UNIVERSITY of LIMERICK

OLLSCOIL LUIMNIGH

Web Based Application Design Group Project

Arjun Ramesh	16044215
Dakshina Murthy	16064879
Gowtham NR	16029143
Ruwaid Khateeb	16055314
Vivek Padmanaban chinnaraj	15113582

Contents

1 OWASP Vulnerabilities prevention technique used in the Project

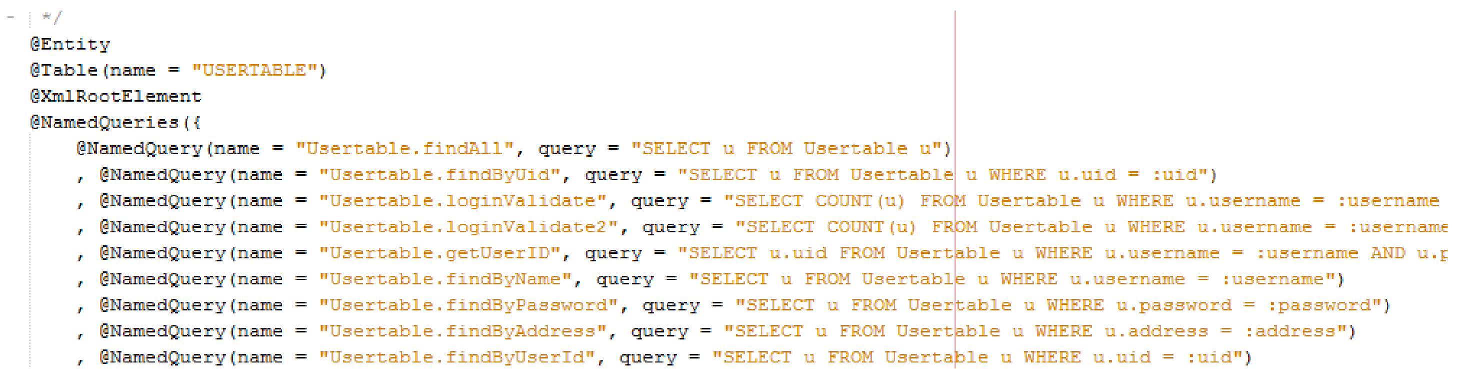
1.1 A1: Injection

We are using Named queries in our application which is static in nature and will be strong against any SQL injection.

Dynamic queries are prone to injection and hence we have not used any such queries in our application.

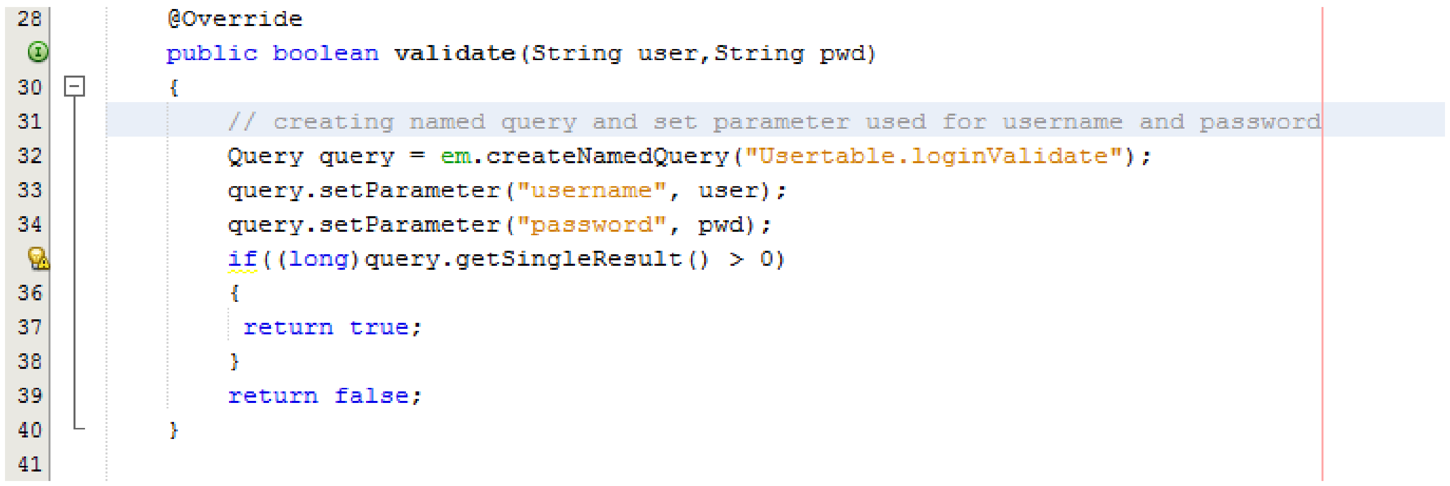
Named queries are more readable, maintainable as well as performant. Named queries in JPA are defined using the @NamedQuery annotation.

Below is the example from UserTable.java.



```
 */
@Entity
@Table(name = "USERTABLE")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Usertable.findAll", query = "SELECT u FROM Usertable u")
    , @NamedQuery(name = "Usertable.findByUid", query = "SELECT u FROM Usertable u WHERE u.uid = :uid")
    , @NamedQuery(name = "Usertable.loginValidate", query = "SELECT COUNT(u) FROM Usertable u WHERE u.username = :username
    , @NamedQuery(name = "Usertable.loginValidate2", query = "SELECT COUNT(u) FROM Usertable u WHERE u.username = :username
    , @NamedQuery(name = "Usertable.getUserID", query = "SELECT u.uid FROM Usertable u WHERE u.username = :username AND u.p
    , @NamedQuery(name = "Usertable.findByName", query = "SELECT u FROM Usertable u WHERE u.username = :username")
    , @NamedQuery(name = "Usertable.findByPassword", query = "SELECT u FROM Usertable u WHERE u.password = :password")
    , @NamedQuery(name = "Usertable.findByAddress", query = "SELECT u FROM Usertable u WHERE u.address = :address")
    , @NamedQuery(name = "Usertable.findByUserId", query = "SELECT u FROM Usertable u WHERE u.uid = :uid")
```

Parameters uses named variables that can be supplied values using the setParameter as shown below.



```
    @Override
    public boolean validate(String user,String pwd)
    {
        // creating named query and set parameter used for username and password
        Query query = em.createNamedQuery("Usertable.loginValidate");
        query.setParameter("username", user);
        query.setParameter("password", pwd);
        if((long)query.getSingleResult() > 0)
        {
         return true;
        }
        return false;
    }
```

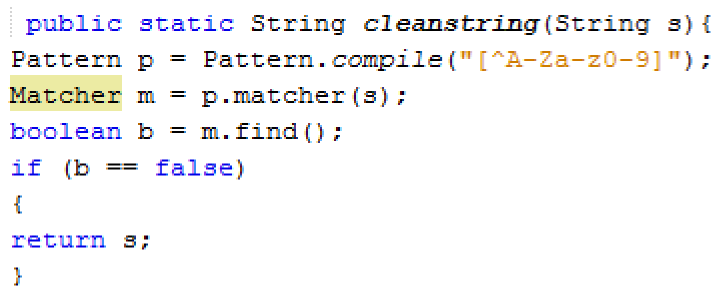
1.2 A3: Cross-Site Scripting (XSS)

We have used regex Pattern and Matcher to validate input from the user and if the invalid input is entered, it will be validated and cleaned before storing into database. We also used OWASP antisamy library to clean input from the user. Attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script.

We have used XSS cleanup in edit profile and admin product add, remove, increment and decrement functionalities.

Below is the code used for checking for pattern and ignore special characters like *!@#$%&<>? and special characters in the input which may break html page or have injection issues.

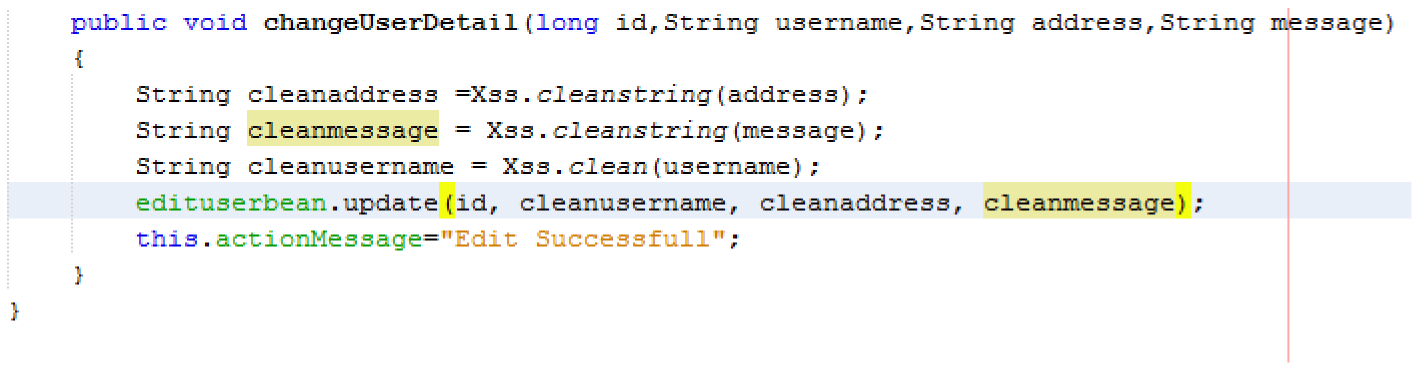
XSS. Java



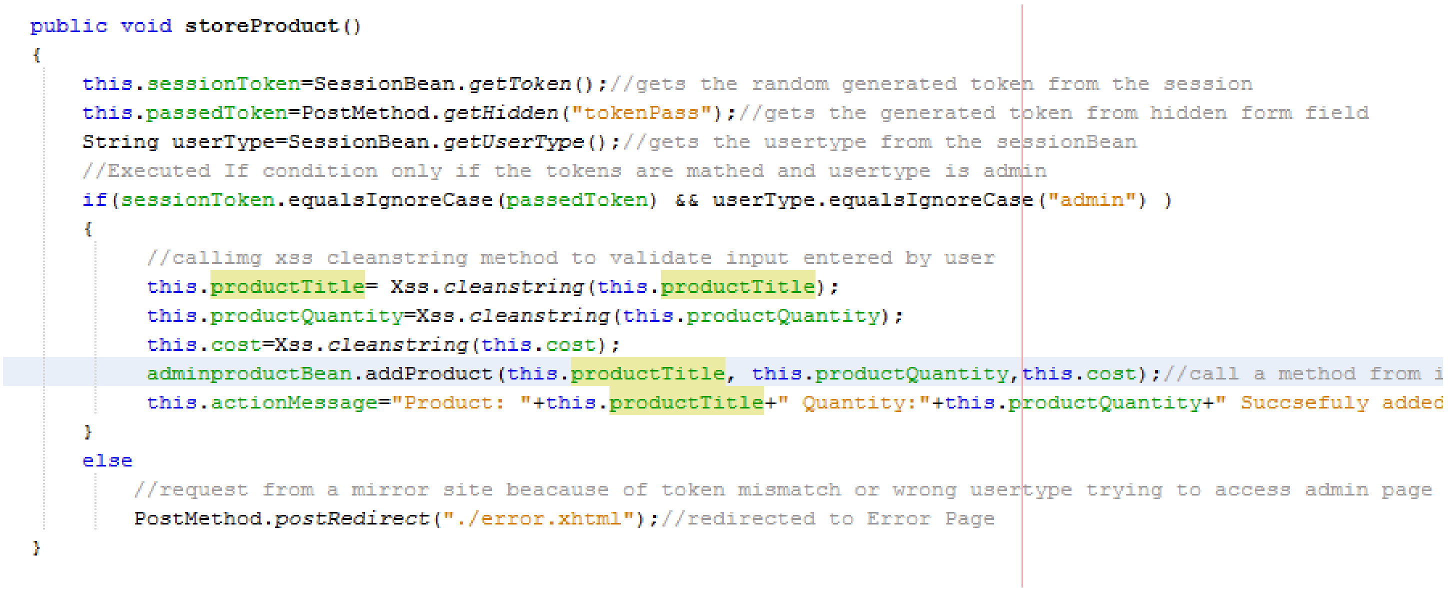
```
 public static String cleanstring(String s){
Pattern p = Pattern.compile("[^A-Za-z0-9]");
Matcher m = p.matcher(s);
boolean b = m.find();
if (b == false)
{
return s;
}
```

Below are the places where we have used XSS cleanstring method to avoid xss attack.

EditProfileBean.java and Admin.java



```
    public void changeUserDetail(long id,String username,String address,String message)
    {
        String cleanaddress =Xss.cleanstring(address);
        String cleanmessage = Xss.cleanstring(message);
        String cleanusername = Xss.clean(username);
        edituserbean.update(id, cleanusername, cleanaddress, cleanmessage);
        this.actionMessage="Edit Successfull";
    }
}
```



```
public void storeProduct()
{
    this.sessionToken=SessionBean.getToken();//gets the random generated token from the session
    this.passedToken=PostMethod.getHidden("tokenPass");//gets the generated token from hidden form field
    String userType=SessionBean.getUserType();//gets the usertype from the sessionBean
    //Executed If condition only if the tokens are mathed and usertype is admin
    if(sessionToken.equalsIgnoreCase(passedToken) && userType.equalsIgnoreCase("admin") )
    {
         //callimg xss cleanstring method to validate input entered by user
         this.productTitle= Xss.cleanstring(this.productTitle);
         this.productQuantity=Xss.cleanstring(this.productQuantity);
         this.cost=Xss.cleanstring(this.cost);
         adminproductBean.addProduct(this.productTitle, this.productQuantity,this.cost);//call a method from i
         this.actionMessage="Product: "+this.productTitle+" Quantity:"+this.productQuantity+" Succsefuly added
    }
    else
        //request from a mirror site beacause of token mismatch or wrong usertype trying to access admin page
        PostMethod.postRedirect("./error.xhtml");//redirected to Error Page
}
```

1.3 A7: Missing Function Level Access Control

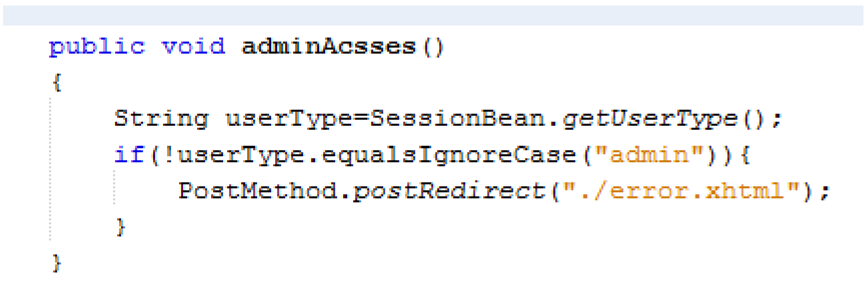
In our Project adminPanel.xhtml page has the admin functionalities and it can be accessed only by user type-admin. Also, before performing any admin related functionalities, user type is validated in each method

If the normal customer tries to access admin page it will be redirected to error page, this is how we are preventing missing function level access control.

Even if the normal user tries to change their login URL to http://localhost:8080/Project_Group12-war/faces/adminPanel.xhtml and tries to access, it will be redirected to error page.

Below is the code used to check user type access rights.

Admin.Java



```
public void adminAcsses()
{
    String userType=SessionBean.getUserType();
    if(!userType.equalsIgnoreCase("admin")){
        PostMethod.postRedirect("./error.xhtml");
    }
}
```

1.4 A8: Cross-Site Request Forgery (CSRF)

In our project, we have prevented CSRF attack using token transfer technique. In each page a random token will be generated for a session and token will be changed each time the page is redirected or refreshed.

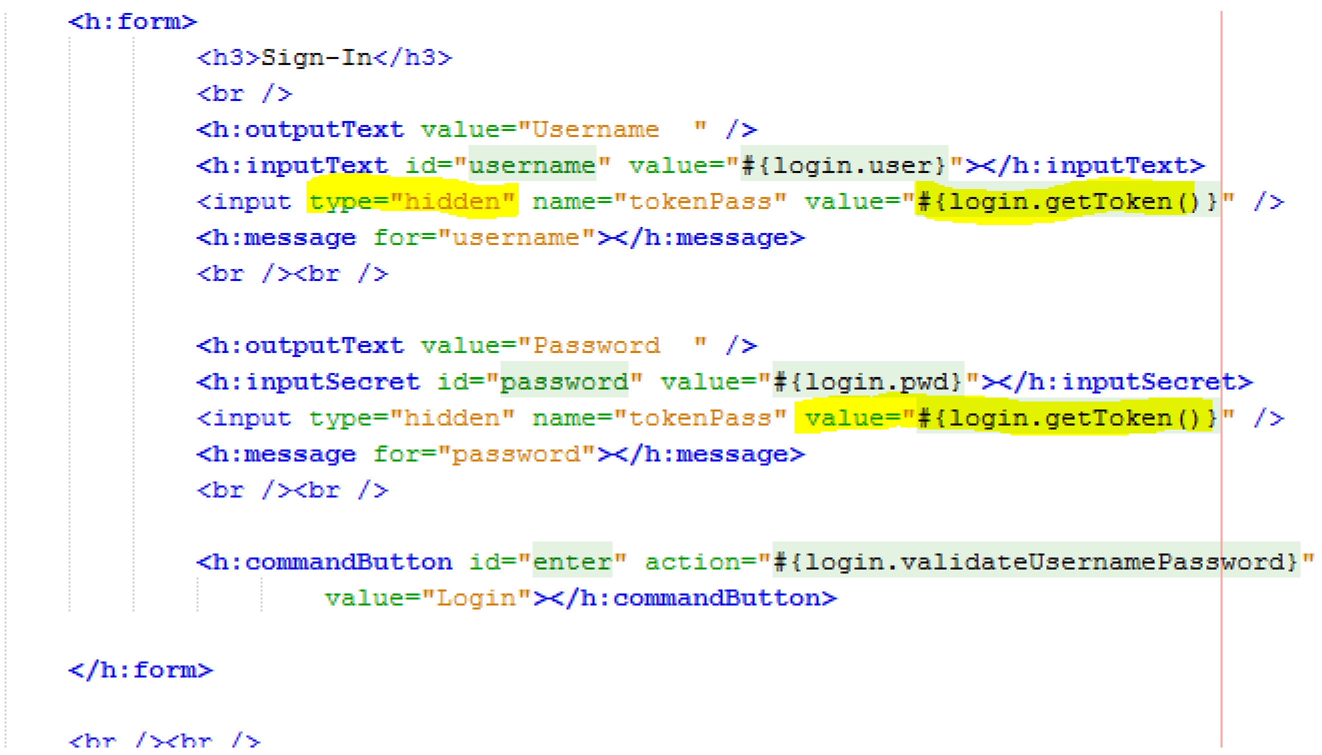
Token will be sent from each page in a hidden field and will be validated in the next page, if the token is mismatch then error page will be displayed.

When the user is logged in and tries to navigate to different page, attacker will send forged HTTP request to another site which will compromise user's session information and cookie details.

Cross-Site Request Forgery (CSRF) is an attack that occurs when a malicious web site or program that causes a user's web browser to perform an unwanted action on a trusted site for which the user is currently authenticated.

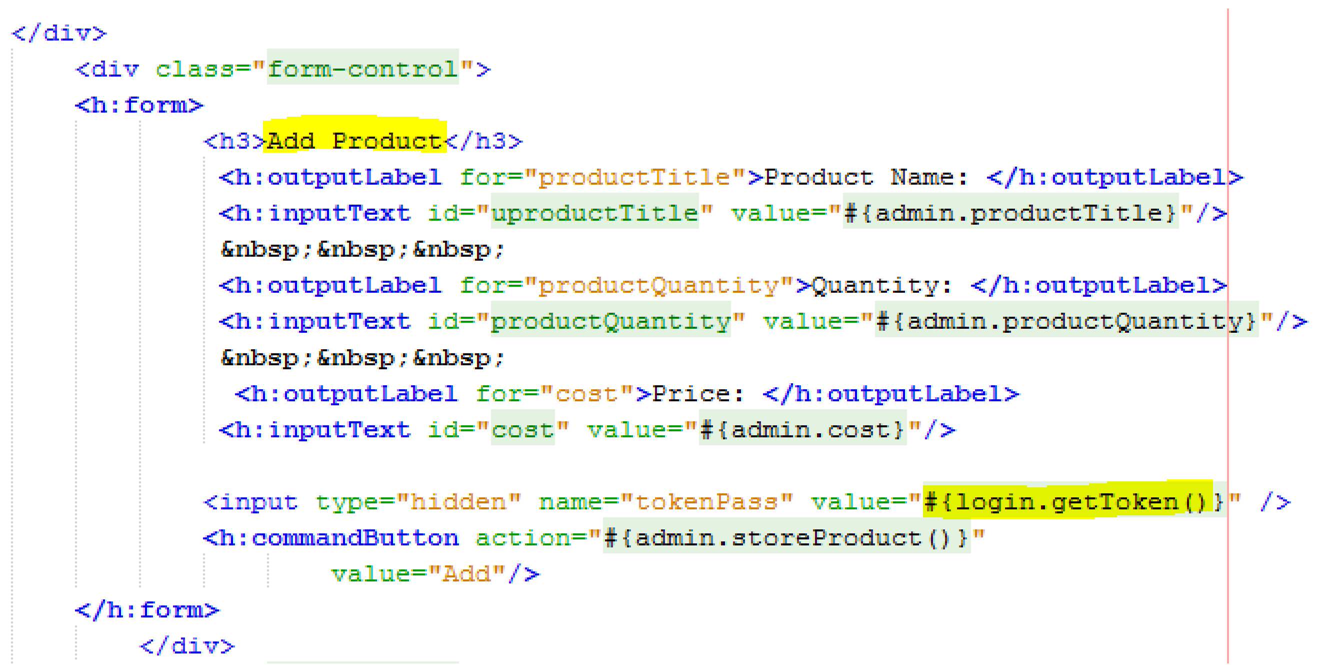
Below are the XHTML pages where we are passing generated token to next redirected page.

login.xhtml



```
<h:form>
        <h3>Sign-In</h3>
        <br />
        <h:outputText value="Username  " />
        <h:inputText id="username" value="#{login.user}"></h:inputText>
        <input type="hidden" name="tokenPass" value="#{login.getToken()}" />
        <h:message for="username"></h:message>
        <br /><br />

        <h:outputText value="Password  " />
        <h:inputSecret id="password" value="#{login.pwd}"></h:inputSecret>
        <input type="hidden" name="tokenPass" value="#{login.getToken()}" />
        <h:message for="password"></h:message>
        <br /><br />

        <h:commandButton id="enter" action="#{login.validateUsernamePassword}"
                value="Login"></h:commandButton>

</h:form>

<br /><br />
```

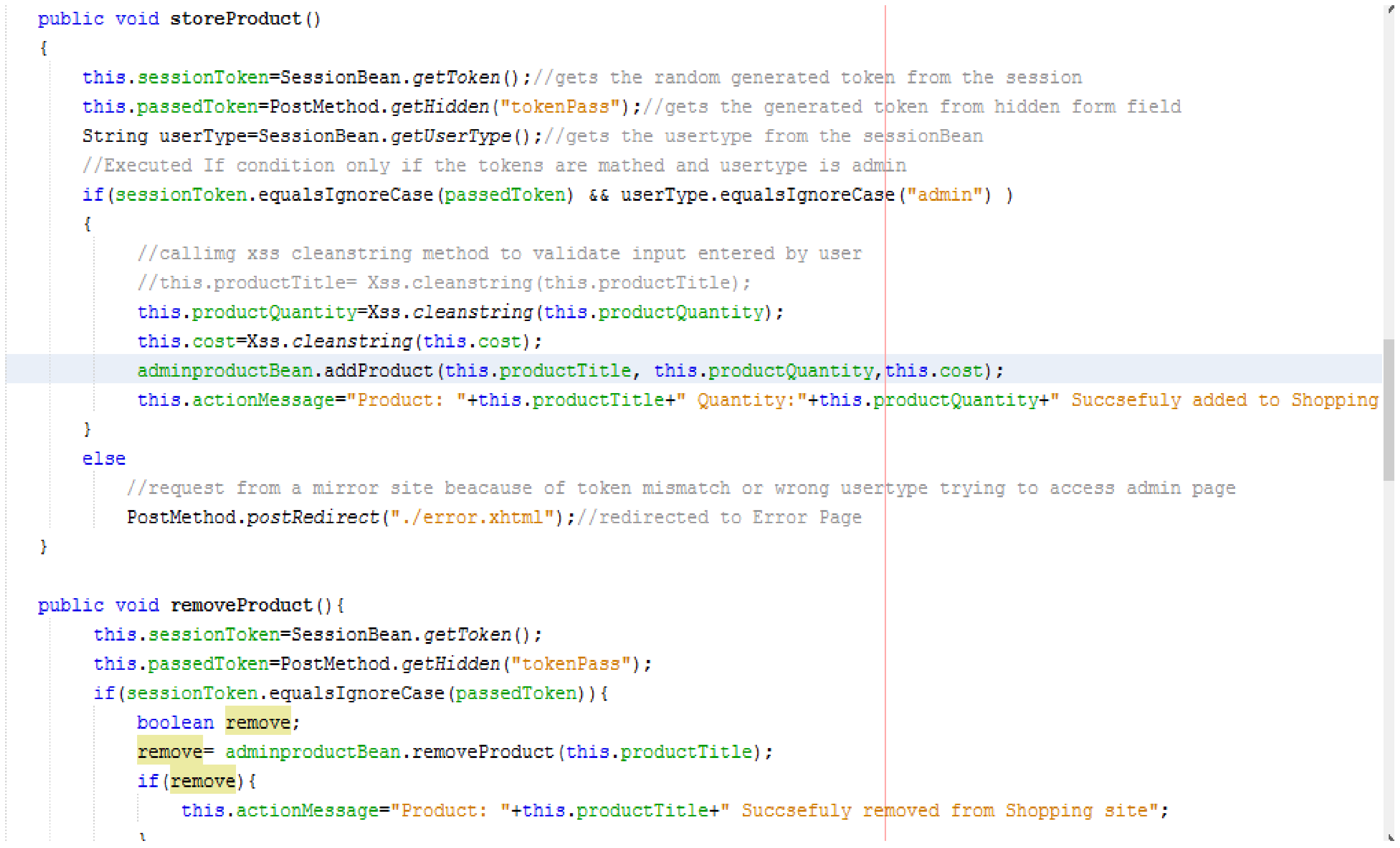
adminPanel.xhtml



```
</div>
    <div class="form-control">
    <h:form>
            <h3>Add Product</h3>
             <h:outputLabel for="productTitle">Product Name: </h:outputLabel>
             <h:inputText id="uproductTitle" value="#{admin.productTitle}"/>
             &nbsp;&nbsp;&nbsp;
             <h:outputLabel for="productQuantity">Quantity: </h:outputLabel>
             <h:inputText id="productQuantity" value="#{admin.productQuantity}"/>
             &nbsp;&nbsp;&nbsp;
              <h:outputLabel for="cost">Price: </h:outputLabel>
             <h:inputText id="cost" value="#{admin.cost}"/>

            <input type="hidden" name="tokenPass" value="#{login.getToken()}" />
            <h:commandButton action="#{admin.storeProduct()}"
                    value="Add"/>
    </h:form>
        </div>
```

Tokens are validated in redirected pages and below is the code.

Admin.Java



```
public void storeProduct()
{
    this.sessionToken=SessionBean.getToken();//gets the random generated token from the session
    this.passedToken=PostMethod.getHidden("tokenPass");//gets the generated token from hidden form field
    String userType=SessionBean.getUserType();//gets the usertype from the sessionBean
    //Executed If condition only if the tokens are mathed and usertype is admin
    if(sessionToken.equalsIgnoreCase(passedToken) && userType.equalsIgnoreCase("admin") )
    {
         //callimg xss cleanstring method to validate input entered by user
         //this.productTitle= Xss.cleanstring(this.productTitle);
         this.productQuantity=Xss.cleanstring(this.productQuantity);
         this.cost=Xss.cleanstring(this.cost);
         adminproductBean.addProduct(this.productTitle, this.productQuantity,this.cost);
         this.actionMessage="Product: "+this.productTitle+" Quantity:"+this.productQuantity+" Succsefuly added to Shopping
    }
    else
        //request from a mirror site beacause of token mismatch or wrong usertype trying to access admin page
        PostMethod.postRedirect("./error.xhtml");//redirected to Error Page
}

public void removeProduct(){
     this.sessionToken=SessionBean.getToken();
     this.passedToken=PostMethod.getHidden("tokenPass");
     if(sessionToken.equalsIgnoreCase(passedToken)){
         boolean remove;
         remove= adminproductBean.removeProduct(this.productTitle);
         if(remove){
             this.actionMessage="Product: "+this.productTitle+" Succsefuly removed from Shopping site";
         }
```

2 Testing Application Security

2.1 A1: Injection

Injection is tested in 2 tools i.e. OWASP ZAP 2.6.0 and using sqlmap python script.

Initially when the application is attacked using ZAP tool, SQL injection was unsuccessful and hence we have created another test scenario to test in sqlmap tool where we tried to inject string '? query=24' and 'query=www.google.com', both the Sql injection failed and below are the screenshots of testing the application.

Below is the command used in sqlmap to test,
sqlmap -r post.txt **http://localhost:8080/Project_Group12-war/faces/login.xhtml**
we have used both POST and GET request to test the application and below is the file used for testing.

PostMethod.txt file

GET http://localhost:8080/Project_Group12-war/faces/login.xhtml?query=%40 HTTP/1.1

User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:39.0) Gecko/20100101 Firefox/39.0

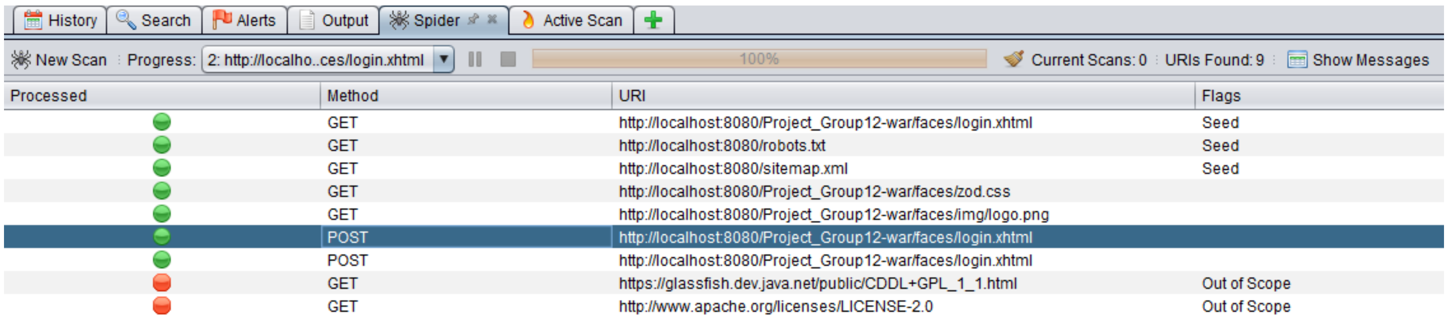
Pragma: no-cache

Cache-Control: no-cache

Content-Length: 0

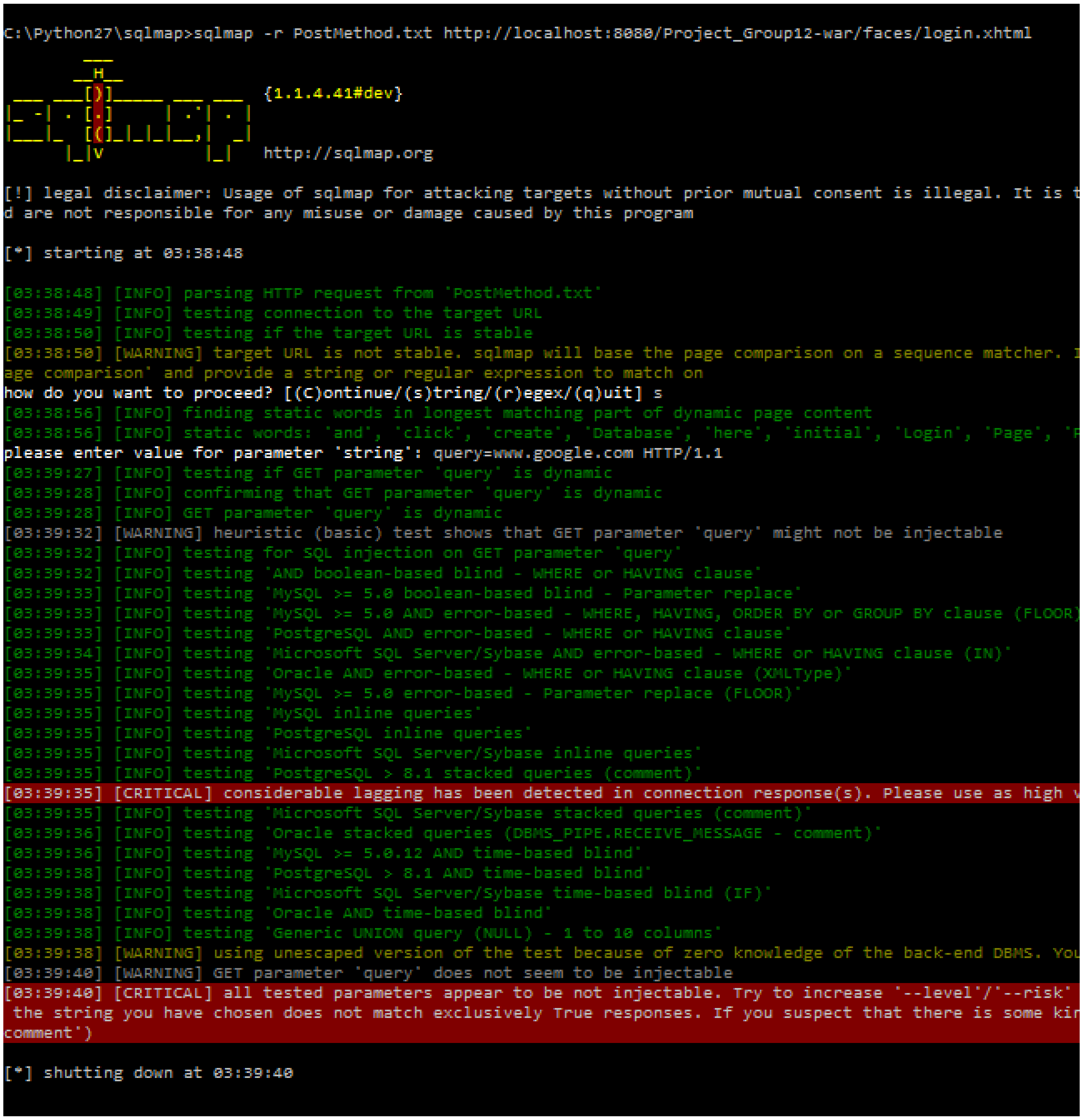
Host: localhost:8080

Below is the screenshot from ZAP tool to capture POST and GET request for sqlmap tool.



Processed	Method	URI	Flags
🟢	GET	http://localhost:8080/Project_Group12-war/faces/login.xhtml	Seed
🟢	GET	http://localhost:8080/robots.txt	Seed
🟢	GET	http://localhost:8080/sitemap.xml	Seed
🟢	GET	http://localhost:8080/Project_Group12-war/faces/zod.css	
🟢	GET	http://localhost:8080/Project_Group12-war/faces/img/logo.png	
🟢	POST	http://localhost:8080/Project_Group12-war/faces/login.xhtml	
🟢	POST	http://localhost:8080/Project_Group12-war/faces/login.xhtml	
🔴	GET	https://glassfish.dev.java.net/public/CDDL+GPL_1_1.html	Out of Scope
🔴	GET	http://www.apache.org/licenses/LICENSE-2.0	Out of Scope

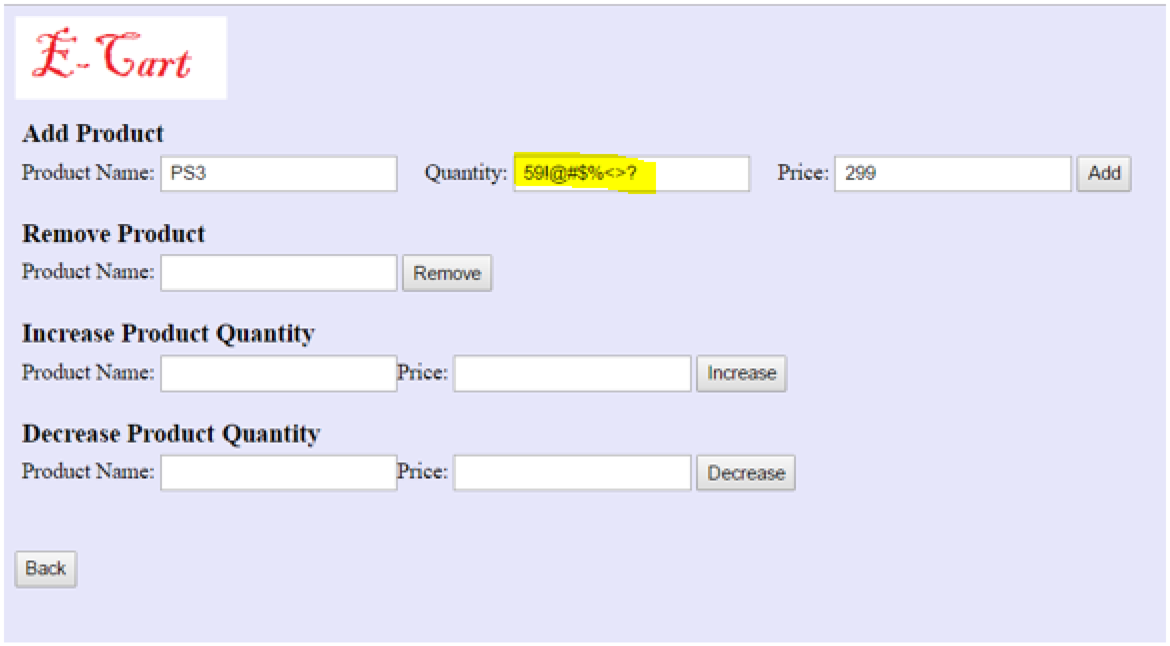
Below is the screenshot of sqlmap testing and all tested parameter appear to be not injectable.



```
C:\Python27\sqlmap>sqlmap -r PostMethod.txt http://localhost:8080/Project_Group12-war/faces/login.xhtml
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.1.4.41#dev}
|_ -| . [.]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is t
d are not responsible for any misuse or damage caused by this program

[*] starting at 03:38:48

[03:38:48] [INFO] parsing HTTP request from 'PostMethod.txt'
[03:38:49] [INFO] testing connection to the target URL
[03:38:50] [INFO] testing if the target URL is stable
[03:38:50] [WARNING] target URL is not stable. sqlmap will base the page comparison on a sequence matcher. I
age comparison' and provide a string or regular expression to match on
how do you want to proceed? [(C)ontinue/(s)tring/(r)egex/(q)uit] s
[03:38:56] [INFO] finding static words in longest matching part of dynamic page content
[03:38:56] [INFO] static words: 'and', 'click', 'create', 'Database', 'here', 'initial', 'Login', 'Page', 'P
please enter value for parameter 'string': query=www.google.com HTTP/1.1
[03:39:27] [INFO] testing if GET parameter 'query' is dynamic
[03:39:28] [INFO] confirming that GET parameter 'query' is dynamic
[03:39:28] [INFO] GET parameter 'query' is dynamic
[03:39:32] [WARNING] heuristic (basic) test shows that GET parameter 'query' might not be injectable
[03:39:32] [INFO] testing for SQL injection on GET parameter 'query'
[03:39:32] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[03:39:33] [INFO] testing 'MySQL >= 5.0 boolean-based blind - Parameter replace'
[03:39:33] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
[03:39:33] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[03:39:34] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[03:39:35] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[03:39:35] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
[03:39:35] [INFO] testing 'MySQL inline queries'
[03:39:35] [INFO] testing 'PostgreSQL inline queries'
[03:39:35] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[03:39:35] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[03:39:35] [CRITICAL] considerable lagging has been detected in connection response(s). Please use as high v
[03:39:35] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[03:39:36] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[03:39:36] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[03:39:38] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[03:39:38] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[03:39:38] [INFO] testing 'Oracle AND time-based blind'
[03:39:38] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[03:39:38] [WARNING] using unescaped version of the test because of zero knowledge of the back-end DBMS. You
[03:39:40] [WARNING] GET parameter 'query' does not seem to be injectable
[03:39:40] [CRITICAL] all tested parameters appear to be not injectable. Try to increase '--level'/'--risk'
 the string you have chosen does not match exclusively True responses. If you suspect that there is some kin
comment')

[*] shutting down at 03:39:40
```

2.2 A3: Cross-Site Scripting (XSS)

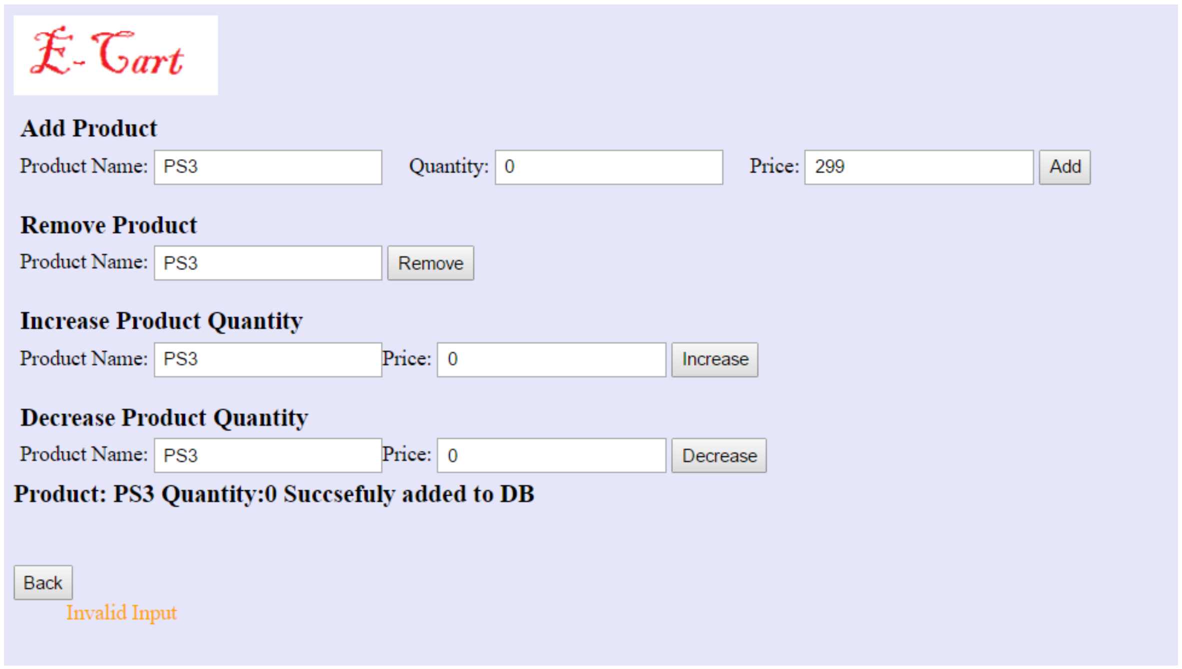
We tried to input special characters in the quantity field and since we used xss filter, it will detect any such special character and will replace that string with '0'.



Add Product

Product Name: PS3 Quantity: 59!@#$%<>? Price: 299

Below is the page with invalid input error message and product will be added with quantity as zero.



Add Product

Product Name: PS3 Quantity: 0 Price: 299

Remove Product

Product Name: PS3

Increase Product Quantity

Product Name: PS3 Price: 0

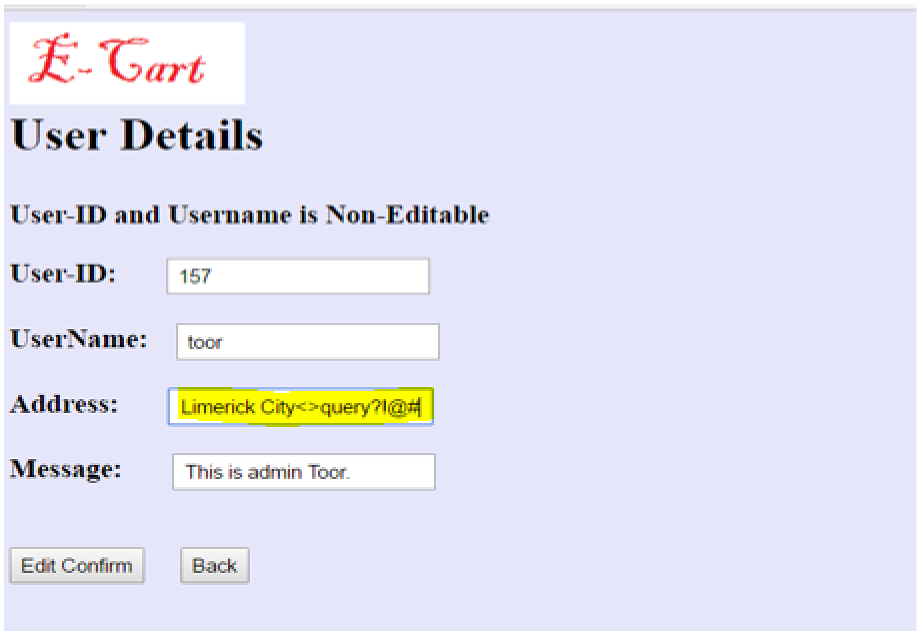
Decrease Product Quantity

Product Name: PS3 Price: 0

Product: PS3 Quantity:0 Succsefuly added to DB

Invalid Input

Below is another test case for user profile edit page. We have tried to input angular bracket for address field and if xss is not implemented, page will break with EJB exception. Since we used filter ,bad string will be replaced by ‘0’.



User Details

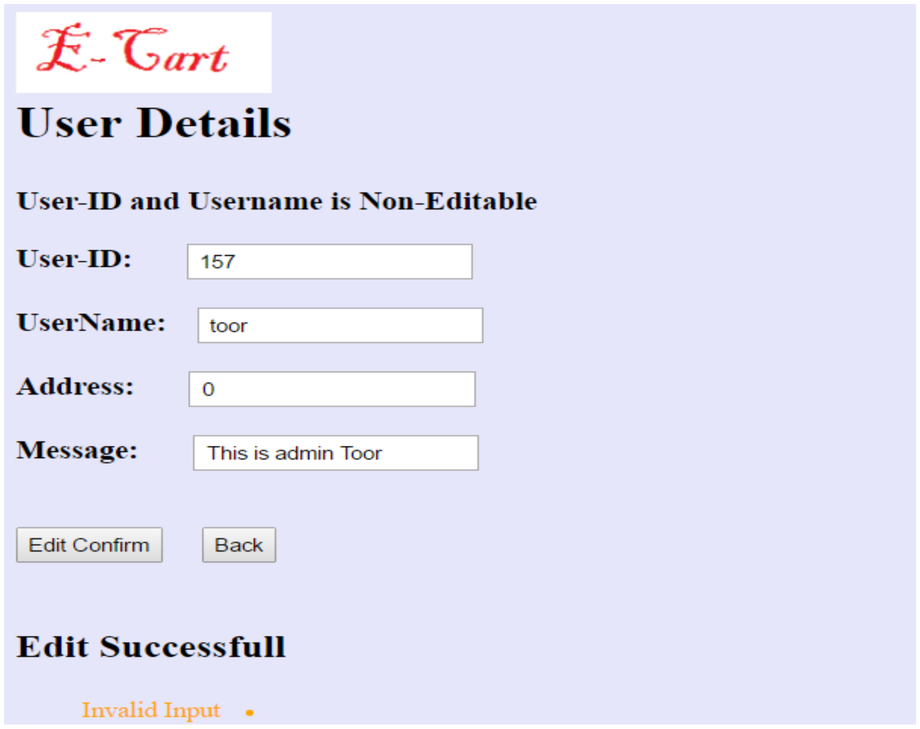
User-ID and Username is Non-Editable

User-ID: 157

UserName: toor

Address: Limerick City<>query?!@#

Message: This is admin Toor.



User Details

User-ID and Username is Non-Editable

User-ID: 157

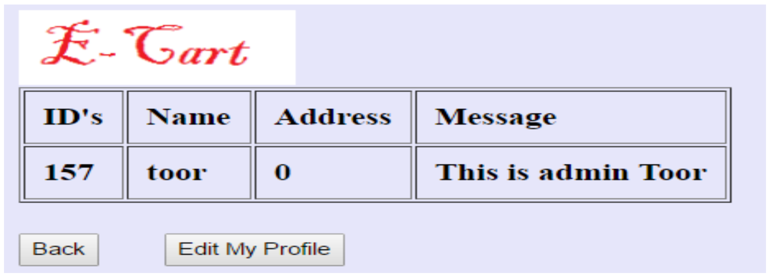
UserName: toor

Address: 0

Message: This is admin Toor

Edit Successfull

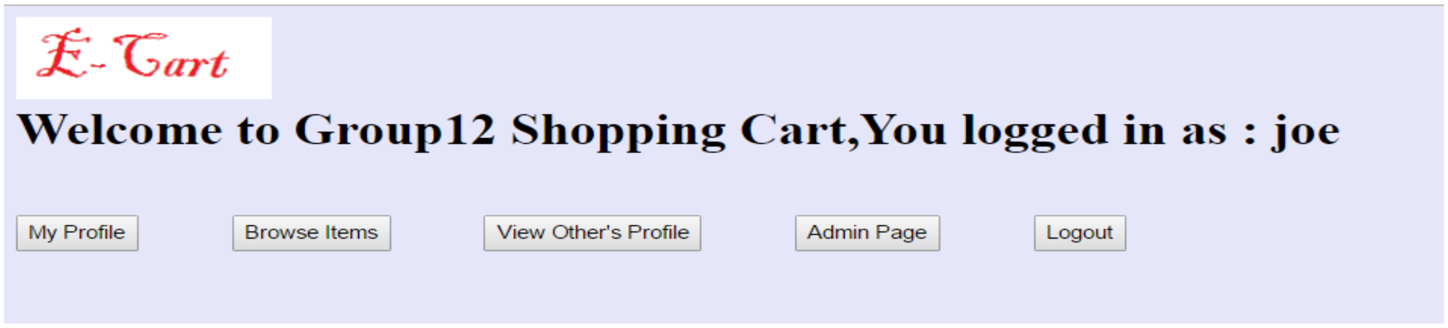
Invalid Input



ID's	Name	Address	Message
157	toor	0	This is admin Toor

2.3 A7: Missing Function Level Access Control

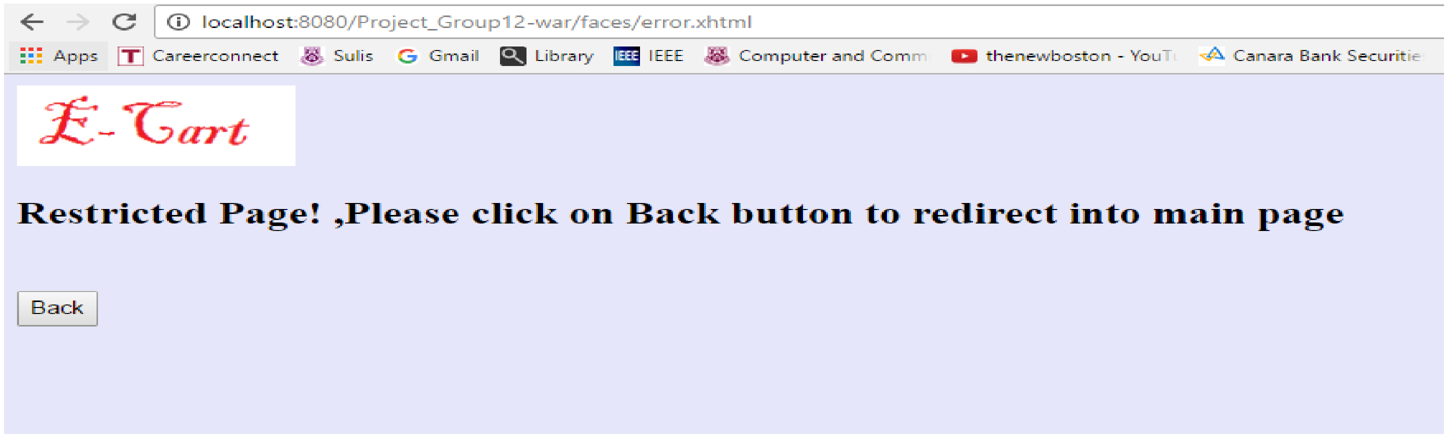
Joe is a customer level access user and if he tries to login to admin page, an error page will be redirected and access will be restricted. joe will be redirected to custom error page and hence page will not be broken and user can come back to landing page.



E-Cart

Welcome to Group12 Shopping Cart,You logged in as : joe

After clicking admin page

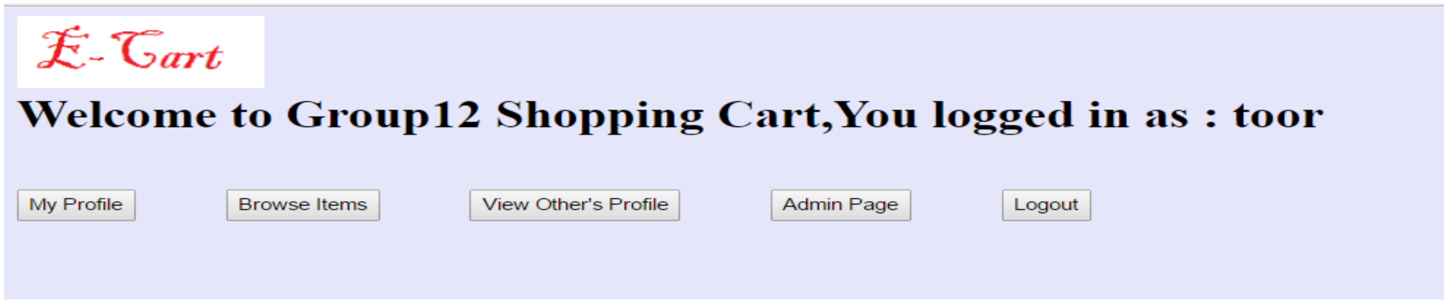


localhost:8080/Project_Group12-war/faces/error.xhtml

E-Cart

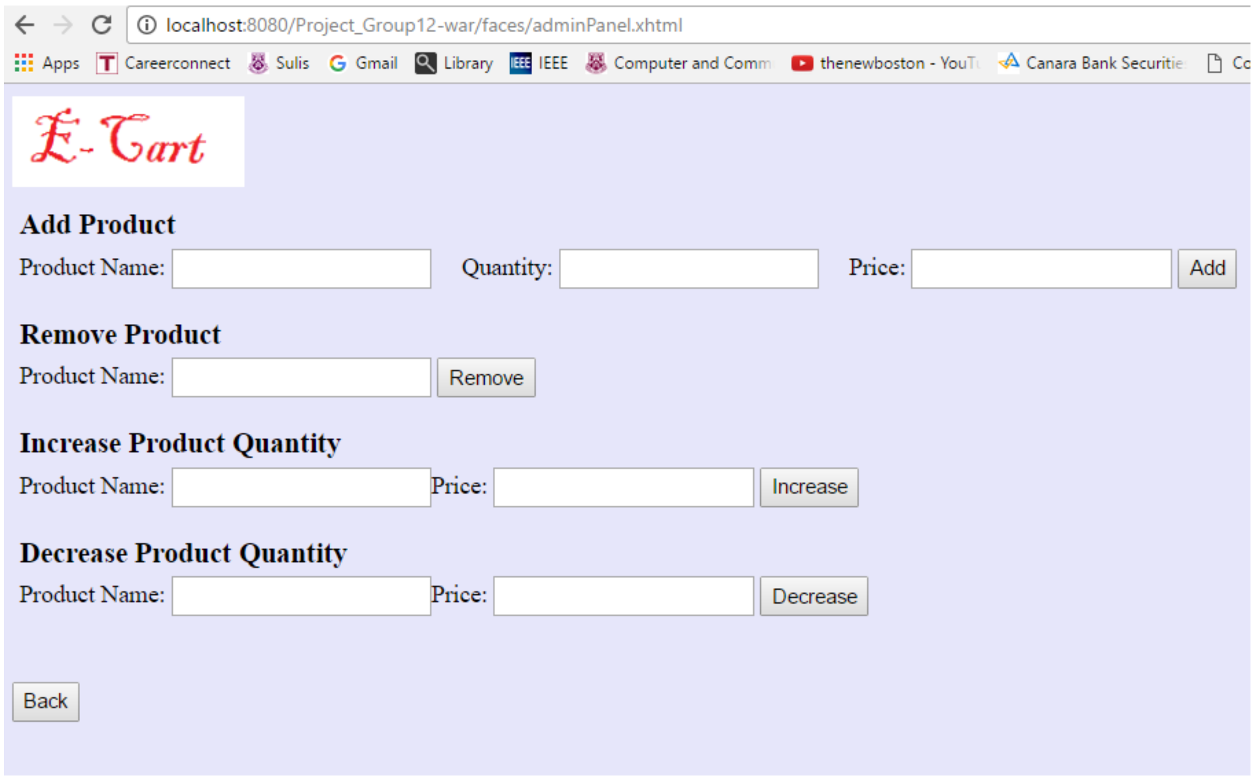
Restricted Page! ,Please click on Back button to redirect into main page

When admin toor tries to access admin page it will be redirected as shown below.



E-Cart

Welcome to Group12 Shopping Cart,You logged in as : toor


localhost:8080/Project_Group12-war/faces/adminPanel.xhtml

Apps | Careerconnect | Sulis | Gmail | Library | IEEE | Computer and Comm | thenewboston - YouT | Canara Bank Securitie | Co

E-Cart

Add Product

Product Name: [ ] Quantity: [ ] Price: [ ] Add

Remove Product

Product Name: [ ] Remove

Increase Product Quantity

Product Name: [ ] Price: [ ] Increase

Decrease Product Quantity

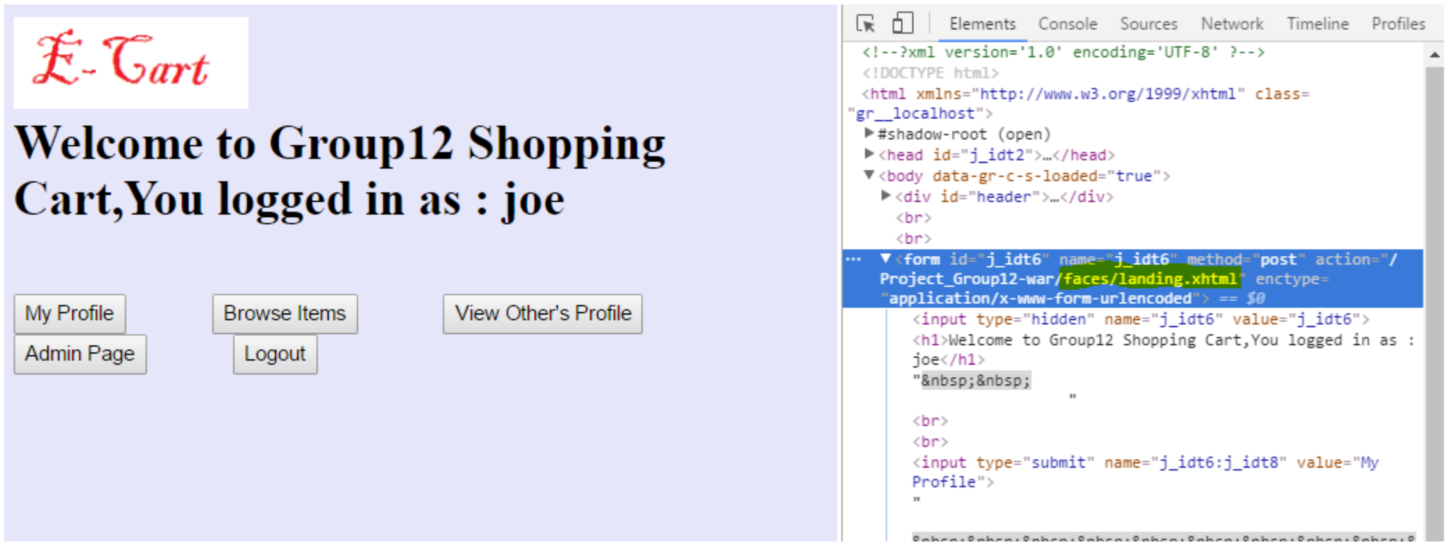
Product Name: [ ] Price: [ ] Decrease

Back

2.4 A8: Cross-Site Request Forgery (CSRF)

joe logged in as a customer and he has restricted access to admin page, if he manages to get admin page url and tries to forge in form element as shown below, he will be restricted and error page will be thrown.

Before forging in the form element

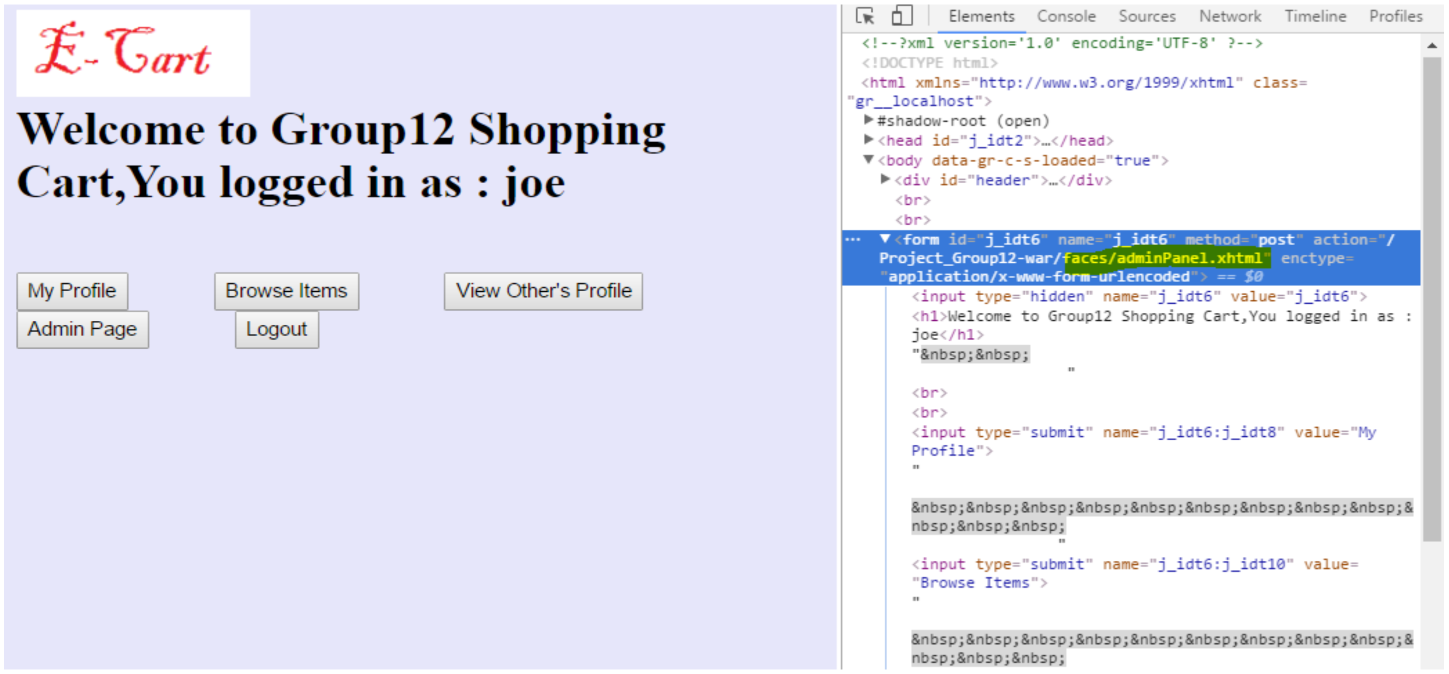

E-Cart

Welcome to Group12 Shopping Cart,You logged in as : joe

My Profile | Browse Items | View Other's Profile | Admin Page | Logout

```
<!--?xml version='1.0' encoding='UTF-8' ?-->
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" class=
"gr__localhost">
  ▶#shadow-root (open)
  ▶<head id="j_idt2">…</head>
  ▼<body data-gr-c-s-loaded="true">
    ▶<div id="header">…</div>
      <br>
      <br>
…   ▼<form id="j_idt6" name="j_idt6" method="post" action="/
    Project_Group12-war/faces/landing.xhtml" enctype=
    "application/x-www-form-urlencoded"> == $0
        <input type="hidden" name="j_idt6" value="j_idt6">
        <h1>Welcome to Group12 Shopping Cart,You logged in as :
        joe</h1>
        "&nbsp;&nbsp;
                     "
        <br>
        <br>
        <input type="submit" name="j_idt6:j_idt8" value="My
        Profile">
        "
```

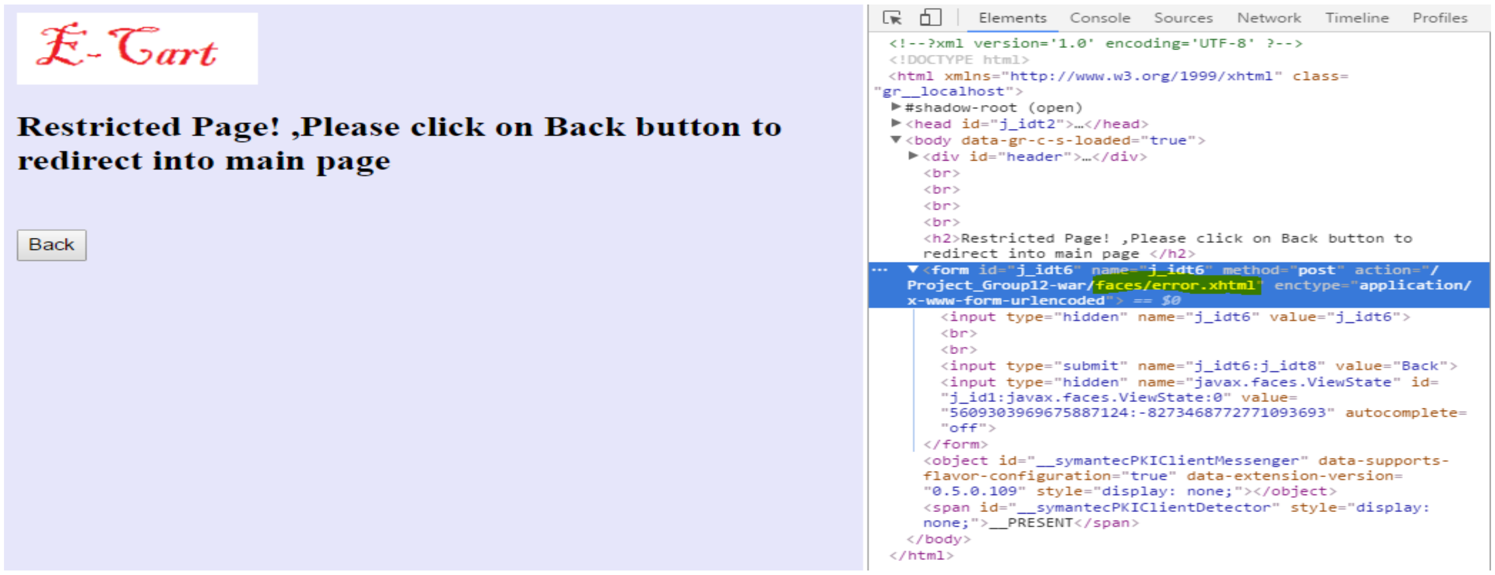
After forging in the form element



Welcome to Group12 Shopping Cart,You logged in as : joe

```
<!--?xml version='1.0' encoding='UTF-8' ?-->
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" class=
"gr__localhost">
  ▶#shadow-root (open)
  ▶<head id="j_idt2">…</head>
  ▼<body data-gr-c-s-loaded="true">
    ▶<div id="header">…</div>
      <br>
      <br>
...   ▼<form id="j_idt6" name="j_idt6" method="post" action="/
      Project_Group12-war/faces/adminPanel.xhtml" enctype=
      "application/x-www-form-urlencoded"> == $0
        <input type="hidden" name="j_idt6" value="j_idt6">
        <h1>Welcome to Group12 Shopping Cart,You logged in as :
        joe</h1>
        "&nbsp;&nbsp;
                          "
        <br>
        <br>
        <input type="submit" name="j_idt6:j_idt8" value="My
        Profile">
        "

        &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
        nbsp;&nbsp;&nbsp;
                         "
        <input type="submit" name="j_idt6:j_idt10" value=
        "Browse Items">
        "

        &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
        nbsp;&nbsp;&nbsp;
```

error.html is redirected without page break or redirecting to admin page with 404 page not found exception.



Restricted Page! ,Please click on Back button to redirect into main page

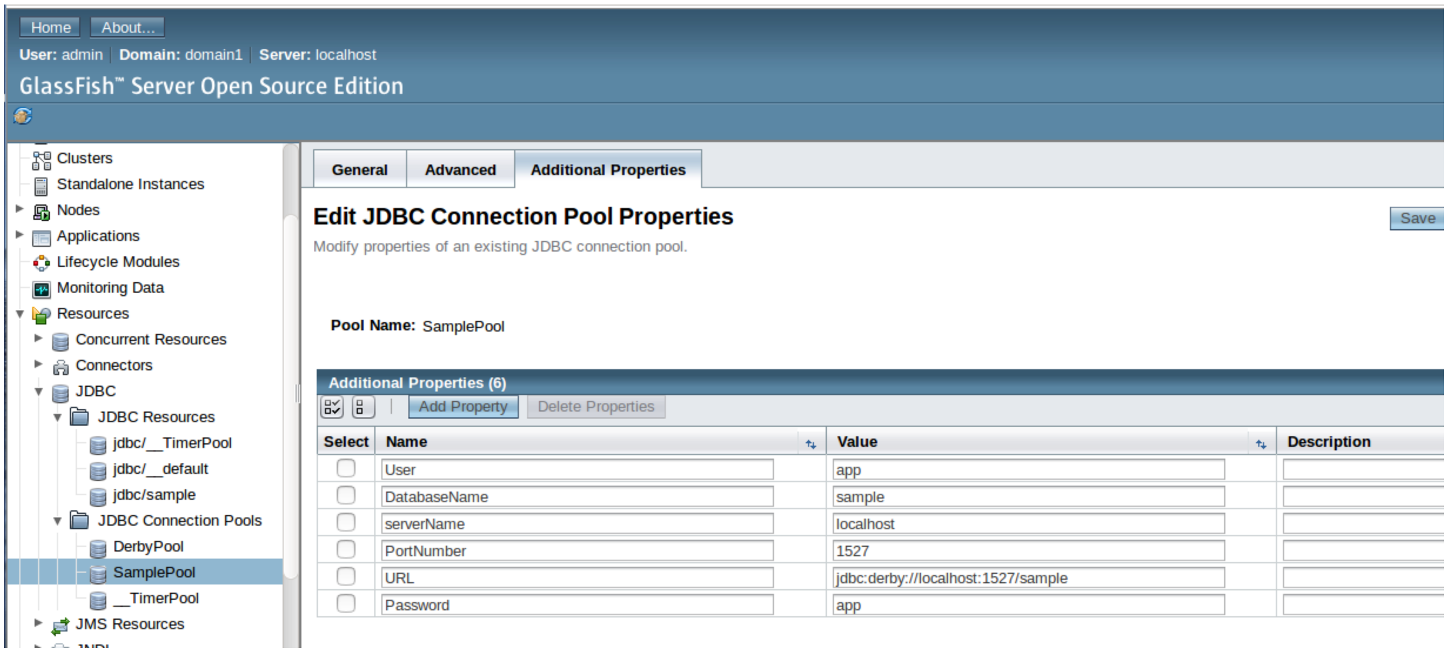
```
<!--?xml version='1.0' encoding='UTF-8' ?-->
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" class=
"gr__localhost">
  ▶#shadow-root (open)
  ▶<head id="j_idt2">…</head>
  ▼<body data-gr-c-s-loaded="true">
    ▶<div id="header">…</div>
      <br>
      <br>
      <br>
      <br>
      <h2>Restricted Page! ,Please click on Back button to
      redirect into main page </h2>
...  ▼<form id="j_idt6" name="j_idt6" method="post" action="/
     Project_Group12-war/faces/error.xhtml" enctype="application/
     x-www-form-urlencoded"> == $0
        <input type="hidden" name="j_idt6" value="j_idt6">
        <br>
        <br>
        <input type="submit" name="j_idt6:j_idt8" value="Back">
        <input type="hidden" name="javax.faces.ViewState" id=
        "j_id1:javax.faces.ViewState:0" value=
        "5609303969675887124:-8273468772771093693" autocomplete=
        "off">
      </form>
      <object id="__symantecPKIClientMessenger" data-supports-
      flavor-configuration="true" data-extension-version=
      "0.5.0.109" style="display: none;"></object>
      <span id="__symantecPKIClientDetector" style="display:
      none;">__PRESENT</span>
    </body>
  </html>
```

3 Deployment

Deployment is successful in local Virtual machine and below are the steps followed for deployment.

We have used Database name as ‘Sample’ and it is configured in SamplePool Connection pools.

SamplePool is selected for JDBC resource and we have used **“jdbc/sample”.**



Edit JDBC Connection Pool Properties

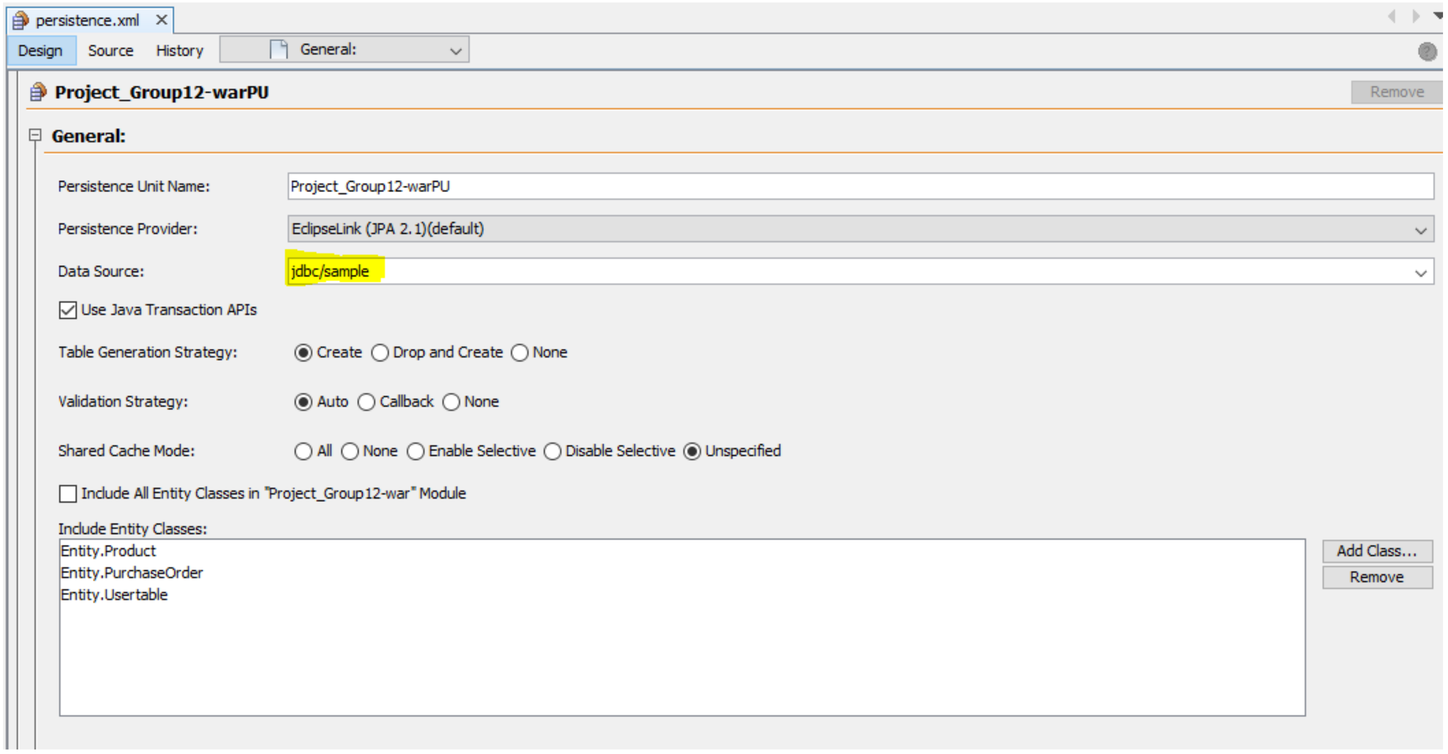
Modify properties of an existing JDBC connection pool.

Pool Name: SamplePool

Additional Properties (6)

Select	Name	Value	Description
☐	User	app	
☐	DatabaseName	sample	
☐	serverName	localhost	
☐	PortNumber	1527	
☐	URL	jdbc:derby://localhost:1527/sample	
☐	Password	app	

Below is the configuration in persistence.xml in the project. Datasource is set as **jdbc/sample/**



Project_Group12-warPU

General:

Persistence Unit Name:	Project_Group12-warPU
Persistence Provider:	EclipseLink (JPA 2.1)(default)
Data Source:	jdbc/sample

[✓] Use Java Transaction APIs

Table Generation Strategy: [✓ Create] Drop and Create None

Validation Strategy: [✓ Auto] Callback None

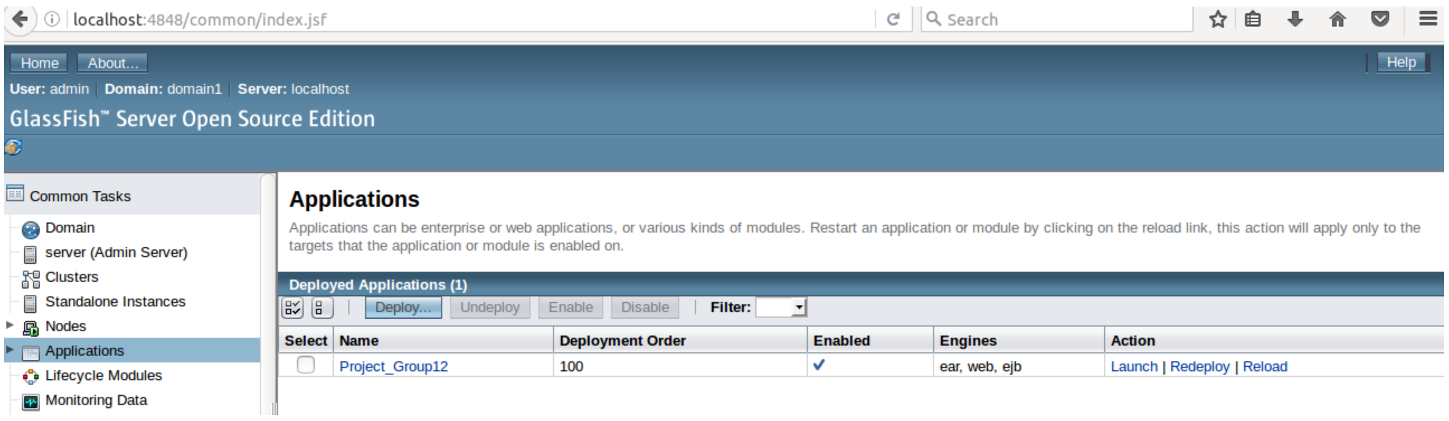
Shared Cache Mode: All None Enable Selective Disable Selective [✓ Unspecified]

☐ Include All Entity Classes in "Project_Group12-war" Module

Include Entity Classes:

Entity.Product
Entity.PurchaseOrder
Entity.Usertable

Project is deployed in glassfish server in the application window and below is the screenshot for successful build and deployment.



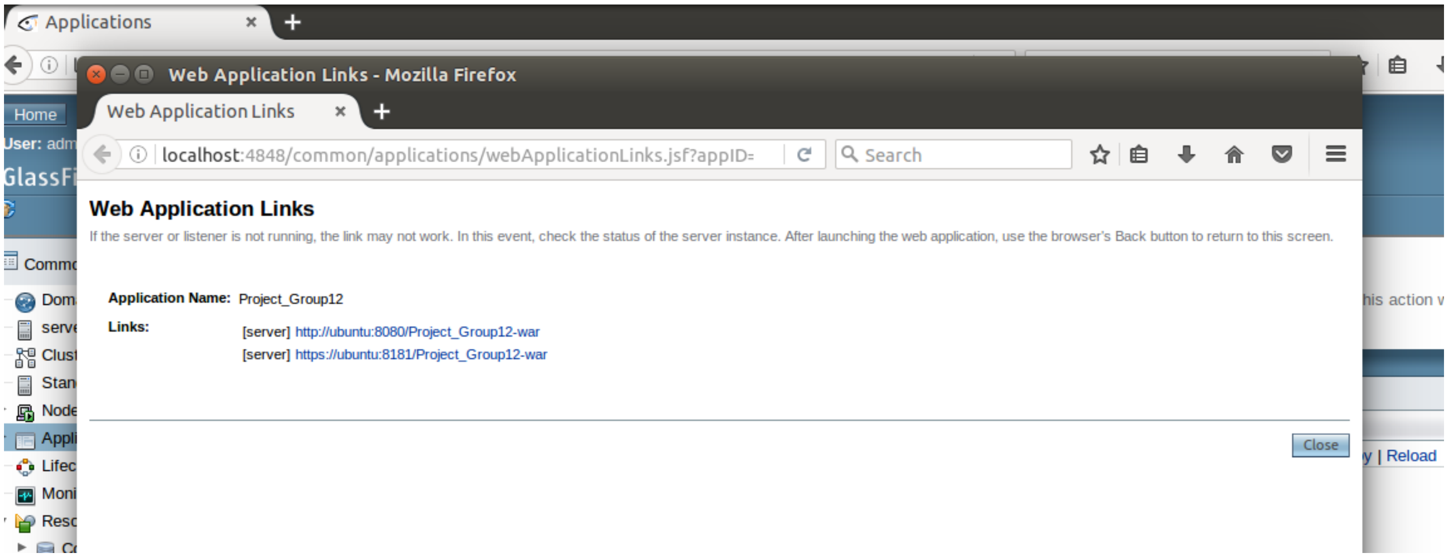
Applications

Applications can be enterprise or web applications, or various kinds of modules. Restart an application or module by clicking on the reload link, this action will apply only to the targets that the application or module is enabled on.

Deployed Applications (1)

Select	Name	Deployment Order	Enabled	Engines	Action
☐	Project_Group12	100	✓	ear, web, ejb	Launch | Redeploy | Reload

Below is the launch window from glassfish server and 2 links are populated,http://Ubuntu:8080/Project_Group12.war is the link generated.


Applications
Web Application Links - Mozilla Firefox
Web Application Links
localhost:4848/common/applications/webApplicationLinks.jsf?appID=
Close

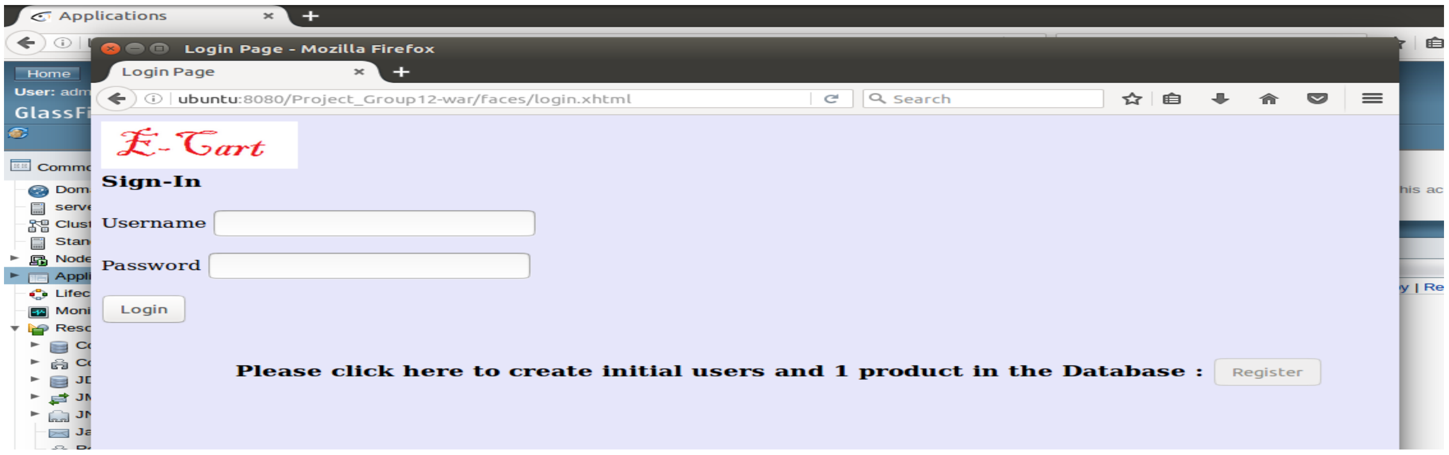
Web Application Links

If the server or listener is not running, the link may not work. In this event, check the status of the server instance. After launching the web application, use the browser's Back button to return to this screen.

Application Name: Project_Group12

Links: [server] http://ubuntu:8080/Project_Group12-war
[server] https://ubuntu:8181/Project_Group12-war

Below is the screenshot for successful application launch.


Applications
Login Page - Mozilla Firefox
Login Page
ubuntu:8080/Project_Group12-war/faces/login.xhtml
E-Cart
Login
Register

Sign-In

Username

Password

Please click here to create initial users and 1 product in the Database :