IPv4

Validator

Student
Mohammed Riyaan

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 thataddresses 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 quicklyvalidate 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

Backend - Server

1. Programming Languages - JavaScript
2. Runtime Environment - Node JS

NPM Packages

1. readline-sync:- Synchronous Readline for interactively running to have a conversation with the user via a console.

2. cli-color:- Colors, formatting and other goodies for the console. This package won't mess with built-ins and provides neat way to predefine formatting patterns.

3. Proposed Solution

Procedure

Installing 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

```
const red = color.xterm(1)
const blue=color.xterm(31)
const orange=color.xterm(166)
const green=color.xterm(79)
const pink=color.xterm(5)
```

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

```
function generateIp(){
    console.clear()
    console.log(red("**********************************"))
    console.log(green("\n        IP Address Project          "))
    console.log(red("\n**********************************"))

    let ip=readline.question(blue("\nEnter your IP Address : "))
    console.log(blue(ip))

    let split=ip.split(".")
    if((split.length==4) && (Number(split[0,1,2,3])>=0)&&(Number(split[0,1,2,3])<=255)){
        range()
    }else{
        console.log(pink("\nInvalid IP Address"))
    }
```

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.

```
    function range(){
        console.log(pink("\nValid IP Address"))
        let result=split.join(".")
        let result1=parseFloat(result)
        if(result1>=parseFloat("0.0.0.0")&&result1<=parseFloat("127.255.255.255")){
            if(result1>=parseFloat("10.0.0.0")&&result1<=parseFloat("10.255.255.255")){
            console.log(orange("CLASS A and Private IP"))
        }else{
            console.log(orange("CLASS A and Public IP"))
        }
        }else if(result1>=parseFloat("128.0.0.0")&&result1<=parseFloat("191.255.255.255")){
            if(result1>=parseFloat("172.16.0.0")&&result1<=parseFloat("172.31.255.255")){
            console.log(orange("CLASS B and Private IP"))
        }else{
            console.log(orange("CLASS B and Public IP"))
        }
        }else if(result1>=parseFloat("192.0.0.0")&&result1<=parseFloat("223.255.255.255")){
            if(result1>=parseFloat("192.168.0.0")&&result1<=parseFloat("192.168.255.255")){
            console.log(orange("CLASS C and Private IP"))
        }else{
            console.log(orange("CLASS C and Public IP"))
        }
        }else if(result1>=parseFloat("224.0.0.0")&&result1<=parseFloat("239.255.255.255")){
            console.log(orange("CLASS D and Public IP "))
        }else {
            console.log(orange("CLASS E and Public IP"))
        }
    }
}
generateIp()
```

Installation of IPv4 Validator

GitHub Repository: https://github.com/mohdriyaan/IPv4-Validator

To clone and Install packages:

```
riyaan@riyaan:~$ git clone git@github.com:mohdriyaan/IPv4-Validator.git
Cloning into 'IPv4-Validator'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

To Run Project

cd IPv4-Validator

npm i

npm start

Output :-

```
**********************************

        IP Address Project

**********************************

Enter your IP Address : 192.1.8.128
192.1.8.128

Valid IP Address
CLASS C and Public IP
```