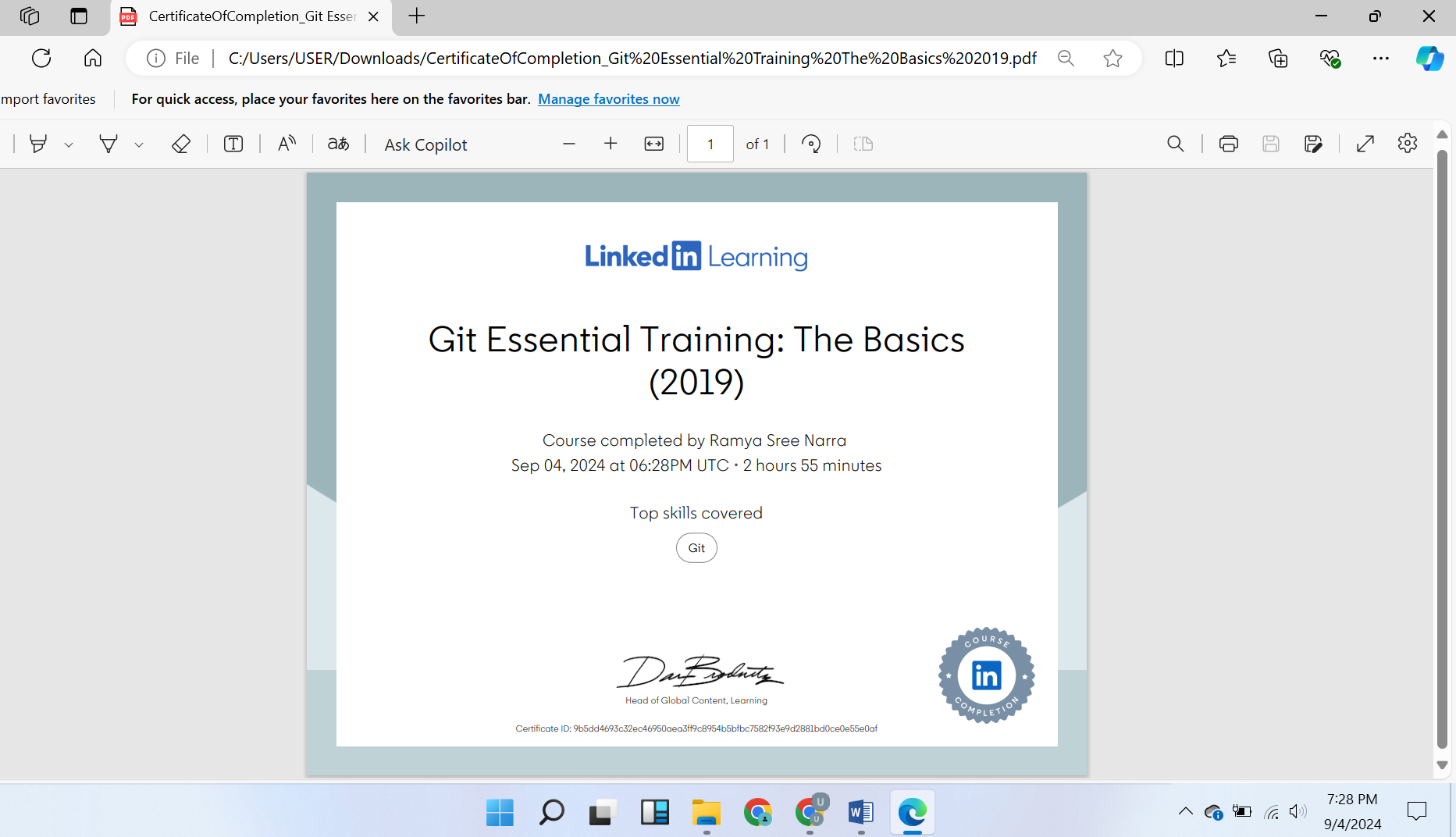
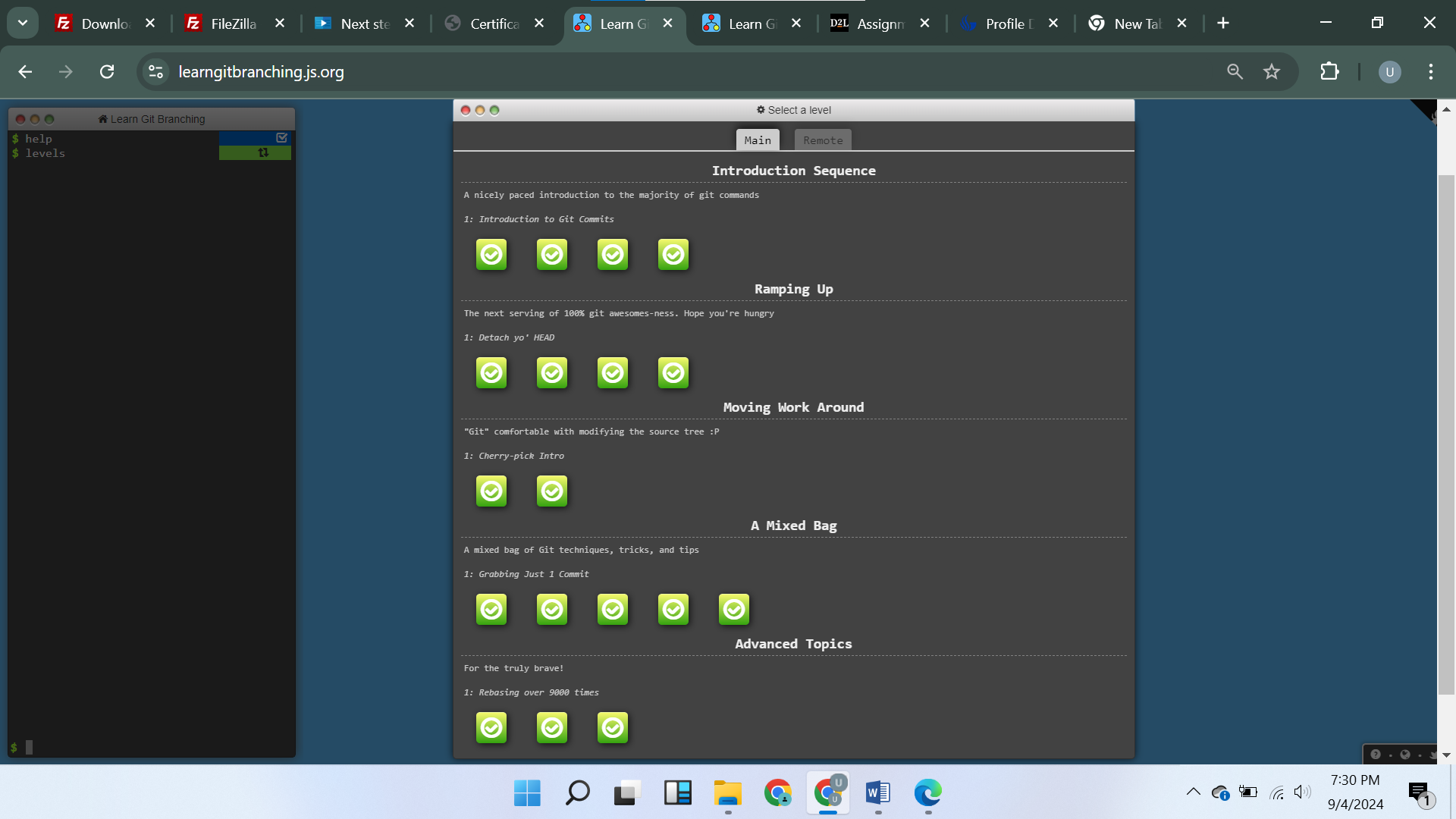
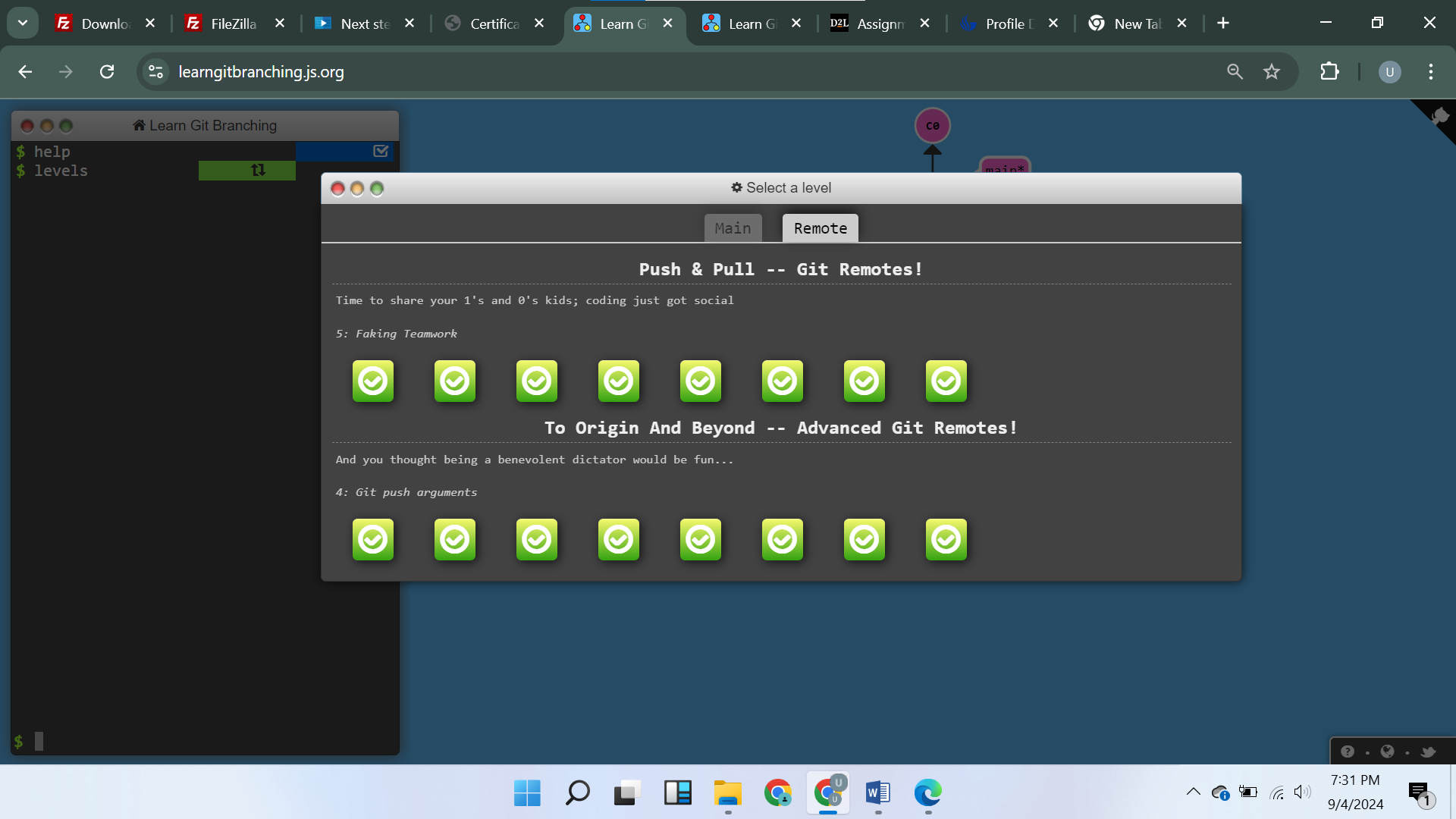
WEB PRO  
Assignment 0

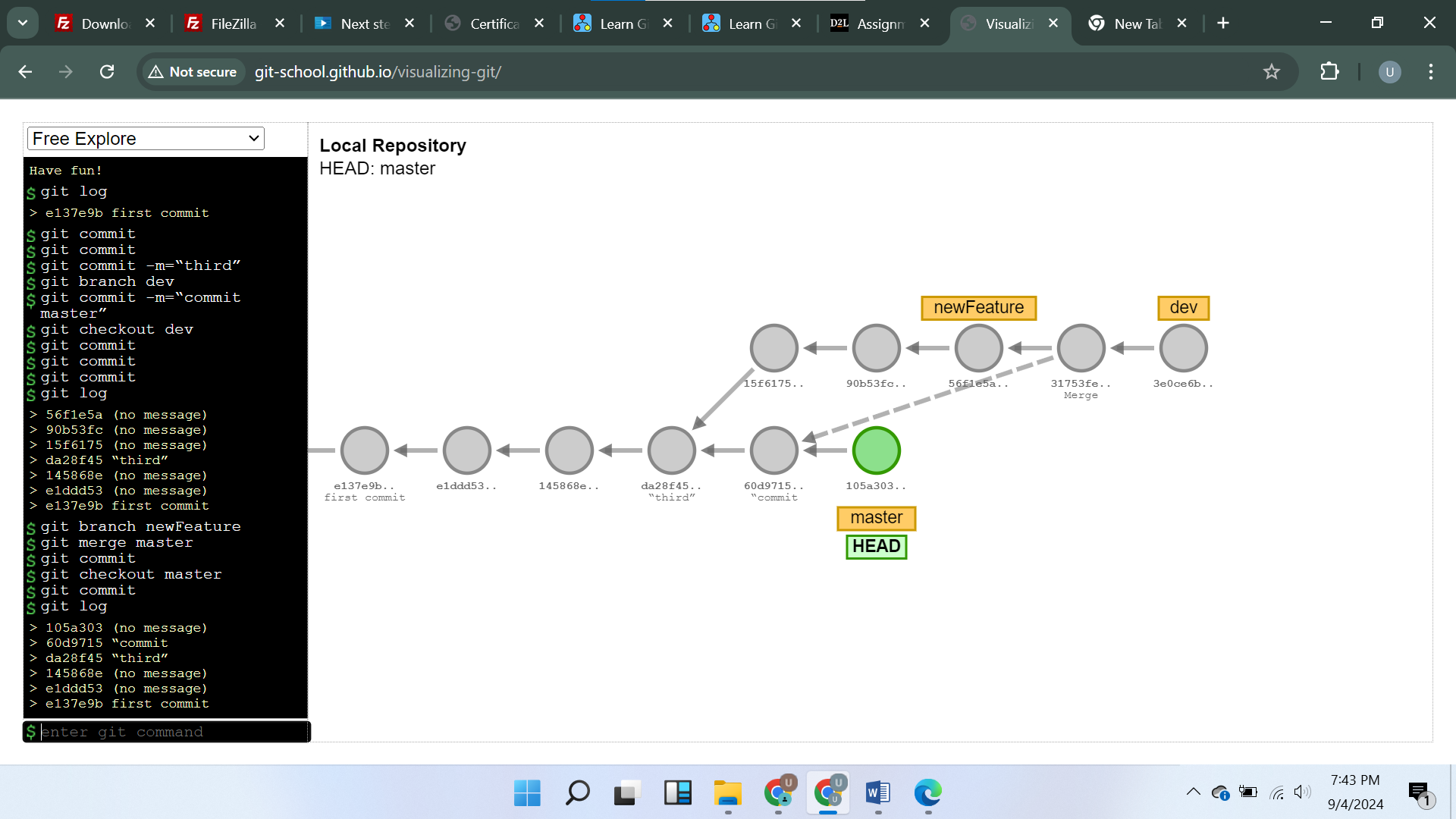
1.Git essential training completion certificate:



2. Learn Git Branching (completion of main and remote levels):





3. Visualizing Git – execution of Git commands

4.

a. .gitignore file is used to untrack specific files and folders in version control and are ignored by Git. It helps the developers to exclude unnecessary files and sensitive data being tracked. This is mostly useful in large projects where many of the files are automatically generated like build files, temporary files, node modules, and local configurations.

.gitignore file content usually consists of patterns that specifies file or directory. These patterns can be generic or specific based on the requirement. We can also use wildcard characters like \* to specify multiple files with that particular pattern. For example, \*.env – git will ignore all the files which has .env extension regardless of file name.

Example of .gitignore file

#dependency modules

node\_modules/

#build files

build/

dist/

#log files

\*.log

#temporary files

\*/temp/\*

#environment files

.env

b. **git diff** command is used to compare the changes in two commits in Git. It helps developers to keep track of every minor change done at different points of time in the project.

syntax: git diff <commit-hash1> <commit-hash2> (commit-hash1 and commit-hash2 are the hash value of the respective commits)

example: git diff edf837 akh927

The above command is used to compare the changes between two commits edf837 and akh927.

c. HEAD pointer always points to current commit on the currently checkedout branch.

While switching from feature branch to main using git checkout the head pointer moves from latest commit in main to the latest commit in the feature branch.

When we make a new commit HEAD points to the new commit in that specific branch.

It is possible to make HEAD point to specific commit directly through git checkout but not connected to branch reference. This is referred as detached head in git

Syntax: git checkout <commit-hash1>

Example: git checkout edf837

If head is point to the recent commit, the above command detaches the head and points to the edf837 commit.

d. **git checkout** command is used to undo the changes in the working directory.

To discard the changes in a single file in the working directory

Syntax: git checkout -- filename

Example: git checkout -- index.html

The above command discards any change in the working directory and revert to the state of last commit in the repository.

To discard all the changes in the working directory

Example: git checkout .

e. **git revert** command is used to undo the changes done by the particular commit

**syntax**: git revert <commit-hash> (commit-hash is the hash value of the commit that we want to revert)

**example**: git revert edf837

The above command creates a new commit that undo the changes done by edf837 commit.

It is possible to undo the changes done by multiple commits by specifying the range of commits.

**Syntax**: git revert <commit-hash1> <commit-hash2> <commit-hash2>...<commit-hashN>

5. GitHub username: ramya036

https://github.com/ramya036