**Project Overview**

The Eduroam statistics project is divided into two sections, namely the logfile extraction program and the data visualisation tool. The log file extraction program extracts and processes the data from the eduroam log file obtained from the Eduroam server. The data visualisation tool utilises the extracted data from the previous program to display the usage statistics in the form of graphs on the SingAren website for eduroam members to inspect the usage data.

**Part 1: Log File Extraction**

The log file extraction program, named as “convert.py”, takes in the Eduroam log file of the previous day as input, and returns the UniqueUsers files for each IHL and the extracted data containing the number of users using the Eduroam system as outputs. The contents of the log file is first extracted and saved into a list of log entries for processing in the convert.py script. The UniqueUsers files which contain the lists of accepted/rejected unique users using eduroam for each IHL during the particular month/year, are also extracted and saved into the respective variables.

Next, the script sorts each entry according to whether the user access is accepted or rejected. If access is rejected for the specific user, the script takes note of the source of the user access request, and then adds the user to RejectUniqueUsers files of the user’s home institution if it is not present in the file previously. The number of unique users for the certain IHL will also be collated. If access is accepted for the user, the script takes note of the entry and exit points of the user access request, and then adds the user to UniqueUsers files of the user’s home institution if it is not in the file before the running of the script. The number of unique users for the certain IHL will be collected, together with the number of unique users from overseas using eduroam at each IHL.  The extraction process will end after all the log entries have been checked by the script.

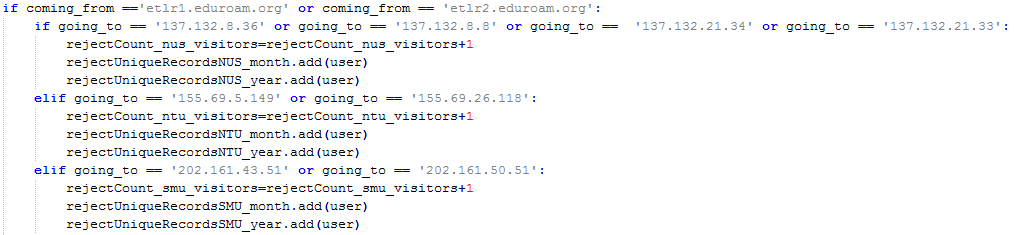
The extracted data will then be written into a daily results text file to summarise the usage statistics obtained in the operation. The extracted data will also be saved into a database as storage such that the visualisation tool can use it as a data source for displaying the graphs.

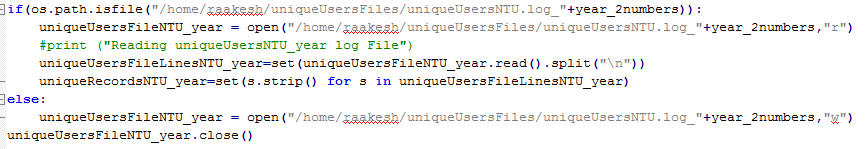
**Part 2: Data Visualisation Tool**

The data visualisation tool is represented by CreateHTML.py, which produces the HTML files for each IHL present in the eduroam system. The HTML files act as a container for structuring the graphs in a neat fashion. The graphs themselves are created using a javascript charting library and they utilise the database as a data source for displaying. The HTML files are then put on the web server for display as part of the statistics webpages of the SingAren website.

**Revamp of Log Extraction Component**

**Problem**

The existing Python code of the Log file extraction program was written in a procedural format, such that there are instances of spaghetti code whereby there are many if-statements for log file data extraction for each IHL without any proper segmentation. At the same time, the details of each IHL are hard-coded in the program, making maintenance difficult as the whole program has to be taken offline to perform changes. An example of this issue can be seen in the code excerpt below. 

Multiple instances of repeated logic are detected since the code has to perform the same actions for each IHL. For example, the code for opening the uniqueUserFiles for every IHL shown below is replicated frequently for each IHL and time period, namely monthly and yearly.

There is heavy reliance of global variables in the code and proper structuring of code is non-existent with the lack of functions. This makes the code hard to test for validity as there are several different portions in the conversion program such that it would be difficult to pinpoint the exact location of the error. Adding new functionalities would also be tedious due to the various dependencies between each component of the programs.

In conclusion, the existing code is unable to handle the expected changes such as the addition of new IHLs to the statistics system without needing a significant amount of code to be edited accordingly.

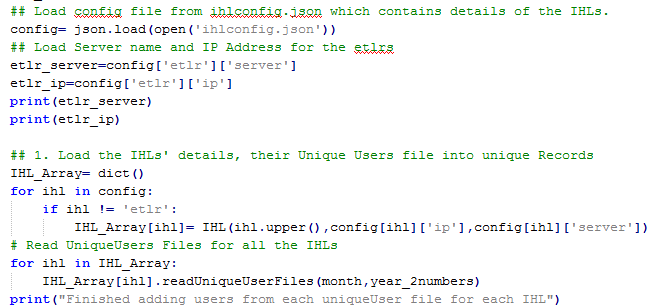
**Design & Implementation**

In lieu of the new requirements and the problems of the legacy code, there is a need for redesigning the structure of the program in order to fulfil these conditions.

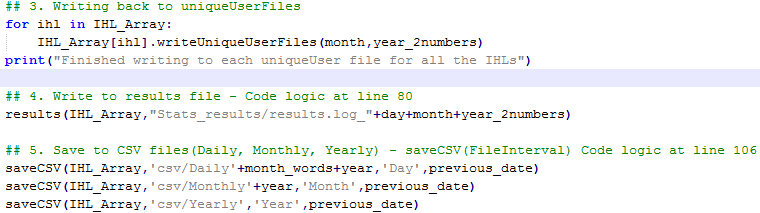
The conversion program is divided into 4 components: main(), logExtract(), results() and saveCSV(). The main() module is in charge of coordinating the different components and direct the control flow of the program. The logExtract() module extracts the required data from the contents of the log file into local variables and the results() saves the extracted data in a daily results text file. The saveCSV() module stores the same extracted data into CSV files that would be used for the data source of the displayed graphs.

IHL-specific operations such as the reading and writing of uniqueUserFiles are consolidated under the class IHL. The IHL class also contains variables to assist in the gathering of related statistics through the log extraction. This design simplifies the process of repeated file I/O operations and the sorting of data into the respective IHLs.

**Main() module**

The main() module first opens the previous day’s log file and stores its contents into a list variable logData. It also loads the configuration file ihlconfig.json which contains the server names and ip addresses of every IHL and the ETLR servers which indicates overseas users. These details are inserted dynamically into the log extraction system as shown in the code excerpt below.

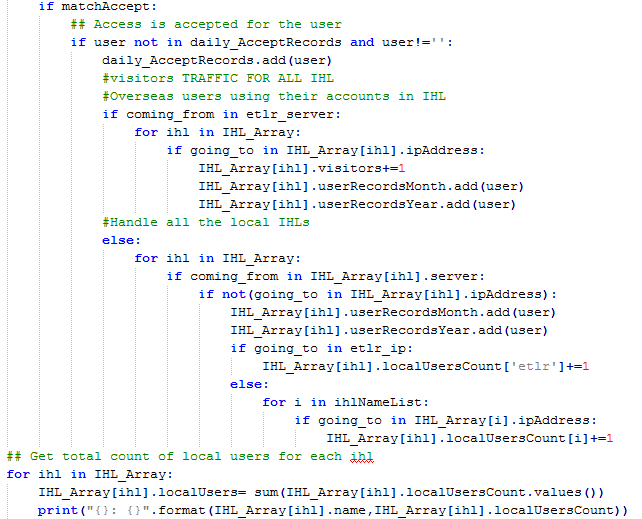
The module then calls logExtract() to process logData and IHL\_Array to contain the extracted statistics. Next, it proceeds to write all the unique user files for all the IHLs. Finally, it proceeds to call the functions results() and saveCSV() to save the data in a portable format as illustrated below.



This method of structuring eases the process of tracking the code logic during debugging sessions.

**LogExtract()**

The module logExtract() takes in the arguments logData, IHL\_Array and the details of the ETLRs. It uses the Python in-built regex library to sort the entries in logData for accepted and rejected entries. The source and destination of each authentication request is then extracted and added to the statistics. The process can be seen in the following code.



One key improvement in the log extraction component is the usage of the dictionary variable “localUsersCount” to indicate the users from the specific IHL who are accessing eduroam at the other IHLs and overseas. E.g. IHL\_Array[ihl].localUsersCount[‘etlr’] refers to the users who using eduroam overseas. The variable “localUsersCount” reduces the amount of variables needed in total to store the data and hence makes the code less prone to errors.

**Results()**

The results() module writes the summary of extracted data obtained from the log file for the day. It uses IHL\_Array as an argument to save the results into an text file for references purpose. The contents of the results file are as follows:

1. Total number of localUsers from each IHL who are abroad and in other IHLs.
2. Total number of localUsers from each IHL in total.
3. Total number of visitors to each IHL.
4. Total number of unique/rejectUnique users to each IHL for the month.
5. Total number of unique/rejectUnique users to each IHL for the year.
6. Total number of rejected accesses from each IHL for the day.

**SaveCSV()**

The saveCSV() module is a crucial component of the log processing system as it bridges the gap between the Python-based log file extraction program and the web-based data visualisation tool. Since it is a new component, more time was needed to ensure the seamless operation of the module with testing using mock data. Three CSV files were used for the three graphs of different intervals, daily, monthly and yearly,  and hence the module is called three times by the main() method with different parameters.

The format in which the data is arranged was also important since the charting library used had a strict requirement in terms of the organisation of the data. The saveCSV() module utilises the Python in-built csv library to handle the reading and writing of the CSV files. Since there is a difference in the definition of newlines for Windows and Linux systems which affect file I/O, I decided to let Python handle the differences automatically through its usage of universal newlines and not explicitly define the newline manually.

**Data Visualisation Component**

**Problem**

The old data visualisation tool implemented by the previous engineer, upon initial inspection, was not operational since the statistics has not been updated since January 2015. The problem was found to be in the change of authentication details for the Google account associated with the statistics program, and the online Google spreadsheets are not updated. Since the charting library for the old visualisation tool, Google Charts, was dependant on the online spreadsheets as a data source, therefore the graphs on the statistics were not up-to-date.

In addition, the Google Data API used in the legacy code has been deemed deprecated and it may not work in the future. This meant that the Google Charts API was not a viable solution with the present situation. Hence, a new charting library and a new data source have to be used.

The existing code also delegates one html creation python script for each IHL, which is inefficient since the logic behind the scripts is the same for all IHLs and the approach is not scalable if many IHLs get involved in the Eduroam statistics.

**Design & Implementation**

The data visualisation process has to be redesigned from scratch in order to create accurate graphs on the webpages. In addition to the saveCSV() module in convert.py, an external charting library was used in place of the preceding Google Chart API.

Dimple.js, an external Javascript charting library by Align Alytics, was used for creating the graphs on the statistics web pages at the SingAren website. Described as an object-oriented API designed for business analytics, Dimple.js allows the creation of flexible axis-based graphs with minimal code using d3js, a popular Javascript library that performs DOM manipulation driven by data. The advantage of using Dimple rather than d3js by itself lies within the gentler learning curve and the various features offered by Dimple such as chart templates compared to the low-level approach of pure d3js. Furthermore, Dimple also allows exposure of core d3js components for customization such as adding the title to the chart.

The creation of the daily, monthly and yearly charts follow a similar method. First, the CSV file is obtained through a HTTP GET request encapsulated by d3js’ parser method. Next, the data is filtered according to IHL such that the correct data will be displayed for the IHL’s statistics page. The graph is then created by adding the code determining the type of x and y axis, and the series of data that is needed to be displayed.

Only one html generation script, CreateHTML.py, is needed for creating the web pages of all the IHLs. The script also depends on ihlconfig.json to determine the names and the amount of web pages needed to be created. The html code is encoded within a multi-line string and written to the respective files as illustrated in the following snippet.

