

**Student Name: Wisani Baloyi ST10442713**

**Module: Programming / Application Development**

**Project: Chat Application (Part 1 – Part 3)**

## **1. Introduction**

The purpose of this Portfolio of Evidence (PoE) is to document the development process, technical implementation, challenges, and learning outcomes experienced during the creation of a Chat Application across three main development phases: **Part 1, Part 2, and Part 3.**

The project required the application of fundamental and advanced Java programming principles including user authentication, message handling, data validation, string manipulation, array handling, and automated testing. The final objective was the creation of a functional ChatApp that allows users to:

- Register and log in
- Send, store, and discard messages
- View sent messages
- Search and delete messages
- Generate message reports
- Pass automated unit tests

This report also highlights the challenges faced, solutions implemented, and growth experienced throughout the project.

## **2. Part 1 – User Registration and Login System**

### **2.1 Objectives**

The objectives of Part 1 were to:

- Design and implement a user registration system
- Validate a username and password using specific rules
- Create a working login mechanism
- Display appropriate feedback messages to users

### **2.2 Features Implemented**

- **Username validation:**  
A valid username had to contain an underscore (\_) and be no more than 5 characters long.
- **Password complexity validation:**  
Passwords were required to have:
  - At least 8 characters

- One uppercase letter
- One numeric digit
- One special character
- **Cell phone validation:**  
The application checked that the number included an international code and a valid number of digits.
- **Registration & Login logic:**  
After registration, the user could log in using their credentials and receive a personalised welcome message.

### 2.3 Challenges Faced in Part 1

Some of the main struggles experienced during this phase included:

- Understanding **regular expressions** used for password and number validation.
- Handling **null inputs and edge cases**.
- Ensuring that error messages matched the exact wording expected by tests.
- Working with object-oriented principles such as **encapsulation and constructors**.
- Creating a logical flow between registration, validation, and login processes.

These challenges are common for college-level Java students, especially when transitioning from basic concepts to slightly more complex real-world implementations.

## 3. Part 2 – Message Creation, Storage, and Handling

### 3.1 Objectives

The main goals of Part 2 were:

- Create a Message class
- Generate a message ID and hash
- Allow the user to choose whether to send, store, or discard a message
- Keep track of the total number of messages sent

### 3.2 Features Implemented

- A **Message object** that stores:
  - Recipient
  - Message text
  - Message ID
  - Message hash
- Support for three user actions:

- Send
- Store
- Disregard
- Static tracking using:
  - totalMessages
  - sentMessages array or list
- Basic reporting functionality to print out stored messages

### 3.3 Challenges Faced in Part 2

Some key difficulties experienced:

- Generating a correct and consistent **message hash**.
- Managing **static vs non-static variables** correctly.
- Organising data into lists and preventing duplication.
- Understanding how objects are saved in arrays and later retrieved.

For many college programmers, this stage is difficult because it requires understanding the relationship between multiple classes and how they interact with each other.

## 4. Part 3 – Arrays, Reports, Search & Testing

### 4.1 Objectives

Part 3 was the most advanced section of the project. The goals were to:

#### Populate multiple arrays:

- Sent Messages
- Disregarded Messages
- Stored Messages
- Message Hashes
- Message IDs

#### Use these arrays to:

- Display senders and recipients
- Show the longest message
- Search by Message ID
- Search by Recipient
- Delete messages using a Message Hash
- Generate a full report
- Write automated **JUnit tests**

## 4.2 How the Requirements Were Met

The following features were successfully implemented:

- Arrays correctly populated from test data
- Messages sorted into correct categories (Sent / Stored / Disregarded)
- Longest message identified using length comparison
- Message search implemented using loops and condition matching
- Message deletion done using message hash comparison

### Full reporting completed listing:

- Message Hash
- Recipient
- Message Content

### Unit tests created to verify:

- Correct data in arrays
- Correct outputs for each function
- Proper deletion of specific messages

### GUI created using JOptionPane to allow:

- Login & Registration
- Message sending
- Viewing messages
- Report generation

## 4.3 Challenges Faced in Part 3

This stage presented the most challenges, including:

- Difficulties with **duplicate class names** and package structure in NetBeans.
- JUnit dependencies not linking in Maven correctly.
- Classpath and **Main Class errors** in Maven builds.
- Confusion between **GUI logic and business logic**.
- Handling multiple arrays in parallel.

These struggles are extremely common among college Java students, as Part 3 requires a mix of:

- Problem-solving
- Debugging
- Logical thinking

- Project organisation
- Software design skills

Through persistence and debugging, these challenges were eventually resolved.

## **5. Skills Developed Through This Project**

This PoE allowed the following skill development:

- Advanced Java programming
- Object Oriented Design (OOP)
- Data validation using regex
- String manipulation
- Array and List handling
- Unit testing using JUnit
- GUI development (JOptionPane)
- Problem solving and debugging
- Real-world project organisation
- Software development lifecycle experience

## **6. Conclusion**

The creation of the Chat Application across the three development stages has significantly strengthened both programming ability and problem-solving skills. Although many obstacles were encountered, the final application meets all required criteria and functions as intended.

This project demonstrates a clear understanding of:

- Handling and manipulating strings
- Working with arrays
- Creating and managing classes and objects
- Using automated unit tests
- Building functional user interfaces
- Debugging and correcting errors in a large-scale application

The knowledge and experience gained from this task will be extremely valuable for further programming and software development projects in the future.

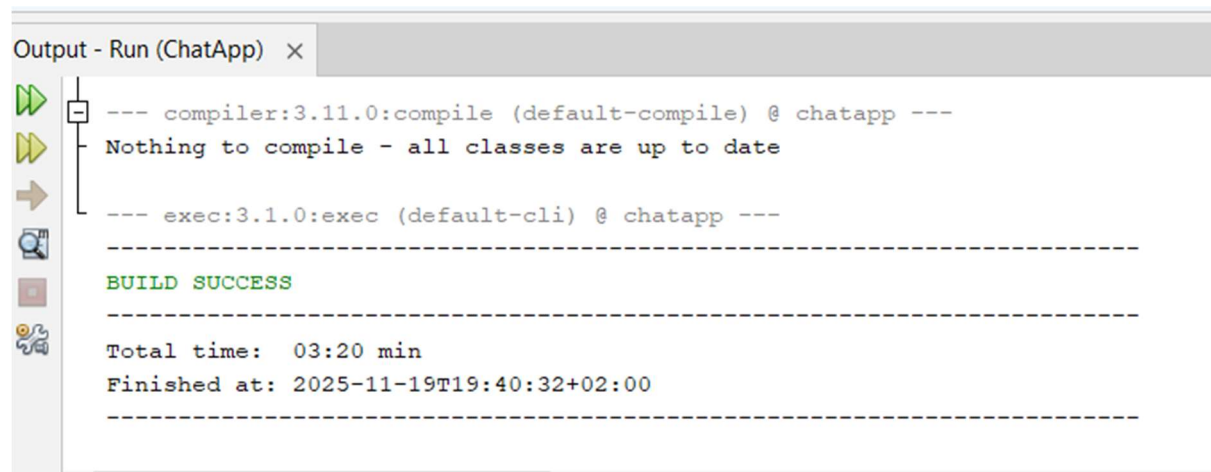
## **7. Final Reflection**

This project was not only a coding assignment but a complete learning journey. From simple login validation in Part 1, to complex data management in Part 3, it required persistence, logic, and determination.

While challenging, the project ultimately strengthened confidence in Java programming and gave a real sense of achievement upon completion.

## 8. Screenshots of the Function

### THE CHATAPP WAS SUCCESSFULLY BUILT AND ALL CLASSES ARE UP TO DATE

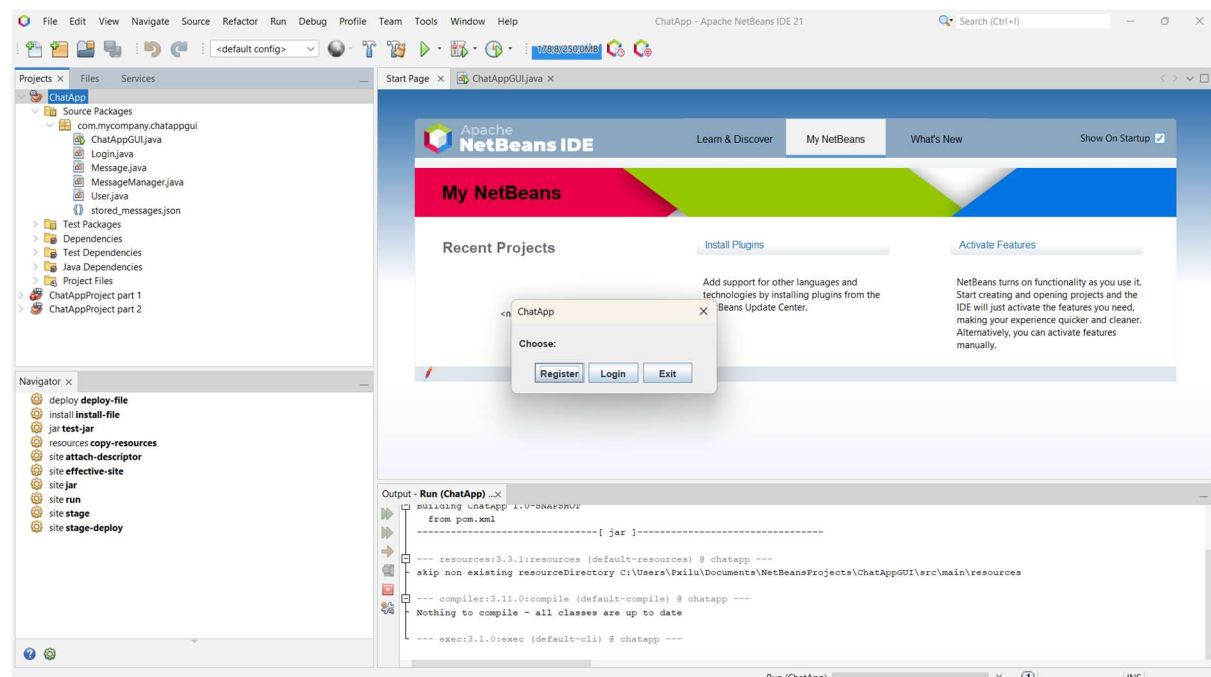


```
Output - Run (ChatApp) x
--- compiler:3.11.0:compile (default-compile) @ chatapp ---
Nothing to compile - all classes are up to date
--- exec:3.1.0:exec (default-cli) @ chatapp ---

-----
BUILD SUCCESS
-----

Total time: 03:20 min
Finished at: 2025-11-19T19:40:32+02:00
-----
```

### OPTIONS TO REGISTER, LOGIN OR EXIT THE CHATAPP



ATTEMPTING TO REGISTER TO THE CHATAPP SO I CAN PROCEED TO THE LOGIN PAGE.



Input

technology Beans

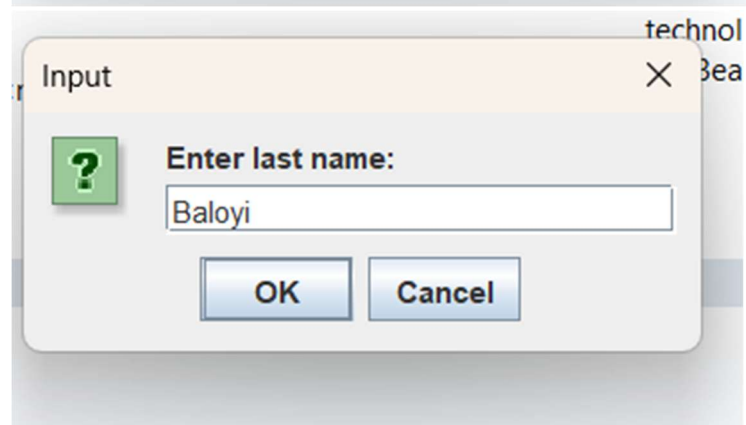
?

Enter first name:

Wisan

OK Cancel

This is a Java Swing input dialog box. It has a title bar with the text 'Input' and a close button. The dialog contains a green question mark icon, a label 'Enter first name:', a text field containing the text 'Wisan', and two buttons labeled 'OK' and 'Cancel'.



Input

technol Bea

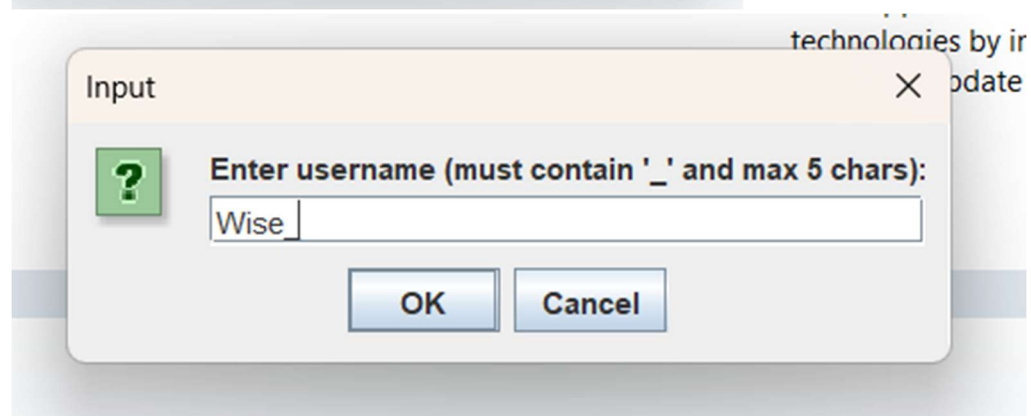
?

Enter last name:

Baloyi

OK Cancel

This is a Java Swing input dialog box. It has a title bar with the text 'Input' and a close button. The dialog contains a green question mark icon, a label 'Enter last name:', a text field containing the text 'Baloyi', and two buttons labeled 'OK' and 'Cancel'.



Input

technologies by ir update

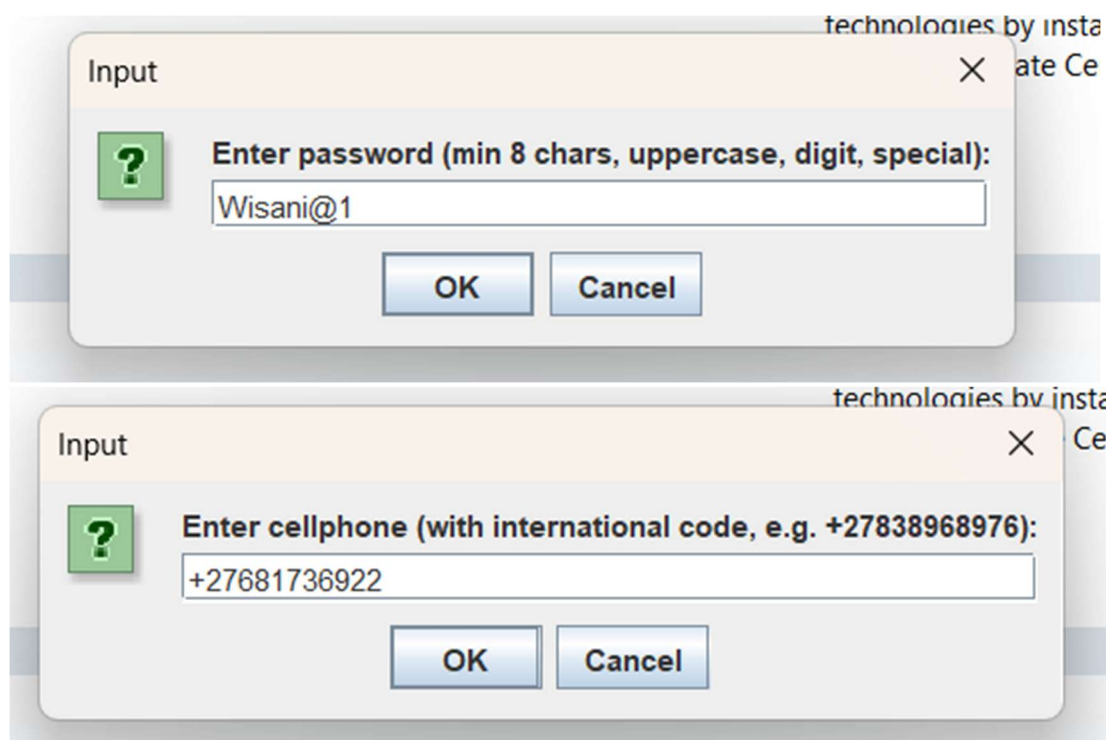
?

Enter username (must contain '\_' and max 5 chars):

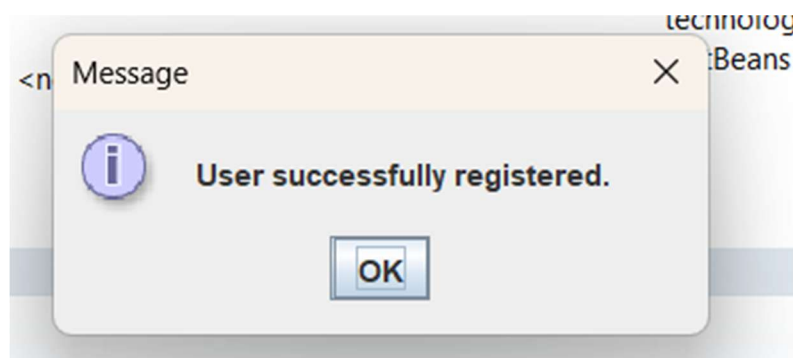
Wise

OK Cancel

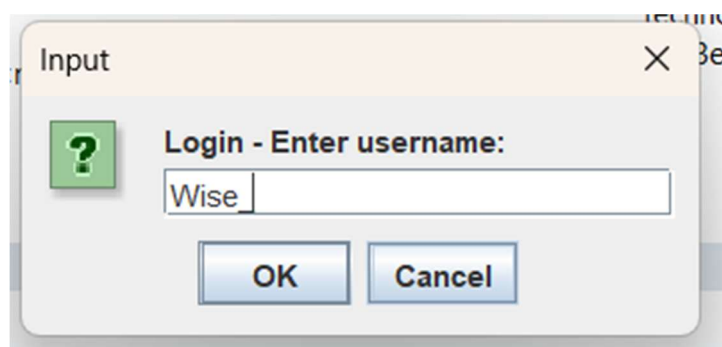
This is a Java Swing input dialog box. It has a title bar with the text 'Input' and a close button. The dialog contains a green question mark icon, a label 'Enter username (must contain '\_' and max 5 chars):', a text field containing the text 'Wise', and two buttons labeled 'OK' and 'Cancel'.




**SUCCESSFULLY REGISTERED.**



**ATTEMPTING TO LOGIN AFTER REGISTRATION.**




Input

 **Login - Enter password:**


**SUCCESSFULLY LOGGED IN**

Message

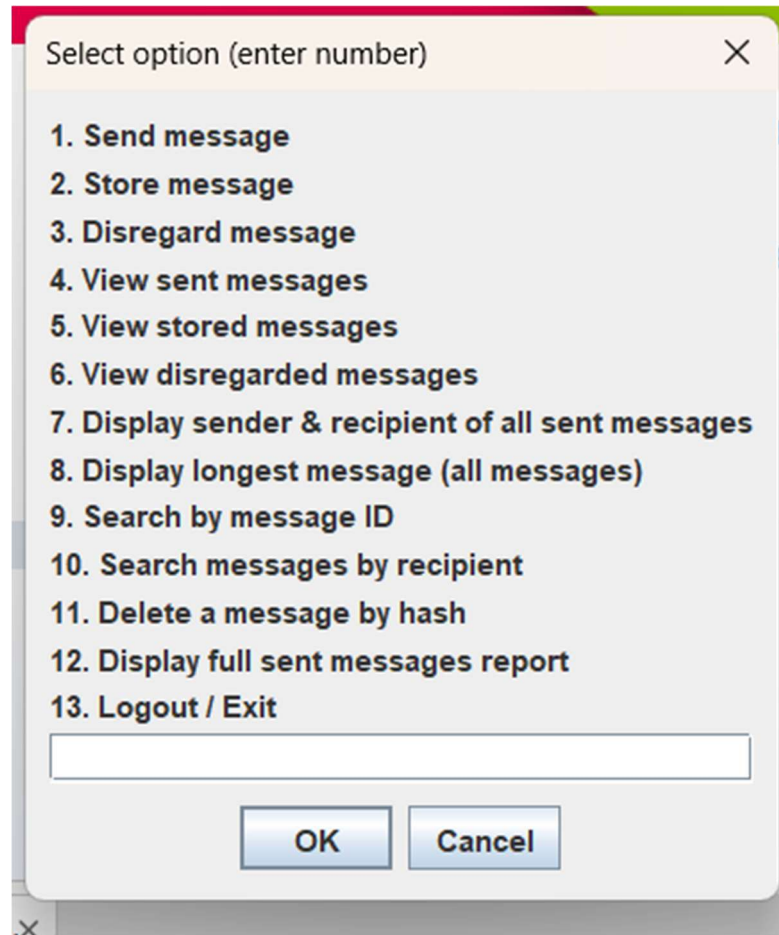
 **Welcome Wisani ,Baloyi it is great to see you.**

**LIMIT OF HOW MANY MESSAGES SHOULD be allowed to be sent.**

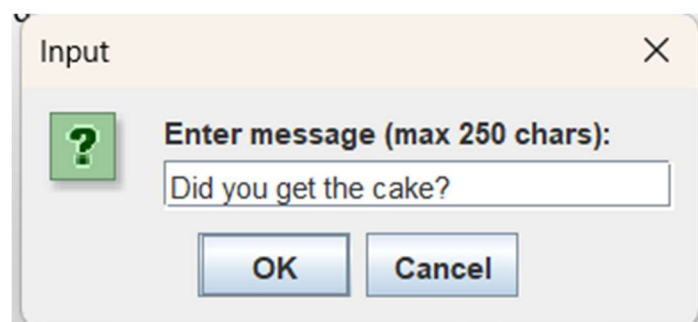
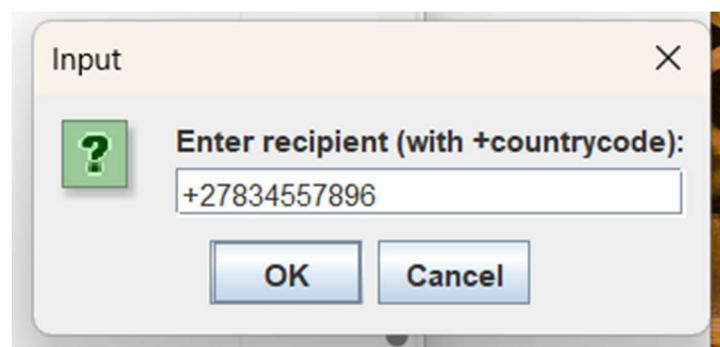
Input

 **How many messages should this user be allowed to send? (enter integer)**

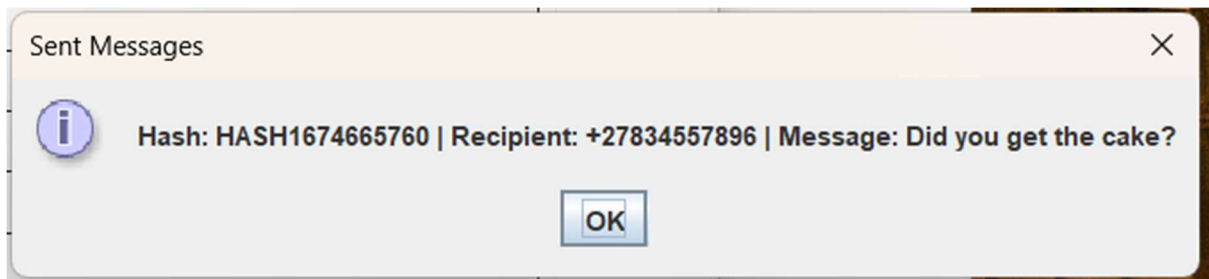
**ALL THE AVAILABLE SELECT OPTIONS.**



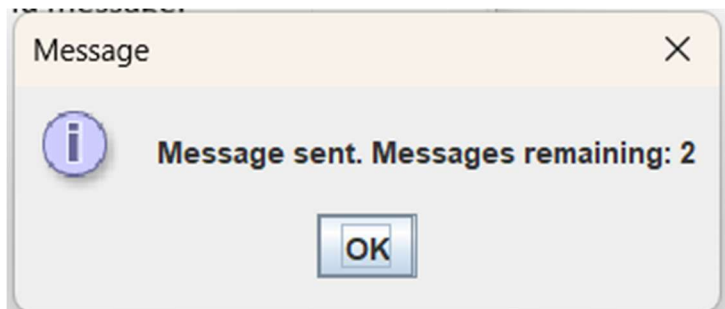
**ATTEMPTING TO SEND MESSAGE.**



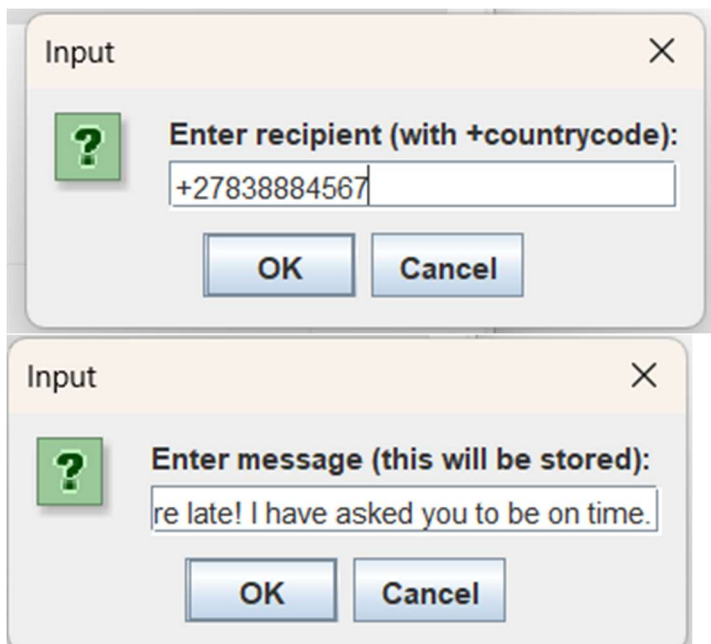
### DISPLAYING THE SENT MESSAGES.



### NUMBER OF MESSAGES REMAINING, AFTER ONE WAS SENT.



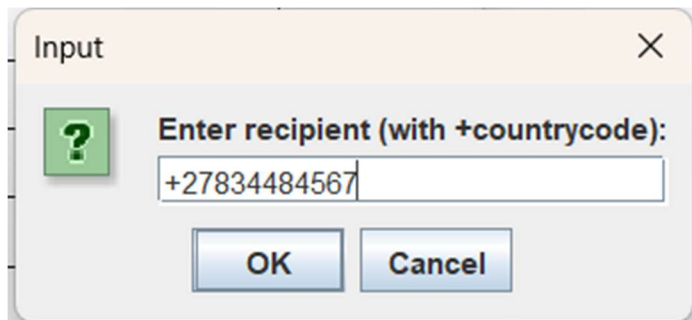
### ATTEMPTING TO STORE MESSAGE.



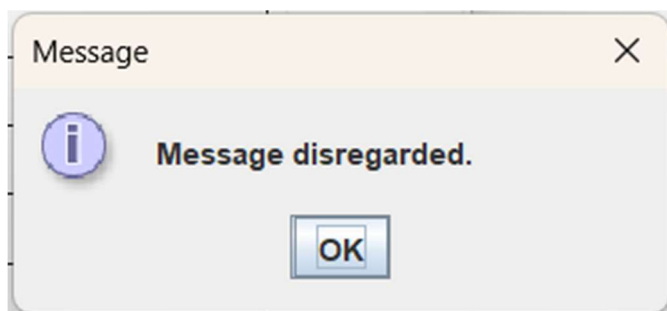
#### MESSAGE SUCCESSFULLY STORED TO JSON



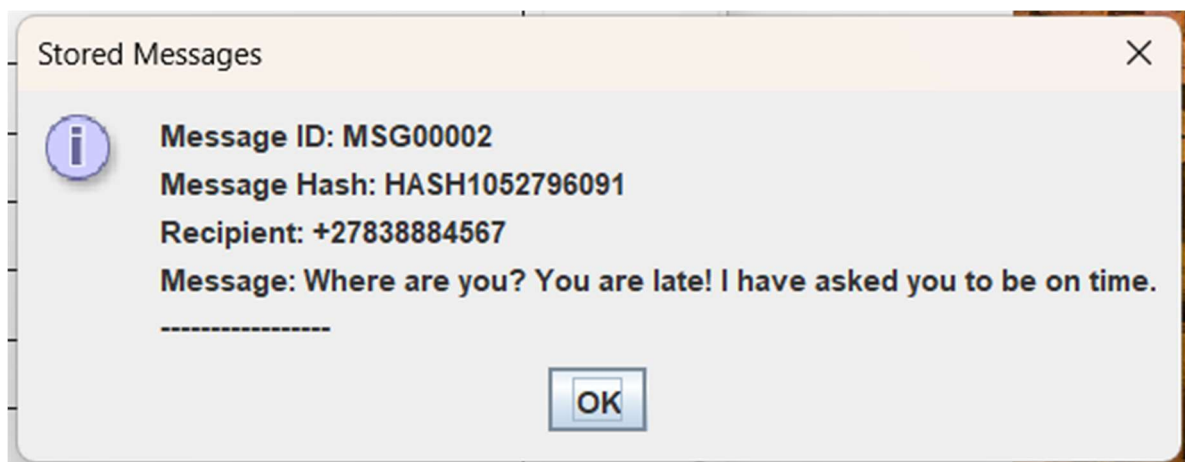
#### ATTEMPTING TO DISREGARD A MESSAGE



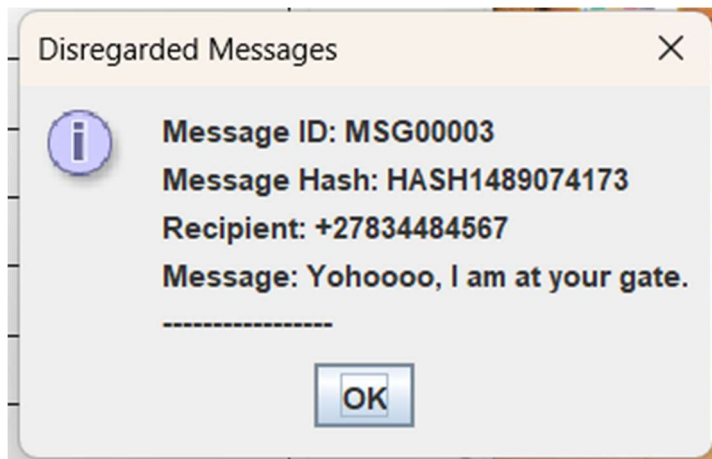
#### MESSAGE SUCCESSFULLY DISREGARDED



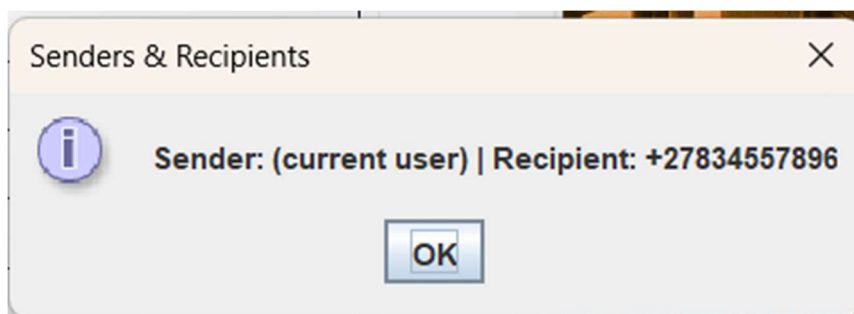
#### DISPLAYING THE STORED MESSAGES.



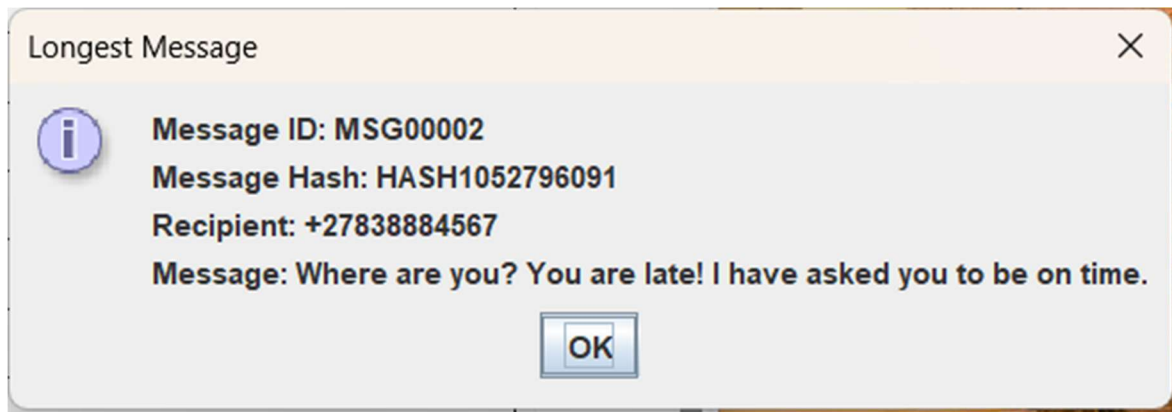
#### DISPLAYING THE DISREGARDED MESSAGES.



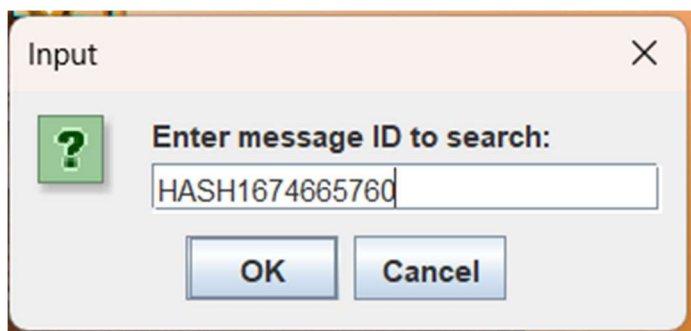
#### SENDERS & RECIPIENTS BOTH DISPLAYED.



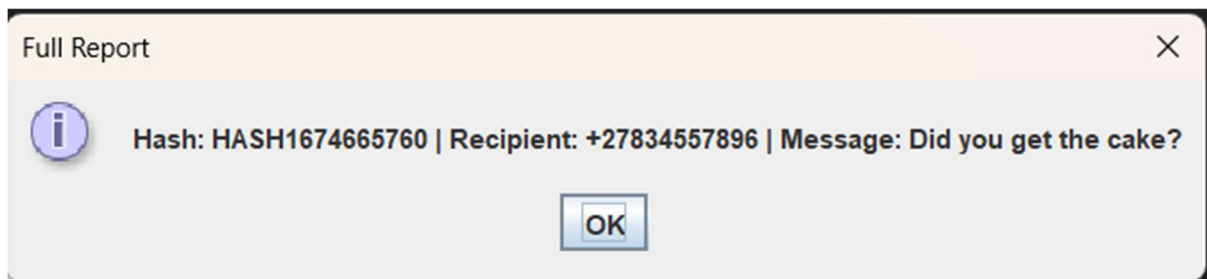
**LONGEST MESSAGE DISPLAYED.**



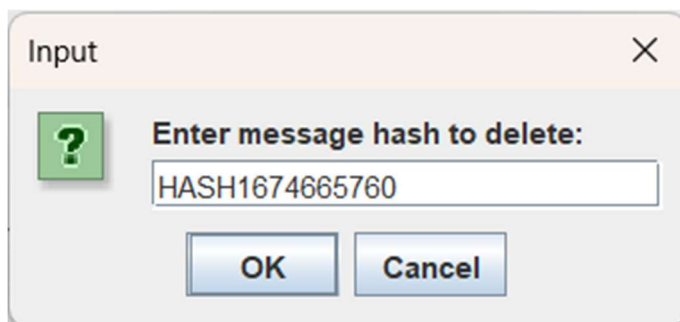
**SEARCH MESSAGE USING ID.**



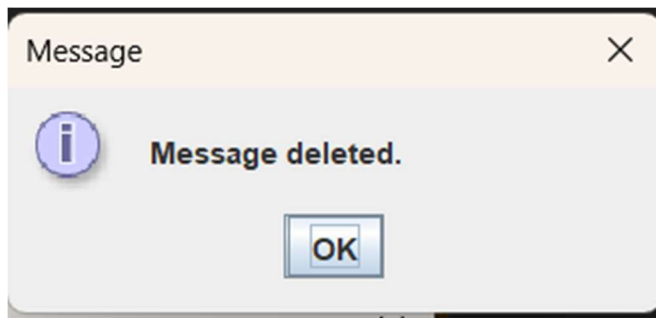
**FULLREPORT OF THE AVAILABLE MESSAGES.**



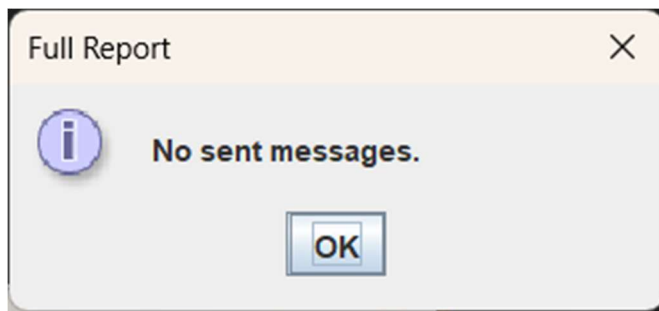
**ATTEMPTED TO DELETE A MESSAGE USING THE HASH.**



**MESSAGE WAS SUCCESSFULLY DELETED.**



**FULL REPORT OF AVAILABLE MESSAGES AFTER DELETING.**



**LOGGED OUT OF THE CHAT AND IT DISPLAYED A "Goodbye!" message.**

