IP Address Manager Using Doubly Linked List

Author: Ishtiaq Ahmed
Instructor: Mr.M.Shayan Shah
Affiliation: Department of Artificial Intelligence, Institute of Management Sciences (IMSciences), Peshawar, Pakistan

PROBLEM STATEMENT

Managing and validating multiple IP addresses efficiently requires a system that can store and traverse them in both directions. The task is to read IP addresses from a file, store them in a **doubly linked list**, and then display them **forward (head to tail)** and **backward (tail to head)** — ensuring all memory is managed properly.

SOLUTION

The solution is a **C++ program** that:

- Reads IP addresses from a text file .
- Validates each address to ensure it follows correct IPv4 formatting.
- Inserts each valid address into a **doubly linked list**, where each node holds:
  - The IP address
  - A pointer to the next node
  - A pointer to the previous node
- Displays the list from start to end and from end to start.
- Properly deletes all nodes before exiting to prevent memory leaks.

This approach provides efficient traversal and flexible data handling without using built-in STL containers.

CONCLUSION

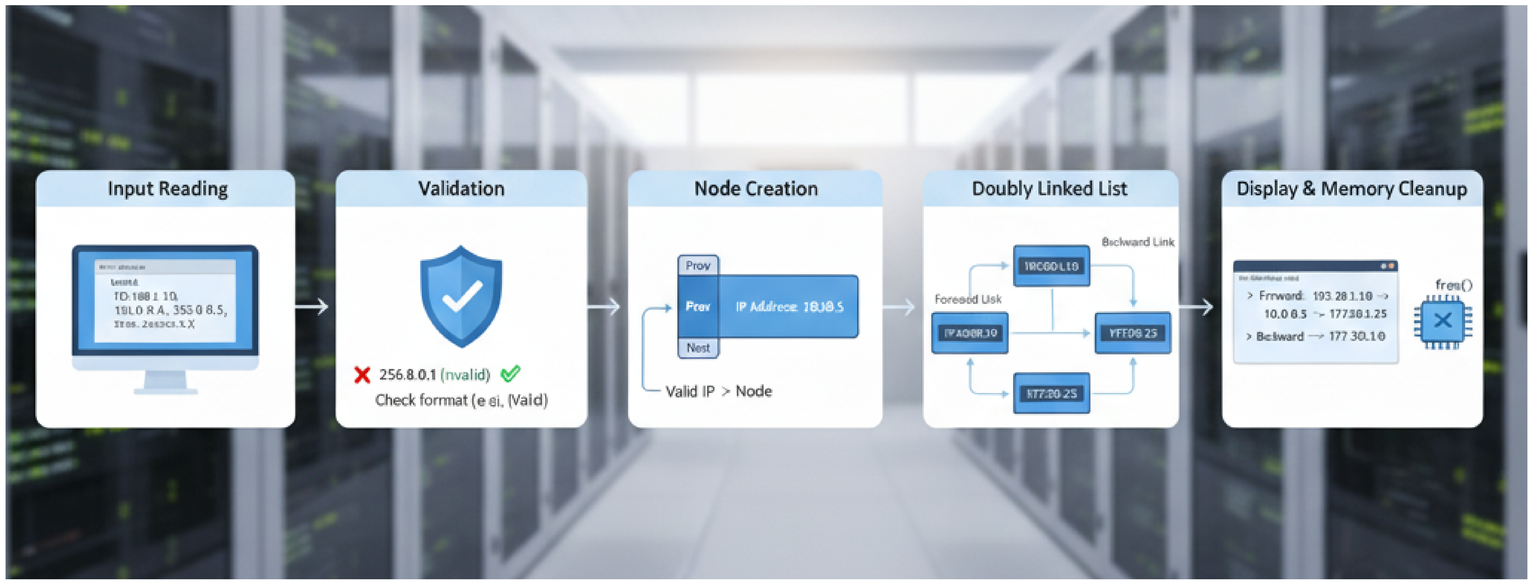
The **IP Address Manager** successfully loads, validates, and displays IP addresses using a **doubly linked list**. It demonstrates effective use of:

- File handling
- String processing
- Dynamic memory allocation
- Bidirectional traversal

This project highlights how data structures can efficiently solve real-world problems such as managing network data.

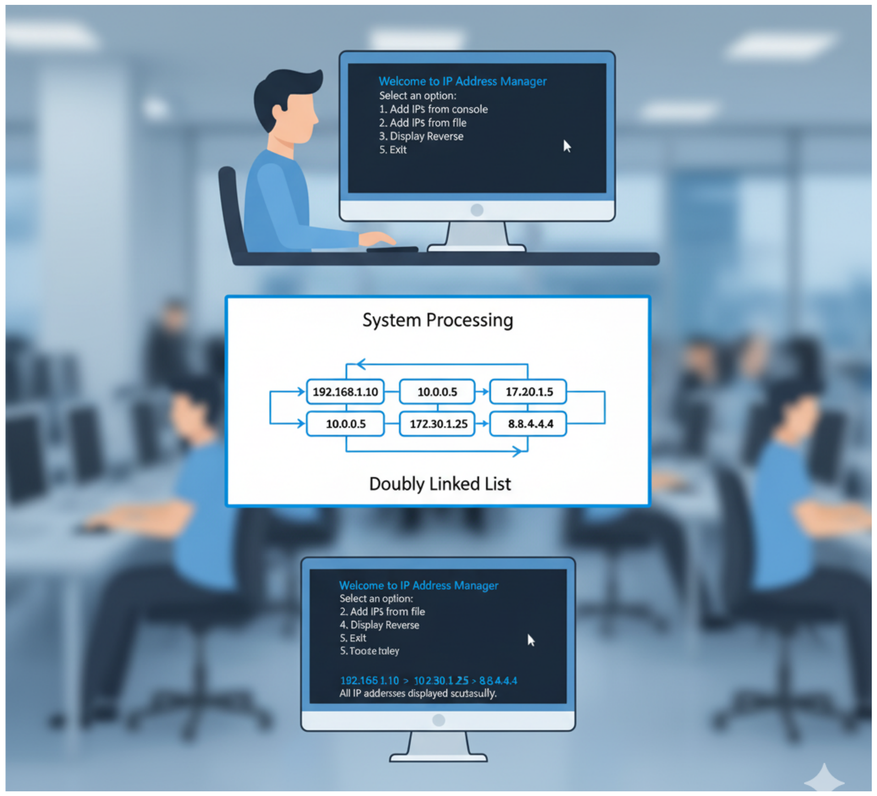
WORKING METHODOLOGY

The program follows a structured approach to manage IP addresses efficiently using a **doubly linked list**. It begins by reading a list of IP addresses from a text file or console input. Each IP address is separated by commas and validated for proper IPv4 format. Valid addresses are then stored in dynamically created nodes, where each node contains the IP address and two pointers — one to the next node and one to the previous node — enabling bidirectional traversal.
After insertion, the program allows the user to display the stored IP addresses **from start to end** or **from end to start** using traversal functions. Finally, before the program exits, all nodes are deleted to ensure proper memory management and avoid memory leaks.



USER INTERACTION FLOW

This diagram illustrates how the user interacts with the IP Address Manager program. The process begins when the user runs the C++ console application and is presented with a menu of options. The user can choose to add IP addresses manually or load them from a file. Each entered IP address is automatically validated before being added to the doubly linked list. Once stored, the user can display all IP addresses in forward or reverse order. The system continuously updates the list as operations are performed and ensures all memory is released properly upon exit.



Created with BioRender Poster Builder