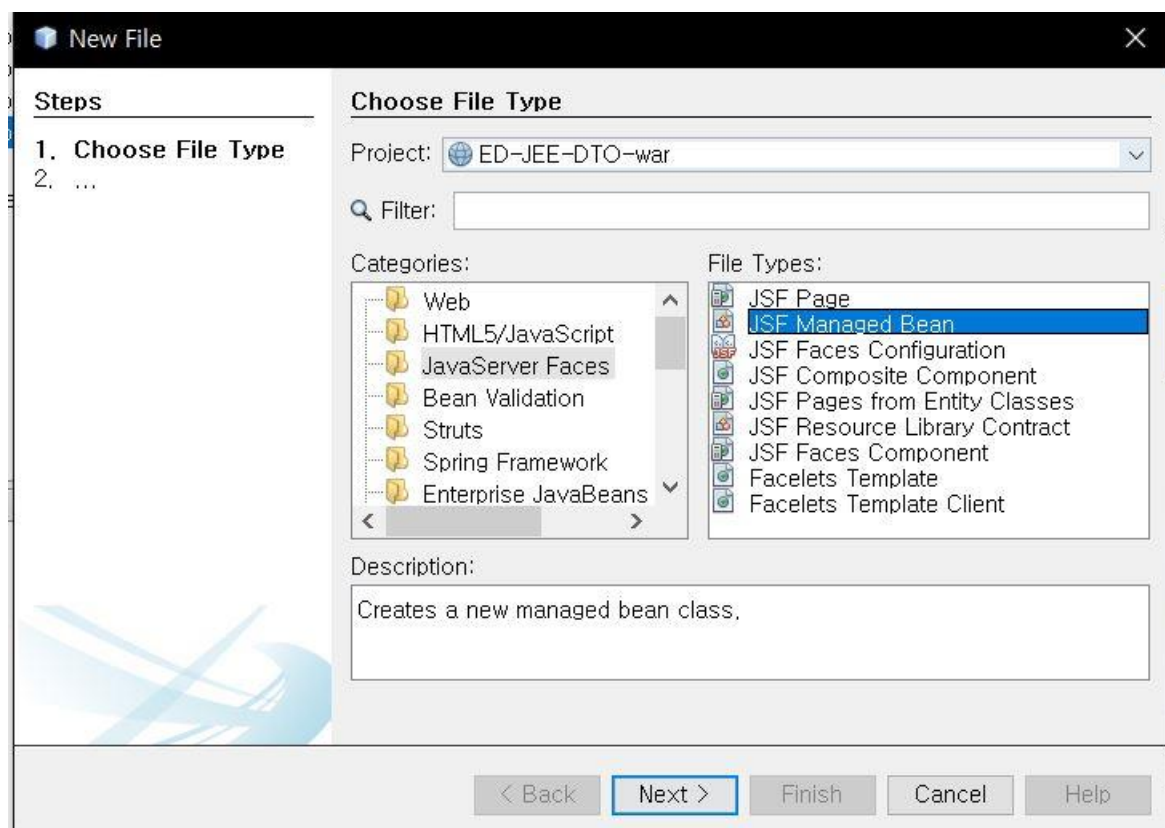
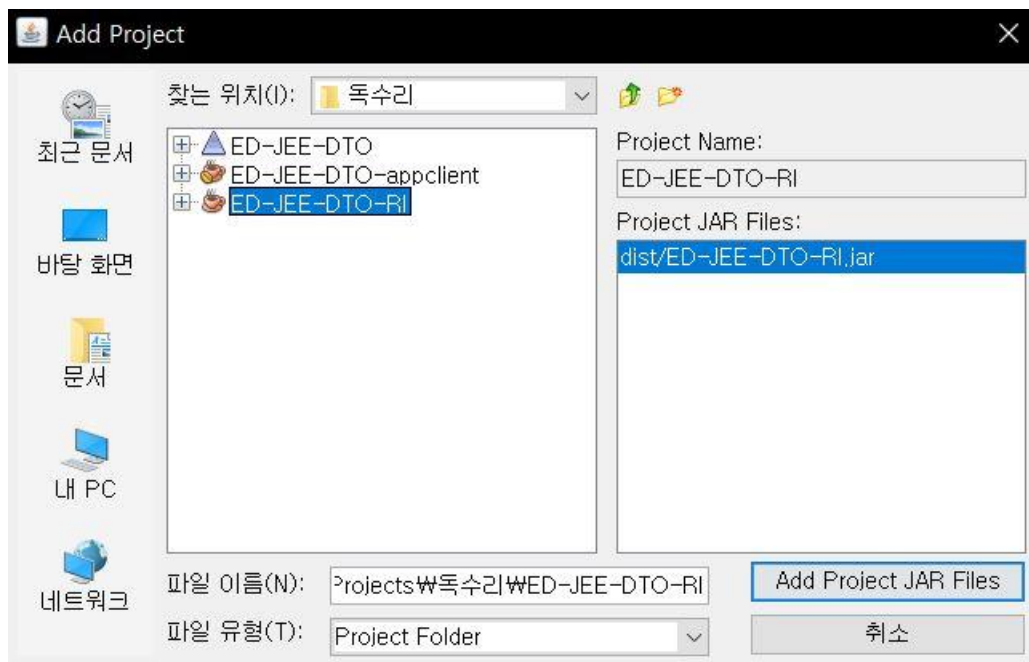


Task 1



New JSF Managed Bean

Steps

1. Choose File Type

2. Name and Location

Name and Location

Class Name: MyuserManagedBean

Project: ED-JEE-DTO-war

Location: Source Packages

Package: web

Created File: I:\JEE-DTO-war\src\java\web\MyuserManagedBean.java

☐ Add data to configuration file

Configuration File:

Name: myuserManagedBean

Scope: request

< Back

Next >

Finish

Cancel

Help

Call Enterprise Bean

Select an enterprise bean from open projects.

ED-JEE-DTO-ejb

MyuserFacade

ED-JEE-DTO-war

ED-JEE-DTO-war

ED-JEE-DTO-ejb

Reference Name: MyuserFacade

Referenced Interface: ☐ No Interface ☐ Local ☒ Remote

OK

Cancel

Help

```

private String userid;
private String name;
private String password;
private String cPassword; // for confirmed password field
private String email;
private String phone;
private String address;
private String secQn;
private String secAns;

/**
 * Creates a new instance of MyuserManagedBean
 */
public MyuserManagedBean() {

}

public MyuserFacadeRemote getMyuserFacade() {
    return myuserFacade;
}

public void setMyuserFacade(MyuserFacadeRemote myuserFacade) {
    this.myuserFacade = myuserFacade;
}

public String getUserid() {
    return userid;
}

public void setUserid(String userid) {
    this.userid = userid;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

```

```
public String getPassword() {  
    return password;  
}  
  
public void setPassword(String password) {  
    this.password = password;  
}  
  
public String getCPassword() {  
    return cPassword;  
}  
  
public void setCPassword(String cPassword) {  
    this.cPassword = cPassword;  
}  
  
public String getEmail() {  
    return email;  
}  
  
public void setEmail(String email) {  
    this.email = email;  
}  
  
public String getPhone() {  
    return phone;  
}  
  
public void setPhone(String phone) {  
    this.phone = phone;  
}  
  
public String getAddress() {  
    return address;  
}
```

```
    public void setAddress(String address) {  
        this.address = address;  
    }  
  
    public String getSecQn() {  
        return secQn;  
    }  
  
    public void setSecQn(String secQn) {  
        this.secQn = secQn;  
    }  
  
    public String getSecAns() {  
        return secAns;  
    }  
  
    public void setSecAns(String secAns) {  
        this.secAns = secAns;  
    }  
}
```

```

    public String addUser() {
        String result = "failure";
        /*
        * are all data entered valid?
        * and password the same as cPassword (case sensitive)
        * before calling the facade's createRecord() method
        */
        if (isValidUserId(userid) && isValidName(name)
            && isValidPassword(password) && isValidPassword(cPassword)
            && isValidEmail(email) && isValidPhone(phone)
            && isValidAddress(address) && isValidSecQn(secQn)
            && isValidSecAns(secAns) && password.equals(cPassword)) {
            MyuserDTO myuserDTO = new MyuserDTO(userid, name,
                password, email, phone, address, secQn, secAns);
            if (myuserFacade.createRecord(myuserDTO)) {
                result = "success";
            }
        }
        return result;
    }

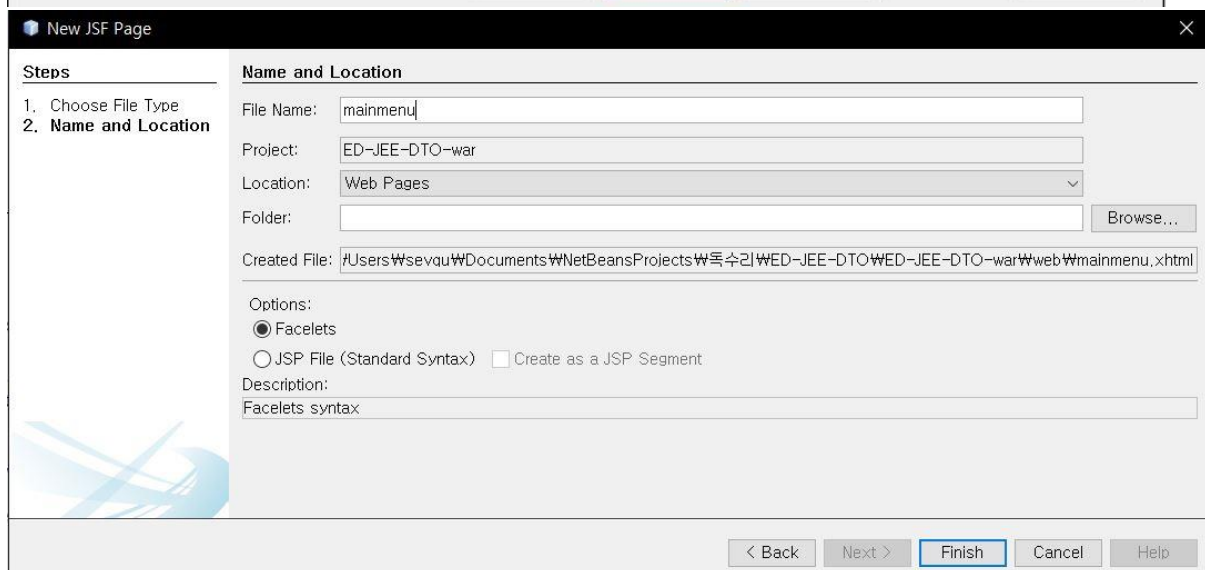
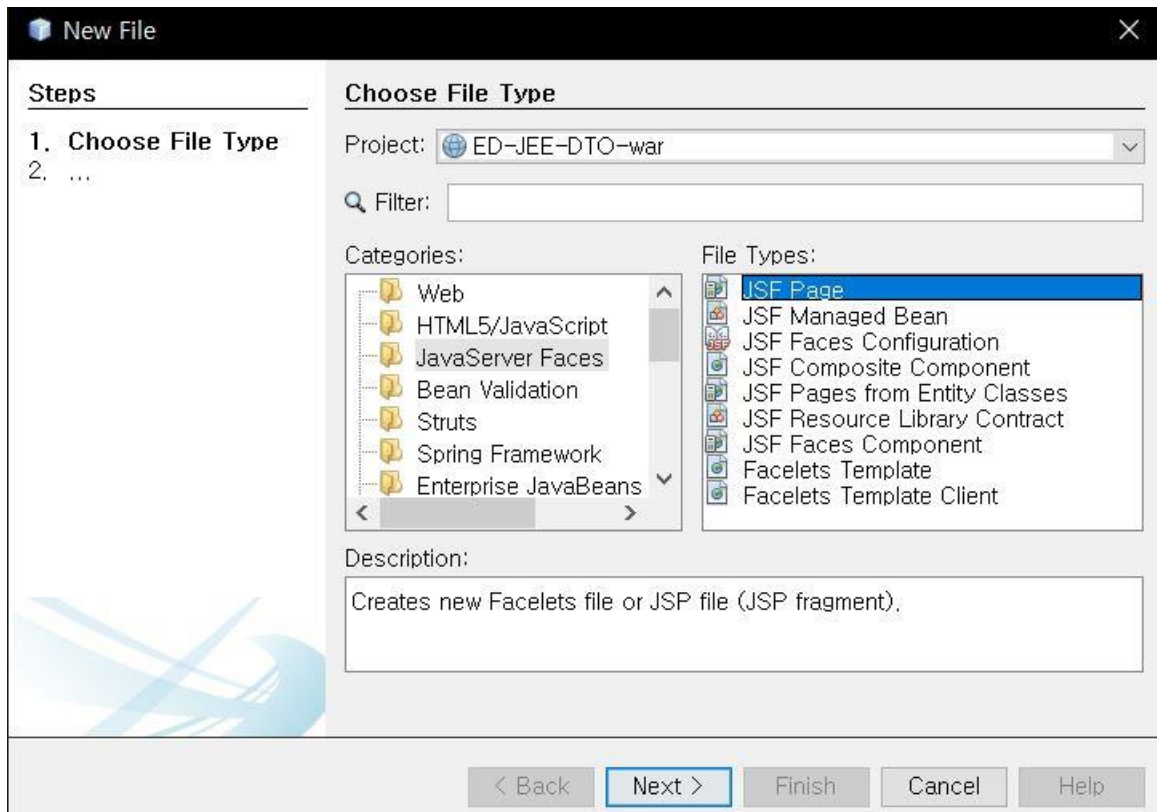
    /* Some basic checking, complicated checking can be done later
    * not a good way of doing this
    * Should use JSF's validator method to do this - left as C task
    */
    public boolean isValidUserId(String userid) {
        return (userid != null);
    }

    public boolean isValidName(String name) {
        return (name != null);
    }

    public boolean isValidPassword(String password) {
        return (password != null);
    }

```

```
1 public boolean isValidUserId(String userid) {  
2     return (userid != null);  
3 }  
  
4 public boolean isValidName(String name) {  
5     return (name != null);  
6 }  
7  
8 public boolean isValidPassword(String password) {  
9     return (password != null);  
10 }  
  
11 public boolean isValidEmail(String email) {  
12     return (email != null);  
13 }  
14  
15 public boolean isValidPhone(String phone) {  
16     return (phone != null);  
17 }  
18  
19 public boolean isValidAddress(String address) {  
20     return (address != null);  
21 }  
22  
23 public boolean isValidSecQn(String secQn) {  
24     return (secQn != null);  
25 }  
26  
27 public boolean isValidSecAns(String secAns) {  
28     return (secAns != null);  
29 }
```




```

<h:head>
  <title>Main Menu</title>
</h:head>
<h:body>
  <h:form>
    <h1>
      <h:outputText value="Welcome to Myuser Web Application!"/>
    </h1>

    <h2>
      <h:outputText value="Please select one of the following options"/>
    </h2>
    <h3>
      <ol>
        <li><a href="addUser.xhtml">Add a new user</a></li>
        <li><a href="undercon.html">Display a user</a></li>
        <li><a href="undercon.html">Edit a user</a></li>
        <li><a href="undercon.html">Delete a user</a></li>
      </ol>
    </h3>
  </h:form>
</h:body>

```

New JSF Page

Steps

1. Choose File Type

2. Name and Location

Name and Location

File Name:

Project:

Location:

Folder:

Created File:

Options:

☒ Facelets

☐ JSP File (Standard Syntax) ☐ Create as a JSP Segment

Description:

< Back

Next >

Finish

Cancel

Help

```

3  <h:head>
    <title>Add a User Page</title>
-  </h:head>
3  <h:body>
    <h1>
        Add a User
-    </h1>
3    <h2>
        <p>Please enter the user's details below</p>
-    </h2>
3    <h3>
        <h:form>
3            <h:panelGrid columns="2">
3                <h:outputText value="User Id: "/>
3                <h:inputText id="userid" value="#{myuserManagedBean.userid}"
                    required="true"
                    requiredMessage="The userid field cannot be empty!"
                    size="6" />
-                <h:outputText value="Name: "/>
3                <h:inputText id="name" value="#{myuserManagedBean.name}"
                    required="true"
                    requiredMessage="The name field cannot be empty!"
                    size="30"/>
-                <h:outputText value="Password "/>
3                <h:inputText id="password" value="#{myuserManagedBean.password}"
                    required="true"
                    requiredMessage="The password field cannot be empty!"
                    size="6"/>
-                <h:outputText value="Confirm Password "/>
3                <h:inputText id="cPassword" value="#{myuserManagedBean.cPassword}"
                    required="true"
                    requiredMessage="The confirm password field cannot be empty!"
                    size="6"/>
-                <h:outputText value="Email: "/>
3                <h:inputText id="email" value="#{myuserManagedBean.email}"
                    required="true"

```

```

<h:inputText id="email" value="#{myuserManagedBean.email}"
    required="true"
    requiredMessage="The email field cannot be empty!"
    size="30" />
<h:outputText value="Telephone: " />
<h:inputText id="phone" value="#{myuserManagedBean.phone}"
    required="true"
    requiredMessage="The telephone field cannot be empty!"
    size="10" />
<h:outputText value="Address: " />
<h:inputText id="address" value="#{myuserManagedBean.address}"
    required="true"
    requiredMessage="The email field cannot be empty!"
    size="30" />
<h:outputText value="Security Question: " />
<h:inputText id="secQn" value="#{myuserManagedBean.secQn}"
    required="true"
    requiredMessage="The security question field cannot be empty!"
    size="60" />
<h:outputText value="Security Answer: " />
<h:inputText id="secAns" value="#{myuserManagedBean.secAns}"
    required="true"
    requiredMessage="The security answer field cannot be empty!"
    size="60" />
</h:panelGrid>
<p></p>
<h:commandButton id="submit" value="Submit"
    action="#{myuserManagedBean.addUser}" />
</h:form>
</h3>
</h:body>

```

New JSF Page

Steps

1. Choose File Type
2. Name and Location

Name and Location

File Name:

Project:

Location:

Folder:

Browse...

Created File:

Options:

☒ Facelets
☐ JSP File (Standard Syntax)

☐ Create as a JSP Segment

Description:

< Back

Next >

Finish

Cancel

Help

```

<h:head>
    <title>User Added</title>
</h:head>
<h:body>
    <h:form>
        <h1>
            User Added - Success
        </h1>
        <h2>
            User whose userid is
            <h:outputText value="#{myuserManagedBean.userid}"/>
            has been added to the system.
        </h2>
        <p></p>
        <h3>
            Back to <h:commandButton value="Main Menu" action="mainmenu.xhtml"/>
        </h3>
    </h:form>
</h:body>

```

New JSF Page

Steps

1. Choose File Type
2. Name and Location

Name and Location

File Name:

Project:

Location:

Folder:

Created File:

Options:

☒ Facelets
☐ JSP File (Standard Syntax) ☐ Create as a JSP Segment

Description:

Facelets syntax

< Back

Next >

Finish

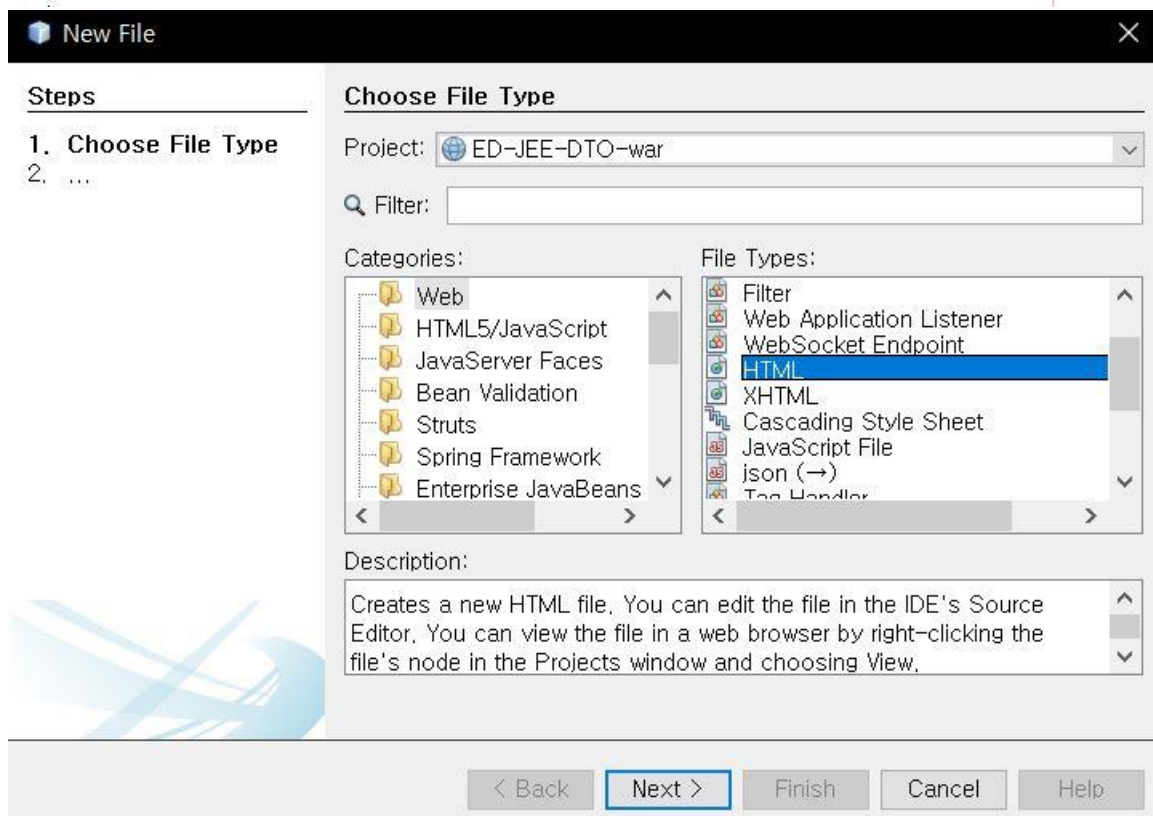
Cancel

Help

```

<h:head>
    <title>User Not Added</title>
</h:head>
<h:body>
    <h:form>
        <h1>
            User Added - Failure
        </h1>
        <h2>
            User whose userid is
            <h:outputText value="#{myuserManagedBean.userid}"/>
            cannot be added to the system.
        </h2>
        <p></p>
        <h3>
            Possibly there is an existing user with the same userid.
        </h3>
        <p></p>
        Back to <h:commandButton value="Main Menu" action="mainmenu.xhtml"/>
    </h:form>
</h:body>

```



New HTML

Steps

1. Choose File Type
2. Name and Location

Name and Location

HTML File Name:

Project:

Location:

Folder:

Created File:

< Back

Next >

Finish

Cancel

Help

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
  <title>Under Construction Page</title>
</head>
<body>
  <h1>
    This page is still under construction!
  </h1>
  Back to <a href="mainmenu.xhtml">Main Menu</a>.
```

New File

Steps

1. Choose File Type
2. ...

Choose File Type

Project:

Filter:

Categories:

Web
HTML5/JavaScript
JavaServer Faces
Bean Validation
Struts
Spring Framework
Enterprise JavaBeans

File Types:

JSF Page
JSF Managed Bean
JSF Faces Configuration
JSF Composite Component
JSF Pages from Entity Classes
JSF Resource Library Contract
JSF Faces Component
Facelets Template
Facelets Template Client

Description:
Creates a new faces-config.xml.

< Back

Next >

Finish

Cancel

Help

New JSF Faces Configuration

Steps

1. Choose File Type
2. Name and Location

Name and Location

File Name: faces-config

Project: ED-JEE-DTO-war

Folder: Browse...

Created File: DTOWED-JEE-DTO-war\web\WEB-INF\faces-config.xml

< Back

Next >

Finish

Cancel

Help

Browse Files

Folders:

web

WEB-INF

mainmenu.xhtml

addUserSuccess.xhtml

addUserFailure.xhtml

index.html

addUser.xhtml

undercon.html

Select File

Cancel

```
<navigation-rule>
  <from-view-id>/addUser.xhtml</from-view-id>
</navigation-rule>
```

Add Navigation Case

From View:

From Action:

From Outcome:

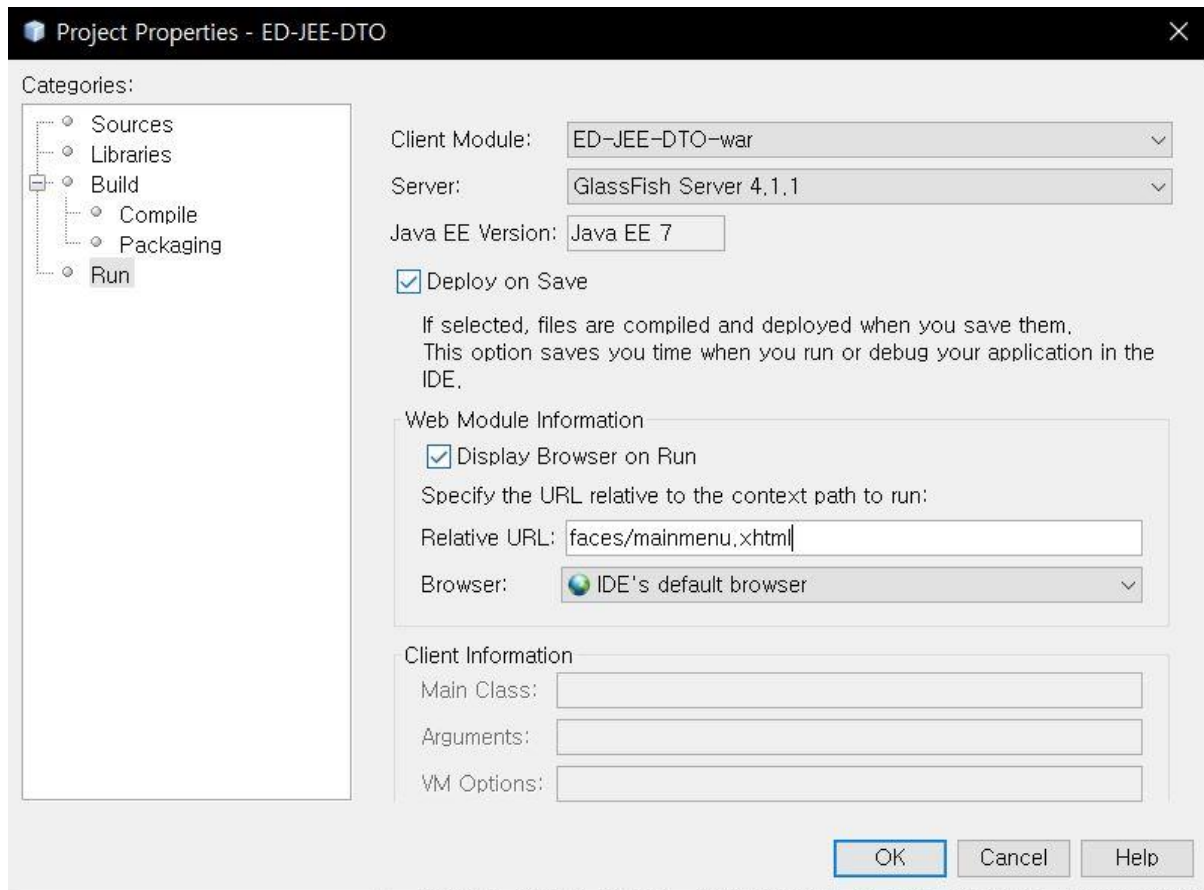
To View:

☐ Redirect

Rule Description:

Sun Oct 18 18:49:50 SGT 2020 : 기본 서버 보안 정책을 사용하여 보안 관리자가 설치됩니다.

Sun Oct 18 18:49:55 SGT 2020 : Apache Derby 네트워크 서버 - 10.10.2.0 - (1582446)이(가) 시작되어 1527 포트에서 접속을 승인할 준비가 되었습니다.



Welcome to Myuser Web Application!

Please select one of the following options

1. [Add a new user](#)
2. [Display a user](#)
3. [Edit a user](#)
4. [Delete a user](#)

Task 2

1) UserRecord

```
1 public String userRecord() {
    String result = "a";
    String a = userid;

    if (myuserFacade.getRecord(a) != null) {
        result = "b";
        name = myuserFacade.getRecord(a).getName();
        password = myuserFacade.getRecord(a).getPassword();
        email = myuserFacade.getRecord(a).getEmail();
        phone = myuserFacade.getRecord(a).getPhone();
        address = myuserFacade.getRecord(a).getAddress();
        secQn = myuserFacade.getRecord(a).getSecQn();
        secAns = myuserFacade.getRecord(a).getSecAns();
        return result;
    }

    return result;
}
```

2) Edit User - MyuserManagedBean

```
public String editUser() {
    String result = "failure";
    String smtpServer = "smtp.gmail.com";
    String from = "sevuqy@gmail.com";
    String to = "sevuqy@gmail.com";
    String subject = "Testing from gmail";
    String body = "your personal information update has been detected Check if you didn't make an update, send email to abc@gmail.com";
    String emailUser = from;
    String password22 = "xxxx";

    if (isValidUserId(userid) && isValidName(name)
        && isValidPassword(password) && isValidPassword(cPassword)
        && isValidEmail(email) && isValidPhone(phone)
        && isValidAddress(address) && isValidSecQn(secQn)
        && isValidSecAns(secAns) && password.equals(cPassword)) {
        MyuserDTO myuserDTO = new MyuserDTO(userid, name,
            password, email, phone, address, secQn, secAns);
        if (myuserFacade.updateRecord(myuserDTO)) {
            result = "success";
        }
    }

    try {
        Properties props = System.getProperties();
        // -- Attaching to default Session, or we could start a new one --
        props.put("mail.smtp.host", smtpServer);
        props.put("mail.smtp.port", 587);
        props.put("mail.smtp.auth", true);
        props.put("mail.smtp.starttls.enable", true);
        // -- prepare a password authenticator --
        MyAuthenticator myPA = new MyAuthenticator(emailUser, password22); // see MyAuthenticator class
        // get a session
        Session session = Session.getInstance(props, myPA);
        // -- Create a new message --
        Message msg = new MimeMessage(session);
        // -- Set the FROM and TO fields --
        msg.setFrom(new InternetAddress(from));
    } catch (Exception e) {
        result = "failure";
    }

    return result;
}
```

```

// -- Set the FROM and TO fields --
msg.setFrom(new InternetAddress(from));
msg.setRecipients(Message.RecipientType.TO, InternetAddress.parse(to, false));
// -- Set the subject and body text --
msg.setSubject(subject);
msg.setText(body);
// -- Set some other header information --
msg.setHeader("X-Mailer", "Gmail");
msg.setSentDate(new Date());
// -- Send the message --
Transport.send(msg, emailUser, password22);
System.out.println("Message sent OK.");
} catch (Exception ex) {
    ex.printStackTrace();
}
return result;
}

```

3) Edit user(Search Record) – JSF file

The screenshot shows the source code of the 'editUser.xhtml' file. The code is as follows:

```

<h:body>
<h1>
    edit User
</h1>
<h2>
<p>Please enter the user's details below</p>
</h2>
<h3>
<h:form>
    <h:panelGrid columns="2">
        <h:outputText value="User id: "/>
        <h:inputText id="userid" value="#{myuserManagedBean.userid}"
            required="true"
            validator="#{myuserManagedBean.validate}"
            requiredMessage="The userid field cannot be empty!"
            size="6" />
    </h:panelGrid>
    <h:commandButton id="submit" value="research"
        action="#{myuserManagedBean.userRecord}" />
    </h:form>
</h3>
</h:body>
</html>

```

4) Edit Record(Search Record) Success – JSF file

The screenshot shows the source code of the 'editRecordSuccess.xhtml' file. The code is as follows:

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
    <xhtml xmlns="http://www.w3.org/1999/xhtml"
        xmlns:h="http://www.w3.org/1999/xhtml"
        xmlns:f="http://www.w3.org/1999/xhtml" />
</!DOCTYPE>
<h:head>
    <title>User Record</title>
</h:head>
<h:body>
    <h:form>
        <h1>
            userid existed
        </h1>
        <h2>
            <h:outputText value="User id: "/>
            <h:outputText value="#{myuserManagedBean.userid}" />
            <h:inputHidden id="userid" value="#{myuserManagedBean.userid}" />
        </h2>
        <p></p>
        <h:outputText value="Name: "/>
        <h:inputText id="name" value="#{myuserManagedBean.name}"
            required="true"
        </h:inputText>
    </h:form>
</h:body>
</html>

```

```

<h:outputText value="password: " />
<h:inputText id="password" value="#{myuserManagedBean.password}"
    required="true"
    binding="#{passwordComponent}"

    requiredMessage="The password field cannot be empty!"
    size="6" />

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

<p></p>
<h:outputText value="Email: " />
<h:inputText id="email" value="#{myuserManagedBean.email}"
    required="true"
    requiredMessage="The email field cannot be empty!"
    size="30" />

<p></p>
<h:outputText value="Phone: " />
<h:inputText id="phone" value="#{myuserManagedBean.phone}"
    required="true"

    requiredMessage="The telephone field cannot be empty!"
    size="10" />

<p></p>
<h:outputText value="Address: " />
<h:inputText id="address" value="#{myuserManagedBean.address}"
    required="true"
    requiredMessage="The email field cannot be empty!"
    size="30" />

<p></p>
<h:outputText value="secQn: " />

```

```

        size="30" />
    <p></p>
    <h:outputText value="secQn: " />
    <h:inputText id="secQn" value="#{myuserManagedBean.secQn}"
        required="true"
        requiredMessage="The security question field cannot be empty!"
        size="60" />
    <p></p>
    <h:outputText value="secAns: " />
    <h:inputText id="secAns" value="#{myuserManagedBean.secAns}"
        required="true"
        requiredMessage="The security answer field cannot be empty!"
        size="60" />
</h2>
<p></p>
<h:commandButton id="submit" value="Update"
    action="#{myuserManagedBean.editUser}" />

</h:form>
<h:form>
    <p></p>
    <h3>
        Back to <h:commandButton value="Main Menu" action="mainmenu.xhtml" />
    </h3>
</h:form>
</h:body>
</html>

```

5) Edit Record Fail(Search Record) – JSF FILE



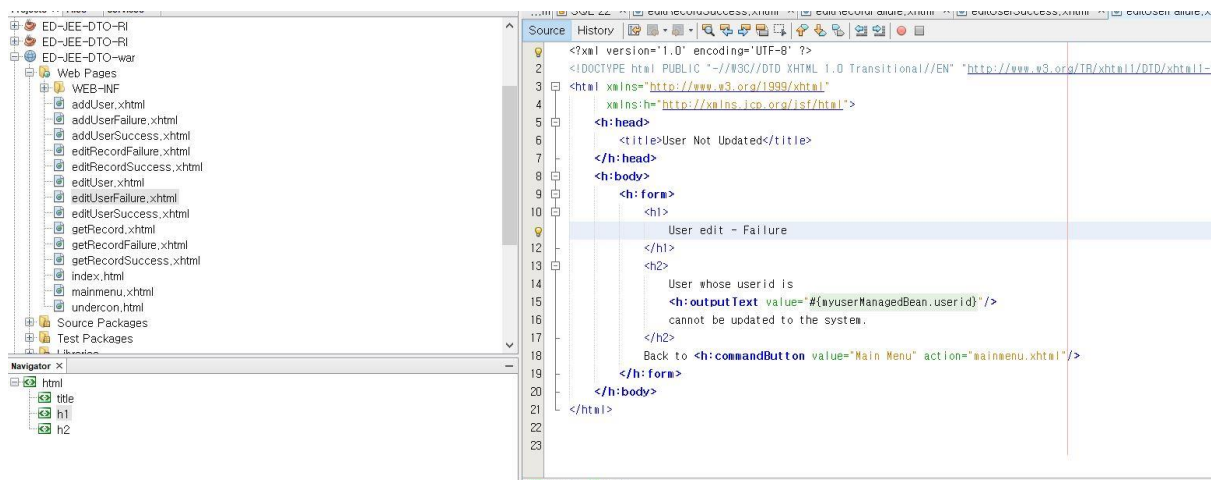
The screenshot shows a web application interface. On the left, a tree view lists various XHTML files under 'Web Pages' and 'WEB-INF'. The main area displays the content of a selected file, which is an XHTML page. The page structure is as follows:

```

<?xml version="1.0" encoding="UTF-8" ?>
<DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://xmlns.icp.org/jsf/html">
  <h:head>
    <title>Facelet Title</title>
  </h:head>
  <h:body>
    FAIL
  </h:body>
</html>

```

6) Edit User Failure – JSF FILE



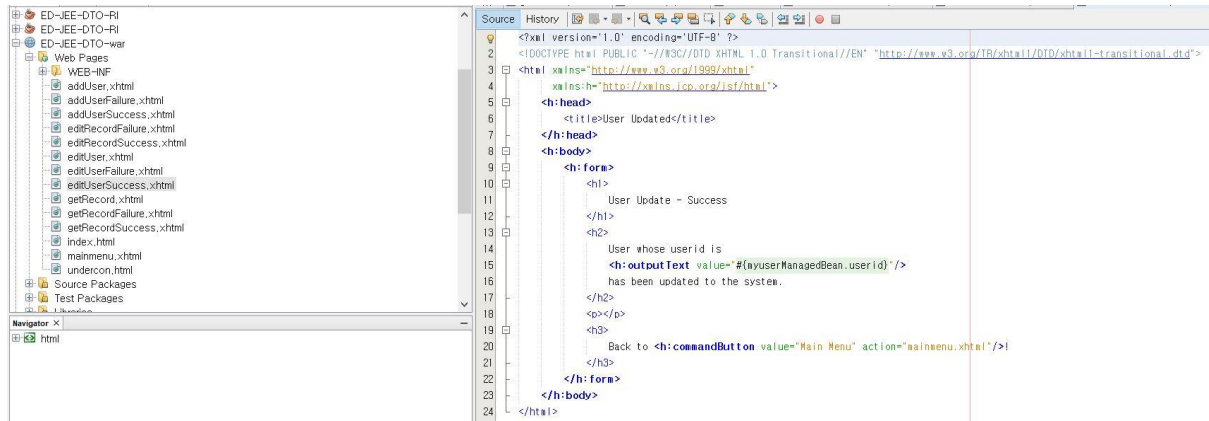
The screenshot shows a web application interface. On the left, a tree view lists various XHTML files under 'Web Pages' and 'WEB-INF'. The main area displays the content of a selected file, which is an XHTML page. The page structure is as follows:

```

<?xml version="1.0" encoding="UTF-8" ?>
<DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://xmlns.icp.org/jsf/html">
  <h:head>
    <title>User Not Updated</title>
  </h:head>
  <h:body>
    <h:form>
      <h1>
        User edit - Failure
      </h1>
      <h2>
        User whose userid is
        <h:outputText value="#{myuserManagedBean.userid}" />
        cannot be updated to the system.
      </h2>
      Back to <h:commandButton value="Main Menu" action="mainmenu.xhtml" />
    </h:form>
  </h:body>
</html>

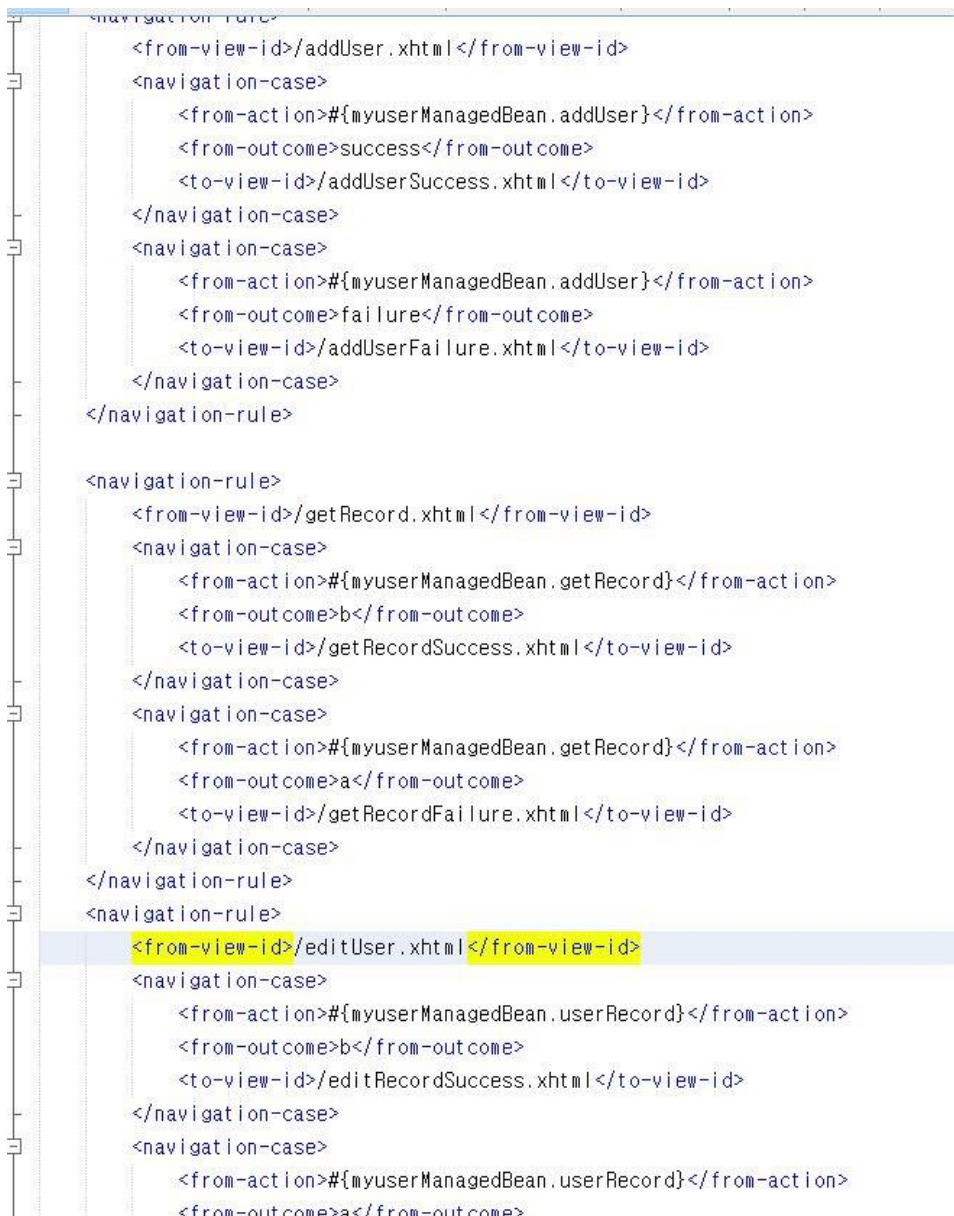
```


7) Edit User Success – JSF FILE



```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>User Updated</title>
  </h:head>
  <h:body>
    <h:form>
      <h1>
        User Update - Success
      </h1>
      <h2>
        User whose userid is
        <h:outputText value="#{myuserManagedBean.userid}"/>
        has been updated to the system.
      </h2>
      <p></p>
      <h3>
        Back to <h:commandButton value="Main Menu" action="mainmenu.xhtml"/>
      </h3>
    </h:form>
  </h:body>
</html>
```

8) Faces-config



```
<from-view-id>/addUser.xhtml</from-view-id>
<navigation-case>
  <from-action>#{myuserManagedBean.addUser}</from-action>
  <from-outcome>success</from-outcome>
  <to-view-id>/addUserSuccess.xhtml</to-view-id>
</navigation-case>
<navigation-case>
  <from-action>#{myuserManagedBean.addUser}</from-action>
  <from-outcome>failure</from-outcome>
  <to-view-id>/addUserFailure.xhtml</to-view-id>
</navigation-case>
</navigation-rule>

<navigation-rule>
  <from-view-id>/getRecord.xhtml</from-view-id>
  <navigation-case>
    <from-action>#{myuserManagedBean.getRecord}</from-action>
    <from-outcome>b</from-outcome>
    <to-view-id>/getRecordSuccess.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-action>#{myuserManagedBean.getRecord}</from-action>
    <from-outcome>a</from-outcome>
    <to-view-id>/getRecordFailure.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
<navigation-rule>
  <from-view-id>/editUser.xhtml</from-view-id>
  <navigation-case>
    <from-action>#{myuserManagedBean.userRecord}</from-action>
    <from-outcome>b</from-outcome>
    <to-view-id>/editRecordSuccess.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-action>#{myuserManagedBean.userRecord}</from-action>
    <from-outcome>a</from-outcome>
```

```
        <to-view-id>/editRecordFailure.xhtml</to-view-id>
    </navigation-case>
</navigation-rule>
<navigation-rule>
    <from-view-id>/editRecordSuccess.xhtml</from-view-id>
    <navigation-case>
        <from-action>#{myuserManagedBean.editUser}</from-action>
        <from-outcome>success</from-outcome>
        <to-view-id>/editUserSuccess.xhtml</to-view-id>
    </navigation-case>
    <navigation-case>
        <from-action>#{myuserManagedBean.editUser}</from-action>
        <from-outcome>failure</from-outcome>
        <to-view-id>/editUserFailure.xhtml</to-view-id>
    </navigation-case>
</navigation-rule>
</aces-config>
```

Task 3 : Testing

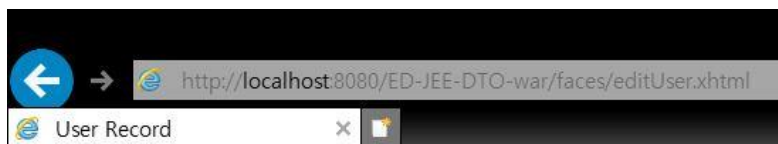


edit User

Please enter the user's details below

User Id:

research



userID existed

User Id: 000004

Name:

password:

Confirm Password

Email:

Phone:

Address:

secQn:

secAns:

Update

Back to [Main Menu](#) !



userID existed

User Id: 000004

Name:

password:

Confirm Password

Email:

Phone:

Address:

secQn:

secAns:

Back to



User Update - Success

User whose userid is 000004 has been updated to the system.

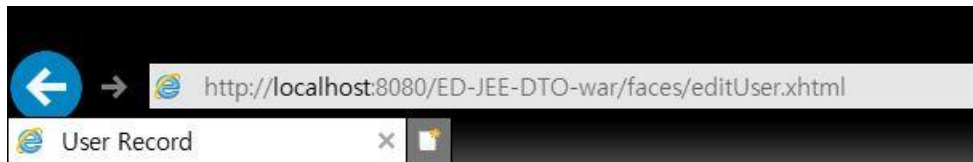
Back to

Testing from gmail 📧 받은편지함 ✕

sevquy@gmail.com

나에게 ▾

your personal information update has been detected Check if you didn't make an update, send email to abc@gmail.com



userID existed

User Id: 000004

Name: ✕

password:

Confirm Password

Email:

Phone:

Address:

secQn:

secAns:

Back to !



User Added - Failure

User whose userid is 000004 cannot be updated to the system.

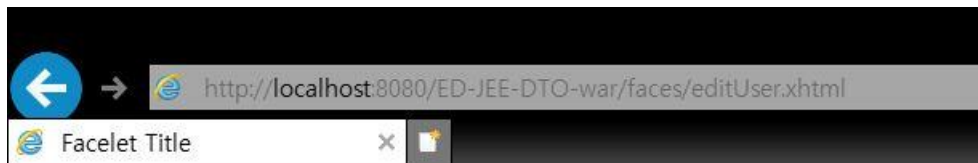
Back to [Main Menu](#)



edit User

Please enter the user's details below

User Id:



FAIL

Add a User

Please enter the user's details below

User Id:	<input type="text" value="000004"/>
Name:	<input type="text" value="test"/>
Password	<input type="text" value="test"/>
Confirm Password	<input type="text" value="test"/>
Email:	<input type="text" value="test@test.com"/>
Telephone:	<input type="text" value="1212"/> <input type="button" value="x"/>
Address:	<input type="text" value="test"/>
Security Question:	<input type="text" value="test"/>
Security Answer:	<input type="text" value="test"/>

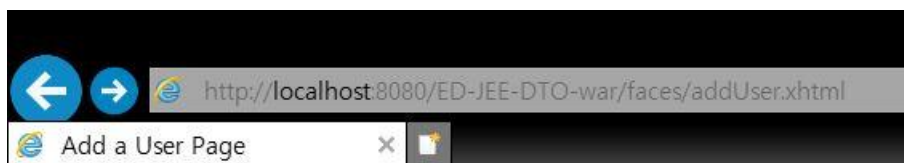


User Added - Failure

User whose userid is 000004 cannot be added to the system.

Possibly there is an existing user with the same userid.

Back to [Main Menu](#)



Add a User

Please enter the user's details below

User Id:	<input type="text" value="000010"/>
Name:	<input type="text" value="test"/>
Password	<input type="text" value="test"/>
Confirm Password	<input type="text" value="test"/>
Email:	<input type="text" value="test@test.com"/>
Telephone:	<input type="text" value="1212"/>
Address:	<input type="text" value="test"/>
Security Question:	<input type="text" value="test"/>
Security Answer:	<input type="text" value="test"/>

[Submit](#)



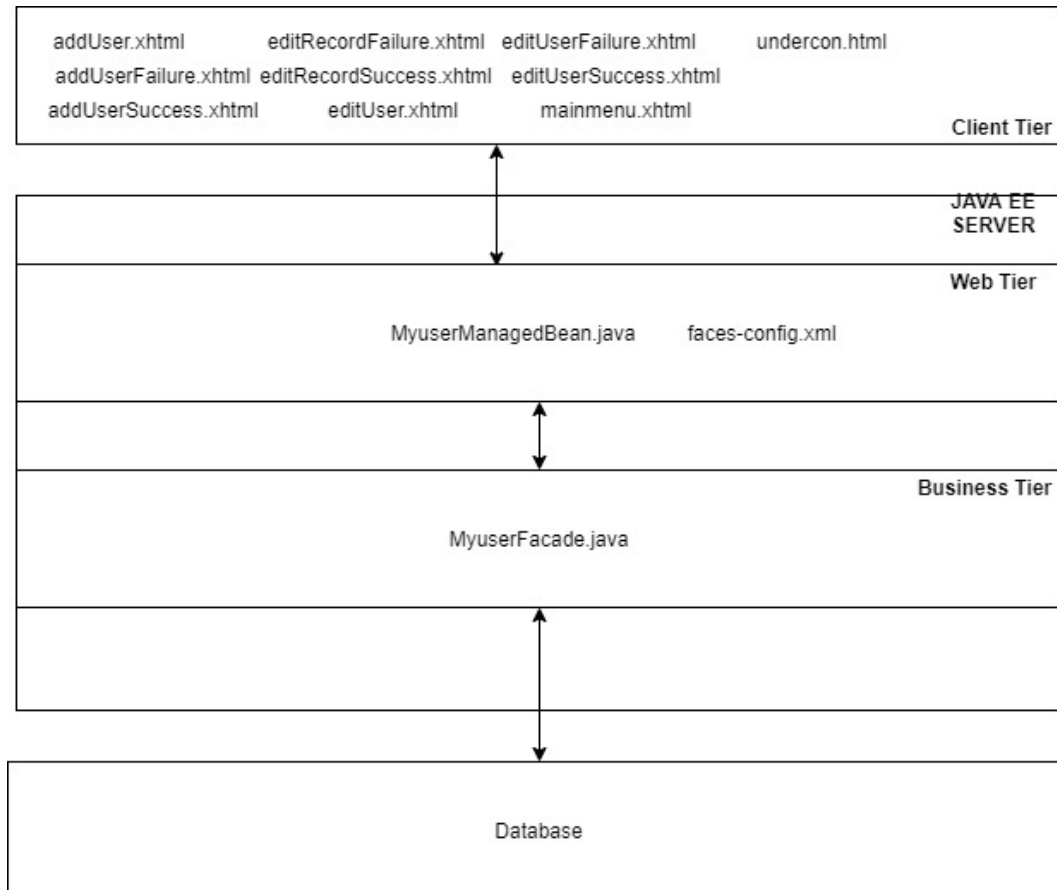
User Added - Success

User whose userid is 000010 has been added to the system.

Back to [Main Menu](#) !

#	USERID	NAME	PASSWORD	EMAIL	PHONE	ADDRESS
1	000001	Peter Smith	123456	psmith@swin.edu.au	9876543210	Swinburne EN510f
2	000002	abc	456789	a@google.com	134679	Swinburne EN5f
3	000003	Sheldon Cooper	345678	scooper@swin.edu.au	7654321098	Swinburne EN512a
4	000004	qwerty	456789	a@google.com	134679	Swinburne EN5f
5	000005	Harry Potter	567890	hpotter@swin.edu.au	6543210987	Swinburne EN514a
6	000006	David Lee	654321	dlee@swin.edu.au	0123456789	Swinburne EN510g
7	000007	a	qwe123	a@amail.com	123456	aa
8	000008	aa	aaaa	a@amail.com	123	aa
9	000456	aa	aa	aa	aa	aa
10	000010	test	test	test@test.com	1212	test

Task 4 : Design



Describe and discuss the role of each component in your diagram

Client Tier

1. addUser.xhtml : Allow user to enter the information to add users.
2. addUserSuccess.xhtml. : Print success message to users.
3. addUserFailure.xhtml : Print failure message to users.
4. editUser.xhtml : Allow User to enter the userid to find the record.
5. editRecordFailure.xhtml : Print failure message to users.
6. editRecordSuccess.xhtml : Print and accept User to enter the values to update.
7. editUserFailure.xhtml : Print failure message to users
8. editUserSuccess.xhtml : Print success message of update to users.
9. mainmenu.xhtml : Print mainmenu which navigate to update and add users.
10. undercon.html : Print under constructing message to users

JAVA EE SERVER

Web Tier

1. MyuserManagedBean.java : get user input values and deliver the result of success or failure to "faces-config.xml". Moreover, it sends/links values to "MyuserFacade.java" to do update, edit, get Record functions.
2. faces-config.xml : according to MyuserManagedBean.java's result(success/failure), decide to right webpage to print.

Business Tier

1. MyuserFacade.java : get values from MyuserManagedBean.java and perform update, get record, and add user.

EIS Tier

1. Database : information is saved in the database.

When designing your web pages, what decisions have been made by you (e.g. providing all information in one web page for user to key in the changes or providing only just one information per page)? What assumptions have been made in making such decisions? Explain and justify why you think this is a reasonable choice.

- I decided to print all information in one web page because if user have to type only one information per page, it could be more work on the developer to create multiple pages and for the users, if they made a mistake they have to return to first page to correct the values which is not convenient for users.

Task 5

Which email address(es) should you use to alert the user when their information has just been changed? The old one? Or the new one? What procedures / steps would you use to make sure the “update” is genuine? Remember to justify your answer.

- To prevent other users to update Email to steal/get Email alert, I think that the alert email needs to send to old email address. Moreover, to make sure the update is genuine, email need to include a button or confirmation code which pass “true” value to the database to agree with the request(update). If users don’t click the button or type the confirmation code, false value will be passed to database to cancel the update of database.

What are the things that you will change in your program to make this happen? In terms of business logic? In terms of the database? Remember to justify your answer. [You do not need to write a “new” program, just “talk” about it. Present your idea in a clear and concise manner.]

- Within Email, we need to add a button or validation code to verify the request is made by correct users. Moreover, to prevent data overwrite, we need to add concurrency on the request to prevent 1st request’s data is overwritten by 2nd request’s data to prevent it. Moreover, database need to be modified to accept the requests’ data such as extra field for name, userid etc.