

Project Development Phase(Sprint 3)

Team ID	PNT2022TMID04674
Project Name	Skill/Job Recommender Application

Pages:

Current Question : 3/10

15 minutes left

Submit Test

Which of the following is immutable by default in Rust but can be made mutable?

variable

const

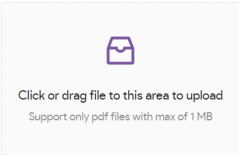
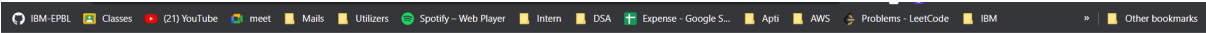
user defined object

Depends on usage

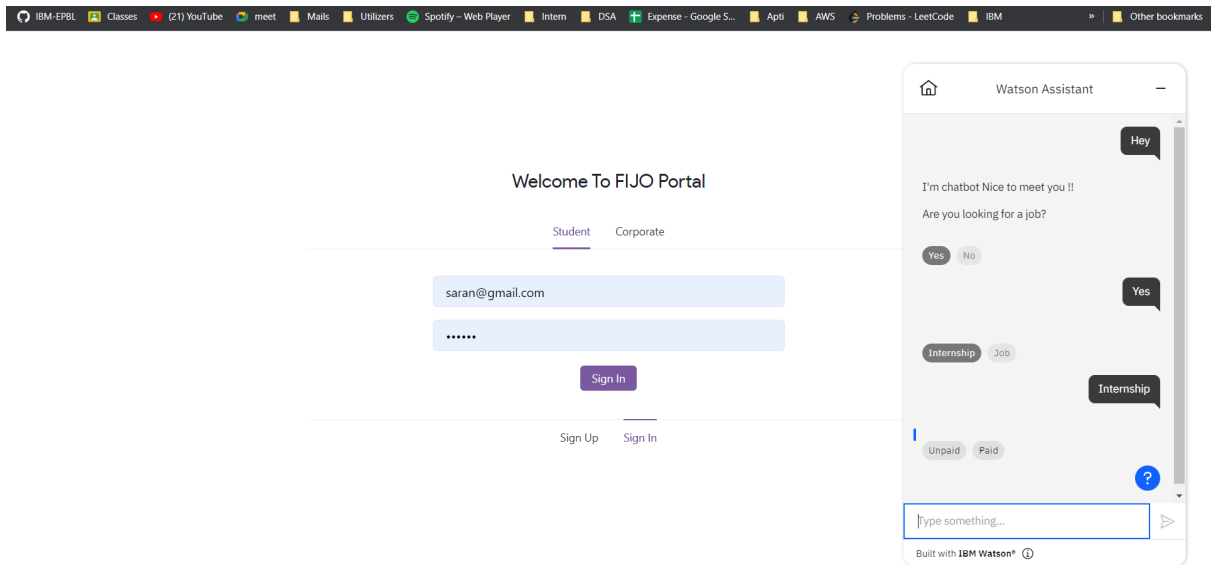
Next

Saved answers : 3

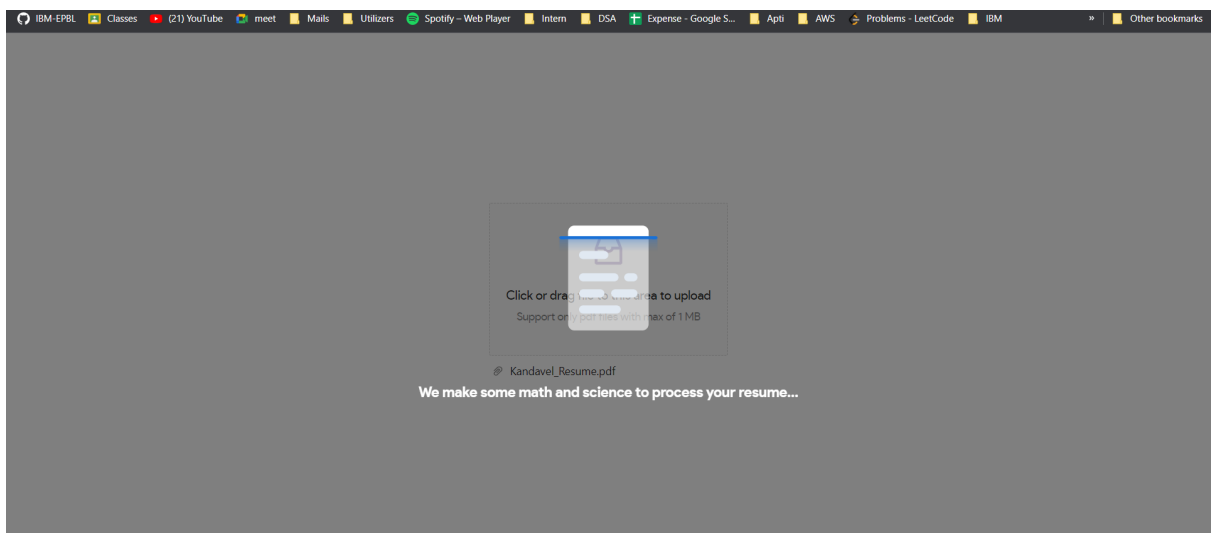
Assessment Page



Resume upload Page



Chatbot



Resume uploading

Personal Details

Email (same as account)
sanjai@gmail.com

Name
Kongu Engineering College

Education
ECE KEC 9

[Add Another Education Field](#)

Experience
1

Mobile Number
+919962857600

Details fetch from resume

Build pixel-perfect, robust and, accessible user interfaces on the web - Drive independently a project from inception to production deployment with a strong focus on performance and robustness - Follow our software development process i...

[Read more](#)

Recommended Assessments based on your skills

Javascript

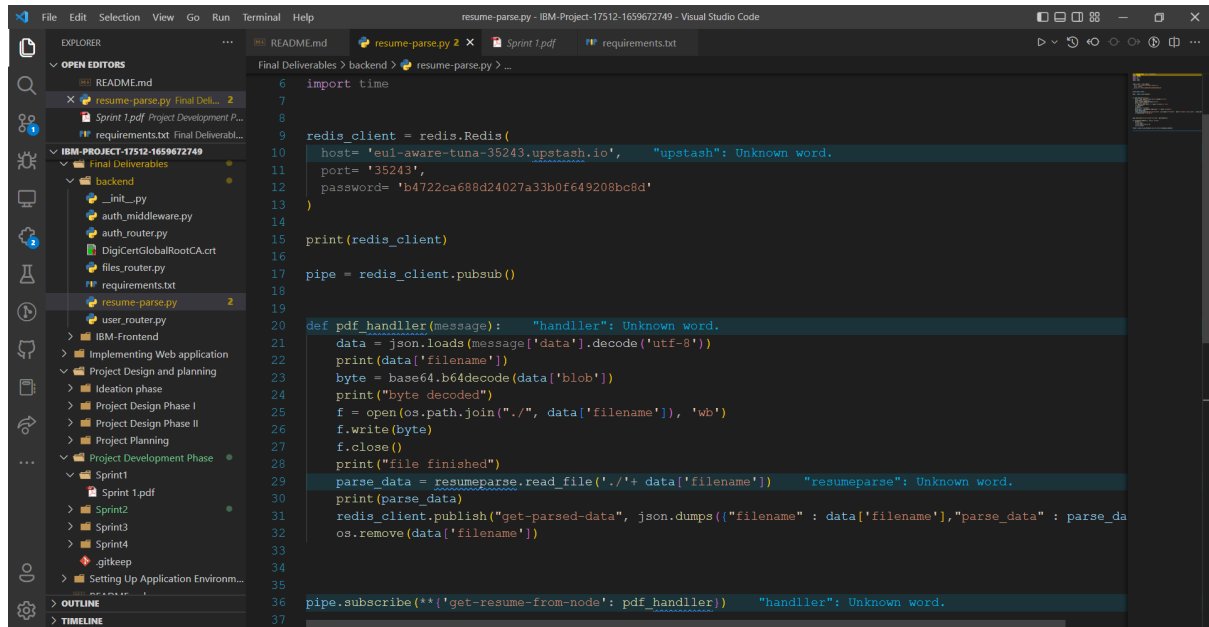
Total Questions : 10
Pass Percentage: 65%

[Take Test](#)

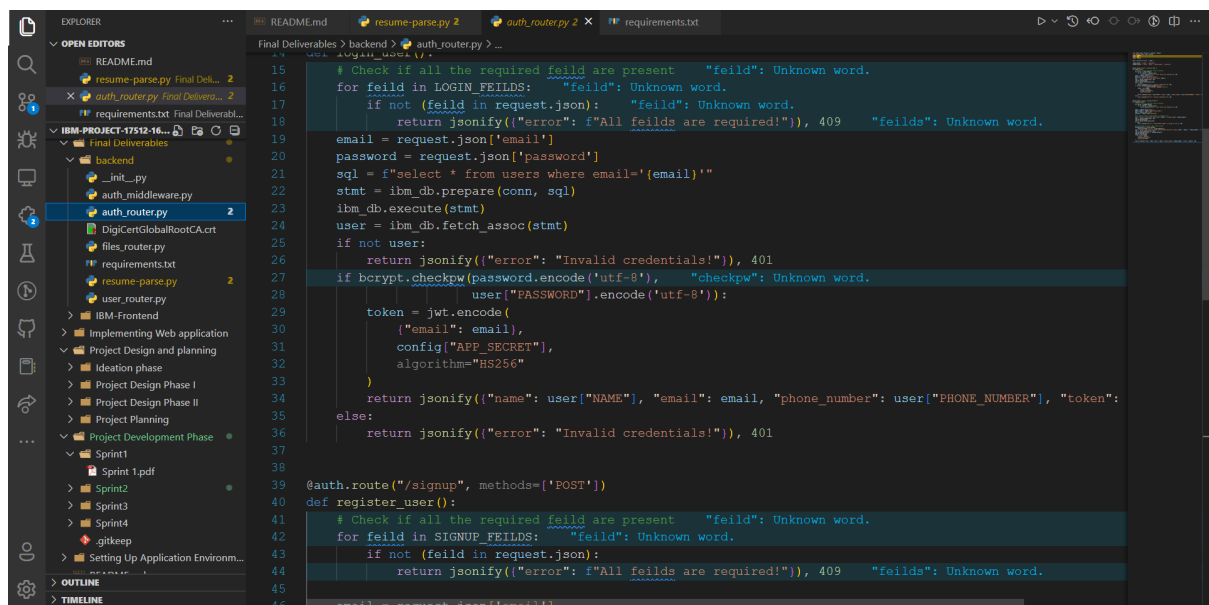
Are you wanna groom your skills? Here are some hackathons for you

Assessment based on my Skillset

Code Examples:



```
6 import time
7
8
9
10 redis_client = redis.Redis(
11     host= 'eul-aware-tuna-35243.upstash.io', "upstash": Unknown word.
12     port= '35243',
13     password= 'b4722ca688d24027a33b0f649208bc8d'
14 )
15
16 print(redis_client)
17
18 pipe = redis_client.pubsub()
19
20 def pdf_handler(message): "handler": Unknown word.
21     data = json.loads(message['data'].decode('utf-8'))
22     print(data['filename'])
23     byte = base64.b64decode(data['blob'])
24     print("byte decoded")
25     f = open(os.path.join("./", data['filename']), 'wb')
26     f.write(byte)
27     f.close()
28     print("file finished")
29     parse_data = resumeparse.read_file('./' + data['filename']) "resumeparse": Unknown word.
30     print(parse_data)
31     redis_client.publish("get-parsed-data", json.dumps({"filename" : data['filename'], "parse_data" : parse_data}))
32     os.remove(data['filename'])
33
34
35
36 pipe.subscribe(("get-resume-from-node": pdf_handler)) "handler": Unknown word.
37
```



```
15 # Check if all the required feild are present "feild": Unknown word.
16 for feild in LOGIN_FEILDS: "feild": Unknown word.
17     if not (feild in request.json): "feild": Unknown word.
18         return jsonify({"error": f"All feilds are required!"}), 409 "feilds": Unknown word.
19
20 email = request.json['email']
21 password = request.json['password']
22 sql = f"select * from users where email='{email}'"
23 stmt = ibm_db.prepare(conn, sql)
24 ibm_db.execute(stmt)
25 user = ibm_db.fetch_assoc(stmt)
26 if not user:
27     return jsonify({"error": "Invalid credentials!"}), 401
28 if bcrypt.checkpw(password.encode('utf-8'), user["PASSWORD"].encode('utf-8')):
29     token = jwt.encode(
30         {"email": email},
31         config["APP_SECRET"],
32         algorithm="HS256"
33     )
34     return jsonify({"name": user["NAME"], "email": email, "phone_number": user["PHONE_NUMBER"], "token":
35     })
36 else:
37     return jsonify({"error": "Invalid credentials!"}), 401
38
39 @auth.route("/signup", methods=['POST'])
40 def register_user():
41     # Check if all the required feild are present "feild": Unknown word.
42     for feild in SIGNUP_FEILDS: "feild": Unknown word.
43         if not (feild in request.json):
44             return jsonify({"error": f"All feilds are required!"}), 409 "feilds": Unknown word.
45
46     email = request.json['email']
```

```
13 files = Blueprint("files", __name__)
14
15
16
17 def multi_part_upload(bucket_name, item_name, file_path):
18     try:
19         print("Starting file transfer for (0) to bucket: (1)\n".format(
20             item_name, bucket_name))
21         # set 5 MB chunks
22         part_size = 1024 * 1024 * 5
23
24         # set threshold to 15 MB "threshold": Unknown word.
25         file_threshold = 1024 * 1024 * 15
26
27         # set the transfer threshold and chunk size
28         transfer_config = ibm_boto3.s3.transfer.TransferConfig( "boto": Unknown word.
29             multipart_threshold=file_threshold,
30             multipart_chunksize=part_size "chunksize": Unknown word.
31         )
32
33         # the upload_fileobj method will automatically execute a multi-part upload
34         # in 5 MB chunks for all files over 15 MB
35         with open(file_path, "rb") as file_data:
36             cos.Object(bucket_name, item_name).upload_fileobj(
37                 Fileobj=file_data,
38                 Config=transfer_config
39             )
40
41         print("Transfer for (0) Complete!\n".format(item_name))
42     except ClientError as be:
43         print("CLIENT ERROR: (0)\n".format(be))
44     except Exception as e:
```

Full code - [Github\(Team id -PNT2022TMID04674\)](#)