

DEVELOPMENT PHASE SPRINT - 2

Team ID	PNT2022TMID35639
Project Name	Personal Expense Tracker Application

SCREENSHOTS:

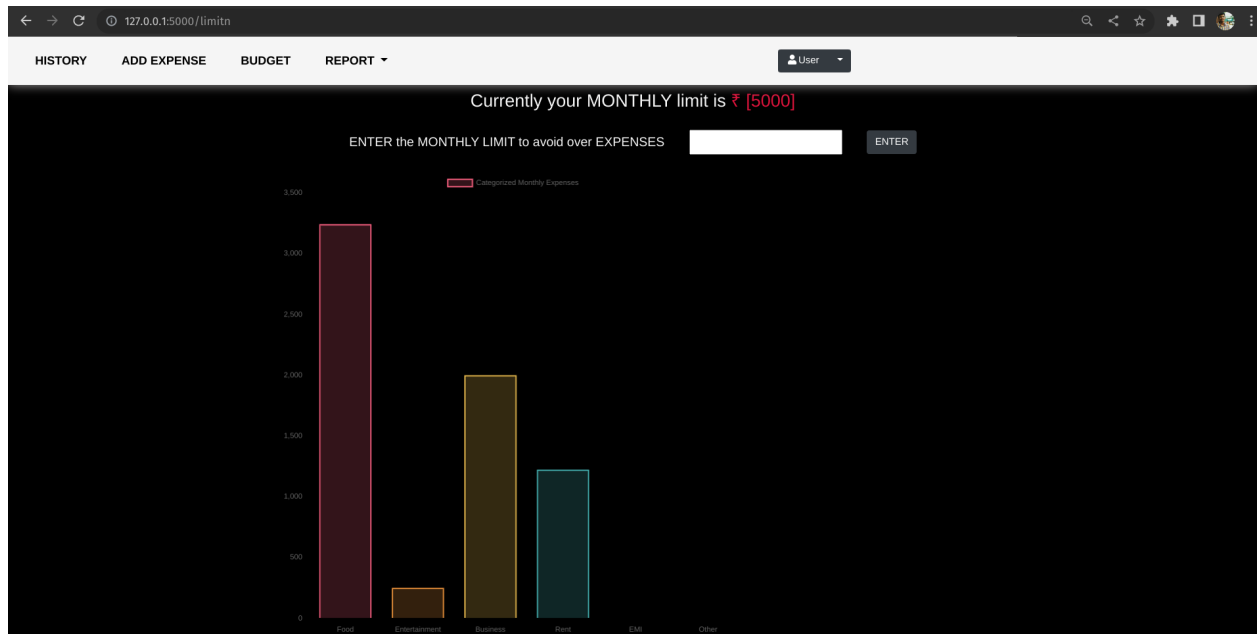
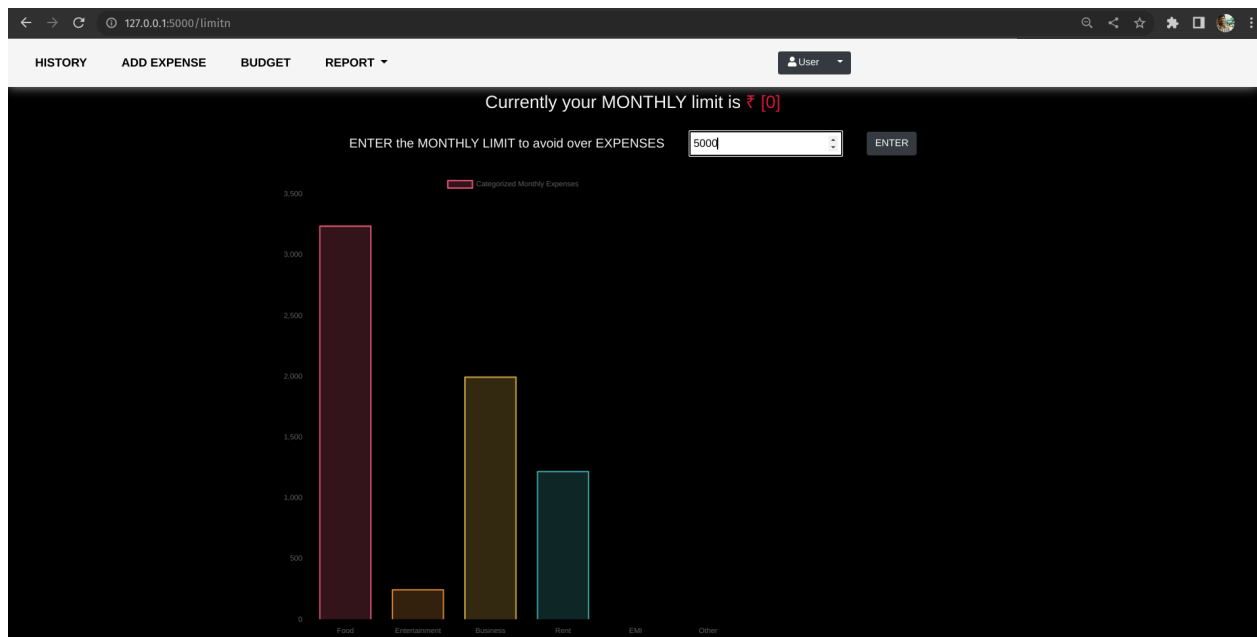
The screenshot shows the 'Add Expense' form in a web application. The form is titled 'Add Expense' and contains the following fields:

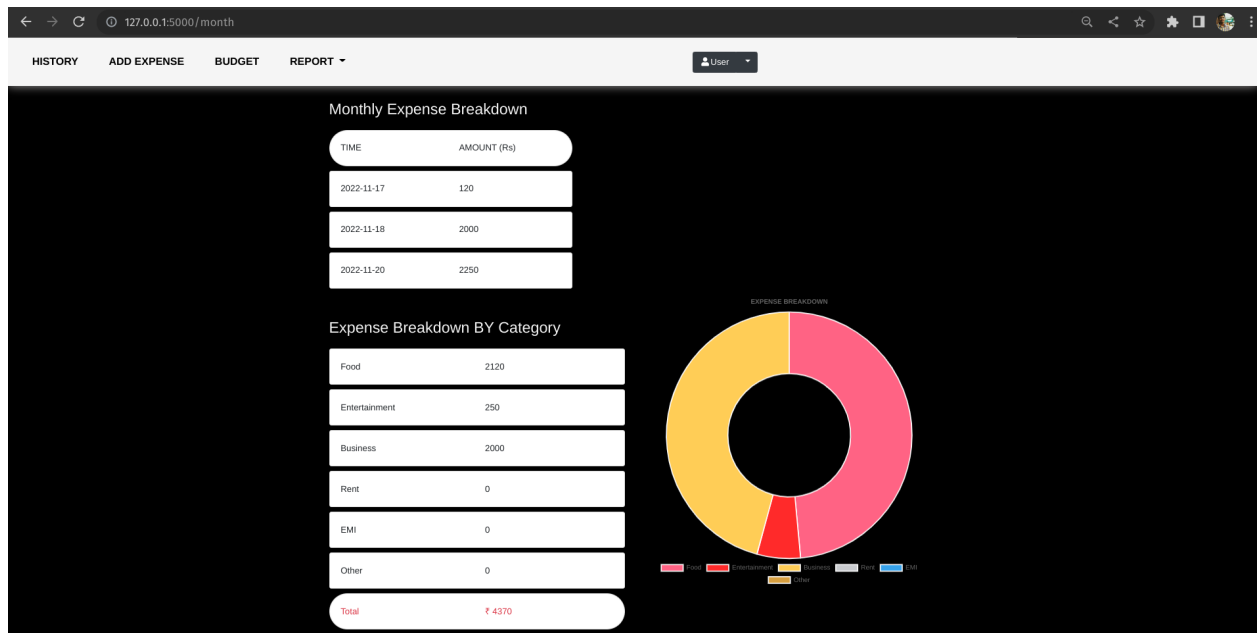
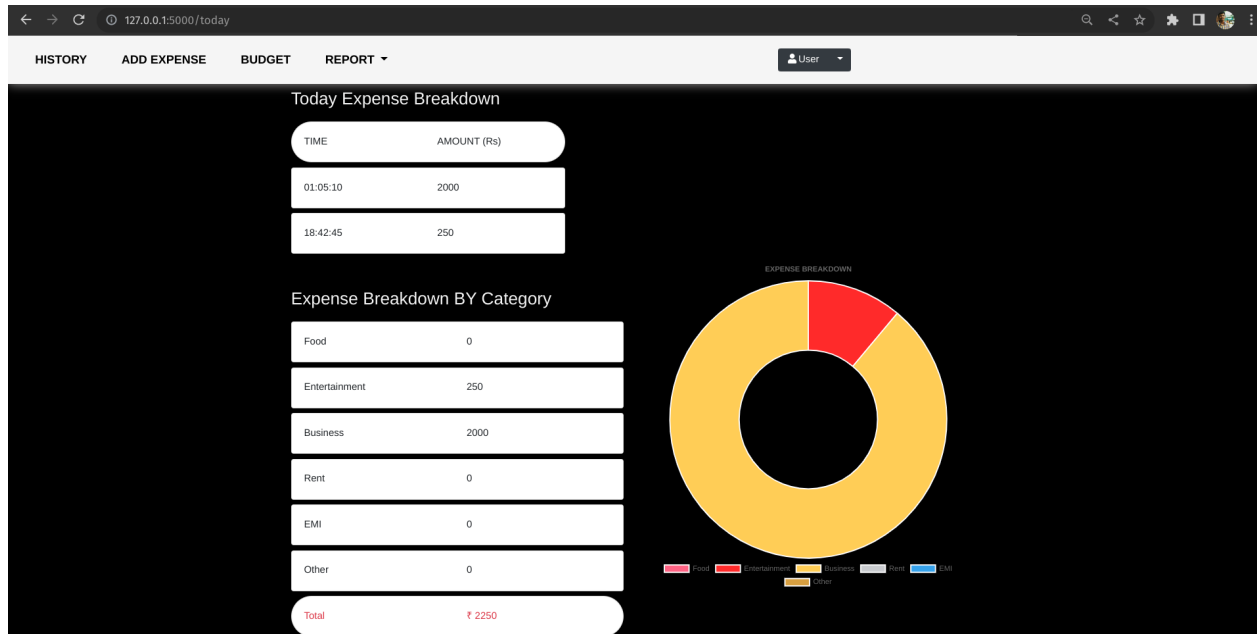
- Date: 11/20/2022, 06:42:45 PM
- Expense name: Wakanda Forever
- Expense Amount: 250
- Payment method: epayment
- Category: Entertainment

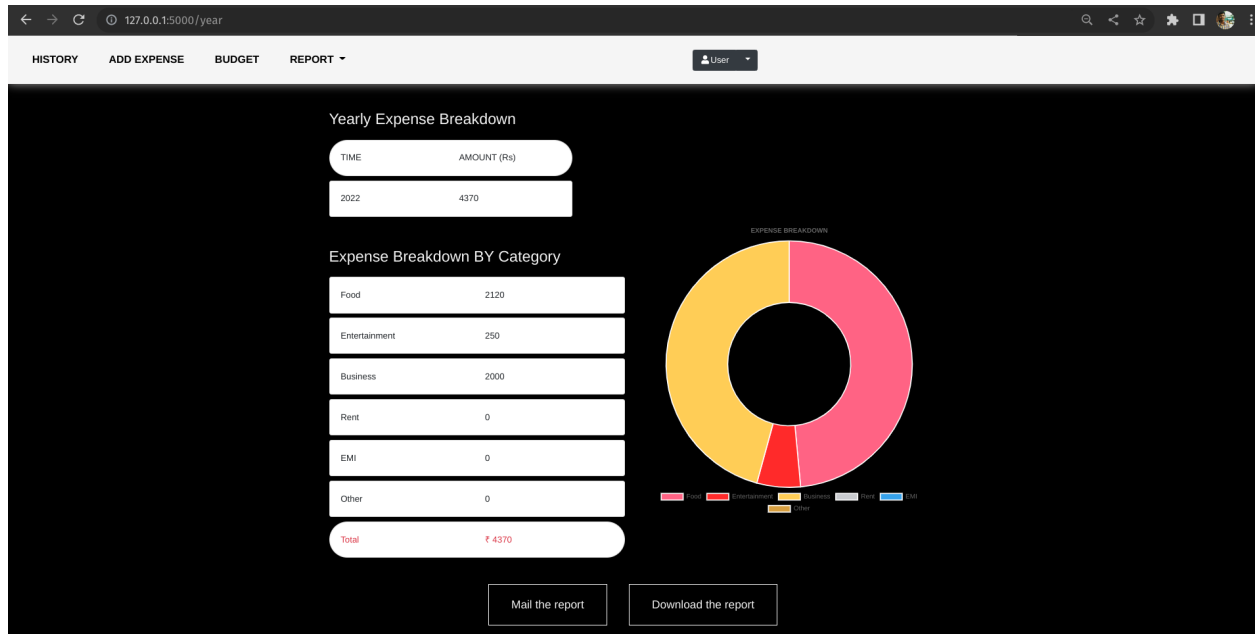
There is an 'Add' button at the bottom of the form.

The screenshot shows the 'EXPENSES' list in the application. The list contains the following entries:

Date	Expense name	Amount	Payment method	Category	Edit	Delete
2022-11-20 18:42:45	Wakanda Forever	₹ 250	epayment	entertainment	Edit	Delete
2022-11-20 12:35:16	hostel fee	₹ 1122	debitcard	food	Edit	Delete
2022-11-20 01:09:23	test	₹ 1223	creditcard	rent	Edit	Delete
2022-11-20 01:05:10	ibm	₹ 2000	cash	business	Edit	Delete
2022-11-18 12:00:11	SANDY	₹ 2000	cash	food	Edit	Delete
2022-11-17					Edit	Delete







CODE:

```
# -*- coding: utf-8 -*-  
"""
```

```
Spyder Editor  
This is a temporary script file.  
"""
```

```
from flask import Flask, render_template, request, redirect, session , make_response, url_for,  
json, flash  
#from flask_mysqlDb import MySQL  
#import MySQLdb.cursors  
import re, ibm_db  
import ibm_db_dbi ,pandas as pd  
from flask_mail import Mail, Message  
import os, datetime  
from pandas import Timestamp  
from pretty_html_table import build_table  
import pdfkit  
  
app = Flask(__name__)  
  
app.secret_key = 'SECRET_KEY'
```

```
conn = ibm_db.connect("DATABASE=bludb;  
HOSTNAME=fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdo  
main.cloud; PORT=32731; SECURITY=SSL; SSLServerCertificate=DigiCertGlobalRootCA.crt;  
UID=rfh77431;PWD=1WClhcJWgdCoeAk5",",")  
pd_conn = ibm_db_dbi.Connection(conn)
```

```
#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 = "
```

```
    if request.method == 'POST' :
```

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

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

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

```
        sql = 'SELECT * from REGISTER WHERE USERNAME = ?'
```

```
        stmt = ibm_db.prepare(conn, sql)
```

```
        ibm_db.bind_param(stmt, 1, username)
```

```
        ibm_db.execute(stmt)
```

```
        account = ibm_db.fetch_assoc(stmt)
```

```
        print(account)
```

```
        if account:
```

```
            msg = 'Account 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:
    sql = "INSERT INTO REGISTER(USER_ID,USERNAME,EMAIL,PASSWORD)
VALUES(DEFAULT,?,?,?)"
    stmt=ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,username)
    ibm_db.bind_param(stmt,2,email)
    ibm_db.bind_param(stmt,3,password)
    ibm_db.execute(stmt)

    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']
```

```
        sql = 'SELECT * from REGISTER WHERE USERNAME = ? AND PASSWORD = ?'
```

```
        stmt = ibm_db.prepare(conn, sql)
```

```
        ibm_db.bind_param(stmt, 1, username)
```

```
        ibm_db.bind_param(stmt, 2, password)
```

```
        ibm_db.execute(stmt)
```

```
        account = ibm_db.fetch_assoc(stmt)
```

```
        print (account)
```

```
        if account:
```

```
            session['loggedin'] = True
```

```

        session['id'] = account['USER_ID']
        userid = account['USER_ID']
        session['username'] = account['USERNAME']

        return redirect('/add')
    else:
        msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)

```

#ADDING----DATA

@app.route("/add")

def adding():

```

    if session.get("id")== None or session.get("username") == None:
        return redirect('/')

```

```

    return render_template('add.html')

```

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

def addexpense():

```

    if session.get("id")== None or session.get("username") == None:
        redirect('/')

```

```

    date = request.form['date']

```

```

    date = AlterDate(date)

```

```

    expensename = request.form['expensename']

```

```

    amount = request.form['amount']

```

```

    paymode = request.form['paymode']

```

```

    category = request.form['category']

```

```

    print(date)

```

```

    rep = generateReport('Monthly')

```

```

    monthly_expense = rep['total'][0]

```

```

    print(type(monthly_expense),type(amount))

```

```

    sql = "INSERT INTO EXPENSES VALUES(DEFAULT,?,?,?,?,?,?)"

```

```

    stmt = ibm_db.prepare(conn, sql)

```

```

    ibm_db.bind_param(stmt, 1, session['id'])

```

```

    ibm_db.bind_param(stmt, 2, date)

```

```

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)

```

```

sql = 'SELECT limitss FROM LIMITS WHERE USER_ID = {}'.format(session['id'])
df = pd.read_sql(sql,pd_conn)
row = df.values.tolist()

```

```

if(len(row) > 0 and int(amount) + int(monthly_expense) > int(row[0][0]) ):
    flash("ALERT!!! You have crossed your monthly limit")
    sendLimitAlert(int(amount) + int(monthly_expense) - int(row[0][0]))

```

```

print(date + " " + expensename + " " + amount + " " + paymode + " " + category)

```

```

return redirect("/display")

```

```

def AlterDate(date):
    temp = date.split('T')
    time = temp[1].replace(':', '.')

    return temp[0]+'-'+time

```

```

#DISPLAY---graph
@app.route("/display")
def display():
    if session.get("id")== None or session.get("username") == None:
        return redirect('/')

    print(session["username"],session['id'])

    id = str(session['id'])

    sql = 'SELECT * FROM EXPENSES WHERE USER_ID = {} ORDER BY DATE
DESC'.format(id)
    df = pd.read_sql(sql,pd_conn)
    expense = df.values.tolist()

    print(expense)

```



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

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

```
@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
def delete(id):
    sql = "DELETE FROM expenses WHERE EXPENSE_ID = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, id)
    ibm_db.execute(stmt)

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

```
#UPDATE---DATA
```

```
@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    sql = 'SELECT * FROM EXPENSES WHERE EXPENSE_ID = {}'.format(id)
    df = pd.read_sql(sql,pd_conn)
    row = df.values.tolist()

    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']
        print(AlterDate(date))
        expensename = request.form['expensename']
        amount = request.form['amount']
        paymode = request.form['paymode']
        category = request.form['category']
```

```
        sql = "UPDATE expenses SET DATE = ?, EXPENSE_NAME = ?, AMOUNT = ?,PAYMODE
= ?, CATEGORY = ? WHERE EXPENSE_ID = ?"
        stmt = ibm_db.prepare(conn, sql)
```

```
ibm_db.bind_param(stmt, 1, AlterDate(date))
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":
        if session.get("id")== None or session.get("username") == None:
            return redirect('/')
```

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

```
    sql = 'SELECT limitss FROM LIMITS WHERE USER_ID = {}'.format(session['id'])
    df = pd.read_sql(sql,pd_conn)
    row = df.values.tolist()
```

```
    if(len(row) > 0):
```

```
        sql = "UPDATE LIMITS SET LIMITSS = ? WHERE USER_ID = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,number)
        ibm_db.bind_param(stmt,2,session['id'])
        ibm_db.execute(stmt)
```

```
    else:
```

```
        sql = "INSERT INTO LIMITS VALUES(DEFAULT,?,?)"
        stmt = ibm_db.prepare(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():
```

```
    if session.get("id") == None or session.get("username") == None:
```

```
        return redirect('/')

    sql = 'SELECT limitss FROM LIMITS WHERE USER_ID = {}'.format(session['id'])
    df = pd.read_sql(sql,pd_conn)
    row = df.values.tolist()

    s = 0
    if(len(row) > 0):
        s = row[0]

    rep = generateReport('Monthly')
    monthly_expense = list()
    for key,val in rep.items():
        if(key != 'texpanse' and key!='total'):
            print(key)
            monthly_expense.append(val[0])

    return render_template("limit.html", type="Monthly",expense_data=monthly_expense, y=s)

#Generate Report Today, Monthly, Yearly
def generateReport(report_type):
    if(session.get('id') == None):
        return

    id = str(session['id'])

    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0
    expense = []
    texpanse = []
```

```

if(report_type == 'Today'):
    sql = 'SELECT TIME(date), AMOUNT FROM EXPENSES WHERE USER_ID = {} AND
DATE(date) = DATE(NOW())'.format(id)
    df = pd.read_sql(sql,pd_conn)
    texpanse = df.values.tolist()

    sql = 'SELECT * FROM EXPENSES WHERE USER_ID={} AND DATE(DATE) =
(CURRENT_DATE) ORDER BY AMOUNT DESC, DATE DESC'.format(id)
    df = pd.read_sql(sql,pd_conn)
    expense = df.values.tolist()

elif(report_type == 'Monthly'):
    sql = 'SELECT DATE(date), SUM(AMOUNT) FROM EXPENSES WHERE USER_ID = {}
AND MONTH(DATE(date)) = MONTH(CURRENT_DATE) GROUP BY DATE(date)'.format(id)
    df = pd.read_sql(sql,pd_conn)
    texpanse = df.values.tolist()

    sql = 'SELECT * FROM EXPENSES WHERE USER_ID={} AND MONTH(DATE(DATE)) =
MONTH(CURRENT_DATE) ORDER BY AMOUNT DESC, DATE DESC'.format(id)
    df = pd.read_sql(sql,pd_conn)
    expense = df.values.tolist()

elif(report_type == 'Yearly'):
    sql = 'SELECT YEAR(date), SUM(AMOUNT) FROM EXPENSES WHERE USER_ID = {}
AND YEAR(DATE(date)) = YEAR(CURRENT_DATE) GROUP BY YEAR(date)'.format(id)
    df = pd.read_sql(sql,pd_conn)
    texpanse = df.values.tolist()

    sql = 'SELECT * FROM EXPENSES WHERE USER_ID={} AND YEAR(DATE(DATE)) =
YEAR(CURRENT_DATE) ORDER BY AMOUNT DESC, DATE DESC'.format(id)
    df = pd.read_sql(sql,pd_conn)
    expense = df.values.tolist()

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]

    return {'texpanse':texpanse,'t_food':[t_food],'t_entertainment':[t_entertainment],
            't_business':[t_business],'t_rent':[t_rent],
            't_EMI':[t_EMI],'t_other':[t_other],'total':[total] }

#REPORT
#daily report
@app.route("/today")
def today():
    if(session.get('id') == None):
        return redirect('/')

    rep = generateReport('Today')
    print(rep)
    return render_template("report.html", type="Today", texpanse = rep['texpanse'], total =
rep['total'][0] ,
        t_food = rep['t_food'][0],t_entertainment = rep['t_entertainment'][0],
        t_business = rep['t_business'][0], t_rent = rep['t_rent'][0],
        t_EMI = rep['t_EMI'][0], t_other = rep['t_other'][0] )

# Monthly report
@app.route("/month")
def month():

    if(session.get('id') == None):
        return redirect('/')

    rep = generateReport('Monthly')
    return render_template("report.html", type="Monthly", texpanse = rep['texpanse'], total =
rep['total'][0] ,
        t_food = rep['t_food'][0],t_entertainment = rep['t_entertainment'][0],
        t_business = rep['t_business'][0], t_rent = rep['t_rent'][0],
        t_EMI = rep['t_EMI'][0], t_other = rep['t_other'][0] )

```

```

# Yearly report
@app.route("/year")
def year():
    if(session.get('id') == None):
        return redirect('/')

    rep = generateReport('Yearly')
    return render_template("report.html", type="Yearly", texpanse = rep["texpanse"], total =
rep["total"][0] ,
        t_food = rep["t_food"][0], t_entertainment = rep["t_entertainment"][0],
        t_business = rep["t_business"][0], t_rent = rep["t_rent"][0],
        t_EMI = rep["t_EMI"][0], t_other = rep["t_other"][0] )

#log-out
@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    return render_template('home.html')

if __name__ == "__main__":
    app.run(debug=True)

```