**Graduate Diploma in Information and Communication Technology**

**BCCE301 Cooperative Education Project**

**Analysis of JavaScript Testing Frameworks in Canterbury**

**Methodology Essay**

**On Test Driven Development (TDD)**



Swathi Soman

99162569

**Department of Computing**

**Te Horo Rorohiko**

Contents

[Introduction 2](#_Toc498537711)

[Part A – Test Driven Development(TDD) 2](#_Toc498537712)

[What is TDD? 2](#_Toc498537713)

[Why TDD? 3](#_Toc498537714)

[History of TDD 3](#_Toc498537715)

[Techniques of TDD 4](#_Toc498537716)

[Advantages of TDD 5](#_Toc498537717)

[Disadvantages of TDD 5](#_Toc498537718)

[Part B – TDD in Practice 6](#_Toc498537719)

[Reason to choose TDD 6](#_Toc498537720)

[Applying TDD 6](#_Toc498537721)

[Test-driven development cycle 6](#_Toc498537722)

[Additional code Practice 8](#_Toc498537723)

[Part C - Compare and Contrast 11](#_Toc498537724)

[Tasks based on theory and the project 11](#_Toc498537725)

[Summary 12](#_Toc498537726)

[Bibliography 13](#_Toc498537727)

# Introduction

In this report, the methodology essay topic for the project is Test Driven Development (TDD). The report is divided into three sections: 1) Theory on TDD, 2) Process on how TDD is used within the project and industry 3) a compare and contrast of TDD used in the project with the theory

# Part A – Test Driven Development(TDD)

## What is TDD?

The Test-driven development (TDD) is a software development process which mainly relies on the re-occurrence of a very short development cycle. "TDD" refers a programming style which has three activities: coding, testing and design (refactor) (Agile Alliance, n.d.).

Requirements are analysed to design certain automated test cases, and then the improved software is used to pass the new tests. It is a transformative way to deal with development, which combines TDD where tests are written before the code to satisfy that test and refactoring (Test-driven\_development, n.d.). Refactoring is the process of changing the structure of the code without changing its behaviour; with this, the code developed would be easier to understand.

For all the small application functions it starts with designing and developing tests in TDD. In the below given figure1 the developer writes an automated test case that fails initially and requires improvement. Then a code will be produced after that so that test pass and then the new code would be refactored to the acceptable standards.

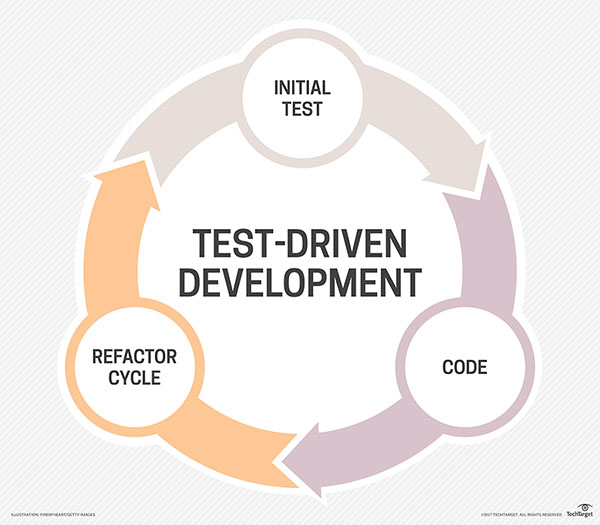


Fig1: Test-Driven Development (Rouse, n.d.)

It starts with the designing and developing tests for all little functionality of an application. In TDD approach, the test which is developed first specifies and validates what the code must do. It can be shown as “red-green-refactor cycle”. Before the code is written and any new functionality is added to an application, an automated test would be written by the developer stating how the code should behave, and wait for it to turn red (fail to pass). Then the code would be written to the specification which would turn the test to green (test passes). Finally, the developer will make sure that code written will be clean and executable (refactoring) (wikipedia, 2017).

## Why TDD?

TDD is part of holistic design, development and testing process, it’s not a testing technique. The main concept of TDD is that the test is written first and not the code to make sure that the code works. Since the test is written first and then the code, this will get all the tests to pass. This is known as a Test-First approach.

There are two views to practice TDD in software development

* In the first view before the actual code is written the TDD is mentioned as the technique which can be used to specify the requirements and design
* In the second view the actual code can be developed by the programmers using TDD.

Regardless of the views mentioned, TDD not just improves the code, but it also improves the implementation of the software system and its complete design. (Barber, 2017).

By using TDD it solves many issues by implementing refactoring, good practice of coding and testing can be followed by the programmers. A good program not just works, but it’s simple and easy to maintain. Code which is written well not only saves the time and money it can change the test time according to the requirement in business. At starting stage, the programmers feel that lot of effort must be put forth, but gradually it facilitates easier ongoing delivery. It is always difficult to understand the code written by another person, it happens when each person has their own style. But it is not necessary that the code written by each person should be good and efficient. Many coders do not follow certain coding practices such as, commenting properly, providing good indentation, vertical coding, etc (helpingtesters, 2017).

## History of TDD

The latest rediscovery of TDD was attributed by Kent Beck, he is the creator of the extreme programming. Software Development community widely accepted TDD through Extreme programming and Agile software development.

* Extreme programming (XP) is a methodology in software development, which can be used to improve the software quality and its responsiveness according to the requirements of the customer. It is an agile software development which promotes short development cycles, which is used to improve the productivity of the customer and introduce checkpoints at which new customer requirements can be acquired (Extreme programming, n.d.).
* It’s known that Kent Beck “rediscovered” TDD since the early days of computing which was in 1960s dates the prototype of TDD. The biggest difference between the old and the new TDD is that the old TDD requires manual testing whereas the modern TDD is facilitated to automate. As of today, TDD is fully automated test-driven (Barber, 2017).

## Techniques of TDD

The hardest thing in TDD is writing tests. The syntax or rules can be learned easily in very less time. The problem is how to obtain an idea of what is to be done and then verify to check if the function developed works.

The question arises of how can it be used and how many functions are required, later test it. TDD is the method of developing or thinking about the code in a unique way.

Think of any kind of idea and something which is different and can be tested. Once the tests are written and it becomes a routine like writing conditions and loops. Test can be written without thinking about it lot.

* A technique is built up to address the constraints of TDD examined above with respect to the production of new techniques on teaming up classes under an easy falling flat unit test. When another strategy is created on an alternate class, TDD is recursively applied. So before making the new technique, another test is made on alternate test class, which is compared to the different class. This indicates that there are two simple tests which is coming up. So the TDD rules are changed and run the test of second class while with the new method. Once the method is done it can be confirmed that there is only one failing test and can start working on the original method. (Vandegriend, 2009)
* Upon getting the current test to pass, there are few scenarios which can be chosen to write the upcoming tests. While choosing a scenario choose those which would fail at first and not to choose a scenario which passes automatically. This shows the pattern between the pass and fail of the test. When all the scenario which indicates failure is tested, a scenario is added which can be passed. The production code gets over and the tests are still written until it confirms that the other scenarios given are correct. (Vandegriend, 2009)

#### Refactoring

Refactoring is one of the procedure of TDD and that is its main advantage. It is mainly because refactoring requires code which finishes the test scope, hence when the programmer is following TDD, the code developed would follow the requirements constantly. Refactoring of the code is the reconstructing the present code to extent the design and the structure of the code. Before refactoring test coverage is required, since test will guarantee that it will work as required and the functionality of it do not change.

Once the programmer refactors’ the function, they can run the test again to ensure that the behaviour of the test works as expected and the test pass. Thus, it will give great confidence to the developer to refactor the code that they would have avoided. The developers will never know if the changes they made have broken anything (wikipedia, 2017).

## Advantages of TDD

TDD is a software development method in which the code is tested constantly. TDD develops software of high quality in a short span of time. It saves cost, effort and much more. Below list mentions few advantages of TDD:

* It is first a safety net. All the existing code is tested. As human programmers make mistakes when the code is changed or improved
* TDD will allow to spot bugs quicker. All the errors in the program while testing is identifiable easily.
* TDD focuses on good design. To make the code testable, it must have a good structure.
* TDD helps to maintain the code easily and refactor. The code implemented is of good clarity.
* Helps prevents defects – If there is any issue with the requirements or with design at the beginning it is found. If there are any small mistakes it is caught immediately and thus helps the programmers to find the mistakes.
* Helps the programmers really understand their code – The code written by the programmers are easily understandable
* Helps support refactoring as needs and design changes
* Provides early warning to design problems: If there is any errors in the code. Early warnings will be provided
* Programmers learn how to write other kinds of tests: Programmers learn different form of tests
* It encourages small steps and the principle that it is easier to keep a system working than fix it after you break it (Grenning, 2016).

## Disadvantages of TDD

Even though there are many advantages for TDD, there are few disadvantages, which makes the TDD less efficient. Below list shows some of the main disadvantages:

* It consumes a lot of time and effort for the development - Takes lot of time thinking of how to write the test, which requires lot of effort.
* Focusing on the less hard plan now and not thinking ahead can mean major refactoring essentials.
* It's difficult to write wonderful tests that cover all the basics and keep away from the unnecessary.
* Significant investment is required and a push to keep up the test suite – it must be reconfigured for most extreme esteem.
* If there is rapid change in the design, the test should be changed frequently. Most of the time would be wasted by writing tests (Hill, 2015).

# Part B – TDD in Practice

## Reason to choose TDD

TDD was chosen due to my familiarity and my experience on working with it. I had a heap of knowledge on the subject as I have worked with it and have studied the subject in SE101 JavaScript Programming. This knowledge helped me in choosing TDD which would help in developing a well-structured code. TDD fulfils the project requirements and can be used by the staffs, students or developers for their projects. It is equally beneficial for me and the future developers.

## Applying TDD

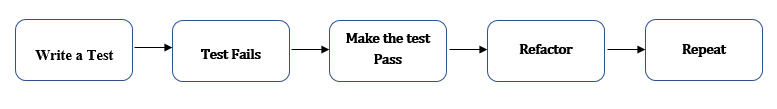
The below mentioned steps describes about how TDD is applied and I was introduced to this topic through SE101. Which demonstrate the systematic programming on Jasmine Framework and its development using TDD. The TDD is used along with BDD for the test practices.

Fig2: Process of TDD

## Test-driven development cycle

The below given steps shows the cycle of Test Driven Development cycle.

###### 1: Write a failing test

Writing a test is first step of TDD. The programmer should clearly analyse the specifications and requirements before the test is written. Once a specification and requirements has been understood, the programmer can write an automated test to test given requirements. The developer writes the test and minimal code to run the test. The test fails initially as the actual functionality is not developed. The test which fails initially is the actual aspect of TDD and that is how the test begins.

###### 2: Code to pass the test

When the automated test is failed, the programmer writes a code to pass the test. As the programmer works on the code an automated test can be executed to see if the code is working. The programmer gets feedback instantly on this which increases the productivity of the programmer. The developer works on the code until the test is passing. Thus, the developer can be confident on the program developed as it is passing and the code meets the test requirements.

###### 3: Refactor

The detail of refactoring is given below; once the test passes the code can be refactored as required. The base of the code should be cleaned regularly during TDD. The newly developed code can be moved from its convenience of passing the test to where it belongs. The code duplication should be removed. There are specific guidelines for creating clean code. Re-running

through each phase of refactoring the programmer can be confident that the process does not redo any functionality.

###### 4: Repetition

When the program has the test passed and has a clean and refactored code, then code can be repeated. Starting with other new test function and proceed in the same manner. If the code which was developed newly do not satisfy the test or if the test fails, the programmer must undo the new code or return to immoderate troubleshooting. As the developer goes through all the functions, the application which is developed would be actively working. One of the main advantage is TDD is that as the programmers complete the full test with the test coverage. It is extremely uncommon that a programmer would backpedal to their code and write tests for everything after having a completely actualized application (wikipedia, 2017).

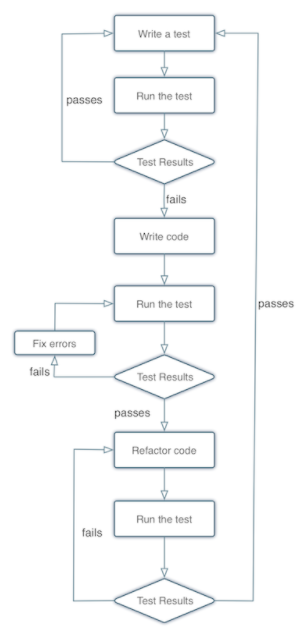
The flowchart given below describes about each phase of the TDD life cycle. At first the test is written and its run to check the test result. If the test passes the process will be repeated. If the test fails, write the code so that the test pass. When the code is written, and the code passes the test, the code will be refactored, and the test would be run to check if there are any further errors. Any further failures in the code should be refactored again. If the code passes, the process will be repeated.

Fig3: TDD Cycle (Barber, 2017)

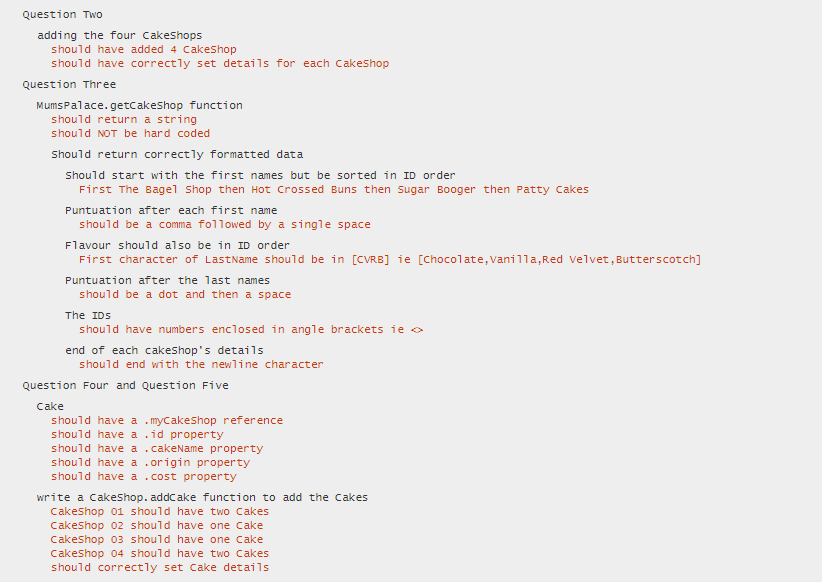
## Additional code Practice

When writing code for TDD, I usually prefer to go with a usual and an easy approach, by making a simple code it can be read and understood easily. I will also make sure that the software which I use for the code is relevant and is of latest version.

###### Example:

The example given below shows the after and before versions of the use TDD in the project. The tests are written first which fails at first and then the code is written to pass the test. The tests are written in BDD.

Before: The tests are written at first which fails.

Fig4: Failed TDD test

When the test fails, the code is written further so that the test passes. The code is written for the above mentioned test to pass.

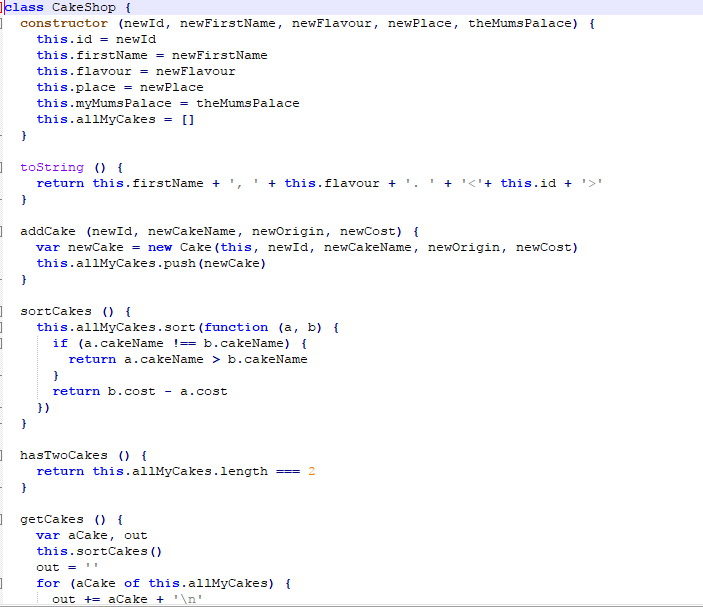


Fig5: Code to pass TDD

The above given is the code which is written for the test to pass.

After: The below images shows the test which passes after the code is written.

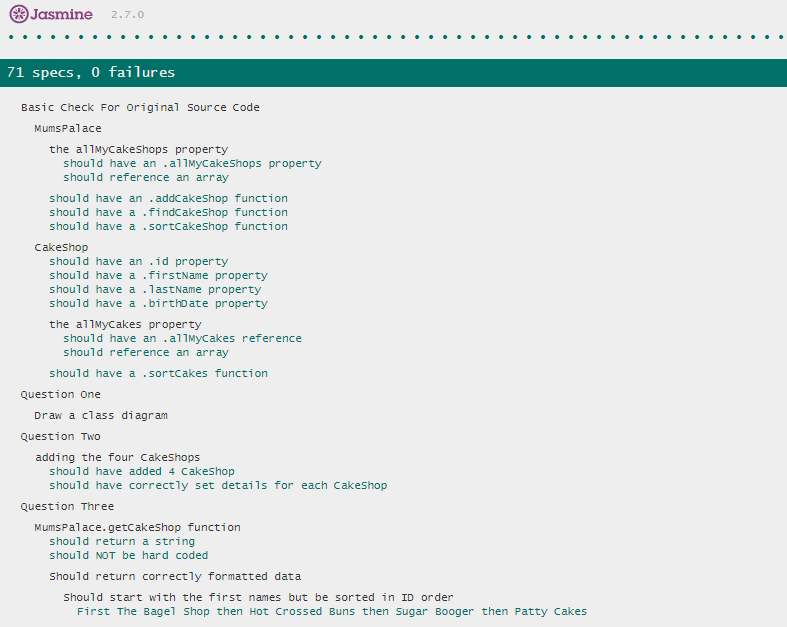


Fig6: Passed Tests

The above given example shows about the test which was written first and it failed. The test was written in Jasmine testing framework one of the popular testing framework in JavaScript Testing.

The version of Jasmine used is 2.7 and which contains of lib, spec and src folders. The lib folder contains library files which is required for the program execution. Spec folder is used for writing tests, there were 7 scenarios which is edited into different test in the above mentioned example and src folder is used for writing the code which matches the test.

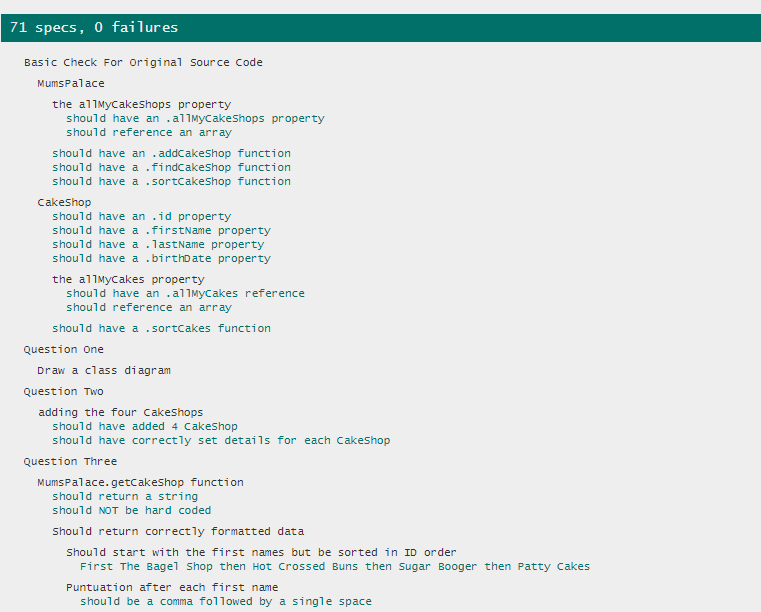
Seven scenarios are executed to fail at first and then the code is written to make them pass. Once the code is passed it is then refactored to make sure that the code is well structured and behaviour driven. The spec runner is run again to check if the code is passing fine.

Fig7: Refactored Code

The image above shows the code which is refactored and works efficiently.

###### TDD in Industry

TDD is the powerful practice in the industry. TDD helps to deliver value quickly to the user while reducing the stress of developing the program and its release to the production environment.

In industry when TDD is being coded it is stopped now and then to check if the code can be released safely. Most of the time that can be done even if it is unfinished. TDD pattern is not red, green and refactoring it always. It may also be red, green and release at times. And after doing that there would be several refactor and release cycles in the process. TDD can also be considered as a great tool for practising evolutionary design. (Quora, 2016)

# Part C - Compare and Contrast

In the part c section the TDD will be compared and contrasted with the theory and how it is being used within the project. From the recap of the above mentioned sections, TDD is a software development cycle in which the developer writes the test first and a small code is written to check if the test passes and then the code is refactored to the standard of acceptance.

## Tasks based on theory and the project

As seen from the above sections TDD is a software development process that relies on the repetition of a very short development cycle. "TDD" referred to a programming style which has three activities: coding, testing and design (refactor). The same process is applied in the project from the developing stage and it worked.

###### Theory recommends to do the below points and its done.

* Each developer mainly focusses on the results which are necessary to validate the test.
* The test are designed time related which allows the execution of the test even in the un-realistic operating system
* The test code should be as same as that of the production code. The code should work well even for the positive and the negative case, which is readable and maintainable also it lasts long.
* Share the essential techniques required for the test and get along with the team.

From the theory there were different methods which is mentioned in the development of TDD. All of which is simple and used during the development cycle. By just focusing on writing the code which is necessary so that the tests will pass, the design is clearer and cleaner when compared to the other methods. For the tests to achieve design pattern which is advanced, it is written to generate that design. The code will be simpler and it would pass all the required test which makes the developer to focus only on what is important. (wikipedia, 2017)

###### Theory says to do it and it’s not done

Advances practices on TDD results in acceptance test driven development (ATDD) and simplifying by using an example. The criteria specified by the customer are further automated, which later drives the traditional Unit test driven development. This process is used to identify if the customer has the automated mechanism to check if the customer meets all of their requirements. The team has specific target when using ATDD which can be satisfied and will keep focused on what the user or customer wants. Thus as per the theory the process wasn’t done and it appeared to be ok without any problem. (wikipedia, 2017)

###### Theory says not to do and it is not done

* All the unit test should be started with a known and pre-configured state.
* Depending on test cases and the test suite in which the test case are dependent are complex and fragile.
* Execution request should not be assumed. Refactoring of the tests or the structure can cause extensive effects in the tests.
* The tests can cause falling false negatives. An error in an early tests breaks a later tests regardless of the possibility that no real blame exists in the unit test, which increases the defect analysis and efforts on debugging.
* Testing exact execution behaviour based on the timing or execution.
* Running of tests which is slow.

###### Issues encountered

* Theory says it's difficult to write wonderful tests that cover all the basics and keep away from the unnecessary. But the test were written which sometimes caused the design to be unclear and hence the code had to be rewritten which had big loss of time.
* Maintaining test code: Breaking a test or a very small test is actually good. By this it is understandable that how the new code would affect the system .But if the tests are written badly, coupled tightly or not structured in a proper order. Maintaining of the code would be tough. Thus, it causes more work and the test will be the thing which will be deleted first, which will cause loss of time. Thus the theory says not to do certain things and it is done which encountered few issues.

# Summary

TDD is an approach of development in which the test is written by the developer which shows the piece of work which is worked, and then the code is written further so that the test passes. It is development technique in which the tests are written which fails before a new code is written. The result of the TDD is best when the code is improved. It can drive the way the program has to be designed their application, without any fear of breaking it.

TDD is an extensible and modularized code. The software can be considered as small units where the tested and written independently by the developers. By using TDD in organisation it helps to update the software according to the business requirements painlessly. If there is any changes in the environment and the organisation reacts to it, only then the organisation is successful and it can make further changes in the environment. By using TDD it makes maintainability, extensibility and flexibility possible. (Novoseltseva, 2017)

# Bibliography

*Agile Alliance*. (n.d.). Retrieved from www.agilealliance.org: https://www.agilealliance.org/glossary/tdd/#q=~(filters~(postType~(~'page~'post~'aa\_book~'aa\_event\_session~'aa\_experience\_report~'aa\_glossary~'aa\_research\_paper~'aa\_video)~tags~(~'tdd))~searchTerm~'~sort~false~sortDirection~'asc~page~1)

Barber, D. (2017, March 27). Retrieved from http://derekbarber.ca/blog: http://derekbarber.ca/blog/2012/03/27/why-test-driven-development/

*Extreme programming*. (n.d.). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Extreme\_programming

Grenning, J. (2016, May 11). *quora*. Retrieved from www.quora.com: https://www.quora.com/What-are-the-pros-and-cons-of-test-driven-development

Guru99. (2017). *Test Driven Development (TDD): Learn with Example*. Retrieved from test-driven-development.html: https://www.guru99.com/test-driven-development.html

*helpingtesters*. (2017, August 31). Retrieved from www.helpingtesters.com: http://www.helpingtesters.com/guide-to-test-driven-development-tdd/

Hill, S. (2015, February 23). *lean testing*. Retrieved from https://leantesting.com: https://leantesting.com/test-driven-development/

Novoseltseva, E. (2017, Jan 19). *Dzone*. Retrieved from https://dzone.com: https://dzone.com/articles/20-benefits-of-test-driven-development

Rouse, M. (n.d.). *Tech Target*. Retrieved from searchsoftwarequality.techtarget.com: http://searchsoftwarequality.techtarget.com/definition/test-driven-development

*Test-driven\_development*. (n.d.). Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Test-driven\_development

*tutorialspoint*. (2017). Retrieved from www.tutorialspoint.com: https://www.tutorialspoint.com/software\_testing\_dictionary/test\_driven\_development.htm

*wikipedia*. (2017). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Test-driven\_development