Convert.py Process (Part 1)

1. Open the copied log file “radsecproxy.log\_\*previous day date\*”
2. Save all the information from log file into a variable logData, a list of log entries in the file and close the file.
3. Load the ihlconfig.json file into the program. The json file contains the IP addresses and Server names for all the IHLs and the Euro TLR servers.
4. Initialise all the IHLs into a dictionary IHL\_Array to facilitate the log extraction process.
5. Load uniqueUsers.log from subdirectory “uniqueUsersFiles” for each month, year and each IHL (done through IHL.py method readUniqueUserFiles)
   1. Os.path.isfile checks whether the specific uniqueUsers.log is a regular file and not a directory.
      1. If true, open the file for reading and create uniqueRecords as a set of existing unique users to check with the extracted log file.
      2. If false, open the file for writing the new set of unique users. uniqueRecords will be an empty set.
   2. Close each uniqueUsers.log file
6. Run Log Extract.For each entry in logData, use regex to search for ‘Access-Accept/Reject for user’
   1. re.search (r’Access-Accept for user’, line, re.MULTILINE|re.IGNORECASE)
   2. If Access-Reject, matchReject is set.
      1. If user not in daily\_RejectedRecords, add user to it.
      2. Check where user is coming from (Identity Provider) and where he is going to (Service Provider/Access Point)
      3. Keep count of rejected users, whether local or overseas, for each IHL.
   3. If Access-Accept, matchAccept is set
      1. If user not in daily\_AcceptRecords, add user to it.
      2. Check where user is coming from (Identity Provider) and where he is going to (Service Provider/Access Point)
      3. Keep count of localusers overseas, in different IHLs, for each IHL they come from through the variable IHL\_Array[ihl].localUserCount.
      4. Keep count of overseas visitors accessing the network for each IHL.
7. Total the number of rejected localUsers and visitors from each IHL in IHL\_Array[ihl].reject\_localUsers and IHL\_Array[ihl].reject\_visitors.
8. Total the number of localusers in different IHLs from each IHL in IHL\_Array[ihl].localUsersCount.
9. Open the uniqueUsers.log file for each month, year, each IHL, reject/accept.
   1. For each user in the uniqueRecords, count the number of unique users for the IHL.
   2. Write back the unique user names in uniqueUsersFile.
10. Create a log file “results.log\_\*date\*” to store the total number of localUsers from each IHL and the total number of visitors to each IHL.
11. Results content should contain
    1. Total number of localUsers from NTU/NUS/SMU/ASTAR/NIE who are abroad and in other IHLs. Also total number of localUsers from each IHL in total.
    2. Total number of visitors to NTU/NUS/SMU/ASTAR/NIE.
    3. Total number of unique/rejectUnique users to each IHL for the month.
    4. Total number of unique/rejectUnique users to each IHL for the year.
    5. Total number of rejected accesses from each IHL for the day.

Convert.py Process (Part 2: Saving to Daily,Monthly,Yearly CSV files)

1. Open the csv file to extract the present list of usage statistics into the local list variable “csv\_list”.
2. If the csv file is not found, create a new csv file and close it. Add in the first row of labels in the list “csv\_list”
   1. If file is daily, add [“Date”,”IHL”,”Users”,”Category”] as labels. Categories are LocalUsers, Visitors and Rejected.
   2. If file is monthly, add [“Month”,”IHL”,”UniqueUsers”,”Category”] as labels. Categories are Accepted and Rejected.
   3. If file is yearly, add [“Year”,”IHL”,”UniqueUsers”,”Category”] as labels. Categories are Accepted and Rejected.
3. Check for duplicate entries before appending the stats to the list “csv\_list” by using the first item of the last row as reference. Set it as the variable “last\_checked”.
   1. For daily file, check if the current date is the same as “last\_checked” and “last\_checked” is not the label “Date”. If true, we filter out the duplicate data and append the new data to the list.
   2. For monthly file, check if the current month is the same as “last\_checked” and “last\_checked” is not the label “Month”. If true, we filter out the duplicate data and append the new data to the list.
   3. For yearly file, check if the current year is the same as “last\_checked” and “last\_checked” is not the label “Year”. If true, we filter out the duplicate data and append the new data to the list.
4. After the duplicate check, we can append the new statistics values to the list “csv\_list”
   1. For daily file, append the numbers of localUsers, Visitors and Rejected Users as each row for each IHL to csv\_list.
   2. For monthly file, append the numbers of Accepted UniqueUsers for the month and the numbers of Rejected UniqueUsers for the month as rows, for each IHL to csv\_list.
   3. For yearly file, append the numbers of Accepted UniqueUsers for the year and the numbers of Rejected UniqueUsers for the year as rows, for each IHL to csv\_list.
5. Write back the contents of csv\_list to the csv files. (File format is DailyMay2015.csv, Monthly2015.csv and Yearly.csv).

CreateHTML.py Process (Data Visualisation on the singaren stats webpages for each ihl)

1. Set the previous date and the filepath to ensure the right csv files are referenced by the data visualisation tool. ( filepath=’html files/’ by default)
2. Get the names of the IHLs involved from ihlconfig.json and store them in the variable ihlNames.
3. Open the html files for each IHL and overwrite those files with the new HTML code produced by the createHTML() function.
4. In the createHTML() function run for each IHL,
   1. Insert source links to external libraries used for drawing the graphs
      1. <https://cdnjs.cloudflare.com/ajax/libs/d3/3.5.5/d3.min.js> for D3 dependency
      2. <http://dimplejs.org/dist/dimple.v2.1.3.min.js> for dimple library
   2. Create daily graph for the current month with the combination of Line plot and Scatter plot to show a line graph with clear indicated values. Data-source is obtained from the Daily CSV file.
   3. Create monthly graph for the current year with a stacked bar plot graph showing Accepted and Rejected UniqueUsers. Data-source is obtained from Monthly CSV file.
   4. Create yearly graph for the IHL with a stacked bar plot graph showing Accepted and Rejected UniqueUsers. Data-source is obtained from Yearly CSV file.
   5. Return the whole HTML code as a string.
5. Write and save the HTML code string to the html files for all the IHLs.

convertSingaren.py

1. Open the copied log file “radsecproxy.log\_\*previous day date\*”
2. Save all the information from log file into a variable logData, a list of log entries in the file.and close the file.
3. Initialise the class serverLoad with its class variables, accepts and rejects, to keep track of the number of accepted/rejected requests hourly.
4. Run Log Extract.For each entry in logData, use regex to search for ‘Access-Accept/Reject for user’
   1. re.search (r’Access-Accept for user’, line, re.MULTILINE|re.IGNORECASE)
   2. If Access-Reject, matchReject is set.
      1. The variable rejects is incremented for the specific hour.
   3. If Access-Accept, matchAccept is set
      1. The variable accepts is incremented for the specific hour.
5. Open the csv file to extract the present list of usage statistics into the local list variable “csvlist”.
6. If the csv file is not found, create a new csv file and close it. Add in the first row of labels in the list “csvlist”, ["Date","Month","Hour","Requests","Category"].
   1. Check for duplicate entries before appending the stats to the list “csvlist” by using the first item of the last row as reference. Set it as the variable “last\_checked”. check if the current date is the same as “last\_checked” and “last\_checked” is not the label “Date”. If true, we filter out the duplicate data and append the new data to the list.
7. After the duplicate check, we can append the new statistics values to the list “csv\_list”
   1. Append the number of requests for each hour of the day as each row to csv\_list.
8. Write back the contents of csv\_list to the csv files. (File format is ServerLoad<year>.csv).

CreateHTML\_singaren.py Process (Data Visualisation on the singaren stats webpages for total.html)

1. Set the previous date and the filepath to ensure the right csv files are referenced by the data visualisation tool. ( filepath=’html files/’ by default)
2. Get the names of the IHLs involved from ihlconfig.json and store them in the variable ihlNames.
3. Open the total.html file and overwrite it with the new HTML code produced by the createSingarenHTML() function.
4. In the createSingarenHTML() function run for each IHL,
   1. Insert source links to external libraries used for drawing the graphs
      1. <https://cdnjs.cloudflare.com/ajax/libs/d3/3.5.5/d3.min.js> for D3 dependency
      2. <http://dimplejs.org/dist/dimple.v2.1.3.min.js> for dimple library
   2. Create hourly graph for the current month with the area plot to show a line graph with clear indicated values. Data-source is obtained from the ServerLoad CSV file.
   3. Create daily graph for the current year with the areaplot graph showing Accepted and Rejected requests. Data-source is obtained from ServerLoad CSV file.
   4. Create monthlly graph for the IHL with a stacked bar plot graph showing Accepted and Rejected requests. Data-source is obtained from ServerLoad CSV file.
   5. Return the whole HTML code as a string.
5. Write and save the HTML code string to the html files for total.html..