PERSONAL EXPENSE TRACKER

PROJECT REPORT

Submitted by

MEENA A (19EUEC081)

MITHRA R V (19EUEC082)

MOHAMED IRFAN (19EUEC083)

MOUNA RANGEETHA V (19EUEC084)

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

CHAPTER 1

INTRODUCTION

In today's busy and expensive lives we are in a great rush to make money. But at the end of the day we broke off. As we are unknowingly spending money on little and unwanted things. So, we have come over with the idea to track our earnings. Daily Expense Tracker (DET) aims to help everyone who are planning to know their expenses and save from it. DTE is a website in which user can add expenses on daily basis and its table will get generated and at the end based on user expenses report will be generated. User can select date range to calculate his/her expenses come over with the idea to track our earnings. Personal Expense Tracker aims to help everyone who are planning to know their expenses and save from it. Personal Expense Tracker is a website in which user can add expenses on daily basis and at the end, based on user expenses report will be generated. User can select date range to calculate his/her expenses.

1.1 Project Overview

In day to day life , people spend a lot of money in many things and its mostly costs our expenses. This is an integrated project that aims to track the people expenses on daily, weekly and monthly basis. certain tools like Flask, Docker, IBM cloud which helps us to complete this project successfully.

This Expense Tracker is a web application that facilitates the users to keep track and manage their personal as well as business expenses. This application helps the users to keep a digital diary. It will keep track of a user's income and expenses on a daily basis. The user will be able to add his/her expenditures instantly and can review them anywhere and anytime with the help of the internet.

1.2 Purpose

- The app will track all your payment dates, whether it is for credit card dues, phone bills, utility bills and so on.
- It will send alerts to your phone so that you do no miss any payments. Advance alerts help you manage large payments like credit card dues. This means that you do not have to pay late payment charges.
- It can categorise the expenses spent by people.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing problem

The existing problem is the tracking the expenses of the people money. It cannot be done just by uploading the dataset but real time money calculation is not done efficiently.

2.2 References

(1) A research at university on Tennessee on expense tracker of by (Dan Underwood, 2011): In which using excel accounting team designed a Cost Allocation tool 1 in which a spreadsheet is used to allocate the product category both by site and the cooperation and a Cost allocation tool 2 which is developed to further integrate and allocate cost to identify which manufacturer is profitable or which is not. This research used excel and designed this CAT tool in which both the spreadsheets are required to use to identify where we could reduce expenses or better managed it.

(2) (Girish Bekaroo, 2007) did a research on intelligent online budget that manages the expenses and used to give the graphical analysis of data. It uses a Rational Unified Method (RUP) which was way more efficient and advantageous in the way it used to promote code reuse and encapsulation in which CSS and xml technologies has been used.

(3) Researchers of Nandha and Anna university (2016) created an android version of expense manager in with they used post and remark techniques for underlining the expenses and some of the data mining features for analyzing the market value well.

(4) (R N Rajprabha, 2017) created an android version of family budget manager with later evolved in PDA and tablet features.

(5) (Ravi Sharma, 2017) stated users sometimes feels uncomfortable in sharing their personal information with an app and he suggested security and usability are two major concerns. Even the advanced UI needs to maintain retention.

(6) Intelligent Online Budget Tracker

The development of this application has been conducted in a stepwise manner using the well defined methodology, RUP, customized according to the requirements of the system. Most of the goals set at the start of the development phase have been met. Security problems like web security or network security have also been treated in the design and development of the system, thus increasing the reliability of the system. Quality management issues have also been handled satisfactorily.

(7) Online Income and Expense Tracker

This project is work more efficient than the other income and expense tracker. The project successfully avoids the manual calculation for calculating the income and expense per month. The modules are developed efficiently and also in an attractive manner.

(8) Family Expense Manager Application

As the result, the user can make use of this application in his/her daily life. After being used it can be a part of daily life to update and view daily expenses and family expenses. This helps to keep track of expenses & manage it for the user as they are busy in their daily routine, they are not able to keep track of their incomes & expenses.

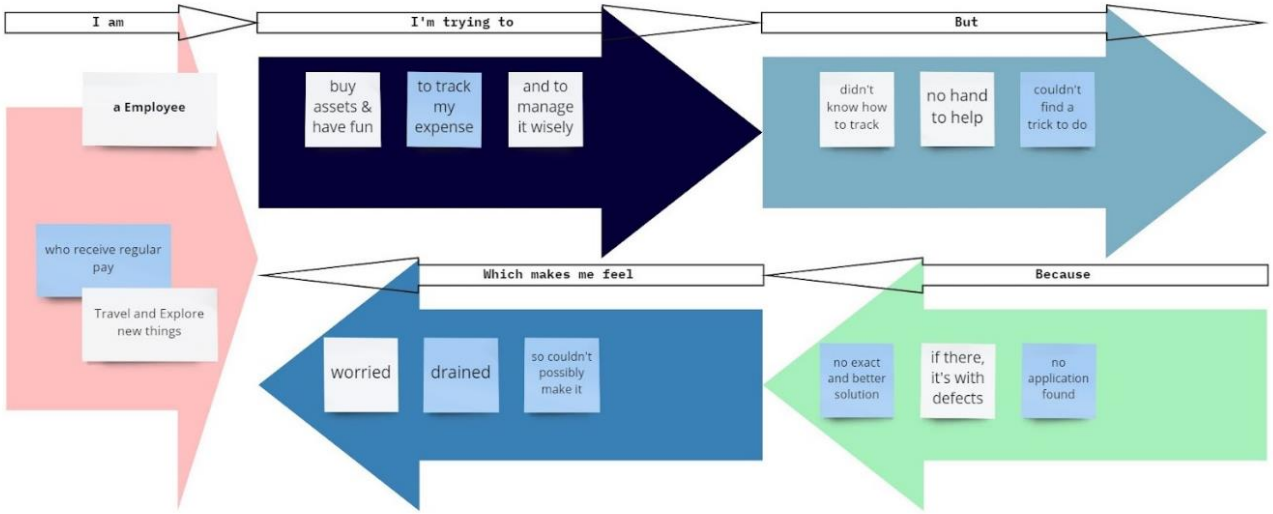
(9) Personalized Expense Managing Assistant Using Android

Some of the features are like enabling users to register to the application using an existing email or social network account, it will synchronize the user's profile information to the application. Apart from this, the application can be used to gather samples of data related to user's expenses with consents and use those sample data as parameters to assess patterns of spending. Using some data mining techniques expenses can be classified and can be used in market analysis and planning

2.3 Problem Statement Definition

Numerous companies utilize their own system for recording their revenue and expenses, which they believe to be the most important indicator of their company's progress. It is a good habit for a person to keep track of their daily expenses and profits, but due to ignorance and a lack of suitable applications, many people will fail to do so. People who lack decision-making capacity use conventional note-taking methods to maintain their privacy. There are two constant overloads to rely on daily entry of expenses and total estimate until the end of the month.

The Issue Flowed Below:



Exactly who is the issue affecting?	People who receive regular pay.
What's the real worry?	The paper-based expense tracker system does not assist user portability, the existing system relies solely on paper-based records and is thus incapable of allowing user portability. Update any expenses that have been completed but are unable to alter the location of the expense data. The proposed system is much more disruptive.
When did the defect occur?	When the digits could not be found to be correct. When transactions fail to complete. When elderly people are unable to understand the smaller handwritten digits. When a paper-based expense tracker is being used, they are vulnerable to fire, flood, and certain other natural disasters.
Where is the site of the troubles?	The trouble comes when the individual is unable to keep a record of his expenses and earnings.

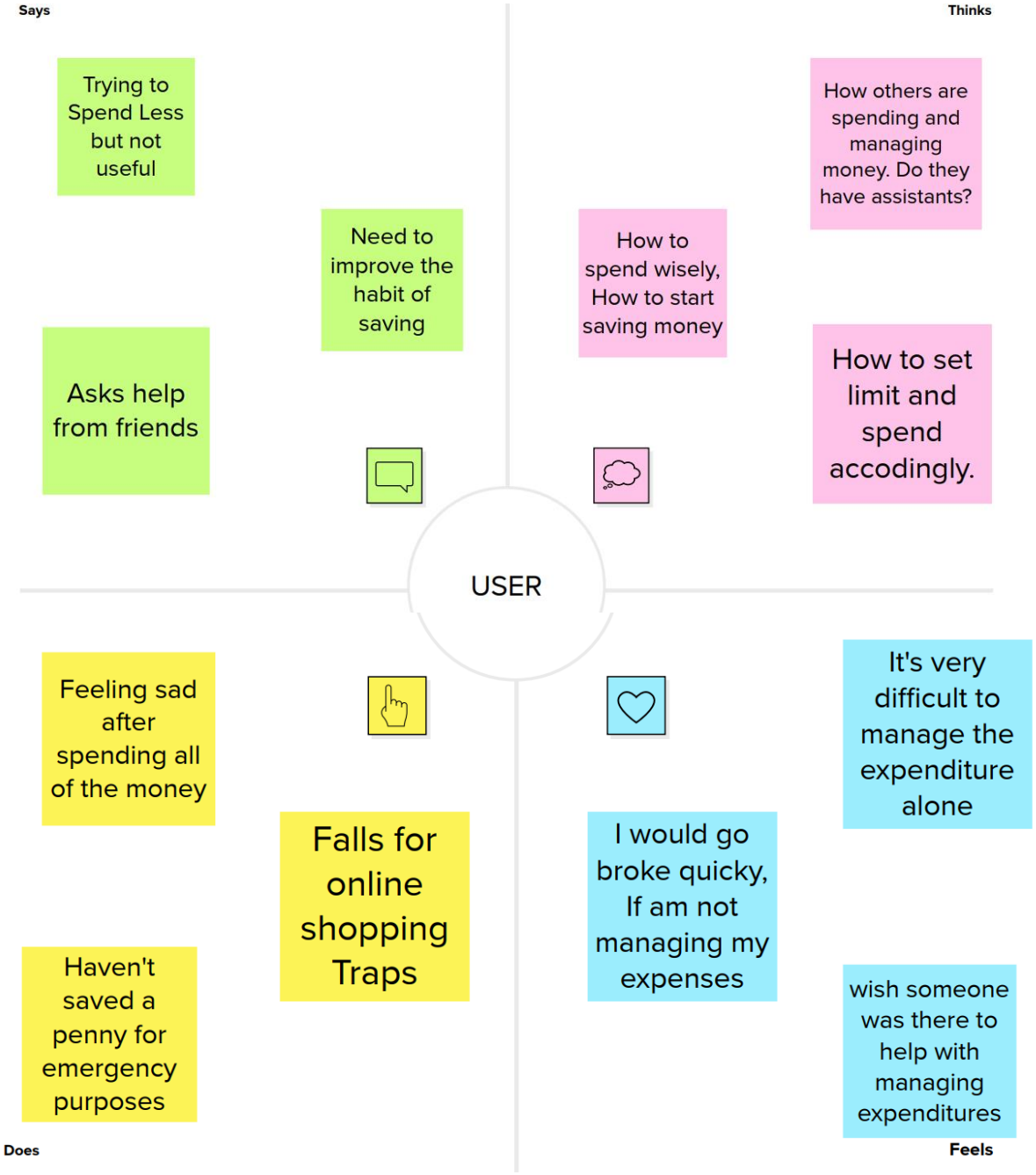
Why is it important that we fix the problem?	By resolving this issue, those receiving regular wages will be able to track their expenses and avoid unnecessary expenses.

Example:

Problem Statement (PS)	**I am (Customer)**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	a Freelance Photographer	Balance my expense	Struggling to cope up with it	Couldn't find any resolution	Perturbed
PS-2	a Marine or Wildlife Biologist	Budget my outlays	Tired of mapping it	Had no tricks to get it done	Rattled

CHAPTER 3
IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



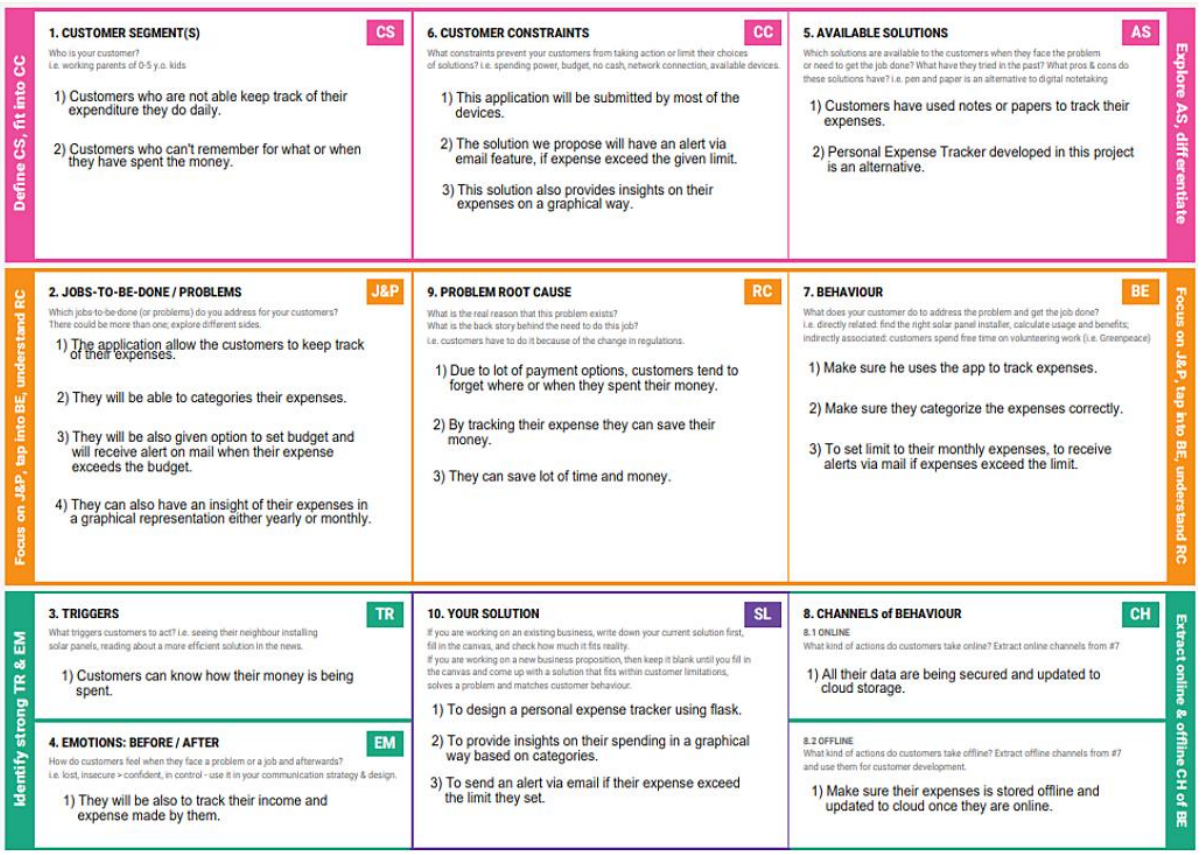
3.2 Ideation & Brainstorming

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	1. It is challenging for the people to manage their cash flow day-to-day. 2. They always prepare a monthly budget for their expenses manually.
2.	Idea / Solution description	1. Due to the busy hectic lifestyle, people tend to overlook their budget and end up spending an excessive amount of money since they usually didn't plan their budget wisely. 2. User cannot predict future expenses. While they can write down their expenses in a excel spreadsheet, their lack of knowledge in managing finances will be a problem.
3.	Novelty / Uniqueness	1. This application tracks your every expenses anywhere and anytime without using the paper work. Just click and enter your expenditure. 2. To avoid data loss, quick settlements and reduce human error. 3. To provide the charts or graph lines in this application.
4.	Social Impact/ Customer Satisfaction	1. Using the application one can track their personal expenses and frame a monthly/annual budget. 2. If your expense exceeded than the specified limit, the application will show you an alert message in the form of an e-mail.
5.	Business Model(Revenue Model)	1. We can provide the application on a subscription basis. 2. We can provide pop-up ads, overlay ads, and other advertising services from third party advertisers.
6.	Scalability of the Solution	1. This application will be available to all the user for life long time to maintain their own expenses in an effective way.

		2. IBM cloud will automatically allocate the storage for the users.

3.4 Problem Solution fit



CHAPTER 4
REQUIREMENT ANALYSIS

4.1 Functional requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Application Registration through E-mail
FR-2	User Confirmation	Confirmation via E-mail Confirmation via OTP
FR-3	User monthly expense tentative data	Data to be registered in the app
FR-4	User monthly income data	Data to be registered in the app
FR-5	Alert/ Notification	Alert through E-mail Alert through SMS
FR-6	User Budget Plan	Planning and Tracking of user expense vs budget limit

4.2 Non-Functional requirements

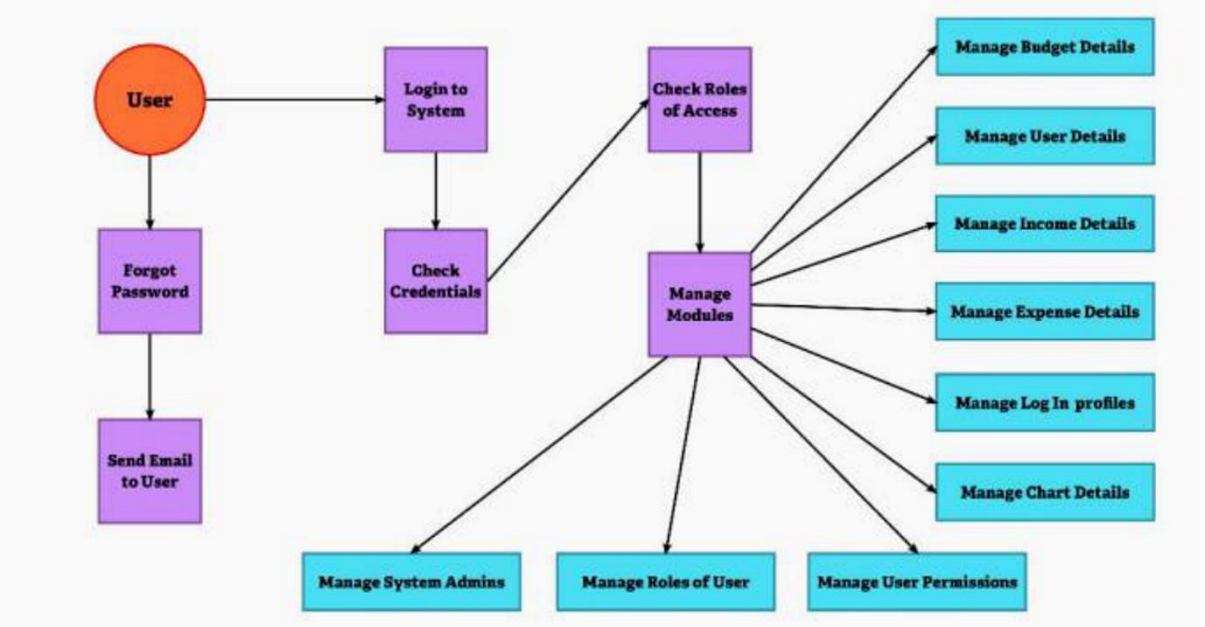
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Effectiveness, efficiency and overall satisfaction of the user while interacting with our application.
NFR-2	**Security**	Authentication, authorization, encryption of the application.
NFR-3	**Reliability**	Probability of failure-free operations in a specified environment for a specified time.
NFR-4	**Performance**	How the application is functioning and how responsive the application is to the end-users.
NFR-5	**Availability**	Without near 100% availability, application reliability and the user satisfaction will affect the solution.
NFR-6	**Scalability**	Capacity of the application to handle growth, especially in handling more users.

CHAPTER 5
PROJECT DESIGN

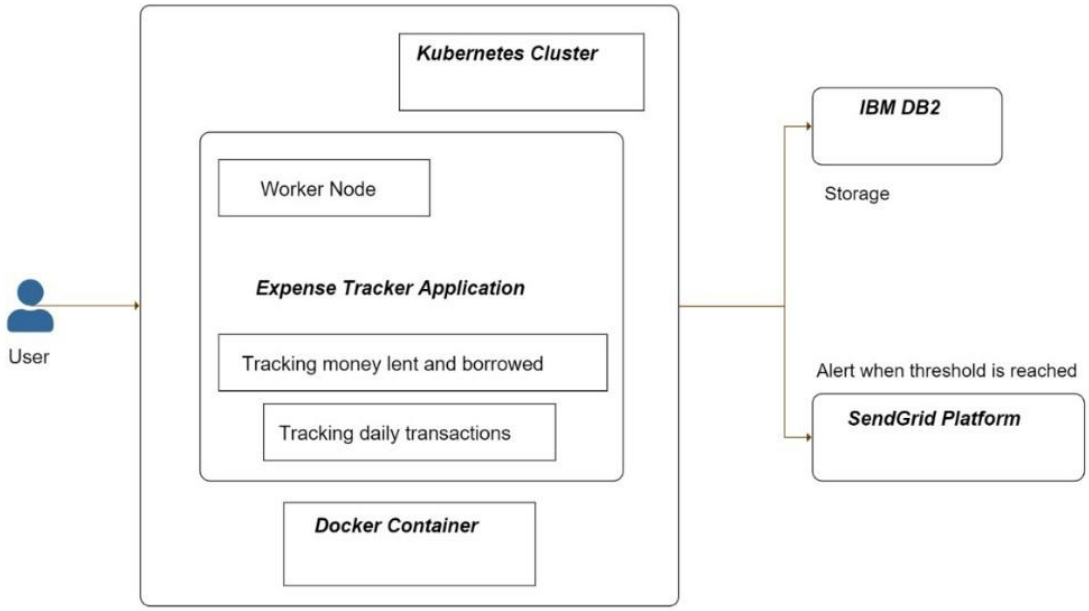
5.1 Data Flow Diagrams

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



5.2 Solution & Technical Architecture

Solution Architecture:



Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

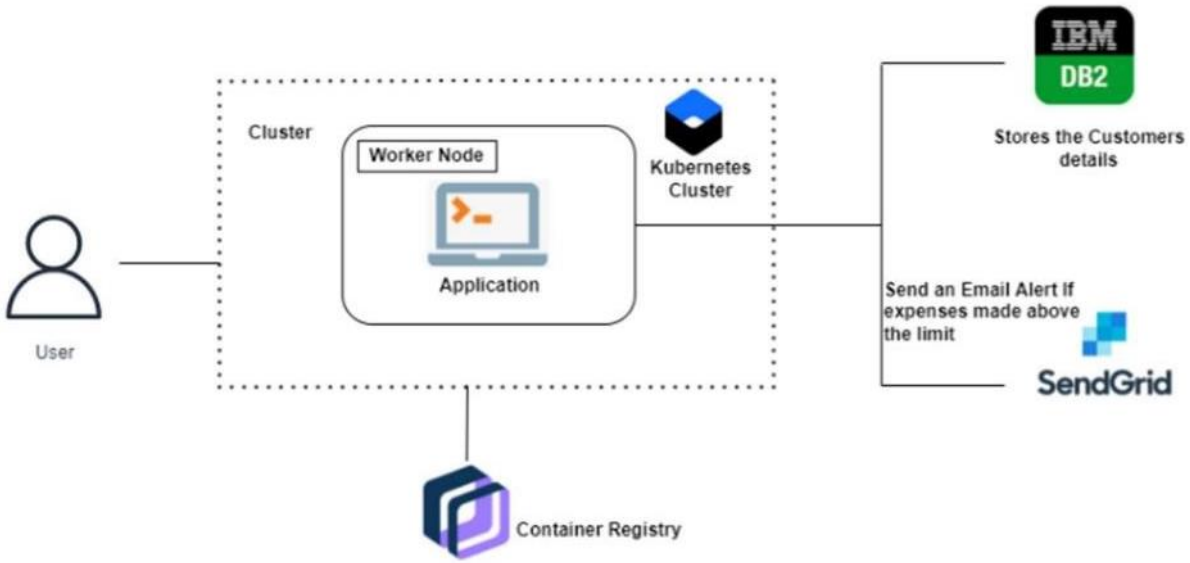


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	The user can Interact with the application with use of Chatbot.	HTML, CSS, JavaScript / ReactJS, etc.
2.	Application Logic-1	The application contains the sign in/sign up where the user will login into the main dashboard.	Python
3.	Application Logic-2	Dashboard contains the fields like Add income, Add Expenses.	IBM Watson STT service
4.	Application Logic-3	The user will get the expense report in the graph form and also get alerts if the expense limit exceed.	IBM Watson Assistant, SendGrid.
5.	Database	The Income and Expense data are stored in the MySQL database.	MySQL, NoSQL, etc.
6.	Cloud Database	With use of Database Service on Cloud, the User data are stored in a well secured Manner.	IBM DB2, IBM Cloudant, etc.
7.	File Storage	IBM Block Storage used to store the financial data of the user.	IBM Block Storage or Other Storage Service or Local Filesystem.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Flask Framework in Python is used to implement this Application.	Python-Flask.
2.	Security Implementations	This Application Provides high security to the user financial data. It can be done by using the Container Registry in IBM cloud.	Container Registry, Kubernetes Cluster.
3.	Scalable Architecture	Expense Tracker is a life time access supplication. Its demand will increase when the user's incomes are high.	Container Registry, Kubernetes Cluster.
4.	Availability	This application will be available to the user at any part of time.	Container Registry, Kubernetes Cluster.
5.	Performance	The performance will be high because there will be no network traffics in the application	Kubernetes Cluster.

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user & Web user)	Registration	USN - 1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard.	High	Sprint – 1
		USN - 2	As a user, I can track my expenses and	I can track my expenses and manage	High	Sprint – 3

			manage my monthly budget.	my monthly budget.		
		USN - 3	As a user, I can see if there is an excessive expense and if there is such condition, I will be notified via e-mail.	I can receive e-mail, if there is an excessive expense.	Low	Sprint – 3
	Login	USN - 4	As a user, I can login to user dashboard and see the information about my incomes and expenses.	I can login to user dashboard and see the information.	High	Sprint – 1
	Dashboard	USN - 5	As a user, I can enter my income and expenditure details.	I can view my daily expenses.	High	Sprint – 2
Customer Care Executive		USN – 6	As a customer care executive, I can solve the log in issues and other issues of the application.	I can provide support or solution at any time 24*7	Medium	Sprint – 4
Administrator	Application	USN - 7	As an administrator, I can upgrade or update the application.	I can fix the bug which arises for the customers and users of the application.	Medium	Sprint - 4

CHAPTER 6
PROJECT PLANNING & SCHEDULING

6.1 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 application by entering my email,password, and confirming my password.	2	High	Meena A, Mithra R V
		USN-2	As a user, I will receive confirmation email once I have registered for theapplication	1	High	Mouna Rangeetha, Mohamed Irfan
	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Meena A, Mouna Rangeetha
	Dashboard	USN-4	Logging in takes to the dashboard for the logged user.	`2	High	Mohamed Irfan, Mithra R V
Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only						
Sprint 2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Mouna Rangeetha, Mithra R V
	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Meena A, Mithra R V
	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Mouna Rangeetha, Mohamed Irfan
		USN-4	Making dashboard interactive with JS	2	High	Meena A, Mohamed Irfan
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Mithra R V, Mohamed Irfan
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user's query	1	Medium	Meena A, Mouna Rangeetha
	SendGrid	USN-3	Using SendGrid to send mail to the user about their expenses	1	Low	Mohamed Irfan, Mithra R V
		USN-4	Integrating both frontend and backend	2	High	Mouna Rangeetha, Mithra R V
Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only						
Sprint-4	Docker	USN-1	Creating image of website using docker/	2	High	Meena A, Mithra R V, Mouna Rangeetha, Mohamed Irfan
	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Meena A, Mithra R V, Mouna Rangeetha, Mohamed Irfan
	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Meena A, Mithra R V, Mouna Rangeetha, Mohamed Irfan
	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Meena A, Mithra R V, Mouna Rangeetha, Mohamed Irfan

6.2 Sprint Delivery Schedule

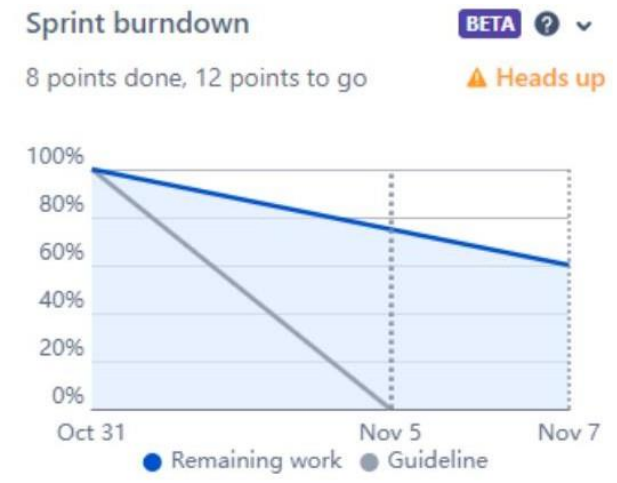
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	23 Oct 2022	28 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	30 Oct 2022	04 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	06 Nov 2022	11 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	13 Nov 2022	18 Nov 2022	20	19 Nov 2022

Project Tracker, Velocity & Burndown Chart:

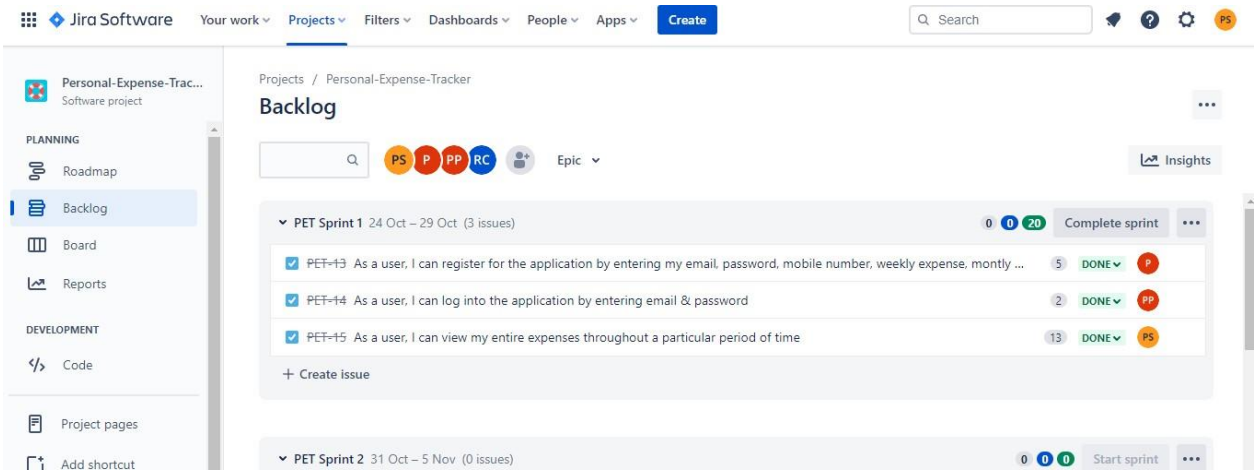
Velocity
We have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). Calculating the team's average velocity (AV) per iteration unit (story points per day)

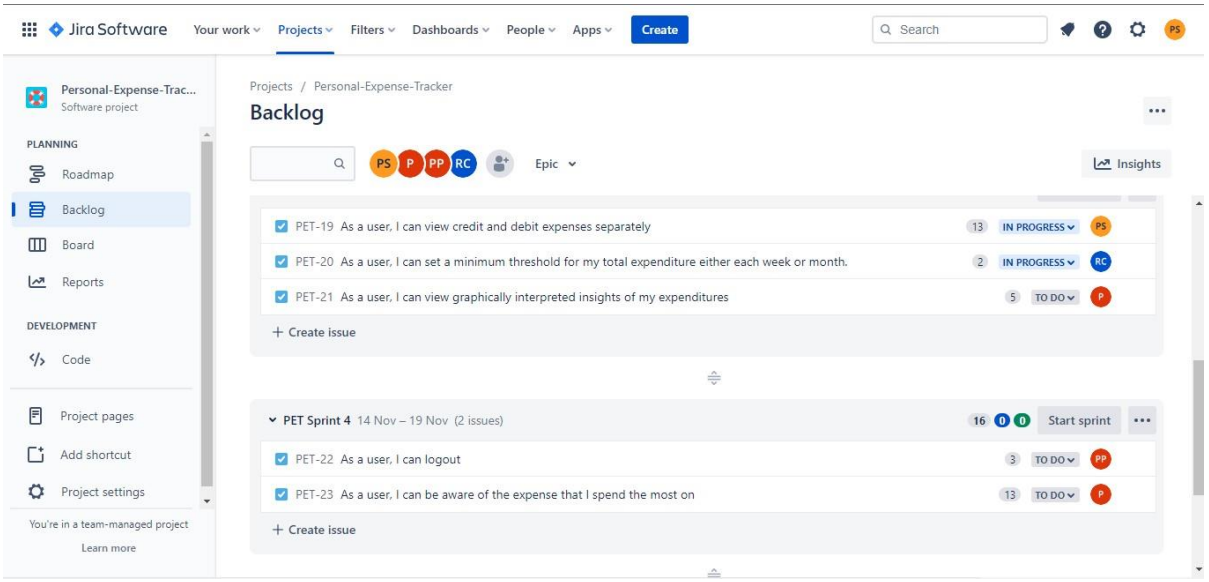
AV = sprint duration / velocity = 20/6 = 3.33

Burndown Chart:
A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



6.3 Reports from JIRA





CHAPTER 7
CODING AND SOLUTIONING

base_template.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <!-- Required meta tags -->
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1">

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

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

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

   {% block title %}
      <title>Base Template</title>
   {% endblock title %}
</head>
<body>
   <div class="container-fluid">
      <div class="row flex-nowrap">
         <div class="col-auto col-md-3 col-xl-2 px-sm-2 px-0" style="background-color: #B2D3C2">
            <div class="d-flex flex-column align-items-center align-items-sm-start px-3 pt-2 min-vh-100"
style="color:black">
               <p class="d-flex align-items-center pb-3 mb-md-0 me-md-auto text-white text-decoration-none">
                  <span class="fs-5 d-none d-sm-inline" style="color:black; font-weight: bold;">Personal Expense
Tracker</span>
                  <img src="https://s3.jp-tok.cloud-object-
storage.appdomain.cloud/personalexpensetrackercapd/piggybank.png" style="width:50px;height: 50px;">
               </p>
               <ul class="nav nav-pills flex-column mb-sm-auto mb-0 align-items-center align-items-sm-start"
id="menu">
                  <li class="nav-item mt-2" style="background-color: {{'#00AD83' if highlight == 'dashboard'}};
height: 50px; width: 150px; border-radius: 5px;">
                     <a href="dashboard" class="nav-link align-middle px-0" style="color:black;">
                        <img src="https://s3.jp-tok.cloud-object-
storage.appdomain.cloud/personalexpensetrackercapd/house-outline.svg" style="width:20px;height:20px;margin-
left: 5px;">
                        <span class="ms-1 d-none d-sm-inline">Home</span>
                     </a>
                  </li>
```

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

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

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

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

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

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

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

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

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

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

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

addcategory.html

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

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

{% set highlight = 'addcategory' %}
```

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

            </div>
            <div class="card-body">
                  <div class="mb-3">
                  <label for="category" class="form-label">Category Name: </label>
                  <input type="text" class="form-control" name="category" id="category"></input>
                  </div>
                  <div class="mb-3">
                  <label for="description" class="form-label">Description of Category: </label>
                  <input type="text" class="form-control" name="description" id="description"></input>
                  </div>
            </div>
            <div class="card-footer text-muted" style="text-align:center">
               <button type="submit" style="background-color:#00AD83; border-color:#00AD83; border-radius:5px;">Add category</button>
            </div>
         </form>
      </div>
   </div>
</div>
{% endblock content %}
```

addexpense.html

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

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

{% set highlight = 'addexpense' %}

{% block content %}
<div class="col py-3" style="background-color:#00AD83">
   <h3 style="color:white; text-align: center;">Add expense</h3>
   <div class="container mt-3" style="width: 600px;">
      <div class="card shadow-lg bg-white rounded">
         <form action="/addexpense" method="POST">
            <div class="card-header" style="text-align: center;">
               <span style="display:inline-flex"><h4>Expense Made</h4><img src="https://s3.jp-tok.cloud-object-storage.appdomain.cloud/personalexpensetrackercapd/pay-15.png" style=" margin-left:10px; width:30px; height:30px"></span>
            </div>
```

```
            <div class="card-body">
               <div class="mb-3">
                  <label for="amountspent" class="form-label">Amount Spent: (Rs) </label>
                  <input type="number" class="form-control" name="amountspent" id="amountspent"
placeholder="100.00" required>
               </div>
               <div class="mb-3">
                  <label for="expensecategory" class="form-label">Expense Category: </label>
                  <select name="category" id="category" class="form-control" placeholder="Select a category"
required>
                     <option value="">Select a category</option>
                     {% for cat in categories %}
                        <option value="{{ cat[0] }}">{{ cat[1] }}</option>
                     {% endfor %}
                  </select>
               </div>
               <div class="mb-3">
                  <label for="date" class="form-label">Date of Expense: </label>
                  <input type="date" class="form-control" name="date" id="date" required></input>
               </div>
               <div class="mb-3">
                  <label for="description" class="form-label">Description of Expense: </label>
                  <input type="text" class="form-control" name="description" id="description"></input>
               </div>
               <div class="mb-3">
                  <label for="group" class="form-label">Group(if needed): </label>
                  <div title="New group" style="float:right" value="Create group" onclick="addGroup()">ADD
GROUP</div>
                     <br/>

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

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

      const formData = new FormData();
```

```
    formData.append("groupname", group);

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

addgoal.html

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

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

{% block content %}
<div class="col py-3" style="background-color:#00AD83">
  <h3 style="color:white; text-align: center;">Add Goal and Reward</h3>
  <div class="container mt-3" style="width: 600px;">
    <div class="card shadow-lg bg-white rounded">
      <form action="/addgoal" method="POST">
        <div class="card-header" style="text-align: center;">
          <span style="display:inline-flex"><h4>Goal & Reward</h4><img src="https://s3.jp-tok.cloud-object-
storage.appdomain.cloud/personalexpensetrackercapd/goal-icon.webp" style=" margin-left:10px; width:30px;
height:30px"></span>
        </div>
        <div class="card-body">
          <div class="mb-3">
            <label for="amountspent" class="form-label">Goal Wallet Balance: (Rs) </label>
            <input type="number" class="form-control" name="goal_amount" id="goal_amount"
placeholder="100.00" required>
          </div>

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

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

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

analysis.html

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

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

{% set highlight = 'analysis' %}

{% block content %}
<div class="col-auto px-0 col-lg-10 col-md-6 col-sm-4">
  <div class="card min-vh-100" style="background-color: #00ad83">
    <h4 class="card-header">Analysis of my expenses</h4>
    <div class="card-body">
      <div class="row flex-nowrap">
        <div class="col col-lg-5 col-md-3 px-4" style="background-color: #00ad83">
          <img
            id="picture"
            src="{{ 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;SSLServerCertificat
e=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	Login page	log into the application by entering email & password	User1@gmail.com *****	Login successful	Login sucessfull	pass
3	TC – ST 01	UI	Stats page	view my entire expenses throughout a particular period of time		Expenses are displayed For particular time	Expenses are displayed For particular time	pass
4	TC – DB 01	UI	Dash-board	Display graph in dashboard		Graph is displayed	Graph is displayed	pass
5	TC – ST 02	Func-tional	Stats page	generate reports based on my previous expenditures		Reports generated in graphical form	Reports generated in graphical form	pass
6	TC – SI 02	Func-tional	Dash-board	can logout		Go to sign page	Sign in page displayed	pass

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

8.2 User Acceptance Testing

1. Defect Analysis

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

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

2. Test Case Analysis

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

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

CHAPTER 9
RESULT

9.1 Performance Metrics

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

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

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

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

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

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

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

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

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

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

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

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

CHAPTER 10
ADVANTAGES AND DISADVANTAGES

Advantages :

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

Disadvantages :

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

CHAPTER 11
CONCLUSION

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

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

CHAPTER 12
FUTURE SCOPE

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

CHAPTER 13
APPENDIX

GitHub Link:
https://github.com/IBM-EPBL/IBM-Project-17881-1659676916

Google Drive Link:
https://drive.google.com/file/d/1GgQs2F297uxdqoSfHhbvWGLKgUzkvSBE/view?usp=share_link