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: Voting System using Blockchain

Section: 6 A

Team Members:

Akhilarka Jayanthi - PES1201700050

Atul Anand Gopalkrishna - PES1201701452

Kevin Arulraj - PES1201700659

Unit Testing

By: Kevin Arulraj - PES1201700659

1. *Introduction*

The Voting System has a Node JS frontend and a Flask Python backend and uses Blockchain as its intelligent functionality. I performed Unit Testing on the Flask Python backend of the code and tested the PUT, POST and GET methods. I used the *unittest* module and used the *requests* module to send requests to Flask backend.

2. *Objective*

With Unit testing, I aimed to test each PUT, POST and GET method and prove each method's functionality. For example, only PUT and POST can be used to add to the Flask backend but not GET. PUT and POST methods fail if no data is passed to add to the backend.

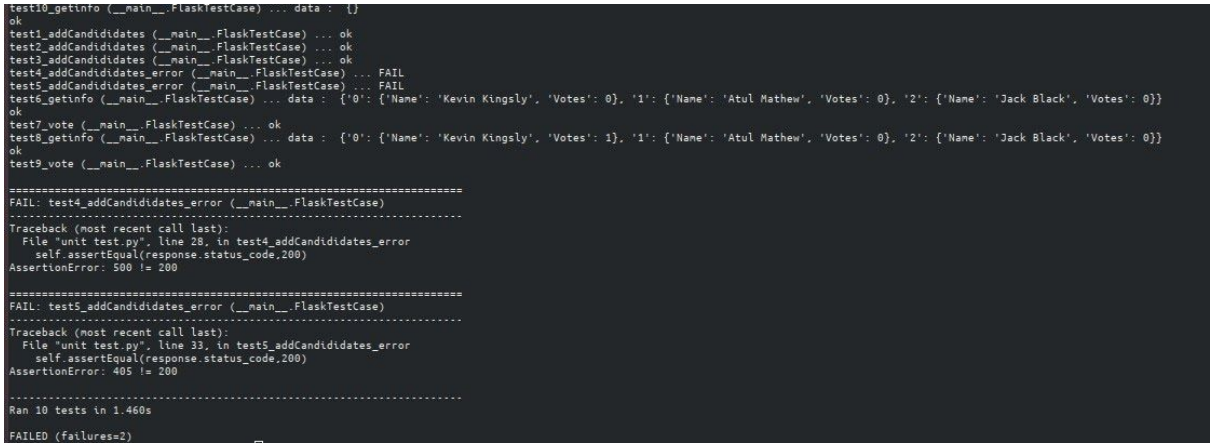
3. *Test Report*

Test case No.	Test Case Description	Expected Output	Actual Output	Test Result (Pass/Fail)
1.	POST method to add Kevin Kingsley as a candidate	Candidate added to Flask backend	Candidate added to Flask backend	Pass
2.	POST method to add Atul Mathew as a candidate	Candidate added to Flask backend	Candidate added to Flask backend	Pass
3.	PUT method to add Jack Black as a candidate	Candidate added to Flask backend	Candidate added to Flask backend	Pass
4.	No data added using PUT method	Data added to Flask backend	Assertion Error	Fail

5.	GET method to add a candidate	Candidate added to Flask backend	Candidate not added to Flask backend	Fail
6.	POST method to display election data on the screen	Election results displayed with 0 votes	Each candidate has 0 votes	Pass
7.	POST method to add a vote for Kevin Kingsley	Vote added to Kevin Kingsley	Vote added to Kevin Kingsley	Pass
8.	PUT method to display election data on the screen	Election results displayed	Kevin Kingsley has 1 vote while others have 0	Pass
9.	PUT method to add a vote for Atul Mathew	Vote added to Atul Mathew	Vote added to Atul Mathew	Pass
10.	PUT method to display election data on the screen	Election results displayed	Kevin Kingsley and Atul Mathew have 1 vote each while Jack Black has 0	Pass

4. *Observation and Conclusion*

80% of test cases passed, as expected. The program functioned flawlessly.



Load Testing

By: Akhilarka Jayanthi - PES1201700050

1. ***Introduction***

   Our application is a voting system using blockchain. The backend is made with flask and restAPI. The tool used to load test this backend is locust. A script is written which then is used to load test the flask backend. The script tested the three api used in the backend which were used for adding candidates, casting votes, and getting poll information.

2. ***Objective***

   The Objective here is to observe the response times and check up to how many number of users the application performs optimally. This was done by writing a locust script which simulates a user. Each user makes requests to add a candidate, cast a vote and then send a request to retrieve poll information.

3. ***Test Report***

   All response time calculations were done in milliseconds.

Test case No.	Test Case Description	Expected Output	Actual Output	Test Result (Pass/Fail)
1	Simulated 10 users arriving at a rate of 2 users per second. Total of 71 requests were sent. Each request is unique even if it was to the same API the data needed was randomly generated and sent.	Since there were a small number of users arriving at a relatively slow rate the response was expected to be fast.	The average time taken for the three API are 217 ms,116ms, 692 ms with a total average time of 456 ms.	Pass
2	Simulated 50 users arriving at a rate of 5 per second. A total of 218 requests were sent. Each request is unique even if it was to the same API the	This was a slightly higher number than the previous case but we still	The average time taken for the three API are 3602ms, 2173ms, 46720 ms with a	Fail

	data needed was randomly generated and sent.	expected fast responses.	total average time of 25747 ms. This took exponentially longer than the previous case and is very slow.	

4. *Observation and Conclusion*

The application works normally for a low number of users up to 20. Once the number of users exceeds this like 50, for example, the response time taken by the flask server is much longer than lower values. It is a slightly exponential increase in response times.

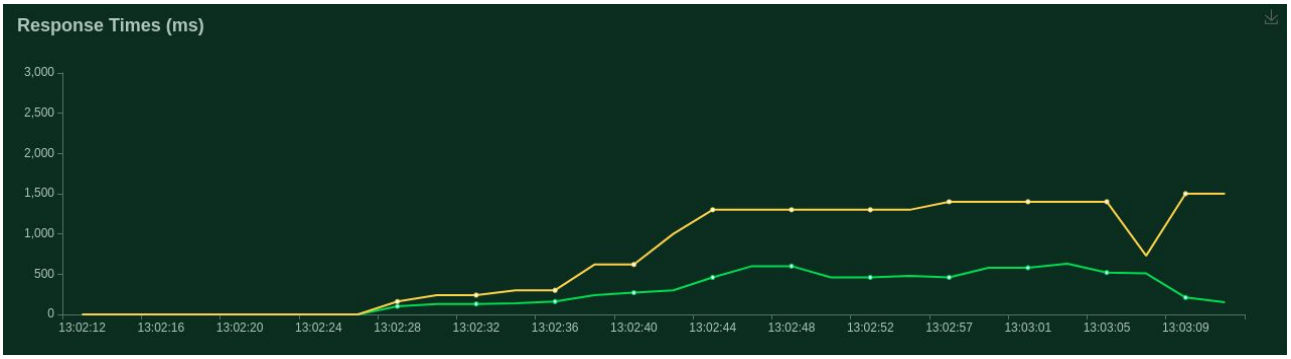
<u>The graphs of time taken for the first test case are given below.</u>

Statistics

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	//poll/add/candidate	22	0	140	460	217	72	1020	3	0.3	0
POST	//poll/cast/vote	11	0	100	220	116	36	292	17	0.4	0
POST	//poll/get/information	38	0	580	1400	692	63	1509	398	0.7	0
	Aggregated	**71**	**0**	**260**	**1300**	**456**	**36**	**1509**	**216**	**1.4**	**0**

Total requests per second

The green line in the graph below is the median response time(y-axis), vs time.

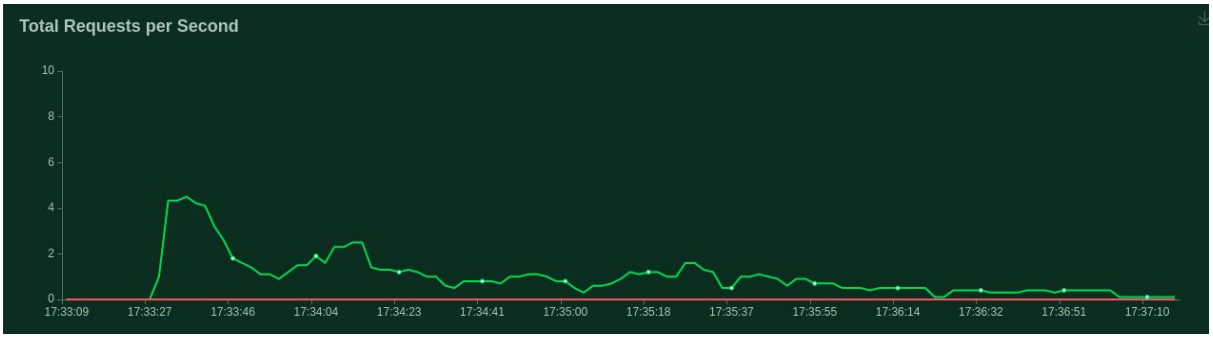


The graphs of time taken for the Second test case are given below.

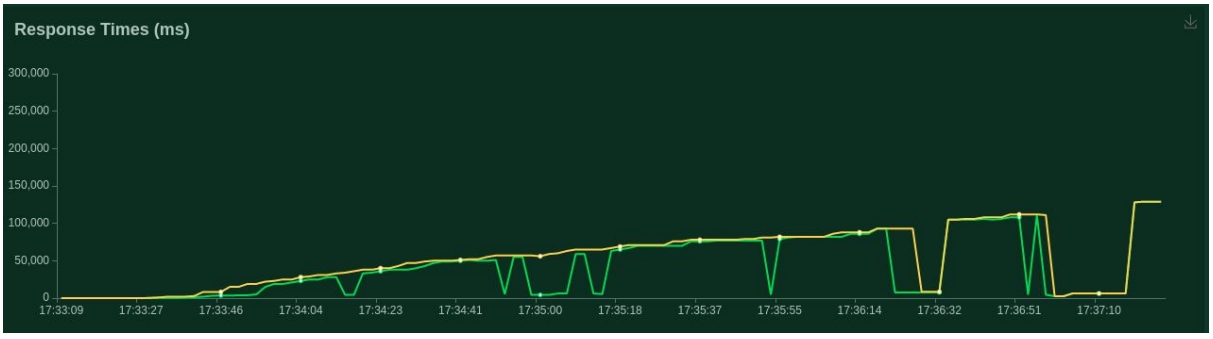
Statistics

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	//poll/add/candidate	74	0	3800	6700	3602	102	8472	3	0	0
POST	//poll/cast/vote	31	0	1600	5000	2173	111	6099	17	0.1	0
POST	//poll/get/information	113	0	49000	82000	46720	85	111628	1063	0.3	0
	Aggregated	**218**	**0**	**5300**	**77000**	**25749**	**85**	**111628**	**555**	**0.4**	**0**

Total Requests per second



The green line in the graph below is the median response time(y-axis), vs time.



System Testing

By: Atul Anand G - PES1201701452

1. ***Introduction***

   The application we built was a voting system using blockchain. The user interaction with the application happens using the react web page. Here we perform system testing which is used to test a system completely end to end to find bugs.

2. ***Objective***

   The objective of the system test is to exercise the system completely end to end and check if any bugs, errors or crashes occur in the process. So this is realized by using a tool called selenium. This tool is used to record interaction with the UI. Then these interactions are stored so that it can be replicated automatically by the tool which can then check for bugs and errors.

3. ***Test Report***

Test case No.	Test Case Description	Expected Output	Actual Output	Test Result (Pass/Fail)
1	The interactions with the web page were as follows. ● The web page was opened ● A navigation button was clicked to access the voting page ● A name atul was entered into the form ● A blockchain account id was entered into the form ● Button was clicked to add a name to the form. ● Another name kevin was added to the form.	Since the duplicate name is entered into the database, the application may not react properly when a vote is cast.	The interactions with the webpage were recorded and then replicated b the selenium tool. The name atul and the name kevin were all added to the list giving a duplicate entry for kevin	Fail

	• Button was clicked twice to add the name and a duplicate to the from • The cast vote button was clicked for the name atul. • The cast vote button was clicked for the name kevin.		which is a bug. Then when a vote was cast for the duplicate entry the application crashed.	
2	The interaction with the web page are as follows: • First the web page was opened and interactions recorded using selenium. • The navigation button was clicked to redirect to the voting page. • In the voting page a name atul was entered and then added to the list by clicking the add button. • Then another name kevin was added to the list. • Now an account address was entered into the form. • Using this account a button was clicked to vote for atul • Then using the same account address , a vote was cast for kevin.	The backend is built using blockchain so it is expected that duplicate votes will not be considered and then discarded.	The application performed in a proper manner after the interactions with the web page were stored and then replicated by the tool. The vote was counted for atul but not for kevin which means the backend worked and did not allow the double vote.	Pass

4. Observation and Conclusion

The 14 out of the 16 interactions with the webpage passed. The application malfunctions when duplicates are entered into the database. The application performs optimally and prevents double voting.