|  |  |  |
| --- | --- | --- |
| Unit Test Point | Testing | Result |
| SQL statement to get user | Write SQL statement with test inputs and run it  using phpMyAdmin.  Test input: username = Ebenezer.dzobo  Expected Result: find this user | The user was gotten |
| SQL statement to search for the username | Write SQL statement with test inputs and run it  using phpMyAdmin.  Test input: username was Ebenezer.dzobo  Expected Result: to find this user | The user was found |
| getUser method in  projectUsers2 class | Write a code to create object of the class and  call the method with test inputs and compare  output in database  Test input:Ebenezer.dzobo  Expected Result: the user was obtained | The user was gotten |
| getUser method in  projectUsers2 class | Write a PHP unit test to test the code  Input:ebenezer.dzobo  Output:OK(1 test,3assertions)  Assert Condition: assertCount()  assertEquals() |  |
| searchUser method in  User class | Write a code to create object of the class and  call the method with test inputs and compare  output in database  Test input:Ebenezer.dzobo  Expected Result: |  |
| searchUser method in  User class | Write a PHP unit test to test the code  Input:ebenezer.dzobo  Output:OK(1 test, 3assertions)  Assert Condition:assertCount()  assertEquals() |  |

**Sequence Diagram for “Logging In”**

DB

Proposal

Student

2. Send get user query

And check the password of the user

Enter username and passsword

3.Open welcome page if successful

4. Display Page

**Functional Requirements: -**

The user should be able to: Login

**Input:**

-First page to select which type of account you want to login to (student or faculty) using a button click.

-User credentials (for login)

-username

-password

**Output:**

New page depending on the type of user

Welcome Page

-Student: submission of IRB

-FACULTY: submission of IRB ; view submitted IRB’s.

**DATA ENTITIES**

USER

**Database Procedure**

//RUN Input against database via query

After the user has put in an input, input in the user name section is run across the database to check for an occurrence in the database using a select statement. When a result is obtained the password is the checked across the password from the textbox if they are the same, the usr is then granted access according to the account type.

**Class/Methods**

User class

-login(String username, String password)

**Range of Inputs**

All forms of alphanumeric keys on the keyboard

**Conditions**

Password requirements

Passwords must not contain the user's entire samAccountName (Account Name) value or entire displayName (Full Name) value. Both checks are not case sensitive:

* The samAccountName is checked in its entirety only to determine whether it is part of the password. If the samAccountName is less than three characters long, this check is skipped.
* The displayName is parsed for delimiters: commas, periods, dashes or hyphens, underscores, spaces, pound signs, and tabs. If any of these delimiters are found, the displayName is split and all parsed sections (tokens) are confirmed not to be included in the password. Tokens that are less than three characters in length are ignored, and substrings of the tokens are not checked. For example, the name "Erin M. Hagens" is split into three tokens: "Erin," "M," and "Hagens." Because the second token is only one character long, it is ignored. Therefore, this user could not have a password that included either "erin" or "hagens" as a substring anywhere in the password.

Passwords must contain characters from three of the following five categories:

* Uppercase characters of European languages (A through Z, with diacritic marks, Greek and Cyrillic characters)
* Lowercase characters of European languages (a through z, sharp-s, with diacritic marks, Greek and Cyrillic characters)
* Base 10 digits (0 through 9)
* Nonalphanumeric characters: ~!@#$%^&\*\_-+=`|\(){}[]:;"'<>,.?/
* Any Unicode character that is categorized as an alphabetic character but is not uppercase or lowercase. This includes Unicode characters from Asian languages.
  1. functional requirements

-login

b. description

-Depending on the type of user the login will, if successful will open the correct page, that is user/faculty

d. test plan

I will try a correct username but a wrong password and vice versa, just to if theses scenario’s will cause any issues.

**Professional Test Result**

The upload functional requirement works in an order as follows:

-Click choose file

-File chooser pops up, choose File

-Click on upload, show details of the document

-display (name, document type, date,size) in array

-Click save

This is how the upload file works. This branch is able to open a file chooser and traverse through directories to search for files. This is facilitated easier since you will be seeing only word documents and pdf’s. Also this is very efficient since you see all the details of files after you have chosen the file. After all this you can choose to save or cancel the process.