Cyrus Edgren 101606991
Tutor: Wei Lai

5.2C

Task 1.

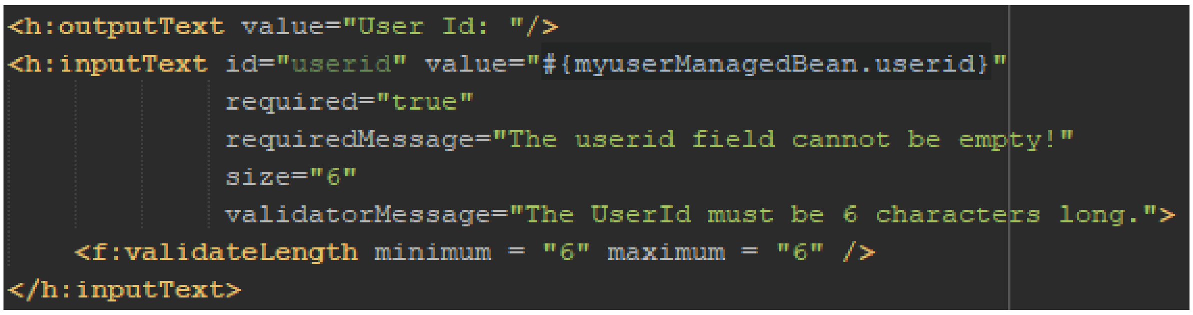
For checking the user id, since the check was just matching exactly 6 characters this was most simply done with the standard validator tag ‘<f:validateLength>’. This simply checks if the input is between a min and max number of characters, which with both set to 6 works perfectly for this use. The attribute ‘validatorMessage’ in the input text was used to set a custom message for if the validation failed.

When matching the password and confirm password against the list of requirements it was simplest to go with the standard validator tag ‘<f:validateRegex>’. This tag allows the input to be checked against a regular expression which can check for all the use cases of passwords that we would want.

There is no standard validator tag that would work to compare the password and the confirm password, so this prompted the slightly more complicated process of added a user-defined validation method in the MyuserManagedBean class. In this method it was possible to retrieve the other value that was required and test for the match.

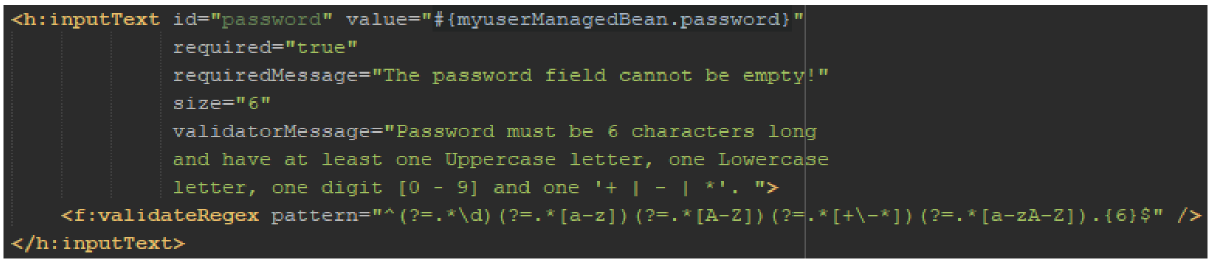
Task 2.

User Id validation check with <f:validateLength>



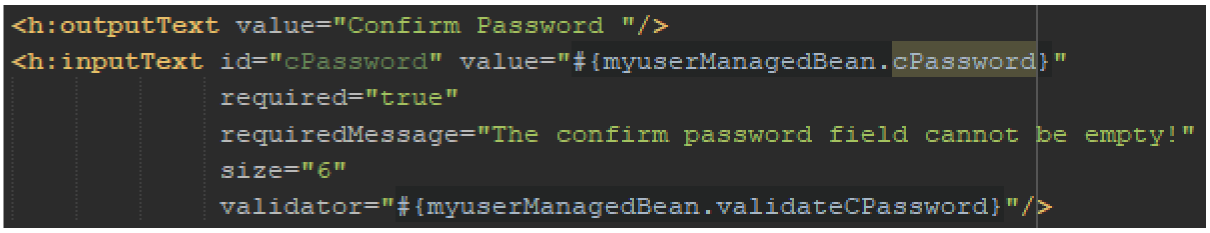
```
<h:outputText value="User Id: "/>
<h:inputText id="userid" value="#{myuserManagedBean.userid}"
             required="true"
             requiredMessage="The userid field cannot be empty!"
             size="6"
             validatorMessage="The UserId must be 6 characters long.">
    <f:validateLength minimum = "6" maximum = "6" />
</h:inputText>
```

Password field with regular expression check:



```
<h:inputText id="password" value="#{myuserManagedBean.password}"
             required="true"
             requiredMessage="The password field cannot be empty!"
             size="6"
             validatorMessage="Password must be 6 characters long
             and have at least one Uppercase letter, one Lowercase
             letter, one digit [0 - 9] and one '+ | - | *'. ">
    <f:validateRegex pattern="^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[+\-*])(?=.*[a-zA-Z]).{6}$" />
</h:inputText>
```

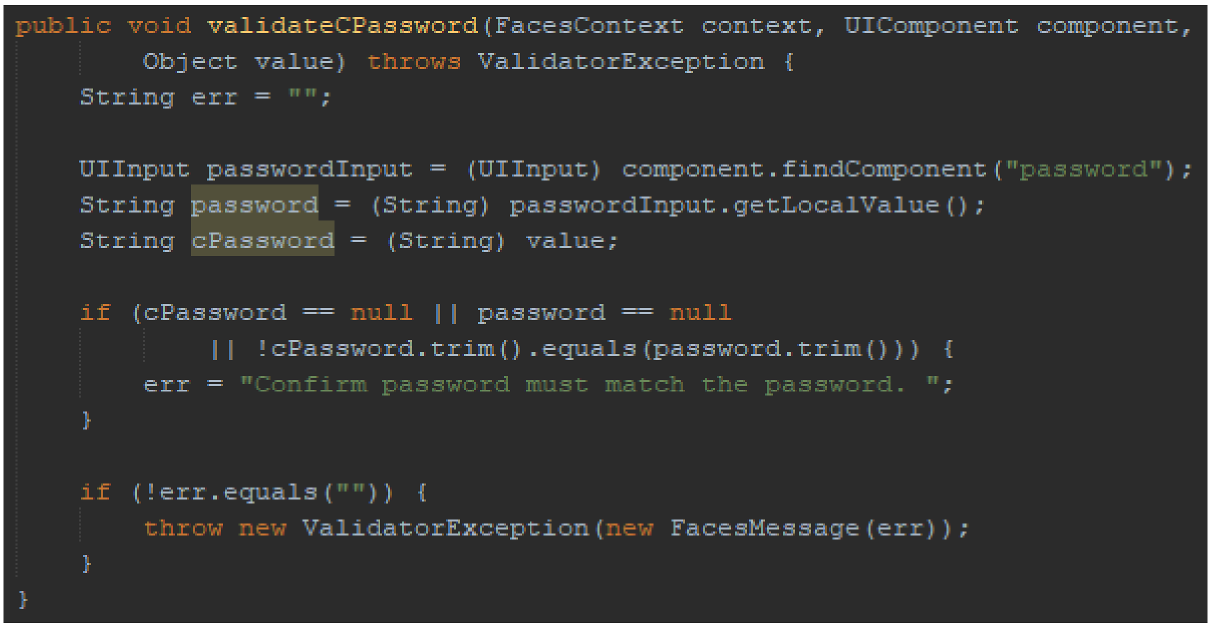
Confirm password with custom method validation:



```
<h:outputText value="Confirm Password "/>
<h:inputText id="cPassword" value="#{myuserManagedBean.cPassword}"
             required="true"
             requiredMessage="The confirm password field cannot be empty!"
             size="6"
             validator="#{myuserManagedBean.validateCPassword}"/>
```

Cyrus Edgren 101606991
Tutor: Wei Lai

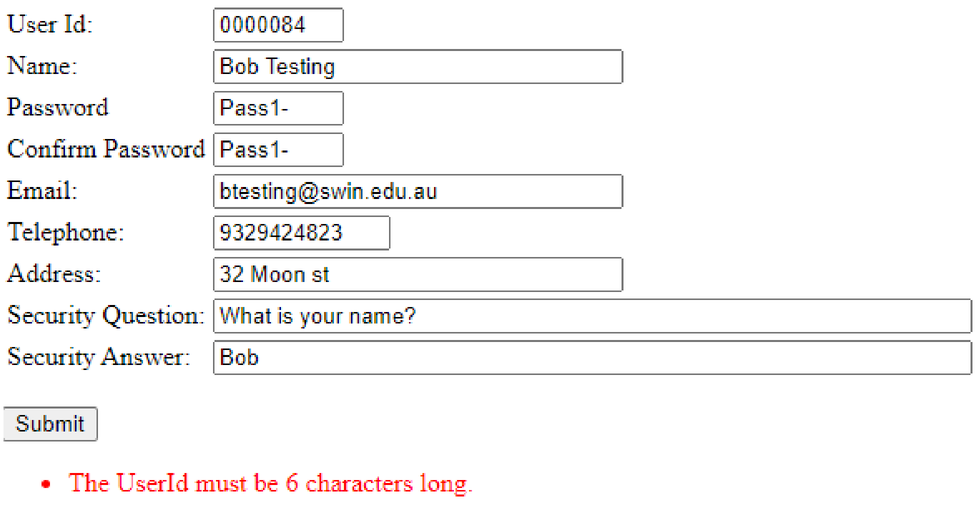
Function for making sure both passwords matched:



```
public void validateCPassword(FacesContext context, UIComponent component,
        Object value) throws ValidatorException {
    String err = "";

    UIInput passwordInput = (UIInput) component.findComponent("password");
    String password = (String) passwordInput.getLocalValue();
    String cPassword = (String) value;

    if (cPassword == null || password == null
            || !cPassword.trim().equals(password.trim())) {
        err = "Confirm password must match the password. ";
    }

    if (!err.equals("")) {
        throw new ValidatorException(new FacesMessage(err));
    }
}
```

Task 3.

User Id validation:

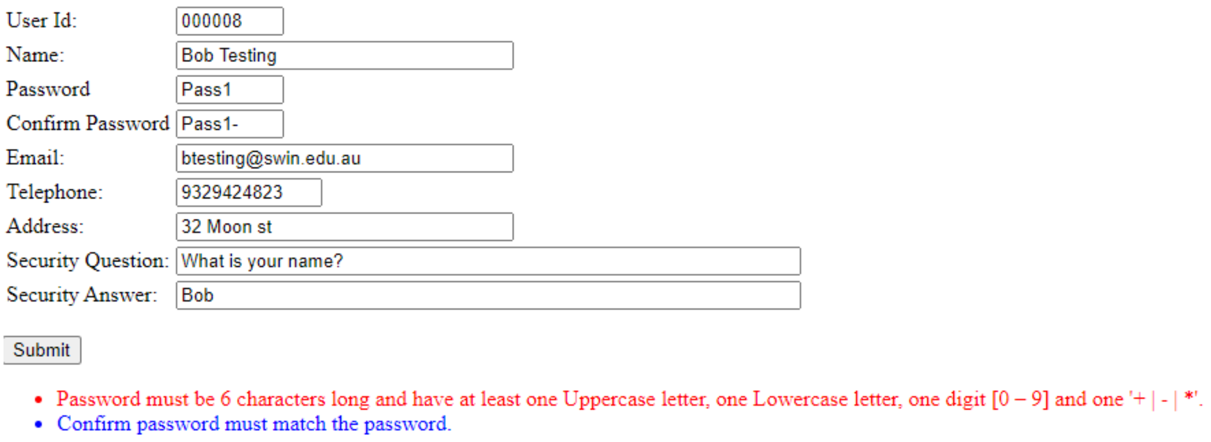


User Id:	0000084
Name:	Bob Testing
Password	Pass1-
Confirm Password	Pass1-
Email:	btesting@swin.edu.au
Telephone:	9329424823
Address:	32 Moon st
Security Question:	What is your name?
Security Answer:	Bob

Submit

- The UserId must be 6 characters long.

Password regular expression match:



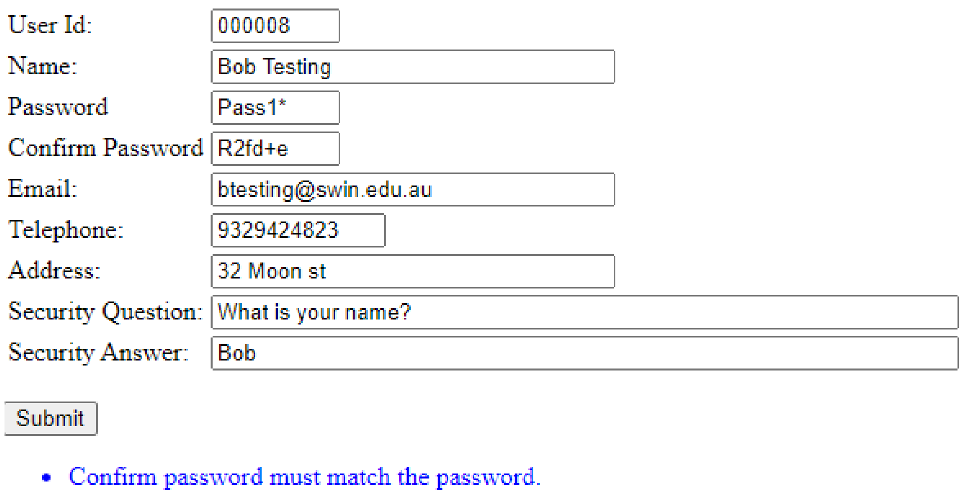
User Id:	000008
Name:	Bob Testing
Password	Pass1
Confirm Password	Pass1-
Email:	btesting@swin.edu.au
Telephone:	9329424823
Address:	32 Moon st
Security Question:	What is your name?
Security Answer:	Bob

Submit

- Password must be 6 characters long and have at least one Uppercase letter, one Lowercase letter, one digit [0 – 9] and one '+ | - | *'.
- Confirm password must match the password.

Cyrus Edgren 101606991
Tutor: Wei Lai

Different passwords:

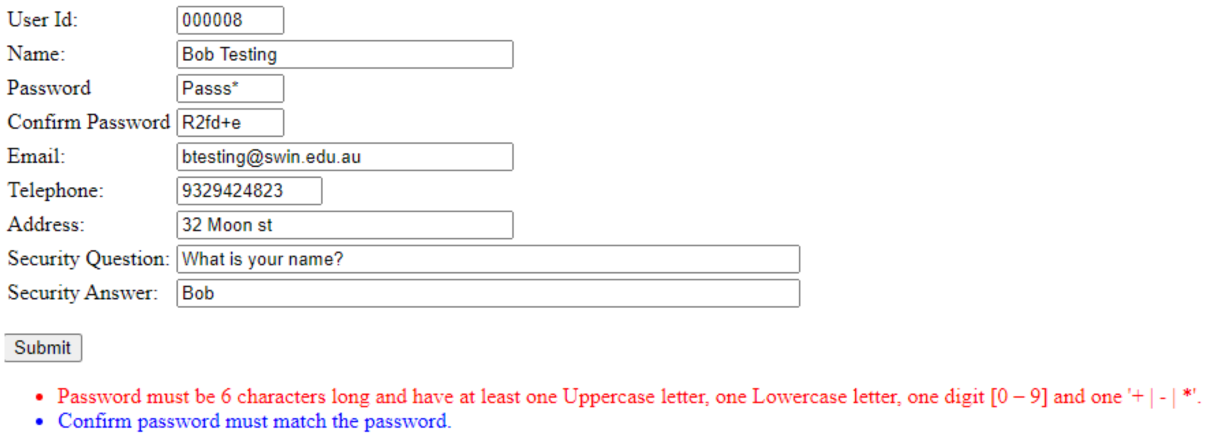


User Id:	000008
Name:	Bob Testing
Password	Pass1*
Confirm Password	R2fd+e
Email:	btesting@swin.edu.au
Telephone:	9329424823
Address:	32 Moon st
Security Question:	What is your name?
Security Answer:	Bob

Submit

- Confirm password must match the password.

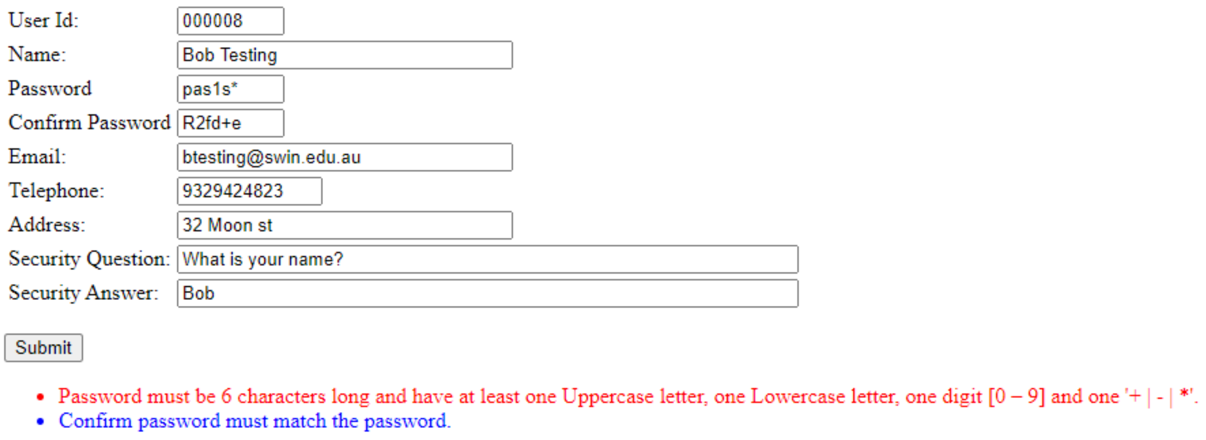
More regular expression examples



User Id:	000008
Name:	Bob Testing
Password	Pass*
Confirm Password	R2fd+e
Email:	btesting@swin.edu.au
Telephone:	9329424823
Address:	32 Moon st
Security Question:	What is your name?
Security Answer:	Bob

Submit

- Password must be 6 characters long and have at least one Uppercase letter, one Lowercase letter, one digit [0 – 9] and one '+ | - | *'.
- Confirm password must match the password.



User Id:	000008
Name:	Bob Testing
Password	pas1s*
Confirm Password	R2fd+e
Email:	btesting@swin.edu.au
Telephone:	9329424823
Address:	32 Moon st
Security Question:	What is your name?
Security Answer:	Bob

Submit

- Password must be 6 characters long and have at least one Uppercase letter, one Lowercase letter, one digit [0 – 9] and one '+ | - | *'.
- Confirm password must match the password.

Cyrus Edgren 101606991
Tutor: Wei Lai

Task 4.

Finding the information to complete this task was not too hard. Information on the validator tags was quick to find on Tutorial Point [1] and was easy to understand and use. Finding out about making custom validation functions was also quick as there was a nice blog post from Turreta [2] about this that was very useful in how to create these kinds of functions. One point that almost proved to be a stumbling block was when figuring out how to grab the password to confirm against the confirm password. Thankfully there was another blog post, this time from Omnijava [3], that showed how to solve for this exact problem. With these resources this task went smoothly, after solving a problem I had caused by spreading the regular expression across two lines in the validator tag. Aside from that issue, there were no other roadblocks in the completion of this task.

[1] JSF - Validator Tags - Tutorialspoint. (2021). Retrieved 6 May 2021, from https://www.tutorialspoint.com/jsf/jsf_validation_tags.htm
[2] Gabriel, K. (2017). JSF - Managed Bean Custom Validation Methods - Turreta. Retrieved 7 May 2021, from https://turreta.com/2017/07/01/jsf-managed-bean-custom-validation-methods/
[3] Fogel, K. (2018). Password Confirmation on a JSF Page – Part 1 A Simple Model – Omni Java. Retrieved 7 May 2021, from https://www.omnijava.com/2018/04/24/password-confirmation-on-a-jsf-page-part-1-a-simple-model/