PERSONAL EXPENSE TRACKER
(TEAM ID : PNT2022TMID34929)

PROJECT REPORT

Submitted by

SHERIN.F (962819106040)
M.MAARI MAHESWARI (962819106028)
J.S.RAAM SIVANI (962819106030)
M.S.SOWMIYA (962819106041)

in partial fulfillment for the award of degree of
BACHELOR OF ENGINEERING
in
ELECTRONICS AND COMMUNICATION ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING, NAGERCOIL

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2022

INDEX

13. **APPENDIX**

Source Code

GitHub & Project Demo Link

1) INTRODUCTION

Personal finance management is an important part of people's lives. However, everyone does not have the knowledge or time to manage their finances in a proper manner. And, even if a person has time and knowledge, they do not bother with tracking their expenses as they find it tedious and time-consuming. Now, you don't have to worry about managing your expenses, as you can get access to an expense tracker that will help in the active management of your finances.

People tend to overspend without realizing, and this can prove to be disastrous. Using a daily expense manager can help you keep track of how much you spend every day and on what.

a.Project Overview

An expense tracker app allows you to monitor and categorize your expenses across different bank and investment accounts and credit cards. Some of these apps also offer budgeting tools, credit monitoring, mileage tracking, receipt keeping, and advice to grow your net worth.

b.Purpose

Expense tracker is a software or application that helps to keep an accurate record of your money inflow and outflow.

2) LITERATURE SURVEY

a.Existing problem

In a study conducted by Forrester in 2016 surveying small and medium businesses (SMBs) across the world, 56% companies reported expense management as being the biggest challenge for their finance departments.

In another survey conducted by Levvel Research in 2018 in North America, respondents reported the following pain points in expense management before adopting automation:

- Manual entry and routing of expense reports (62%)
- Lack of visibility into spend data (42%)
- Inability to enforce travel policies (29%)
- Lost expense reports (24%)
- Lengthy expense approval system and reimbursement cycles (23%)

b.References

Link: https://blog.coupler.io/personal-expenses-tracker-google-sheets/#Expense_Tracker

Link:http://www.inappsettingskit.com/home

Link: http://www.themobileinnovation.net/smartphones-operating-systems-war-android-vsblackberry-vs-_ios-vs-symbian

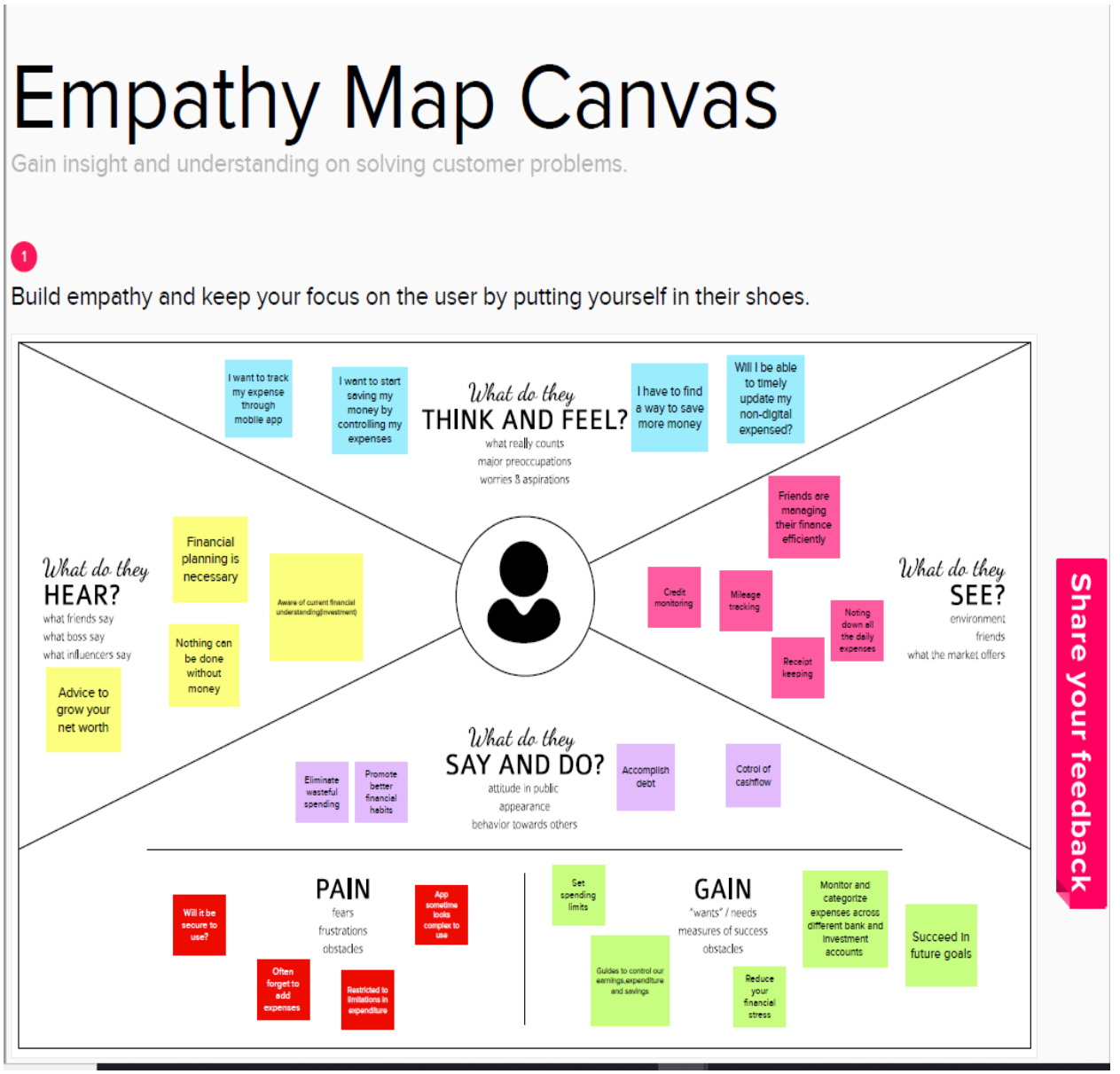
Link: http://www.omnigroup.com/products/omnigraffle/

c.Problem Statement Definition

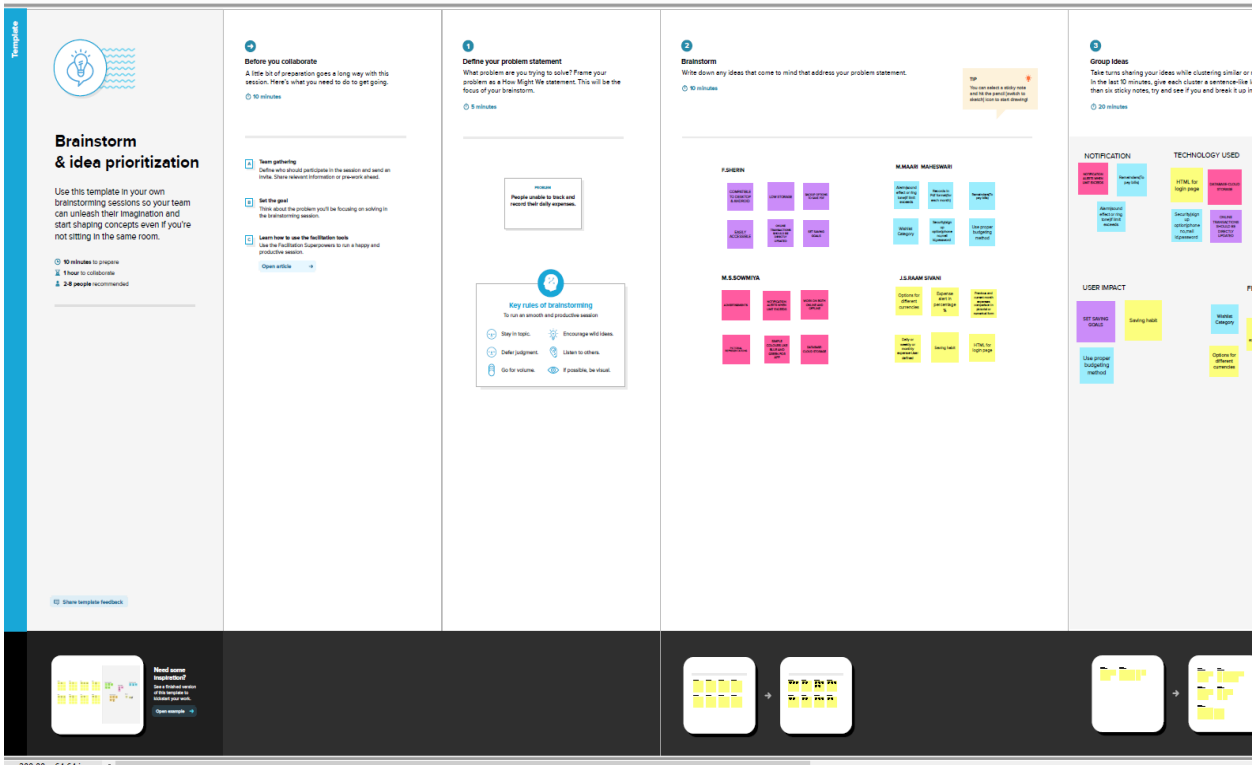
Almost everything in our world turned digital like making transactions , payments ,booking tickets ,etc through our handy(mobile phones). Keeping track of our expenses is an important part of managing our overall finances. People having more money don’t have control over spending them in useful manner. This is due to lack of budgeting and saving habit. This make them suffer at money needed situations. Also people are very busy, unable to make entries of their income, expenditure and savings in diaries or data sheets cause they are very time consuming.

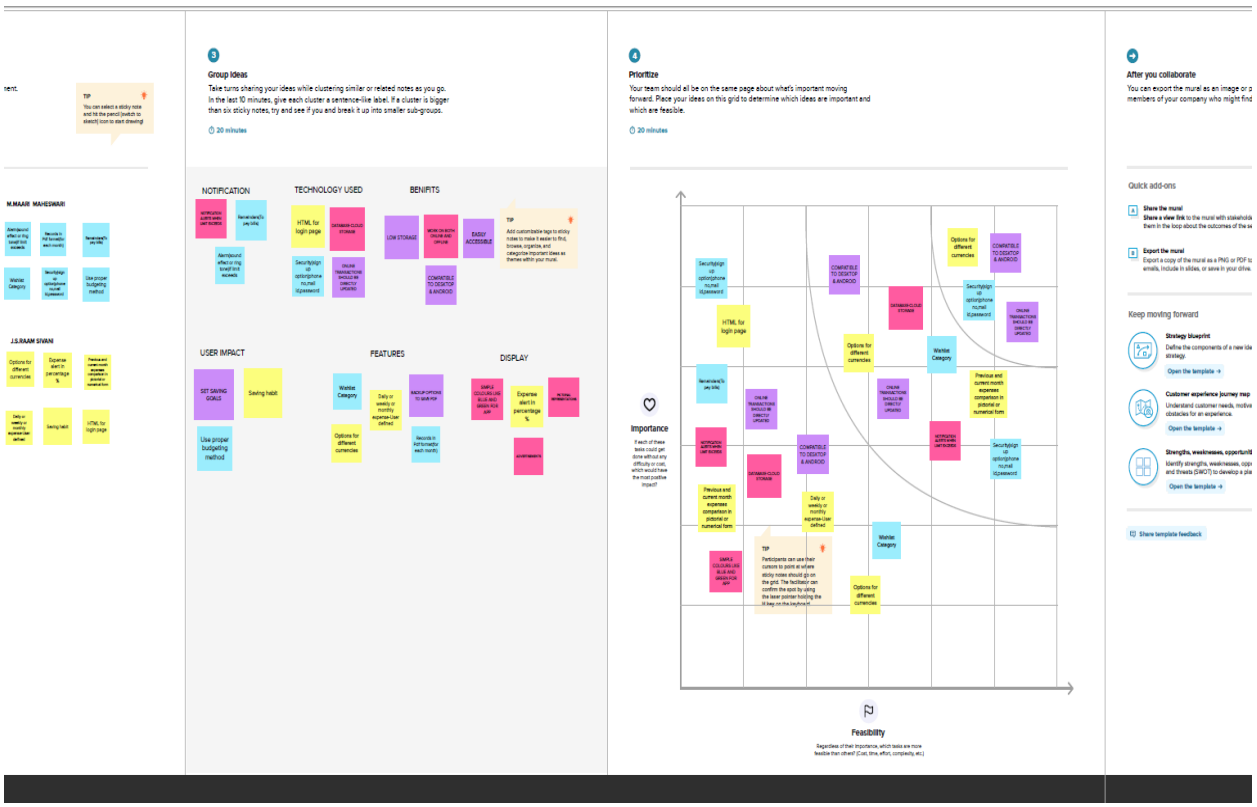
3) IDEATION & PROPOSED SOLUTION

a.Empathy Map Canvas



b.Ideation & Brainstorming





c.Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Almost everything in our world turned digital like making transactions , payments ,booking tickets ,etc through our handy(mobile phones). Keeping track of our expenses is an important part of managing our overall finances. People having more money don't have control over spending them in useful manner. This is due to lack of budgeting and saving habit. This make them suffer at money needed situations. Also people are very busy, unable to make entries of their income, expenditure and savings in diaries or data sheets cause they are very time consuming.
2.	Idea / Solution description	To create a low power consuming app which gets supported to both laptop and mobile phones with no limitations to network(4G,5G,...).It allows user to connect bank account or credit cards to track and it makes direct entries to expense category. An important feature is that we include educational tools like blogs or courses to help people learn more about budgeting and saving. Users can customize their own expense ,income & saving categories. Also anyone can download for free and use it with no subscriptions. We will make it Ad-free so that people will not get irritated.
3.	Novelty / Uniqueness	Ad-free App and have no limits to the data network. We include options to search and make entries through voice. Options for different currencies(₹,$,..)is included.
4.	Social Impact / Customer Satisfaction	We will make sure our app is secure to users so people recommend it to others. Money management skill gets improved. People under debts gets reduced.
5.	Business Model (Revenue Model)	A game(optional) is added, if they score certain points in it, rewards(offers) will be given to the user. Additional points will also be added based on the duration of using the app. The rewards will be like offer to buy a product from the recommended shop.
6.	Scalability of the Solution	Useful to all people in their daily life

d. Problem Solution fit

Team ID: PNT2022TMID34929
Project title: Personal Expense Tracker

Define CS, fit into CC

1. CUSTOMER SEGMENT(S)

- Part time workers
- Business people
- Workers
- Home makers

6. CUSTOMER STATE LIMITATIONS

- Slow response time
- Network connection problem
- Power consumption issues
- Spending money without any care
- No budget

5. AVAILABLE SOLUTIONS

- Low data rate
- Low power consumption
- Remainder(alert) is provided when limit exceeds
- Proper budgeting

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS

- Slow reimbursement
- Human errors
- Expense frauds
- Difficulty in enforcing policies
- Limited access

9. PROBLEM ROOT CAUSE

- Inaccurate receipts
- Lack of visibility into spending trends and patterns
- Delayed submission of expense reports
- Data entry errors
- Lost revenues

7. BEHAVIOUR

- Try to limit their expenses
- While facing issues they are directed in a wrong way
- Labour intensive while using traditional methods
- Use manual methods because of that there maybe miscalculations

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

3. TRIGGERS

Getting interest when we have to deposit cash

4. EMOTIONS: BEFORE / AFTER

- Emphasis on your savings
- Help to get a good idea of your purchasing behaviour

After:

- Happy
- Increase feelings of security and peace of mind.

10. YOUR SOLUTION

- receipt uploading
- Chat support
- Security
- Limited notifications
- Break down barriers and easy to collaborate

8.CHANNELS of BEHAVIOUR

- Online Cookies: Text files with data that identifies a user's computer to improve their browsing experience
- Device IDs: Numbers that identify and track mobile devices for applications and advertisers
- Mapping: Records behavior in a specific place and time
- Geo-location: Uses location tracking through GPS and IP addresses to reveal the location of electronic devices

4)REQUIREMENT ANALYSIS

a. Functional requirements

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Login	Login through Name and Password Login through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Expense & Income category	User can add categories as they wish . Directly or Indirectly the amount can be updated.
FR-4	Chatbot	To clear queries or any technical issues using Watson assistant.
FR-5	Data storage	IBM Cloud used to store and retrieve data through flask API(connect between the app and database).

b. Non-Functional requirements

Non-functional Requirements:

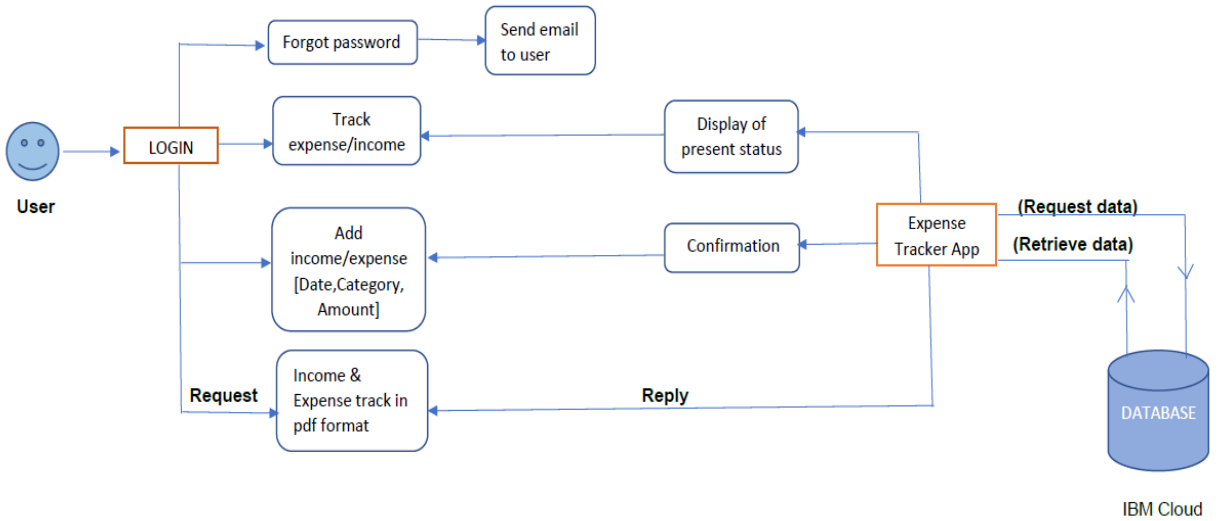
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Helps to keep an accurate record of your money inflow and outflow. Easy to use.
NFR-2	**Security**	Encrypted data packed up regularly to avoid data loss. The users data cannot be accessed without the right credentials.
NFR-3	**Reliability**	Reliable in terms of accuracy & security.
NFR-4	**Performance**	Helps track the particular bills,income,expenses, invoices and transactions that are recurring in nature or the ones that keep coming every once in a while.
NFR-5	**Availability**	Available on android phones, so anyone can use. And available any where , any time.
NFR-6	**Scalability**	Useful to all people in their daily life.

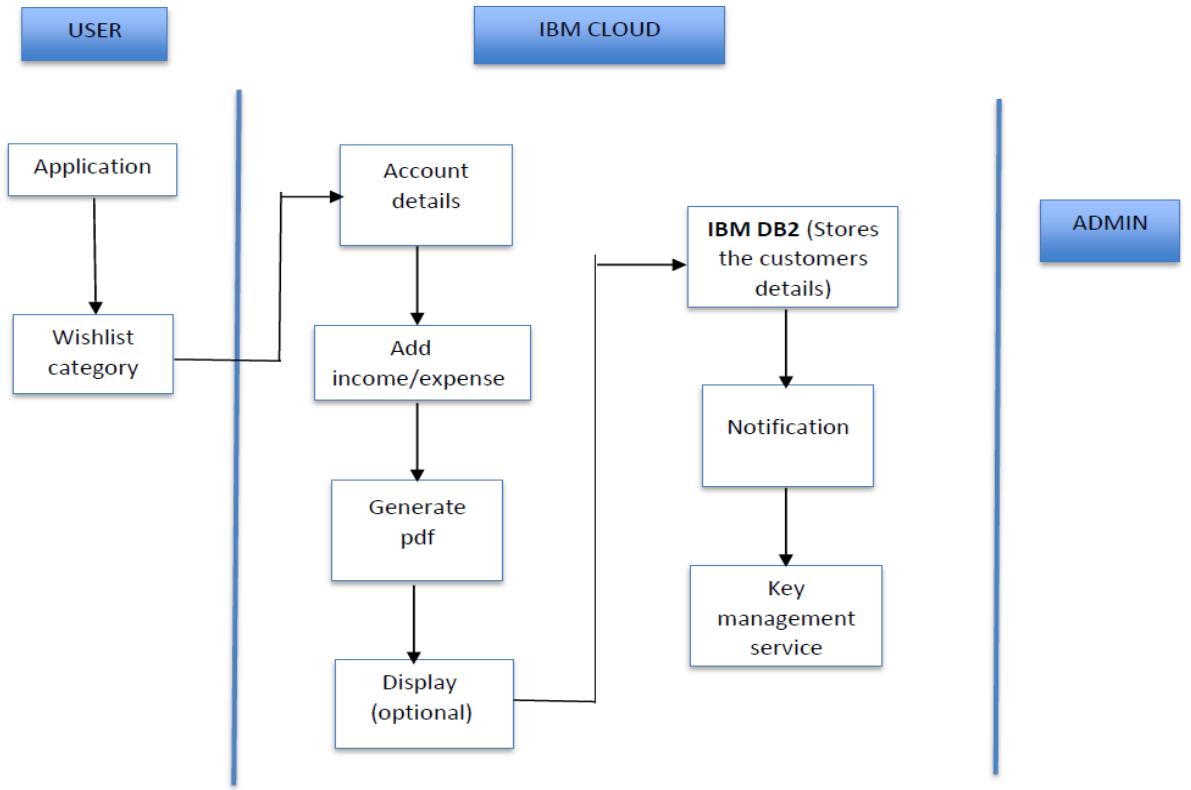
5) PROJECT DESIGN

a.Data Flow diagram

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



b.Solution & Technical Architecture



c.User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile/Web user)	User Login	USN-1	Login through Name and Password	As a registered user and logged out, if I go to the log in page and enter my username and password and click on Log in, then the data associated to my user should be accessible.	High	Sprint-1
		USN-2	Login through Gmail	Continue with google account option if given the login becomes easy.	High	Sprint-2
		USN-3	As a user, I can log into the application by entering email & password	As a registered user and logged out, if I go to the log in page and enter my email and password and click on Log in, then the data associated to my user should be accessible.	High	Sprint-1
	User Confirmation	USN-4	Confirmation via OTP	If I login into any device I need to get verification code as OTP so that no one can access my account	High	Sprint-3
	Expense & Income category	USN-2	Directly the amount can be updated	Whenever I make payments through online it should directly gets updated into the category.	High	Sprint-2
		USN-3	As a user, I can add categories as I wish	I can add expenses/income daily under categories created by my own and those changes should be saved.	High	Sprint-2
	Chatbot	USN-5	To clear queries or any technical issues	If I come across any issue/doubts with the app,I should be able to report and get solution.So chatbots will be supportive .	High	Sprint-3
	Data storage	USN-6	Entered data should be secure, stored and retrieved safely	I can keep my data up to date and make changes easily whenever I need, like adding income/expense.	High	Sprint-4

6) PROJECT PLANNING & SCHEDULING

a. Sprint Planning & Estimation

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

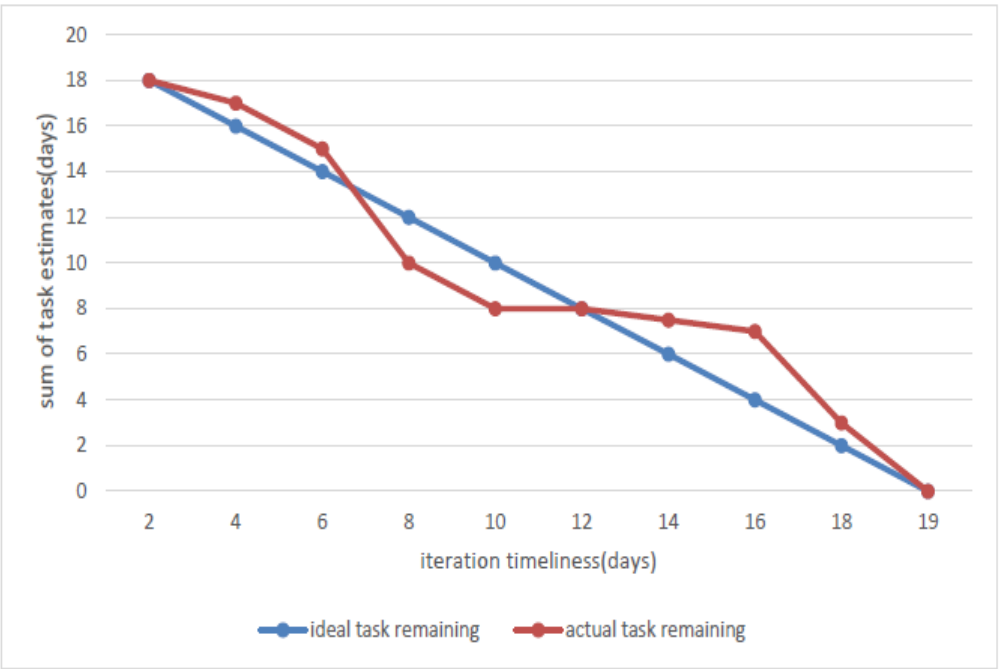
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Maari Maheswari,Sowmiya
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Sherin,Raam Sivani
Sprint-1		USN-3	As a user, I can register for the application through Gmail	2	Medium	Sherin,Maari Maheswari
Sprint-1		USN-4	As a new user, I want to register by creating a username and password so that the system can remember me and my data.	2	High	Maari Maheswari,Raam Sivani
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	1	High	Sherin,Sowmiya
Sprint-1		USN-6	As a registered user, I want to log in with my username and password so that the system can authenticate me and I can trust it.	2	High	Sherin,Sowmiya
Sprint-1		USN-7	As a registered user, I want to be able to occasionally change my password so that I can keep it secure.	3	High	Maari Maheswari,Raam Sivani
Sprint-1		USN-8	As a user, I want a "forgot password" option so that even if I forgot old password I can create new one immediately.	3	High	Sowmiya,Raam Sivani

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-4	Chatbot	USN-10	Equip your expense tracking app with a bot that can understand and answer all user queries and address their needs such as account balance, credit score, etc.	4	Low	Maari Maheswari,Raam Sivani
Sprint-3	Data storage	USN-11	Entered data should be secure, stored and retrieved safely	4	High	Sherin,Sowmiya
Sprint-3		USN-12	Generate reports as PDF files	4	Medium	Maari Maheswari,Sowmiya
Sprint-2	Income and expense category	USN-13	As a user, I can create a category so that I can track expenses by specific categories	4	High	Sherin,Raam Sivani
Sprint-2		USN-14	As a user, I can edit a category so that I can fix any typos I may accidentally make.	3	Low	Sherin,Maari Maheswari
Sprint-2		USN-15	As a user, I can delete a category from an expense/income so that I can remove any mistakes	3	High	Sowmiya, Raam Sivani
Sprint-2		USN-16	As a use, I want the app to remind all the recurrent expenses on a timely basis.	4	High	Sherin,Maari Maheswari
Sprint-2		USN-17	The app should track the payments through online and automatically updates it onto the expense category	4	High	Sowmiya,Raam Sivani
Sprint-2		USN-18	As a user, I want a budget category	2	Medium	Maari Maheswari,Sowmiya
Sprint-3	Visuals	USN-19	As a user, I want to change the background colour as I wish	4	Low	Sherin,Raam Sivani
Sprint-3		USN-20	Statistical report of income vs expenditure should be in easy-to-understand pie chart or bar graph.	4	Medium	Sherin,Sowmiya
Sprint-3		USN-21	Statistical report of income vs expenditure should be in percentage	3	High	Maari Maheswari,Sowmiya
Sprint-4	Currency symbols	USN-22	As a user, I want the app to have many currency symbols	5	Low	Sherin,Raam Sivani

b. Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	16	5 Days	01 Nov 2022	05 Nov 2022	10	16 Nov 2022
Sprint-2	20	5 Days	06 Nov 2022	10 Nov 2022	13	16 Nov 2022
Sprint-3	19	5 Days	11 Nov 2022	15 Nov 2022	15	18 Nov 2022
Sprint-4	12	4 Days	16 Nov 2022	19 Nov 2022	12	19 Nov 2022

c. Reports from JIRA



7) CODING AND SOLUTIONING

Final code -> app.py

```
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 = "3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"
dsn_uid = "sbb93800"
dsn_pwd = "wobsVLm6ccFxcNLe"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "31498"
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'] = '3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'
app.config['port'] = '31498'
app.config['protocol'] = 'tcpip'
app.config['uid'] = 'sbb93800'
app.config['pwd'] = 'wobsVLm6ccFxcNLe'
app.config['security'] = 'SSL'
try:
    mysql = DB2(app)

    conn_str='database=bludb;hostname=3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=31498;pro
tocol=tcpip;\
            uid=sbb93800;pwd=wobsVLm6ccFxcNLe;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 Shyam","shyam123@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 ,
```

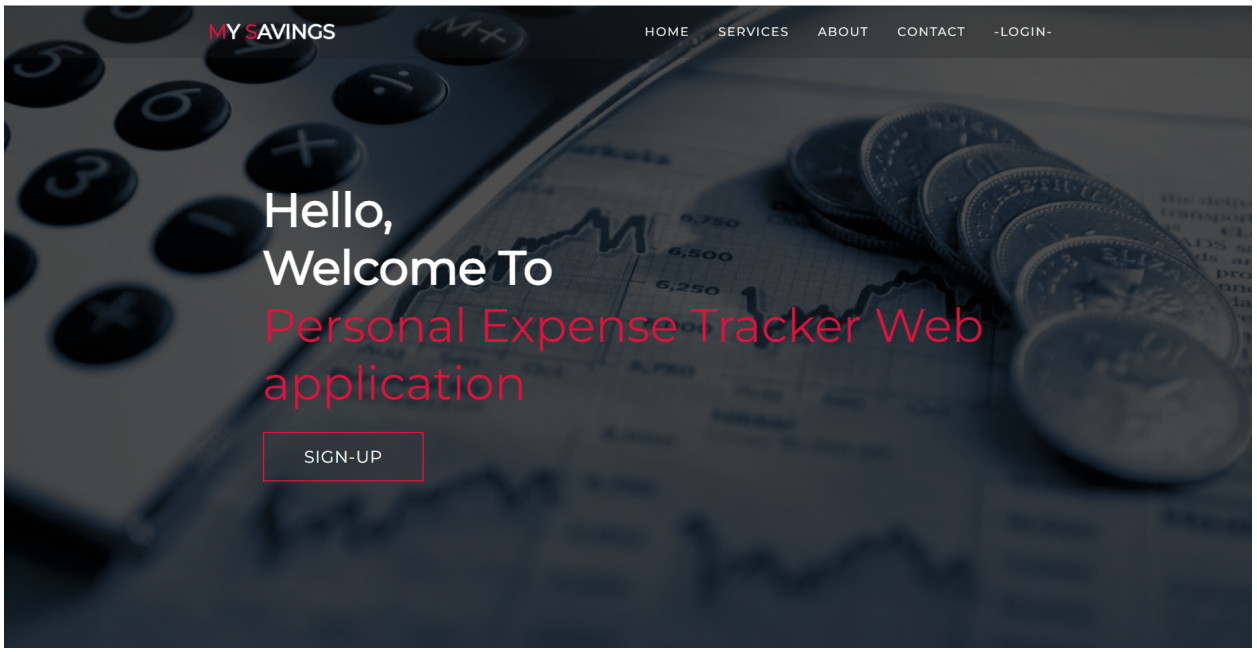
```
                                t_food = t_food,t_entertainment =
t_entertainment,
                                t_business = t_business,  t_rent =  t_rent,
                                t_EMI =  t_EMI,  t_other =  t_other )
#log-out

@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)
```

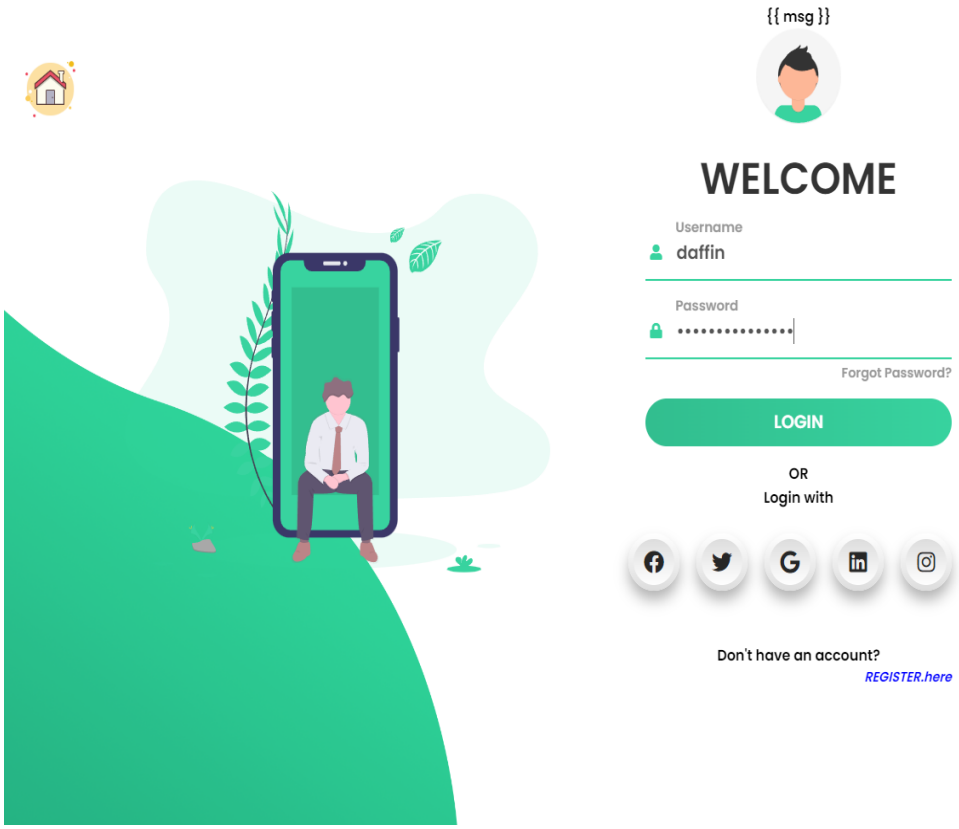
-> sendemail.py

```
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.shyam123@gmail.com", "oms@1Shyam")
    s.login("shyam123@gmail.com", "lxixbmpnexbkiemh")
    message  = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)
    # s.sendmail("il.shyam123@gmail.com", email, message)
    s.sendmail("il.shyam123@gmail.com", email, message)
```

```
    s.quit()
def sendgridmail(user,TEXT):

    # from_email = Email("sneham789@gmail.com")
    from_email = Email("shyam123@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)
```

8) TESTING


MY SAVINGS
HOME
SERVICES
ABOUT
CONTACT
-LOGIN-
Hello,
Welcome To
Personal Expense Tracker Web application
SIGN-UP


{{ msg }}
WELCOME
Username
daffin
Password
Forgot Password?
LOGIN
OR
Login with
Don't have an account?
REGISTER.here

9) RESULTS

● **Tracking income and expenses**: Monitoring the income and tracking all expenditures (through bank accounts).

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

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

10) ADVANTAGES & DISADVANTAGES

ADVANTAGES

->Track your expenses anywhere, anytime.

->Seamlessly manage your money and budget without any financial paperwork.

->Just click and submit your invoices and expenditures.

->Access, submit, and approve invoices irrespective of time and location.

->Avoid data loss by scanning your tickets and bills and saving in the app.

Approval of bills and expenditures in real-time and get notified instantly.

Quick settlement of claims and reduced human errors with an automated and streamlined billing process.

11) CONCLUSION

Monitoring your everyday expenses can set aside you cash, yet it can likewise help you set your monetary objectives for what's to come. On the off chance that you know precisely where your sum is going much of a stretch see where a few reductions and bargains can be made. Expense Tracker project is for keeping our day-to-day expenditures will helps us to keep record of our money daily. The project what we have created is work more proficient than the other income andexpense tracker. The project effectively keeps away from the manual figuring for trying not to ascertain the pay and cost each month. It's a user-friendly application.

12) FUTURE SCOPE

1) It will have various options to keep record (for example Food, Travelling Fuel, Salary etc.)
2) Automatically it will keep on sending notifications for our daily expenditure.
3) In today’s busy and expensive life, we are in a great rush to make moneys, but at the end of the month we broke off. As we are unknowingly spending money on title and unwanted things. So, we have come over with the plan to follow our profit.
4) Here user can define their own categories for expense type like food, clothing, rent and bills where they have to enter the money that has been spend and likewise can add some data in extra data to indicate the expense.

13) APPENDIX

Source code

```
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 = "3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"
dsn_uid = "sbb93800"
dsn_pwd = "wobsVLm6ccFxcNLe"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "31498"
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'] = '3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'
app.config['port'] = '31498'
app.config['protocol'] = 'tcpip'
app.config['uid'] = 'sbb93800'
app.config['pwd'] = 'wobsVLm6ccFxcNLe'
app.config['security'] = 'SSL'
try:
    mysql = DB2(app)

    conn_str='database=bludb;hostname=3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=31498;pro
tocol=tcpip;\
            uid=sbb93800;pwd=wobsVLm6ccFxcNLe;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 Shyam","shyam123@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)
```

Github link:

https://github.com/IBM-EPBL/IBM-Project-44146-1660722589.git

Project Demo link :

https://drive.google.com/file/d/1lUm-zoesHMFuqEmajCsJ2T5KeHWE76dU/view?usp=drivesdk