OSS_Term_Project_18

1. Project introduction

- GitHub Repository address: https://github.com/cokestrawberry/OSS_Term_Project_18
- Team Member: 김서해, 김승진, 임종승, 허지범
- License: MIT License

Repository for Chungang UNIV. Dept. of SW.
OSS term project team 18

FMOS is an open source file manager that supports git features. The use of git, which had to be accessed with the existing CLI, can be used simultaneously with file search in the GUI environment.

FMOS started by referring to <WIFIle>, an open source project for File Manager.
<WIFIle> Github address(https://github.com/reallyrehan/flask-fileexplorer)

A brief summary of the FMOS open source project is as follows:

License	MIT
language	python
Platform	windows
Framework	Flask/Bootstrap 4

FMOS can use git functions simultaneously with file search in the GUI environment, and at the same time, it can help the overall understanding of git in the process of modifying/supplementing them.

2. How to Start

1) Clone this project

```
git clone https://github.com/cokestrawberry/OSS_Term_Project_18.git
```

2) Install library(in flask-fileexplorer folder)

- If you are not in flask-fileexplorer folder, do 'cd' command

```
cd flask-fileexplorer
pip install -r requirements.txt
```

3) Run file manager program

```
python setup.py
```

- If you enter this command, you can get a IPv4 address as return. That is your file manager address. Do ctrl + click or copy that address, paste on web-browser and enter.
- Alternatively, it can be accessed through the loopback address (127.0.0.1:80).
- Then, you can see your C: directory. You can browse by click folder or add local address of your directory behind the address.

4) Set Downloads/Documents folder (Optional, Not neccessaty)

- You will have to configure the config.json file with your paths

```
"Favorites":    ["C://Users//Administrator//Documents","C://Users//Administrator//Downloads"],
```

3. How to use and Feature that need to be implemented

With name of command on buttons, you can intuitively know how to use.

- git init
  - ☞ If there is no '.git repository' in the current folder, that is, if the folder is not under version control with git, the git init menu is displayed on the left.
  - ☞ On left side, you can see 'git init' button.
  - ☞ If you press that, .git will be generated.
- git add
  - ☞ With files which is untracked, there is U sign on icon's right-bottom side.
  - ☞ Below the icon, there is 'git add' button.
  - ☞ If you press that, 'git add' command will excute and at the left side bar(for status), staged list will be shown.
- git restore
  - ☞ If the file is in a modified or staged state, a 'git restore' button is provided.
  - ☞ When you click the button, 'git restore --staged' or 'git restore' will be executed depending on the state of the file.
    - ◆ 'git restore --staged': Executes when a file is uploaded to the staging area.
    - ◆ 'git restore': The file is not uploaded to the staging area and is executed in a modified state.
- git commit
  - ☞ Above the left side bar, you can do commit if there is any staged files.
  - ☞ When you click commit button, text box for commit message will be genereted.
  - ☞ After fill in it with your commit message, press the enter key, or press the commit button to commit.
- git mv
  - ☞ For easy version control, the mv menu is only available when file is committed or unmodified state.
  - ☞ Clicking the button creates a text box where you can enter the name to be modified.
  - ☞ Enter the name to be modified, including the extension, in the text box and press the Enter key or click the renaming button to execute the git mv command.
- git rm
  - ☞ 'git rm' deletes files from both git and working directory, 'git rm --cached' deletes files only from git and makes them untracked. So, a separate menu is provided.
  - ☞ 'git rm --cached': It works by pressing the Untracking button.
  - ☞ 'git rm': It works by pressing the Delete button.
- git branch
  - ☞ Create / Delete / Rename / Checkout / Merge are provided as Branch Menu under Git Menu.
  - ☞ The above menus provide a text box to enter a new branch name or a list modal to select an existing branch.
  - ☞ Error Message:

    - ☞ For Create / Delete / Rename / Checkout, the error messages are provided by the modal window itself.
    - ☞ In the case of Merge, the conflict error message informs you of the conflict file as an alert.
- git clone
  - ☞ public/private repository clone
  - ☞ You can designate public or private with the radio button.
  - ☞ In the case of private, the user's id and token are stored and used in the ./flask-fileexplorer/config.json file.
- git history(git log)
  - ☞ print out the git commit's log
  - ☞ The history basically includes the workflow of the current branch.
  - ☞ Each commit object in the graph includes its author name and message.
  - ☞ If a user chooses a commit object, then it provides the detailed information about the commit

4. Contribution guidelines

If you want to contribute to FMOS, be sure to review the Google Python Style Guide. This project adheres to FMOS's code of conduct. By participating, you are expected to uphold this code.
Google Python Style Guide : https://github.com/google/styleguide/blob/gh-pages/pyguide.md
FMOS's code of conduct :
https://github.com/cokestrawberry/OSS_Term_Project_18/blob/main/CODE_OF_CONDUCT.md

5. Implementation

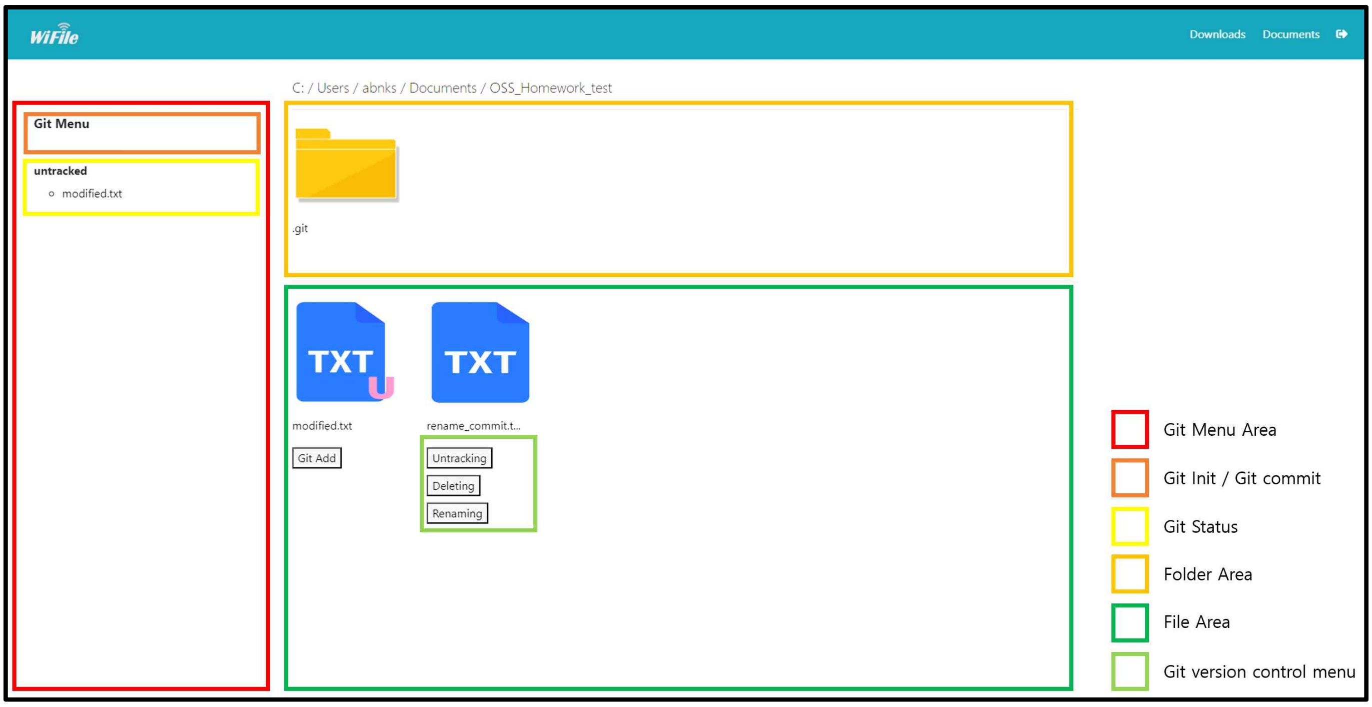
- file system analysis : <WIFIle> (https://github.com/reallyrehan/flask-fileexplorer)
  - ▪ features: Python Web App.
    - ☞ Language: Python
    - ☞ Platform: Windows / Mac / Linux
      => We only developed and tested Windows in consideration of the development environment of our team members.
    - ☞ Frameworks: Flask / Bootstrap4
      => Flask: run the server, serve up the html pages
      => Bootstrap4: show the folders and the files
  - ▪ How to change
    - ☞ icon:
      - ◆ file icons -> ./static/files_icon/*
      - ◆ folder icon -> ./static/folder5.png
    - ☞ html:
      - ◆ main page -> ./templates/home.html
    - ☞ main logic: setup.py
      - ◆ Responding to route movement, event invocation, etc. can be composed of the following template.

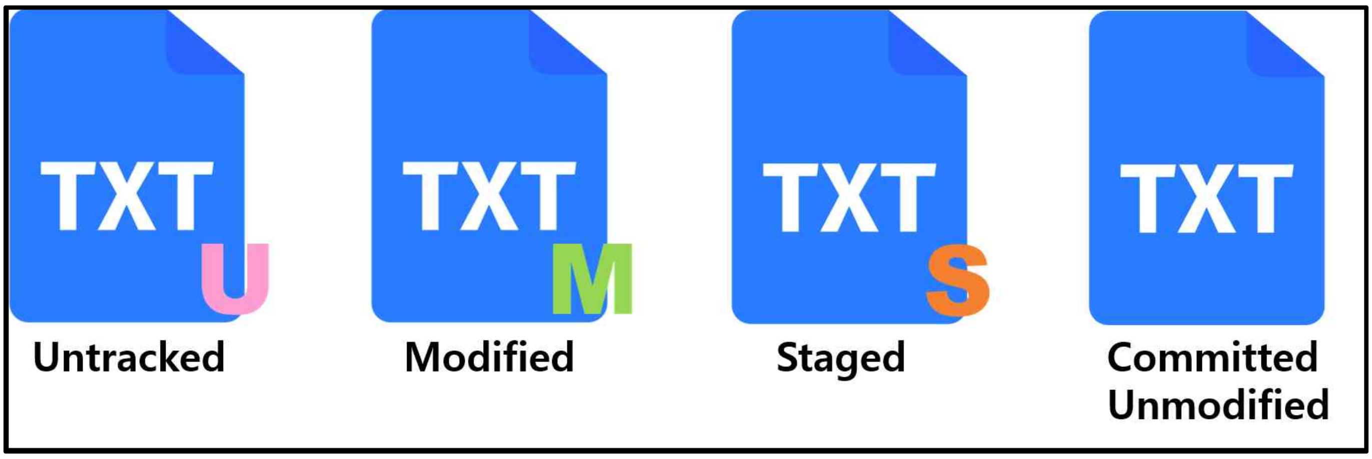
```
@app.route('/route/', methods=['POST'])
@app.route('/route/<path:var>', methods=['POST'])
def function_name(var=""):
    ~~~

    return render_template(~~~)
```

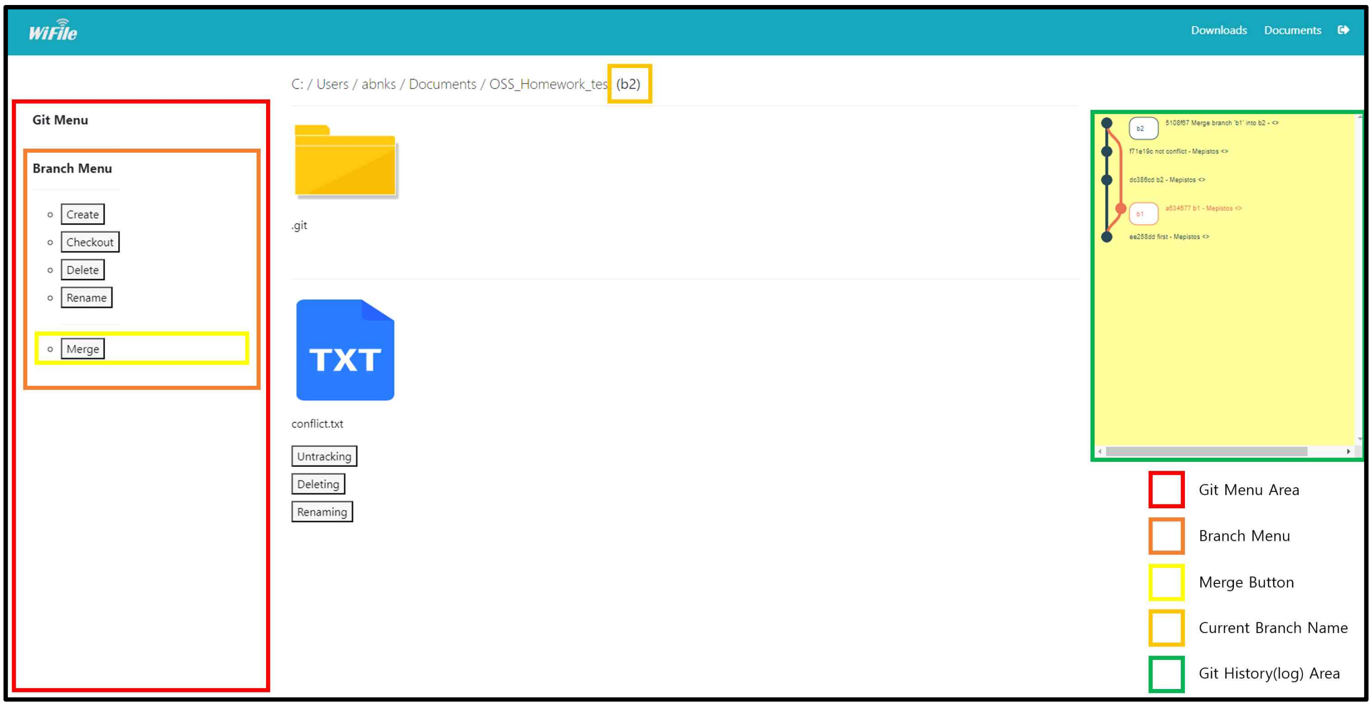
- Use decorators to respond to calls to specific routes.
- The return value for the call is render_template(~~) that renders html.

- UI
  - ☞ first term.



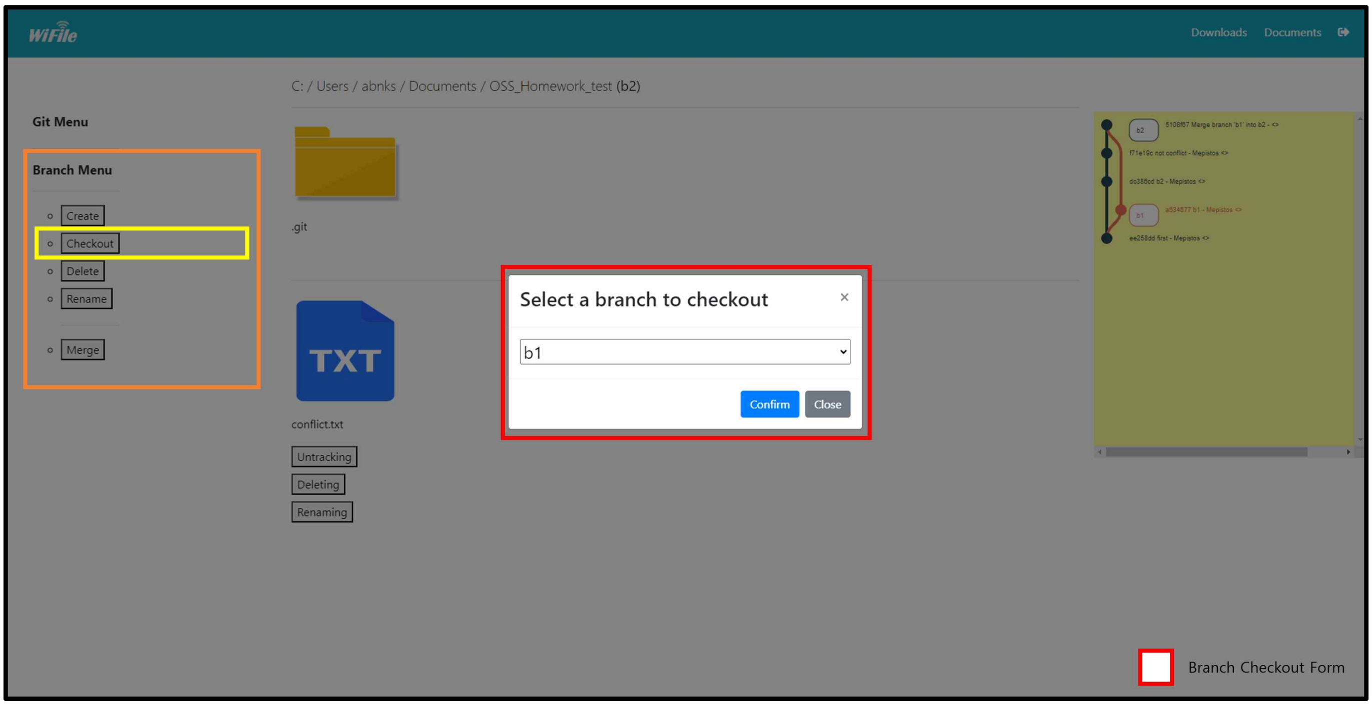


  - ☞ second term



C: / Users / abnks / Documents / OSS_Homework_tes (b2)

```
5108f67 Merge branch 'b1' into b2 - <>
f71e19c not conflict - Mepistos <>
dc386cd b2 - Mepistos <>
a634677 b1 - Mepistos <>
ee258dd first - Mepistos <>
```



C: / Users / abnks / Documents / OSS_Homework_test (b2)

Select a branch to checkout ×

b1

- Discussion
  - License
    => MIT
    : First, following base file explorer program's license, we selected MIT License. Furthermore, we found that there is lots of groups to try implementation of git to existing file system. To contribute other developer group, even it's small amount, we selected MIT License.

  - Expression method when one file has multiple states
    => Parsing the execution result of git status into a table

    : There have been many discussions, such as how to express multiple statuses together, how to prioritize statuses and express only the status with the highest priority, but in the end, it was determined by git that following the results of git status was the most accurate. Therefore, the result of git status is parsed and displayed.

Icons are displayed in the order of untracked, modified, staged, and committed in order of priority.

- Project Progress method

  ① Modify the filesystem make possible to add git commands.

  ② Parallel development of functions

  Each function is related to the state of the file and affects each other, but the function itself operates independently and unrelated to each other, so parallel development is possible.

  ⁓ first term

Name	Task
김서해	init / add
김승진	initial work(1) / commit
임종승	rm / mv
허지범	restore

  ⁓ second term

Name	Task
김서해	commit histroy
김승진	git clone / maintenance
임종승	git branch merge
허지범	git branch manage

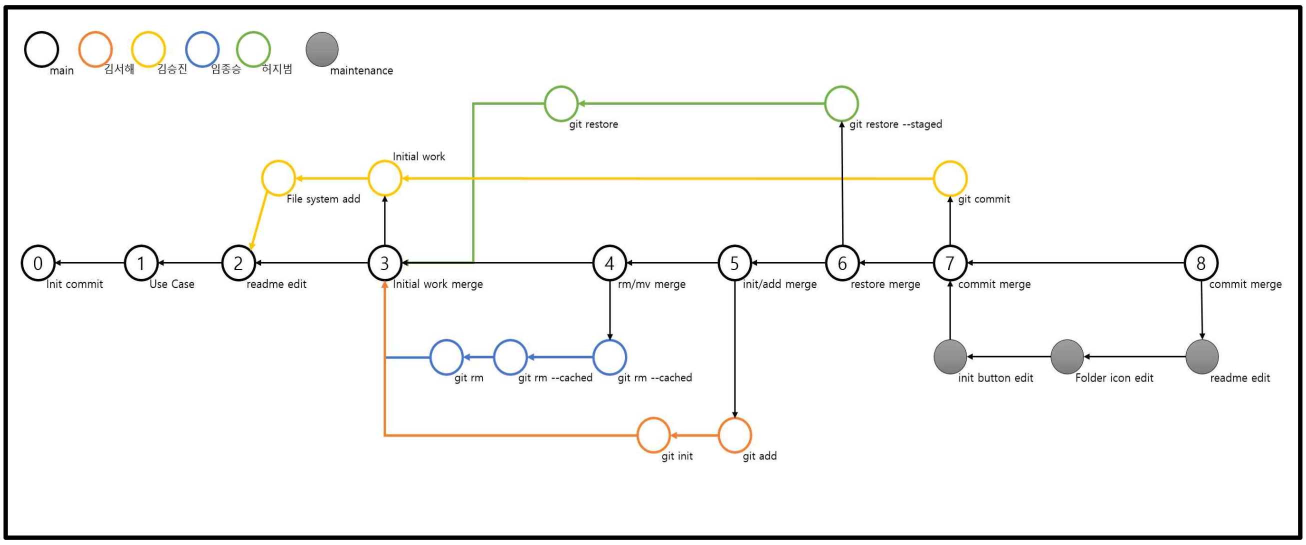
  ③ Merge and maintenance

  Since each function operates independently of each other, there is a low possibility of conflict when merging in logical operation (setup.py). However, in the process of editing html, there may be errors in UI implementation, such as changing the order of buttons during merge.

  Therefore, after everyone has completed development, proceed with the merge Create a "maintenance" branch to hotfix errors on the UI.

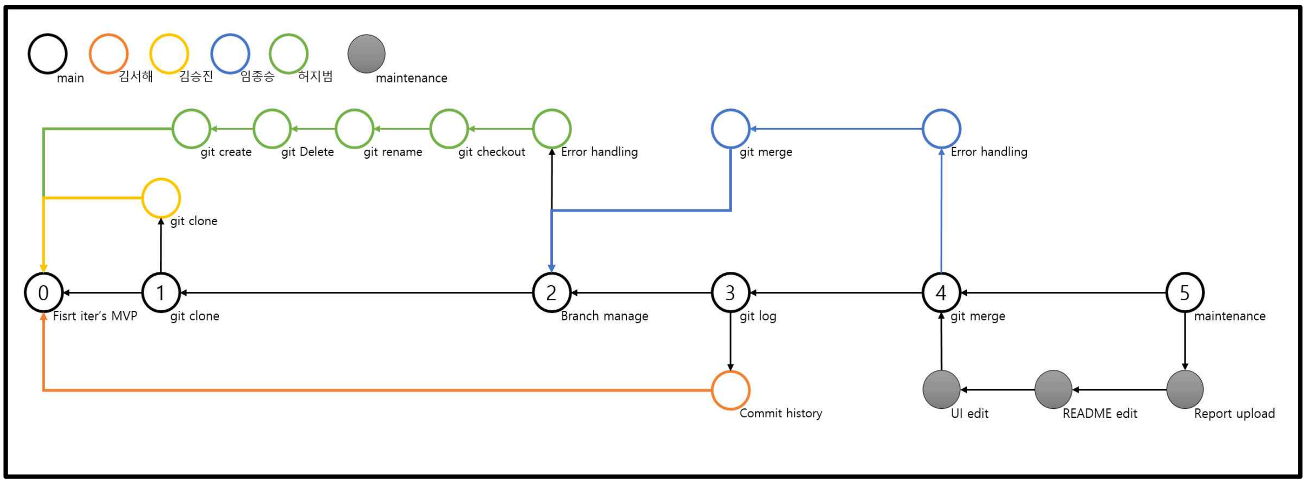
6. Collaboration History

- Simplified commit history – first term



There are omitted or increased parts to make the overall flow easier to understand.

- Simplified commit history – second term



There are omitted or increased parts to make the overall flow easier to understand.

7. for future development

There are a lot of features in git and GitHub that could not be covered in this project due to time constraints.
Therefore, I think it is good to study and develop by considering the following points in studying git and GitHub.

- git

  ① git merge:
  The FMOS project only warns the user in case of a conflict in "git merge" and provides only the option to go back.
  There are a number of ways to track this more usefully and give you the option to fix it, so we encourage you to study it.

  ② git clone:
  In receiving the user's id and token, FMOS stores and uses them in 'config.json'. There is no problem in the immediate implementation, but it is a very big risk in

terms of security. Therefore, it is good to devise a way to use session/cookie, etc.

- GitHub

  ③ action:

  GitHub provides actions as a tool to track repo issues or to automatically respond to 'pull requests'. Therefore, although it is not currently implemented in FMOS, it is good to study tools that can speed up code_review by adding actions such as checking 'conding_convention'.

  ④ security:

  If you look at the Dependabot alerts in the security tab of the repository, you can see that there are 4 security advisories.

  I think it's also good to study how to solve this, and it would be good to think about it in relation to the security issue of 'git clone' mentioned above.