5.2C

Task 1. Analysis

a. User id length
b. Password complexity

JSF has a set of standard validator tasks that can do both of these very easily.

F:validateLength allows me to set the minimum & maximum size of a string.

F:validateRegex lets me write a regex pattern for the string to match.

These are both appropriate as they're standard JSF tags and will be extremely simple to implement.

c. Passwords Match

JSF also has a F:Validator tag which lets me point to a custom validator class to validate the input. A custom validator class is necessary here as I need to programmatically compare the two values of two input fields.

Task 2. Programming

Length validation:

```
<f:validateLength minimum="6" maximum="6" for="userid"/>
```

Complexity validation for password:

I used a regex testing website to write a regex that would only accept a 6 character string that contained at least 1 capital letter, 1 lower case letter, 1 number, and one special character (-, +, *)

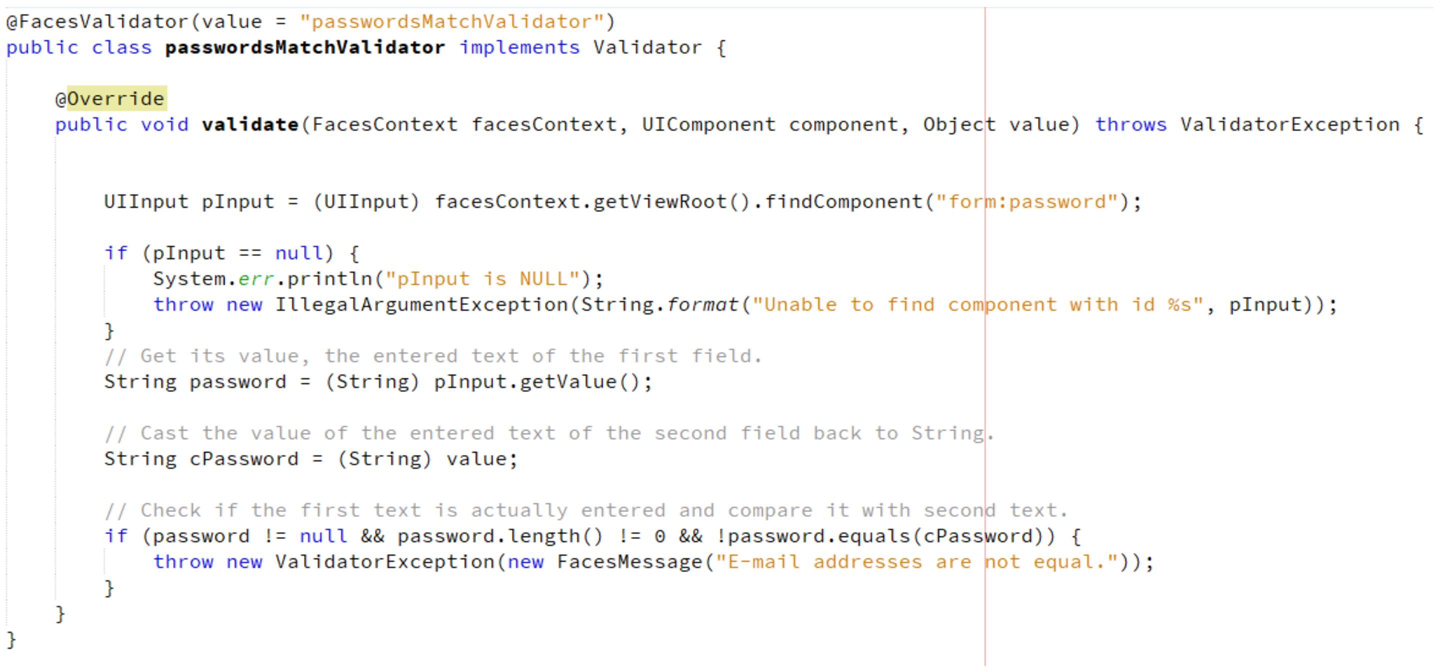
The 'at least 1' condition is met with the following pattern: (?=.*[])

?= means look ahead, and .* means any character except line break, and [] contains the character we want to search for. So its looking ahead for any character that exists in the square brackets.
This pattern is repeated for each of the characters we want at least one of.

```
<f:validateRegex for="password"
    pattern="^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[+\-*])[a-zA-Z0-9*+-]{6}$"  />
```

Passwords Match validation:

I created the following class to validate that the passwords match:



```java
@FacesValidator(value = "passwordsMatchValidator")
public class passwordsMatchValidator implements Validator {

    @Override
    public void validate(FacesContext facesContext, UIComponent component, Object value) throws ValidatorException {


        UIInput pInput = (UIInput) facesContext.getViewRoot().findComponent("form:password");

        if (pInput == null) {
            System.err.println("pInput is NULL");
            throw new IllegalArgumentException(String.format("Unable to find component with id %s", pInput));
        }
        // Get its value, the entered text of the first field.
        String password = (String) pInput.getValue();

        // Cast the value of the entered text of the second field back to String.
        String cPassword = (String) value;

        // Check if the first text is actually entered and compare it with second text.
        if (password != null && password.length() != 0 && !password.equals(cPassword)) {
            throw new ValidatorException(new FacesMessage("E-mail addresses are not equal."));
        }
    }
}
```

Its attached to the custom password field so it needs to grab the password input to compare.
To do this it uses the findcomponent method with the passwords id with its containers id in front of it. Here our form has the id of 'form' and our password has the id of 'password' so we find the component using 'form:password'.

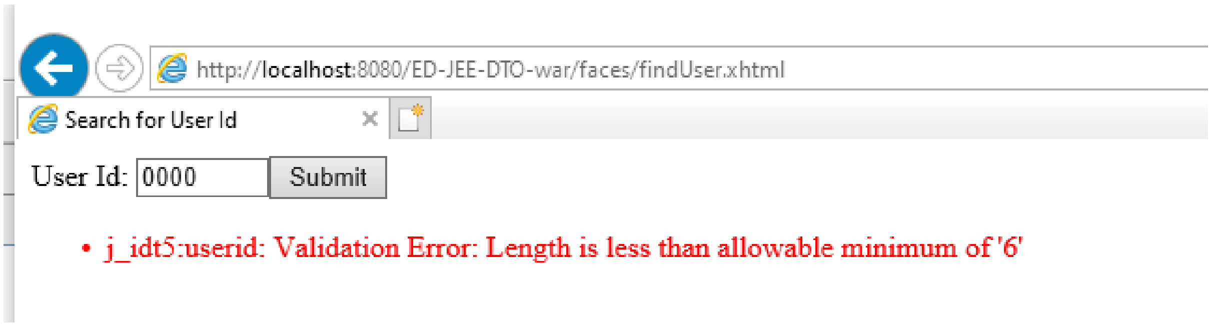
If its not null then it grabs the string value inside, and the string value in the confirm password field.
It then compares the two and throws an error if they are not the same.

To attach this to the field I use the following tag:

```xml
<f:validator validatorId="passwordsMatchValidator"/>
```

3. Test cases

Password length

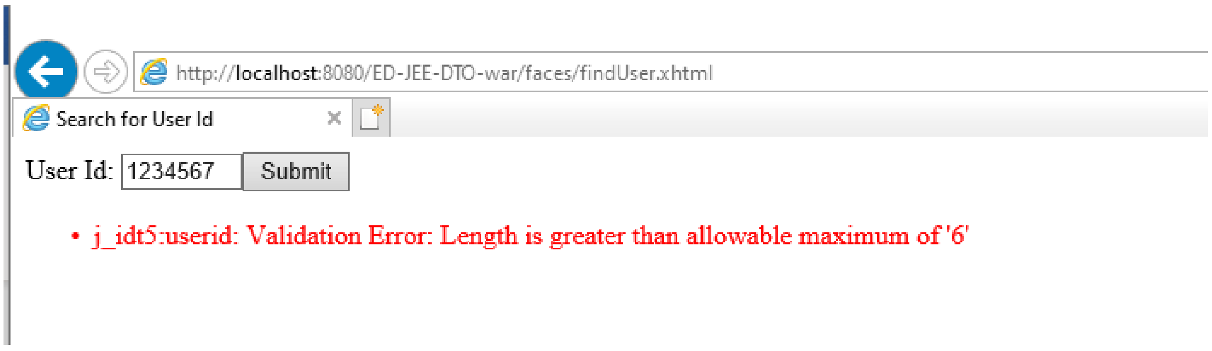


http://localhost:8080/ED-JEE-DTO-war/faces/findUser.xhtml

Search for User Id

User Id: 0000 Submit

- j_idt5:userid: Validation Error: Length is less than allowable minimum of '6'



http://localhost:8080/ED-JEE-DTO-war/faces/findUser.xhtml

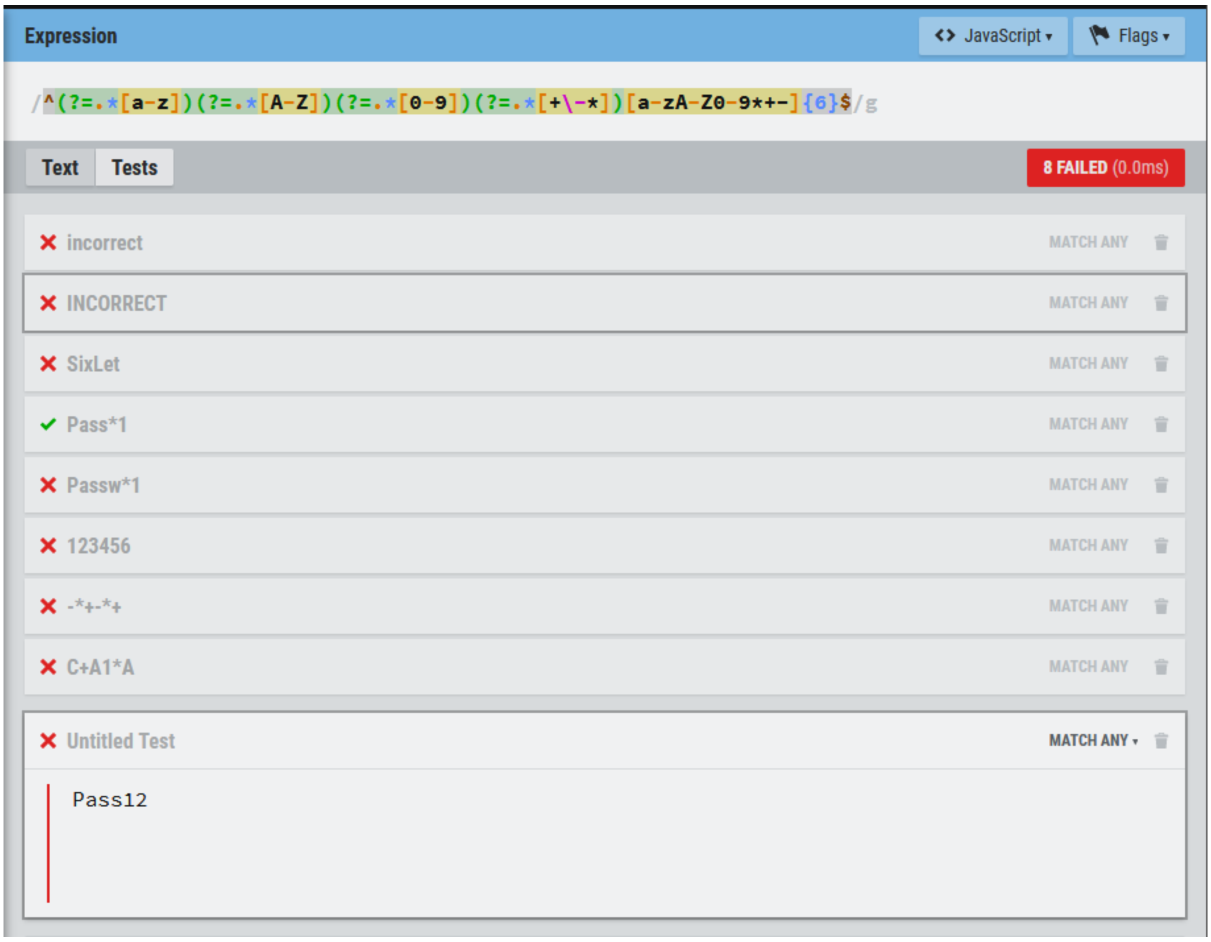
Search for User Id

User Id: 1234567 Submit

- j_idt5:userid: Validation Error: Length is greater than allowable maximum of '6'

Password Complexity:

To test if my regex works, I used the tester on the website. https://regexr.com/



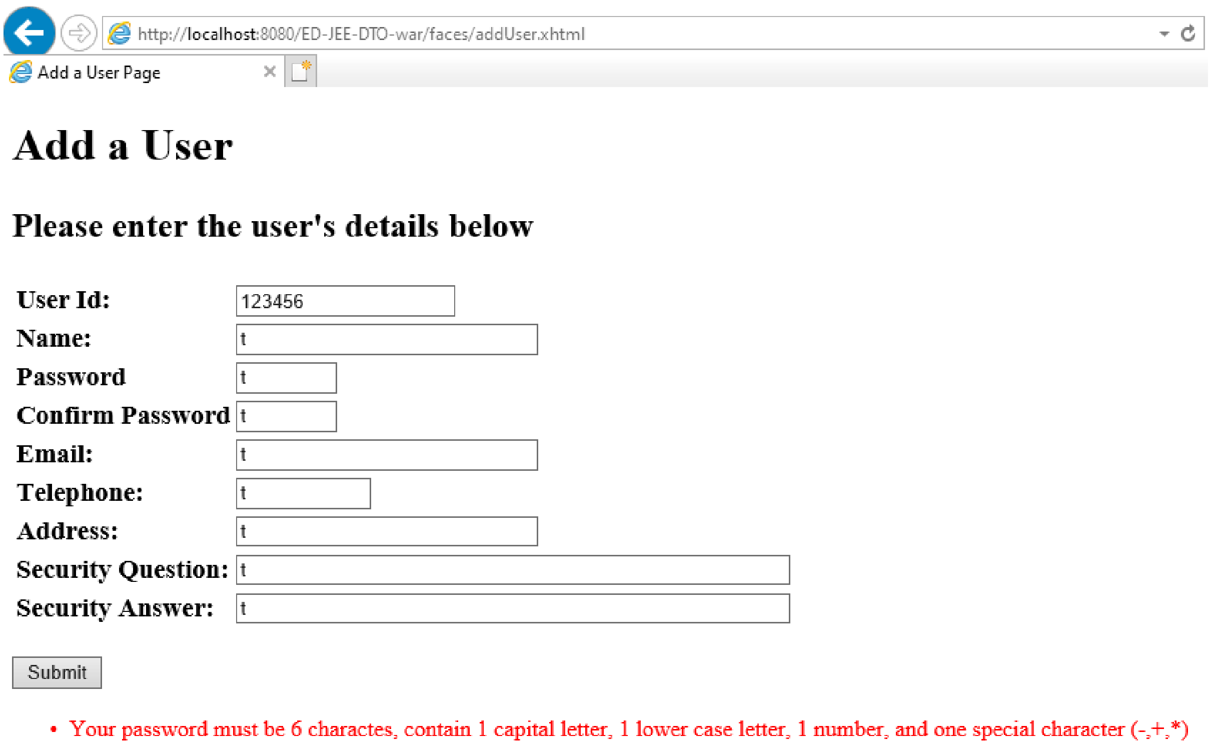
Expression

```
/^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[+\-*])[a-zA-Z0-9*+-]{6}$/g
```

Text | Tests — 8 FAILED (0.0ms)

- ✗ incorrect — MATCH ANY
- ✗ INCORRECT — MATCH ANY
- ✗ SixLet — MATCH ANY
- ✓ Pass*1 — MATCH ANY
- ✗ Passw*1 — MATCH ANY
- ✗ 123456 — MATCH ANY
- ✗ -*+-*+ — MATCH ANY
- ✗ C+A1*A — MATCH ANY
- ✗ Untitled Test — MATCH ANY

```
Pass12
```



http://localhost:8080/ED-JEE-DTO-war/faces/addUser.xhtml

Add a User Page

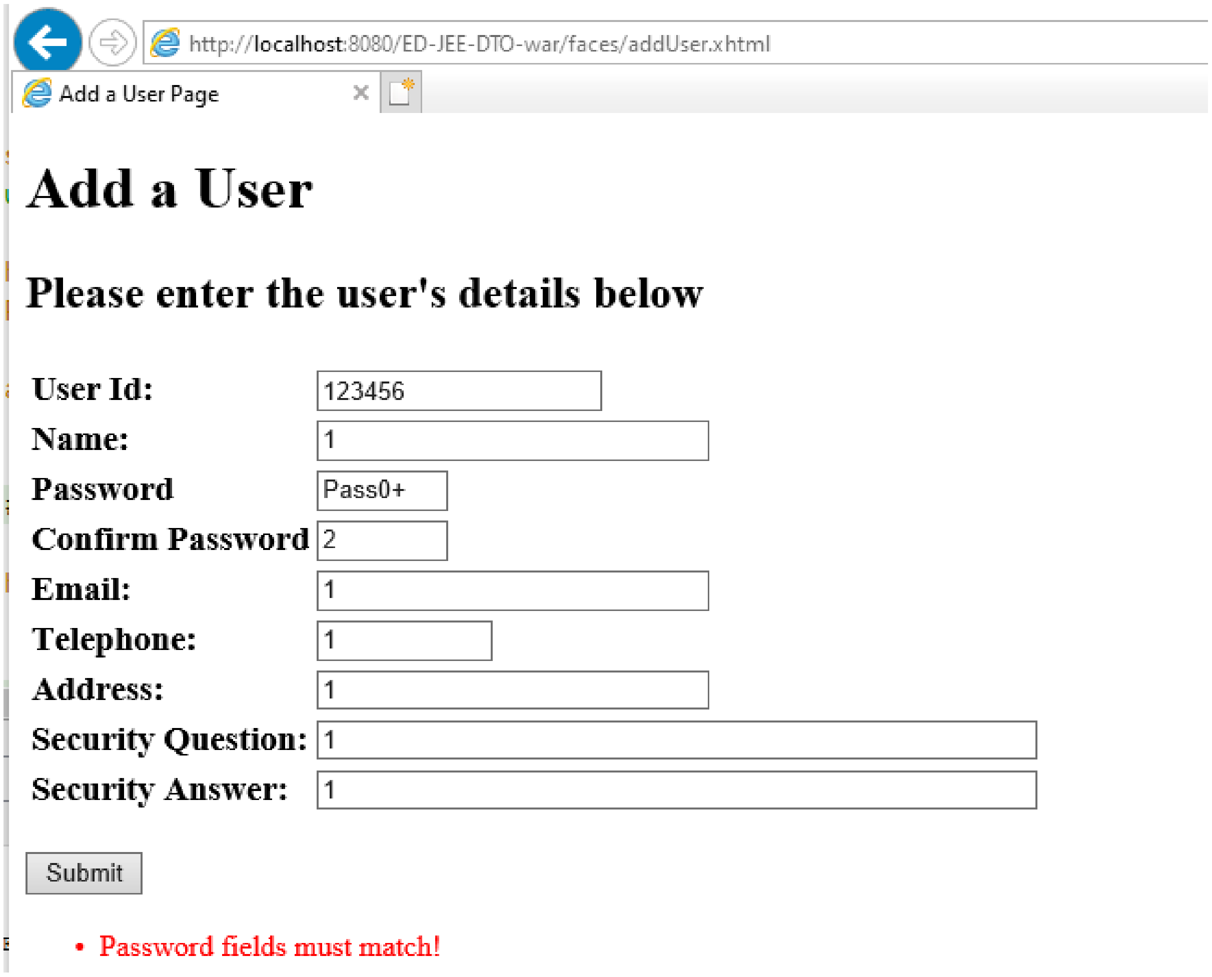
Add a User

Please enter the user's details below

User Id:	123456
Name:	t
Password	t
Confirm Password	t
Email:	t
Telephone:	t
Address:	t
Security Question:	t
Security Answer:	t

Submit

- Your password must be 6 charactes, contain 1 capital letter, 1 lower case letter, 1 number, and one special character (-,+,*) Passwords Matching:



Add a User

Please enter the user's details below

User Id:	123456
Name:	1
Password	Pass0+
Confirm Password	2
Email:	1
Telephone:	1
Address:	1
Security Question:	1
Security Answer:	1

- Password fields must match!

Task 4. Learning

A quick google search led me to this page: https://www.tutorialspoint.com/jsf/jsf_validation_tags.htm
From this I quickly implemented the length and regex validation.
Then for the custom validator I found this stackoverflow post https://stackoverflow.com/questions/2909021/jsf-2-0-validate-equality-of-2-inputsecret-fields-confirm-password-without-wr and basically copied the implementation.

Task 5. Research

Apparently it is possible to have multiple tags for each. https://stackoverflow.com/questions/4980616/multiple-validators-for-one-input