Test Plan v1.0

Unravel

November 20, 2017

0.1 Introduction

0.1.1 Project Description

Every student who enters in a domain which is completely new to him/her, he faces a problem of finding relevant study materials. He/she can easily find study materials, but it is very difficult to separate relevant study materials from it. Also, additional effort and time is required to do so, which can be applied in some productive work, if one gets relevant materials in the first place.

0.1.2 Objective of Testing

In this project, primary objective of Software Testing is to verify that our Web and Android application is running smoothly and to ensure that adding new feature doesn't breaks the existing code.

The final project of after testing phase will be a software ready to be deployed on production server and a set of stable test scripts which can be reused for testing, if some new feature is being added to website.

0.2 Features to be tested

- Website.
  - Template loading.
  - Data passage between client and server.
  - URL existence at desired location.
  - Pagination[1].
  - User registration and authentication.
  - Search feature.
  - Code coverage.
- Server
  - Rate limit hitting of YouTube/GitHub API
  - CSRF token validity.

[1] Assuming that 10 items are allowed on a given page

0.3 Features not to be tested

- DTL[2] to HTML conversion validation.
- Youtube/Github API endpoints response(s).
- Django backend in-built features (except CSRF validation).
- Database connections.
- Other standard libraries used.

0.4 Approach

Testing is done using a dummy web browser which can be simulated using *django.test.Client*. Documentation of this class can be found in references section.

0.4.1 Website

- **Template Loading**: Primary objective of testing this feature is to ensure that correct template is being loaded for a given URL. It can be ensured by makes a *GET* requests at given URL, and then verifying that the template name of loaded HTML page is same as template name of desired HTML page.
- **Data passing between client and server**: Primary objective of testing this feature is to ensure that the context which is sent by server and the context required by templates is the same, confirming that template is getting all the data which it needs to render a web page correctly. It can be ensured by making a *GET* request from an instance of *django.test.Client* to a given URL and then extracting context from the response.
- **URL existence at desired location**: Primary objective of testing this feature is to verify that desired URL and URL embedded in source code are same. A method to perform this testing is by making a *GET/POST* request at desired URL, getting a response and checking that status code of response is 200.

[2]Django Template Language

- **Pagination**: Primary objective of testing this feature is to verify that items are paginated correctly. This requires two underlying tests.

  1. Context contains some boolean variable which controls paging.
  2. If boolean variable is set to "True", then total number of items on page is exactly 10. If it is "False", then total number of items on a page is less than or equal to 10.

  This can be ensured by making a *GET* request, getting the response and then verifying that there exist a boolean variable which controls paging and length of *item_list* is 10 if the variable value is "True" and less than or equal to 10, if it's value is "False".

- **User registration and Authentication**: Primary objective of testing this feature is to verify that user registration and user authentication is working correctly. User registration can be tested by creating a user from dummy client, saving it to database and then retrieving the user from database.

  For testing user authentication, we can make a *POST* request at */login*, with context consisting of *username* and *password* and then verify that the redirect URL and profile page URL[3] is same.

- **Search feature**: Primary objective of testing this feature is ensure that behaviour of web application remains stable and appropriate response is returned on entering some specific types of search queries, like blank search query and search queries which have zero search results. This can be tested by making a *GET* request, containing search query verb, retrieving the response and verifying response is contains no search result in both cases. However, when search query is blank, then user remains at the same web page.

- **Code coverage**: In computer science, code coverage is a measure used to describe the degree to which the source code of a program is executed when a particular test suite runs. A program with high code coverage, measured as a percentage, has had more of its source code executed during testing which suggests it has a lower chance of containing undetected software bugs compared to a program with low code coverage. Many different metrics can be used to calculate code coverage; some of the most basic are the percentage of program subroutines and the

[3]Immediately after login, user will be redirected to it's profile page

percentage of program statements called during execution of the test suite.[4]

0.4.2 Server

- **Rate Limit hitting of Youtube and GitHub API**: Primary objective of testing this feature is to ensure that rate limit of Youtube/GitHub API hasn't reached. This is most important test as it controls *search* and *trending* feature.

  This can be done by making API call to Youtube/GitHub server and check the number of API calls left. If total number of calls left is less than 10, then replace corresponding API key with another fresh API key[5].

- **CSRF tokens validity**: The CSRF token provides easy-to-use protection against Cross Site Request Forgeries. This type of attack occurs when a malicious website contains a link, a form button or some JavaScript that is intended to perform some action on your website, using the credentials of a logged-in user who visits the malicious site in their browser.

  Due to this reason, it is important to ensure that, on *POST* request, browser is sending valid CSRF token or not. This can be tested by setting *enforce_csrf_checks* argument to True, while creating the dummy client instance.

0.5 Testing environment

0.5.1 Web

Software environment

- Hosted on: Localhost(For development), Amazon Web Service(For production).
- OS: CentOS 7, Ubuntu 16.04.
- Django Version: 1.11.7
- Templating Engine: Django Template Language (DTL)

[4]Taken from Wikipedia.
[5]hard-coded in source code file or OS environment.

- Interpreter: python 2.7.9
- Web browsers: Mozilla Firefox 52.4.0, Google Chrome 62.0.3202.94

Other library dependencies are mentioned in **requirements.txt** file.

Hardware environment

- Primary Memory: 4GB(Development), 1GB(Production)

0.6 References

1. `https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Testing`
2. `https://docs.djangoproject.com/en/1.10/intro/tutorial05/`
3. `https://docs.djangoproject.com/en/1.11/topics/testing/tools/`
4. `http://coverage.readthedocs.io/en/coverage-4.4.1/`
5. `https://docs.djangoproject.com/en/1.10/topics/testing/advanced/`
6. `https://en.wikipedia.org/wiki/Code_coverage`