



Summer Fellowship Report

On

YAKSH: Testing Automation

Submitted by

Vivek Kumar

Under the guidance of

Prof. Prabhu Ramachandran

Department of Aerospace Engineering

IIT Bombay

July 8, 2019

Acknowledgment

I, Vivek Kumar, a FOSSEE intern of the YAKSH Project is overwhelmed in all humbleness and gratefulness to acknowledge my deep gratitude to all those who have helped me accomplish the tasks assigned to me.

I am highly indebted to my project mentor Mr. Ankit R. Javalkar for his continuous support, supervision, motivation and guidance throughout the tenure of my project in spite of his hectic schedule who truly remained driving spirit in my project and his experience gave me the light in handling this project and helped me in clarifying the abstract concepts, requiring knowledge and perception, handling critical situations and in understanding the objective of my work.

With Regards,

Vivek Kumar

(GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI)

Contents

1	Introduction	3
2	Moderator and Student mode on Yaksh	4
2.1	Moderator Mode	4
2.2	Student Mode	6
3	About Selenium	7
3.1	Selenium 1 (aka. Selenium RC or Remote Control)	7
3.2	Selenium 2 (aka. Selenium WebDriver)	7
4	Flow of Testing	8
4.1	Flow of moderator functions testing	8
4.2	Flow of student functions testing	9
5	Using Selenium WebDriver for browser automation	10

Chapter 1

Introduction

Yaksh is an Online Test Interface for Conducting online programming quiz. It supports various programming languages like :- C, C++, Python and simple Bash.

User can solve any questions by using these languages. Yaksh uses test cases to test the the implementations of the students. It also supports simple multiple choice questions and file uploads so that user can easily submit his code.

Not only you can practice the questions even you can also conduct a programming quiz that supports various languages. There is a separate moderator section for this where a moderator can create questions,quizzes and courses. Yaksh provides various programming courses created by moderators to be enrolled by the students.

Chapter 2

Moderator and Student mode on Yaksh

2.1 Moderator Mode

On logging in moderators see the following dashboard.

YAKSH Questions Courses Monitor Grade User Regrade Vivek Kumar

Moderator's Dashboard

List of quizzes! Click on the given links to have a look at answer papers for a quiz.

Courses	Quizzes								
demo_course	<table border="1"><thead><tr><th>Quiz</th><th>Taken By</th><th>No. of users Passed</th><th>No. of users Failed</th></tr></thead><tbody><tr><td>demo_quiz</td><td>1 user(s)</td><td>0</td><td>1</td></tr></tbody></table>	Quiz	Taken By	No. of users Passed	No. of users Failed	demo_quiz	1 user(s)	0	1
Quiz	Taken By	No. of users Passed	No. of users Failed						
demo_quiz	1 user(s)	0	1						
Yaksh Demo course	<table border="1"><thead><tr><th>Quiz</th><th>Taken By</th><th>No. of users Passed</th><th>No. of users Failed</th></tr></thead><tbody><tr><td>Yaksh Demo quiz</td><td>1 user(s)</td><td>1</td><td>0</td></tr></tbody></table>	Quiz	Taken By	No. of users Passed	No. of users Failed	Yaksh Demo quiz	1 user(s)	1	0
Quiz	Taken By	No. of users Passed	No. of users Failed						
Yaksh Demo quiz	1 user(s)	1	0						

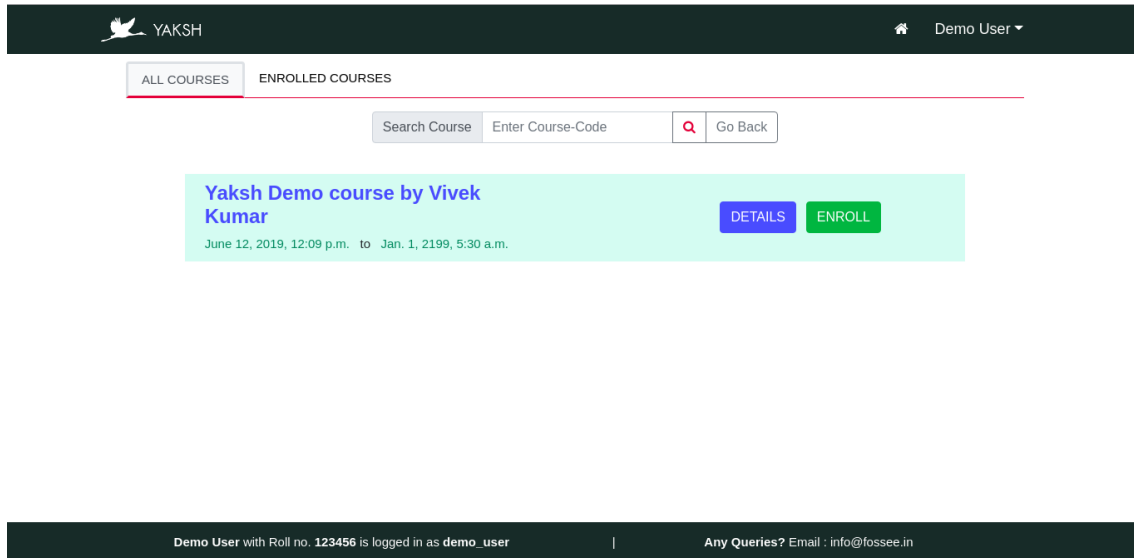
[Add New Course](#) [Create Demo Course](#) [What's This](#)

Following functions are available for moderator:

- Courses
 - Setting up a new course
 - Design a Course
- Quizzes
 - Creating a Quiz
 - Creating a Exercise
 - Designing Question Paper
 - Editing a Quiz/Exercise
 - Editing a QuestionPaper
- Questions
 - Setting up questions
- Lessons and Modules
 - Setting up a Lesson
 - Setting up a Module
 - Design a Module

2.2 Student Mode

On logging in students see the following dashboard.



The screenshot displays the YAKSH student dashboard. At the top, there is a dark header with the YAKSH logo on the left and 'Demo User' on the right. Below the header, there are two tabs: 'ALL COURSES' and 'ENROLLED COURSES'. A search bar is present with the text 'Search Course' and 'Enter Course-Code', followed by a search icon and a 'Go Back' button. The main content area features a light blue card for a course titled 'Yaksh Demo course by Vivek Kumar'. Below the title, it shows the course duration: 'June 12, 2019, 12:09 p.m. to Jan. 1, 2199, 5:30 a.m.'. To the right of the card are two buttons: 'DETAILS' (purple) and 'ENROLL' (green). At the bottom, a dark footer contains the text 'Demo User with Roll no. 123456 is logged in as demo_user' and 'Any Queries? Email : info@fossee.in'.

Following functions are available for students:

- Enrolling in a quiz/course
- Taking a quiz

Chapter 3

About Selenium

Selenium is a set of different software tools each with a different approach to supporting test automation. Most Selenium QA Engineers focus on the one or two tools that most meet the needs of their project, however learning all the tools will give you many different options for approaching different test automation problems. The entire suite of tools results in a rich set of testing functions specifically geared to the needs of testing of web applications of all types. These operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behavior. One of Seleniums key features is the support for executing ones tests on multiple browser platforms.

Selenium is composed of multiple software tools. Each has a specific role.

3.1 Selenium 1 (aka. Selenium RC or Remote Control)

Selenium RC was the main Selenium project for a long time, before the WebDriver/Selenium merge brought up Selenium 2, the newest and more powerful tool.

Now Selenium 1 is deprecated and is not actively supported (mostly in maintenance mode).

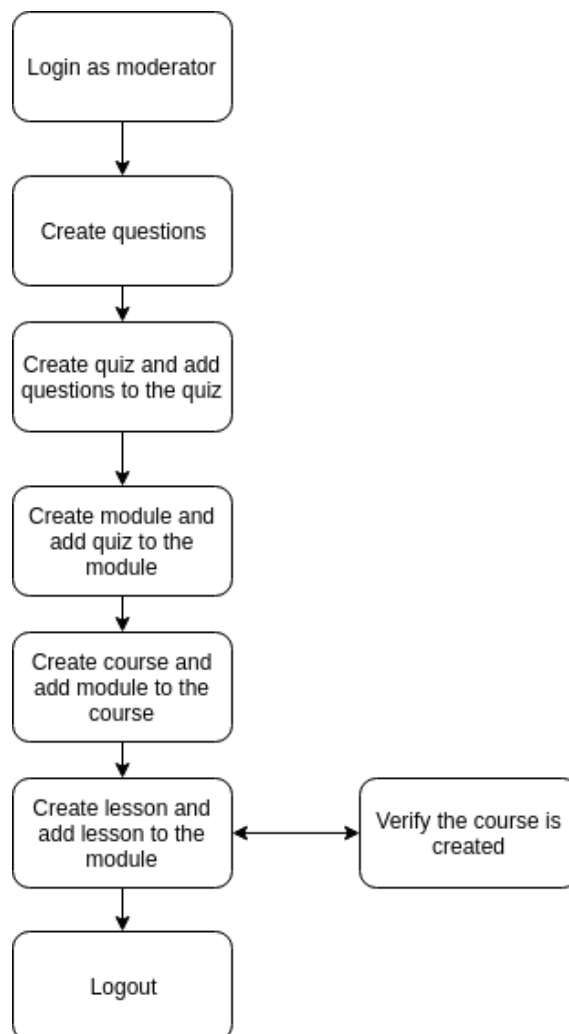
3.2 Selenium 2 (aka. Selenium WebDriver)

Selenium 2 is the future direction of the project and the newest addition to the Selenium toolkit. This brand new automation tool provides all sorts of awesome features, including a more cohesive and object oriented API as well as an answer to the limitations of the old implementation. I have used selenium 2 in my task for automation the testing of platform Yaksh. We will look in more detail about selenium 2 in upcoming chapters.

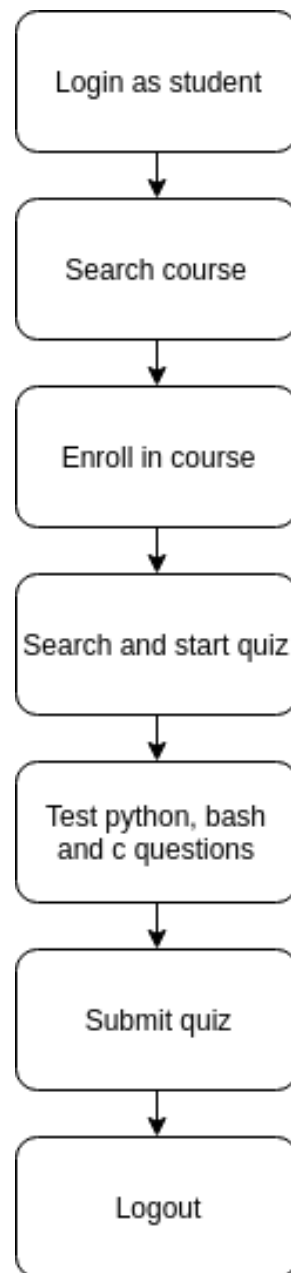
Chapter 4

Flow of Testing

4.1 Flow of moderator functions testing



4.2 Flow of student functions testing



Chapter 5

Using Selenium WebDriver for browser automation

I have used Selenium 2 (aka. Selenium WebDriver) for automating the testing of the Yaksh as the flow mentioned in section 4.1 and 4.2.

The selenium.webdriver module provides all the WebDriver implementations. Currently supported WebDriver implementations are Firefox, Chrome, IE and Remote.

```
1 from selenium import webdriver
```

Next, we have to create an instance of webdriver. I have used instance of Firefox WebDriver in my code.

```
1 driver = webdriver.Firefox()
```

The driver.get method will navigate to a page given by the URL. WebDriver will wait until the page has fully loaded (that is, the on-load event has fired) before returning control to your test or script. Its worth noting that if your page uses a lot of AJAX on load then WebDriver may not know when it has completely loaded.

```
1 driver.get('/exam/login/')
```

For automating a task with selenium on a web page for example clicking a button we have find that button on the web page and then click on it using selenium webdriver methods.

There are various strategies to locate elements in a page. You can use the most appropriate one for your case. Selenium provides the following methods to locate elements in a page:

- `find_element_by_id`
- `find_element_by_name`
- `find_element_by_xpath`
- `find_element_by_link_text`
- `find_element_by_partial_link_text`
- `find_element_by_tag_name`
- `find_element_by_class_name`
- `find_element_by_css_selector`

For example here's a code snippet for automating logging in to Yaksh:

```
1 def login(self, username, password):
2     # get the username, password and submit form elements
3     username_elem = self.driver.find_element_by_id("id_username")
4     password_elem = self.driver.find_element_by_id("id_password")
5     submit_login_elem = self.driver.find_element_by_css_selector(
6         'button.btn')
7
8     # Type in the username, password and submit form
9     username_elem.send_keys(username)
10    password_elem.send_keys(password)
11    submit_login_elem.click()
```

Sometimes we can get element not found error in spite of element being present on the page. Most of the time this type of error happens because of slow loading of page. We can overcome this issue by using WebDriverWait method, with the help of this function we can make selenium driver to wait until the presence of required element on a web page. Here's an example of its usage:

```
1 WebDriverWait(self.driver, 5).until(
2     EC.presence_of_element_located((By.ID, "home")))
3     ).click()
```

Reference

- <https://www.seleniumhq.org/docs/>
- <https://selenium-python.readthedocs.io/>