package ChatApplication;

import java.awt.\*;

import java.awt.event.\*;

import java.util.\*;

public class ChatApplicationGUI extends Frame implements ActionListener {

private static final long serialVersionUID = 1L;

// Data Structures

private static final HashMap<String, User> users = new HashMap<>();

private final HashMap<String, Set<String>> contacts = new HashMap<>();

private User currentUser;

private String currentChatRoomName;

private final Set<String> chatRoomUsers = new HashSet<>();

// GUI Components

private TextField usernameField, passwordField, contactField, chatRoomNameField, messageField;

private TextArea registerOutput, loginOutput, contactOutput, chatRoomOutput, messageOutput;

private Button registerButton, loginButton, addContactButton, createChatRoomButton, sendMessageButton, showContactsButton, backButton, nextButton, closeButton, setChatRoomNameButton, inviteUserButton, oneToOneChatButton, sendOneToOneMessageButton;

private Panel contactListPanel, cardPanel, oneToOneChatPanel;

private CardLayout cardLayout;

private TextField oneToOneContactField, oneToOneMessageField;

private TextArea oneToOneMessageOutput;

// Notification Module

private final NotificationModule notificationModule = new NotificationModule();

public ChatApplicationGUI() {

setTitle("Chat Application");

setSize(600, 500);

setLayout(new BorderLayout());

// Card layout for navigation

cardLayout = new CardLayout();

cardPanel = new Panel(cardLayout);

// Add panels to the card layout

cardPanel.add(createRegisterLoginPanel(), "RegisterLogin");

cardPanel.add(createAddContactPanel(), "AddContact");

cardPanel.add(createChatRoomPanel(), "CreateChatRoom");

cardPanel.add(createSendMessagePanel(), "SendMessage");

cardPanel.add(createOneToOneChatPanel(), "OneToOneChat");

add(cardPanel, BorderLayout.CENTER);

// Navigation panel

Panel navigationPanel = new Panel(new FlowLayout());

backButton = new Button("Back");

nextButton = new Button("Next");

closeButton = new Button("Close");

oneToOneChatButton = new Button("One-to-One Chat");

navigationPanel.add(backButton);

navigationPanel.add(nextButton);

navigationPanel.add(closeButton);

navigationPanel.add(oneToOneChatButton);

add(navigationPanel, BorderLayout.SOUTH);

// Event listeners

backButton.addActionListener(this);

nextButton.addActionListener(this);

closeButton.addActionListener(this);

oneToOneChatButton.addActionListener(e -> cardLayout.show(cardPanel, "OneToOneChat"));

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent we) {

dispose();

}

});

}

private Panel createRegisterLoginPanel() {

Panel panel = new Panel(new GridLayout(7, 2, 10, 10));

panel.add(new Label("Username:"));

usernameField = new TextField(20);

panel.add(usernameField);

panel.add(new Label("Password:"));

passwordField = new TextField(20);

passwordField.setEchoChar('\*');

panel.add(passwordField);

registerButton = new Button("Register");

loginButton = new Button("Login");

panel.add(registerButton);

panel.add(loginButton);

registerOutput = new TextArea(5, 40);

registerOutput.setEditable(false);

panel.add(registerOutput);

loginOutput = new TextArea(5, 40);

loginOutput.setEditable(false);

panel.add(loginOutput);

registerButton.addActionListener(this);

loginButton.addActionListener(this);

return panel;

}

private Panel createAddContactPanel() {

Panel panel = new Panel(new GridLayout(5, 1, 10, 10));

panel.add(new Label("Enter Contact Name:"));

contactField = new TextField(20);

panel.add(contactField);

addContactButton = new Button("Add Contact");

panel.add(addContactButton);

showContactsButton = new Button("Show Contacts");

panel.add(showContactsButton);

contactOutput = new TextArea(5, 40);

contactOutput.setEditable(false);

panel.add(contactOutput);

addContactButton.addActionListener(this);

showContactsButton.addActionListener(this);

return panel;

}

private Panel createChatRoomPanel() {

Panel panel = new Panel(new GridLayout(6, 1, 10, 10));

chatRoomNameField = new TextField(20);

setChatRoomNameButton = new Button("Set Chat Room Name");

inviteUserButton = new Button("Invite User");

contactListPanel = new Panel(new GridLayout(0, 1));

chatRoomOutput = new TextArea(5, 40);

chatRoomOutput.setEditable(false);

panel.add(new Label("Chat Room Name:"));

panel.add(chatRoomNameField);

panel.add(setChatRoomNameButton);

panel.add(inviteUserButton);

panel.add(contactListPanel);

panel.add(chatRoomOutput);

setChatRoomNameButton.addActionListener(e -> {

currentChatRoomName = chatRoomNameField.getText().trim();

if (currentChatRoomName.isEmpty()) {

chatRoomOutput.setText("Chat room name cannot be empty.");

} else {

chatRoomOutput.setText("Chat room created: " + currentChatRoomName);

}

});

inviteUserButton.addActionListener(e -> {

contactListPanel.removeAll(); // Clear the current contact list

if (contacts.containsKey(currentUser.getUsername())) {

Set<String> userContacts = contacts.get(currentUser.getUsername());

if (!userContacts.isEmpty()) {

for (String contact : userContacts) {

Button contactButton = new Button(contact);

contactButton.addActionListener(contactEvent -> {

chatRoomUsers.add(contact); // Add the contact to the chat room

chatRoomOutput.setText("Invited: " + contact);

});

contactListPanel.add(contactButton); // Add each contact as a button

}

contactListPanel.revalidate();

contactListPanel.repaint();

} else {

chatRoomOutput.setText("No contacts available.");

}

} else {

chatRoomOutput.setText("No contacts available.");

}

});

return panel;

}

private Panel createSendMessagePanel() {

Panel panel = new Panel(new GridLayout(4, 1, 10, 10));

panel.add(new Label("Enter Message:"));

messageField = new TextField(20);

panel.add(messageField);

sendMessageButton = new Button("Send Message");

panel.add(sendMessageButton);

messageOutput = new TextArea(5, 40);

messageOutput.setEditable(false);

panel.add(messageOutput);

sendMessageButton.addActionListener(e -> {

String message = messageField.getText().trim();

if (message.isEmpty()) {

messageOutput.setText("Message cannot be empty.");

return;

}

if (currentUser == null) {

messageOutput.setText("You must be logged in first.");

return;

}

if (currentChatRoomName == null) {

messageOutput.setText("Please create a chat room first.");

return;

}

messageOutput.append("You to " + currentChatRoomName + ": " + message + "\n");

notificationModule.sendNotification("Message sent to chat room: " + currentChatRoomName);

});

return panel;

}

private Panel createOneToOneChatPanel() {

Panel panel = new Panel(new GridLayout(4, 1, 10, 10));

panel.add(new Label("Enter Contact Name:"));

oneToOneContactField = new TextField(20);

panel.add(oneToOneContactField);

panel.add(new Label("Enter Message:"));

oneToOneMessageField = new TextField(20);

panel.add(oneToOneMessageField);

sendOneToOneMessageButton = new Button("Send Message");

panel.add(sendOneToOneMessageButton);

oneToOneMessageOutput = new TextArea(5, 40);

oneToOneMessageOutput.setEditable(false);

panel.add(oneToOneMessageOutput);

sendOneToOneMessageButton.addActionListener(e -> {

String contactName = oneToOneContactField.getText().trim();

String message = oneToOneMessageField.getText().trim();

if (contactName.isEmpty() || message.isEmpty()) {

oneToOneMessageOutput.setText("Contact name or message cannot be empty.");

return;

}

oneToOneMessageOutput.append("To " + contactName + ": " + message + "\n");

notificationModule.sendNotification("One-to-one message sent to " + contactName);

});

return panel;

}

// Event handler for buttons

public void actionPerformed(ActionEvent e) {

String command = e.getActionCommand();

if (command.equals("Register")) {

register();

} else if (command.equals("Login")) {

login();

} else if (command.equals("Add Contact")) {

addContact();

} else if (command.equals("Show Contacts")) {

showContacts();

} else if (command.equals("Back")) {

cardLayout.previous(cardPanel);

} else if (command.equals("Next")) {

cardLayout.next(cardPanel);

} else if (command.equals("Close")) {

dispose();

}

}

private void register() {

String username = usernameField.getText().trim();

String password = passwordField.getText().trim();

if (username.isEmpty() || password.isEmpty()) {

registerOutput.setText("Username or password cannot be empty.");

return;

}

if (users.containsKey(username)) {

registerOutput.setText("Username already exists.");

return;

}

User newUser = new User(username, password);

users.put(username, newUser);

registerOutput.setText("User registered successfully!");

}

private void login() {

String username = usernameField.getText().trim();

String password = passwordField.getText().trim();

if (username.isEmpty() || password.isEmpty()) {

loginOutput.setText("Username or password cannot be empty.");

return;

}

User user = users.get(username);

if (user == null || !user.getPassword().equals(password)) {

loginOutput.setText("Invalid username or password.");

return;

}

currentUser = user;

loginOutput.setText("Welcome, " + username + "!");

}

private void addContact() {

String contactName = contactField.getText().trim();

if (contactName.isEmpty()) {

contactOutput.setText("Contact name cannot be empty.");

return;

}

// Add contact to the current user's contact list

contacts.computeIfAbsent(currentUser.getUsername(), k -> new HashSet<>());

Set<String> currentUserContacts = contacts.get(currentUser.getUsername());

// If the contact is not already in the user's contact list, add them

if (currentUserContacts.contains(contactName)) {

contactOutput.setText("Contact already added.");

} else {

currentUserContacts.add(contactName); // Add the contact to the current user's contact list

contactOutput.setText("Contact added: " + contactName);

// Also add the current user to the contact's contact list

contacts.computeIfAbsent(contactName, k -> new HashSet<>());

Set<String> contactUserContacts = contacts.get(contactName);

contactUserContacts.add(currentUser.getUsername()); // Add current user to the contact's list

}

}

private void showContacts() {

if (contacts.containsKey(currentUser.getUsername())) {

Set<String> userContacts = contacts.get(currentUser.getUsername());

StringBuilder sb = new StringBuilder();

for (String contact : userContacts) {

sb.append(contact).append("\n");

}

contactOutput.setText(sb.toString());

} else {

contactOutput.setText("No contacts found.");

}

}

// User class for handling user details

public static class User {

private final String username;

private final String password;

public User(String username, String password) {

this.username = username;

this.password = password;

}

public String getUsername() {

return username;

}

public String getPassword() {

return password;

}

}

// Notification module for sending notifications

public class NotificationModule {

public void sendNotification(String message) {

System.out.println("Notification: " + message);

}

}

public static void main(String[] args) {

ChatApplicationGUI app = new ChatApplicationGUI();

app.setVisible(true);

}

}