Research Article

Implementation of Synchronous Messaging with Visual Basic as a Support for Learning the Parallel and Distributed Systems Course

[1,*]Muhammad Romi Nasution, [2]Basorudin

[1,2] Department of Information Systems, Faculty of Computer Science, Pasir Pengaraian University, Riau, Indonesia

* Corresponding Author: m.romi.nst@upp.ac.id

Abstract: There are two types of message delivery, namely synchronous and asynchronous messages. Synchronous communication refers to communication that occurs directly and simultaneously, where both parties interact in real-time. Common examples of synchronous communication in online learning include video conferencing, web conferencing, and live chat. Asynchronous communication, which is indirect, allows for more flexible learning through tools such as discussion forums, emails, and blogs. In this article, a chatting application will be designed and built as part of the implementation of the parallel and distributed systems course to support student learning. The application was built using Visual Basic software with important supporting components such as Winsock, TCP/IP protocol, and Local Port, to communicate directly (synchronously) between the client and server. From the test results of conducting chatting between the server and client, it can be concluded that the application runs well as expected without any errors, with an error rate of 0% and a success accuracy rate of 100%. For future researchers, this application can be used as teaching material in the parallel and distributed systems course, and can be developed with other models and the latest software, for example, not only sending messages but also sending files such as images and the like. This research approach can look at multiple sides, for example, from AI-Enhanced Communication, Hybrid Legacy-Modern Integration, Adaptive Protocol Switching, Dynamic Key Exchange, Auto-Discovery Port Management, ML-Optimized UI Components, Context-Aware UX, Predictive Load Balancing, Behavioral Pattern Analysis, AI-Assisted Bug Detection, QoE Assessment Model, dan Open Source Framework.

Keywords: Client-Server, TCP/IP Protocol, Synchronous, Visual Basic, Resource-constrained Environments, AI-Enhanced Communication

check for updates

Citation: Nasution, M. R., & Basorudin. (2025). Implementation of synchronous messaging with Visual Basic as a support for learning the Parallel and Distributed Systems course. *Iota*, 5(2). https://doi.org/10.31763/iota.v5i2.963

Academic Editor: Adi, P.D.P

Received: Maret 22, 2025

Accepted: April 25, 2025

Published: June 11, 2025

1. Introduction

In the continuously evolving era of digitalization, technology has become an important actor in various fields of our lives, one of which is in the field of education. One of the significant developments in the world of education is the application of technology to support learning. Technical and concept-based courses such as "Parallel and Distributed Systems" often require deep understanding and innovative learning approaches to help students grasp complex material. In the research by Indra Riyana Rahadjeng et al. (2022), it is explained that chat applications are usually used for chatting activities conducted by two or more people, either offline or over the internet. In the current era, chatting applications are rapidly evolving. In chatting applications, they are not only used for sending text messages, but chatting activities today can also be used to send various types of emoticons, image messages, files, data, audio messages, and even video messages. [1].

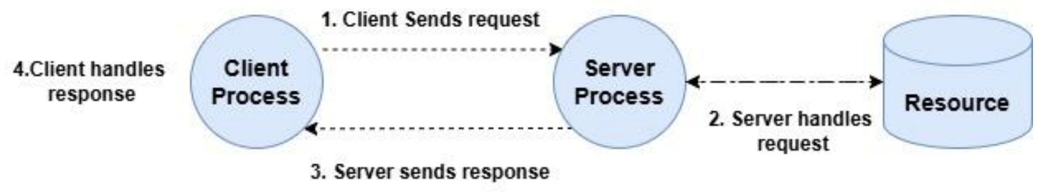


Figure 1. Message Transaction on Server-Client [7].

There are two types of message delivery, namely synchronous and asynchronous messages. Synchronous communication refers to communication that occurs directly and simultaneously, where both parties interact in real time. Common examples of synchronous communication in online learning include video conferencing, web conferencing, and live chat. Asynchronous communication, which is indirect, allows for more flexible learning through tools such as discussion forums, email, and blogs. [2]. Visual Basic, or more commonly known as VB, is a programming language developed by Microsoft.

Furthermore, first released in 1991, VB is part of the software development environment commonly referred to as Visual Studio. VB is designed to facilitate the development of Windows-based applications by providing an intuitive graphical interface. One of the main advantages of Visual Basic is its ease of use, especially for developers who are just starting out. [3]. In this article, a chat program or application will be designed and built as part of the implementation of the parallel and distributed systems course to support student learning. The application is built using Visual Basic software with essential supporting components such as Winsock, TCP/IP protocol, and Local Port, to communicate directly (synchronously) between the client and server.

Previous research conducted by Bambang Kelana Simpony (2017). Explains the importance of using WinSock in the development of socket-based applications for more reliable and secure network communication, as well as providing an understanding of how to build chat and messaging applications using TCP as the basic protocol [4]. Meanwhile, the function of the TCP/IP Protocol is used for communication between computers in a network, providing stability and efficiency in data transmission. [5]. In the course on parallel and distributed systems, one of the topics discussed is message exchange conducted synchronously and asynchronously. However, this research focuses solely on the implementation of synchronous message exchange. Therefore, this study aims to implement synchronous messaging, one of which is by building a chat application with the help of Visual Basic software. This application will be based on Client-Server architecture, with the expected outcome being that the application on the client can communicate with the application on the server in real-time.

The transfer of data and information through a network is very feasible to implement, as it can certainly accelerate and facilitate the process of data or information exchange. For example, the transfer of data and information carried out by a host or client is shown to the server. [6]. Figure 1 shows an example of Message Transmission on Server-Client, which consists of three basic components, i.e., Client and Server Process, and Resource, which consists of four activities, i.e., 1. Client Sent Request, 2. The server handles the request, 3. The server sends a response, and 4. Client handles response.

Every application on the network has its transactions based on the client-server concept. A server and a client, or several clients, request services from the server. The server's function is to manage the clients connected to it, in other words, to manage the existing resources, which will provide services by utilizing resources for the needs of the connected clients. [7]. The research conducted by Ripo Saputra et al. (2023) discusses a chat application that has been integrated with security using Classic Cryptography in its encoding. This research has been successfully developed, and the chat application will be safer if the process of sending and receiving messages includes data encryption. [8]. Another chatting application that has been developed by previous researchers is a multi-user chatting application, where the file transfer facility in the software can make communication between one user and another more effective, efficient, and practical, and can support users in completing their tasks. [9].

Besides being used as a chatting application, Visual Basic can also be used to design alarm programs. This research was successfully created by Andi Dwi Riyanto (2008). This alarm application can be applied as a reminder for schedules and agendas when someone is busy working or sitting in front of a computer. This alarm software is very helpful for its users in managing schedules and agendas in daily life, for example, as a reminder for waking up, study time, class time, and so on. [10]. The expected solution in this research is to design and build a chatting application, so that the development of this application will provide a learning solution, especially very useful for lecturers teaching parallel and distributed systems courses.

2. Method

In this research, the steps to be taken include preparing Visual Basic software to build a real-time chatting application. The components used are the TCP/IP protocol, local port, and several important components such as label, textbox, listbox, commandbutton, and Microsoft Winsock. Figure 2 shows the chart of the research methodology to be developed. This research can be developed comprehensively as shown in the development approach in Figure 3.

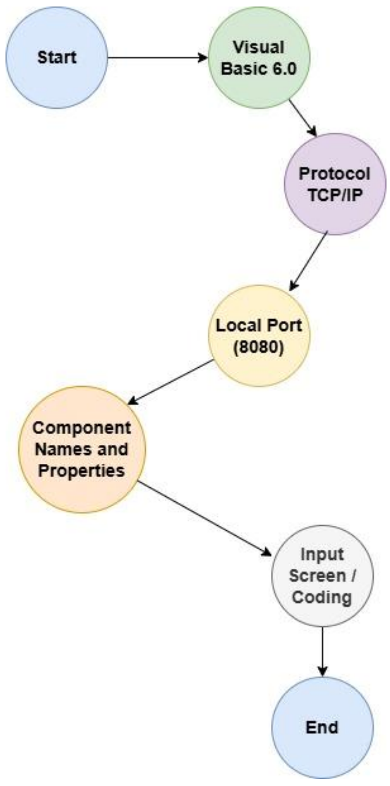


Figure 2. Research Methodology

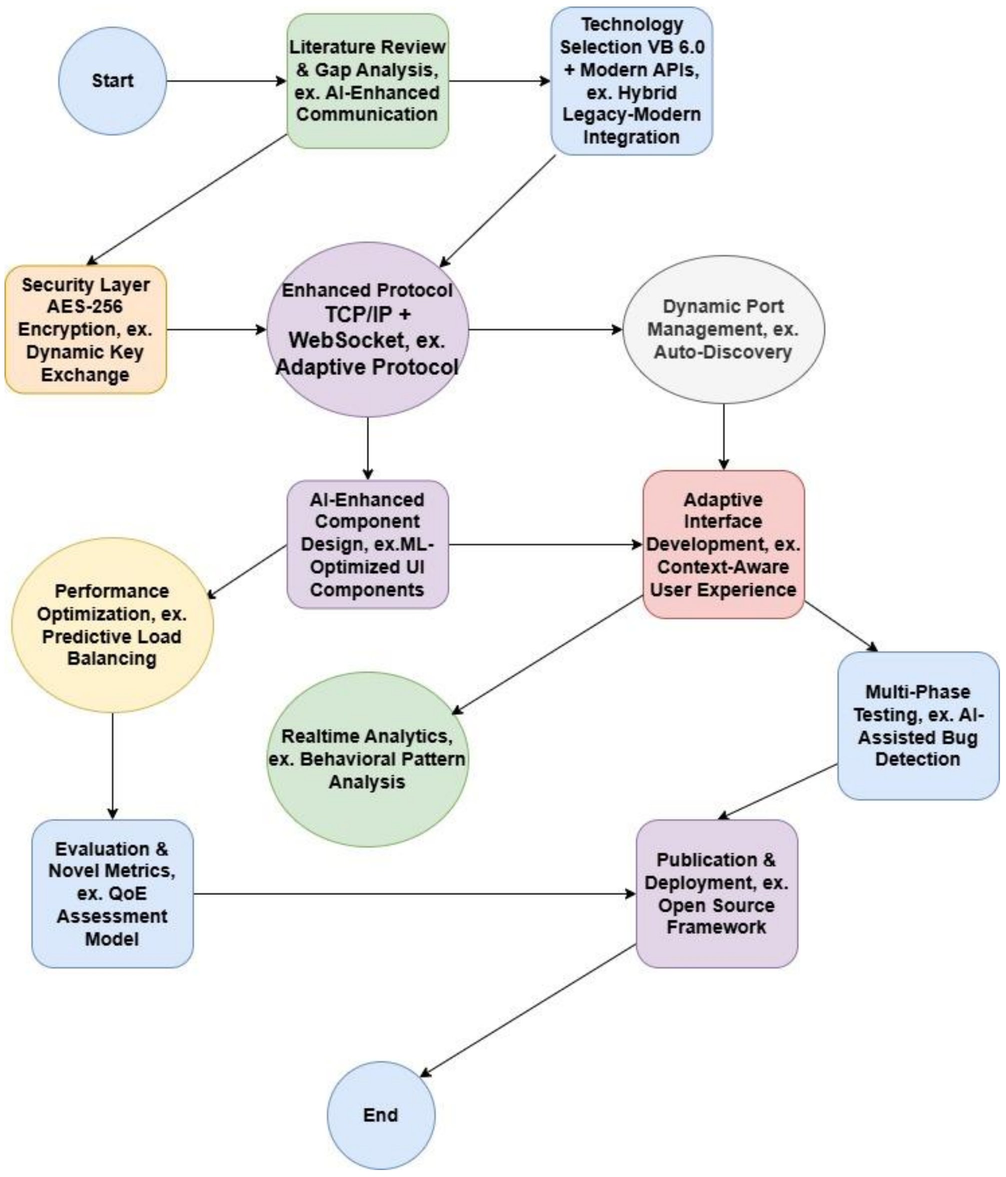
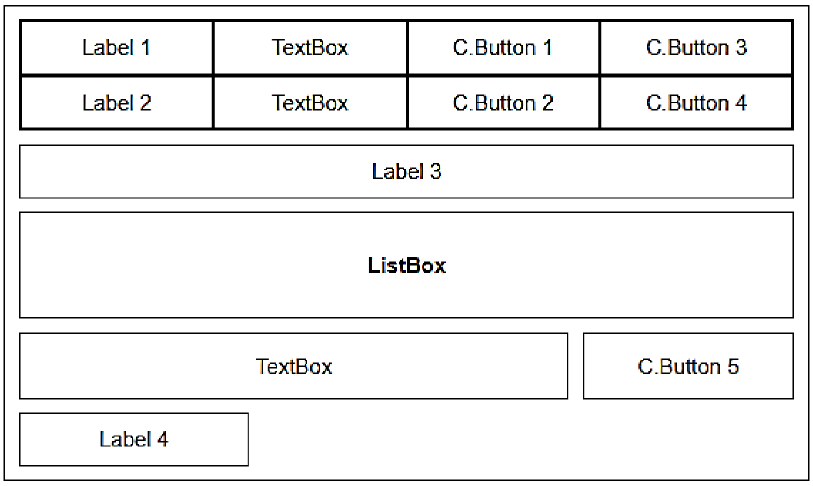

Start
Literature Review & Gap Analysis, ex. AI-Enhanced Communication
Technology Selection VB 6.0 + Modern APIs, ex. Hybrid Legacy-Modern Integration
Security Layer AES-256 Encryption, ex. Dynamic Key Exchange
Enhanced Protocol TCP/IP + WebSocket, ex. Adaptive Protocol
Dynamic Port Management, ex. Auto-Discovery
AI-Enhanced Component Design, ex.ML-Optimized UI Components
Adaptive Interface Development, ex. Context-Aware User Experience
Performance Optimization, ex. Predictive Load Balancing
Realtime Analytics, ex. Behavioral Pattern Analysis
Multi-Phase Testing, ex. AI-Assisted Bug Detection
Evaluation & Novel Metrics, ex. QoE Assessment Model
Publication & Deployment, ex. Open Source Framework
End

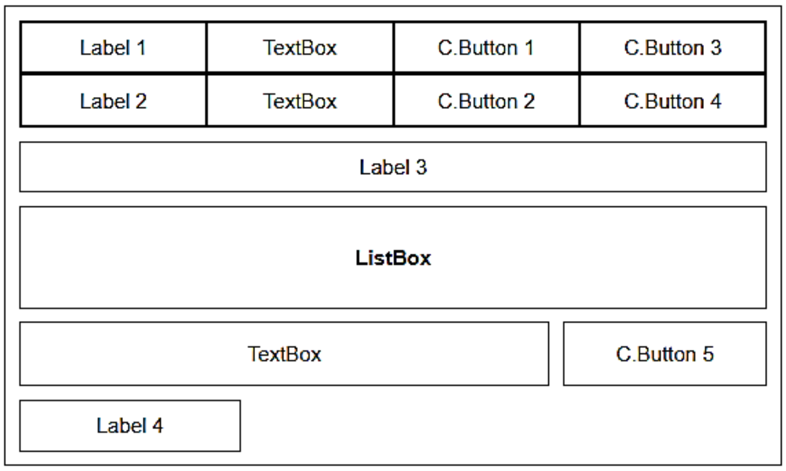
Figure 3. Approach to the development of research methodology

The Chat Platform as shown in Figure 3, has been integrated with Hybrid Legacy-Modern Integration as shown in Research (S. K. K L, S. M, S. A, S. J. U and V. S. P V. 2023), (S. Tirpude., et.al. 2024), Visual Basic development has led to a more flexible platform and can be applied to mobile applications such as research on the development of WhatsApp and LINE Chat. (S. Yaqub, et.al. 2024), (Y. Mitarai, et.al., 2023), (K. V. Rajkumar, et.al., 2024), (R. Leila, et.al., 2024), (V. A., S. LSS., et.al., 2025), (J. N. Singh, et.al., 2024). A Chat feature for conducting conversation analysis on Higher Education Students has also been developed and analyzed. And also other comprehensive research, do development from the platform side, analysis from Chat comprehensively (Y. U. Chandra and S. Ardiyansyah, 2023), (S. Singh, et.al., 2023, R. Rajasekar, et.al, 2023, Y. -A. Hsieh and N. -C. Tai, 2023, A. Seufert, et.al., 2023, N. Nie, et.al., 2023, R. Srishti Gulecha, et.al., 2023, M. M. Rovnyagin, et.al., 2024).

The methodology that will be developed in this research can be explained as follows: The first stage is preparing Visual Basic software to build a real-time chatting application. This software is chosen because of its ease in software development, especially for creating chatting applications by implementing components provided by Visual Basic itself. The second stage is the use of the TCP/IP Protocol, which serves as the basis for communication between computers in the network. This ensures stable and efficient communication between the client and server in the chat application. The third stage uses local port 8080, which will be used to ensure smooth communication between the client and server. This port allows connections to be opened between the two applications (client and server) on the network. The next stage is to implement the necessary components. These components are used to build the application interface and handle message delivery. The final stage, which is the most important, is the input screen or coding, which will ensure whether the designed application can run as expected or receive an error message.



(a)



(b)

Figure 4. (a, b) Client Application Design

Furthermore, from Figures 4a and b, the design is the same, but what distinguishes them is the caption. For example, on CommandButton 1 on the server, it is labeled "Listen," while on the client, it is labeled "Connect." Moreover, Table 1 is the Component Properties of the Server, while Table 2 is the Component Properties in the Client. Tables 1 and 2 can be explained that the component name column refers to the components used for application design, both server and client, while the name column is the property name that must be input according to the name of the component used, and the caption is used as the variable name for each component. The script used for the server can use the VB.NET code, which is described as follows the Pseudocode.

Table 1. Component Properties on the Server

No	**Component Name**	(Name)	Caption / Text	Action
1	Label	Label 1	IP Address	-
2	Label	Label 2	Local Port	-
3	TextBox	TxtServerIP	-	-
4	TextBox	TxtServerPort	-	-
5	CaommandButton	CmdListen	Listen	-
6	CaommandButton	CmdStop	Stop	-
7	CaommandButton	cmdClear	Clear Chat	-
8	CaommandButton	cmdExit	Exit	-
9	Label	LblStatus	Server is Closed…	-
10	ListBox	ListChat	-	-
11	TextBox	TxtMsg	-	-
12	CaommandButton	CmdSend	Send	-
13	Label	Label3	Copyright@2025	-
14	Winsock	wskServer	-	(Protocol) 0-sckTCPProtocol

Table 2. Properties Components on Client

No	Component Name	(Name)	Caption / Text	Action
1	Label	Label 1	Server IP	-
2	Label	Label 2	Server Port	-
3	TextBox	TxtServerIP	-	-
4	TextBox	TxtServerPort	-	-
5	CaommandButton	CmdConenct	Connect	-
6	CaommandButton	CmdDisconect	Disconenct	-
7	CaommandButton	cmdClear	Clear Chat	-
8	CaommandButton	cmdExit	Exit	-
9	Label	LblStatus	Ready to Connect…	-
10	ListBox	ListChat	-	-
11	TextBox	TxtMsg	-	-
12	CaommandButton	CmdSend	Send	-
13	Label	Label3	Copyright@2025	-
14	Winsock	wskClient	-	(Protocol) 0-sckTCPProtocol

Moreover, it is necessary to describe each component of the Socket Communication System in Pseudocode. Which consists of Main Program Structure, Server Side Operations, and components as detailed in the script.

Socket Communication System Pseudocode 1

Main Program Structure
```
DECLARE server socket
DECLARE client list
DECLARE user interface components (text fields, buttons, list box)
DECLARE connection status variables
```

Server Side Operations
Initialize Server
```
FUNCTION InitializeServer():
    SET server local port = user input port
    CREATE server socket
    SET server.Enabled = False
    SET connection listener enabled = False
    SET stop enabled = True
    DISPLAY "Server initialized"
END FUNCTION
```

Start Listening for Connections
```
FUNCTION StartListening():
    IF server state = connected THEN
        SET server.Listen()
        SET connection listener enabled = False
        SET stop enabled = False
        ADD TO chat list: "Server listening on port " + port
        SEND DATA through server socket
        SET message text = empty
    END IF
END FUNCTION
```

Handle Client Connection Request
```
FUNCTION HandleConnectionRequest(requestID):
    CLOSE server socket
    ACCEPT connection request with requestID
    SET status caption = "Connected to Client"
    ADD TO chat list: "Successfully Connected to Client..."
END FUNCTION
```

Handle Data Arrival from Client
```
FUNCTION HandleDataArrival(bytesTotal):
    GET DATA from server socket into message variable
    DISPLAY client message in events list: "<Client> " + message
END FUNCTION
```
Handle Server Errors
```
FUNCTION   HandleServerError(errorNumber,   description,   source,
helpFile, helpContext):
    IF error is critical THEN
```

```
        DISPLAY error message: "Fatal Error on connection. Check
IP Address/Port, try again"
        SET status = "Error"
    END IF
END FUNCTION
```

Client Side Operations

Load and Initialize Client

```
FUNCTION LoadClient():
    SET server IP text = local server IP
    SET server IP enabled = False
    SET server port text = default port
    SET stop enabled = False
END FUNCTION
```

Handle Server State Changes

```
FUNCTION HandleServerStateChange():
    IF server state = closed THEN
        CLOSE server connection
        SET connection listener click enabled = False
        SET stop enabled = False
        ADD TO chat list: "Waiting for Connection..."
        ADD TO chat list: "Disconnected..."
    END IF
END FUNCTION
```

Connect to Server

```
FUNCTION ConnectToServer():
    CLOSE any existing connections
    SET server.Accept request ID
    SET status caption = "Connected to Client"
    ADD TO chat list: "Successfully Connected to Client..."
END FUNCTION
```

Common Communication Functions

Send Message

```
FUNCTION SendMessage():
    IF connection is established THEN
        GET message from input field
        SEND message through socket
        ADD TO chat list: "<Server/Client> " + message
        CLEAR input field
    ELSE
        DISPLAY "Not connected" error
    END IF
END FUNCTION
```

Close Connection

```
FUNCTION CloseConnection():
    IF socket is connected THEN
        CLOSE socket connection
        SET status = "Disconnected from Client/Server"
        ADD TO chat list: "Disconnected..."
        RESET connection controls
    END IF
END FUNCTION
```

Event Handlers

Button Click Events

```
FUNCTION OnConnectButtonClick():
    CALL InitializeServer()
    CALL StartListening()
END FUNCTION

FUNCTION OnSendButtonClick():
    CALL SendMessage()
END FUNCTION

FUNCTION OnStopButtonClick():
    CALL CloseConnection()
END FUNCTION
```

Socket Event Handlers

```
FUNCTION OnConnectionRequest(requestID):
    CALL HandleConnectionRequest(requestID)
END FUNCTION

FUNCTION OnDataArrival(bytesTotal):
    CALL HandleDataArrival(bytesTotal)
END FUNCTION

FUNCTION OnError(errorNumber, description, source, helpFile,
helpContext):
    CALL HandleServerError(errorNumber, description, source,
helpFile, helpContext)
END FUNCTION
```

Main Program Flow

```
BEGIN MAIN PROGRAM
    INITIALIZE user interface
    INITIALIZE socket components
    SET default values for server IP and port
    ENABLE appropriate controls based on initial state

    WHILE application is running:
        LISTEN for user input events
        HANDLE socket events as they occur
        UPDATE user interface accordingly
    END WHILE

    ON APPLICATION EXIT:
        CLOSE all socket connections
        CLEANUP resources
END MAIN PROGRAM
```

Error Handling Strategy

```
FOR ALL socket operations:
    TRY:
        EXECUTE socket operation
    CATCH connection errors:
        DISPLAY appropriate error message
        RESET connection state
        ENABLE reconnection controls
    CATCH data transmission errors:
        LOG error details
        ATTEMPT to maintain connection
    END TRY-CATCH
```

Pseudocode Client Application or Pseudocode 2

Variable Declarations

```
DECLARE svIP As String
DECLARE svPort As String
DECLARE txtChat As String
DECLARE msg As String
DECLARE wskClient As WinsockClient
DECLARE TxtServerPort As TextBox
DECLARE TxtServerIP As TextBox
DECLARE TxtMsg As TextBox
DECLARE LblStatus As Label
DECLARE ListChat As ListBox
DECLARE CmdConnect As Button
DECLARE CmdDisconnect As Button
```

Event Handlers

1. Clear Button Click Event

```
PROCEDURE cmdClear_Click()
    CLEAR ListChat items
END PROCEDURE
```

2. Connect Button Click Event

```
PROCEDURE cmdConnect_Click()
    SET svPort = TxtServerPort.Text
    SET svIP = TxtServerIP.Text

    CALL wskClient.Close()
    SET wskClient.RemoteHost = svIP
    SET wskClient.RemotePort = svPort
    CALL wskClient.Connect()

    SET TxtServerPort.Enabled = False
    SET TxtServerIP.Enabled = False
    SET CmdConnect.Enabled = False
    SET CmdDisconnect.Enabled = True
    SET LblStatus.Caption = "Connecting to Server..."
END PROCEDURE
```

3. Disconnect Button Click Event

```
PROCEDURE cmdDisconnect_Click()
    CALL wskClient.Close()
    SET LblStatus.Caption = "Disconnected to Server"
    SET TxtServerPort.Enabled = True
    SET TxtServerIP.Enabled = True
    SET CmdConnect.Enabled = True
    SET CmdDisconnect.Enabled = False
    ADD "Disconnected..." TO ListChat
END PROCEDURE
```

4. Exit Button Click Event

```
PROCEDURE cmdExit_Click()
    END APPLICATION
END PROCEDURE
```

5. Send Button Click Event

```
PROCEDURE CmdSend_Click()
    SET msg = TxtMsg.Text

    IF wskClient.State = sckConnected THEN
        ADD "<Client> " + msg TO ListChat
        SEND msg TO wskClient
        CALL DoEvents()
        SET TxtMsg.Text = ""
    ELSE
        ADD "---Not Connected to Server---" TO ListChat
    END IF
END PROCEDURE
```

6. Form Load Event

```
PROCEDURE Form_Load()
    SET TxtServerPort.Text = "8080"
    SET TxtServerIP.Text = wskClient.LocalIP
    SET CmdDisconnect.Enabled = False
END PROCEDURE
```

7. Client Close Event

```
PROCEDURE wskClient_Close()
    IF wskClient.State <> sckClosed THEN
        SET LblStatus.Caption = "Connected to Server"
        ADD "---Succesfully Connected to Server---" TO ListChat
    END IF
END PROCEDURE
```

8. Data Arrival Event

```
PROCEDURE wskClient_DataArrival(ByVal bytesTotal As Long)
    CALL wskClient.GetData(msg)
    ADD "<Server> " + msg TO ListChat
    CALL DoEvents()
END PROCEDURE
```

```
9. Error Event Handler
PROCEDURE wskClient_Error(ByVal number As Integer, description As String, ByVal scode As Long, ByVal source As String, ByVal helpfile As String, ByVal helpcontext As Long, cancelDisplay As Boolean)

    DISPLAY "Could Not Connect to Server !!!" + vbCrLf + "Error"
END PROCEDURE

Main Program Flow
BEGIN PROGRAM
    INITIALIZE Form
    INITIALIZE Controls
    ENABLE Connect Button
    DISABLE Disconnect Button
    SET Default Server Port = "8080"
    SET Default Server IP = Local IP

    WAIT FOR User Events:
        - Connect Button Click
        - Disconnect Button Click
        - Send Button Click
        - Clear Button Click
        - Exit Button Click
        - Network Events (Data Arrival, Connection Close, Errors)

END PROGRAM
```

3. Results and Analysis

The result of this research is a chatting application that will allow communication between the server and client.

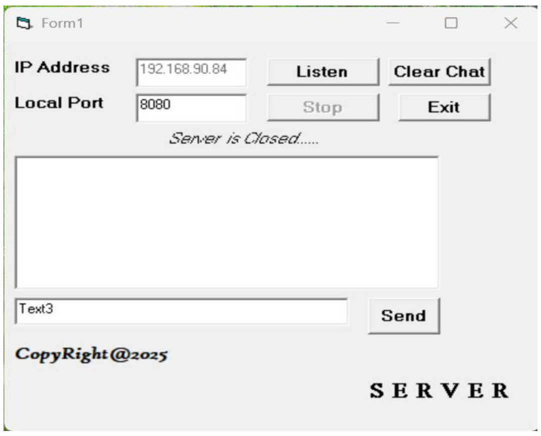
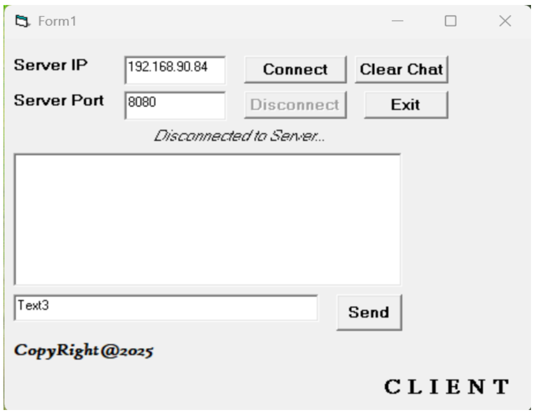


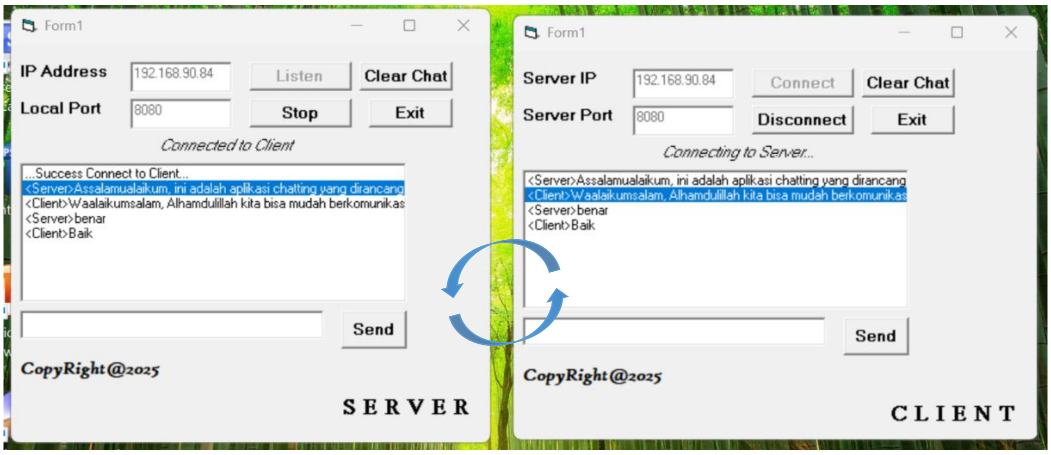
Figure 5. Chatting Client Application Design



Server IP 192.168.90.84
Server Port 8080
Disconnected to Server...
Text3

Figure 6. Client Chatting Application Design

Figures 5 and 6 are applications that have been designed and created, as explained in the descriptions of Figures 5 and 6. The difference between the server and client side lies in the name and caption. In terms of design, these applications are the same, but in terms of functionality, they have their respective roles. For example, in Figure 8, the left side of the image is the server and the right side is the client.



IP Address 192.168.90.84
Local Port 8080
Connected to Client

```
...Success Connect to Client...
<Server>Assalamualaikum, ini adalah aplikasi chatting yang dirancang
<Client>Waalaikumsalam, Alhamdulillah kita bisa mudah berkomunikas
<Server>benar
<Client>Baik
```

Server IP 192.168.90.84
Server Port 8080
Connecting to Server...

```
<Server>Assalamualaikum, ini adalah aplikasi chatting yang dirancang
<Client>Waalaikumsalam, Alhamdulillah kita bisa mudah berkomunikas
<Server>benar
<Client>Baik
```

Figure 7. The chat process from server to client and vice versa

Figure 7 shows that the application built is running as expected or functioning well and can communicate with each other synchronously or in real-time. The sentence typed on the server is sent to the client:

```
<server> Assalamualaikum, this is a chatting application
designed for real-time communication.
```

```
<client> Waalaikumsalam, Alhamdulillah we can easily
```

```
<server> true
```

```
<client> Good
```

4. Discussion

System development in this research can be developed towards Resource-constrained Environments, or in terms of a more comprehensive context, it can also focus on Hybrid Systems, Cross-Cultural Distributed Learning. In terms of evaluation and development of analysis, it can add a Learning Analytics Dashboard, Performance Prediction Model, Social Network Analysis, and Cognitive Load Assessment. When viewed from various Innovative Pedagogical aspects, some sample research developments are on Gamification-Driven, which integrates game mechanics elements, a collaborative Problem-Solving Framework, which is a collaborative problem-solving-based learning approach, building learning systems between students, such as peer-to-peer knowledge Exchange and Microlearning-based or learning in small modules.

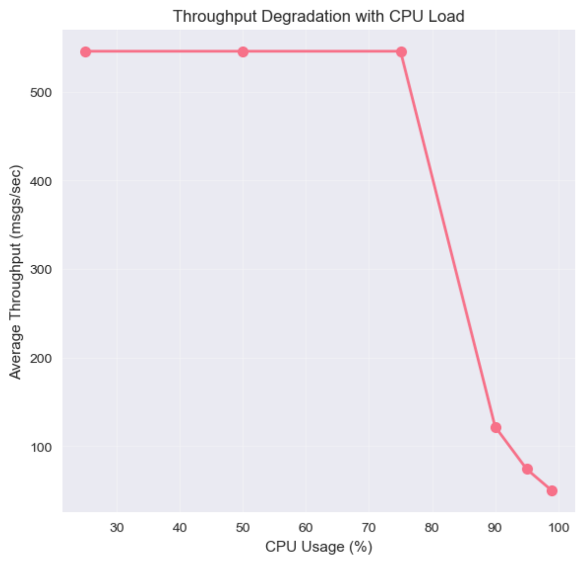


Figure 8. Resource-Constrained Environment Analysis for Synchronous Messaging, specifically on Throughput Degradation with CPU Load

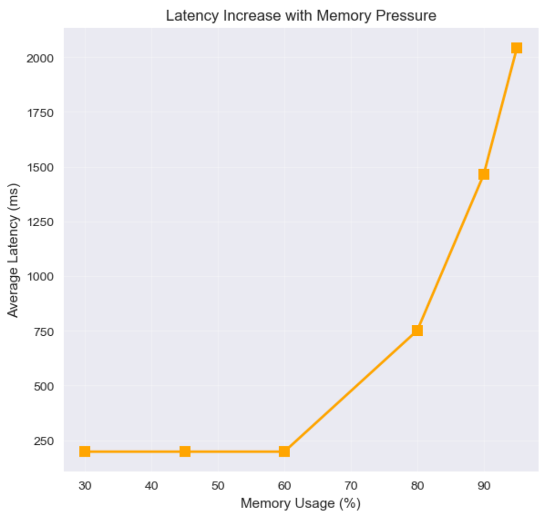


Figure 9. Resource-Constrained Environment Analysis for Synchronous Messaging, specifically on Latency Increase with Memory Pressure

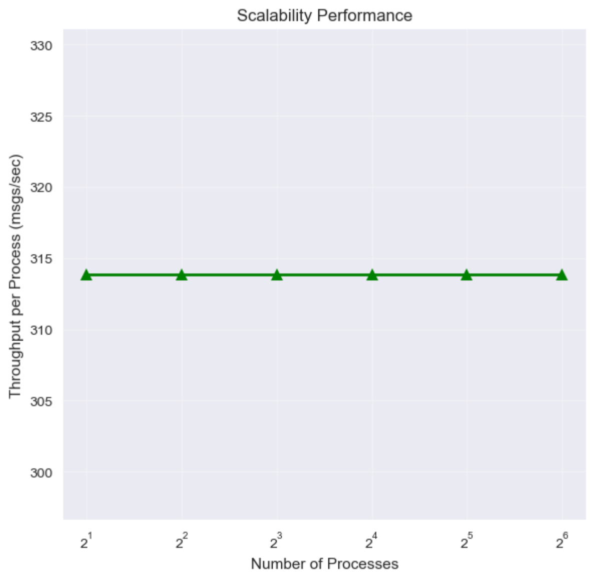


Figure 10. Resource-Constrained Environment Analysis for Synchronous Messaging, specifically on Scalability Performance

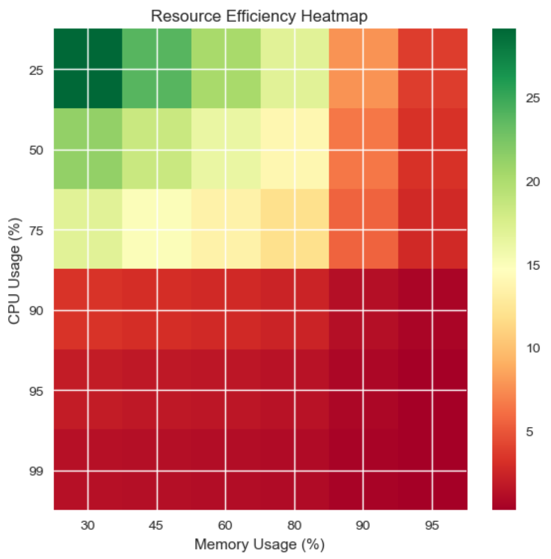


Figure 11. Resource-Constrained Environment Analysis for Synchronous Messaging, specifically on Resource Efficiency Heatmap

Furthermore, in the approach to the Resource-Constrained Environment, there are several analyses that need to be deepened, among others, this Messaging System will cause these three parameters, namely Throughput Calculation, Latency Modeling, and Success Rate. Moreover, Throughput is based on CPU, Memory, and Bandwidth, while Latency is based on contention considerations between processes, and success rate is determined by packet loss simulation under stress conditions.

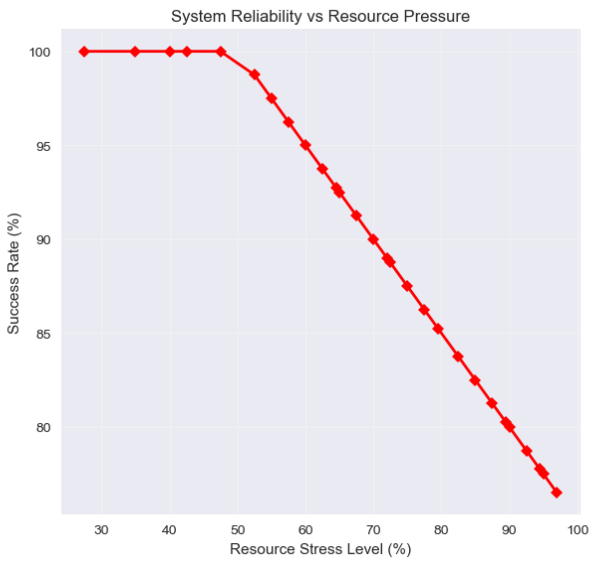


Figure 12. Resource-Constrained Environment Analysis for Synchronous Messaging, specifically on System Reliability vs Resource Pressure

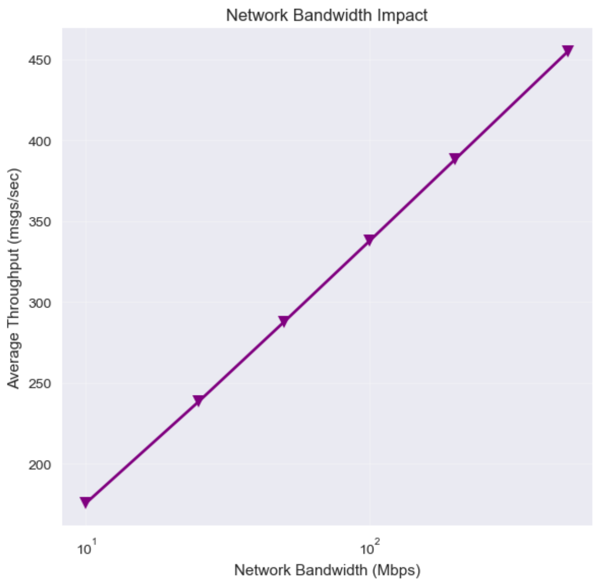


Figure 13. Resource-Constrained Environment Analysis for Synchronous Messaging, specifically on Network Bandwidth Impact

Figure 8 to 13 shows a comprehensive visualization of the Throughput vs CPU usage condition which shows performance degradation, then on the Latency vs Memory Pressure side is an attempt to increase delay when memory is high, then from the Scalability Analysis side which shows the performance side per process when scaling, then Resource Efficiency Heatmap which shows resource combination optimization, then Success Rate vs Resource Stress which shows System Reliability, and Bandwidth Impact which shows Diminishing Returns from network Speed. And Performance Metrics consisting of Resource Efficiency, Critical Thresholds, and Correlation Analysis. So that an approach to Intelligent Recommendations emerges consisting of optimal CPU usage

<75%, Memory Pressure <80%, Load Balancing for >16 processes, and Network bandwidth sweet spot ~ 50 Mbps. And the key insights generated are Performance Degradation Pattern, Resource Contention, Network Optimization, and Scalability Limit.

5. Conclusion and Future Research

From the results of the trial by conducting chatting between the server and client, it can be concluded that the application built meets the expectations, in other words, the application runs well without experiencing any errors at all, with an error rate of 0%, and a success accuracy rate of 100%. For future researchers, it is hoped that with the development of this application, it can be used as teaching material for teachers and lecturers in the course on parallel and distributed systems, and can be further developed with other models and the latest software, for example, not only sending messages but also being able to send files such as images and the like. This research approach can look at multiple sides, for example, from AI-Enhanced Communication, Hybrid Legacy-Modern Integration, Adaptive Protocol Switching, Dynamic Key Exchange, Auto-Discovery Port Management, ML-Optimized UI Components, Context-Aware UX, Predictive Load Balancing, Behavioral Pattern Analysis, AI-Assisted Bug Detection, QoE Assessment Model, dan Open Source Framework.

Acknowledgments: Thanks to the entire research team, and all the people in the Department of Information System, Faculty of Computer Science, Pasir Pengaraian University, Riau, who have helped the author in completing this research, hopefully this research can be maximally useful and applied, especially in the development of novelty research, such as AI-Enhanced Communication, and other parameters that have novelty value.

Author contributions: The authors are responsible for building Conceptualization, Methodology, analysis, investigation, data curation, writing—original draft preparation, writing—review and editing, visualization, supervision of project administration, funding acquisition, and have read and agreed to the published version of the manuscript.

Funding: The study was conducted without any financial support from external sources.

Availability of data and Materials: All data are available from the authors.

Conflicts of Interest: The authors declare no conflict of interest.
Additional Information: No Additional Information from the authors.

References

[1] I. R. Rahadjeng, M. N. H. Siregar, and A. P. Windarto, (2022). "Pemanfaatan Sistem Keputusan Dalam Mengevaluasi Penentuan Aplikasi Chatting Terbaik Dengan Multi Factor Evaluation Process," vol. 6. ISSN 2614-5278 (media cetak), ISSN 2548-8368 (media online) Available Online at https://ejurnal.stmik-budidarma.ac.id/index.php/mib DOI: 10.30865/mib.v6i2.4021.

[2] Mohamad Firdaus, Indra Bakti, (2024) Perancangan dan Pembuatan Desain Aplikasi OPNAME dengan Visual Basic Menggunakan Metode UML." Journal on Pustaka Cendekia Informatika Volume 1 No. 3, Januari-Maret 2024, pp 169-178.

[3] W. A. Prabowo and S. N. Hutagalung, "Perancangan Aplikasi Penyandian Pesan Chat Client dan Server Berdasarkan Algoritma Spritz," vol. 7, no. 3, 2020.

[4] M. Kannangara, C. Rajapakse, D. Asanka and N. Jayalath, "Development of a Chat Assistant Using Large Language Models for Personalized Mathematics Tutoring to Overcome Educational Disparities in Sri Lanka," 2025 5th International Conference on Advanced Research in Computing (ICARC), Belihuloya, Sri Lanka, 2025, pp. 1-6, doi: 10.1109/ICARC64760.2025.10963323.

[5] S. K. K L, S. M, S. A, S. J. U and V. S. P V, "Video Chat using WebRTC with In-built Sign Language Detection," 2023 Third International Conference on Ubiquitous Computing and Intelligent Information Systems (ICUIS), Gobichettipalayam, India, 2023, pp. 167-170, doi: 10.1109/ICUIS60567.2023.00036.

[6] S. Tirpude, Y. Thakre, S. Sudan, S. Agrawal, and A. Ganorkar, "Mining Comments and Sentiments in YouTube Live Chat Data," 2024 4th International Conference on Intelligent Technologies (CONIT), Bangalore, India, 2024, pp. 1-6, doi: 10.1109/CONIT61985.2024.10626352.

[7] S. Yaqub, S. Gochhait, H. A. H. Khalid, S. N. Bukhari, A. Yaqub and M. Abubakr, "WhatsApp Chat Analysis: Unveiling Insights through Data Processing and Visualization Techniques," 2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS), Manama, Bahrain, 2024, pp. 862-865, doi: 10.1109/ICETSIS61505.2024.10459604.

[8] Y. Mitarai, A. Mutoh, K. Shima, K. Moriyama, T. Matsui, and N. Inuzuka, "Automatic Construct of Communication Structures in LINE Group Chat," 2023 IEEE 12th Global Conference on Consumer Electronics (GCCE), Nara, Japan, 2023, pp. 310-313, doi: 10.1109/GCCE59613.2023.10315557.

[9] K. V. Rajkumar, N. Jayasree, L. V. Kommireddi, Y. Rayudu, M. Ikramuddin, and P. K. Aouti, "Voice Chat - Bringing Chats to Life Using Deep Learning," 2024 9th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2024, pp. 2001-2006, doi: 10.1109/ICCES63552.2024.10860258.

[10] R. Leila, P. Tazhibayeva, I. Abulkhair, D. Pogolovkin and K. Yelzhas, "Development of a Chat History Analysis Module Using Advanced Machine Learning Models and Vector Databases for Digital Forensic Tools," 2024 International Conference on Modeling, Simulation & Intelligent Computing (MoSICom), Dubai, United Arab Emirates, 2024, pp. 490-495, doi: 10.1109/MoSICom63082.2024.10882040.

[11] V. A, S. LSS, H. TS, K. Jaspin, A. K. Selvaraj, and N. D. M, "Visualization of Hidden Patterns in WhatsApp Chat Using Sentimental Analysis," 2025 3rd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), Bengaluru, India, 2025, pp. 8-13, doi: 10.1109/IDCIOT64235.2025.10914980.

[12] J. N. Singh, Y. Kumar, A. Srivastava, D. Baghel, and K. Al-Attabi, "WhatsApp Chat Analyzer," 2024 1st International Conference on Advances in Computing, Communication and Networking (ICAC2N), Greater Noida, India, 2024, pp. 78-83, doi: 10.1109/ICAC2N63387.2024.10895439.

[13] Y. U. Chandra and S. Ardiyansyah, "Analysis of the Use of E-Stickers in Chat Conversations for Higher Education Students," 2023 International Conference on Information Management and Technology (ICIMTech), Malang, Indonesia, 2023, pp. 413-417, doi: 10.1109/ICIMTech59029.2023.10277862.

[14] S. Singh, S. Singh and A. Sharma, "Real-Time Secure Web-Based Chat Application using Django," 2023 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2023, pp. 1560-1565, doi: 10.1109/ICAC3N60023.2023.10541532.

[15] R. Rajasekar, T. Shreya, P. Sandya, V. S. S. Reddy and B. Deepak, "Active Chat Monitoring and Detection Over Internet," 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2023, pp. 949-954, doi: 10.1109/ICICCS56967.2023.10142690.

[16] Y. -A. Hsieh and N.-C. Tai, "An Expressive Chat Room with Personalized Fonts Mimicking Actual Inconsistencies in Human Handwriting," 2023 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), PingTung, Taiwan, 2023, pp. 197-198, doi: 10.1109/ICCE-Taiwan58799.2023.10227039.

[17] A. Seufert, C. Baur, F. Poignée, M. Seufert and T. Hoßfeld, "Sitting, Chatting, Waiting: Influence of Loading Times on Mobile Instant Messaging QoE," 2024 16th International Conference on Quality of Multimedia Experience (QoMEX), Karlshamn, Sweden, 2024, pp. 100-103, doi: 10.1109/QoMEX61742.2024.10598245.

[18] N. Nie, H. Guo, and W. Song, "Authenticity Classification of WeChat Group Chat Messages Based on LDA and NLP," 2024 9th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), Chengdu, China, 2024, pp. 313-319, doi: 10.1109/ICCCBDA61447.2024.10569975.

[19] R. Srishti Gulecha, K. Muthu Reshmi, and S. Abirami, "Exploratory Data Analysis of WhatsApp group chat," 2023 12th International Conference on Advanced Computing (ICoAC), Chennai, India, 2023, pp. 1-6, doi: 10.1109/ICoAC59537.2023.10250143.

[20] M. M. Rovnyagin, D. M. Sinelnikov, A. A. Eroshev, T. A. Rovnyagina, and A. V. Tikhomirov, "Optimizing Cache Memory Usage Methods for Chat LLM-models in PaaS Installations," 2024 Conference of Young Researchers in Electrical and Electronic Engineering (ElCon), Saint Petersburg, Russian Federation, 2024, pp. 277-280, doi: 10.1109/ElCon61730.2024.10468250.