Name, Vorname : Mauladendra, Imdi Melvana

Matrikelnummer: 569252

- Feature und Function

1. Authentication
   Function:
   - Login
   - Register
   - Logout
2. Chat Room
   Function:
   - Create
   - Open
   - Find
3. Chat
   Function:
   - Write
   - Find
   - Delete

- Gherkin für Feature

1. Login

```
Feature: User login
        Scenario Outline: successful login
                Given the user is already registered in the DB with <username> and
                <password>
                And the given input (<username> and <password>) are correct
                Then the user can successfully login with <username> and <password>

                Examples:
| username | password |
| "Idokli22" | "Passw0rd" |
| "TestUser1" | "Passw0rd" |


        Scenario Outline: unsuccessful login – user is not registered
                Given the user with the <username> is not registered in the DB
                And the given input (<username> and <password>) are correct
                Then the user can not login with <username> and <password>

        Examples:
        | username | password |
        | "Idokli22" | "Passw0rd" |
                | "TestUser1" | "Passw0rd" |

        Scenario Outline: unsuccessful login - user name or password is wrong
        Given the user is already registered in the DB with <username> and <password>
        And the given input (<falseusername> and <password>) are not correct
        Then the user can not login with <falseusername> and <password>

    Examples:
    | username | password | falseusername |
    | "Idokli22" | "Passw0rd" | "Idokli"  |
    | "TestUser1" | "Passw0rd" | "testuser1" |

    Scenario: User logout successfully
        When User logouts from application
        Then Message displays Logout successfully
```

2. All chat rooms view

```
Feature: All chat rooms view
    Scenario Outline: user with existing chat rooms

        Given the user is already registered in the DB with <username> and <password>
        And the user (<username>) has User-Chats entities
        Then after login (<username> and <password>),he should see the different chat rooms that are available to him

        Examples:
        | username | password |
        | "Idokli22" | "Passw0rd" |
        | "TestUser1" | "Passw0rd" |

    Scenario Outline: user without existing chat rooms
        Given the user is already registered in the DB with <username> and <password>
        And the user (<username>) has no User-Chats entities
        Then after login (<username> and <password>),he should not see any chat rooms available

         Examples:
        | username | password |
        | "Idokli22" | "Passw0rd" |
        | "TestUser1" | "Passw0rd" |

    Scenario Outline: open a new chat
        Given the user is already registered in the DB with <username> and <password>
        And the user (<username>) has no User-Chats entities
        When the user (with <username>) wants to open a chat room he needs to enter the <chatName> and participants.
        Then a new chat room with the <chatName> should be opened for user (with <username>).

        Examples:
```

```
| username | password | chatName|
| "Idokli22" | "Passw0rd" | "room1"|
| "TestUser1" | "Passw0rd" | "room2"|
```

3. Messaging

```
Feature: Send a receive messages in chat rooms
        Scenario Outline: successful sending of a message

                Given the user is already registered in the DB with <username1> and <password1>
                And another user is already registered in the DB with <username2> and <password2>
                And chat room (with name <chatName> and Admin <username1>)for both users (<username2> is open
                When <username1> writes a message in <chatName> with <content> and sents it
                Then the message with <content> should be shown in <chatName> for all users <username1> and <username2>.

                Examples:
                | username1 | password1 | username2 | password2 | chatName | content |
                | "Idokli22" | "Passw0rd" | "Maxb22289" | "Passw0rd" | "Test1" | "what's up?" |
                | "TestUser1" | "Passw0rd" | "TestUser2" | "Passw0rd" | "Test2" | "what's up? 2" |
```

4. Registration

```
Feature: User registration
        Scenario Outline: successful registration
                Given the <username> is not registered in the app
                And the given input (<username> and <password>) are correct
                When the user tries to register with <username> and <password>
                Then the user with the <username> should successful registered

                Examples:
                | username | password |
                | "Idokli22" | "Passw0rd" |
                | "TestUser1" | "Passw0rd" |

        Scenario Outline: unsuccessful registration – username taken
                Given the <username> is already registered in the app
                And the given input (<username> and <password>) are correct
                When the user tries to register with <username> and <password>
                Then the user should get an error message that tells him that the <username> is taken.

                Examples:
                | username | password |
                | "Idokli22" | "Passw0rd" |
                | "TestUser1" | "Passw0rd" |
```

❖ Verfeinerung der algorithmischen Funktionen

- Authentication

1. Login

```
Login(username, password)
  If username-defined
          load-username
            return OK
return error
```

2. Logout

```
Logout(sid)
  Destroy-sid
  Delete-sid in account
```

3. register

```
registerUser(username, password)
  if username defined
          create user
          store user to DB
          return ok
  else
          return error
```

- Chat Room

1. Create

```
openNewChatRoom(chatName, admin, participants)
    load-chat
    create chatroom for admin
    while participants:=participants
          create chatroom for participants
    store chat room to admin
    store chat room to participants
    return chat
```

2. Open

```
getChatRoomsForUser(username)
    if username-defined
        return Chat list from username
    else
        return error
```

3. Find

```
findChatRoomByUserAndChatName(userName, chatName)
    load-chat list
    if chat list-defined
          while chat:=chat list
```

```
            if chatname-defined
            return chat
    return error
```

- Chat

1. Write

```
addMessageToChatRoom(from, chatName, content)
    load-chat room
    load-user
    if chat name-defined and chat room-defined
            create content
            load-chat room messages
            store messages
            return messages
    return error
```

2. Find

```
findChatbyChatUser(username)
    if username-defined
            return chat
    else
            return null
```

3. Delete

```
deleteChat(username, content)
    if username-defined
            if content-defined
                    return content
            delete chat(username, content)
```