Project Name:

Education-Cube (make higher education selection easy!!)

Project Description:

As we all know in today's competitive world everyone are chasing top emerging careers, got enrollment in those courses and then they didn't get the jobs due to lack of enough skills which ends up with stress, disappointment and depression.

What could happen if person gets the job which he really loves and have the skills!!!!

This Project is all about to mitigate the troubles between to find correct education program which matches with an individual's true skills for which he is actually passionate and wants to work in that field.

Technical Workflow:

Extraction of data

- We extracted data from some standard pdfs which is having information about Colleges and Universities of India using 'Docparser'.Here,we did some manipulations to make data standardized using MS Excel
- We also covered some colleges and universities from Canada by extracting them manually or with web scrapping tool like 'import.io'

Launch RDS instance - MYSQL on AWS

- After that we have launched one RDS instance on AWS, where we made datawarehouse and uploaded already extracted data.Here we used python scripts to upload data from Excel files to datawarehouse

Parse and Feel Geographical information

- Here,we only had colleges or universities names and some abstracted addressess. So, we need to more detail level geographical information to suggest good options by considering distance as one of important parameter
- For that we used Google Geocoding API to get the more deep informations for addressess like latitude, longitude, street number,county,formated address and more

Launch ElasticBeanStalk on AWS

- At this stage we needed something that exports required resultant data in JSON format on web page,from there an android application can easily fetch those information to show them to end users
- Therefore, we decided to take leverage AWS services which can connect AWS RDS instance easily and we launched one EB Application with dedicated enviroment having python flask web server script
- This flask web server script contains sql queries which take the conditional parameter from the URLs and return resultant data in JSON format

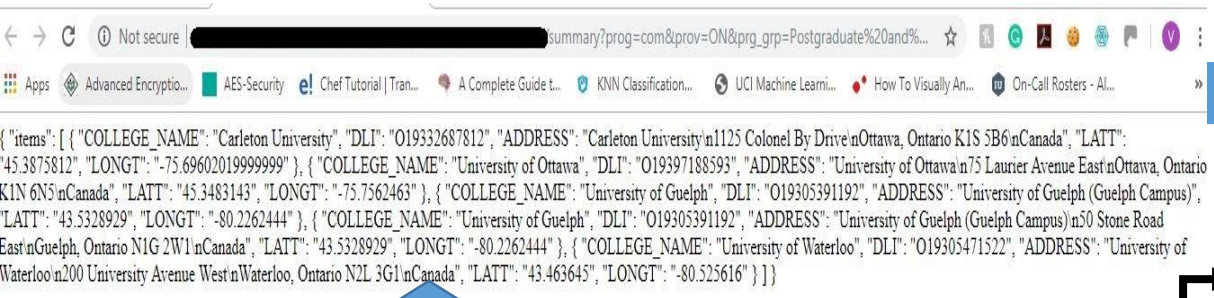
Technical Tools:

Serial No.	Name	Usage	Web Links
1	DocParser	To extract data from pdfs	https://docparser.com/
2	Import.io	To extract information from web data	https://www.import.io/
3	AWS RDS - MYSQL	To build data warehouse in MYSQL on AWS	
4	MYSQL Workbench 8.0 CE	Client tool to access database	Needed to be download
5	Python 3.6(64 bit)	Scripting language to make and deploy logic for various tasks	Needed to be download We use with Spyder framework supported by Anaconda
6	Google Geocoding API	To parse geographical information for addresses like latitude, longitude etc.	Register for API and get the key
7	AWS EB App (has to be in same region)	To run the web application and easily access AWS RDS instance	Use EB CLI and deploy your FLASK script in the EB environment

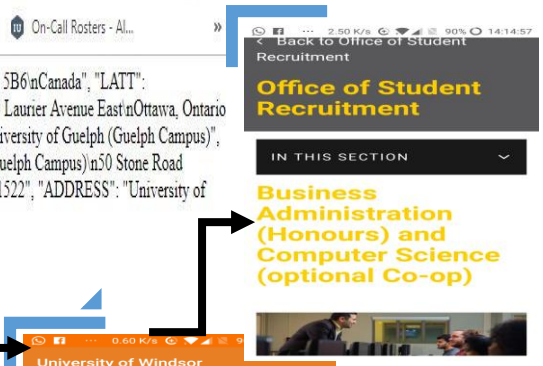
Data Utilization over Android Application side:

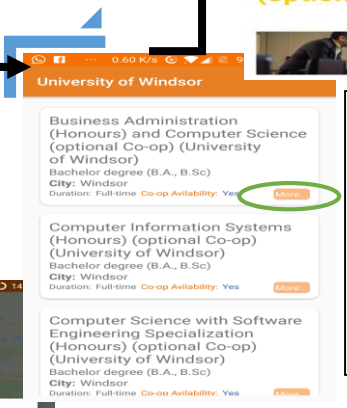
My Bake-End part is being utilized over here

Below one is the cycle that shows how the whole system works

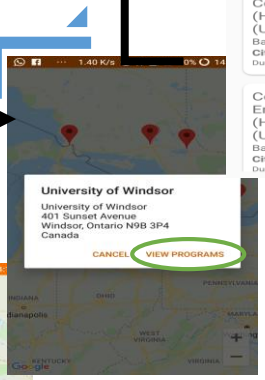


Above picture shows one of the EB web page, which takes conditional arguments in URL and retrieve resultant data in the JSON format

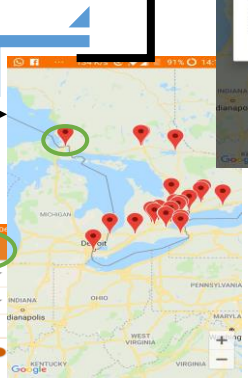




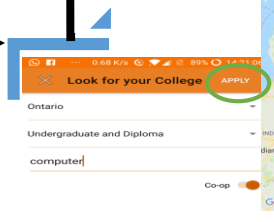
By tapping on more button on previous screen, it redirects you to official program page in the college website

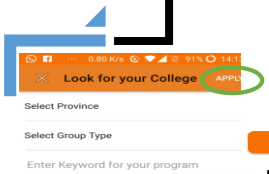


Showing education programs of the chosen college having searched keyword



Selecting College by tapping on college pointer





Showing colleges pointers on map which has education program matching with program keyword

Adding Values in Search Page

Search Page

This is project I did for a corporate company, so due to project confidentiality I can't reveal the whole code, but I can provide some code snippets over here.

Elastic Beanstalk Python script snippet (to fetch resultant data from RDS instance):

```
from flask import Flask,request
from flask import app as application
import pymysql
import json

application = Flask(__name__)

@application.route('/summary')
def summary():
    prog = request.args.get('prog')
    prov=request.args.get('prov')
    prg_grp= request.args.get('prg_grp')
    co_op= request.args.get('co_op')


    view_query = """select distinct COLLEGE_NAME,DLI,ADDRESS,LATT,LONGT from [redacted]
                where instr(upper(PROGRAM_NAME),upper(%s))>0
                    and upper(PROV_CODE) = upper(%s)
                    and upper(PRG_LVL_GRP)= upper(%s)
                    and upper(CO_OP_AVB)=upper(%s)"""
    value= (prog,prov,prg_grp,co_op)
    cnx= pymysql.connect(host="[redacted]", user="[redacted]", passwd="[redacted]",
           port=3306, db="[redacted]", autocommit=True)
    cursor=cnx.cursor()

    cursor.execute(view_query,value)
    result=cursor.fetchall()
            # print(result)
    items = [dict(zip([key[0] for key in cursor.description], row)) for row in result]

    results_list = json.dumps({'items':items},indent=4)
    print(results_list)
    return results_list
    cursor.close()

```

Geographical Information Processing of given places (i.e. Latitude, Longitude, Formatted Addresses)

```
from requests import get
from json import dump
import mysql.connector as mc
API_KEY = '[redacted]'

mydb = mc.connect(host = '[redacted]'
                  ,user = '[redacted]'
                  ,passwd='[redacted]'
                  ,database='[redacted]')

mycur = mydb.cursor()

query_1 = "select concat(CAMPUSES,CITY,NAME) from [redacted]"
mycur.execute(query_1)
addresses = mycur.fetchall()

for add in addresses:
    url = 'https://maps.googleapis.com/maps/api/geocode/json?components=&language=&region=&bounds=&key='+API_KEY
    url = url + '&address='+ str(add[0]).replace(" ","+")
    print(url)
    response = get(url)
    print(response.content)
    json = response.json()
    if json['results']:
     data = json['results'][0]
     for item in data['address_components']:
        for category in item['types']:
            data[category] = {}
            data[category] = item['long_name']
            street = data.get("route", None)
            state = data.get("administrative_area_level_1", None)
            city = data.get("locality", None)
            county = data.get("administrative_area_level_2", None)
            country = data.get("country", None)
            postal_code = data.get("postal_code", None)
            neighborhood = data.get("neighborhood",None)
            sublocality = data.get("sublocality", None)
            housenumber = data.get("housenumber", None)
            postal_town = data.get("postal_town", None)
            subpremise = data.get("subpremise", None)
            latitude = data.get("geometry", {}).get("location", {}).get("lat", None)
            longitude = data.get("geometry", {}).get("location", {}).get("lng", None)
            location_type = data.get("geometry", {}).get("location_type", None)
            postal_code_suffix = data.get("postal_code_suffix", None)
            street_number = data.get('street_number', None)
            address =data.get("formatted_address",None)

            up_sql = """update [redacted]
                        set ADDRESS= %s,LATT=%s,LONGT=%s
                        where concat(CAMPUSES,CITY,NAME) = %s"""
            values=(address,latitude,longitude,add[0])
            mycur.execute(up_sql,values)
            mydb.commit()
            print(mycur.rowcount,"number rows affected")

print(mycur.rowcount,"number rows affected")
mydb.commit()
mycur.close()
mydb.close()
```

Large PDF Divider Python Script:

```
import PyPDF2 as pdf
import pandas as pd

pdfFileObj = open('[redacted]','rb')
pdfFileReader= pdf.PdfFileReader(pdfFileObj)
pageNo= pdfFileReader.numPages
pdfWriter=pdf.PdfFileWriter()

for i in range (3,25):
    pdfPageReader = pdfFileReader.getPage(i)
    pdfWriter.addPage(pdfPageReader)

    part_1 =open('[redacted]','wb')
    pdfWriter.write(part_1)

part_1.close()

for j in range(25,49):
    pdfPageReader = pdfFileReader.getPage(j)
    pdfWriter.addPage(pdfPageReader)

    part_2 =open('[redacted]','wb')
    pdfWriter.write(part_2)

part_2.close()

pdfFileObj.close()
```

Demo Web Scraper Script:


```
{"_id":"oc_info","startUrl":["[redacted]"],"selectors":[{"id":"colleges_info","type":"Selector
electors":["program_element"],"selector":"p.college-name","multiple":false,"regex":"(?<=\\:)(.*?)(?=\\|)","delay":0},{"id":"Pr
an","multiple":false,"regex":"","delay":0},{"id":"program_level","type":"SelectorText","parentSelectors":["program_element"],"
t","parentSelectors":["program_element"],"selector":"dd:nth-of-type(10)","multiple":false,"regex":"","delay":0}]}
```


You can download this application from URL listed at below:

https://play.google.com/store/apps/details?id=com.educationcube.cube