Installation

1. Setting up a virtual environment

For the project to operate, a python 3.7+ virtual environment is needed to be installed at this path: `ProjectRoot/venv`. That is achievable by executing the following command at the root directory of the project: `python3 -m venv venv`. Note that you might need to replace 'python3' with 'python' depending on your PATH configuration.

2. Installing all the required packages

All the required packages are stored in `requirements.txt` (pip freeze). This file can be fed directly to the pip module; like this:

```
venv\Scripts\activate
```

```
pip install -r requirements.txt
```

Note that you need to execute those commands in the project's root directory.

Almost done! How to activate it (schedule it)?

Simply by running the `schedule-task.bat` script. that will schedule the scraping, processing, and uploading tasks for every Monday at 09:00 AM. Note that those are the 3 tasks that are done by the python modules. The rest of the automation process (form creation and emailing) is done by a separate and independent google apps script project that is scheduled separately for every Monday between 10:00 to 11:00 AM.

To check if the task is scheduled correctly, you can run the `check-task.bat` script; and the result should be something similar to this:

```
TaskName                                 Next Run Time          Status
======================================== ====================== ===============
MIB-Automation                           4/19/2021 9:00:00 AM   Ready
```

For running the whole thing out of schedule, two simple steps are ahead of you; first running the `run-agent.bat` script which is the same script that windows scheduler executes. Second, running the google apps script manually from the apps script web platform.

How does it work?

Let me walk you through the whole process with a good amount of detail. The automation process has 5 steps; the last step, which is scheduling, is done separately for each of the two platforms we are working with. The first two steps of the process, web scraping, processing the data, and uploading the results to google drive, are done by the python project. The second two steps, form generation, sending emails, are done by a google apps script. Which uses the data that is uploaded by the python module to a specific spreadsheet on google drive.

These are the files that contain almost all the goods within the python project:

- `cast_spider.py`
  - Contains an extended Scrapy spider class definition named `CastSpider`.
  - Does the scraping process and returns instances of `ImdbcastItem` class (custom scrapy item type defined based on our need).
  - By default, its parameters are set to scrap 100 pages that each contain 50 movies. That parameter can be changed in order to achieve larger or smaller scales of data.
- `agent.py`
  - This script combines all the parts together and handles all the small details.
  - It runs the spider using the scrapy core API, pipes the scraped data to a file named `movies.json`, runs the `most_frequent_members.py` script and grabs the results; and finally uploads the top 5 cast list to google drive.
- `most_frequent_members.py`
  - There is a single function defined in this script that does all the magic.
  - The function loads the `movies.json` file as a pandas Dataframe object.
  - Returns the final result as a python list containing 5 actor/actress names.
- `client_secret.json`
  - This file plays an important role in the project.
  - It's a google drive credentials file created from a google account using the google cloud console.
  - The `agent.py` script uses this to authenticate into the target google account and hand over the results.

Some additional details

- If you force the python project to run using the run-agent.bat script, the output should be a consistent, real-time, and detailed log from the spider. Scraped items will show up in the log one after another. A single report of an scraped item should look something similar to this:

```
2021-04-13 23:18:21 [scrapy.core.scraper] DEBUG: Scraped from <200
https://www.imdb.com/title/tt0848228/>
{'Cast': ['Robert Downey Jr.',
          'Chris Evans',
          'Mark Ruffalo',
          'Chris Hemsworth',
          'Scarlett Johansson',
          'Jeremy Renner',
          'Tom Hiddleston',
          'Clark Gregg',
          'Cobie Smulders',
          'Stellan Skarsgård',
          'Samuel L. Jackson',
          'Gwyneth Paltrow',
          'Paul Bettany',
          'Alexis Denisof',
          'Tina Benko'],
 'Id': 'tt0848228',
 'Name': 'The Avengers',
 'Year': '2012'}
```

- Running the run-agent.bat out of schedule would not complete the automation process by itself. You will need to run the apps script manually. In other words, the program scraps, processes, uploads, and then stops there!
- A copy of the apps script module's code is in a file named `Code.gs` in this path: `ProjectRoot\google-apps-script`
- You can see a sample form created by the program here: https://docs.google.com/forms/d/e/1FAIpQLSfsQr0jyNaQ7-IjHVuD0S5csSNM5ZQ1gaB6WQFWKi33sY_TNA/viewform