CS 458
Software Validation and Verification

Introduction to Test Automation (WEB)

Okan Şen	21202377
Hassam Abdullah	21701610
Mustafa Tuna Acar	21703639
Ömer Olkun	21100999

Introduction to Test Automation (WEB)

1 Code Details/Information

1.1 Screenshot of Web Application

Login Page

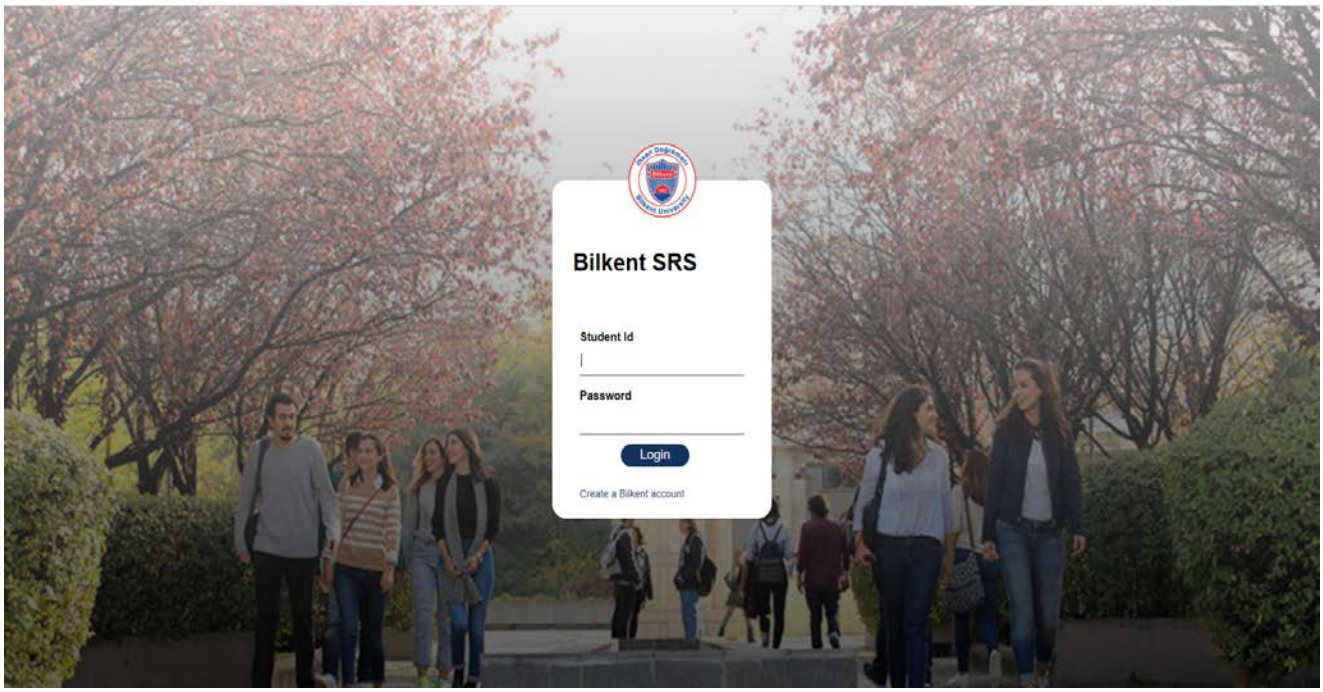


Figure 1: Login Page View

This is the Website's main login page from where the user can redirect to either the registration page by clicking on the "Create a Bilkent Account" link or login successfully by entering valid credentials and getting redirected to the Success page.

Registration Page

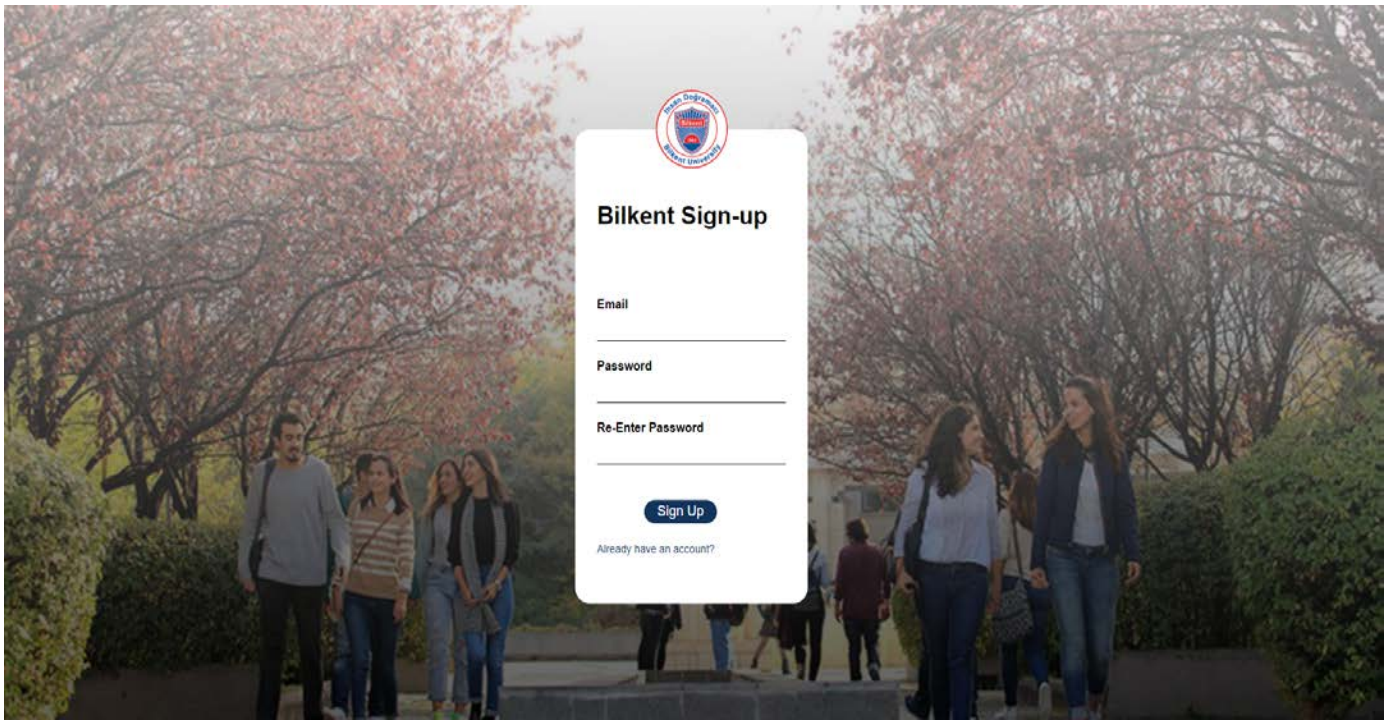


Figure 2: Sign-up View

This is the website's Registration page where the user can enter his email/password to receive a unique Bilkent ID which he can use as valid credentials for the Login page. The Sign Up button registers the user with the credentials he adds in the form, otherwise the user can go back to the Login page by clicking on the "Already have an account?" link.

Success Page

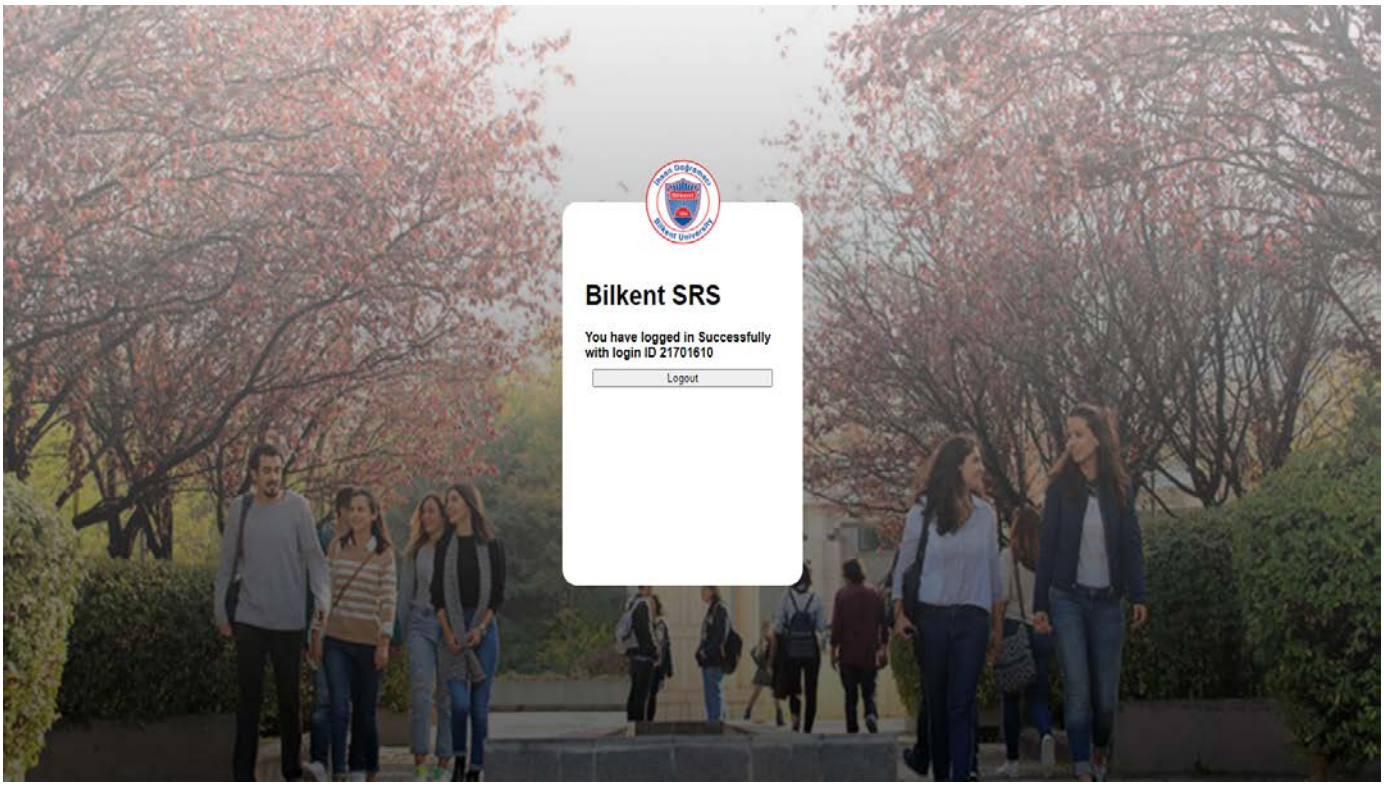


Figure 3: Successful Login View

This is the website's Success page and the user is redirected to this after he enters valid credentials in the Login page that are verified in the database. The user has to press the Logout button to exit the session otherwise the user will remain logged in and will not be able to access the login page. Furthermore, the user will be automatically logged out if he remains inactive for a specific duration of time.

1.2 Excerpts of Web Application Code

Important excerpts of code containing php/html/javascript etc can be viewed below that are integral to the functioning and understanding of the website.

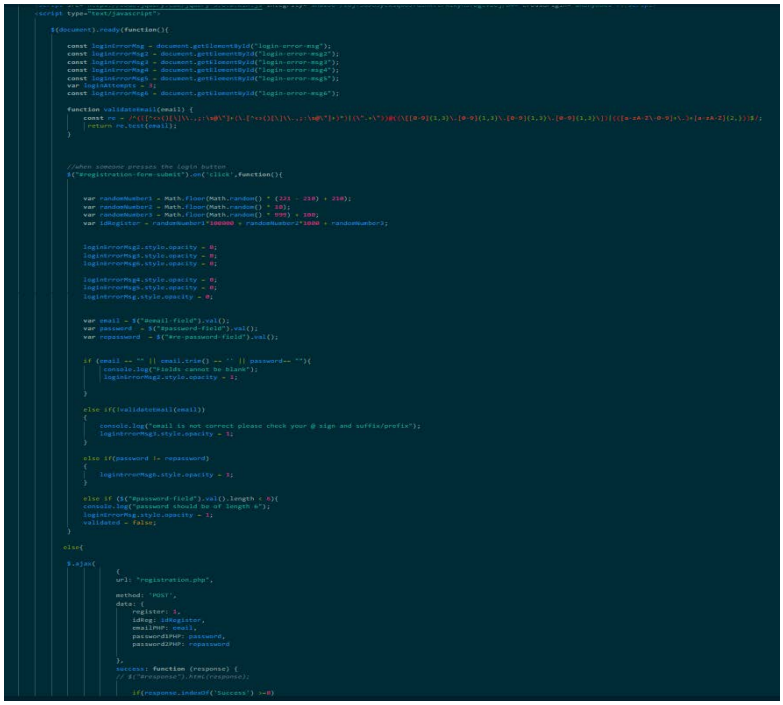
The server.php file acts as a server for both the login and registration page since the database is connected to them and Figure 4. shows a snippet of code that shows the server-side logic implemented for login.php. Figure 5. shows the html code which is standard in all the other files as well. Lastly, Figure 6. shows the javascript code that uses ajax and jquery for server to client and database communication in both Login.php and Registration,php, while Success.php only uses Javascript/Jquery since it does not need to communicate with the database.

```php
//THIS DEALS WITH LOG IN LOGIC
//if already logged in
if(isset($_SESSION['loggedIN'])) {
    echo "Im inside here";
    header('Location: Success.php');
    exit();
}


    //connect php and mysqllite
    if(isset($_POST['login'])){ //everything is fine ready to login
        $connection = new mysqli('localhost','root','', 'srs');



        $id = $connection->real_escape_string($_POST['idPHP']);
        $password = $connection->real_escape_string($_POST['passwordPHP']);

        $sql="select * from loginform where Username ='".$id."'AND Password='".$password."'
        limit 1";

        $result = mysqli_query($connection, $sql);

        if(mysqli_num_rows($result)!=0){
        $_SESSION['loggedIN'] = 1;
        $_SESSION['id'] = $id;
            exit("Success");
        }

        else{

            exit("Failure");
        }


    }
```

```html
<html lang="en">
    <head>

        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Login Form</title>
        <link rel="stylesheet" type="text/css" href="registration_style.css">


    </head>
    <body>


        <div class="loginbox">
            <img src="bilkent2.png" class="avatar">

                <h1 id="login-header">Bilkent Sign-up </h1>

                <div id="login-error-msg-holder">
                    <p id="login-error-msg">Password is too short (minimum is 6 characters). </p>
                    <p id="login-error-msg2">Any field cannot be blank!</p>
                    <p id="login-error-msg3">Email is not correct please check your @ sign and suffix/prefix</p>
                    <p id="login-error-msg4">The email id already exists</p>
                    <p id="login-error-msg5">The email is now registered</p>
                    <p id="login-error-msg6">The passwords do not match check your password entries!</p>
                </div>


                <form method="post" id="registration-form" action="registration.php">
                    <p id=id-header>Email</p>
                    <input type="text" name="email" id="email-field" class="registration-form-field" placeholder="">
                    <p id=id-pass>Password</p>
                    <input type="password" name="password" id="password-field" class="registration-form-field" placeholder="">
                    <p id=id-repass>Re-Enter Password</p>
                    <input type="password" name="re-password" id="re-password-field" class="registration-form-field" placeholder="">
                    <input type="button" value="Sign Up" id="registration-form-submit">
                    <a href="login.php" class= "ca"> Already have an account?</a>
                </form>
```

Figure 4: PHP code in Server.php

Figure 5: Html Code in Registration.php



```javascript
<script type='text/javascript">

    $(document).ready(function(){

        const loginErrorMsg = document.getElementById("login-error-msg");
        const loginErrorMsg2 = document.getElementById("login-error-msg2");
        const loginErrorMsg3 = document.getElementById("login-error-msg3");
        const loginErrorMsg4 = document.getElementById("login-error-msg4");
        const loginErrorMsg5 = document.getElementById("login-error-msg5");
        var loginAttempts = 3;
        const loginErrorMsg6 = document.getElementById("login-error-msg6");

        function validateEmail(email) {
            const re = /^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;
            return re.test(email);
        }


        //when someone presses the login button
        $("#registration-form-submit").on('click',function(){


            var randomNumber1 = Math.floor(Math.random() * (221 - 210) + 210);
            var randomNumber2 = Math.floor(Math.random() * 10);
            var randomNumber3 = Math.floor(Math.random() * 999) + 100;
            var idRegister = randomNumber1*100000 + randomNumber2*1000 + randomNumber3;


            loginErrorMsg2.style.opacity = 0;
            loginErrorMsg3.style.opacity = 0;
            loginErrorMsg5.style.opacity = 0;

            loginErrorMsg4.style.opacity = 0;
            loginErrorMsg6.style.opacity = 0;
            loginErrorMsg.style.opacity = 0;


            var email = $("#email-field").val();
            var password  = $("#password-field").val();
            var repassword  = $("#re-password-field").val();


            if (email == "" || email.trim() == "" || password== ""){
                console.log("Fields cannot be blank");
                loginErrorMsg2.style.opacity = 1;
            }

            else if(!validateEmail(email))
            {
                console.log("email is not correct please check your @ sign and suffix/prefix");
                loginErrorMsg3.style.opacity = 1;
            }

            else if(password != repassword)
            {
                loginErrorMsg6.style.opacity = 1;
            }

            else if ($("#password-field").val().length < 6){
            console.log("password should be of length 6");
            loginErrorMsg.style.opacity = 1;
            validated = false;
            }

        else{

            $.ajax(
                    {
                    url: "registration.php",

                    method: 'POST',
                    data: {
                        register: 1,
                        idReg: idRegister,
                        emailPHP: email,
                        passwordPHP: password,
                        passwordPHP: repassword

                    },
                    success: function (response) {
                    // $("#response").html(response);

                        if(response.indexOf('Success') >=0)
```

Figure 6: Javascript code using Jquery and Ajax in Registration.php

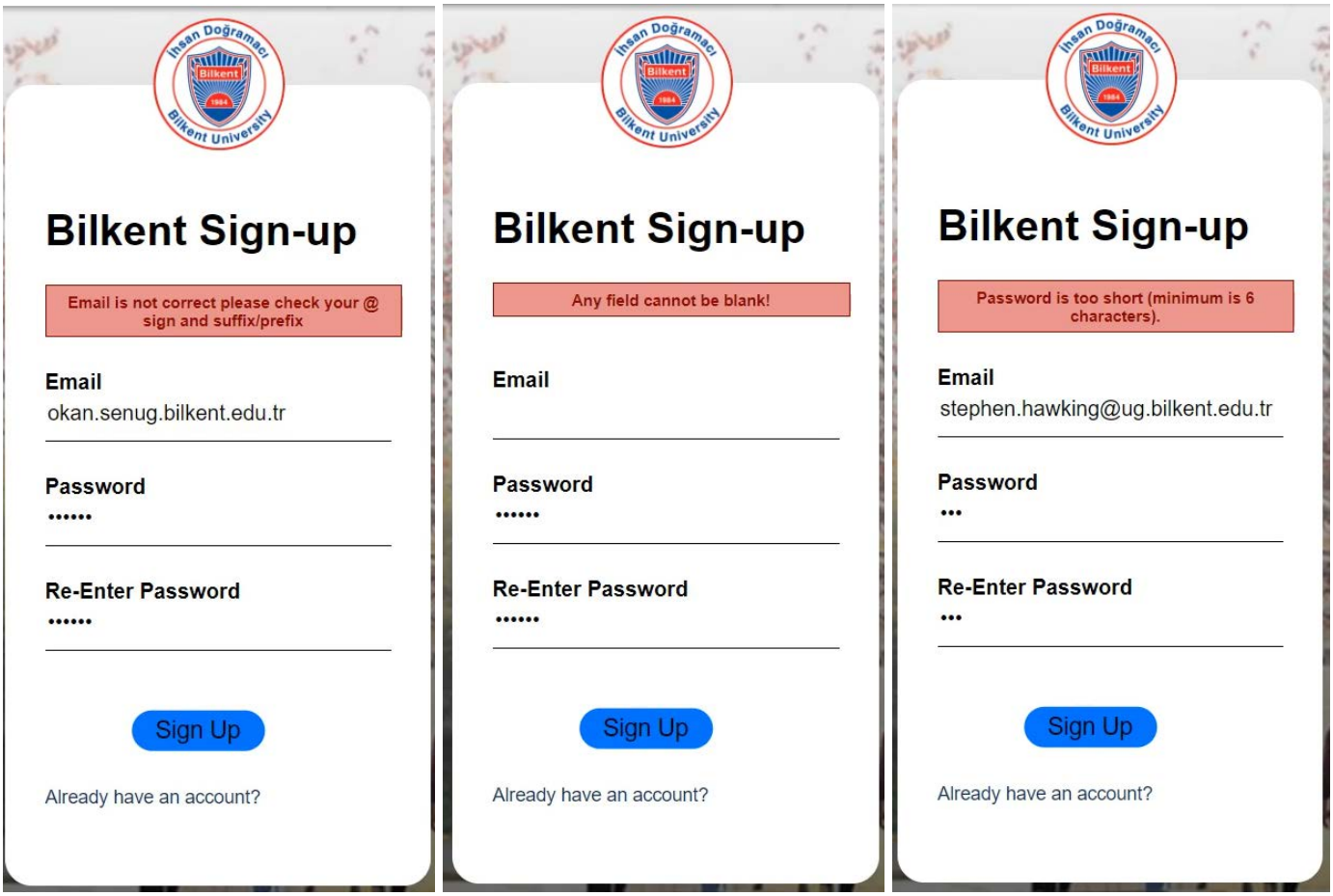
1.3 Screenshot of Test Code and Test Cases

We have implemented a registration system to our website in order to have a more immersive experience when testing out the cases. This is why we will start testing with the registration page. In order to explain what is happening in these following screenshots, the test cases are listed below.

- Test Cases

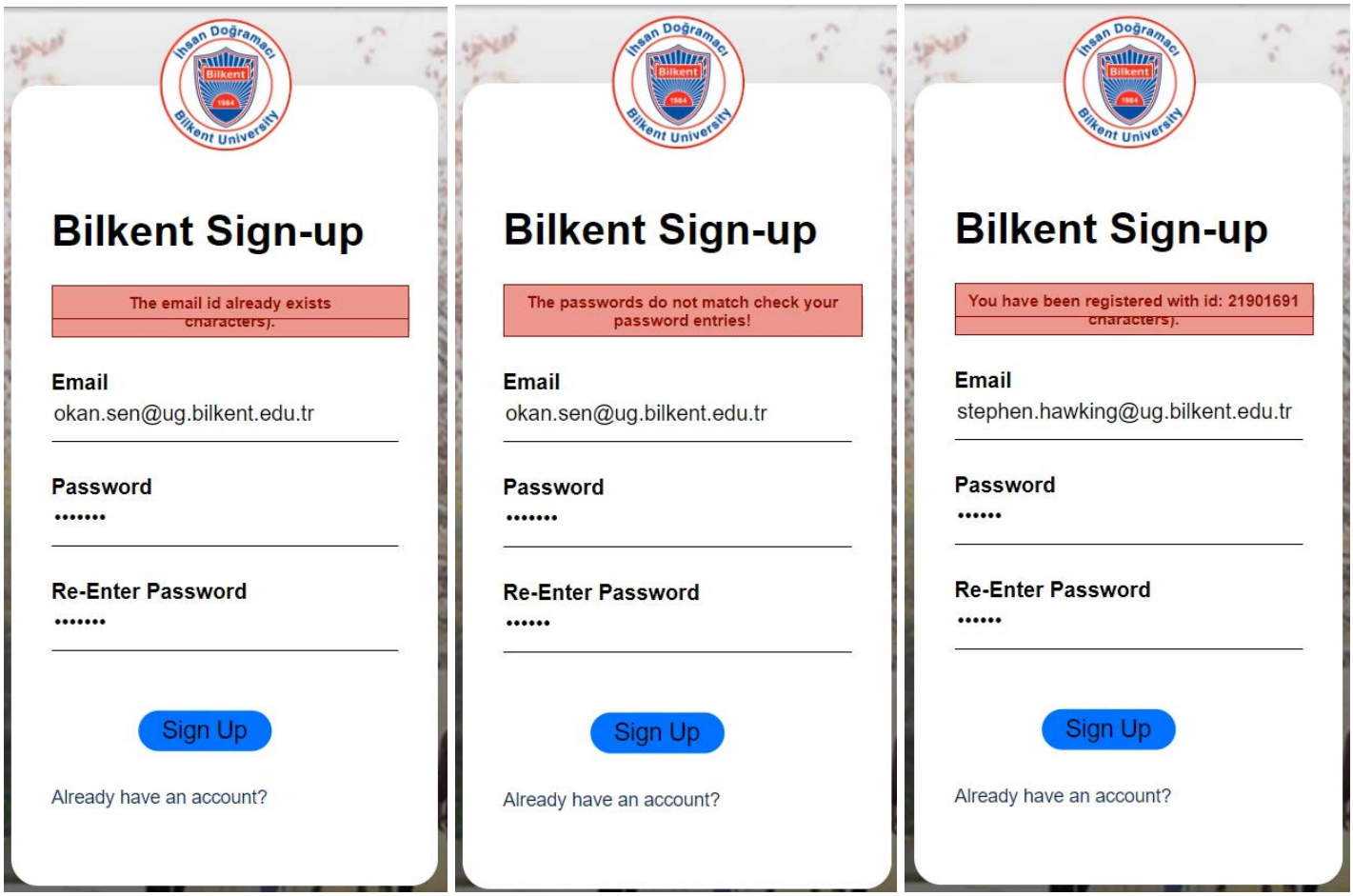
- Wrong/correct credentials test case for login and register pages
- No multi sessions/no multiple different account logins at same machine allowed/ Cannot use back to go to login page after activating a session/ Cannot use back to get in user page after logout - sessions system
- Timeout for multiple failed login tries
- Enter key is login / either id or password fields should be active when pressed enter / by default id field should be active
- Automatic logout after x amount of time of inactivity - during testing it is set to 3 seconds

1.3.1 Wrong/correct credentials Test Case



Figures 7, 8, 9: Invalid Entries For Registration

- On the registration page, the entered email should have an '@' sign in it to be considered a valid email.
- The fields cannot be left empty. Whitespaces count as empty, as well.
- If however, the entered email is valid, the password must be at least 6 characters long for security reasons.



Figures 10, 11, 12: Invalid and a valid register

- When the submission is sent to the server, the entered values are checked, and if the email is in use within our database, the user cannot get a new account with that email.

- If the passwords entered do not match the user gets an error.

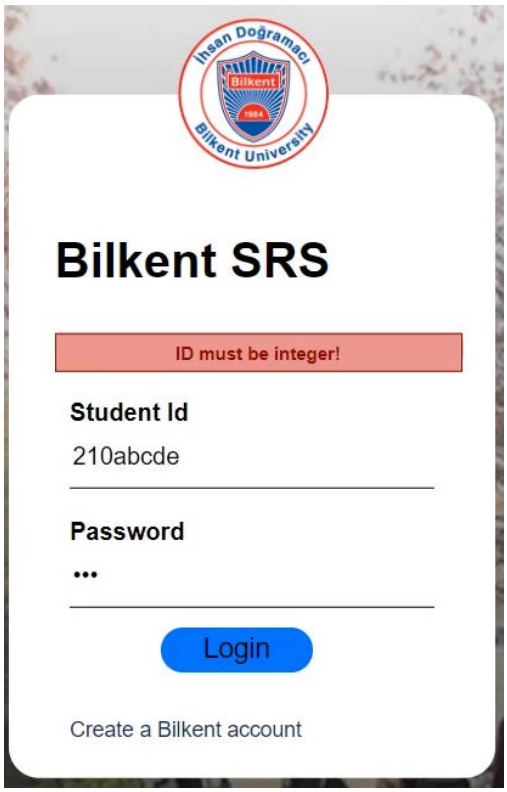
- If all conditions are met, the system generates the user a random available id, as can be seen in figure X.

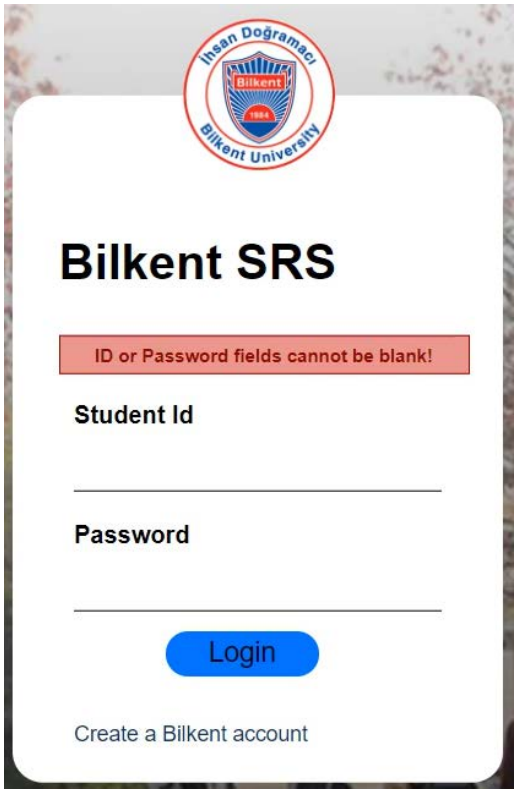
	Id	Username	Password	Email
Edit Copy Delete	1	21202377	asd1asd	okan.sen@ug.bilkent.edu.tr
Edit Copy Delete	3	21902025	91942803	merve.tural@ug.bilkent.edu.tr
Edit Copy Delete	5	21809595	asdasdasd	jack.ripper@ug.bilkent.edu.tr
Edit Copy Delete	7	21206567	olmholmh	nina.sky@ug.bilkent.edu.tr
Edit Copy Delete	11	21501931	bbbbbb	walter.white@ug.bilkent.edu.tr
Edit Copy Delete	14	21701610	asdasdasd	hassam.abdullah@ug.bilkent.edu.tr

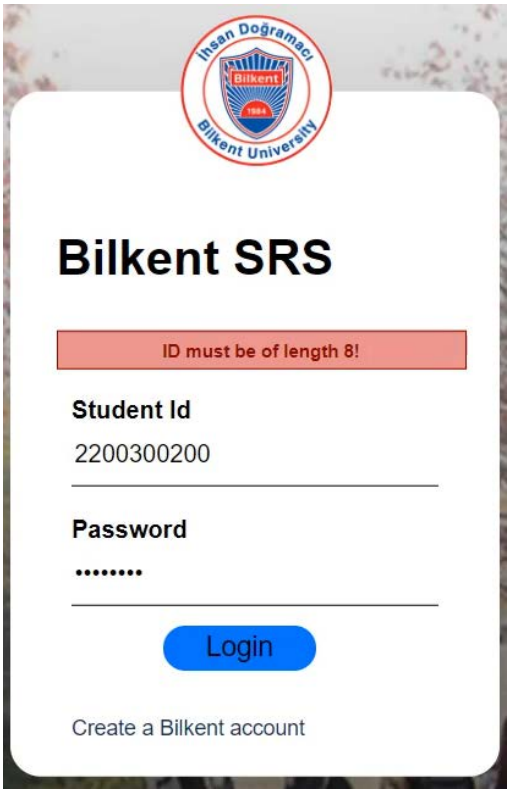
Figure 13: Database before any entry

		Id	Username	Password	Email
☐	Edit Copy Delete	1	21202377	asd1asd	okan.sen@ug.bilkent.edu.tr
☐	Edit Copy Delete	3	21902025	91942803	merve.tural@ug.bilkent.edu.tr
☐	Edit Copy Delete	5	21809595	asdasdasd	jack.ripper@ug.bilkent.edu.tr
☐	Edit Copy Delete	7	21206567	olmholmh	nina.sky@ug.bilkent.edu.tr
☐	Edit Copy Delete	11	21501931	bbbbbb	walter.white@ug.bilkent.edu.tr
☐	Edit Copy Delete	14	21701610	asdasdasd	hassam.abdullah@ug.bilkent.edu.tr
☐	Edit Copy Delete	28	21007922	asdasd	stephen.hawking@ug.bilkent.edu.tr

Figure 14: Database after a successful new registration

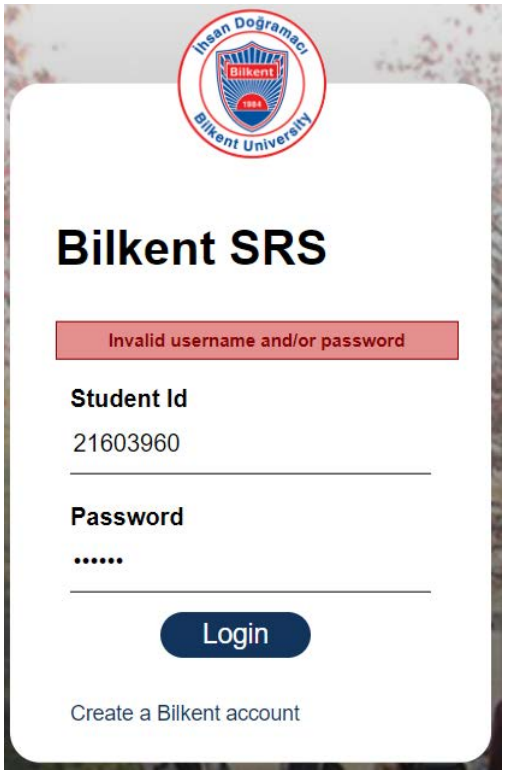


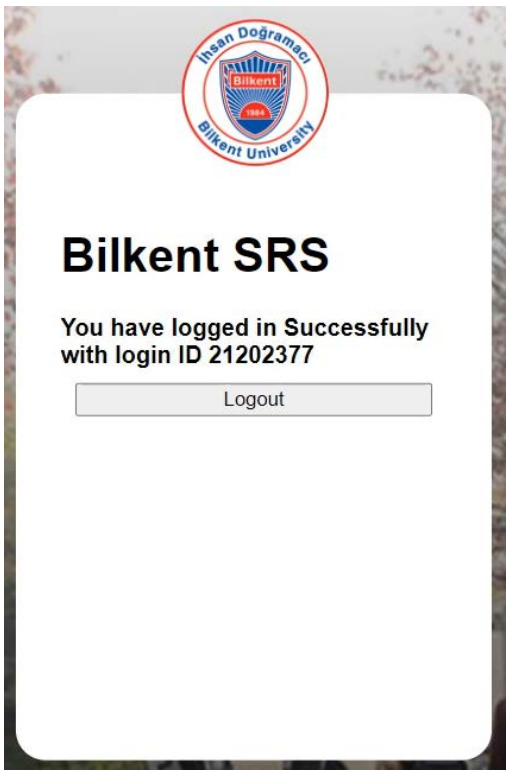




Figures 15, 16, 17:

- ID must contain only integers
- No blank input. This includes full whitespaces, or completely empty inputs.
- ID length must be of 8 characters long, no less no more

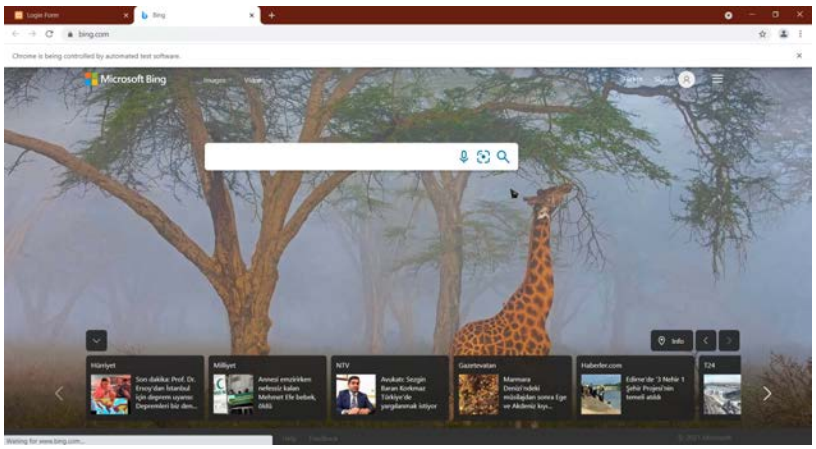


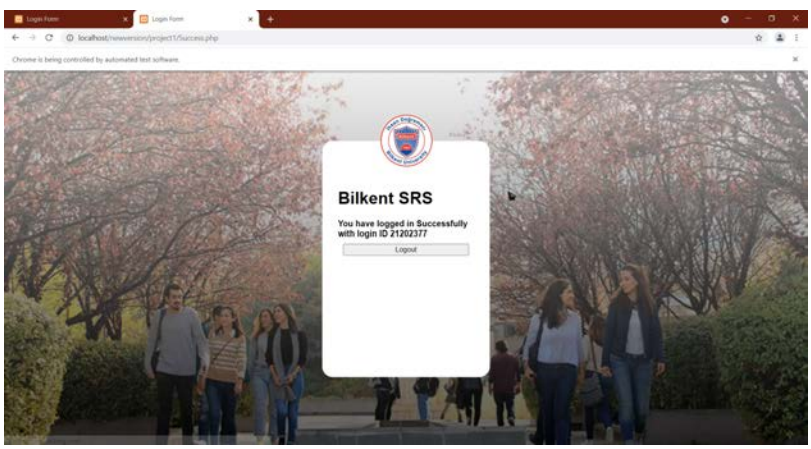


Figures 18, 19:

- A wrong id/password combination
- After a successful entry to the system the user is directed to “Success.php”.

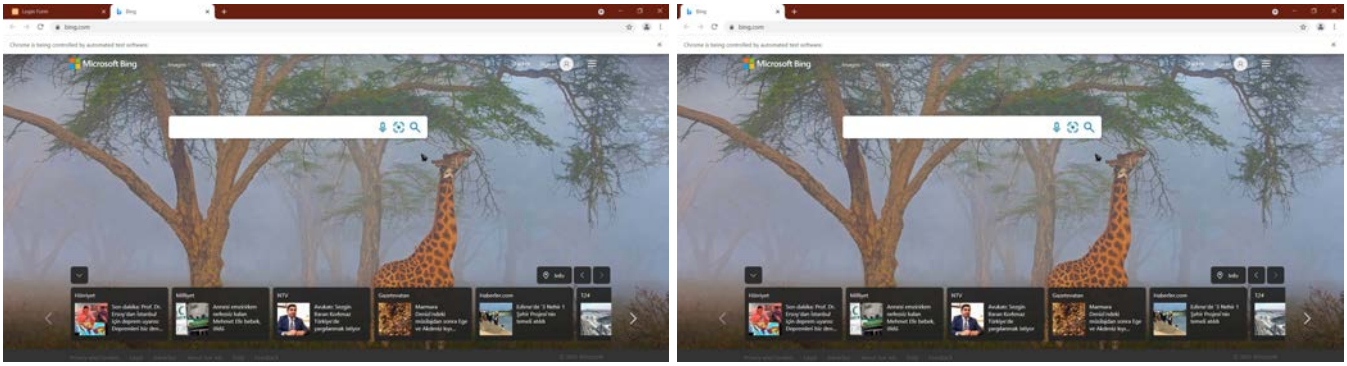
1.3.2 No Multi Sessions/Single Session Case

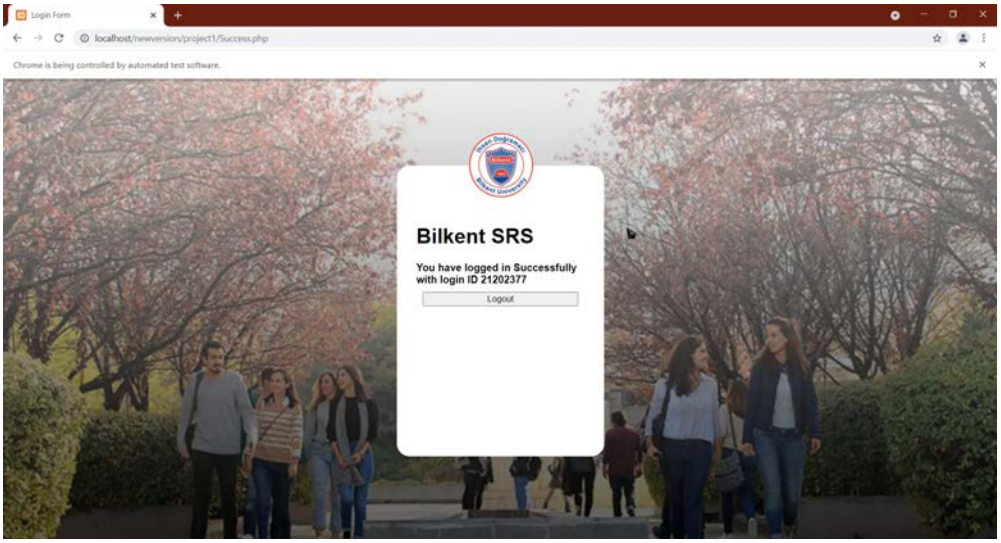




Figures 20, 21:

- A new empty tab is opened and automation goes to the login page. However, because of session systems, the page is automatically redirected to “Success.php”, because of the user’s active session.





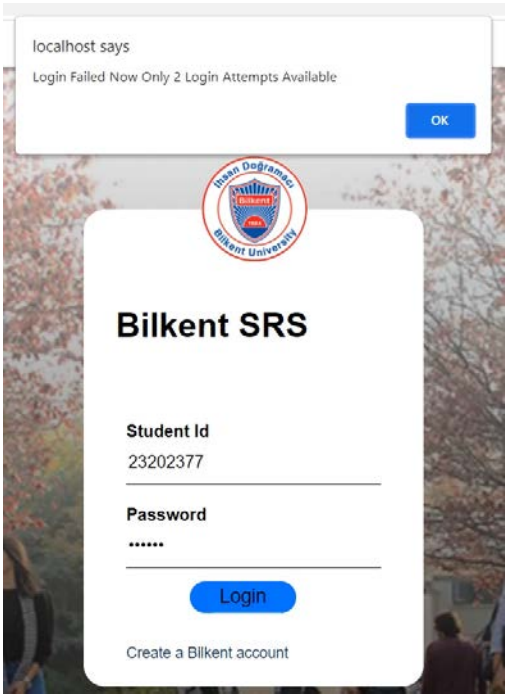
Figures 22, 23, 24:

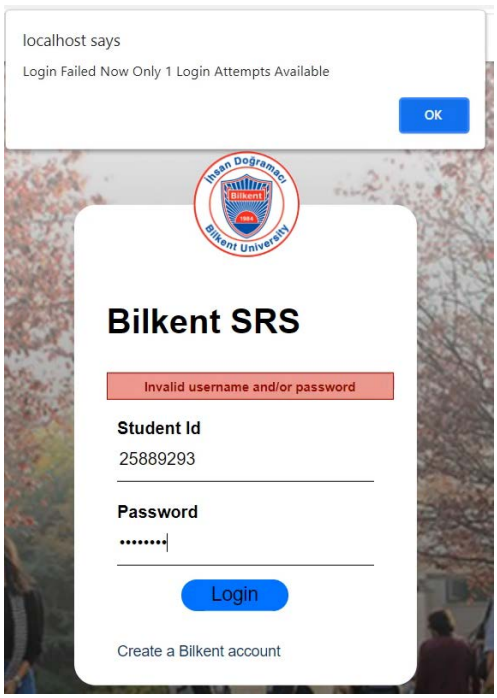
- The same feature can be seen when the user opens up a new empty tab, closes the active session without logging out, and then the user goes to “loging.php”, only to be redirected to their still active session in “Success.php”. The links entered by our code can be seen in the excerpts section of the report. The reason this is mentioned is because, the redirection is almost instant, causing the viewers not to be able to see the change of the page.

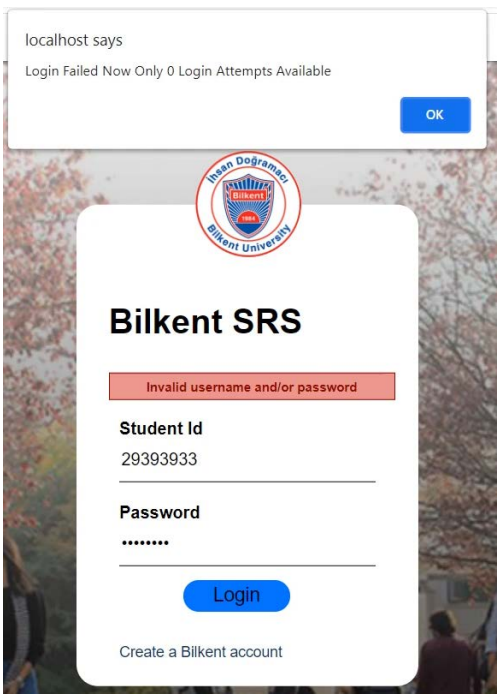
- Another case tested for this feature is, when the user has an active session and presses the back button on the browser, they do not go back to the login page. The active session has to be terminated for the user to be able to open up the login page.

  This case can be observed better in the video demo of the project available on the GitHub page.

1.3.3 Three Incorrect Login Attempts Result in Timeout







Figures 25, 26, 27:

- After attempting three incorrect logins, the system gives the user a warning, and locks any further attempts by five seconds. After those five seconds have passed, the system is unlocked again, and the user can attempt to login again.

Figures 28, 29:

1.3.4 Enter Key works as Login/Sign Up Button

- This feature allows the user to submit their entered inputs by pressing the ENTER key on their keyboards, while either Id, or password fields are active. By default, the Id field is active.

- This feature is best demonstrated by the video demo, images will not serve for the demo purposes.

1.3.5 User is Automatically Logged out After X Seconds of Inactivity

- This feature automatically logs out a user if they stay inactive for X amount of seconds. For quicker testing purposes we have set the X to be 3 seconds. By default, it can be a value from 10 minutes up to 15 minutes.

- This feature also is best demonstrated by the video demo.

1.4 Excerpts of Test Code

Important excerpts of test code can be viewed below that are integral to the functioning and understanding of the automation code using selenium.

First of all, we set up Selenium to test multiple cases.

```
# set the wait time between each action
t = 1

# Selenium setup for chrome using webdrivers
options = webdriver.ChromeOptions()

options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')
options.add_argument("--mute-audio")


driver = webdriver.Chrome('C:\\webdrivers\\chromedriver.exe', options=options)
driver.maximize_window()
```

Figure 30: Setting up Selenium in the Python code

The automation code runs on two separate text files, inside a loop. First text file includes the test entries for the registration page, and the second text file includes all of the test cases for the login page.

The first for loop traverses inside the registration test cases.txt file testing out invalid/valid inputs to the input fields. This one is simpler than the second since it is pretty straightforward, adding inputs and pressing the sign up button.

The second for loop iterates inside a more sophisticated text file. And it keeps a count of how many test cases have been done. Each test case consists of multiple different cases, so a count helps us to get to the next test case.

```
for line in lines1:

    #Split the line by white spaces to get two indexes of information
    # 0 index is email, 1 index is password
    credentials = line.split("\t")

    print("testing id: ", credentials[0],", password: ", credentials[1], credentials[2])
    email = credentials[0]
    password = credentials[1]
    password2 = credentials[2]

    email_field = driver.find_element_by_id("email-field")
    password_field = driver.find_element_by_id("password-field")
    re_password_field = driver.find_element_by_id("re-password-field")

    time.sleep(t)

    #Clear the text fields
    email_field.clear()
    password_field.clear()
    re_password_field.clear()

    # Enter info
    email_field.send_keys(email)
    password_field.send_keys(password)
    re_password_field.send_keys(password2)


    # Click sign up
    driver.find_element_by_id("registration-form-submit").click()
```

Figure 31: Registration Test Cases

```
# Switch tab
driver.switch_to.window(driver.window_handles[0])
time.sleep(t)
print("testing pressing back button at success.php page")

# Press back button. This will also result in redirection to Success.php since we have a session system.
driver.execute_script("window.history.go(-1)") # results in success.php again bcz of sessions
time.sleep(t)

# The session also stays online even when the tab is
# closed and a new tab is opened and the user went to
# login page. The user will be redirected to success.php as expected.
print("testing opening new empty tab, closing success page, then going to login page with the empty tab")

# Open new tab
driver.execute_script('''window.open("http://bings.com","_blank");''')
time.sleep(t)

# close first tab
driver.switch_to.window(driver.window_handles[0])
driver.close()
time.sleep(t)

# switch to new tab
driver.switch_to.window(driver.window_handles[0])

# go to login page
driver.get("http://localhost/project1/login.php") # this page will automatically be directed to success.php
```

Figure 32: Switching between tabs for test case 2

```
def enter_credentials(id, password):
    # Finds id and password text areas
    id_field = driver.find_element_by_id("id-field")
    pass_field = driver.find_element_by_id("password-field")

    #Clear the text fields
    id_field.clear()
    pass_field.clear()

    # Enters the given parameters into text fields
    id_field.send_keys(id)
    pass_field.send_keys(password)

    # Press submit
    #submit = driver.find_element_by_id("login-form-submit")
    #submit = driver.find_element_by_xpath("html/body/div/form/input[3]")
    submit = WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.ID, "login-form-submit")))
    submit.click()
```

Figure 33: Most used method in the automation(input login info)

```
# Show that enter works by pressing enter without filling fields
print("pressing enter with empty fields")
submit.send_keys(u'\ue007')
time.sleep(t)

#Clear the text fields
id_field.clear()
pass_field.clear()

# Enters the given parameters into text fields
id_field.send_keys(id)
pass_field.send_keys(password)
time.sleep(t)

print("pressing enter with correct entries")
submit.send_keys(u'\ue007')
time.sleep(t)
```

Figure 34: Pressing enter key for test case 4

```
elif line_count  < 16:
    # TEST CASE 5
    # After 3 seconds of inactivity the user is actually logged out automatically
    # 3 seconds here can be changed from server.php, for the sake of quick testing it is set to 3
    #
    print("TEST CASE 5")
    print("testing logout after 3 seconds of inactivity")
    print()

    # Wait for 5 seconds to be safe
    time.sleep(5)

    # Now reload the page
    print("refreshing after 5 seconds of inactivity")
    driver.refresh()
    time.sleep(t)
```

Figure 35: Inactivity logout test for case 5

2 UML Diagrams

2.1 Class Diagram

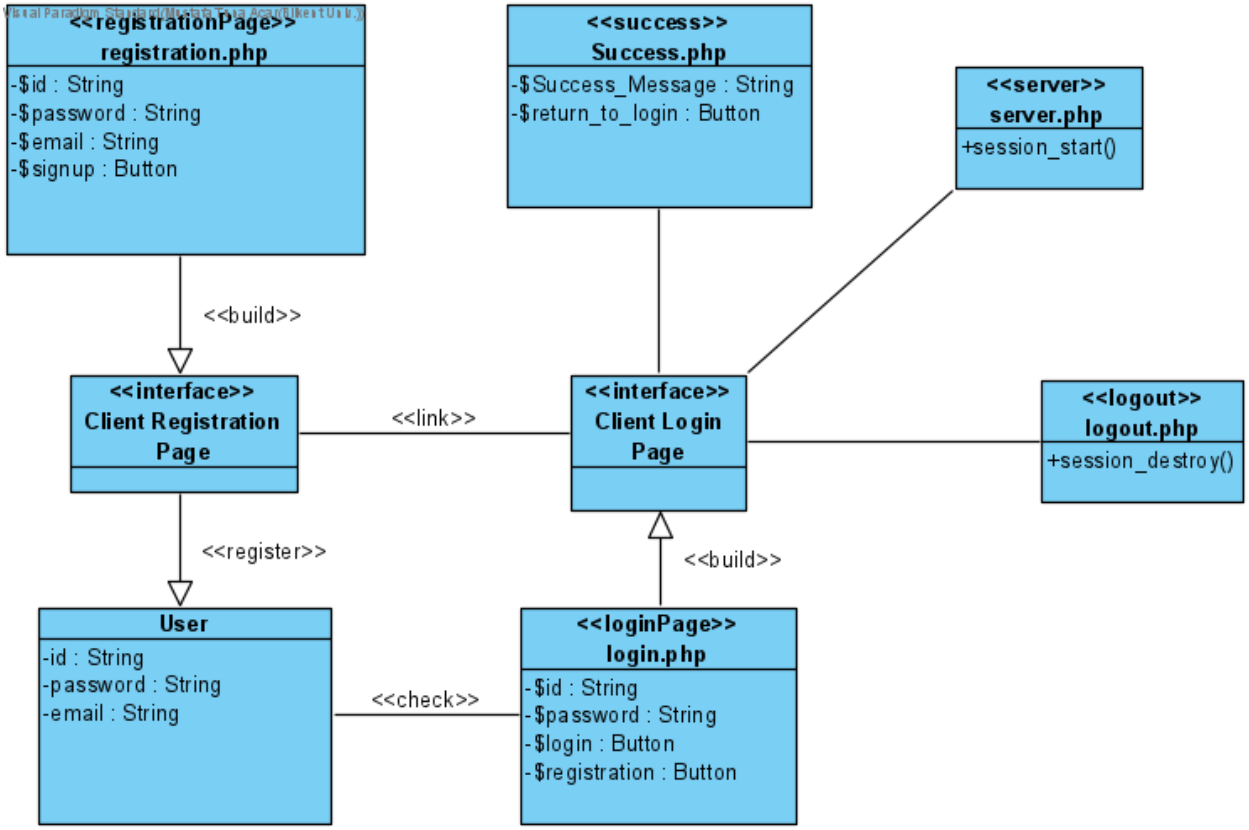


Figure 36: Class diagram representing the structure of the Website

2.2 Use Case Diagram

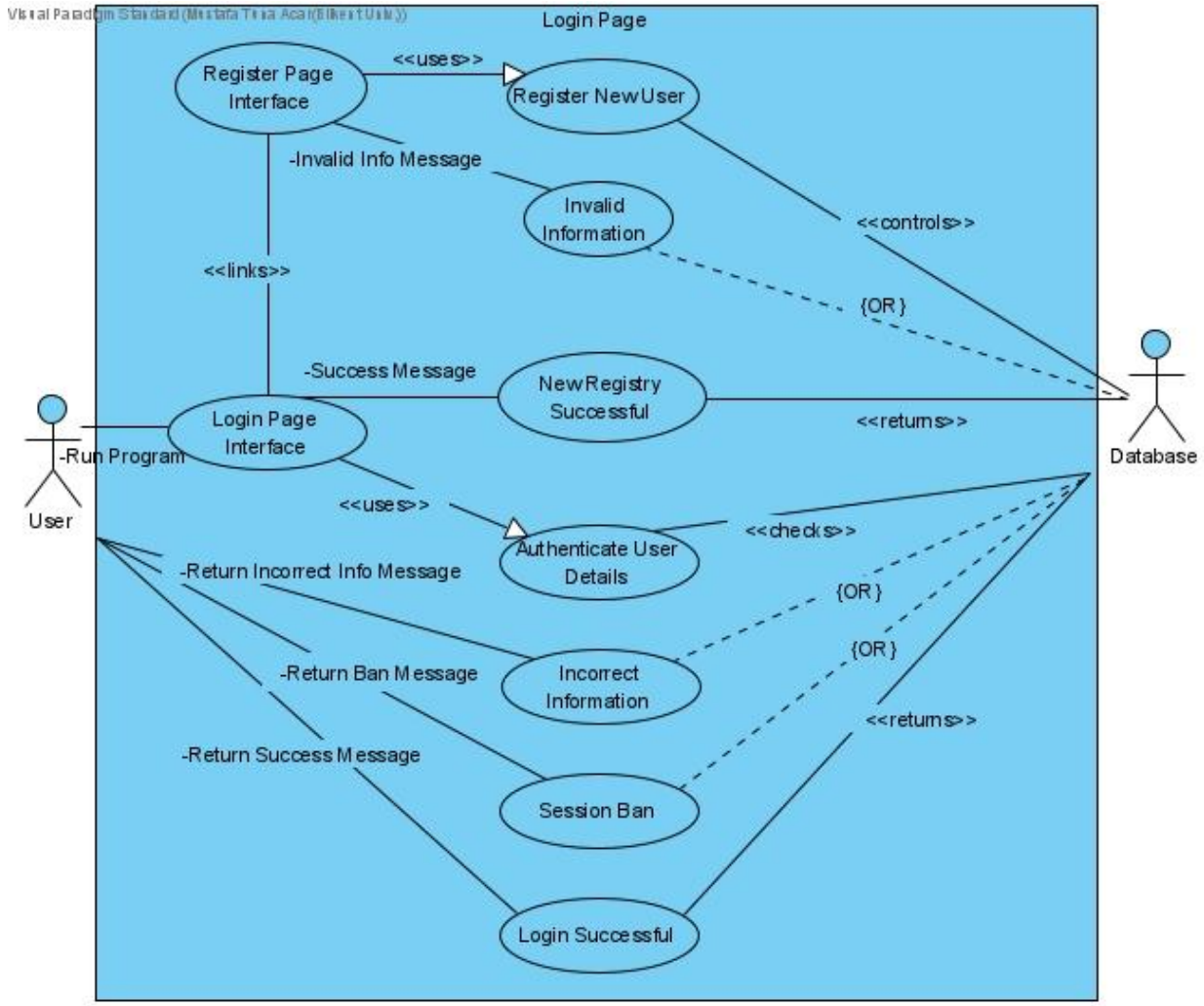


Figure 37: Use-Case Diagram for using Website

2.3 Sequence Diagram

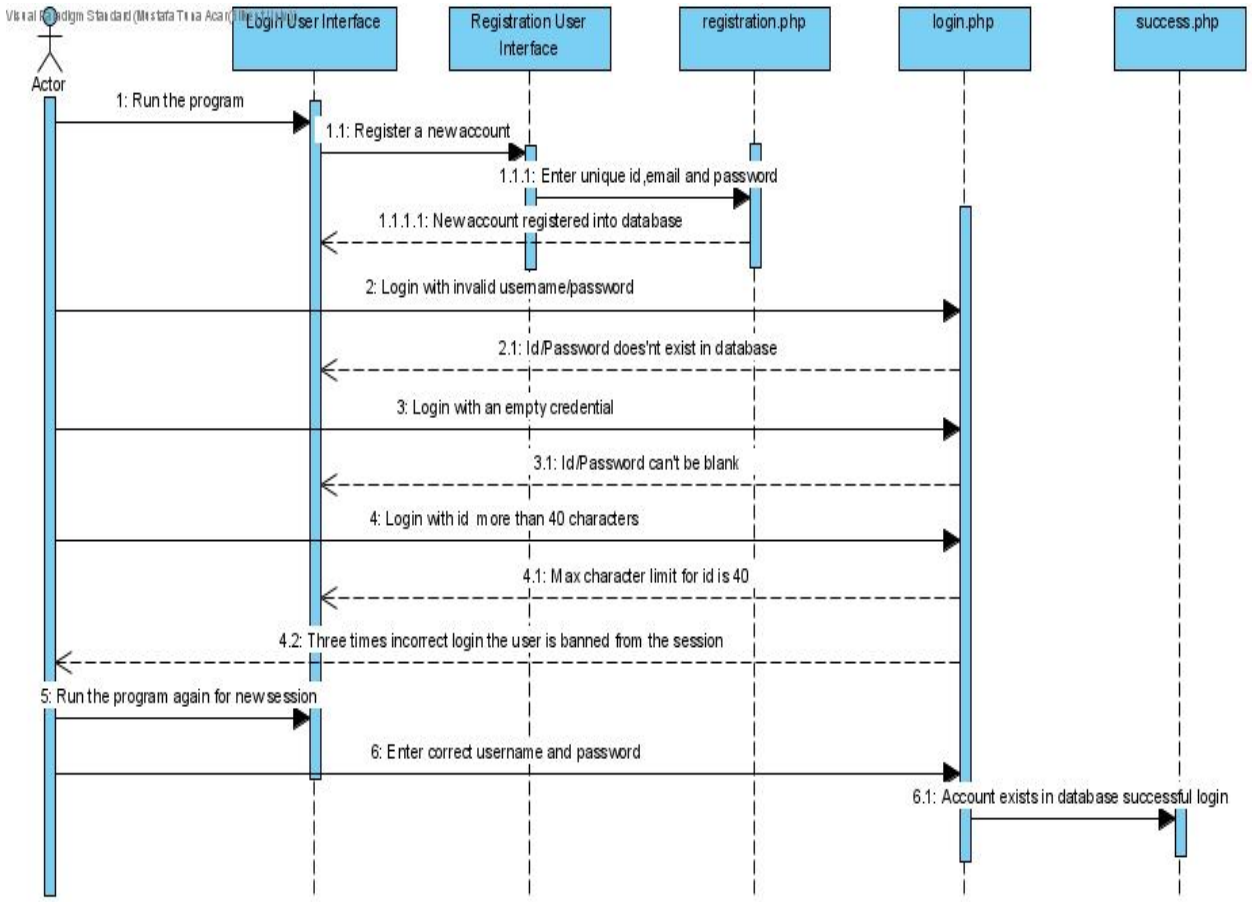


Figure 38: Representing the process of Test Case 1

3 Analysis of Selenium’s Capabilities

Selenium is a framework which enables running automated tests on several different browsers. It's popularity rises from its ability to be adapted into different multiple browsers using it's open-source features. Using WebDriver as a core of the interface, Selenium provides a user-friendly GUI for its test automation.

List of Selenium's Capabilities:

1. Reusability with Integrations:- Since Selenium is not a fully inclusive web automation testing tool Using third party frameworks and add-ons help Selenium be reusable and let it be tested with different browsers and operating systems.
2. Support Of Different Operating Systems:- Selenium is advantageous for it's portability for being used in various operating systems like Windows, Mac OS, Linux, Unix.
3. Open Source:- Allowing every engineer to freely use and develop Selenium projects allows the community and creativity to grow rapidly.
4. Language Support:- Selenium projects can be written using Java, Python ,C#, Ruby, JavaScript and Kotlin languages. Allowing Selenium to be written in these popular programming languages increases the popularity and the readability of Selenium projects.
5. Regular Updates:- Since Selenium is growing more popular with it's open source and other features, the growing community allows Selenium to get updated constantly and improve
6. Multiple Browser Support:- Using different browsers is supported because each browser has a popular user base and Selenium allows it's script to be compatible with different browsers such as Chrome, Firefox, Safari, Internet Explorer, Opera, and Edge.
7. Less Required Hardware Usage:- When compared with the other test automation tools such as QTP,SilkTest and Uft, Selenium proves to require less hardware usage.
8. Flexibility:- Selenium is flexible when it comes to regrouping and refactoring of the test cases which allows quickly changing the code while reducing complexity and duplication.

9. User Friendly Implementation:- The interface and framework are made user friendly and let the user create and run test scripts while letting them watch it.
10. Parallel Testing:- Using Selenium Grid, Selenium lets the execution of multiple tests in parallel which reduces the time to execute the program.

4 Automation Experience

4.1 Automation Experience

During the past two weeks, we have had the opportunity to work on many coding languages. In order to get the project working, we first implemented simple HTML pages, stylised with CSS files. Then used Javascript to make them more functional. We started automating using Python. However, we were not satisfied with the simple webpage we had, so we implemented a database using Xampp, and changed the whole structure of the webpages, transferring them to php's that include HTML, JS, and PHP code in one file for each page. It was a great experience to work on this many languages in such short notice. It proved to be a challenge.

Another challenge was to implement all of the test cases in Python using Selenium. More complicated cases would require manual testing, although we wanted to make it fully automated, without any tester having to interfere with the process, and it turned out well. We had to learn to plan ahead, and consider many perspectives in order to consider a wide variety of test cases.

There were critical timing issues and we had to deal with those issues since some elements of the webpage took a longer time to reload and there had to be a pause between some activities for the testing process to function properly therefore a timer was set at appropriate intervals.

4.2 How Test Automation contributes to SDLC

Velocity

It may be disadvantageous because if the automation is done for the software's short-term process, creating an automation code with the same amount as the software itself is insufficient and sustainable. Because the overall process for creating the automation system is far longer than doing it manually. However in the long term as the development grows, automation comes in handy as it creates time-efficiency rather than doing it manually thus it becomes an efficient tool for the developer.

Quality

Compared to testing manually, the automated testing systems help the process to be less risky. As the constant well state of the software is compact consistently, this reduces the overall risk thus it improves the quality of the software. Also, the effort to run such a test is reduced and minimized therefore the quality of the software is ensured.