**CST-236 Activity 2 Guide**

Contents

[Part 1: nLayer Design for Software Services 2](#_Toc530474223)

[Part 2: Security Class 5](#_Toc530474224)

[Activity 2 Overall Submissions 6](#_Toc530474225)

## Part 1: nLayer Design for Software Services

**Overview**

In this activity, students will learn how to use SQL Wildcard Searches to search for a pattern within a column in a table, create a table and bulk load data into the table as well as create a simple search form.

**Execution**

Execute this activity according to the following guidelines:

The design in this application uses the n-layered approach. You can trace the flow of the code through three levels. The SearchHandler.php depends on the UserBusinessService.php which depends on the UserDataService.php

Beginning programmers will see this approach as unnecessary layers. It would be possible to put SQL select statements directly in the SearchHander.php file. However, the benefits of having multiple layer design allows better modularity. For example, in a future application upgrade, the UserDataService module could be rewritten so it connects with a database different than MySQL or it could even connect with a JSON web service instead of a SQL interface. The method names, such as databaseService->findByFirstName('alice') would not change.

1. Initialize database:
   1. Using MySQL Workbench, create a database schema named *ica17*.
   2. Create a table *users* that has three columns:
      1. ID of type int that is auto-incremented
      2. FIRST\_NAME of type varchar(100)
      3. LAST\_NAME of type varchar(100)
   3. Create a data load text file with 100 users and use MySQL Workbench to bulk load the users into the users table.
2. Write and Test a Search Query in MySQL Workbench:
   1. Write a search query to find all users in the user table where the Last Name has the letter a in it.

SELECT \* FROM users WHERE LAST\_NAME LIKE '%a%';

* 1. Test the search query in MySQL Workbench.
  2. Write a search query to find all users in the user table where the First Name has the letter a in it.

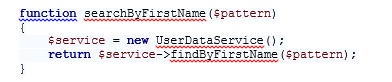
SELECT \* FROM users WHERE FIRST\_NAME LIKE '%a%';

* 1. Test the search query in MySQL Workbench.
  2. Write a search query to find all users in the user table where the First Name has the letter a in it OR the Last Name has the letter e in it.

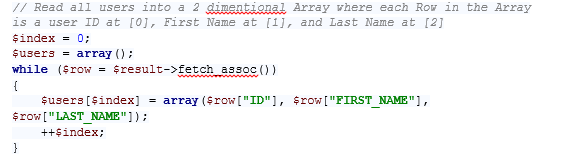
SELECT \* FROM users WHERE FIRST\_NAME LIKE '%a%' OR LAST\_NAME LIKE '%a%';

* 1. Test the search query in MySQL Workbench.
  2. Take a screenshot of the results of running the query from Step 1f.

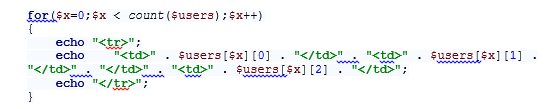
1. Write a simple Search Form in HTML and PHP:
   1. Create a new Project in Eclipse PHP named *ica16*.
   2. Create a Search Form in HTML:
      1. Create a new HTML file called search.html.
      2. Add a single edit control to enter a First Name search pattern.
      3. Add a submit button to POST to the form to a PHP Handler.
      4. Specify a Form action of searchHandler.php and a Form method of POST.
   3. Create support files in PHP:
      1. Create a new PHP Class named *Database* that contains a function getConnect() that gets a connection to the database.
      2. Add an Autoloader.php script to autoload all dependent PHP files.
   4. Create a User Business Service class in PHP:
      1. Create a User Business Service class named *UserBusinessService.php*.
      2. Add a new function, *searchByFirstName()* in *UserBusinessService.php* that instantiates an instance of the *UserDataService* and calls the *findByFirstName()* and return the value from this method from *searchByFirstName()*.



* 1. Add a new function, *findByFirstName()* in *UserDataService.php* that create a User Data Service class in PHP:
     1. Create a User Data Service class named *UserDataService.php*.
     2. Add a new function, *findByFirstName()* in *UserDataService.php* that searches for the database for users first name with the specified pattern passed as an argument to the function and returns the results in a 2-dimensional array. If no results are found then return null.



* 1. Create a Display Users Partial page in PHP:
     1. Create a partial page named \_*displayAllUsers.php* that takes as an argument a 2-dimensional array and displays the user data in a table.



* 1. Create a Search Handler in PHP:
     1. Create a new PHP file called searchHandler.php.
     2. Read the POST search pattern into a PHP variable called searchPattern.
     3. In the searchHandler call the *searchByFirstName()* from the *UserBusinessService* and display the users array in a HTML Table using the \_*displayAllUsers.php*. If the return users array is a null value simply display an error message that there were no users found.
  2. Deploy and test your application:
     1. Take a screenshot of your users displayed in the table for desired search results and when no users are found.

**Notables and Extra Practice:**

1. Make a more complicated Search Form that allows users to search on First Name, Last Name, or First and Last Name.
2. Make a more complicated Search Form that allows users to specify if they want to do a \*.\* search, \*. search, or .\* search.
3. Make a more complicated Search Form that allows users to search lower and upper case search phrases if they want to so a case sensitive search.

**Documentation**

All documentation will be submitted at the end of the activity to the learning management system. Ensure documentation of the following:

1. Screenshot of the SQL query from Step 2g
2. Screenshot of the Users Search Results from Step 3h

Part 2: Security Class

**Overview**

In this activity, students will create the necessary PHP scripts that can be used to provide Page Security in a PHP application.

**Execution**

Execute this assignment according to the following guidelines:

1. Create a new project in Eclipse.
2. Create a new PHP class.
   1. Name your class *SecurityService* (notice this is a noun).
   2. Create two private class variables for a Username and Password (your state).
   3. Create a constructor that takes a Username and Password (initialize your state).
   4. Create a method that could be used to login / authenticate a User (notice this is a verb). The method should return true if authentication passes and false if authentication fails. Your authentication logic can be hard coded to simply check for a desired username and password combination.
3. Create an application Login form page and Login handler:
   1. Create a Login Handler *LoginHandler.php* that will validate your form data, create an instance of the *SecurityService* class, call the authenticate method, and display the results of the authentication with an echo statement.
   2. Create a Login HTML page *L.phpogin.html* with a Login Form, 2 edit controls for a Username and Password, and a submit Login Button.
   3. Bind the Login Form to the Login Handler using a POST method.
4. Test your Login Form and Security Service Class:
   1. Test your Login in your browser:
      1. http://localhost:[port]/ projectName /Login.html
5. Create a file called *header.php* that simply calls session\_start().
6. Create a file called *securePage.php* that includes header.php and checks to see if $\_SESSION["principle"] is set OR $\_SESSION["principle"] is null OR $\_SESSION["principle"] is false and if any of these conditions are true redirect the user to the login.html page by calling header("Location: index.html").
7. Create a file called *loginFailed.php* that simply echoes a login failed message on the screen.
8. Create a file called *loginPassed.php* that simply echoes a login passed message on the screen as well as includes the *header.php* and *securePage.php* files.
9. Change the logic in the *LoginHandler.php* so that if the authentication passes set $\_SESSION["principle"] to a value of true and then includes the *loginPassed.php* page else if authentication fails set $\_SESSION["principle"] to a value of false and then includes the *loginFailed.php* page.
10. Take a screenshot of the output when logging successfully and failed in your browser.
    1. Test your Login in your browser:
       1. http://localhost:[port]/ projectName /Login.html
11. Close your browser and validate that *loginPassed.php* is secured and that the user is safely navigated to the Login Page.
    1. Test your Login in your browser:
       1. http://localhost:[port]/projectName/loginPassed.php

**Documentation**

All documentation will be submitted at the end of the activity to the learning management system. Ensure documentation of the following:

1. ZIP file of the project folder containing the PHP source code
2. Screenshot when login was successful
3. Screenshot when login failed

## Activity 2 Overall Submissions

Submit the following to the learning management system:

**Part 1: nLayer Design for Software Services**

Screenshot of the SQL query from Step 2g

Screenshot of the Users Search Results from Step 3h

**Part 2: Security Class**

ZIP file of the project folder containing the PHP source code

Screenshot when login was successful

Screenshot when login failed