NATIONAL INSTITUTE OF TECHNOLOGY TIRUCHIRAPALLI, TAMIL NADU 620015

PROJECT REPORT: IP SUBNET CALCULATOR

Team Members:

Name: Parth Danagaya, Roll No: 205124063
Name: Prasad Sahil, Roll No: 205124069
Name: Tej Prakash Carpenter, Roll No: 205124101
Name: Vivek Singh, Roll No: 205124114

Subject Code – CA716
Name: Object Oriented Programming
Faculty: Dr. S Ghanshyam Bopche
Course Details: MCA (Masters in Computer Applications)
Academic Session: 2024–2025
Date: 10-April-2025

Introduction

The Network ID & IP Range Extractor is a Java-based desktop application developed using the Swing framework. This utility allows users to input an IP address along with its subnet mask and instantly retrieve essential network details such as the **Network ID**, **Broadcast ID**, and the complete **range of valid host IP addresses**. Designed with simplicity and precision in mind, the tool provides a user-friendly interface for students, network admins, and anyone looking to understand or debug IP-based configurations effectively.

Problem Statement

In many scenarios, users need to calculate and understand network parameters such as the Network ID, Broadcast ID, and valid IP address ranges—especially while configuring networks, managing subnets, or learning IP addressing. However, performing these calculations manually or through command-line tools can be time-consuming and error-prone. This project aims to simplify the process by providing a graphical desktop application where users can input an IP address and subnet mask, and instantly retrieve all the necessary network information in an intuitive and user-friendly interface.

Functional and Usability Requirements

- Functional Requirements:

  - Accept user input for **IP address** and **Subnet mask**.
  - Calculate and display the **Network ID** based on the input.
  - Calculate and display the **Broadcast ID**.
  - Generate and list the complete **range of valid host IP addresses**.
  - Include responsive **GUI buttons** for each operation using Java Swing.
  - Validate input to ensure correct IP and subnet format.

- Usability Requirements:

  - The application should be simple and intuitive for users with minimal technical knowledge.
  - Results (like Network ID, Broadcast ID, etc.) should be shown in a **clean, formatted, and easy-to-read** manner.
  - The GUI should be **visually appealing** with clear fonts, button styling, and organized layout for a better user experience.

Proposed Solution

Here is the source code for the given problem

//GUI Part

```
import java.awt.*;

import java.awt.event.*;

import java.io.File;

import java.net.InetAddress;

import javax.swing.*;


public class IPSubnetCalculatorGUI {

   private static JLabel titleLabel, descriptionLabel, subnetMaskLabel, ipLabel;

   private static JTextField ipAddressField, subnetMaskField;

   private static JLabel networkAddressLabel, broadcastAddressLabel, usableIPsLabel;

   private static JLabel ipv6Label, ipv6SubnetMaskLabel, ipv6NetworkAddressLabel,
ipv6BroadcastAddressLabel, ipv6UsableIPsLabel;

   private static JTextField ipv6AddressField, ipv6PrefixLengthField;

   private static JFrame frame;

   private static Font jedarFont;

   private static JTabbedPane tabbedPane;

   private static JPanel ipv6ResultPanel; // Declare ipv6ResultPanel


   public static void main(String[] args) {

      SwingUtilities.invokeLater(() -> createAndShowGUI());

   }


   private static void createAndShowGUI() {

      frame = new JFrame("IP Subnet Calculator");
```

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(800, 600);
frame.setLocationRelativeTo(null);

// Background image setup
try {
    ImageIcon backgroundImage = new ImageIcon("network_BG.jpg");
    JLabel backgroundLabel = new JLabel(backgroundImage);
    backgroundLabel.setLayout(new BorderLayout());
    frame.setContentPane(backgroundLabel);
} catch (Exception e) {
    System.err.println("Background image not found: " + e.getMessage());
}

try {
    jedarFont = Font.createFont(Font.TRUETYPE_FONT, new File("Jedar.ttf")).deriveFont(24f);
    GraphicsEnvironment.getLocalGraphicsEnvironment().registerFont(jedarFont);
} catch (Exception e) {
    jedarFont = new Font("Arial", Font.BOLD, 24);
}

JPanel mainPanel = new JPanel();
mainPanel.setOpaque(false);
mainPanel.setLayout(new BorderLayout());
mainPanel.setBorder(BorderFactory.createEmptyBorder(60, 80, 60, 80)); // 10% padding approx

JPanel topPanel = new JPanel();
topPanel.setLayout(new BoxLayout(topPanel, BoxLayout.Y_AXIS));
topPanel.setOpaque(false);
```

```
titleLabel = new JLabel("IP Subnet Calculator", SwingConstants.CENTER);

titleLabel.setFont(jedarFont);

titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

titleLabel.setForeground(Color.WHITE); // Set to white

titleLabel.setBorder(BorderFactory.createEmptyBorder(10, 0, 0, 0));

topPanel.add(titleLabel);

    descriptionLabel = new JLabel("<html><div style='text-align:center;'>The IP Subnet Calculator
    tool calculates network values.<br>It uses network class, IP address, and subnet mask to
    calculate and return a list of data regarding IPv4 and IPv6 subnets.</div></html>",
    SwingConstants.CENTER);

descriptionLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

descriptionLabel.setFont(new Font("Arial", Font.PLAIN, 14));

descriptionLabel.setForeground(Color.WHITE); // Set to white

descriptionLabel.setBorder(BorderFactory.createEmptyBorder(10, 20, 20, 20));

topPanel.add(descriptionLabel);


tabbedPane = new JTabbedPane(JTabbedPane.TOP);

tabbedPane.setAlignmentX(Component.CENTER_ALIGNMENT);

tabbedPane.setFont(new Font("Arial", Font.BOLD, 17));

tabbedPane.addTab("IPv4", createIPV4Panel());

tabbedPane.addTab("IPv6", createIPv6Panel());


JPanel centerTabPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));

centerTabPanel.add(tabbedPane);

centerTabPanel.setOpaque(false);


mainPanel.add(topPanel, BorderLayout.NORTH);

mainPanel.add(centerTabPanel, BorderLayout.CENTER);

frame.add(mainPanel);


frame.addComponentListener(new ComponentAdapter() {

   @Override
```

```
        public void componentResized(ComponentEvent e) {
            resizeFonts();
        }
    });

    frame.setVisible(true);
}

private static JPanel createIPV4Panel() {
    JPanel panel = new JPanel(new GridBagLayout());
    panel.setOpaque(false);

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets(10, 10, 10, 10);
    gbc.fill = GridBagConstraints.HORIZONTAL;

    ipLabel = new JLabel("IP Address:");
    ipLabel.setFont(new Font("Arial", Font.BOLD, 14));
    gbc.gridx = 0; gbc.gridy = 0;
    panel.add(ipLabel, gbc);

    ipAddressField = new JTextField("127.0.0.1", 15);
    ipAddressField.setFont(new Font("Arial", Font.PLAIN, 14));
    ipAddressField.setBorder(BorderFactory.createLineBorder(Color.GRAY));
    gbc.gridx = 1;
    panel.add(ipAddressField, gbc);

    subnetMaskLabel = new JLabel("Subnet Mask:");
    subnetMaskLabel.setFont(new Font("Arial", Font.BOLD, 14));
    gbc.gridx = 0; gbc.gridy = 1;
```

```
panel.add(subnetMaskLabel, gbc);

subnetMaskField = new JTextField("255.255.255.0", 15);
subnetMaskField.setFont(new Font("Arial", Font.PLAIN, 14));
subnetMaskField.setBorder(BorderFactory.createLineBorder(Color.GRAY));
gbc.gridx = 1;
panel.add(subnetMaskField, gbc);

JButton calculateButton = new JButton("Calculate");
calculateButton.setFont(new Font("Arial", Font.BOLD, 14));
calculateButton.setFocusPainted(false);
calculateButton.setBackground(new Color(33, 150, 243));
calculateButton.setForeground(Color.WHITE);
calculateButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
gbc.gridx = 0; gbc.gridy = 2;
gbc.gridwidth = 2;
panel.add(calculateButton, gbc);

networkAddressLabel = new JLabel("");
broadcastAddressLabel = new JLabel("");
usableIPsLabel = new JLabel("");

gbc.gridy = 3;
panel.add(networkAddressLabel, gbc);
gbc.gridy = 4;
panel.add(broadcastAddressLabel, gbc);
gbc.gridy = 5;
panel.add(usableIPsLabel, gbc);

calculateButton.addActionListener(e -> {
```

```
try {
    calculateButton.setEnabled(false);
    calculateButton.setText("Calculating...");

    SwingWorker<String[], Void> worker = new SwingWorker<String[], Void>() {
        @Override
        protected String[] doInBackground() throws Exception {
            return IPSubnetCalculatorLogic.calculateIPv4(
                ipAddressField.getText().trim(),
                subnetMaskField.getText().trim()
            );
        }

        @Override
        protected void done() {
            try {
                String[] results = get();
                networkAddressLabel.setText("Network Address: " + results[0]);
                broadcastAddressLabel.setText("Broadcast Address: " + results[1]);
                usableIPsLabel.setText("Usable IP Range: " + results[2]);
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(frame, "Invalid IP Address or Subnet Mask", "Error",
                JOptionPane.ERROR_MESSAGE);
            } finally {
                calculateButton.setEnabled(true);
                calculateButton.setText("Calculate");
            }
        }
    };
    worker.execute();
```

```
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(frame, "Invalid IP Address or Subnet Mask", "Error",
            JOptionPane.ERROR_MESSAGE);
            calculateButton.setEnabled(true);
            calculateButton.setText("Calculate");
        }
    });

    return panel;
}

private static JPanel createIPv6Panel() {
    JPanel panel = new JPanel(new GridBagLayout());
    panel.setOpaque(false);

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets(10, 10, 10, 10);
    gbc.fill = GridBagConstraints.HORIZONTAL;

    ipv6Label = new JLabel("IPv6 Address:");
    ipv6Label.setFont(new Font("Arial", Font.BOLD, 14));
    gbc.gridx = 0; gbc.gridy = 0;
    panel.add(ipv6Label, gbc);

    ipv6AddressField = new JTextField("2001:db8::1", 30);
    ipv6AddressField.setFont(new Font("Arial", Font.PLAIN, 14));
    gbc.gridx = 1;
    panel.add(ipv6AddressField, gbc);

    ipv6SubnetMaskLabel = new JLabel("Prefix Length:");
    ipv6SubnetMaskLabel.setFont(new Font("Arial", Font.BOLD, 14));
```

```
gbc.gridx = 0; gbc.gridy = 1;
panel.add(ipv6SubnetMaskLabel, gbc);

ipv6PrefixLengthField = new JTextField("64", 5);
ipv6PrefixLengthField.setFont(new Font("Arial", Font.PLAIN, 14));
gbc.gridx = 1;
panel.add(ipv6PrefixLengthField, gbc);

JButton calculateButton = new JButton("Calculate");
calculateButton.setFont(new Font("Arial", Font.BOLD, 14));
calculateButton.setFocusPainted(false);
calculateButton.setBackground(new Color(33, 150, 243));
calculateButton.setForeground(Color.WHITE);
calculateButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
gbc.gridx = 0; gbc.gridy = 2;
gbc.gridwidth = 2;
panel.add(calculateButton, gbc);

ipv6NetworkAddressLabel = new JLabel("");
ipv6BroadcastAddressLabel = new JLabel("");
ipv6UsableIPsLabel = new JLabel("");

ipv6ResultPanel = new JPanel(); // Initialize ipv6ResultPanel
ipv6ResultPanel.setLayout(new BoxLayout(ipv6ResultPanel, BoxLayout.Y_AXIS));
ipv6ResultPanel.setOpaque(false);
ipv6ResultPanel.add(ipv6NetworkAddressLabel);
ipv6ResultPanel.add(ipv6BroadcastAddressLabel);
ipv6ResultPanel.add(ipv6UsableIPsLabel);
ipv6ResultPanel.setVisible(false); // Initially hidden
```

```
gbc.gridy = 3;
panel.add(ipv6ResultPanel, gbc);

calculateButton.addActionListener(e -> {
  try {
    calculateButton.setEnabled(false);
    calculateButton.setText("Calculating...");

    SwingWorker<String[], Void> worker = new SwingWorker<String[], Void>() {
      @Override
      protected String[] doInBackground() throws Exception {
        return IPSubnetCalculatorLogic.calculateIPv6(
          ipv6AddressField.getText().trim(),
          Integer.parseInt(ipv6PrefixLengthField.getText().trim())
        );
      }

      @Override
      protected void done() {
        try {
          String[] results = get();
          ipv6NetworkAddressLabel.setText("Network Address: " + results[0]);
           ipv6BroadcastAddressLabel.setText("Full IPv6 Address: " +
           expandIPv6Address(ipv6AddressField.getText().trim())); // Display full IPv6 address in
           expanded format
          ipv6UsableIPsLabel.setText("Usable IP Range: " + results[0] + " - " + results[1]);

          ipv6ResultPanel.setVisible(true);
        } catch (Exception ex) {
           JOptionPane.showMessageDialog(frame, "Invalid IPv6 Address or Prefix Length",
           "Error", JOptionPane.ERROR_MESSAGE);
        } finally {
```

```
                calculateButton.setEnabled(true);
                calculateButton.setText("Calculate");
            }
        }
    };
    worker.execute();

  } catch (Exception ex) {
    JOptionPane.showMessageDialog(frame, "Invalid IPv6 Address or Prefix Length", "Error",
    JOptionPane.ERROR_MESSAGE);
    calculateButton.setEnabled(true);
    calculateButton.setText("Calculate");
  }
});

return panel;
}

private static void resizeFonts() {
  double scaleFactor = frame.getWidth() / 800.0;

  titleLabel.setFont(scaleFont(jedarFont, scaleFactor));
  descriptionLabel.setFont(scaleFont(new Font("Arial", Font.PLAIN, 14), scaleFactor));
  ipLabel.setFont(scaleFont(new Font("Arial", Font.BOLD, 14), scaleFactor));
  subnetMaskLabel.setFont(scaleFont(new Font("Arial", Font.BOLD, 14), scaleFactor));
  ipAddressField.setFont(scaleFont(new Font("Arial", Font.PLAIN, 14), scaleFactor));
  subnetMaskField.setFont(scaleFont(new Font("Arial", Font.PLAIN, 14), scaleFactor));
  networkAddressLabel.setFont(scaleFont(new Font("Arial", Font.PLAIN, 14), scaleFactor));
  broadcastAddressLabel.setFont(scaleFont(new Font("Arial", Font.PLAIN, 14), scaleFactor));
  usableIPsLabel.setFont(scaleFont(new Font("Arial", Font.PLAIN, 14), scaleFactor));
  ipv6Label.setFont(scaleFont(new Font("Arial", Font.BOLD, 14), scaleFactor));
```

```
    ipv6SubnetMaskLabel.setFont(scaleFont(new Font("Arial", Font.BOLD, 14), scaleFactor));
    ipv6AddressField.setFont(scaleFont(new Font("Arial", Font.PLAIN, 14), scaleFactor));
    ipv6PrefixLengthField.setFont(scaleFont(new Font("Arial", Font.PLAIN, 14), scaleFactor));
    ipv6NetworkAddressLabel.setFont(scaleFont(new Font("Arial", Font.PLAIN, 14), scaleFactor));
    ipv6BroadcastAddressLabel.setFont(scaleFont(new Font("Arial", Font.PLAIN, 14), scaleFactor));
    ipv6UsableIPsLabel.setFont(scaleFont(new Font("Arial", Font.PLAIN, 14), scaleFactor));
}

private static Font scaleFont(Font font, double scaleFactor) {
    try {
        return font.deriveFont((float) (font.getSize() * scaleFactor));
    } catch (Exception e) {
        return font;
    }
}

private static String expandIPv6Address(String ipv6Address) {
    try {
        // Convert to InetAddress to validate and get byte representation
        InetAddress inetAddress = InetAddress.getByName(ipv6Address);
        if (inetAddress instanceof java.net.Inet6Address) {
            byte[] bytes = inetAddress.getAddress();

            // Build full IPv6 address with all 8 segments properly formatted
            StringBuilder sb = new StringBuilder();
            for (int i = 0; i < 16; i += 2) {
                sb.append(String.format("%02x%02x", bytes[i] & 0xFF, bytes[i + 1] & 0xFF));
                if (i < 14) {
                    sb.append(":");
                }
```

```
            }

            // Convert to array of segments
            String[] segments = sb.toString().split(":");

            // Format each segment to have 4 characters with leading zeros
            for (int i = 0; i < segments.length; i++) {
                segments[i] = String.format("%4s", segments[i]).replace(' ', '0');
            }

            // Join segments back together with colons
            return String.join(":", segments);
        }
    } catch (Exception e) {
        return ipv6Address; // Return original if expansion fails
    }
    return ipv6Address;
  }
}
```

//Logic

```
import java.net.InetAddress;
import java.net.UnknownHostException;
import javax.swing.JOptionPane;

public class IPSubnetCalculatorLogic {
```

```
// IPv4 Calculation Logic
public static String[] calculateIPv4(String ipAddress, String subnetMask) throws
UnknownHostException {
    try {
        InetAddress ip = InetAddress.getByName(ipAddress);
        InetAddress mask = InetAddress.getByName(subnetMask);

        byte[] ipBytes = ip.getAddress();
        byte[] maskBytes = mask.getAddress();

        byte[] networkBytes = new byte[4];
        byte[] broadcastBytes = new byte[4];

        for (int i = 0; i < 4; i++) {
            networkBytes[i] = (byte) (ipBytes[i] & maskBytes[i]); // Network address
            broadcastBytes[i] = (byte) (networkBytes[i] | ~maskBytes[i]); // Broadcast address
        }

        InetAddress networkAddress = InetAddress.getByAddress(networkBytes);
        InetAddress broadcastAddress = InetAddress.getByAddress(broadcastBytes);

        byte[] firstUsableBytes = networkBytes.clone();
        byte[] lastUsableBytes = broadcastBytes.clone();

        boolean carry = false;
        for (int i = 3; i >= 0; i--) {
            if (i == 3 || carry) {
                firstUsableBytes[i] = (byte) ((firstUsableBytes[i] & 0xFF) + 1);
```

```
            carry = (firstUsableBytes[i] & 0xFF) == 0;
        }
    }

    boolean borrow = false;
    for (int i = 3; i >= 0; i--) {
        if (i == 3 || borrow) {
            lastUsableBytes[i] = (byte) ((lastUsableBytes[i] & 0xFF) - 1);
            borrow = (lastUsableBytes[i] & 0xFF) == 255;
        }
    }

    if (isMaskLength31or32(maskBytes)) {
        firstUsableBytes = networkBytes;
        lastUsableBytes = broadcastBytes;
    }

    InetAddress firstUsableAddress = InetAddress.getByAddress(firstUsableBytes);
    InetAddress lastUsableAddress = InetAddress.getByAddress(lastUsableBytes);

    return new String[]{
        networkAddress.getHostAddress(),
        broadcastAddress.getHostAddress(),
        firstUsableAddress.getHostAddress() + " - " + lastUsableAddress.getHostAddress()
    };

} catch (UnknownHostException e) {
```

```
        JOptionPane.showMessageDialog(null, "Invalid IP Address or Subnet Mask", "Error",
JOptionPane.ERROR_MESSAGE);
        throw e;
    }
  }

  // Helper method to determine if mask is /31 or /32
  private static boolean isMaskLength31or32(byte[] maskBytes) {
    int ones = 0;
    for (byte b : maskBytes) {
        ones += Integer.bitCount(b & 0xFF);
    }
    return ones >= 31; // True if /31 or /32
  }

  // IPv6 Calculation Logic
  public static String[] calculateIPv6(String ipv6Address, int prefixLength) throws
UnknownHostException {
    try {
        if (prefixLength < 1 || prefixLength > 128) {
            JOptionPane.showMessageDialog(null, "Invalid Prefix Length. It must be between 1 and 128.",
"Error", JOptionPane.ERROR_MESSAGE);
            throw new IllegalArgumentException("Invalid Prefix Length");
        }

        InetAddress ipv6InetAddress = InetAddress.getByName(ipv6Address);
        byte[] ipv6Bytes = ipv6InetAddress.getAddress();
        byte[] networkBytes = ipv6Bytes.clone();
```

```
int fullBytes = prefixLength / 8;
int remainingBits = prefixLength % 8;

for (int i = fullBytes; i < 16; i++) {
    if (i == fullBytes && remainingBits > 0) {
        networkBytes[i] &= (byte) (0xFF << (8 - remainingBits));
    } else if (i > fullBytes) {
        networkBytes[i] = 0;
    }
}

InetAddress networkAddress = InetAddress.getByAddress(networkBytes);

byte[] broadcastBytes = networkBytes.clone();
for (int i = fullBytes; i < 16; i++) {
    if (i == fullBytes && remainingBits > 0) {
        broadcastBytes[i] |= (byte) ~(0xFF << (8 - remainingBits));
    } else if (i > fullBytes) {
        broadcastBytes[i] = (byte) 0xFF;
    }
}

InetAddress lastUsableAddress = InetAddress.getByAddress(broadcastBytes);

return new String[]{
    networkAddress.getHostAddress(),
    lastUsableAddress.getHostAddress()
};
```

```
    } catch (UnknownHostException e) {
        JOptionPane.showMessageDialog(null, "Invalid IPv6 Address", "Error",
JOptionPane.ERROR_MESSAGE);
        throw e;
    } catch (IllegalArgumentException e) {
        throw e;
    }
  }
}
```

Code Explanation:

Swing GUI:

- Used **Java Swing** to create a **modern tabbed interface** (JTabbedPane) with responsive design.
- Custom fonts (Jedar.ttf) and background image (network_BG.jpg) add aesthetic value.
- Contains **IPv4 and IPv6 calculators** in separate tabs.
- Uses JTextField, JLabel, and JButton to take input and display results.

Ip Address Retrieval:

- Uses InetAddress.getByName() to **convert user-entered IP/subnet into byte arrays**.
- For **IPv4**, calculates:
    - **Network Address** using ip & subnet mask.
    - **Broadcast Address** using network | ~mask.
    - **Usable IP Range** (first and last usable IPs) by adjusting the host portion.

Subnet Mast Calculation:

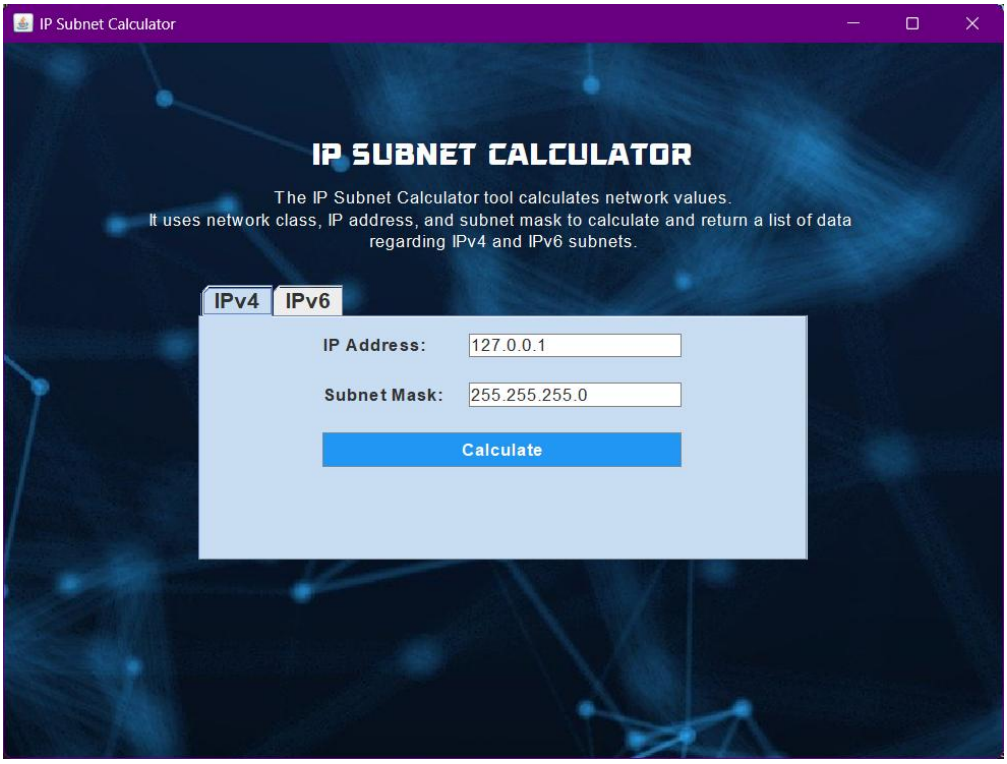
For **IPv4**:

- calculateIPv4() method computes Network, Broadcast, and Usable Range.

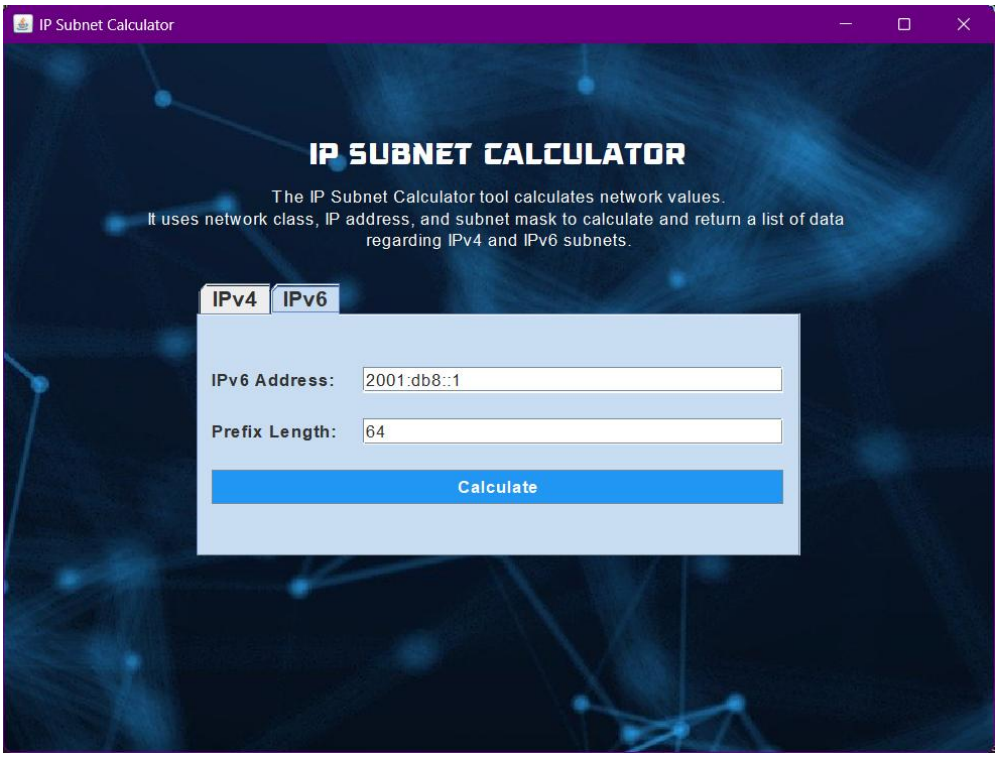
For **IPv6**:

- calculateIPv6() method applies **prefix-length based masking** to extract the network portion.
- Calculates the **last usable address** using bit-level manipulation.

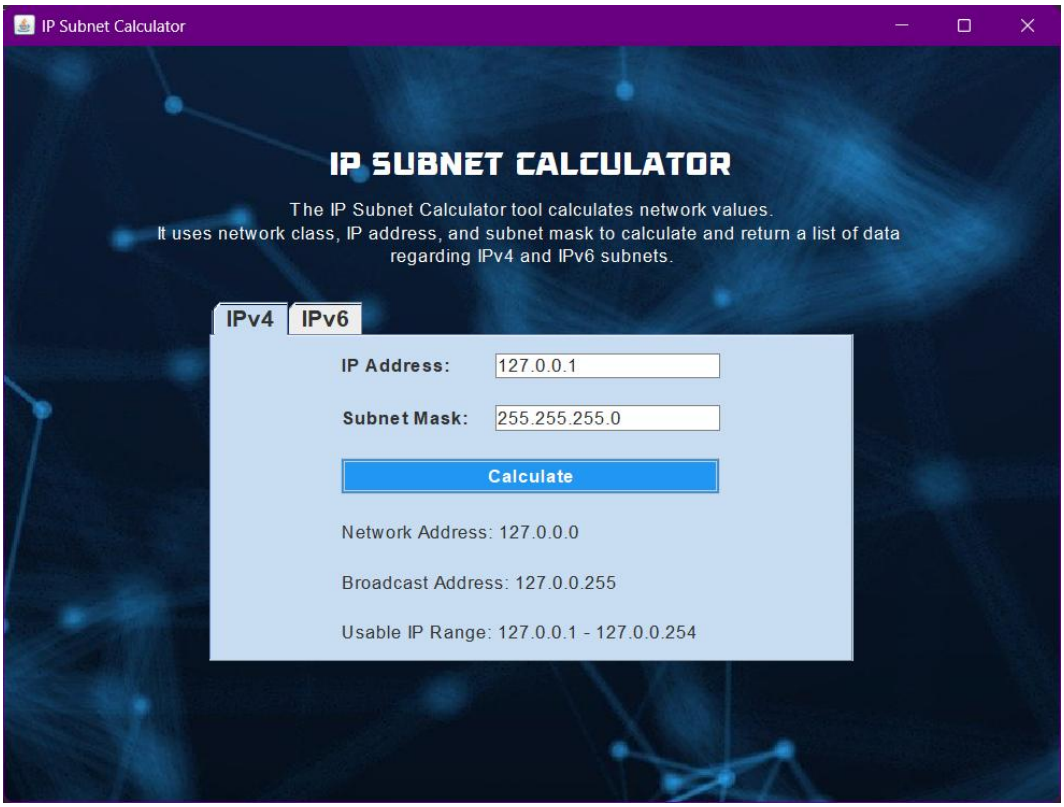
IPv4 Tab:



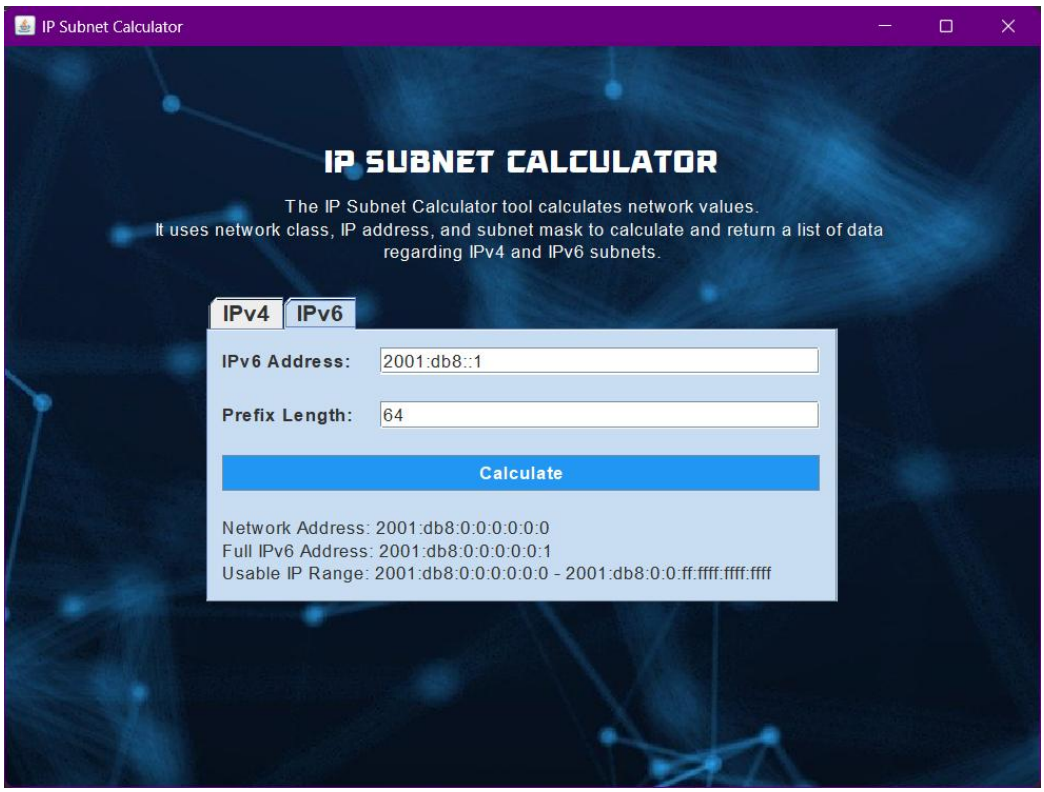
IPv6 Tab:



IPv4 Tab Output:



IPv6 Tab Output:



Conclusion:

The IP Subnet Calculator simplifies the complex task of subnetting by providing accurate and instant results for IP address segmentation. This tool is especially useful for students learning networking concepts, as well as for network administrators managing IP infrastructure. With scope for enhancements like CIDR notation support, visual subnet mapping, and export options, this project offers a solid base for further development in the field of network utilities.