



BILKENT UNIVERSITY
FACULTY OF ENGINEERING
CS458 SOFTWARE VERIFICATION/VALIDATION

Project #1
Introduction to Test Automation (Web)

YUSUF DALVA 21602867
EGE ÖZCAN 21602150
CAGLA SOZEN 21602547

March 7, 2020

CONTENTS

1 Selenium	3
1.1 Introduction	3
1.2 Capabilities	3
2 Login Page Implementation	3
2.1 Back-end part of the implementation	4
2.2 Front-end part of the implementation	5
3 Test Cases	7
3.1 Test Case 1: User logs in to the system successfully	7
3.2 Test Case 2: A very long string entered as an input	8
3.3 Test Case 3: Wrong Password Entered For a User	9
3.4 Test Case 4: Loading the contents of the login page	10
3.5 Test Case 5: Protection Against SQL Injection	10
4 Evaluation	11
5 Discussion	13
6 Appendix	13
References	27

1 SELENIUM

1.1 Introduction

Selenium is an open source automation tool for web browsers which emulates the interaction of a user and enables remote controlling of browser instances. It can simulate the interaction of end-users to simulate possible actions of the user which might fail some of the test cases. It supports automation across different platforms and particularly browsers for different programming languages.

The capabilities of Selenium are listed and explained below.

1.2 Capabilities

- Simulating common activities done by the users
- Providing a single interface for all of the commonly used browser technologies
- Ability to detect HTML elements and performing actions on these elements
- Introducing implicit and explicit waits for DOM elements
- Ability to execute tests remotely
- Cross platform testing
- Adding cookies to the current browsing context
- Arbitrary JavaScript execution
- Record and playback feature for testing without scripts
- Parallel test execution

2 LOGIN PAGE IMPLEMENTATION

Following the project description given for the project, a login page has been implemented which originates from the Bilkent University SRS system. In the implementation we tried to implement all of the functionalities that are present in the login page of the SRS Login page. The information boxes are set accordingly and Turkish language support has been added to follow the original functionality of the application. Below, some screenshots are provided to give a general idea about the application:

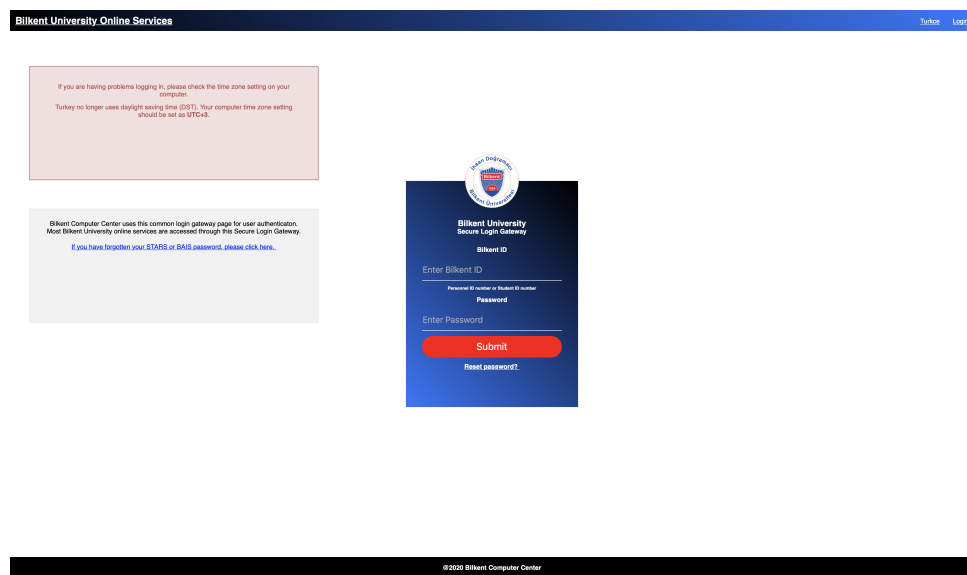


Figure 1: Screenshot of the English version of the login page implemented

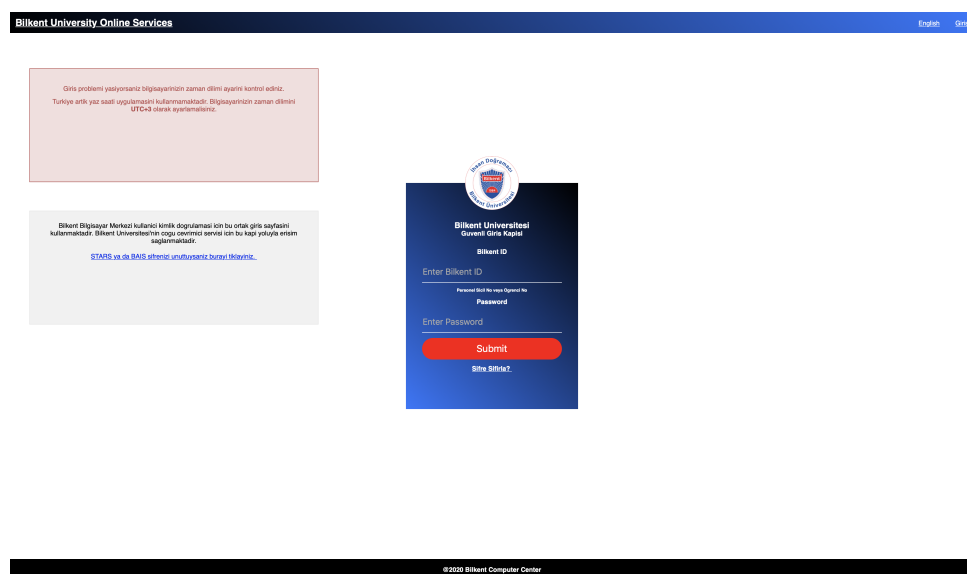


Figure 2: Screenshot of the Turkish version of the login page implemented

The application consists of different software elements like front-end and back-end part. The login functionality and the user interactions are being performed by these two sub parts integrated altogether. The following sections explains how these sub parts work with necessary UML diagrams.

2.1 Back-end part of the implementation

As the application implemented as our project consists of a login functionality, the most logical idea to perform this is by using a database for student records who will be allowed to enter to the SRS

system. For our application we established a MySQL database which is setup on a local server. That database consists of one table named students which consists of student records. The E/R diagram representation of the database table is shown below:

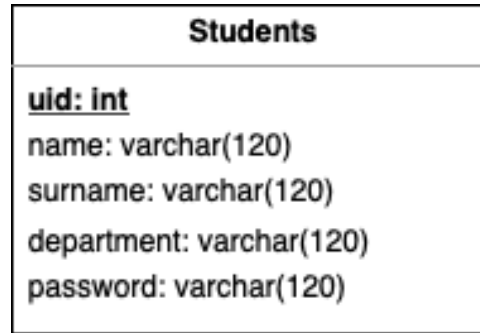


Figure 3: E/R diagram representation of the database table storing login information

Also the contents of the table created are also given below:

uid	name	surname	department	password
21602150	Ege	Ozcan	CS	admin
21602547	Cagla	Sozen	CS	admin
21602867	Yusuf	Dalva	CS	admin

By using these records from the MySQL database created for this project the webpage performs login operation. For the interaction between the webpage and the MySQL database, a local web development solution named MAMP has been used. As the implementation of the back-end functionality uses PHP, both the database and the webpage required a webpage to operate. For the webpage part, the back-end functionality and the front-end functionality has not been separated and implemented in a file named index.php (for Turkish version of the webpage, it is index-tr.php). The implementation of these files can be found in the Appendix. Also the database credentials are provided in the Appendix as a file named config.php.

2.2 Front-end part of the implementation

As mentioned in the part given above, the front-end and back-end functionality are implemented altogether. The implementation of the webpage can be found in the appendix with the mentioned PHP files. For the styling of the webpage, the layout of the webpage is taken as a model but the main layout has been changed without creating any difficulty for the user in terms of usability. This fact is not that testable for our use case, but the best effort has been made by testing the experience as the team members. The layout of the webpage is given in Figure 1 and Figure 2 respectively. The styling code can be found in the appendix with the file name style.css. The webpage is programmed with an intended behavior, which is the login functionality. In order to make this objective visual and more

understandable at the stage of coding, a Activity Diagram and a Use Case Diagram have been created. The diagram is given below:

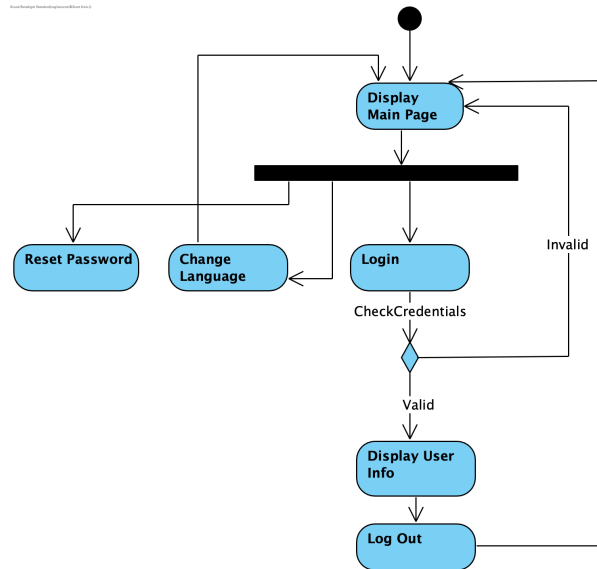


Figure 4: Activity Diagram of the System

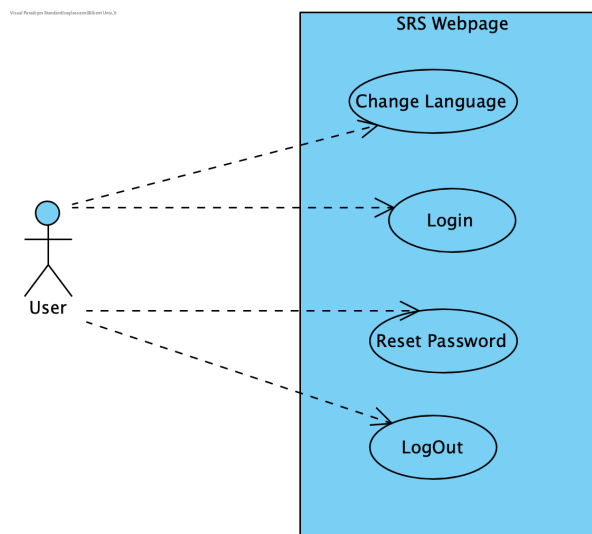


Figure 5: Use Case Diagram of the System

3 TEST CASES

As mentioned in the assignment, this project introduces 5 test cases for the application. As specified, for testing Selenium framework has been used. For the coding environment for the tests, Python has been selected since it is a language which every team member is comfortable in coding. As another tool, the Chrome Driver is being used to run the webpage for the tests. The initialization code is given below for clarity:

```
def setUp(self):
    parameter_path = os.path.join(sys.path[0], 'parameters.json')
    with open(parameter_path) as f:
        my_dict = json.load(f)
    op = webdriver.ChromeOptions()
    op.add_argument('headless')
    self.driver = webdriver.Chrome(my_dict["chromedriver"], options=op)
    self.driver.implicitly_wait(5)
    self.url = my_dict["url"]
```

This part of our report includes the ideas behind the test cases and the implementation details are given.

3.1 Test Case 1: User logs in to the system successfully

For the most optimistic test case for the login page, the user enters valid credentials and these credentials match with one of the records in the Students table that is introduced in *Section 2.1*. In case of a match the system is expected to be logged in to the system, which opens a webpage (main.php) with a welcome message in it. In the implementation, a record that is known that it is in the database is used as credentials using Selenium. For our case, the student record that is used is for the one that has name 'Ege' and surname 'Ozcan'. As the user is registered to the database that runs locally, it is expected that the login page is going to direct the user to the webpage with the welcome message. The code snippet for this test case is given below:

```
def test_successful_login(self):
    driver = self.driver
    driver.get(self.url)
    self.login(driver, "21602150", "admin")
    timeout = 5
    try:
        element_present = EC.presence_of_element_located((By.ID, 'Logout'))
        WebDriverWait(driver, timeout).until(element_present)
    except TimeoutException:
```

```

        print("Successful Login and Logout FAIL")
        raise TimeoutException
    except Exception:
        print("Successful Login and Logout FAIL")
        raise Exception
    try:
        text_area = driver.find_element_by_xpath("/html/body/center/h1")
        self.assertEqual(text_area.text, "Welcome 21602150")
    except Exception:
        print("Successful Login and Logout FAIL")
        raise Exception
    logout_button = driver.find_elements_by_xpath('//*[@id="Logout"]')[0]
    logout_button.click()
    try:
        self.assertTrue("Login" in driver.title)
        print("Successful Login and Logout OK")
    except Exception:
        print("Successful Login and Logout FAIL")
        raise Exception

```

In this test case, a successful login will be assessed by three different aspects. Initially, the form presented is filled with default credentials, and login button is pressed by using Selenium. As the first aspect of the test case, the browser is expected to be directed to the new webpage where the welcome message is presented to the user. Other than that, since the login operation is expected to be successful, the logout option has to be present with the welcome message. Finally, the title of the redirected webpage is examined to finally determine that the login operation is successful. As the final stage of the test case, the logout functionality has been used.

As the result of this test case the statement "Successful Login and Logout OK" as expected since the credentials are valid for the use case.

3.2 Test Case 2: A very long string entered as an input

In order to test that whether a very long string causes a crash as a result of web development environment properties, a test case is created. Here a Bilkent ID that is a very long string (500 characters) is inputted and the webpage is expected to be responding to this input. As it is not possible for our database to hold a 500 character username according to our table definition given in section 2.1. The test case is implemented in the following way:

```

def test_very_long_string(self):
    driver = self.driver

```



```
driver.get(self.url)
very_long_string = "x" * 500
self.login(driver, very_long_string, very_long_string)
try:
    alert = driver.switch_to.alert
    self.assertEqual(alert.text,
        "No such user with the given password in the system")
    print("Very Long String OK")
except Exception:
    print("Very Long String FAIL")
    raise Exception
```

In the implementation of the webpage, in order to prevent these kinds of problems, the input taken is filtered using a method called *mysqli_real_escape_string* which filters the escape string into a format which is applicable for database, this functionality is also being used for testing the effects of SQL Injection. As the string is formatted accordingly and the text boxes are selected in a non-limiting way the test case is expected to give a positive result. As the result of this test, the line "Very Long String OK" is printed as console output.

3.3 Test Case 3: Wrong Password Entered For a User

As our third test case, we have considered the case that for a valid user ID that is registered to the database is attempting to enter to the system with a password that is not valid for that user. For this use case, again the credentials for the user that has name attribute equal to "Ege" and surname attribute equal to "Ozcan" is used. For getting the information of this record, the primary key of the user is entered, which is the Bilkent ID. Then by specifying a wrong password for login, it is expected that the login operation will not be successful. In the implementation of the "index.php" page, the logic follows that if the user enters wrong credentials the system gives the error that indicates there is no such user in the form of an alert. The implementation of the test case in Selenium framework is given below:

```
def test_wrong_password(self):
    driver = self.driver
    driver.get(self.url)
    self.login(driver, "21602150", "amdin")
    try:
        alert = driver.switch_to.alert
        self.assertEqual(alert.text,
            "No such user with the given password in the system")
        print("Wrong Password OK")
```

```
except Exception:
    print("Wrong Password FAIL")
    raise Exception
```

As the result of the test case, it is expected that an alert with text "No such user with the given password in the system" will be entered. However there is no certain filter that checks whether Bilkent ID is valid in that case. It manually checks the ID, password pair and if there is no such pair the alert is raised. For our test case, with Bilkent ID 21602150 and password "amdin", the test case is expected to print the line "Wrong Password OK", which is the observed behavior in our test code.

3.4 Test Case 4: Loading the contents of the login page

As a trivial test for our application, the loading status of the webpage has been tested. The first action that the user performs in order to use the application is to enter to the webpage. In order to test that capability of the application a test case controlling whether the components of the webpage are being loaded successfully has been tested. As the webpage is coded in PHP, in case of an error in programming or a crash in the server (either Apache server or MySQL server), the webpage will not open. In order to test that, we decided to come up with a test case to check that the web page functions error free at the first place. Another objective of this test is to check whether the capabilities used in our webpage are supported or attempting to use them result with error in the current setting. The implementation of the test case is given below:

```
def test_page_opens(self):
    driver = self.driver
    driver.get(self.url)
    try:
        self.assertTrue("Login" in driver.title)
        driver.find_elements_by_xpath("/html/body/div[1]/img")
        print("Web Page Loading OK")
    except Exception:
        print("Web Page Loading FAIL")
```

As a result of this test case, if the webpage is loaded successfully (error-free), the testing script would output a line to the console which has the contents of the string "Web Page Loading OK". This was the result that was expected and when the test script executed the expected result was obtained by the test. After testing, when we had a look at the application manually we also saw that the web page is error-free initially. This also shows the truth of the test case.

3.5 Test Case 5: Protection Against SQL Injection

For our final test case for the login page, one of the common methods that are being used for skipping authentication stages, SQL Injection, and its effect on our program has been tested. In order to

test the resistance of the program against this threat, the user name and password "" **OR 1=1** is entered which makes the SQL statement return a true value for the authentication information. If there would be no extra measurements for SQL Injection the user can skip authentication and the program would respond in unexpected behavior. As a measure against SQL Injection, the method **mysqli_real_escape_string** is used. This method detects whether SQL Injection is intended and if so, formats the input accordingly such that the attempt by the hackers would not result in success. The implementation of the test case is given below:

```
def test_sql_injection(self):
    driver = self.driver
    driver.get(self.url)
    injection = ' "" or 1=1 '
    self.login(driver, injection, injection)
    try:
        alert = driver.switch_to.alert
        self.assertEqual(alert.text,
            "No such user with the given password in the system")
        print("SQL Injection OK")
    except Exception:
        print("SQL Injection FAIL")
        raise Exception
```

As our program does take the necessary precautions against SQL Injection, the input line entered would be expected to be treated as wrong credentials. Just like in the previous test cases, in this case application is expected to raise an alert indicating that the credentials entered by the user are wrong. As a result, for our test case it is expected that a single line to the console will be outputted with the content "SQL Injection OK". Just as expected, when we run the test case the mentioned line is outputted.

4 EVALUATION

For this project, the application that we have developed was a basic login application that applies a basic level of authentication. There were basic functionalities that are applied by our webpage which enables the user to log in to the system. As we have not implemented the whole SRS system, when the user logs in a symbolic page that represents the user with a certain Bilkent ID logged in to the system. As mentioned before the application was developed as a PHP application. With the constraints of the language, detecting an error was not an easy task just like identifying an ordinary problem was not easy either.

Also since the application developed was a web based application, testing would be a problem since without a framework only manual tests could be performed as the web application was not composed

of modules that perform certain functions. The webpage implemented contains all of the implemented functionality in a single file and it is not distinguishable in terms of dividing it into simpler units. Due to this fact, the functionalities that Selenium offers was helpful in terms of verifying the webpage in terms of its function. As the verification concept corresponds to assuring that the application meets the pre-determined specifications. In order to do that, the initial step we took for the project was to think about the qualifications that the software needed to meet. That was the point where we came up with the 5 test cases that are introduced in the third section of this report. While determining the test cases we have considered the quality goals introduced in the coverage of the course and came up with test cases that look for different aspects of dependability requirements for the software. The relation between each requirement and the dependability requirement is given below:

- **Correctness:** In order to ensure that the application responds with correct behavior. The test cases that the user enters to the system with correct credentials (Bilkent ID and Password match) and false credentials (Password is wrong for a valid Bilkent ID) whether the login functionality can be performed correctly.
- **Reliability:** For our application the reliability and correctness factors do not differ significantly and the test cases mentioned for the correctness criteria is also valid for the reliability criteria. As the distinguishing units is not possible for our application th correctness and reliability criteria did not differ.
- **Safety:** For the safety criteria, the main issue was granting access to a user with credentials that are not valid. In order to maintain this criteria, the test case regarding whether the application is resistant to SQL injection is implemented. By ensuring that this test case does not fail, any illegal access for our application by maintaining the queries sent to the database will be taken under control.
- **Robustness:** For the robustness criteria, the main test case was involving whether the webpage was operating as a web element, without considering the login functionality. For this criteria, a test case which checks whether the page elements are loaded is implemented. This case checks that the webpage is accessible no matter what.

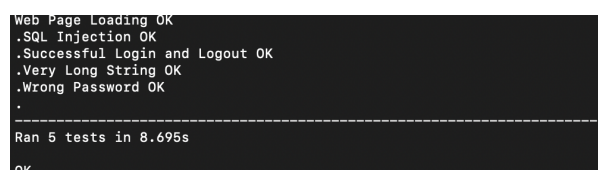
The other aspect of external qualities to be checked consist of usefulness of the application. As the application was not a sophisticated application and at this stage no internal quality factor like maintainability is considered, the internal quality testing was not considered. In terms of usefulness of the application, all of the team members used the application for logging in to see whether there is a problem with the user experience. This was not a perfect approach since we are also the developers of the project but this was the best effort that we could have come with.

In our testing experience regarding dependability, which was tested by the test cases that we have implemented, the automated testing process helped on checking the cases that we have written. By the help of a simple script, we eliminated the requirement of testing the webpage functionality

by hand after making the changes to the application. After implementing the core functionality of the application, with every change we made running the test script was enough to verify the webpage implemented. Also with some of the functionalities that Selenium offers (like inspecting DOM elements) testing that cannot be done in a feasible way with human eye could be performed. Other than this capability, with the web page manipulation abilities of Selenium some certain inputs could be entered automatically. By this way, the whole testing process has been automated and could be done around 10 seconds after each change we have performed. In addition to that, since we have created the test cases before programming the webpage, the application implemented in a way that is compatible with the test cases written. By this way, we came up with a way that keeps the application complying to the requirements and testing it in an easy and a practical way.

5 DISCUSSION

Resulting from our experience in testing, we experienced several benefits of the automated testing on our project. As explained in the previous section, in terms of velocity Selenium framework gives certain advantages. Shortly, with the predefined test cases in a script, they can be executed quickly rather than performing tests manually. To be more specific about the advantage gained in terms of velocity, the whole testing process took around 9 seconds on average which would not be possible without automated tests. Furthermore for the quality aspect, automated testing is more consistent, efficient and reliable compared to manual testing for manual testing may fail to perform specific cases repeatedly occasionally whereas automated testing performs these tests without any possible deviations. Therefore improves the verification as well as assuring the quality.



```
Web Page Loading OK
.SQL Injection OK
.Successful Login and Logout OK
.Very Long String OK
.Wrong Password OK
.
-----
Ran 5 tests in 8.695s
OK
```

Figure 6: Output received at console after executing tests

6 APPENDIX

File index.php:

```
<?php
include "config.php";
session_start();
if(isset($_POST['submitCred'])) {
    $user = mysqli_real_escape_string($mysqli, $_POST['bil_id']);
    $pass = mysqli_real_escape_string($mysqli, $_POST['pass']);
```

```
if (\$user != "" && \$pass != "") {
    \$user = strtolower(\$user);
    \$cred_query = "SELECT uid, password FROM students
WHERE uid = '" . \$user . "' and password = '" . \$pass . "'";
    \$result = mysqli_query(\$mysqli, \$cred_query);
    \$flag = mysqli_fetch_row(\$result);
    if(\$flag === null) {
        echo '<script language="javascript">';
        echo 'alert("No such user with the
given password in the system)';
        echo '</script>';
    } else{
        \$_SESSION['user_name'] = \$user;
        \$_SESSION['password'] = \$pass;
        header('Location: main.php');
    }
}
else
{
    echo '<script language="javascript">';
    echo 'alert("Please enter a username and a password)';
    echo '</script>';
}
}

?>
```

```
<html>
<head>
  <title>Bilkent STARS Login</title>
  <link rel = "stylesheet" type = "text/css" href = "style.css">
  <script type="text/javascript">
    function checkCred()
    {
      var bil_id = document.forms["login_form"]["bil_id"].value;
      var pass = document.forms["login_form"]["pass"].value;
      if (bil_id==null || pass=="")
      {
        alert("ID can't be blank");
        return false;
      }
      else if (pass==null || bil_id=="")
      {
        alert("Password can't be blank");
        return false;
      }
      return true;
    }
  </script>
<body>
  <div class = "loginbox">
    
    <h1> Bilkent University </h1>
    <h2> Secure Login Gateway </h2>
    <form name="login_form" method="post" action="">
      <p> Bilkent ID </p>
      <input type = "text" name="bil_id"
        placeholder="Enter Bilkent ID" >
      <label for="id">Personnel ID number or
        Student ID number</label>
      <p> Password </p>
      <input type = "password" name="pass"
        placeholder="Enter Password" >
      <input type = "submit" name="submitCred"
        placeholder="Login" >
```

```

        <a href= "resetpage.html"> Reset password? </a><br>
    </form>

</div>
<div class = "topbar">
    <form>

        <a href= "#" class = "onservices"> Bilkent University
        Online Services </a><br>
        <a href= "index_tr.php" class = "lang"> Turkce </a><br>
        <a href= "#" class = "log"> Login </a><br>

    </form>

</div>
<div class = "botbar">
    <h1> @2020 Bilkent Computer Center <h1>
</div>
<div class = "warningbox">
    <h1> If you are having problems logging in, please check the time zone
    setting on your computer. <h1>
    <h1> Turkey no longer uses daylight saving time (DST).
    Your computer time zone setting should be set as <b>UTC+3</b>. <h1>
</div>
<div class = "infobox">
    <h1> Bilkent Computer Center uses this common login gateway page
    for user authentication. Most Bilkent University online services are accessed
    <a href= "resetpage.html"> If you have forgotten your
    STARS or BAIS password, please click here. </a><br>
</div>
</body>
</head>
</html>

```

File config.php:

```

<?php
    // The MySQL database is setup locally.
    // The credentials for that database is given in this file
    $host_name = "localhost";
    $username = "root";
    $password = "10lovers";
    $db_name = "bilkent";

```



```
$mysqli = mysqli_connect($host_name, $username, $password, $db_name);  
if (!$mysqli) {  
    echo "Error: Unable to connect to MySQL." . PHP_EOL;  
    echo "Debugging errno: " . mysqli_connect_errno() . PHP_EOL;  
    echo "Debugging error: " . mysqli_connect_error() . PHP_EOL;  
    exit;  
}  
?>
```

File main.php:

```
<?php  
include "config.php";  
session_start();  
function logOut() {  
    echo "You are logging out of the system...";  
    session_destroy();  
    header('Location: index.php');  
}  
$user = $_SESSION['user_name'];  
$userPrint = strtoupper($user);  
$c_id = $_SESSION['password'];  
if(!isset($user)){  
    header('Location: index.php');  
}  
if(array_key_exists('Logout',$_POST)){  
    logOut();  
}  
?>
```

```
<!doctype html>
<html>
<head></head>
<body>
<center> <h1>Welcome <?php echo \$userPrint ?> </h1></center>
<form method="post"> <center>
<input type="submit" name="Logout" id="Logout" value="Logout" /><br/>
</form>
</body>
</html>
```

File style.css:

```
body{
    margin: 0;
    padding: 0;
    background-image: white;
    background-size: cover;
    background-position: center;
    font-family: sans-serif;
}
.loginbox{
    width: 320px;
    height: 420px;
    background: linear-gradient(45deg, #3374ff, black);
    color: #fff;
    top: 50%;
    left: 50%;
    position: absolute;
    transform: translate(-50%, -50%);
    box-sizing: border-box;
    padding: 70px 30px ;
}
.topbar{
    width: 100%;
    height: 10px;
    background: linear-gradient(45deg, black, #3374ff);
    color: #fff;
    top: 0%;
```

```
        left: 0%;
        position: absolute;
        box-sizing: border-box;
        padding: 20px 30px ;
    }
    .botbar{
        width: 100%;
        height: 5px;
        background: black;
        color: #fff;
        bottom: 0%;
        left: 0%;
        position: absolute;
        box-sizing: border-box;
        padding: 20px 30px ;
    }
    .warningbox{
        width: 30%;
        height: 20%;
        background: #f2dede;
        color: #fff;
        top: 10%;
        left: 2%;
        position: absolute;
        box-sizing: border-box;
        padding: 20px 30px ;
        border: solid;
        border-color: #b84e4e;
        border-width: 1px;
    }
    .infobox{
        width: 30%;
        height: 20%;
        background: #f1f1f1;
        top: 35%;
        left: 2%;
        position: absolute;
        box-sizing: border-box;
```

```
padding: 20px 30px ;
border: solid;
border-width: 1px;
border-color: #e5e5e5;
}
.resetbox{
width: 400px;
height: 250px;
background: linear-gradient(45deg, #3374ff, black);
color: #fff;
top: 50%;
left: 50%;
position: absolute;
transform: translate(-50%, -50%);
box-sizing: border-box;
padding: 70px 30px ;
}
.onservices{
border: none;
color: white;
font-size: 18px;
top: 0%;
left: 0%;
margin-top: 10px;
margin-left: 10px;
position: absolute;
font-weight: bold;
}
.lang{
border: none;
color: white;
font-size: 12px;
top: 0%;
right: 3%;
margin-right: 10px;
margin-top: 15px;
position: absolute;
}
```

```
.log{
    border: none;
    color: white;
    font-size: 12px;
    top: 0%;
    right: 0%;
    margin-right: 10px;
    margin-top: 15px;
    position: absolute;
}
.logo{
    width: 100px;
    height: 100px;
    border-radius: 50%;
    position: absolute;
    height: -50px;
    left: calc(50% - 50px);
    top: -50px;
}
.loginbox h1{
    margin: 0;
    text-align: center;
    font-size: 15px;
}
.botbar h1{
    margin: -7px;
    text-align: center;
    font-size: 12px;
    top : 0%;
}
.loginbox h2{
    margin: 0;
    text-align: center;
    font-size: 12px;
    padding: 0 0 10px;
}
.warningbox h1{
    margin: 10 0 0px;
```

```
        text-align: center;
        font-size: 12px;
        color: #b84e4e;
        font-weight: normal;
    }
    .infobox h1{
        margin: 0 0 15px;
        text-align: center;
        font-size: 12px;
        color: black;
        font-weight: normal;
    }
    .resetbox h1{
        margin: 0;
        text-align: center;
        font-size: 15px;
    }
    .resetbox h2{
        margin: 0;
        text-align: center;
        font-size: 12px;
        padding: 0 0 10px;
    }
    .loginbox p{
        margin: 10px;
        padding: 0;
        font-weight: bold;
    }
    .resetbox p{
        margin: 10px;
        padding: 0;
        font-weight: bold;
    }
    .loginbox label[for = "id"]{
        font-size: 8px;
        vertical-align: bottom;
        padding: 0 0 15px;
    }
```

```
.loginbox input{
    width: 100%;
    margin-bottom: 10px;
}

.resetbox input{
    width: 40%;
    margin-bottom: 10px;
    margin-top: 20px;
    margin-right: 10px;
    margin-left: 10px;
}

.loginbox input[type = "text"], input[type = "password">{
    border: none;
    border-bottom: 1px solid #fff;
    background: transparent;
    outline: none;
    height: 40px;
    color: white;
    font-size: 16px;
}

.loginbox input[type = "submit">{
    border: none;
    border-radius: 20px;
    background: red;
    outline: none;
    height: 40px;
    color: #fff;
    font-size: 18px;
}

.resetbox input[type = "submit">{
    border: none;
    border-radius: 20px;
    background: red;
    outline: none;
    height: 40px;
    color: #fff;
    font-size: 75\%;
```

```
}
.loginbox input[type = "submit"]:hover{
    background: white;
    color: #000;
}
.resetbox input[type = "submit"]:hover{
    background: white;
    color: #000;
}
.loginbox a{
    color: white;
}
.loginbox a:hover{
    color: red;
}

import os
import sys
import json
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

class SRSTest(unittest.TestCase):
    @classmethod
    def setUp(self):
        parameter_path = os.path.join(sys.path[0], 'parameters.json')
        with open(parameter_path) as f:
            my_dict = json.load(f)
        op = webdriver.ChromeOptions()
        op.add_argument('headless')
        self.driver = webdriver.Chrome(my_dict["chromedriver"], options=op)
        self.driver.implicitly_wait(5)
        self.url = my_dict["url"]
    @classmethod
```



```
def tearDown(self):
    self.driver.quit()
def test_successful_login(self):
    driver = self.driver
    driver.get(self.url)
    self.login(driver, "21602150", "admin")
    timeout = 5
    try:
        element_present = EC.presence_of_element_located((By.ID, 'Logout'))
        WebDriverWait(driver, timeout).until(element_present)
    except TimeoutException:
        print("Successful Login and Logout FAIL")
        raise TimeoutException
    except Exception:
        print("Successful Login and Logout FAIL")
        raise Exception
    try:
        text_area = driver.find_element_by_xpath("/html/body/center/h1")
        self.assertEqual(text_area.text, "Welcome 21602150")
    except Exception:
        print("Successful Login and Logout FAIL")
        raise Exception
    logout_button = driver.find_elements_by_xpath('//*[@id="Logout"]')[0]
    logout_button.click()
    try:
        self.assertTrue("Login" in driver.title)
        print("Successful Login and Logout OK")
    except Exception:
        print("Successful Login and Logout FAIL")
        raise Exception
def test_very_long_string(self):
    driver = self.driver
    driver.get(self.url)
    very_long_string = "x" * 500
    self.login(driver, very_long_string, very_long_string)
    try:
        alert = driver.switch_to.alert
        self.assertEqual(alert.text, "No such user with the given password")
```

```
        "in the system")
    print("Very Long String OK")
except Exception:
    print("Very Long String FAIL")
    raise Exception

def test_wrong_password(self):
    driver = self.driver
    driver.get(self.url)
    self.login(driver, "21602150", "amdin")
    try:
        alert = driver.switch_to.alert
        self.assertEqual(alert.text, "No such user with the given "
                                   "password in the system")
        print("Wrong Password OK")
    except Exception:
        print("Wrong Password FAIL")
        raise Exception

def test_page_opens(self):
    driver = self.driver
    driver.get(self.url)
    try:
        self.assertTrue("Login" in driver.title)
        driver.find_elements_by_xpath("/html/body/div[1]/img")
        print("Web Page Loading OK")
    except Exception:
        print("Web Page Loading FAIL")

def test_sql_injection(self):
    driver = self.driver
    driver.get(self.url)
    injection = ' "" or 1=1 '
    self.login(driver, injection, injection)
    try:
        alert = driver.switch_to.alert
        self.assertEqual(alert.text, "No such user with the given "
                                   "password in the system")
        print("SQL Injection OK")
    except Exception:
        print("SQL Injection FAIL")
```

```
        raise Exception

    @classmethod
    def login(cls, driver, username, password):
        user_input = driver.find_element_by_name("bil_id")
        password_input = driver.find_element_by_name("pass")
        user_input.clear()
        user_input.send_keys(username)
        password_input.clear()
        password_input.send_keys(password)
        password_input.send_keys(Keys.RETURN)

if __name__ == "__main__":
    unittest.main()
```

REFERENCES

- [1] "The Selenium Browser Automation Project," *The Selenium Browser Automation Project :: Documentation for Selenium*. [Online]. Available: <https://www.selenium.dev/documentation/en/>. [Accessed: 02-Mar-2020].
- [2] MAMP GmbH, "MAMP & MAMP PRO - your local web development solution for PHP and Word-Press development," *MAMP & MAMP PRO - Your local web development solution*. [Online]. Available: <https://www.mamp.info/en/>. [Accessed: 02-Mar-2020].
- [3] Pezzè Mauro and M. Young, *Software testing and analysis: process, principles, and techniques*. Hoboken, NJ: Wiley, 2008.