A

PROJECT REPORT

ON

MINI PROJECT

ENTITLED

“*IPv4 &IPv6 Conversion*”

SUBMITTED BY

Raval Vedant M. (216090307026)

Doshi Yug N. (216090307028)

Narshindani Krish S. (216090307031)

Shah Parth V. (216090307037)

Shah Smit Y. (216090307039)

To

COMPUTER ENGINEERING DEPARTMENT

C U SHAH GOVT. POLYTECHNIC – SURENDRANAGAR

GUJARAT TECHNOLOGICAL UNIVERSITY – AHMEDABAD

JANUARY – 2023

Introduction to Project

The IPv4 to IPv6 Conversion Mini Project aims to develop a tool or application that facilitates the transition from the IPv4 (Internet Protocol version 4) addressing system to IPv6 (Internet Protocol version 6). The project acknowledges the limitations of the IPv4 addressing scheme, such as address exhaustion, and addresses the need for a more robust and scalable solution provided by IPv6.

IPv4, the fourth iteration of the Internet Protocol, has been the primary addressing scheme used on the internet for several decades. However, due to the exponential growth of internet-connected devices and networks, the available IPv4 address space has become increasingly scarce. IPv6, the next-generation protocol, was developed to overcome the limitations of IPv4 and provide a significantly larger address space.

- **Project Objectives:**

- **Address Mapping:** Develop an algorithm or mechanism to convert IPv4 addresses to IPv6 addresses, ensuring compatibility and preserving relevant information during the conversion process.

- **Data Integrity:** Ensure that the conversion process accurately reflects the original IPv4 address and maintains data integrity, minimizing the potential for errors or loss of information during the conversion.

- **User Interface:** Design a user-friendly interface that allows users to input IPv4 addresses and obtain their corresponding IPv6 counterparts, promoting ease of use and accessibility.

- **Validation and Testing:** Implement a comprehensive validation and testing framework to verify the accuracy of the conversion tool and identify any potential issues or bugs.

- **Documentation and Reporting:** Prepare clear and concise documentation that outlines the conversion process, including the underlying methodology, algorithm, and instructions for users. Additionally, generate reports to track the progress of the project and highlight any challenges encountered.

- Benefits:

- Enhanced Address Space: By facilitating the migration from IPv4 to IPv6, the project contributes to the availability of a vastly larger address space, accommodating the growing number of devices and networks connected to the internet.

- Future-Proofing: As IPv6 adoption becomes more prevalent, this project ensures that systems and networks are prepared for the transition, enabling seamless communication across both IPv4 and IPv6 environments.

- Learning Opportunity: The project provides an opportunity for developers to gain insights into IPv4 and IPv6 protocols, address conversion techniques, and network infrastructure, enhancing their knowledge and skills in networking technologies.

- Conclusion:

- The IPv4 to IPv6 Conversion Mini Project aims to address the limitations of IPv4 and facilitate the transition to IPv6. By developing a user-friendly tool or application that converts IPv4 addresses to IPv6 addresses accurately, the project contributes to the evolution of internet addressing systems and supports the future growth and scalability of networks.

Working of Project

1. User Interface:
    - Design a user-friendly interface that allows users to input an IPv4 address.
    - Provide a "Convert" button to trigger the conversion process.
    - Display the corresponding IPv6 address to the user after conversion.
2. Conversion Algorithm:
    - Accept the user's input of an IPv4 address.
    - Parse the IPv4 address into its four octets (e.g., 192.168.0.1).
    - Expand each octet into its binary representation (e.g., 11000000.10101000.00000000.00000001).
    - Append the necessary zeros to each binary octet to make them 8 bits long (e.g., 11000000.10101000.00000000.00000001).
    - Concatenate the four binary octets together (e.g., 11000000101010000000000000000001).
    - Insert the necessary separators to create the IPv6 format (e.g., 1100:0000:0000:0000:0000:0000:0101:0403).
    - Convert any consecutive groups of zeros into "::" (e.g., 1100::0101:0403).
    - Present the converted IPv6 address to the user.
3. Validation and Error Handling:
    - Verify that the user's input is a valid IPv4 address.
    - Check for any potential errors or invalid inputs during the conversion process.
    - Provide appropriate error messages or notifications to the user if any issues occur.
4. Testing:
    - Develop a comprehensive testing framework to ensure the accuracy of the conversion algorithm.
    - Test the tool with various IPv4 addresses, including edge cases and common scenarios.
    - Verify that the converted IPv6 addresses are correct and comply with the IPv6 format.

5. Documentation:
    - Prepare clear and concise documentation that explains the project's purpose, the conversion algorithm used, and instructions for users on how to operate the tool.
    - Include any prerequisites, dependencies, or installation instructions if applicable.
    - Provide examples and usage scenarios to guide users effectively.

Code of Project

Put code of your project here

In main.py

```python
import eel

# Initialize Eel
eel.init('web')


@eel.expose
def ipv4_to_ipv6(ipv4Address):
    ipv4_parts = ipv4Address.split('.')
    # print(len(ipv4_parts))
    ipv4_binary = ''
    # Generating a 32 bit binary number of ipv4 address
    for i in ipv4_parts:
        if (i != '0'):
            binary_num = str(bin(int(i)))[2:]
            num = binary_num.zfill(8)
            ipv4_binary = ipv4_binary + num
        else:
            ipv4_binary = ipv4_binary + '00000000'

    # print(ipv4_binary)
    result = ''
    for i in range(0, len(ipv4_binary), 4):
        result = result + ipv4_binary[i:i+4]
        if (i < 28):
            result = result + ':'

    # print(len(ipv4_binary))
    print(result)
    return result

@eel.expose
def compress_ipv6(ipv6Address):
    # Split the IPv6 address into its individual parts
    parts = ipv6Address.split(':')

    # Find the longest sequence of consecutive zeros
    longest_zeros_start = -1
    longest_zeros_length = -1
    current_zeros_start = -1
    current_zeros_length = -1

    for i, part in enumerate(parts):  # Iterate in form of index parts pair
```

```
        if part == '0000':
            if current_zeros_length == -1:
                # it will give us the position of current zeros start because
i is index
                current_zeros_start = i
            current_zeros_length += 1  # length of available zeros is
increased by 1
        else:
            if current_zeros_length > longest_zeros_length:
                longest_zeros_start = current_zeros_start
                longest_zeros_length = current_zeros_length
            current_zeros_length = -1

    if current_zeros_length > longest_zeros_length:
        longest_zeros_start = current_zeros_start
        longest_zeros_length = current_zeros_length

    # Compress the IPv6 address
    compressed_parts = []
    flag = 0
    for i, part in enumerate(parts):
        if (flag == 0):
            if i == longest_zeros_start:
                compressed_parts.append('')
                if longest_zeros_length > 1:
                    flag = 1
                    compressed_parts.append('')
            elif part != '0000':
                compressed_parts.append(part)
            elif part == '0000':
                compressed_parts.append(part)

    # Construct the compressed IPv6 address
    compressed_address = ':'.join(compressed_parts)

    return compressed_address


@eel.expose
def convertohex_compress(final_ip):
    final_ip = str(final_ip).split(':')
    index = final_ip.index('')
    while ('' in final_ip):
        final_ip.remove('')

    ip = ''
    count = 0
    # print(index)
    for i in final_ip:
        if (index != count):
            ip = ip + hex(int(i, 2))[2:]
            count = count + 1
            if (count != len(final_ip)):
                ip = ip + ':'

        else:
            ip = ip + ':'
            count = count + 1

    if (index == count):
        ip = ip + '::'

    return ip


@eel.expose
def convertohex(final_ip):
    final_ip = str(final_ip).split(':')
    # print(final_ip)
    ip = ''
    count = 1
    # print(index)
    for i in final_ip:
        if (count == 8):

            ip = ip + hex(int(i, 2))[2:]
            count = count + 1
        else:
            ip = ip + hex(int(i, 2))[2:]
            ip = ip + ":"
            count = count + 1

    return ip


eel.start('index.html', size=(700, 500))
ip_input=input("Enter IP address")
ans = ipv4_to_ipv6(ip_input)
final_ip = compress_ipv6(ans)
print(convertohex(ans))
print(convertohex_compress(final_ip))
```

In index.html

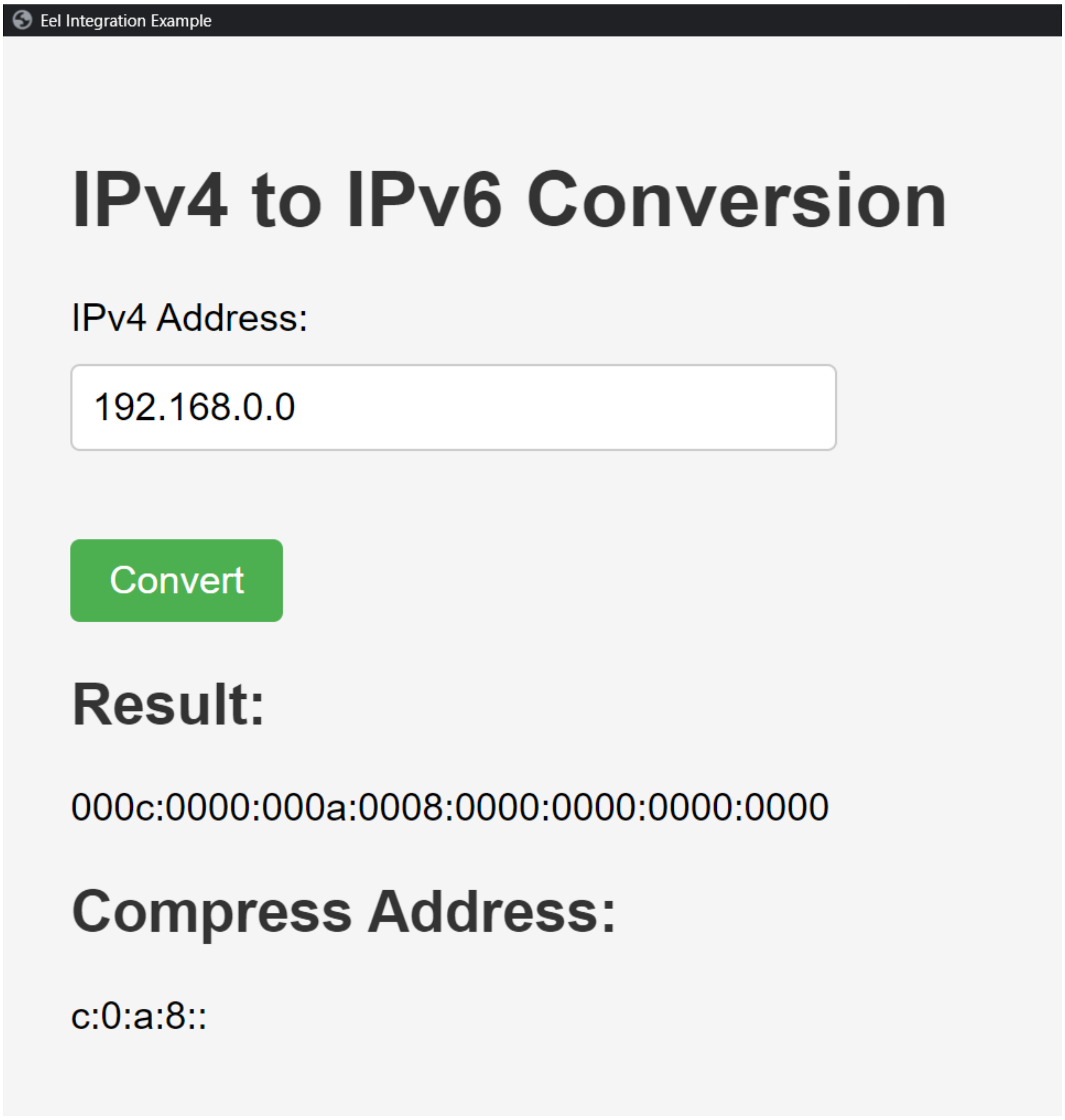
```
<!DOCTYPE html>
<html>
<head>
    <title>Eel Integration Example</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script type="text/javascript" src="/eel.js"></script>
</head>
<body>
    <div class="container">
        <h1>IPv4 to IPv6 Conversion</h1>

        <div class="input-container">
            <label for="ipv4Address">IPv4 Address:</label>
            <input type="text" id="ipv4Address" name="ipv4Address"
placeholder="Enter IPv4 Address">
        </div>
        <br><br>
        <div class="button-container">
            <button onclick="convert()">Convert</button>
        </div>

        <h2>Result:</h2>
        <p id="result"></p>
        <h2>Compress Address:</h2>
        <p id="compress"></p>
    </div>

    <script type="text/javascript">
        async function convert() {
            var ipv4Address = document.getElementById("ipv4Address").value;

            // Call the exposed Python function using Eel
            var result = await eel.ipv4_to_ipv6(ipv4Address)();

            // Update the result on the web page
            document.getElementById("result").innerHTML = result;

            var compress = eel.convertohex(result)();
            document.getElementById("compress").innerHTML = compress;
        }
    </script>
</body>
</html>
```

In style.css

```css
/* main.css */

body {
  font-family: Arial, sans-serif;
  background-color: #f5f5f5;
  padding: 20px;
}

h1 {
  color: #333;
}

label {
  display: block;
  margin-bottom: 10px;
}

input[type="text"] {
  width: 300px;
  padding: 8px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

button {
  padding: 8px 16px;
  font-size: 16px;
  background-color: #4CAF50;
  color: #fff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

button:hover {
  background-color: #45a049;
}

h2 {
  color: #333;
  margin-top: 20px;
}

p {
  font-size: 16px;
}
```

Screenshots

Put screenshot of your project here



Eel Integration Example

IPv4 to IPv6 Conversion

IPv4 Address:

192.168.0.0

Result:

000c:0000:000a:0008:0000:0000:0000:0000

Compress Address:

c:0:a:8::