TITLE: Mentorship Program -SMART ASSIGNMENT ENGINE

1.Introduction:

Overview: This document serves as a comprehensive guide to represent the mentorship program between the teachers, students, administrators. This may have an application where the student can access the assignments and upload the work done by the student.

Purpose: The application allows for secure file uploads, ensuring the integrity and confidentiality of submitted work. The system provides notifications and reminders to students about upcoming assignment due dates and any updates or clarifications from teachers. Smart Assignment aims to enhance the assignment management process, promote collaboration, and improve educational outcomes.

2.Literature Survey:

Existing Problem:

The inefficiency and lack of collaboration in the traditional assignment management process. In many educational institutions, the assignment workflow involves manual creation, distribution, submission, grading, and feedback processes. The manual assignment creation and distribution process can be time-consuming for teachers. They have to prepare assignments, print them, distribute them in class, and later collect physical copies or emails from students. Students might miss assignment details due to the lack of a centralized platform. Without a single source of information, students may struggle to keep track of assignment deadlines and requirements.

Proposed Solution:

Smart Assignment is a web-based application designed to streamline the process of assignment management and enhance collaboration among students, teachers, and administrators. This project aims to leverage web development technologies to create an intelligent platform that automates assignment creation, submission, grading, and feedback, thereby improving the efficiency and effectiveness of the assignment process. Students can access the platform to view and submit assignments. The application allows for secure file uploads, ensuring the integrity and confidentiality of submitted work.

3.Theoretical Analysis:

Block Diagram:
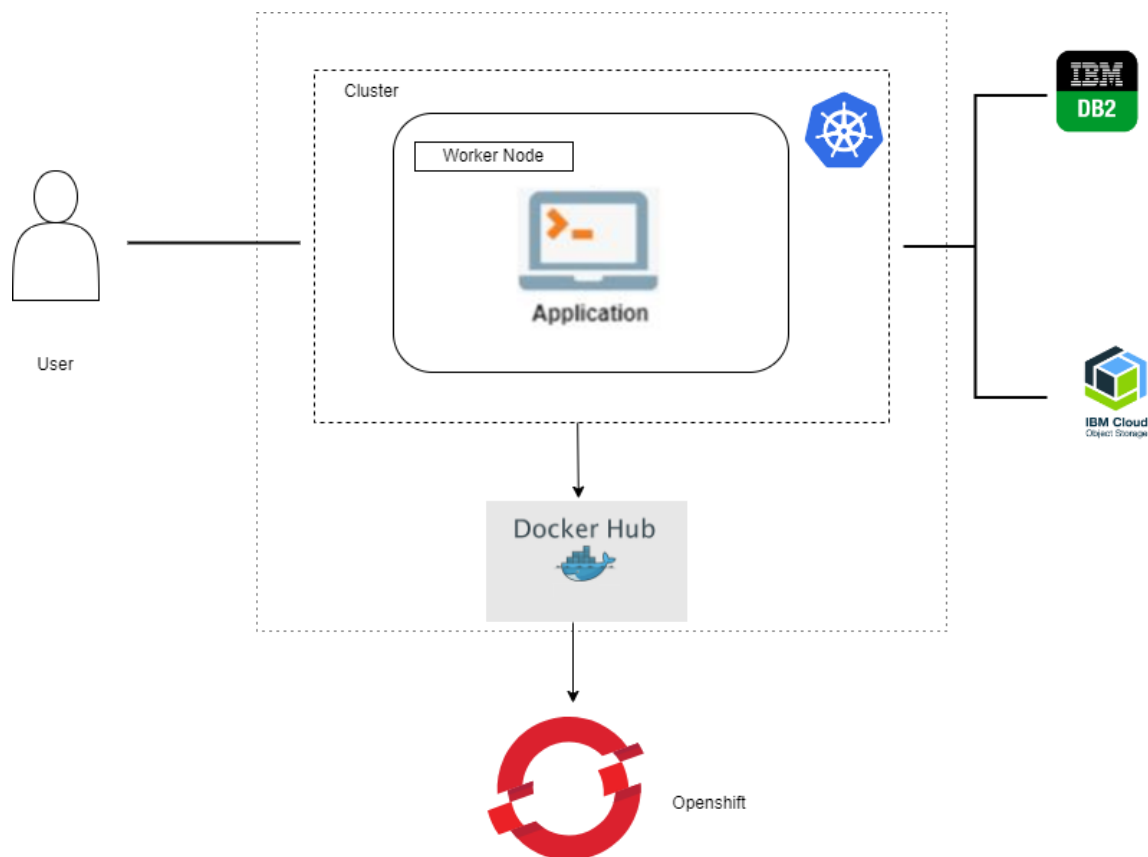
Software Requirements: IBM Cloud, HTML, CSS, GitHub, IBM Cloud Object Storage and Databases, Docker.

4.Experimental Investigation:

Smart Assignment Engine is the solution as a web application is designed to streamline assignment management and enhance collaboration between students, teachers and administrators. This project aims to leverage web development technologies to create an intelligent platform that automates assignment creation, submission, grading and feedback. Throughout the development process we found that this will be used to faculty to given assignment notifications through it and students can submit their assignments through this web software.
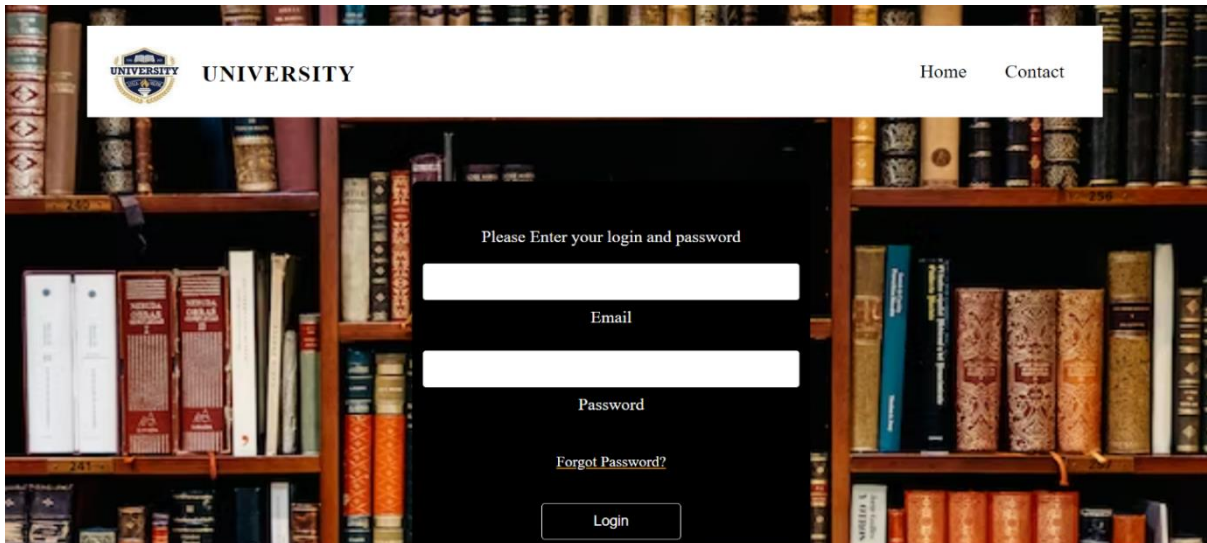
5.Flow Chart:
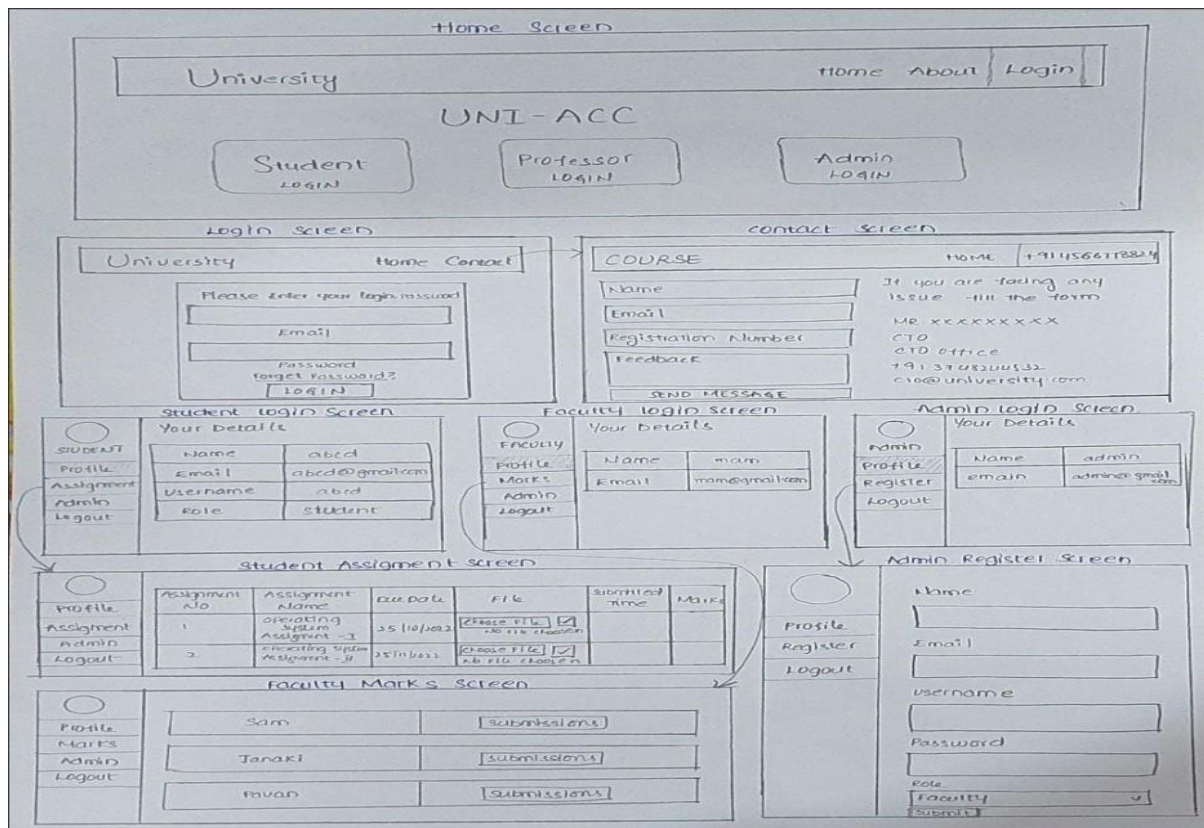
## 6.Result:
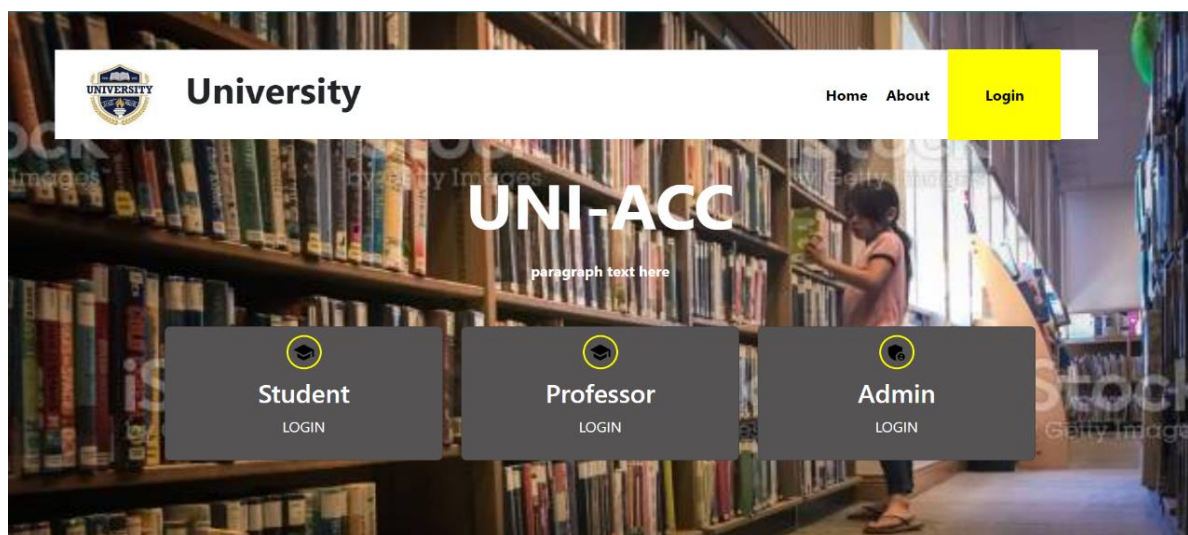
Screen1: Home Page

- In above Screen1 we have the Home page of the Mentorship Program Smart Assignment Engine.

- In that we include the logins regarding to category and we also provide the login page and contact page which can be redirect.
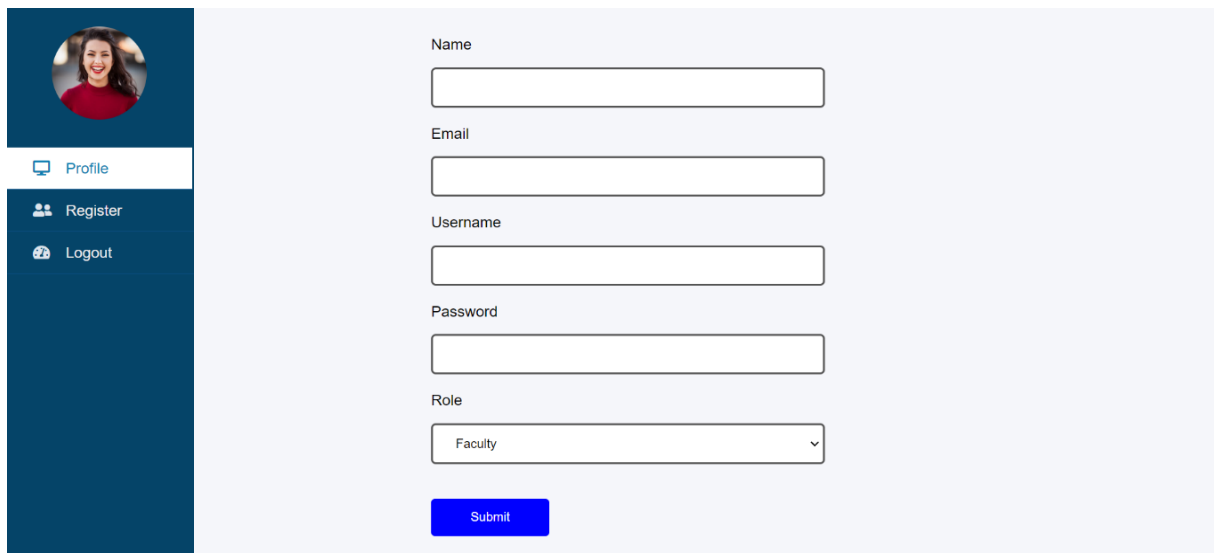

Screen2: Login Page

- In this page the Student, Faculty and Admin can login into the application to access the resources provided
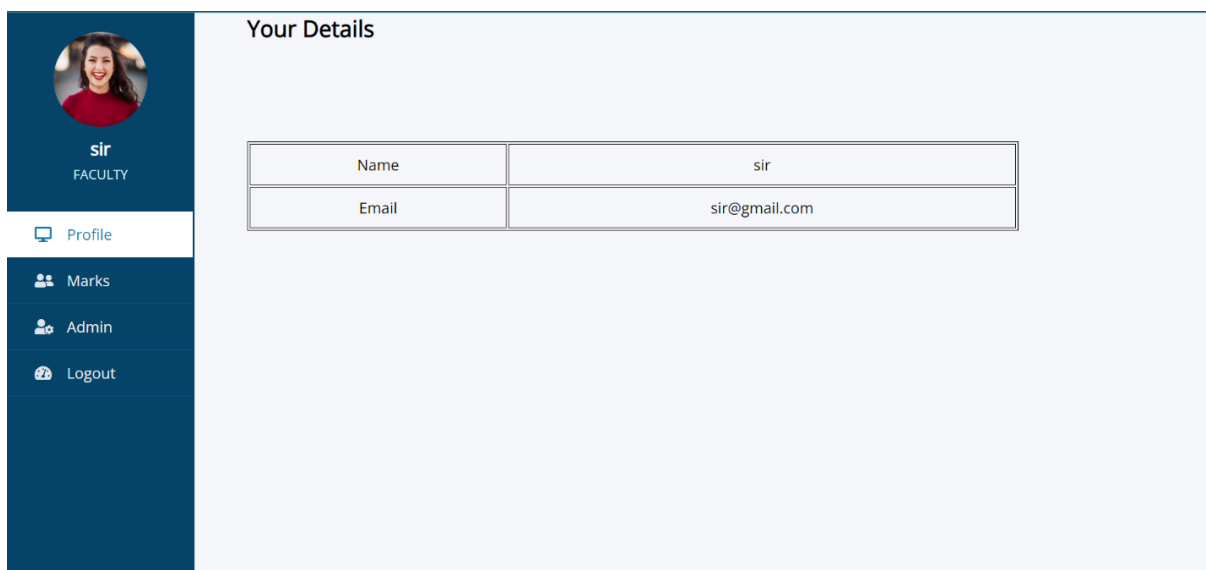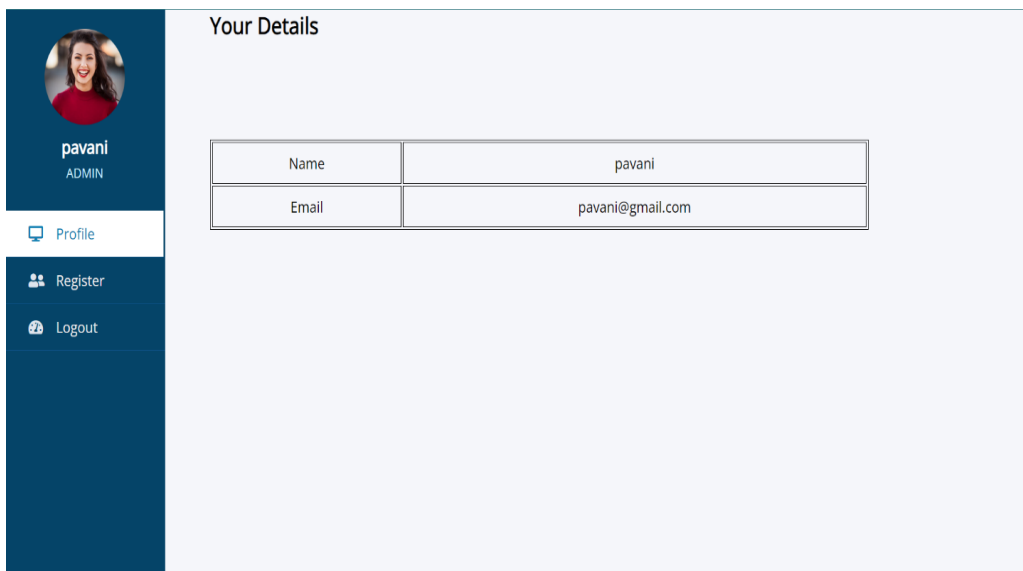

Screen3: Admin Profile Page

- In this page the Admin profile is viewed and actions performed by the admin.
- The admin can register the new users from his/her profile.

Screen4: Admin Register Page.

- In this Admin Register page the admin can register Students, Faculty and other Admins too.
- And the registered data will be stored in the same table where all the information is stored.

Screen5: Faculty Profile Page

- .This is the Faculty Profile page where faculty can view his/her details and also view the marks of the students who upload the work in the application

Screen6: Marks page

- In this page the Faculty can view the submissions received by the students and can a lot the marks.

Screen7: Contact Page

- When the Faculty clicks on the Admin page it redirect to the contact where any one can address the issues and also may provide the feed back about the application.

Screen8: Student Page

- In this page the Student can view his details and also upload the assignments. Here he/her can give the feedback form by clicking the Admin route.

Screen9: Student Assignment Submission page

- In this the students can submit their assignments and also view the submitted time and marks. Here the marks can shown when the faculty the update the marks based their performance.

Advantages:

- Student can submit their assignments to their faculty through smart assignment engine.
- Student can view their marks gained for their assignment which was submitted through online.
- Faculty can give assignments to the students through online.
- The student data for the respected faculty can be known my faculty login.
- Faculty can view the students submitted assignments and give marks to students.

Disadvantages:

- There is no separate access to faculty to upload assignments.
- No separate access to multiple subjects to give assignments.

Applications:

- This application can be used in colleges for the submissions of assignments.
- This can also used in schools to create the intelligent platform that automates assignment creation.

Conclusion:

The overall conclusion for the project is the web development software which will enhances the assignment management process, and make collaboration between students and teachers.

This will gives separate user interface based on the role the user registered which is very interactive to users.

Future Scope:

As the web development is done using IBM cloud, this can enhance in many educational areas where students and faculty are collaborated. We can develop further improvements to give access to multiple subjects and manage assignment process.

Appendix:

Source code:

```
from flask import Flask, render_template, request, session
import ibm_db
import ibm_boto3
```

```python
from ibm_botocore.client import Config, ClientError
import os
import re
import random
import string
import datetime
import requests

app = Flask(__name__)
app.secret_key = 'a'
conn       =       ibm_db.connect("DATABASE=bludb;HOSTNAME=1bbf73c5-d84a-4bb0-85b9-
ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=32286;SECURITY=SSL;
SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=wrj97286;PWD=yr9TGZdqnKAYr7Tx",",")
url = "https://rapidprod-sendgrid-v1.p.rapidapi.com/mail/send"
@app.route("/")
def index():
    return render_template("index.html")


@app.route("/index")
def index2():
    return render_template("index.html")


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


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


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


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


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


@app.route("/login", methods=['POST','GET'])
def loginentered():
    global Userid
    global Username
    msg = ''
```

```python
    if request.method == "POST":
        email = str(request.form['email'])
        print(email)
        password = request.form["password"]
        sql = "SELECT * FROM REGISTER WHERE EMAIL=? AND PASSWORD=?"  # from db2 sql
table
        stmt = ibm_db.prepare(conn, sql)
        # this username & password is should be same as db-2 details & order also
        ibm_db.bind_param(stmt, 1, email)
        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['EMAIL']
            Userid = account['EMAIL']
            session['email'] = account['EMAIL']
            Username = account['USERNAME']
            session['Username'] = account['USERNAME']
            Name = account['NAME']
            msg = "logged in successfully !"
            sql = "SELECT ROLE FROM register where email = ?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, email)
            ibm_db.execute(stmt)
            r = ibm_db.fetch_assoc(stmt)
            print(r)
            if r['ROLE'] == 1:
                print("STUDENT")
                return render_template("studentprofile.html", msg=msg, user=email, name = Name, role=
"STUDENT", username=Username, email = email)
            elif r['ROLE'] == 2:
                print("FACULTY")
                return render_template("facultyprofile.html", msg=msg, user=email, name = Name, role=
"FACULTY", username=Username, email = email)
            else:
                return render_template('adminprofile.html', msg=msg, user=email, name = Name, role=
"ADMIN", username=Username, email = email)
        else:
            msg = "Incorrect Email/password"

        return render_template("login.html", msg=msg)
    else:
        return render_template("login.html")
@app.route("/studentsubmit", methods=['POST','GET'])
def sassignment():
```

```python
#u = Username.strip()
u=session.get('Username')
subtime = []
ma = []
sql = "SELECT CSUBMITTIME, MARKS from SUBMIT WHERE STUDENTNAME = ? "
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, u)
ibm_db.execute(stmt)
st = ibm_db.fetch_tuple(stmt)
while st !=False:
    subtime.append(st[0])
    ma.append(st[1])
    st = ibm_db.fetch_tuple(stmt)
print(subtime)
print(ma)
if request.method=="POST":
    for x in range (1,5):
        x = str(x)
        y = str("file"+x)
        print(type(y))
        f=request.files[ y ]
        print(f)
        print(f.filename)
        if f.filename != '':
            basepath=os.path.dirname(__file__) #getting the current path i.e where app.py is present
            #print("current path",basepath)
            filepath=os.path.join(basepath,'uploads',u+x+".pdf") #from anywhere in the system we can give
image but we want that image later  to process so we are saving it to uploads folder for reusing
            #print("upload folder is",filepath)
            f.save(filepath)
            # connecting with cloud object storage

            COS_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud"
            COS_API_KEY_ID = "feQ_YXP6d0HktdnQwEH6YFh545ZpgacGGQJkguFhthrb"
            COS_INSTANCE_CRN                 =                "crn:v1:bluemix:public:cloud-object-
storage:global:a/2396b07efb5e497894432383c651d668:2883dbef-d638-4e0a-922e-2ad686a7ad08::"
            cos                                                                                  =
ibm_boto3.resource("s3",ibm_api_key_id=COS_API_KEY_ID,ibm_service_instance_id=COS_INST
ANCE_CRN, config=Config(signature_version="oauth"),endpoint_url=COS_ENDPOINT)
            cos.meta.client.upload_file(Filename=          filepath,Bucket='studentassignmentsb',Key=
u+x+".pdf")
            msg = "Uploding Successful"
            ts = datetime.datetime.now()
            t = ts.strftime("%Y-%m-%d %H:%M:%S")
            sql1   =   "SELECT   *   FROM   SUBMIT   WHERE   STUDENTNAME   =   ?   AND
ASSIGNMENTNUM = ?"
            stmt = ibm_db.prepare(conn, sql1)
```

```python
        ibm_db.bind_param(stmt, 1, u)
        ibm_db.bind_param(stmt, 2, x)
        ibm_db.execute(stmt)
        acc = ibm_db.fetch_assoc(stmt)
        print(acc)
        if acc == False:
            sql = "INSERT into SUBMIT (STUDENTNAME, ASSIGNMENTNUM, CSUBMITTIME)
values (?,?,?)"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, u)
            ibm_db.bind_param(stmt, 2, x)
            ibm_db.bind_param(stmt, 3, t)
            ibm_db.execute(stmt)
        else:
            sql = "UPDATE SUBMIT SET CSUBMITTIME = ? WHERE STUDENTNAME = ? and
ASSIGNMENTNUM = ?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, t)
            ibm_db.bind_param(stmt, 2, u)
            ibm_db.bind_param(stmt, 3, x)
            ibm_db.execute(stmt)
        return render_template("studentsubmit.html", msg=msg, datetime=subtime, Marks=ma)
    return     render_template("studentsubmit.html",     datetime=subtime,     Marks=ma)     def
marksassign(stdname):
    # Initialize variables
    global u
    global g
    global file
    global name
    name = stdname
    da = []
    file = set()
    # COS configuration
    COS_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud"
    COS_API_KEY_ID = "feQ_YXP6d0HktdnQwEH6YFh545ZpgacGGQJkguFhthrb"
    COS_INSTANCE_CRN            =            "crn:v1:bluemix:public:cloud-object-
storage:global:a/2396b07efb5e497894432383c651d668:2883dbef-d638-4e0a-922e-2ad686a7ad08::"
    # Initialize COS client
    cos = ibm_boto3.client("s3",
            ibm_api_key_id=COS_API_KEY_ID,
            ibm_service_instance_id=COS_INSTANCE_CRN,
            config=Config(signature_version="oauth"), endpoint_url=COS_ENDPOINT)
    output = cos.list_objects(Bucket="studentassignmentsb")
    # Extract object keys from the output
    object_keys = [item['Key'] for item in output.get('Contents', [])]
    # Filter object keys based on student name
    for key in object_keys:
```

```python
        if key.startswith(stdname):
            file.add(key)
    g = len(file)
    # Database query for submission times
    sql = "SELECT CSUBMITTIME FROM SUBMIT WHERE STUDENTNAME = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt , 1, stdname)
    ibm_db.execute(stmt)
    # Fetch submission times and store in da list
    while True:
        m = ibm_db.fetch_tuple(stmt)
        if not m:
            break
        da.append(m[0])
    # Render template with data
    return render_template("facultymarks.html", file=file, g=g, marks=0, datetime=da)
@app.route("/logout")
def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    return render_template("logout.html")
if __name__ == "__main__":
    app.run(debug=True,port=5000, host="0.0.0.0")
```

Demo Link:

https://drive.google.com/file/d/149GFeOT5BdBFDUF7PO0IhB-6Q6utNOhm/view?usp=drivesdk

Docker Link:

we use docker and red hat, open shift to containerize the application

link:-https://studentassignment-moturisireesha-dev.apps.sandbox-m2.ll9k.p1.openshiftapps.com/