CYPRUS INTERNATIONAL UNIVERSITY

SOFTWARE ENGINEERING

GIT AND GITHUB

FILE READER PROJECT

Submitted by:

OMBILI SANTOS **20164910**

MUKHTAR ILIYASU GARBA **20154439**

EUGENE KOFI OFOSU **20156247**

27th April, 2018

Course Lecturer: Asst. Prof. Dr. Çağın KAZIMOĞLU

Table of Contents

Overview

The purpose of the project is to develop a File Reading application through Git and GitHub using the C# programming language and by making use of the Crystal Agile software process model. The team collaboration of this project is to be done using GitHub. A repository, which is the main file system of the project will be created by one account user and distributed to other team member accounts on GitHub. Although team members will make use of Git and GitHub tools to collaborate anytime and anywhere, team members are encouraged to have face-to-face meetings. The project is expected to be completed in a period of one month and no extensions will be granted. Throughout the development process, at least 3 versions are to be developed.

What are Git and GitHub?

Git is a **Version Control System (VCS)** that maintains and tracks the changes of the software system being developed. Although a **VCS** is divided into two types, the members will make use of the **Distributed Version Control System.** This will allow team members to have their own local copy of the remote repository. Should anything go wrong with the remote repository, team members will have their own local copies of the repository to work from and can restore the remote repository from these copies.

GitHub is a web based application that offers a visual interface to the team's repository. The online repository is referred to the remote repository. The biggest advantage of GitHub is the fact that it makes collaboration easier as team members can collaborate anytime and anywhere provided they are connected to the internet.

Therefore, Git is not GitHub and vice versa. **Git** is a Version Control System while **GitHub** is an online repository system. **Git** can be used with other online repository systems and **GitHub** can be used with other Version Control Systems.

Project Scope and objectives

The objective of the project is to develop the File Reading Application in versions in a period of one month. By using Git, the team members are able to develop the application in versions. This will assist team members to keep track of the changes of the applications through versions. Each version should be different from the other. Each team will have their own working directory which is a copy of the remote repository which will be the master branch.

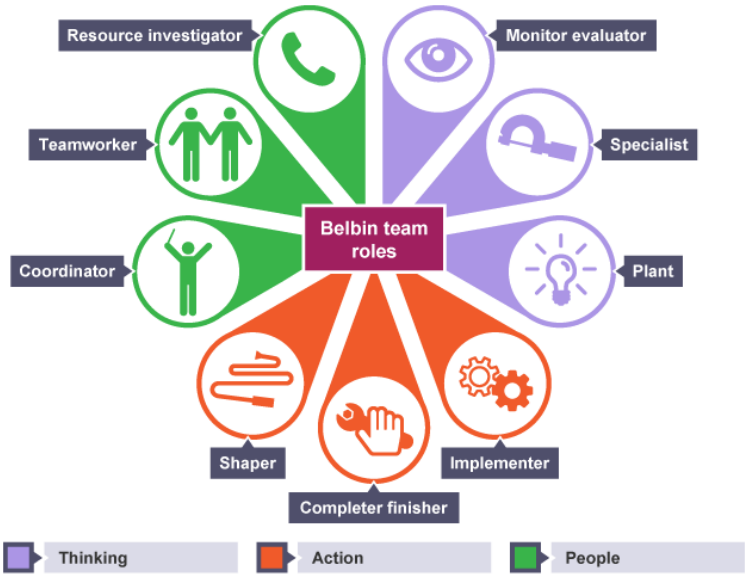
Assignment of Roles and Responsibilities

The team consists of 3 (three) group members with each team member having a different role:

1. Team leader (Senior Designer/Programmer): The team leader is responsible for the principal design of the application and programming. Other responsibilities is to ensure that other group members meet their contributions to the project on time and the right tools have been utilized to develop the project.

2. Business class designer (developer and tester): The business class designer will be responsible for ensuring that the design laid out by the team leader meets the requirements of the user and the programming codes do exactly what they were supposed to do. He/she will also be able to add extra functionality and features to the application. During each development process, the application will be tested.

3. User (System acceptance tester and document): The user is responsible with testing the application throughout each process and will also be commenting and documenting on the whole development process. By making use of the user experiences as well as the lead programmer and developer methods used to develop the application, the user will have to document all these events.

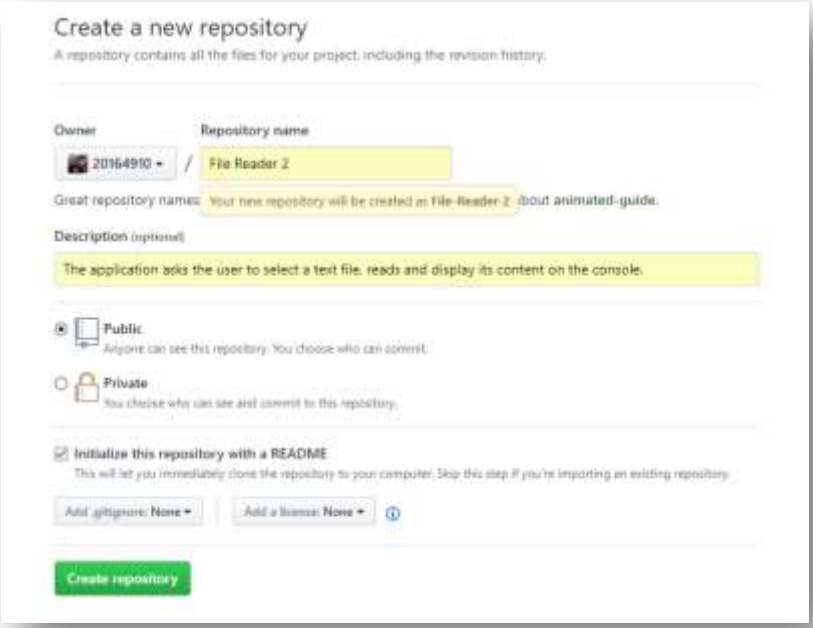


Design

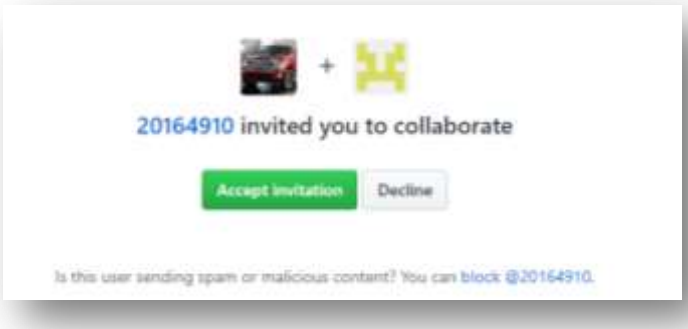
The development team decided to make use of **Microsoft Visual Studios** as their main development application tool. The developers were not only limited to Git's tool Git Bash to push their commits nor pull their commits. With the use of the Visual Studio **GitHub Extension,** developers had a choice to make use of this extension. But In order to develop versions, Git Bash with its commands was used. For this project, 4 versions were released.

Development Processes:

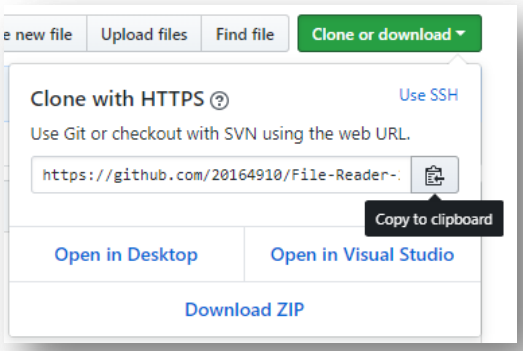
1. Creation of the repository: After Signing up on GitHub, the team leader creates the repository

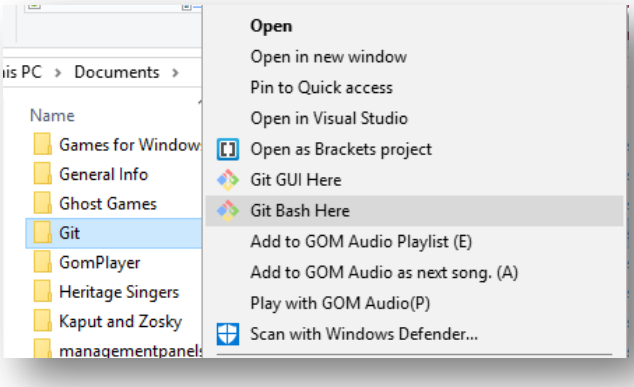


2. Invite: The team leader of the project invited other team members



3.1 Creating Branches: Each member creates their own local branches by copying the link of the remote repository on GitHub. User then right-click at the clones repository and selects "Git Bash Here" to open Git Bash.





The Git Bash application will have to be configured with GitHub user's name and e-mail before cloning of the repository. This prevents prompting the user to enter these requirements if a push command to the remote repository is required.

Commands used to configure: **git config --global user.name "usersname"**

git config --global user.email "usersemail"

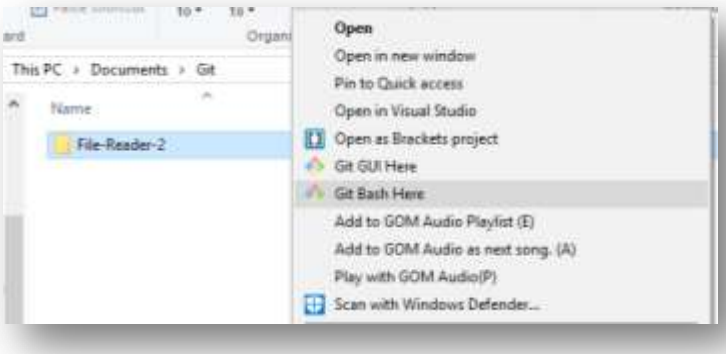
```
OYDS@THE-MEAN-MACHINE MINGW64 ~/Documents/Git
$ git config --global user.name 20164910

OYDS@THE-MEAN-MACHINE MINGW64 ~/Documents/Git
$ git config --global user.email oyds2014@icloud.com
```

Command used to clone: **git clone "url link of the repository"**

```
OYDS@THE-MEAN-MACHINE MINGW64 ~/Documents/Git (master)
$ git clone https://github.com/20164910/File-Reader-2.git
Cloning into 'File-Reader-2'...
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

3.2 Using Git to create local repositories and branches: Each user including the team leader should have their own local repositories. Each local repository should have a master branch and an extra branch where the user can make any commits of their choice at will.

After cloning the remote repository, the user then right-click at the clones repository and selects "Git Bash Here" to open and be active in the master branch



Code used to create branch: git branch "name-of-branch"

```
OYDS@THE-MEAN-MACHINE MINGW64 ~/Documents/Git/File-Reader-2 (master)
$ git branch 20164910-Own-Branch
```

To confirm if branch was created, migrate to the newly created branch:
git checkout "name-of-branch"

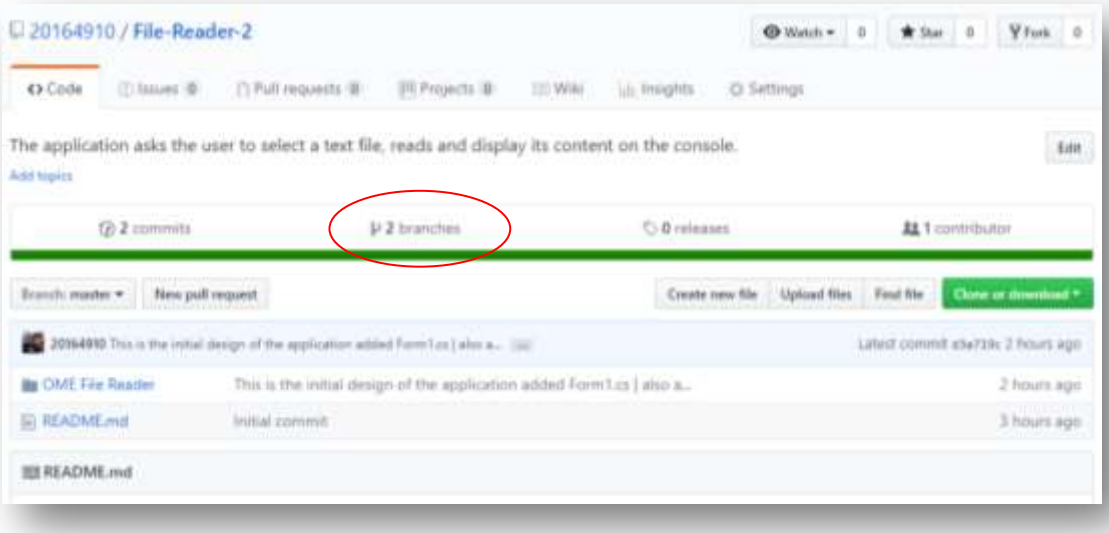
```
OYDS@THE-MEAN-MACHINE MINGW64 ~/Documents/Git/File-Reader-2 (master)
$ git checkout 20164910-Own-Branch
Switched to branch '20164910-Own-Branch'

OYDS@THE-MEAN-MACHINE MINGW64 ~/Documents/Git/File-Reader-2 (20164910-Own-Branch
)
$
```

4. Pushing the local own branch: After creating the user's own local repository, the user will still have to push it to the remote repository

Code used to push user's own branch to remote: git push –u origin "name-of-branch"

```
OYDS@THE-MEAN-MACHINE MINGW64 ~/Documents/Git/File-Reader-2 (20164910-Own-Branch)
$ git push -u origin 20164910-Own-Branch
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/20164910/File-Reader-2.git
 * [new branch]      20164910-Own-Branch -> 20164910-Own-Branch
Branch '20164910-Own-Branch' set up to track remote branch '20164910-Own-Branch'
from 'origin'.
```

Now there are two branches:



5. Committing Changes: After every modification on the user's own branch, these modifications should be added and then committed to their own branches. Git status command is used to display the status of the branch. It can be used to check if its ahead or behind the master branch.

Command used to check status: git status

Command used to add changes to the user's own local branch: git add .

```
OYDS@THE-MEAN-MACHINE MINGW64 ~/Documents/Git/File-Reader-2 (master)
$ git status
On branch master
Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   OME File Reader/.vs/OME File Reader/v15/.suo
        modified:   OME File Reader/.vs/OME File Reader/v15/Server/sqlite3/storage.ide
        deleted:    OME File Reader/.vs/OME File Reader/v15/Server/sqlite3/storage.ide-shm
        deleted:    OME File Reader/.vs/OME File Reader/v15/Server/sqlite3/storage.ide-wal
        modified:   OME File Reader/bin/Debug/OME File Reader.exe
        modified:   OME File Reader/bin/Debug/OME File Reader.pdb
        modified:   OME File Reader/obj/Debug/OME File Reader.csproj.FileListAbsolute.txt
        modified:   OME File Reader/obj/Debug/OME File Reader.exe
        modified:   OME File Reader/obj/Debug/OME File Reader.pdb

no changes added to commit (use "git add" and/or "git commit -a")

OYDS@THE-MEAN-MACHINE MINGW64 ~/Documents/Git/File-Reader-2 (master)
$ git add .
```

Command used to commit the changes comments: git commit –m "place comments here"

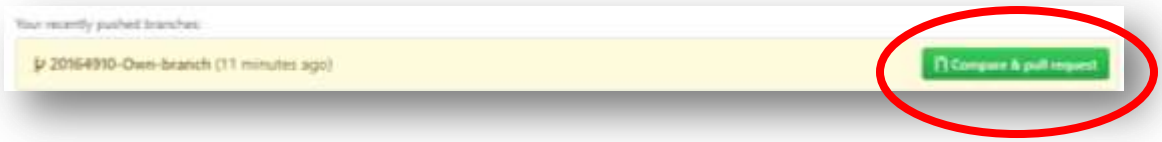
```
OYDS@THE-MEAN-MACHINE MINGW64 ~/Documents/Git/File-Reader-2 (20164910-Own-Branch
)
$ git commit -m "Changed the design for the new version and added some new featu
res"
[20164910-Own-Branch 411ad13] Changed the design for the new version and added s
ome new features
 14 files changed, 1381 insertions(+), 88 deletions(-)
 rewrite OME File Reader/.vs/OME File Reader/v15/.suo (85%)
 rewrite OME File Reader/bin/Debug/OME File Reader.exe (84%)
 rewrite OME File Reader/obj/Debug/OME File Reader.exe (84%)
 create mode 100644 OME File Reader/obj/Debug/OME_File_Reader.Form1.resources
```

6. Pushing commits: The commits will then be added to the remote user's branch

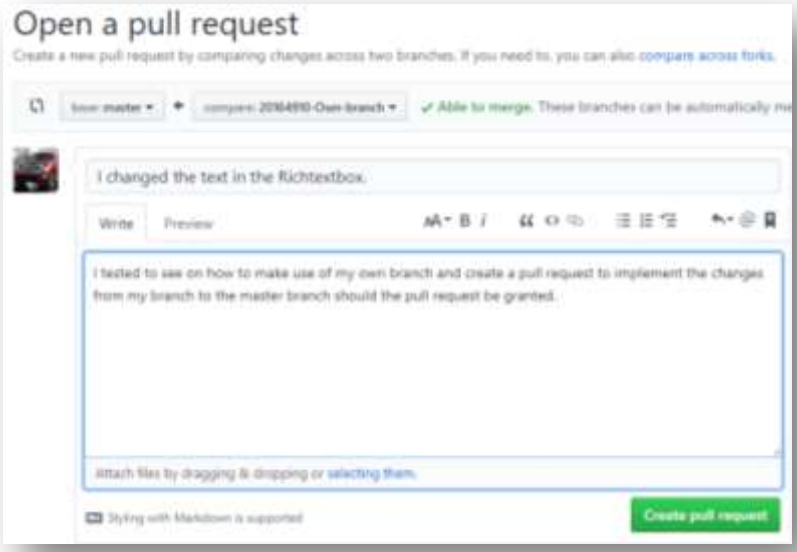
Command used to push the commits to remote user's branch:
git push –u origin 20164910-Own-Branch

```
OYDS@THE-MEAN-MACHINE MINGW64 ~/Documents/Git/File-Reader-2 (20164910-Own-Branch
)
$ git push -u origin 20164910-Own-Branch
Counting objects: 24, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (18/18), done.
Writing objects: 100% (24/24), 59.54 KiB | 1.75 MiB/s, done.
Total 24 (delta 9), reused 0 (delta 0)
remote: Resolving deltas: 100% (9/9), completed with 8 local objects.
To https://github.com/20164910/File-Reader-2.git
   cd64be6..411ad13  20164910-Own-Branch -> 20164910-Own-Branch
Branch '20164910-Own-Branch' set up to track remote branch '20164910-Own-Branch'
 from 'origin'.
```

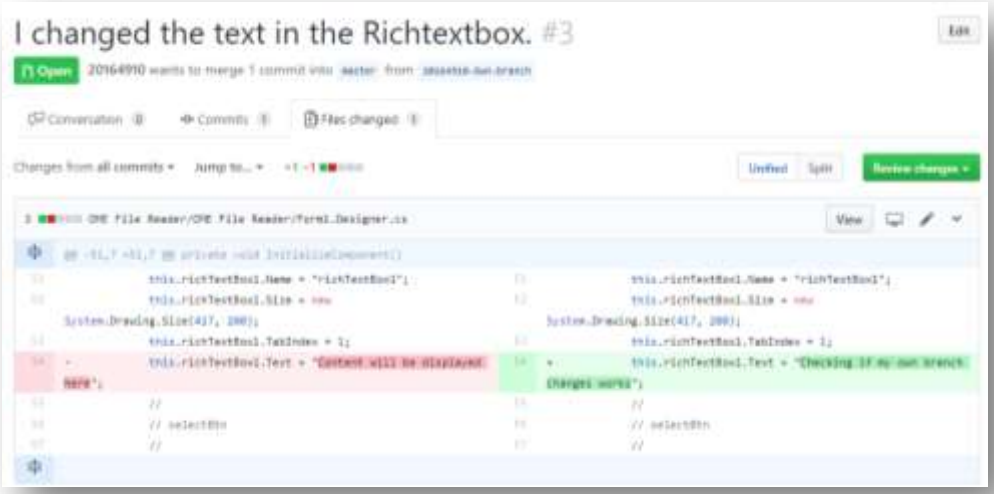
7. Creating a Pull request: After a commit has been submitted on the remote branch, a pull request will be created on GitHub to ask other team members to have a look at the commitments and if they are ok to be added to the master branch.



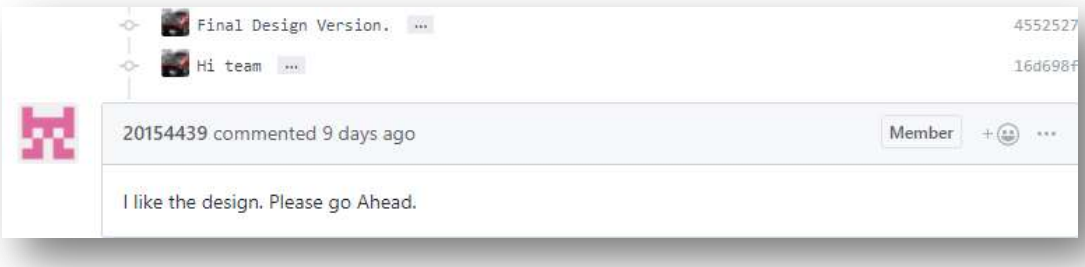
User will then insert comments for the commits made for other team members to get an idea of what the commits are about.



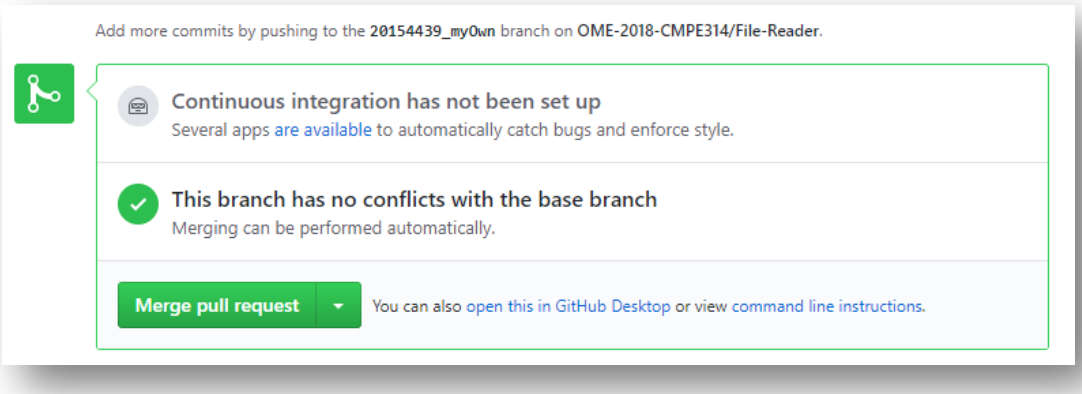
Changes before are highlighted in red, changes after are highlighted in green while unchanged codes are not highlighted.



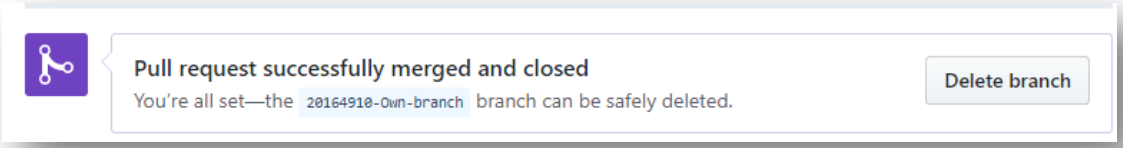
Once the other team members approve of the changes, merging can take place



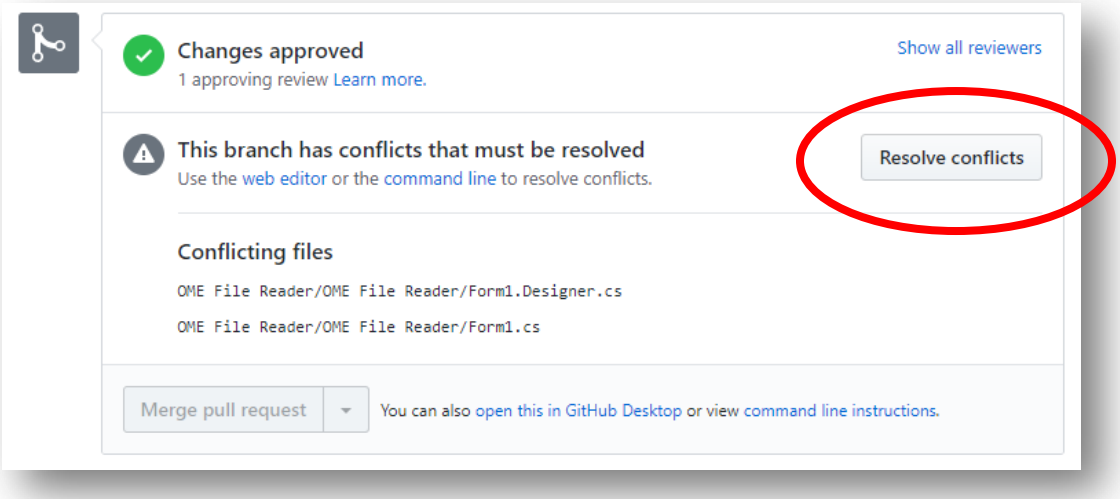
8. Merging a successful pull request: Before merge, a pull request icon is green

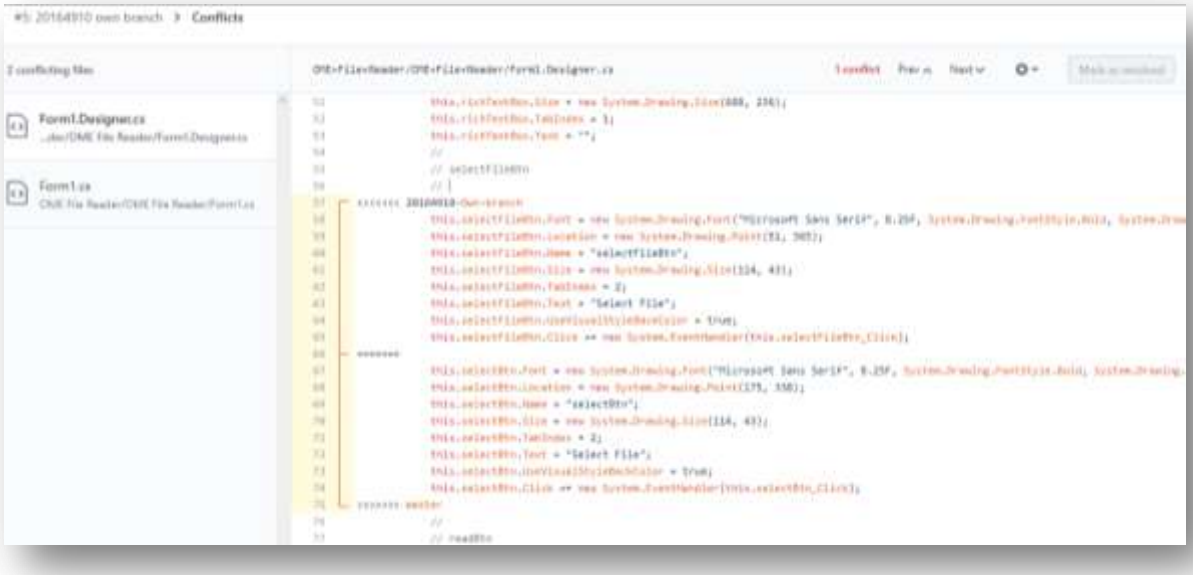


After a merge, a pull request turns purple



9. Correcting pull requests: At times, a pull request may not be automatically merged because it contains conflicts. Conflicts are commits that the GitHub component cannot automatically merge with the master branch as it is confused on which commits to choose; keep the old with the new or delete the old and replace with new. It is therefore up to the user to manually resolve these conflicts.





After these conflicts have been resolved, the commit merge button will be clicked

Implementation

Creating Versions

Version implementations are referred to as tags. They are implemented within the master local branch first and then uploaded to the remote master branch on GitHub

Command used to create tag: git tag "tag-name"

```
OYDS@THE-MEAN-MACHINE MINGW64 ~/Documents/Git/File-Reader-2 (master)
$ git tag v1.0
```

Confirmation of the creation of the version: git tag

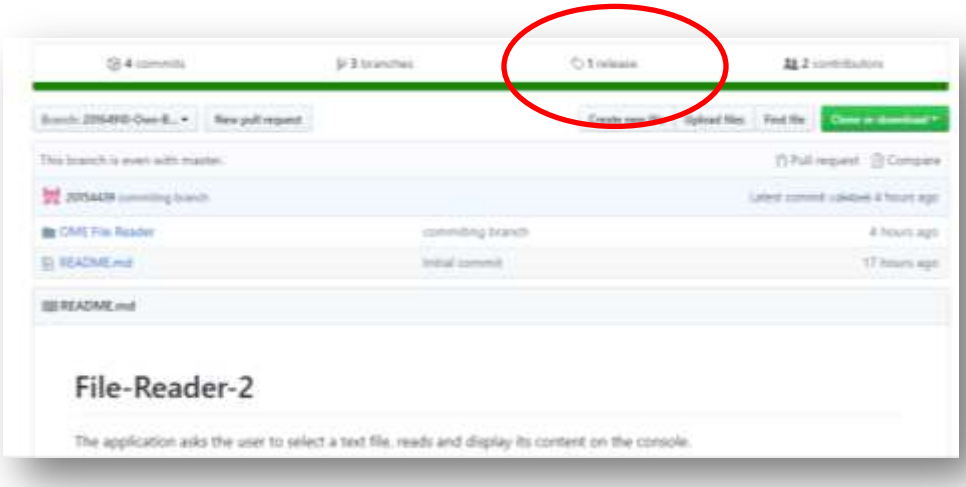
```
OYDS@THE-MEAN-MACHINE MINGW64 ~/Documents/Git/File-Reader-2 (master)
$ git tag
v1.0
```

The version is then pushed to the remote master branch on GitHub

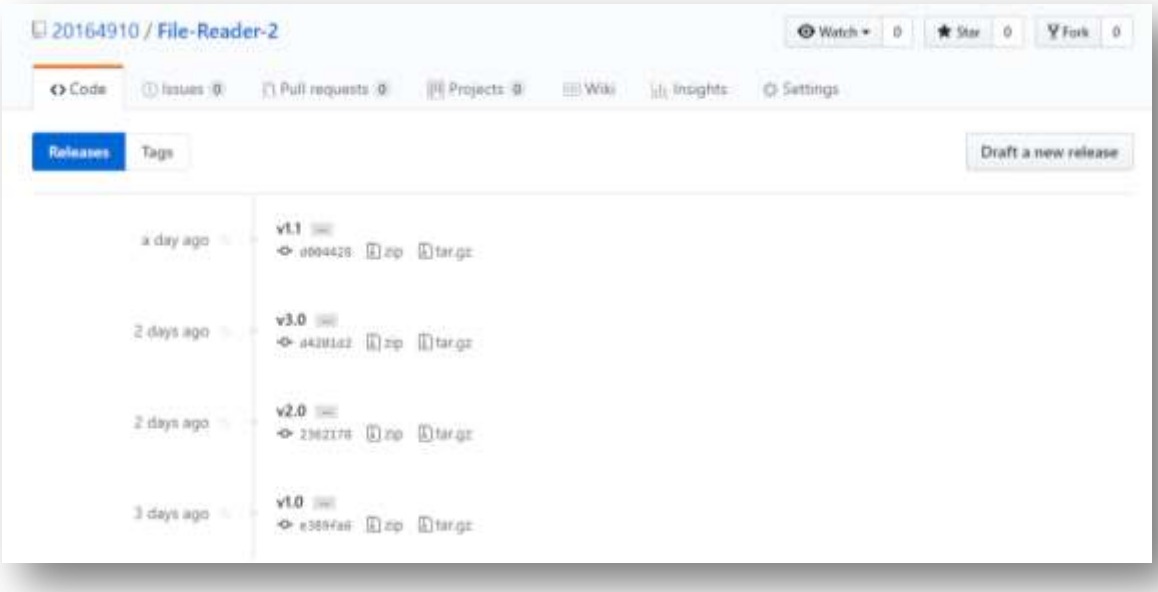
Command used: git push origin v1.0

```
OYDS@THE-MEAN-MACHINE MINGW64 ~/Documents/Git/File-Reader-2 (master)
$ git push origin v1.0
Counting objects: 30, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (22/22), done.
Writing objects: 100% (30/30), 8.46 KiB | 509.00 KiB/s, done.
Total 30 (delta 13), reused 0 (delta 0)
remote: Resolving deltas: 100% (13/13), completed with 9 local objects.
To https://github.com/20164910/File-Reader-2.git
 * [new tag]         v1.0 -> v1.0
```

Now the version has been added



After all versions have been created



Different versions and their visible changes

The File reader was designed to Read text from certain files only. The versions got better and better giving an ultimate perfect user experience .

v1.0

The initial version was quite simple with minor functionalities and features.

User Functionalities:

- When user hovers over the select file, a tip balloon will appear prompting user to click and select required file to be read by the application.
- User will be able to save a brand new text file.
- User will also be able to clear the text currently in the view window

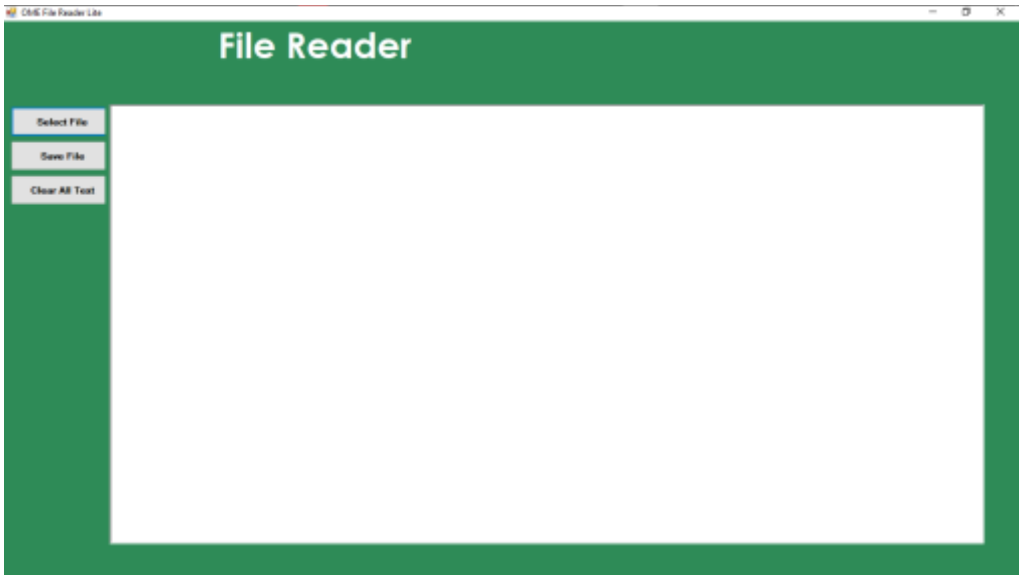


Fig 1.0 v1.0

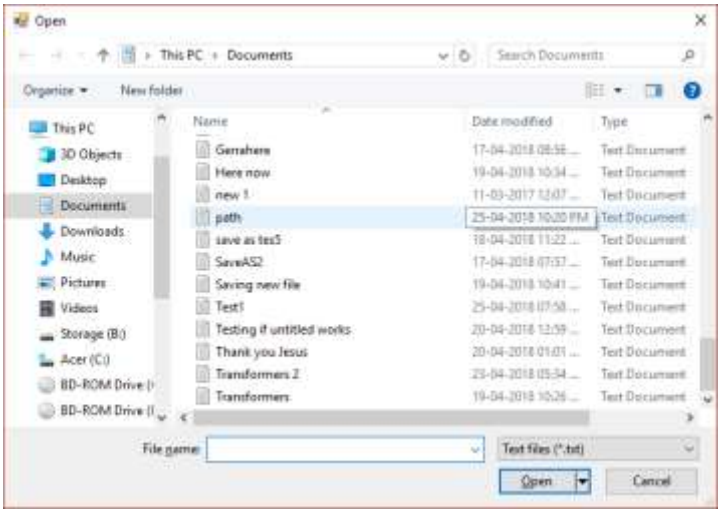


Fig 1.1 Open file dialog when Select File button is clicked

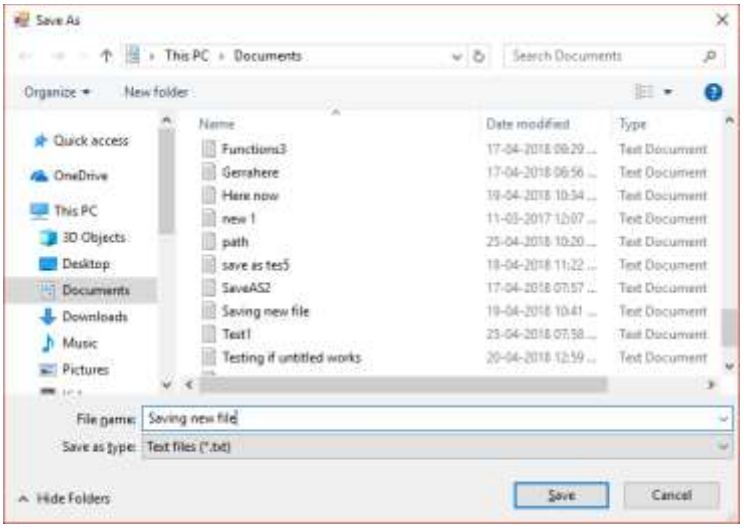


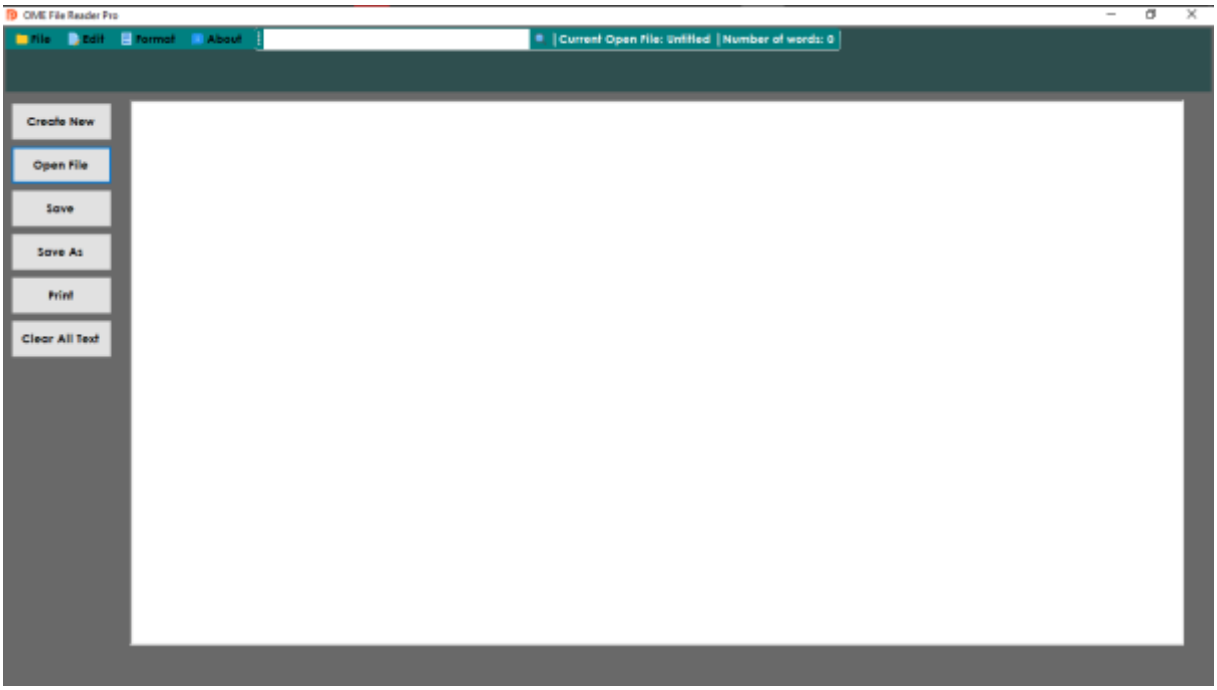
Fig 1.2 Save file dialog when Save File button is clicked

Fig 1.3 Before Clear All Text button click and after the button is clicked

v1.1 was also created after the programmers noticed a bug caused the function codes not to work in v1.0

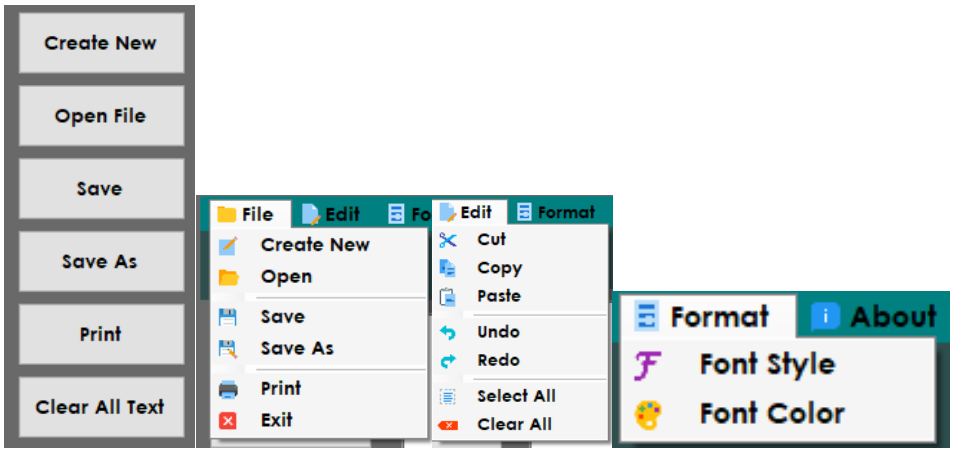
v2.0

A Brand design with new features were introduced in the second version



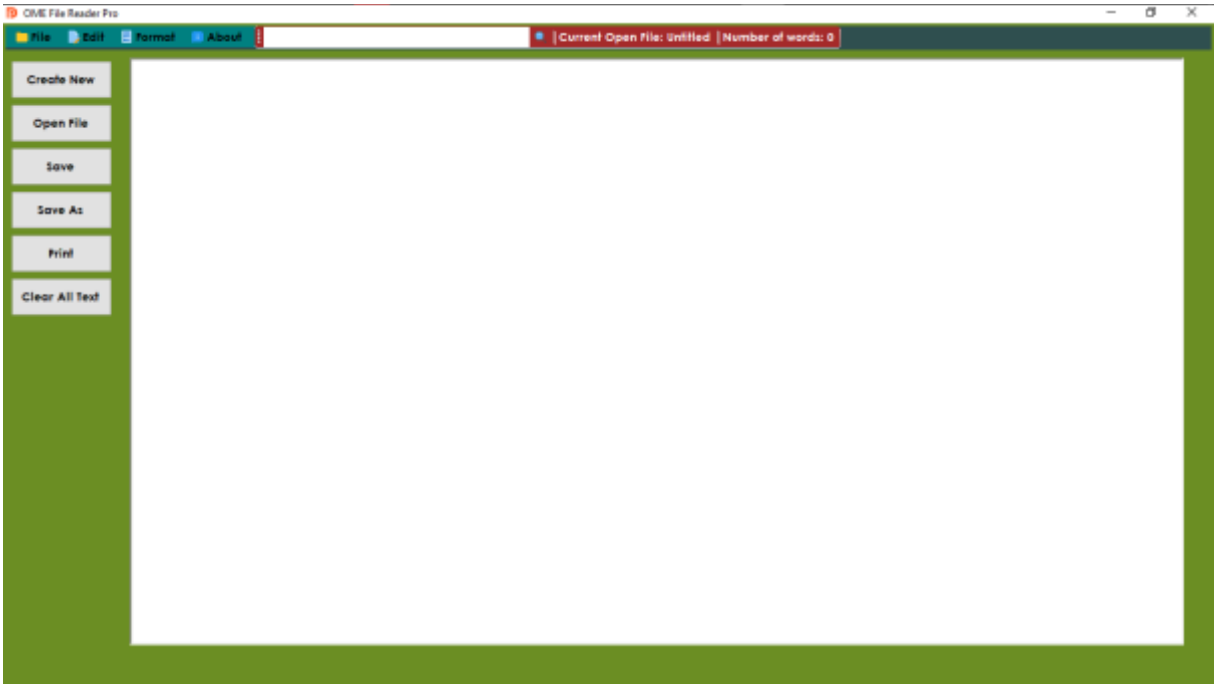
Features

- **Menu strip**
- **Extra buttons and functionalities**
- **User can now the current open document and number of words in the text file**



v3.0

New features and functionalities were added. New features are the colour change at the menu strip (red) and main window background.



The extra functionality is the addition of the context menu strip which enables the user to make use of commands when the right-click mouse button is clicked. Word counter bug was removed to show actual number of words in the text file.