Software Analytics Report

Quality Assessment for C402

Done By: Femi Sowemimo

PROJECT OVERVIEW

Name: Votetell
Website: https://votetell.com/
Languages: html, javascript, css
Github: https://github.com/awaseem/votetell.com

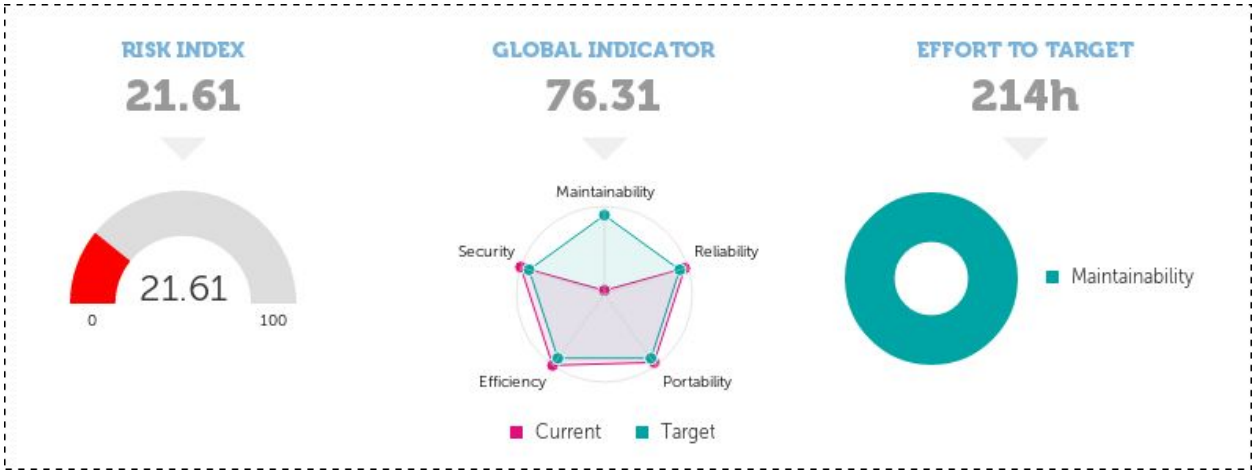
Analysis Date: 2016/02/20

Kiuwan Username: ssowemim@ualberta.ca
Kiuwan Name: Femi Sowemimo

Votetell is a website created by a close friend of mine in Calgary. Its is a simple poll creator that allows users to vote and also send a custom response to the creator. This program was developed with Meteor JS and Semantic UI.

The initial requirement for this project is to use our previous Cmput 401 project and run the Kiuwan Code analysis software on it, but my Cmput 401 project was done in a VM environment, and there wasn't a way to try and wrap all this into a zip file and run the analysis software. A secondary issue with using my Cmput 401 project was the fact that our code was to make changes to an already functioning moodle environment, this having hundred of thousands of code and the Kiuwan wouldn't have been able to fully analyze this. Talked to professor prior to this and she recommended using an open source project, and I felt there was no better open source project rather than something created by a friend of mine.

Summary from running Kiuwan Assessment:



Initial requirements given for the Kiuwan assessments were:

- **Maintainability:** 90
- **Security:** 90
- **Efficiency:** 90
- **Portability:** 90
- **Reliability:** 90

No specific reason for giving it all a performance rating of 90, it was mentioned by Prof. Teresa that this shouldn't be a big deal but as long as the minimum is a PR (performance rating) of 70. I decided to give it all 90s due to the fact that I believe on a web application all these play a major role in the way a web application runs, especially to keep the site up and running effectively. Different website application vulnerabilities are brought to light everyday and always being on top of these vulnerabilities mean that the maintainability, security, efficiency, portability and reliability is always kept at its best.

Maintainability of 90 works, because seeing as this is a web application the idea of having such a high PR means the web application can be repaired in an ease and fast manner so the system is never down for an extended period of time for any reason.

Security having a PR of 90, also means user inputs are all valid, where the user wouldn't be able to perform any code injection to cause the web app to run into any problems.

Efficiency having a similar high PR shows how reliable the code is, having a consistent coding style and following the appropriate code ethics. This can be in terms of assigning variables with proper names along with proper type attribute.

Portability talks about how easily the software can move from one platform onto another, for a web app this is very key because we are faced with people using platforms such as Tablets, Phones, Laptops & Monitors along with a variety of web browsers. Making sure all this checks out is why I gave the portability such a high PR.

Reliability focuses on the software having a specified level of performance. Making sure a web application loads all appropriate information in a fast smooth manner is very key. Hence why the 90 PR.

Risk Index, Kiuwan also provides a Risk index and this represents potential problems that arises for not paying full attention to one's quality of source code. It provides concrete evidence found in source code of application. The way it works is if you have a poor quality, but the efforts needed to get better is low you are not assuming a high risk application because problems will be easily fixed. But if the effort needed to get better is very high risk index will be high as well.

In a perfect world where the quality of application was ideal, no bugs, minimum code duplication and all we would’ve a Risk Index of 0.0. This meaning no risk is identified in the quality of the code.

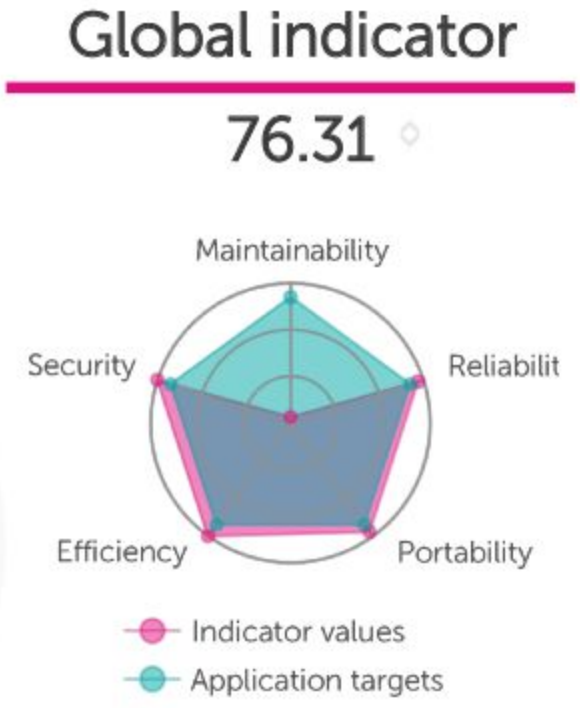
QUALITY

Result from the Kiuwan Assessment after running the assesment.

Characteristic	Quality	App. target
Maintainability	4	90
Reliability	96	90
Portability	96	90
Efficiency	100	90
Security	100	90

Global Indicator: 76.31

The global indicator is a way of showing how well performance is on all basis of the quality assessment. Maintainability is the biggest issue in my code and this is clearly shown in the graph below.



Maintainability scored very low in terms of this web app project, taking a look at a more detailed look at repairs I am faced with the following issues that resulted in my quality assesment.

Defects	Files	Rule Name	Characteristic
11	7	Avoid statements without semicolon	Maintainability
2	2	Do not update control vars in 'for' loop body	Maintainability
1	1	Avoid popup windows	Reliability
13	6	Define Variables with var	Reliability
2	2	Avoid declaring with names already used	Reliability
64	27	Non-portable function check	Portability
48	**18**	**Duplicated code: big block**	**Maintainability**

Maintainability having such a low score greatly has to do with the problem of "Duplicated code: big block". This issue occurring 48 times and having shown up in about 18 files and totaling up a time of 192h in being able to fix this issue.
Going through the full kiuwan website assessment and observing the Maintainability analysis code, the problem with the duplicated code is due to the fact that a semantic-ui is being used.
Semantic-ui is a development framework that creates HTML with ease, along with possible templates that can be used. This making sense for codes being duplicated so much, because if we are using a button element from a template it doesn't group them into one. It simply duplicates the code for each button created since this is how the Semantic-ui works along with most template platforms.

REPARATION EFFORTS

The reparation efforts shows you the cost of fixing the quality of your code in other to reach your goal.
It has calculated the minimum set of defects that will be corrected to achieve it. Showing you the amount of time needed to invest into the source code purely for fixing the quality.

Characteristic	Effort to target
Maintainability	214 h
Reliability	0 h
Portability	0 h
Efficiency	0 h
Security	0 h

This chart/graph drives home this point of Maintainability being the biggest issue, and taking a look at previous charts we realize the main problem with Maintainability is the case of ‘Duplicated Code: big blocks’.

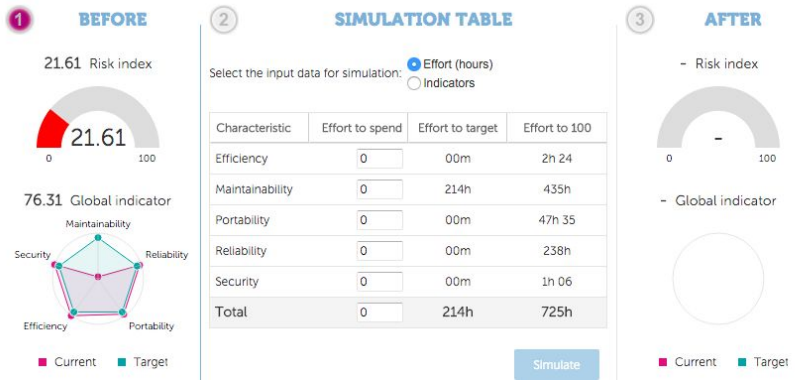
ACTION PLAN

Action plan will be first make appropriate repairs first, Kiuwan is also great for this because it provides a list of TOP 10 Repairs to take on. These defects are the ones that once eliminated give more benefit by unit time of effort.

Rank	Defects	Files	Rule name	Lang.	Characteristic	Priority	Effort
1	1	1	Never use JavaScript 'history' object or navigation-based positioning functions.	javascript	Security	1	6m
2	11	7	Avoid statements without semicolon.	javascript	Maintainability	2	33m
3	2	2	Do not update control vars in 'for' loop body.	javascript	Maintainability	1	1h
4	1	1	Avoid popup windows.	javascript	Reliability	1	30m
5	13	6	Define variables with var.	javascript	Reliability	2	1h 18
6	2	2	Avoid declaring a variable with a name that is already used.	javascript	Reliability	2	12m
7	10	8	Avoid using a comma at the end of the last element in the declaration of an array or object.	javascript	Portability	2	30m
8	64	27	Non-portable function check.	javascript	Portability	2	6h 24
9	2	1	Avoid using navigator.userAgent ('browser detecting') for writing portable code.	javascript	Portability	2	12m
10	48	18	Duplicated code: big block	javascript	Maintainability	1	192h

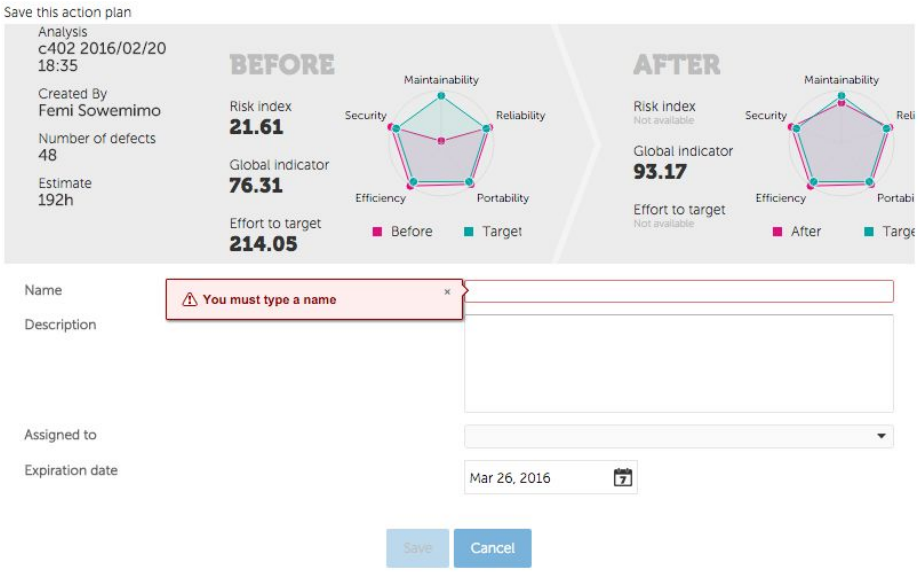
Taking into account the last defect, the issue of fixing ‘duplicated code: big block’ does not have to mainly due to code written by the programmer. Especially when the programmer is using applications such as seismic UI that rely on duplicating already existing code.

My action plan will be to first fix the top 9 repairs on this list. Then going through my code and making sure no duplicated code made by the programmer is found. Once this is done, I’ll now rerun the Kiuwan assessment once more but making a big adjustment to the Maintainability target, likely setting it down to about 60-70, while improving others to likely 95-100. Just so a big focus isn’t being put on this idea of having duplicated code in the application and we can aim to perfect the application.



This image shows a secondary main function from the Kiuwan assessment. It is being able to create ‘ACTION PLANS’. It is very handy because it shows you a current view of where you are at and also helps you distribute hours into the project to be able to increase the quality of your code to your set target or to the ideal spot of 100%. It also lets you assign issues to users if you’re on a team, this holding people accountable for getting all their issues sorted out.

This image shows the process of assigning an issue to someone.



OTHER

The Kiuwan pdf report also provides more information but due to the fact they're not all relevant to this assignment, they wouldn’t be included in this writeup. Please look for the C402 Software Analytics Report pdf to view more informations. Especially informations regarding Metric Values, Quality Distribution in Files and Metric Distribution in Files.