IPv4

Validator

Student
Ayesha Khatoon

Index

About project IPv4 Validator

An IPv4 validator project is a CLI designed to validate and verify IPv4 addresses.

IPv4 (Internet Protocol version 4) is the fourth version of the Internet Protocol, which is commonly used to identify and locate devices on a network. An IPv4 address consists of four sets of numbers separated by periods (e.g., 192.168.0.1).

The primary purpose of an IPv4 validator project is to ensure that an IPv4 address is valid and correctly formatted. It performs various checks and validations to determine if the provided address is a valid IPv4 address and it also determines in which class does it lay on

Some of the benefits of an IPv4 validator project include:

1. **Address Validation**: The project helps to verify the correctness and validity of an IPv4 address. It checks if the address has the correct format and follows the rules defined for IPv4 addresses.

2. **Preventing Errors**: By validating IPv4 addresses, the project helps prevent errors that may occur due to incorrect or malformed addresses. It ensures that addresses used in networking configurations are accurate and properly formatted, reducing the chances of network issues.

3. **Enhancing Security**: An IPv4 validator project can also contribute to enhancing security by preventing malicious inputs. It can detect invalid or

potentially harmful addresses that may be used for malicious purposes, such as IP spoofing or unauthorized access.

4. **Automated Validation**: With an IPv4 validator project, the validation process can be automated, saving time and effort for network administrators or developers. Instead of manually checking each address, the project can quickly validate a large number of addresses, making it easier to handle address management tasks.

5. **Network Efficiency**: Validating IPv4 addresses ensures that the correct addresses are used in networking configurations, leading to improved network efficiency. It helps in reducing communication errors and enables smooth communication between devices on the network.

Tech stack used in IPv4 Validator

1. Programming Languages – JavaScript
2. Runtime Environment – Node.js

Proposed Solution

Procedure

Installing axios, readline-sync , cli-color packages

Importing packages

```
import readline from "readline-sync";
import color from "cli-color";
```

Using Cli-color package, naming the cli package codes to the new color variables

```
console.clear()
const pink = color.xterm(212)
const peach =  color.xterm(224)
const blue = color.xterm(153)
const lime = color.xterm(229)
```

```
function main(){

    let a=[];
    let input = readline.question(pink("Enter your IPv4 address: "))
    let ipv4 = input.split('.')
    ipv4.forEach((input)=>{
        a.push(Number(input))
    })
    if (a.length===4){
         ipcheck(a)
    }

else{
    console.log(peach("It is not ipv4"))
}
```

1.Asking user an input to check the given number is valid IPv4 address

2.Splitting the input from ` . ` and checking the length of the given input

3. If the given input length after splitting is 4 then we're going to next of validation or else it returns not an IPv4 address

```
function ipcheck(a){
    if(a[0]>=0 && a[0]<=127 && a[1]<=255 && a[2]<=255 && a[3]<=255){
        console.log(peach("It is a IPV4 address"))
        console.log(blue("It comes under class A"))
        if(a[ipv4]==10){
            console.log(lime("Private"))
        }
        else{
            console.log(lime("Public"))
        }
}
else if(a[0]>=128 && a[0]<=191 && a[1]<=255 && a[2]<=255 && a[3]<=255){
    console.log(peach("It is a IPV4 address"))
    console.log(blue("It comes under class B"))
    if(a[0]==172 && a[1]>=16 && a[1]<=31){
        console.log(lime("Private"))
    }
    else{
        console.log(lime("Public"))
    }
}
```

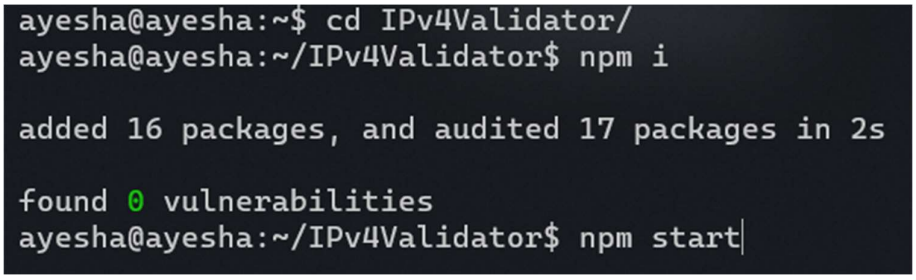
4. If it's an IPv4 address it checks if it is an valid IPv4 address and in which class does it lay on and is it public or private.

Installation of IPv4 Validator

GitHub Repository: https://github.com/ayeshakhatoon17/IPv4Validator

1.Clone the repository

2. Installing dependencies & running project



3. Output: