PROG POE

```
import java.util.Scanner;
import java.util.regex.Pattern;

public class RegistrationLoginApp {
    private static Login loginSystem = new Login();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        System.out.println("=== Registration and Login System ===");

        boolean running = true;
        while (running) {
            System.out.println("\n1. Register");
            System.out.println("2. Login");
            System.out.println("3. Exit");
            System.out.print("Choose an option: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    registerUser();
                    break;
                case 2:
                    loginUser();
                    break;
```

```
            case 3:
                running = false;
                System.out.println("Goodbye!");
                break;
            default:
                System.out.println("Invalid option. Please try again.");
        }
    }
}

private static void registerUser() {
    System.out.println("\n=== User Registration ===");

    System.out.print("Enter username: ");
    String username = scanner.nextLine();

    System.out.print("Enter password: ");
    String password = scanner.nextLine();

    System.out.print("Enter cell phone number: ");
    String cellPhone = scanner.nextLine();

    System.out.print("Enter first name: ");
    String firstName = scanner.nextLine();

    System.out.print("Enter last name: ");
    String lastName = scanner.nextLine();
```

```
    // Set user details
    loginSystem.setUsername(username);
    loginSystem.setPassword(password);
    loginSystem.setCellPhone(cellPhone);
    loginSystem.setFirstName(firstName);
    loginSystem.setLastName(lastName);

    // Register user
    String result = loginSystem.registerUser();
    System.out.println(result);
}

private static void loginUser() {
    System.out.println("\n=== User Login ===");

    System.out.print("Enter username: ");
    String username = scanner.nextLine();

    System.out.print("Enter password: ");
    String password = scanner.nextLine();

    // Set login credentials
    loginSystem.setLoginUsername(username);
    loginSystem.setLoginPassword(password);

    // Attempt login
    boolean loginSuccess = loginSystem.loginUser();
    String status = loginSystem.returnLoginStatus();
```

```
        System.out.println(status);
    }
}
```

Login Class

```
import java.util.regex.Pattern;

/**
```

- Login class handling user registration and authentication
- AI Tool Reference: This code was developed with assistance from ChatGPT (OpenAI)
- Reference: OpenAI. (2024). ChatGPT (Version 3.5) [Large language model]. https://chat.openai.com

```
*/ public class Login { private String username; private String password; private String cellPhone; private String firstName; private String lastName; private String loginUsername; private String loginPassword;

// Regular expression for password complexity
private static final String PASSWORD_PATTERN = "^(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$%^&*()_+\\-=\\[\\]{};':\"\\\\|,.<>\\/?]).{8,}$";

// Regular expression for South African cell phone numbers with international code
// AI Tool Reference: This regex pattern was developed with assistance from ChatGPT (OpenAI)
// Reference: OpenAI. (2024). ChatGPT (Version 3.5) [Large language model]. https://chat.openai.com
private static final String CELL_PHONE_PATTERN = "^\\+27[0-9]{9}$";

/**
 * Checks if username contains underscore and is no more than 5 characters
 * @return true if username is correctly formatted
 */
public boolean checkUserName() {
```

```
    if (username == null) return false;
    return username.length() <= 5 && username.contains("_");
}

/**
 * Checks if password meets complexity requirements
 * @return true if password meets complexity requirements
 */
public boolean checkPasswordComplexity() {if (password == null) return false;

    // Check length
    if (password.length() < 8) return false;

    // Check for capital letter
    boolean hasCapital = false;
    // Check for number
    boolean hasNumber = false;
    // Check for special character
    boolean hasSpecial = false;

    for (char c : password.toCharArray()) {
        if (Character.isUpperCase(c)) hasCapital = true;
        if (Character.isDigit(c)) hasNumber = true;
        if (!Character.isLetterOrDigit(c) && !Character.isWhitespace(c)) hasSpecial = true;
    }

    return hasCapital && hasNumber && hasSpecial;

    // Alternative using regex (commented out as per requirement to use manual
checking)
    // return Pattern.matches(PASSWORD_PATTERN, password);
}

/**
 * Checks if cell phone number is correctly formatted with international code
 * @return true if cell phone number is correctly formatted
 */
public class Login {

    private String username;

    private String password;
```

```
    private String cellPhone;
    private String firstName;
    private String lastName;
    private String loginUsername;
    private String loginPassword;
}
  public boolean checkUserName() {
    if (username == null) return false;
    return username.length() <= 5 && username.contains("_");
}
public boolean checkPasswordComplexity() {if (password == null) return false;

if (password.length() < 8) return false;

boolean hasCapital = false;

boolean hasNumber = false;

boolean hasSpecial = false;

for (char c : password.toCharArray()) {
      if (Character.isUpperCase(c)) hasCapital = true;
      if (Character.isDigit(c)) hasNumber = true;
      if (!Character.isLetterOrDigit(c) && !Character.isWhitespace(c)) hasSpecial = true;
    }

return hasCapital && hasNumber && hasSpecial; }
```

```
}

/**
 * Registers a new user with validation
 * @return registration status message
 */
public String registerUser() {
   if (!checkUserName()) {
      return "Username is not correctly formatted, please ensure that your username
contains an underscore and is no more than five characters in length.";
   }   if (!checkPasswordComplexity()) {
      return "Password is not correctly formatted; please ensure that the password
contains at least eight characters, a capital letter, a number, and a special character.";
   }

   if (!checkCellPhoneNumber()) {
      return "Cell phone number incorrectly formatted or does not contain international
code.";
   }

   return "Username successfully captured.\nPassword successfully captured.\nCell
phone number successfully added.\nUser registered successfully.";
}

/**
 * Verifies login credentials
 * @return true if login is successful
 */
public boolean loginUser() {
   return username != null &&
         password != null &&
         username.equals(loginUsername) &&
         password.equals(loginPassword);
}

/**
 * Returns login status message
 * @return login status message
 */
public String returnLoginStatus() {
   if (loginUser()) {
```

```
        return "Welcome " + firstName + " " + lastName + " it is great to see you again.";
    } else {
        return "Username or password incorrect, please try again.";
    }
}

// Getters and Setters
public String getUsername() { return username; }
public void setUsername(String username) { this.username = username; }

public String getPassword() { return password; }
public void setPassword(String password) { this.password = password; }

public String getCellPhone() { return cellPhone; }
public void setCellPhone(String cellPhone) { this.cellPhone = cellPhone; }

public String getFirstName() { return firstName; }
public void setFirstName(String firstName) { this.firstName = firstName; }

public String getLastName() { return lastName; }
public void setLastName(String lastName) { this.lastName = lastName; }

public String getLoginUsername() { return loginUsername; }
public void setLoginUsername(String loginUsername) { this.loginUsername =
loginUsername; }

public String getLoginPassword() { return loginPassword; }
public void setLoginPassword { this.loginPassword = loginPassword; }
```

Messasge class part 2

```
import java.util.; import java.util.regex.Pattern; import org.json.JSONArray; import
org.json.JSONObject; import java.io.; import java.nio.file.Files; import
java.nio.file.Paths;

/**
```

- Message class handling message creation, validation, and storage

- AI Tool Reference: JSON storage method developed with assistance from ChatGPT (OpenAI)
- Reference: OpenAI. (2024). ChatGPT (Version 3.5) [Large language model]. https://chat.openai.com

*/ public class Message { private String messageID; private int numMessagesSent; private String recipient; private String message; private String messageHash; private static int totalMessages = 0; private static List sentMessages = new ArrayList<>(); private static List storedMessages = new ArrayList<>();

// Regular expression for South African cell phone numbers with international code private static final String CELL_PHONE_PATTERN = "^\\+27[0-9]{9}$";

```
public Message() {
 this.messageID = generateMessageID();
  this.numMessagesSent = ++totalMessages;
}

/**
 * Generates a random 10-digit message ID
 */
private String generateMessageID() {
  Random random = new Random();
  long id = 1000000000L + (long)(random.nextDouble() * 9000000000L);
  return String.valueOf(id);
}

/**
 * Checks if message ID is exactly 10 characters
 */
public boolean checkMessageID() {
  return messageID != null && messageID.length() == 10;
}

/**
 * Checks recipient cell number format
 */
public int checkRecipientCell() {
  if (recipient == null) return -1;

  if (Pattern.matches(CELL_PHONE_PATTERN, recipient)) {
```

```
        return 1; // Success
    } else {
        return 0; // Failure
    }
}

/**
 * Creates message hash in format: first2Digits:messageNumber:firstWordLastWord
 */
public String createMessageHash() {
    if (message == null || message.isEmpty()) return "";

    // Get first two digits of message ID
    String firstTwoDigits = messageID.substring(0, 2);

    // Get message number
    String messageNum = String.valueOf(numMessagesSent);

    // Extract first and last words
    String[] words = message.trim().split("\\s+");
    String firstWord = words[0].replaceAll("[^a-zA-Z]", "").toUpperCase();
    String lastWord = words.length > 1 ?
        words[words.length - 1].replaceAll("[^a-zA-Z]", "").toUpperCase() :
        firstWord;

    this.messageHash = firstTwoDigits + ":" + messageNum + ":" + firstWord + lastWord;
    return this.messageHash;
}

/**
 * Handles message sending options
 */
public String sentMessage(int choice) {
    switch (choice) {
        case 1: // Send
            sentMessages.add(this);
            storeMessage(); // Also store in JSON
            return "Message successfully sent.";
        case 2: // Disregard
            return "Press 0 to delete message.";
        case 3: // Store for later
```

```
            storedMessages.add(this);
            storeMessage();
            return "Message successfully stored.";
        default:
            return "Invalid choice.";
    }
}

/**
 * Stores message in JSON file
 * AI Tool Reference: This JSON storage method was developed with assistance from ChatGPT (OpenAI)
 * Reference: OpenAI. (2024). ChatGPT (Version 3.5) [Large language model]. https://chat.openai.com
 */
public void storeMessage() {
    try {
        JSONObject messageJson = new JSONObject();
        messageJson.put("messageID", this.messageID);
        messageJson.put("numMessagesSent", this.numMessagesSent);
        messageJson.put("recipient", this.recipient);
        messageJson.put("message", this.message);
        messageJson.put("messageHash", this.messageHash);
        messageJson.put("timestamp", new Date().toString());

        // Read existing messages
        JSONArray messagesArray;
        File file = new File("messages.json");
        if (file.exists()) {
            String content = new String(Files.readAllBytes(Paths.get("messages.json")));
            messagesArray = new JSONArray(content);
        } else {
            messagesArray = new JSONArray();
        }

        // Add new message
        messagesArray.put(messageJson);

        // Write back to file
        Files.write(Paths.get("messages.json"), messagesArray.toString(4).getBytes());
```

```
    } catch (Exception e) {
        System.out.println("Error storing message: " + e.getMessage());
    }
}

/**
 * Returns formatted list of all sent messages
 */
public String printMessage() {
    StringBuilder sb = new StringBuilder();
    sb.append("=== ALL SENT MESSAGES ===\n");

    for (Message msg : sentMessages) {
        sb.append("MessageID: ").append(msg.getMessageID()).append("\n");
        sb.append("Message Hash: ").append(msg.getMessageHash()).append("\n");
        sb.append("Recipient: ").append(msg.getRecipient()).append("\n");
        sb.append("Message: ").append(msg.getMessage()).append("\n");
        sb.append("------------------------\n");
    }

    sb.append("Total Messages Sent: ").append(returnTotalMessages());
    return sb.toString();
}

/**
 * Returns total number of messages sent
 */
public int returnTotalMessages() {
    return sentMessages.size();
}

/**
 * Checks if message length is valid (<= 250 characters)
 */
public String checkMessageLength() {
    if (message == null) {
        return "Message is null.";
    }

 if (message.length() <= 250) {
        return "Message ready to send.";
```

```
    } else {
        int excess = message.length() - 250;
        return "Message exceeds 250 characters by " + excess + ", please reduce size.";
    }
}

/**
 * Validates recipient cell number and returns appropriate message
 */
public String validateRecipientNumber() {
    int result = checkRecipientCell();
    if (result == 1) {
        return "Cell phone number successfully captured.";
    } else {
        return "Cell phone number is incorrectly formatted or does not contain an
international code. Please correct the number and try again.";
    }
}

// Getters and Setters
public String getMessageID() { return messageID; }
public void setMessageID(String messageID) { this.messageID = messageID; }

public int getNumMessagesSent() { return numMessagesSent; }
public void setNumMessagesSent(int numMessagesSent) { this.numMessagesSent =
numMessagesSent; }

public String getRecipient() { return recipient; }
public void setRecipient(String recipient) { this.recipient = recipient; }

public String getMessage() { return message; }
public void setMessage(String message) { this.message = message; }

public String getMessageHash() { return messageHash; }
public void setMessageHash(String messageHash) { this.messageHash =
messageHash; }

public static List<Message> getSentMessages() { return sentMessages; }
public static List<Message> getStoredMessages() { return storedMessages; }
```

```
}
```

Enhanced main class

```
import javax.swing.JOptionPane;
import java.util.Scanner;

public class EnhancedRegistrationLoginApp {
    private static Login loginSystem = new Login();
    private static Scanner scanner = new Scanner(System.in);
    private static Message messageSystem = new Message();

    public static void main(String[] args) {
        System.out.println("=== Enhanced Registration, Login and Messaging System ===");

        boolean running = true;
        while (running) {
            System.out.println("\n1. Register");
            System.out.println("2. Login");
            System.out.println("3. Send Messages");
            System.out.println("4. View Sent Messages");
            System.out.println("5. Exit");
            System.out.print("Choose an option: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline
```

```
        switch (choice) {
            case 1:
                registerUser();
                break;
            case 2:
                loginUser();
                break;
            case 3:
                sendMessages();
                break;
            case 4:
                viewSentMessages();
                break;
            case 5:
                running = false;
                System.out.println("Goodbye!");
                break;
            default:
                System.out.println("Invalid option. Please try again.");
        }
    }
}

private static void registerUser() {
    System.out.println("\n=== User Registration ===");

    System.out.print("Enter username: ");
    String username = scanner.nextLine();
```

```
        System.out.print("Enter password: ");
        String password = scanner.nextLine();

        System.out.print("Enter cell phone number: ");
        String cellPhone = scanner.nextLine();

        System.out.print("Enter first name: ");
        String firstName = scanner.nextLine();

        System.out.print("Enter last name: ");
        String lastName = scanner.nextLine();

        loginSystem.setUsername(username);
        loginSystem.setPassword(password);
        loginSystem.setCellPhone(cellPhone);
        loginSystem.setFirstName(firstName);
        loginSystem.setLastName(lastName);

        String result = loginSystem.registerUser();
        System.out.println(result);
    }

    private static void loginUser() {
        System.out.println("\n=== User Login ===");

        System.out.print("Enter username: ");
        String username = scanner.nextLine();
```

```
    System.out.print("Enter password: ");
    String password = scanner.nextLine();

    loginSystem.setLoginUsername(username);
    loginSystem.setLoginPassword(password);

    boolean loginSuccess = loginSystem.loginUser();
    String status = loginSystem.returnLoginStatus();
    System.out.println(status);
}

private static void sendMessages() {
    System.out.println("\n=== Message Sending System ===");

    System.out.print("How many messages do you wish to enter? ");
    int numMessages = scanner.nextInt();
    scanner.nextLine(); // consume newline

    for (int i = 0; i < numMessages; i++) {
        System.out.println("\n--- Message " + (i + 1) + " of " + numMessages + " ---");

        Message message = new Message();

        // Get recipient number
        System.out.print("Enter recipient cell number: ");
        String recipient = scanner.nextLine();
        message.setRecipient(recipient);
```

```
        // Validate recipient
        String recipientValidation = message.validateRecipientNumber();
        System.out.println(recipientValidation);

        if (!recipientValidation.contains("successfully")) {
            System.out.println("Skipping this message due to invalid recipient.");
            continue;
        }

        // Get message content
        System.out.print("Enter your message (max 250 characters): ");
        String messageContent = scanner.nextLine();
        message.setMessage(messageContent);

        // Validate message length
        String lengthValidation = message.checkMessageLength();
        System.out.println(lengthValidation);

        if (!lengthValidation.contains("ready to send")) {
            System.out.println("Message too long. Please try again with a shorter
message.");
            i--; // Retry this message
            continue;
        }

        // Create message hash
        String messageHash = message.createMessageHash();
```

```
        System.out.println("Message Hash: " + messageHash);

        // Display message details using JOptionPane
        displayMessageDetails(message);

        // Get user choice for message handling
        System.out.println("\nChoose an option:");
        System.out.println("1. Send Message");
        System.out.println("2. Disregard Message");
        System.out.println("3. Store Message to send later");
        System.out.print("Enter your choice (1-3): ");

        int messageChoice = scanner.nextInt();
        scanner.nextLine(); // consume newline

        String result = message.sentMessage(messageChoice);
        System.out.println(result);

        if (messageChoice == 2) { // Disregard
            System.out.print("Press 0 to confirm deletion: ");
            int confirm = scanner.nextInt();
            scanner.nextLine();
            if (confirm == 0) {
                System.out.println("Message deleted.");
            }
        }
    }
```

```
        System.out.println("\nTotal messages sent in this session: " +
messageSystem.returnTotalMessages());
    }

    private static void displayMessageDetails(Message message) {
        String details = "Message Details:\n" +
                "MessageID: " + message.getMessageID() + "\n" +
                "Message Hash: " + message.getMessageHash() + "\n" +
                "Recipient: " + message.getRecipient() + "\n" +
                "Message: " + message.getMessage() + "\n" +
                "Message Number: " + message.getNumMessagesSent();

        JOptionPane.showMessageDialog(null, details, "Message Details",
JOptionPane.INFORMATION_MESSAGE);
        System.out.println(details);
    }

    private static void viewSentMessages() {
        System.out.println("\n" + messageSystem.printMessage());
    }
}
```