PERSONAL EXPENSE TRACKER

PROJECT REPORT

Submitted by

PARTHIBAN K (19EUEC097)

PAVITHRA SK (19EUEC098)

POOVITHA G (19EUEC099)

PRANAV SUNDAR (19EUEC100)

in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

COIMBATORE

(An Autonomous Institution)

ANNA UNIVERSITY: CHENNAI

MAY 2022

ACKNOWLEDGEMENT

We express our sincere thanks to the management and **Dr.J.JANET, M.E.,Ph.D.,** Principal, Sri Krishna College of Engineering and Technology, Coimbatore for providing us the facilities to carry out this project work.

We are highly indebt to **Dr.S. SASIPRIYA M.E.,Ph.D.,** Head of Electronics and Communication Engineering for her continuous evaluation, valuable suggestions and comments given during the course of the project work.

We express our deep sense of gratitude to our guide**,** Professors in the department of Electronics and Communication Engineering for her valuable advice, guidance and support during the course of our project work.

By this, we express our heartfelt sense of gratitude and thanks to our beloved parents, family and friends who have all helped in collecting the resources and materials needed for this project and for their support during the study and implementation this project.

TABLE OF CONTENTS

ABSTRACT

The Expense Tracker is developed to manage the daily expenses in a more efficient and manageable way. Tracking regular expense is a key factor to maintain a budget. People often track expense using pen and paper method or take notes in a mobilephone or a computer. These processes of storing expense require further computations and processing for these data to be used as a trackable record. In this work, we are proposing an automated system to store and calculate these data. This application extracts thetextual information from the receiptsand saves the amount and description for further processing.

It also monitors user's income by tracking thereceived SMS's from the user's saving accounts. By calculating income and expense it produces the user's balance in monthly and yearly basis. Overall, this is a smart automated solutionfor tracking expense.

1.INTRODUCTION

1.1 PROJECT OVERVIEW:

The overview of the project was simple where there is a process for collecting receipts and organize them once per month. From the other tracking system, the information can be tracked and they have one common thing in their mind about tracking.

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 vendingmachine isn’t worthtracking — although those little

expenses can add up amazingly fast.This project will requestthe clients to add their expenses and in view of their costs, wallet status will be refreshed which will be noticeable to the client.

i. The user interacts with the application.

ii. Application will ask users
toadd their expensesand based on their expenseswallet balance will be updated which will be visibleto the user.

iii. Also, users can get ananalysis of their expenditure in graphical forms.

iv. Work with IBM Cloud CLI, Docker CLI, Sendgrid

v. Create UI to Interactwith the application.

1.2 PURPOSE:

The main purpose of our project to help people by tracking and alters to linits their budget. In this application, user can provide the income to calculate his total expenses per day and the result will be stored for each user. The expense tracker will help any organization to deal with all their expenses mo.A personalfinance app will not only help
you with budgeting and accounting but also give you helpfulinsights about financial management.

2. LITERATURE SURVEY:

2.1 EXISTING PROBLEM :

In existing system, most of the applications are used only for personal use and

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. 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 persons 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.

2.2 REFERENCES

EXPENSE TRACKER USINGSTATISTICAL ANALYSIS

Authors: Muskaan Sharma,Ayush Bansal, Dr. Raju Ranjan,Shivam Sethi

Description:

In this paper, an approach has beenproposed on how to efficiently manage house-old budget. This application will allow users to keep track of their expenses.This novel expense trackeruses statistical analysis which is going to keepa track of your expenses and would even give you results accordingly.

Year: 2021

Technologies: Java

STUDENT EXPENSETRACKING APPLICATION

Authors: Saumya Dubey , PragyaDubey , Rigved RishabhKumar , Aaisha Khatoon

Description:

This is an android application which is used to track the daily expenses of a Student. It is likea digital diarythat keeps a recordof expenses done by a student.The Application keeps track of money spent and the earnings of both of the students on a Day-to-day basis. It also has the feature that it gives warning messages if we are Exceeding our expenses and hence, we can limit our expenses and avoid Overspending. If you spend less money than the daily expense allowedamount, the Money left after spending is added into the user’s savings.

Year: 2022
Technologies: Java

SPENDING TRACKER :
A Smart Approach to Track Daily Expenses

Authors:

UP Singh, AK Gupta,Dr. B. Balamurugan

Description:

In this paper, a Java GUI based application was proposed to assure that it will Help its users to manage the cost of their daily expenditure. It will guide themand aware them of their daily expenses. The proposed design contained the basic modules
for Adding and viewingexpenses, managing expense categories.
Supports CRUD Operations on expense data.
Year: 2021
Technologies: Java

SAWANT-EXPENSE TRACKER

Authors: Atiya Kazi, Praphulla S. Kherade , Raj S. Vilankar , Parag M

Description:

In this approach, the application keeps track of the Incomeand Expenses of both users on a day-to-day basis.This application takes the income of a user andmanages its daily expenses so that the user can save money. If you exceed the daily expense allowed amount it will give you a warning, so that you don't spend much and that specific day. If you spend less money than the daily expense allowed amount, the money left after spending is added into the user's savings. The application generates report of the expensesof each end of the month.
Year: 2021
Technologies: Java

BUDGET ESTIMATORANDROID APPLICATION

Authors: Namita Jagtap, PriyankaJoshi, Aditya Kamble

Description:

The system known as Budget Estimator is designed to manage the application user ‘s daily expenses in a more efficient and manageable way. This project is about mobile application Expenses system with geo-location tracking, based on the location of the user, it using GooglePlaces, to check, the availablestore inthe area, provides a notification for offers purpose, In term of security design, this system may implement a login authentication such as OTP message to yourmobile device, this function may bring more security confidence to user. To reduce manual calculations, we propose an application which is developed by android.This application allowsusers to maintain a digital automated diary.

Year: 2021

Technologies: Java

1.3 PROMBLEMSTATEMENT :

Who doesthe problem affect?	Investors, savers, big spenders, debtors ,shoppers, budgetconsciousconsumers.
What are the boundaries of the pr oblem?	Expense tracker for working individuals, students, commonpeople.
What is the issue?	To be vigilant aboutthe expensespent,increases fin ancial stress. Being indecisive about the finances may resultin less financial security and exceedthe budget.

When does thisissu eoccur?	When usingwrong budgeting techniques. When nottracking the expenses doesn’t helpyou to know the a mountthat is actually spent.

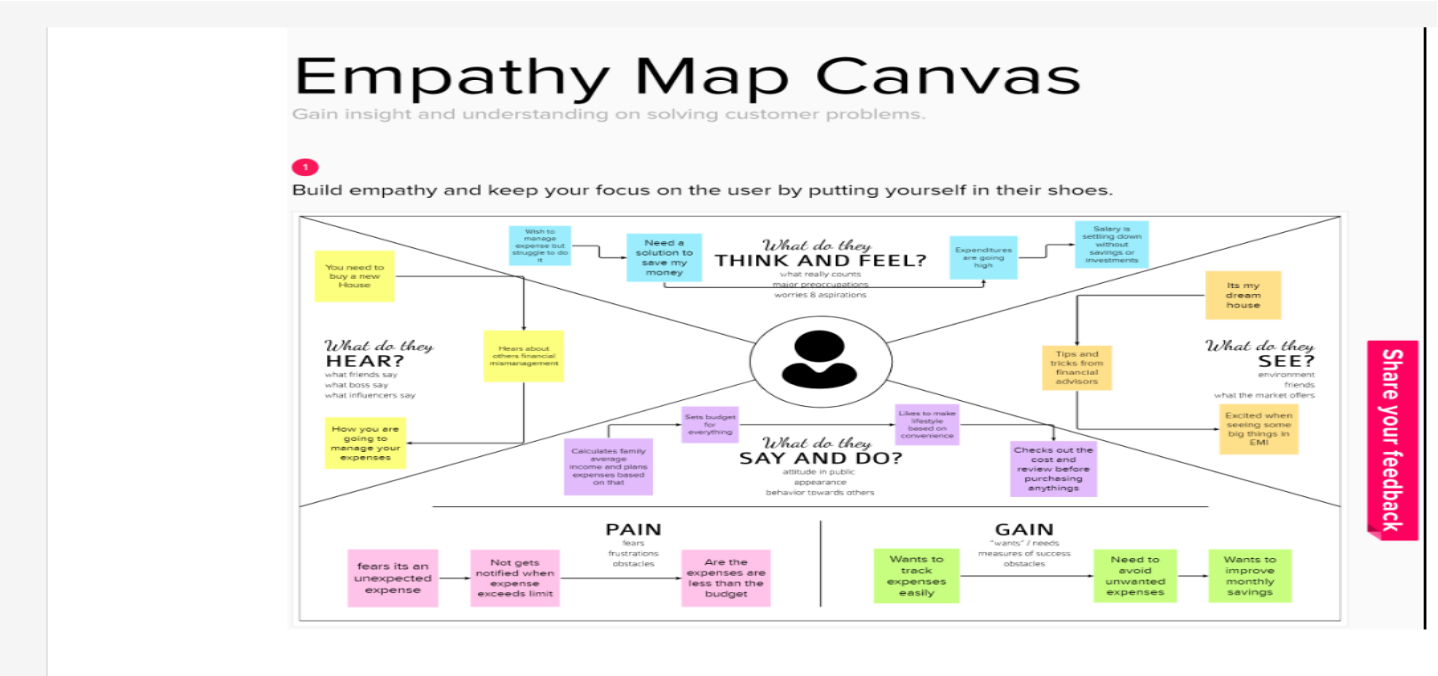
Where is the issueoccurring?	Working individuals who find it difficult to track theirexpenses
Why is it important thatwefix thep roblem?	Fixing this issue, brings accountability and helps to be intentional with the income by assignit to spending, saving and giving.Thisleads to fina ncial stability.

1. Abella, who is a shopholic,
   finds it hard to control her desire to shop. To stop her fromoverindulging
   in impulsive purchases, she needsto track her expenses and hold herselfaccountable.
2. John, who is interested to
   invest in stocks, finds itdifficult to figureout the expensethathe can spend on investing stocks. With the helpof expense tracking,he can easily plan out the expensesfor investing in an ef ficient way.
3. Akshay, is a high schoolstudent, who usuallygets a limited allowance from his parents.So tracki ng his expensesand good budgetingtechnique allows him to spend on his regularexpenses as well as on himself.

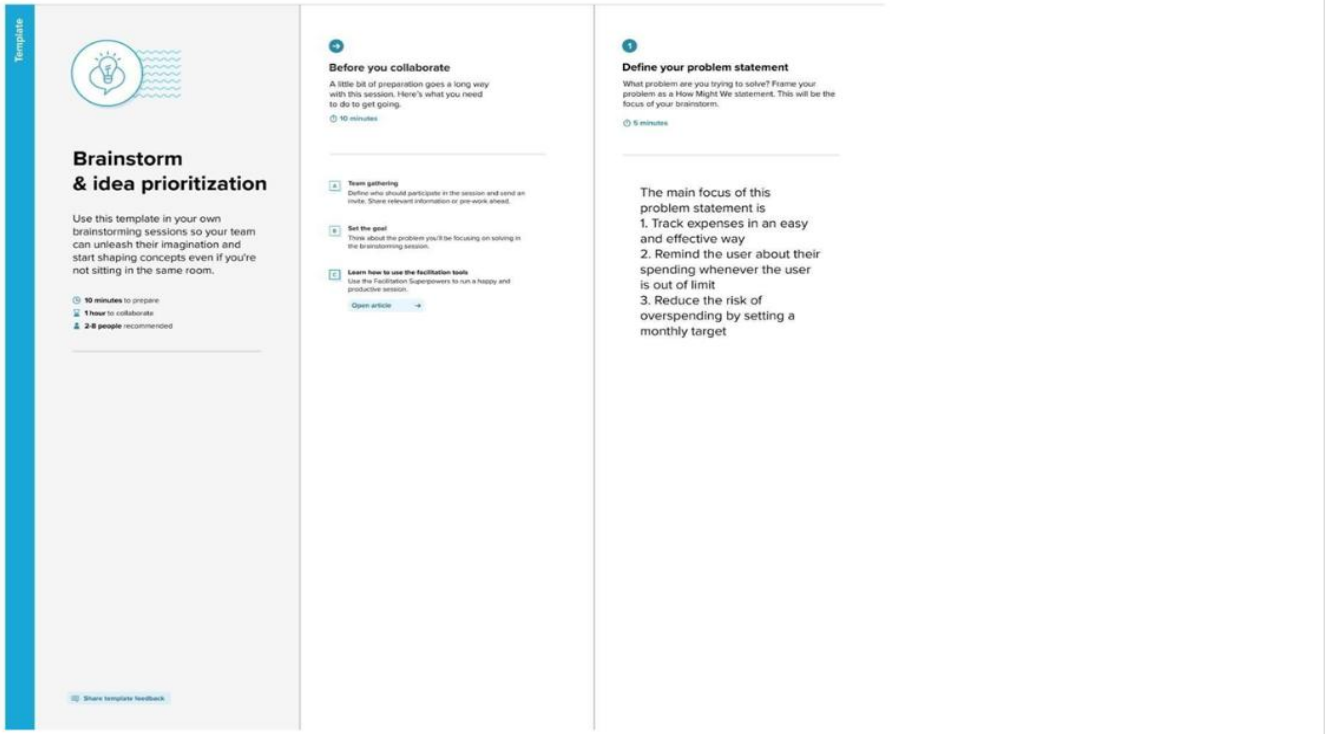
4. Udhay, who is a novice budgeter, finds it tedious to track and manage the expensesamongst his busy schedule. Prioritizing hisexpenses will help him to curtail his unnecessary expenditures.

3. IDEATION &PROPOSED SOLUTION :

3.1 EMPATHY MAP CANVAS



3.2 IDEATION &BRAINSTORMING:



Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Share template feedback

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

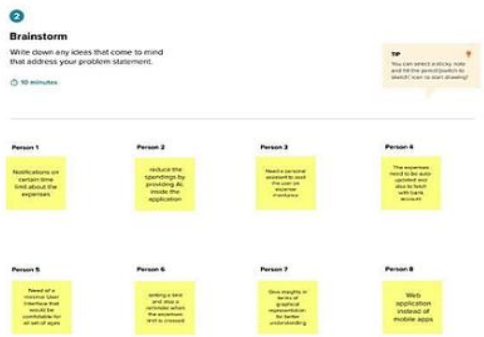
Open article →

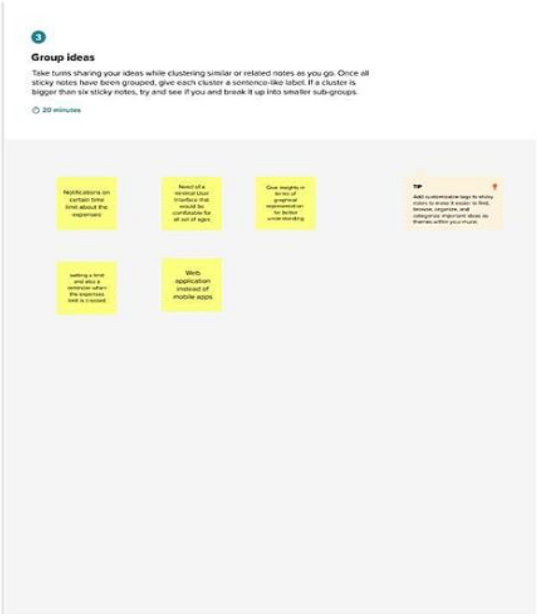
1 **Define your problem statement**

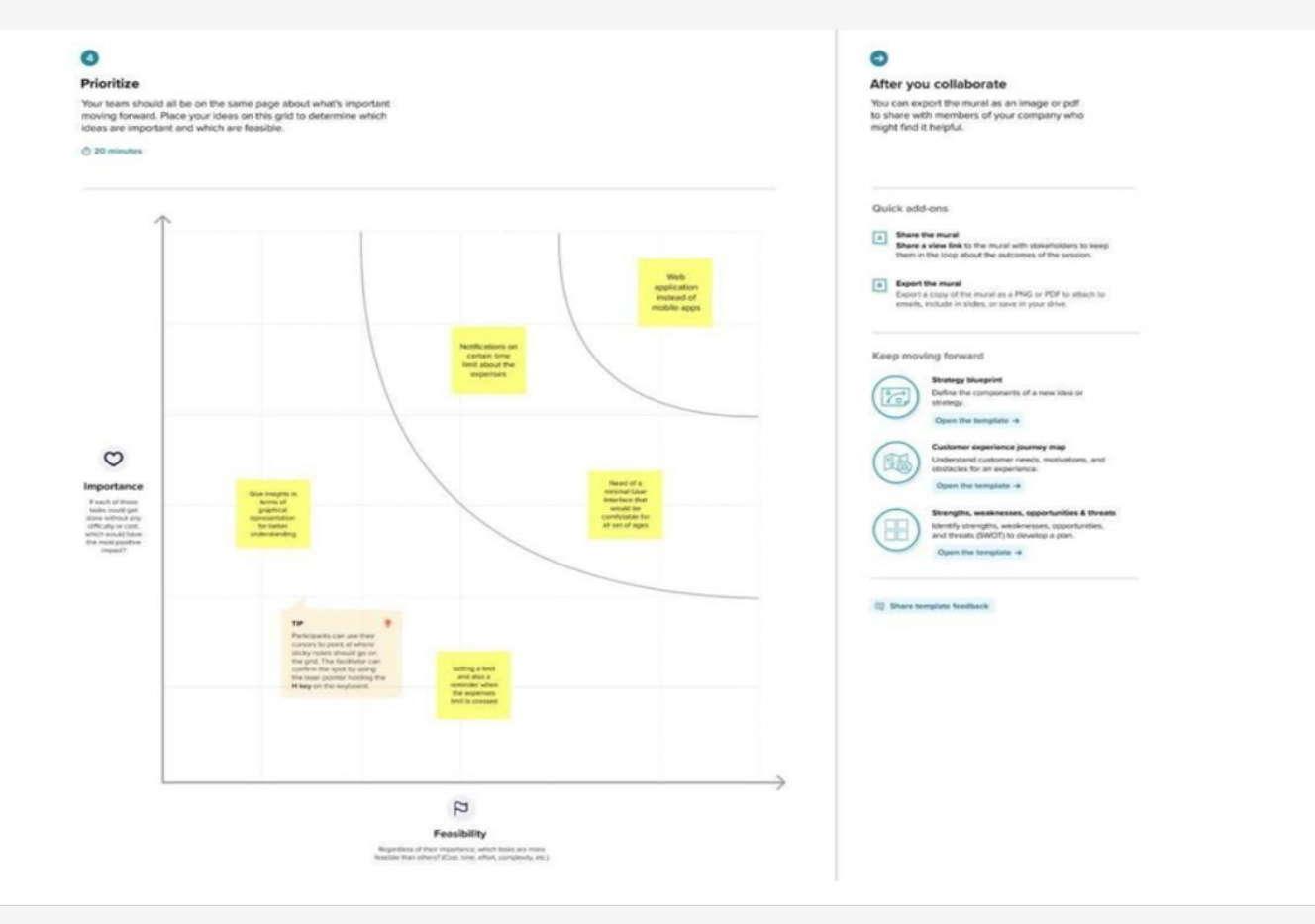
What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

The main focus of this problem statement is
1. Track expenses in an easy and effective way
2. Remind the user about their spending whenever the user is out of limit
3. Reduce the risk of overspending by setting a monthly target


2
Brainstorm
Write down any ideas that come to mind that address your problem statement.
10 minutes
TIP
Person 1
Notifications on certain time limit about the expenses
Person 2
reduce the spendings by providing AI inside the application
Person 3
Person 4
Person 5
Person 6
Person 7
Person 8
Web application instead of mobile apps


3
Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.
20 minutes
Notifications on certain time limit about the expenses
Web application instead of mobile apps
TIP

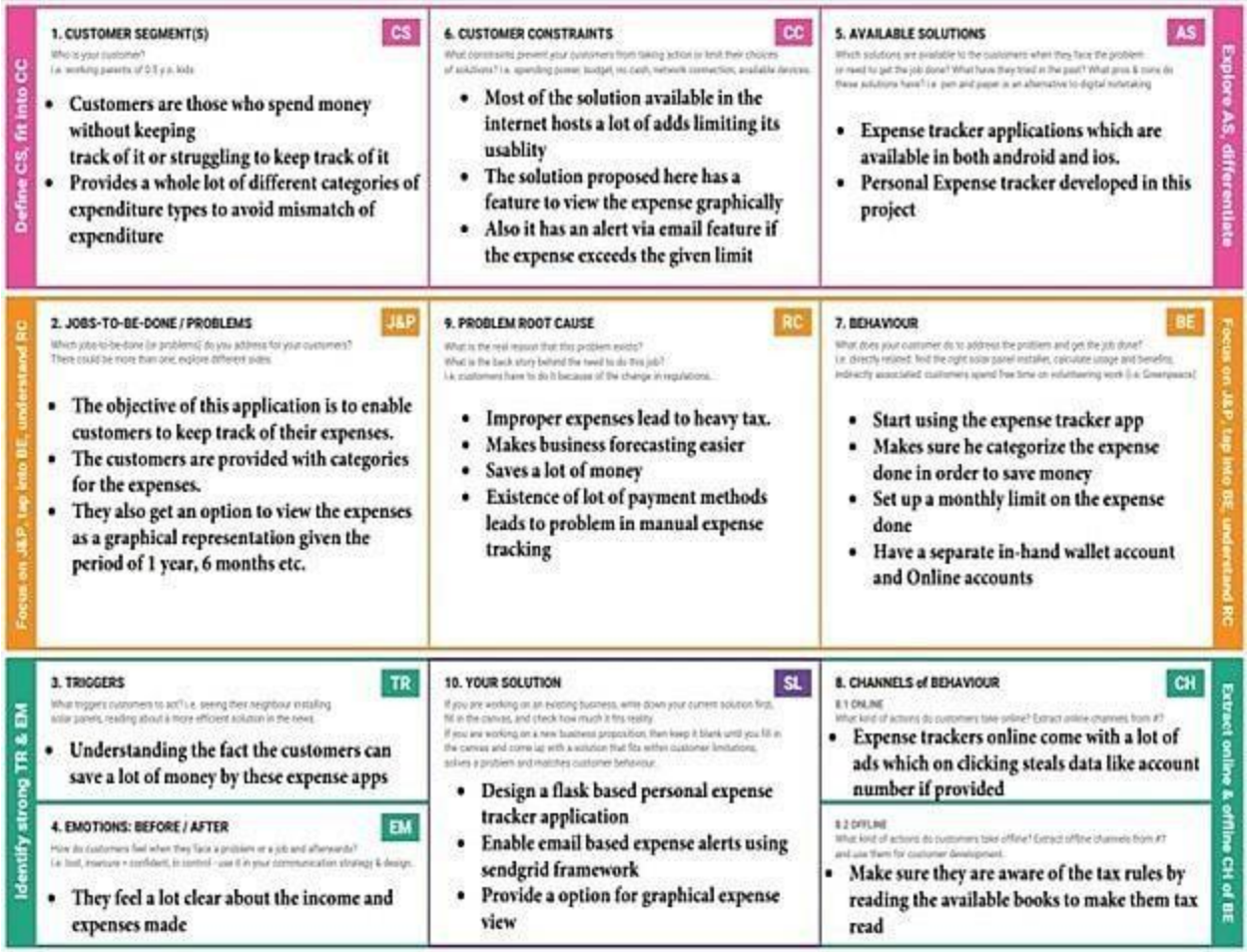


3.3 PROPOSED SOLUTION:

we propose this application to reduce manual calculations. Tracker application which will keep a track of

Income-Expense of a user on a day to day basis. The best organizations have a way of tracking and handling these reimbursements. This ideal practice guarantees that the expenses tracked are accurately and in a timely manner. Effective expense tracking and reporting to avoid conflict. As a project manager or business owner, you can set clear policies for the expense types and reimbursement limits to avoid misunderstandings are about costs. Tracking the project expenses by asking team members to provide receipts is helpful to avoid conflict and maintain compliance also. An excellent reporting mechanism is extremely helpful to support the amount to be reimbursed to your team and also invoicing to your customer. Tracking the amount of money spent on the projects is important to invoice customers and determine the cost & profitability analysis when your company is providing services to another company. On the other hand, expense tracking or internal project is important for cost and ROI calculation. Understanding how this money is being utilized across the project is such a significant issue. The consequence for not properly tracking and reporting project expenses may lead to a budgetary issues.

3.4 PROBLEM SOLUTION FIT:



3. REQUIREMENT ANALYSIS

3.1 REQUIREMENT ANALYSIS :

FUNCTIONAL REQUIREMENT :

FR No.	Functional Requirement (Epic)	Sub Requirement (Story/ Sub-Task)
FR-1	User Registration	Registration through Form Registration throughGmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via EmailConfirmatio n via OTP
FR-3	Multiple login	Many userscan log in by usingseparate mail identities.Also,using the mail identity ,theuser can log into anydevice.
FR-4	Alerting the user	A limitmust be set on the amount of money to bespent.Whenever the user exceeds thel imit, he will benotified through mailor tex message.
FR-5	Reporting	An analysis on the expenses should be done.Based ontheanalysis ,a detailed report (in any graphical form) must be generated to help the user in accounting and budgeting.

4.2 NON FUNCTIONAL REQUIREMENT :

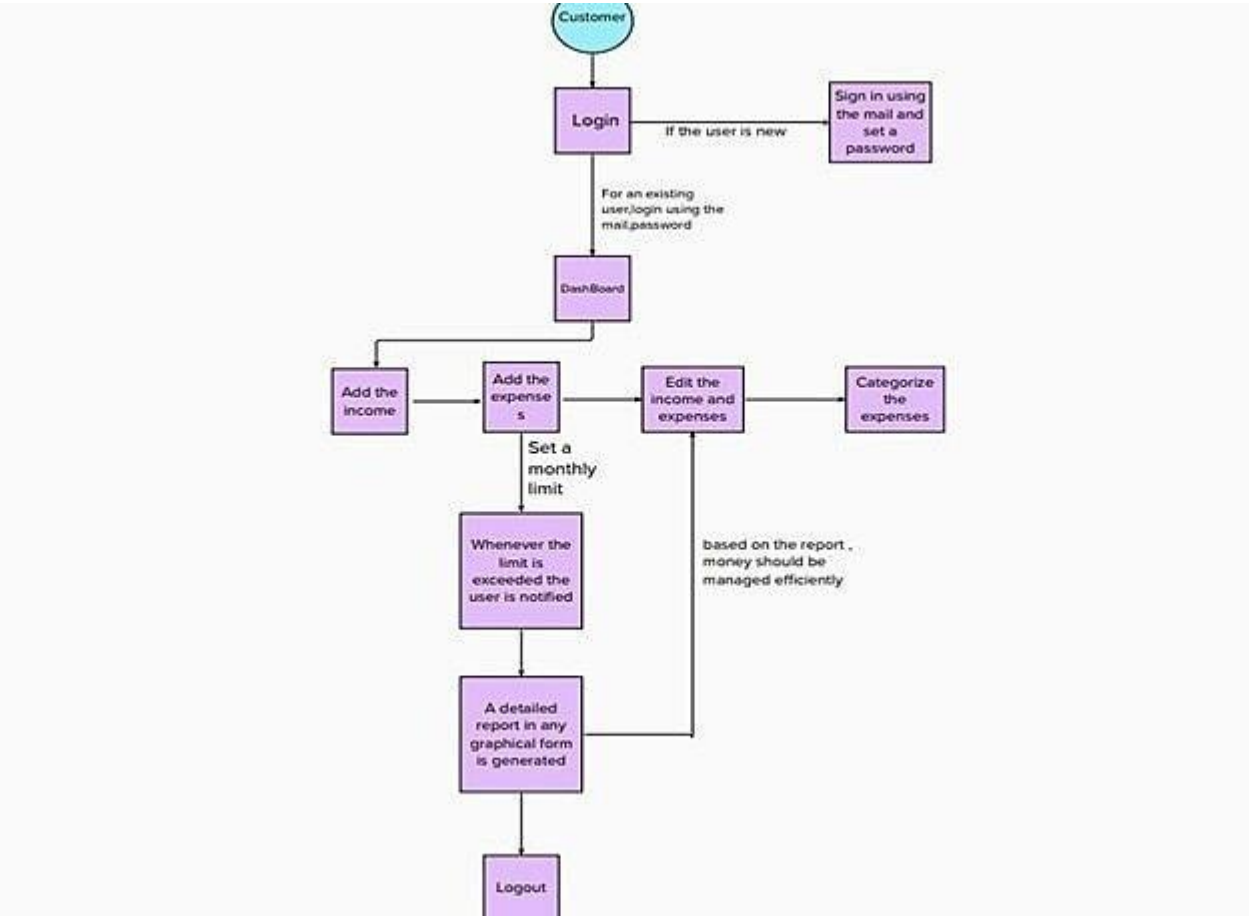
FR No.	Non-Functional Requireme	Description

	nt	
NFR-1	**Usability**	The interface must be user friendly that makes iteasyto use for all types of users.The basicfeaturesmust be available free of cost to us ers.
NFR-2	**Security**	The application should have multi-factor authentication when logging in.Also, banking datamust be securedby some encryption technology.
NFR-3	**Reliability**	The transaction must rollback if there is any technical or network issue .The data must be savedwhen updation of datafailsin between theprocess.Even if there is a failure,it should be restored within a few minutes.
NFR-4	**Performance**	The application must not takemore than 30 secondsto load. The response time should be quick even when there is heavy traffic.
NFR-5	**Availability**	When the app is being updated,except for the module that is beingupdated,the rest can be used.
NFR-6	**Scalability**	The app must be designed to work efficiently evenwhen thereis heavy traffic.

3. PROJECT DESIGN :

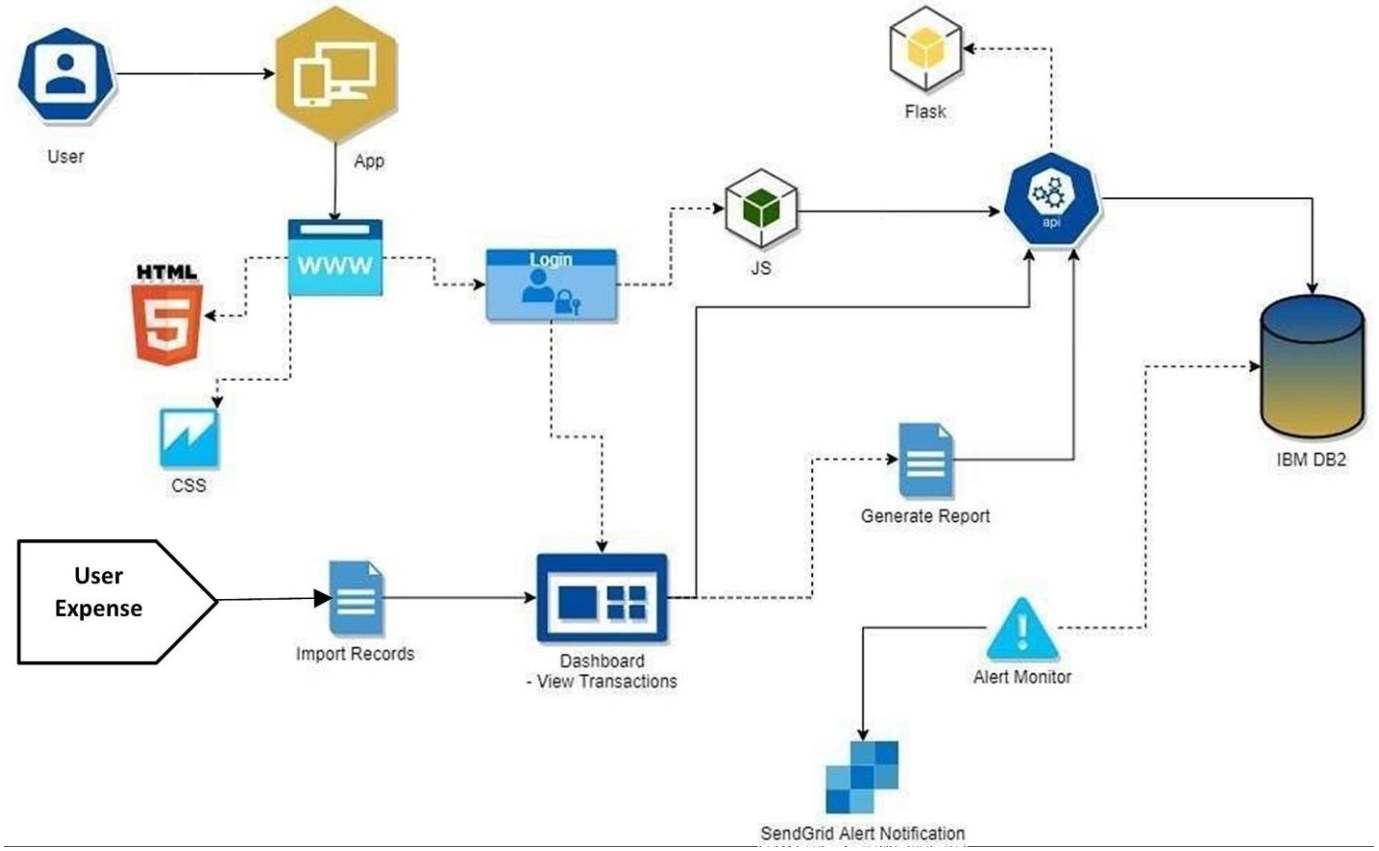
3.1 DATAFLOW DIAGRAM :

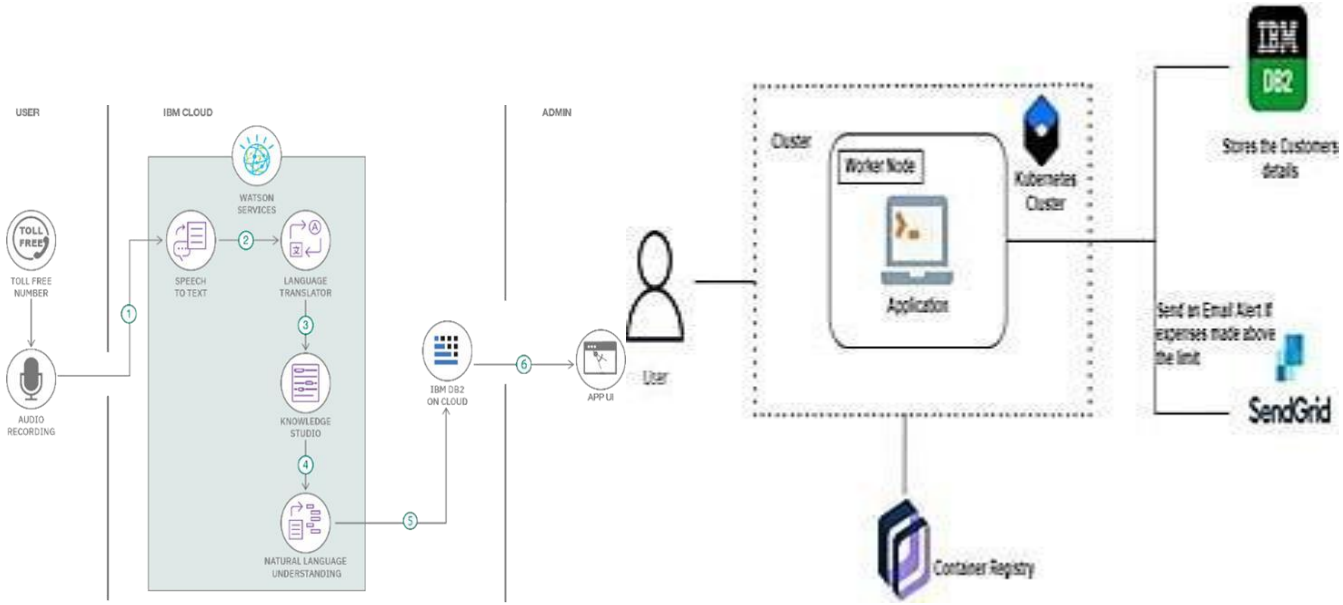
A Data Flow Diagram(DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depictthe right amount of the system requirement graphically. It shows how data enter and leaves the system, what changes the information, and where datais stored.



SOLUTION AND TECHNICAL ARCHITECTURE :

SOLUTION ARCHITECTURE





3.1 USER STORIES:

Use the below template to list all the user stories for the product.

Table-1 : Components & Technologies:

S. No	Component	Description	Technology
1.	User Interface	The usercan Interact withthe application with use of IBM WatsonChatbot.	HTML, CSS, JavaScript /Angular-js / React-js etc.
2.	Application Logic-1	The application contains the sign in/sign up where the userwill loginintothe main dashboard.	Java / Python
3.	Application Logic-2	Dashboard contains the fields like Add income, AddExpenses, SaveMoney, Add budget, Profileetc…	IBM Watson STT service
4.	Application Logic-3	The user will get the expense report in the Statistics formand get alertsif the expense limit exceeds.	IBM WatsonAssistant
5.	Database	TheIncome and Expensedata are stored in the IBM Cloud database.	MySQL, NoSQL,etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM-Cloudant etc.
7.	File Storage	IBM CloudStorage used to store the financial dataof the user	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External APIused in the application	IBM Weather API, etc.

9.	External API-2	Purpose of External API usedin the application	Aadhar API, etc.
10	Machine Learning Model	Purpose of MachineLearning Model	Object Recognition Model,
11	Infrastructure (Server / Cloud)	Application Deployment on Local System/ CloudLocal Server Configuration: Cloud Server Configuration :	Local, CloudFoundry, Kubernetes, etc.

6.PROJECT PLANNING AND SCHEDULING

6.2 SPRINT PLANNING &ESTIMATION

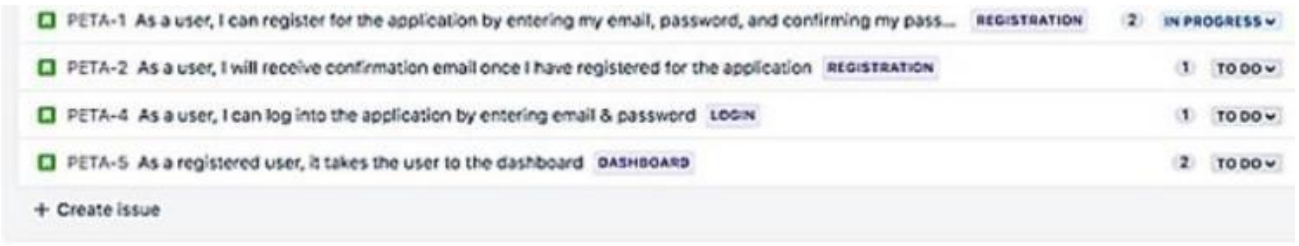
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the applicationby entering my email, password, and confirming mypassword.	2	High	Pavithra
Sprint-1		USN-2	As a user,I will receive confirmation emailonceI have registered for the application	1	High	Poovitha
Sprint-1	Login	USN-3	As a user, I can register for the applicationthroughGmail	1	High	Parthiban
Sprint-1	Dashboard	USN-4	As a user,I canlog into the application byenteringemail& password	2	High	Pranav sundar
Sprint-2	Workspace	USN-1	Workspace for personal expensetracking	2	High	Pavithra
Sprint-2	Charts	USN-2	Creating variousgraphsandstatistics ofcustomer's data	1	Medium	Poovitha
	Connecting to IBMDB2					
Sprint-2		USN-4	Making dashboard interactive with JS	2	High	Parthiban
			frontend			

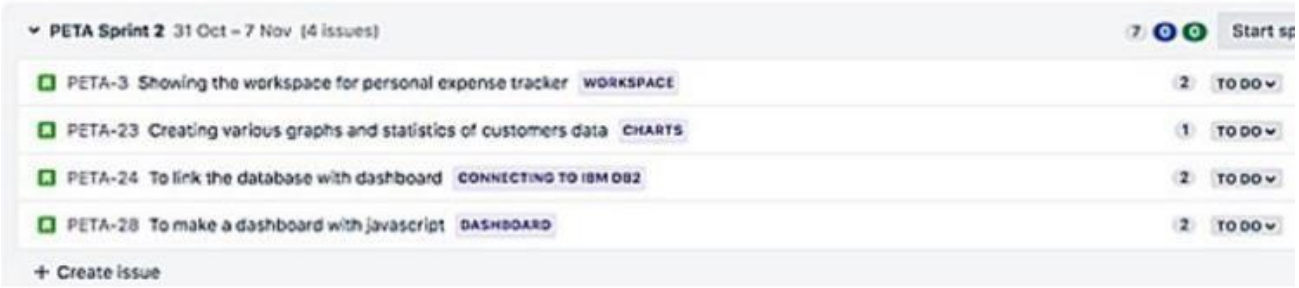
Sprint -3	Watson Assist ant	USN -2	Creating Chatbot for expense tracking and forcalrifying user'squ ery	1	Mediu m	Poovith a
Sprint -3	SendGrid	USN -3	Using SendGrid to send mail to the userabout	1	Low	Pranav sundar

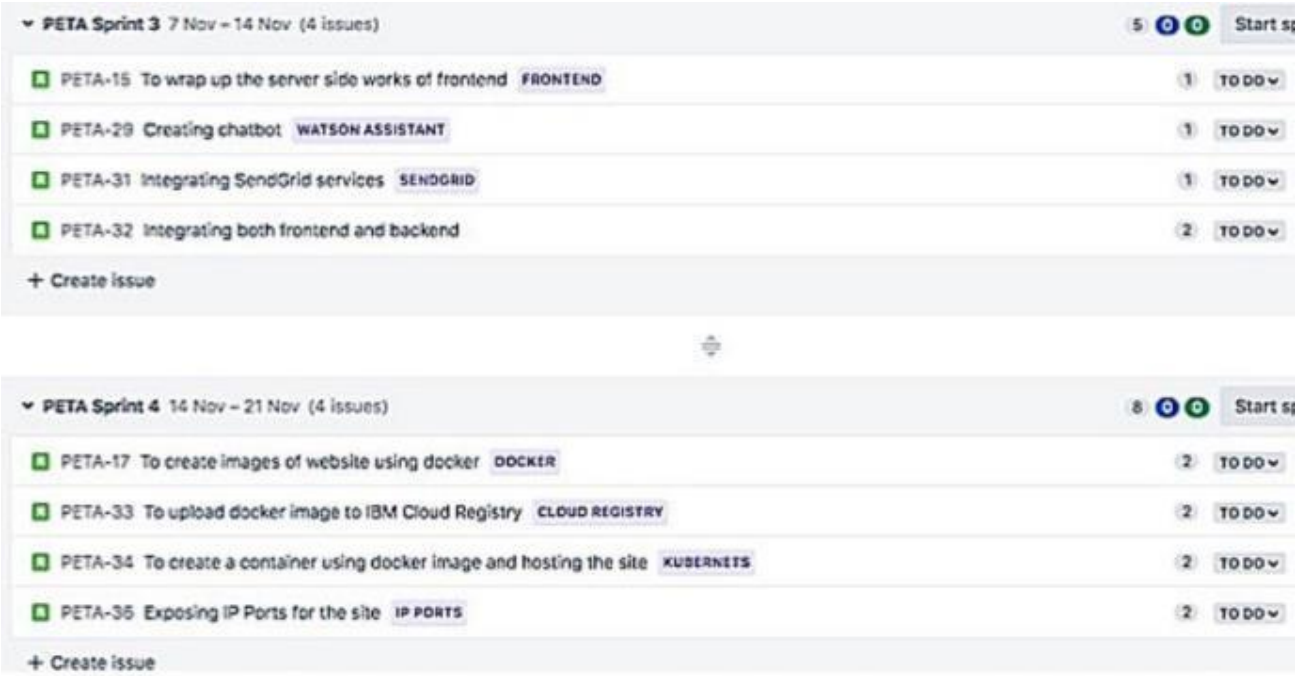
Sprint-4	Docker	USN-1	Creating image of website usingdocker	2	High	Pavithra
Sprint-4	Cloud Registry	USN-2	Uploading docker imageto IBM Cloudregistry	2	High	Poovitha
Sprint-4	kubernetes	USN-3	Create container usingthe dockerimageandhosting thesite	2	High	Parthiban
Sprint-4	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Pranav sundar

6.3 REPORT FROM JIRA:

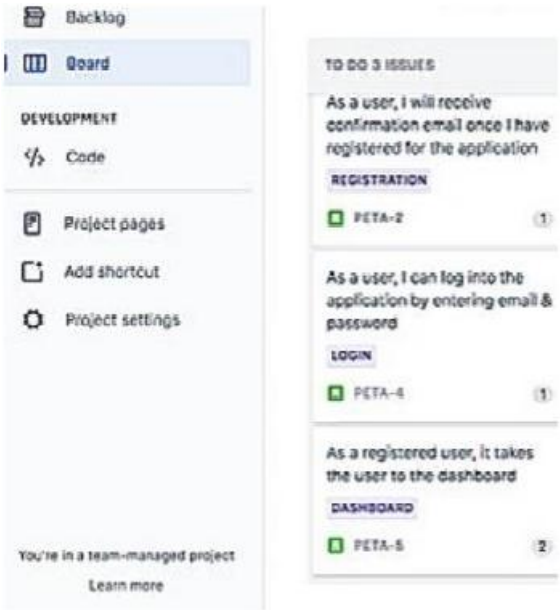
BACKLOG:



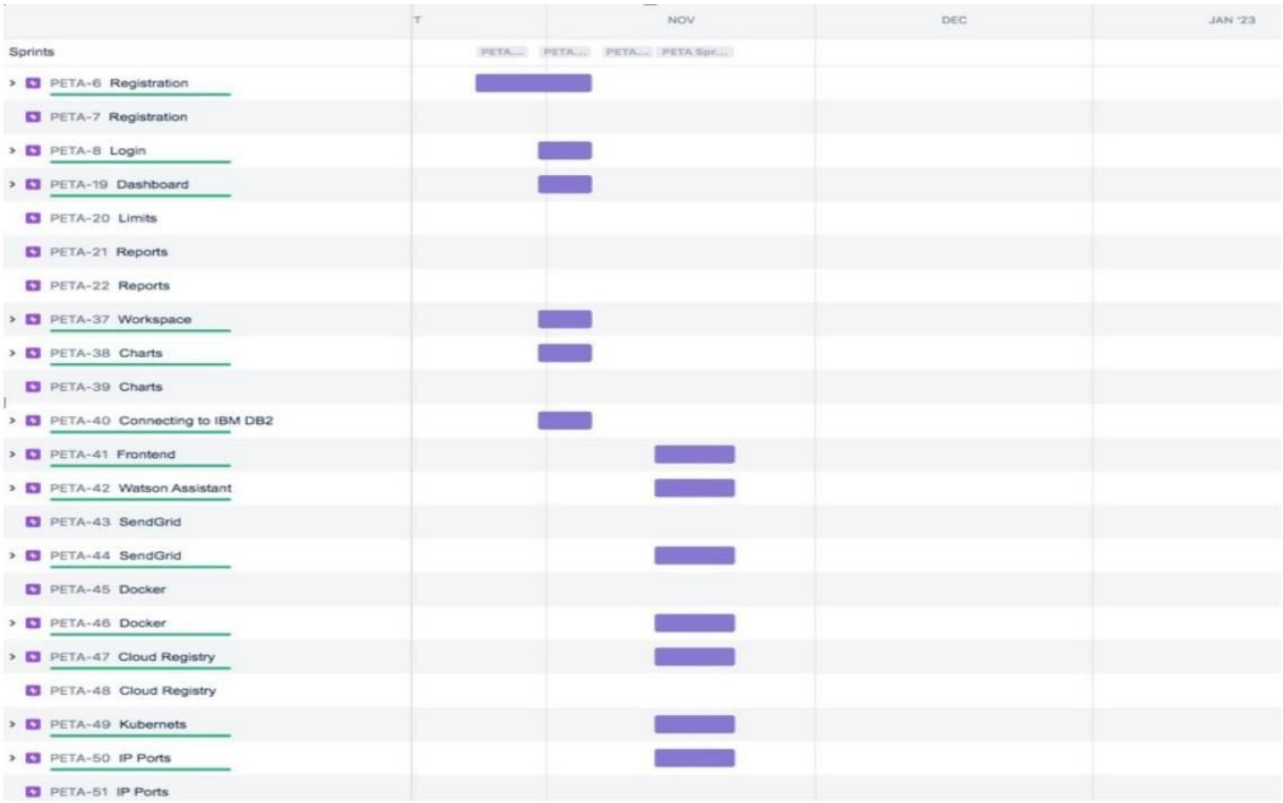




BOARD:



ROAD MAP:


T
NOV
DEC
JAN '23
Sprints
PETA...
PETA...
PETA...
PETA Spr...
PETA-6 Registration
PETA-7 Registration
PETA-8 Login
PETA-19 Dashboard
PETA-20 Limits
PETA-21 Reports
PETA-22 Reports
PETA-37 Workspace
PETA-38 Charts
PETA-39 Charts
PETA-40 Connecting to IBM DB2
PETA-41 Frontend
PETA-42 Watson Assistant
PETA-43 SendGrid
PETA-44 SendGrid
PETA-45 Docker
PETA-46 Docker
PETA-47 Cloud Registry
PETA-48 Cloud Registry
PETA-49 Kubernets
PETA-50 IP Ports
PETA-51 IP Ports

CHAPTER 7

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="{{ img_data1 }}"
          />
        </div>
        <div class="col col-lg-5 col-md-3 px-4" style="background-color: #00ad83">
          <img
            id="picture"
            src="{{ 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'] = 'SG.rRPqo3ZyRhWUD6RhljE1CA.894zN6QMM9UjOpgPlO-4KT-
_mjT9-KwXZ9ArygkEnis'
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;SSLServerCe
rtificate=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)
```

CHAPTER 8
TESTING

8.1 TEST CASES :

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	Logi n 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 i sdisplayed	Graph i sdisplayed	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 thatI 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

```
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'] =
'SG.rRPqo3ZyRhWUD6RhljE1CA.894zN6QMM9UjOpgPlO-4KT-_mjT9-
KwXZ9ArygkEnis'
```

```
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=CXtQLA
GZ06fD0fhC;", "", "")
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)
```