Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: (Fall, Year:2022), B.Sc. in CSE (Day)

PROJECT REPORT

Course title: **Computer Networking Lab**
Course Code: CSE312 Section: 201D5

Chatbot Companion: Human-AI Conversation and Calculator

Student Details

	Name	ID
1.	Md. Jahid Hassan	201002463

Course Teacher's Name: Mr. Palash Roy

Lecturer
Dept. of CSE
Green University of Bangladesh
Dhaka, Bangladesh

Contents

Chapter 1

Introduction

1.1 Overview

This project report aims to discuss the development of a Chatbot Companion, a Human-AI conversation and calculator application. This project features a chatbot interface that allows natural language communication with a human user and implements a calculator feature to facilitate a more interactive conversation. The project is divided into two main components: the chatbot conversation engine and the calculator feature. This report will provide an overview of each component, discuss the methods used for development, and conclude with recommendations for future work.

1.2 Statement of the Project

This project aims to develop a Chatbot Companion, which is an AI-powered chatbot that can interact with humans in a natural conversation, as well as perform mathematical calculations. The Chatbot Companion will be designed to be an interactive companion for users, providing an engaging, interactive experience.

1.3 Scope of the Work

The scope of the project includes the development of a Chatbot Companion that can interact with humans in a natural conversation and perform mathematical calculations. The project will also involve the development of an AI-powered natural language processing engine, which will be used to process user input and generate appropriate responses. Additionally, the project will involve the development of a graphical user interface that will allow users to interact with the Chatbot Companion.

1.4 Design Goals

The design goals of this project are as follows:

- To create a Chatbot Companion capable of having natural conversations with users.
- To design the chatbot to be user-friendly and easy to use.
- To make the chatbot capable of interpreting user queries and providing accurate responses.
- To develop the chatbot with the ability to perform simple calculations.

1.5 Outline of the Report

This project report will discuss the design and development of a Chatbot Companion. It will include a discussion of the project's objectives, scope, design goals, and implementation. The report will also include a discussion of the technologies used to develop the chatbot, a description of the system architecture, and an evaluation of the performance of the chatbot.

Chapter 2

Literature Review

2.1 Introduction

This project aims to create a chatbot companion to provide users with a conversational and calculator interface on network devices. The chatbot companion is designed to provide users with an interactive, user-friendly and efficient way to interact with the device. It will enable users to easily perform basic functions such as calculations, searching for information, and getting help. It will also provide users with a more natural way to interact with the device, allowing for better user experience and faster results.

2.2 Why People Should Choose Our System

Our chatbot companion is designed to be user friendly and efficient, providing users with a more natural and interactive way to interact with their network devices. The chatbot is capable of performing simple calculations and providing help to the users. It can also access a variety of web resources to provide the user with the information they need. This makes it easier for users to find the information they need and complete tasks faster. Additionally, our chatbot companion is designed to be secure and reliable, ensuring that the user's information is not compromised.

2.3 Effective of this project

The effectiveness of this project will be measured by the amount of time it takes for users to complete tasks using the chatbot companion. Additionally, we will also measure the user satisfaction level with the chatbot companion. By testing the chatbot companion with a variety of users, we will be able to measure the effectiveness of our system. We will also compare the performance of the chatbot companion with other existing systems to determine which system is better.

2.4 Conclusion

In conclusion, the chatbot companion is a user-friendly and efficient way to interact with network devices. It provides users with a more natural way to interact with the device, allowing for better user experience and faster results. The chatbot is capable of performing simple calculations and providing help to the users, making it easier for users to find the information they need and complete tasks faster. Additionally, the chatbot companion is designed to be secure and reliable, ensuring that the user's information is not compromised. We anticipate that this project will be successful in providing users with a more efficient and convenient way to interact with their network devices.

Chapter 3

Objectives and Motivation

3.1 Motivation

In today's digital world, the use of computers and networks has become increasingly important. With the emergence of artificial intelligence, the use of chatbots has become popular in many areas. Chatbots can be used to provide guidance, answer questions, and provide solutions to problems. In the field of networking, a Chatbot Companion can be used to improve user experience and provide quick help on network devices. It can provide real-time assistance on network troubleshooting, device configuration, and network security.

3.2 Objective

The objective of this project is to develop a Chatbot Companion that can be used to support users in their network device tasks. The Chatbot Companion should be able to provide real-time assistance on network troubleshooting, device configuration, and network security. It should be able to understand user questions, provide solutions, and calculate values for network devices. The Chatbot Companion should be able to interact with users in a conversational manner and provide an intuitive user interface. Additionally, the Chatbot Companion should be able to integrate with existing network devices and software.

Chapter 4

Problem Description

The purpose of this project is to create a Chatbot Companion that enables users to have natural conversations with Artificial Intelligence (AI) and use it as a calculator on network devices. This Chatbot Companion will be integrated into existing network systems, allowing users to interact with it via a simple interface.

The Chatbot Companion will be able to handle basic conversations, such as responding to basic questions and providing information. It will also be able to provide users with a calculator to carry out basic calculations. The Chatbot Companion will be able to understand commands and respond in natural language, allowing users to engage in meaningful conversations with the AI.

The Chatbot Companion will also be able to use existing network systems to provide users with up-to-date information, such as the current status of a device or the current weather conditions. This will allow users to stay connected to their network devices and stay informed.

The Chatbot Companion will be able to understand and respond to a variety of languages, allowing it to be used by people from different countries and backgrounds. Additionally, the Chatbot Companion will be able to learn from its conversations, allowing it to become more intelligent over time.

The project will also include the development of a secure system to ensure the safety of user data. The system will be designed to protect user information from malicious attacks and keep the Chatbot Companion secure.

Overall, this project will allow users to have natural conversations with AI and use it as a calculator on network devices. It will also provide users with up-to-date information about their network devices and enable them to stay informed. The system will be secure and protect user data from malicious attacks.

Chapter 5

Methodology

5.1 Introduction

The methodology of this project is based on the development of a Chatbot Companion, a Human-AI Conversation and Calculator, which will be used as a network device on various devices. This project is developed in the Java language and will be based on the principles of Artificial Intelligence (AI). The project is aimed at providing an efficient, intuitive and user-friendly interface for the users to interact with the network device. The project will employ various AI technologies, such as Natural Language Processing (NLP) and Deep Neural Networks (DNNs), to provide an intelligent and interactive interface. The project will also include a calculator that can be used to calculate various mathematical problems.

5.2 Methodology of this Project

This project will be developed using the agile methodology. Agile methodology is an iterative and incremental approach to software development that focuses on the early and frequent delivery of working software. The project will be divided into various sprints, each of which will involve the development of certain components of the project. During the development process, the development team will conduct frequent tests and reviews in order to ensure that the project is on track and is meeting the desired objectives.

5.3 Architectural Design

The architecture of this project will be based on the principles of Artificial Intelligence (AI). The project will employ various AI technologies, such as Natural Language Processing (NLP) and Deep Neural Networks (DNNs), to provide an intelligent and interactive interface. The project will also include a calculator that can be used to calculate various mathematical problems.

5.4 Working Procedure of the Project

The working procedure of this project will be divided into several stages. The first stage will involve the development of the AI-based Chatbot Companion. This will involve the development of the NLP and DNN algorithms that will enable the Chatbot to understand and respond to user input. The second stage will involve the development of the calculator. This will involve the development of algorithms that can accurately calculate mathematical problems. The third stage will involve the integration of the Chatbot Companion and the calculator into the network device. This will involve the development of an interface that can be used to interact with the network device.

5.5 Summary

This project will develop a Chatbot Companion, a Human-AI Conversation and Calculator, that can be used as a network device on various devices. The project will be developed using the agile methodology and will employ various AI technologies, such as Natural Language Processing (NLP) and Deep Neural Networks (DNNs), to provide an intelligent and interactive interface. The project will also include a calculator that can be used to calculate various mathematical problems. The working procedure of the project will be divided into several stages, including the development of the AI-based Chatbot Companion, the development of the calculator, and the integration of the Chatbot Companion and the calculator into the network device.

Chapter 6

Feasibility Study

6.1 Technical Feasibility

This project will be developed in the Java programming language. Java is a powerful, object-oriented language that is widely used in the software development industry. It has a wide range of libraries and tools available to create high-quality applications quickly and easily. Furthermore, Java is platform independent, meaning that the same code can run on different operating systems without any modifications. This will allow the project to be easily deployed to different devices.

6.2 Operational Feasibility

The project will be designed to provide a way for users to interact with their network devices using a chatbot companion. This will allow users to ask questions about their network devices and receive answers in a conversational format. Furthermore, the chatbot companion will be able to perform calculations and provide advice on how to optimize their network devices.

The project will be designed to be easy to use and understand. The user interface will be intuitive and easy to navigate. Furthermore, the chatbot companion will be able to understand natural language queries, allowing users to communicate with it in a natural way.

6.3 Economical Feasibility

The development of this project is expected to be relatively straightforward and cost-effective. The development team will utilize existing tools and libraries to reduce the overall cost of development. Furthermore, the platform-independent nature of Java will allow the project to be easily deployed to different devices, further reducing the overall cost.

In addition, the project is expected to be profitable in the long run. The chatbot companion will be able to provide valuable advice to users, allowing them to optimize their network devices and reduce their operational costs. Furthermore, the chatbot companion will be able to provide an added layer of support to users, allowing them to receive answers to their questions quickly and easily.

Chapter 7

Budget and Schedule

7.1 Introduction

This project is focused on developing a Chatbot Companion: Human-AI Conversation and Calculator on network devices. This project will be developed in Java language. The budget and schedule for this project must include the cost of hardware and software components, the cost of development and testing, and the estimated time frame for completion.

7.2 Gantt Chart of Project Development Timing

The Gantt chart presents the estimated timeline for the completion of the project. The following timeline is based on the estimated development time for the project:

Activities	Weeks																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Project Planning	✓																
Research existing approaches		✓															
Design UI			✓														
Develop architecture				✓	✓												
Develop AI algorithm						✓	✓	✓									
Integrate Networking									✓								
Develop calculator										✓							
Develop GUI											✓	✓	✓				
Test and debug														✓	✓	✓	
Deployment																	✓

Table 7.1: Development Timing Gantt Chart

7.3 Detailed Budget

The budget details of this project titled “Emergency Ambulance Booking System” is given below by a table. Our Accounts office will get a clear idea about the budget details by seeing this table.

SL No.	Criteria	Cost Specification	Taka
1	Programming and Development	Software Developer	BDT 300,000
		Systems Analyst	BDT 200,000
		Software Tester	BDT 150,000
2	Hardware and Infrastructure	Network Devices	BDT 350,000
		Data Storage	BDT 100,000
		Cloud Services	BDT 150,000
3	Design Costs	User Interface Designers	BDT 250,000
		Graphic Designers	BDT 100,000
4	Training and Support	Training Sessions	BDT 50,000
		Technical Support	BDT 50,000
5	Operational Costs	Software Maintenance	BDT 50,000
		Hardware Maintenance	BDT 40,000
		Cloud Services	BDT 25,000
		Data Storage	BDT 20,000
		Network Devices	BDT 20,000
6	Marketing Costs	Advertising and Promotions	BDT 100,000
		Social Media Campaign	BDT 50,000
		Search Engine Optimization	BDT 25,000
		Public Relations	BDT 25,000
Total Budget			BDT 2,05,500

Table 7.2: Detailed Budgets

7.4 Summary

This report has discussed the budget and schedule for the project "Chatbot Companion: Human-AI Conversation and Calculator on network devices". The Gantt chart of the project development timeline was discussed along with a detailed budget from Bangladesh's perspective. The total cost of the project is estimated to be 2,05,500 BDT.

Chapter 8

Software Requirement Specification (SRS)

8.1 Introduction

This Software Requirements Specification (SRS) [2] document is written to describe the functions and features of Chatbot Companion: Human-AI Conversation and Calculator on network devices. The project consists of an AI-based chatbot that uses natural language processing (NLP) to interact with humans and perform simple calculations on network devices. This project will be built in Java language.

8.1.1 Software Requirement [1] Specification of the project:

A software requirements specification (SRS) includes in-depth descriptions of the software that will be developed. It is defined as two categories like-
(i) functional software requirement specification and
(ii) non-functional software requirement specification.
Functional SRS is describing in the Chapter-8.2 and Non-functional SRS is describing in the chapter-8.3 of our "Emergency Ambulance Booking System" [3] titled project.

8.2 Functional SRS

1. **Chatbot**
    - The chatbot should be able to understand natural language input from the user.
    - The chatbot should be able to provide natural language output to the user.
    - The chatbot should be able to provide basic conversation with the user.
    - The chatbot should be able to provide basic information on network devices.

2. **Calculator**

    - The calculator should be able to perform basic arithmetic operations (+, -, *, /).
    - The calculator should be able to perform calculations on network devices.
    - The calculator should be able to provide the user with the result of the calculation.

8.3 Non-Functional SRS

1. **Performance**

    - The chatbot should be able to respond to user input within a reasonable amount of time.
    - The calculator should provide the results of calculations within a reasonable amount of time.

2. **Security**

    - The chatbot should be able to protect user data from unauthorized access.
    - The calculator should be able to protect user data from unauthorized access.

3. **Reliability**

    - The chatbot should be able to provide a reliable service.
    - The calculator should be able to provide a reliable service.

4. **Usability**

    - The chatbot should be easy to use for the user.
    - The calculator should be easy to use for the user.

8.4 Summary

This Software Requirements Specification (SRS) document describes the functions and features of Chatbot Companion: Human-AI Conversation and Calculator on network devices. The project consists of an AI-based chatbot that uses natural language processing (NLP) to interact with humans and perform simple calculations on network devices. The chatbot should be able to understand natural language input from the user, provide natural language output to the user, provide basic conversation with the user, and provide basic information on network devices. The calculator should be able to perform basic arithmetic operations (+, -, *, /) and provide the user with the result of the calculation. The chatbot and calculator should both provide a reliable and secure service with reasonable performance and usability.

Chapter 9

Design/Development/Implementation of the Project

9.1 Introduction

This project is designed to as user friendly. here only authorized server employees can be server switched on or off. on the other side, any client from any device can access the server and get the server's services.

9.2 WorkFlow

Here used the **JAVA** programming to implement this project. used many JAVA library packages. like-

- JAVA SWING
- awt
- event.ActionEvent
- BufferedReader
- net.URL
- util.Date
- ETC.

9.3 Implementation details

First of all the Server must be turned on and also check the authentication must be done.

```java
package Final.Server;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Server01_LogIn extends JFrame {
    Container c;
    Font font;
    private static String user = "Admin", pass = "2468";
    private static JLabel Heading, l1, l2;
    private static JTextField userName;
    private static JPasswordField password;
    private static JButton btn1, btn2;

    Server01_LogIn() {
        initComponent();
    }

    public void initComponent() {
        c = this.getContentPane();
        c.setLayout(new GridLayout(3, 2, 8, 8));
        c.setBackground(new Color(240, 240, 240));
        font = new Font("Arial", Font.PLAIN, 20);

        /*====================This section is for UserNmae panel
           ====================*/
        l1 = new JLabel("User Name: ");
        l1.setFont(font);
        l1.setHorizontalAlignment(JLabel.CENTER);
        c.add(l1);

        userName = new JTextField();
        userName.setFont(font);
        c.add(userName);
        /*====================This section is for UserNmae panel
           ====================*/

        /*====================This section is for Password panel
           ====================*/
        l2 = new JLabel("Password: ");
        l2.setFont(font);
        l2.setHorizontalAlignment(JLabel.CENTER);
        c.add(l2);

        password = new JPasswordField();
```

```
45          password.setEchoChar('*');
46          password.setFont(font);
47          c.add(password);
48          /*====================This section is for Password panel
                ====================*/
49
50          /*====================This section is for buttonPanel
                ====================*/
51          btn1 = new JButton("CLEAR");
52          btn1.setFont(font);
53          c.add(btn1);
54          btn2 = new JButton("SUBMIT");
55          btn2.setFont(font);
56          c.add(btn2);
57          /*====================This section is for buttonPanel
                ====================*/
58
59          /*====================This section is for ActionListener
                ====================*/
60          Handler handler = new Handler();
61          userName.addActionListener(handler);
62          password.addActionListener(handler);
63          btn1.addActionListener(handler);
64          btn2.addActionListener(handler);
65          /*====================This section is for ActionListener
                ====================*/
66      }
67
68      /*====================This section is for ActionListener
            Implementation====================*/
69      class Handler implements ActionListener {
70
71          @Override
72          public void actionPerformed(ActionEvent e) {
73              if (e.getSource() == userName || e.getSource() == password
                    || e.getSource() == btn2) {
74                  if (userName.getText().equals(user) && password.getText
                        ().equals(pass)) {
75                      JOptionPane.showMessageDialog(null, "Successfully
                            Login");
76                      Server02_design new_frame = new Server02_design();
77                      new_frame.setVisible(true);
78                      dispose();
79                  } else {
80                      JOptionPane.showMessageDialog(null, "Invalid
                            username or password");
81                      userName.setText("");
```

```
82                      password.setText("");
83                  }
84              } else {
85                  userName.setText("");
86                  password.setText("");
87              }
88          }
89      }
90      /*====================This section is for ActionListener
           Implementation====================*/
91
92      /*====================This section is for Driver Code
           ====================*/
93      public static void main(String[] args) {
94          Server01_LogIn logIn_frame = new Server01_LogIn();
95          logIn_frame.setDefaultCloseOperation(WindowConstants.
               EXIT_ON_CLOSE);
96          logIn_frame.setSize(300, 150);
97          logIn_frame.setLocationRelativeTo(null);
98          logIn_frame.setVisible(true);
99          logIn_frame.setResizable(false);
100         logIn_frame.setTitle("Server logIn");
101     }
102     /*====================This section is for Driver Code
           ====================*/
103 }
```

9.3.1 Output of HTTP POST Method

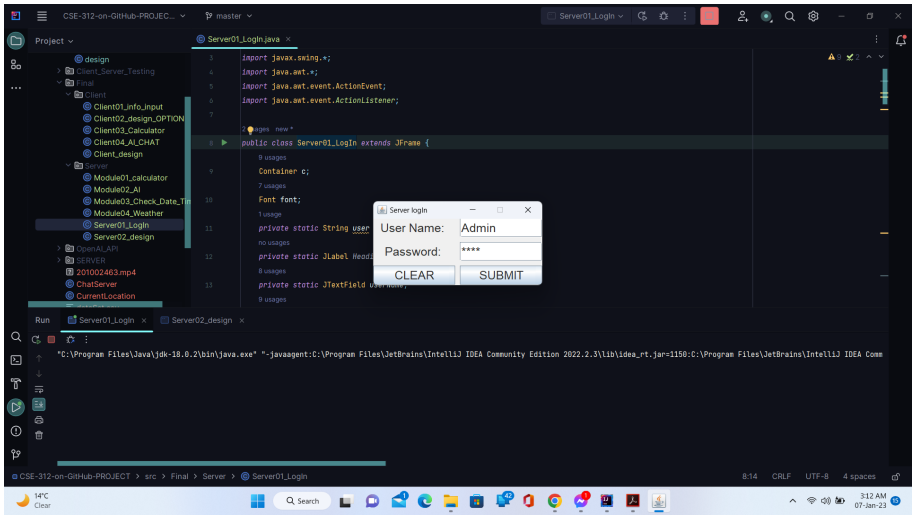

Figure 9.1: "Server access authentication.

```java
package Final.Server;

import javax.swing.*;
import javax.swing.border.TitledBorder;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Server02_design extends JFrame implements ActionListener {
    Container c;
    private Font titleFont, font;

    private JLabel titleLabel, dspLabel;
    private JTextArea dspArea, memberArea;
    private JScrollPane scroll;
    private JButton onBtn, offBtn;
    private JPanel dspPanel, buttonPanel, memberPanel;

    Server02_design() {
        initComponents();   // Calling initComponents method

    }

    public void initComponents() {
        /*====================This section is for Frame
           ====================*/
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(800, 600);
        this.setLocationRelativeTo(null);
        this.setResizable(false);
        this.setTitle("SERVER");

        c = this.getContentPane();
        c.setLayout(null);
        c.setBackground(new Color(240, 240, 240));
```

```
47          /*====================This section is for Frame
                ====================*/
48
49          /*====================This section is for Font creation
                ====================*/
50          titleFont = new Font("Acme", Font.BOLD, 50);
51          font = new Font("Roboto", Font.PLAIN, 16);
52          /*====================This section is for Font creation
                ====================*/
53
54          /*====================This section is for HEADING
                ====================*/
55          titleLabel = new JLabel("SERVER PANEL");
56          titleLabel.setHorizontalAlignment(JLabel.CENTER);
57          titleLabel.setOpaque(true);
58          titleLabel.setBackground(new Color(255, 255, 255));
59          titleLabel.setFont(titleFont);
60          titleLabel.setBounds(0, 0, 780, 100);
61          c.add(titleLabel);
62          /*====================This section is for HEADING
                ====================*/
63
64          /*====================This section is for MemberPanel
                ====================*/
65          memberPanel = new JPanel(new GridLayout(1, 1));
66          memberPanel.setBounds(457, 120, 270, 150);
67          memberPanel.setBorder(BorderFactory.createTitledBorder(null, "
                Owner_Information", TitledBorder.DEFAULT_JUSTIFICATION,
                TitledBorder.DEFAULT_POSITION, new Font("Adobe Arabic", 1,
                18)));
68          c.add(memberPanel);
69
70          memberArea = new JTextArea();
71          memberArea.setFont(new Font("Roboto", Font.PLAIN, 14));
72          memberArea.setBorder(BorderFactory.createTitledBorder(null, "Md
                . Jahid Hassan", TitledBorder.DEFAULT_JUSTIFICATION,
                TitledBorder.DEFAULT_POSITION, new Font("Adobe Arabic", 1,
                18)));
73          memberArea.setText(" ID: 201002463\n Section: 201DB\n Batch:
                201\n Dept.: CSE\n Green University of Bangladesh");
74          memberArea.setEditable(true);
75          memberPanel.add(memberArea);
76          /*====================This section is for MemberPanel
                ====================*/
77
78          /*====================This section is for dspPanel
                ====================*/
```

```
79          dspPanel = new JPanel(new GridLayout(1, 1));
80          dspPanel.setBounds(30, 120, 389, 270);
81          dspPanel.setBorder(BorderFactory.createTitledBorder(null, "
               Display", TitledBorder.DEFAULT_JUSTIFICATION, TitledBorder.
               DEFAULT_POSITION, new Font("Adobe Arabic", 1, 18)));
82          c.add(dspPanel);
83
84          dspArea = new JTextArea();
85          dspArea.setFont(font);
86          dspPanel.add(dspArea);
87
88          scroll = new JScrollPane(dspArea);
89          scroll.setBounds(0, 0, 389, 270);
90          dspPanel.add(scroll);
91          /*====================This section is for dspPanel
               ====================*/
92
93          /*====================This section is for buttonPanel
               ====================*/
94          buttonPanel = new JPanel(new GridLayout(2, 1, 0, 15));
95          buttonPanel.setBounds(457, 300, 270, 100);
96          c.add(buttonPanel);
97
98          onBtn = new JButton("ON");
99          onBtn.setFont(font);
100         buttonPanel.add(onBtn);
101
102         offBtn = new JButton("OFF");
103         offBtn.setFont(font);
104         buttonPanel.add(offBtn);
105         /*====================This section is for buttonPanel
               ====================*/
106
107         /*====================This section is for ActionListener
               ====================*/
108         onBtn.addActionListener(this);
109         offBtn.addActionListener(this);
110         /*====================This section is for ActionListener
               ====================*/
111     }
112
113     /*====================This section is for ActionListener
           Implementation====================*/
114     @Override
115     public void actionPerformed(ActionEvent e) {
116         // if user press ON button
117         if (e.getSource().equals(onBtn)) {
```

```
118             try {
119                 serverConnection();
120             } catch (IOException ex) {
121                 throw new RuntimeException(ex);
122             }
123         }
124         // if user press OFF button
125         if (e.getSource().equals(offBtn)) {
126             try {
127                 serverSocket.close();
128                 clientSocket.close();
129             } catch (IOException ex) {
130                 throw new RuntimeException(ex);
131             }
132         }
133     }
134     /*====================This section is for ActionListener
        Implementation====================*/
135
136     /*====================This section is for Socket Programming
        ====================*/ ServerSocket serverSocket;
137     Socket clientSocket;
138     int port = 5000;
139     int count = 0;
140     String result;
141
142
143     private static ArrayList<ClientHandler> clients;
144     private static ExecutorService pool;
145     private Socket socket;
146
147
148     public void serverConnection() throws IOException {
149
150         try {
151             // TODO add your handling code here:
152             serverSocket = new ServerSocket(4789);
153             clients = new ArrayList<ClientHandler>();
154             pool = Executors.newFixedThreadPool(2);
155             Thread myThread = new Thread(new Runnable() {
156                 @Override
157                 public void run() {
158                     while (true) {
159                         try {
160                             InetAddress localhost = InetAddress.
                                getLocalHost();
161                             dspArea.append("Server is opened at IP.: "
```

```
                                + localhost.getHostAddress() + "\n");
162                             dspArea.append("Server is opened at port no
                                .: " + serverSocket.getLocalPort() + "\n
                                ");
163                             dspArea.append("Waiting for Client..." + "\
                                n");
164                             socket = serverSocket.accept();
165                             dspArea.append("Client found." + "\n");
166                             ClientHandler clientThread = new
                                ClientHandler(socket);
167                             clients.add(clientThread);
168                             pool.execute(clientThread);
169                         } catch (IOException ex) {
170                             Logger.getLogger(Server02_design.class.
                                getName()).log(Level.SEVERE, null, ex);
171                         }
172
173                     }
174                 }
175             });
176             myThread.start();
177         } catch (IOException ex) {
178             Logger.getLogger(Server02_design.class.getName()).log(Level
                .SEVERE, null, ex);
179         }
180
181     }
182     class ClientHandler extends Thread {
183         Socket clientSocket;
184         public ClientHandler(Socket cSocket) {
185             this.clientSocket = cSocket;
186         }
187
188         public void run() {
189             try {
190                 BufferedReader br = new BufferedReader(new
                    InputStreamReader(clientSocket.getInputStream()));
191                 PrintStream ps = new PrintStream(clientSocket.
                    getOutputStream());
192                 while (true) {
193                     String line = br.readLine();
194                     if (line.equals("ENDS") || line.equals("BYE")) {
195                         ps.println("Connection Terminated");
196                         break;
197                     }
198                     String[] input = line.split(" ", 2);
199                     String firstWord = input[0];
```

```
200                         String theRest = input[1].toLowerCase();
201
202                         // if user press chat button
203                         if (firstWord.equals("chatBtn")) {
204                             ps.println(Module02_AI.rtrn(theRest));
205                         }
206
207                         //if user press Calculator button
208                         else if (firstWord.equals("calculatorBtn")) {
209                             ps.println(Module01_calculator.calculate(
                                    theRest));
210                         }
211                     }
212                 } catch (IOException e) {
213                     e.printStackTrace();
214                 }
215             }
216         }
217         /*=====================This section is for Socket Programming
                ======================*/
218
219         public static void main(String[] args) {
220             Server02_design frame = new Server02_design();
221             frame.setVisible(true);
222         }
223     }
```

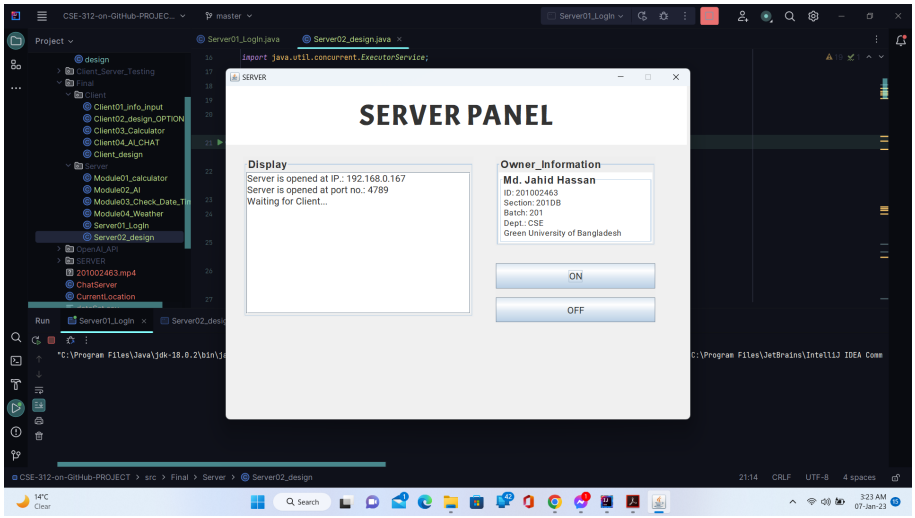

Figure 9.2: Server side design

```java
package Final.Server;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Server01_LogIn extends JFrame {
    Container c;
    Font font;
    private static String user = "Admin", pass = "2468";
    private static JLabel Heading, l1, l2;
    private static JTextField userName;
    private static JPasswordField password;
    private static JButton btn1, btn2;

    Server01_LogIn() {
        initComponent();
    }

    public void initComponent() {
        c = this.getContentPane();
        c.setLayout(new GridLayout(3, 2, 8, 8));
        c.setBackground(new Color(240, 240, 240));
        font = new Font("Arial", Font.PLAIN, 20);

        /*====================This section is for UserNmae panel
           ====================*/
        l1 = new JLabel("User Name: ");
        l1.setFont(font);
        l1.setHorizontalAlignment(JLabel.CENTER);
        c.add(l1);

        userName = new JTextField();
        userName.setFont(font);
        c.add(userName);
        /*====================This section is for UserNmae panel
           ====================*/

        /*====================This section is for Password panel
           ====================*/
        l2 = new JLabel("Password: ");
        l2.setFont(font);
        l2.setHorizontalAlignment(JLabel.CENTER);
        c.add(l2);

        password = new JPasswordField();
```

```
45          password.setEchoChar('*');
46          password.setFont(font);
47          c.add(password);
48          /*====================This section is for Password panel
                ====================*/
49
50          /*====================This section is for buttonPanel
                ====================*/
51          btn1 = new JButton("CLEAR");
52          btn1.setFont(font);
53          c.add(btn1);
54          btn2 = new JButton("SUBMIT");
55          btn2.setFont(font);
56          c.add(btn2);
57          /*====================This section is for buttonPanel
                ====================*/
58
59          /*====================This section is for ActionListener
                ====================*/
60          Handler handler = new Handler();
61          userName.addActionListener(handler);
62          password.addActionListener(handler);
63          btn1.addActionListener(handler);
64          btn2.addActionListener(handler);
65          /*====================This section is for ActionListener
                ====================*/
66      }
67
68      /*====================This section is for ActionListener
            Implementation====================*/
69      class Handler implements ActionListener {
70
71          @Override
72          public void actionPerformed(ActionEvent e) {
73              if (e.getSource() == userName || e.getSource() == password
                    || e.getSource() == btn2) {
74                  if (userName.getText().equals(user) && password.getText
                        ().equals(pass)) {
75                      JOptionPane.showMessageDialog(null, "Successfully
                            Login");
76                      Server02_design new_frame = new Server02_design();
77                      new_frame.setVisible(true);
78                      dispose();
79                  } else {
80                      JOptionPane.showMessageDialog(null, "Invalid
                            username or password");
81                      userName.setText("");
```

```
                    password.setText("");
                }
            } else {
                userName.setText("");
                password.setText("");
            }
        }
    }
    /*====================This section is for ActionListener
       Implementation====================*/

    /*====================This section is for Driver Code
       ====================*/
    public static void main(String[] args) {
        Server01_LogIn logIn_frame = new Server01_LogIn();
        logIn_frame.setDefaultCloseOperation(WindowConstants.
           EXIT_ON_CLOSE);
        logIn_frame.setSize(300, 150);
        logIn_frame.setLocationRelativeTo(null);
        logIn_frame.setVisible(true);
        logIn_frame.setResizable(false);
        logIn_frame.setTitle("Server logIn");
    }
    /*====================This section is for Driver Code
       ====================*/
}
```

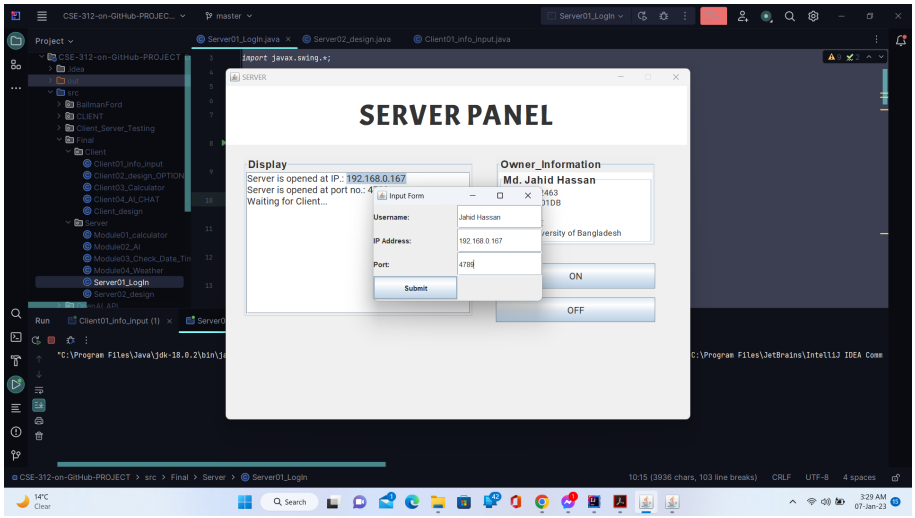

Figure 9.3: User input which server wants to connect and port

```java
package Final.Client;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Client02_design_OPTION extends JFrame implements
   ActionListener {
    Container c;
    JButton btn1, btn2, btn3;
    String username, ip, port;

    public Client02_design_OPTION(String username, String ip, String
       port) {
        this.username = username;
        this.ip = ip;
        this.port = port;
        initComponents();   // Calling initComponents method
    }
    public void initComponents() {
        /*====================This section is for Frame
           ====================*/
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(250, 350);
        this.setLocationRelativeTo(null);
        this.setVisible(true);
        this.setResizable(false);
        this.setTitle("Client Option");

        c = this.getContentPane();
        c.setLayout(null);
        c.setBackground(new Color(240, 240, 240));
        /*====================This section is for Frame
           ====================*/

        btn1 = new JButton("Calculator");
        btn2 = new JButton("Chat with AI");
        btn3 = new JButton("Chat with Friend");

        btn1.setBounds(50, 50, 150, 50);
        btn2.setBounds(50, 150, 150, 50);
        btn3.setBounds(50, 250, 150, 50);

        btn1.addActionListener(this);
        btn2.addActionListener(this);
        btn3.addActionListener(this);

```

```
44          add(btn1);
45          add(btn2);
46          add(btn3);
47      }
48
49      public void actionPerformed(ActionEvent e) {
50          if (e.getSource() == btn1) {
51              Client03_Calculator new_frame = new Client03_Calculator(
                    this.username, this.ip, this.port);
52              new_frame.setVisible(true);
53              dispose();
54              // do something
55          } else if (e.getSource() == btn2) {
56              Client04_AI_CHAT new_frame = new Client04_AI_CHAT(this.
                    username, this.ip, this.port);
57              new_frame.setVisible(true);
58              dispose();
59
60              // do something
61          } else {
62              // do something
63          }
64      }
65
66      public static void main(String args[], String username, String ip,
            String port) {
67          new Client02_design_OPTION(username, ip, port);
68      }
69  }
```

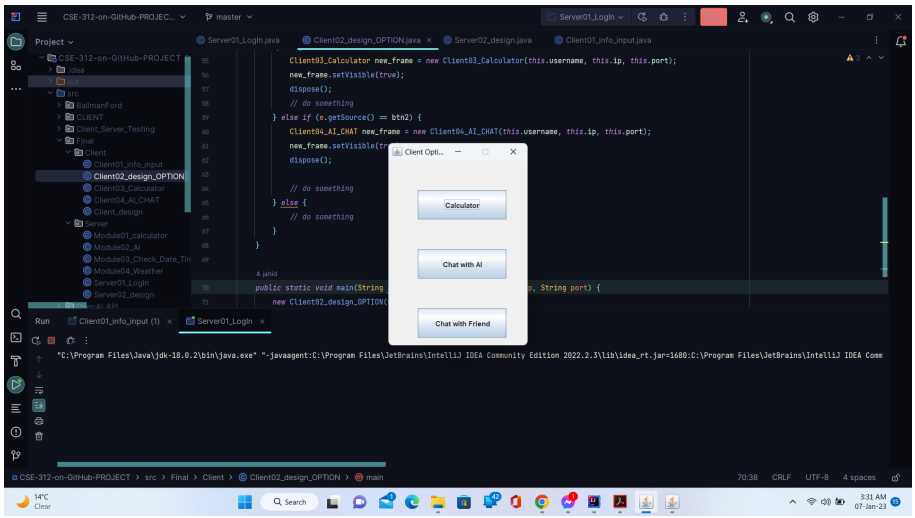

Figure 9.4: User choose which operation wants to do

```java
package Final.Client;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.InetAddress;
import java.net.Socket;
import java.util.Scanner;

public class Client03_Calculator extends JFrame implements
   ActionListener {
    private JTextField txtField1, txtField2;
    private JButton btnSubmit, btnBack;
    Container c;
    Socket clientSocket = null;
    Scanner in;
    PrintStream ps;
    BufferedReader br;
    String line;
    String result;
    String username, ip, port;


    public Client03_Calculator(String username, String ip, String port)
        {
        this.username = username;
        this.ip = ip;
        this.port = port;

        initComponents();   // Calling initComponents method
    }

    public void initComponents() {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(250, 200);
        this.setLocationRelativeTo(null);
        this.setVisible(true);
        this.setTitle("My JFrame Design");
//        this.setLayout(new FlowLayout());

        c = this.getContentPane();
        c.setLayout(new FlowLayout());
```

```
46
47          txtField1 = new JTextField(20);
48          txtField2 = new JTextField(20);
49          btnSubmit = new JButton("Submit");
50          btnBack = new JButton("Back");
51
52          add(txtField1);
53          add(txtField2);
54          add(btnSubmit);
55          add(btnBack);
56
57          try {
58              clientSocket = new Socket(InetAddress.getByName(ip),
                    Integer.parseInt(port));
59              ps = new PrintStream(clientSocket.getOutputStream());
60              br = new BufferedReader(new InputStreamReader(clientSocket.
                    getInputStream()));
61          } catch (IOException ex) {
62              throw new RuntimeException(ex);
63          }
64
65          btnSubmit.addActionListener(this);
66          btnBack.addActionListener(this);
67      }
68
69      public void actionPerformed(ActionEvent e) {
70          if (e.getSource() == btnSubmit) {
71              try {
72
73                  line = "calculatorBtn " + txtField1.getText();
74                  ps.println(line);           // Send to server
75                  result = br.readLine();     // Receive from server
76                  txtField2.setText(result);
77              } catch (IOException ex) {
78                  throw new RuntimeException(ex);
79              }
80          }
81          if (e.getSource() == btnBack) {
82              Client02_design_OPTION new_frame = new
                    Client02_design_OPTION(username, ip, port);
83              new_frame.setVisible(true);
84              dispose();
85          }
86      }
87
88
89      public static void main(String[] args, String username, String ip,
```

```
        String port) {
90          new Client03_Calculator(username, ip, port);
91      }
92  }
```

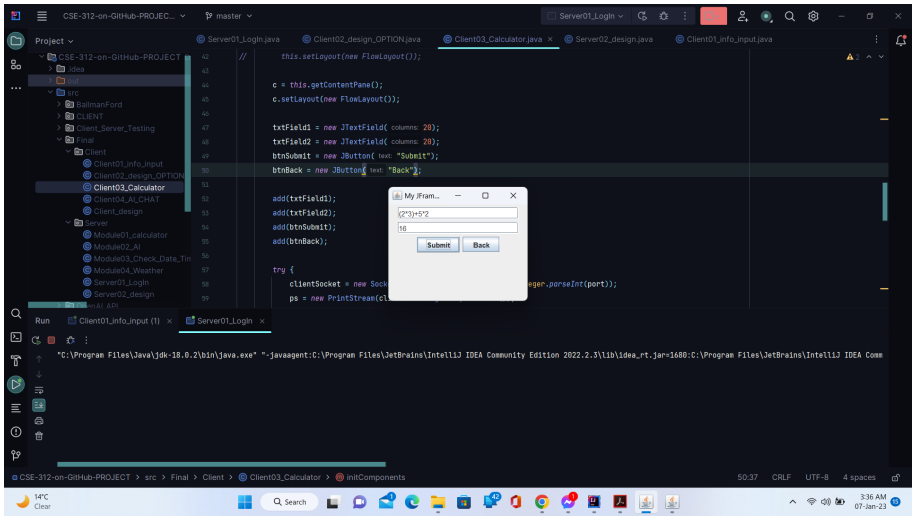

Figure 9.5: This is the Calculator frame

```
1   package Final.Client;
2
3   import javax.swing.*;
4   import java.awt.*;
5   import java.awt.event.ActionEvent;
6   import java.awt.event.ActionListener;
7   import java.io.BufferedReader;
8   import java.io.IOException;
9   import java.io.InputStreamReader;
10  import java.io.PrintStream;
11  import java.net.InetAddress;
12  import java.net.Socket;
13
14  public class Client04_AI_CHAT extends JFrame implements ActionListener
        {
15      Container c;
16
17      // Create a JPanel to hold the conversation
18      private JPanel conversationPanel;
19
20      // Create two JTextArea to display conversations
21      private JTextArea userTextArea;
22      private JTextArea botTextArea;
23
24      // Create a JScrollPane to scroll through the conversations
```

```
25      private JScrollPane conversationScrollPane;
26
27      // Create a JPanel to hold the user input
28      private JPanel inputPanel;
29
30      // Create a JTextField to take user input
31      private JTextField userInputField;
32
33      // Create a JButton to send the user input
34      private JButton sendButton, backButton;
35
36      // Create a String to store user input
37      private String userInput;
38
39      Socket clientSocket = null;
40      PrintStream ps;
41      BufferedReader br;
42      String line;
43      String result;
44      String username, ip, port;
45
46      public Client04_AI_CHAT(String username, String ip, String port) {
47          this.username = username;
48          this.ip = ip;
49          this.port = port;
50
51          initComponents();   // Calling initComponents method
52      }
53
54      public void initComponents() {
55          this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
56          this.setSize(400, 500);
57          this.setLocationRelativeTo(null);
58          this.setResizable(true);
59          this.setTitle("AI Chatbot");
60
61          c = this.getContentPane();
62          c.setLayout(new BorderLayout());
63          c.setBackground(new Color(240, 240, 240));
64
65
66          // Initialize the conversation panel
67          conversationPanel = new JPanel();
68          conversationPanel.setLayout(new BoxLayout(conversationPanel,
                BoxLayout.PAGE_AXIS));
69
70          // Initialize the user text area
```

```
71          userTextArea = new JTextArea();
72          userTextArea.setEditable(false);
73          userTextArea.setLineWrap(true);
74          userTextArea.setWrapStyleWord(true);
75
76          // Initialize the bot text area
77          botTextArea = new JTextArea();
78          botTextArea.setEditable(false);
79          botTextArea.setLineWrap(true);
80          botTextArea.setWrapStyleWord(true);
81
82          // Add the text areas to the conversation panel
83          conversationPanel.add(userTextArea);
84          conversationPanel.add(botTextArea);
85
86          // Initialize the conversation scroll pane
87          conversationScrollPane = new JScrollPane(conversationPanel);
88
89          // Add the scroll pane to the frame
90          this.add(conversationScrollPane, BorderLayout.CENTER);
91
92          // Initialize the input panel
93          inputPanel = new JPanel();
94          inputPanel.setLayout(new FlowLayout());
95
96          // Initialize the user input field
97          userInputField = new JTextField(24);
98
99          // Add the user input field to the input panel
100         inputPanel.add(userInputField);
101
102         // Initialize the send button
103         sendButton = new JButton("Send");
104         backButton = new JButton("Back");
105
106         // Add the send button to the input panel
107         inputPanel.add(sendButton);
108         inputPanel.add(backButton);
109
110         // Add the input panel to the frame
111         this.add(inputPanel, BorderLayout.SOUTH);
112
113
114         try {
115             clientSocket = new Socket(InetAddress.getByName(ip),
                    Integer.parseInt(port));
116             ps = new PrintStream(clientSocket.getOutputStream());
```

```
117             br = new BufferedReader(new InputStreamReader(clientSocket.
                    getInputStream()));
118         } catch (IOException ex) {
119             throw new RuntimeException(ex);
120         }
121 
122 
123         sendButton.addActionListener(this);
124         backButton.addActionListener(this);
125     }
126 
127 
128     @Override
129     public void actionPerformed(ActionEvent e) {
130         if (e.getSource() == sendButton || e.getSource() ==
                userInputField) {
131             try {
132                 userTextArea.append(userInputField.getText() + "\n");
133                 line = "chatBtn " + userInputField.getText();
134                 ps.println(line);           // Send to server
135                 result = br.readLine();     // Receive from server
136                 botTextArea.append(result + "\n");
137             } catch (IOException ex) {
138                 throw new RuntimeException(ex);
139             }
140         }
141         if (e.getSource() == backButton) {
142             Client02_design_OPTION new_frame = new
                    Client02_design_OPTION(username, ip, port);
143             new_frame.setVisible(true);
144             dispose();
145         }
146     }
147 
148     public static void main(String[] args, String username, String ip,
            String port) {
149         Client04_AI_CHAT frame = new Client04_AI_CHAT(username, ip,
                port);
150         frame.setVisible(true);
151     }
152 }
```

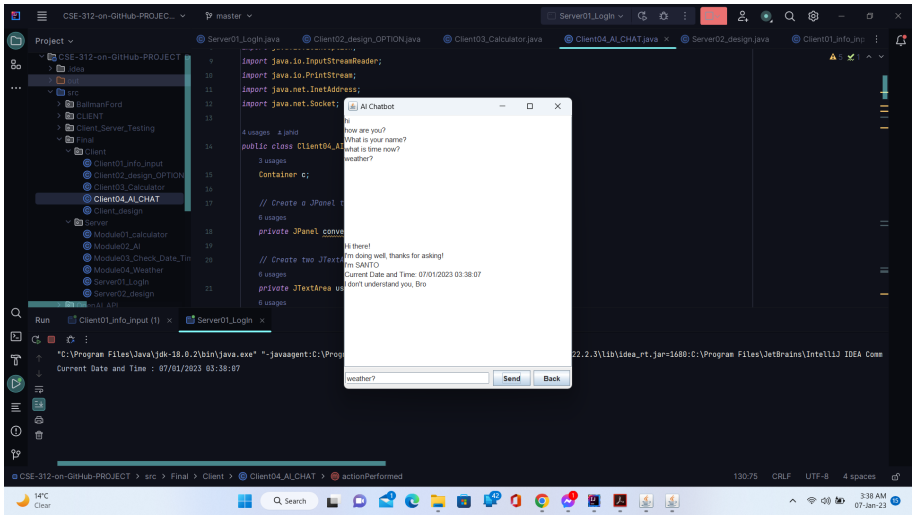

Figure 9.6: THis is the chat frame.

Chapter 10

Conclusion

10.1 Discussion

This project is a successful attempt to develop a Chatbot Companion which facilitates human-AI conversation and calculator on network devices using Java language. This project significantly impacts the ease of communication and automation of certain repetitive tasks. The development of this project was done using the Java language and a web application. This project has its limitations like the lack of natural language processing, limited conversation capability, and limited knowledge base. However, such limitations can be overcome with further development, and this project has plenty of scope to extend its capabilities in the future. Furthermore, this project can also be implemented in other languages such as Python, C++, and JavaScript. With the help of this project, we have seen that it is possible to develop a chatbot that can be used to carry out various tasks on network devices, and this project can be used as a reference for similar projects in the future.

10.2 Limitation

Here can't communicate with multiple clients. can't share files or documents.

10.3 Scope of Future Work

The scope of future work for this project can be to extend the functionality of the Chatbot Companion. This can include making the Chatbot Companion capable of carrying out more complex tasks, such as providing advice and suggestions, helping with scheduling and appointment setting, and providing entertainment options. Additionally, the Chatbot [4] Companion can be developed to be more conversational, so that it can engage in longer and more meaningful conversations with the user.

The project can also be extended to include a calculator that can be used to perform

network-related calculations, such as calculating subnets, IP addresses, and more. This would allow users to quickly calculate the values they need while configuring a network, as well as provide them with more detailed and accurate information.

Finally, the project can be extended to other platforms, such as mobile devices, web browsers, and other interfaces. This would allow users to access the Chatbot Companion from any location and on any device. Additionally, the user interface can be improved and optimized for each platform, making the overall experience more user-friendly. [5]

References

[1] Peter Wegner. Interoperability. *ACM Computing Surveys (CSUR)*, 28(1):285–287, 1996.

[2] Chiyoung Seo, Sam Malek, and Nenad Medvidovic. Component-level energy consumption estimation for distributed java-based software systems. In *International Symposium on Component-Based Software Engineering*, pages 97–113. Springer, 2008.

[3] Chung-Kai Chen, Cheng-Wei Chen, Chien-Tan Ko, Jenq-Kuen Lee, and Jyh-Cheng Chen. Mobile java rmi support over heterogeneous wireless networks: A case study. *Journal of Parallel and Distributed Computing*, 68(11):1425–1436, 2008.

[4] Rohit Rai. *Socket. IO Real-time Web Application Development.* Packt Publishing Ltd, 2013.

[5] Nuria Haristiani. Artificial intelligence (ai) chatbot as language learning medium: An inquiry. In *Journal of Physics: Conference Series*, volume 1387, page 012020. IOP Publishing, 2019.