limit</h4>
                        <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/pay-15.png" style=" margin-
left:10px; width:30px; height:30px">
                    </span>
                    <div class="card-body">
                        <div class="mb-3">
                            <label for="monthlylimit" class="form-label">Maximum amount allowed this month: (Rs) </label>
                            <input type="number" class="form-control" name="monthlylimit" id="monthlylimit" placeholder="5000.00" required>
                        </div>
                    </div>
                    <div class="card-footer text-muted" style="text-align:center">
                        <button type="submit" value="submit" style="background-color:#00AD83; border-color:#00AD83; border-radius:5px;">Set
Monthly Limit</button>
                    </div>
                </div>
            </form>
        </div>
    </div>
</div>
{% endblock content %}
```

updatebalance.html

```
{% extends 'base_template.html' %}
{% block title %}

<title>Update Balance</title>

{% endblock title %}


{% block content %}

<div class="col py-3" style="background-color:#00AD83">

   <h3 style="color:white; text-align: center;">Update Balance</h3>

   <div class="container mt-3" style="width: 600px;">

      <div class="card shadow-lg bg-white rounded">

         <form action="/updatebalance" method="POST">

            <div class="card-header" style="text-align: center;">

               <span style="display:inline-flex"><h4>Wallet Balance</h4><img src="https://s3.jp-tok.cloud-object-
storage.appdomain.cloud/personalexpensetrackercapd/wallet_money.webp" style=" margin-left:10px; width:30px; height:30px"></span>

            </div>

            <div class="card-body">

                  <div class="mb-3">

                  <label for="category" class="form-label">Current Balance: </label>

                  <input type="text" value={{wallet}} readonly>

                  </div>

                  <div class="mb-3">

                  <label for="description" class="form-label">New Balance: </label>

                  <input type="text" class="form-control" name="balanceupdated" id="balanceupdated"></input>

                  </div>

            </div>

            <div class="card-footer text-muted" style="text-align:center">

               <button type="submit" style="background-color:#00AD83; border-color:#00AD83; border-radius:5px;">Update
Balance</button>

            </div>

         </form>

      </div>

   </div>

</div>

{% endblock content %}
```

viewrecurring.html

```
{% extends 'base_template.html' %}


{% block title %}

<title>View Recurring Expenses</title>
```

```
{% endblock title %}


{% set highlight = 'viewrecurring' %}


{% block content %}
<div class="col py-3" style="background-color:#00AD83">
   <h4 style="color:red;">{{ msg }}</h4>
   <h3 style="color:black; text-align: center;">Welcome Back! {{ email }}</h3>
   <div class="d-flex justify-content-end">
      <img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/wallet_money.webp"
         style="height: 30px; width:30px">
      <h4 style="margin-left:10px;">Wallet Balance: <span><h5 style="display:inline"><i>{{wallet}}</i></h5></span></h4>
      <a href="updatebalance"><img src="https://s3.jp-tok.cloud-object-
storage.appdomain.cloud/personalexpensetrackercapd/edit_pencil.png"
         style="margin-left:5px; height: 30px; width:30px"></a>
   </div>
   <h3>Here are your expenses:</h3>
   <div class="card-deck">
      <!-- {% set count = 1 %} -->
      {% for expense in expenses %}


      <div class="card shadow-lg bg-white rounded" style="margin: 20px;width:20rem; height:20rem;">
         <div class="card-header" style="text-align: center;">
            <h4>Expense {{loop.index}}</h4>
         </div>
         <div class="card-body">
            <h6 class="card-text">Amount Spent: <span style="color:#00AD83"> Rs {{expense[0]}}</span>
               <br><br>Reason: <span style="color:#00AD83">{{expense[1]}}</span>
            <!-- <br><br>Category: <span style="color:#00AD83">{{expense[3]}}</span></h6> -->
            <br><br> <button type = "button" name = "{{expense[1]}}" onclick="removeExpense(name)"> Remove Expense </button>
         </div>
         <div class="card-footer text-muted" style="text-align:center">
            <h6>Date on which Expense was initiated: <span style="color:#00AD83">{{expense[2]}}</span></h6>
         </div>
      </div>
      {% endfor %}
   </div>


</div>
{% endblock content %}
{% block script %}
<script>
   function removeExpense(e) {
      console.log("hello");
      // e.preventDefault();
      // group = window.prompt('Enter group name: ')
      // console.log('PROMPT WINDOW SHOWN'+group);

      window.alert("cancelling " + e + " autorenewal");
      const formData = new FormData();
      formData.append("description", e);

      const xhttp = new XMLHttpRequest();
      xhttp.onload = function() {
         if (this.readyState == 4 && this.status == 200) {
            window.location.reload
         }
      }
      xhttp.open("POST", "http://localhost:5000/removerecurring");
      xhttp.send(formData);
   }
</script>
{% endblock script %}
```

app.py

```
from flask import Flask, render_template, request, redirect, url_for
from flask_mail import Mail, Message
from datetime import datetime
from flask_cors import CORS, cross_origin
import ibm_db
import json
import plotly
import plotly.graph_objs as go
import pandas as pd
from flask import send_file
from io import BytesIO
import matplotlib.pyplot as plt
import numpy as np
import base64
from PIL import Image
```

```
import time

import atexit

from datetime import datetime

from apscheduler.schedulers.background import BackgroundScheduler


app = Flask(__name__, template_folder='templates')

app.config['SECRET_KEY'] = 'top-secret!'

app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'

app.config['MAIL_PORT'] = 587

app.config['MAIL_USE_TLS'] = True

app.config['MAIL_USERNAME'] = 'apikey'

app.config['MAIL_PASSWORD'] = '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'

app.config['MAIL_DEFAULT_SENDER'] = 'nunnaaarthi@gmail.com'

mail = Mail(app)

cors = CORS(app)

app.config['CORS_HEADERS'] = 'Content-Type'


# GLobal variables

EMAIL = ''

USERID = ''

print()

try:

    conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=54a2f15b-5c0f-46df-8954-
7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32733;Security=SSL;SSLServerCertificate=DigiCertGlobalRootC
A.crt;UID=nlg66799;PWD=CXtQLAGZ06fD0fhC;", "", "")

except Exception as e:

    print(e)

# FUNCTIONS INTERACTING WITH DB #

print('hello')


def fetch_walletamount():

    sql = 'SELECT WALLET FROM PETA_USER WHERE EMAIL=?'

    stmt = ibm_db.prepare(conn, sql)

    ibm_db.bind_param(stmt, 1, EMAIL)

    ibm_db.execute(stmt)

    user = ibm_db.fetch_assoc(stmt)

    # print(user['WALLET'])

    return user['WALLET']  # returns int




def fetch_categories():
    sql = 'SELECT * FROM PETA_CATEGORY WHERE USERID = ?'
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, USERID)
    ibm_db.execute(stmt)

    categories = []
    while ibm_db.fetch_row(stmt) != False:
        categories.append([ibm_db.result(stmt, "CATEGORYID"),
                           ibm_db.result(stmt, "CATEGORY_NAME")])

    sql = 'SELECT * FROM PETA_CATEGORY WHERE USERID IS NULL'
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)

    while ibm_db.fetch_row(stmt) != False:
        categories.append([ibm_db.result(stmt, "CATEGORYID"),
                           ibm_db.result(stmt, "CATEGORY_NAME")])

    # print(categories)
    return categories  # returns list



def fetch_userID():
    sql = 'SELECT USERID FROM PETA_USER WHERE EMAIL=?'
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, EMAIL)
    ibm_db.execute(stmt)
    user = ibm_db.fetch_assoc(stmt)
    # print(user['USERID'])
    return user['USERID']  # returns int



def fetch_groups():
    sql = 'SELECT * FROM PETA_GROUPS'
    stmt = ibm_db.exec_immediate(conn, sql)
    groups = []
    while ibm_db.fetch_row(stmt) != False:
        groups.append([ibm_db.result(stmt, "GROUPID"),
                       ibm_db.result(stmt, "GROUPNAME")])
    # print(groups)
    return groups  # returns list


def fetch_expenses():
    sql = 'SELECT * FROM PETA_EXPENSE where USERID = ' + str(USERID)
    # print(sql)
    stmt = ibm_db.exec_immediate(conn, sql)
    expenses = []
    while ibm_db.fetch_row(stmt):
        category_id = ibm_db.result(stmt, "CATEGORYID")
        category_id = str(category_id)
        sql2 = "SELECT * FROM PETA_CATEGORY WHERE CATEGORYID = " + category_id
        stmt2 = ibm_db.exec_immediate(conn, sql2)
        category_name = ""
        while ibm_db.fetch_row(stmt2) != False:
            category_name = ibm_db.result(stmt2, "CATEGORY_NAME")
        expenses.append([ibm_db.result(stmt, "EXPENSE_AMOUNT"), ibm_db.result(
            stmt, "DATE"), ibm_db.result(stmt, "DESCRIPTION"), category_name])
    # print(expenses)
    return expenses


def fetch_rec_expenses_cron():
    sql = 'SELECT * FROM PETA_REC_EXPENSES;'
    stmt = ibm_db.exec_immediate(conn, sql)
    rec_expenses = []
    while ibm_db.fetch_row(stmt) != False:
        amt = ibm_db.result(stmt, "AMOUNT")
        amt = str(amt)
        description = ibm_db.result(stmt, "DESCRIPTION")
        userid = ibm_db.result(stmt, "USERID")
        date = ibm_db.result(stmt, "RECDATE")
        rec_expenses.append([amt, description, date, userid])
    # print(rec_expenses)
    return rec_expenses


def fetch_rec_expenses():
    sql = 'SELECT * FROM PETA_REC_EXPENSES WHERE USERID = ' + str(USERID)
    stmt = ibm_db.exec_immediate(conn, sql)
    rec_expenses = []
    while ibm_db.fetch_row(stmt) != False:
        amt = ibm_db.result(stmt, "AMOUNT")
        amt = str(amt)
        description = ibm_db.result(stmt, "DESCRIPTION")
        userid = ibm_db.result(stmt, "USERID")
        date = ibm_db.result(stmt, "RECDATE")
        rec_expenses.append([amt, description, date, userid])
    # print(rec_expenses)
    return rec_expenses


def fetch_limits():
    now = datetime.now()
    year = now.year

    limits = [0 for i in range(12)]

    sql = 'SELECT LIMITAMOUNT, LIMITMONTH FROM PETA_LIMIT WHERE USERID = ? AND LIMITYEAR = ?'
    statement = execute_sql(sql, USERID, year)

    while ibm_db.fetch_row(statement):
        limit_amount = int(ibm_db.result(statement, 'LIMITAMOUNT'))
        limit_month = int(ibm_db.result(statement, 'LIMITMONTH'))
        limits[limit_month] = limit_amount

    return limits


# HELPER FUNCTIONS #
def fetch_latest_expenses(expenses):
    # must return expenses of last month
    latest_month = datetime.today().month
    latest_expenses = []
    for exp in expenses:
        if exp[1].month == latest_month:
            latest_expenses.append(exp)

    return latest_expenses
```

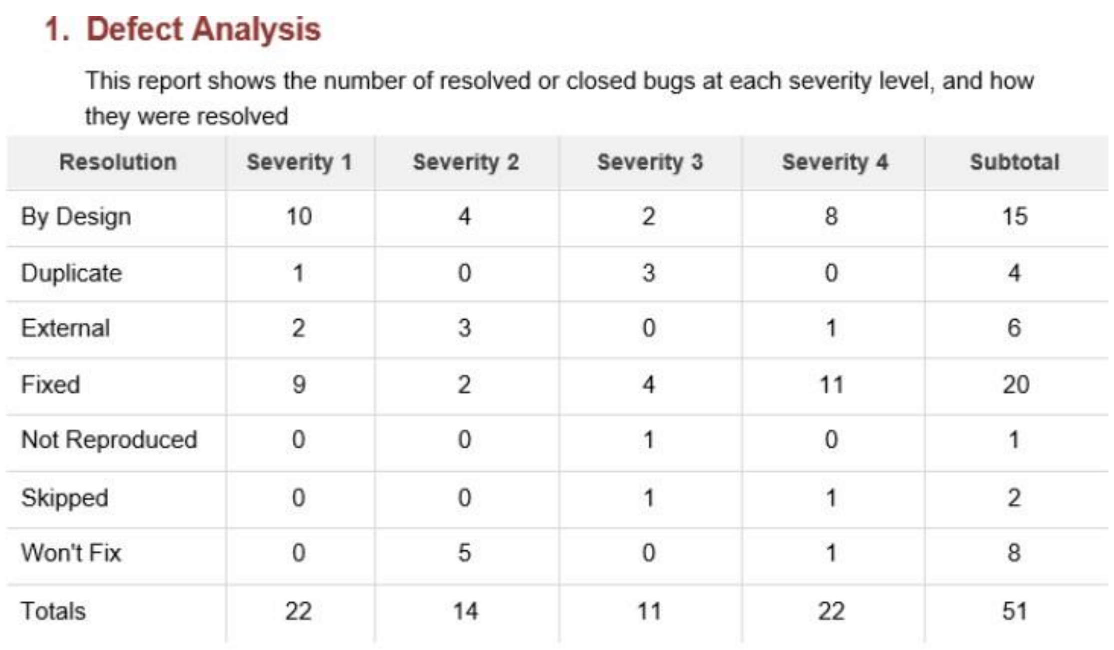
```
def fetch_monthly_expenses(expenses):
    latest_year = datetime.today().year
    monthly_expenses = {}

    for month in range(1, 13):
        monthly_expenses[month] = 0

    for exp in expenses:
        if exp[1].year == latest_year:
            monthly_expenses[exp[1].month] += exp[0]

    return monthly_expenses.values()


def draw_graph1(expenses):
    # TOTAL EXPENSE / DAY OF MONTH
    # x-axis: day , y-axis: expense/day

    latest_expenses = fetch_latest_expenses(expenses)
    mp = {}
    for day in range(1, 31):
        mp[day] = 0

    for exp in latest_expenses:
        mp[exp[1].day] += exp[0]

    x = mp.keys()
    y = mp.values()

    # print(mp)

    plt.figure()
    plt.title('Expense recorded over the past month')
    plt.plot(x, y)
    plt.xlabel('Day of the month')
    plt.ylabel('Recorded expense')
    plt.xlim(1, 32)

    buffer = BytesIO()
    plt.savefig(buffer, format='png')
    encoded_img_data = base64.b64encode(buffer.getvalue())

    return encoded_img_data


def draw_graph2(expenses, limits):
    # limit/month vs expense/month -> 2 line graphs

    monthly_expenses = fetch_monthly_expenses(expenses)
    x = range(1, 13)
    y1 = limits
    y2 = monthly_expenses

    plt.figure()
    plt.title('Month-wise comparison of limit and expense')
    plt.plot(x, y1, label="Limit/month")
    plt.plot(x, y2, label="Expenses/month")
    plt.xlabel('Month')
    plt.legend()

    buffer = BytesIO()
    plt.savefig(buffer, format='png')

    encoded_img_data = base64.b64encode(buffer.getvalue())

    return encoded_img_data

# finds the category id that matches that of the recurring expense category


def fetch_recurring_category_id():
    categories = fetch_categories()
    for category in categories:
        p = ''
        for i in category[1]:
            if (i == ' '):
                break
            p += i
        category[1] = p
    print(categories)
    categoryid = -1
    for category in categories:
        if category[1] == 'recurring':
            categoryid = category[0]
    print(categoryid)
    return categoryid



# cron to autodeduct the expenses each day
def auto_renew():
    global USERID
    # print(time.strftime("%A, %d. %B %Y %I:%M:%S %p"))
    rec_expenses = fetch_rec_expenses_cron()
    print(rec_expenses)
    current_day = time.strftime("%d")
    print(current_day)
    for expense in rec_expenses:
        here = str(expense[2])
        here = here.split('-')
        here = here[2]
        print(here)
        if (here == current_day):
            sql = "INSERT INTO PETA_EXPENSE(USERID, EXPENSE_AMOUNT, CATEGORYID, DESCRIPTION, DATE)
VALUES(?,?,?,?,?);"
            USERID = str(expense[3])
            categoryid = fetch_recurring_category_id()
            print(categoryid)
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, expense[3])
            ibm_db.bind_param(stmt, 2, expense[0])
            ibm_db.bind_param(stmt, 3, categoryid)
            ibm_db.bind_param(stmt, 4, expense[1])
            d3 = time.strftime("%Y-%m-%d")
            ibm_db.bind_param(stmt, 5, d3)
            print(d3, categoryid, expense[0],
                  expense[1], expense[2], expense[3])
            ibm_db.execute(stmt)

            check_monthly_limit(datetime.now().month, datetime.now().year)
            # print(here, d3, expense[0], expense[1], expense[2])
            sql = "UPDATE PETA_USER SET WALLET = WALLET - ? WHERE USERID = ?"
            statement = ibm_db.prepare(conn, sql)
            print(USERID)
            ibm_db.bind_param(statement, 1, expense[0])
            ibm_db.bind_param(statement, 2, expense[3])
            print("deducted")
            ibm_db.execute(statement)


# caller code for the cron
scheduler = BackgroundScheduler()
scheduler.add_job(func=auto_renew, trigger="interval", seconds=3600 * 24)
print('hello')
# END POINTS #
scheduler.start()
print('hello')
atexit.register(lambda: scheduler.shutdown())


@app.route('/', methods=['GET', 'POST'])
@cross_origin()
def registration():
    global EMAIL
    print("hello")
    if request.method == 'GET':
        return render_template('registration.html')
    if request.method == 'POST':
        email = request.form['email']
        EMAIL = email
        password = request.form['password']
        wallet = request.form['wallet']
        sql = "INSERT INTO PETA_USER(EMAIL,PASSWORD,WALLET) VALUES(?,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.bind_param(stmt, 3, wallet)
        print(stmt)
        ibm_db.execute(stmt)
        # msg = Message('Registration Verfication',recipients=[EMAIL])
        # msg.body = ('Congratulations! Welcome user!')
        # msg.html = ('<h1>Registration Verfication</h1>'
        #             '<p>Congratulations! Welcome user!'
        #             '<b>PETA</b>!</p>')
        # mail.send(msg)
        EMAIL = email
    return redirect(url_for('dashboard'))


@app.route('/login', methods=['GET', 'POST'])
def login():
    global EMAIL
    print("login")
    if request.method == 'POST':
        email = request.form['email']
        EMAIL = email
        print(EMAIL)
        password = request.form['password']
        sql = "SELECT * FROM PETA_USER WHERE email=? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        if account:
            return redirect(url_for('dashboard'))
        else:
            return redirect(url_for('login'))
    elif request.method == 'GET':
        return render_template('login.html')


@app.route('/logout', methods=['GET'])
def logout():
    if request.method == 'GET':
        global USERID
        global EMAIL
        USERID = ""
        EMAIL = ""
        return redirect(url_for('login'))


@app.route('/dashboard', methods=['GET'])
def dashboard():
    global USERID
    global EMAIL
    print("dashboard")
    if USERID == '' and EMAIL == '':
        print("null email")
        return render_template('login.html')
    elif USERID == '':
        USERID = fetch_userID()
        print(USERID)

    sql = "SELECT EXPENSEID, EXPENSE_AMOUNT, DESCRIPTION, CATEGORY_NAME, DATE FROM PETA_EXPENSE,
PETA_CATEGORY WHERE PETA_EXPENSE.USERID = ? AND PETA_EXPENSE.CATEGORYID =
PETA_CATEGORY.CATEGORYID"
    statement = execute_sql(sql, USERID)

    expenses = []
    while True:
        expense = ibm_db.fetch_assoc(statement)
        if expense:
            expenses.append(expense)
        else:
            break

    wallet = fetch_walletamount()
    return render_template('dashboard.html', expenses=expenses, wallet=wallet, email=EMAIL)



@app.route('/updatebalance', methods=['GET', 'POST'])
def update_balance():
    if request.method == 'GET':
        wallet = fetch_walletamount()
        return render_template('updatebalance.html', wallet=wallet)
    elif request.method == 'POST':
        global EMAIL
        global USERID
        if EMAIL == '':
            return render_template('login.html', msg='Login before proceeding')
        if (USERID == ''):
            # get user using email
            USERID = fetch_userID()

        new_balance = request.form['balanceupdated']
        sql = 'UPDATE PETA_USER SET WALLET = ? WHERE USERID = ?'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, new_balance)
        ibm_db.bind_param(stmt, 2, USERID)
        ibm_db.execute(stmt)

        return redirect(url_for('dashboard'))



@app.route('/addcategory', methods=['GET', 'POST'])
def add_category():
    if request.method == 'GET':
        # categories = fetch_categories()
        return render_template('addcategory.html')

    elif request.method == 'POST':
        categoryname = request.form['category']
        sql = 'INSERT INTO PETA_CATEGORY(CATEGORY_NAME, USERID) VALUES(?,?)'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, categoryname)
        ibm_db.bind_param(stmt, 2, USERID)
        ibm_db.execute(stmt)

        return redirect(url_for('dashboard'))



@app.route('/addgroup', methods=['POST'])
def add_group():
    if request.method == 'POST':
        if USERID == '':
            return render_template('login.html', msg='Login before proceeding')
        sql = "INSERT INTO PETA_GROUPS(GROUPNAME, USERID) VALUES(?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, request.form['groupname'])
        ibm_db.bind_param(stmt, 2, USERID)
        ibm_db.execute(stmt)

        group_info = {}

        sql = "SELECT * FROM PETA_GROUPS WHERE GROUPNAME=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, request.form['groupname'])
        ibm_db.execute(stmt)
        group_info = ibm_db.fetch_assoc(stmt)
        return {"groupID": group_info['GROUPID'], 'groupname': group_info['GROUPNAME']}



@app.route('/addexpense', methods=['GET', 'POST'])
def add_expense():
    if request.method == 'GET':
        groups = fetch_groups()
        categories = fetch_categories()
        if len(categories) == 0:
            return redirect(url_for('add_category'))
        return render_template('addexpense.html', categories=categories, groups=groups)

    elif request.method == 'POST':
        global EMAIL
        global USERID
        if EMAIL == '':
            return render_template('login.html', msg='Login before proceeding')
        if (USERID == ''):
            # get user using email
            USERID = fetch_userID()

        amount_spent = request.form['amountspent']
        category_id = request.form.get('category')
        description = request.form.get('description')
        date = request.form['date']

        groupid = request.form.get('group')
        groupid = None if groupid == '' else groupid

        print(amount_spent, category_id, description, date, groupid, USERID)

        sql = "INSERT INTO PETA_EXPENSE(USERID, EXPENSE_AMOUNT, CATEGORYID, GROUPID, DESCRIPTION, DATE)
VALUES(?,?,?,?,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, USERID)
        ibm_db.bind_param(stmt, 2, amount_spent)
        ibm_db.bind_param(stmt, 3, category_id)
        ibm_db.bind_param(stmt, 4, groupid)
        ibm_db.bind_param(stmt, 5, description)
        ibm_db.bind_param(stmt, 6, date)
        ibm_db.execute(stmt)
        print(date, amount_spent, category_id)
        sql = "UPDATE PETA_USER SET WALLET = WALLET - ? WHERE USERID = ?"
        statement = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(statement, 1, amount_spent)
        ibm_db.bind_param(statement, 2, USERID)
        ibm_db.execute(statement)

        return redirect(url_for('dashboard'))


@app.route('/viewrecurring', methods=['GET'])
def viewrecurring():
    global USERID
    global EMAIL
    print("viewrecurring")
    if USERID == '' and EMAIL == '':
        print("null email")
        return render_template('login.html')
    elif USERID == '':
        USERID = fetch_userID()
        # print(USERID)
    print(USERID)
    expenses = fetch_rec_expenses()
    wallet = fetch_walletamount()
    return render_template('viewrecurring.html', expenses=expenses, wallet=wallet, email=EMAIL)


@app.route('/recurringexpense', methods=['GET', 'POST'])
def recurring_expense():
    global USERID, EMAIL
    if request.method == 'GET':
        groups = fetch_groups()
        categories = fetch_categories()
        if len(categories) == 0:
            return redirect(url_for('add_category'))
        USERID = fetch_userID()
        # check if user has added a category for recurring category, if not redirect and ask her to
        recur_id = fetch_recurring_category_id()
        if (recur_id == -1):
            return (redirect(url_for('add_category')))
        return render_template('recurringexpense.html', categories=categories, groups=groups)

    elif request.method == 'POST':
        if EMAIL == '':
            return render_template('login.html', msg='Login before proceeding')
        if (USERID == ''):
            # get user using email
            USERID = fetch_userID()
            # check if user has added a category for recurring category, if not redirect and ask her to
            recur_id = fetch_recurring_category_id()
            if (recur_id == -1):
                return (redirect(url_for('add_category')))
        amount_spent = request.form['amountspent']
        category_id = request.form.get('category')
        description = request.form['description']
        date = request.form['date']
        # months = request.form['autorenewals']
        # groupid = request.form.get('group')
        print("recurring : ")
        print(amount_spent, description, date, USERID)

        sql = "INSERT INTO PETA_REC_EXPENSES(AMOUNT, RECDATE, USERID, DESCRIPTION) VALUES (?,?,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, amount_spent)
        ibm_db.bind_param(stmt, 2, date)
        ibm_db.bind_param(stmt, 3, USERID)
        ibm_db.bind_param(stmt, 4, description)
        ibm_db.execute(stmt)

        sql = "INSERT INTO PETA_EXPENSE(USERID, EXPENSE_AMOUNT, CATEGORYID, DESCRIPTION, DATE)
VALUES(?,?,?,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, USERID)
        ibm_db.bind_param(stmt, 2, amount_spent)
        ibm_db.bind_param(stmt, 3, category_id)
        ibm_db.bind_param(stmt, 4, description)
        ibm_db.bind_param(stmt, 5, date)
        ibm_db.execute(stmt)
        sql = "UPDATE PETA_USER SET WALLET = WALLET - ? WHERE USERID = ?;"
        statement = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(statement, 1, amount_spent)
        ibm_db.bind_param(statement, 2, USERID)
        ibm_db.execute(statement)

        return redirect(url_for('dashboard'))



@app.route('/removerecurring', methods=['POST'])
def remove_recurring():
    print("remove recurring")
    if request.method == 'POST':
        global EMAIL
        global USERID
        if EMAIL == '':
            return render_template('login.html', msg='Login before proceeding')
        if (USERID == ''):
            # get user using email
            USERID = fetch_userID()
        description = request.form['description']
        print(description, USERID)
        sql = 'DELETE FROM PETA_REC_EXPENSES WHERE USERID = ? AND DESCRIPTION = ?;'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, USERID)
        ibm_db.bind_param(stmt, 2, description)
        ibm_db.execute(stmt)

        return redirect(url_for('dashboard'))



@app.route('/analysis', methods=['GET', 'POST'])
def analyse():
    if request.method == 'GET':
        expenses = fetch_expenses()
        limits = fetch_limits()

        graph1 = draw_graph1(expenses=expenses)
        graph2 = draw_graph2(expenses=expenses, limits=limits)

        return render_template("analysis.html", img_data1=graph1.decode('utf-8'), img_data2=graph2.decode('utf-8'))
    elif request.method == 'POST':
        return render_template('analysis.html')



def execute_sql(sql, *args):
    stmt = ibm_db.prepare(conn, sql)
    for i, arg in enumerate(args):
        ibm_db.bind_param(stmt, i + 1, arg)
    ibm_db.execute(stmt)
    return stmt



def check_monthly_limit(month, year):
    sql = 'SELECT SUM(EXPENSE_AMOUNT) FROM PETA_EXPENSE WHERE USERID = ? AND MONTH(DATE) = ? AND
YEAR(DATE) = ?'
    statement = execute_sql(sql, USERID, month, year)
    amt_spent = ibm_db.fetch_tuple(statement)

    sql = 'SELECT LIMITAMOUNT FROM PETA_LIMIT WHERE USERID = ? AND LIMITMONTH = ? AND LIMITYEAR = ?'
    statement = execute_sql(sql, USERID, month, year)
    monthly_limit = ibm_db.fetch_tuple(statement)

    if amt_spent and monthly_limit and int(amt_spent[0]) > int(monthly_limit[0]):
        diff = int(amt_spent[0]) - int(monthly_limit[0])
        msg = Message('Monthly limit exceeded', recipients=[EMAIL])
        msg.body = (
            f'Monthly limit exceeded by {diff} for the month of {month}, {year}')
        mail.send(msg)



def update_monthly_limit(monthly_limit, month, year):
    sql = 'SELECT LIMITAMOUNT FROM PETA_LIMIT WHERE USERID = ? AND LIMITMONTH = ? AND LIMITYEAR = ?'
    statement = execute_sql(sql, USERID, month, year)

    if ibm_db.fetch_row(statement):
        sql = 'UPDATE PETA_LIMIT SET LIMITAMOUNT = ? WHERE USERID = ? AND LIMITMONTH = ? AND LIMITYEAR = ?'
        execute_sql(sql, monthly_limit, USERID, month, year)
    else:
        sql = 'INSERT INTO PETA_LIMIT VALUES(?, ?, ?, ?)'
        execute_sql(sql, USERID, monthly_limit, month, year)
    check_monthly_limit(month, year)



@app.route('/setmonthlylimit', methods=['GET', 'POST'])
def set_monthly_limit():
    if request.method == 'GET':
        return render_template('setmonthlylimit.html')
    elif request.method == 'POST':
        new_monthly_limit = request.form['monthlylimit']
        now = datetime.now()
        update_monthly_limit(new_monthly_limit, now.month, now.year)
        return redirect(url_for('dashboard'))



@app.route('/modifyexpense', methods=['GET', 'POST'])
def modify_expense():
    if request.method == 'GET':
        expenseid = request.args.get('expenseid')
        sql = "SELECT * FROM PETA_EXPENSE WHERE EXPENSEID = ?"
        statement = execute_sql(sql, expenseid)
        expense = ibm_db.fetch_assoc(statement)
        categories = fetch_categories()
        groups = fetch_groups()
        return render_template('modifyexpense.html', expense=expense, categories=categories, groups=groups)
    elif request.method == 'POST':
        amount_spent = request.form['amountspent']
        category_id = request.form.get('category')
        description = request.form['description']
        date = request.form['date']
        groupid = request.form.get('group')

        expenseid = request.form['expenseid']
        old_amount_spent = request.form['oldamountspent']

        sql = "UPDATE PETA_EXPENSE SET EXPENSE_AMOUNT = ?, CATEGORYID = ?, GROUPID = ?, DESCRIPTION = ?, DATE
= ? WHERE EXPENSEID = ?"
        execute_sql(sql, amount_spent, category_id,
                    groupid, description, date, expenseid)

        sql = "UPDATE PETA_USER SET WALLET = WALLET + ?"
        execute_sql(sql, float(old_amount_spent) - float(amount_spent))

        return redirect(url_for('dashboard'))


def fetch_goals():
    sql = 'SELECT * FROM PETA_GOALS WHERE USERID = ?'
    statement = execute_sql(sql, USERID)

    goals = []
    while True:
        goal = ibm_db.fetch_tuple(statement)
        if goal:
            goals.append(goal[2:])
        else:
            break

    print(goals)
    return goals


@app.route('/rewards', methods=['GET'])
def rewards_and_goals():
    goals = fetch_goals()
    return render_template('rewards.html', goals=goals)


@app.route('/addgoal', methods=['GET', 'POST'])
def add_goal():
    if request.method == 'GET':
        return render_template('addgoal.html')
    elif request.method == 'POST':
        goal_amount = request.form['goal_amount']
        date = request.form['date']
        reward = request.form['reward']
        sql = "INSERT INTO PETA_GOALS(USERID, GOAL_AMOUNT, DATE, REWARD) VALUES(?, ?, ?, ?)"
        execute_sql(sql, USERID, goal_amount, date, reward)
        return redirect(url_for('dashboard'))


def check_goals():
    sql = "SELECT A.GOALID, A.USERID, A.GOAL_AMOUNT, A.DATE, A.REWARD, B.WALLET FROM PETA_GOALS AS A,
PETA_USER AS B WHERE A.USERID = B.USERID"
    statement = execute_sql(sql)


    now = datetime.now()
    while True:
        row = ibm_db.fetch_assoc(statement)
        if not row:
            break
        if row['DATE'] == now:
            if row['GOAL_AMOUNT'] <= row['WALLET']:
                msg = Message('Goal achieved!', recipients=[EMAIL])
                msg.body = (
                    f'You are eligible for your reward:\n{row["REWARD"]}')
                mail.send(msg)
            else:
                msg = Message('Goal limit exceeded', recipients=[EMAIL])
                msg.body = (
                    f'You are not eligible for the reward:\n{row["REWARD"]}\nBetter luck next time!')
                mail.send(msg)
            sql = "DELETE FROM PETA_GOALS WHERE GOALID = ?"
            execute_sql(sql, row['GOALID'])



scheduler.add_job(func=check_goals, trigger="interval", seconds=3600 * 24)


if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True).
```

CHAPTER 8
TESTING

s. no	Test Case id	Feature Type	component	Test description	Input test Data	Actual output	Expected output	remarks
1	TC – RG 01	Func-tional	Register page	register for the application by entering my name, email, password, monthly limit	User1 User1@gmail.com ***** 10000	Registration successful	Registration successful	pass
2	TC – SI 01	Func-tional	Login page	log into the application by entering email & password	User1@gmail.com *****	Login successful	Login sucessfull	pass
3	TC – ST 01	UI	Stats page	view my entire expenses throughout a particular period of time		Expenses are displayed For particular time	Expenses are displayed For particular time	pass
4	TC – DB 01	UI	Dash-board	Display graph in dashboard		Graph is displayed	Graph is displayed	pass
5	TC – ST 02	Func-tional	Stats page	generate reports based on my previous expenditures		Reports generated in graphical form	Reports generated in graphical form	pass
6	TC – SI 02	Func-tional	Dash-board	can logout		Go to sign page	Sign in page displayed	pass

7	TC – ST 03	Func-tional	Stats page	create expense	14-11-2022 100 Food Debit Night food	Expenses created	Expenses created	pass
8	TC – ST 04	Func-tional	Stats page	can edit ,delete, update expense		Expenses updated	Updated of expenses	pass
9	TC – ST 05	UI	Stats page	can view credit and debit expenses separately.		Expenses are listed separately	Expenses are listed separately	pass
10	TC – ST 06	UI	Stats page	aware of the expense that I spend the most on		Expenses are listed for particular category	Expenses are listed for particular category	pass
11	TC – PG 01	Func-tional	Profile page	able to update my set monthly limit		Monthly limit updated	Monthly limit updated	pass
12	TC – PG 01	UI	Profile page	able to view my profile		Profile details displayed	Profile details displayed	pass

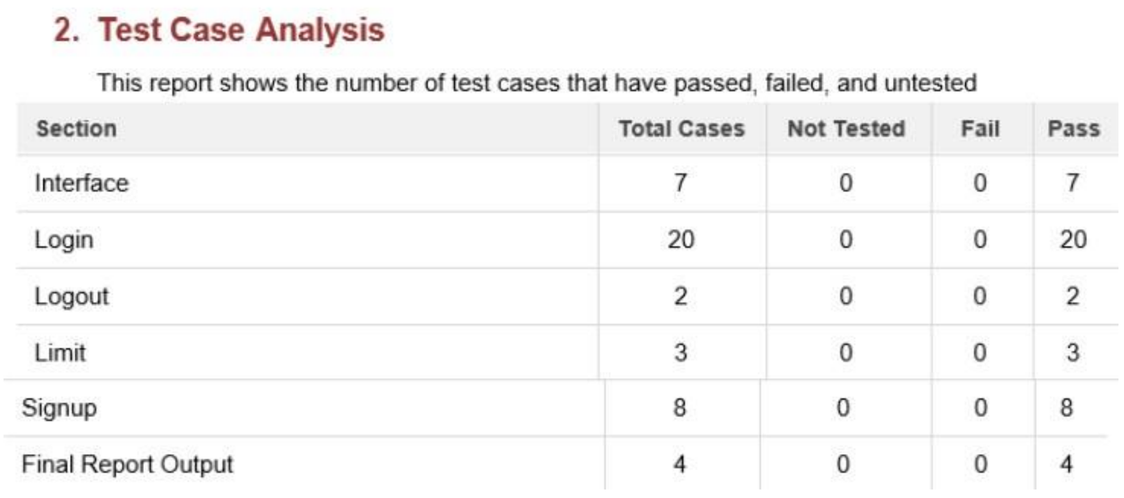
8.2 User Acceptance Testing



1. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	8	15
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	9	2	4	11	20
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	0	1	8
Totals	22	14	11	22	51



2. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Interface	7	0	0	7
Login	20	0	0	20
Logout	2	0	0	2
Limit	3	0	0	3
Signup	8	0	0	8
Final Report Output	4	0	0	4

CHAPTER 9

RESULT

9.1 Performance Metrics

i. Tracking income and expenses: Monitoring the income and tracking all expenditures (through bank accounts, mobile wallets, and credit & debit cards).

ii. Transaction Receipts: Capture and organize your payment receipts to keep track of your expenditure.

iii. Organizing Taxes: Import your documents to the expense tracking app, and it will streamline your income and expenses under the appropriate tax categories.

iv. Payments & Invoices: Accept and pay from credit cards, debit cards, net banking, mobile wallets, and bank transfers, and track the status of your invoices and bills in the mobile app itself. Also, the trackingapp sends remindersfor payments an d automatically matches the payments with invoices.

v. Reports: The expense tracking app generates and sends reports to give a detailed insight about profits, losses, budgets, income, balance sheets, etc.,

vi. Ecommerce integration: Integrateyour expense trackingapp wit h your eCommerce store and track your sales through payments received via multiple payment methods.

vii. Vendors and Contractors: Manage and track all the payments to the vendors and contractors added to the mobile app.

viii. Access control: Increase your team productivity by providing access control to particular users through custom permissions.

ix. Track Projects: Determine project profitability by tracking labor costs, payroll, expenses, etc., of your ongoing project.

x. Inventory tracking: An expense tracking app can do it all. Right from tracking products or the cost of goods, sending alert notifications when the product is running out of stock or the product is not selling, to purchase orders.

xi. In-depth insights and analytics: Provides in-built tools to generate reports with easy-to- understand visuals and graphics to gain insights about the performance of your business.

xii. Recurrent Expenses: Rely on your budgeting app to track, streamline, and automate all the recurrent expenses and remind you on a timely basis.

CHAPTER 10

ADVANTAGES AND DISADVANTAGES

Advantages :

- It allows users to track their expenses daily, weekly, monthly, and yearly interms of summary, bar graphs, and pie-charts.
- Separate view for credit and debit transactions
- No burden of manual calculations
- Generate and save reports.
- You can insert, delete records
- You can track expenses by categories like food, automobile, entertainment,education etc..
- You can track expenses by time, weekly, month, year etc..
- Setting monthly limits and we can update it later
- Customized email alerts when limit exceeds.

Disadvantages :

- User have entry every records manually
- The category divided may be blunder or messy
- Can’t able to customized user defined categories

CHAPTER 11
CONCLUSION

In this project, after making this application we assure that this application will help its users to manage the cost of their daily expenditure. It will guide them and make them aware about their daily expenses. It will prove to be helpful for the people who are frustrated with their daily budget management, irritated because of the amount of expenses and wish to manage money and to preserve the record of their daily cost which may be useful to change their way of spending money. In short, this application will help its users to overcome the wastage of money.

From this project, we are able to manage and keep tracking the daily expenses as well as income. While making this project, we gained a lot of experience of working as a team. We discovered various predicted and unpredicted problems and we enjoyed a lot solving them as a team. We adopted things like video tutorials, text tutorials, internet and learning materials to make our project complete.

CHAPTER 12

FUTURE SCOPE

- In further days, there will be mails and payment embedded with the app. Also, backup details will be recorded on cloud.
- Here user can define their own categories for expense type like food, clothing, rent and bills where they have to enter the money that has been spend .
- Alerts for paying dues and remainders to record input at particular user defined time.

CHAPTER 13

APPENDIX

SOURCE CODE:

```
from flask
import
Flask,
render_tem
plate,
request,
redirect,
url_for
             from flask_mail import Mail, Message
             from datetime import datetime
             from flask_cors import CORS, cross_origin
             import ibm_db
             import json
             import plotly
             import plotly.graph_objs as go
             import pandas as pd
             from flask import send_file
             from io import BytesIO
             import matplotlib.pyplot as plt
             import numpy as np
             import base64
             from PIL import Image
             import time
```

```python
import atexit
from datetime import datetime
from apscheduler.schedulers.background import BackgroundScheduler
app = Flask(__name__, template_folder='templates')
app.config['SECRET_KEY'] = 'top-secret!'
app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'apikey'
app.config['MAIL_PASSWORD'] = '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'
app.config['MAIL_DEFAULT_SENDER'] = 'nunnaaarthi@gmail.com'
mail = Mail(app)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'
# GLobal variables
EMAIL = ''
USERID = ''
print()
try:
    conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=54a2f15b-5c0f-46df-8954-
7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32733;Security=SSL;SSLServerCertificate
=DigiCertGlobalRootCA.crt;UID=nlg66799;PWD=CXtQLAGZ06fD0fhC;", "", "")
except Exception as e:
    print(e)
# FUNCTIONS INTERACTING WITH DB #
print('hello')
def fetch_walletamount():
    sql = 'SELECT WALLET FROM PETA_USER WHERE EMAIL=?'
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, EMAIL)
    ibm_db.execute(stmt)
    user = ibm_db.fetch_assoc(stmt)
    # print(user['WALLET'])
    return user['WALLET']  # returns int
def fetch_categories():
    sql = 'SELECT * FROM PETA_CATEGORY WHERE USERID = ?'
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, USERID)
    ibm_db.execute(stmt)
    categories = []
    while ibm_db.fetch_row(stmt) != False:
        categories.append([ibm_db.result(stmt, "CATEGORYID"),
                           ibm_db.result(stmt, "CATEGORY_NAME")])
    sql = 'SELECT * FROM PETA_CATEGORY WHERE USERID IS NULL'
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    while ibm_db.fetch_row(stmt) != False:
        categories.append([ibm_db.result(stmt, "CATEGORYID"),
                           ibm_db.result(stmt, "CATEGORY_NAME")])
    # print(categories)
    return categories  # returns list
def fetch_userID():
    sql = 'SELECT USERID FROM PETA_USER WHERE EMAIL=?'
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, EMAIL)
    ibm_db.execute(stmt)
    user = ibm_db.fetch_assoc(stmt)
    # print(user['USERID'])
    return user['USERID']  # returns int
def fetch_groups():
    sql = 'SELECT * FROM PETA_GROUPS'
    stmt = ibm_db.exec_immediate(conn, sql)
    groups = []
    while ibm_db.fetch_row(stmt) != False:
        groups.append([ibm_db.result(stmt, "GROUPID"),
                      ibm_db.result(stmt, "GROUPNAME")])
    # print(groups)
    return groups  # returns list
def fetch_expenses():
    sql = 'SELECT * FROM PETA_EXPENSE where USERID = ' + str(USERID)
    # print(sql)
    stmt = ibm_db.exec_immediate(conn, sql)
    expenses = []
    while ibm_db.fetch_row(stmt):
        category_id = ibm_db.result(stmt, "CATEGORYID")
        category_id = str(category_id)
        sql2 = "SELECT * FROM PETA_CATEGORY WHERE CATEGORYID = " + category_id
        stmt2 = ibm_db.exec_immediate(conn, sql2)
        category_name = ""
        while ibm_db.fetch_row(stmt2) != False:
            category_name = ibm_db.result(stmt2, "CATEGORY_NAME")
        expenses.append([ibm_db.result(stmt, "EXPENSE_AMOUNT"), ibm_db.result(
            stmt, "DATE"), ibm_db.result(stmt, "DESCRIPTION"), category_name])
    # print(expenses)
    return expenses
def fetch_rec_expenses_cron():
    sql = 'SELECT * FROM PETA_REC_EXPENSES;'
    stmt = ibm_db.exec_immediate(conn, sql)
    rec_expenses = []
    while ibm_db.fetch_row(stmt) != False:
        amt = ibm_db.result(stmt, "AMOUNT")
        amt = str(amt)
        description = ibm_db.result(stmt, "DESCRIPTION")
        userid = ibm_db.result(stmt, "USERID")
        date = ibm_db.result(stmt, "RECDATE")
        rec_expenses.append([amt, description, date, userid])
    # print(rec_expenses)
    return rec_expenses
def fetch_rec_expenses():
    sql = 'SELECT * FROM PETA_REC_EXPENSES WHERE USERID = ' + str(USERID)
    stmt = ibm_db.exec_immediate(conn, sql)
    rec_expenses = []
    while ibm_db.fetch_row(stmt) != False:
        amt = ibm_db.result(stmt, "AMOUNT")
        amt = str(amt)
        description = ibm_db.result(stmt, "DESCRIPTION")
        userid = ibm_db.result(stmt, "USERID")
        date = ibm_db.result(stmt, "RECDATE")
        rec_expenses.append([amt, description, date, userid])
    # print(rec_expenses)
    return rec_expenses
def fetch_limits():
    now = datetime.now()
    year = now.year
    limits = [0 for i in range(12)]
    sql = 'SELECT LIMITAMOUNT, LIMITMONTH FROM PETA_LIMIT WHERE USERID = ? AND LIMITYEAR = ?'
    statement = execute_sql(sql, USERID, year)
    while ibm_db.fetch_row(statement):
        limit_amount = int(ibm_db.result(statement, 'LIMITAMOUNT'))
        limit_month = int(ibm_db.result(statement, 'LIMITMONTH'))
        limits[limit_month] = limit_amount
    return limits
# HELPER FUNCTIONS #
def fetch_latest_expenses(expenses):
    # must return expenses of last month
    latest_month = datetime.today().month
    latest_expenses = []
    for exp in expenses:
        if exp[1].month == latest_month:
            latest_expenses.append(exp)
    return latest_expenses
def fetch_monthly_expenses(expenses):
    latest_year = datetime.today().year
    monthly_expenses = {}
    for month in range(1, 13):
        monthly_expenses[month] = 0
    for exp in expenses:
        if exp[1].year == latest_year:
            monthly_expenses[exp[1].month] += exp[0]
    return monthly_expenses.values()
def draw_graph1(expenses):
    # TOTAL EXPENSE / DAY OF MONTH
    # x-axis: day , y-axis: expense/day
    latest_expenses = fetch_latest_expenses(expenses)
    mp = {}
    for day in range(1, 31):
        mp[day] = 0
    for exp in latest_expenses:
        mp[exp[1].day] += exp[0]
    x = mp.keys()
    y = mp.values()
    # print(mp)
    plt.figure()
    plt.title('Expense recorded over the past month')
    plt.plot(x, y)
    plt.xlabel('Day of the month')
    plt.ylabel('Recorded expense')
    plt.xlim(1, 32)
    buffer = BytesIO()
    plt.savefig(buffer, format='png')
    encoded_img_data = base64.b64encode(buffer.getvalue())
    return encoded_img_data
def draw_graph2(expenses, limits):
    # limit/month vs expense/month -> 2 line graphs
    monthly_expenses = fetch_monthly_expenses(expenses)
    x = range(1, 13)
    y1 = limits
    y2 = monthly_expenses
    plt.figure()
    plt.title('Month-wise comparison of limit and expense')
    plt.plot(x, y1, label="Limit/month")
    plt.plot(x, y2, label="Expenses/month")
    plt.xlabel('Month')
    plt.legend()
    buffer = BytesIO()
    plt.savefig(buffer, format='png')
    encoded_img_data = base64.b64encode(buffer.getvalue())
    return encoded_img_data
# finds the category id that matches that of the recurring expense category
def fetch_recurring_category_id():
    categories = fetch_categories()
    for category in categories:
        p = ''
        for i in category[1]:
            if (i == ' '):
                break
            p += i
        category[1] = p
    print(categories)
    categoryid = -1
    for category in categories:
        if category[1] == 'recurring':
            categoryid = category[0]
    print(categoryid)
    return categoryid
# cron to autodeduct the expenses each day
def auto_renew():
    global USERID
    # print(time.strftime("%A, %d. %B %Y %I:%M:%S %p"))
    rec_expenses = fetch_rec_expenses_cron()
    print(rec_expenses)
    current_day = time.strftime("%d")
    print(current_day)
    for expense in rec_expenses:
        here = str(expense[2])
        here = here.split('-')
        here = here[2]
        print(here)
        if (here == current_day):
            sql = "INSERT INTO PETA_EXPENSE(USERID, EXPENSE_AMOUNT, CATEGORYID, DESCRIPTION, DATE)
VALUES(?,?,?,?,?);"
            USERID = str(expense[3])
            categoryid = fetch_recurring_category_id()
            print(categoryid)
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, expense[3])
            ibm_db.bind_param(stmt, 2, expense[0])
            ibm_db.bind_param(stmt, 3, categoryid)
            ibm_db.bind_param(stmt, 4, expense[1])
            d3 = time.strftime("%Y-%m-%d")
            ibm_db.bind_param(stmt, 5, d3)
            print(d3, categoryid, expense[0],
                  expense[1], expense[2], expense[3])
            ibm_db.execute(stmt)
            check_monthly_limit(datetime.now().month, datetime.now().year)
            # print(here, d3, expense[0], expense[1], expense[2])
            sql = "UPDATE PETA_USER SET WALLET = WALLET - ? WHERE USERID = ?"
            statement = ibm_db.prepare(conn, sql)
            print(USERID)
            ibm_db.bind_param(statement, 1, expense[0])
            ibm_db.bind_param(statement, 2, expense[3])
            print("deducted")
            ibm_db.execute(statement)
# caller code for the cron
scheduler = BackgroundScheduler()
scheduler.add_job(func=auto_renew, trigger="interval", seconds=3600 * 24)
print('hello')
# END POINTS #
scheduler.start()
print('hello')
atexit.register(lambda: scheduler.shutdown())
@app.route('/', methods=['GET', 'POST'])
@cross_origin()
def registration():
    global EMAIL
    print("hello")
    if request.method == 'GET':
        return render_template('registration.html')
    if request.method == 'POST':
        email = request.form['email']
        EMAIL = email
        password = request.form['password']
        wallet = request.form['wallet']
        sql = "INSERT INTO PETA_USER(EMAIL,PASSWORD,WALLET) VALUES(?,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.bind_param(stmt, 3, wallet)
        print(stmt)
        ibm_db.execute(stmt)
        # msg = Message('Registration Verfication',recipients=[EMAIL])
        # msg.body = ('Congratulations! Welcome user!')
        # msg.html = ('<h1>Registration Verfication</h1>'
        #             '<p>Congratulations! Welcome user!'
        #             '<b>PETA</b>!</p>')
        # mail.send(msg)
        EMAIL = email
    return redirect(url_for('dashboard'))
@app.route('/login', methods=['GET', 'POST'])
def login():
    global EMAIL
    print("login")
    if request.method == 'POST':
        email = request.form['email']
        EMAIL = email
        print(EMAIL)
        password = request.form['password']
        sql = "SELECT * FROM PETA_USER WHERE email=? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        if account:
            return redirect(url_for('dashboard'))
        else:
            return redirect(url_for('login'))
    elif request.method == 'GET':
        return render_template('login.html')
@app.route('/logout', methods=['GET'])
```

```
def logout():
    if request.method == 'GET':
        global USERID
        global EMAIL
        USERID = ""
        EMAIL = ""
        return redirect(url_for('login'))
@app.route('/dashboard', methods=['GET'])
def dashboard():
    global USERID
    global EMAIL
    print("dashboard")
    if USERID == '' and EMAIL == '':
        print("null email")
        return render_template('login.html')
    elif USERID == '':
        USERID = fetch_userID()
        print(USERID)
    sql = "SELECT EXPENSEID, EXPENSE_AMOUNT, DESCRIPTION, CATEGORY_NAME, DATE FROM PETA_EXPENSE,
PETA_CATEGORY WHERE PETA_EXPENSE.USERID = ? AND PETA_EXPENSE.CATEGORYID = PETA_CATEGORY.CATEGORYID"
    statement = execute_sql(sql, USERID)
    expenses = []
    while True:
        expense = ibm_db.fetch_assoc(statement)
        if expense:
            expenses.append(expense)
        else:
            break
    wallet = fetch_walletamount()
    return render_template('dashboard.html', expenses=expenses, wallet=wallet, email=EMAIL)
@app.route('/updatebalance', methods=['GET', 'POST'])
def update_balance():
    if request.method == 'GET':
        wallet = fetch_walletamount()
        return render_template('updatebalance.html', wallet=wallet)
    elif request.method == 'POST':
        global EMAIL
        global USERID
        if EMAIL == '':
            return render_template('login.html', msg='Login before proceeding')
        if (USERID == ''):
            # get user using email
            USERID = fetch_userID()
        new_balance = request.form['balanceupdated']
        sql = 'UPDATE PETA_USER SET WALLET = ? WHERE USERID = ?'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, new_balance)
        ibm_db.bind_param(stmt, 2, USERID)
        ibm_db.execute(stmt)
        return redirect(url_for('dashboard'))
@app.route('/addcategory', methods=['GET', 'POST'])
def add_category():
    if request.method == 'GET':
        # categories = fetch_categories()
        return render_template('addcategory.html')
    elif request.method == 'POST':
        categoryname = request.form['category']
        sql = 'INSERT INTO PETA_CATEGORY(CATEGORY_NAME, USERID) VALUES(?,?)'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, categoryname)
        ibm_db.bind_param(stmt, 2, USERID)
        ibm_db.execute(stmt)
        return redirect(url_for('dashboard'))
@app.route('/addgroup', methods=['POST'])
def add_group():
    if request.method == 'POST':
        if USERID == '':
            return render_template('login.html', msg='Login before proceeding')
        sql = "INSERT INTO PETA_GROUPS(GROUPNAME, USERID) VALUES(?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, request.form['groupname'])
        ibm_db.bind_param(stmt, 2, USERID)
        ibm_db.execute(stmt)
        group_info = {}
        sql = "SELECT * FROM PETA_GROUPS WHERE GROUPNAME=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, request.form['groupname'])
        ibm_db.execute(stmt)
        group_info = ibm_db.fetch_assoc(stmt)
        return {"groupID": group_info['GROUPID'], 'groupname': group_info['GROUPNAME']}
@app.route('/addexpense', methods=['GET', 'POST'])
def add_expense():
    if request.method == 'GET':
        groups = fetch_groups()
        categories = fetch_categories()
        if len(categories) == 0:
            return redirect(url_for('add_category'))
        return render_template('addexpense.html', categories=categories, groups=groups)
    elif request.method == 'POST':
        global EMAIL
        global USERID
        if EMAIL == '':
            return render_template('login.html', msg='Login before proceeding')
        if (USERID == ''):
            # get user using email
            USERID = fetch_userID()
        amount_spent = request.form['amountspent']
        category_id = request.form.get('category')
        description = request.form.get('description')
        date = request.form['date']
        groupid = request.form.get('group')
        groupid = None if groupid == '' else groupid
        print(amount_spent, category_id, description, date, groupid, USERID)
        sql = "INSERT INTO PETA_EXPENSE(USERID, EXPENSE_AMOUNT, CATEGORYID, GROUPID, DESCRIPTION, DATE)
VALUES(?,?,?,?,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, USERID)
        ibm_db.bind_param(stmt, 2, amount_spent)
        ibm_db.bind_param(stmt, 3, category_id)
        ibm_db.bind_param(stmt, 4, groupid)
        ibm_db.bind_param(stmt, 5, description)
        ibm_db.bind_param(stmt, 6, date)
        ibm_db.execute(stmt)
        print(date, amount_spent, category_id)
        sql = "UPDATE PETA_USER SET WALLET = WALLET - ? WHERE USERID = ?"
        statement = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(statement, 1, amount_spent)
        ibm_db.bind_param(statement, 2, USERID)
        ibm_db.execute(statement)
        return redirect(url_for('dashboard'))
@app.route('/viewrecurring', methods=['GET'])
def viewrecurring():
    global USERID
    global EMAIL
    print("viewrecurring")
    if USERID == '' and EMAIL == '':
        print("null email")
        return render_template('login.html')
    elif USERID == '':
        USERID = fetch_userID()
        # print(USERID)
    print(USERID)
    expenses = fetch_rec_expenses()
    wallet = fetch_walletamount()
    return render_template('viewrecurring.html', expenses=expenses, wallet=wallet, email=EMAIL)
@app.route('/recurringexpense', methods=['GET', 'POST'])
def recurring_expense():
    global USERID, EMAIL
    if request.method == 'GET':
        groups = fetch_groups()
        categories = fetch_categories()
        if len(categories) == 0:
            return redirect(url_for('add_category'))
        USERID = fetch_userID()
        # check if user has added a category for recurring category, if not redirect and ask her to
        recur_id = fetch_recurring_category_id()
        if (recur_id == -1):
            return (redirect(url_for('add_category')))
        return render_template('recurringexpense.html', categories=categories, groups=groups)
    elif request.method == 'POST':
        if EMAIL == '':
            return render_template('login.html', msg='Login before proceeding')
        if (USERID == ''):
            # get user using email
            USERID = fetch_userID()
            # check if user has added a category for recurring category, if not redirect and ask her to
            recur_id = fetch_recurring_category_id()
            if (recur_id == -1):
                return (redirect(url_for('add_category')))
        amount_spent = request.form['amountspent']
        category_id = request.form.get('category')
        description = request.form['description']
        date = request.form['date']
        # months = request.form['autorenewals']
        # groupid = request.form.get('group')
        print("recurring : ")
        print(amount_spent, description, date, USERID)
        sql = "INSERT INTO PETA_REC_EXPENSES(AMOUNT, RECDATE, USERID, DESCRIPTION) VALUES (?,?,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, amount_spent)
        ibm_db.bind_param(stmt, 2, date)
        ibm_db.bind_param(stmt, 3, USERID)
        ibm_db.bind_param(stmt, 4, description)
        ibm_db.execute(stmt)
        sql = "INSERT INTO PETA_EXPENSE(USERID, EXPENSE_AMOUNT, CATEGORYID, DESCRIPTION, DATE)
VALUES(?,?,?,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, USERID)
        ibm_db.bind_param(stmt, 2, amount_spent)
        ibm_db.bind_param(stmt, 3, category_id)
        ibm_db.bind_param(stmt, 4, description)
        ibm_db.bind_param(stmt, 5, date)
        ibm_db.execute(stmt)
        sql = "UPDATE PETA_USER SET WALLET = WALLET - ? WHERE USERID = ?;"
        statement = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(statement, 1, amount_spent)
        ibm_db.bind_param(statement, 2, USERID)
        ibm_db.execute(statement)
        return redirect(url_for('dashboard'))
@app.route('/removerecurring', methods=['POST'])
def remove_recurring():
    print("remove recurring")
    if request.method == 'POST':
        global EMAIL
        global USERID
        if EMAIL == '':
            return render_template('login.html', msg='Login before proceeding')
        if (USERID == ''):
            # get user using email
            USERID = fetch_userID()
        description = request.form['description']
        print(description, USERID)
        sql = 'DELETE FROM PETA_REC_EXPENSES WHERE USERID = ? AND DESCRIPTION = ?;'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, USERID)
        ibm_db.bind_param(stmt, 2, description)
        ibm_db.execute(stmt)
        return redirect(url_for('dashboard'))
@app.route('/analysis', methods=['GET', 'POST'])
def analyse():
    if request.method == 'GET':
        expenses = fetch_expenses()
        limits = fetch_limits()
        graph1 = draw_graph1(expenses=expenses)
        graph2 = draw_graph2(expenses=expenses, limits=limits)
        return render_template("analysis.html", img_data1=graph1.decode('utf-8'),
img_data2=graph2.decode('utf-8'))
    elif request.method == 'POST':
        return render_template('analysis.html')
def execute_sql(sql, *args):
    stmt = ibm_db.prepare(conn, sql)
    for i, arg in enumerate(args):
        ibm_db.bind_param(stmt, i + 1, arg)
    ibm_db.execute(stmt)
    return stmt
def check_monthly_limit(month, year):
    sql = 'SELECT SUM(EXPENSE_AMOUNT) FROM PETA_EXPENSE WHERE USERID = ? AND MONTH(DATE) = ? AND
YEAR(DATE) = ?'
    statement = execute_sql(sql, USERID, month, year)
    amt_spent = ibm_db.fetch_tuple(statement)
    sql = 'SELECT LIMITAMOUNT FROM PETA_LIMIT WHERE USERID = ? AND LIMITMONTH = ? AND LIMITYEAR = ?'
    statement = execute_sql(sql, USERID, month, year)
    monthly_limit = ibm_db.fetch_tuple(statement)
    if amt_spent and monthly_limit and int(amt_spent[0]) > int(monthly_limit[0]):
        diff = int(amt_spent[0]) - int(monthly_limit[0])
        msg = Message('Monthly limit exceeded', recipients=[EMAIL])
        msg.body = (
            f'Monthly limit exceeded by {diff} for the month of {month}, {year}')
        mail.send(msg)
def update_monthly_limit(monthly_limit, month, year):
    sql = 'SELECT LIMITAMOUNT FROM PETA_LIMIT WHERE USERID = ? AND LIMITMONTH = ? AND LIMITYEAR = ?'
    statement = execute_sql(sql, USERID, month, year)
    if ibm_db.fetch_row(statement):
        sql = 'UPDATE PETA_LIMIT SET LIMITAMOUNT = ? WHERE USERID = ? AND LIMITMONTH = ? AND LIMITYEAR =
?'
        execute_sql(sql, monthly_limit, USERID, month, year)
    else:
        sql = 'INSERT INTO PETA_LIMIT VALUES(?, ?, ?, ?)'
        execute_sql(sql, USERID, monthly_limit, month, year)
    check_monthly_limit(month, year)
@app.route('/setmonthlylimit', methods=['GET', 'POST'])
def set_monthly_limit():
    if request.method == 'GET':
        return render_template('setmonthlylimit.html')
    elif request.method == 'POST':
        new_monthly_limit = request.form['monthlylimit']
        now = datetime.now()
        update_monthly_limit(new_monthly_limit, now.month, now.year)
        return redirect(url_for('dashboard'))
@app.route('/modifyexpense', methods=['GET', 'POST'])
def modify_expense():
    if request.method == 'GET':
        expenseid = request.args.get('expenseid')
        sql = "SELECT * FROM PETA_EXPENSE WHERE EXPENSEID = ?"
        statement = execute_sql(sql, expenseid)
        expense = ibm_db.fetch_assoc(statement)
        categories = fetch_categories()
        groups = fetch_groups()
        return render_template('modifyexpense.html', expense=expense, categories=categories,
groups=groups)
    elif request.method == 'POST':
        amount_spent = request.form['amountspent']
        category_id = request.form.get('category')
        description = request.form['description']
        date = request.form['date']
        groupid = request.form.get('group')
        expenseid = request.form['expenseid']
        old_amount_spent = request.form['oldamountspent']
        sql = "UPDATE PETA_EXPENSE SET EXPENSE_AMOUNT = ?, CATEGORYID = ?, GROUPID = ?, DESCRIPTION = ?,
DATE = ? WHERE EXPENSEID = ?"
        execute_sql(sql, amount_spent, category_id,
                    groupid, description, date, expenseid)
        sql = "UPDATE PETA_USER SET WALLET = WALLET + ?"
        execute_sql(sql, float(old_amount_spent) - float(amount_spent))
        return redirect(url_for('dashboard'))
def fetch_goals():
    sql = 'SELECT * FROM PETA_GOALS WHERE USERID = ?'
    statement = execute_sql(sql, USERID)
    goals = []
    while True:
        goal = ibm_db.fetch_tuple(statement)
        if goal:
            goals.append(goal[2:])
        else:
            break
    print(goals)
    return goals
@app.route('/rewards', methods=['GET'])
def rewards_and_goals():
    goals = fetch_goals()
    return render_template('rewards.html', goals=goals)
@app.route('/addgoal', methods=['GET', 'POST'])
def add_goal():
    if request.method == 'GET':
        return render_template('addgoal.html')
    elif request.method == 'POST':
        goal_amount = request.form['goal_amount']
        date = request.form['date']
        reward = request.form['reward']
        sql = "INSERT INTO PETA_GOALS(USERID, GOAL_AMOUNT, DATE, REWARD) VALUES(?, ?, ?, ?)"
        execute_sql(sql, USERID, goal_amount, date, reward)
        return redirect(url_for('dashboard'))
def check_goals():
    sql = "SELECT A.GOALID, A.USERID, A.GOAL_AMOUNT, A.DATE, A.REWARD, B.WALLET FROM PETA_GOALS AS A, 
PETA_USER AS B WHERE A.USERID = B.USERID"
    statement = execute_sql(sql)
    now = datetime.now()
    while True:
        row = ibm_db.fetch_assoc(statement)
        if not row:
            break
        if row['DATE'] == now:
            if row['GOAL_AMOUNT'] <= row['WALLET']:
                msg = Message('Goal achieved!', recipients=[EMAIL])
                msg.body = (
                    f'You are eligible for your reward:\n{row["REWARD"]}')
                mail.send(msg)
            else:
                msg = Message('Goal limit exceeded', recipients=[EMAIL])
                msg.body = (
                    f'You are not eligible for the reward:\n{row["REWARD"]}\nBetter luck next time!')
                mail.send(msg)
            sql = "DELETE FROM PETA_GOALS WHERE GOALID = ?"
            execute_sql(sql, row['GOALID'])
scheduler.add_job(func=check_goals, trigger="interval", seconds=3600 * 24)
if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-18101-1668782604

DEMO LINK:

https://drive.google.com/file/d/1e-1roJ3OmR27sYgrjUVBwwbV3lYKC7gw/view?usp=drivesdk