eVisit Technical Challenge

Amudhan Manisekaran

Coding Approach

- How does your code work?
  I have implemented a class '**IPTracker'** with four functions.
  - **Constructor:** This function initializes the cache that will store IP address along with the request count. I am using a defaultdict for this purpose.
  - **request_handled(ip_address):** This function handles the request made by IP addresses and updates the count in the cache based on the request made.
  - **top100():** This function returns the top 100 IP addresses by request count, with the highest traffic IP address first.
  - **clear():** This function clears all the IP addresses and tallies

- Why did you choose this approach?
  I chose the data structure for the IP address cache as a **Dictionary** in **Python** because it can store data values like a map(key-value pairs). I chose the **defaultdict** subclass of the dictionary as it provides a default value for the keys that do not exist. Thereby eliminating the chances of a Key error.

  To find the top 100 IP addresses, I use the **nlargest** function. This is an inbuilt function in heapq library that performs the task of sorting the cache based on the count. I chose this function because we do not need the count of the addresses to be returned.

- What other approaches did you decide not to pursue?
  This cache implementation can also be done by using lists in Python. In that case, we would have to maintain two lists, one for the IP addresses and one for the respective request count.
  It would take more time to insert, update as well as retrieve data values from a list when compared to dictionaries. Also, creating two lists would use up extra space.

- What is the runtime complexity of each function?
  To maintain the dictionary, all operations (request_handled, clear) be done in constant time. **O(1)**
  To retrieve the top 100 addresses, it would be done in **O(n log n)** where n is the number of unique entries in the cache.

- How would you test this?
  I have included the test code (**Test.py**) which performs the unit testing on the implemented functions.