CS7NS1 Scalable Computing Coursework

Student Name: Vimanyu Kachroo
Student Number: 17311910
Strand: MSc computer science – data science

Problem statement:

To deliver a REST service, that provides an interface for submitting a set of GitHub repositories and to identify the set of GitHub developers who have contributed to any of these repositories in 2018, and for this user set, provide an ordering that ranks these users according to one of the following criteria:

1. The total number of commit contributions to any project to which a user has a contributed.
2. The total number of commit contributions as above but restricted to projects that are members of the original submitted set.
3. The number of known programming languages for each user.
4. The weekly commit rate of users (provide a weekly rank ordering) for the submitted project set, for 2018.
5. The average commit rate of each user to any project, for 2018.
6. The total number of collaborators in 2018 i.e. a count of other users who have contributed to any project that the user has contributed to).

Tools used:

Python – Used Flask library to implement rest service. I have used “render_templates” and “request” to interact with the created HTML pages, these two are part of the Flask library. I have also used the “nested_lookup” library to help search for keywords in nested JSON files. Also, used the “DateTime” library to convert the Unix (GitHub Default) date format to YYYY-MM-DD format.

Smtplib library in python is used to send the results to the user via mail. Smtplib utilizes Simple Mail Transfer Protocol to do this.

HTML – I have used HTML web pages to create an interface to interact with my rest service using a web browser.

Approach taken:

I started with accessing GitHub API using python. To start using the GitHub API I required a list of usernames and repository names from the user, to achieve this is created a simple input code block and stored the usernames and repository names in two different lists and later combined them to form a dictionary which is easier to convert into JSON and use for our rest service. Also, I created an empty dictionary wherever required to append/update the results after running one of the ranking criteria.

Next, I have used the "requests" module in python to send simple HTTP requests to the GitHub API and fetch data as required using the information in the stored dictionary and process it accordingly.

The following are the URLs used to fetch the data for the respective ranking criteria:

- Rank1

  https://api.github.com/users/"+ <username> +"/repos
  https://api.github.com/repos/"+ <username> +"/"+ <repository name> +"/commits?since=2018-01-01

To get the desired results here I used two GitHub API calls, the first call gets all the repository names for the given user and the second call gives us the list of commits made in the year 2018 for each of these repositories, if this results to zero that means the user has no commits for that repository in the year 2018 and isn't counted, and if the number of commits is not zero I add them towards the final result and order them accordingly.

- Rank2

  https://api.github.com/repos/+ <username> +/+ <repository name> +/commits?since=2018-01-01

  This ranking criterion is similar to the first one but in this case, we are only concerned about the submitted repository. The above URL gives us the information for all the commits done by the submitted user in the given repository for the year 2018.

- Rank3

  https://api.github.com/users/"+ <username> +"/repos"

  https://api.github.com/repos/+ <repository name> +/languages"

  For this ranking criterion, I have used two URLs to get the required data, in the first URL I fetch the list of all the repository names for a given username and store them in a list. I use this list to check if there are any commits for the year 2018, if it has at least one commit then I fetch the language list from it and save them in a dictionary according to their respective usernames.

- Rank4

  https://api.github.com/repos/"+ <username> +"/"+ <repo> +"/stats/commit_activity

  The data returned for this URL contains the weekly commit activity for the specified user's repository. It shows the number of commits done by the user in the last 52 weeks. The data pertaining to date and time is given in the Unix format and needs to be converted. I checked for at least one commit in a given week and if it was made in the year 2018, if the condition is satisfied I store the week, date and time along with the number of commits for that week.

- Rank5

  https://api.github.com/users/"+ <username> +"/repos

  https://api.github.com/repos/"+ <username> +"/"+ <repository name> +"/commits?since=2018-01-01

  Like previously I have used the first URL to get the list of all the repository names for a given username and the second URL to get commits for these repositories in the year 2018. To calculate the average commit rate for the given user in the year 2018, I checked for the repositories with at least one commit in the year 2018 and count the number of commits, to keep track of the repositories with commits in 2018 I kept a separate counter and finally I used It to calculate the mathematical average using the total number of commits and the repo counter (total no. of commits divided by total no. of repositories committed in 2018).

- Rank6

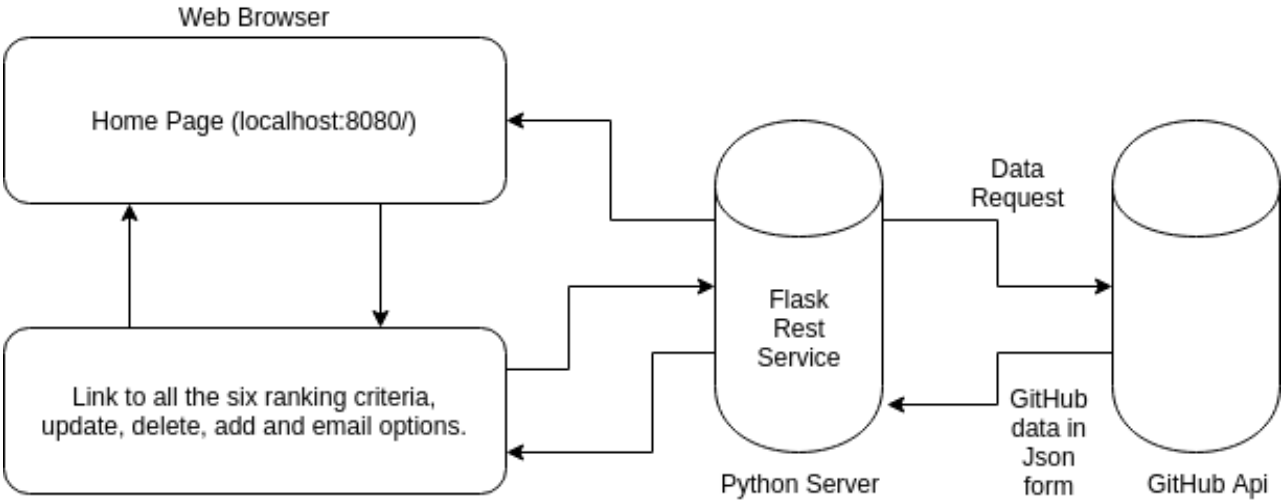
  https://api.github.com/repos/"+ <username> +"/"+ <repository name> +"/contributors

  After fetching all the repository names for a given user and checking them for any commits made in the year 2018. If the conditions are satisfied I look for the number of contributors for that specific repository since in each repository user itself is considered as a contributor, therefore I used an offset counter to get the correct output.

If the user wants to send the results to any specific email ID he or she can use the email option which leads to an HTML form where they will enter their email login credentials and the receiver email ID. Further, I have created HTML templates to implement update, delete, add new entries and to show the results on a web browser.

Network architecture

The below diagram shows the work flow of the rest service created. The user interacts with the rest service through a web browser (we can start the server by entering zero username/repository name and add username/repository name through the browser). Based on user selection the rest service accesses the GitHub API and gets the data which is then further used to get the required output and displayed on the web browser.



How to use the service

You can clone the bellow repository to your system.

Link: https://github.com/VimanyuK/CS7NS1-RestServiceAPI

Please make sure the file “RestService.py” and the “templates” folder are in the same directory at the time of execution.

To start the service, please run the RestService.py file. When you run the service, you'll get a console prompt in the console asking you the number of users you want to enter (enter the number of users you want to work with). Then in the consecutive prompts, you will be asked to enter the usernames and their corresponding repository names (you can enter more or edit these through the HTML interface in your browser).

The python local server will run on port 8080. You can go to http://localhost:8080 on your browser and access the service.

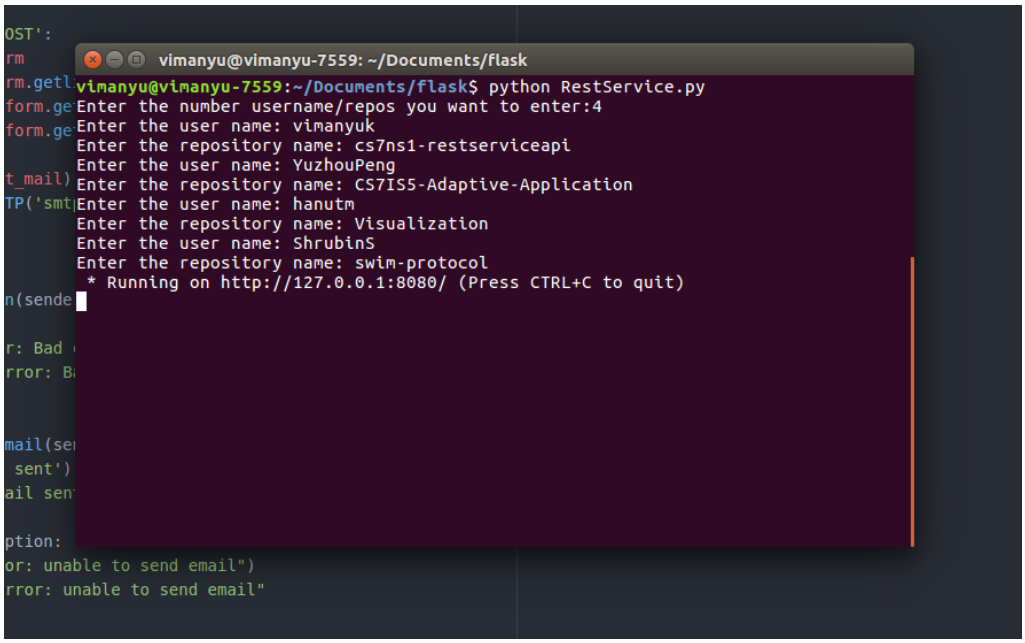
(The main.py file is not required to run the service at any time, it just contains the initial attempts to access the GitHub API and work with the data to get the desired outputs.)

Output

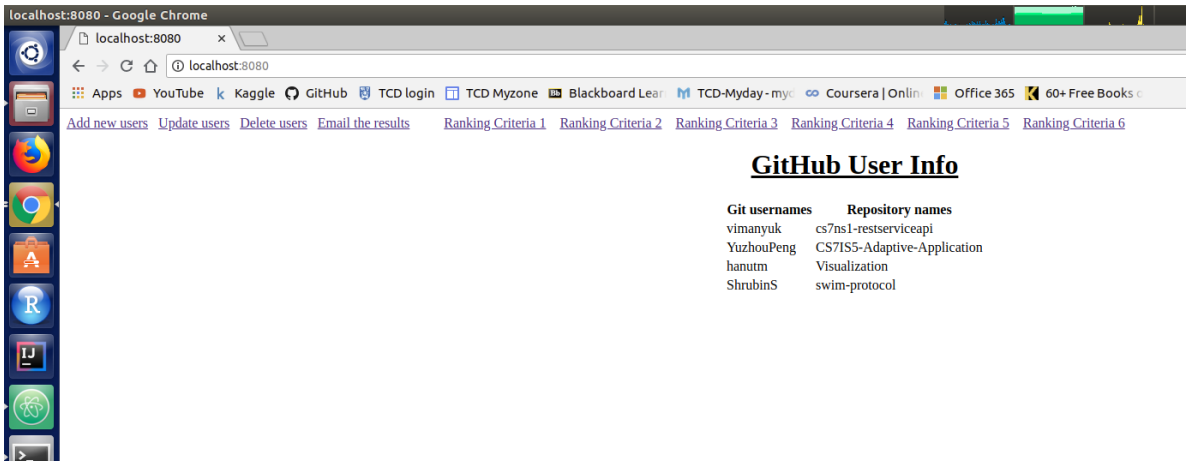
Following are the results of the rest service with 5 usernames and their repositories:

The first step is to run our RestService.py file, the screenshot shows its execution file which gives us the option to enter the number of users we want to work with. Here I have given four and provided their GitHub usernames and repository names.

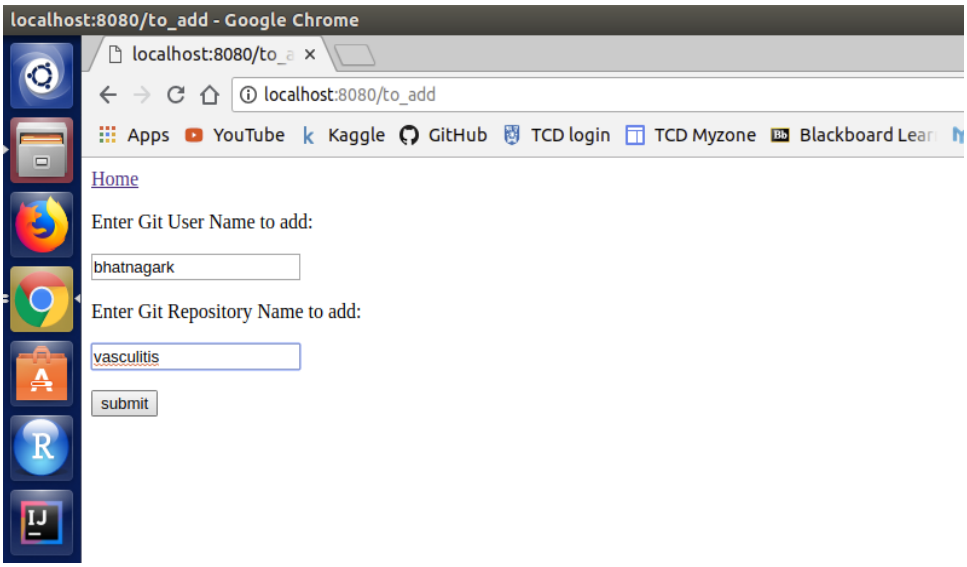
You can enter zero and proceed ahead by pressing enter if you wish to enter the username/repository name through the web interface on localhost:8080.

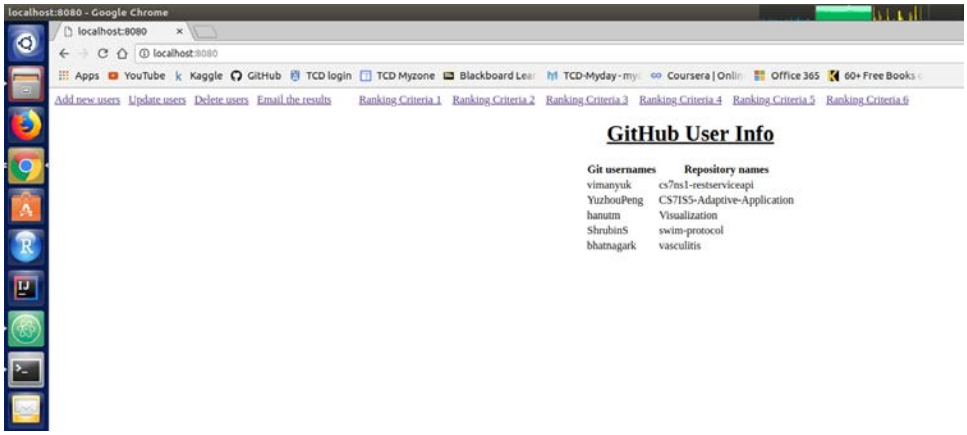


After running our server, open a web browser (tested on google chrome) and enter localhost:8080 in the address bar to get the following view. Here we can see the entered usernames and repository names along with options to add, update, delete and email results options.

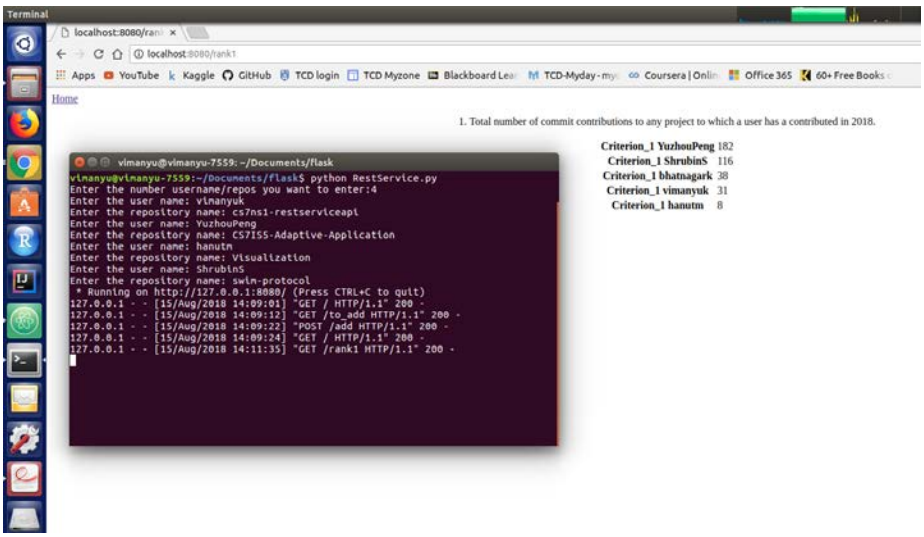


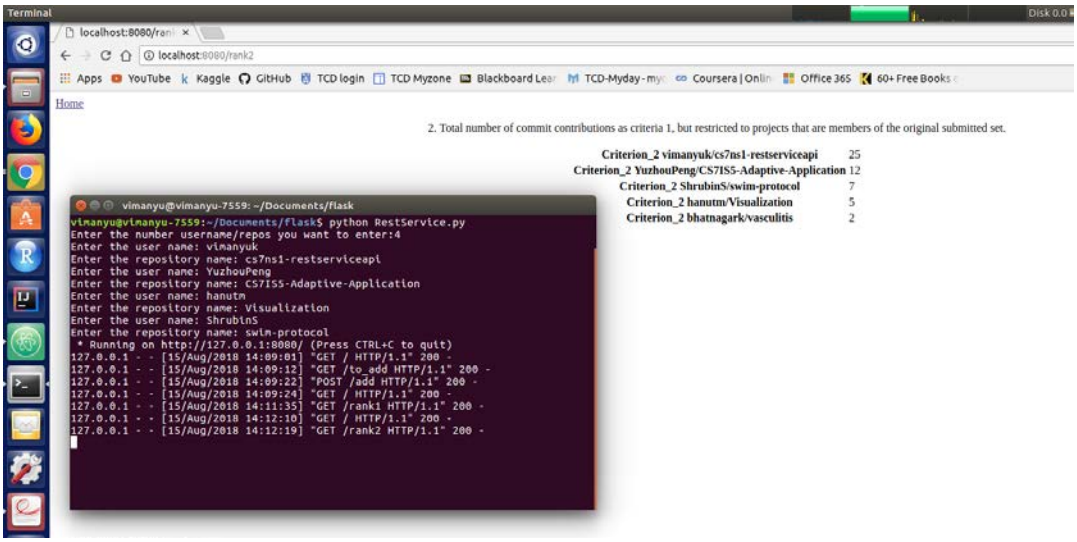
The below screenshot shows the link to add a new user and repo name.

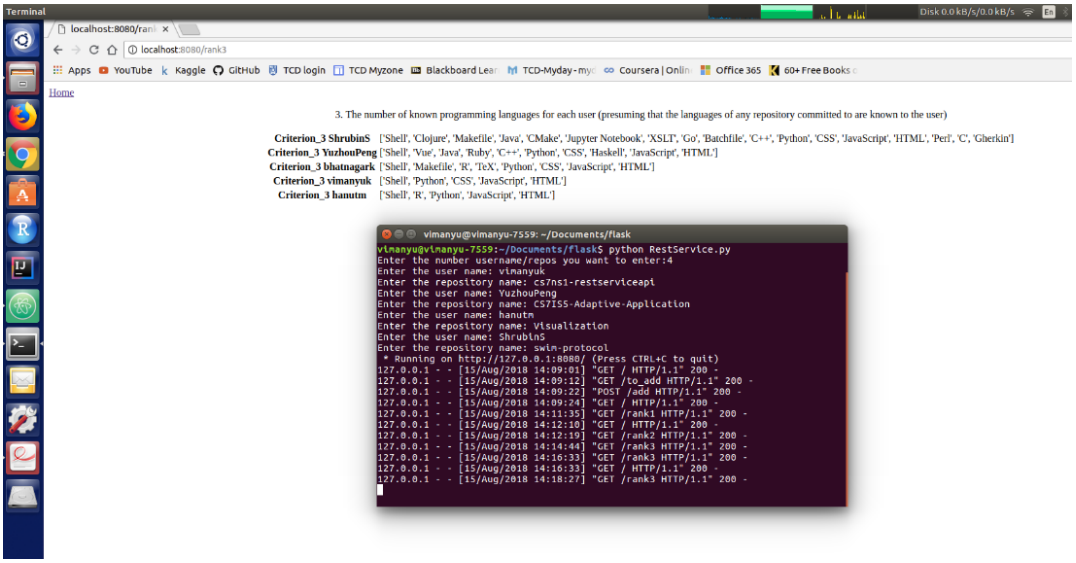


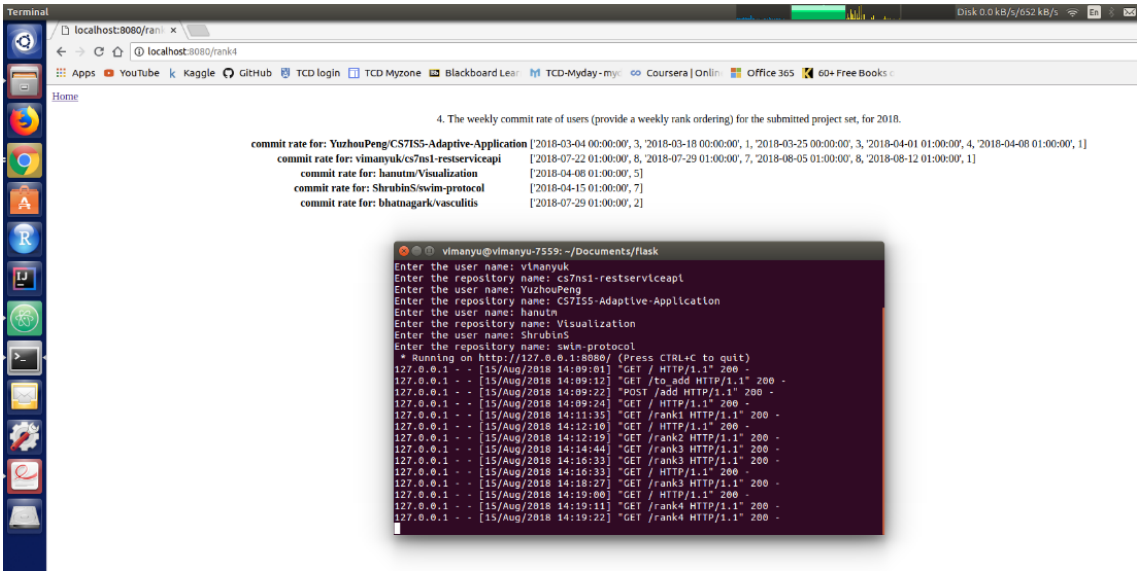


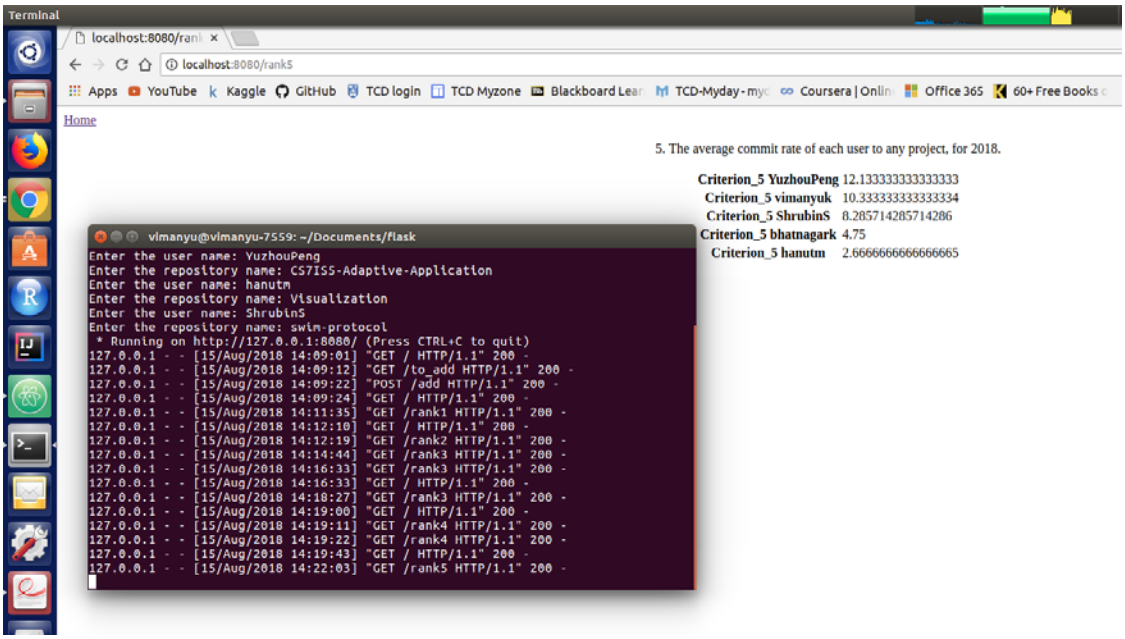
The next six screenshots show the results along with the console output for all the six ranking criteria mentioned above.

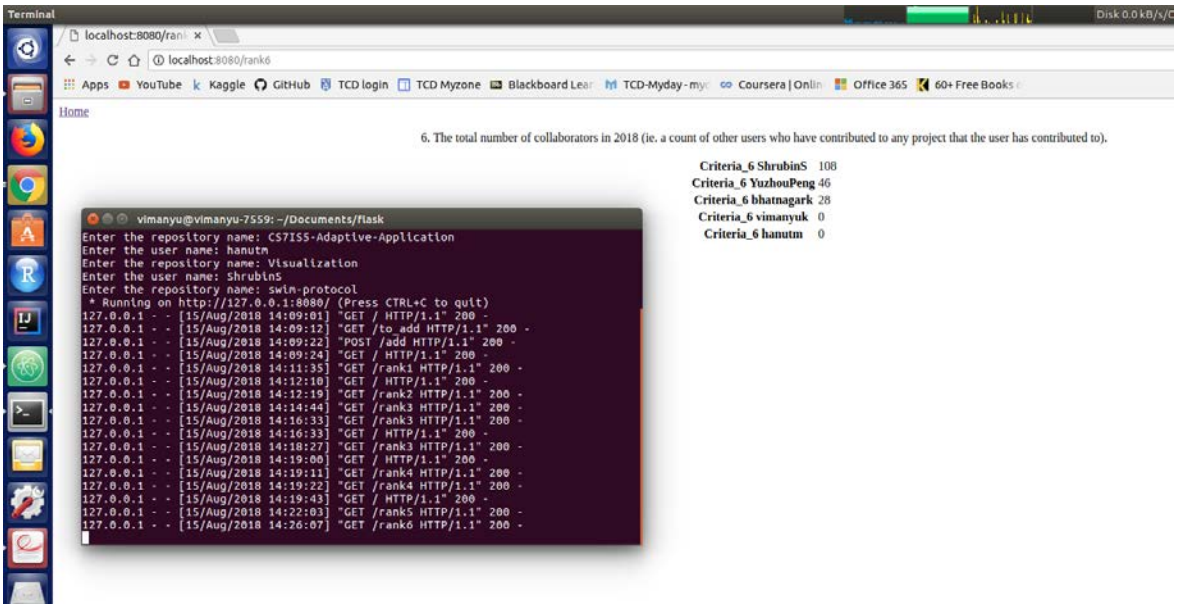



Terminal
localhost:8080/rank2
Apps YouTube Kaggle GitHub TCD login TCD Myzone Blackboard Lear TCD-Myday-myc Coursera | Onlin Office 365 60+ Free Books
Home
2. Total number of commit contributions as criteria 1, but restricted to projects that are members of the original submitted set.
Criterion_2 vimanyuk/cs7ns1-restserviceapi 25
Criterion_2 YuzhouPeng/CS7IS5-Adaptive-Application 12
Criterion_2 ShrubinS/swim-protocol 7
Criterion_2 hanutm/Visualization 5
Criterion_2 bhatnagark/vasculitis 2
vimanyu@vimanyu-7559: ~/Documents/flask
vimanyu@vimanyu-7559:~/Documents/flask$ python RestService.py
Enter the number username/repos you want to enter:4
Enter the user name: vimanyuk
Enter the repository name: cs7ns1-restserviceapi
Enter the user name: YuzhouPeng
Enter the repository name: CS7IS5-Adaptive-Application
Enter the user name: hanutm
Enter the repository name: Visualization
Enter the user name: ShrubinS
Enter the repository name: swim-protocol
* Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
127.0.0.1 - - [15/Aug/2018 14:09:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:12] "GET /to_add HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:22] "POST /add HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:24] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:11:35] "GET /rank1 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:12:10] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:12:19] "GET /rank2 HTTP/1.1" 200 -

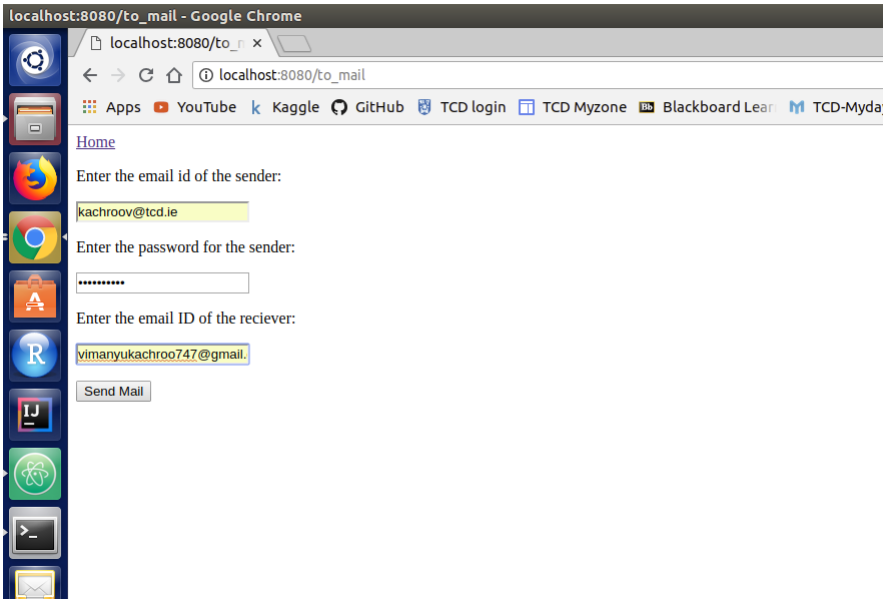

Terminal
localhost:8080/rank3
Apps YouTube Kaggle GitHub TCD login TCD Myzone Blackboard Lear TCD-Myday-myc Coursera | Onlin Office 365 60+ Free Books
Home
3. The number of known programming languages for each user (presuming that the languages of any repository committed to are known to the user)
Criterion_3 ShrubinS ['Shell', 'Clojure', 'Makefile', 'Java', 'CMake', 'Jupyter Notebook', 'XSLT', 'Go', 'Batchfile', 'C++', 'Python', 'CSS', 'JavaScript', 'HTML', 'Perl', 'C', 'Gherkin']
Criterion_3 YuzhouPeng ['Shell', 'Vue', 'Java', 'Ruby', 'C++', 'Python', 'CSS', 'Haskell', 'JavaScript', 'HTML']
Criterion_3 bhatnagark ['Shell', 'Makefile', 'R', 'TeX', 'Python', 'CSS', 'JavaScript', 'HTML']
Criterion_3 vimanyuk ['Shell', 'Python', 'CSS', 'JavaScript', 'HTML']
Criterion_3 hanutm ['Shell', 'R', 'Python', 'JavaScript', 'HTML']
vimanyu@vimanyu-7559: ~/Documents/flask
vimanyu@vimanyu-7559:~/Documents/flask$ python RestService.py
Enter the number username/repos you want to enter:4
Enter the user name: vimanyuk
Enter the repository name: cs7ns1-restserviceapi
Enter the user name: YuzhouPeng
Enter the repository name: CS7IS5-Adaptive-Application
Enter the user name: hanutm
Enter the repository name: Visualization
Enter the user name: ShrubinS
Enter the repository name: swim-protocol
* Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
127.0.0.1 - - [15/Aug/2018 14:09:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:12] "GET /to_add HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:22] "POST /add HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:24] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:11:35] "GET /rank1 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:12:10] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:12:19] "GET /rank2 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:14:44] "GET /rank3 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:16:33] "GET /rank3 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:16:33] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:18:27] "GET /rank3 HTTP/1.1" 200 -


Terminal
localhost:8080/rank4
Apps YouTube Kaggle GitHub TCD login TCD Myzone Blackboard Lear TCD-Myday-myc Coursera | Onlin Office 365 60+ Free Books
Home
4. The weekly commit rate of users (provide a weekly rank ordering) for the submitted project set, for 2018.
commit rate for: YuzhouPeng/CS7IS5-Adaptive-Application ['2018-03-04 00:00:00', 3, '2018-03-18 00:00:00', 1, '2018-03-25 00:00:00', 3, '2018-04-01 01:00:00', 4, '2018-04-08 01:00:00', 1]
commit rate for: vimanyuk/cs7ns1-restserviceapi ['2018-07-22 01:00:00', 8, '2018-07-29 01:00:00', 7, '2018-08-05 01:00:00', 8, '2018-08-12 01:00:00', 1]
commit rate for: hanutm/Visualization ['2018-04-08 01:00:00', 5]
commit rate for: ShrubinS/swim-protocol ['2018-04-15 01:00:00', 7]
commit rate for: bhatnagark/vasculitis ['2018-07-29 01:00:00', 2]
vimanyu@vimanyu-7559: ~/Documents/flask
Enter the user name: vimanyuk
Enter the repository name: cs7ns1-restserviceapi
Enter the user name: YuzhouPeng
Enter the repository name: CS7IS5-Adaptive-Application
Enter the user name: hanutm
Enter the repository name: Visualization
Enter the user name: ShrubinS
Enter the repository name: swim-protocol
* Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
127.0.0.1 - - [15/Aug/2018 14:09:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:12] "GET /to_add HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:22] "POST /add HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:24] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:11:35] "GET /rank1 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:12:10] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:12:19] "GET /rank2 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:14:44] "GET /rank3 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:16:33] "GET /rank3 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:16:33] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:18:27] "GET /rank3 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:19:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:19:11] "GET /rank4 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:19:22] "GET /rank4 HTTP/1.1" 200 -


Terminal
localhost:8080/rank5
Apps YouTube Kaggle GitHub TCD login TCD Myzone Blackboard Lear TCD-Myday - myc Coursera | Onlin Office 365 60+ Free Books
Home
5. The average commit rate of each user to any project, for 2018.
Criterion_5 YuzhouPeng 12.133333333333333
Criterion_5 vimanyuk 10.333333333333334
Criterion_5 ShrubinS 8.285714285714286
Criterion_5 bhatnagark 4.75
Criterion_5 hanutm 2.6666666666666665
vimanyu@vimanyu-7559: ~/Documents/flask
Enter the user name: YuzhouPeng
Enter the repository name: CS7ISS-Adaptive-Application
Enter the user name: hanutm
Enter the repository name: Visualization
Enter the user name: ShrubinS
Enter the repository name: swim-protocol
* Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
127.0.0.1 - - [15/Aug/2018 14:09:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:12] "GET /to_add HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:22] "POST /add HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:24] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:11:35] "GET /rank1 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:12:10] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:12:19] "GET /rank2 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:14:44] "GET /rank3 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:16:33] "GET /rank3 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:16:33] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:18:27] "GET /rank3 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:19:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:19:11] "GET /rank4 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:19:22] "GET /rank4 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:19:43] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:22:03] "GET /rank5 HTTP/1.1" 200 -


Terminal
localhost:8080/rank6
Apps YouTube Kaggle GitHub TCD login TCD Myzone Blackboard Lear TCD-Myday - myc Coursera | Onlin Office 365 60+ Free Books
Home
6. The total number of collaborators in 2018 (ie. a count of other users who have contributed to any project that the user has contributed to).
Criteria_6 ShrubinS 108
Criteria_6 YuzhouPeng 46
Criteria_6 bhatnagark 28
Criteria_6 vimanyuk 0
Criteria_6 hanutm 0
vimanyu@vimanyu-7559: ~/Documents/flask
Enter the repository name: CS7ISS-Adaptive-Application
Enter the user name: hanutm
Enter the repository name: Visualization
Enter the user name: ShrubinS
Enter the repository name: swim-protocol
* Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
127.0.0.1 - - [15/Aug/2018 14:09:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:12] "GET /to_add HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:22] "POST /add HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:09:24] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:11:35] "GET /rank1 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:12:10] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:12:19] "GET /rank2 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:14:44] "GET /rank3 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:16:33] "GET /rank3 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:16:33] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:18:27] "GET /rank3 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:19:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:19:11] "GET /rank4 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:19:22] "GET /rank4 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:19:43] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:22:03] "GET /rank5 HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2018 14:26:07] "GET /rank6 HTTP/1.1" 200 -

After getting all the results if the user wants to email the results, they can click the email the results option to get the bellow view where they can enter their login credentials and receivers email ID.



The below screenshot shows the email received by the mentioned receiver.

The execution time depends on the number of users we are working with.

References

GitHub API documentation: https://developer.github.com/v3/

Flask documentation: http://flask.pocoo.org/docs/1.0/

END