Table of Content

[Version Control 2](#_Toc23862462)

[Implementation of Version Control 3](#_Toc23862463)

[GIT and GitHub 3](#_Toc23862464)

[GIT commands: 4](#_Toc23862465)

[Benefits of using Version Control 7](#_Toc23862466)

[Test Driven Development (TDD): 8](#_Toc23862467)

[Behavior Driven Development (BDD): 12](#_Toc23862468)

[BDD Workflow: 12](#_Toc23862469)

[Tool used for BDD: 13](#_Toc23862470)

# Version Control

Simply, Version Control Systems (VCS) can be understood as a category of software tool that helps an individual or a team to manage the changes to the source code of a product over the time of production. The greatest merit of using a version control system is that if an error occurs during the production phase, the developer can go back to an earlier version of the product which allows the developer to compare the code and to fix the error/mistake without disturbing other members of the team.

The code for a program or a product is generally divided within a folder structure or file tree where codes related to the database, frontend, scripts etc. are stored. One developing member of the team maybe busy on adding new features for the product whereas another developer maybe trying to fix bugs at the same time, each developer may change specific features in several portions of the file tree.

Version control traces every individual change by the respective contributor and helps prevent concurrent work from conflicting with each other. A good version control software facilitates a smooth and uninterrupted flow of modification to the code without imposing one particular way of working. (Anon., 2019)

Some of the free or open source Version Control tools that are popular now-a-days are as follows: (Anon., 2019)

* **CVS (Concurrent Versions System):** This version control system is an important component of SCM (Source Configuration Management). It is free to use and it can record the history of the source files and documents it.
* **SVN (Apache Subversion):** Subversion control system is a popular system that many large products use such as Python apache, Ruby etc. even google uses it to distribute the required codes.
* **GIT:**  GIT is relatively a newer version control system that have become well known in a short amount of time. Due to the unique features of this system many people chose this system over others.
* **Mercurial:** This version control system is similar to GIT but it mainly focuses on large projects instead of independent small-scale projects.

## Implementation of Version Control

The project of our team is an Online Auction House. Users can bid with each other to obtain a product auctioned at the system site. The system or application is fully web based with functions similar to an authentic auction. For the backend portion of the system “NodeJS” has been implemented and for the database portion of the system MongoDB has been used. For the frontend portion of the system HTML has been used along with Bootstrap and CSS for better outlook as well as for better user experience. AJAX as well as JavaScript has been used for the logical portions of the frontend as well as to connect the Frontend system with the Backend portion of the system. For the Version Control for the system GIT has been used.

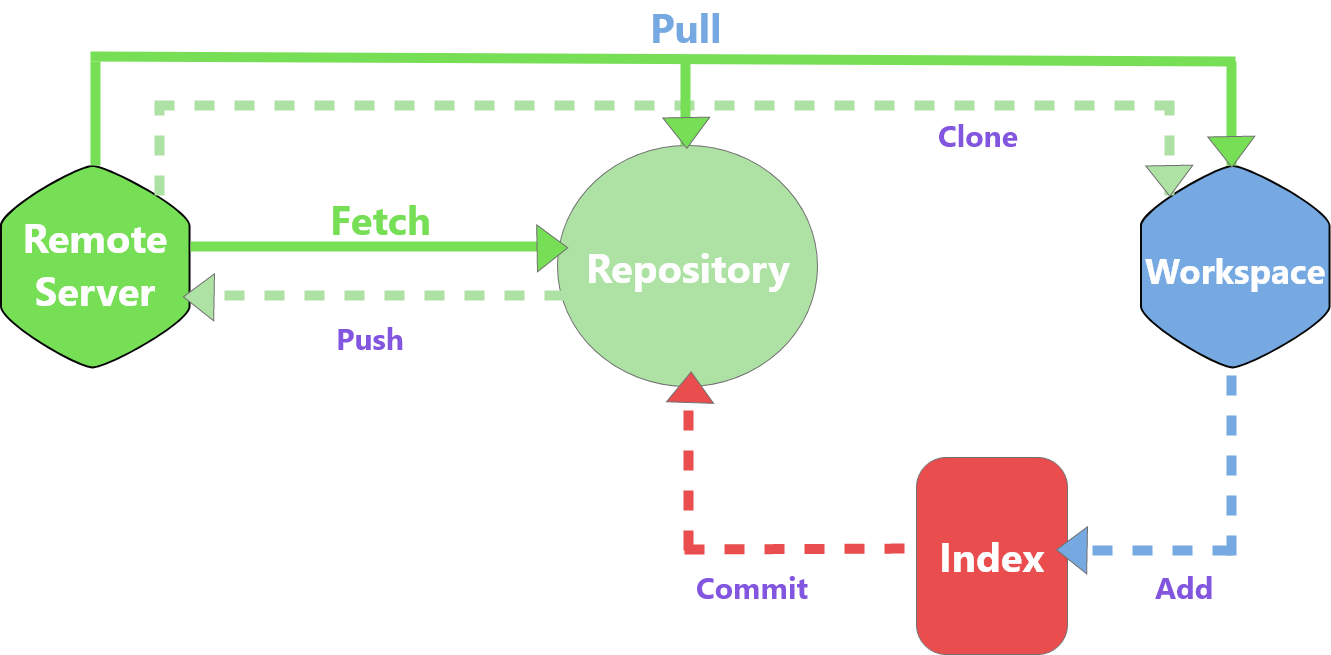
## GIT and GitHub

GIT is a version control system that helps developers to manage or control the project files for a software or application. The primary function or feature of GIT is to keep track of every changes made in a project by the developers. GIT mainly keeps track of changes of the codes used to develop the product. To develop a system the developer first needs to build a basic version for the system and needs to improve the system over time by adding new features and develop functions as required for the system to work. While developing a system various types of error and bugs can arise that can affect the whole work which can affect the functions the developer is currently working on. Hence, if a developer wants to revert or rewind back to a previous version where there were no bugs or errors, he/she can do it easily with the help of GIT. GIT also allows more than one person to modify codes for the system as a team at the same time without disturbing other developers with the help of “branching”. (Anon., 2019)

GitHub is basically a web-based service for version controlling using GIT. GitHub is basically a social networking site for developers where one developer can view other people’s codes as well as work with their code if permitted. It can basically be categorized as an online database where developers can store and track the changes in their project/system. (Anon., 2019)

## GIT commands:

Some of the basic GIT commands used for Version controlling are as follows: (Anon., 2018)



**Git init:** This command is used to create a local GIT repository for the project.

*Command: git init*

**Git clone:** This command is used to copy a repository from a remote server to the workspace.

*Command: git clone*

**Git add:** This command is used to add files to the working area of the project.

*Command: git add .*

**Git commit:** This command creates an image file or snapshot of the saved changes made to the files of the project along with a message for the change code.

*Command: git commit -m “Something”*

**Git config:** It is used to set user-specific configurations like user-email, password, username etc.

*Command: git config --global*

**Git status:** This command displays the list of altered files along with uncommitted files.

*Command: git status*

**Git push:** This command uploads the local commits of the project to the main branch of the remote server.

*Command: git push*

**Git checkout:** This command creates branches as well as helps the developer to navigate between those branches.

*Command: git checkout branch name*

**Git remote:** This command shows all remote repositories.

*Command: git remote -v*

**Git branch:**  It lets you create, list or delete the branches of the system/project.

*Command: git branch*

**Git pull:** This command merges the present code in the remote repository with the working directory.

*Command: git pull*

**Git merge:** This is used to combine a branch into the active one.

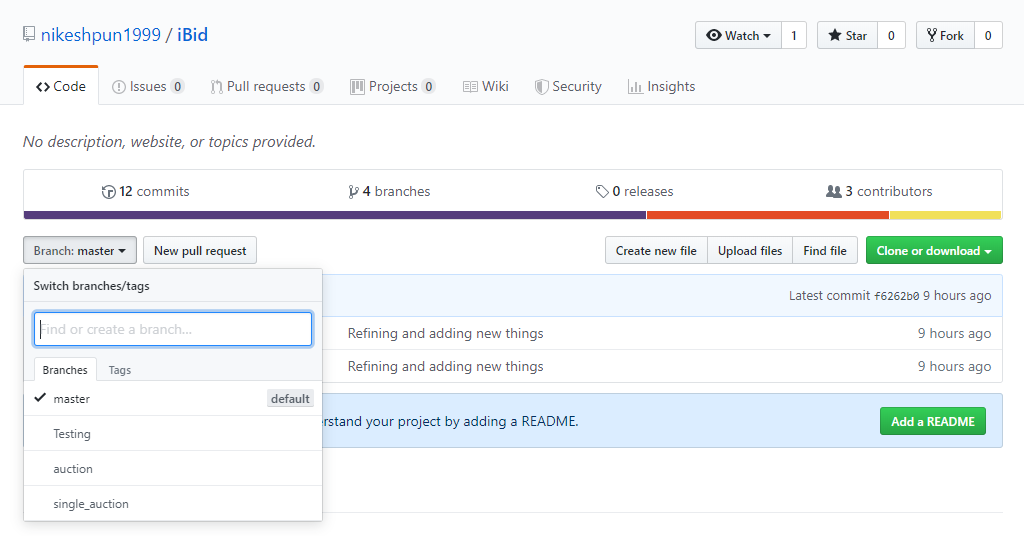
*Command: git merge branch name*

**Git reset:** This command is used to reset the working directory to the previous commit’s state.

*Command: git reset --hard*

**GIT in our System**

Our agile team consists of six members in total. There are two testers, three developers and one scrum master. All the members of our team have the privilege to edit and manage the required code to make the system better. Branches have also been divided according to the roles of the members in the team. The testers work on their own respective branch whereas developers also work on their respective branch. The main folder that GIT keeps track of is called a repository, it is the main branch for the whole system. First of all, the team members need to clone the main repository on their respective computers/laptops in order to start the development process. After cloning the main repository, the members/developers can begin to update the system. If the team member wants to push the work done by him, he needs to commit the work before pushing the change to the respective branch.



The above snippet is the GitHub view page of our current project “iBid”, here we can see the branches made for the respective roles of the team members as well the number of commits to the branch. The number of commits differ for each branch since different work is being done in each branch.

Things don’t always go as planned while working as a team. Many misunderstandings may arise while working as a team. For example, not able to finish the given work at time, lack of communication between the members, repeating the same work by different members as well as managing and dividing the work between the members of the team.

## Benefits of using Version Control

Version Control System has gone through many improvements over the past few years. There are many version control systems, people have their own preferences but the primary benefits that should be expected from a version control system are as follow: (Anon., 2019)

* **Change history of every related file:** Long term history of the changes made by every individual developer to the product. Changes made to the project include updates to the files as well as deletion of files. The main purpose of storing the complete history of every change made is to help in the rollback function or rewinding to a previous version to analyze and compare codes in hopes to find the specific bugs as well as to find the root cause of a problem or error, it is crucial at the time of fixing errors.
* **Branching and merging:** In normal situations without version control technique, team member working in a parallel pace or concurrently is very difficult. The work between the members would collide as well as coexisting codes would be repeated without knowledge. With the help of a feature of Version Control system called “Branching” this problem can be resolved easily. A “Branch” used in a Version Control system keeps numerous streams of work independent from each other while providing the feature to merge the whole project when required. This enables the developers to confirm that the changes on each individual branch do not have any type of conflicts with each other. There are various types of workflows that a team can adopt when they decide how to use the feature of Version control system “Branching and Merging “.
* **Traceability:** Version Control Systems provide the ability to trace each and every change made to a program and connect it to the development of the project as well as can be used to track bugs in the software. The developers can even annotate each change with a short message explaining the motive of the change. The annotated history of the program will benefit in the future while fixing bugs or even adding new features, having the notes will help the developer understand what that piece of code is doing and why it is important. This will help the developer in making the correct and balanced changes in the system.

# Test Driven Development (TDD):

If we look for the meaning of TDD, it is basically a programming practice where the developers bypass the duplication of code by writing new code only when the automated test fails. TDD’s fundamental aim is clearer, bug-free and simple code. The automated tests are run even before starting the actual development of the product and hence, people also refer it as **Test First Development.**

Here, we begin by drafting and establishing tests for single one of the functionalities of the product. This will later validate and specify the outcome of the code. But, since the tests are written before the development process, there is always a higher probability of failure. Therefore, developers need to constantly apply changes to the code i.e. refactoring and continue to do so until the test is passed.

(Anon., 2019)

The general steps to perform a TDD test is as follows:

1. Add a test
2. Run all tests and see if the new one fails
3. Write some code
4. Run tests
5. Refactor code
6. Repeat

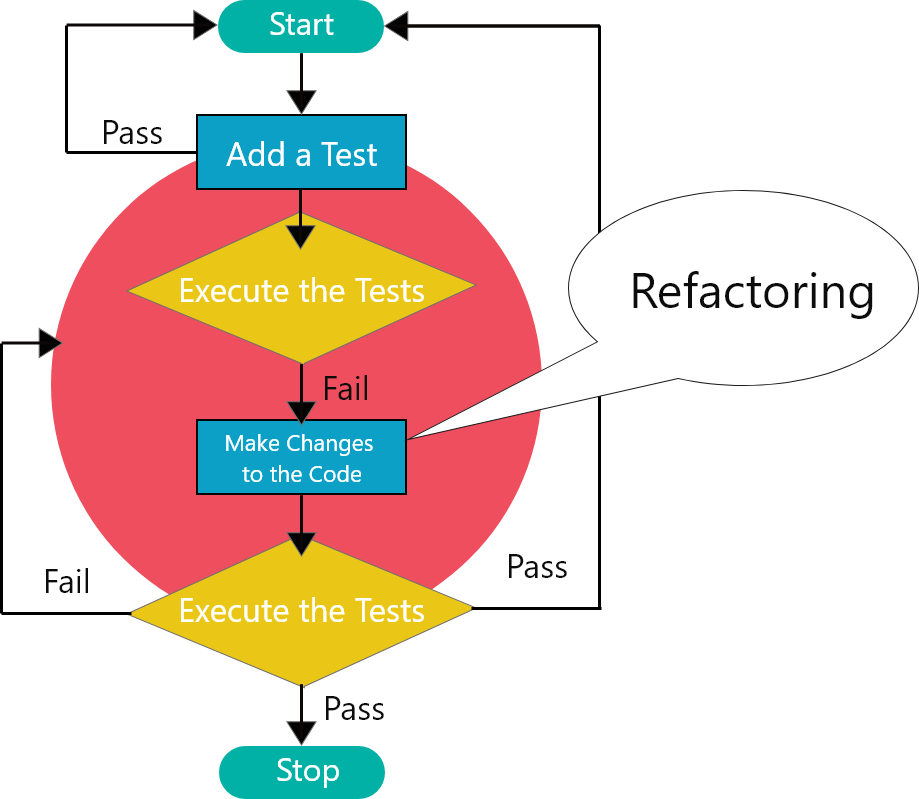


Figure 1: Steps of TDD test

A common misconception about automated tests is that it is time consuming. But in reality, automated tests help the developers along with the team to build the product much more quickly and with a higher quality. The followings are some of the benefits of TDD which might help to clear out the doubts and misconceptions of people.

* It asks us to acknowledge our desired outcome of the code.
* Quick feedback.
* Increases time efficiency via refactoring.
* Less loss of time in the debugger.
* Creation of a thorough specification.
* Encourage evolution in design and comply with varying understanding of the problem.
* Forces radical simplification of code.
* Final unit tests are simple and can be utilized as documentation for the code.
* Fewer bugs and enhanced quality.
* Cost efficient development along with increased productivity of the developers.
* Leads to more flexible, modularized and extensible code which further leads to cleaner interfaces, relaxed coupling, and smaller but highly focused classes.
* Grants ability to painlessly update the software in case of new business requirements or other unexpected variables.

(Anon., 2019)

For TDD also, we have executed Unit testing of all the functionalities of the system. The unit test was carried out in **Node.js** with the help of code editor **Visual Studio Code.**

The following steps were pursued in the process of Unit testing:

1. At first, the whole system was divided into small functional units.
2. Next up, state-based-testing was chosen over the interaction-based testing for our system.
3. The data required to run the test was analyzed.
4. Test case for each of the functionalities were written.
5. Thus prepared test scenarios were then run.
6. After running the test, the behavior of the program was then observed.
7. Final result of the test was compared with the expected result of the test.
8. In case of failed tests, the test was re-written and run until it passed.

(Anon., 2019)

An image illustrating the procedure followed for unit testing in our project is shown below:



Figure 2: Unit test case of a functionality



Figure 3: Result of the Unit test cases

# Behavior Driven Development (BDD):

As for the definition, BDD is a union and refinement of processes of Test Driven Development (TDD) and Acceptance Test Driven Development (ATDD) which utilizes samples in order to represent the characteristics/behavior of the system that are noted in a readable and recognizable language for all the people associated with development. One of the target of BDD is to offer a shared procedure and tools to promote communication among business analysts and software developers during the development phase in order to hand over a product carrying business value. Next up, it looks up into what a system should do and not do on its implementation method. It also sheds light on the visibility and readability. Last but not the least, it ensure that the product not only functions well but also meets the expectations of the client. (Anon., 2019)

Furthermore, it expands ATDD and TDD with accompanying strategies:

* Apply the "Five Why's" rule to each proposed client story, so its motivation is obviously identified with business results.
* To limit waste, implement thinking “from the outside in”, which means implementing just those practices that contribute more precisely to the same business results.
* Depict practices in a solitary documentation which is directly available to domain specialists, developers and testers, in order to improve communication.
* Apply these procedures right down to the most reduced degrees of deliberation of the product, giving specific consideration to the dispersion of behavior, with the goal that evolution stays modest.

## BDD Workflow:

The workflow if BDD can be essentially divided into three main parts:

1. **Define:** At first, the domain specialist or the business analyst drafts behaviors into a feature file which is typically written as standard sentences and is presents it to the team.
2. **Create:** With respect to the behaviors listed in the feature file, developers prepare a code known as “glue code”, which pictures behaviors to certain administration. An example is as follows:

***Given*** I am the user

***When*** I enter my login credentials in the login page

***Then*** I should be able to access the website

1. **Verify:** Next, the testers compare the glue code against the feature file in order to ensure that the application code correlates to the behaviors. This process is executed with the help of a BDD tool such as HipTest, Cucumber, etc. For the output, the feature file is compiled and pulled off to confirm the functionality.

Moving on to the advantages of using BDD, here are some of the expected benefits of it:

* It grants more clear-cut guidance on arranging the discussion between domain specialists, developers and testers.
* Documentations starting in the BDD method, specifically the **given-when-then** canvas, are nearer to ordinary day to day language and have a trivial learning curve contrasted with those of tools, for example, Fit/FitNesse.
* Tools focusing on a BDD approach for the most part bear the automatic generation of technical and end user documentation from BDD “specifications”.

(Anon., 2019)

## Tool used for BDD:

Out the varieties of tools available in the market, to name a few, HipTest, Easy B, J Behave, etc., we have used **Cuke Test**. The tool is most importantly free and it aids in drafting tests that are simple to understand for people even without Technical knowledge. Additionally, it helps to keep the test case scenarios and the requirement statement in the same place. Similarly, it is confined only for Web-driven automation, so it was also the selling point for us. The whole process of tracking the implementation is also easy with it as it displays the behavior sets that have been implemented and in working status.

## BDD Code

Some of the BDD test cases used to test the system are as follows:

* The First case is for the scenario of a user login. When the user inputs the required credentials with the registered username & password, then he must be able to enter the system.



Figure 4: BDD test case for User Login

* The second BDD case scenario is to update the username of the registered user. The bellow picture shows the success of updating a username.

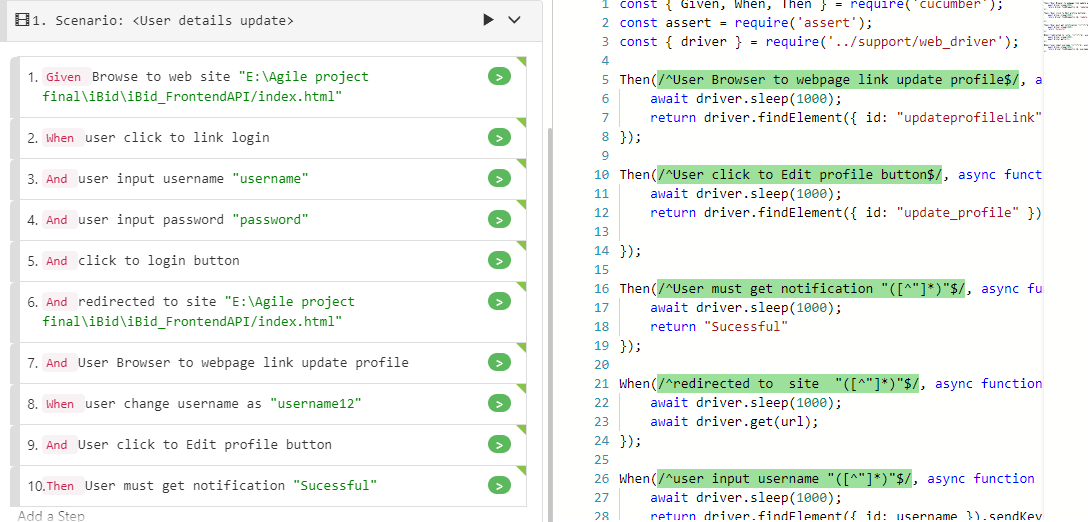


Figure 5: BDD test case for updating user

* The third BDD case scenario is of the admin login. The admin should be able to log in to the admin dashboard.

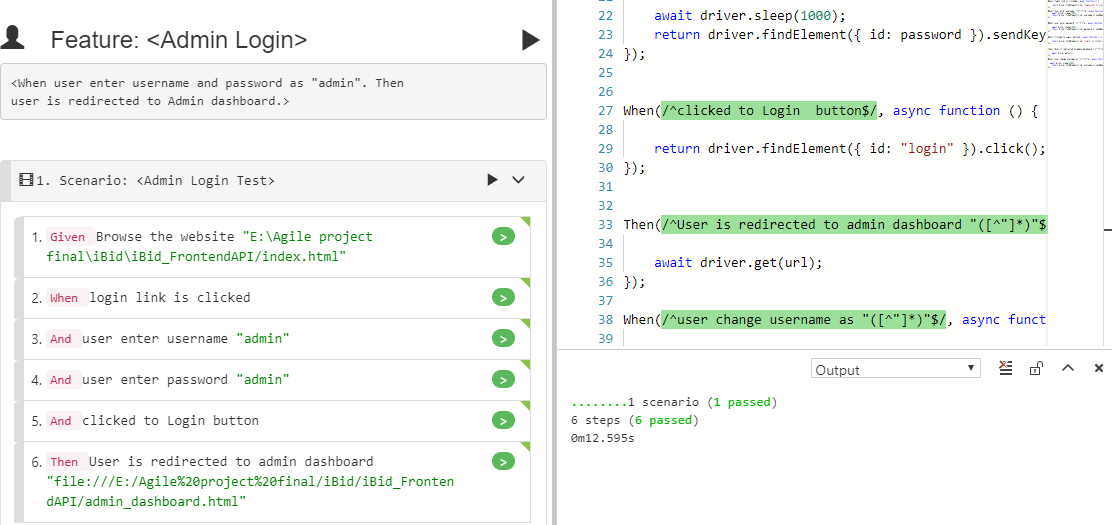


Figure 6: BDD test case for Admin Login.

# References

Anon., 2018. *dzone.com.* [Online]   
Available at: https://dzone.com/articles/top-20-git-commands-with-examples  
[Accessed 14 October 2019].

Anon., 2019. *Behavior Driven Development (BDD).* [Online]   
Available at: https://www.agilealliance.org/glossary/bdd/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa\_book~'aa\_event\_session~'aa\_experience\_report~'aa\_glossary~'aa\_research\_paper~'aa\_video)~tags~(~'bdd))~searchTerm~'~sort~false~sortDirection~'asc~page~1)

Anon., 2019. *Benefits of Test Driven Development.* [Online]   
Available at: https://dzone.com/articles/20-benefits-of-test-driven-development

Anon., 2019. *github.com.* [Online]   
Available at: https://github.com  
[Accessed 11 October 2019].

Anon., 2019. *git-scm.com/.* [Online]   
Available at: https://git-scm.com/  
[Accessed 11 October 2019].

Anon., 2019. *How to Write Unit Tests.* [Online]   
Available at: wikihow.com/Write-Unit-Tests

Anon., 2019. *Introduction to Behavior Driven Development.* [Online]   
Available at: https://www.tutorialspoint.com/behavior\_driven\_development/behavior\_driven\_development\_introduction.htm

Anon., 2019. *seguetech.com.* [Online]   
Available at: https://www.seguetech.com/a-review-of-software-version-control-systems-benefits-and-why-it-matters/  
[Accessed 15 October 2019].

Anon., 2019. *What is Test Driven Development (TDD).* [Online]   
Available at: https://www.guru99.com/test-driven-development.html

Anon., 2019. *what-is-version-control.* [Online]   
Available at: https://www.atlassian.com/git/tutorials/what-is-version-control

Anon., 2019. *www.getfilecloud.com.* [Online]   
Available at: https://www.getfilecloud.com/blog/2015/02/top-5-open-source-version-control-tools-for-system-admins/#.XcFSLugza00  
[Accessed 15 October 2019].