Homework 4

8.1.8 If a session has been idle for more than 15 minutes, require the user to re-authenticate to re-activate the terminal or session.

Change session-config-> session-timeout to 15 mins

```
<session-config>
    <session-timeout>
        15
    </session-timeout>
</session-config>
```

8.1.6 Limit repeated access attempts by locking out the user ID after not more than six attempts.

- Create a counter to count the number of login attempts

- After the 6th attempt disable all text fields and buttons on the login page

```
    // Set the session variable
    if (isValid) {

        // Create a session object if it is already not  created.
        session = request.getSession(true);

        session.setAttribute("UMUCUserEmail", request.getParameter("emailAddress"));
        session.setAttribute("UMUCUserID", String.valueOf(USER.getUser_id()));

        response.sendRedirect("Authenticate?failed=false");

    } else {

        countLoginAttemnpt++;
        response.sendRedirect("Authenticate?failed=true");

    }
} catch (Exception ex) {
    Logger.getLogger(Authenticate.class.getName()).log(Level.SEVERE, null, ex);
}
```

Counter incremented after every failed login attempt

```
if (request.getParameter("failed").equals("true")) {

    if (countLoginAttemnpt == 6) {

        request.setAttribute("ErrorMessage", "30 Second Login Suspension");
        RequestDispatcher dispatcher = request.getRequestDispatcher("login.jsp");
        dispatcher.forward(request, response);
        countLoginAttemnpt = 0;
```

On 6th attempt display lockout message

```
<% if (session.getAttribute("UMUCUserEmail") == null) {

        application.setAttribute("attempts", SDEV425_HW4.Authenticate.countLoginAttemnpt);
%>
```

Set the count variable as an applications attribute called attempts

```
${ attempts eq 6 ? 'disabled="disabled"' : ''}>
```

The EL expression is used to check disable text fields "emailAddress", "pfield" and the button "SignIn"

```
window.onload = function () {
    document.getElementById('sub-id').onclick = function () {
        document.getElementById('form-id').submit();
        return false;
    };
};

setTimeout(function () {
    document.getElementById('sub-id').disabled = false;
    document.getElementById('email').disabled = false;
    document.getElementById('password').disabled = false;
    document.getElementById("counter").value = 0;

}, 30000);
</script>
```

JavaScript function **setTimeout** is used to set a 30 second disable period. While addressing this issue I noticed each time the login page was refreshed it would re-submit the form details. To mitigate this, I added the onclick JavaScript function which only submits when the “SignIn” button is clicked. This lead to a redesign of the pattern used to submit request within the application.

Post/ Redirect/Get

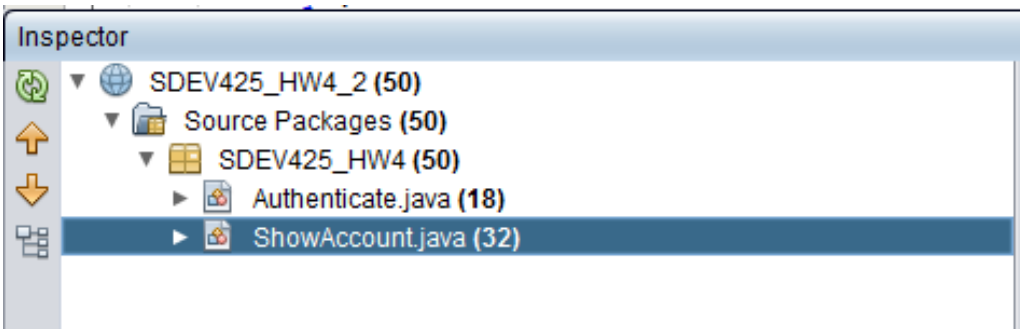
To prevent duplicated form submission, we redirect all HTTP POST request including HTTP GET response

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    HttpSession session;

    try {

        // Get the post input
        Boolean isValid = validate(request);
        response.setContentType("text/html;charset=UTF-8");

        // Set the session variable
        if (isValid) {

            // Create a session object if it is already not  created.
            session = request.getSession(true);

            session.setAttribute("UMUCUserEmail", request.getParameter("emailAddress"));
            session.setAttribute("UMUCUserID", String.valueOf(USER.getUser_id()));

            response.sendRedirect("Authenticate?failed=false");

        } else {

            countLoginAttemnpt++;
            response.sendRedirect("Authenticate?failed=true");

        }
    } catch (Exception ex) {
        Logger.getLogger(Authenticate.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Post request are redirected to locations, in this case HTTP Post request are redirected to the Authenticate servlet with the appropriate response

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    if (request.getParameter("failed").equals("false")) {

        // Send to the Welcome JSP page
        RequestDispatcher dispatcher = request.getRequestDispatcher("welcome.jsp");
        dispatcher.forward(request, response);
    }

    if (request.getParameter("failed").equals("true")) {

        if (countLoginAttemnpt == 6) {

            request.setAttribute("ErrorMessage", "30 Second Login Suspension");
            RequestDispatcher dispatcher = request.getRequestDispatcher("login.jsp");
            dispatcher.forward(request, response);
            countLoginAttemnpt = 0;

        } else {

            request.setAttribute("ErrorMessage", "Invalid Username or Password. Try aga
            RequestDispatcher dispatcher = request.getRequestDispatcher("login.jsp");
            dispatcher.forward(request, response);

        }
    }
}
```

HTTP GET method handles appropriated response.



FindBug vulnerabilities prior to the addition Classes Users and ConnectToDb.

Additional Classes/ FindBug vulnerabilities

The application was scanned for security vulnerabilities using findbug and over 50 issues were detected. Most issues originated from the use of mutable fields and properly closing the database resources. To address this, a Class called **Users** was created to handle the assignment and retrieval of values associated with each user (Users class acts as a model of the CustomerAccount table) and the and a Class called **ConnectToDb** was created to handle database connective and querying.

```
public static ConnectToDb newInstance() {

    return new ConnectToDb();

}
```

Creates a new instance of ConnectToDb

```
private Connection connectionToDatabase() {

    Connection conn = null;
    try {

        ClientDataSource ds = new ClientDataSource();
        ds.setDatabaseName("SDEV425");
        ds.setServerName("localhost");
        ds.setPortNumber(1527);
        ds.setUser("sdev425");
        ds.setPassword("sdev425");
        ds.setDataSourceName("jdbc:derby");
        conn = ds.getConnection();

    } catch (SQLException ex) {
        Logger.getLogger(ConnectToDb.class.getName()).log(Level.SEVERE, null, ex);
    }
    return conn;
}
```

Method used to connect to database

```
public ResultSet newRequest(String... parameters) throws SQLException {

    PreparedStatement stmt = connectionToDatabase().prepareStatement(parameters[0]);

    for (String param : parameters) {

        int index = Arrays.asList(parameters).indexOf(param);

        if (index != 0) {
            stmt.setString(index, param);

        }

    }

    return stmt.executeQuery();

}
```

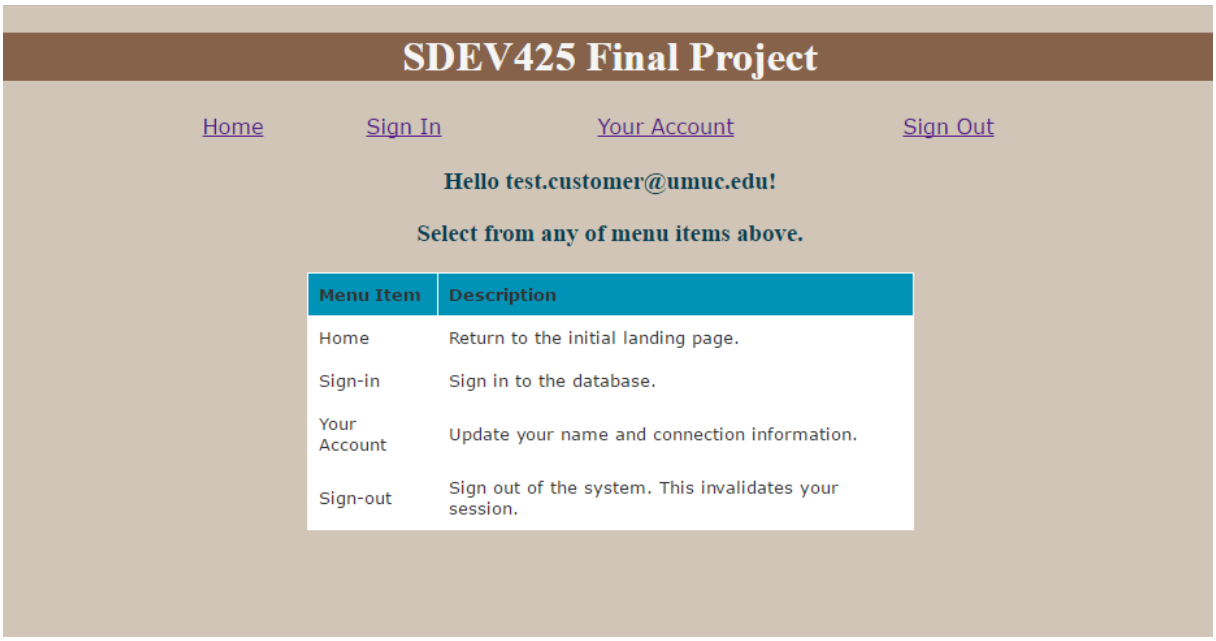
Method used to query the database

```
public boolean validate(HttpServletRequest request) throws SQLException {

    boolean status = false;
    int hitcnt = 0;
    ResultSet rs_1 = null;

    try {
        USER.setUser_id(0);

        rs_1
                = ConnectToDb.newInstance().newRequest(ConnectToDb.QUERYREQUEST_2,
                        String.valueOf(request.getParameter("emailAddress")));

        while (rs_1.next()) {

            USER.setUser_id(rs_1.getInt(1));
        }

        System.out.println("THis users id is " + USER.getUser_id());
        if (USER.getUser_id() > 0) {

            ResultSet rs_2
                    = ConnectToDb.newInstance().newRequest(ConnectToDb.QUERYREQUEST_3,
                            String.valueOf(USER.getUser_id()),
                            String.valueOf(request.getParameter("pfield")));

            while (rs_2.next()) {
                hitcnt++;
            }
            // Set to true if userid/password match
            if (hitcnt > 0) {
                status = true;
                rs_2.close();
            }
        }
    } catch (SQLException ex) {
        System.out.println(ex);
    } finally {

        if (rs_1 != null) {
            rs_1.close();
        }

    }
```

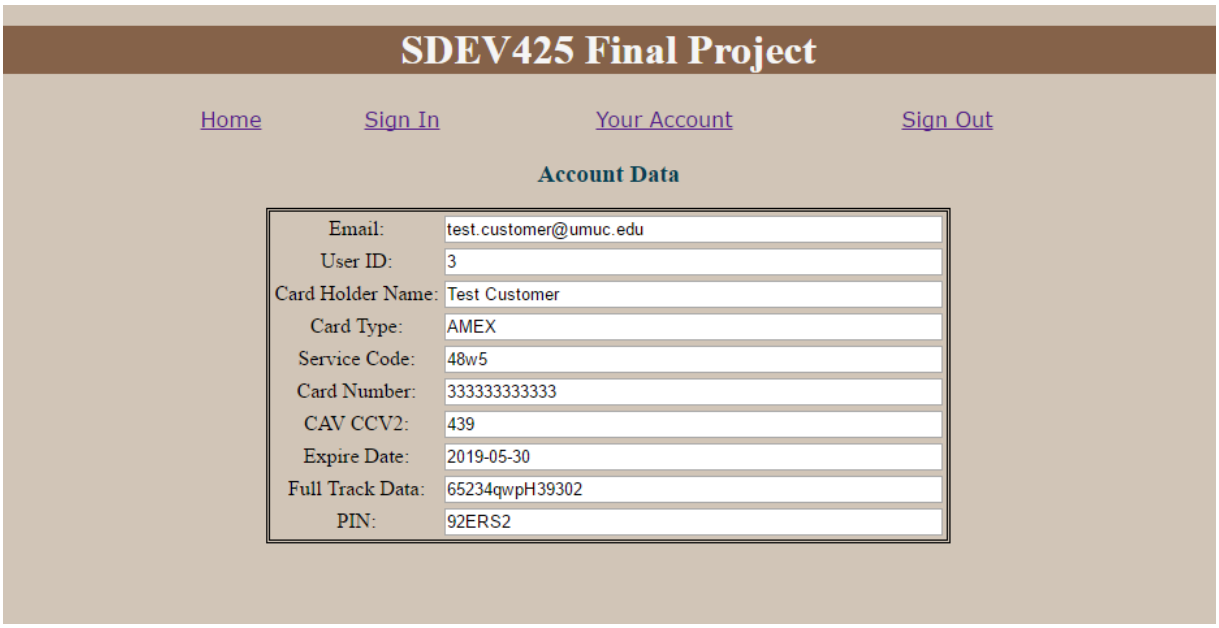
Classes ConnectToDb and Users are used to Validate the login Creditentials submitted

```
public void getData(HttpServletRequest request, HttpSession session) {

    try {

        ResultSet rs
                = ConnectToDb.newInstance().newRequest(ConnectToDb.QUERYREQUEST_1,
                        String.valueOf(session.getAttribute("UMUCUserID")));

        // Assign values
        while (rs.next()) {

            USER.setUser_id(rs.getInt(1));
            USER.setCardholdername(rs.getString(2));
            USER.setCardType(rs.getString(3));
            USER.setServiceCode(rs.getString(4));
            USER.setCardNumber(rs.getString(5));
            USER.setCav_ccv2(rs.getInt(6));
            USER.setExpiredate(rs.getDate(7));
            USER.setFullTrackData(rs.getString(8));
            USER.setPin(rs.getString(9));
        }


    } catch (Exception e) {
        System.out.println(e);
    }

}
```

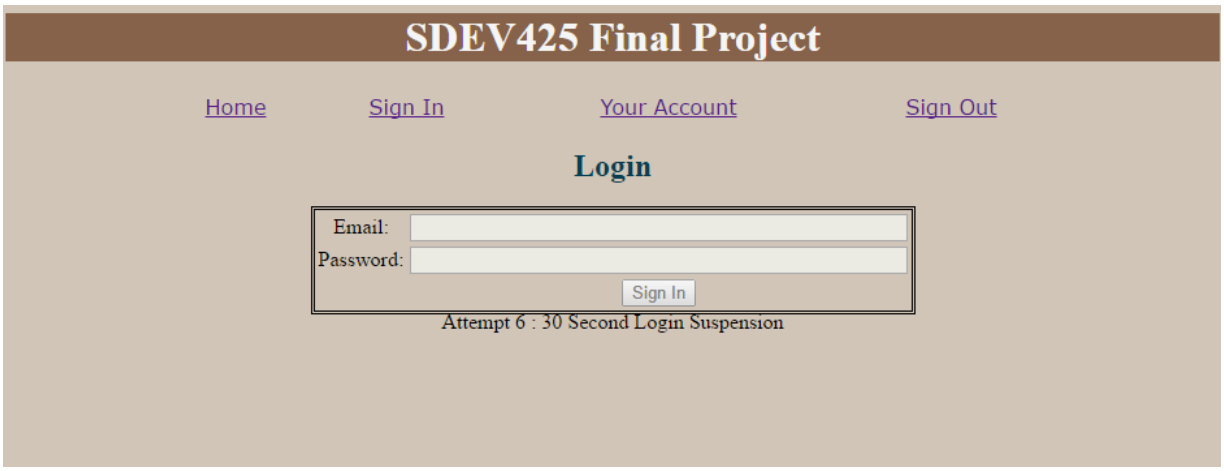
Classes ConnectToDb and Users are used to set user data 0



Sucessfully logged in



Display user data



Suspended login page