

MSCS-632: Residency Project: Deliverable 2 Report: Simple Chat Application: Core Functionality

Implementation

Shabnam Shaikh, Sakchham Sangroula, Sandesh Pokharel, Nihar Turumelle, Romika Souda

University of the Cumberlands

MSCS-632 Advanced Programming Languages

Dr. Vanessa Cooper

April 05, 2025

1. Introduction

This document outlines the **core functionality implementation** for our chat application in two languages: **Rust** and **Go**. Both implementations are functional, meet the assignment's key requirements, and make use of language-specific features.

The purpose of this deliverable is to highlight how each language handles the same problem differently — from design to execution — using appropriate data structures, concurrency models, and memory management tools available in each ecosystem.

GitHub Repo: <https://github.com/sanspokharel26677/MSCS-632-Residency-Project>

2. Rust Implementation

Our Rust version is a **turn-based chat system** that allows users to take turns sending messages. The application uses a combination of structs, a message queue, and the tokio runtime for async behavior.

Structs and Memory Safety

Rust relies heavily on strong typing and memory safety. We defined a Message struct and used VecDeque to store chat history. Shared access is managed using Arc<Mutex<_>>.

```
struct Message {  
    user_id: String,  
    content: String,  
    timestamp: String,
```

```
    }

struct ChatServer {

    messages: VecDeque<Message>,

    message_count: HashMap<String, usize>,

    active_users: Vec<String>,

}
```

 **Screenshot** : main.rs struct definitions and shared state initialization.

```
11 // Struct to represent a message      You, 1 hour ago • Added chat app with rust implementation
12 struct Message {
13     user_id: String,
14     content: String,
15     timestamp: String,
16 }
17
18 // Chat server struct to store messages and stats
19 struct ChatServer {
20     messages: VecDeque<Message>,
21     message_count: HashMap<String, usize>,
22     active_users: Vec<String>,
23 }
24
25 impl ChatServer {
26     fn new(users: Vec<String>) -> Self {
27         ChatServer {
28             messages: VecDeque::new(),
29             message_count: HashMap::new(),
30             active_users: users,
31         }
32     }
33
34     fn add_message(&mut self, msg: Message) {
35         println!("[{}][{}]: {}", msg.timestamp, msg.user_id, msg.content);
36         *self.message_count.entry(msg.user_id.clone()).or_insert(0) += 1;
37         self.messages.push_back(msg);
38     }
39
40     fn show_summary(&self) {
41         println!("\n=====");
42         println!("| Chat History Summary");
43         println!("=====");
44         for msg in &self.messages {
45             println!("| [{}][{}]: {}", msg.timestamp, msg.user_id, msg.content);
46         }
47     }
48 }
```

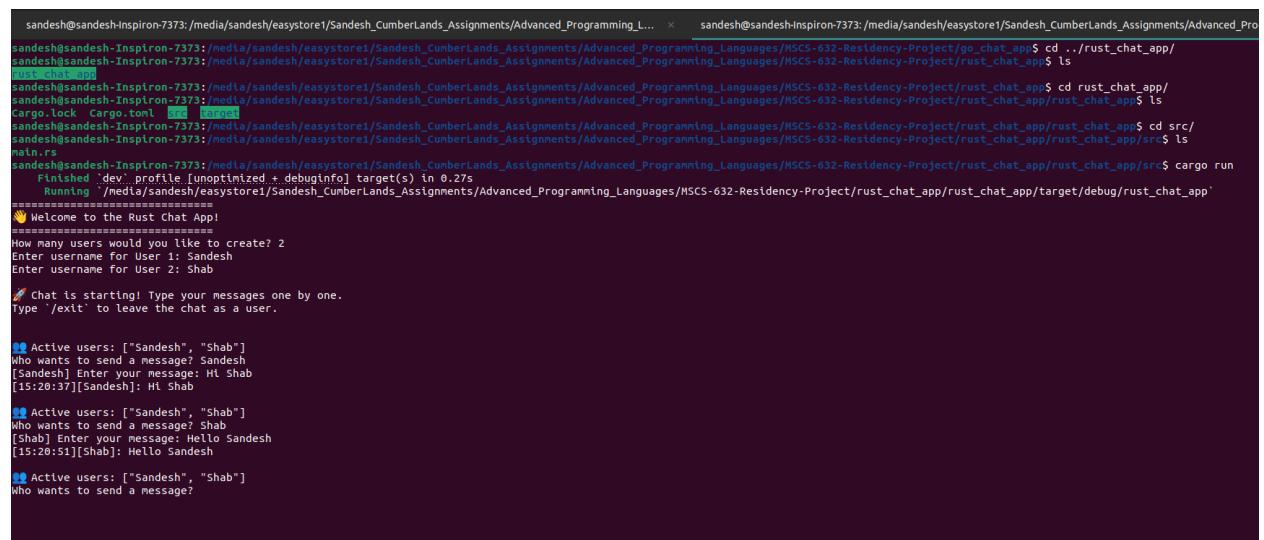
Async Concurrency

Rust's concurrency is handled using tokio and async fn. For example, the interactive user message loop is defined as an async function:

```
async fn start_user(id: &str, server: Arc<Mutex<ChatServer>>) {  
    ...  
    let mut srv = server.lock().unwrap();  
    srv.add_message(msg);  
    sleep(Duration::from_millis(200)).await;  
}
```

This ensures non-blocking behavior while maintaining thread safety through ownership and borrowing rules.

Screenshot : CLI in action with users taking turns to chat.



```
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_L... x sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/go_chat_app$ cd ..../rust_chat_app/  
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app$ ls  
rust_chat_app  
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app$ cd rust_chat_app/  
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app/rust_chat_app$ ls  
Cargo.lock Cargo.toml lib cargo  
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app/rust_chat_app$ cd src/  
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app/rust_chat_app/src$ ls  
main.rs  
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app/rust_chat_app/src$ cargo run  
=====  
 Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.27s  
 Running `/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app/rust_chat_app/target/debug/rust_chat_app'  
=====  
👋 Welcome to the Rust Chat App!  
=====  
How many users would you like to create? 2  
Enter username for User 1: Sandesh  
Enter username for User 2: Shab  
📝 Chat is starting! Type your messages one by one.  
Type '/exit' to leave the chat as a user.  
  
🕒 Active users: ["Sandesh", "Shab"]  
Who wants to send a message? Sandesh  
[Sandesh] Enter your message: Hi Shab  
[15:20:37][Sandesh]: Hi Shab  
🕒 Active users: ["Sandesh", "Shab"]  
Who wants to send a message? Shab  
[Shab] Enter your message: Hello Sandesh  
[15:20:51][Shab]: Hello Sandesh  
🕒 Active users: ["Sandesh", "Shab"]  
Who wants to send a message?
```

Message Handling Logic

Users are dynamically added and prompted for turn-based chat like:

Choose user [User1/User2]: User1

[User1] Enter message: Hello!

After all users exit, a summary of messages and stats is printed.

```
server.filter_by_user("User1");  
server.filter_by_keyword("hello");
```



Screenshot : Chat summary showing filters and user message counts.

```

👥 Active users: ["Sandesh", "Shab"]
Who wants to send a message? /exit
⚠User '/exit' is not active or doesn't exist.

👥 Active users: ["Sandesh", "Shab"]
Who wants to send a message? Sandesh
[Sandesh] Enter your message: /exit
👋 Sandesh has left the chat.

👥 Active users: ["Shab"]
Who wants to send a message? Shab
[Shab] Enter your message: exit
[15:24:16][Shab]: exit

👥 Active users: ["Shab"]
Who wants to send a message? Shab
[Shab] Enter your message: /exit
👋 Shab has left the chat.

🎉 All users have left the chat.

=====
📋 Chat History Summary
=====
[15:20:37][Sandesh]: Hi Shab
[15:20:51][Shab]: Hello Sandesh
[15:23:13][Sandesh]: How are you?
[15:23:31][Shab]: Doing good
[15:24:16][Shab]: exit

=====
📈 Messages Sent Per User
=====
Sandesh: 2 message(s)
Shab: 3 message(s)

💡 Total Messages: 5
👥 Users Participated: 2
✓ Chat Ended Successfully.
sandesh@sandesh-Inspiron-7373:/media/sandesh/easy
```



3. Go Implementation

Our Go version is also a CLI-based application where users can be added dynamically using prompts. Messages are routed between users using **goroutines and channels**, showcasing Go's concurrency strengths.

Goroutines and Channels

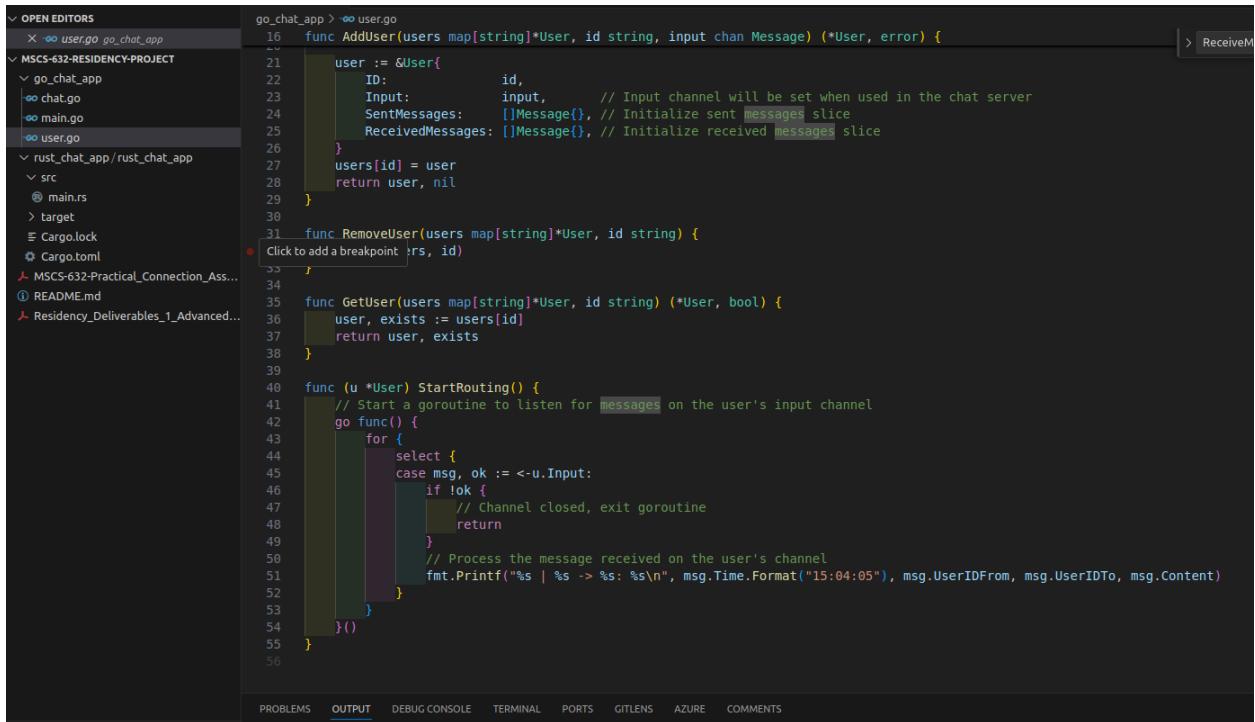
Each user has a dedicated goroutine running `ReceiveMessages()`. All messages are routed via the `ChatServer`:

```

func (u *User) ReceiveMessages() {
    for {
        msg := <-u.Received
        fmt.Printf("[From %s to %s] %s\n", msg.SenderID, msg.ReceiverID, msg.Content)
    }
}

```

 **Screenshot :** Code showing goroutine setup and ReceiveMessages()



```

OPEN EDITORS
  user.go go_chat_app
MSCS-632-RESIDENCY-PROJECT
  go_chat_app
    chat.go
    main.go
    user.go
rust_chat_app/rust_chat_app
  src
    main.rs
  target
Cargo.lock
Cargo.toml
MSCS-632-Practical_Connection_Ass...
README.md
Residency_Deliverables_1_Advanced...

func AddUser(users map[string]*User, id string, input chan Message) (*User, error) {
    user := &User{
        ID:           id,
        Input:        input, // Input channel will be set when used in the chat server
        SentMessages: []Message{}, // Initialize sent messages slice
        ReceivedMessages: []Message{}, // Initialize received messages slice
    }
    users[id] = user
    return user, nil
}

func RemoveUser(users map[string]*User, id string) {
    // Click to add a breakpoint
    if _, ok := users[id]; ok {
        delete(users, id)
    }
}

func GetUser(users map[string]*User, id string) (*User, bool) {
    user, exists := users[id]
    return user, exists
}

func (u *User) StartRouting() {
    // Start a goroutine to listen for messages on the user's input channel
    go func() {
        for {
            select {
            case msg, ok := <-u.Input:
                if !ok {
                    // Channel closed, exit goroutine
                    return
                }
                // Process the message received on the user's channel
                fmt.Printf("%s | %s -> %s: %s\n", msg.Time.Format("15:04:05"), msg.UserIDFrom, msg.UserIDTo, msg.Content)
            }
        }()
    }
}

```

Data Structures

Go uses basic structs and slices, keeping things flexible and idiomatic:

```
type Message struct {  
    SenderID string  
    ReceiverID string  
    Content string  
}
```

ChatServer stores users and their message history:

```
type ChatServer struct {  
    Users map[string]*User  
    Messages []Message  
}
```

📸 **Screenshot :** Screenshot showing the struct definitions from the file chat.go.

```

OPEN EDITORS
go_chat_app > go_chat.go
MSCS-632-RESIDENCY-PR... D R C S
go_chat_app
go_chat.go
main.go
user.go
rust_chat_app/rust_chat_app
src
main.rs
target
Cargo.lock
Cargo.toml
MSCS-632-Practical_Connection_Ass...
README.md
Residency_Deliverables_1_Advanced...

```

```

12 // ChatServer stores and displays messages, and supports filtering by user and keyword.
13
14
15 // Message holds the content sent by a user along with metadata.
16 type Message struct {
17     UserIDFrom string // ID of the user who sent the message
18     UserIDTo   string // ID of the user to whom the message is directed
19     Content    string // Actual message text
20     Time       time.Time // Timestamp of when message was sent
21 }
22
23 // ChatServer handles storing and displaying messages.
24 type ChatServer struct {
25     Users map[string]*User // Map of users to allow user management
26     Messages []Message    // Slice to store chat history
27 }
28
29 // FilterByUser prints all messages from a specific user.
30 func (c *ChatServer) FilterByUser(userID string) {
31     // Check if the user exists in the chat server
32     if _, exists := c.Users(userID); !exists {
33         fmt.Printf("User with ID %s does not exist.\n", userID)
34         return // Exit if the user does not exist
35     }
36     var sentMessages = c.Users(userID).SentMessages
37     for _, msg := range sentMessages {
38         if msg.UserIDFrom == userID || msg.UserIDTo == userID {
39             fmt.Printf("%s | %s -> %s: %s\n", msg.Time.Format("15:04:05"), msg.UserIDFrom, msg.UserIDTo, msg.Content)
40         }
41     }
42
43     var receivedMessages = c.Users(userID).ReceivedMessages
44     for _, msg := range receivedMessages {
45         if msg.UserIDFrom == userID || msg.UserIDTo == userID {
46             fmt.Printf("%s | %s -> %s: %s\n", msg.Time.Format("15:04:05"), msg.UserIDFrom, msg.UserIDTo, msg.Content)
47         }
48     }
49 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS AZURE COMMENTS

Menu-Driven CLI

The Go version uses a command-based prompt:

1. Add User
2. Send Message
3. Filter Messages
4. Exit

Example send flow:

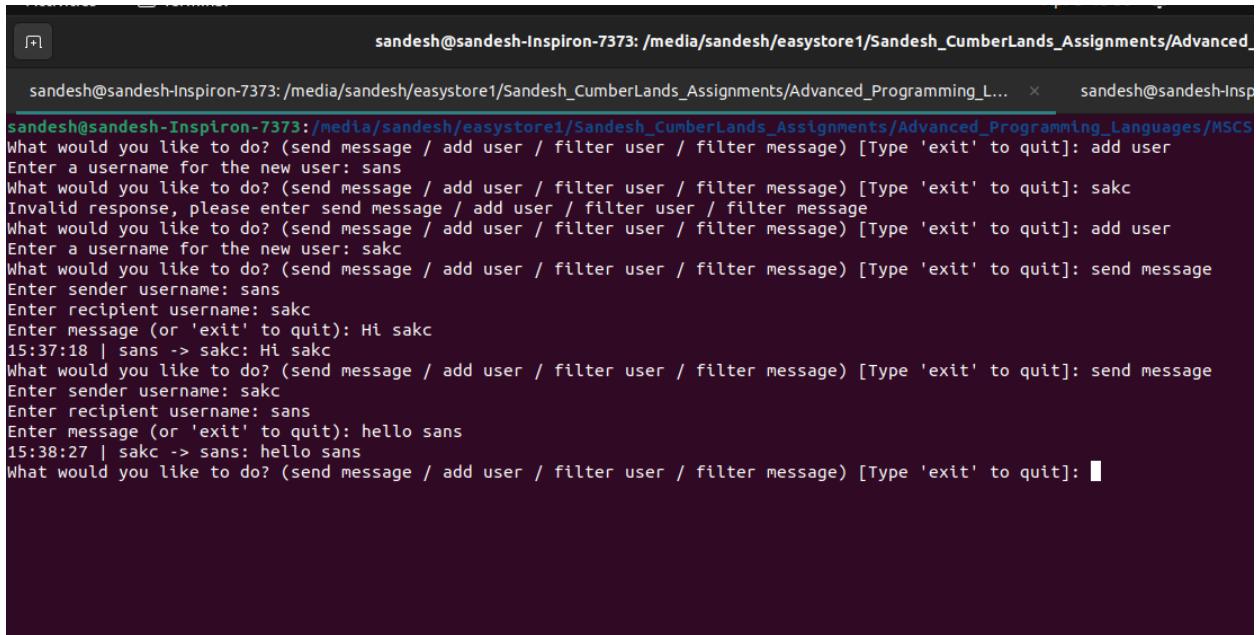
```
fmt.Println("Sender Username: ")
```

```
senderID := getInput()
```

...

server.SendMessage(senderID, receiverID, content)

📸 **Screenshot :** Terminal showing Go CLI in action with commands



A terminal window titled "sandesh@sandesh-Inspiron-7373: /media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS". The terminal shows a Go CLI session where a user adds a new user ("sans") and sends a message ("Hello sans") to another user ("sakc"). The session ends with a command to quit.

```
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: sans
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: sakc
Invalid response, please enter send message / add user / filter user / filter message
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: sakc
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: send message
Enter sender username: sans
Enter recipient username: sakc
Enter message (or 'exit' to quit): Hi sakc
15:37:18 | sans -> sakc: Hi sakc
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: send message
Enter sender username: sakc
Enter recipient username: sans
Enter message (or 'exit' to quit): hello sans
15:38:27 | sakc -> sans: hello sans
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: █
```

4. Comparison Summary

Feature	Rust	Go
User Creation	At the start, predefined	Dynamically via prompt
Message Flow	Turn-based	Menu-based CLI
Concurrency Model	tokio async + Arc<Mutex>	Goroutines + channels

Data Structures	Structs + VecDeque, HashMap	Structs + slices + maps
Message History	Global queue with summary	Per-user and global history
CLI Experience	Turns + summaries + filters	Menu system with live receiver goroutines
Language-Specific Focus	Memory safety, async handling	Goroutines, channels, performance focus

💡 Screenshot : Side-by-side CLI views of Rust and Go versions

```

sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/go_chat_app$ go run *
what would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: sans
what would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: sakc
Invalid response, please enter send message / add user / filter user / filter message
what would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: sakc
what would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: send message
Enter sender username: sans
Enter recipient username: sakc
Enter message (or type 'exit' to quit): Hi sakc
15:37:18 | sans->sakc: Hi sakc
what would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: filter message
Enter the keyword to filter by: hello
... Messages containing 'hello' ...
15:38:27 | sakc->sans: hello sans
what would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: exit
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app$ ./rust_chat_app/
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app$ ls
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app$ cd rust_chat_app/
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app$ ls
Cargo.lock  Cargo.toml  lib.rs  main.rs
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app$ cd src/
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app/rust_chat_app$ ls
main.rs
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app$ cargo run
   Finished 'dev' profile [unoptimized + debugInfo] target(s) in 0.27s
     Running `./rust_chat_app`
Welcome to the Rust Chat App!
Please enter your message one by one.
Type 'exit' to leave the chat as a user.
=====
[+] Active users: ["Sandesh", "Shab"]
Who wants to send a message? Sandesh
[Sandesh] Enter your message: Hi Shab
[15:20:13][Sandesh]: Hi Shab
[+] Active users: ["Sandesh", "Shab"]
Who wants to send a message? Shab
[Shab] Enter your message: Hello Sandesh
[15:20:51][Shab]: Hello Sandesh
[+] Active users: ["Sandesh", "Shab"]
Who wants to send a message? Sandesh
[Sandesh] Enter your message: How are you?
[15:23:13][Sandesh]: How are you?

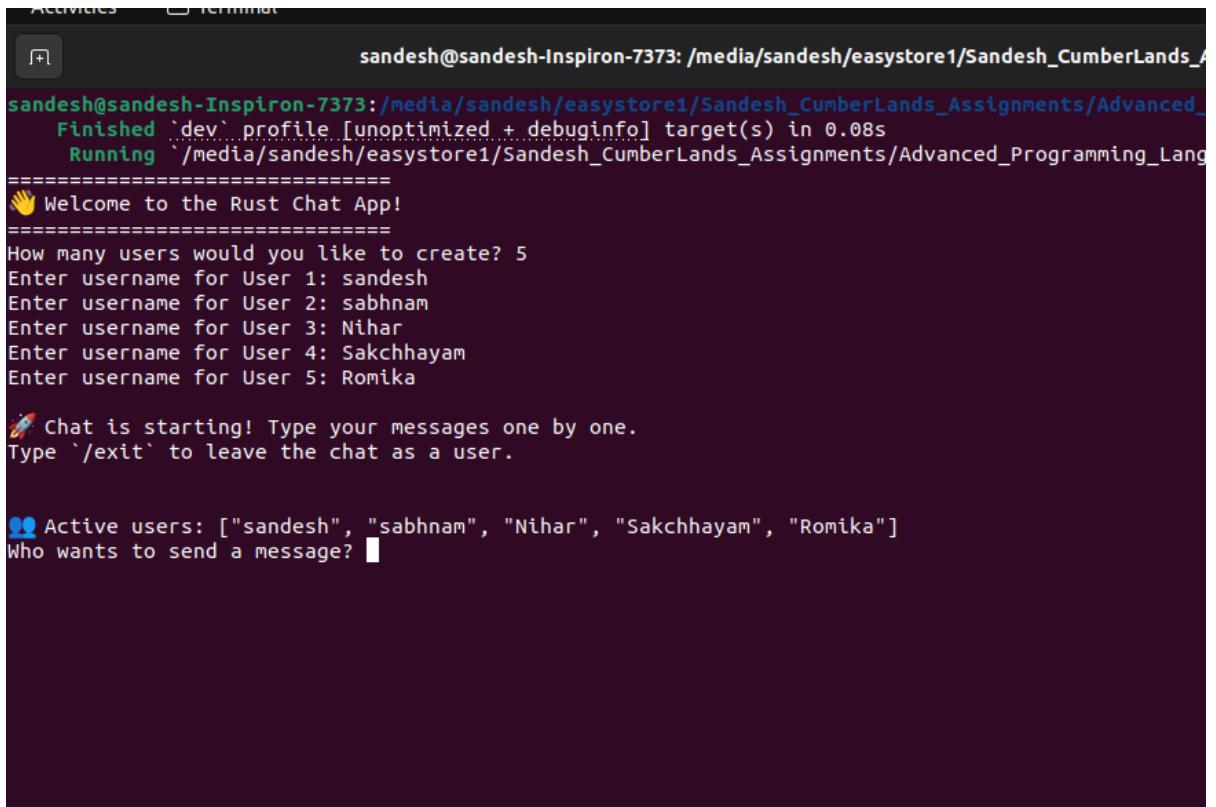
```

5. Testing and Debugging

Both applications were tested with:

- *Multiple users*

Rust Impl:



```

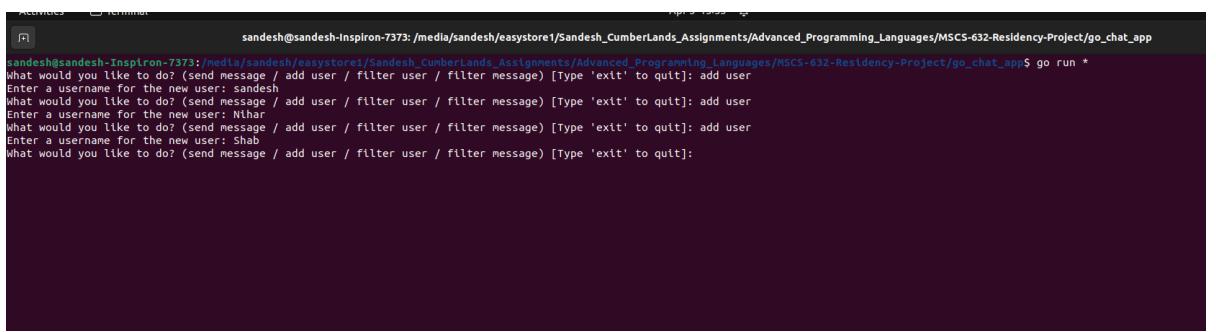
Activities Terminal sandesh@sandesh-Inspiron-7373: /media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/go_chat_app$ cargo build --release
   Compiling go_chat_app v0.1.0 (/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/go_chat_app)
Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.08s
Running `/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/go_chat_app`
=====
👋 Welcome to the Rust Chat App!
=====
How many users would you like to create? 5
Enter username for User 1: sandesh
Enter username for User 2: sabhnam
Enter username for User 3: Nihar
Enter username for User 4: Sakchhayam
Enter username for User 5: Romika

🚀 Chat is starting! Type your messages one by one.
Type '/exit' to leave the chat as a user.

👤 Active users: ["sandesh", "sabhnam", "Nihar", "Sakchhayam", "Romika"]
Who wants to send a message? █

```

Go impl:



```

Activities Terminal sandesh@sandesh-Inspiron-7373: /media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/go_chat_app$ go run *
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: sandesh
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: Nihar
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: Shab
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]:

```

- *Edge cases (empty messages, invalid usernames)*

Rust Impl:

```

Activities Terminal sandesh@sandesh-Inspiron-7373: /media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app/rust_chat_app/src$ cargo run
   Finished dev profile [unoptimized + debuginfo] target(s) in 0.08s
     Running `./rust_chat_app`
=====
👋 Welcome to the Rust Chat App!
=====
How many users would you like to create? 5
Enter username for User 1: sandesh
Enter username for User 2: sabhnam
Enter username for User 3: Nihar
Enter username for User 4: Sakchhayam
Enter username for User 5: Romika
=====
📝 Chat is starting! Type your messages one by one.
Type '/exit' to leave the chat as a user.

👤 Active users: ["sandesh", "sabhnam", "Nihar", "Sakchhayam", "Romika"]
Who wants to send a message? someuser
⚠️ User 'someuser' is not active or doesn't exist.

👤 Active users: ["sandesh", "sabhnam", "Nihar", "Sakchhayam", "Romika"]
Who wants to send a message?

```

Go Impl:

```

Activities Terminal sandesh@sandesh-Inspiron-7373: /media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/go_chat_app$ go run *
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: sandesh
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: Nihar
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: Shab
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter sender username: sandesh
Enter recipient username: muks
Enter message (or 'exit' to quit): hello
Error: User muks does not exist
Please create the user first or check the username.
Failed to send message, please try again.
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]:

```

- *User exit conditions*

Rust Impl:

```

👤 Active users: ["sandesh", "sabhnam", "Nihar", "Sakchhayam", "Romika"]
Who wants to send a message? sandesh
[sandesh] Enter your message: /exit
👋 sandesh has left the chat.

👤 Active users: ["sabhnam", "Nihar", "Sakchhayam", "Romika"]
Who wants to send a message? █

```

Go Impl:

```

Activities Terminal sandesh@sandesh-Inspiron-7373: /media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/go_chat_app$ go run *
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: sandesh
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: Nihar
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: Shab
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter sender username: sandesh
Enter recipient username: muks
Enter message (or 'exit' to quit): hello
Error: User muks does not exist
Please create the user first or check the username.
Failed to send message, please try again.
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: exit
Exiting the chat application.
sandesh@sandesh-Inspiron-7373: /media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/go_chat_app$ █

```

- *Chat history filtering*

Rust Impl:

```

👥 Active users: ["Sandesh", "Shab"]
Who wants to send a message? /exit
⚠User '/exit' is not active or doesn't exist.

👥 Active users: ["Sandesh", "Shab"]
Who wants to send a message? Sandesh
[Sandesh] Enter your message: /exit
👋 Sandesh has left the chat.

👥 Active users: ["Shab"]
Who wants to send a message? Shab
[Shab] Enter your message: exit
[15:24:16][Shab]: exit

👥 Active users: ["Shab"]
Who wants to send a message? Shab
[Shab] Enter your message: /exit
👋 Shab has left the chat.

👋 All users have left the chat.

=====
📋 Chat History Summary
=====
[15:20:37][Sandesh]: Hi Shab
[15:20:51][Shab]: Hello Sandesh
[15:23:13][Sandesh]: How are you?
[15:23:31][Shab]: Doing good
[15:24:16][Shab]: exit

=====
📈 Messages Sent Per User
=====
Sandesh: 2 message(s)
Shab: 3 message(s)

💡 Total Messages: 5
👤 Users Participated: 2
✅ Chat Ended Successfully.
sandesh@sandesh-Inspiron-7373:/media/sandesh/easy
```



Go Impl:

```
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/go_chat_app$ go run *
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/go_chat_app$ go run *
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: sans
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: ram
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: send message
Enter sender username: sans
Enter recipient username: ram
Enter message (or 'exit' to quit): hi ram
16:11:50 | sans -> ram: hi ram
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: send message
Enter sender username: ram
Enter recipient username: sans
Enter message (or 'exit' to quit): hi sans
16:12:04 | ram -> sans: hi sans
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: filter message
Enter the keyword to filter by: hi
--- Messages containing 'hi' ---
16:11:50 | sans -> ram: hi ram
16:12:04 | ram -> sans: hi sans
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: filter user
Enter the username to filter messages by: sans
16:11:50 | sans -> ram: hi ram
16:12:04 | ram -> sans: hi sans
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]:
```

Debugging tools like `println!` (Rust) and `fmt.Println` (Go) helped in identifying flow issues and confirming data integrity. Both versions handle errors gracefully with simple condition checks.

💡 Screenshot : Error handling or test run showing clean exit

Go impl:

```
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/go_chat_app$ go run *
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/go_chat_app$ go run *
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: sans
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: add user
Enter a username for the new user: ram
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: send message
Enter sender username: sans
Enter recipient username: ram
Enter message (or 'exit' to quit): hi ram
16:11:50 | sans -> ram: hi ram
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: send message
Enter sender username: ram
Enter recipient username: sans
Enter message (or 'exit' to quit): hi sans
16:12:04 | ram -> sans: hi sans
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: filter message
Enter the keyword to filter by: hi
--- Messages containing 'hi' ---
16:11:50 | sans -> ram: hi ram
16:12:04 | ram -> sans: hi sans
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: filter user
Enter the username to filter messages by: sans
16:11:50 | sans -> ram: hi ram
16:12:04 | ram -> sans: hi sans
What would you like to do? (send message / add user / filter user / filter message) [Type 'exit' to quit]: exit
Exiting the chat application.
sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/go_chat_app$
```

Rust impl:

```
[sabhnam] Enter your message: /exit
👋 sabhnam has left the chat.

👤 Active users: ["Sakchhayam", "Romika"]
Who wants to send a message? Sakchhyam
⚠ User 'Sakchhayam' is not active or doesn't exist.

👤 Active users: ["Sakchhayam", "Romika"]
Who wants to send a message? Sakchhayam
[Sakchhayam] Enter your message: /exit
👋 Sakchhayam has left the chat.

👤 Active users: ["Romika"]
Who wants to send a message? Romika
[Romika] Enter your message: /exit
👋 Romika has left the chat.

🎉 All users have left the chat.

=====
📁 Chat History Summary
=====
[16:13:47][Nihar]: Hello all

=====
✅ Messages Sent Per User
=====
Nihar: 1 message(s)

💡 Total Messages: 1
👤 Users Participated: 1
✅ Chat Ended Successfully.

sandesh@sandesh-Inspiron-7373:/media/sandesh/easystore1/Sandesh_CumberLands_Assignments/Advanced_Programming_Languages/MSCS-632-Residency-Project/rust_chat_app/rust_chat
```

6. GitHub Repository

All code is available at:

 <https://github.com/sanspokhare126677/MSCS-632-Residency-Project>

Folders:

- /rust_chat_app – Turn-based chat with summary reporting
- /go_chat_app – Dynamic CLI chat with live routing

7. Conclusion

Both implementations meet the project requirements and show how different programming paradigms (Rust's safety-first async vs Go's goroutine-based concurrency) can be applied to solve the same problem.

This side-by-side exercise helped us deeply understand language-specific idioms, best practices, and strengths.