INDEX

1. **INTRODUCTION**
   - 1.1 Project Overview
   - 1.2 Purpose
2. **LITERATURE SURVEY**
   - 2.1 Existing problem
   - 2.2 References
   - 2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   - 3.1 Empathy Map Canvas
   - 3.2 Ideation & Brainstorming
   - 3.3 Proposed Solution
   - 3.4 Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   - 4.1 Functional requirement
   - 4.2 Non-Functional requirements
5. **PROJECT DESIGN**
   - 5.1 Data Flow Diagrams
   - 5.2 Solution & Technical Architecture
   - 5.3 User Stories
6. **PROJECT PLANNING & SCHEDULING**
   - 6.1 Sprint Planning & Estimation
   - 6.2 Sprint Delivery Schedule
   - 6.3 Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
   - 7.1 Feature 1
   - 7.2 Feature 2
   - 7.3 Database Schema (if Applicable)
8. **TESTING**
   - 8.1 Test Cases
   - 8.2 User Acceptance Testing
9. **RESULTS**
   - 9.1 Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

    Source Code

    GitHub & Project Demo Link

1. INTRODUCTION

1.1 Project Overview

This report entails the **Personal Expense Tracker Application** built with HTML, Flask and Docker. This project was developed by Aarthi.N, Aarthi Suresh Kumar, Anirudh.T.N, Joseph Amrithraj, students of SSN College Of Engineering under the guidance of A.Beulah, guide and Khusboo, our industrial mentor from IBM for the Naalaiyathiran scheme of 2022.

1.2 Purpose

A personal finance app makes life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management.

Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert

2. LITERATURE SURVEY

2.1 Existing problem

The current problem is that users are not motivated to monitor their expenses and not willing to alter their spending trends. Our system introduces a goal and reward system to beat the problem. Secondly, the feature of sending reminders for recurring bills is a key feature in this app. Most often, users do not have the right guidance with spending and it is hard for them to analyze their mistakes. To make this process simpler, our app displays the spending trands as a graph to help users analyse and find the real reason behind his inability to meet their goals.

2.2 References

MINT-INDIVIDUAL/PERSONAL EXPENSES:

a. Mint is a free budgeting app that allows you to connect all of your financial accounts in one digital space
b. After connecting your financial accounts, Mint tracks your transactions and categorises them into budget categories to simplify tracking. Users can keep the default categories provided by Mint or create custom categories to fit their needs
c. Track all of your monthly bills through Mint and receive reminders so you can easily pay your bills on time
d. Mint is meant for individual users and there's no option for joint Mint accounts. Users with joint financial accounts can each create their own Mint account and sync the same accounts to view the same information
e. Mint is free for everyone to use.

GOOD BUDGET -INDIVIDUAL/PERSONAL EXPENSES:

a. "Envelope method" - Users allocate a certain amount of their income into categories of expenses(groceries,clothing).
    i. Users can visualise their spending better and define new goals
b. Not linked to bank accounts/credit cards->manual entry of expenses make users aware of spending
c. 256-bit bank grade encryption in a secure data centre
d. Custom/predefined categories of expenses
e. One year of transaction history

QUICKBOOKS - BUSINESS EXPENSES:

a. Tracks business expenses on the fly, as well as income

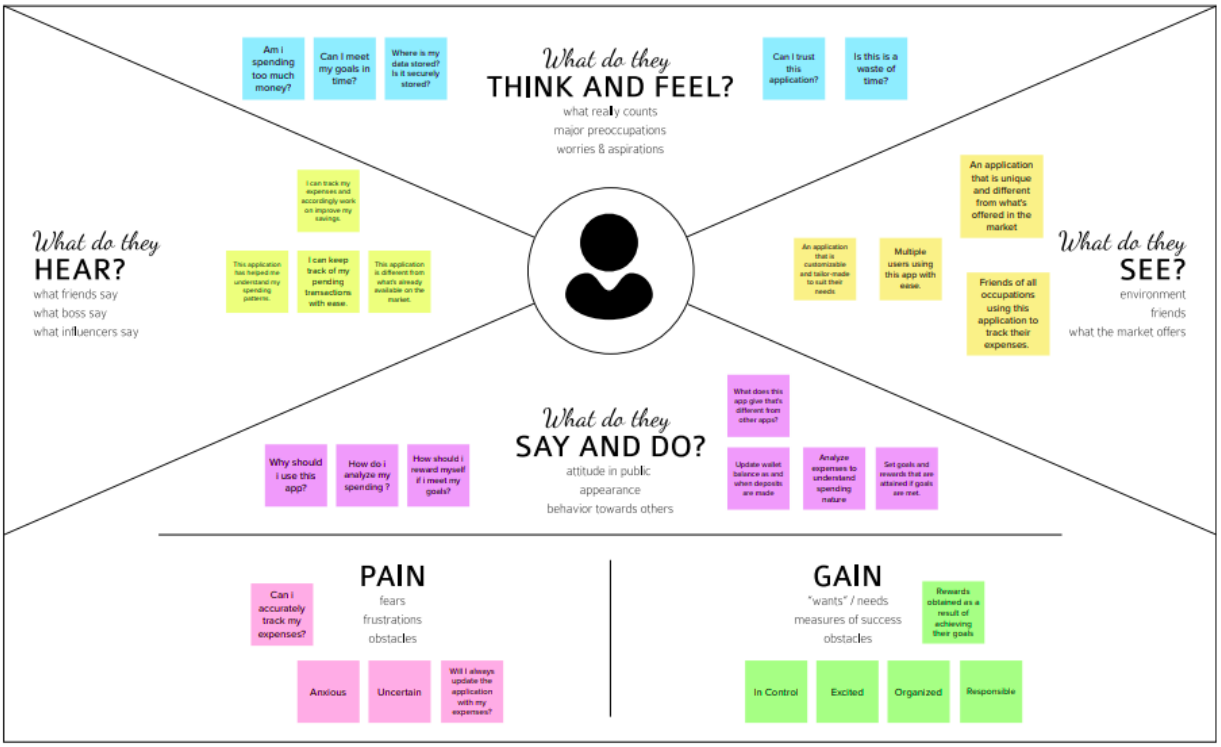
b. Syncs to bank accounts, credit cards and third-party apps like PayPal and Square
c. Offers real-time dashboards to keep user updated on all financial transactions related to their business
d. Different plans according to size of business and requirements(analytics, fast transactions)
e. Full-service payroll is a feature of some plans

2.3 Problem Statement Definition:

Users spend money on various transactions that may be a part of their daily routine or could be a one-time transaction. Every user has different priorities and thus different expenses. Our team aims to develop a customizable Personal Expense Tracker that allows users to tailor-make the application to suit their needs. We aim to do so through the provision of user-defined expense categories, rewards, goals, and limits to name a few. The application will also provide users with the feature to view a graphical analysis of their expenditure to understand their spending patterns and reach conclusions accordingly.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



For a clear picture:
[Empathy Map Github]

3.2 Ideation & Brainstorming

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

4

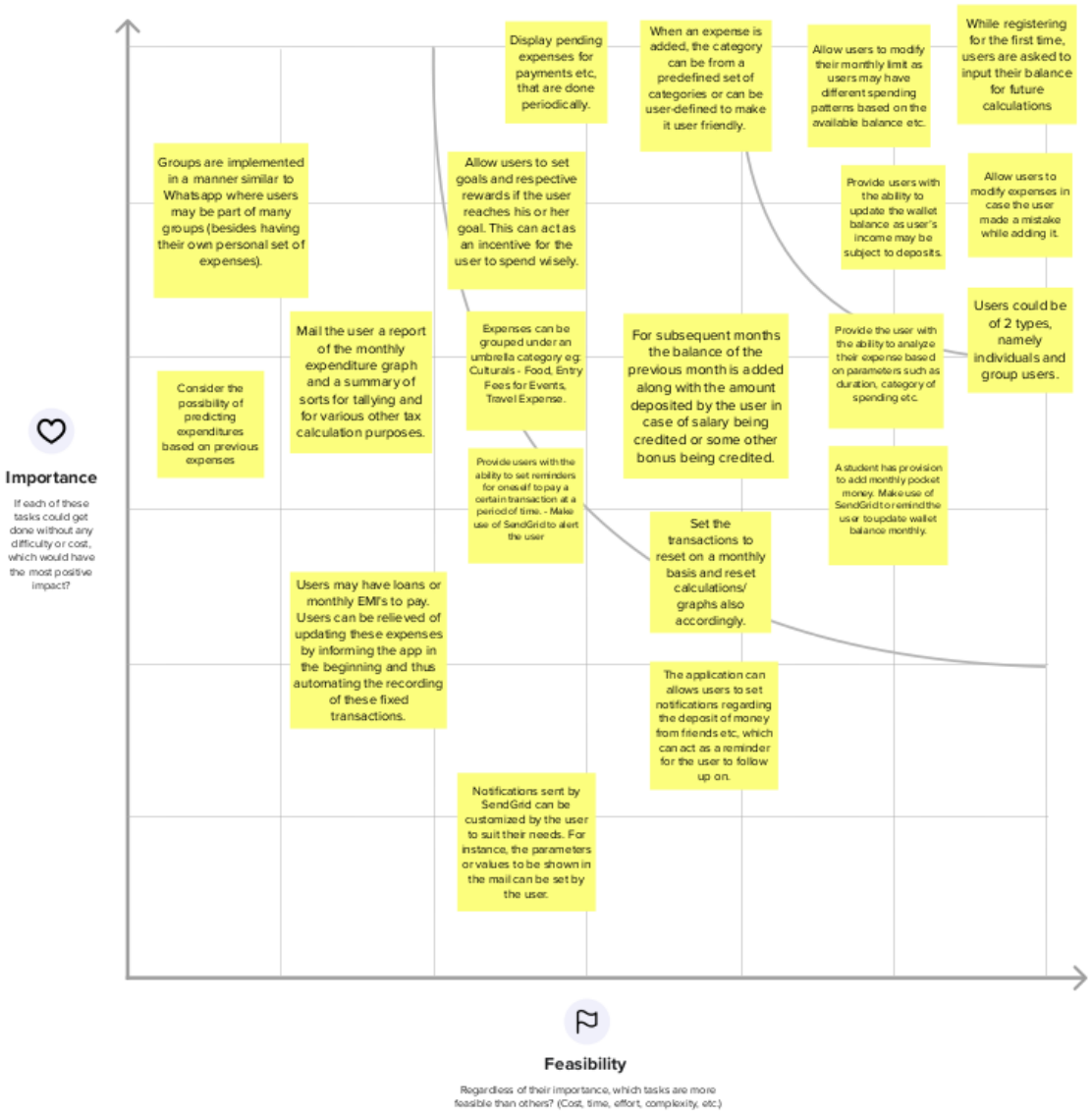
Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.



Link: [Brainstorming&Ideation]

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Users spend money on various transactions that may be a part of their daily routine or could be a one-time transaction. Every user has different priorities and thus different expenses. To keep track of these expenses would be a herculean task if the user were to do the same using pen and paper or by relying solely on their memory. Using these poor alternatives would result in an increase in the probability of errors and thus lead to wrong conclusions about spending.
2.	Idea / Solution description	Our team aims to develop a customizable Personal Expense Tracker that allows users to tailor-make the application to suit their needs. We aim to do so through the provision of user-defined expense categories, rewards, goals, and limits to name a few. The application will also provide users with the feature to view a graphical analysis of their expenditure to understand their spending patterns and reach conclusions accordingly.
3.	Novelty / Uniqueness	There exist certain applications that allow users to track expenses. Our project aims to make an application that is customizable, thus providing the user with the most access. The ability of the application to be customized introduces the main novelty of our project. Apart from this, other features such as allowing users to set rewards and goals for themselves to spend wisely introduces further novelty to our project.

4.	Social Impact / Customer Satisfaction	This application will help our users track their expenses effortlessly. Customers will be put at ease when it comes to their spendings as the only requirement is for the user to update the application when an expense occurs. The social impact of the web application would be far-reaching as the application is applicable to users of different stratas. With its ability to be customized, the application is developed to suit the needs of all our users alike.
5.	Business Model (Revenue Model)	**Key partners:** The members of the team alongside SSN and IBM mentors will work towards the development of this application.

		Key resources: The resources for the development are obtained using our personal equipment, various IDE, IBM's database and software, college systems etc. **Activities:** The main activities include developing the application using Flask, interfacing it with IBM DB2, SendGrid, containerizing the application, and hosting it on the cloud. **Value Proposition:** The users will be provided with a web application that has a friendly GUI and serves all the tasks of the application in a transparent manner. Security compliance will be strictly monitored to ensure that the user's data is safeguarded against any form of threats.
		Cost Structure: Cost is levied due to the usage of proprietary software. However, IBM's software is provided to us due to the fortunate initiative. Other such software that are non-IBM may add on to the expenses if inevitable. **Revenue Streams:** Subscription fees, unlocking premium features, expanding the storage of the application etc. **Customer Segments:** Students, Interested individuals, Family Members, Working Professionals **Customer Relationships:** All the customer segments will be treated alike. Thus, all the users will be treated in a strictly professional manner, i.e every user will be treated in a fair manner, a prospective customer with no additional priorities etc.

		Channels: The application will be publicized through the usage of various social media platforms and through word of mouth. As users begin to use the application, ratings in Google, Play Store, and App Store would increase, resulting in a huge influx of customers.
6.	Scalability of the Solution	The application will be scalable based on the requirements of the future. For instance, the application could partner with Payment applications like GPay and provide users with an incentive to spend wisely through GPay rewards. Similarly, the application could be made more advanced and modern by integrating a service bot that could aid users in customizing the app with ease.

Link: Proposed solution

3.4 Problem Solution Fit

Link : [Problem-Solution fit]

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

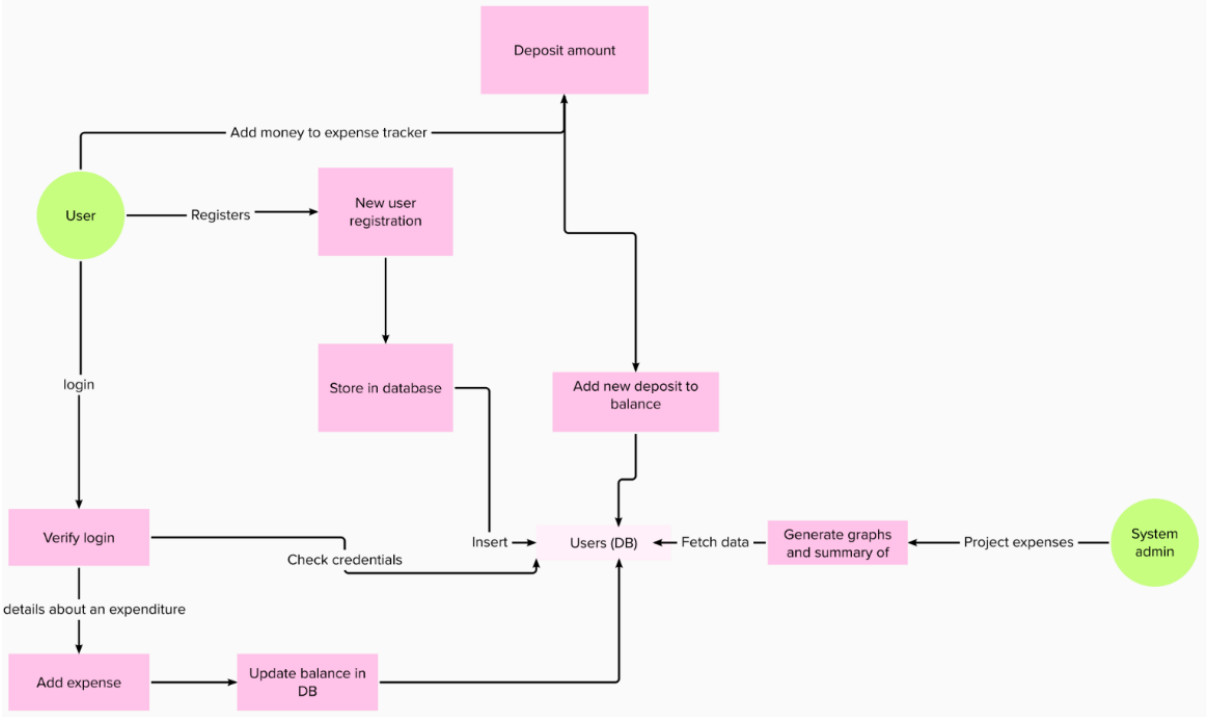
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through App's UI that is connected to IBM DB2.
FR-2	User Login	Login through App's UI that is connected to IBM DB2.
FR-3	User Confirmation	Confirmation via Email using Sendgrid.
FR-4	User Balance Initialization	Initialization of user balance through App's UI that is connected to DB2.
FR-5	Add an expense	Obtain details through App's UI and insert the same in IBM DB2.
FR-6	Modify an expense	Obtain details through App's UI an update the same in IBM DB2.
FR-7	Create a category of expenses	Obtain details of category and insert the same in IBM DB2. Used to render the list of categories while adding/modifying an expense.
FR-8	Summary of expenses	Summarize expenses in a particular time period as per the user's request. Summarize expenses in a selected category. Display analytics of the expenses in a visual form.
FR-9	Monthly expenditure limit	User may set a monthly limit and will be alerted through Sendgrid when expenditure exceeds the limit.
FR-10	Add a periodic expense.	User defines an expense to be made periodically. User is alerted to pay the expense through Sendgrid. Obtain details such as time period and amount through App's UI and insert the same in IBM DB2.
FR-11	Add a delayed expense.	User defines an expense to be made after a fixed amount of time. User is alerted to pay the expense through Sendgrid. Obtain details such as time period and amount through App's UI and insert the same in IBM DB2.

4.2 Non-functional Requirements

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	User-friendly UI with internationally recognized icons and colours that go easy on the eye. Highly responsive and error-friendly.
NFR-2	**Security**	User information such as login information and expense data are stored securely in the cloud. Access to IBM DB2 is purely restricted to developers.
NFR-3	**Reliability**	All transactions are independent follow the ACID properties.
NFR-4	**Performance**	Near instantaneous retrieval of information/services requested by the user.
NFR-5	**Availability**	Application is available 24/7 provided the user has Internet access.
NFR-6	**Scalability**	Since this is a cloud-based application, CPU time is supplied on demand.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution and Technical Architecture

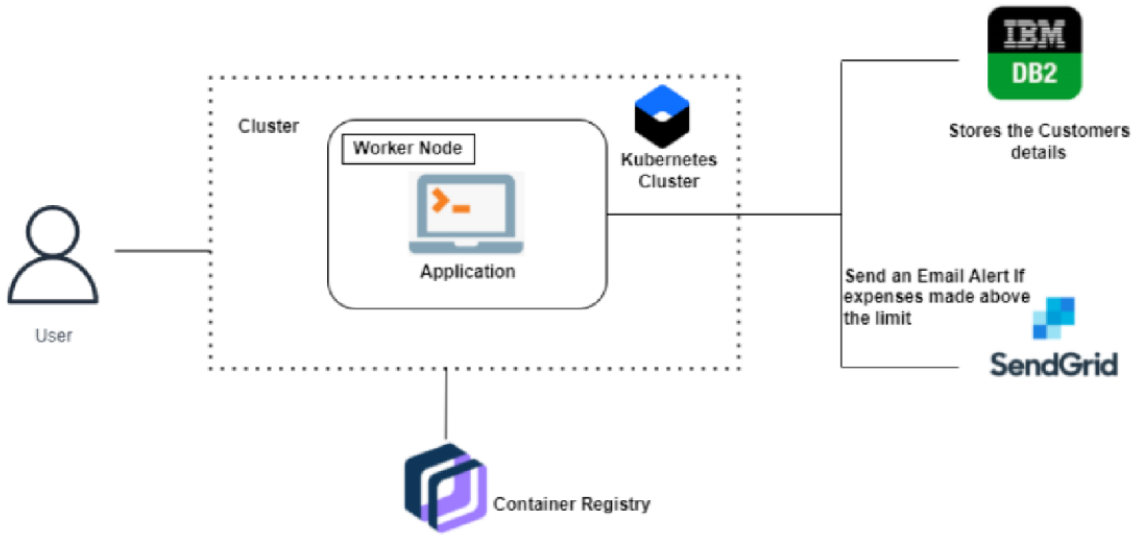
MVP Idea: Web application that tracks a user's expenses.

Product Requirements:

1. Initialize balance on signing up.
2. Update balance whenever necessary.
3. Add expenses with optional descriptions.
4. Group related expenses.
5. Modify an expense.
6. Display remaining balance.
7. Display graph of expenditure (over a user-specified period of time).
8. Set a monthly limit on expenditure.
9. Notify users when they exceed their monthly limit.
10. Add specialized support for different types of users such as students, family, etc.

Priority Product Requirements:

1. User authentication.
2. Initialize balance on signing up.
3. Update balance whenever necessary.
4. Add expenses with optional descriptions.
5. Display remaining balance.
6. Set a monthly limit on expenditure.
7. Display summary of expenses.



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	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
Customer	Confirmation	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
Customer	Login	USN-3	As a user, I can login and access my dashboard and expenses page	I can login and if I have been previously registered I can access my account	High	Sprint-1
Customer	Add expense	USN-4	As a user, I can add an expense	Spent amount must be greater than existing amount and balance must be updated accordingly	High	Sprint-2
Customer	Add amount	USN-5	As a user, I can add money to my account anytime necessary	The amount I added must be updated for future transactions	Medium	Sprint-2
Customer	Dashboard	USN-6	As a user, I can view my dashboard to see balance remaining and last transactions and where I have spent them.	All transactions previously must be updated accordingly and details must be fetched accurately	High	Sprint-2
Customer	Alert the user about crossing expenditure	USN-7	As a user, if I exceed my limit I should be warned with an email.	Limit and amount spent up until then must be tracked accordingly.	Low	Sprint-3
Customer	Show customer summary projections	USN-8	Every month end, as a user, I should be able to view my monthly expenses, projections in the form of dashboards and graphs.	Monthly expenses are kept track of and aesthetic projections are being presented to the user	Low	Sprint-4

6. PROJECT PLANNING AND SCHEDULING

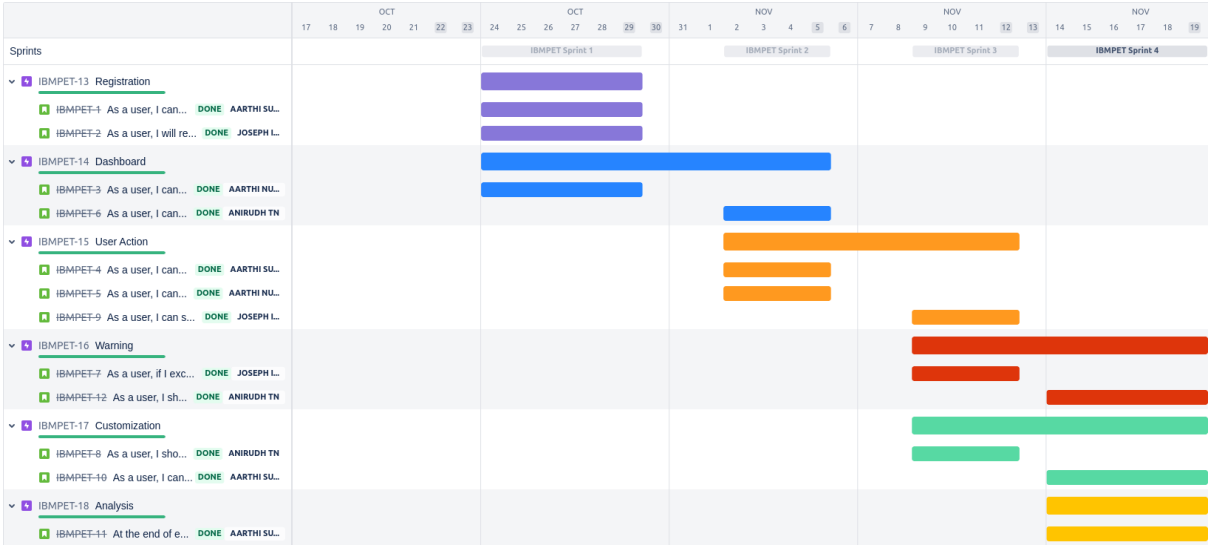
6.1 Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Aarthi N & Aarthi S K
Sprint-1	Registration	USN-2	As a user, I will receive confirmation email once I have registered for the application.	2	High	Anirudh T N & Joseph Amirtharaj
Sprint-1	Dashboard	USN-3	As a user, I can login and access my dashboard and expenses page.	3	High	Aarthi N & Aarthi S K & Anirudh T N & Joseph Amritharaj
Sprint-2	User Action	USN-4	As a user, I can add an expense.	2	High	Aarthi S K
Sprint-2	User Action	USN-5	As a user, I can add money to my account anytime necessary.	2	Medium	Aarthi N
Sprint-2	Dashboard	USN-6	As a user, I can view my dashboard to see balance remaining, the last transactions made, and where I have spent them.	3	High	Anirudh T N & Joseph Amirtharaj
Sprint-3	Warning	USN-7	As a user, if I exceed my limit, I should be warned with an email.	3	Low	Anirudh T N
Sprint-3	Customization	USN-8	As a user, I should be able to set rewards and goals for myself to feel inclined to continue to spend wisely.	5	High	Aarthi N & Aarthi S K & Anirudh T N & Joseph Amritharaj
Sprint-3	User Action	USN-9	As a user, I can set a monthly limit for my expenses.	2	Medium	Aarthi N
Sprint-4	Customization	USN-10	As a user, I can create custom categories that are given to me as a choice when I upload/update an expense.	3	Medium	Aarthi S K
Sprint-4	Analysis	USN-11	At the end of every month, as a user, I should be able to view my monthly expenses, projections in the form of dashboards and graphs.	3	High	Aarthi N & Aarthi S K
Sprint-4	Warning	USN-12	As a user, I should be able to set reminders to alert me of periodic transactions or delayed expenses that are to be completed.	3	Medium	Joseph Amirtharaj

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	06	6 Days	24 Oct 2022	29 Oct 2022	07	- (Meet Planned Date)
Sprint-2	06	6 Days	31 Oct 2022	05 Nov 2022	07	- (Meet Planned Date)
Sprint-3	06	6 Days	07 Nov 2022	12 Nov 2022	10	- (Meet Planned Date)
Sprint-4	06	6 Days	14 Nov 2022	19 Nov 2022	09	- (Meet Planned Date)

6.3 Reports from JIRA



7. CODING AND SOLUTIONING

7.1 Feature - Add Expense

This feature allows the user to add an expense, it takes in the amount spent, the date of the expense, the category under which the expense falls and the group to which the expense belongs. It also takes an optional description for the expense.

7.2 Feature - Update Wallet

This feature allows the user to update the wallet amount. This is useful, for example to reflect the receiving of salary.

7.3 Feature - Dashboard

This allows the user to view the most recent expenses made. It displays each expense along with its details. It is also the home page which the user sees on logging in.

7.4 Feature - Rewards and Goals

This allows the user to specify certain goals regarding expenses, for example, to spend less than a certain amount over a certain period of time. The user can also specify a reward on completion of the goal.

7.5 Feature - Set Monthly Limit

This allows the user to set a limit on how much is spent in a given month. This can be changed. If the user's expenses exceeds the monthly limit, then a warning email is sent to the user.

7.6 Feature - Add Category

This feature allows the user to add his/her own categories. This allows for high customizability.

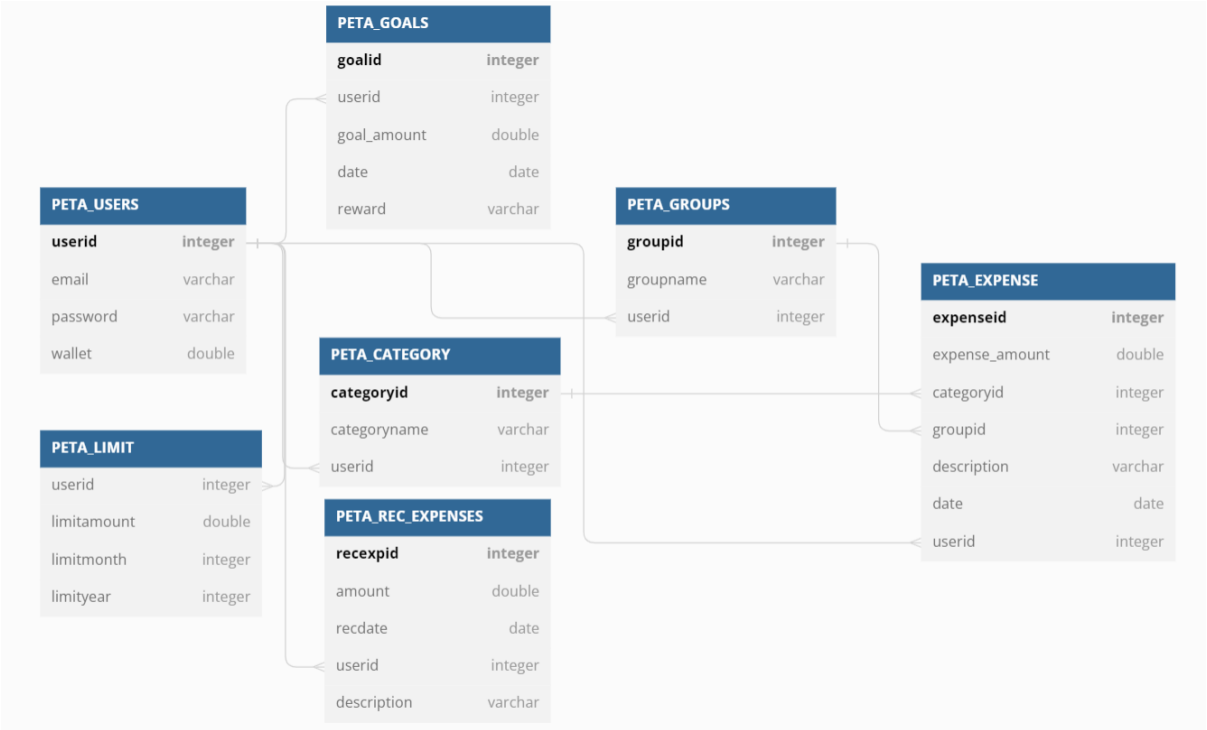
7.7 Feature - Analyse Expenses

This feature allows the user to view how much is spent in a given period of time, and other analyses based on the user's expenses.

7.8 Feature - Add Recurring/Periodic Expenses

This feature allows the user to add an expense that is recurring for every month. This is useful, for example, to model bills that need to be paid.

7.9 Database Schema



8. TESTING

8.1 Test Cases

- Verify that the user is able to login successfully on entering appropriate credentials.
- The UI elements, such as the card for the login, the button to submit etc are functional and are rendered properly in all devices.
- Users who enter invalid credentials will not be redirected to the dashboard
- Verify that users are able to register themselves to the application.
- The UI elements, such as the card for the registration, the button to submit etc are functional and are rendered properly in all devices.
- The user should not be able to register successfully if any of the fields are left empty.
- The user should receive the email from nunnaaarthi@gmail.com, stating a successfully registration.
- "The UI part of the dashboard,
- the side navbar, the logout option, the cards showing various expenses, and the wallet edit icon must be functioning and rendered properly"
- The UI part of the add expense page must be rendered and functioning.
- "The user must be able to
- add an expense. "
- The user should not be able to add the same expense again.
- The user should be able to update the balance and it must be reflected in the dashboard where the wallet balance is displayed.
- The user views all the UI components rendered properly and functioning accordingly.
- The user should be able to set a monthly limit for his expenditures.
- The page must render properly and function appropriately.
- The user must be able to view the analysis of their expenses.
- The UI which contains 2 graphs must be rendered properly.
- The user must be able to view the analysis of their expenses.
- The UI is rendered properly and the various components are functioning properly.
- The UI is rendered properly and the various components are functioning properly.
- This page allows the user to view the recurring expenses that he or she has created using the add recurring expense page.
- The user must be able to view their existing rewards and goals . They can also use the add reward and goal icon to add the reward and goal.
- The user must be able to add a reward and goal.
- The user must be able to view their existing rewards and goals in a properly rendered manner.
- The user must be able view the UI components to add a goal in the appropriate manner.
- The user must be able to create a new category for themselves.
- The UI components of the add category page is rendered properly and functions appropriately.
- The user views the expense cards, and clicks on a particular card's edit icon to modify the expense
- The user views the expense cards displaying the expenses available to edit.

8.2 User Acceptance Testing

Link to the UAT:

https://github.com/IBM-EPBL/IBM-Project-17227-1659631598/blob/main/Final%20Deliverables/Testcases%20Report.xlsx

9. RESULTS

9.1 Performance Metrics

NFT-Risk assessment:

S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes
1	Personal Expense Tracker v1	New	High	No Changes
2	Personal Expense Tracker v2	Existing	Moderate	No Changes

Software Changes	Impact of Downtime	Load/Volume Changes
Moderate	Considerable time taken to insert into DB	No Changes
Moderate	Faster DB operations	No Changes

Risk Score	Justification
ORANGE	Adding this feature makes is coherant with features like adding group and hence, functional changes is high
ORANGE	Updated SQL statements for faster DB operations

NFT - Detailed Test Plan:

S.No	Project Overview	NFT Test approach	Assumptions/Dependencies/Risks
1	Personal Expense Tracker v1	Load testing	Locust installed, app deployed with docker
2	Personal Expense Tracker v2	Stress testing	Locust installed, app deployed with docker
3	Personal Expense Tracker v2	Spike testing	Locust installed, app deployed with docker
4	Personal Expense Tracker v2	Reliability testing	Locust installed, app deployed with docker

End Of Test Report:

S.No	Project Overview	NFT Test approach	NFR - Met
1	Personal Expense Tracker v1	Load testing	Failures
2	Personal Expense Tracker v2	Stress testing	Response time
3	Personal Expense Tracker v2	Spike testing	Response time
4	Personal Expense Tracker v2	Reliability testing	Number of failures

Test Outcome	GO/NO-GO decision
Number of failures spikes from 0.3 to 0.9 with increase in number of users from 70 to 100	GO
Increases from 3100ms for one user to 19000ms for 20 users	GO
Varies from minimum if 1367ms and maximum of 25609ms for a specific feature of adding expenses	GO
41% failures with adding expense with spike in user data and incorrect input values, but maintained average of 13000ms response time	GO

Recommendations	Identified Defects (Detected/Closed/Open)
Provide checks from input value before inserting into database	Bad request error on adding expense without primary, required rows
Provide checks from input value before inserting into database	internal server error on adding expense without primary, required rows

Link : Risk Assessment

10. ADVANTAGES & DISADVANTAGES

Advantages

1. An added expense can also be modified. Hence an error that is made while adding an expense is easily rectified.
2. Rewards and goals may be used to motivate the user to spend less.
3. The expenses can be analysed to understand the patterns of expenditure.
4. Recurring expenses can be added to model bills and such.
5. Categories can be added for higher customizability.

Disadvantages

1. Complex analysis based on categories is not supported.
2. Limits cannot be imposed on arbitrary periods of time, but only on a single month.
3. Recurring expenses have a period of exactly one month, which is not customizable.
4. It requires the user to manually enter expense information accurately.
5. The user is the one who is responsible for rewarding himself on the completion of a goal.

11. CONCLUSION

A Personal Expense Tracker supporting myriad features has been designed and implemented using IBM's cloud based tools such as DB2, IBM Object Storage, etc. SendGrid is used to send mails to users. Docker and Kubernetes are used for deployment of the application.

12. FUTURE SCOPE

Future avenues of work in this application would include:

1. Design and implementation of complex analyses based on various parameters such as categories, groups, etc.
2. Prediction of expenses based on machine learning.
3. Adding support for limits imposed on arbitrary periods of time.
4. Adding support for recurring expenses that have a user-specified period.
5. Having the application itself reward the user for accomplishment of various goals, such as coupons, vouchers, etc.

13. APPENDIX

13.1 Source Code

app.py

```
from flask import Flask, render_template, request, redirect, url_for
```

```
from flask_mail import Mail, Message
from datetime import datetime
from flask_cors import CORS, cross_origin
import ibm_db
import json
import plotly
import plotly.graph_objs as go
import pandas as pd
from flask import send_file
from io import BytesIO
import matplotlib.pyplot as plt
import numpy as np
import base64
from PIL import Image
import time
import atexit
from datetime import datetime
from apscheduler.schedulers.background import BackgroundScheduler

app = Flask(__name__, template_folder='templates')
app.config['SECRET_KEY'] = 'top-secret!'
app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'apikey'
app.config['MAIL_PASSWORD'] =
'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'
app.config['MAIL_DEFAULT_SENDER'] = 'nunnaaarthi@gmail.com'
mail = Mail(app)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'

# GLobal variables
EMAIL = ''
USERID = ''

conn =
ibm_db.connect("DATABASE=bludb;HOSTNAME=54a2f15b-5c0f-46df-8954-7e38e612c2
bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32733;Security=SSL;
SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=nlg66799;PWD=CXtQLAGZ06f
D0fhC;", "", "")

# FUNCTIONS INTERACTING WITH DB #


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)

# END POINTS #
scheduler.start()
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)
```

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/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65Vohhpuu
COmLASjC" 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/t
WtIaxVXM" 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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65Vohhpuu
COmLASjC" 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/t
WtIaxVXM" 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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65Vohhpuu
COmLASjC" 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/t
WtIaxVXM" 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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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/personalexpens
etrackercapd/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 %}
```

Dockerfile

```
FROM debian:stable
COPY . ./server
WORKDIR /server
RUN apt-get update
RUN apt-get install -y apt-utils
RUN apt install -y build-essential libxml2
RUN apt install -y python3
RUN apt install -y python3-pip
RUN rm -rf /var/lib/apt/lists/*
RUN apt-get update
RUN pip3 install -r requirements.txt
EXPOSE 5000
CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0"]
```

requirements.txt

```
APScheduler==3.9.1.post1
blinker==1.5
click==8.1.3
contourpy==1.0.6
cycler==0.11.0
Flask==2.2.2
Flask-Cors==3.0.10
Flask-Mail==0.9.1
fonttools==4.38.0
ibm-db==3.1.3
itsdangerous==2.1.2
Jinja2==3.1.2
kiwisolver==1.4.4
MarkupSafe==2.1.1
matplotlib==3.6.2
numpy==1.23.4
packaging==21.3
pandas==1.5.1
Pillow==9.3.0
plotly==5.11.0
pyparsing==3.0.9
python-dateutil==2.8.2
pytz==2022.6
pytz-deprecation-shim==0.1.0.post0
six==1.16.0
tenacity==8.1.0
```

```
tzdata==2022.6
tzlocal==4.2
Werkzeug==2.2.2
```

13.2 Github and Project Demo Link

Github Project Link
Project Demo Link