COMP208 Group Project – Group 13

Technical Documentation

Table of Contents

[Introduction 3](#_Toc69127225)

[Front End 4](#_Toc69127226)

[Flutter App 4](#_Toc69127227)

[Back End 5](#_Toc69127228)

[Overview 5](#_Toc69127229)

[Python Modules 5](#_Toc69127230)

[Directory Structure 5](#_Toc69127231)

[How to Use 5](#_Toc69127232)

[Config.py 5](#_Toc69127233)

[Flask Web Server 6](#_Toc69127234)

[Overview 6](#_Toc69127235)

[\_\_init\_\_.py 6](#_Toc69127236)

[Errors.py 6](#_Toc69127237)

[Models.py 6](#_Toc69127238)

[Routes.py 8](#_Toc69127239)

[Database 10](#_Toc69127240)

[Overview 10](#_Toc69127241)

[SQL Alchemy Explained 10](#_Toc69127242)

[MySQL Database 10](#_Toc69127243)

[Migrations 10](#_Toc69127244)

[Comp208.db 10](#_Toc69127245)

[Web Scraping 11](#_Toc69127246)

[Overview 11](#_Toc69127247)

[Indeed\_import.py 11](#_Toc69127248)

[Scraping Directory 11](#_Toc69127249)

[Job Recommender 12](#_Toc69127250)

[Overview 12](#_Toc69127251)

[Job\_suggestions.py 12](#_Toc69127252)

# Introduction

# Front End

## Flutter App

# Back End

## Overview

### Python Modules

|  |  |  |
| --- | --- | --- |
| Module | Description | Use |
| alembic==1.5.8 |  |  |
| appnope==0.1.2 |  |  |
| autopep8==1.5.6 |  |  |
| backcall==0.2.0 |  |  |
| beautifulsoup4==4.9.3 |  |  |
| bs4==0.0.1 |  |  |
| certifi==2020.12.5 |  |  |
| chardet==4.0.0 |  |  |
| click==7.1.2 |  |  |
| decorator==5.0.5 |  |  |
| Flask==1.1.2 |  |  |
| Flask-HTTPAuth==4.2.0 |  |  |
| Flask-Migrate==2.7.0 |  |  |
| Flask-SQLAlchemy==2.5.1 |  |  |
| greenlet==1.0.0 |  |  |
| idna==2.10 |  |  |
| ipykernel==5.5.3 |  |  |
| ipython==7.22.0 |  |  |
| ipython-genutils==0.2.0 |  |  |
| itsdangerous==1.1.0 |  |  |
| jedi==0.18.0 |  |  |
| Jinja2==2.11.3 |  |  |
| joblib==1.0.1 |  |  |
| jupyter-client==6.1.12 |  |  |
| jupyter-core==4.7.1 |  |  |
| Mako==1.1.4 |  |  |
| MarkupSafe==1.1.1 |  |  |
| numpy==1.20.2 |  |  |
| pandas==1.2.3 |  |  |
| parso==0.8.2 |  |  |
| pexpect==4.8.0 |  |  |
| pickleshare==0.7.5 |  |  |
| prompt-toolkit==3.0.18 |  |  |
| ptyprocess==0.7.0 |  |  |
| pycodestyle==2.7.0 |  |  |
| Pygments==2.8.1 |  |  |
| python-dateutil==2.8.1 |  |  |
| python-editor==1.0.4 |  |  |
| pytz==2021.1 |  |  |
| pyzmq==22.0.3 |  |  |
| requests==2.25.1 |  |  |
| scikit-surprise==1.1.1 |  |  |
| scipy==1.6.2 |  |  |
| selenium==3.141.0 |  |  |
| six==1.15.0 |  |  |
| soupsieve==2.2.1 |  |  |
| SQLAlchemy==1.4.11 |  |  |
| toml==0.10.2 |  |  |
| tornado==6.1 |  |  |
| traitlets==5.0.5 |  |  |
| urllib3==1.25.8 |  |  |
| wcwidth==0.2.5 |  |  |
| Werkzeug==1.0.1 |  |  |
| xlwt==1.3.0 |  |  |

### Directory Structure

### How to Use

### Config.py

## Flask Web Server

### Overview

### \_\_init\_\_.py

### Errors.py

|  |  |  |
| --- | --- | --- |
| Error | Code | Description |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Models.py

|  |  |  |
| --- | --- | --- |
| Class | Attribute | Description |
| User |  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Rating |  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Job |  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

### Routes.py

REST API

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| API Route | HTTP Method | Auth | Client Body | Server Response Success | Description |
| “/api/job” | GET | **✓** |  |  |  |
| “/api/job” | POST | **✓** |  |  |  |
| “/api/rating” | GET | **✓** |  |  |  |
| “/api/rating” | POST | **✓** |  |  |  |
| “/api/create/user” | POST | **✖** |  |  |  |
| “/api/user” | GET | **✓** |  |  |  |
| “/api/user” | PUT | **✓** |  |  |  |
| “/api/user” | DELETE | **✓** |  |  |  |

## Database

### Overview

### First Time Set Up

“flask db init”

“flask db migrate -m “comp208”

“flask db migrate”

“flask db upgrade”

### SQL Alchemy Explained

### MySQL vs SQLite Database

### Changing Database Layout

## Web Scraping

### Overview

Web scraping is the process of automatically parsing a web page for data and is a technique I have implemented in this project to harvest job postings from Indeed.com, a well-known jobs website.

The process is relatively straight-forward in Python in principle as there are a number of modules which massively simplify the process, such as “Beautiful Soup”. However, in practice, it is slightly more complex.

While web scraping is completely legal (in fact Indeed.com, where I am getting the job data from gets many of its jobs from scraping other sites) as all the information is already publicly available, many websites don’t like the practice, as it may reduce the number of people visiting their site, or it could cost them money as their servers are placed under a higher than usual load due to the increased number of requests. Due to this, they can create a number of obstacles to discourage this practice.

One of the major barriers are websites attempting to stop you from collecting their data by limiting, or completely block, your IP if they detect too many requests in a specific timeframe. In order to counter this, in my script I employ the use of proxy addresses, the details of which are elaborated upon in the next section.

### Indeed\_import.py

This is where all of the functions which make the web scraping happen are.

### Scraping Directory

### Proxies.JSON

### Indeed.JSON

## Job Recommender

### Overview

When a client makes a GET API call to the

### Job\_suggestions.py

# Sources

<https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world> - all flask and flask-sql stuff

<https://realpython.com/build-recommendation-engine-collaborative-filtering/> - AI recommendations

<https://flutterawesome.com/a-flutter-package-that-helps-you-create-on-boarding-screen/>

<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html>

<https://realpython.com/python-web-scraping-practical-introduction/>

<https://www.scrapehero.com/how-to-rotate-proxies-and-ip-addresses-using-python-3/>

<https://uk.indeed.com/?r=us>