Clinic Management System

Team No: 04
Team Members: Rashmi Nayak S
Malavika K Nair
Nithin Peter

Git Guidelines Manual

Create branches under the main branch as per your requirements to keep the work sorted in the repository. In our project it is as follows.

1. Development/RestAPI - Backend- (Asp.Net)
2. Development/UI - frontend (angular)
3. Documents - contains documentations related to the project

Steps:

1. Initializing git repository

Step 1: create a remote repository on github.
Step 2:create a project on a local device.
Step 3: initialise project folder as git folder by using 'git init' command
Step 4: make initial commit by using following command:

- Git add . (Stage changes)
- Git commit -m "developer message"

Step 5: connect local project folder to remote git repository by following command:

- Git add remote

Eg: git add remote origin https://github.com/team4/sample.git
Step 6: make new branches if needed

- Git checkout -b Example: git checkout -b development/ui

Step 7: push committed changes to remote repository

- Git push -u -f Example: git push -u -f origin development/ui

Note: The contents in the local repository will now be uploaded to the remote repository from which other team members can pull and work with their respective tasks.

2. Using the existing git repository by Cloning

Step1 : use following command to clone existing remote git repository and use it in our local system.

- Git clone - -branch

Example: git clone https://github.com/team4/sample.git - - branch development/ui

Step 2: after making any changes and want to upload it to a remote repository use the following commands.

- Git add or git add -A
- Git commit -m “developer message”
- Git push Example: git push origin development/ui

Note: Every time when a developer/team member wants to upload the changes that they have made, one should perform a ‘pull’ from the remote repository.

- Git checkout Example:git checkout development/ui
- Git pull

3. Resolving Conflicts

Note: When you perform “Pull”, most likely it will result in conflicts in the files so we have to check where the conflict is and resolve the conflict using three options. They are :

- Accept incoming changes
- Accept current changes
- Accept both changes

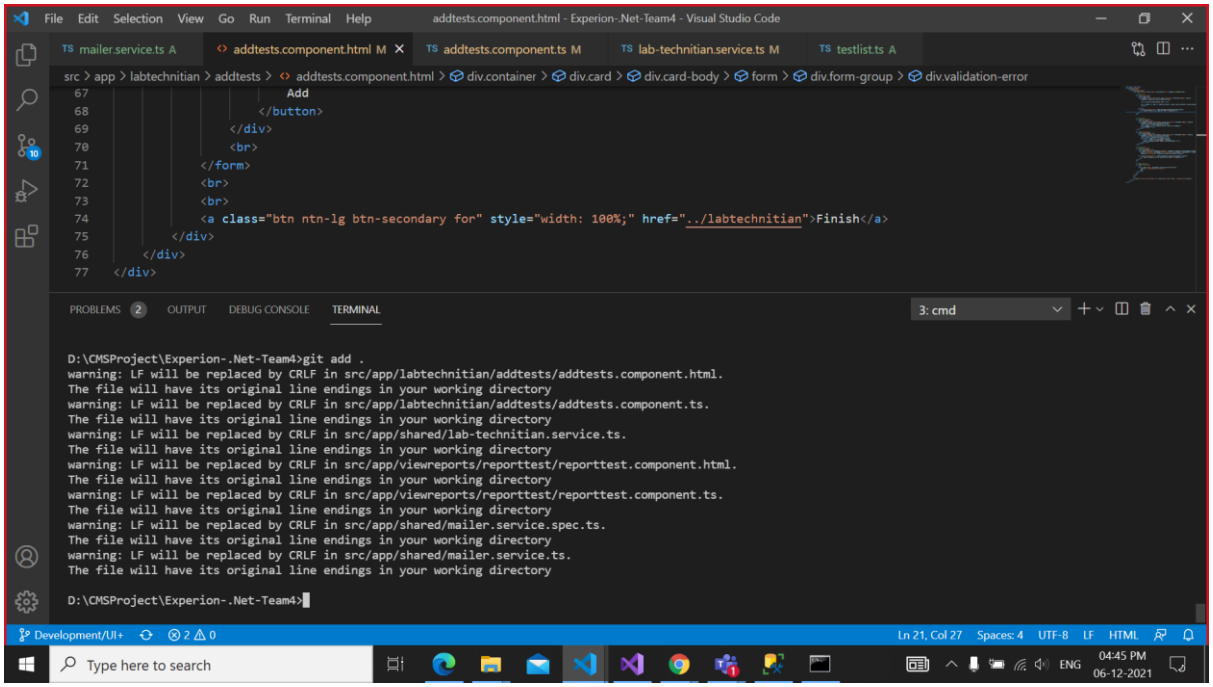
It depends on the type of operation leading to that conflict.
In your case, a merge, where:

- current change represents what you have (the destination of the merge)
- incoming change represents what you merge (the source of the merge)
  Then:
- Option 1 ("Accept Incoming changes") would ignore completely what you had, and keep what you merge.
- Option 2 ("Accept current changes") would ignore completely what you merge, and keep what you had.
- Option 3 (“Accepts both incoming and current changes”)

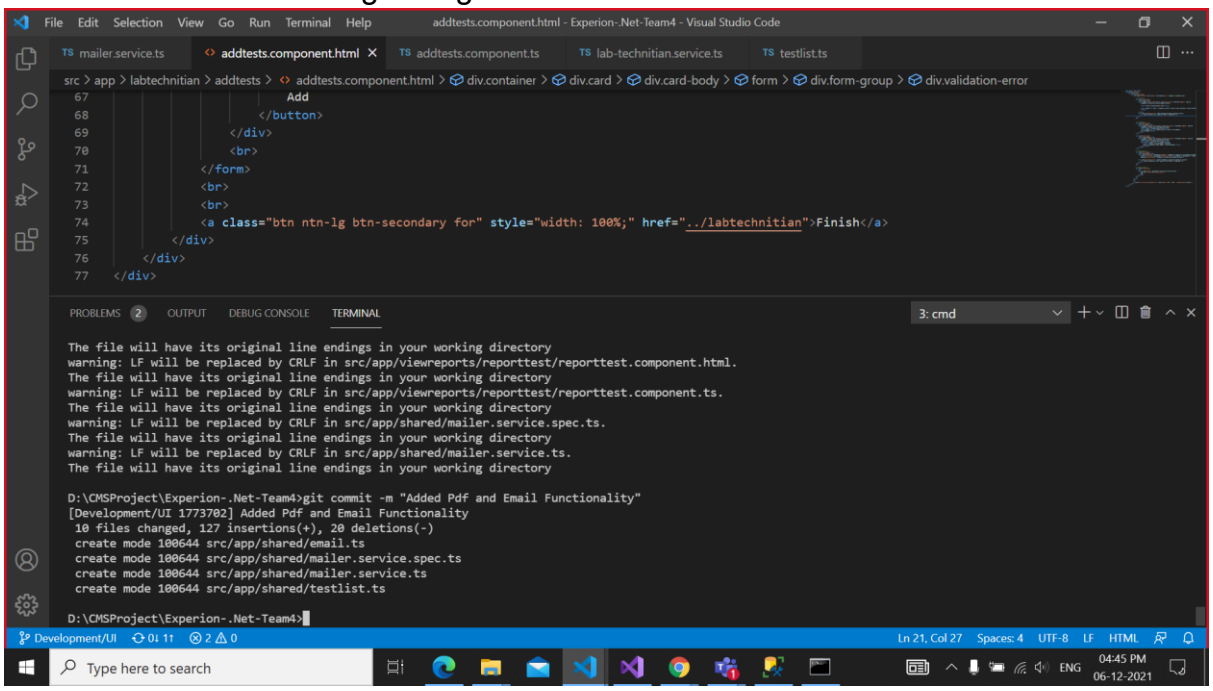
After choosing the right option and resolving the conflict click on stage changes.

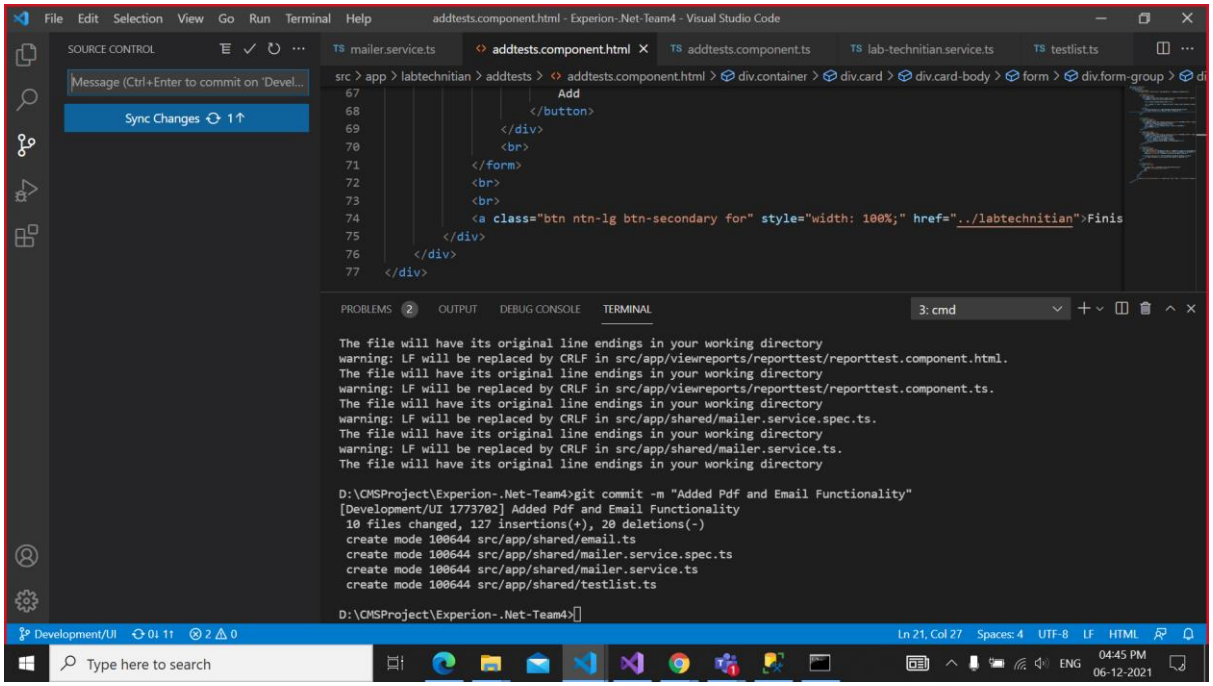
Screenshots showing some operations in git using gui and cmd.

Git add . to stage changes

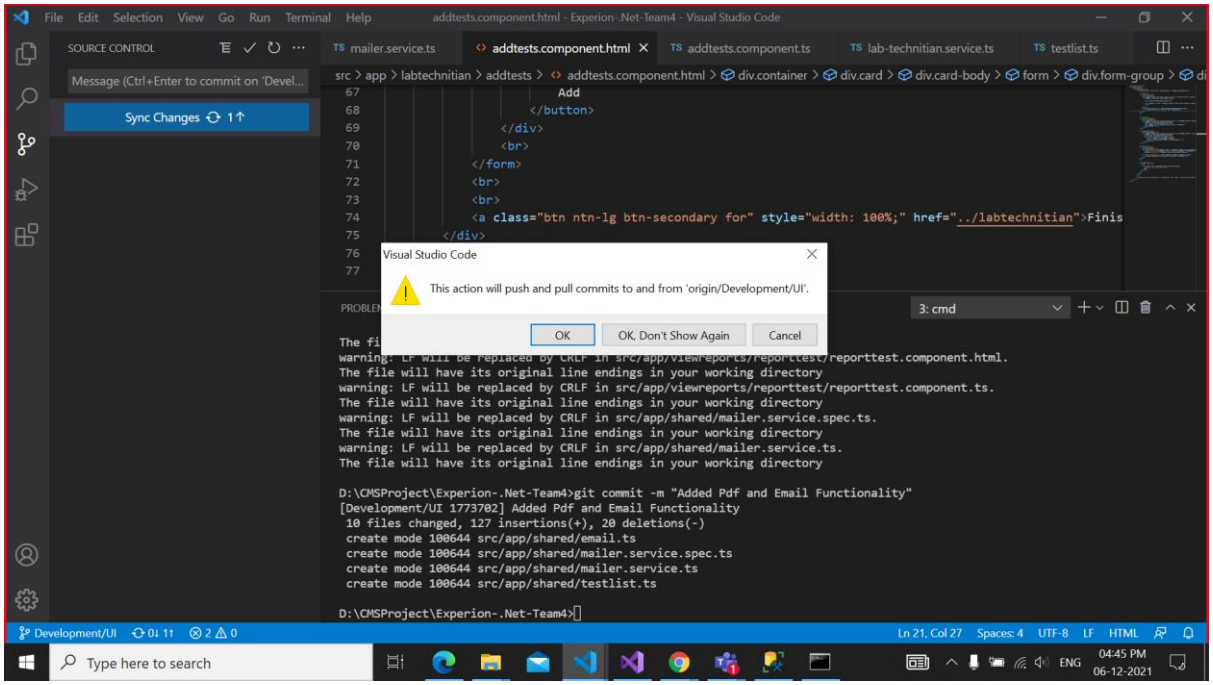


Git commit -m “committing changes”

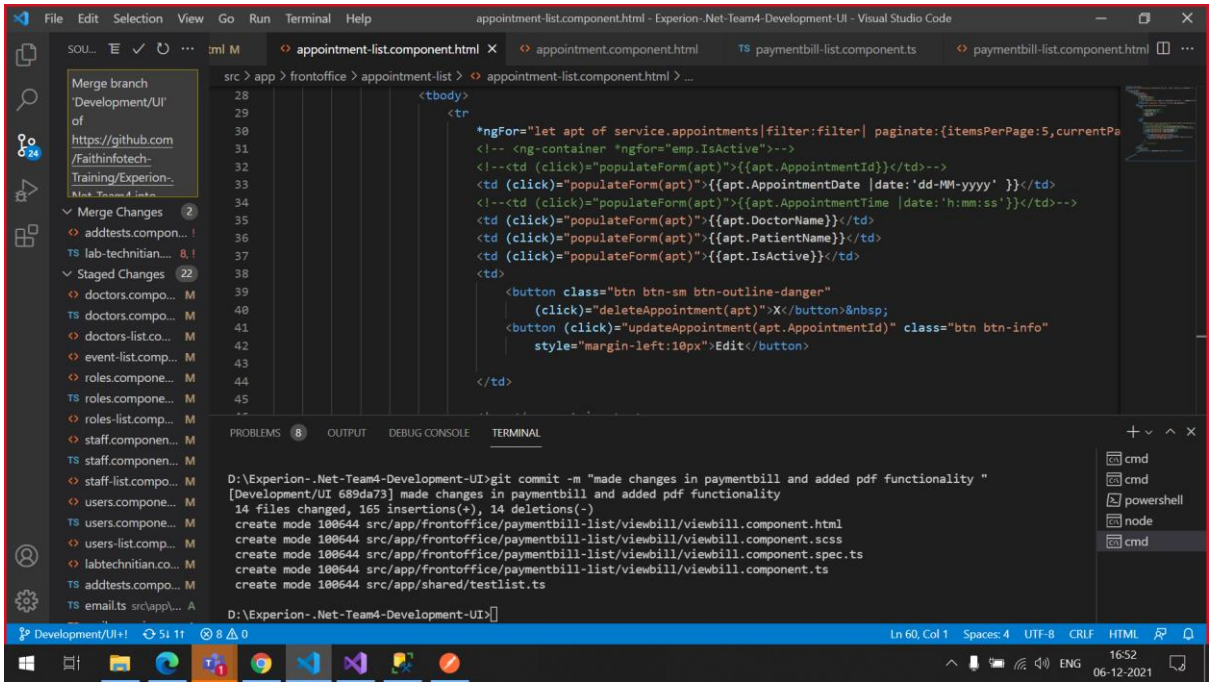




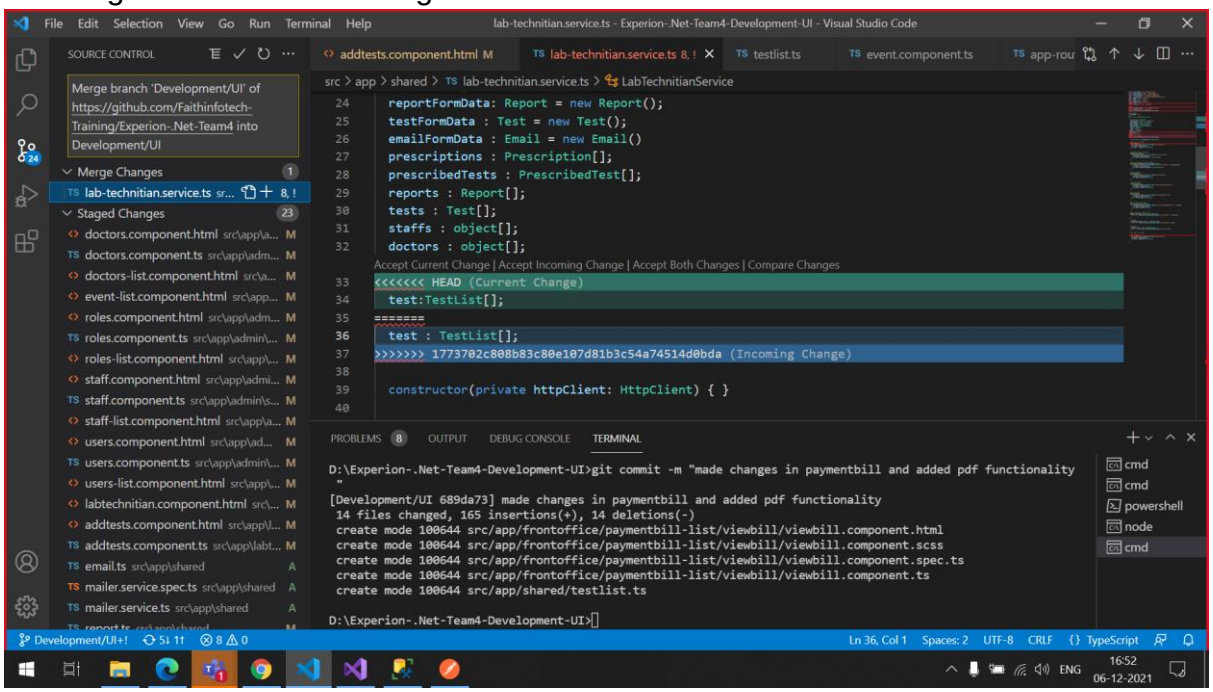
After committing changes click sync the changes. This action will push and pull commits from the branch.



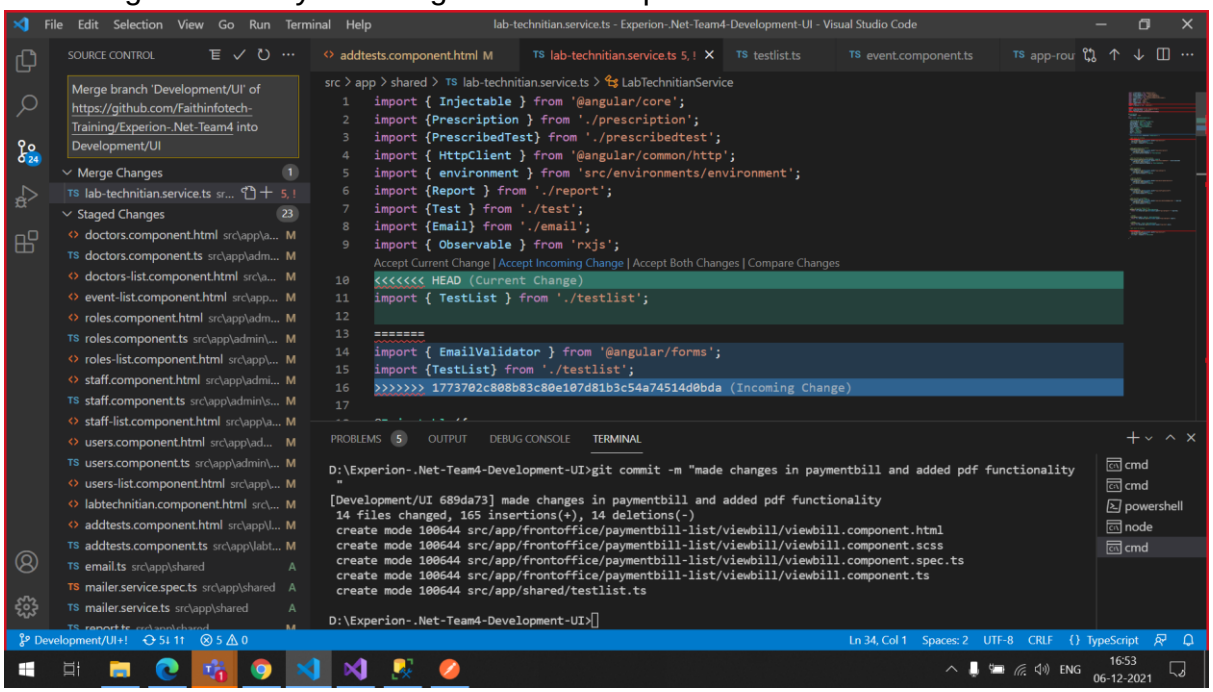
Another author committing their changes to the repository



Showing the conflicts occuring.



Resolving conflicts by choosing the desired options.



After resolving conflicts, commit--->commit staged