P.E.S. UNIVERSITY

Department of Computer Science and Engineering

Session: Jan-May 2020

UE17CS355 – Web Technologies-II Lab

Project Phase – II

Test Report

Project Title: FitBeast Fitness Website

Section: 6B

Team Members:

Sakshi Goel PES1201700148

Amogh Rajesh Desai PES1201700180

Tanvi PK PES1201700646

Unit Testing

Introduction

Unit Testing is a type of software testing where individual units of a software are tested. A unit is the smallest piece of code that can be logically isolated in a system, for example an individual function, method, procedure, module, or object. The purpose is to validate that each unit of the software performs as expected. It is concerned with functional correctness of the standalone modules.

The units of this application were tested using the "*unittest*" tool provided by python. "*unittest*" is a python standard library, which provides a rich set of tools for constructing and running tests. It supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework.

Objective

The purpose of unit testing is to validate that each unit of the software performs as expected, by isolating each unit of the system to identify, analyze and fix the defects. The objectives of unit testing are:

- Isolation of a section of code.
- Verification of the correctness of code, that is if the application code functions as per requirements.
- Testing every function and procedure.
- Fixing bugs early in the development cycle and saving costs.
- Allowing developers to learn what functionality is provided by a unit and how to use it to gain a basic understanding of the unit API.
- Make the code more reusable.

Test Report

Test Case	Description	Expected Output	Actual Output	Test Result
1.	Registering a new user with valid credentials	Status Code: 200	Status Code: 200	PASS
2.	Registering a new user with invalid email format	Status Code: 500	Status Code: 200	FAIL
3.	Registering an existing user	Status Code: 409	Status Code: 409	PASS
4.	Registering a user with PUT method instead of POST	Status Code: 405	Status Code: 405	PASS
5.	Adding a valid item to the cart	Status Code: 200	Status Code: 200	PASS
6.	Adding an invalid item to the cart	Status Code: 409	Status Code: 200	FAIL
7.	Getting contents of the cart	Status Code: 200 JSON Object: Items of Cart	Status Code: 200 JSON Object: Items of Cart	PASS
8.	Logging out without logging in	Status Code: 500	Status Code: 200	FAIL
9.	Accessing the recommendations	Status Code: 200 JSON Object: Recommendations	Status Code: 200 JSON Object: Recommendations	PASS
10.	Accessing all the other webpages (7 different pages)	Status Code: 200	Status Code: 200	PASS

Observation and Conclusion

As it can be observed, 70% of the test cases have passed. The application is failing for three test cases due to the particular validity being checked in the front-end portion of the application, implemented by using the JQuery framework combined with JavaScript checks in the HTML files and we know that unit testing targets the backend functionality, that includes the API endpoints. Otherwise, the application is working as per requirements.

System Testing

Introduction

System Testing is a type of software testing that is performed on a complete and fully integrated system to evaluate the compliance of the system with the specified requirements. In system testing, the functionalities of the system are tested from an end-to-end perspective. It is a series of different tests where the sole purpose is to test the design and behavior of the system. It also involves the external workings of the software from the user's perspective.

This application was system tested using Selenium IDE. Selenium is a web testing tool which uses simple scripts to run tests directly within a browser. Selenium IDE is a portable software testing framework for web applications that provides a record-and-playback tool for authoring tests without learning a test scripting language. It is a browser plugin that allows software testers to record their test scenarios that can be used to develop test cases.

Objective

The aim of system testing is to ensure that the system will function correctly and properly when all the features are bundled as a whole, because specific features can be working perfectly on their own, but they might not when connected to each other. System Testing involves testing the software code for following:

- Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole.
- Verify in-depth/detailed testing of every input in the application to check for desired outputs.
- Testing of the user's experience with the application.

Test Report

Test Case	Description	Expected Output	Actual Output	Test Result
1.	Sign Up with unmatched passwords	User can not be registered	User can not be registered	PASS
2.	Sign Up a valid user	Sign Up successful	Sign Up successful	PASS
3.	Log In a valid user and Add *MMA, Yoga, Zumba*	Log In successful & 3 sessions added to cart	Log In successful & 3 sessions added to cart	PASS
4.	Add *American Chicken Salad, Veg Thai Noodles, Jamaican Meal, American Garden Salad*	4 food items added to cart	4 food items added to cart	PASS
5.	*For You* Page	*For You* Page loaded with recommendations	*For You* Page loaded with recommendations	PASS
6.	See Cart	Cart shown successfully	Cart shown successfully	PASS
7.	Log In a valid user and Logout	Log In and Log Out successful	Log In and Log Out successful	PASS
8.	Sign Up and Log In a valid user -> Add *Men's Sleeveless, Men's Tracksuit* -> Make Payment	Sign Up & Log In successful & items are added to cart & payment is received	Sign Up & Log In successful & items are added to cart & payment is received	PASS
9.	Twitter Page	Twitter Page loaded	Twitter Page loaded	PASS
10.	*For You* Page -> Add *American Chicken Salad, American Garden Salad, Ethiopian Salad, Men's Joggers, Men's Sleeveless* -> Make Payment -> *For You* Page	*For You* Page shows updated recommendations on payment of the new order	*For You* Page shows updated recommendations on payment of the new order	PASS

Observation and Conclusion

As observed from the table above, 100% of the test cases have passed. Hence, the application is working as per requirements. Also, the cases that failed during the unit testing are successfully tested here, as system testing combines both the front-end functionality, that is the User Interface and the back-end functionality, that includes the APIs.

Performance Testing

Introduction

Performance testing is a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workloads. Load testing is one of the performance testing techniques that is conducted to understand the behaviour of the system under a specific load. It measures the quality attributes of the system such as:

- Speed
- Scalability
- Stability
- Reliability
- Resource usage

This application was load tested by using Postman and k6-script. Postman is used to create requests and organise these requests into a collection. The exported Postman collection is then fed to a postman-to-k6 converter script to generate a k6 script, which is then run using k6 to test the APIs in the application.

Objective

Performance testing can be used as a diagnostic aid to locate computing or communications bottlenecks within a system. Bottlenecks are a single point or component within a system's overall function that hold back overall performance. Performance testing can help identify the nature or location of a software-related performance problem by highlighting where an application might fail or lag.

Test Report

One Postman Collection includes:

S. No.	Request Description	Method	URI
1.	Start Page	GET	/
2.	Signup Page	GET	/signup
3.	Registering a new user	POST	/register_user
4.	Extended Signup Page	GET	/extended_signup
5.	Registering an existing user	POST	/register_user
6.	Login Page	GET	/login
7.	Successful Login Attempt	POST	/verify_login
8.	Failed Login Attempt	POST	/verify_login
9.	Home Page	GET	/homepage
10.	Recommendations Page	GET	/recommend
11.	Merchandise Page	GET	/merch
12.	Adding merchandise to cart	POST	/add_to_cart
13.	Food page	GET	/eat
14.	Adding food to cart	POST	/add_to_cart
15.	Workout Page	GET	/cult
16.	Adding workout to cart	POST	/add_to_cart
17	Cart Page	GET	/cart
18.	Payment Page	GET	/payment

This collection of requests is sent to the application as different loads by varying the number of virtual users and the time duration for which these requests keep getting iterated, i.e. number of epochs.

Observations

1) For one virtual user and varying time durations:

- Checks being performed:

✓ Response time is less than 200ms
✓ Content-Type is present
✓ Your test name
✓ Successful POST request
✓ Status code is 200

- Observed Statistics:

Duration	Iterations	Requests	Avg. Waiting Time	Checks
10s	163	2937	1.79 ms	100%
20s	320	5766	1.86 ms	100%
30s	367	6619	2.44 ms	100%
40s	643	11577	1.92 ms	100%
50s	848	15276	1.89 ms	100%
60s	558	10052	4.30 ms	100%

2) For ten virtual users and varying time durations:

- Checks being performed:

A: Your test name
B: Successful POST request
C: Status code is 200
D: Response time is less than 200ms
E: Content-Type is present

- Observed Statistics:

Duration	Iterations	Requests	Avg. Waiting Time	Checks
10s	100	1909	45.40 ms	A, B, E: 100% C: 99% passed D: 82% passed
20s	201	3695	47.40 ms	A, B, C, E: 100% D: 94% passed
30s	286	5256	50.34 ms	A, B, E: 100% C: 99% passed D: 84% passed
40s	293	5393	65.92 ms	A, B, E: 100% C: 99% passed D: 79% passed
50s	463	8434	52.07 ms	A, B, E: 100% C: 99% passed D: 93% passed
60s	538	9811	54.04 ms	A, B, E: 100% C: 99% passed D: 87% passed

3) For time duration of 10 seconds and varying number of users:

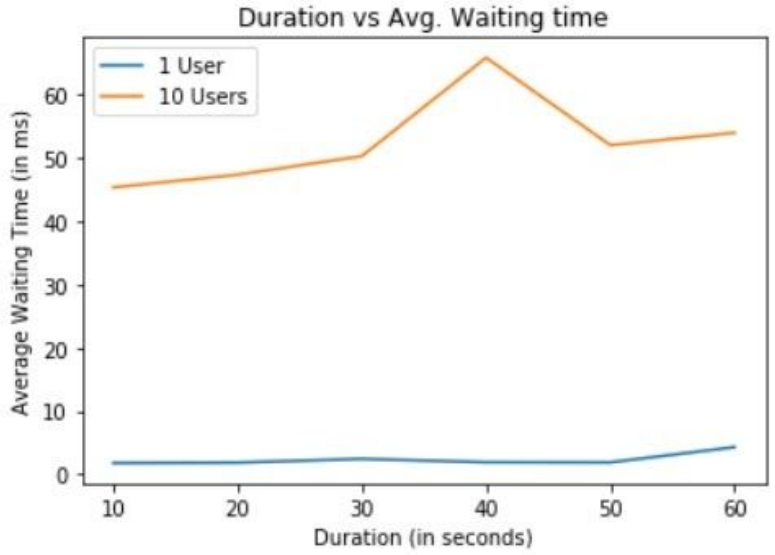
- Checks being performed:
  A: Your test name
  B: Successful POST request
  C: Status code is 200
  D: Response time is less than 200ms
  E: Content-Type is present

- Observed Statistics:

Users	Iterations	Requests	Avg. Waiting Time	Checks
1 user	163	2937	1.79 ms	A, B, C, D, E: 100%
5 users	136	2491	16.85 ms	A, D, E: 100% B: 96% passed C: 99% passed
10 users	130	2406	37.98 ms	A, E: 100% B: 91% passed C: 99% passed D: 97% passed
15 users	120	2345	60.06 ms	A, E: 100% B: 86% passed C: 99% passed D: 89% passed
20 users	120	2306	81.98 ms	A, E: 100% B: 90% passed C: 99% passed D: 47% passed
25 users	125	2450	97.74 ms	A, E: 100% B: 99% passed C: 92% passed D: 41% passed
30 users	120	2441	117.65 ms	A, E: 100% B: 99% passed C: 89% passed D: 35% passed

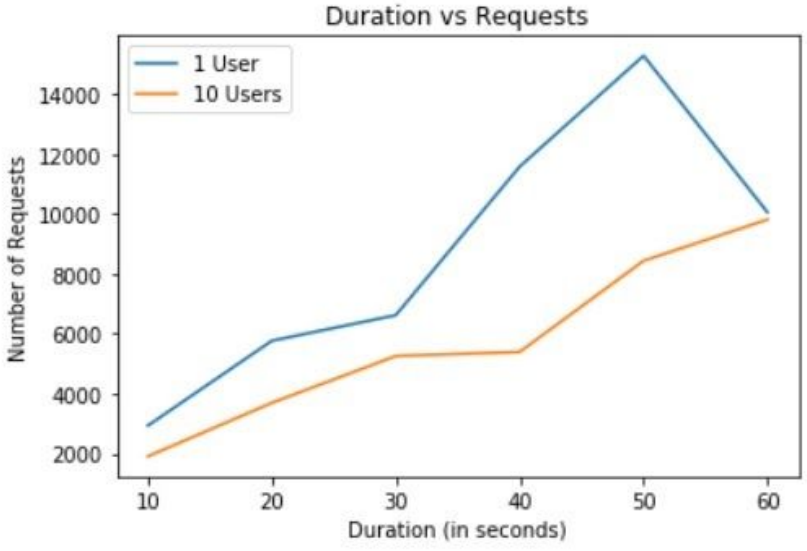
Visualisations

a) Average Waiting Time vs Durations (1 Virtual User and 10 Virtual Users):



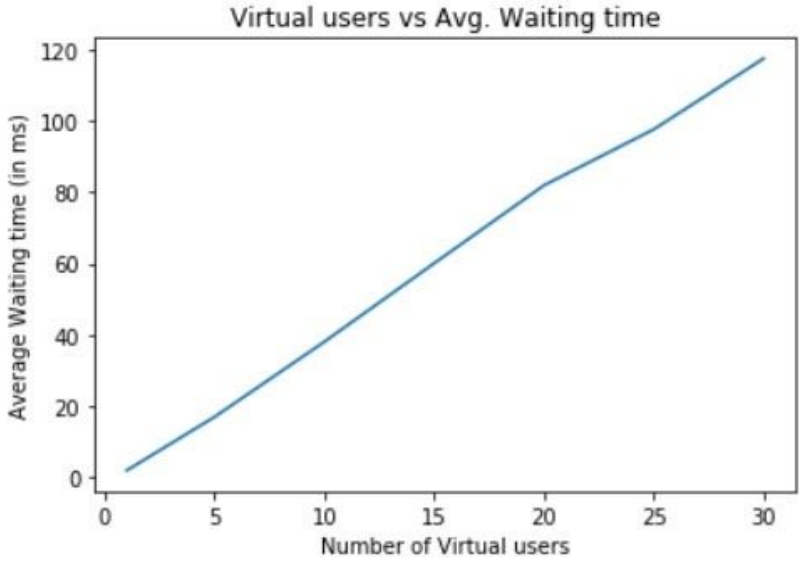
As observed from the above graph, the average waiting time has increased exponentially for 10 virtual users as compared to one virtual user. It can also be observed that the average waiting time for the same number of virtual users with varying durations linearly with a small slope.

b) Number of Requests vs Duration (1 Virtual User and 10 Virtual Users):



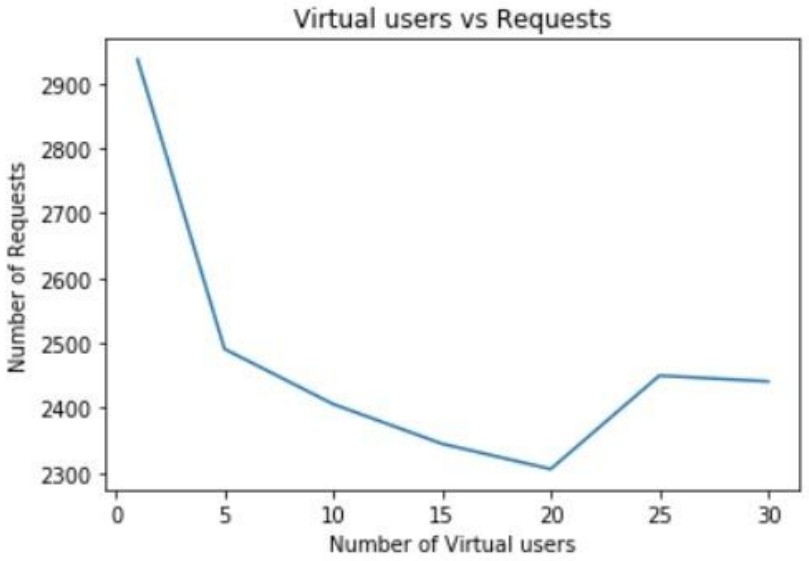
This graph shows that the number of requests responded to increases with increase in the duration for both one virtual user and ten virtual users. Also, it can be inferred that the number of requests for ten virtual users is higher compared to the number of requests for one virtual user.

c) Average Waiting Time vs Number of Virtual Users (For 10s duration):



It can be observed from this graph that the average waiting time increases linearly with the increase in the number of virtual users.

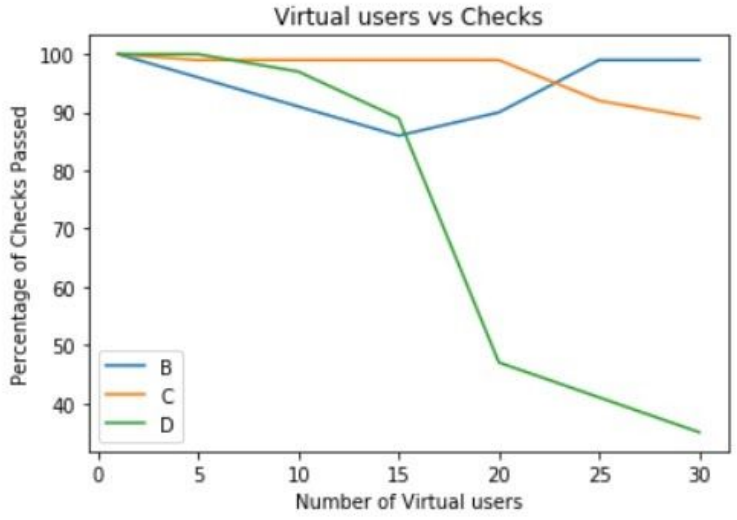
d) Number of Requests vs Number of Virtual Users (For 10s duration):



As the number of virtual users increases, it can be seen that the number of requests responded to by the application tends to decrease. This may happen due to the fact that the waiting time increases with an increase in the number of virtual users and hence, some of the requests in the waiting queue might get dropped.

e) Percentage of checks passed vs Number of Virtual Users (For 10s duration):

B: Successful POST request
C: Status code is 200
D: Response time is less than 200ms



The check 'B', i.e. *Successful POST request* has been passed for most of the cases and has an almost consistent curve for varying numbers of virtual users, while the number of cases that pass the check 'D', i.e. *Response time is less than 200 ms*, decreases with the increase in the number of virtual users. This decrease might be caused due to the increase in response time, which in turn is increased because of the increase in average waiting time as observed from the visualisation (c).

Conclusion

It can be concluded that with an increase in the load, i.e. the number of requests, the average waiting time increases linearly, resulting in an increase in the response time. This might be caused due to the queuing of requests in the waiting queues. It can also be inferred that the number of requests/iterations that the application responds to decrease with an increase of virtual users because of the dropping of requests in the queue.