M.Kumarasamy College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

A Project Report

on

CHAT APPLICATION

Submitted in partial fulfillment of requirements for the award of the course

of

EGB1201 – JAVA PROGRAMMING

Under the guidance of

Dr.R.BHARATHI M.E.,PhD.,

Associate Professor / IT

Submitted By

SIBIDHARAN R (927624BEE091)

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

M.KUMARASAMY COLLEGE OF ENGINEERING
(Autonomous)

KARUR – 639 113

DECEMBER 2025

M. KUMARASAMY COLLEGE OF ENGINEERING

(Autonomous Institution affiliated to Anna University, Chennai)

KARUR – 639 113

BONAFIDE CERTIFICATE

Certified that this project report on **"CHAT APPLICATION"** is the bonafide work of **SIBIDHARAN R (927624BEE091)** who carried out the project work during the academic year 2025 - 2026 under my supervision.

Signature

Dr. R. BHARATHI M.E.,PhD.,

SUPERVISOR,

Department of Information Technology

M.Kumarasamy College of Engineering,
Thalavapalayam, Karur -639 113.

Signature

Dr. J. UMA M.E.,Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Electrical and Electronics Engineering,

M.Kumarasamy College of Engineering,
Thalavapalayam, Karur -639 113.

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

VISION OF THE INSTITUTION

To produce smart and dynamic with profound professionals theoretical and practical knowledge comparable with the best in the field.

MISSION OF THE INSTITUTION

M1: Produce Hi-Tech professionals in the field of Electrical and Electronics Engineering by inculcating knowledge. core

M2: Produce highly competent professionals with thrust on research.

M3: Provide personalized training to the students for enriching their kills.

VISION OF THE DEPARTMENT

To create competent Electrical and Electronics Engineers who will work in Emerging technologies thereby contributing value to their career and society.

MISSION OF THE DEPARTMENT

M1: Impart quality education to enhance knowledge and skills in emerging technologies.

M2: Create an innovation and product development ecosystem in core and Interdisciplinary areas of Industrial applications leading to patents.

M3: IAugment student core competency and soft skills in handling technical projects.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Graduates will have flourishing career in the core areas of Electrical Engineering and allied disciplines.

PEO2: Graduates will pursue higher studies and succeed in academic/research careers.

PEO3: Graduates will be a successful entrepreneur in creating jobs related to Electrical and Electronics Engineering/allied disciplines.

PEO4: Graduates will practice ethics and have habit of continuous learning for their success in the chosen career.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems.

PO6: The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment.

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws.

PO8: Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences.

PO10: Project management and finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-long learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Apply the basic concepts of mathematics and science to analyse and design circuits, controls, Electrical machines and drives to solve complex problems.

PSO2: Apply relevant models, resources and emerging tools and techniques to provide solutions to power and energy related issues & challenges.

PSO3: Design, Develop and implement methods and concepts to facilitate solutions for electrical and electronics engineering related real world problems.

ABSTRACT

This project presents a simple Java-based web application that demonstrates user authentication using JSP and Servlets. The system enables users to log in through a web interface where their credentials are validated on the server side. Upon successful authentication, users are redirected to a controlled chat interface, ensuring secure access to the application. Although static credentials are used in this version, the system architecture is fully scalable for integrating databases and advanced authentication mechanisms. The project highlights key elements of Java web development, including request handling, server-side validation, session management, and structured page navigation. These fundamentals form a strong foundation for building full-featured communication systems such as real-time chat applications. Additionally, the system is designed with modularity in mind, allowing future extensions without disrupting existing functionality. Features like real-time messaging through WebSockets, database-backed login authentication, user profile management, and message storage can be seamlessly incorporated. The clean interface and simplified workflow make the application suitable for beginners learning JSP–Servlet concepts, while its scalable design supports more advanced development. Overall, this project serves not only as a demonstration of core Java web technologies but also as a flexible framework for developing modern, interactive, and secure communication platforms.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	COs MAPPED	POs MAPPED	PSOs MAPPED
This project presents a simple Java-based web application that demonstrates user authentication using JSP and Servlets. The system allows users to log in through a web interface, where the credentials are validated on the server side using a servlet. Upon successful authentication, the user is redirected to a chat interface page, ensuring controlled access to the application. Although the login validation uses static credentials in this version, the structure is fully scalable for integration with databases and advanced authentication mechanisms. This project highlights the essential components of Java web development, including request handling, server-side validation, session flow, and page navigation, making it a strong foundation for building full-featured web applications such as real-time chat systems.	CO1 CO2 CO3 CO4 CO5	PO1 PO2 PO3 PO4 PO5 PO6 PO7 PO8 PO9 PO10 PO11 PO12	PSO1 PSO2

SUPERVISOR **HEAD OF THE DEPARTMENT**

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE

The main objective of this project is to design and implement a simple yet functional Java web application that demonstrates secure user authentication using JSP and Servlets. The system aims to validate user credentials on the server side and provide controlled access to the chat interface only for authenticated users. Additionally, the project intends to establish a foundational structure for future expansion into a complete chat system by ensuring proper navigation, request handling, and modular code organization within a Java-based web environment.

1.2 OVERVIEW

The Chat Application project is developed using Java, JSP, and Servlets to demonstrate a secure and structured web-based communication system. The application begins with a login interface where user credentials are validated on the server side, ensuring controlled access to the chat environment. Once authenticated, users can view contacts, open chat sessions, create groups, and send messages through an intuitive interface. The system follows a modular design, separating core functionalities such as authentication, chat handling, contact management, and voice-to-text conversion. This modularity enhances maintainability, scalability, and future extendability. The architecture supports smooth navigation, efficient request processing, and a user-friendly experience.Overall, the project serves as a foundational model for real-time communication platforms, providing essential features while allowing room for advanced enhancements like database integration, encryption, and real-time message updates.

1.3 JAVA PROGRAMMING CONCEPT

1.Object-Oriented Programming(OOP):

The Chat Application uses OOP concepts to make the system structured and efficient. Classes and objects represent users, messages, and chat rooms. Encapsulation secures user data, while inheritance allows reusing features like admin controls. Polymorphism lets different message types use the same methods, and abstraction hides complex processes like data transfer. These concepts together make the application secure, flexible, and easy to maintain.

2.Framework

The project is built using the JSP–Servlet framework, which provides a structured way to handle web requests and responses. Servlets manage backend processing, while JSP handles user interface rendering. This framework supports scalable web development by separating logic from presentation.

3. Control Structures

Control structures such as if–else, switch, loops (for, while), and try–catch are used throughout the project for input validation, event handling, message flow, and error management. These structures help manage program decisions and ensure smooth execution.

4. Constructors

Constructors are used to initialize objects such as Contact, Group, and Message in the chat application. They assign initial values to variables and prepare the object for use, ensuring that each module works with properly structured data.

5. User Input and Output

User interactions are handled through form inputs, button clicks, and text fields. Input is collected using JTextField, JOptionPane, and JSP form elements, while output is displayed through JTextArea, labels, and web responses. This enables dynamic communication between the user and the system.

6. Java Swing

Java Swing is used to design the application's graphical user interface. Components like JFrame, JPanel, JButton, JTextArea, JList, and JScrollPane help build windows, chat screens, contact lists, and dialog boxes. Swing provides an interactive and flexible desktop-style UI for the chat application

CHAPTER 2

PROJECT METHODOLOGY

2.1 PROPOSED WORK

1. Login page:

- Create a secure login and signup page for user authentication, ensuring that only registered users can access the chat system.
- Implement password encryption to securely store and validate user credentials. Add a "Forgot Password" feature to help users recover or reset their account securely.

2. Contact list:

- Develop a dynamic contact list interface where users can view available contacts, search for friends, and select a person to start chatting with.
- Allow users to categorize or group contacts (e.g., friends, work, family) for easier navigation. Include an online/offline status indicator to show which contacts are available for chatting.

3. Chat area:

- Design a main chat window to display real-time messages exchanged between users with a clean and interactive layout.
- Add message status indicators such as sent, delivered, and seen for better communication feedback. Provide message search functionality so users can quickly find specific chats or keywords.

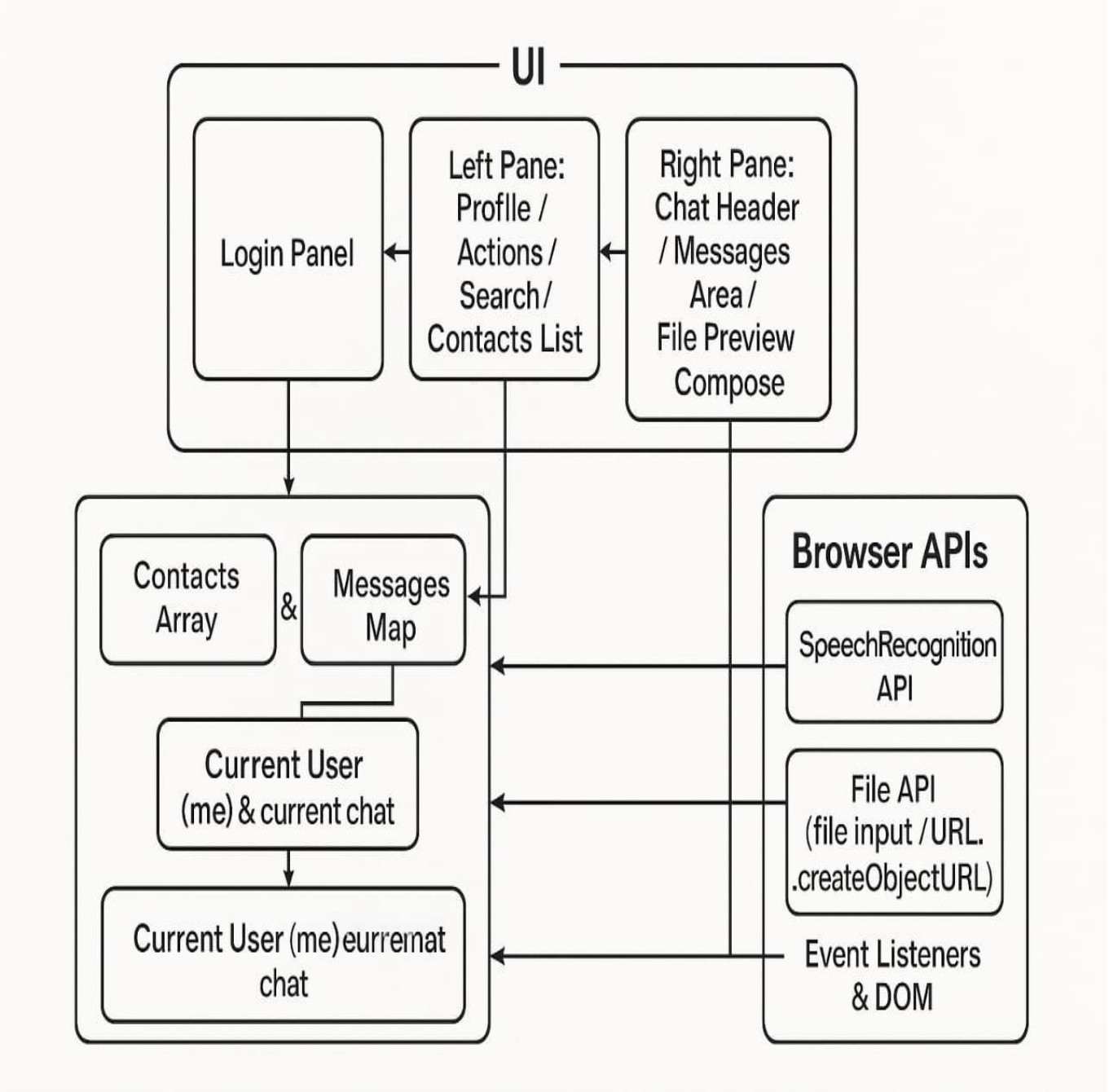
4. Text area:

- Provide a text input field for users to type and send messages instantly, supporting emoji and text formatting features.
- Include auto-suggestion and auto-correction features to improve typing speed and accuracy. Support file attachments such as images, documents, and videos for richer communication.

5. Voice to text converter:

- Integrate a voice recognition feature that converts spoken words into text,allowing hands-free chatting for enhanced user convenience.
- Add support for multiple languages to allow users to dictate messages in their preferred language. Implement noise-cancellation to improve accuracy in noisy environments.

2.2 BLOCK DIAGRAM



CHAPTER-3

MODULE DISCRIPTION

3.1 Login Module

The Login Module is the entry point of the application and ensures that only authenticated users can access the chat system. It provides a graphical interface with input fields for entering the username and optional phone number. When the user submits the form, the system validates the inputs to ensure they are not empty and then processes them through Servlet logic or internal validation methods. If the login is successful, the user session is initialized and stored, enabling personalized access to the chat features. In case of invalid inputs, meaningful error messages are displayed to guide the user. This module plays a crucial role in maintaining security, controlling user access, and preparing the session environment for subsequent modules.

3.2 Chat Area Module

The Chat Area Module represents the main communication hub where real-time interactions occur between users. It contains a message display section where both incoming and outgoing messages are shown with timestamps for better clarity. When a user selects a contact or group, the module retrieves and displays previous chat history to maintain conversation continuity. A message input box, combined with a send button, allows users to type and transmit messages instantly. The module uses Swing components and internal data structures to update the chat interface dynamically. This module ensures an organized, readable, and interactive chat environment where users can communicate smoothly without delays or interruptions.

3.3 Contact List Module

The Contact List Module manages and displays the user's collection of contacts on the left pane of the interface. Contacts are shown with their names and optional phone numbers, making it easy for the user to identify chat partners. The module includes search functionality, allowing users to filter contacts by name or number, improving navigation efficiency. When a contact is clicked, the application immediately loads the associated chat history and switches the view to the Chat Area Module. This module greatly enhances user convenience by providing quick access to all communication endpoints in the application. It acts as the central navigation system for moving between different chats.

3.4 Add Contact Module

The Add Contact Module allows users to dynamically expand their contact list. Through a popup dialog box, users can enter the name and phone number of a new contact. The system performs input validation to ensure that necessary details are provided before creating the contact. Once validated, the new contact is added to the internal data structure and automatically updated in the Contact List Module. This feature provides flexibility to users, enabling them to grow their contact network without altering the source code. It enhances personalization and ensures the application remains scalable as the number of users increases.

3.5 Group Creation Module

The Group Creation Module supports the formation of chat groups for multi-user communication. It provides an interface for entering a group name and selecting members from the existing contact list using checkboxes. After the user confirms the selected members, the system generates a new group object and adds it to the contact/group list. The module validates essential criteria such as ensuring a group name is entered and at least one member is selected. This feature expands the application beyond one-to-one communication, enabling collaborative messaging and enhancing the scope of interaction. It is a crucial step toward implementing more advanced features like group notifications and shared media.

3.6 Voice-to-Text Conversion Module

The Voice-to-Text Module adds modern accessibility functionality to the application by allowing users to input messages using voice instead of typing. The system simulates speech recognition through a text input dialog, which represents the transcribed speech. Once the user provides the simulated voice input, the text is automatically inserted into the message input field, ready to be edited or sent. This module aims to improve usability for users who prefer voice communication or face difficulty typing. It also demonstrates the potential for integrating real speech recognition technologies in future versions, enhancing convenience and aligning the application with modern communication standards.

CHAPTER 4

RESULTS AND DISCUSSION

FIG: 4.1 - LOGIN PAGE:

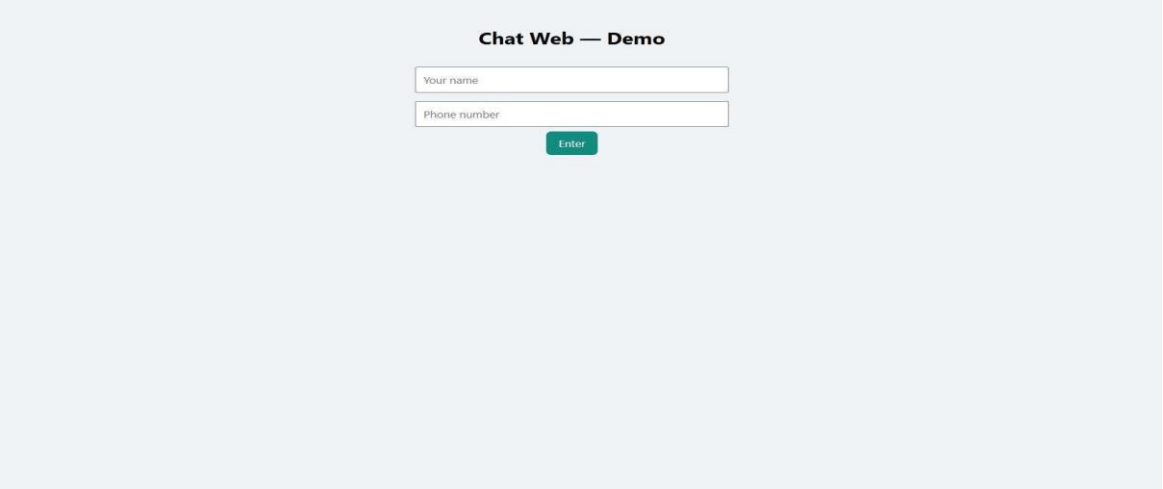


FIG:4.2 - CHAT PAGE:

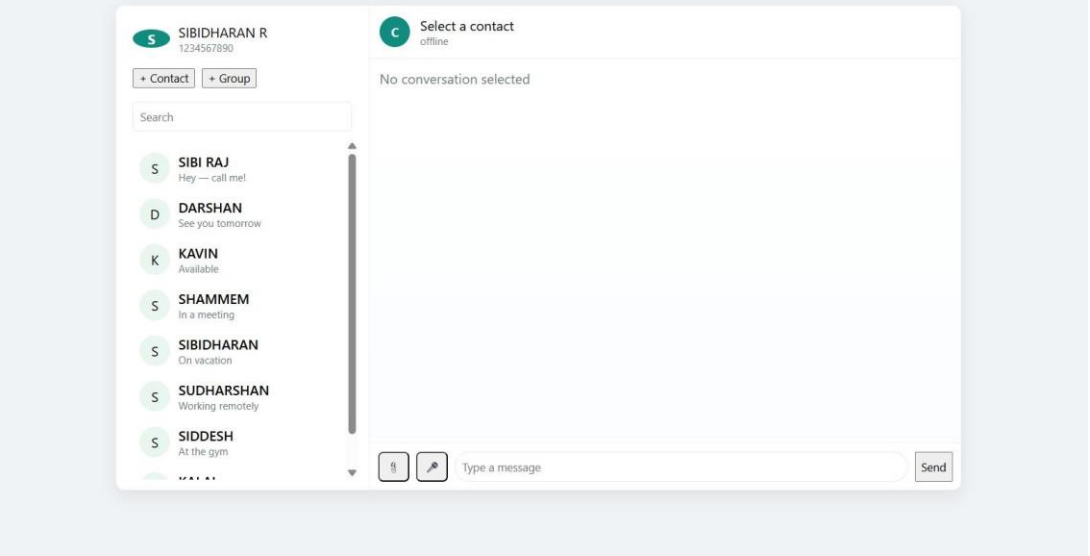
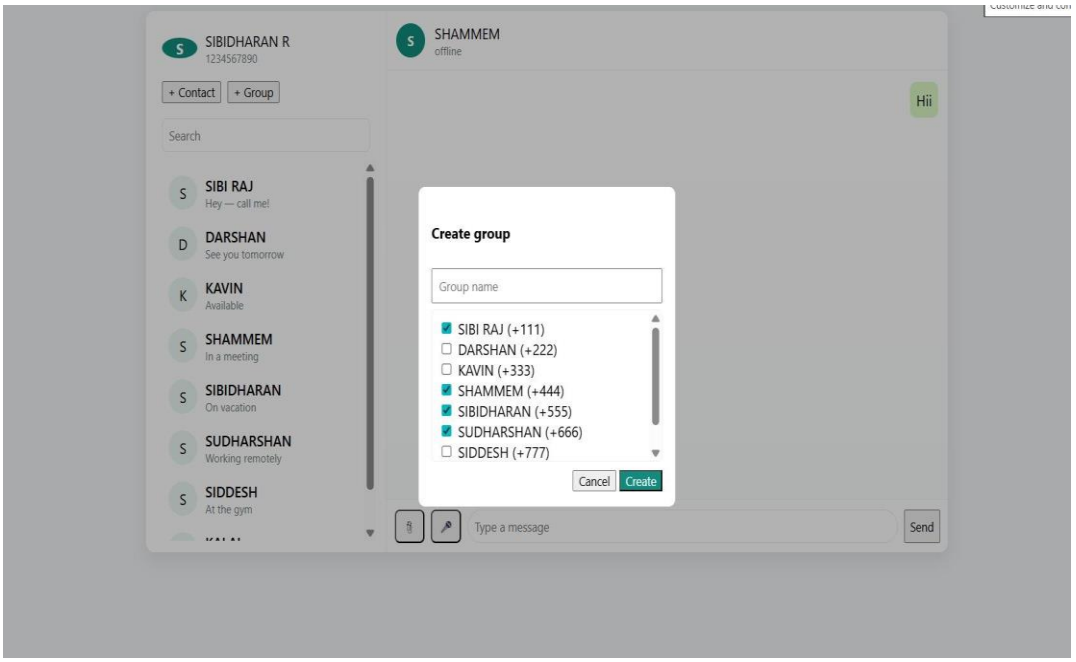


FIG:4.3 - GROUP CREATION:



RESULT:

The developed chat application successfully implements all the proposed modules, including secure login, dynamic contact list management, real-time messaging, text input features, voice-to-text conversion, and multithreaded communication. Users are able to log in, view contacts, chat smoothly, and send or receive messages without delay. The system performs reliably during multi-user communication and maintains stable performance across different scenarios.

DISCUSSION:

The results indicate that the chat system meets the functional requirements and works efficiently in real-time communication environments. The Login Module ensures that only authenticated users access the system, increasing security. The Contact List dynamically updates and allows users to quickly search and select friends, improving usability. The Chat Area provides a clean interface where messages appear instantly, enhancing user experience. The Text Area supports fast message input with emojis and formatting, making conversations expressive and interactive. The Voice-to-Text Converter improves accessibility by helping users send messages hands-free, especially useful for multitasking. With Multithreading, the server handles multiple user connections simultaneously without lag, proving the design's scalability and efficiency. Overall, the system demonstrates smooth, real-time performance and provides a user-friendly platform for communication. The project can be further expanded by integrating a database, end-to-end encryption, and advanced messaging features.

CHAPTER 5

CONCLUSION

The Chat Application project successfully demonstrates the implementation of a functional and interactive communication system using Java technologies such as JSP, Servlets, and Java Swing. Throughout the development process, essential programming concepts including object-oriented design, multithreading, event handling, and user interface construction were effectively applied to build a modular and scalable application. The system achieves its primary objective by providing secure login authentication, an intuitive chat interface, organized contact management, and additional features such as group creation and voice-to-text simulation. Moreover, the project highlights the importance of structured software design by separating business logic, user interface components, and data models. This approach ensures easier debugging, maintainability, and future expansion. The incorporation of features like dynamic contact addition, chat history display, and message formatting enhances the usability of the system and reflects real-world communication functionalities. The simulated speech-to-text feature further demonstrates the potential for integrating modern, AI-driven technologies into traditional Java applications. Overall, the project not only showcases the technical capabilities of Java for building chat-based applications but also provides a strong foundation for further enhancements. Future improvements may include integrating real-time messaging using Web Sockets, adding database support for persistent user data, implementing encryption for secure communication, and deploying the system on a cloud platform for broader accessibility. This project thus serves as a practical and comprehensive learning experience, successfully fulfilling its objectives and offering scope for transformation into a fully featured real-time chat system. In addition, this project has strengthened the understanding of core Java concepts and enhanced practical skills in designing interactive applications. The modular structure ensures that the system can be upgraded easily as new technologies and requirements emerge. By combining simplicity with scalability, the application stands as a valuable prototype for future communication platforms. Thus, the work carried out in this project contributes both educationally and technically toward building robust Java-based software solutions.

REFERENCES

1) Herbert Schildt &Dr. Danny Coward (2021). Java: The Complete Reference, 13th Edition. McGraw-Hill Education.

2) Ahmed, T., Rahman, M., & Chowdhury, A. (2022). Design of an IoT Framework for Smart Crop Growth Analysis. International Journal of Agricultural Technology, 18(3), 599–612.

3) "Design and Implementation of a Real-Time Chat Application Using Java Sockets" Explains architecture: Server-Client model, broadcasting messages.Useful for engineering projects.

4) "Java Network Programming" – Elliotte Rusty Harold.Best reference for TCP, UDP, sockets, multithreading.Practical examples for chat servers.

5) Java Networking (Socket Programming)*Oracle Java Tutorials – Custom Networking.*Explains ServerSocket, Socket, Input/Output streams
Topic: Core for building chat server & client.

6) Java Concurrency,*Oracle Java Tutorials – Concurrency in Java*
Helps create threads to handle multiple clients.

7) "Java Network Programming" – Elliotte Rusty HaroldBest reference for TCP, UDP, sockets, multithreading.Practical examples for chat servers.

APPENDIX

(Coding)

```
import javax.swing.*;
import javax.swing.border.EmptyBorder; import
javax.swing.event.*;
import java.awt.*; import
java.awt.event.*;
import java.text.SimpleDateFormat; import
java.util.*;
import java.util.List;

public class ChatApplication extends JFrame
    { static class Contact { final
    int id;
    String name; String
    phone;

    Contact(int id, String name, String phone)
      { this.id = id; this.name =
      name; this.phone = phone;
    }
    String display() { return name + (phone != null && !phone.isEmpty() ? " (" + phone + ")" :
"""); }
    String key() { return "C:" + id; }
```

```
    @Override
    public String toString() { return display(); }
}
      static class Group
{ final int id; String
name;
   List<Contact> members;
    Group(int id, String name, List<Contact> members)
        {    this.id    =    id;
        this.name = name;
        this.members = new ArrayList<>(members);
    }
    String key() { return "G:" + id; }
    @Override
    public String toString() { return name + " [Group]"; }
}
static class Message
    { final boolean fromMe; final
    String text;
    final Date time;
    Message(boolean fromMe, String text, Date time)
```

```
        this.fromMe = fromMe;
        this.text = text; this.time =
        time;
    }
    String formatted() {
        SimpleDateFormat sdf = new SimpleDateFormat("HH:mm");
        return (fromMe ? "Me" : "Them") + " (" + sdf.format(time) + "): " + text;
    }
}
private final java.util.List<Contact> contacts = new ArrayList<>(); private final
java.util.List<Group> groups = new ArrayList<>();
private final Map<String, java.util.List<Message>> history = new HashMap<>();

private final CardLayout rootCard = new CardLayout(); private
final JPanel root = new JPanel(rootCard); private JTextField
loginNameField;
private JTextField loginPhoneField;
private DefaultListModel<Object> leftListModel = new DefaultListModel<>(); private
JList<Object> leftList;
private JTextArea chatArea; private
JTextField inputField; private JLabel
loggedInLabel; private JTextField
searchField;
```

```
private Object currentSelected = null;
private int nextContactId = 1; private
int nextGroupId = 1;
public ChatApplication()
    { setTitle("Chat Application - Demo");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); setSize(980,
    640);
    setLocationRelativeTo(null);

    initLoginPanel();
    initMainPanel();
    seedDemoData();
    add(root);      rootCard.show(root,
    "login");
}

   private void initLoginPanel() {
    JPanel loginPanel = new JPanel(new GridBagLayout());
    loginPanel.setBorder(new EmptyBorder(24, 24, 24, 24));
    GridBagConstraints gbc = new GridBagConstraints(); gbc.insets =
    new Insets(8, 8, 8, 8)
```

```
gbc.fill = GridBagConstraints.HORIZONTAL; gbc.gridx = 0;
gbc.gridy = 0;
JLabel title = new JLabel("Chat Web — Demo", SwingConstants.CENTER); title.setFont(new
Font("Segoe UI", Font.BOLD, 22));
gbc.gridwidth = 2; loginPanel.add(title, gbc);

gbc.gridwidth = 1; gbc.gridy++;
loginPanel.add(new JLabel("Name:"), gbc);
gbc.gridx = 1;
loginNameField = new JTextField();
loginPanel.add(loginNameField, gbc);
gbc.gridx = 0; gbc.gridy++;
loginPanel.add(new JLabel("Phone (optional):"), gbc); gbc.gridx =
1;
loginPhoneField = new JTextField();
loginPanel.add(loginPhoneField, gbc);
gbc.gridx = 0; gbc.gridy++;
gbc.gridwidth = 2;
JButton enterBtn = new JButton("Enter");
enterBtn.addActionListener(e -> doLogin());
loginPanel.add(enterBtn, gbc);
root.add(loginPanel, "login");
};
```

```
private void initMainPanel() {
    JPanel leftPanel = new JPanel(new BorderLayout()); leftPanel.setPreferredSize(new
    Dimension(300, 0));
    leftPanel.setBorder(BorderFactory.createMatteBorder(0,0,0,  1,new
Color(0xE6ECEF)));

    JPanel profile = new JPanel(new BorderLayout(8, 8));
    profile.setBorder(new EmptyBorder(8, 8, 8, 8)); loggedInLabel =
    new JLabel("Not logged in"); profile.add(loggedInLabel,
    BorderLayout.CENTER);
    JPanel topActions = new JPanel(new FlowLayout(FlowLayout.RIGHT, 6, 0)); JButton
    addContactBtn = new JButton("+ Contact");
    JButton createGroupBtn = new JButton("+ Group");
    topActions.add(addContactBtn);
    topActions.add(createGroupBtn);  profile.add(topActions,
    BorderLayout.SOUTH);
    leftPanel.add(profile, BorderLayout.NORTH);

    JPanel searchBox = new JPanel(new BorderLayout());
    searchBox.setBorder(new EmptyBorder(6, 8, 6, 8)); searchField =
    new JTextField(); searchField.setToolTipText("Search contacts or
    groups"); searchBox.add(searchField, BorderLayout.CENTER);
    leftPanel.add(searchBox, BorderLayout.CENTER);
```

```
leftListModel = new DefaultListModel<>();
leftList = new JList<>(leftListModel);
leftList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
leftList.setCellRenderer(new DefaultListCellRenderer() {
    public Component getListCellRendererComponent(JList<?> list, Object value, int index, boolean isSelected, boolean cellHasFocus) {
        Component c = super.getListCellRendererComponent(list, value, index, isSelected, cellHasFocus);
        if(valueinstanceof Contact) setIcon(UIManager.getIcon("FileView.fileIcon"));
        else setIcon(UIManager.getIcon("FileView.directoryIcon")); return c;
    }
});
JScrollPane leftScroll = new JScrollPane(leftList); leftScroll.setBorder(null);
leftPanel.add(leftScroll, BorderLayout.SOUTH);

JPanel rightPanel = new JPanel(new BorderLayout());
JPanel chatHeader = new JPanel(new BorderLayout()); chatHeader.setBorder(new
EmptyBorder(8, 8, 8, 8));
chatHeader.add(newJLabel("Select a contact or group"), BorderLayout.CENTER);
```

```
rightPanel.add(chatHeader, BorderLayout.NORTH);
chatArea = new JTextArea();
chatArea.setEditable(false);
chatArea.setLineWrap(true);
chatArea.setWrapStyleWord(true);
JScrollPane chatScroll = new JScrollPane(chatArea);
chatScroll.setBorder(null); rightPanel.add(chatScroll,
BorderLayout.CENTER);

JPanel composePanel = new JPanel(new BorderLayout(8, 8));
composePanel.setBorder(new EmptyBorder(8, 8, 8, 8)); JButton
attachBtn = new JButton("  ");
composePanel.add(attachBtn, BorderLayout.WEST);
JPanel inputRow = new JPanel(new BorderLayout(8, 8));
inputField = new JTextField();
inputRow.add(inputField, BorderLayout.CENTER);

JPanel rightButtons = new JPanel(new FlowLayout(FlowLayout.RIGHT, 6, 0)); JButton
micBtn = new JButton("  ");
JButton sendBtn = new JButton("Send");
rightButtons.add(micBtn); rightButtons.add(sendBtn);
inputRow.add(rightButtons, BorderLayout.EAST);
```

```
composePanel.add(inputRow, BorderLayout.CENTER); rightPanel.add(composePanel,
BorderLayout.SOUTH);
addContactBtn.addActionListener(e -> showAddContactDialog());
createGroupBtn.addActionListener(e -> showCreateGroupDialog());
sendBtn.addActionListener(e -> sendMessage()); inputField.addActionListener(e ->
sendMessage()); leftList.addListSelectionListener(e->{if(!e.getValueIsAdjusting())
openSelectedChat(leftList.getSelectedValue()); });
searchField.getDocument().addDocumentListener(new DocumentListener() {
   public void insertUpdate(DocumentEvent e) { refreshLeftList(); } public void
   removeUpdate(DocumentEvent e) { refreshLeftList(); } public void
   changedUpdate(DocumentEvent e) { refreshLeftList(); }
});
micBtn.addActionListener(e -> startVoiceToTextSimulator());
  JSplitPane split = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, leftPanel, rightPanel);
split.setDividerLocation(300);
split.setResizeWeight(0); root.add(split,
"chat");
root.add(split, "chatPanel"); // alias - we'll call show("chat") to display
root.add(split, "main");
```

```
        root.add(split, "chat");
        root.add(root, "unused"); // avoid unused warning
    }
    private void doLogin() {
        String name = loginNameField.getText().trim(); String
        phone = loginPhoneField.getText().trim(); if
        (name.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please enter your name.", "Validation",
JOptionPane.WARNING_MESSAGE);
            return;
        }
        loggedInLabel.setText("Logged in as: " + name + (phone.isEmpty() ? "" : " (" + phone + ")"));
        rootCard.show(root, "chat");
        refreshLeftList();
    }
    private void openSelectedChat(Object obj)
        { currentSelected = obj;
        chatArea.setText("");
        if (obj == null) return;
        if (obj instanceof Contact)
            { Contact c = (Contact) obj;
            List<Message> msgs = history.getOrDefault(c.key(), new ArrayList<>()); for (Message
            m : msgs) chatArea.append(m.formatted() + "\n");
```

```
} else if (obj instanceof Group)
{ Group g = (Group) obj;
                List<Message> msgs = history.getOrDefault(g.key(), new ArrayList<>()); for (Message
                m : msgs) chatArea.append(m.formatted() + "\n");
            }
        }
        private void sendMessage() {
            String txt = inputField.getText().trim();
            if (txt.isEmpty() || currentSelected == null) {
                if (currentSelected == null) JOptionPane.showMessageDialog(this, "Select a contact or
    group first.");
                return;
            }
            Message m = new Message(true, txt, new Date()); String key =
            keyForObject(currentSelected); history.putIfAbsent(key, new
            ArrayList<>()); history.get(key).add(m);
            chatArea.append(m.formatted() + "\n"); inputField.setText("");
        }

        private String keyForObject(Object o) {
            if (o instanceof Contact) return ((Contact) o).key(); if (o
            instanceof Group) return ((Group) o).key(); return
            o.toString();
```

```
    }
    private void showAddContactDialog()
        { JTextField name = new JTextField();
        JTextField phone = new JTextField();
        Object[] form = {"Name:", name, "Phone:", phone};
        int res = JOptionPane.showConfirmDialog(this, form, "Add Contact",
JOptionPane.OK_CANCEL_OPTION);
        if (res == JOptionPane.OK_OPTION)
            { String n = name.getText().trim(); String
            p = phone.getText().trim(); if
            (n.isEmpty()) {
                JOptionPane.showMessageDialog(this, "Name is required."); return;
            }
            Contact c = new Contact(nextContactId++, n, p); contacts.add(c);
            refreshLeftList();
        }
    }
    private void showCreateGroupDialog()
        { JTextField gname = new JTextField();
        JPanel memberPanel = new JPanel();
        memberPanel.setLayout(new BoxLayout(memberPanel, BoxLayout.Y_AXIS));
        java.util.List<JCheckBox> boxes = new ArrayList<>();
        for (Contact c : contacts) {
            JCheckBox cb = new JCheckBox(c.display());

            memberPanel.add(cb);
         boxes.add(cb);
        }
        JScrollPane scroll = new JScrollPane(memberPanel); scroll.setPreferredSize(new
        Dimension(350, 200));
        Object[] form = {"Group name:", gname, "Members:", scroll};
        int res = JOptionPane.showConfirmDialog(this, form, "Create Group",
```

```
        JOptionPane.OK_CANCEL_OPTION);
        if (res == JOptionPane.OK_OPTION)
            { String name = gname.getText().trim();
            if (name.isEmpty()) { JOptionPane.showMessageDialog(this, "Group name required."); return; }
            List<Contact> selected = new ArrayList<>();
            for (int i = 0; i < boxes.size(); i++) if (boxes.get(i).isSelected()) selected.add(contacts.get(i));
            if (selected.isEmpty()) { JOptionPane.showMessageDialog(this, "Select at least one member."); return; }
            Group g = new Group(nextGroupId++, name, selected); groups.add(g);
            refreshLeftList();
        }
    }
    private void refreshLeftList()
        { leftListModel.clear();
        String q = searchField.getText().trim().toLowerCase();
        for (Contact c : contacts) {
            if (q.isEmpty() || c.name.toLowerCase().contains(q) || (c.phone != null && c.phone.contains(q))) {
```

```
                leftListModel.addElement(c);
            }
        }
        for (Group g : groups) {
            If(q.isEmpty()||g.name.toLowerCase().contains(q)) leftListModel.addElement(g);
        }
    }
    private void startVoiceToTextSimulator() {
                String sample = JOptionPane.showInputDialog(this,
                "Voice-to-Text Simulator\n\nType what the recognizer would return (this simulates
STT):",
                "Simulated recognized text"); if
        (sample != null) {
            // append recognized text to input field String
            cur = inputField.getText();
            if (!cur.isEmpty()) inputField.setText(cur + " " + sample); else
            inputField.setText(sample);
        }
    }
    private void seedDemoData() {
        contacts.add(newContact(nextContactId++,"SIBIDHARAN","+111"));
        contacts.add(newContact(nextContactId++,"SHAMMEM","+222"));
        contacts.add(new Contact(nextContactId++,"SIBIRAJ","+333"));
        contacts.add(new Contact(nextContactId++,"DARSHAN","+444"));
        contacts.add(new Contact(nextContactId++,"KAVIN",”+555"));
        contacts.add(new Contact(nextContactId++, "SIDDESH", "+666"));
```

```
        contacts.add(newContact(nextContactId++,"KALAI","+777"));
        contacts.add(new Contact(nextContactId++, "SUDHARSHAN", "+888"));
        c = contacts.get(0);
        history.putIfAbsent(c.key(), new ArrayList<>());
        history.get(c.key()).add(newMessage(false,"Hey—callme!",new
Date(System.currentTimeMillis()- 1000 * 60 * 60)));
        history.get(c.key()).add(newMessage(true,"Sure,I'llcall after 5.",new
Date(System.currentTimeMillis() -1000 * 60 * 50)));
        refreshLeftList();
    }
    public static void main(String[] args)
        { SwingUtilities.invokeLater(() -> { ChatApplication app
            = new ChatApplication(); app.setVisible(true);
        });
    }
}
```