SPRINT – 4

Date	**19/101/2022**
Team ID	**PNT2022TMID32597**
Project Name	**Personal Expense Tracker**

FRONTEND AND BACKED CONNECTION :

```
from __future__ import print_function
from datetime import datetime
from flask import Flask, request, json, jsonify
from flask_json import FlaskJSON, json_response
from flask_cors import CORS
import ibm_db
from template import *

# Initializing flask app
app = Flask(__name__)
jsonObj = FlaskJSON(app)
cors = CORS(app,resources={r'*':{'origins':'http://localhost:3000'}})

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=125f9f61-9715-46f9-9399-
c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30426;Security=S
SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=mpr37327;PWD=k5iRh2q3KBMmZr7
Z;","","")

# Route for seeing a data
@app.route('/login')
def login():
    email = request.args.get('email')
    password = request.args.get('password')
    sql = "SELECT * FROM login where email ='{}'".format(email)
    out = ibm_db.exec_immediate(conn, sql)
    document = ibm_db.fetch_assoc(out)
    if document == False :
        response = json_response(value=0)
    elif document['PASSWORD'] == password :
        response = json_response(value=1)
    else :
        response = json_response(value=2)
    return response

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

```
def register():
    if request.method == "POST":
        credentials = json.loads(request.data)
        sql = "INSERT INTO login
VALUES('{}','{}')".format(credentials['email'],credentials['password'])
        out = ibm_db.exec_immediate(conn, sql)
        sql = "INSERT INTO personal_info(email,name)
VALUES('{}','{}')".format(credentials['email'],credentials['name'])
        out = ibm_db.exec_immediate(conn, sql)
        response = json_response(200)
        return response

@app.route('/loadData')
def loadData():
    email = request.args.get('email')
    sql = "select sum(amount) as expense from expenses where email='{}' and
month(timestamp)=month(current_timestamp)".format(email)
    out = ibm_db.exec_immediate(conn, sql)
    document = ibm_db.fetch_assoc(out)

    totalExpense = document['EXPENSE']
    print(totalExpense)
    resultData = {
        'totalExpense' : document['EXPENSE'],
    }
    sql = "select walletlimit from personal_info where email =
'{}'".format(email)
    out = ibm_db.exec_immediate(conn, sql)
    document = ibm_db.fetch_assoc(out)

    # if resultData['totalExpense'] == None:
    #     resultData['balance'] = 0
    # else:
    #     resultData['balance'] = document['WALLETLIMIT'] - totalExpense
    sql = "select category, sum(amount) as expense from expenses where email='{}'
and month(timestamp)=month(current_timestamp) group by category".format(email)
    out = ibm_db.exec_immediate(conn, sql)
    document = ibm_db.fetch_assoc(out)
    piegraphData = []
    piegraphLabel = []
    while document != False:
        piegraphLabel.append(document["CATEGORY"])
        piegraphData.append(document["EXPENSE"])
        document = ibm_db.fetch_assoc(out)
    resultData['piegraphdata'] = piegraphData
```

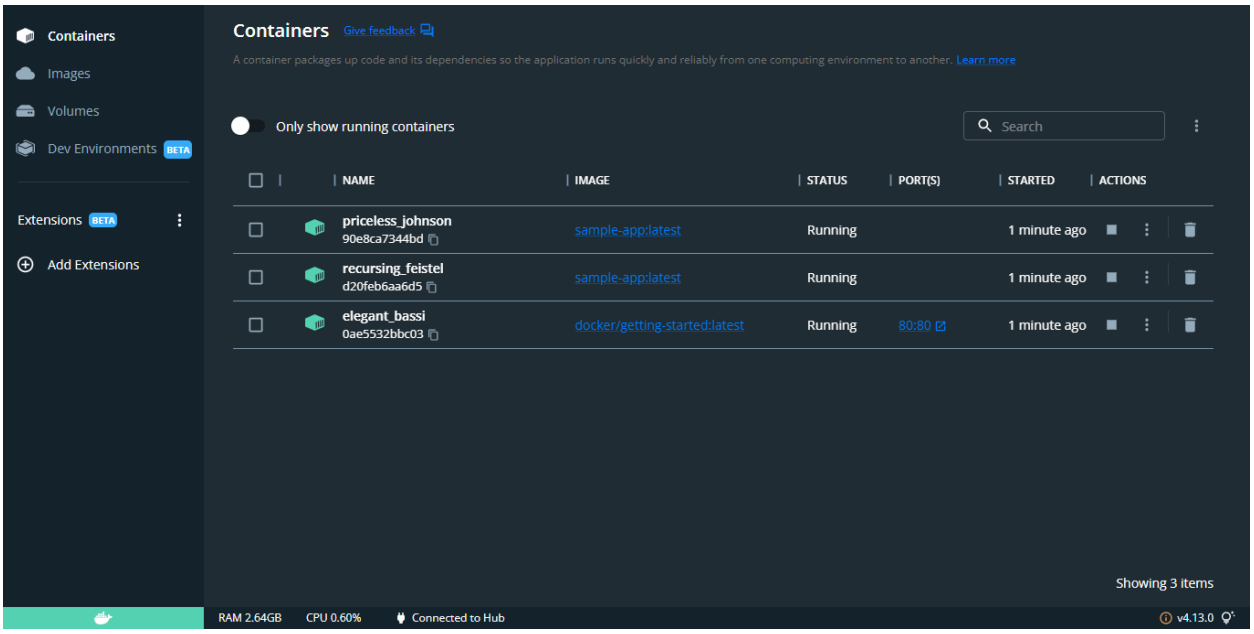
```
    resultData['piegraphlabel'] = piegraphLabel
    sql = "select dayname(cast(timestamp as date)) as day, sum(amount) as expense
from expenses,sysibm.sysdummy1 where email='{}' and
week(timestamp)=week(current_timestamp) group by cast(timestamp as
date)".format(email)
    out = ibm_db.exec_immediate(conn, sql)
    document = ibm_db.fetch_assoc(out)
    bargraphData = []
    bargraphLabel =[]
    while document != False:
        bargraphLabel.append(document['DAY'])
        bargraphData.append(document["EXPENSE"])
        document = ibm_db.fetch_assoc(out)
    resultData['bargraphdata'] = bargraphData
    resultData['bargraphlabel'] = bargraphLabel
    sql = "select sum(amount) as expense from expenses where email='{}' and
date(timestamp)=date(current_timestamp)".format(email)
    out = ibm_db.exec_immediate(conn, sql)
    document = ibm_db.fetch_assoc(out)
    resultData['dailyExpense']=document['EXPENSE']
    response = json_response(resultData=resultData)
    return response

@app.route('/addExpense', methods=['POST'])
def addExpense():
    if request.method == "POST":
        expense = json.loads(request.data)
        sql = "INSERT INTO expenses(email,category,amount,timestamp)
VALUES('{}','{}',{},'{}')".format(expense['email'],expense['category'],expense['a
mount'], datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
        out = ibm_db.exec_immediate(conn, sql)
        response = json_response(200)
        return response

# @app.route('/limitExceed')
# def limitExceed():
#     email = request.args.get('email')
#     SendDynamic()
#     return json_response(200)

@app.route('/personalData')
def personalData():
    email = request.args.get('email')
    sql = "select * from  personal_info where email='{}'".format(email)
    out = ibm_db.exec_immediate(conn, sql)
```

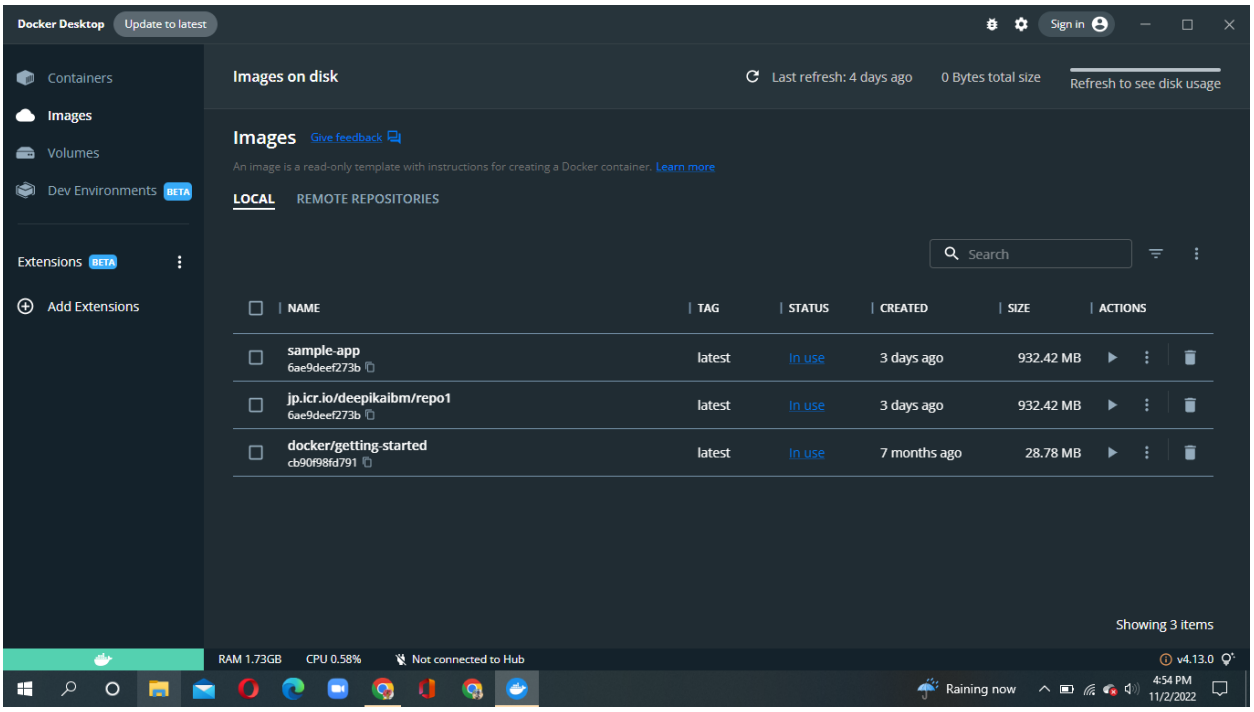
```
    document = ibm_db.fetch_assoc(out)
    resultData = {
        'name' : document['NAME'],
        'email' : email,
        'walletlimit':document['WALLETLIMIT'],
        'gender': document['GENDER'],
        'location': document['LOCATION'],
        'phone' : document['PHONE'],

    }
    sql = "select * from login where email='{}'".format(email)
    out = ibm_db.exec_immediate(conn, sql)
    document = ibm_db.fetch_assoc(out)
    resultData['password'] = document['PASSWORD']
    response = json_response(resultData=resultData)
    return response

@app.route('/updateProfile',methods=['POST'])
def updateProfile():
    if request.method == "POST":
        credentials = json.loads(request.data)
        sql = "UPDATE login SET password='{}' where
email='{}'".format(credentials['password'],credentials['email'])
        out = ibm_db.exec_immediate(conn, sql)
        print("Cred:",credentials)
        sql = "UPDATE personal_info SET name='{}', walletlimit={}, gender='{}',
location='{}', phone='{}' where email='{}'
".format(credentials['name'],credentials['walletlimit'],credentials['gender'],cre
dentials['location'],credentials['phone'], credentials['email'])
        out = ibm_db.exec_immediate(conn, sql)
        response = json_response(200)
        return response
    print("hai out")


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

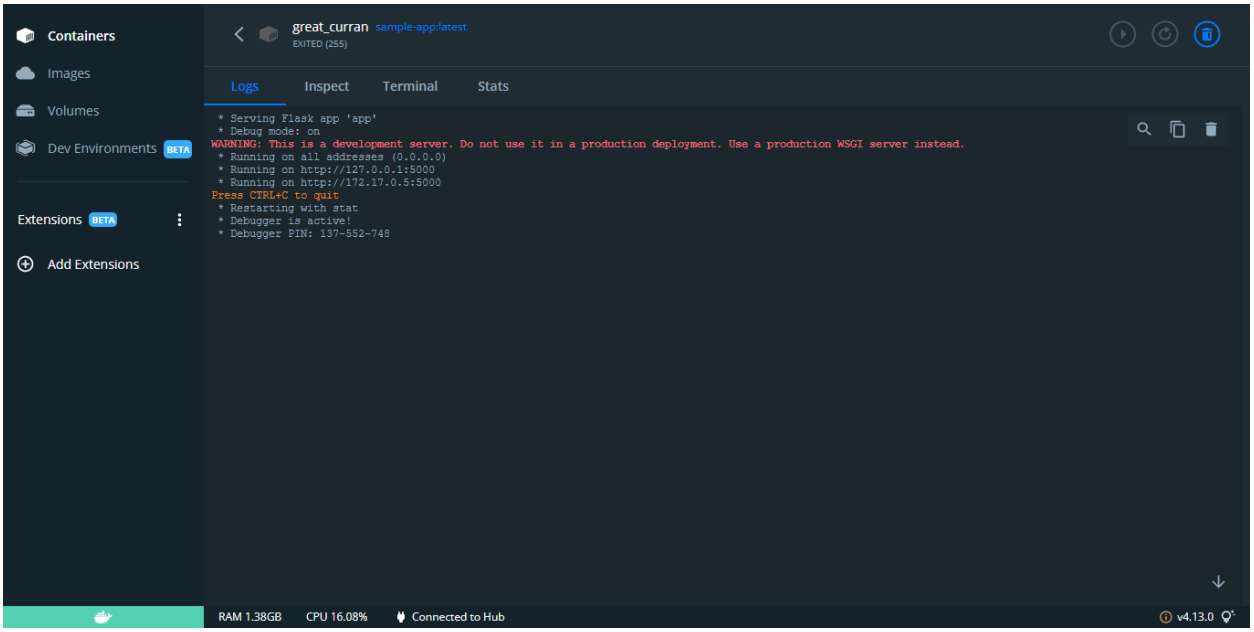
DOCKER :

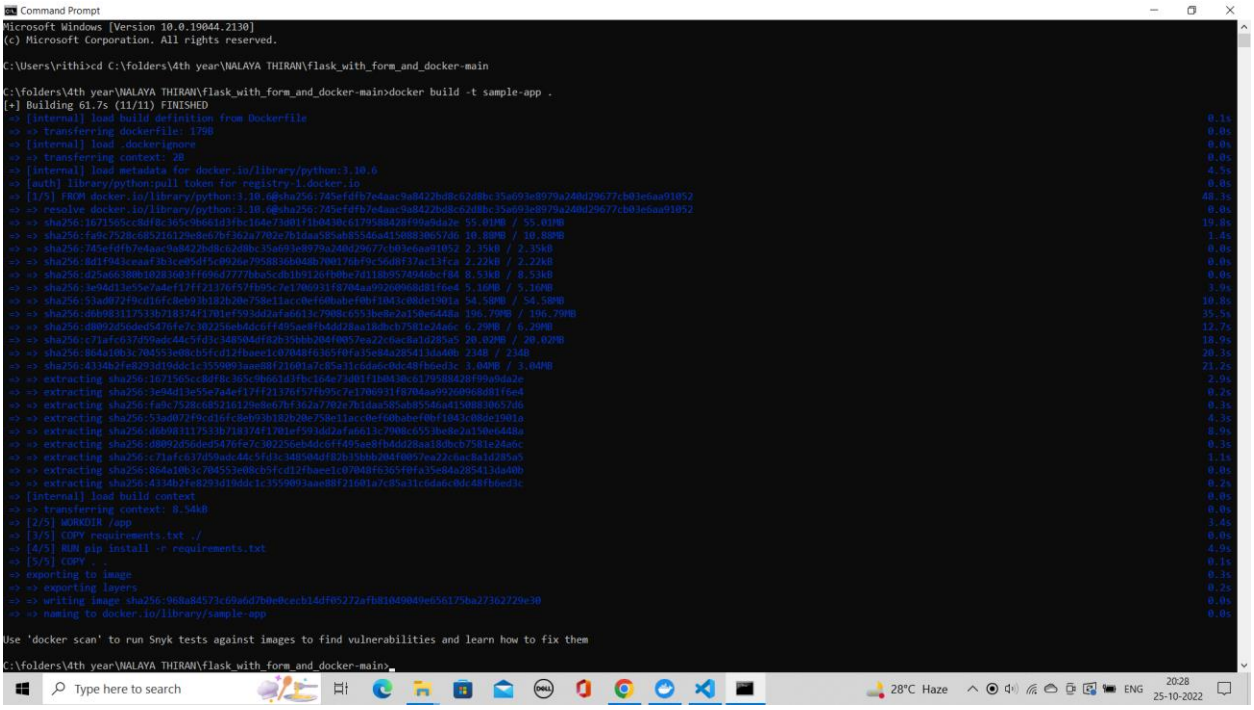
```
FROM node:16.6.2-alpine as build-step

WORKDIR /app
ENV PATH /app/node_modules/.bin:$PATH
COPY package.json ./
COPY package-lock.json ./
COPY nginx.conf ./
COPY ./src ./src
COPY ./public ./public
RUN npm install

# RUN npm run build

# FROM nginx:1.17.10-alpine
# COPY --from=build-step /app/build/ /usr/share/nginx/html
# COPY --from=build-step /app/nginx.conf /etc/nginx/conf.d/default.conf

EXPOSE 3000

CMD ["npm", "start"]
```

DOCKER SCREENSHOTS :




Docker Desktop
Update to latest
Sign in
Containers
Images
Volumes
Dev Environments BETA
Extensions BETA
Add Extensions
Images on disk
Last refresh: 4 days ago
0 Bytes total size
Refresh to see disk usage
Images
Give feedback
An image is a read-only template with instructions for creating a Docker container. Learn more
LOCAL
REMOTE REPOSITORIES
Search
NAME
TAG
STATUS
CREATED
SIZE
ACTIONS
sample-app
6ae9deef273b
latest
In use
3 days ago
932.42 MB
jp.icr.io/deepikaibm/repo1
6ae9deef273b
latest
In use
3 days ago
932.42 MB
docker/getting-started
cb90f98fd791
latest
In use
7 months ago
28.78 MB
Showing 3 items
RAM 1.73GB
CPU 0.58%
Not connected to Hub
v4.13.0
Raining now
4:54 PM
11/2/2022


Containers
Images
Volumes
Dev Environments BETA
Extensions BETA
Add Extensions
great_curran
sample-app:latest
EXITED (255)
Logs
Inspect
Terminal
Stats
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.5:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 137-552-748
RAM 1.38GB
CPU 16.08%
Connected to Hub
v4.13.0


Command Prompt
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.
C:\Users\rithi>cd C:\folders\4th year\NALAYA THIRAN\flask_with_form_and_docker-main
C:\folders\4th year\NALAYA THIRAN\flask_with_form_and_docker-main>docker build -t sample-app .
[+] Building 61.7s (11/11) FINISHED
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 179B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/python:3.10.6 4.5s
=> [auth] library/python:pull token for registry-1.docker.io 0.0s
=> [1/5] FROM docker.io/library/python:3.10.6@sha256:745efdfb7e4aac9a8422bd8c62d8bc35a693e8979a240d29677cb03e6aa91052 48.3s
=> => resolve docker.io/library/python:3.10.6@sha256:745efdfb7e4aac9a8422bd8c62d8bc35a693e8979a240d29677cb03e6aa91052 0.0s
=> => sha256:1671565cc8df8c365c9b661d3fbc164e73d01f1b0430c6179588428f99a9da2e 55.01MB / 55.01MB 19.0s
=> => sha256:fa9c7528c685216129e8e67bf362a7702e7b1daa585ab85546a4150883065 7d6 10.88MB / 10.88MB 1.4s
=> => sha256:745efdfb7e4aac9a8422bd8c62d8bc35a693e8979a240d29677cb03e6aa91052 2.35kB / 2.35kB 0.0s
=> => sha256:8d1f943ceaaf3b3ce05df5c0926e7958836b048b700176bf9c56d8f37ac13fca 2.22kB / 2.22kB 0.0s
=> => sha256:d25a66380b10283603ff696d7777bba5cdb1b9126fb0be7d118b9574946bcf84 8.53kB / 8.53kB 0.0s
=> => sha256:3e94d13e55e7a4ef17ff21376f57fb95c7e1706931f8704aa99260968d81f6e4 5.16MB / 5.16MB 3.9s
=> => sha256:53ad072f9cd16fc8eb93b182b20e758e11acc0ef60babef0bf1043c08de1901a 54.58MB / 54.58MB 10.8s
=> => sha256:d6b983117533b718374f1701ef593dd2afa6613c7008c6553be8e2a150e6448a 196.79MB / 196.79MB 35.5s
=> => sha256:d8092d56ded5476fe7c302256eb4dc6ff495ee8fb4dd28aa18dbcb7581e24a6c 6.29MB / 6.29MB 12.7s
=> => sha256:c71afc637d59adc44c5fd3c148504df82b35bbb204f0057ea22c6ac8a1d285a5 20.02MB / 20.02MB 18.9s
=> => sha256:864a10b3c704553e08cb5fcd12fbaee1c07048f6365f0fa35e84a285413da40b 234B / 234B 20.3s
=> => sha256:4334b2fe8293d19ddc1c3559093aae88f21601a7c85a31c6da6c0dc48fb6ed3c 3.04MB / 3.04MB 21.2s
=> => extracting sha256:1671565cc8df8c365c9b661d3fbc164e73d01f1b0430c6179588428f99a9da2e 2.9s
=> => extracting sha256:3e94d13e55e7a4ef17ff21376f57fb95c7e1706931f8704aa99260968d81f6e4 0.2s
=> => extracting sha256:fa9c7528c685216129e8e67bf362a7702e7b1daa585ab85546a4150883065 7d6 0.3s
=> => extracting sha256:53ad072f9cd16fc8eb93b182b20e758e11acc0ef60babef0bf1043c08de1901a 4.3s
=> => extracting sha256:d6b983117533b718374f1701ef593dd2afa6613c7008c6553be8e2a150e6448a 8.9s
=> => extracting sha256:d8092d56ded5476fe7c302256eb4dc6ff495ee8fb4dd28aa18dbcb7581e24a6c 0.3s
=> => extracting sha256:c71afc637d59adc44c5fd3c148504df82b35bbb204f0057ea22c6ac8a1d285a5 1.1s
=> => extracting sha256:864a10b3c704553e08cb5fcd12fbaee1c07048f6365f0fa35e84a285413da40b 0.0s
=> => extracting sha256:4334b2fe8293d19ddc1c3559093aae88f21601a7c85a31c6da6c0dc48fb6ed3c 0.2s
=> [internal] load build context 0.0s
=> => transferring context: 8.54kB 0.0s
=> [2/5] WORKDIR /app 3.4s
=> [3/5] COPY requirements.txt ./ 0.0s
=> [4/5] RUN pip install -r requirements.txt 4.9s
=> [5/5] COPY . . 0.1s
=> exporting to image 0.3s
=> => exporting layers 0.2s
=> => writing image sha256:968a84571c69a6d7b0e0cecb14df05272afb81049049e656175ba27362729e30 0.0s
=> => naming to docker.io/library/sample-app 0.0s
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
C:\folders\4th year\NALAYA THIRAN\flask_with_form_and_docker-main>
Type here to search
28°C Haze
ENG
20:28
25-10-2022