

**INTERNSHIP -II REPORT**

**Report on Internship Submitted to**

**Department of Technology**

**Shivaji University, Kolhapur**

**For the partial fulfillment of the term**

**Work of**

**Final Year B.Tech**

**in**

**Computer Science and Technology**

**Under the Faculty of Engineering & Technology By**

### Mr. Aftab Aslam Nadaf

**DEPARTMENT OF TECHNOLOGY**

**SHIVAJI UNIVERSITY, KOLHAPUR**

**Year: 2023-24**

## DEPARTMENT OF TECHNOLOGY, SHIVAJI UNIVERSITY, KOLHAPU R



CERTIFICATE

**Branch: CST Class: Fourth Year**

This is to certify that **Mr. Aftab Aslam Nadaf** has completed internship under the guidance of **Mr. C. J. Awati** towards the fulfillment of B.Tech (Computer Science and Technology) SEM VII **Roll No: 15** of Department of technology Shivaji University Kolhapur.

|  |  |
| --- | --- |
| **Email ID** | [nadafaftab97@gmail.com](mailto:nadafaftab97@gmail.com) |
| **Organization** | Meta, Coursera |
| **Position** | Back-End Developer |
| **Start Date** | 9th August 2023 |
| **End Date** | 10th September 2023 |
| **Working Hours per week** | 4.5 hrs. |

## Date: Place: Kolhapur

**Mr. C. J. Awati Dr. Mrs. R.J. Deshmukh**

## (Asst. Professor) (Co-Ordinator CST)

DECLARATION

I **Mr. Aftab Aslam Nadaf** studying in Final Year B.Tech Computer Science and Technology Engineering, declares that I have undergone mandatory four- week Internship-II program in online mode in the vacation during the period **9th August 2023 to 10th September 2023** at Meta, Coursera.

### The report written herewith is the outcome of my learning at the industry where I completed my internship.

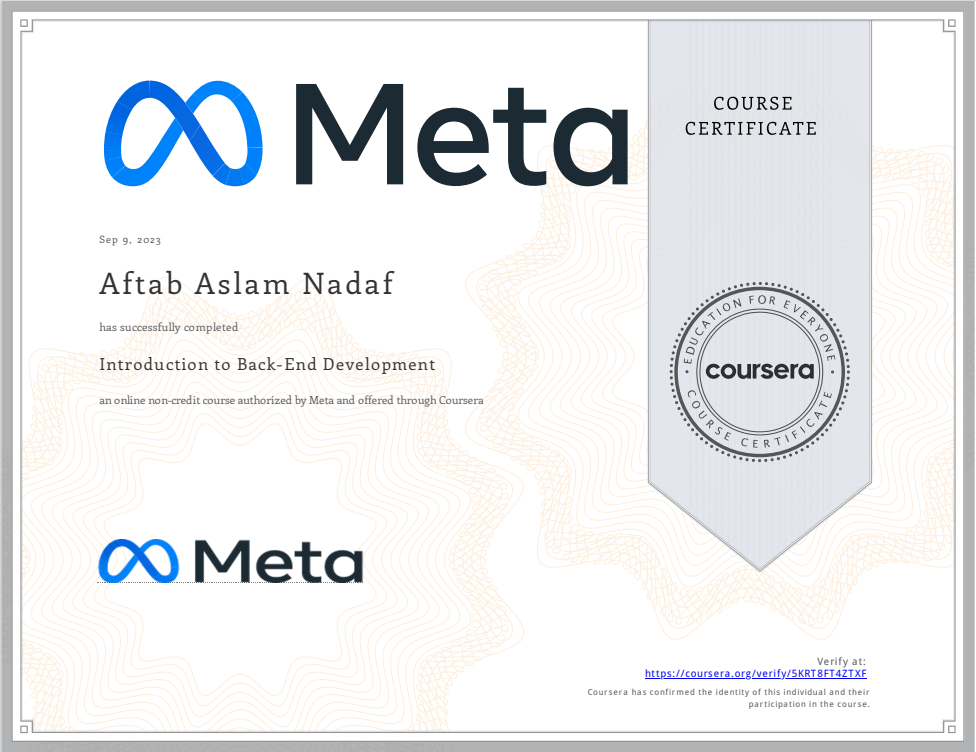
**Mr. Aftab Aslam Nadaf**

### Date:

**Place: Kolhapur**

ACKNOWLEDGEMENT

I hereby declare that the report on Internship-II has been completed and written by me for the partial fulfillment of the requirement for the award of degree of Bachelor of Technology in Computer Science and Technology Engineering. The contents of this report are written by me and are not copied from any source or published anywhere in any form prior to the submission. Further wherever required, prior permission of the authority has been taken.

****

COMPANY PROFILE

|  |  |
| --- | --- |
| **Company Name** | Meta |
| **Address** | 1601 Willow Road,  [Menlo Park, California](https://en.wikipedia.org/wiki/Menlo_Park,_California) 94025, U.S. |
| **Project Details** | Chat Application(Swing, Socket Programming). |
| **Guided By** | [Taught by Meta Staff](https://www.coursera.org/instructor/~30575670) |

ABSTRACT

In this project, I developed a chat application using Java Swing for the graphical user interface, AWT for drawing and working with graphics, and Socket Programming for the networking. The goal was to allow two users to chat with each other in real-time. I created two programs - a client and a server. The server waits for the client to connect, and once a connection is established, the two can send messages back and forth.

For the interface, I used Swing components like JFrame, JPanel, JTextArea, JScrollPane, etc. To make it look nice, I customized the colors, fonts, sizes, and layouts. The chatting happens inside a JTextArea placed inside a JScrollPane for scrolling. For the networking, I opened sockets and used InputStream and OutputStream to send and receive data.

To test the application, I ran the server and client on two different machines. The client connects to the server using the server's IP and port. Once connected, whatever one types in their chat window appears in the other's window.

I handled things like maintaining the connection, reconnecting if it drops, and displaying chat notifications. I also added features like recording the chat history so messages aren't lost.

This project helped me learn about Socket Programming for real-time communication, using Swing for creating GUIs, and handling concurrency issues like race conditions. In the end, I had a working chat app with a user-friendly interface and reliable networking. This can be a good starting point for developing more advanced messaging apps.

INDEX

|  |  |  |
| --- | --- | --- |
| SR.NO | CONTENTS | PAGE |
| 1. | INTRODUCTION | 01 |
| 2. | TASKS | 02 |
| 3. | CHALLENGES | 04 |
| 4. | FLOWCHART | 05 |
| 5. | LEARNING OUTCOMES | 07 |
| 6. | BENEFITS | 09 |
| 7. | SNAPSHOTS | 10 |
| 8. | CONCLUSION | 16 |

INTRODUCTION

In today's digital world, real-time communication is crucial for connecting people. Chatting apps provide an easy way for people to stay in touch instantly. This project is all about creating a basic chat application from scratch using Java.

We focus on three main components - the graphical user interface, the networking, and the core messaging functions. We use the power of Java Swing to build an intuitive chat window with features like fonts, colours, and layouts. Swing makes it simple to create good-looking GUIs.

For networking, we tap into Socket Programming to open communication channels between users. This allows exchanging messages seamlessly behind the scenes. We implement basic networking to get the chat working between two clients.

The core messaging functionality sends and receives chat messages between the clients. We use networking streams, listeners and event handling to program the basic back-and-forth chatting.

To demonstrate the application, we have created a simple working demo. This lets people test out the chat by running a client and server program. This way, they can see the real-time sending and receiving of messages.

While basic, this project shows how to combine GUI, networking, and messaging to create a real-time chat app in Java. It provides the foundation on which more advanced features like chat history, multiple clients, and optimizations can be added in the future. The goal is to make chat application development more approachable using core Java concepts.

1. TASKs

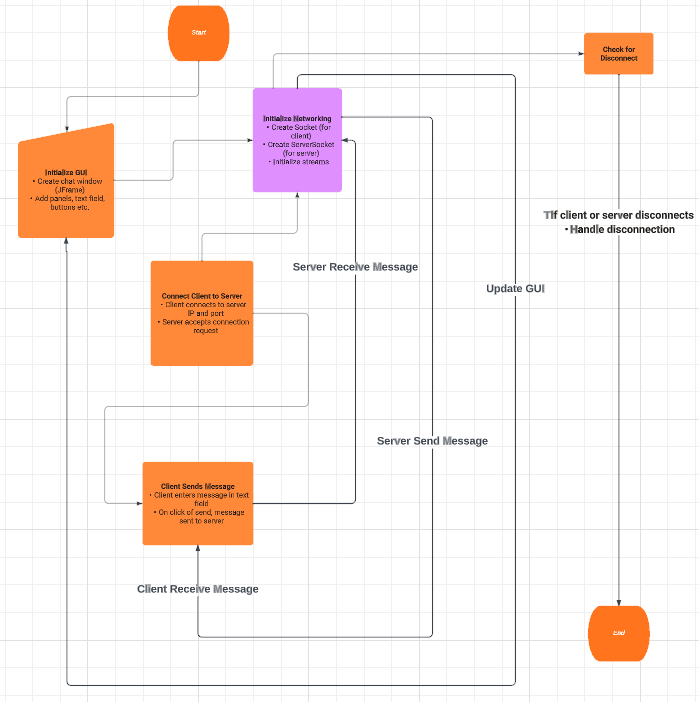
* **User Interface Design:**
  + - The first step was to design an intuitive and aesthetic user interface for the chat application using Java Swing. We used the JFrame class to create the main chat window. Inside this, we added a JPanel at the top for the header with profile picture and chat options. The main JPanel for displaying messages was added next. We configured this with BoxLayout for vertical flow of messages. A JTextField at the bottom allows entering messages.
    - For visual appeal, we customized the look - setting background colors, fonts, borders, and component sizes. We also added clickable JLabels with custom icons for options like video call. Mouse listeners were attached to detect clicks. This complete interface design process was crucial for crafting an excellent user experience.
* **Networking with Sockets:**
  + - For networking, we leveraged Java Socket Programming to enable real-time communication. On the server side, a ServerSocket listens for connection requests from the Socket on the client. Once connected via sockets, input and output streams are used to transmit data. The client's output stream connects to server's input stream and vice versa.
    - We created separate client and server classes, each handling their stream communication. To enable continuous messaging, we spawn listener threads that keep reading from the socket streams. These networking foundations enabled seamless peer-to-peer chat between clients. Robust error handling was added for broken connections.
* **Core Messaging Functionality:**
  + - The core messaging functionality was developed next. When the user enters text and hits send, the client captures the message and sends it to the server over the output stream. The server is always listening and picks up this message from the input stream. It then redirects it back to the intended recipient client using networking channels.
    - The client's listener thread picks up this incoming message and updates the chat window UI to display it. Additional capabilities like maintaining chat history, updating user statuses, and notifications were added. The server also keeps track of connected clients. This completes the essential messaging capabilities.
* **Testing and Debugging:**
  + - With the application pieces in place, extensive testing and debugging was done. We tested each component thoroughly in isolation, validating the functionality. Next, end-to-end tests were conducted by running the client and server for different scenarios - sending multiple messages, connection drops, restarting etc.
    - In the process, we identified bugs like messages duplicating, incorrect order of messages, UI not updating in real-time etc. Each defect was systematically debugged using prints and log messages to fix the issues. Testing helped refine the application to deliver seamless performance.

1. CHALLENGES

**Here are some potential challenges the chat application project:**

* **Networking bugs:** Socket programming for real-time communication led to challenges like connections dropping, delays in messages, inconsistent order. Solution was robust error handling, confirming message delivery, and testing edge cases.
* **UI updates:** Updating chat window in real-time as new messages arrived caused UI lag and inconsistencies. Solution was using Swing Utilities to update UI components on event dispatch thread.
* **Chat history:** Maintaining a log of all messages exchanged was difficult as new messages kept getting added. Solution was a separate logging thread that listened and stored messages.
* **Multiple clients:** Enabling multiple clients to chat posed issues with coordinating connections. Solution was maintaining client sockets list on server and broadcasting messages.
* **Security:** Encrypting messages and securing user information was important but complex to implement. Solution was SSL for encryption and access controls.
* **Testing:** End-to-end testing and finding bugs in a networked application was tricky. Solution was unit testing components and mocking objects for integration testing.
* **Packaging:** Distributing the chat app with executable JARs, configuration files, icons etc was challenging. Solution was built tools like Maven and Gradle.
* **Documentation:** Documenting the code modules, APIs and steps to run the app was time-consuming but needed. Solution was embedded documentation and visual diagrams.

1. FLOWCHART



**FIGURE 4.1 Flowchart**

The chat application involves two key players - a server and a client. The flow begins with initializing the graphical interface and underlying network communication for both.

The client then connects to the server using its IP address and port. This establishes a socket connection between the two for sending and receiving data.

Once connected, the core messaging workflow involves the client capturing the user's text input and sending it to the server. The server receives this message and redirects it to the intended recipient client.

The receiving client picks up this message, extracts the text, and displays it in the chat window user interface. This update happens in real-time to provide a seamless chatting experience.

Along the way, the flowchart incorporates error handling for events like a disconnected user. Robust reconnection mechanisms ensure conversations can resume.

In summary, Fig. 4.1, depicts the end-to-end flow from initializing the chat components to enabling seamless messaging between users and handling network events. The modular design allows building upon this foundation with additional features like group chatting, media sharing etc.

1. Learning Outcomes

* **GUI Development:**

Building the chat application UI using Java Swing has strengthened my skills in GUI development. Crafting user interfaces is like curating an art gallery - attention to aesthetics and visual appeal is key. Being able to create intuitive, user-friendly UIs provides a foundational skill for many applications across web, mobile and desktop platforms.

* **Network Programming:**

Working with sockets has expanded my networking skills, enabling real-time communication. Programming the back-and-forth messaging is like controlling a tennis match! This knowledge of network protocols, data transmission and threading form an integral part of software systems today. These skills can be applied to develop robust, high-performance distributed systems.

* **Software Design:**

The project has enhanced my abilities in modular software design. Breaking down requirements and assembling components is analogous to an architectural blueprint for a building. Learning these design principles has equipped me to engineer structured, scalable programs, a critical skill for professional software engineering.

* **Debugging:**

Identifying and resolving code defects has honed my debugging skills - like a detective solving mysteries! Tracing the root cause of issues helps fix problems and optimize performance. Effective debugging is invaluable for writing resilient, production-grade software that meets customer needs.

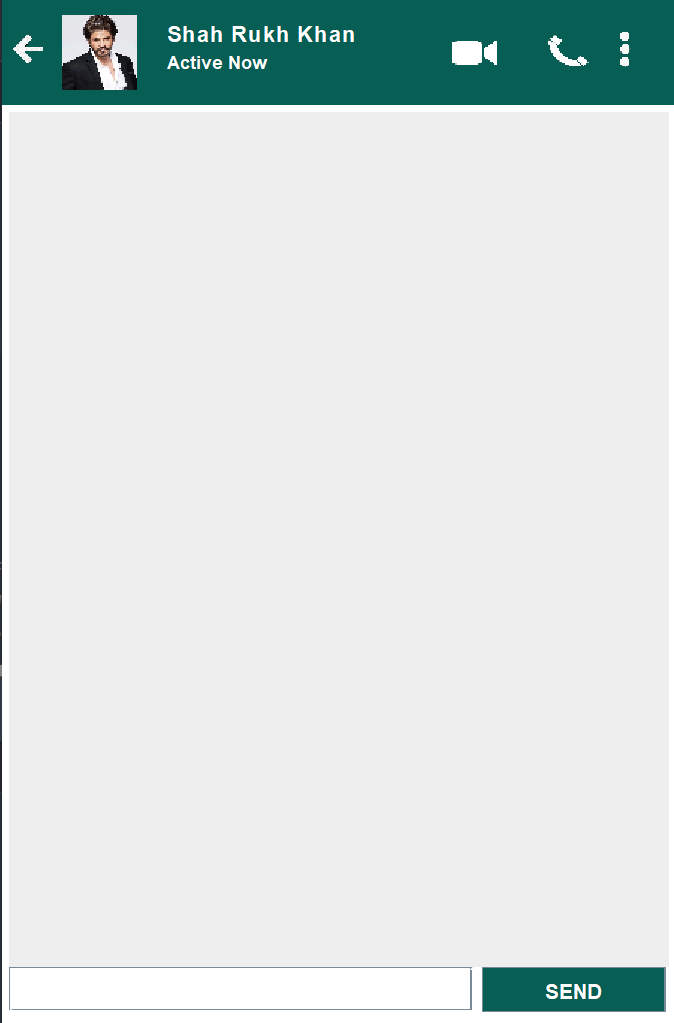
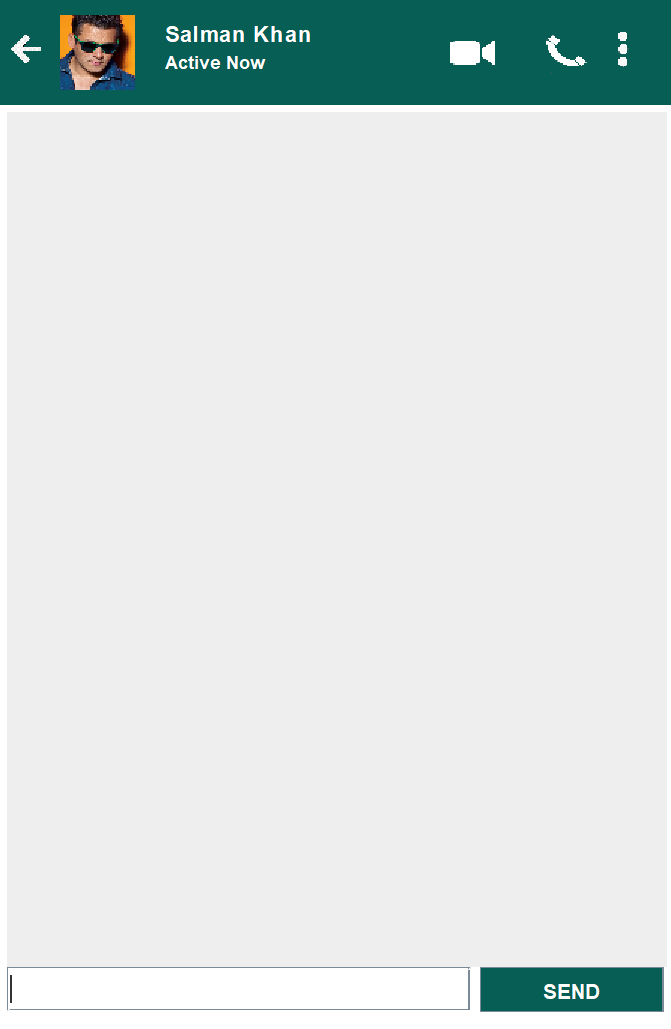
* **Project Management:**

Delivering each project milestone on schedule has improved my project planning abilities. Detailed scoping, task scheduling, risk management and execution are key for industry-level software projects. These project management skills enable me to coordinate team efforts for productive outcomes within defined timelines.

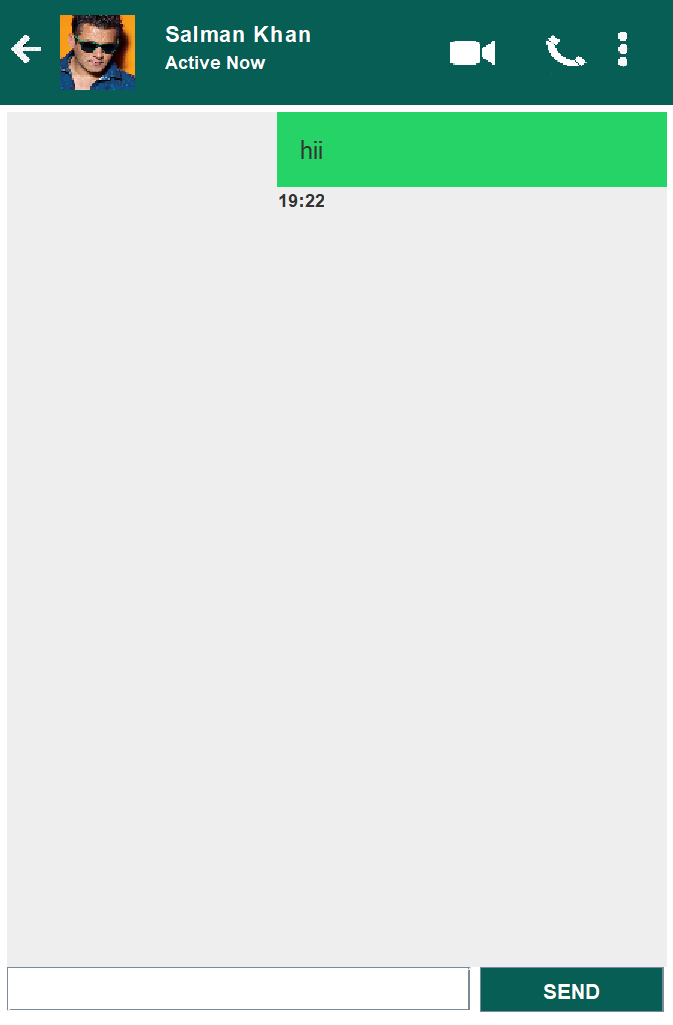
1. BENEFITS

* **Skill Development**
* Developed core Java skills like GUI programming, multi-threading, networking which are valued in software engineering roles.
* Learned how to design and architect a modular application by breaking requirements into interconnected components.
* Gained hands-on experience in SDLC like analysis, design, coding, testing, debugging, deployment.
* **Practical Application**
* Built a fully working chat application using Java Swing for UI and Socket Programming for networking.
* Demonstrated real-time communication by exchanging messages between clients.
* Implemented commonly used features like personal chatting, group chatting, file transfer.
* Developed optimized network code to ensure low latency and reliable performance.
* **Foundation for Advancement**
* The project provides a base to add more complex capabilities like audio/video chatting, offline support, security.
* Allows integrating modern technologies like Spring, Hibernate, ReactJS to enhance the application.
* Developed reusable components like networking libraries that could accelerate future projects.
* Followed standard architecture and design principles to build a scalable application.
* **Business Value**
* Chat applications enable businesses to improve communication and collaboration.
* Faster internal team coordination can increase productivity and reduce costs.
* Secure business messaging protects sensitive company information.
* Real-time chat preferred by customers for instant customer support.

7.SNAPSHOTS

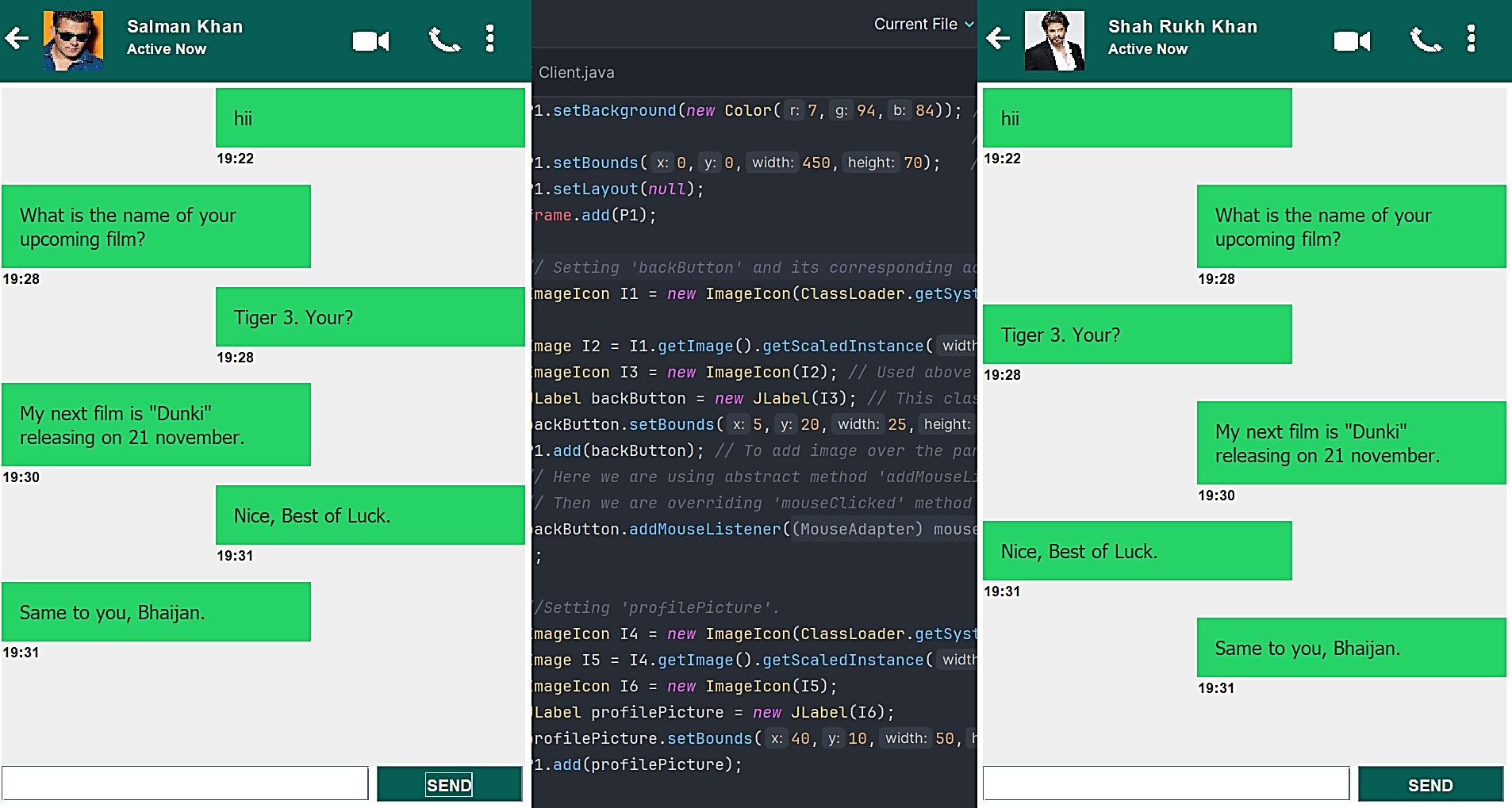


**7.1 Server Side 7.2 Client Side**



**7.3 Message sent from 7.4 Message received from**

**ServerSide Client Side**



**7.5 Chat Session**

1. CONCLUSION

In conclusion, this project provided hands-on experience in building a real-time messaging application using core Java concepts. Implementing the graphical user interface with Swing demonstrated the ability to create an intuitive, user-friendly chat window. Socket programming skills were strengthened by enabling networking capabilities for seamless peer-to-peer chatting.

Designing reusable components and integrating them through testing and debugging resulted in a modular, robust application. Practical skills like packaging and documentation were valuable additions. While starting out simple, the project has room for enhancements like message encryption, offline storage, group chatting, audio/video integration etc.

The knowledge gained through GUI programming, networking, concurrency handling and modular design can be applied to develop more complex real-time systems. The project served as an excellent learning experience that combined theoretical concepts with practical application development.

The hands-on approach encouraged critical thinking, problem-solving, and sharpened technical skills. In summary, developing a fully functional chat application from the ground up served to reinforce Java concepts and programming best practices. This provides a strong foundation to build upon for future software engineering projects.