Chat App – Part 3

https://github.com/IIEVCPMB/programming-1a-poe-ST10474044/compare/master?expand=1

src/main/java/com/mycompany/chatapp2/ChatApp2.java

|  |
| --- |
| /\* |
|  |  | \* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license |
|  |  | \*/ |
|  |  |  |
|  |  | package com.mycompany.chatapp2; |
|  |  |  |
|  |  | import java.io.\*; |
|  |  | import org.json.simple.JSONArray; |
|  |  | import org.json.simple.JSONObject; |
|  |  | import org.json.simple.parser.JSONParser; |
|  |  | import javax.swing.\*; |
|  |  | import java.awt.\*; |
|  |  | import java.util.HashMap; |
|  |  | import java.util.Random; |
|  |  | import org.json.simple.parser.ParseException; |
|  |  |  |
|  |  |  |
|  |  | /\*\* |
|  |  | \* |
|  |  | \* @author lab\_services\_student |
|  |  | \*/ |
|  |  | public class ChatApp2 { |
|  |  |  |
|  |  | public static void main(String[] args) { |
|  |  | SwingUtilities.invokeLater(ChatAppGUI::showMainMenu); |
|  |  | } |
|  |  | public class ChatAppGUI { |
|  |  | static HashMap<String, ChatAppGUI.User> users = new HashMap<>(); |
|  |  | static int messageCount = 0; |
|  |  | static final String MESSAGE\_FILE = "messages.json"; |
|  |  |  |
|  |  | static void showMainMenu() { |
|  |  | String[] options = {"Register", "Login", "Exit"}; |
|  |  | while (true) { |
|  |  | int choice = JOptionPane.showOptionDialog(null, "QuickChat", "Main Menu", |
|  |  | JOptionPane.DEFAULT\_OPTION, JOptionPane.INFORMATION\_MESSAGE, null, options, options[0]); |
|  |  |  |
|  |  | switch (choice) { |
|  |  | case 0 -> registerUser(); |
|  |  | case 1 -> loginUser(); |
|  |  | default -> { |
|  |  | JOptionPane.showMessageDialog(null, "Goodbye!"); |
|  |  | System.exit(0); |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | static void registerUser() { |
|  |  | String username = JOptionPane.showInputDialog("Create username (must contain '\_' and max 5 chars):"); |
|  |  | if (username == null || !username.contains("\_") || username.length() > 5) { |
|  |  | JOptionPane.showMessageDialog(null,"Invalid username format."); |
|  |  | return; |
|  |  | } |
|  |  |  |
|  |  | String password = JOptionPane.showInputDialog("Create password (min 8 chars, 1 capital, 1 number, 1 special char):"); |
|  |  | if (password == null || !isValidPassword(password)) { |
|  |  | JOptionPane.showMessageDialog(null,"Invalid password format."); |
|  |  | return; |
|  |  | } |
|  |  |  |
|  |  | String cellNumber = JOptionPane.showInputDialog("Enter SA phone number (+27XXXXXXXXX):"); |
|  |  | if (cellNumber == null || !isValidSAphone(cellNumber)) { |
|  |  | JOptionPane.showMessageDialog(null,"Invalid phone number format."); |
|  |  | return; |
|  |  | } |
|  |  |  |
|  |  | if (users.containsKey(username)) { |
|  |  | JOptionPane.showMessageDialog(null,"Username already exists."); |
|  |  | } else { |
|  |  | users.put(username, new User(username, password, cellNumber)); |
|  |  | JOptionPane.showMessageDialog(null,"Registration successful."); |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | static void loginUser() { |
|  |  | String username = JOptionPane.showInputDialog("Enter username:"); |
|  |  | String password = JOptionPane.showInputDialog("Enter password:"); |
|  |  |  |
|  |  | if (users.containsKey(username) && users.get(username).password.equals(password)) { |
|  |  | JOptionPane.showMessageDialog(null, "Login successful. Welcome, " + username); |
|  |  |  |
|  |  | String limitStr = JOptionPane.showInputDialog("How many messages would you like to send?"); |
|  |  | int limit; |
|  |  | try { |
|  |  | limit = Integer.parseInt(limitStr); |
|  |  | } catch (NumberFormatException e) { |
|  |  | JOptionPane.showMessageDialog(null,"Invalid number. Returning to menu."); |
|  |  | return; |
|  |  | } |
|  |  |  |
|  |  | int userMessageCount = 0; |
|  |  | while (userMessageCount < limit) { |
|  |  | String[] options = {"Send Message", "Show Stored Messages", "Logout"}; |
|  |  | int opt = JOptionPane.showOptionDialog(null, "Choose option:", "User Menu", |
|  |  | JOptionPane.DEFAULT\_OPTION, JOptionPane.INFORMATION\_MESSAGE, null, options, options[0]); |
|  |  |  |
|  |  | switch (opt) { |
|  |  | case 0 -> { |
|  |  | handleMessage(); |
|  |  | userMessageCount++; |
|  |  | } |
|  |  | case 1 -> showStoredMessages(); |
|  |  | default -> { |
|  |  | JOptionPane.showMessageDialog(null, "Logging out..."); |
|  |  | return; |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  | JOptionPane.showMessageDialog(null,"Message limit reached."); |
|  |  | } else { |
|  |  | JOptionPane.showMessageDialog(null,"Login failed."); |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | static void handleMessage() { |
|  |  | String recipient = JOptionPane.showInputDialog("Enter recipient number (+27XXXXXXXXX):"); |
|  |  | if (recipient == null || !isValidSAphone(recipient)) { |
|  |  | JOptionPane.showMessageDialog(null,"Invalid SA number."); |
|  |  | return; |
|  |  | } |
|  |  |  |
|  |  | String message = JOptionPane.showInputDialog("Enter your message (max 50 chars):"); |
|  |  | if (message == null || message.length() > 50) { |
|  |  | JOptionPane.showMessageDialog(null,"Message must be under 50 characters."); |
|  |  | return; |
|  |  | } |
|  |  |  |
|  |  | String[] options = {"Send Now", "Disregard", "Store for Later"}; |
|  |  | int action = JOptionPane.showOptionDialog(null,"Choose an action:", "Message Options", |
|  |  | JOptionPane.DEFAULT\_OPTION, JOptionPane.INFORMATION\_MESSAGE, null, options, options[0]); |
|  |  |  |
|  |  | switch (action) { |
|  |  | case 0 -> sendMessage(recipient, message); |
|  |  | case 1 -> JOptionPane.showMessageDialog(null,"Message disregarded."); |
|  |  | case 2 -> { |
|  |  | storeMessageLater(recipient, message); |
|  |  | JOptionPane.showMessageDialog(null,"Message stored for later."); |
|  |  | } |
|  |  | default -> { |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | static void sendMessage(String recipient, String content) { |
|  |  | long messageId = 1000000000L + new Random().nextInt(900000000); |
|  |  | messageCount++; |
|  |  | String[] words = content.trim().split("\\s+"); |
|  |  | String hash = String.format("%s:%d:%s:%s", |
|  |  | String.valueOf(messageId).substring(0, 2), |
|  |  | messageCount, |
|  |  | words[0].toUpperCase(), |
|  |  | words[words.length - 1].toUpperCase()); |
|  |  |  |
|  |  | JSONObject messageObject = new JSONObject(); |
|  |  | messageObject.put("MessageID", messageId); |
|  |  | messageObject.put("MessageHash", hash); |
|  |  | messageObject.put("Recipient", recipient); |
|  |  | messageObject.put("Message", content); |
|  |  |  |
|  |  | saveMessageToJson(messageObject); |
|  |  |  |
|  |  | JOptionPane.showMessageDialog(null,"Message sent!\nMessage ID: " + messageId + "\nMessage Hash: " + hash); |
|  |  | } |
|  |  |  |
|  |  | static void storeMessageLater(String recipient, String content) { |
|  |  | JSONObject messageObject = new JSONObject(); |
|  |  | messageObject.put("Recipient", recipient); |
|  |  | messageObject.put("Message", content); |
|  |  | messageObject.put("Status", "StoredForLater"); |
|  |  |  |
|  |  | saveMessageToJson(messageObject); |
|  |  | } |
|  |  |  |
|  |  | static void showStoredMessages() { |
|  |  | try { |
|  |  | File file = new File(MESSAGE\_FILE); |
|  |  | if (!file.exists()) { |
|  |  | JOptionPane.showMessageDialog(null, "No stored messages."); |
|  |  | return; |
|  |  | } |
|  |  |  |
|  |  | JSONParser parser = new JSONParser(); |
|  |  | JSONArray messages = (JSONArray) parser.parse(new FileReader(MESSAGE\_FILE)); |
|  |  |  |
|  |  | StringBuilder display = new StringBuilder("--- Stored Messages ---\n"); |
|  |  | for (Object msg : messages) { |
|  |  | JSONObject obj = (JSONObject) msg; |
|  |  | display.append(obj.toJSONString()).append("\n\n"); |
|  |  | } |
|  |  |  |
|  |  | JTextArea textArea = new JTextArea(display.toString()); |
|  |  | textArea.setEditable(false); |
|  |  | JScrollPane scrollPane = new JScrollPane(textArea); |
|  |  | scrollPane.setPreferredSize(new Dimension(500, 400)); |
|  |  |  |
|  |  | JOptionPane.showMessageDialog(null, scrollPane, "Stored Messages", JOptionPane.INFORMATION\_MESSAGE); |
|  |  |  |
|  |  | } catch (HeadlessException | IOException | ParseException e) { |
|  |  | JOptionPane.showMessageDialog(null, "Error reading messages."); |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | static void saveMessageToJson(JSONObject message) { |
|  |  | JSONArray messages = new JSONArray(); |
|  |  | try { |
|  |  | File file = new File(MESSAGE\_FILE); |
|  |  | if (file.exists() && file.length() > 0) { |
|  |  | JSONParser parser = new JSONParser(); |
|  |  | messages = (JSONArray) parser.parse(new FileReader(MESSAGE\_FILE)); |
|  |  | } |
|  |  | } catch (IOException | ParseException ignored) {} |
|  |  |  |
|  |  | messages.add(message); |
|  |  |  |
|  |  | try (FileWriter writer = new FileWriter(MESSAGE\_FILE)) { |
|  |  | writer.write(messages.toJSONString()); |
|  |  | } catch (IOException e) { |
|  |  | JOptionPane.showMessageDialog(null, "Failed to save message."); |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | static boolean isValidPassword(String password) { |
|  |  | return password.length() >= 8 && |
|  |  | password.matches(".\*[A-Z].\*") && |
|  |  | password.matches(".\*\\d.\*") && |
|  |  | password.matches(".\*[^a-zA-Z0-9].\*"); |
|  |  | } |
|  |  |  |
|  |  | static boolean isValidSAphone(String number) { |
|  |  | return number.matches("\\+27\\d{9}"); |
|  |  | } |
|  |  |  |
|  |  | static class User { |
|  |  | String username, password, cellNumber; |
|  |  |  |
|  |  | User(String username, String password, String cellNumber) { |
|  |  | this.username = username; |
|  |  | this.password = password; |
|  |  | this.cellNumber = cellNumber; |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | } |