PROJECT REPORT

Abstraction:

The main motive of this project is to provide a platform where every person can track and analysis monthly expenses in easy way. In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your 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 expense 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.

INTRODUCTION

1.1 Project Overview

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

1.2 Purpose

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

LITERATURE SURVEY

2.1 Existing problem

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

2.2 References

[1] TITLE: Expense Tracker : A Smart Approach to Track Everyday Expense (Year2020)
AUTHOR: Ms.J. Angelin Blessy, Anant Prakash Singh, Navneet Kumar, HrithikGupta

SUMMARY:
Expense Tracker is a day-to-day expense management system designed to easily and efficiently track the daily expenses through a computerized system that eliminates the need for manual paper tasks that systematically maintains records and easily accesses data stored by the user.

[2] TITLE: Budget Estimator Android Application (Year 2019) AUTHOR: Namita Jagtap, Priyanka Joshi, Aditya Kamble SUMMARY:
This application takes income from user and divides in daily expense allowed, if user exceed that days expense it will cut if from your income and give new daily expense allowed amount, and if that days expense is less it will add it in savings. Expense tracking application will generate report at the end of month to show income-expense via multiple graphs.

[3] TITLE: VoiceControlled Web Application(Year 2022)
AUTHOR: Raj Gandhi, RomilDesai, Marmik Modi,Dr. Suvarna Pansambal
SUMMARY:
Voice or speech recognition systems allow a user to make a hands-free request to the computer, which in turn processes the request and serves the user with appropriate responses. A voice-controlled system embedded in a web application can enhance user experience and can provide voice as a means to control the functionality

2.3 Problem Statement Definition

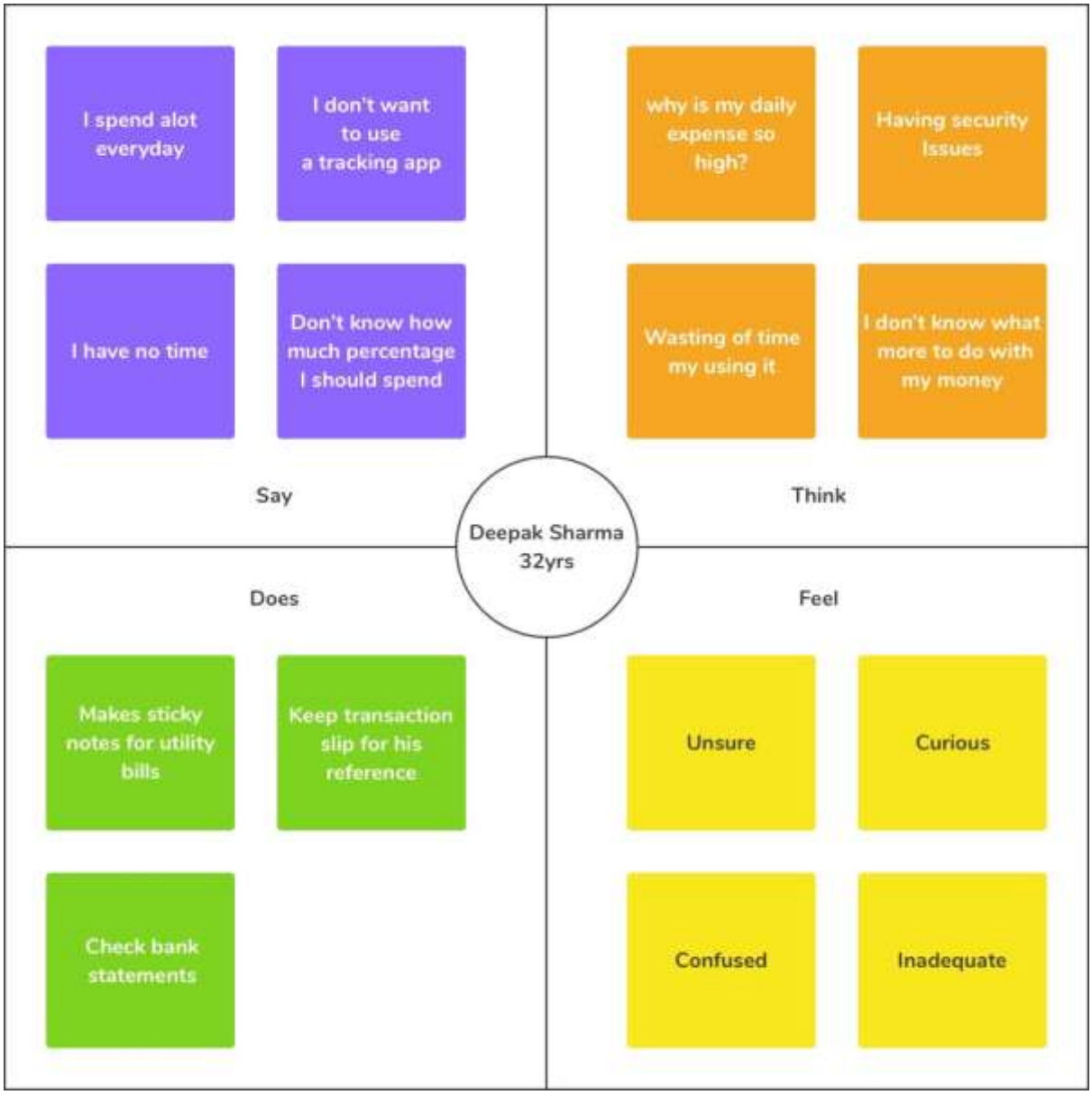
Customer Problem Statement Template:

A well-articulated customer problem statement allows us to find the ideal solution for the challenges our customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

IDEATION &PROPOSED SOLUTION

3.1 Empathy Map Canvas

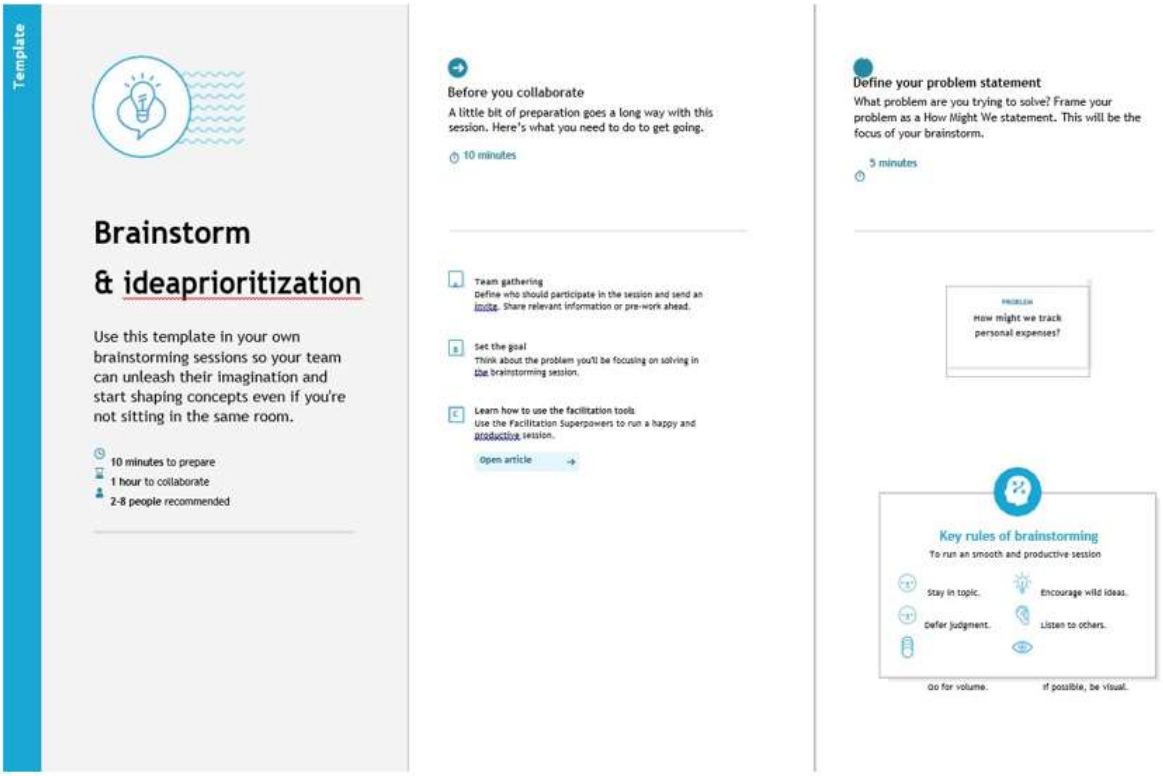
The empathy map was created as a tool to help you gain an understanding of a targeted persona. Thus you can use it when you want to deliver a better user experience of your product/service. In the process, the exercise can also help you identify the things you don't know about your users yet so you can carry out new research to fill in those gaps.



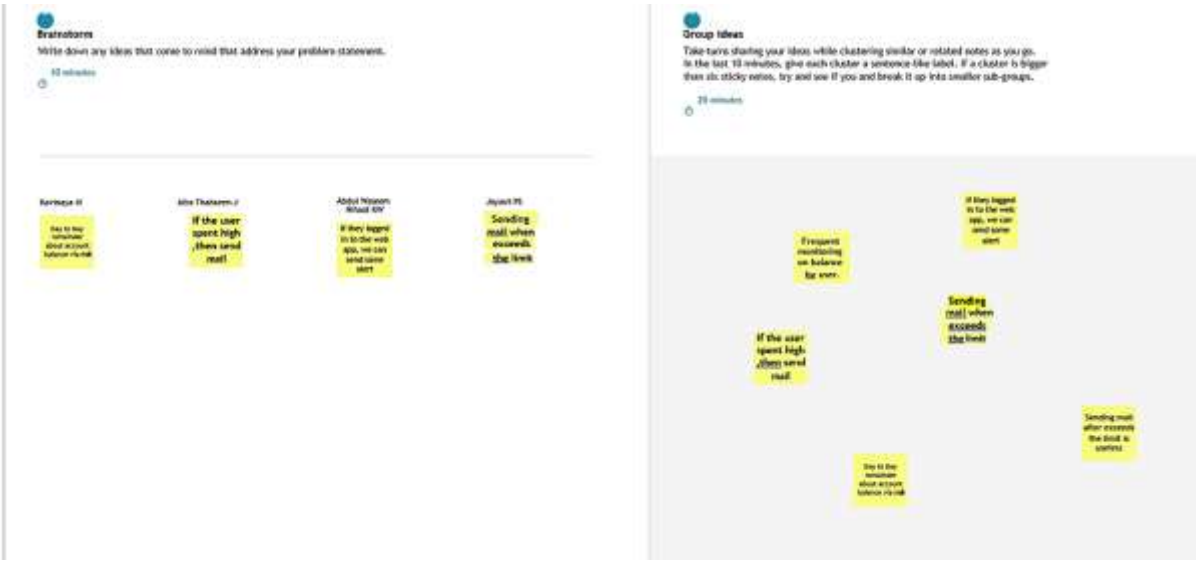
3.2 Ideation & Brainstroming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

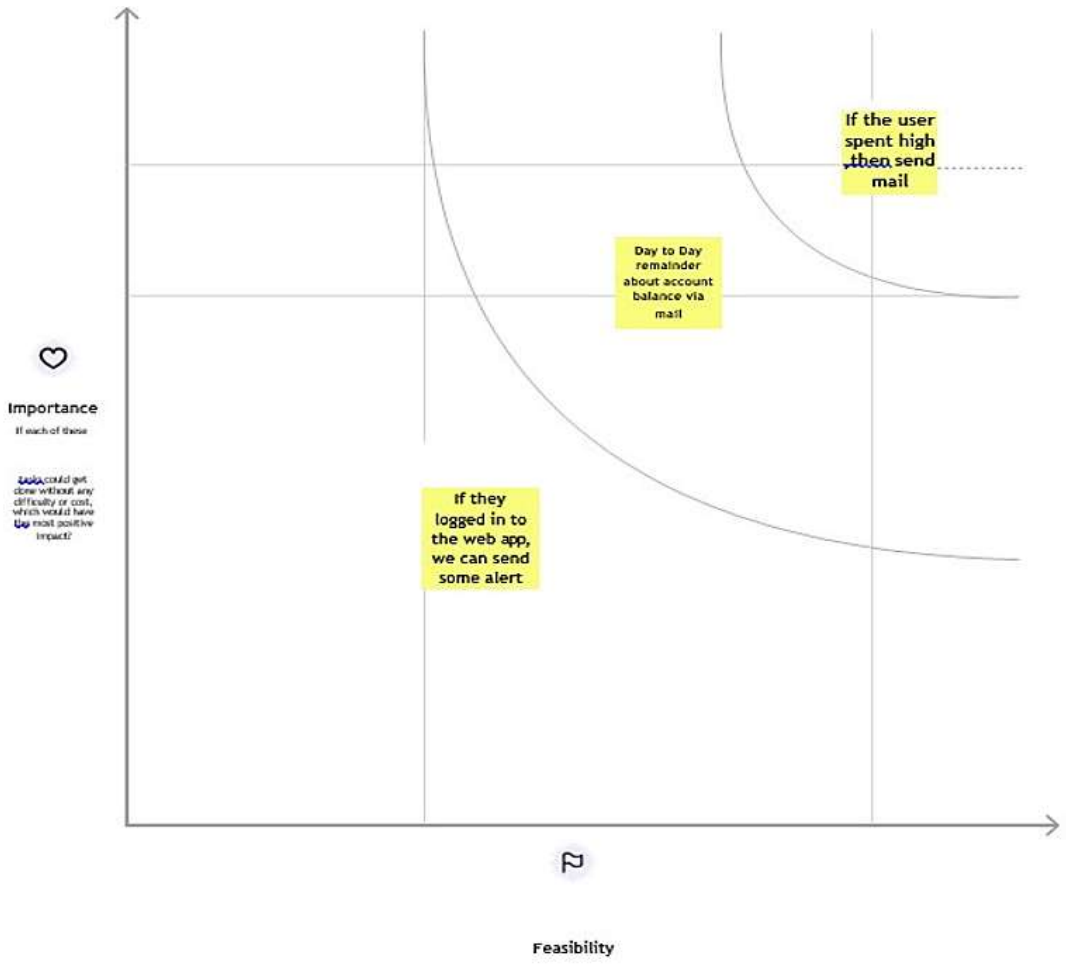
Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization



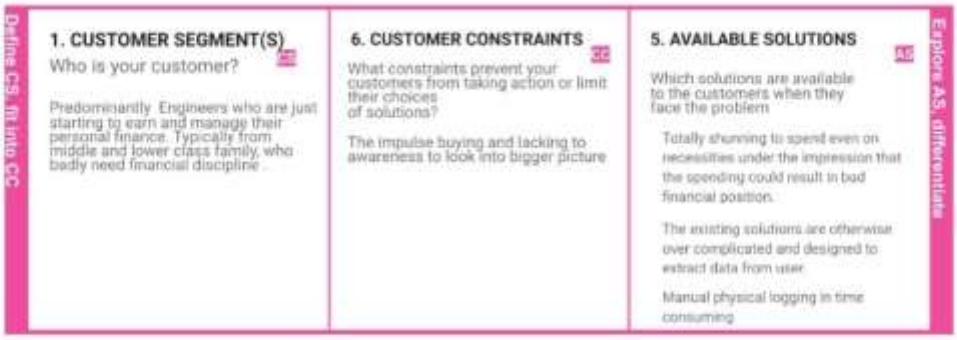
3.3 Proposed Solution

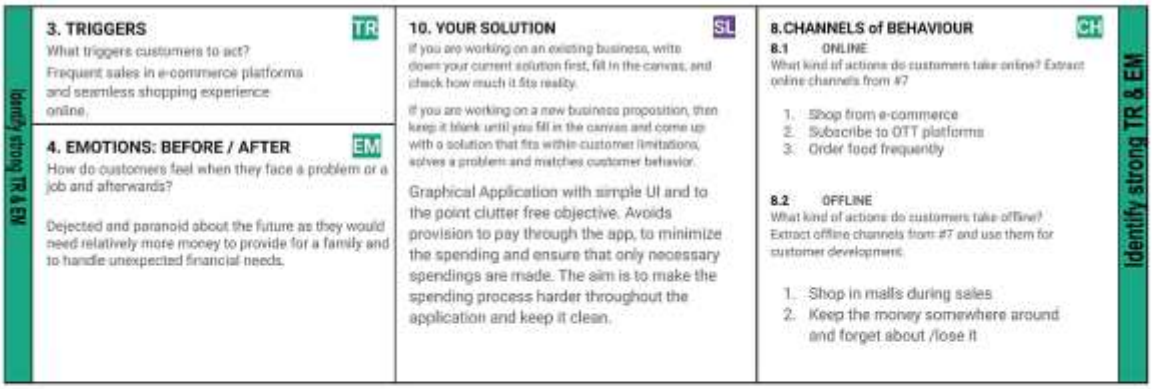
Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Keeping Proper track of our daily expenses is becoming challenging in today's world. Without the proper money management knowledge people overspend on their wants instead of focusing on their needs. Especially when using online applications for purchasing their requirements consumers tend to over spend. This problem leads to improper distribution of their daily expenses. Without proper knowledge on managing money poor are becoming poorer and rich are becoming richer.
2.	Idea / Solution description	An attempt to develop an app to manage our daily expenses and give us insights on managing our money would be a good idea. This app will be able to track expenses on various online platforms and apps. The app can help with proper budgeting and give alerts when the user over spends or crosses the limit previously set by them. This will lead to proper spending habits and make them knowledgeable about money management. IBM cloud can be used to handle the data safely.
3.	Novelty / Uniqueness	The speciality for the app will be the data security with IBM cloud being used for data storage and this app genuinely helps with the money management. The proper and personalized budgeting of the user's money leads them to trust the app and they wouldn't have to worry about their expenditure on unnecessary things.
4.	Social Impact / Customer Satisfaction	People using the app will be becoming better at their spending habits and will be able to save more than their peers who are not using the app. This application aims to improve the users' savings sustainably and steadily which leads them to trust the app without worrying about their money.

5.	Business Model (Revenue Model)	This application leads to a business model, the user can be suggested the right products to buy based on their budget and this can lead to targeted business approaching the right consumers. The model leads to systematic and structured expenses of the user and also leads to predictive analysis of the future expenses of the consumer. This model makes the user more careful with expenses as they are provided with the money management insights.
6.	Scalability of the Solution	This application can be created as a multi user model nationwide. The model can also be modified based on the country's law on applications and data security which leads to international implementation of this application by maintaining proper gateway rules. This app when developed for multiple nations can be modified to their requirements. The app can also be modified for a particular group of people or organization.

3.4 Problem Solution





Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act?

Frequent sales in e-commerce platforms and seamless shopping experience online.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?

Dejected and paranoid about the future as they would need relatively more money to provide for a family and to handle unexpected financial needs.

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.

If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behavior.

Graphical Application with simple UI and to the point clutter free objective. Avoids provision to pay through the app, to minimize the spending and ensure that only necessary spendings are made. The aim is to make the spending process harder throughout the application and keep it clean.

8.CHANNELS of BEHAVIOUR CH

8.1 ONLINE

What kind of actions do customers take online? Extract online channels from #7

1. Shop from e-commerce
2. Subscribe to OTT platforms
3. Order food frequently

8.2 OFFLINE

What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

1. Shop in malls during sales
2. Keep the money somewhere around and forget about /lose it

Identify strong TR & EM

4.FUNCTIONAL REQUIREMENT

4.1 Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Email/SignUp Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Add expenses	Enter the everyday expenses Split it into categories(example : food, petrol,movies)
FR-4	Reminder mail	Sending reminder mail on target (for ex : if user wants a reminder when his/her balance reaches some amount(5000)) Sending reminder mail to the user if he/she has not filled that day's expenses.
FR-5	Creating Graphs	Graphs showing everyday and weekly expenses. Categorical graphs on expenditure.
FR-6	Add salary	Users must enter the salary at the start of the month.
FR-7	Export CSV	User can export the raw data of their expenditure as CSV

4.2 Non-Functional requirements

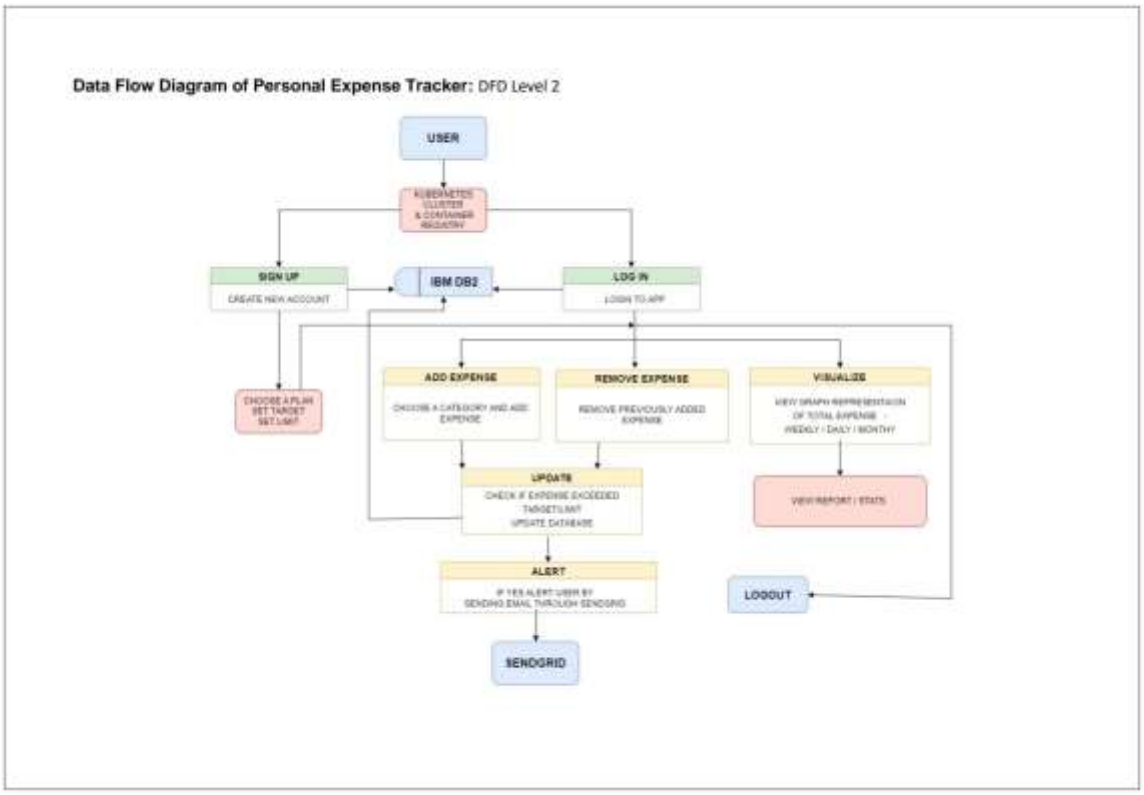
Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	A simple web application which is accessible across devices
NFR-2	**Security**	The OAuth Google sign in and email login are secure with hashed and salted secure storage of credentials.
NFR-3	**Reliability**	Containerized service ensures that new instance can kick up when there is a failure
NFR-4	**Performance**	The load is managed through the load balancer used with docker. Thus ensuring good performance
NFR-5	**Availability**	With load balancing and multiple container instances, the service is always available.
NFR-6	**Scalability**	Docker and Kubernetes are designed to accommodate scaling based on need

5. PROJECT DESIGN

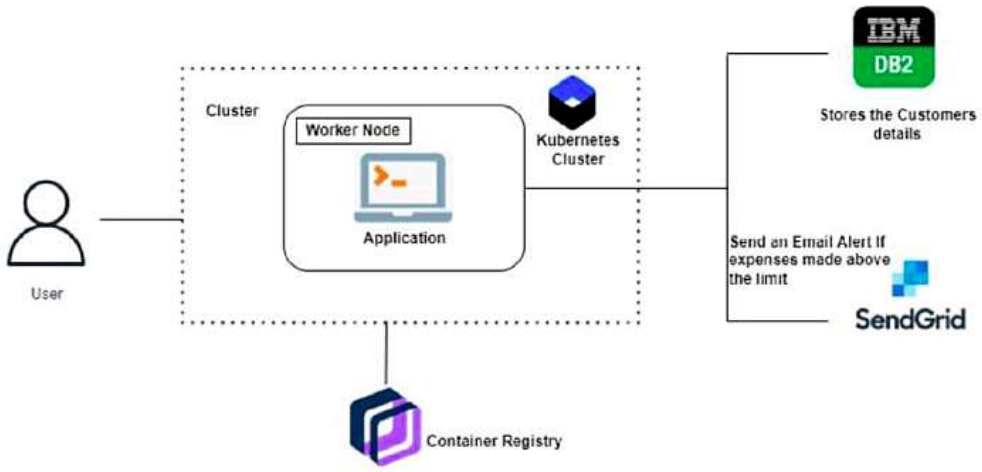
5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture

Technical Architecture:

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



5.3 User Stories

User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.
	Login	USN-2	As a user, I can log into the application by entering email & password
	Add	USN -3	As a user , I can add in new expenses.
	Remove	USN – 4	As a user , I can remove previously added expenses.
	View	USN - 5	As a user , I can view my expenses in the form of graphs and get insights.
	Get alert message	USN - 6	As a user , I will get alert messages if I exceed my target amount.
Administrator	Add / remove user	USN – 7	As admin , I can add or remove user details on db2 manually.
		USN - 8	As admin , I can add or remove user details on sendgrid.

SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement(Epic)	User StoryNumber	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for t heapplication by entering my email,password, andconfirming my password.	2	High	Nandakisore R, Nandha Kumar D
Sprint-1		USN-2	As a user, I will receive confirmation email once I haveregistered for the application	1	High	Naveen VP, Nandhini P
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email &password	2	High	Nanda kisore R,Nandhini P
Sprint- 2	Dashboard	USN-4	As a user, I can update the daily expenses.	3	High	Nandha kumar D,Naveen VP
Sprint- 2		USN-5	As a user, I can visualize their expenditure.	2	High	Naveen VP, Nandhini P
Sprint- 4		USN-6	As a user I can view the stats page which shows the average daily expenditure,compares with previous month expenditure.	2	Medium	Nanda kisore R,Nandha kumar D

Sprint- 3		USN-7	As a user , I can update the icome and salary	2	High	Nandhini P
Sprint - 4	Chatbot	USN-8	As a user,I can get answers tomy queriesusing the chatbot	2	Medium	Nandhakumar D
Sprint-3	Notification	USN-9	As a user, I will get expenditure statmentvia email upona request	1	High	Nandakisore R
Sprint-3		USN-10	As a user, I will get notification on high expenses made and monthly expenses	1	High	Naveen VP

6.2 SPIRIT DELIVERY SCHEDULE

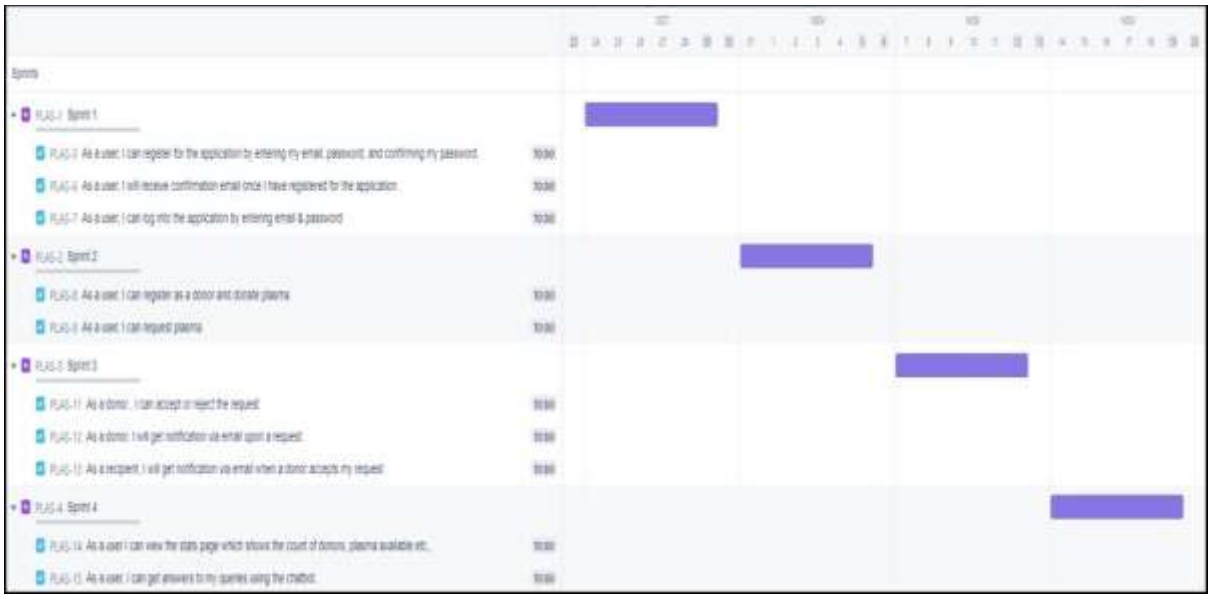


Fig 3.1.Sprint Delivery Schedule

6.3 REPORTS FROM JIRA

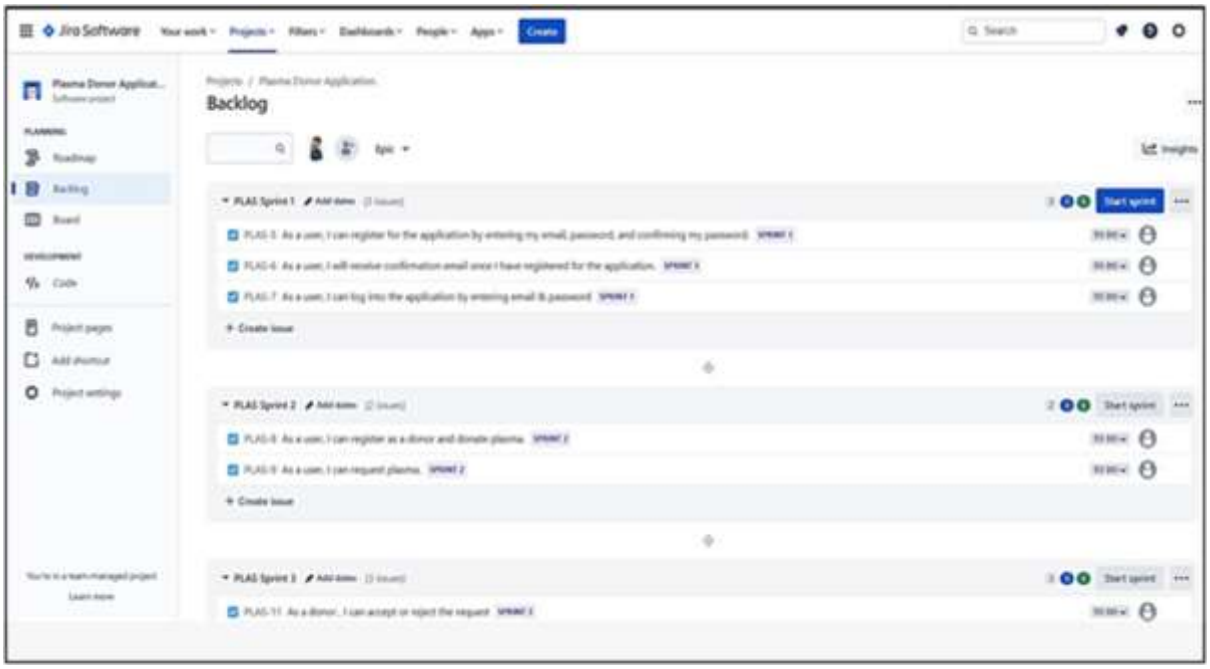


Fig 3.2.Sprint report

CODING & SOLUTIONING

FEATURE 1: REGISTRATION OF USER

```
app.config['database'] = 'bludb'
app.config['hostname'] = 'ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud'
app.config['port'] = '31321'
app.config['protocol'] = 'tcpip'
app.config['uid'] = 'vmk08423'
app.config['pwd'] = '3KfJl6HGDtPdbIWye'
app.config['security'] = 'SSL'
try:
   mysql = DB2(app)

   conn_str='database=bludb;hostname=ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;port=31321;protocol=tcpi
p;\
         uid=vmk08423;pwd=3KfJl6HGDtPdbIWy;security=SSL'
   ibm_db_conn = ibm_db.connect(conn_str,'','')

   print("Database connected without any error !!")
except:
   print("IBM DB Connection error   :     " + DB2.conn_errormsg())
```

```
# app.config['']

# mysql = MySQL(app)


#HOME--PAGE
@app.route("/home")
def home():
    return render_template("homepage.html")

@app.route("/")
def add():
    return render_template("home.html")



#SIGN--UP--OR--REGISTER


@app.route("/signup")
def signup():
    return render_template("signup.html")



@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    print("Break point1")
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

        print("Break point2" + "name: " + username + "------" + email + "------" + password)

        try:
            print("Break point3")
            connectionID = ibm_db_dbi.connect(conn_str, '', '')
            cursor = connectionID.cursor()
            print("Break point4")
        except:
            print("No connection Established")

        # cursor = mysql.connection.cursor()
        # with app.app_context():
        #     print("Break point3")
        #     cursor = ibm_db_conn.cursor()
        #     print("Break point4")

        print("Break point5")
        sql = "SELECT * FROM register WHERE username = ?"
```

```
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.execute(stmt)
    result = ibm_db.execute(stmt)
    print(result)
    account = ibm_db.fetch_row(stmt)
    print(account)

    param = "SELECT * FROM register WHERE username = " + "\'" + username + "\'"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    print("---- ")
    dictionary = ibm_db.fetch_assoc(res)
    while dictionary != False:
      print("The ID is : ", dictionary["USERNAME"])
      dictionary = ibm_db.fetch_assoc(res)

    # dictionary = ibm_db.fetch_assoc(result)
    # cursor.execute(stmt)

    # account = cursor.fetchone()
    # print(account)

    # while ibm_db.fetch_row(result) != False:
    #     # account = ibm_db.result(stmt)
    #     print(ibm_db.result(result, "username"))

    # print(dictionary["username"])
    print("break point 6")
    if account:
      msg = 'Username already exists !'
    elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
      msg = 'Invalid email address !'
    elif not re.match(r'[A-Za-z0-9]+', username):
      msg = 'name must contain only characters and numbers !'
    else:
      sql2 = "INSERT INTO register (username, email,password) VALUES (?, ?, ?)"
      stmt2 = ibm_db.prepare(ibm_db_conn, sql2)
      ibm_db.bind_param(stmt2, 1, username)
      ibm_db.bind_param(stmt2, 2, email)
      ibm_db.bind_param(stmt2, 3, password)
      ibm_db.execute(stmt2)
      # cursor.execute('INSERT INTO register VALUES (NULL, % s, % s, % s)', (username,
email,password))
      # mysql.connection.commit()
      msg = 'You have successfully registered !'
    return render_template('signup.html', msg = msg)

@app.route("/signin")
def signin():
  return render_template("login.html")

@app.route('/login',methods =['GET', 'POST'])
```

```
def login():
    global userid
    msg = ''


    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        # cursor = mysql.connection.cursor()
        # cursor.execute('SELECT * FROM register WHERE username = % s AND password = %
s', (username, password ),)
        # account = cursor.fetchone()
        # print (account)

        sql = "SELECT * FROM register WHERE username = ? and password = ?"
        stmt = ibm_db.prepare(ibm_db_conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        result = ibm_db.execute(stmt)
        print(result)
        account = ibm_db.fetch_row(stmt)
        print(account)

        param = "SELECT * FROM register WHERE username = '' + "\'" + username + "\'" + '' and
password = '' + "\'" + password + "\'"
        res = ibm_db.exec_immediate(ibm_db_conn, param)
        dictionary = ibm_db.fetch_assoc(res)

        # sendmail("hello sakthi","sivasakthisairam@gmail.com")

        if account:
            session['loggedin'] = True
            session['id'] = dictionary["ID"]
            userid = dictionary["ID"]
            session['username'] = dictionary["USERNAME"]
            session['email'] = dictionary["EMAIL"]

            return redirect('/home')
        else:
            msg = 'Incorrect username / password !'

    return render_template('login.html', msg = msg)
```

#ADDING----DATA

```
@app.route("/add")
def adding():
    return render_template('add.html')


@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():

    date = request.form['date']
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category = request.form['category']
    print(date)
    p1 = date[0:10]
    p2 = date[11:13]
    p3 = date[14:]
    p4 = p1 + "-" + p2 + "." + p3 + ".00"
    print(p4)
    # cursor = mysql.connection.cursor()
    # cursor.execute('INSERT INTO expenses VALUES (NULL,  % s, % s, % s, % s, % s, % s)',
(session['id'] ,date, expensename, amount, paymode, category))
    # mysql.connection.commit()
    # print(date + " " + expensename + " " + amount + " " + paymode + " " + category)
    sql = "INSERT INTO expenses (userid, date, expensename, amount, paymode, category)
VALUES (?, ?, ?, ?, ?, ?)"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, session['id'])
    ibm_db.bind_param(stmt, 2, p4)
    ibm_db.bind_param(stmt, 3, expensename)
    ibm_db.bind_param(stmt, 4, amount)
    ibm_db.bind_param(stmt, 5, paymode)
    ibm_db.bind_param(stmt, 6, category)
    ibm_db.execute(stmt)

    print("Expenses added")

    # email part

    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND
MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp)
ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
```

```
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    total=0
    for x in expense:
         total += x[4]

    param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER
BY id DESC LIMIT 1"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = 0
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMITSS"])
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[0]

    if total > int(s):
        msg = "Hello " + session['username'] + " , " + "you have crossed the monthly limit of Rs. "
+ s + "/- !!!" + "\n" + "Thank you, " + "\n" + "Team Personal Expense Tracker."
        sendmail(msg,session['email'])

    return redirect("/display")



#DISPLAY---graph

@app.route("/display")
def display():
    print(session["username"],session['id'])

    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT * FROM expenses WHERE userid = % s AND date ORDER BY
`expenses`.`date` DESC',(str(session['id'])))
    # expense = cursor.fetchall()

    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " ORDER BY
date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
```

```
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    return render_template('display.html' ,expense = expense)




#delete---the--data

@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
def delete(id):
    #  cursor = mysql.connection.cursor()
    #  cursor.execute('DELETE FROM expenses WHERE  id = {0}'.format(id))
    #  mysql.connection.commit()

    param = "DELETE FROM expenses WHERE  id = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)

    print('deleted successfully')
    return redirect("/display")


#UPDATE---DATA

@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT * FROM expenses WHERE  id = %s', (id,))
    # row = cursor.fetchall()

    param = "SELECT * FROM expenses WHERE  id = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
```

```
        row.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    print(row[0])
    return render_template('edit.html', expenses = row[0])




@app.route('/update/<id>', methods = ['POST'])
def update(id):
  if request.method == 'POST' :

      date = request.form['date']
      expensename = request.form['expensename']
      amount = request.form['amount']
      paymode = request.form['paymode']
      category = request.form['category']

    #   cursor = mysql.connection.cursor()
    #   cursor.execute("UPDATE `expenses` SET `date` = % s , `expensename` = % s , `amount`
= % s, `paymode` = % s, `category` = % s WHERE `expenses`.`id` = % s ",(date, expensename,
amount, str(paymode), str(category),id))
    #   mysql.connection.commit()

      p1 = date[0:10]
      p2 = date[11:13]
      p3 = date[14:]
      p4 = p1 + "-" + p2 + "." + p3 + ".00"

      sql = "UPDATE expenses SET date = ? , expensename = ? , amount = ?, paymode = ?,
category = ? WHERE id = ?"
      stmt = ibm_db.prepare(ibm_db_conn, sql)
      ibm_db.bind_param(stmt, 1, p4)
      ibm_db.bind_param(stmt, 2, expensename)
      ibm_db.bind_param(stmt, 3, amount)
      ibm_db.bind_param(stmt, 4, paymode)
      ibm_db.bind_param(stmt, 5, category)
      ibm_db.bind_param(stmt, 6, id)
      ibm_db.execute(stmt)

      print('successfully updated')
      return redirect("/display")
```

FEATURE 2 - UDAPTE DAILY EXPENSES

PYTHON SNIPPET:

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

```
# from flask_mysqldb import MySQL
# import MySQLdb.cursors
import re

from flask_db2 import DB2
import ibm_db
import ibm_db_dbi
from sendemail import sendgridmail,sendmail

# from gevent.pywsgi import WSGIServer
import os


app = Flask(__name__)

app.secret_key = 'a'

# app.config['MYSQL_HOST'] = 'remotemysql.com'
# app.config['MYSQL_USER'] = 'D2DxDUPBii'
# app.config['MYSQL_PASSWORD'] = 'r8XBO4GsMz'
# app.config['MYSQL_DB'] = 'D2DxDUPBii'
"""
dsn_hostname = "ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
dsn_uid = "vmk08423"
dsn_pwd = "3KfJl6HGDtPdbIWy"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "31321"
dsn_protocol = "tcpip"

dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
    "PORT={3};"
    "PROTOCOL={4};"
    "UID={5};"
    "PWD={6};"
).format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol, dsn_uid,
dsn_pwd)
"""
# app.config['DB2_DRIVER'] = '{IBM DB2 ODBC DRIVER}'
app.config['database'] = 'bludb'
app.config['hostname'] = 'ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud'
app.config['port'] = '31321'
app.config['protocol'] = 'tcpip'
app.config['uid'] = 'vmk084230'
app.config['pwd'] = '3KfJl6HGDtPdbIWy'
app.config['security'] = 'SSL'
try:
```

```
    mysql = DB2(app)

    conn_str='database=bludb;hostname=ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;port=31321;protocol=tcpi
p;\
            uid=vmk08423;pwd=3KfJl6HGDtPdbIWy;security=SSL'
    ibm_db_conn = ibm_db.connect(conn_str,'','')

    print("Database connected without any error !!")
except:
    print("IBM DB Connection error   :     " + DB2.conn_errormsg())

# app.config['']

# mysql = MySQL(app)


#HOME--PAGE
@app.route("/home")
def home():
    return render_template("homepage.html")

@app.route("/")
def add():
    return render_template("home.html")



#SIGN--UP--OR--REGISTER


@app.route("/signup")
def signup():
    return render_template("signup.html")



@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    print("Break point1")
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

        print("Break point2" + "name: " + username + "------" + email + "------" + password)

        try:
            print("Break point3")
            connectionID = ibm_db_dbi.connect(conn_str, '', '')
            cursor = connectionID.cursor()
```

```
    print("Break point4")
except:
    print("No connection Established")

# cursor = mysql.connection.cursor()
# with app.app_context():
#     print("Break point3")
#     cursor = ibm_db_conn.cursor()
#     print("Break point4")

print("Break point5")
sql = "SELECT * FROM register WHERE username = ?"
stmt = ibm_db.prepare(ibm_db_conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.execute(stmt)
result = ibm_db.execute(stmt)
print(result)
account = ibm_db.fetch_row(stmt)
print(account)

param = "SELECT * FROM register WHERE username = " + "\'" + username + "\'"
res = ibm_db.exec_immediate(ibm_db_conn, param)
print("---- ")
dictionary = ibm_db.fetch_assoc(res)
while dictionary != False:
    print("The ID is : ", dictionary["USERNAME"])
    dictionary = ibm_db.fetch_assoc(res)

# dictionary = ibm_db.fetch_assoc(result)
# cursor.execute(stmt)

# account = cursor.fetchone()
# print(account)

# while ibm_db.fetch_row(result) != False:
#     # account = ibm_db.result(stmt)
#     print(ibm_db.result(result, "username"))

# print(dictionary["username"])
print("break point 6")
if account:
    msg = 'Username already exists !'
elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
    msg = 'Invalid email address !'
elif not re.match(r'[A-Za-z0-9]+', username):
    msg = 'name must contain only characters and numbers !'
else:
    sql2 = "INSERT INTO register (username, email,password) VALUES (?, ?, ?)"
    stmt2 = ibm_db.prepare(ibm_db_conn, sql2)
    ibm_db.bind_param(stmt2, 1, username)
    ibm_db.bind_param(stmt2, 2, email)
    ibm_db.bind_param(stmt2, 3, password)
```

```
        ibm_db.execute(stmt2)
        # cursor.execute('INSERT INTO register VALUES (NULL, % s, % s, % s)', (username,
email,password))
        # mysql.connection.commit()
        msg = 'You have successfully registered !'
    return render_template('signup.html', msg = msg)




#LOGIN--PAGE

@app.route("/signin")
def signin():
  return render_template("login.html")

@app.route('/login',methods =['GET', 'POST'])
def login():
  global userid
  msg = ''


  if request.method == 'POST' :
    username = request.form['username']
    password = request.form['password']
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT * FROM register WHERE username = % s AND password = %
s', (username, password ),)
    # account = cursor.fetchone()
    # print (account)

    sql = "SELECT * FROM register WHERE username = ? and password = ?"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.bind_param(stmt, 2, password)
    result = ibm_db.execute(stmt)
    print(result)
    account = ibm_db.fetch_row(stmt)
    print(account)

    param = "SELECT * FROM register WHERE username = '' + "\'" + username + "\'" + '' and
password = '' + "\'" + password + "\'"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)

    # sendmail("hello sakthi","sivasakthisairam@gmail.com")

    if account:
      session['loggedin'] = True
      session['id'] = dictionary["ID"]
      userid = dictionary["ID"]
      session['username'] = dictionary["USERNAME"]
```

```
            session['email'] = dictionary["EMAIL"]

            return redirect('/home')
        else:
            msg = 'Incorrect username / password !'

    return render_template('login.html', msg = msg)









#ADDING----DATA


@app.route("/add")
def adding():
    return render_template('add.html')


@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():

    date = request.form['date']
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category = request.form['category']

    print(date)
    p1 = date[0:10]
    p2 = date[11:13]
    p3 = date[14:]
    p4 = p1 + "-" + p2 + "." + p3 + ".00"
    print(p4)
    # cursor = mysql.connection.cursor()
    # cursor.execute('INSERT INTO expenses VALUES (NULL,  % s, % s, % s, % s, % s, % s)',
(session['id'] ,date, expensename, amount, paymode, category))
    # mysql.connection.commit()
    # print(date + " " + expensename + " " + amount + " " + paymode + " " + category)

    sql = "INSERT INTO expenses (userid, date, expensename, amount, paymode, category)
VALUES (?, ?, ?, ?, ?, ?)"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, session['id'])
    ibm_db.bind_param(stmt, 2, p4)
    ibm_db.bind_param(stmt, 3, expensename)
    ibm_db.bind_param(stmt, 4, amount)
```

```
    ibm_db.bind_param(stmt, 5, paymode)
    ibm_db.bind_param(stmt, 6, category)
    ibm_db.execute(stmt)

    print("Expenses added")

    # email part

    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND
MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp)
ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    total=0
    for x in expense:
         total += x[4]

    param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER
BY id DESC LIMIT 1"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = 0
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMITSS"])
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[0]

    if total > int(s):
        msg = "Hello " + session['username'] + " , " + "you have crossed the monthly limit of Rs. "
+ s + "/- !!!" + "\n" + "Thank you, " + "\n" + "Team Personal Expense Tracker."
        sendmail(msg,session['email'])

    return redirect("/display")
```

```
#DISPLAY---graph

@app.route("/display")
def display():
   print(session["username"],session['id'])

   # cursor = mysql.connection.cursor()
   # cursor.execute('SELECT * FROM expenses WHERE userid = % s AND date ORDER BY
`expenses`.`date` DESC',(str(session['id'])))
   # expense = cursor.fetchall()

   param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " ORDER BY
date DESC"
   res = ibm_db.exec_immediate(ibm_db_conn, param)
   dictionary = ibm_db.fetch_assoc(res)
   expense = []
   while dictionary != False:
      temp = []
      temp.append(dictionary["ID"])
      temp.append(dictionary["USERID"])
      temp.append(dictionary["DATE"])
      temp.append(dictionary["EXPENSENAME"])
      temp.append(dictionary["AMOUNT"])
      temp.append(dictionary["PAYMODE"])
      temp.append(dictionary["CATEGORY"])
      expense.append(temp)
      print(temp)
      dictionary = ibm_db.fetch_assoc(res)

   return render_template('display.html' ,expense = expense)




#delete---the--data

@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
def delete(id):
   #  cursor = mysql.connection.cursor()
   #  cursor.execute('DELETE FROM expenses WHERE  id = {0}'.format(id))
   #  mysql.connection.commit()

   param = "DELETE FROM expenses WHERE  id = " + id
   res = ibm_db.exec_immediate(ibm_db_conn, param)

   print('deleted successfully')
   return redirect("/display")


#UPDATE---DATA
```

```
@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT * FROM expenses WHERE  id = %s', (id,))
    # row = cursor.fetchall()

    param = "SELECT * FROM expenses WHERE  id = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        row.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    print(row[0])
    return render_template('edit.html', expenses = row[0])




@app.route('/update/<id>', methods = ['POST'])
def update(id):
  if request.method == 'POST' :

      date = request.form['date']
      expensename = request.form['expensename']
      amount = request.form['amount']
      paymode = request.form['paymode']
      category = request.form['category']

    #   cursor = mysql.connection.cursor()
    #   cursor.execute("UPDATE `expenses` SET `date` = % s , `expensename` = % s , `amount`
= % s, `paymode` = % s, `category` = % s WHERE `expenses`.`id` = % s ",(date, expensename,
amount, str(paymode), str(category),id))
    #   mysql.connection.commit()

      p1 = date[0:10]
      p2 = date[11:13]
      p3 = date[14:]
      p4 = p1 + "-" + p2 + "." + p3 + ".00"

      sql = "UPDATE expenses SET date = ? , expensename = ? , amount = ?, paymode = ?,
category = ? WHERE id = ?"
```

```
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, p4)
    ibm_db.bind_param(stmt, 2, expensename)
    ibm_db.bind_param(stmt, 3, amount)
    ibm_db.bind_param(stmt, 4, paymode)
    ibm_db.bind_param(stmt, 5, category)
    ibm_db.bind_param(stmt, 6, id)
    ibm_db.execute(stmt)

    print('successfully updated')
    return redirect("/display")
```

```
#limit
@app.route("/limit" )
def limit():
     return redirect('/limitn')

@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
    if request.method == "POST":
       number= request.form['number']
       #  cursor = mysql.connection.cursor()
       #  cursor.execute('INSERT INTO limits VALUES (NULL, % s, % s) ',(session['id'], number))
       #  mysql.connection.commit()

       sql = "INSERT INTO limits (userid, limitss) VALUES (?, ?)"
       stmt = ibm_db.prepare(ibm_db_conn, sql)
       ibm_db.bind_param(stmt, 1, session['id'])
       ibm_db.bind_param(stmt, 2, number)
       ibm_db.execute(stmt)

       return redirect('/limitn')


@app.route("/limitn")
def limitn():
   # cursor = mysql.connection.cursor()
   # cursor.execute('SELECT limitss FROM `limits` ORDER BY `limits`.`id` DESC LIMIT 1')
   # x= cursor.fetchone()
   # s = x[0]

   param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER
BY id DESC LIMIT 1"
   res = ibm_db.exec_immediate(ibm_db_conn, param)
   dictionary = ibm_db.fetch_assoc(res)
```

```
    row = []
    s = " /-"
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMITSS"])
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[0]

    return render_template("limit.html" , y= s)

#REPORT

@app.route("/today")
def today():
    #   cursor = mysql.connection.cursor()
    #   cursor.execute('SELECT TIME(date)   , amount FROM expenses  WHERE userid = %s
AND DATE(date) = DATE(NOW()) ',(str(session['id'])))
    #   texpense = cursor.fetchall()
    #   print(texpense)

      param1 = "SELECT TIME(date) as tn, amount FROM expenses WHERE userid = " +
str(session['id']) + " AND DATE(date) = DATE(current timestamp) ORDER BY date DESC"
      res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
      dictionary1 = ibm_db.fetch_assoc(res1)
      texpense = []

      while dictionary1 != False:
          temp = []
          temp.append(dictionary1["TN"])
          temp.append(dictionary1["AMOUNT"])
          texpense.append(temp)
          print(temp)
          dictionary1 = ibm_db.fetch_assoc(res1)

    #   cursor = mysql.connection.cursor()
    #   cursor.execute('SELECT * FROM expenses WHERE userid = % s AND DATE(date) =
DATE(NOW()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
    #   expense = cursor.fetchall()

      param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND
DATE(date) = DATE(current timestamp) ORDER BY date DESC"
      res = ibm_db.exec_immediate(ibm_db_conn, param)
      dictionary = ibm_db.fetch_assoc(res)
      expense = []
      while dictionary != False:
          temp = []
          temp.append(dictionary["ID"])
          temp.append(dictionary["USERID"])
          temp.append(dictionary["DATE"])
          temp.append(dictionary["EXPENSENAME"])
          temp.append(dictionary["AMOUNT"])
```

```
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0

    for x in expense:
        total += x[4]
        if x[6] == "food":
            t_food += x[4]

        elif x[6] == "entertainment":
            t_entertainment  += x[4]

        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]

        elif x[6] == "EMI":
            t_EMI  += x[4]

        elif x[6] == "other":
            t_other  += x[4]

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)
return render_template("today.html", texpense = texpense, expense = expense,  total = total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )

@app.route("/month")
def month():
    #   cursor = mysql.connection.cursor()
    #   cursor.execute('SELECT DATE(date), SUM(amount) FROM expenses WHERE userid=
%s AND MONTH(DATE(date))= MONTH(now()) GROUP BY DATE(date) ORDER BY
DATE(date) ',(str(session['id'])))
```

```
    #   texpense = cursor.fetchall()
    #   print(texpense)

      param1 = "SELECT DATE(date) as dt, SUM(amount) as tot FROM expenses WHERE
userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current timestamp) AND
YEAR(date) = YEAR(current timestamp) GROUP BY DATE(date) ORDER BY DATE(date)"
      res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
      dictionary1 = ibm_db.fetch_assoc(res1)
      texpense = []

      while dictionary1 != False:
         temp = []
         temp.append(dictionary1["DT"])
         temp.append(dictionary1["TOT"])
         texpense.append(temp)
         print(temp)
         dictionary1 = ibm_db.fetch_assoc(res1)
  #   cursor = mysql.connection.cursor()
    #   cursor.execute('SELECT * FROM expenses WHERE userid = % s AND
MONTH(DATE(date))= MONTH(now()) AND date ORDER BY `expenses`.`date`
DESC',(str(session['id'])))
    #   expense = cursor.fetchall()
param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND
MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp)
ORDER BY date DESC"
      res = ibm_db.exec_immediate(ibm_db_conn, param)
      dictionary = ibm_db.fetch_assoc(res)
      expense = []
      while dictionary != False:
         temp = []
         temp.append(dictionary["ID"])
         temp.append(dictionary["USERID"])
         temp.append(dictionary["DATE"])
         temp.append(dictionary["EXPENSENAME"])
         temp.append(dictionary["AMOUNT"])
         temp.append(dictionary["PAYMODE"])
         temp.append(dictionary["CATEGORY"])
         expense.append(temp)
         print(temp)
         dictionary = ibm_db.fetch_assoc(res)


      total=0
      t_food=0
      t_entertainment=0
      t_business=0
      t_rent=0
      t_EMI=0
      t_other=0


      for x in expense:
```

```
        total += x[4]
        if x[6] == "food":
            t_food += x[4]

        elif x[6] == "entertainment":
            t_entertainment  += x[4]

        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]

        elif x[6] == "EMI":
            t_EMI  += x[4]

        elif x[6] == "other":
            t_other  += x[4]

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)
return render_template("today.html", texpense = texpense, expense = expense,  total = total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )

@app.route("/year")
def year():
   #   cursor = mysql.connection.cursor()
   #   cursor.execute('SELECT MONTH(date), SUM(amount) FROM expenses WHERE userid=
%s AND YEAR(DATE(date))= YEAR(now()) GROUP BY MONTH(date) ORDER BY
MONTH(date) ',(str(session['id'])))
   #   texpense = cursor.fetchall()
   #   print(texpense)

    param1 = "SELECT MONTH(date) as mn, SUM(amount) as tot FROM expenses WHERE
userid = " + str(session['id']) + " AND YEAR(date) = YEAR(current timestamp) GROUP BY
MONTH(date) ORDER BY MONTH(date)"
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpense = []

    while dictionary1 != False:
        temp = []
        temp.append(dictionary1["MN"])
        temp.append(dictionary1["TOT"])
        texpense.append(temp)
```

```
        print(temp)
        dictionary1 = ibm_db.fetch_assoc(res1)


    #   cursor = mysql.connection.cursor()
    #   cursor.execute('SELECT * FROM expenses WHERE userid = % s AND
YEAR(DATE(date))= YEAR(now()) AND date ORDER BY `expenses`.`date`
DESC',(str(session['id'])))
    #   expense = cursor.fetchall()

    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND
YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)


    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0


    for x in expense:
        total += x[4]
        if x[6] == "food":
            t_food += x[4]

        elif x[6] == "entertainment":
            t_entertainment  += x[4]

        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]

        elif x[6] == "EMI":
```

```
            t_EMI  += x[4]

        elif x[6] == "other":
            t_other  += x[4]

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)



    return render_template("today.html", texpense = texpense, expense = expense,  total =
total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )

#log-out

@app.route('/logout')

def logout():
  session.pop('loggedin', None)
  session.pop('id', None)
  session.pop('username', None)
  session.pop('email', None)
  return render_template('home.html')


port = os.getenv('VCAP_APP_PORT', '8080')
if __name__ == "__main__":
   app.secret_key = os.urandom(12)
   app.run(debug=True, host='0.0.0.0', port=port)
```

TESTING

8.1 TEST CASES

1	Test Cases	Result
2	Verify the user is able to see the Sign up page when the user clicks the signup button in navigation bar	Positive
3	Verify the UI elements in the Sign up page	Positive
4	Verify the user is able to register into the application by providing valid details	Positive
5	Verify the user is able to see the sign in page when the user clicks the signin button in navigation bar	Positive
6	Verify the UI elements in the Sign in page	Positive
7	Verify the user is able to login into the application by providing valid details	Positive
8	Verify the user is able to see the Donor registration page when the user clicks the donate link in navigation bar	Positive
9	Verify the UI elements in the Donor Registration page	Positive
10	Verify the user is able toregister as a donor by providing valid details	Positive
11	Verify the user is able to see the request page when the user clicks the request link in navigation bar	Positive
12	Verify the UI elements in the request page	Positive
13	Verify the user is able to make a request by providing valid details	Positive
14	Verify the user gets a email notification when they sign up	Positive
15	Verify the donor gets a email notification when they make a request	Positive
16	Verify the donor and recipient gets a email notification when the donor accepts the request	Positive
17	Verify the user is able to see the stats page when the user clicks the stage page link in navigation bar	Positive
18	Verify the user is able to interact with the chatbot	Positive

8.2 USER ACCEPTANCE TESTING

1	Test case ID	Feature Type	Component	Test Scenario	Steps to Execute
2	SignUpPage_TC_001	Functional	Sign Up page	Verify the user is able to see the Sign up page when the user clicks the signup button in navigation bar	1. Enter the url and go 2.Click the sign up link in the navigation bar. 3.Verify the sign up page is visible or not.
3	SignUpPage_TC_002	UI	Sign Up page	Verify the UI elements in the Sign up page	1. Enter the url and go 2.Click the sign up link in the navigation bar. 3.Verify the below mentioned ui elements: a.name text box b. email text box. c. password text box. d. repeat password text box. e. sign up button
4	SignUpPage_TC_003	Functional	Sign Up page	Verify the user is able to register into the application by providing valid details	1. Enter the url and go 2.Click the sign up link in the navigation bar. 3.Enter valid details in the text boxes. 4. Verify the confirmation message.
5	SignInPage_TC_001	Functional	Sign In page	Verify the user is able to see the sign in page when the user clicks the signin button in navigation bar	1. Enter the url and go 2.Click the sign in link in the navigation bar. 3.Verify the sign in page is visible or not.
6	SignInPage_TC_002	UI	Sign In page	Verify the UI elements in the Sign in page	1. Enter the url and go 2.Click the sign in link in the navigation bar. 3.Verify the below mentioned ui elements: a. email text box. b. password text box. c. sign in button

6	SigninPage_TC_002	UI	Sign in page	Verify the UI elements in the Sign in page	a. email text box. b. password text box. c. sign in button
7	SigninPage_TC_003	Functional	Sign in page	Verify the user is able to login into the application by providing valid details	1. Enter the url and go 2.Click the sign in link in the navigation bar. 3.Enter valid details in the text boxes. 4. Verify the user is able to login.
8	DonorRegistrationPage_TC_001	Functional	Donor Registration Page	Verify the user is able to see the Donor registration page when the user clicks the donate link in navigation bar	1. Enter the url and go 2.Click the donate link in the navigation bar. 3.Verify the donor registration page is visible or not.
9	DonorRegistrationPage_TC_002	UI	Donor Registration Page	Verify the UI elements in the Donor Registration page	. Enter the url and go 2.Click the donate link in the navigation bar. 3.Verify the below mentioned ui elements: a. name text box b. email text box. c. blood group text box. d. contact number text box. e. city text box f. register as donor button
10	DonorRegistrationPage_TC_003	Functional	Donor Registration Page	Verify the user is able toregister as a donor by providing valid details	1. Enter the url and go 2.Click the donate link in the navigation bar. 3.Enter valid details in the text boxes. 4. Click the donate button. 4. Verify the user is able to register as a donor sucessfully

11	RequestPage_TC_001	Functional	Request Page	Verify the user is able to see the request page when the user clicks the request link in navigation bar	1. Enter the url and go 2.Click the request link in the navigation bar. 3.Verify the request page is visible or not.
12	RequestPage_TC_002	UI	Request Page	Verify the UI elements in the request page	. Enter the url and go 2.Click the request link in the navigation bar. 3.Verify the below mentioned ui elements: a. name text box b. email text box. c. blood group text box. d. contact number text box. e. city text box f. make a request button
13	RequestPage_TC_003	Functional	Request Page	Verify the user is able to make a request by providing valid details	1. Enter the url and go 2.Click the request link in the navigation bar. 3.Enter valid details in the text boxes. 4. Click the request button. 4. Verify the user is able to make a request sucessfully.
14	Notication_TC_001	Functional	Sign up page	Verify the user gets a email notification when they sign up	1. Enter the url and go 2. Go to the sign up page. 3. Enter the details and click sign up button 4. Verify if they get the email on successfull sign up
15	Notication_TC_002	Functional	Request Page	Verify the donor gets a email notification when they make a request	1. Enter the url and go 2. Go to the request page. 3. Enter the details and click make a request button 4. Verify if the donor gets the email on successfully maki request.

16	Notication_TC_003	Functional	Profile Page	Verify the donor and recipient gets a email notification when the donor accepts the request	1. Enter the url and go 2. Go to the profile page 3.Accept the pending request. 4. Verify if they get the email containing contact details
17	StatsPage_TC_001	Functional	Stats Page	Verify the user is able to see the stats page when the user clicks the stage page link in navigation bar	1. Enter the url and go 2.Click the stats page link in the navigation bar. 3.Verify the stats page is visible or not
18	Chatbot_TC_001	Functional	Home Page	Verify the user is able to interact with the chatbot	1. Enter the url and go 2.Click the chatbot icon in the home page 3.Verify the chatbot is working or not

RESULTS

9.1 PERFORMANCE METRICS

Web application performance metrics help determine certain aspects that impact the performance of an application. There are eight key metrics, including: User Satisfaction—also known as Apdex Scores, uses a mathematical formula in order to determine user satisfaction.

Fig 4.1 Performance Metrics

ADVANTAGES & DISADVANTAGES

ADVANTAGES

- It is a user-friendly application.
- It will help people to find plasmaeasily.
- Simple User Interface
- It alleviates the burden of coordinator to manage Users and resources easily.
- Compared to all other mobileapplications, it incorporates provision for daily expense and
  total income Requesting.

- Attracts more, number of users as it is available in the form of Mobile applicationinsteadof What‘‘s app group.
- Usage of this application will greatly reducetime in selecting the right decision in making purchasing.

DISADVANTAGES

- It requires an active internetconnection.
- It relays on the detailsprovided by the user.

CONCLUSION

The main purpose of this rpoduct is to track the daily expense of the people of gen z .The aim to spend the money effficient and keep track of the expenditure.The analysis of daily expenses may help in the improving the way of saving and this product can also come in handy when paying monthly emi and others loans. The planning of the future expenses can be poosible and easy to process with the help of this platform.The analysis of expenditure is made easy with visualization tools to show the graph of the expense.The user can see the expense where they spend their money mostly.

FUTURE SCOPE

- The coupon holder to save the digital coupon and offers for the future shopping.

- To analysis and advertise the product market for the user from the analysis of the data

Appendix

Source code:

request.html :
<!doctype html>

```
<html lang="en">
<head>
<!—Required meta tags ◊
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<link href="https://fonts.googleapis.com/css?family=Roboto:400,700,900&display=swap"
rel="stylesheet">
<!—Vendor CSS Files ◊
<link href="../static/vendor/aos/aos.css" rel="stylesheet">
<link href="../static/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
<link href="../static/vendor/glightbox/css/glightbox.min.css" rel="stylesheet">
<link href="../static/vendor/swiper/swiper-bundle.min.css" rel="stylesheet">
<link href="../static/css/style.css" rel="stylesheet">
<!—Bootstrap CSS ◊
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>
<!—Style ◊
<link rel="stylesheet" href="../static/css/request.css">
<script>
function validateForm() {

let name = document.forms["form"]["name"].value; let email =
document.forms["form"]["email"].value;
let blood_group = document.forms["form"]["blood_group"].value; let contact_no =
document.forms["form"]["contact_no"].value;
let city = document.forms["form"]["city"].value;
if(validname(name) && validemail(email) && validblood_group(blood_group) &&
validcontact_no(contact_no) &&  validcity(city)){
return true;
}
else{
return false;
}
}
function validname(name){ document.getElementById("name_err").innerHTML="" if (name
!= "") {
var letters = /^[a-zA-Z ]*$/; if(name.match(letters))
{
return true;
}
else
{
document.getElementById("name_err").innerHTML="Name should contain only Alphabets"
return false;
}
}
```

```
else{
document.getElementById("name_err").innerHTML="Name should not be empty" return
false;
}
}
function validemail(email){

console.log(email) document.getElementById("email_err").innerHTML=""; if (email != "") {
var emailfor = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/; if(email.match(emailfor))
{
return true;
}
else
{
document.getElementById("email_err").innerHTML="Invalid Email" return false;
}
}
else{
document.getElementById("email_err").innerHTML="Email Should not be empty" return
false;
}
}

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">


  <link rel="stylesheet" href="..\static\css\home.css">
  <title>My Website</title>
</head>

<body>
  <!-- Header -->
  <section id="header">
    <div class="header container">
      <div class="nav-bar">
        <div class="brand">
          <a href="#hero">
            <h1><span>M</span>y <span>B</span>udget</h1>
          </a>
        </div>
        <div class="nav-list">
          <div class="hamburger">
            <div class="bar"></div>
          </div>
          <ul>
            <li><a href="#hero" data-after="Home">Home</a></li>
            <li><a href="#services" data-after="Service">Services</a></li>
```

```
          <li><a href="#about" data-after="About">About</a></li>
          <li><a href="#contact" data-after="Contact">Contact</a></li>
          <LI><a href="/signin" data-after="Login">-Login-</a></LI>
        </ul>
      </div>
    </div>
  </div>
</section>
<!-- End Header -->


<!-- Hero Section  -->
<section id="hero">
  <div class="hero container">
    <div>
      <h1>Hello, <span></span></h1>
      <h1>Welcome To <span></span></h1>
      <h1>Expense Tracker Web application <span></span></h1>
      <a href="/signup" type="button" class="cta">Sign-up</a>
    </div>
  </div>
</section>
<!-- End Hero Section  -->

<!-- Service Section -->
<section id="services">
  <div class="services container">
    <div class="service-top">
      <h1 class="section-title">Serv<span>i</span>ces</h1>
      <p>MyBudget provides a many services  to the customer and industries. Financial
solutions to meet your needs whatever your money goals,there is a MyBudget solution to
help you reach them </p>
    </div>
    <div class="service-bottom">
      <div class="service-item">
        <div class="icon"><img
src="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
        <h2>Personal Expenses</h2>
        <p>Budgeting is more than paying bills and setting aside savings.it's about creating a
money plan for the life you want</p>
      </div>
      <div class="service-item">
        <div class="icon"><img
src="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
        <h2>Investments</h2>
        <p>Follow your investments and bring your portfolio into focus with support for
stocks,bonds,CDs,mutual funds and more</p>
      </div>
      <div class="service-item">
        <div class="icon"><img
src="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
```

```
        <h2>Online Banking</h2>
        <p>MyBudget application can automatically download transactions and send
payments online from many financial institutions</p>
      </div>
      <div class="service-item">
        <div class="icon"><img
src="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
        <h2>Financial Life</h2>
        <p>Get your Complete financial picture at a glance. With MyBudget application you can
view your all the financial activities
        </p>
      </div>
    </div>
  </div>
 </section>
 <!-- End Service Section -->

<!-- About Section -->
 <section id="about">
  <div class="about container">
    <div class="col-left">
      <div class="about-img">
        <img src="..\static\images\img-2.png" alt="img">
        <div><h2>Founders, CSE-C Last Benchers </h2></div>
      </div>
    </div>

    <div class="col-right">
      <h1 class="section-title">About <span>Us</span></h1>
      <h2>Financial Solution</h2>
      <p>MyBudget financial solution  is one among Leading financial company from many
years.MyBudget provides a many services  to the customer and industries. Financial
solutions to meet your needs whatever your money goals,there is a MyBudget solution to
help you reach them.u can Contact our service center for further information and also follow
our social media for update on new services </p>
      <a href="#footer" class="cta">Follow Us</a>

    </div>
  </div>
 </section>
 <!-- End About Section -->

 <!-- Contact Section -->
 <section id="contact">
  <div class="contact container">
    <div>
      <h1 class="section-title">Contact <span>info</span></h1>
    </div>
    <div class="contact-items">
      <div class="contact-item">
        <div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/phone.png"
/></div>
```

```
        <div class="contact-info">
          <h1>Phone</h1>
          <h2>+1 234 123 1234</h2>
          <h2>+1 234 123 1234</h2>
        </div>
      </div>
      <div class="contact-item">
        <div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/new-
post.png" /></div>
        <div class="contact-info">
          <h1>Email</h1>
          <h2>info@gmail.com</h2>
          <h2>abcd@gmail.com</h2>
        </div>
      </div>
      <div class="contact-item">
        <div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/map-
marker.png" /></div>
        <div class="contact-info">
          <h1>Address</h1>
          <h2>4th main-road,Bengaluru,Karnataka,India</h2>
        </div>
      </div>
    </div>
  </div>
 </section>
 <!-- End Contact Section -->

 <!-- Footer -->
 <section id="footer">
  <div class="footer container">
    <div class="brand">
      <h1><span>M</span>y <span>B</span>udget</h1>
    </div>
    <h2>Your Complete Financial Solution</h2>
    <div class="social-icon">
      <div class="social-item">
        <a href="#"><img src="https://img.icons8.com/bubbles/100/000000/facebook-
new.png" /></a>
      </div>
      <div class="social-item">
        <a href="#"><img src="https://img.icons8.com/bubbles/100/000000/instagram-
new.png" /></a>
      </div>
      <div class="social-item">
        <a href="#"><img src="https://img.icons8.com/bubbles/100/000000/behance.png"
/></a>
      </div>
    </div>
    <p>Copyright © 2022 SECE IV-CSE-'C' Last Benchers . All rights reserved</p>
  </div>
 </section>
```

```
  <!-- End Footer -->
  <script src="..\static\js\home.js"></script>
</body>

</html>
//python//
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""

from flask import Flask, render_template, request, redirect, session
# from flask_mysqldb import MySQL
# import MySQLdb.cursors
import re

from flask_db2 import DB2
import ibm_db
import ibm_db_dbi
from sendemail import sendgridmail,sendmail

# from gevent.pywsgi import WSGIServer
import os


app = Flask(__name__)

app.secret_key = 'a'

# app.config['MYSQL_HOST'] = 'remotemysql.com'
# app.config['MYSQL_USER'] = 'D2DxDUPBii'
# app.config['MYSQL_PASSWORD'] = 'r8XBO4GsMz'
# app.config['MYSQL_DB'] = 'D2DxDUPBii'
"""
dsn_hostname = "ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
dsn_uid = "vmk08423"
dsn_pwd = "3KfJl6HGDtPdbIWy"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "31321"
dsn_protocol = "tcpip"

dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
    "PORT={3};"
    "PROTOCOL={4};"
    "UID={5};"
```

```
    "PWD={6};"
).format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol, dsn_uid,
dsn_pwd)
"""
# app.config['DB2_DRIVER'] = '{IBM DB2 ODBC DRIVER}'
app.config['database'] = 'bludb'
app.config['hostname'] = 'ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud'
app.config['port'] = '31321'
app.config['protocol'] = 'tcpip'
app.config['uid'] = 'vmk084230'
app.config['pwd'] = '3KfJl6HGDtPdbIWy'
app.config['security'] = 'SSL'
try:
    mysql = DB2(app)

    conn_str='database=bludb;hostname=ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;port=31321;protocol=tcpi
p;\
            uid=vmk08423;pwd=3KfJl6HGDtPdbIWy;security=SSL'
    ibm_db_conn = ibm_db.connect(conn_str,'','')

    print("Database connected without any error !!")
except:
    print("IBM DB Connection error   :     " + DB2.conn_errormsg())

# app.config['']

# mysql = MySQL(app)


#HOME--PAGE
@app.route("/home")
def home():
    return render_template("homepage.html")

@app.route("/")
def add():
    return render_template("home.html")



#SIGN--UP--OR--REGISTER


@app.route("/signup")
def signup():
    return render_template("signup.html")



@app.route('/register', methods =['GET', 'POST'])
```

```
def register():
    msg = ''
    print("Break point1")
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

        print("Break point2" + "name: " + username + "------" + email + "------" + password)

        try:
            print("Break point3")
            connectionID = ibm_db_dbi.connect(conn_str, '', '')
            cursor = connectionID.cursor()
            print("Break point4")
        except:
            print("No connection Established")

        # cursor = mysql.connection.cursor()
        # with app.app_context():
        #     print("Break point3")
        #     cursor = ibm_db_conn.cursor()
        #     print("Break point4")

        print("Break point5")
        sql = "SELECT * FROM register WHERE username = ?"
        stmt = ibm_db.prepare(ibm_db_conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        result = ibm_db.execute(stmt)
        print(result)
        account = ibm_db.fetch_row(stmt)
        print(account)

        param = "SELECT * FROM register WHERE username = '" + "\'" + username + "\'"
        res = ibm_db.exec_immediate(ibm_db_conn, param)
        print("---- ")
        dictionary = ibm_db.fetch_assoc(res)
        while dictionary != False:
            print("The ID is : ", dictionary["USERNAME"])
            dictionary = ibm_db.fetch_assoc(res)

        # dictionary = ibm_db.fetch_assoc(result)
        # cursor.execute(stmt)

        # account = cursor.fetchone()
        # print(account)

        # while ibm_db.fetch_row(result) != False:
        #     # account = ibm_db.result(stmt)
        #     print(ibm_db.result(result, "username"))
```

```
        # print(dictionary["username"])
        print("break point 6")
        if account:
            msg = 'Username already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            sql2 = "INSERT INTO register (username, email,password) VALUES (?, ?, ?)"
            stmt2 = ibm_db.prepare(ibm_db_conn, sql2)
            ibm_db.bind_param(stmt2, 1, username)
            ibm_db.bind_param(stmt2, 2, email)
            ibm_db.bind_param(stmt2, 3, password)
            ibm_db.execute(stmt2)
            # cursor.execute('INSERT INTO register VALUES (NULL, % s, % s, % s)', (username,
email,password))
            # mysql.connection.commit()
            msg = 'You have successfully registered !'
        return render_template('signup.html', msg = msg)




#LOGIN--PAGE

@app.route("/signin")
def signin():
    return render_template("login.html")

@app.route('/login',methods =['GET', 'POST'])
def login():
    global userid
    msg = ''


    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        # cursor = mysql.connection.cursor()
        # cursor.execute('SELECT * FROM register WHERE username = % s AND password = %
s', (username, password ),)
        # account = cursor.fetchone()
        # print (account)

        sql = "SELECT * FROM register WHERE username = ? and password = ?"
        stmt = ibm_db.prepare(ibm_db_conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        result = ibm_db.execute(stmt)
        print(result)
        account = ibm_db.fetch_row(stmt)
```

```
        print(account)

        param = "SELECT * FROM register WHERE username = " + "\'" + username + "\'" + " and
password = " + "\'" + password + "\'"
        res = ibm_db.exec_immediate(ibm_db_conn, param)
        dictionary = ibm_db.fetch_assoc(res)

        # sendmail("hello sakthi","sivasakthisairam@gmail.com")

        if account:
            session['loggedin'] = True
            session['id'] = dictionary["ID"]
            userid = dictionary["ID"]
            session['username'] = dictionary["USERNAME"]
            session['email'] = dictionary["EMAIL"]

            return redirect('/home')
        else:
            msg = 'Incorrect username / password !'

    return render_template('login.html', msg = msg)









#ADDING----DATA


@app.route("/add")
def adding():
    return render_template('add.html')


@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():

    date = request.form['date']
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category = request.form['category']

    print(date)
    p1 = date[0:10]
    p2 = date[11:13]
    p3 = date[14:]
```

```
    p4 = p1 + "-" + p2 + "." + p3 + ".00"
    print(p4)
    # cursor = mysql.connection.cursor()
    # cursor.execute('INSERT INTO expenses VALUES (NULL,  % s, % s, % s, % s, % s, % s)',
(session['id'] ,date, expensename, amount, paymode, category))
    # mysql.connection.commit()
    # print(date + " " + expensename + " " + amount + " " + paymode + " " + category)

    sql = "INSERT INTO expenses (userid, date, expensename, amount, paymode, category)
VALUES (?, ?, ?, ?, ?, ?)"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, session['id'])
    ibm_db.bind_param(stmt, 2, p4)
    ibm_db.bind_param(stmt, 3, expensename)
    ibm_db.bind_param(stmt, 4, amount)
    ibm_db.bind_param(stmt, 5, paymode)
    ibm_db.bind_param(stmt, 6, category)
    ibm_db.execute(stmt)

    print("Expenses added")

    # email part

    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND
MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp)
ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    total=0
    for x in expense:
         total += x[4]

    param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER
BY id DESC LIMIT 1"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = 0
```

```
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMITSS"])
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[0]

    if total > int(s):
        msg = "Hello " + session['username'] + " , " + "you have crossed the monthly limit of Rs. "
+ s + "/- !!!" + "\n" + "Thank you, " + "\n" + "Team Personal Expense Tracker."
        sendmail(msg,session['email'])

    return redirect("/display")



#DISPLAY---graph

@app.route("/display")
def display():
    print(session["username"],session['id'])

    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT * FROM expenses WHERE userid = % s AND date ORDER BY
`expenses`.`date` DESC',(str(session['id'])))
    # expense = cursor.fetchall()

    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " ORDER BY
date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    return render_template('display.html' ,expense = expense)



#delete---the--data
```

```
@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
def delete(id):
    #  cursor = mysql.connection.cursor()
    #  cursor.execute('DELETE FROM expenses WHERE  id = {0}'.format(id))
    #  mysql.connection.commit()

    param = "DELETE FROM expenses WHERE  id = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)

    print('deleted successfully')
    return redirect("/display")


#UPDATE---DATA

@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT * FROM expenses WHERE  id = %s', (id,))
    # row = cursor.fetchall()

    param = "SELECT * FROM expenses WHERE  id = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        row.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    print(row[0])
    return render_template('edit.html', expenses = row[0])




@app.route('/update/<id>', methods = ['POST'])
def update(id):
  if request.method == 'POST' :

      date = request.form['date']
      expensename = request.form['expensename']
      amount = request.form['amount']
      paymode = request.form['paymode']
```

```
    category = request.form['category']

   #   cursor = mysql.connection.cursor()
   #   cursor.execute("UPDATE `expenses` SET `date` = % s , `expensename` = % s , `amount`
= % s, `paymode` = % s, `category` = % s WHERE `expenses`.`id` = % s ",(date, expensename,
amount, str(paymode), str(category),id))
   #   mysql.connection.commit()

    p1 = date[0:10]
    p2 = date[11:13]
    p3 = date[14:]
    p4 = p1 + "-" + p2 + "." + p3 + ".00"

    sql = "UPDATE expenses SET date = ? , expensename = ? , amount = ?, paymode = ?,
category = ? WHERE id = ?"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, p4)
    ibm_db.bind_param(stmt, 2, expensename)
    ibm_db.bind_param(stmt, 3, amount)
    ibm_db.bind_param(stmt, 4, paymode)
    ibm_db.bind_param(stmt, 5, category)
    ibm_db.bind_param(stmt, 6, id)
    ibm_db.execute(stmt)

    print('successfully updated')
    return redirect("/display")


#limit
@app.route("/limit" )
def limit():
     return redirect('/limitn')

@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
    if request.method == "POST":
       number= request.form['number']
       #  cursor = mysql.connection.cursor()
       #  cursor.execute('INSERT INTO limits VALUES (NULL, % s, % s) ',(session['id'], number))
       #  mysql.connection.commit()

       sql = "INSERT INTO limits (userid, limitss) VALUES (?, ?)"
       stmt = ibm_db.prepare(ibm_db_conn, sql)
       ibm_db.bind_param(stmt, 1, session['id'])
       ibm_db.bind_param(stmt, 2, number)
       ibm_db.execute(stmt)
```

```
        return redirect('/limitn')


@app.route("/limitn")
def limitn():
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT limitss FROM `limits` ORDER BY `limits`.`id` DESC LIMIT 1')
    # x= cursor.fetchone()
    # s = x[0]

    param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER
BY id DESC LIMIT 1"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = " /-"
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMITSS"])
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[0]

    return render_template("limit.html" , y= s)

#REPORT

@app.route("/today")
def today():
    #   cursor = mysql.connection.cursor()
    #   cursor.execute('SELECT TIME(date)   , amount FROM expenses  WHERE userid = %s
AND DATE(date) = DATE(NOW()) ',(str(session['id'])))
    #   texpense = cursor.fetchall()
    #   print(texpense)

      param1 = "SELECT TIME(date) as tn, amount FROM expenses WHERE userid = " +
str(session['id']) + " AND DATE(date) = DATE(current timestamp) ORDER BY date DESC"
      res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
      dictionary1 = ibm_db.fetch_assoc(res1)
      texpense = []

      while dictionary1 != False:
          temp = []
          temp.append(dictionary1["TN"])
          temp.append(dictionary1["AMOUNT"])
          texpense.append(temp)
          print(temp)
          dictionary1 = ibm_db.fetch_assoc(res1)

    #   cursor = mysql.connection.cursor()
```

```
    #   cursor.execute('SELECT * FROM expenses WHERE userid = % s AND DATE(date) =
DATE(NOW()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
    #   expense = cursor.fetchall()

    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND
DATE(date) = DATE(current timestamp) ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)


    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0


    for x in expense:
        total += x[4]
        if x[6] == "food":
            t_food += x[4]

        elif x[6] == "entertainment":
            t_entertainment  += x[4]

        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]

        elif x[6] == "EMI":
            t_EMI  += x[4]

        elif x[6] == "other":
            t_other  += x[4]

    print(total)
```

```
    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)



    return render_template("today.html", texpense = texpense, expense = expense,  total =
total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )


@app.route("/month")
def month():
   #   cursor = mysql.connection.cursor()
   #   cursor.execute('SELECT DATE(date), SUM(amount) FROM expenses WHERE userid=
%s AND MONTH(DATE(date))= MONTH(now()) GROUP BY DATE(date) ORDER BY
DATE(date) ',(str(session['id'])))
   #   texpense = cursor.fetchall()
   #   print(texpense)

    param1 = "SELECT DATE(date) as dt, SUM(amount) as tot FROM expenses WHERE
userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current timestamp) AND
YEAR(date) = YEAR(current timestamp) GROUP BY DATE(date) ORDER BY DATE(date)"
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpense = []

    while dictionary1 != False:
       temp = []
       temp.append(dictionary1["DT"])
       temp.append(dictionary1["TOT"])
       texpense.append(temp)
       print(temp)
       dictionary1 = ibm_db.fetch_assoc(res1)


   #   cursor = mysql.connection.cursor()
   #   cursor.execute('SELECT * FROM expenses WHERE userid = % s AND
MONTH(DATE(date))= MONTH(now()) AND date ORDER BY `expenses`.`date`
DESC',(str(session['id'])))
   #   expense = cursor.fetchall()

    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND
MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp)
ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
```

```
dictionary = ibm_db.fetch_assoc(res)
expense = []
while dictionary != False:
    temp = []
    temp.append(dictionary["ID"])
    temp.append(dictionary["USERID"])
    temp.append(dictionary["DATE"])
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    expense.append(temp)
    print(temp)
    dictionary = ibm_db.fetch_assoc(res)


total=0
t_food=0
t_entertainment=0
t_business=0
t_rent=0
t_EMI=0
t_other=0


for x in expense:
    total += x[4]
    if x[6] == "food":
        t_food += x[4]

    elif x[6] == "entertainment":
        t_entertainment  += x[4]

    elif x[6] == "business":
        t_business  += x[4]
    elif x[6] == "rent":
        t_rent  += x[4]

    elif x[6] == "EMI":
        t_EMI  += x[4]

    elif x[6] == "other":
        t_other  += x[4]

print(total)

print(t_food)
print(t_entertainment)
print(t_business)
print(t_rent)
print(t_EMI)
print(t_other)
```

```
    return render_template("today.html", texpense = texpense, expense = expense,  total =
total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )

@app.route("/year")
def year():
   #   cursor = mysql.connection.cursor()
   #   cursor.execute('SELECT MONTH(date), SUM(amount) FROM expenses WHERE userid=
%s AND YEAR(DATE(date))= YEAR(now()) GROUP BY MONTH(date) ORDER BY
MONTH(date) ',(str(session['id'])))
   #   texpense = cursor.fetchall()
   #   print(texpense)

    param1 = "SELECT MONTH(date) as mn, SUM(amount) as tot FROM expenses WHERE
userid = " + str(session['id']) + " AND YEAR(date) = YEAR(current timestamp) GROUP BY
MONTH(date) ORDER BY MONTH(date)"
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpense = []

    while dictionary1 != False:
       temp = []
       temp.append(dictionary1["MN"])
       temp.append(dictionary1["TOT"])
       texpense.append(temp)
       print(temp)
       dictionary1 = ibm_db.fetch_assoc(res1)


   #   cursor = mysql.connection.cursor()
   #   cursor.execute('SELECT * FROM expenses WHERE userid = % s AND
YEAR(DATE(date))= YEAR(now()) AND date ORDER BY `expenses`.`date`
DESC',(str(session['id'])))
   #   expense = cursor.fetchall()

    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND
YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
       temp = []
       temp.append(dictionary["ID"])
       temp.append(dictionary["USERID"])
       temp.append(dictionary["DATE"])
       temp.append(dictionary["EXPENSENAME"])
       temp.append(dictionary["AMOUNT"])
```

```
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)


    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0


    for x in expense:
        total += x[4]
        if x[6] == "food":
            t_food += x[4]

        elif x[6] == "entertainment":
            t_entertainment  += x[4]

        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]

        elif x[6] == "EMI":
            t_EMI  += x[4]

        elif x[6] == "other":
            t_other  += x[4]

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)



    return render_template("today.html", texpense = texpense, expense = expense,  total =
total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )
```

```
#log-out

@app.route('/logout')

def logout():
   session.pop('loggedin', None)
   session.pop('id', None)
   session.pop('username', None)
   session.pop('email', None)
   return render_template('home.html')


port = os.getenv('VCAP_APP_PORT', '8080')
if __name__ == "__main__":
    app.secret_key = os.urandom(12)
    app.run(debug=True, host='0.0.0.0', port=port)
//flash//
apiVersion: v1
kind: Service
metadata:
  name: flask-app-service
spec:
  selector:
    app: flask-app
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 5000
  type: LoadBalancer
//yaml//
apiVersion: apps/v1
kind: Deployment
metadata:
   name: sakthi-flask-node-deployment
spec:
   replicas: 1
   selector:
     matchLabels:
       app: flasknode
   template:
     metadata:
       labels:
         app: flasknode
     spec:
       containers:
       - name: flasknode
         image: icr.io/sakthi_expense_tracker2/flask-template2
         imagePullPolicy: Always
         ports:
         - containerPort: 5000
//app python//
```

```
import smtplib
import sendgrid as sg
import os
from sendgrid.helpers.mail import Mail, Email, To, Content
SUBJECT = "expense tracker"
s = smtplib.SMTP('smtp.gmail.com', 587)

def sendmail(TEXT,email):
    print("sorry we cant process your candidature")
    s = smtplib.SMTP('smtp.gmail.com', 587)
    s.starttls()
    # s.login("il.tproduct8080@gmail.com", "oms@1Ram")
    s.login("tproduct8080@gmail.com", "lxixbmpnexbkiemh")
    message  = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)
    # s.sendmail("il.tproduct8080@gmail.com", email, message)
    s.sendmail("il.tproduct8080@gmail.com", email, message)
    s.quit()
def sendgridmail(user,TEXT):

    # from_email = Email("mohamedriyazafzal@gmail.com")
    from_email = Email("tproduct8080@gmail.com")
    to_email = To(user)
    subject = "Sending with SendGrid is Fun"
    content = Content("text/plain",TEXT)
    mail = Mail(from_email, to_email, subject, content)

    # Get a JSON-ready representation of the Mail object
    mail_json = mail.get()
    # Send an HTTP POST request to /mail/send
    response = sg.client.mail.send.post(request_body=mail_json)
    print(response.status_code)
    print(response.headers)
//login html//
<!DOCTYPE html>
<html>
<head>
        <title>Animated Login Form</title>
        <link rel="stylesheet" type="text/css" href="..\static\css\login.css">
        <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">
        <script src="https://kit.fontawesome.com/a81368914c.js"></script>
        <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body >
        <img class="wave" src="..\static\images\wave.png">
        <div class="container">

                <div class="img">
                        <div id="png"><a href="/" title="HOME"><img style="width:75px;
height:75px ; " src="..\static\images\home-page.png"></a></div>
                        <img src="..\static\images\bg.svg">
                </div>
```

```
                <div class="login-content">

                        <form action='/login' method="POST">
                                <div class="msg">{{ msg }}</div>
                                <img src="..\static\images\avatar.svg">
                                <h2 class="title">Welcome</h2>
                <div class="input-div one">
                  <div class="i">
                                <i class="fas fa-user"></i>
                  </div>
                  <div class="div">
                                <h5>Username</h5>
                                <input type="text" name="username" class="input" required>
                  </div>
                </div>
                <div class="input-div pass">
                  <div class="i">
                        <i class="fas fa-lock"></i>
                  </div>
                  <div class="div">
                        <h5>Password</h5>
                        <input type="password"  name="password" class="input" required>
            </div>
          </div>
          <a href="#">Forgot Password?</a>
          <input type="submit" class="btn" value="Login">
                                <span>OR</span>

                                <div><b>Login with</b></div>
                                <div>
                                        <ul>
                                                <li><a href="#"><i class="fab fa-facebook" aria-
hidden="true"></i></a></li>
                                                <li><a href="#"><i class="fab fa-twitter" aria-
hidden="true"></i></a></li>
                                                <li><a href="#"><i <i class="fab fa-google"  aria-
hidden="true"></i></a></li>
                                                <li><a href="#"><i class="fab fa-linkedin" aria-
hidden="true"></i></a></li>
                                                <li><a href="#"><i class="fab fa-instagram" aria-
hidden="true"></i></a></li>
                                        </ul>

                                </div>
                                <div class="app" ><b>Don't have an account?</b><a id="app1"
href="\signup">REGISTER.here</a></div>
          </form>

        </div>
```

```
    </div>

    <script type="text/javascript" src="..\static\js\login.js"></script>
</body>
</html>
//sign up html//
<html>
<head>
<meta charset="utf-8">
<title>Sign-up</title>
<link href="..\static\css\signup.css" rel="stylesheet">
<script src="https://kit.fontawesome.com/a81368914c.js"></script>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
<!--container---------------------->
<div class="container" >
<!--sign-up-box-container--->
<div class="sign-up">

    <div id="png"><a href="/" title="HOME"><img style="width:55px; height:55px ; "
src="..\static\images\home-page.png"></a></div>
<!--heading-->
<form action="/register" method="post">
    <div class="msg">{{ msg }}</div>
<h1 class="heading">Hello,Friend</h1>
<!--name-box-->
<div class="text">
<img height="20px" src="..\static\images\user.png" />
<input placeholder="Name" type="text" name="username"/>
</div>
<!--Email-box-->
<div class="text">
<img height="12px" src="..\static\images/email.png" />
<input placeholder=" Example@gmail.com" type="email" name="email"" />
</div>
<!--Password-box-->
<div class="text">
<img height="20px" src="..\static\images\password.png" />
<input placeholder=" Password" type="password" name="password"/>
</div>
<div class="or"><b>OR</b></div>
<div class="s1"><p><b>Sign-up with</b></p></div>

<div>
    <ul>
        <li><a href="#"><i class="fab fa-facebook" aria-hidden="true"></i></a></li>
        <li><a href="#"><i class="fab fa-twitter" aria-hidden="true"></i></a></li>
```

```
        <li><a href="#"><i class="fab fa-google"  aria-hidden="true"></i></a></li>
        <li><a href="#"><i class="fab fa-linkedin" aria-hidden="true"></i></a></li>
        <li><a href="#"><i class="fab fa-instagram" aria-hidden="true"></i></a></li>
    </ul>

</div>
<!--trems-->

<div class="trems">
    <input class="check" type="checkbox" required/>
<p class="conditions">I read and agree to <a href="#">Trems &amp; Conditions</a></p>
</div>
<!--button-->
<div class="toop">
<button type="submit" class="btn btn-primary" >CREATE ACCOUNT</button> </div>

</form>
<!--sign-in-->
<div class="t"><p class="conditions" id="p3">Already have an account <a href="/signin">Sign
in</a></p> </div></div>
</div>
<!--text-container-->
<div class="text-container">

<h1 style="color: #2d2c2c;font-family:cursive;">Glad to see you</h1>

<div class="diag"><img class="fig1" width="100%" height="105%"
src="..\static\images\Inkeddia_LI.jpg"</div>
<div class="para"> <b>Welcome</b>,Please Fill in the blanks for sign up</div>

</div>
</div>
</body>
</html>
//edit html//
{% extends 'base.html' %}

{% block body %}
<div class="container">
    <div class="row">
        <div class="col-md-6">
            <h3>Edit Expense</h3>
            <form action="/update/{{expenses[0]}}" method="POST">


                <input type="hidden" class="form-control"   name="" value = "{{expenses[0]}}"
id="">

            <div class="form-group">
                <label for="">Date</label>
                <input class="form-control"  type="datetime-local" name="date"
value="{{expenses[2]}}"  id="date"></div>
```

```
<script type="text/javascript">
        var d = new Date(value="{{expenses[2]}}" );
        var elem = document.getElementById("date");
        elem.value = d.toISOString().slice(0,16);
      </script>



      <div class="form-group"> <label for="">Expense name</label>
        <input class="form-control" type="text" name="expensename"
value="{{expenses[3]}}" id="expensename">
      </div>
      <div class="form-group">
         <label for="">Expense Amount</label>
          <input class="form-control"  type="number" min="0" name="amount"
value="{{expenses[4]}}" id="amount">
      </div>

      <div class="form-group">
        <label for=""></label>
        <select class="form-control" name="paymode"
value="{{expenses[5]}}"  id="paymode">
          <option selected hidden>{{expenses[5]}}</option>
          <option value="cash">cash</option>
          <option value="debitcard">debitcard</option>
          <option value="creditcard">creditcard</option>
          <option value="epayment">epayment</option>
          <option value="onlinebanking">onlinebanking</option>

        </select>

       <div class="form-group">
        <label for=""></label>
        <select class="form-control" name="category"
value="{{expenses[6]}}"  id="category">

           <option  selected hidden>{{expenses[6]}}</option>
          <option value="food">food</option>
          <option value="entertainment">Entertainment</option>
          <option value="business ">Business</option>
          <option value="rent">Rent</option>
          <option value="EMI">EMI</option>
          <option value="other">other</option>

        </select>
      </div>

      <input class="btn btn-danger" type="submit" value="Update" id="">
```

```
        </form>
      </div>
    </div>

</div>

    {% endblock %}
//manifest yml//
applications:
- name: Python Flask App IBCMR 2022-10-19
  random-route: true
  memory: 512M
  disk_quota: 1.5G
```

GITHUB LINK: https://github.com/IBM-EPBL/IBM-Project-18519-1659686342
PROJECT DEMO LINK: https://drive.google.com/file/d/1I2pnxPuC-Ho7ZkGEdRZK47OjNkCHs1zG/view?usp=drivesdk