Name: Layla Pezeshkmehr
Project: IoT Security Automation Testing
July 2019

IoT Security Game:
This game was designed for students who have taken cyber security course, and by taking this game, they would evaluate their performance on how well they have learned the concept.

In this game, an instructor will start and control the game. After students have logged in, the instructor can either manually create teams of students or having the system automatically classify students into several teams without the instructor's intervention.

By choosing to manually create each team, the instructor will do as follows:

Create a team name; e.g. team1 and drag the selected students one at a time into the specified team and move to the creation of the next team until the instructor is done with creating the teams with the team members.

The instructor will start the game by clicking on the start button, then the game will appear on the screen of each students. Each question has two portions, they would use Kali Linux and a raspberry Pi to complete the first section. By typing the answer, on the text field, each team member will move to the second section of the question which is multiple choice. After selecting the answers, students will move to the next question. There are total of 10 questions.

As discussed, each question consists of two sections: the mission (hands on project) and the quiz section. Each team member could do each section individually, but the team member who completes the question first will take the credit and move the team to the next question. As the team members start the mission their name will be added and will be shown in the list of the students playing the mission or the quiz. If for any reason a student decides to abandon the question (either the mission or the quiz), her or his name will be removed from the list of the players playing the mission or the quiz.

After completing the test, the instructor's screen will show the score of each team and the credit of each team member.

Automation Testing:

Testing tool: Selenium Web driver
Scripting language: Python 3

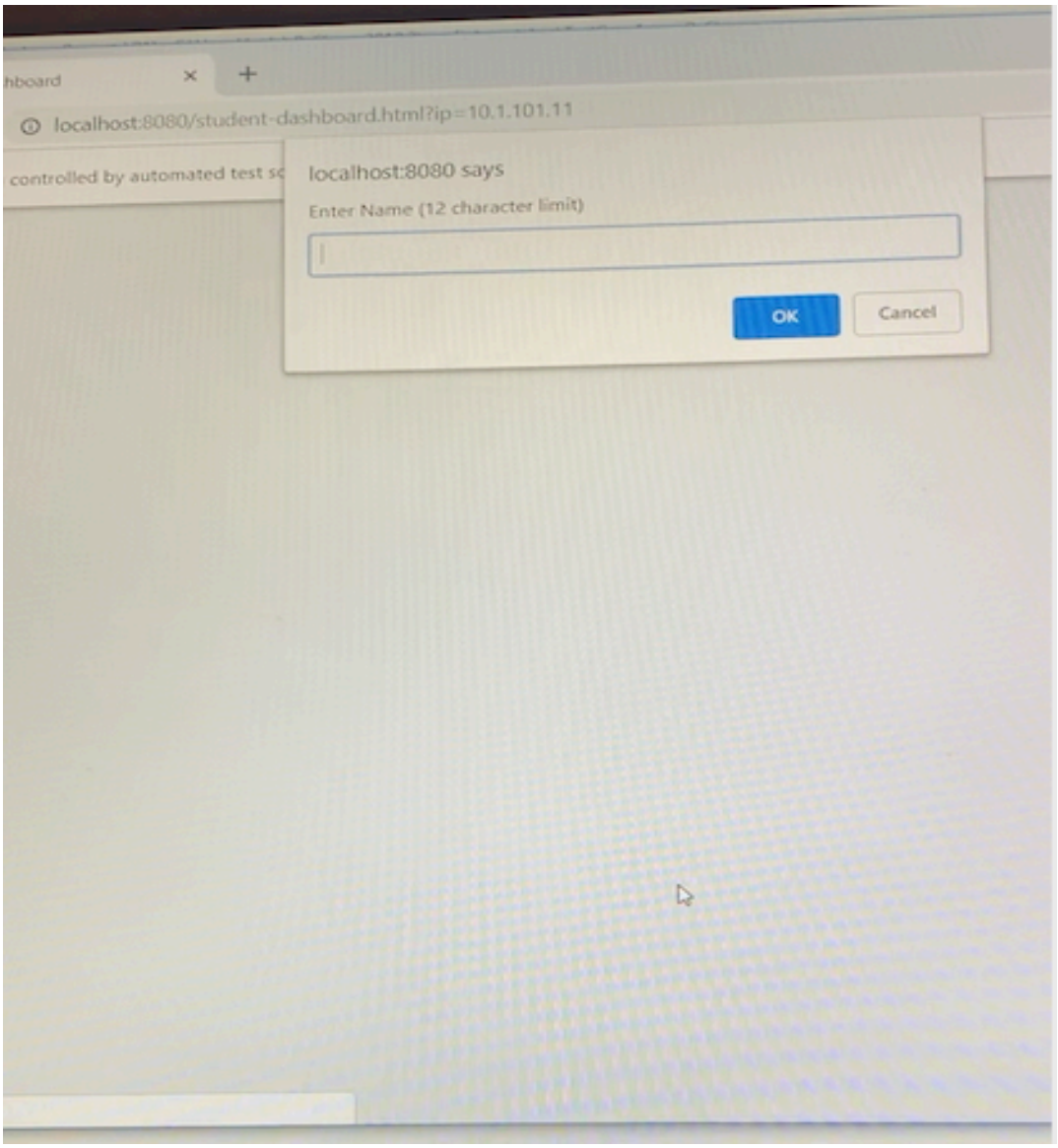
The testing was automated using selenium web driver. One web browser was opened for the Instructor and one browser per each student. (n browsers open for n students). All the features of instructor's web page and student's web page were tested considering the web elements

and using locators to uniquely identify the web elements and assert statements are used to catch all the bugs and to examine the flow of the game.

instructor_actions.py defines all the web elements and uses locators to identify each and every web element uniquely within the instructor's webpage.

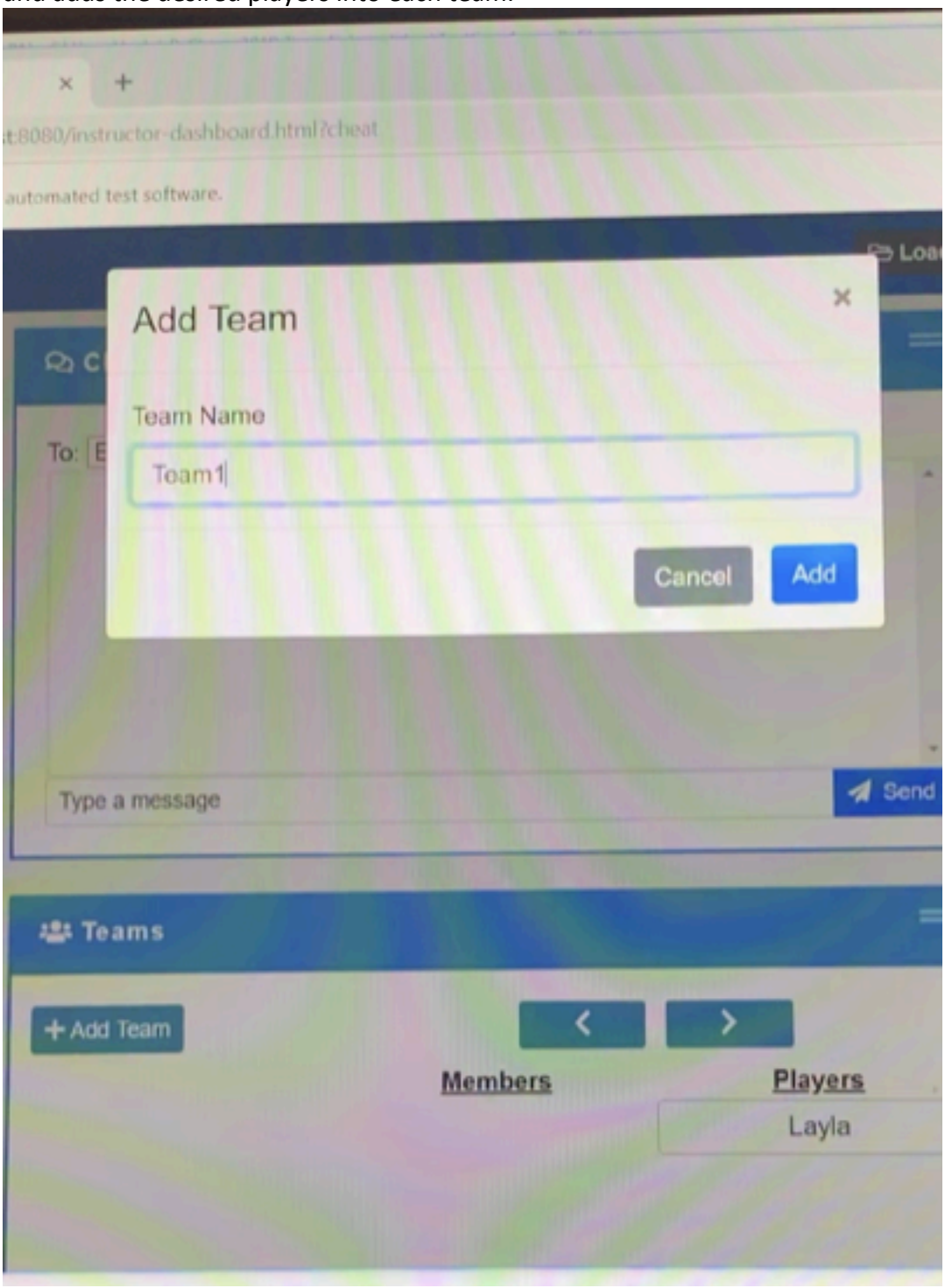
Student_actions.py defines all the web elements and uses locators to identify each and every web element uniquely within the student's webpage.

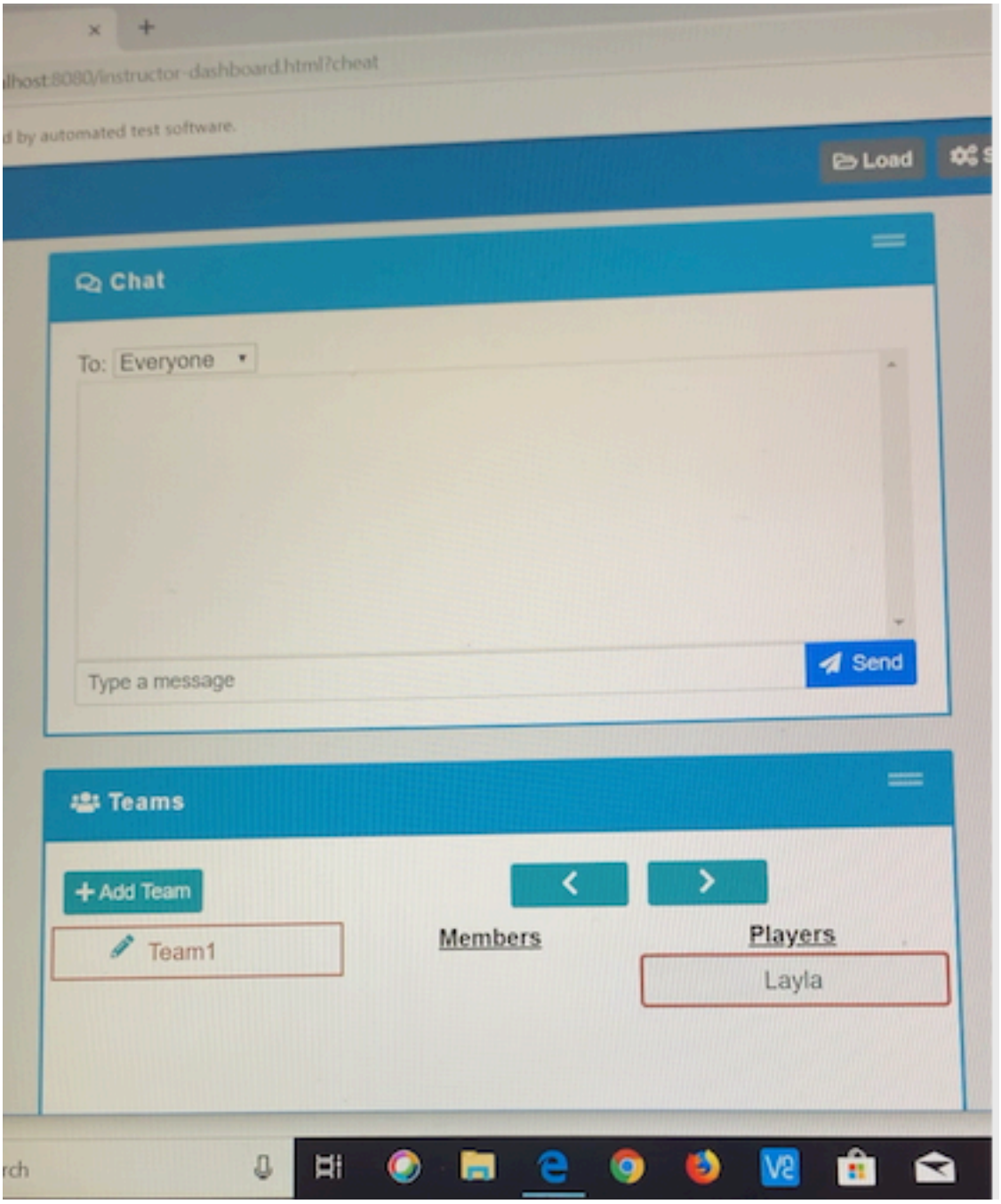
Automation test of student login:
After launching a student web browser, a panel will prompt for student login. The student will enter his/her name. The Ip address for the students starts from 10.1.101.11 the next will be 10. 1. 102.11 and …



After each student logs in, the instructor will create the teams manually or automatically.

By choosing manual button, the instructor creates the team by entering the name of the team and adds the desired players into each team.





After the teams are created and all the team members are assigned to the teams, the instructor starts the game by clicking the start button on the instructor's dashboard. Starting the game will cause the game to be shown in each student browser where they start answering the question and after completion they move to the next question until they finish the game. As mentioned above each question consists of two section. The first question requires student to use Kali Linux and Raspberry Pi to complete the task and entering the final answer in the

desired text field and click the submit button to move to the second part of the question which is quiz with multiple answers.

Description of each test case:

TST_Student_01: test_student_login.py

Preconditions:

1. We have set the number of Teams to 2
   Each team has two members
   Total of 4 web browsers are opened and 4 students to login.
   The first web browser opened has IP address of 10.1.101.11
   The next IP will be calculated by adding the IP address to 256 which will result in 10.1.102.11, …

def test_student_login(self):

This will test the student login. We have an array containing 20 names for the students. I used a loop which will grab a name from the list of the names and perform log in by entering the name in the text field and hit enter. Assert statement is used to make sure that the expected result is acquired. In this case the navbar should show the right student name.

def test_submit_with_Enter_key(self):

This function also performs the same thing however instead of hitting the enter it clicks on the submit button.

def tearDown(self):

close all the student browsers.

TST_Instructor_02: test_manual_team_setup.py

Preconditions:

1. Students have logged in meaning all the student's browsers were opened allowing students to log in. Again, we assume we will have two teams and each team will have two team members. Total of 4 student browsers are opened and students have logged in successfully.

The Instructors will click on the manual button and manually creates teams and assign each team with 2 team members.

Again, at each step, assert statement are being used to verify the result is what exactly we were expecting.

Postcondition:

At last, at the tear down, we close all the browsers (students and the instructor web browser)

TST_Insturctor_03: test_auto_diff_create_team

Preconditions:

1. Students have logged in meaning all the student's browsers were opened allowing students to log in. Again, we assume we will have two teams and each team will have two team members. Total of 4 student browsers are opened and students have logged in successfully.

This test case will click on the automatic button to create teams and assign each team member to a team. Assert statements are used to verify two teams are created and the team members have been assigned correctly to the teams.

TST_Instructor_04: test_auto_team_setup.py

Preconditions:

1. Students have logged in meaning all the student's browsers were opened allowing students to log in. Again, we assume we will have two team and each team will have two team members. Total of 4 student browsers are opened and students have logged in successfully.

def test_auto_team_create(self):
locates the auto button web element using the locator and perform a click action. The rest is using assert statement to verify that every single team member has been assigned to each group having two teams. The assert statement also checks if two teams have been successfully created.

Post condition:

Finally, tear down, close all the browsers including Instructor's browsers and all the students' browsers.

TST_Instructor_05: test_start_game.py
Preconditions:

1. Students have logged in meaning all the student's browsers were opened allowing students to log in. Again, we assume we will have two team and each team will have two team members. Total of 4 student browsers are opened and students have logged in successfully.

2. Teams have been created and team members are all assigned into teams.

def test_start_timer_modal(self):

Automation will locate the web element (start game button) and performs action click. By clicking on the start game button, two web elements will appear showing the hour and minutes left to the start of the game. Depending if the game needs to be started right away or with some delay, hours and minutes will be set with the automation. After the time expires, the game will start. If we want to start the game immediately hour and minute values are set to zero. After setting hour and minute the remaining time to start of the game will be shown using countdown (hours/ minutes left to the beginning of the game).

And Finally, automation perform a click on the game reset button on the instructor's dashboard.

Post condition:

Tear down, will close all the browsers for both instructor and the students.

TST_Instructor_06: test_pause_game.py

Preconditions:

1. Students have logged in meaning all the student's browsers were opened allowing students to log in. Again, we assume we will have two teams and each team will have two team members. Total of 4 student browsers are opened and students have logged in successfully.

2. Teams have been created and team members are all assigned to a right team.

3. The game has been started by the Instructor

def test_pause_modal(self):

If Instructor decides to pause the game, this test case implements and automates the condition by locating the Pause button web element and performing click action on it. Assert statement is used to verify the correct button has been clicked. By clicking the Pause button, the instructor can set the timer (setting the hour and minutes). After the time expires, the game resume and students will be able to continue playing the game.

This test case cancels the pause and just resumes the game and finally click on the reset game button.

Tear down: Closes all the browsers including Instructor's and the student's browsers.

TST_Student_07: test_multiple_users_same_mission.py

This test case automates the condition where first team member in a team starts doing a mission. Assert statement checks to make sure the name of the first player is added to the list of players playing this mission. Next, the second teammate will select the same mission to answer the question. Assert statement checks both players and verifies if both names are in the list of the players playing the same mission. The first player finishes the mission and move to the quiz section which results in getting the whole credit and take the team to next question.

TST_Student_08: test_abondon_mission.py

This test case automates a scenario where first team member selects a mission. Next, the second team member selects the same mission. Assert statement verifies both team members are playing the same mission. The first team member decides to leave/abandon the mission by clicking on the abandon button. Assert statement verifies and checks if the name of the first team member is removed from the list of the players for that specific team. The second team member finishes the mission and complete the next section (the quiz) and completes the question.

TST_Student_09: test_abondon_mission_after_get_mission.py

This test case automates a condition where first team member select the mission next the second teammate select the mission and finishes the mission first. The first team member decides to abandon the mission before the second team member completes the first part. Assert statement check if all the conditions have been met.