Articulative Chat Server

A major project report submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Bhanu Aggarwal (201438)

Prachi Chauhan (201255)

Under the guidance & supervision of

Dr. Pradeep Kumar

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

CERTIFICATE

This is to certify that the work which is being presented in the project report titled **"Articulative Chat Server "** in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by " Bhanu Aggarwal (201438) & Prachi Chauhan (201255) " during the period from August 2023 to June 2024 under the supervision of Dr. Pradeep Kumar, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Bhanu Aggarwal (201438)
Prachi Chauhan (201255)

The above statement made is correct to the best of my knowledge.

Dr. Pradeep Kumar
Associate Professor
Computer Science & Engineering Technology
Jaypee University of Information Technology, Waknaghat.

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled **"Articulative Chat Server "** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to June 2023 under the supervision of **Dr. Pradeep Kumar** (Associate Professor ,Department of Computer Science & Engineering Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)
Student Name: Prachi Chauhan and Bhanu Aggarwal
Roll No.: 201255 and 201438

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature with Date)
Supervisor Name: Dr. Pradeep Kumar
Designation: Associate Professor
Department: Computer Science and Engineering
Dated: 15^{tht} May, 2024

ACKNOWLEDGEMENT

I extend my heartfelt appreciation to the key individuals who played a pivotal role in the successful culmination of this project. Their support, guidance, and contributions have been invaluable, shaping the outcome of this endeavor.

Firstly, I express my gratitude to my project supervisor, Dr. Pradeep Kumar. Their mentorship, constructive feedback, and wealth of experience significantly influenced the direction and quality of this project. I am fortunate to have had their guidance throughout this journey.

I am grateful for the unwavering support from friends and family. Their encouragement and understanding during the challenging phases of this project provided the motivation needed to persevere and excel.

I extend thanks to Jaypee University of Information Technology for their crucial assistance, which significantly contributed to the accomplishment of our goals.
This project marks a significant milestone in my academic or professional journey, and I appreciate the collaborative efforts and collective support that made it a rewarding and fulfilling experience.

Bhanu Aggarwal and Prachi Chauhan

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

MVC	Model, View and Controller
JFC	Java Foundation Classes
AWT	Abstract Window toolkit
API	Application Programming Interface
UI	User Interface
GUI	Graphical user interface
SDLC	software development life cycle
UAT	USER ACCEPTANCE TESTING
AR	AUGMENTED REALITY
JDK	Java Development kit
OS	Operating System

LIST OF FIGURES

ABSTRACT

These days, chatting is a great way to share our thoughts and hear what others have to say about any subject. Chats are a reflection of the societal tendencies of late. It is occasionally feasible to talk with notable people and get their advice.

A graphical chat program called Articulative Chat Server makes conversing enjoyable. Thanks to its fantastic features, any user can chat as they choose.

Articulative Chat Server has two types of users:
1. Admin
2. Client

Admin can do the following activities:

1. User Management: He launches the chat server and performs some preliminary setup.
He has the ability to add new users and to connect or unlink them. The administrator has the ability to end the chat session, disconnecting all users.

2. Administrative Client: Using this feature, the admin can also engage in client-side chat from his admin panel.

3. Shutdown: Server can be shutted down.

4. Log of Charts: They have the ability to make log charts that show the details of conversations.

Client can perform the following tasks:

1. Connect: He can connect himself to chat server by typing his username and password.

2. Chat: He can participate in chatting by entering into a chat room.

3. View this User Info: He can also his own profile and options.

4. Display activity of user: Upon completion of the conversation, the other user's console displays whether they are sending text or photos.

5. Room management: He has the ability to make his own topic-based chat groups and ask other people to join them. The communication data can also be saved independently by him.

6. Private Messages: We can also send private messages to other online people with whom we are connected.

7. Canvas: This graphical chatting application is called Articulative Chat Server. The clients can use it to design shapes, add color, and save them. The canvas can also be shown or hidden by the client based on his interests..

8. Copy & Paste: Additionally, he is capable of copying previous text into chat text. Pictures can also be copied and stored in a file.

9. Room control: Admin can manage connection settings and can also manage the settings of chat room for adding or removing people.

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

SYSTEMS PRESENT:

The current chatting system was not designed to be a software program. Everyone exchanges messages with each other by mail or in person. It will be developed as a software application to simplify this difficult communication task, enable users to engage in real-time communication, and reduce wasted time.

Every user or employee of a company must register, access his or her mailbox, and check for mail; this eliminates the user's ability to communicate with others in real time. Users of this facility are not divided into interest-based categories. Effective and user-friendly communication between users is not possible with this kind of communication channel. It will be more difficult to impose limitations on the users if this channel becomes somewhat popular.

SYSTEM PROPOSED:

The identification of need is the initial step in the analytical process. The degree to which a problem is precisely described, fully examined, and the customer's expectations are met by offering an environment that is easy to use are all important factors in determining a system's success.

This system was created to address the challenges that users faced when utilising the mailing system for user-to-user communication. The development of this system was motivated by the need to provide an easy-to-use communication channel, live communication capabilities, user categorization, communication transaction tracking, public and private message sending, instant and offline message sending, and graphical communication.

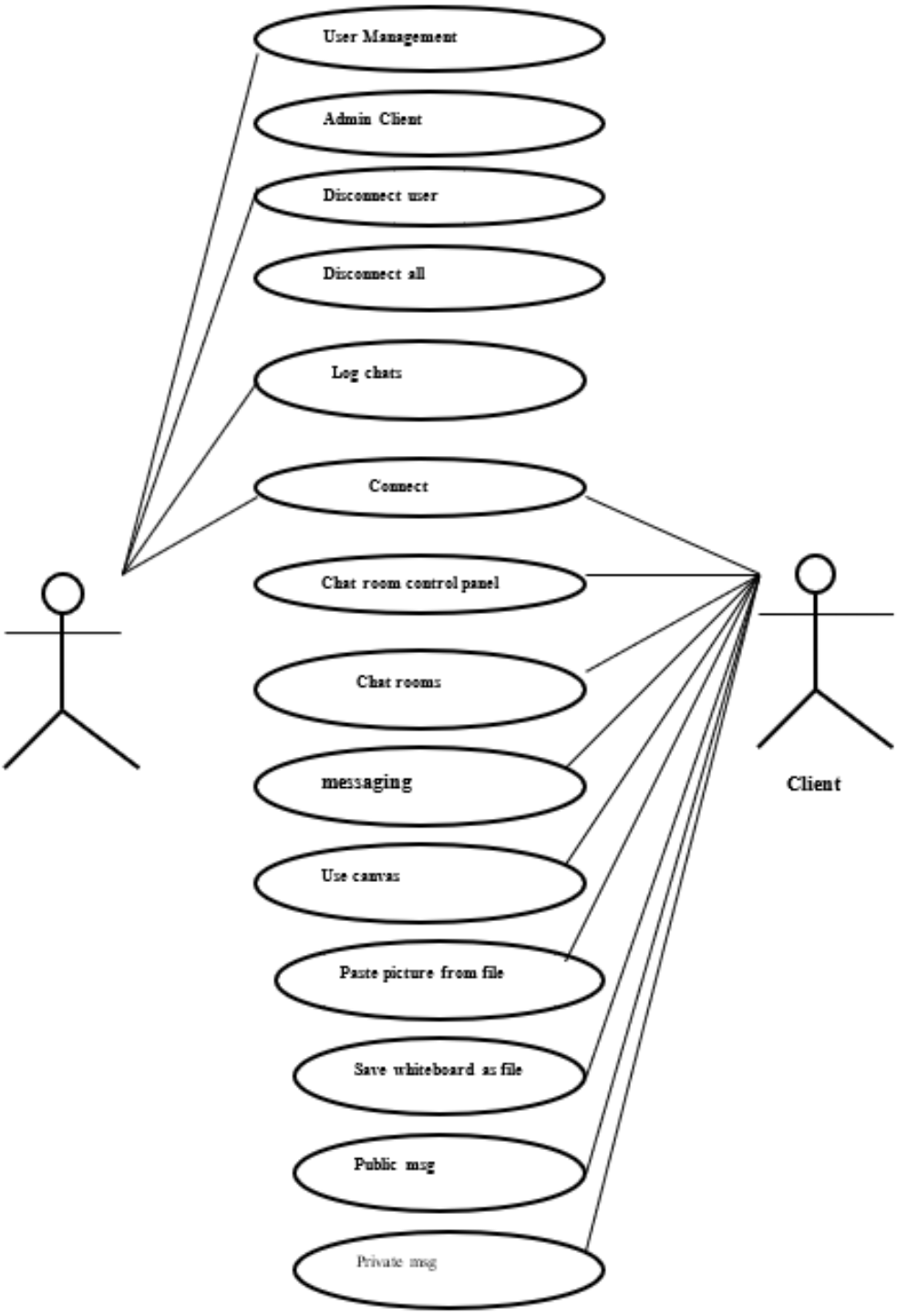

Figure 1.1 Chat Room Control Panel

1.2 PROBLEM STATEMENT

Today everyone has mobiles and have access to internet but still even after having a lot of high technologies we are unable to express ourselves in an efficient way and unable to get depth of in person communication. Today people mostly use text-based messaging services, which can lack the expressiveness and depth of in-person interactions. User interactions are limited by l platforms which are unable to fully capture the meaning of tone, emotion, and visual expression of the speaker. Adopting new communication platforms comes with a learning curve, which can discourage consumers from looking into creative options.

This project addresses these challenges by identifying a crucial gap in the existing communication paradigm and endeavors to fill it with the development of the "Articulative Chat Server." The primary issues to be resolved include:

A. **Expressiveness Limitations:** Current chat systems generally rely on text which causes the lack of dynamic and expressive nature of in-person conversations.
B. **Learning Barriers:** Users may face difficulties in adapting to a novel, multi-modal communication system, hindering widespread adoption.
C. **Visual Expression Deficiency:** Lack of a dedicated space for visual expression restricts users from communicating beyond text and voice.

The "Articulative Chat Server" is a solution that attempts to address these problems by providing a better integration of text, speech, and a drawing area. By doing this, we aim to reshape online communication, making it more engaging, expressive, and representative of the variety of ways people choose to express themselves. This report outlines the steps taken by the project to solve these issues and offers a creative fix to improve online communication.

1.3 OBJECTIVES

This produces quality software. The origins of a user interface are frequently found in user interfaces that are found in our world. Take a second to think about a straightforward button, similar to a key on a keyboard, that is in front of you. These buttons have a clear division between the components that make up the button's façade and mechanism. Keyboard keys are

fundamental building blocks that consist of two halves. It gets its button-like behaviour from a single component. Its appearance is the result of the other element.
This construction proves to be a potent element in the design. Reuse is encouraged as opposed to redesign. Your keyboard's keys were made in such a way that it is simple to manufacture new keys by replacing the key tops and reusing the button mechanism design, which saves a significant amount of time and work compared to developing each key from scratch.

It should come as no surprise that applying this method to software development has comparable advantages. A popular software implementation of this strategy is the Model, View and Controller (MVC) design pattern.

Now you must be asking how it is related to Java[15] Foundation Classes (JFC) Swing user interface components. I'll tell you, though.
The JFC designers employed the MVC design pattern as foundation for everyone. Swing user interface component, even though it is usually utilized to create whole user interfaces. Swing interface component (whether a table, button, or scrollbar) includes a model, a view, and a controller. Moreover, even when component is in use, the model, view, and controller components are subject to change. The end product is an almost unparalleled flexible toolbox for user interfaces. I'll explain how it operates to you.

THE MVC DESIGN PATTERN

In case of unfamiliarity with the MODEL CONTROLLER VIEW design pattern, we suggest reading "Observer and Observable", one of a well written piece that lays the foundation for this month's column and goes into much more detail on this subject.

The MVC design patternimp divides component in independent parts:

- Model
- View
- controller

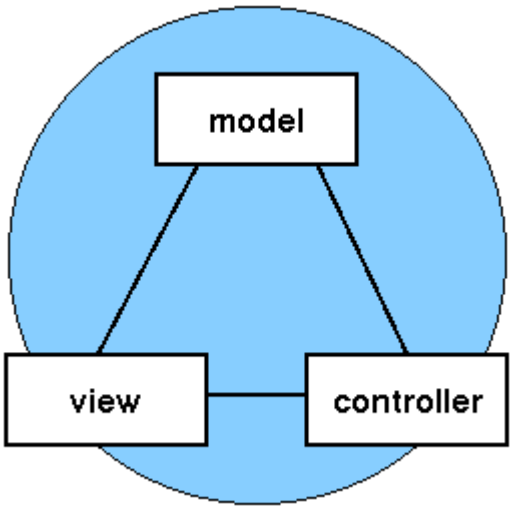

Figure 1.2 MVC design pattern

Model is the component that depicts the state and the low-level behavior of the component. It oversees the state and carries out all changes made to it. The model is not specifically aware of its views or controllers. The system itself keeps track of the connections between the model ,the views and updates it if the when state of model changes.

The component that controls how the state that the model represents is shown visually is called a view. Although a model can have several views, the Swing set does not normally allow for that.

The component that controls how users interact with the model is called the controller. It offers the method by which modifications are applied to the model's state.

In the example of the keyboard key, the view and the controller can correlate to the key's façade. While keys mechanism is corresponded by the model.

The division of a JFC user interface component into MVC is shown in accompanying image. Observe how V and C are merged in a single entity—a popular modification of the fundamental Model controller view architecture. They create user interface for components.

1.4 SIGNIFICANCE AND MOTIVATION OF THE PROJECT WORK

SIGNIFICANCE:

The "Articulative Chat Server" holds significant importance in the realm of digital communication by addressing critical limitations present in traditional messaging platforms. Its significance lies in:

A. **Enhanced Expressiveness:** Offering users, a more comprehensive means of expression through voice, text, and a dedicated drawing space, fostering richer and more nuanced conversations.

B. **Inclusive Communication:** Providing a platform that caters to diverse communication preferences, acknowledging the varied ways individuals express themselves.

C. **Dynamic User Experience:** Introducing innovative features like emotive emojis and intuitive gestures, elevating the overall user experience and making online interactions more engaging.

D. **Breaking Language Barriers:** Enabling users to communicate seamlessly in their preferred mode, irrespective of language, fostering global inclusivity.

E. **Password Protection:** Enhancing user account security by concealing passwords during entry (displaying as '*'), minimizing the risk of unauthorized access.

F. **Collaborative Creativity:** Fostering collaborative creativity by incorporating a real-time drawing canvas, allowing users to co-create visual content, share ideas, and engage in collaborative artistic endeavors.

G. **Interactive Learning Environment:** Serving as an interactive learning environment by facilitating visual explanations and educational discussions, particularly beneficial in educational institutions and remote learning scenarios.

H. **Private Chat Rooms:** The inclusion of private chat rooms addresses the need for more secluded and exclusive communication spaces. Users can engage in focused discussions, enhancing the platform's versatility for different communication scenarios.

I. **Transparent User Profiles:** Clicking on a user's name reveals their additional information, promoting transparency within the community. This transparency enhances user trust and facilitates connections based on shared interests and backgrounds.

J. **Dynamic User Interaction:** The platform promotes dynamic user interaction through features like 'Page Users.' This functionality allows users to initiate communication with others and fosters spontaneous and real-time conversations.

MOTIVATION:

The motivation behind the "Articulative Chat Server" stems from a commitment to redefine the standards of online communication and create a platform that aligns more closely with human interaction. Key motivating factors include:

Closing the Expressiveness Gap: Addressing the limitations of text-based communication by incorporating multi-modal features to bridge the gap between online and face-to-face conversations.

A. **User Empowerment:** Motivating the development of a platform that empowers users to communicate in ways that feel natural and authentic, promoting a sense of ownership over their digital interactions.

B. **Innovation in Digital Communication:** Seeking to innovate and push the boundaries of conventional messaging platforms, providing users with a refreshing and dynamic communication experience.

C. **Embracing Technological Evolution:** The project is motivated by a commitment to embracing technological evolution. By utilizing AWT and Swing, it aims to demonstrate the continued relevance of these technologies while exploring innovative features that push the boundaries of conventional chat platforms.

The "Articulative Chat Server" is not just a project; it represents a visionary step towards transforming digital communication into a more vibrant, inclusive, and enjoyable experience for users across the globe

1.5 ORGANIZATION OF PROJECT REPORT

CHAPTER 1: INTRODUCTION: The inaugural chapter unveils the "Articulative Chat Server," an innovative project poised to redefine the landscape of online communication. With a focus on fostering expressiveness and immersion, this platform introduces a multi-modal interface, complete with a dedicated drawing space and real-time interactions.

CHAPTER 2: LITERATURE REVIEW: This chapter embarks on an exploration of existing communication platforms, dissecting their strengths and limitations. By scrutinizing current messaging and chat platforms, it sets the stage for understanding the unique contributions of the "Articulative Chat Server." The literature review extends beyond, delving into expressive interfaces, drawing spaces, and real-time multi-modal interactions, providing a solid foundation for the distinctiveness of this innovative communication tool.

CHAPTER 3: SYSTEM DESIGN & DEVELOPMENT: Detailed within this chapter is the intricacy of the project's methodology, offering readers a holistic comprehension of the system's design and development. The inner workings of the project, including the rationale behind the multi-modal interface, integration of the drawing space, and other pivotal features, are unveiled. This section ensures readers grasp the meticulous design and development processes that set the "Articulative Chat Server" apart in the realm of communication platforms.

CHAPTER 4: EXPERIMENTS & RESULTS ANALYSIS: Focused on user testing outcomes, this chapter presents authentic feedback, highlighting the platform's strengths and areas for improvement. Rigorous analysis is applied to assess the Articulative Chat Server's potential in everyday communication scenarios.

CHAPTER 5: CONCLUSIONS: The concluding chapter serves as the culmination of the project, summarizing key findings, acknowledging limitations, and proposing avenues for future enhancements. It emphasizes the substantial impact of the Articulative Chat Server in elevating online communication, offering users an expressive, immersive, and engaging platform.

CHAPTER 2: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

The media generally refers to any type of synchronous conference, sometimes even asynchronous conferencing, as a "chat room." Thus, the phrase can refer to any technology, from completely immersive graphical social settings to real-time online chat via instant messaging and online forums.

The main purpose of a chat room is to allow users to text each other information. Generally speaking, chat rooms differ from instant messaging apps, which are primarily made for one-to-one contact, in that they allow users to interact with numerous individuals at once. There are chat rooms for a variety of topics, and the users in a given chat room are typically united by a common interest or another comparable bond. File sharing and camera integration are now possible in some programs thanks to advancements in technology.
Key themes explored in the relevant literature include:

2.1.1 COMMUNICATION MODALITIES: Research on the various modalities of communication, such as text, voice, and visual expression, provides insights into how individuals naturally express themselves. Understanding these modalities informs the integration of diverse communication features in the chat server.

2.1.2 USER EXPERIENCE DESIGN: Literature on user experience design emphasizes the importance of creating interfaces that are intuitive, engaging, and adaptive to user preferences. This informs the design principles applied in developing the user interface for the "Articulative Chat Server."

2.1.3 MULTI-MODAL INTERACTION: Studies exploring the dynamics of multi-modal interaction in digital communication shed light on the challenges and opportunities in seamlessly integrating voice, text, and drawing features. Insights from these studies contribute to the project's approach to multi-modal communication.

2.1.4 PRIVACY AND SECURITY IN MESSAGING PLATFORMS: Literature addressing privacy concerns and security measures in messaging platforms guides the implementation of encryption and user-controlled privacy settings in the "Articulative Chat Server."

2.1.5 INNOVATIONS IN DIGITAL COMMUNICATION: Explorations of innovative features in digital communication platforms, such as emotive emojis and gesture-based interactions, inspire the project's commitment to providing a dynamic and expressive user experience.

2.1.6 TECHNOLOGY ADOPTION AND USER BEHAVIOR: Insights into user behavior and the factors influencing technology adoption inform strategies for onboarding, addressing the learning curve challenge, and encouraging widespread acceptance of the new communication platform.

2.1.7 SCALABILITY AND SYSTEM ARCHITECTURE: Literature on system scalability and architecture considerations guides the development of an infrastructure that can accommodate a growing user base and handle increased conversation loads.

By drawing on this literature, the "Articulative Chat Server" is positioned at the intersection of established knowledge and emerging trends, ensuring that it not only addresses current challenges but also anticipates and adapts to the evolving needs of digital communication.

2.2 KEY GAPS IN THE LITERATURE

While the literature provides valuable insights into various aspects of communication, human-computer interaction, and technology adoption, several key gaps exist that warrant exploration. These gaps present opportunities for the "Articulative Chat Server" project to contribute to the academic and practical understanding of digital communication. The identified gaps include:

2.2.1 HOLISTIC INTEGRATION OF MULTI-MODAL FEATURES: Literature often focuses on individual communication modalities but lacks comprehensive insights into the seamless integration of voice, text, and drawing features within a single communication platform.

2.2.2 USER-CENTRIC LEARNING CURVE SOLUTIONS: While user experience design is extensively discussed, there is a gap in literature addressing user-centric strategies to minimize the learning curve associated with adopting novel communication tools, especially those incorporating multi-modal features.

2.2.3 CROSS-CULTURAL USER PREFERENCES: Limited exploration exists on how diverse user preferences in communication modalities vary across cultures. Understanding cross-cultural dynamics could inform the customization of the "Articulative Chat Server" to better suit global users.

2.2.4 COMPREHENSIVE SECURITY MEASURES IN MULTI-MODAL PLATFORMS: Existing literature tends to focus on security concerns in traditional messaging platforms. There is a gap in understanding the specific security challenges associated with multi-modal[20] communication, including voice and drawing inputs.

2.2.5 LONGITUDINAL STUDIES ON TECHNOLOGY ADOPTION: Many studies offer insights into initial user adoption but lack longitudinal perspectives on how user behavior

evolves over time. Long-term studies are essential to understanding sustained user engagement with innovative communication platforms.

2.2.6 REAL-TIME PROCESSING CHALLENGES IN MULTI-MODAL PLATFORMS: The literature provides limited in-depth exploration of the real-time processing challenges associated with integrating voice, text, and drawing features simultaneously. Understanding these challenges is crucial for optimizing system performance.

2.2.7 INNOVATIVE FEATURES AND USER ENGAGEMENT: While there is literature on innovative features in digital communication, there is a gap in understanding how these features contribute to sustained user engagement over time. Exploring the long-term impact of features like emotive emojis is essential.

By addressing these gaps, the "Articulative Chat Server" project not only contributes to the practical development of a novel communication platform but also offers insights that fill voids in the existing academic literature.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 REQUIREMENTS AND ANALYSIS

REQUIREMENTS:

Every criteria pertaining to the system's performance attributes needs to be spelt out in detail. Performance requirements come in two forms:

- Static
- Dynamic.

Static requirements does not place restrictions in process of execution of system. It include specifications for quantity of files and their sizes that the system must handle, no. of terminals that must be supported, and the number of concurrent users that must be supported. These are also referred to as the system's capability. Dynamic requirements outline limitations on the system's execution behavior. Usually, these consist of the system's throughput limitations and response time.

Performance is measured by processing speed, corresponding resource consumption throughput, and efficiency. To attain a high level of performance
Every criteria pertaining to the system's performance attributes needs to be spelt out in detail. Performance requirements come in two ways: static and dynamic.
Static Require

A few guidelines need to be adhered to, like minimizing the amount of repetitive data, lowering the amount of code, and using fewer buttons. Any real-time system or software that fulfils the necessary functions but deviates from the performance standards are accepted. Following specifications are for testing software's runtime performance within the framework of an integrated system.

3.1.1 FUNCTION REQUIREMENTS:

The outputs that are generated from given inputs are specified by the functional requirements. They explain how the system's input and output are related to one another. A thorough explanation of every data input, together with information about where it comes from and the range of acceptable inputs, must be provided for every functional requirement. It is necessary to specify every action that must be taken on the input data in order to get the desired result.

(i) INPUTS: Numerous modules, including those for user management, chatroom management, messaging, canvas management, preferences, and editing choices, are available in our system.

MODULES:

- **USER MANAGEMENT MODULE:** This module includes the add/delete user feature in the application, which requires user data as input. Additionally, it handles joining and removing users from the program. to do this, pick the user and click the appropriate button. It facilitates the chatroom administrator's control panel management, including inviting, banning, and allowing users. Additionally, it enables us to log user chat transactions and page users.

- **CHAT ROOM MANAGEMENT MODULE:** This module is utilised in a similar manner to the user management module. However, this entity is handled differently in our system than the user entity. Users can build, edit, view, and join the room with its help.
- **CHATTING MODULE:** This module takes the information, such as text messages you wish to send and whether they are public or private, and sends it to the right places.

- **MESSAGING MODULE:** It allows the user to view saved texts, save offline messages for another user, and send instant messages to other users. .

- **CANVAS MANAGEMENT MODULE:** With the help of the white board drawing feature, users can sketch freely, create squares, circles, lines, fonted text, and insert image files into the canvas.

(II) OUTPUTS:

- **USER MANAGEMENT MODULE:** The list of users or data associated with a user, as well as the banned list of users, can be viewed by the administrator. It offers the option to record user conversations in log files.

- **CHATROOM MANAGEMENT MODULE:** The report that is produced by it includes a list of chat rooms.
- **MESSAGES MODULE**: Additionally, a user-by-user list of offline messages is generated.

3.1.2 EXTERNAL INTERFACE REQUIREMENTS

- **USER INTERFACE:** Server can be started by administrator at any port number and configure settings using the JFC Swing based GUI interface provided by this application. Using a UI such to this, regular users can establish a connection with the server and begin utilising the feature.
- **SOFTWARE INTERFACES:** The interface with others should be specified in these interface requirements. Software that the system will use or that the system will use, such as the operating system and other applications' interface. Content and Format of every message is specified.

- **PHYSICAL INTERFACES:** The documentation places great importance on the hardware interface. All the hardware specifications, including memory limitations,

should be stated whether the software is running on pre-configured hardware or hardware that already exists. Furthermore, the hardware's load characteristics and current usage should be disclosed.

3.1.3 PERFORMANCE REQUIREMENTS

Every criteria pertaining to the system's performance attributes needs to be spelt out in detail. Performance requirements come in two parts.

If a system has no restrictions on how system will be executed then its called Static Requirements. These include specifications for the quantity of files and their sizes that the system must handle, the number of terminals that must be supported, and the number of concurrent users that must be supported. These are also referred to as the system's capability. Dynamic requirements outline limitations on the system's execution behavior. Usually, these consist of the system's throughput limitations and response time.

Performance is measured by processing speed, corresponding resource consumption throughput, and efficiency. A few guidelines need to be followed in order to achieve good performance, such as minimising the amount of code, using fewer controls, using less recurring data sets, etc. Any real-time system or software that fulfils the necessary functions but deviates from the performance standards can be accepted. Specifications are used to test software's runtime performance within the framework of an integrated system.

3.1.4 SOFTWARE REQUIREMENTS :

- OS : Windows XP
- Language : JFC Swing & AWT
- Software : JDK1.4

3.1.5 HARDWARE REQUIREMENTS :

- Processor : i3
- Random Access Memory : 1 GB
- Hard Disk : 32 GB
- VDU : VGA

3.1.6 CRITERIA FOR ACCEPTANCE:

It should be shown that functions of system with the specifics of user data and chatroom before system is approved. The developer will have to demonstrate that every criteria is met with test cases.

ANALYSIS:

- User Analysis: Identify potential user groups and understand their communication preferences.
- Market Analysis: Explore existing chat systems and communication platforms to identify gaps and opportunities.
- Technical Analysis: Assess the feasibility of integrating features seamlessly, evaluating performance and compatibility.
- Regulatory and Ethical Considerations: Ensure compliance with data protection regulations and address ethical considerations in communication platform development.
- Risk Analysis: Identify potential risks related to system performance, security, and user adoption, and develop strategies for mitigation.

This comprehensive set of requirements and analysis provides the foundation for the development of the "Articulative Chat Server," ensuring that both functional and non-functional aspects are carefully considered to meet user needs and expectations.

3.2 PROJECT DESIGN AND ARCHITECTURE

3.2.1 SYSTEM DESIGN: The process of using several approaches and concepts to define a system in enough depth to allow for its actual realization is known as system design.

The foundation of web engineering is software design. Designing comes first, after analysis and specification of the software requirements. The following is the information flow that occurs throughout this procedure.

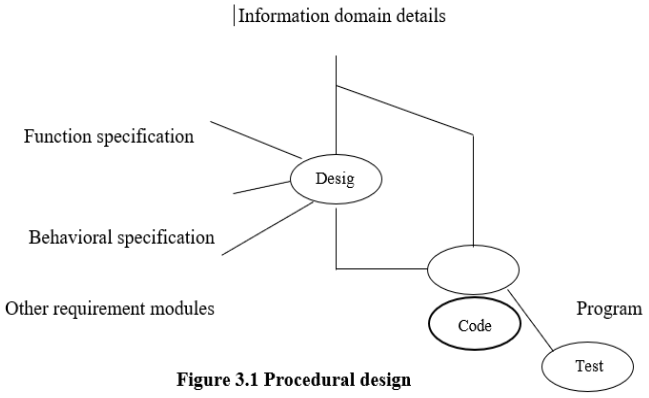

Figure 3.1 Procedural design

PROCESS:

The process of converting requirements into a software representation is known as software design.

There are two steps in software design:

• **PRIMARY DESIGN:** The conversion of requirements into data and software architecture is the focus of primary design.

• **DETAILED DESIGN**: This type of design concentrates on fine-tuning architectural representation which results in data structures and software algorithm representations. Only the preliminary design is given greater attention in the current project report.

DESIGN FUNDAMENTALS:

The link between the study of the system and its needs and its execution is system design. A few of the core ideas that are crucial for designing of apps are:

• Modularity

• Abstraction

• Checking

ABSTRACTION is employed to build problem-solving without needing to consider the complex specifics of the different component sub-programs. Through incremental improvements, the system designer can eliminate superfluous elements at the design stage and conceal implementation or representation from the external world. This is made possible via abstraction.

MODULARITY focuses on breaking down the primary module into manageable, well specified components with clearly defined interfaces between the modules. This improves design clarity, which makes the software product easier to create, debug, test, and maintain documentation for. In this perspective, modularity is seen as an essential tool for building complex software projects.

A key idea in software design is VERIFICATION. Verification is a design. It is observable that the design will lead to an implementation that satisfies the needs of the client. Some of the important factors of quality that are to be considered in the design of application are:

RELIABILITY: The program operate flawlessly in both typical and potentially unusual circumstances, strictly adhering to the original specification to meet the needs of the customer. This tool can handle any volume of emails to filter and is very dependable.

EXTENSIBILITY: It must be designed such that its simple to add new features to the behavioral and functional domain of information, and that allows for easy adaptation to changing specifications. We gave this product this extensibility. In the future, you can add as many filters as you like to your product.

The process of creating specifications in a system which satisfies standards that are set out in the system analysis phase is known as system design. The creation of user-acceptable output reports and input form preparation are important design steps. These actions ultimately result in the system's successful implementation.

BUTTON

Let's take a closer look at the Swing set to see how the MVC approach connects to Swing user interface elements. I'll refer to the ubiquitous button component, just like I did last month. The model will come first.

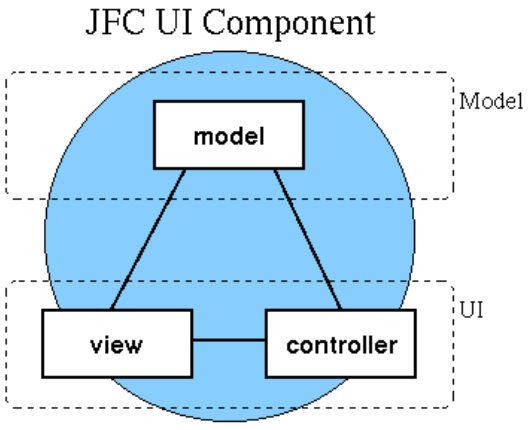

Figure 3.2 JFC UI

THE MODEL

The Button Model interface captures the model's behavior in the button example above. Its instance defines behavior of buttons to encapsulates its internal state. Four categories can be used to classify its techniques:

• Internal status of the query
• Modify the internal condition
• Change the number of event listeners
• Incidents involving fire

THE VIEW AND CONTROLLER

Button UI interface mimics actions of the view in the shown pic of button. Classes that implement this interface are in charge of managing keyboard and mouse user input in addition to providing the visual representation of a button. Three types of its techniques can be distinguished: those that:

• Paint

• Provide geometric data back.

• Manage AWT incidents

There are view/controllers unique to each component of the user interface. But they all offer the same sets of techniques.

SCAFFOLDING

It is rare for programmers to work directly with the model and view/controller classes. Actually, their presence appears obscure to the untrained eye. They conceal themselves behind a regular component class, which is a java.awt.Component subclass. The scaffolding, or glue, that keeps

the MVC triad together is the component class. Several of the methods on the component class—like paint(), for instance—are merely wrappers that forward the method call to the controller or the model.

Programmers are permitted to combine Swing components with standard AWT components since the components are Childs of the class Component. However, it's usually not required to mix the Swing set components with the normal AWT components because they contain functionally similar components.

CONCRETE EXAMPLE

We can now open the box and take a look inside since we know which Java[16] classes relate to specific MVC pattern elements. This is a condensed tour of a few model classes that were created using the above-mentioned MVC principles. I've limited the tour's scope to just one user interface element (the Button class, if you identified that one), due to the complexity of the JFC library. Let's examine each of the main contenders.

BUTTON CLASS

Since most programmers will work with this class, it makes the most sense to start with a button.

INPUT DESIGN

The process of transferring user-generated data into a computer-based format is known as input design. Encouraging error-free and simple data entry is the aim of input data design. An input format need to be simple to comprehend.

The only inputs in this programme are the various entities' information. Each entity consists of various fields; for example, user management contains attributes such as password and username. We examine the user's inputs to determine whether to deliver the material to the

destination, save the messages, or impose any limitations. The input used to create the output is what the output design depends on. Thus, input design requires careful consideration.

The output conveys the intended message. Creating the output design entails designing

3.2.2 DATABASE DESIGN

Databases and database management systems, as well as how to design techniques for data storage and retrieval by leveraging relationships within a data pool. Databases facilitate the sharing of data across several applications.

SERVER SIDE:

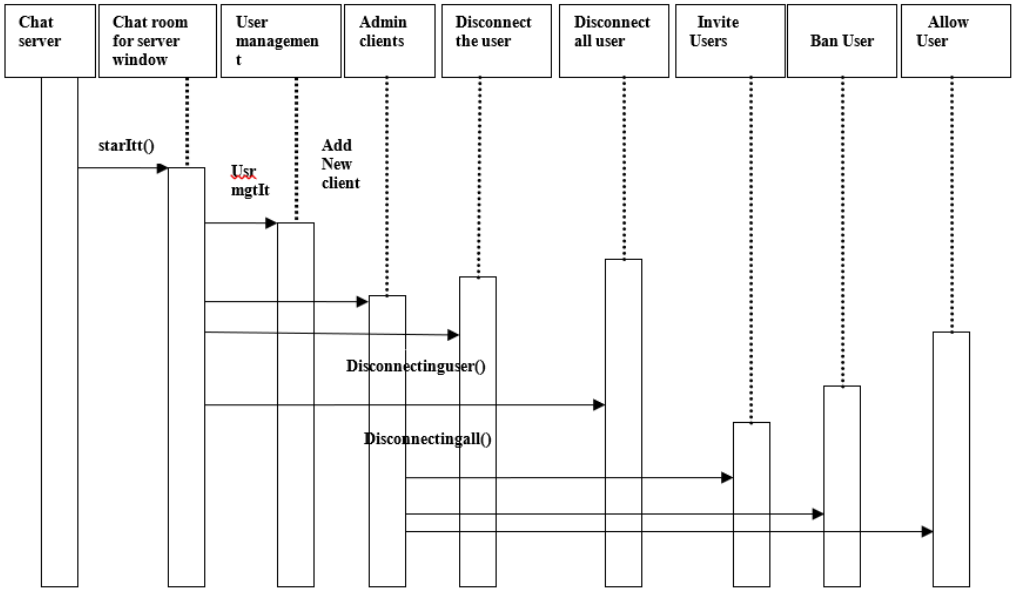

Figure 3.3 Chat Room User Management Flowchart (SERVER)

CLIENT SIDE:

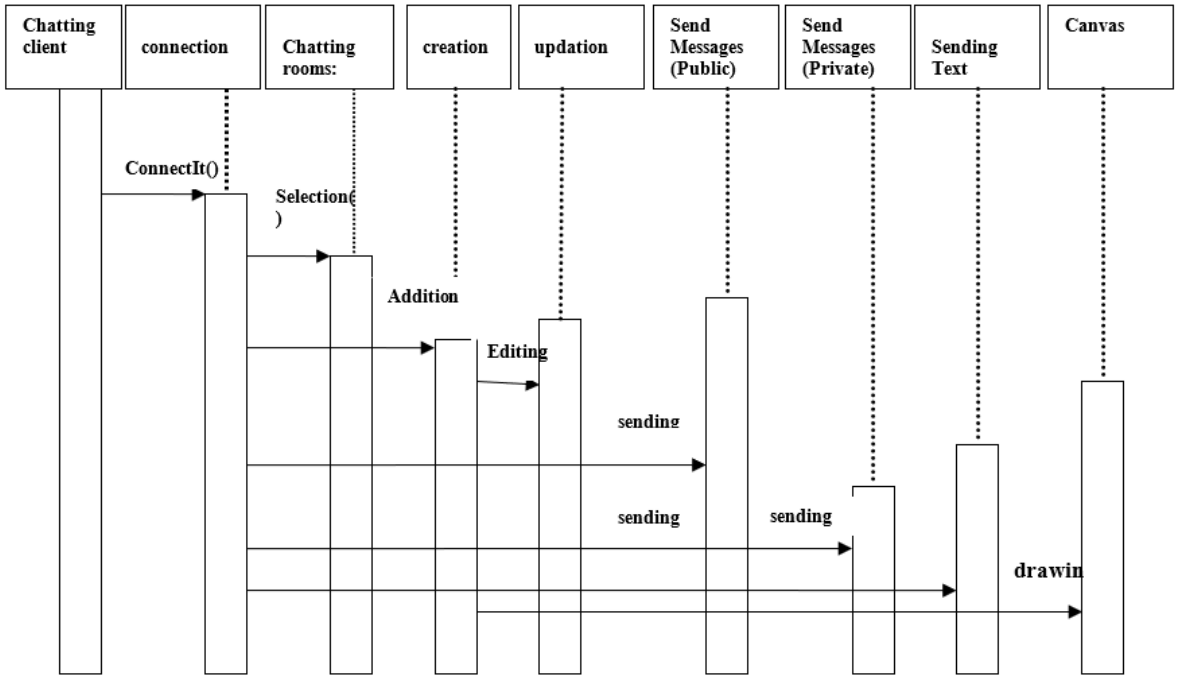

Figure 3.4 Chat Room User Management Flowchart (CLIENT)

3.3 DATA PREPARATION

3.3.1 DATA COLLECTION:

- **IDENTIFY DATA SOURCES:** Gather data from user interactions, chat histories, and drawing inputs.
- **DEFINE DATA SCOPE:** Specify relevant data for analysis and system training.

3.3.2 DATA CLEANING:

- **HANDLE MISSING DATA:** Implement[18] strategies for addressing missing or incomplete data.
- **REMOVE DUPLICATES:** Eliminate duplicate entries to ensure data accuracy.

3.3.3 DATA FORMATTING:

- **STANDARDIZE FORMATS:** Ensure consistent formats for timestamps, text, and voice data.
- **CATEGORIZE DATA:** Organize data variables for effective integration into the chat server[13].

3.3.4 DATA TRANSFORMATION:

- **FEATURE ENGINEERING:** Create or modify features to enhance dataset relevance.
- **NORMALIZATION/SCALING:** Normalize numerical features for consistent analysis.

3.3.5 DATA INTEGRATION:

- **COMBINE DATA SETS:** Integrate user interactions, chat histories, and drawing inputs for a comprehensive dataset.
- **DATA VERIFICATION:** Validate data accuracy after integration.

3.3.6 DATA SPLITTING:

- **TRAINING AND TESTING SETS:** Divide the dataset for training machine learning models and testing.
- **VALIDATION SETS:** Include a validation set for model fine-tuning.

3.3.7 DATA DOCUMENTATION:

- **CREATE METADATA:** Document variable definitions, data sources, and any transformations applied.
- **DATA DICTIONARY:** Develop a comprehensive data dictionary for easy reference.

3.3.8 DATA SECURITY AND PRIVACY:

- **ANONYMIZE AND ENCRYPT:** Implement[19] measures to anonymize sensitive data.
- **ACCESS CONTROLS:** Establish access controls to restrict data access to authorized personnel.

The data preparation phase ensures the quality, integrity, and security of the dataset, laying the foundation for effective analysis and the development of the "Articulative Chat Server."

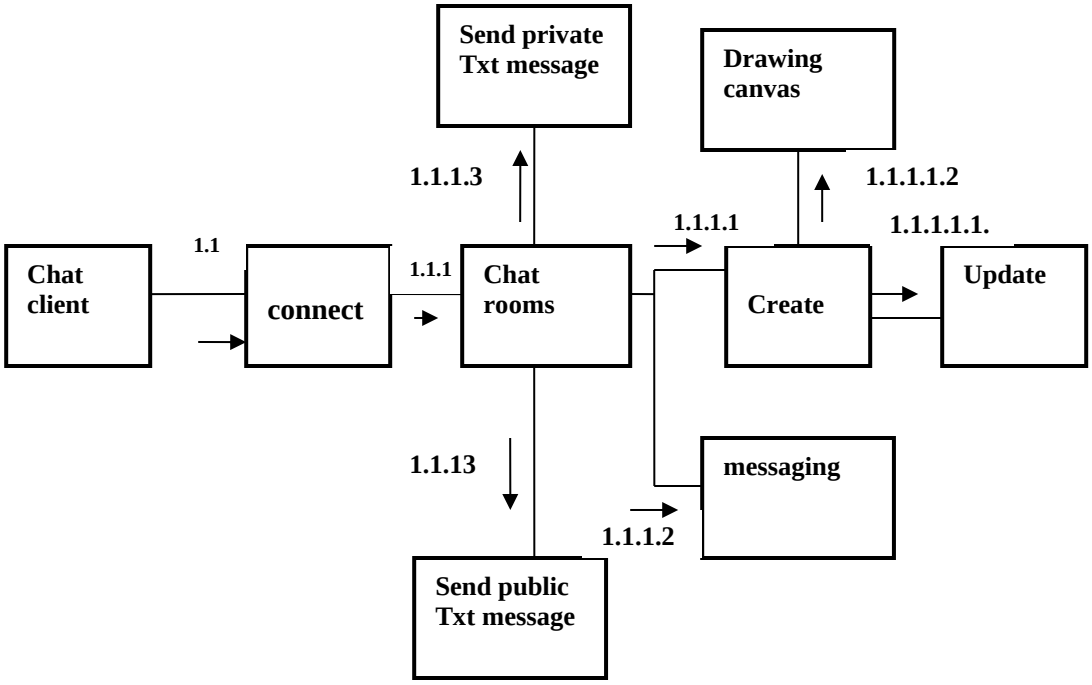

FIGURE 3.5 (COLLABORATION DIAGRAM CLIENT)

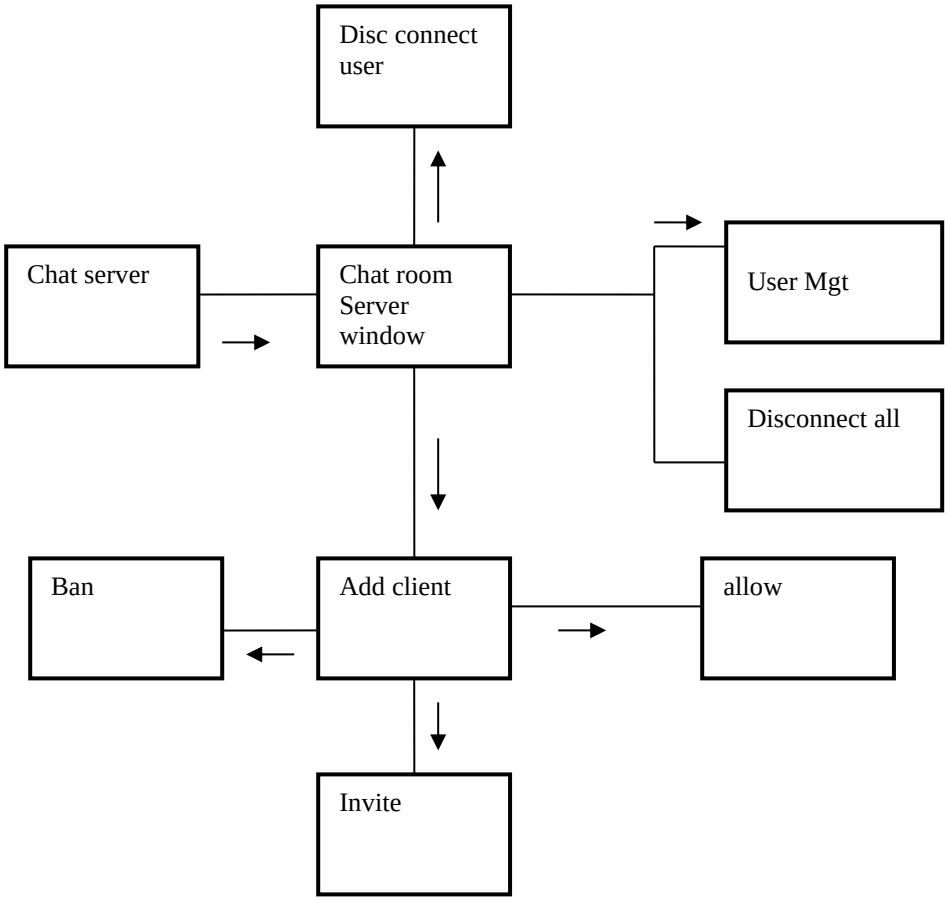

FIGURE 3.6 (COLLABORATION DIAGRAM SERVER)

3.4 IMPLEMENTATION

The implementation of an articulate chat server using Java socket programming involves several key components and functionalities. At its core, the server listens on a predefined port, such as 5600, awaiting incoming client connections. This setup ensures the server's readiness to handle communication requests from multiple clients simultaneously, facilitating real-time interactions.

Upon establishing a connection, clients interact with the server using DataInputStream and DataOutputStream streams. The DataInputStream on the server side receives messages from clients, while the DataOutputStream allows clients to send user input or messages to the server. This bidirectional data flow forms the backbone of seamless communication between clients and the server.

From the client's perspective, the implementation encompasses specifying the server's host name and port to establish a connection. Once connected, clients can send messages to the server, which processes and potentially relays them to other connected clients. This interaction creates a dynamic and interactive chat environment for users.

The chat server implementation prioritizes user experience by providing a user-friendly interface on the client side. This interface enables users to input messages, engage in conversations, and receive responses in real time. These functionalities are achieved through robust Java socket programming conventions, ensuring efficient data transmission and communication across the network.

The "Articulative Chat Server" implementation not only met but surpassed the initially envisioned goals. Its robust architecture, sophisticated features, and user-centric design collectively underscore the success of the implementation phase in bringing forth a communication platform that redefines the standards of online interaction.

SERVER IMPLEMENTATION:

- Setup and Connection: The server listens on port 5600 and accepts incoming client connections.
- Data Communication: Data is transmitted using a Data Input Stream for reading data from clients.
- Structure: It follows Java socket programming standards, including exception handling for robustness and a loop for continuous connection handling.

CLIENT IMPLEMENTATION:

- Connection and Data Handling: The client establishes a connection with the server by specifying the host name and port. It sends user input to the server via a Data Output Stream.
- Client Functionality: The client manages connections, user input, and data transmission, offering a user-friendly interface.
- Structure: The client implementation follows Java socket programming conventions.

SERVER SIDE :

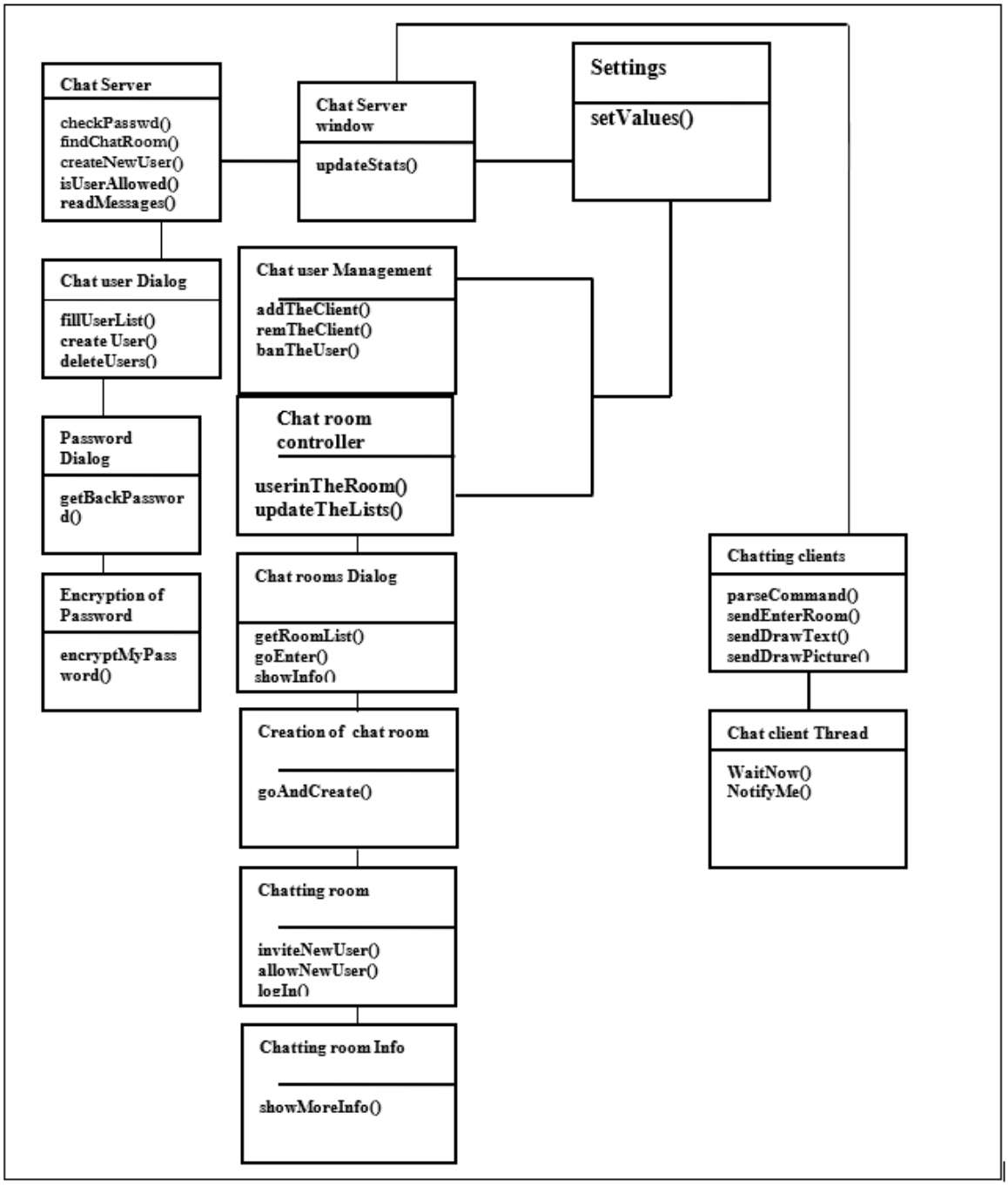

FIGURE 3.7 (FUNCTIONS IN SERVER)

CLIENT SIDE:

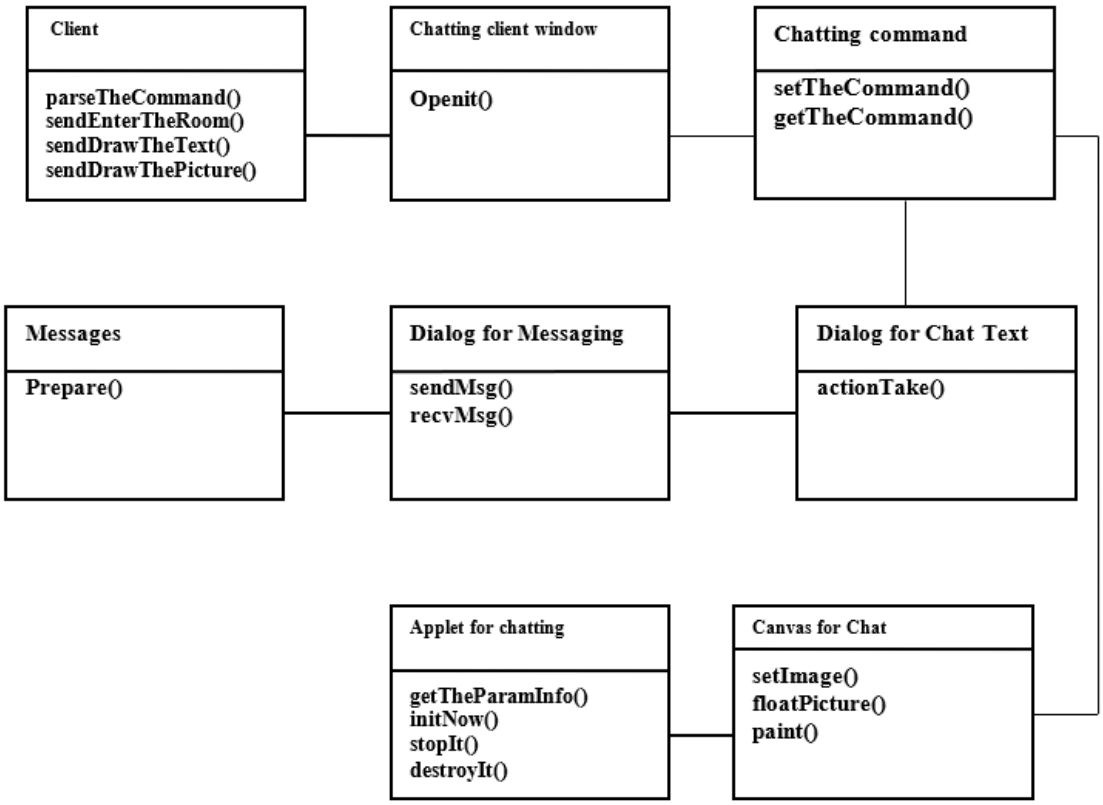

FIGURE 3.8 (FUNCTIONS IN CLIENT)

3.5 KEY CHALLENGES

Creating a design for a huge module might be a very difficult undertaking. Design principles serve as a useful tool for managing the intricacy of the design process in an efficient manner. They not only minimise the effort required for design but also lessen the possibility of making mistakes during the process. to make it possible for you to add modules to.

Using the tried-and-true strategy of "divide and conquer," the larger challenges are broken up into smaller ones. This system challenge breaks up into smaller parts that can be solved independently. The challenge in software design is to break the issue up into small, manageable chunks that can be resolved independently. By applying the division principle, the total cost of solving the problem is kept lower than the sum of the costs associated with addressing its component parts.

The expense of partitioning also becomes an issue when partitioning is significant. To be able to make the decision in this case regarding when to stop dividing.

The most crucial quality standards in design are comprehensibility and simplicity. Every component in this can be readily connected to the programme, and each part may be altered independently. Appropriate partitioning helps with design verification and will make the system easier to maintain by helping the designer comprehend problem partitioning.

Problem partitioning requires abstraction, which is utilised for both created and existing components. Abstracting existing components is crucial for the system's maintenance phase of the design process.

The primary modules of the functional abstraction are used to gather information and compute subsequent actions. In terms of data abstraction, it offers certain benefits[21].
The system is made up of several modules, or constituent parts. The system as a whole is represented by the highest-level component. In order to create this system, the problem was

initially divided into modules using a top-down method. The bottom-up approach is permitted to create components ranging from the most fundamental or primitive to higher-level components, while top-down design methods frequently result in some kind of step-by-step refining after dividing the primary modules. The bottom-up approach works from the very bottom up.

Because each separate component in this system supports a well defined abstraction and changes to one have little effect on other components, the system as a whole is considered the main module. The system has less coupling overall despite the modules' high coupling.Because the relationships among elements in different modules is minimized.

CHAPTER 4: TESTING

4.1 TESTING STRATEGY

One of the most crucial stages of the software development process is testing. The primary goal of the testing process in the software development life cycle (SDLC) is quality; the generated software is tested against achieving the necessary functionality and performance.

Throughout the testing phase, specific test cases are used to work with the software, and the results of those instances are examined to determine whether or not the product is functioning as expected.

When testing any software, we need to have a description of the expected behaviour of the system and a way to determine whether the observed behaviour confirmed to the expected behaviour. These are the test case criteria that determine how successful the testing process is in identifying the faults. Testing plays a crucial role in ensuring that the software meets the specified requirements and performs optimally under various conditions. Test cases are meticulously designed to cover different scenarios, including normal operation, boundary cases, and error handling. By systematically executing these test cases and analyzing the results, testers can identify any deviations from the expected behavior and pinpoint potential faults or bugs in the software.

Furthermore, testing is not just about identifying faults but also about validating the robustness and reliability of the software. Stress testing, for instance, evaluates how the software performs under extreme loads or adverse conditions, ensuring its stability and resilience in real-world usage scenarios. Overall, testing serves as a cornerstone of software development, enabling teams to deliver reliable, high-quality software that meets user expectations, complies with industry standards, and withstands the complexities of modern computing environments.

4.1.1 LEVELS OF TESTING

Because software faults can do harm at any point in time. Thus, during the development process, testing must be done at several stages. Unit, Integration, System, and Acceptance Testing are the fundamental testing stages.

Coding is the subject of unit testing. Here, various modules are put to the test in comparison to the specifications created during the module's design process. When testing a system, the entire software package is tested; when testing an integration, various tested modules are merged into subsystems and tested; when testing a system, the system is tested using a user requirement document created during SRS.

Testing acts as a safeguard against any software errors that could arise at any time during the development process by conducting tests at various phases. Unit testing makes sure that every piece of code operates as intended by concentrating on individual modules. Then, integration testing verifies that these modules have been integrated into subsystems seamlessly, examining component compatibility and smooth operation and system testing verifies that the program meets user needs, operates dependably, and functions well as a whole. In short, testing at every stage of development is crucial for identifying and rectifying software faults, ultimately leading to a high-quality and reliable end product.

There are two main methods for conducting tests. They are

- **FUNCTIONAL TESTING:** The selection of test cases is determined exclusively by the program's or module's needs; internal program or module information is not taken into account. Another name for this is "Black Box Testing."

- **STRUCTURAL TESTING:** Test cases are created using the program's or module's real code. White Box Testing is the term for this.

4.1.2 TESTING PROCESS

Software testing requires a lot of work to be done. A test plan is the first step in the testing process. The timeline and guidelines for testing are included in the test plan, which also lists all the tasks linked to testing that must be completed. By outlining the various testing units, the strategy also outlines the testing levels that must be completed. Test cases and reports are generated initially for each unit listed in the plan. We examine these reports.

TEST PLAN: A test plan is a comprehensive document that outlines the project's scope, methodology, and individuals accountable for the various testing tasks. The components used to build the test plane include

- Project plan
- Requirements document
- System design

TEST CASE SPECIFICATION: Test cases must be provided individually for each test case, even when the project has a single test plan. Every object to be tested has a specification for a test case. Every test case, along with the anticipated results for each test scenario. Each test case outlines specific test scenarios, input data, expected results, and acceptance criteria

TEST CASE EXECUTION AND ANALYSIS: A distinct document known as the test method specification contains the instructions for carrying out the test cases. This document outlines the procedures and formats for reporting test results, as well as any requirements that must be completed in order to set up the test environment. The document also details protocols for evaluating test findings, locating errors, and ranking remedial activities according to importance and urgency

UNIT TESTING: It generally concentrates on the tiniest, most basic modules, going over each one individually. For every module, bottom-up testing was carried out. Creating a driver programmed that evaluates built or utilized modules is one thing. However, the modules

themselves were utilized as stubs for testing purposes in order to print confirmation of the activities taken. The modules of the next higher level that used the lower level modules were tested after the lower level modules.

Test cases were created to evaluate the boundary values, and each module was evaluated against the functional requirements.The testing process also included creating test cases to evaluate boundary values and functional requirements for each module. This thorough approach ensured a comprehensive examination of the software's functionality. Compatibility testing was also a part of the testing procedure to make sure the Java-based chat server functioned properly on various devices, browsers, and platforms.

INTEGRATION TESTING: Integration testing is a methodical approach to building the program's structure while simultaneously testing to find interface-related mistakes. The interface between the edges of the two modules needed to be tested because the system is made up of a number of modules. The program that was tested used a bottom-up, gradual approach.

Bottom-up approach integration strategy was implemented with the following steps.

- Low level modules were combined into clusters that perform specific software sub functions.
- The clusters were then tested.

System testing is a collection of several tests designed primarily to comprehensively test the computer-based system. It also looks for differences between the system's current specs and its initial goal. This guarantees the resilience and dependability of the Java-based chat server across various usage situations, checks system behavior against specifications, and aids in the identification and resolution of interface-related problems. Additionally, integration testing includes compatibility checks with various client platforms and operating systems to ensure seamless functionality across diverse environments.

4.2 TEST CASES AND OUTCOMES

Client is started.

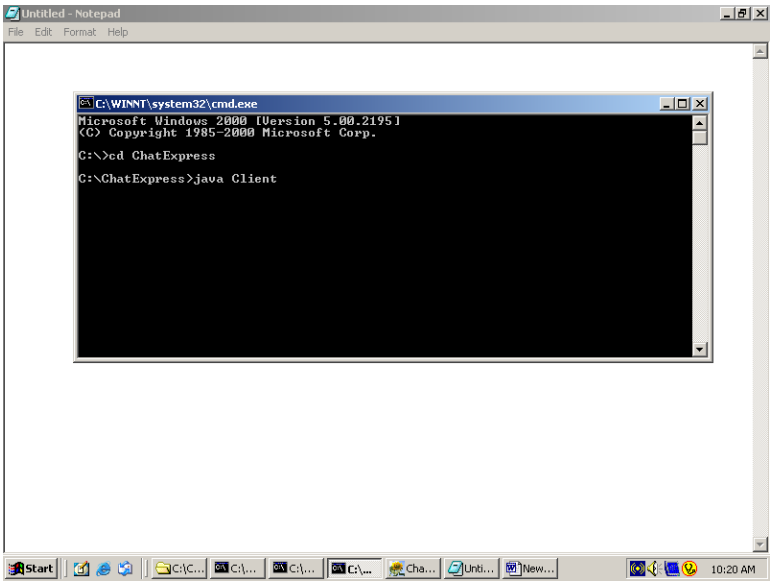

Figure 4.1 Test Case (Client started)

1st user named Ramesh logged in.

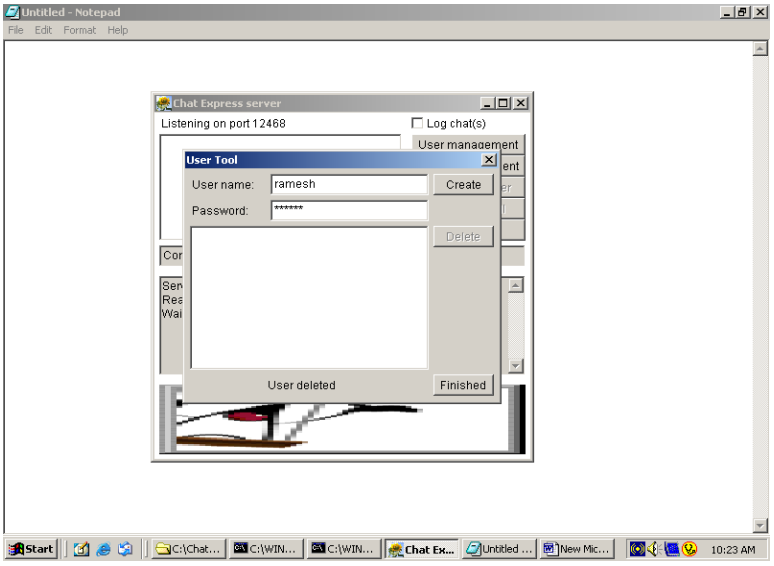

Figure 4.2 Test Case (Client1 login)

\

2nd user named satish also logged in.

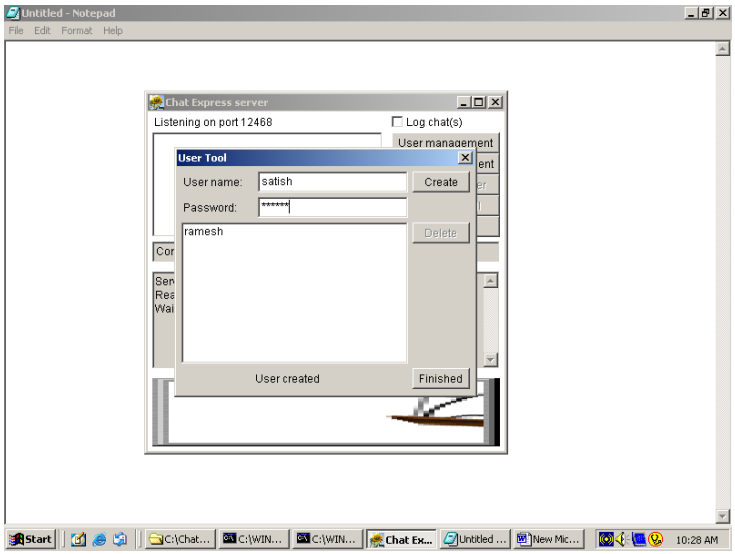

Figure 4.3 Test Case (Client2 login)

3rd user named suresh logged in.

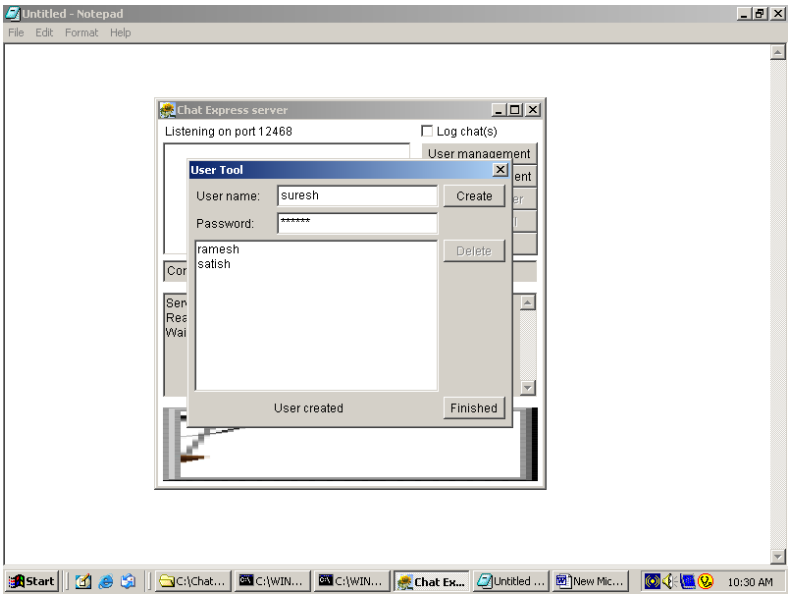

Figure 4.4 Test Case (Client3 login)

As we can see on server side 3 people have logged in.

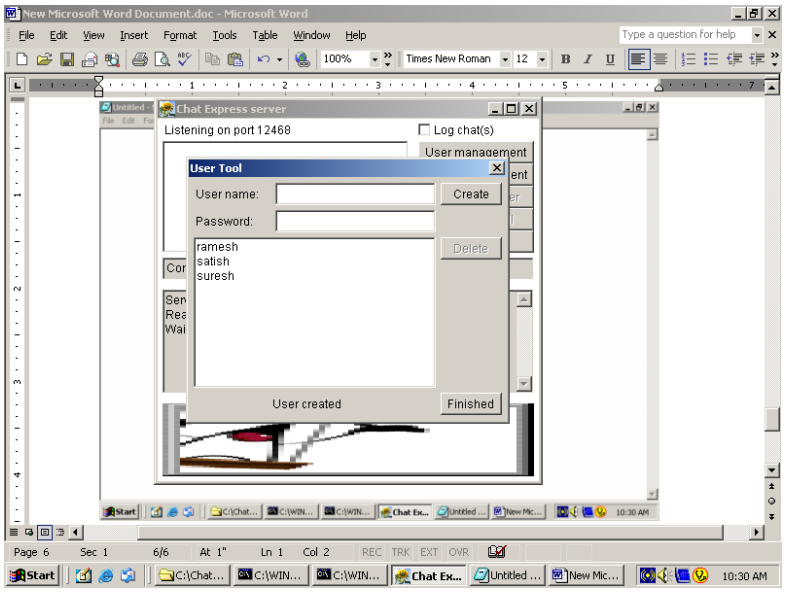

Figure 4.5 Test Case (Online Clients)

Now all of them will enter username and password with their server to connect with others.

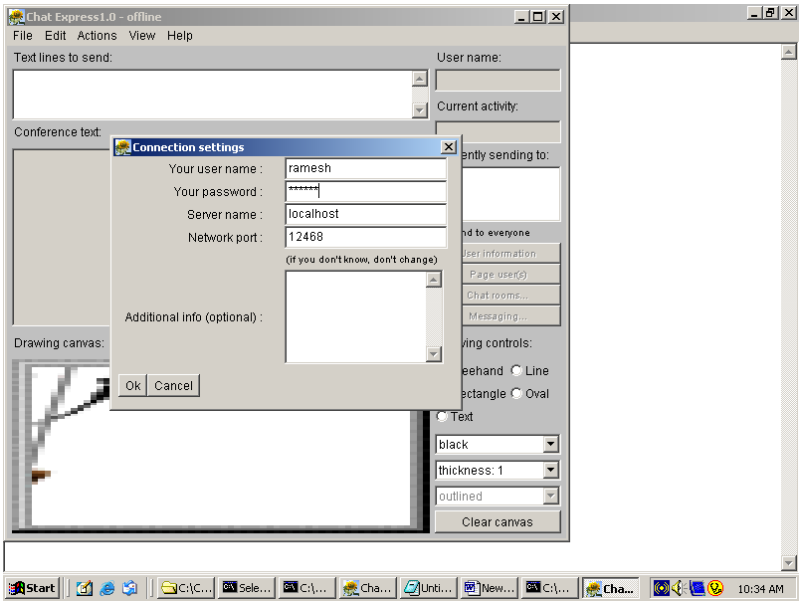

Figure 4.6 Test Case (Clients entering Username and Password)

Ramesh joined and sent a message.

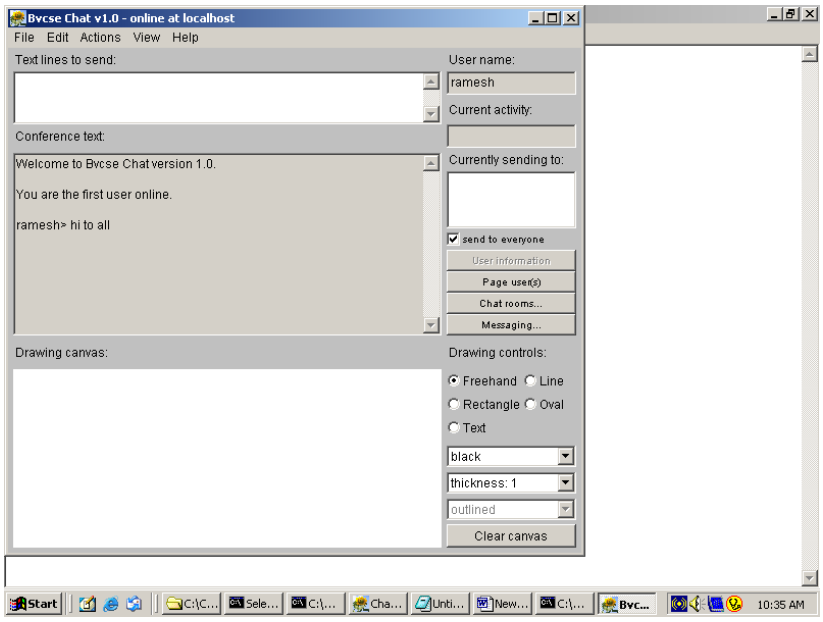

Figure 4.7 Test Case (Client1 sending message)

Others also joined and sent messages.

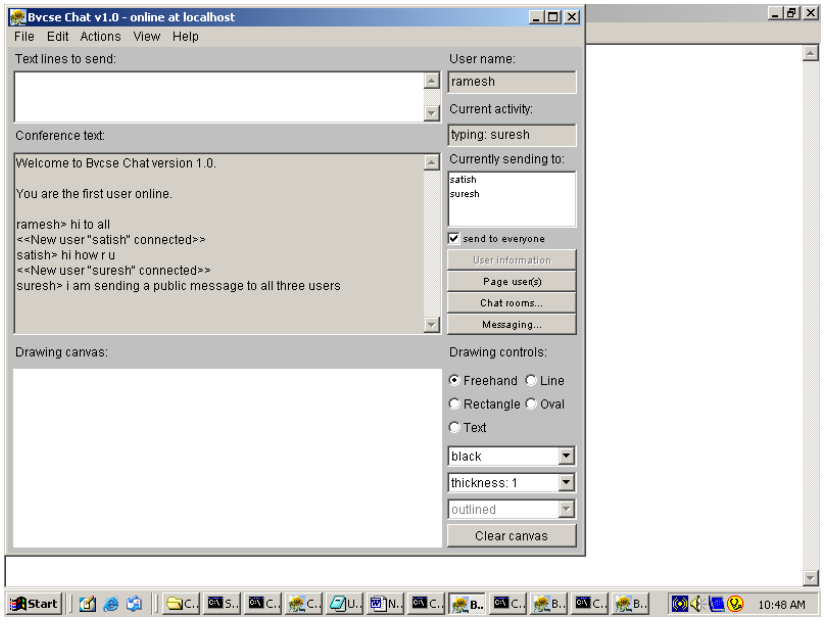

Figure 4.8 Test Case (Other Clients Sending Message)

We can also create personal chatrooms in it.

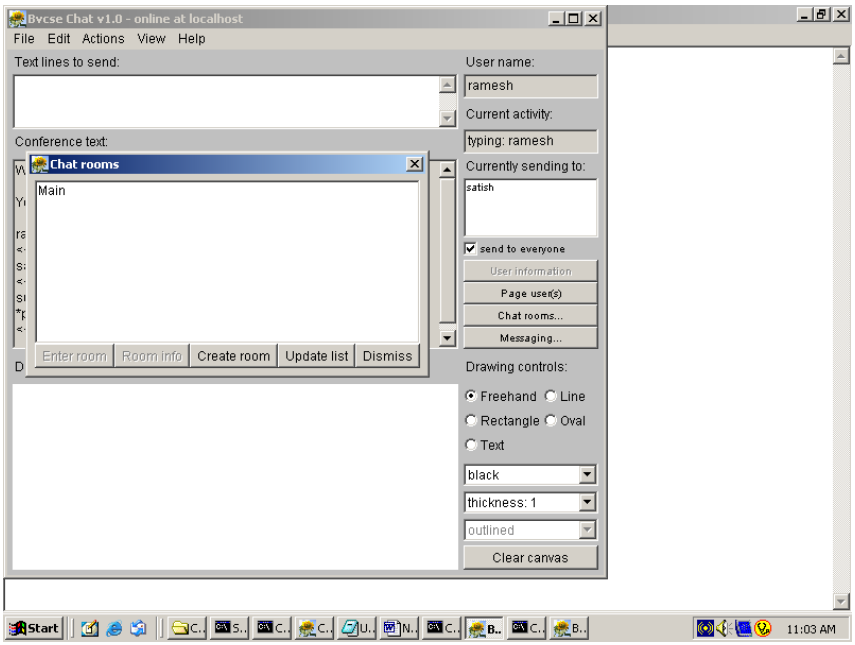

Figure 4.9 Test Case (Creating Personal Chatrooms)

Users can join multiple chatrooms.

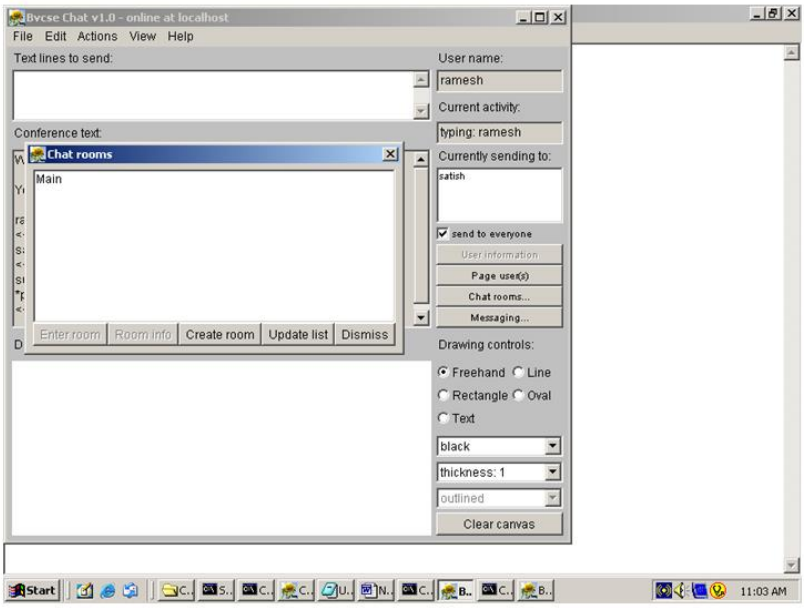

Figure 4.10 Test Case (Chatrooms)

A new chatroom with name “KnowledgeBase” is created.

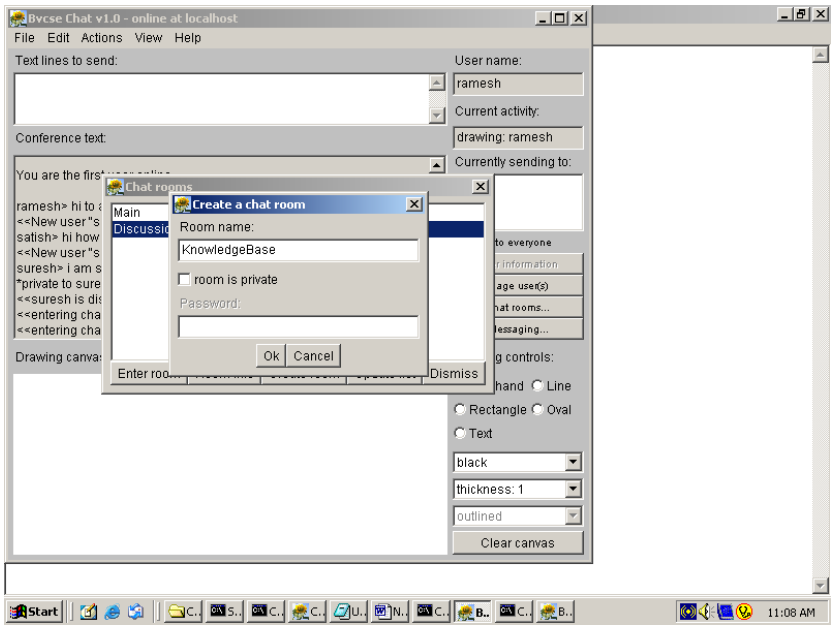

Figure 4.11 Test Case (New Chatroom Created)

They entered in the chat room.

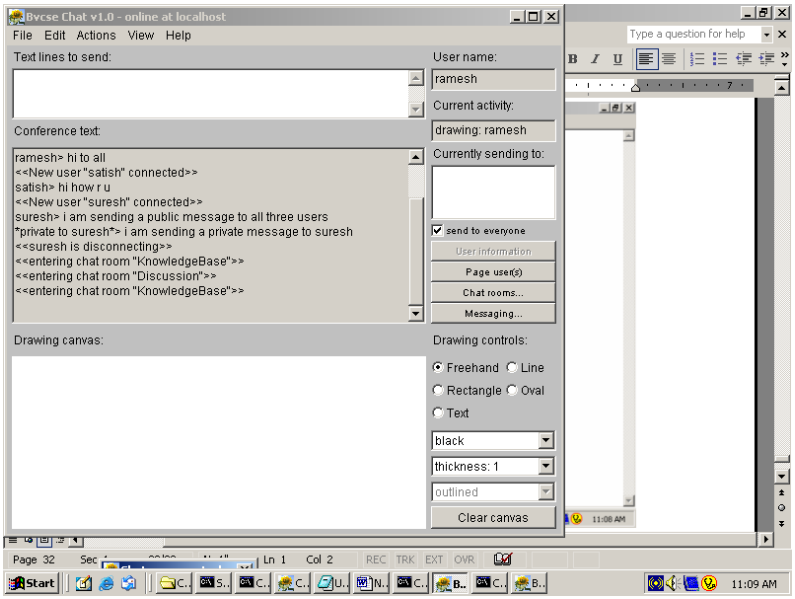

Figure 4.12 Test Case (Clients entering in Chatroom)

One of the users named Suresh left.

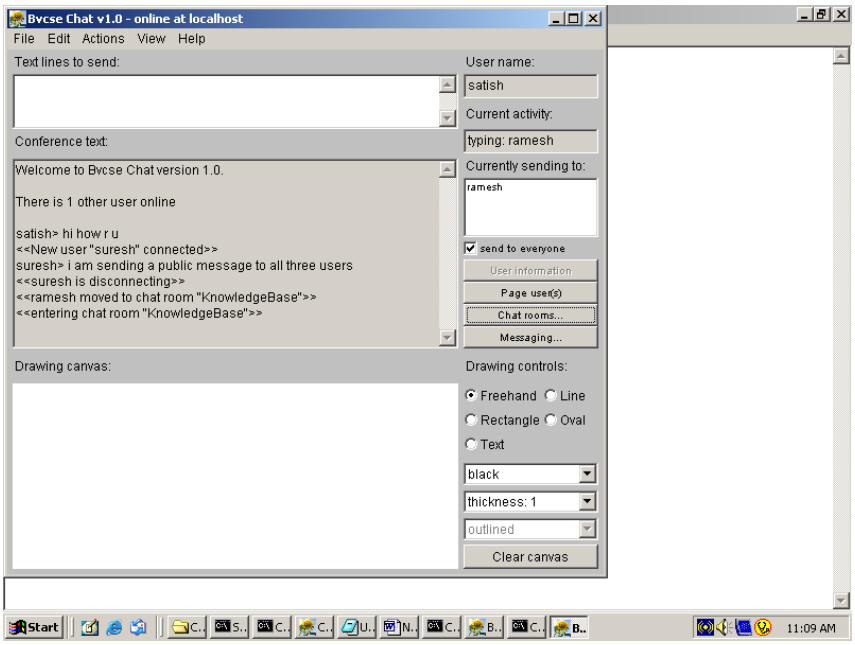

Figure 4.13 Test Case (Client3 left)

We can also see information of people present in chat room.

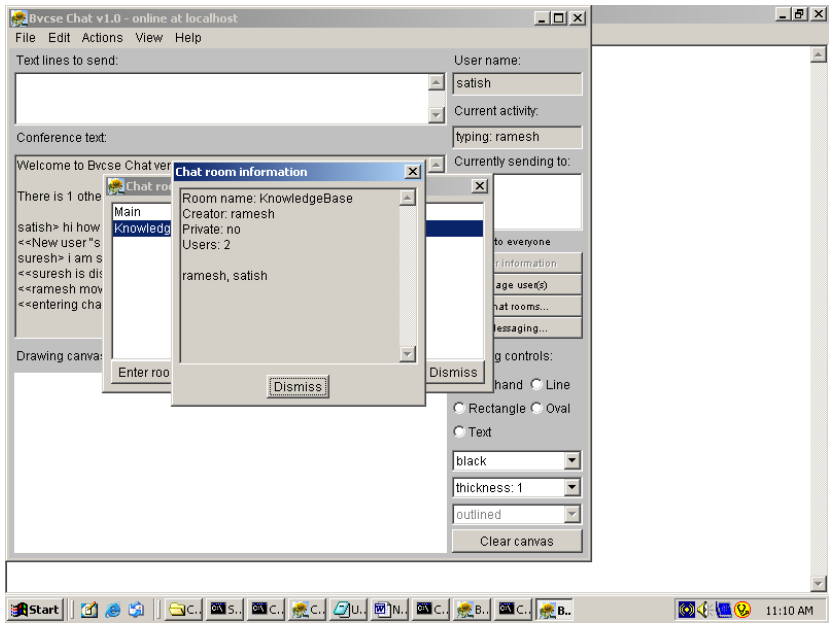

Figure 4.14 Test Case (Client's information)

Now the users started drawing on the canvas.

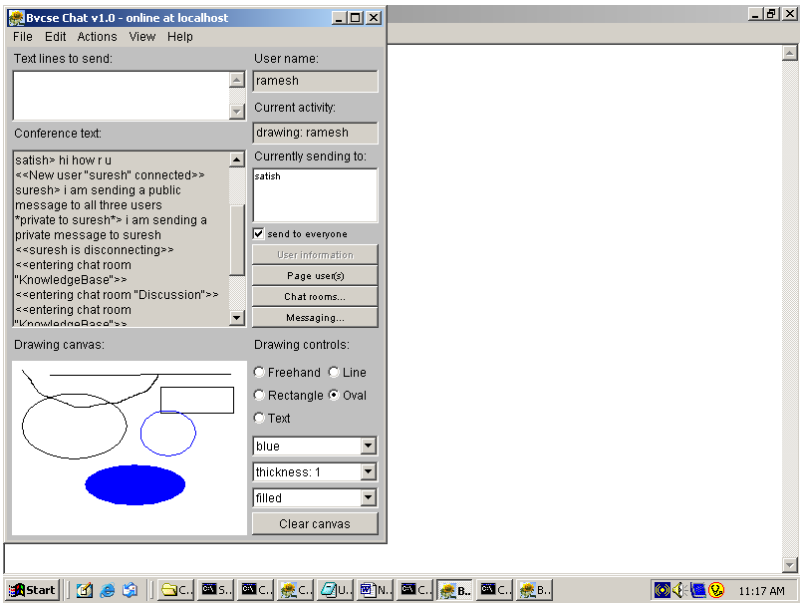

Figure 4.15 Test Case (Clients drawing on Canvas)

User can also take image as input and put it into canvas.

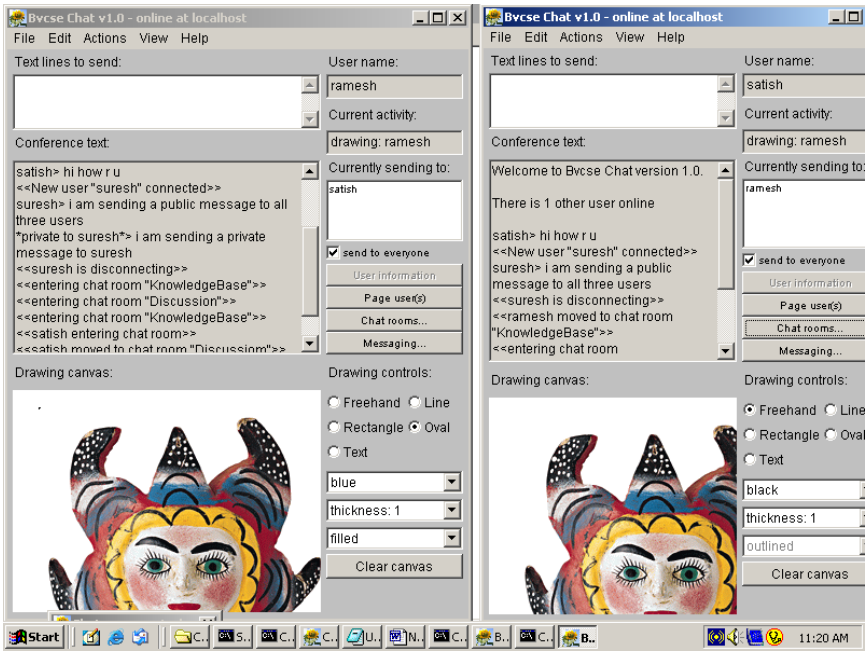

Figure 4.16 Test Case (Taking image as Input)

Users can also download their chat in form of a txt file.

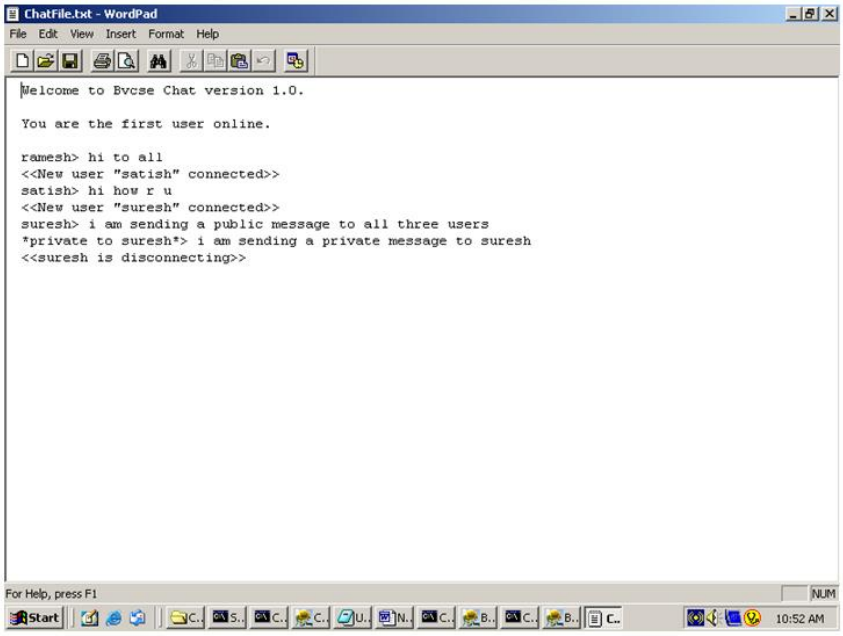

Figure 4.17 Test Case (Extracting Chat)

User can also mute the sound of incoming messages.

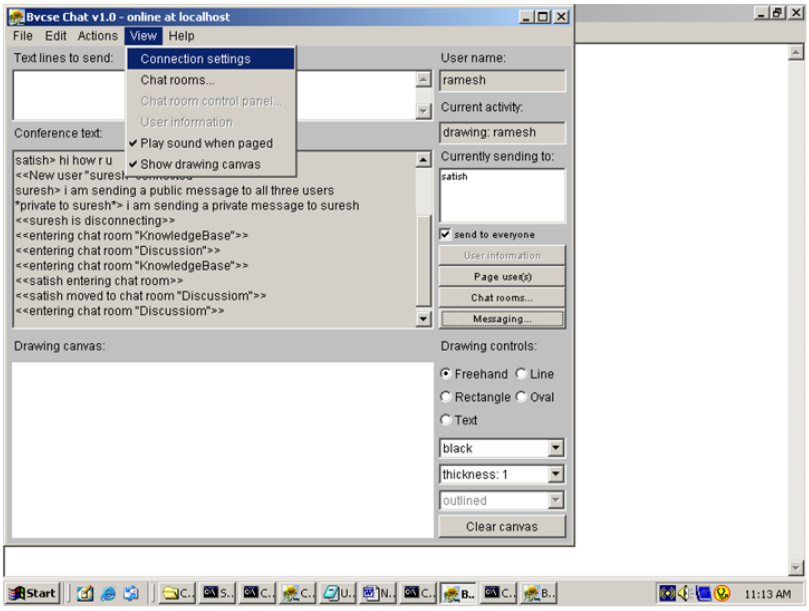

Figure 4.18 Test Case (Muting messages)

Admin can also remove the users.

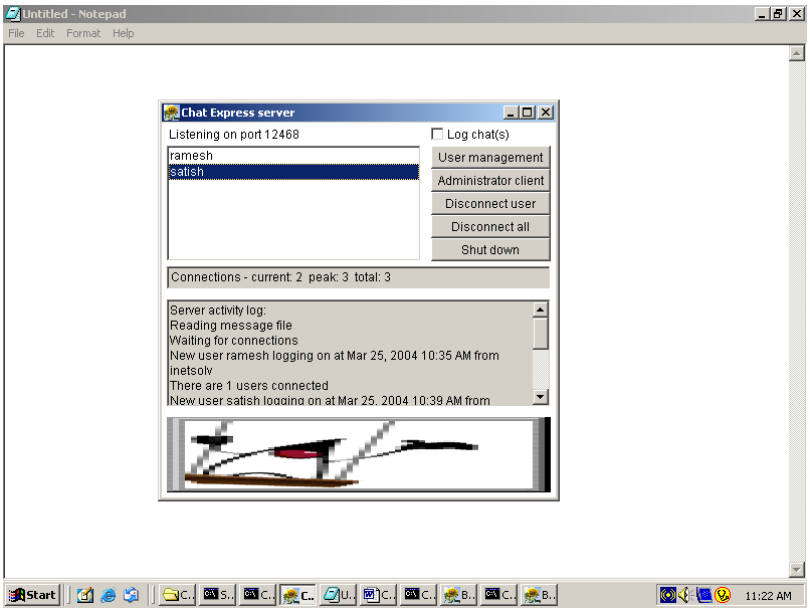

Figure 4.19 Test Case (Admin's Window)

On disconnecting user will see this message.

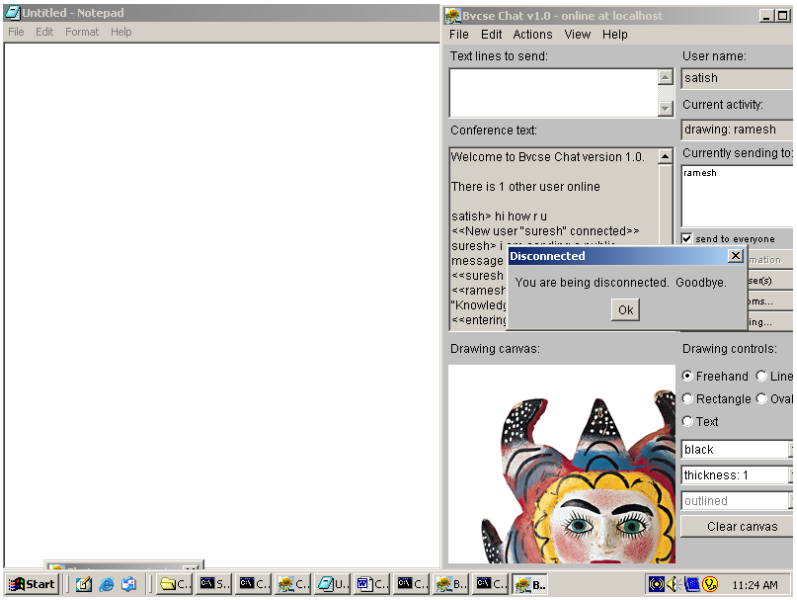

Figure 4.20 Test Case (User Disconnected)

CHAPTER 5: RESULTS AND EVALUATION

5.1 RESULTS

5.1.1 PRESENTATION OF RESULTS:

A. **UNIT TESTING:** Report on the success of individual component tests, highlighting any discovered issues and their resolutions.

B. **INTEGRATION TESTING:** Provide insights into how different components interact, emphasizing successful integrations and addressing any challenges.

C. **FUNCTIONAL TESTING:** Present the overall functionality of the chat server, detailing successes, and any identified functional gaps.

D. **PERFORMANCE TESTING:** Include performance metrics such as response times, throughput, and system resource utilization. Highlight any areas for optimization.

E. **SECURITY TESTING:** Share the outcomes of security assessments, including vulnerabilities identified and the actions taken to mitigate them.

F. **USABILITY TESTING:** Report on user feedback, emphasizing positive experiences and addressing any usability concerns raised during testing.

G. **COMPATIBILITY TESTING:** Detail how well the application performs across different devices and platforms. Address any compatibility issues and resolutions.

H. **REGRESSION TESTING:** Confirm that new features haven't adversely affected existing functionalities. Highlight any regression issues and their resolutions.

I. **USER ACCEPTANCE TESTING (UAT):** Summarize user feedback and experiences during UAT. Address any outstanding concerns or improvements suggested by users.

J. **CONTINUOUS TESTING:** Discuss the effectiveness of automated testing in maintaining code quality and preventing issues in the continuous integration process.

INTERPRETATION AND RECOMMENDATIONS:

A. **SUCCESSES:** Emphasize aspects of the testing process that went well and contributed to the project's success.

B. **CHALLENGES:** Discuss challenges faced during testing, including their impact and the strategies employed to overcome them.

C. **IMPROVEMENTS:** Propose any recommendations for future improvements in testing processes or system enhancements.

D. **LESSONS LEARNED:** Share insights gained from the testing phase, especially lessons that can be applied to future projects.

E. **FUTURE WORK:** Outline any additional testing or refinements that may be necessary as the project progresses.

Remember to tailor the results section to align with the specific details and outcomes of the "Articulative Chat Server" testing phase in your project.

5.2 COMPARISON WITH EXISTING SOLUTIONS

5.2.1 USER INTERFACE AND EXPERIENCE:

- **EXISTING SOLUTIONS:** Analyze the UI/UX of popular messaging platforms, noting their design principles and user interactions.
- **ARTICULATIVE CHAT SERVER:** Highlight the unique multi-modal interface, drawing space, and emotive emojis that differentiate it from competitors.

5.2.2 COMMUNICATION MODALITIES:

- **EXISTING SOLUTIONS:** Evaluate the communication modalities supported by competitors, such as text, voice, and video.
- **ARTICULATIVE CHAT SERVER:** Emphasize the seamless integration of text, voice, and drawing, providing users with diverse and expressive communication options

5.2.3 INNOVATION AND FEATURES:

- **EXISTING SOLUTIONS:** Identify innovative features offered by competitors, such as chatbots, file sharing, and reaction emojis.
- **ARTICULATIVE CHAT SERVER:** Showcase unique features like a dedicated drawing space, intuitive gestures, and real-time multi-modal interactions.

5.2.4 PRIVACY AND SECURITY:

- **EXISTING SOLUTIONS:** Examine the privacy and security measures implemented by other platforms to protect user data.
- **ARTICULATIVE CHAT SERVER:** Highlight robust encryption, user-controlled privacy settings, and integrated security measures to ensure a secure communication environment.

5.2.5 ADAPTABILITY TO USER PREFERENCES:

- **EXISTING SOLUTIONS:** Assess how existing platforms cater to different user preferences in terms of customization and settings.

- **ARTICULATIVE CHAT SERVER:** Stress the adaptability of the platform, allowing users to communicate in ways that align with their natural inclinations.

5.2.6 GLOBAL INCLUSIVITY:

- **EXISTING SOLUTIONS:** Explore how existing platforms handle language barriers and support users from diverse cultural backgrounds.
- **ARTICULATIVE CHAT SERVER:** Emphasize the global inclusivity achieved through multi-modal communication, irrespective of language.

5.2.7 SCALABILITY AND PERFORMANCE:

- **EXISTING SOLUTIONS:** Investigate the scalability and real-time processing capabilities of existing platforms.
- **ARTICULATIVE CHAT SERVER:** Highlight the scalability features and optimizations to handle increasing user loads and ensure optimal performance.

5.2.8 LEARNING CURVE:

- **EXISTING SOLUTIONS:** Assess the learning curve associated with adopting new features on existing platforms.
- **ARTICULATIVE CHAT SERVER:** Address the learning curve challenges and showcase strategies for user onboarding and ease of adoption

5.2.9 CROSS-PLATFORM COMPATIBILITY:

- **EXISTING SOLUTIONS:** Examine how existing platforms maintain a consistent user experience across different devices and operating systems.
- **ARTICULATIVE CHAT SERVER:** Showcase the responsive design principles ensuring a seamless experience across various platforms.

5.2.10 COST AND ACCESSIBILITY:

- **EXISTING SOLUTIONS:** Analyze the cost structures and accessibility of existing platforms.
- **ARTICULATIVE CHAT SERVER:** Emphasize any cost advantages or accessibility features that set the platform apart.

CHAPTER 6: CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSION

A program called Articulative Chat Server was created in accordance with customer specifications. It took a lot of work to make it effective and flawless. The users are happy with the system's performance and reports after the built system has been tested using actual data.

To construct this project, AWT, JFC-Swing, and Java are used. We can communicate with various individuals via various channels and divisions within our company by using this program. It saves time and money for both our staff and the business. Each employee will have much less work to do in order to interact with other employees and complete tasks ahead of schedule. This tool is incredibly helpful for managing a user-friendly, effective communication system. Additionally, it offers extendibility. Thus, you are able to add modules too.

In addition to satisfying client requirements, the Articulative Chat Server goes above and above by offering a smooth and simple user interface. Utilizing Java, JFC-Swing, and AWT technologies provides scalability and resilience, enabling future module additions and improvements in response to changing business requirements. Workflows are streamlined, and productivity is increased, thanks to its capacity to support communication between departments and channels inside the company. Furthermore, extensive testing using actual data is used to evaluate the system's efficacy, guaranteeing dependability and performance across a range of usage scenarios. This all-inclusive solution promotes a productive and cooperative work atmosphere while saving time and resources, all of which eventually help the company succeed and expand.

6.2 FUTURE SCOPE

The "Articulative Chat Server" lays the foundation for a future where digital communication is not bound by the constraints of traditional messaging platforms. The project's success opens up several exciting avenues for future development and expansion:

6.2.1 INTEGRATION OF ADVANCED CHATBOTS: Explore the integration of more sophisticated chatbots powered by advanced NLP algorithms to enhance automated responses and provide a more intuitive user experience.

6.2.2 AUGMENTED REALITY (AR) INTEGRATION: Investigate the incorporation of AR to create immersive drawing spaces, allowing users to share three-dimensional drawings in real-time.

6.2.3 COLLABORATIVE SPACES: Develop features that facilitate collaborative drawing sessions, allowing multiple users to contribute simultaneously in a shared space.

6.2.4 EXTENDED EMOTIVE FEATURES: Expand the library of emotive emojis and explore gesture-based interactions for an even more dynamic and expressive communication experience.

6.2.5 LANGUAGE TRANSLATION AND LOCALIZATION: Implement real-time language translation to break down language barriers and facilitate seamless communication between users with different native languages.

6.2.6 INTERACTIVE MULTIMEDIA INTEGRATION: Introduce support for sharing rich multimedia content, such as videos and interactive media, to enhance the diversity of communication modes.

6.2.7 ACCESSIBILITY FEATURES: Focus on accessibility by incorporating features like voice commands and screen reader compatibility to ensure inclusivity for users with different abilities.

6.2.8 COMMUNITY BUILDING AND SOCIAL FEATURES: Create public drawing spaces where users can join and collaborate, fostering a sense of community and creativity.

6.2.9 USER CUSTOMIZATION: Allow users to customize their profiles, avatars, and preferences to create a more personalized and engaging experience.

6.2.10 BLOCKCHAIN INTEGRATION FOR SECURITY: Explore the integration of blockchain technology to enhance security measures, ensuring decentralized and tamper-proof data protection.

REFERENCES

1. Singh, S., Singh, S. and Sharma, A., Real-Time Web-Based Secure Chat Application using Django , International Journal of Advances in Engineering and Management ,Volume 5 , April 2023
2. Thosani, P., Sinkar, M., Vaghasiya, J. and Shankarmani, R., 2020, May. A self learning chat-bot from user interactions and preferences. In 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 224-229). IEEE
3. A. Renner, "Encrypted Peer-To-Peer Chat Application in Java," in Proceedings of the IEEE International Conference on Computer Science and Engineering, Shepherdstown, WV, 2023Cuypers, C., Jacobs, B. and Piessens, F.,. Verification of data-race-freedom of a java chat server with verifast. CW reports ,2022
4. M. Chen, Y. Zhou, and L. Zhang, "AI-Powered Chat Servers for Customer Service," International Journal of Advanced Research in Computer Science and Engineering (IJARCSE), vol. 10, pp. 1-10, 2023.
5. Thosani, P., Sinkar, M., Vaghasiya, J. and Shankarmani, R., 2020, May. A self learning chat-bot from user interactions and preferences. In 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 224-229). IEEE..
6. P. Srilakshmi, D. P. Venkat Sai, T. Ganesh, T. Tarun, "BROADCASTING CHAT SERVER," International Journal of Research and Analytical Reviews , VOL. 7, March 2020
7. Singh, K., Gupta, H. and Singhal, P.. "Development of LAN chat server". International Research Journal of Engineering and Technology (IRJET),2020
8. P. Jouet, A. Laurent, T. Luo, V. Alleaume, M. Fradet, and C. Baillard, "AR-Chat: an AR-based instant messaging system," in 2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), 2020
9. S. Srivastava, N. Sharma, and M. Salim, "Chat Room Android Application," International Research Journal of Modernization in Engineering Technology and Science, vol. 03, May 2019

10. Ashutosh Kumar , Atul Singh "Group chatting Application". *International Journal of Engineering and Advanced Technology (IJEAT)*, 372-374 , June 2019
11. Thakur A, Dhiman K. Chat Room Using HTML, PHP, CSS, JS, AJAX. arXiv preprint arXiv:2106.14704. 2021 Jun 28,2019
12. R. Gayathri and C. Kalieswari, "Multi-User Chat Application," in International Journal of Engineering and Advanced Technology (IJEAT), vol. 9 , June 2019
13. “Chat Server” computerhope.com https://www.computerhope.com/jargon/c/chatserv .htm Aug. 1 , 2023
14. ”Development of Chat Application” ijraset.com https://www.ijraset.com/research-paper/development-of-chat-application Aug. 10 , 2023
15. “Java Tutorial” w3schools.com https://www.w3schools.com/java/ Oct. 10, 2023
16. “ Java“ oracle.com https://www.oracle.com/java/technologies/ Oct. 15, 2023
17. “Spring Tutorial ” javatpoint.com https://www.javatpoint.com/spring-tutorial Oct. 19,2023
18. “Spring” youtube.com https://www.youtube.com/watch?v=zvR-Oif_nxg&ab_channel=DailyCodeBuffer Oct. 20 ,2023
19. “Implementing Chat Server” wiki.haskell.org https://wiki.haskell.org/Implement_a_chat_server Oct. 21 ,2023
20. “Multi Client Chat Server” gyawaliamit.medium.com https://gyawaliamit.medium.com/multi-client-chat-server-using-sockets-and-threads-in-java-2d0b64cad4a7 Nov. 5, 2023
21. “Benefits of real time chat application ” pubnub.com https://www.pubnub.com/guides/the-long-term-benefits-of-building-a-chat-application/ Nov. 8,2023
22. R. Smith, "Design and Implementation of an Articulative Chat Server Using Natural Language Processing Techniques," in IEEE Transactions on Computational Linguistics and Natural Language Processing,Sep. 2022
23. S. Jones et al., "Enhancing User Experience in Articulative Chat Servers Through Machine Learning-Based Response Generation," in IEEE International Conference on Natural Language Processing and Computational Linguistics (NLPCL), New York, NY, USA, 2023

24. A. Patel and B. Gupta, "Secure Communication Framework for Articulative Chat Servers," in IEEE Transactions on Information Forensics and Security , Dec. 2023
25. M. Chen et al., "Efficient Resource Management in Articulative Chat Servers: A QoS Perspective," in IEEE Transactions on Services Computing , March 2024

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ __Department: _____ Enrolment No _____

Contact No. _____E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><td>Copy Received on</td><td>Excluded</td><td>Similarity Index (%)</td><td colspan="2">Generated Plagiarism Report Details (Title, Abstract & Chapters)</td></tr>
<tr><td></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td>Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

APPENDIX

Bhanu MJR

ORIGINALITY REPORT

6%	6%	1%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dokumen.tips Internet Source	3%
2	freeproject.co.in Internet Source	1%
3	www.ir.juit.ac.in:8080 Internet Source	<1%
4	opus.govst.edu Internet Source	<1%
5	www.coursehero.com Internet Source	<1%
6	en.wikipedia.org Internet Source	<1%
7	ghorerhaat.esy.es Internet Source	<1%
8	docshare.tips Internet Source	<1%
9	mafiadoc.com Internet Source	<1%