Sprint 4
PROJECT_DEVELOPMENT_PHASE

Date	15 November 2022
Team ID	PNT2022TMID49592
Project Name	Project - Personal expense tracker application

CODE:

app.py(updated):

```
from flask import Flask, render_template, request, redirect, url_for, session
import ibm_db
import re
from random import *
from flask_mail import Mail, Message
app = Flask(__name__)
mail = Mail(app) # instantiate the mail class

# configuration of mail
app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'shrikumar13102001@gmail.com'
app.config['MAIL_PASSWORD'] = 'mlzuhzieegwenrhc'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
mail = Mail(app)
otp = randint(000000,999999)
app.secret_key = 'a'
conn =
ibm_db.connect("DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-10cf081900bf.c1o
gj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=SSL;SSLServerCerti
ficate=DigiCertGlobalRootCA.crt;UID=jpg92341;PWD=5HWT2lmD7POkuLnv","","")
@app.route("/")
def home():
  return render_template('index.html')
@app.route('/login', methods =['GET', 'POST'])
def login():
   global userid
   msg = ''
   if request.method == 'POST'and 'email' in request.form and 'password1' in request.form:
      email = request.form['email']
      password1 = request.form['password1']
```

```
        stmt = ibm_db.prepare(conn,'SELECT * FROM regi WHERE email = ? AND password1 =
?')
        ibm_db.bind_param(stmt,1,email)
        ibm_db.bind_param(stmt,2,password1)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_tuple(stmt)
        if account:
            session['id'] = account[0]
            userid =  account[0]
            session['email'] = account[1]
            msg = 'Logged in successfully !'
            return render_template('button.html', name=session['email'])
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)
@app.route('/button')
def button():
    return render_template('button.html')

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

    if 'id' in session and 'email' in session:
        uid = session['id']
        exist = ibm_db.prepare(conn, 'SELECT uid, limit FROM limit WHERE uid = ?')
        ibm_db.bind_param(exist,1, session['id'])
        ibm_db.execute(exist)
        exist = ibm_db.fetch_tuple(exist)
        if request.method == "POST":
            print("Executing INSERT into LIMIT")
            uid = session['id']
            stmt = ""
            if exist == False:
               print("Creating New")
               stmt = ibm_db.prepare(conn, 'INSERT INTO limit (limit, uid) VALUES (?, ?)')
            else:
              print("Updating")
              stmt = ibm_db.prepare(conn, 'UPDATE limit SET \
              limit = ? \
              WHERE uid = ?')


            ibm_db.bind_param(stmt, 1,  request.form['limit'])
            ibm_db.bind_param(stmt, 2,  uid)
```

```
            ibm_db.execute(stmt)

            return render_template('limit.html', status="Success", limit=int(request.form['limit']))
        else:
             print(exist[1])
             limit = int(exist[1])
             return render_template('limit.html', limit=limit)
    return 'Not Authed'

@app.route('/stat')
def stat():
    if 'id' in session:
        stmt = ibm_db.prepare(conn, 'SELECT expense, expensedate FROM expense WHERE
uid = ?')
        ibm_db.bind_param(stmt,1,session['id'])
        ibm_db.execute(stmt)
        tb = ibm_db.fetch_tuple(stmt)

        months = {}
        while tb != False:
            sliced = tb[1].strftime("%b")
            if sliced in months:
                months[sliced] += tb[0]
            else:
                months[sliced] = tb[0]
            tb = ibm_db.fetch_tuple(stmt)

        return render_template('stat.html', data=months)
    return 'Not Authed Please Login'
@app.route('/display')
def display():
    if 'id' in session :
        stmt = ibm_db.prepare(conn, 'SELECT amount, income, expense, expensedate, category
FROM expense WHERE uid = ?')
        ibm_db.bind_param(stmt,1,session['id'])
        ibm_db.execute(stmt)

        tb = ibm_db.fetch_assoc(stmt)
        data = []
        while tb != False:
            data.append(tb)
            tb = ibm_db.fetch_assoc(stmt)
        print(data)
        return render_template('display.html',data=data)
```

```
    return 'Not Authed'

@app.route('/wallet')
def wallet():
    return render_template('design.html')
@app.route('/income', methods =['GET', 'POST'])
def income():
    if 'id' in session and request.method == 'POST' and ('amount' in request.form) and ('income' in
request.form) and ('expense' in request.form) and ('expensedate' in request.form )and
('category' in request.form):
        msg = ''
        uid = session['id']
        amount = request.form['amount']
        income = request.form['income']
        expense = request.form['expense']
        expensedate = request.form['expensedate']
        category = request.form['category']

        limit = ibm_db.prepare(conn, "SELECT limit from limit WHERE uid = ?")
        ibm_db.bind_param(limit, 1, session['id'])
        ibm_db.execute(limit)

        data = ibm_db.fetch_tuple(limit)
        if data != False:
            data = data[0]
            sum = 0
            all_expenses = ibm_db.prepare(conn, "SELECT expense from expense WHERE uid
= ?")
            ibm_db.bind_param(all_expenses, 1, session['id'])
            ibm_db.execute(all_expenses)
            expense_data = ibm_db.fetch_tuple(all_expenses)

            while expense_data != False:
                sum += int(expense_data[0])
                expense_data = ibm_db.fetch_tuple(all_expenses)

        sum += int(expense)
        email = (session['email'] + '@gmail.com').lower()
        if sum >= data:
            #Limit Exceeded
            msg = Message('Limit Exceeded', sender='shrikumar13102001@gmail.com',
recipients=[email])
            msg.body = "You Have Exceeded Your Expense Limit"
```

```
            mail.send(msg)
            msg = "Exceeded Limit"
            return render_template('add.html', a = msg)

        prep_stmt = ibm_db.prepare(conn, "INSERT INTO
expense(uid,amount,income,expense,expensedate,category) VALUES (?, ?, ?, ?, ?, ?)")
        ibm_db.bind_param(prep_stmt, 1, uid)
        ibm_db.bind_param(prep_stmt, 2, amount)
        ibm_db.bind_param(prep_stmt, 3, income)
        ibm_db.bind_param(prep_stmt, 4, expense)
        ibm_db.bind_param(prep_stmt, 5, expensedate)
        ibm_db.bind_param(prep_stmt, 6, category)
        ibm_db.execute(prep_stmt)
        msg = 'You have successfully added !'
        return render_template('add.html', a = msg)
    return render_template('add.html')

@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST':
        firstname = request.form['firstname']
        lastname = request.form['lastname']
        email = request.form['email']
        password1 = request.form['password1']
        cpassword = request.form['cpassword']
        birthdate = request.form['birthdate']
        phonenumber = request.form['phonenumber']
        job = request.form['job']
        income = request.form['income']
        sql = "SELECT * FROM regi WHERE email = ? "
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,email)
        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]+', firstname):
            msg = 'firstname must contain only characters and numbers !'
        elif not email or not firstname or not password1:
            msg = 'Please fill out the form !'
```

```
        else:
            insert_sql = "INSERT INTO
regi(firstname,lastname,email,password1,cpassword,birthdate,phonenumber,job,income)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)"
            stmt = ibm_db.prepare(conn,insert_sql)
            ibm_db.bind_param(stmt, 1, firstname)
            ibm_db.bind_param(stmt, 2, lastname)
            ibm_db.bind_param(stmt, 3, email)
            ibm_db.bind_param(stmt, 4, password1)
            ibm_db.bind_param(stmt, 5, cpassword)
            ibm_db.bind_param(stmt, 6, birthdate)
            ibm_db.bind_param(stmt, 7, phonenumber)
            ibm_db.bind_param(stmt, 8, job)
            ibm_db.bind_param(stmt, 9, income)
            ibm_db.execute(stmt)
            msg = 'You have successfully registered !'
            return render_template('email.html')
    elif request.method == 'POST':
        msg = 'Please fill out the form !'

    return render_template('register.html', msg = msg)
@app.route('/verify', methods=['GET', 'POST'])
def verify():
    if request.method == 'POST':
        email1 = request.form['email1']
        sql = "SELECT * FROM validate WHERE email1 = ? "
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,email1)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account already exists !'
        else:
          insert_sql = "INSERT INTO validate VALUES (?)"
          stmt = ibm_db.prepare(conn,insert_sql)
          ibm_db.bind_param(stmt, 1, email1)
          ibm_db.execute(stmt)
          msg = Message('Hello',sender ='shrikumar13102001@gmail.com',recipients = [email1])
          msg.body = str(otp)
          mail.send(msg)
          return render_template('verify.html')
    return render_template('verify.html')
@app.route('/validate',methods=['GET', 'POST'])
```

```
def validate():
 user_otp = request.form['otp']
 if otp == int(user_otp):
    return render_template('index.html')
 return render_template('index.html')

@app.route('/logout', methods=['GET'])
def logout():
    if 'email' in session:
        del session['email']
        del session['id']
        return redirect('/')
    return redirect('/')


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

limit.html:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.c
ss">
<style>
* {box-sizing: border-box;}

body {
  margin: 0;
  font-family: Arial, Helvetica, sans-serif;
}
#navbar {
  overflow: hidden;
  background-image: linear-gradient(120deg, #d4fc79 0%, #96e6a1 100%);
  padding: 90px 10px;
  transition: 0.4s;
  position: fixed;
  width: 100%;
  top: 0;
```

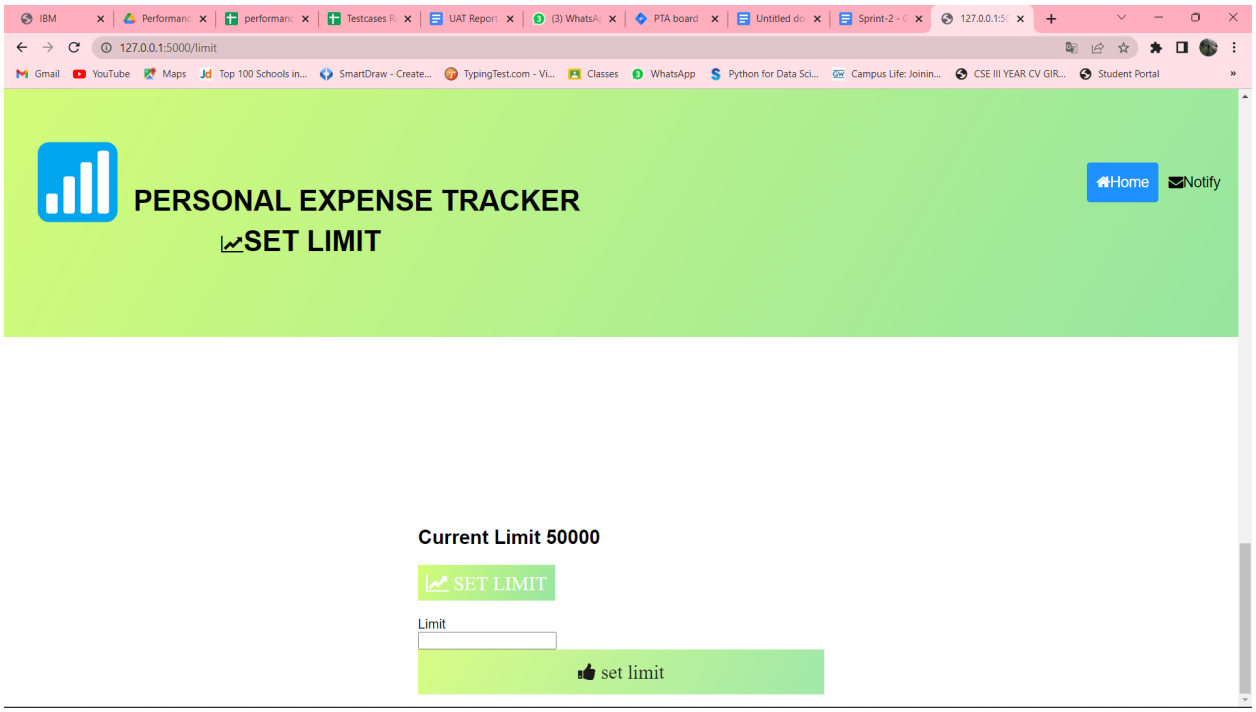
```
  z-index: 99;
}

#navbar a {
  float: left;
  color:black;
  text-align: center;
  padding: 12px;
  text-decoration: none;
  font-size: 18px;
  line-height: 25px;
  border-radius: 4px;
}

#navbar #logo {
  font-size: 35px;
  font-weight: bold;
  transition: 0.4s;
}

#navbar a:hover {
  background-color: #ddd;
  color: black;
}

#navbar a.active {
  background-color: dodgerblue;
  color: white;
}

#navbar-right {
  float: right;
}

@media screen and (max-width: 580px) {
  #navbar {
    padding: 20px 10px !important;
  }
  #navbar a {
    float: none;
```

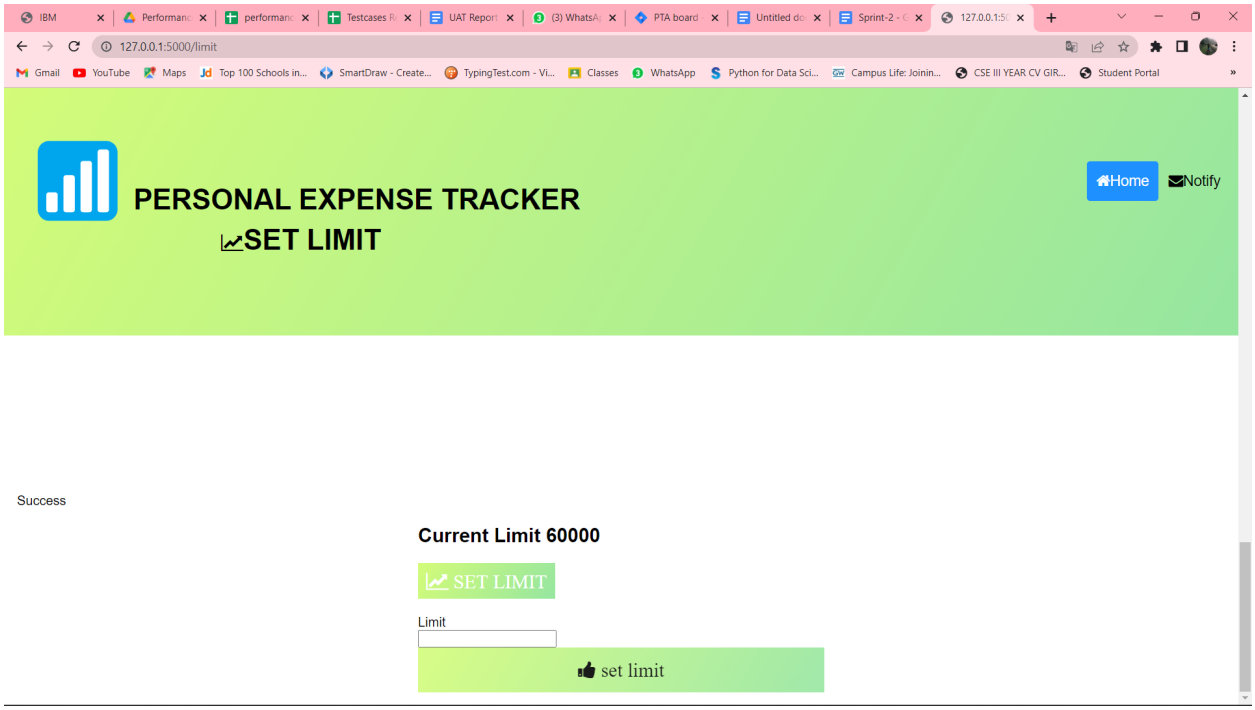
```
    display: block;
    text-align: left;
  }
  #navbar-right {
    float: none;
  }
}
.content {
  padding: 16px;
}

.sticky {
  position: fixed;
  top: 0;
  width: 100%;
}

.sticky + .content {
  padding-top: 60px;
}
.input-container {
  display: -ms-flexbox; /* IE10 */
  display: flex;
  width: 100%;
  margin-bottom: 30px;
}

.icon {
  padding: 10px;
  background-image: linear-gradient(120deg, #d4fc79 0%, #96e6a1 100%);
  color: white;
  min-width: 70px;
  text-align: center;
}

.input-field {
  width: 100%;
  padding: 10px;
  outline: none;
}
```

```
.input-field:focus {
    background-image: linear-gradient(120deg, #d4fc79 0%, #96e6a1 100%);
}

/* Set a style for the submit button */
.btn {
    background-image: linear-gradient(120deg, #d4fc79 0%, #96e6a1 100%);
  padding: 15px 20px;
  border: none;
  cursor: pointer;
  width: 100%;
  opacity: 0.9;
}
.btn:hover {
  opacity: 1;
}

</style>
</head>
<body>

<div id="navbar">
  <a href="#default" id="logo"><span
style='font-size:100px;'>&#128246;</span>PERSONAL EXPENSE
TRACKER<br><br><i class="fa fa-line-chart" style="font-size:24px"></i>SET LIMIT</a>
  <div id="navbar-right">
    <a class="active" href="#index"><i class="fa fa-home"></i>Home</a>
    <a href="#contact"><i class="fa fa-envelope"></i>Notify</a>
  </div>
</div>

<div style="margin-top:210px;padding:15px 15px 2500px;font-size:30px">

</div>
<body>
<div class="content">

          {{status}}
```

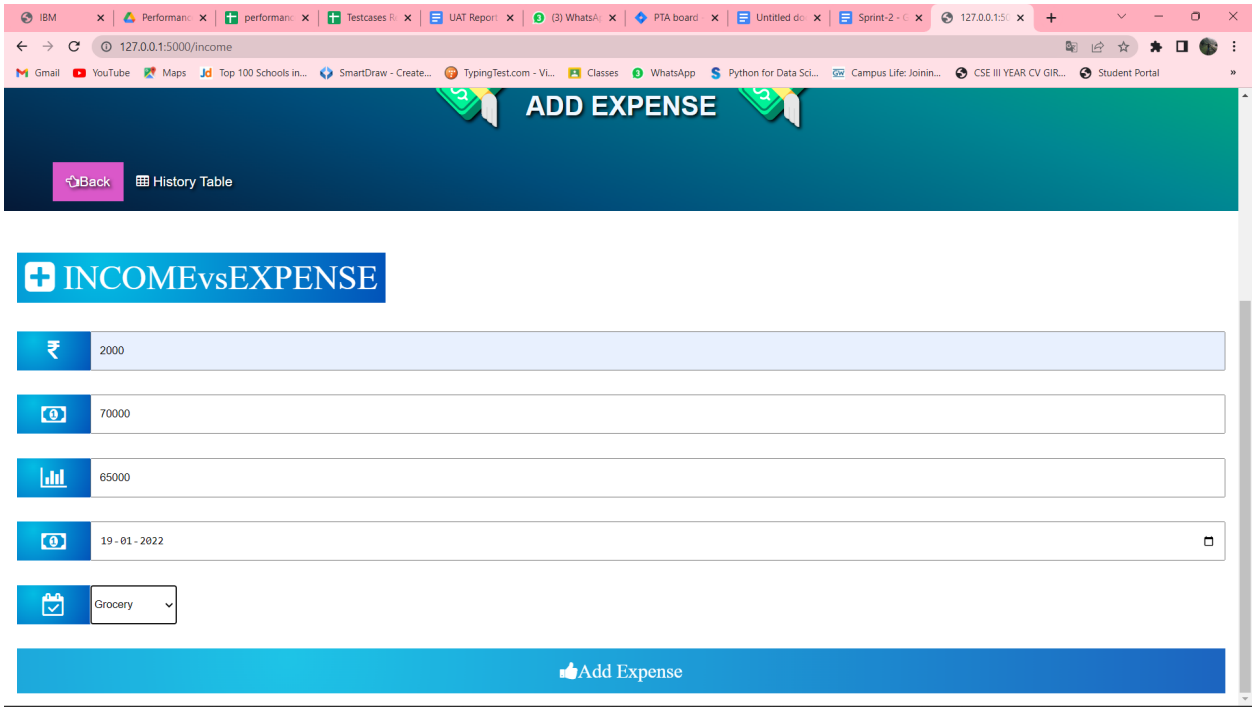
```
   <form action="{{ url_for('limit') }}" method="post"
style="max-width:500px;margin:auto">
            <h2>Current Limit  {{limit}}</h2>
            <h2><i class="fa fa-line-chart icon"> SET LIMIT</i></h2>

          <label for="limit">Limit</label> <br>
          <input name="limit" type="number"/>
             <button type="submit" class="btn"><i class="fa fa-thumbs-up"
style="font-size:24px"> set limit</i></button>
      </form>
</div>

</body>
</html>
```

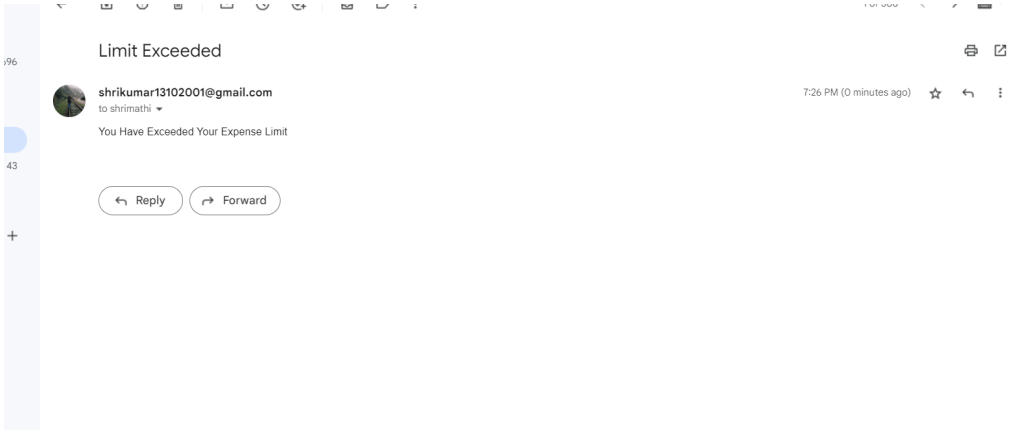
OUTPUT:





When the user try to add the expense which is higher than the limit amount, then the user will get an email alert.




Limit Exceeded
shrikumar13102001@gmail.com
to shrimathi
7:26 PM (0 minutes ago)
You Have Exceeded Your Expense Limit
Reply
Forward