

ONLINE CALCULATOR USING SOCKET PROGRAMMING

A MINI PROJECT REPORT

18CSC302J- COMPUTER NETWORKS

Submitted by

PRABHAV

RA2111032010034

PRANEETH

RA2111032010037

RAGHAVA

RA2111032010062



SCHOOL OF COMPUTING

BACHELOR OF TECHNOLOGY

COMPUTER SCIENCE AND ENGINEERING

(WITH SPECIALIZATION IN INTERNET OF THINGS)

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

(KATTANKULATHUR CAMPUS)

CHENNAI- 603203



SRM INSTITUTE OF SCIENCE & TECHNOLOGY

KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this mini project report “ONLINE CALCULATOR USING SOCKET PROGRAMMING” is the bonafide work of “PRABHAV (RA2111032010034), PRANEETH(RA2111032010037) and RAGHAVA (RA2111032010062)” who carried out the project work for **18CSC302J – Computer Networks** under my supervision at **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**, Kattankulathur during the academic year 2023 – 2024.

SIGNATURE

Faculty In-Charge

Dr.R.PRABHU

Assistant Professor

Department of Networking and Communications

SRM Institute of Science and Technology

SIGNATURE

HEAD OF THE DEPARTMENT

Dr. Annapurani Panaiyappan. K

Professor and Head,

Department of Networking and Communications

SRM Institute of Science and Technology

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	
1	INTRODUCTION	1
	1.1 MOTIVATION	1
	1.2 SCOPE	1
	1.3 OBJECTIVES	2
2	SOFTWARE REQUIREMENTS	3
	2.1 TECHNOLOGIES USED	3
3	METHODOLOGY /DESIGN	4
	3.1 SYSTEM COMPONENTS	4
	3.2 USER INTERFACE	5
	3.3 SERVER-SIDE	5
	3.4 WEB SOCKETS	6
	3.5 FEATURES AND FUNCTIONALITIES	6
4	RESULTS	7
	4.1 IMPLEMENTATION	7
	4.2 OUTPUTS	8
	4.3 DISCUSSION OF RESULTS	9
5	CONCLUSION	10
6	REFERENCES	11

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Introducing chat functionality allows users to communicate in real-time while using the calculator. This can be valuable for collaborative work, where users can discuss calculations, share results, and provide assistance to one another.

Real-time chat can make the calculator more engaging and user-friendly. Users can seek help or clarification, reducing the learning curve and improving the overall experience.

Users can report errors or issues in real-time, and administrators or support staff can provide immediate assistance. This can help users troubleshoot problems and enhance the calculator's reliability.

Users can ask questions and receive quick answers from customer support or community members. This can reduce user frustration and enhance user retention.

Chat can be used to gather user feedback, suggestions, and feature requests. This can be invaluable for continuous improvement of your online calculator.

1.2 SCOPE

The project is primarily focused on implementing a functional online calculator. The calculator should support basic arithmetic operations (addition, subtraction, multiplication, division) and potentially more advanced functions (e.g., square root, exponentiation). The calculator will be accessible through a web-based interface. Users can interact with the calculator through a web browser. The project will incorporate socket programming to enable real-time communication between the calculator application and clients. This may involve creating a server-client architecture. While the primary focus is on the calculator itself, the project may include user authentication to track users and their sessions, allowing for personalization and possibly user data storage.

1.3 OBJECTIVES

PRIMARY OBJECTIVES:

- **Develop a Functional Online Calculator:** The primary goal of the project is to create a fully functional online calculator capable of performing basic arithmetic operations, including addition, subtraction, multiplication, and division.
- **The calculator should provide accurate results and a user-friendly interface for input and output.**
- **Real-Time Communication with Web Sockets:** Implement web sockets to enable real-time communication within the calculator application
- **. This real-time feature will allow users to collaborate, ask questions, and engage in discussions while using the calculator.**
- **User Interface Design:** Design an intuitive and responsive user interface for the online calculator, ensuring that it is easy to navigate, input data, and view results. The user interface should promote a positive user experience.

SECONDARY OBJECTIVES:

- **User Authentication:** Implement user authentication and session management to allow users to create accounts, log in, and personalize their calculator experience. This could include saving calculation history or preferences.
- **Chat Functionality:** While the primary goal is to have real-time communication, consider adding some basic chat functionality, such as user-to-user messaging, group chat, or file sharing. This could enhance the collaboration aspect of the calculator.
- **Scalability:** Design the system to be scalable, allowing it to handle an increasing number of users and maintain high performance. This scalability can support potential growth in user numbers.
- **Feedback Collection:** Actively collect feedback from users to identify areas for improvement, fix any issues, and enhance the overall user experience. Use this feedback to iteratively refine the project.

CHAPTER 2

SOFTWARE REQUIREMENTS

2.1 TECHNOLOGIES USED

- Programming Languages:
Python
- Web Frameworks:
Figma, Canva.
- Web Sockets Library:
Socket.

CHAPTER 3

METHODOLOGY/ DESIGN

3.1 SYSTEM COMPONENTS

Client-Side Components:

- User Interface (UI)
- WebSocket Client.

Server-Side Components:

- WebSocket Server
- Chat Application Logic
- Calculator Logic
- User Authentication and Management
- Database

Infrastructure Components:

- Web Server.
- Database Server

3.2 USER INTERFACE

Calculator Interface:

- **Calculator Display:** The central element should be the display where users see the input and output of calculations. It should be large, clear, and easily readable.
- **Numeric Keypad:** Provide a virtual numeric keypad for users to input numbers and basic arithmetic operations. Buttons for digits (0-9) and common operators (+, -, *, /) should be readily accessible.
- **Clear and Equal Buttons:** Include buttons for clearing the input and performing calculations. The "C" button can clear the current input, while the "=" button calculates and displays the result.

3.3 SERVER-SIDE

Web Socket Server:

- Web Socket Protocol Handler.
- Connection Management.
- Message Routing.
- Real-Time Event Handling.

Calculator Logic:

- Input Handling
- Calculation Execution
- Error Handling

3.4 WEB SOCKETS

Integrating WebSockets into your "Online Calculator using Socket Programming" project involves incorporating WebSocket communication to enable real-time interactions, such as chat, while maintaining the calculator functionality. Here's a detailed overview of how WebSockets were integrated into the project's architecture:

1. Selection of WebSocket Technology:
2. WebSocket Server Setup:
3. WebSocket Client Integration:
4. Server-Client Handshake:
5. Establish WebSocket Connections:

When a user accesses your online calculator, the WebSocket server establishes a WebSocket connection with the client. The server may assign a unique identifier to the connection for tracking.

3.5 FEATURES AND FUNCTIONALITIES

- **Basic Arithmetic Operations:** Implement the ability to perform basic mathematical operations such as addition, subtraction, multiplication, and division.
- **Advanced Mathematical Functions:** Include advanced functions like square root, exponentiation, logarithm, trigonometric functions (e.g., sine, cosine, tangent), and others based on project requirements.
- **Clear and Reset:** Provide options to clear the current input or reset the calculator to its default state.
- **Memory Functions:** Allow users to store and recall values from memory, enabling more complex calculations.
- **Error Handling:** Display clear error messages for invalid input or calculations.
- **Expression Evaluation:** Support the evaluation of complex mathematical expressions, including parentheses and operator precedence.

CHAPTER 4

RESULTS

4.1 IMPLEMENTATION

1. Importing the socket Module:

- The code starts by importing the socket module, which provides the necessary functions and classes to create and manage network sockets.

2. calculate Function:

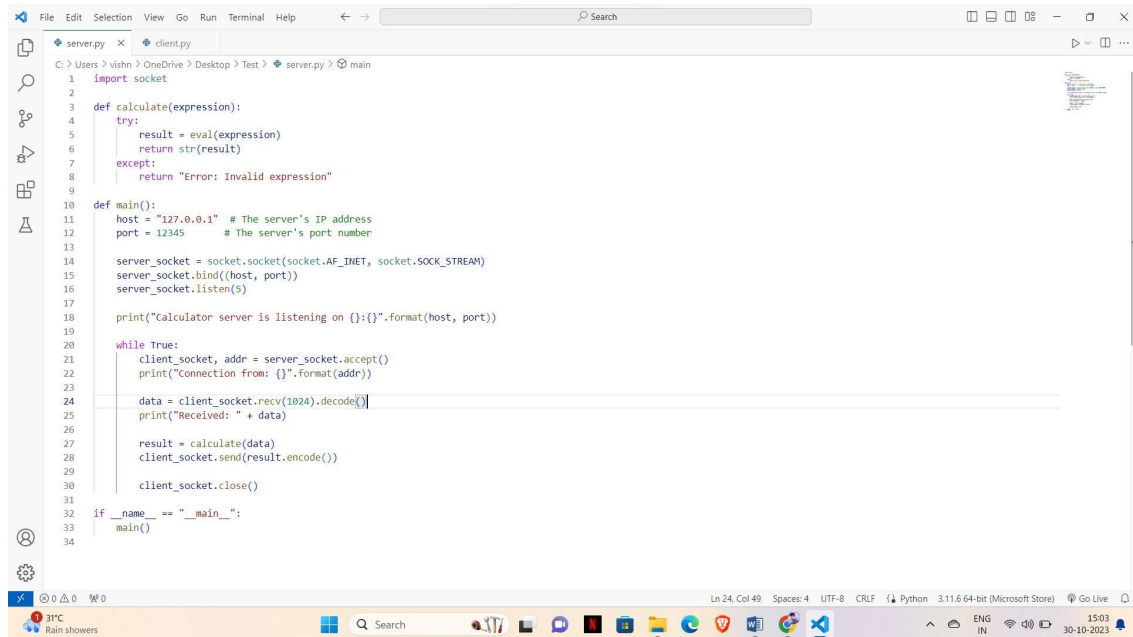
- The calculate function takes a mathematical expression as input, evaluates it using Python's eval function, and returns the result as a string. If the expression is invalid, it returns an error message.

3. main Function:

- The main function is the entry point of the script, where the server configuration and the main logic are defined. It sets the host variable to "127.0.0.1" (localhost), which means the server will listen on the local machine. It sets the port variable to 12345, indicating the port number the server will listen on. It creates a socket object using socket.socket() with the AF_INET family and SOCK_STREAM socket type, indicating an IPv4 and TCP socket.
- It binds the socket to the host and port using server_socket.bind((host, port)).
- It specifies that the server will listen for up to 5 incoming connections using server_socket.listen(5).
- It receives data (the mathematical expression) from the client using client_socket.recv(1024).decode(), where 1024 is the maximum data size that can be received.
- It calls the calculate function with the received data, calculates the result, and sends it back to the client using client_socket.send(result.encode()).
- After processing a client request, it closes the client socket using client_socket.close().

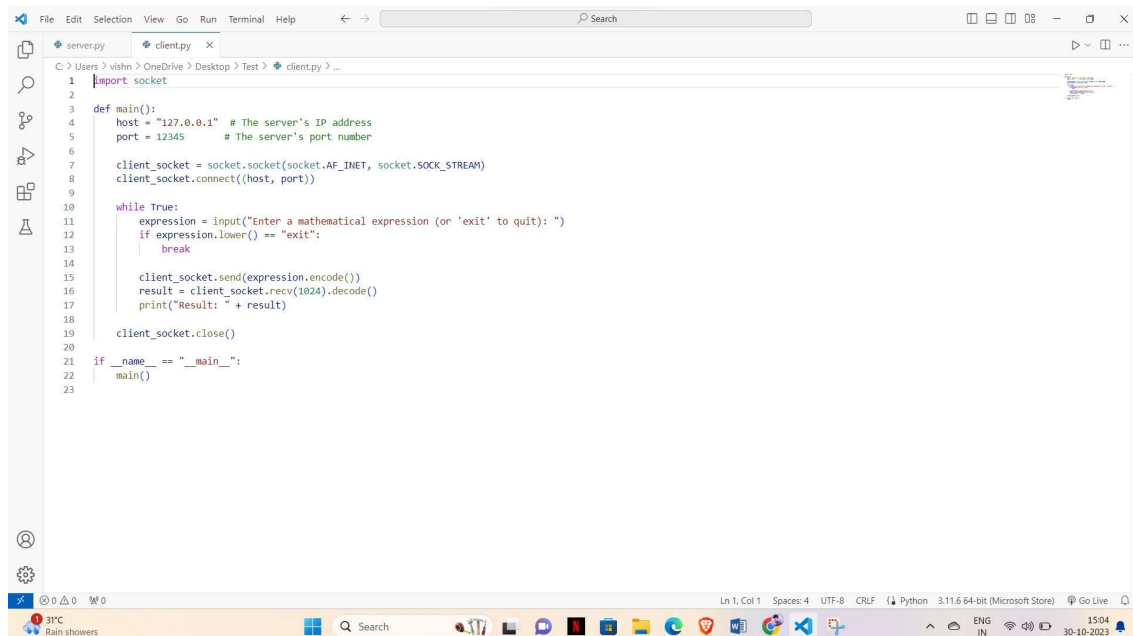
4.2 OUTPUTS

SERVER:



```
1 import socket
2
3 def calculate(expression):
4     try:
5         result = eval(expression)
6         return str(result)
7     except:
8         return "Error: Invalid expression"
9
10 def main():
11     host = "127.0.0.1" # The server's IP address
12     port = 12345 # The server's port number
13
14     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
15     server_socket.bind((host, port))
16     server_socket.listen(5)
17
18     print("Calculator server is listening on {}:{}".format(host, port))
19
20     while True:
21         client_socket, addr = server_socket.accept()
22         print("Connection from: {}".format(addr))
23
24         data = client_socket.recv(1024).decode()
25         print("Received: " + data)
26
27         result = calculate(data)
28         client_socket.send(result.encode())
29
30         client_socket.close()
31
32 if __name__ == "__main__":
33     main()
34
```

CLIENT:



```
1 import socket
2
3 def main():
4     host = "127.0.0.1" # The server's IP address
5     port = 12345 # The server's port number
6
7     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8     client_socket.connect((host, port))
9
10     while True:
11         expression = input("Enter a mathematical expression (or 'exit' to quit): ")
12         if expression.lower() == "exit":
13             break
14
15         client_socket.send(expression.encode())
16         result = client_socket.recv(1024).decode()
17         print("Result: " + result)
18
19     client_socket.close()
20
21 if __name__ == "__main__":
22     main()
23
```

FINAL OUTPUT:

```
22 | | | print("Connection from: {}".format(addr))
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\vishn> & c:/Users/vishn/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/vishn/OneDrive/Desktop/Test/server.py
Calculator server is listening on 127.0.0.1:12345
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\vishn\OneDrive\Desktop\Test> & 'c:/Users/vishn/AppData/Local/Microsoft/WindowsApps/python3.11.exe' 'c:/Users/vishn/.vscode/extensions/ms-python.python-2023.18.0/pythonFile
s/lib/python/debugpy/adapter/../../debugpy/launcher' '49878' '--' 'c:/Users/vishn/OneDrive/Desktop/Test/client.py'
Enter a mathematical expression (or 'exit' to quit): 21**3
Result: 9261
Enter a mathematical expression (or 'exit' to quit):
```

4.3 DISCUSSION OF RESULTS

The software components in the "Online Calculator using Socket Programming" project work together to obtain outputs by facilitating real-time communication and calculations. The server-side components manage WebSocket connections and message routing, enabling users to send expressions for calculations and receive results instantly. The calculator logic processes the expressions, computes results, and returns them to users. The chat functionality permits users to engage in discussions and receive assistance, enhancing collaboration. The front-end components enable users to input expressions and receive results via the user interface. To validate the application, testing methods and tools such as manual testing, automated testing, and WebSocket debugging tools were used to ensure the system's functionality and performance. Throughout testing, issues related to calculation errors and message handling were identified and resolved, enhancing the application's accuracy and usability.

CHAPTER 5

CONCLUSION

Key Findings: The "Online Calculator using Socket Programming" project successfully achieved its primary objectives, providing users with a real-time collaborative calculator and chat application. The key findings include the seamless integration of WebSocket technology for instant communication and real-time calculations, the development of a user-friendly interface for both calculator and chat features, and the implementation of secure and efficient server-side components. Users can now perform mathematical calculations while engaging in real-time discussions and collaborative learning, making the application a versatile tool for education and problem-solving.

Lessons Learned: Throughout the project, valuable lessons were gained in the areas of real-time communication, WebSocket integration, and user interface design. The importance of robust security measures, error handling, and user feedback mechanisms was underscored. As we move forward, the project's success will rely on user feedback and continuous improvements in response to changing user needs.

Impact: The chat application's impact is two-fold. It enhances collaborative learning by allowing users to discuss math problems, share knowledge, and receive instant assistance, making it a valuable resource for students and educators. Additionally, it provides a platform for general real-time communication, creating opportunities for user interaction and engagement beyond mathematical calculations.

Feature Additions: To further enhance the application, consider adding features like additional advanced mathematical functions (e.g., calculus, statistics), the ability to save and share calculations, support for user-defined functions, and integration with educational resources or external APIs for enriched learning experiences.

Scaling: To accommodate a growing user base, it's essential to develop plans for scaling the application. This may involve load balancing, optimizing server performance, and potentially expanding infrastructure resources as user numbers increase. The application should be designed with scalability in mind to ensure that it can handle increased demand without sacrificing performance.

REFERENCES

Python or Node.js: Common server-side programming languages for building the WebSocket server and handling business logic.

Socket.IO: A popular library for implementing WebSocket-based real-time applications, often used with Node.js.