HIGH LEVEL DESIGN

DATE: 25th May, 2022

|  |  |
| --- | --- |
| Version | 1.0 |
| Author | Pardeep |

**Scope of Document:**

This document outlines the high level design of a system which is reading data from a file and checking if line is having IPv4/IPv6 or invalid IP. It highlights the high level flows / Use cases for the system and design of components.

**System Overview:**

This system is taking some input file and reading that file line by line. It is checking if line is representing IPv4 address, IPv6 address or it is invalid string. Based on the type of data in line it increases count for respective type. Then finally, it will display total IPv4, IPv6 and Invalid address along with unique IPv4/IPv6 address counts.

**Requirement of HLD:**

This HLD documents represents designing of a system which is having some requirement as per below section.

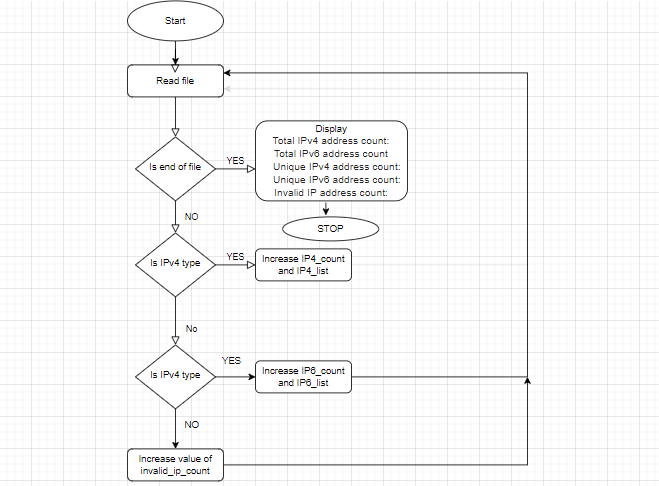
Input:  
 •File consists of millions of records separated by new line character.  
 •Each line can be a valid IPv4/IPv6 or invalid string

Expected output  
 •Total IPv4 address count:  
 •Total IPv6 address count  
 •Unique IPv4 address count:  
 •Unique IPv6 address count:  
 •Invalid IP address count:

**System Design:**

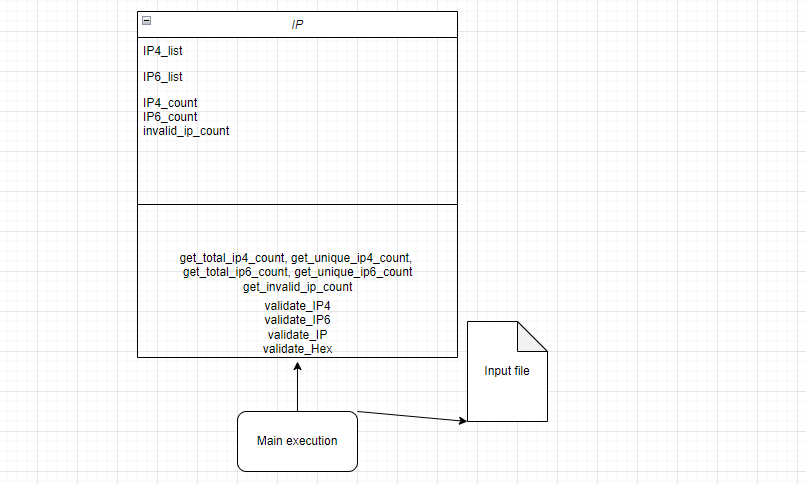
Application Design:

The process flow for this system is as below.



**API Catalogue:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No.** | **Name of API** | **Description** | **Input** | **Output** | **Access Level** |
| 1. | validate\_IP4 | Check if input string represent IPv4 address or not | string | boolean | private |
| 2. | validate\_Hex | Check if input string represent hexadecimal value or not | string | boolean | private |
| 3. | validate\_IP6 | Check if input string represents IPv6 address or not | string | boolean | private |
| 4. | get\_total\_ip4\_count | Prints total count of IPv4 addresses | void | void | public |
| 5. | get\_unique\_ip4\_count | Prints count of unique IPv4 addresses | void | void | public |
| 6. | get\_total\_ip6\_count | Prints total count of IPv6 addresses | void | void | public |
| 7. | get\_unique\_ip6\_count | Prints count of unique IPv6 addresses | void | void | public |
| 8. | get\_invalid\_ip\_count | Prints total count of invalid lines present in file | void | void | public |
| 9. | validate\_ip | Increase count and insert value in set based on type of IP address represented by input string | string | string | public |
| 10. | analyse\_file\_data | It reads data from input file and calls validate\_ip for further processing | char\* | void | public |

**Data Model:**

**Requirement Limitations:**

There are some limitations as per user requirement that we need to process one file. We can enhance our current logic if need to process multiple files using threads. In the main function we will check total number of command line arguments and will create that much threads, each thread will be calling *analyse\_file\_data* API to process request on separate thread.

We can add below logic:

|  |
| --- |
| int main(int argc, char\*\* argv)  {  array<thread, argc-1> threads;  int loop = 0;  for(auto th = threads.begin(); th != threads.end(); th++)  {  loop++;  th = thread(analyse\_file\_data, argv[loop])  }    for(auto &th:threads)  {  th.join();  }  return 0;  } |