SPRINT - 3

Date	14.11.2022
Team ID	PNT2022TMID50144
Project	Skill and Job Recommender

Sendgrid mail integration:

```
import os
import json

from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import *


# NOTE: you will need move this file to the root
# directory of this project to execute properly.


def build_hello_email():
    ## Send a Single Email to a Single Recipient

    message = Mail(from_email=From('from@example.com', 'Example From Name'),
                   to_emails=To('to@example.com', 'Example To Name'),
                   subject=Subject('Sending with SendGrid is Fun'),
                   plain_text_content=PlainTextContent('and easy to do anywhere, even
with Python'),
                   html_content=HtmlContent('<strong>and easy to do anywhere, even
with Python</strong>'))

    try:
        print(json.dumps(message.get(), sort_keys=True, indent=4))
        return message.get()

    except SendGridException as e:
        print(e.message)

    mock_personalization = Personalization()
    personalization_dict = get_mock_personalization_dict()

    for cc_addr in personalization_dict['cc_list']:
        mock_personalization.add_to(cc_addr)

    for bcc_addr in personalization_dict['bcc_list']:
        mock_personalization.add_bcc(bcc_addr)
```

```
    for header in personalization_dict['headers']:
        mock_personalization.add_header(header)

    for substitution in personalization_dict['substitutions']:
        mock_personalization.add_substitution(substitution)

    for arg in personalization_dict['custom_args']:
        mock_personalization.add_custom_arg(arg)

    mock_personalization.subject = personalization_dict['subject']
    mock_personalization.send_at = personalization_dict['send_at']

    message.add_personalization(mock_personalization)

    return message

def get_mock_personalization_dict():
    """Get a dict of personalization mock."""
    mock_pers = dict()

    mock_pers['to_list'] = [To("test1@example.com",
                               "Example User"),
                            To("test2@example.com",
                               "Example User")]

    mock_pers['cc_list'] = [To("test3@example.com",
                               "Example User"),
                            To("test4@example.com",
                               "Example User")]

    mock_pers['bcc_list'] = [To("test5@example.com"),
                             To("test6@example.com")]

    mock_pers['subject'] = ("Hello World from the Personalized "
                            "SendGrid Python Library")

    mock_pers['headers'] = [Header("X-Test", "test"),
                            Header("X-Mock", "true")]

    mock_pers['substitutions'] = [Substitution("%name%", "Example User"),
                                  Substitution("%city%", "Denver")]

    mock_pers['custom_args'] = [CustomArg("user_id", "343"),
                                CustomArg("type", "marketing")]

    mock_pers['send_at'] = 1443636843
    return mock_pers

def build_multiple_emails_personalized():
    # Note that the domain for all From email addresses must match
```

```
    message = Mail(from_email=From('from@example.com', 'Example From Name'),
                   subject=Subject('Sending with SendGrid is Fun'),
                   plain_text_content=PlainTextContent('and easy to do anywhere, even
with Python'),
                   html_content=HtmlContent('<strong>and easy to do anywhere, even
with Python</strong>'))

    mock_personalization = Personalization()
    mock_personalization.add_to(To('test@example.com', 'Example User 1'))
    mock_personalization.add_cc(Cc('test1@example.com', 'Example User 2'))
    message.add_personalization(mock_personalization)

    mock_personalization_2 = Personalization()
    mock_personalization_2.add_to(To('test2@example.com', 'Example User 3'))
    mock_personalization_2.set_from(From('from@example.com', 'Example From Name
2'))
    mock_personalization_2.add_bcc(Bcc('test3@example.com', 'Example User 4'))
    message.add_personalization(mock_personalization_2)

    try:
        print(json.dumps(message.get(), sort_keys=True, indent=4))
        return message.get()

    except SendGridException as e:
        print(e.message)

    return message

def build_attachment1():
    """Build attachment mock. Make sure your content is base64 encoded before
passing into attachment.content.
    Another example: https://github.com/sendgrid/sendgrid-
python/blob/HEAD/use_cases/attachment.md"""

    attachment = Attachment()
    attachment.file_content = ("TG9yZW0gaXBzdW0gZG9sb3Igc2l0IGFtZXQsIGNvbnNl"
                               "Y3RldHVyIGFkaXBpc2NpbmcgZWxpdC4gQ3JhcyBwdW12")
    attachment.file_type = "application/pdf"
    attachment.file_name = "balance_001.pdf"
    attachment.disposition = "attachment"
    attachment.content_id = "Balance Sheet"
    return attachment


def build_attachment2():
    """Build attachment mock."""
    attachment = Attachment()
    attachment.file_content = "BwdW"
    attachment.file_type = "image/png"
    attachment.file_name = "banner.png"
    attachment.disposition = "inline"
    attachment.content_id = "Banner"
```

```
    return attachment

def build_kitchen_sink():
    """All settings set"""
    from sendgrid.helpers.mail import (
        Mail, From, To, Cc, Bcc, Subject, PlainTextContent,
        HtmlContent, SendGridException, Substitution,
        Header, CustomArg, SendAt, Content, MimeType, Attachment,
        FileName, FileContent, FileType, Disposition, ContentId,
        TemplateId, Section, ReplyTo, Category, BatchId, Asm,
        GroupId, GroupsToDisplay, IpPoolName, MailSettings,
        BccSettings, BccSettingsEmail, BypassListManagement,
        FooterSettings, FooterText, FooterHtml, SandBoxMode,
        SpamCheck, SpamThreshold, SpamUrl, TrackingSettings,
        ClickTracking, SubscriptionTracking, SubscriptionText,
        SubscriptionHtml, SubscriptionSubstitutionTag,
        OpenTracking, OpenTrackingSubstitutionTag, Ganalytics,
        UtmSource, UtmMedium, UtmTerm, UtmContent, UtmCampaign)
    import time
    import datetime

    message = Mail()

    # Define Personalizations

    message.to = To('test1@sendgrid.com', 'Example User1', p=0)
    message.to = [
        To('test2@sendgrid.com', 'Example User2', p=0),
        To('test3@sendgrid.com', 'Example User3', p=0)
    ]

    message.cc = Cc('test4@example.com', 'Example User4', p=0)
    message.cc = [
        Cc('test5@example.com', 'Example User5', p=0),
        Cc('test6@example.com', 'Example User6', p=0)
    ]

    message.bcc = Bcc('test7@example.com', 'Example User7', p=0)
    message.bcc = [
        Bcc('test8@example.com', 'Example User8', p=0),
        Bcc('test9@example.com', 'Example User9', p=0)
    ]

    message.subject = Subject('Sending with SendGrid is Fun 0', p=0)

    message.header = Header('X-Test1', 'Test1', p=0)
    message.header = Header('X-Test2', 'Test2', p=0)
    message.header = [
        Header('X-Test3', 'Test3', p=0),
        Header('X-Test4', 'Test4', p=0)
    ]
```

```
    message.substitution = Substitution('%name1%', 'Example Name 1', p=0)
    message.substitution = Substitution('%city1%', 'Example City 1', p=0)
    message.substitution = [
        Substitution('%name2%', 'Example Name 2', p=0),
        Substitution('%city2%', 'Example City 2', p=0)
    ]

    message.custom_arg = CustomArg('marketing1', 'true', p=0)
    message.custom_arg = CustomArg('transactional1', 'false', p=0)
    message.custom_arg = [
        CustomArg('marketing2', 'false', p=0),
        CustomArg('transactional2', 'true', p=0)
    ]

    message.send_at = SendAt(1461775051, p=0)

    message.to = To('test10@example.com', 'Example User10', p=1)
    message.to = [
        To('test11@example.com', 'Example User11', p=1),
        To('test12@example.com', 'Example User12', p=1)
    ]

    message.cc = Cc('test13@example.com', 'Example User13', p=1)
    message.cc = [
        Cc('test14@example.com', 'Example User14', p=1),
        Cc('test15@example.com', 'Example User15', p=1)
    ]

    message.bcc = Bcc('test16@example.com', 'Example User16', p=1)
    message.bcc = [
        Bcc('test17@example.com', 'Example User17', p=1),
        Bcc('test18@example.com', 'Example User18', p=1)
    ]

    message.header = Header('X-Test5', 'Test5', p=1)
    message.header = Header('X-Test6', 'Test6', p=1)
    message.header = [
        Header('X-Test7', 'Test7', p=1),
        Header('X-Test8', 'Test8', p=1)
    ]

    message.substitution = Substitution('%name3%', 'Example Name 3', p=1)
    message.substitution = Substitution('%city3%', 'Example City 3', p=1)
    message.substitution = [
        Substitution('%name4%', 'Example Name 4', p=1),
        Substitution('%city4%', 'Example City 4', p=1)
    ]

    message.custom_arg = CustomArg('marketing3', 'true', p=1)
    message.custom_arg = CustomArg('transactional3', 'false', p=1)
    message.custom_arg = [
        CustomArg('marketing4', 'false', p=1),
```

```
        CustomArg('transactional4', 'true', p=1)
    ]

    message.send_at = SendAt(1461775052, p=1)

    message.subject = Subject('Sending with SendGrid is Fun 1', p=1)

    # The values below this comment are global to entire message

    message.from_email = From('help@twilio.com', 'Twilio SendGrid')

    message.reply_to = ReplyTo('help_reply@twilio.com', 'Twilio SendGrid Reply')

    message.subject = Subject('Sending with SendGrid is Fun 2')

    message.content = Content(MimeType.text, 'and easy to do anywhere, even with
Python')
    message.content = Content(MimeType.html, '<strong>and easy to do anywhere, even
with Python</strong>')
    message.content = [
        Content('text/calendar', 'Party Time!!'),
        Content('text/custom', 'Party Time 2!!')
    ]

    message.attachment = Attachment(FileContent('base64 encoded content 1'),
                                    FileName('balance_001.pdf'),
                                    FileType('application/pdf'),
                                    Disposition('attachment'),
                                    ContentId('Content ID 1'))
    message.attachment = [
        Attachment(FileContent('base64 encoded content 2'),
                FileName('banner.png'),
                FileType('image/png'),
                Disposition('inline'),
                ContentId('Content ID 2')),
        Attachment(FileContent('base64 encoded content 3'),
                FileName('banner2.png'),
                FileType('image/png'),
                Disposition('inline'),
                ContentId('Content ID 3'))
    ]

    message.template_id = TemplateId('13b8f94f-bcae-4ec6-b752-70d6cb59f932')

    message.section = Section('%section1%', 'Substitution for Section 1 Tag')
    message.section = [
        Section('%section2%', 'Substitution for Section 2 Tag'),
        Section('%section3%', 'Substitution for Section 3 Tag')
    ]

    message.header = Header('X-Test9', 'Test9')
    message.header = Header('X-Test10', 'Test10')
```

```
    message.header = [
        Header('X-Test11', 'Test11'),
        Header('X-Test12', 'Test12')
    ]

    message.category = Category('Category 1')
    message.category = Category('Category 2')
    message.category = [
        Category('Category 1'),
        Category('Category 2')
    ]

    message.custom_arg = CustomArg('marketing5', 'false')
    message.custom_arg = CustomArg('transactional5', 'true')
    message.custom_arg = [
        CustomArg('marketing6', 'true'),
        CustomArg('transactional6', 'false')
    ]

    message.send_at = SendAt(1461775053)

    message.batch_id = BatchId("HkJ5yLYULb7Rj8GKSx7u025ouWVlMgAi")

    message.asm = Asm(GroupId(1), GroupsToDisplay([1,2,3,4]))

    message.ip_pool_name = IpPoolName("IP Pool Name")

    mail_settings = MailSettings()
    mail_settings.bcc_settings = BccSettings(False,
BccSettingsTo("bcc@twilio.com"))
    mail_settings.bypass_list_management = BypassListManagement(False)
    mail_settings.footer_settings = FooterSettings(True, FooterText("w00t"),
FooterHtml("<string>w00t!<strong>"))
    mail_settings.sandbox_mode = SandBoxMode(True)
    mail_settings.spam_check = SpamCheck(True, SpamThreshold(5),
SpamUrl("https://example.com"))
    message.mail_settings = mail_settings

    tracking_settings = TrackingSettings()
    tracking_settings.click_tracking = ClickTracking(True, False)
    tracking_settings.open_tracking = OpenTracking(True,
OpenTrackingSubstitutionTag("open_tracking"))
    tracking_settings.subscription_tracking = SubscriptionTracking(
        True,
        SubscriptionText("Goodbye"),
        SubscriptionHtml("<strong>Goodbye!</strong>"),
        SubscriptionSubstitutionTag("unsubscribe"))
    tracking_settings.ganalytics = Ganalytics(
        True,
        UtmSource("utm_source"),
        UtmMedium("utm_medium"),
        UtmTerm("utm_term"),
```

```
        UtmContent("utm_content"),
        UtmCampaign("utm_campaign"))
    message.tracking_settings = tracking_settings

    return message

def send_multiple_emails_personalized():
    # Assumes you set your environment variable:
    # https://github.com/sendgrid/sendgrid-
python/blob/HEAD/TROUBLESHOOTING.md#environment-variables-and-your-sendgrid-api-key
    message = build_multiple_emails_personalized()
    sendgrid_client = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
    response = sendgrid_client.send(message=message)
    print(response.status_code)
    print(response.body)
    print(response.headers)

def send_hello_email():
    # Assumes you set your environment variable:
    # https://github.com/sendgrid/sendgrid-
python/blob/HEAD/TROUBLESHOOTING.md#environment-variables-and-your-sendgrid-api-key
    message = build_hello_email()
    sendgrid_client = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
    response = sendgrid_client.send(message=message)
    print(response.status_code)
    print(response.body)
    print(response.headers)


def send_kitchen_sink():
    # Assumes you set your environment variable:
    # https://github.com/sendgrid/sendgrid-
python/blob/HEAD/TROUBLESHOOTING.md#environment-variables-and-your-sendgrid-api-key
    message = build_kitchen_sink()
    sendgrid_client = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
    response = sendgrid_client.send(message=message)
    print(response.status_code)
    print(response.body)
    print(response.headers)


## this will actually send an email
# send_hello_email()

## this will send multiple emails
# send_multiple_emails_personalized()

## this will only send an email if you set SandBox Mode to False
# send_kitchen_sink()
```

App.py

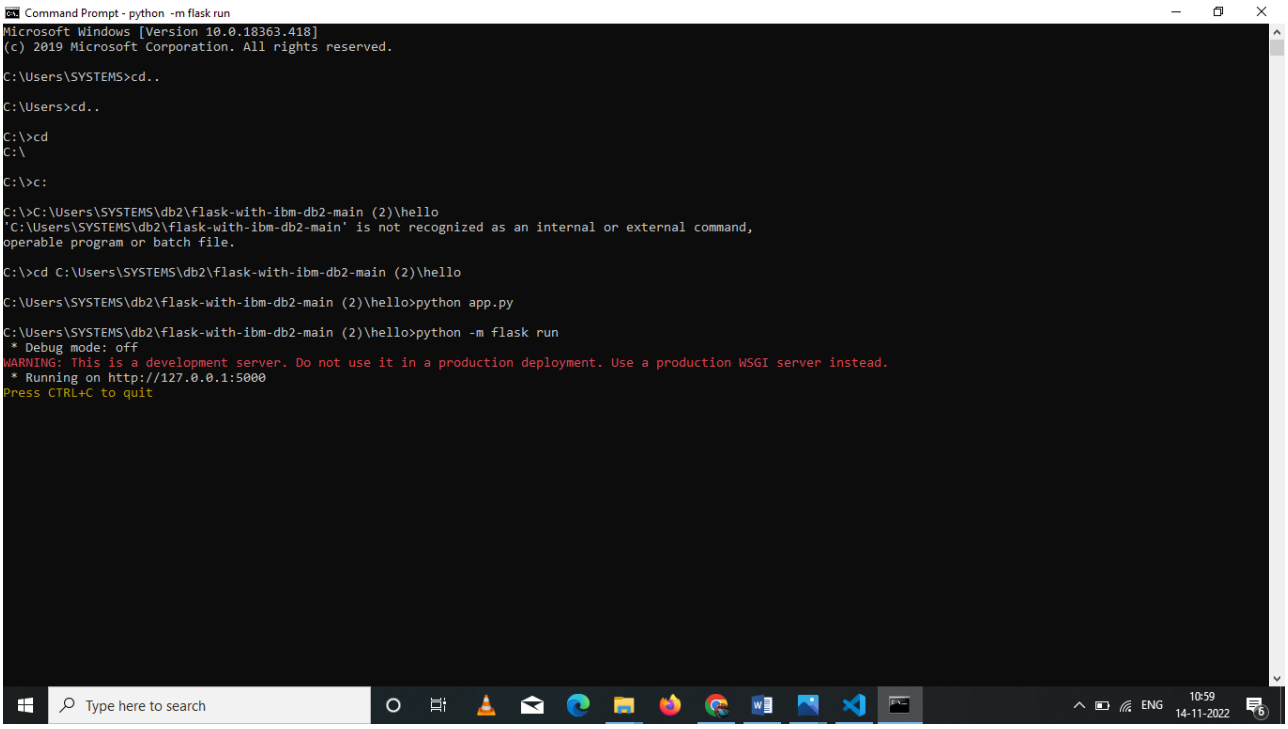
```
from flask import Flask, render_template, request, redirect, url_for, session
import ibm_db
import re

app = Flask(__name__)

app.secret_key = 'a'

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31498;SECURITY=SSL
;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=mzs36647;PWD=tuNhdQMaomMRt7dv",'
','')

@app.route('/')

def homer():
    return render_template('home.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 users 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['USERNAME']
            userid=  account['USERNAME']
            session['username'] = account['USERNAME']
            msg = 'Logged in successfully !'

            msg = 'Logged in successfully !'
            return render_template('dashboard.html', msg = msg)
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)
```

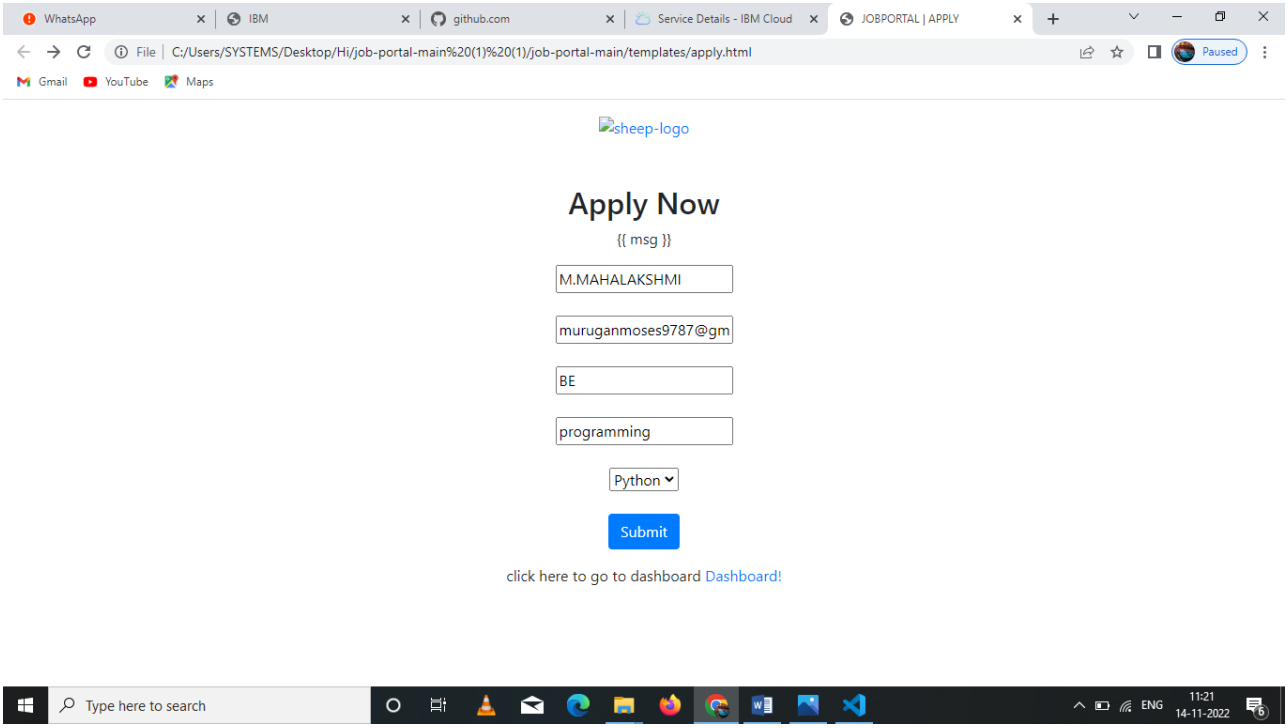
```python
@app.route('/register', methods =['GET', 'POST'])
def registet():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        sql = "SELECT * FROM users WHERE username =?"
        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:
            insert_sql = "INSERT INTO  users VALUES (?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, username)
            ibm_db.bind_param(prep_stmt, 2, email)
            ibm_db.bind_param(prep_stmt, 3, password)
            ibm_db.execute(prep_stmt)
            msg = 'You have successfully registered !'
    elif request.method == 'POST':
        msg = 'Please fill out the form !'
    return render_template('register.html', msg = msg)

@app.route('/dashboard')
def dash():

    return render_template('dashboard.html')

@app.route('/apply',methods =['GET', 'POST'])
def apply():
     msg = ''
     if request.method == 'POST' :
         username = request.form['username']
         email = request.form['email']

         qualification= request.form['qualification']
         skills = request.form['skills']
         jobs = request.form['s']
         sql = "SELECT * FROM users WHERE username =?"
         stmt = ibm_db.prepare(conn, sql)
         ibm_db.bind_param(stmt,1,username)
```

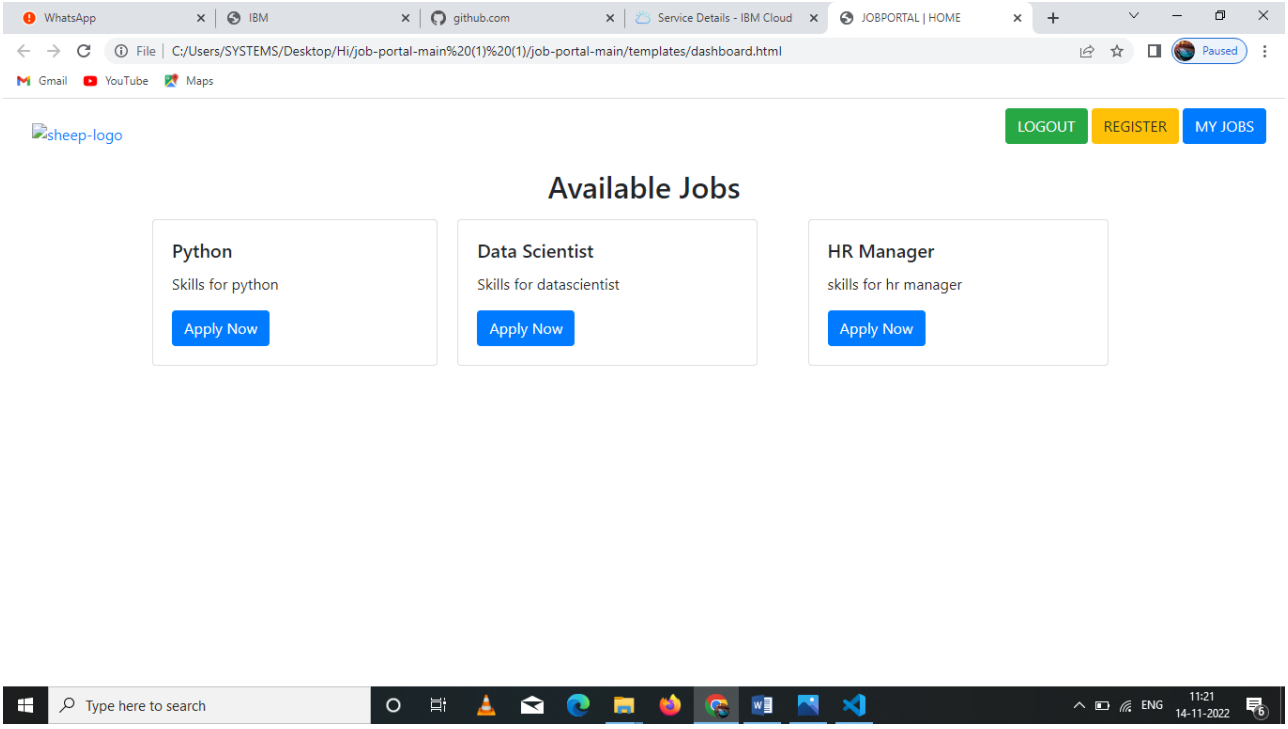
```
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
           msg = 'there is only 1 job position! for you'
           return render_template('apply.html', msg = msg)



        insert_sql = "INSERT INTO  job VALUES (?, ?, ?, ?, ?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, username)
        ibm_db.bind_param(prep_stmt, 2, email)
        ibm_db.bind_param(prep_stmt, 3, qualification)
        ibm_db.bind_param(prep_stmt, 4, skills)
        ibm_db.bind_param(prep_stmt, 5, jobs)
        ibm_db.execute(prep_stmt)
        msg = 'You have successfully applied for job !'
        session['loggedin'] = True
        TEXT = "Hello sandeep,a new appliaction for job position" +jobs+"is
requested"

        #sendmail(TEXT,"sandeep@thesmartbridge.com")
        sendgridmail("sandeep@thesmartbridge.com",TEXT)



    elif request.method == 'POST':
        msg = 'Please fill out the form !'
    return render_template('apply.html', msg = msg)

@app.route('/display')
def display():
    print(session["username"],session['id'])

    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM job WHERE userid = % s', (session['id'],))
    account = cursor.fetchone()
    print("accountdislay",account)


    return render_template('display.html',account = account)

@app.route('/logout')

def logout():
   session.pop('loggedin', None)
   session.pop('id', None)
   session.pop('username', None)
   return render_template('home.html')
```

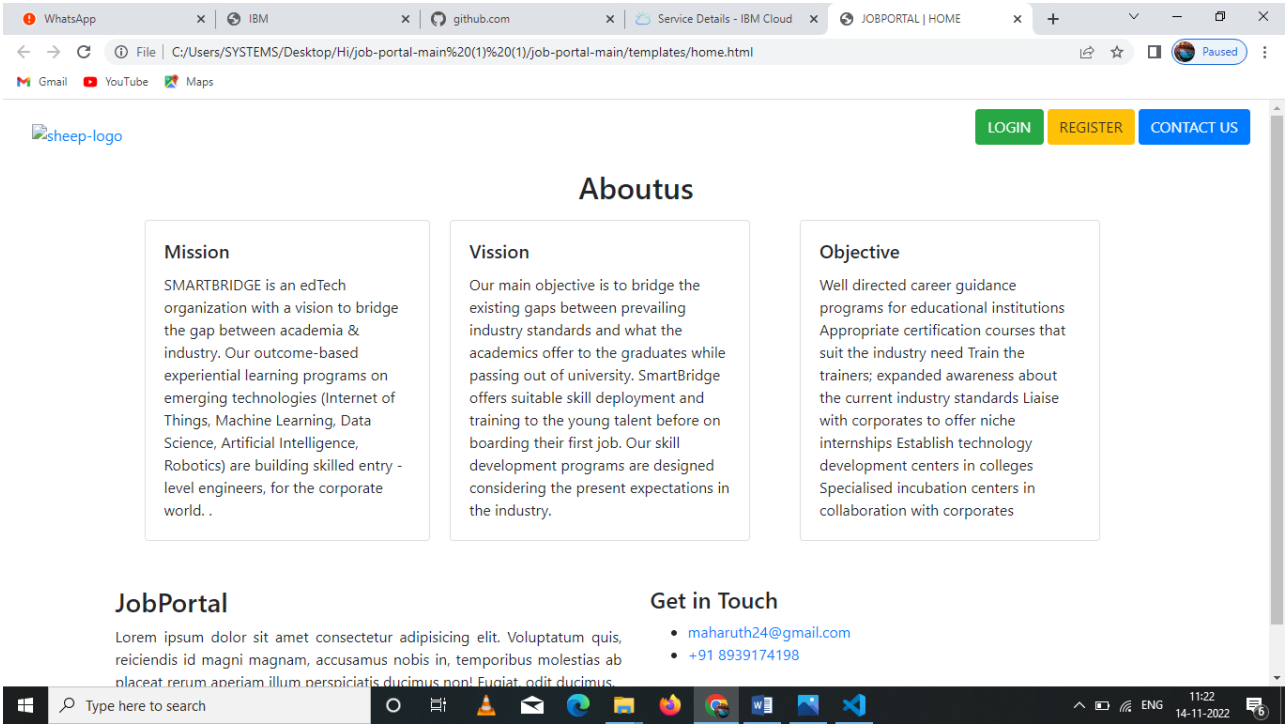
```python
if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

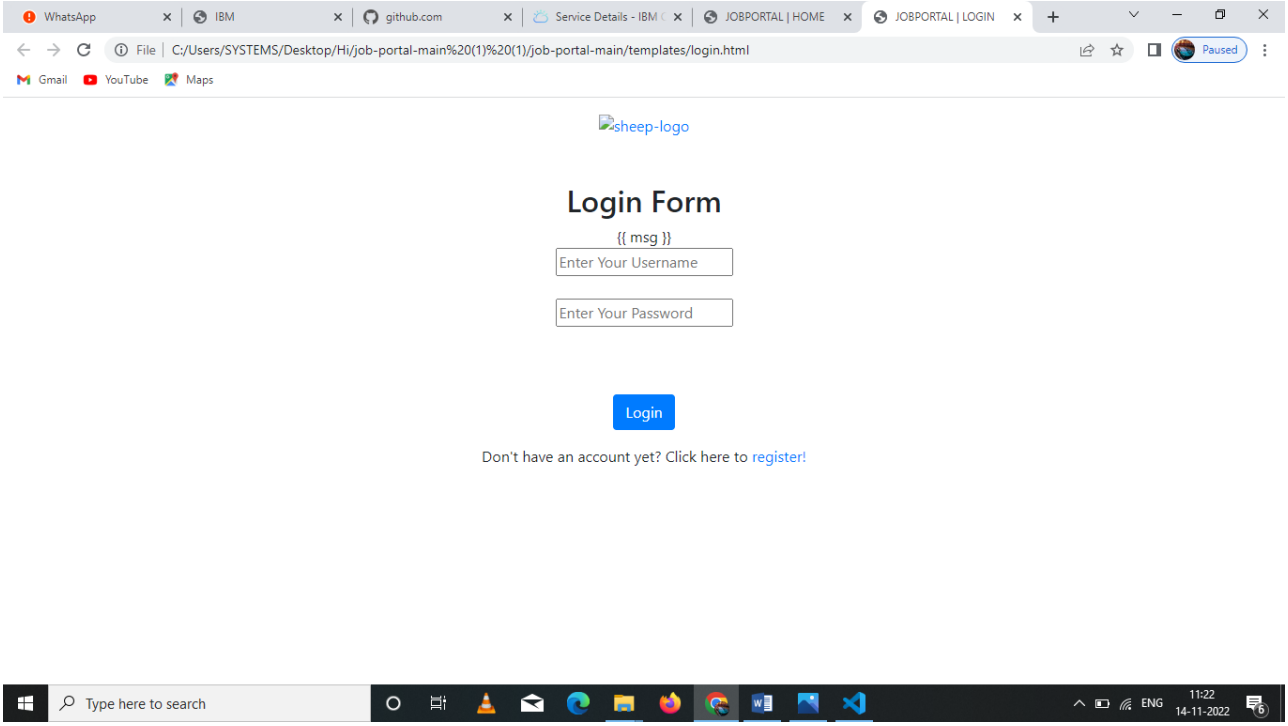
screenshots






WhatsApp
IBM
github.com
Service Details - IBM Cloud
JOBPORTAL | HOME
File | C:/Users/SYSTEMS/Desktop/Hi/job-portal-main%20(1)%20(1)/job-portal-main/templates/dashboard.html
Gmail
YouTube
Maps
Paused
sheep-logo
LOGOUT
REGISTER
MY JOBS
Available Jobs
Python
Skills for python
Apply Now
Data Scientist
Skills for datascientist
Apply Now
HR Manager
skills for hr manager
Apply Now
Type here to search
ENG
11:21
14-11-2022


WhatsApp
IBM
github.com
Service Details - IBM Cloud
JOBPORTAL | HOME
File | C:/Users/SYSTEMS/Desktop/Hi/job-portal-main%20(1)%20(1)/job-portal-main/templates/home.html
Gmail
YouTube
Maps
Paused
sheep-logo
LOGIN
REGISTER
CONTACT US
Aboutus
Mission
SMARTBRIDGE is an edTech organization with a vision to bridge the gap between academia & industry. Our outcome-based experiential learning programs on emerging technologies (Internet of Things, Machine Learning, Data Science, Artificial Intelligence, Robotics) are building skilled entry - level engineers, for the corporate world. .
Vission
Our main objective is to bridge the existing gaps between prevailing industry standards and what the academics offer to the graduates while passing out of university. SmartBridge offers suitable skill deployment and training to the young talent before on boarding their first job. Our skill development programs are designed considering the present expectations in the industry.
Objective
Well directed career guidance programs for educational institutions Appropriate certification courses that suit the industry need Train the trainers; expanded awareness about the current industry standards Liaise with corporates to offer niche internships Establish technology development centers in colleges Specialised incubation centers in collaboration with corporates
JobPortal
Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptatum quis, reiciendis id magni magnam, accusamus nobis in, temporibus molestias ab
Get in Touch
maharuth24@gmail.com
+91 8939174198
Type here to search
ENG
11:22
14-11-2022


WhatsApp
IBM
github.com
Service Details - IBM
JOBPORTAL | HOME
JOBPORTAL | LOGIN
File | C:/Users/SYSTEMS/Desktop/Hi/job-portal-main%20(1)%20(1)/job-portal-main/templates/login.html
Paused
Gmail
YouTube
Maps
sheep-logo
Login Form
{{ msg }}
Enter Your Username
Enter Your Password
Login
Don't have an account yet? Click here to register!
Type here to search
ENG
11:22
14-11-2022