$$\boxed{\text{LAB - MST - 1}}$$

Set - B

Main task :

(i) First, one person in the group should create a public repository using their GitHub account.

(ii) This same person should then follow the instructions from GitHub to add a remote, and then push their repository. Do not forget the -u flag, as suggested by GitHub!

(iii) All of the other members of the group should then be added as collaborators, so they can commit to the repository also.

(iv) Next, everyone else in the group should clone the repository from GitHub. Verify that the content of the repository is what is expected.

(v) One of the group members who just cloned......

. . . .

(i) → open your GitHub profile
→ Go to the 'Repositories' section
→ Click on 'New' button
→ select the owner account and give the name of the repository.
→ select the 'public' option to make the repository public.
→ optionally, you can select to add '.gitignore' and 'licence'.
→ add the (README.md) file.
→ Click on 'create' button.

(ii) Add remote
└→ git remote add origin <URL>

push
└→ git push -u origin main

(iii) → Go to the repository
→ open the 'settings' tab.

(iii) → On the left option-panel, click on 'collaborators' option.

(iv) → Here, you can Invite the collaborators and give them the access right (write or admin).

(iv) <u>Clone → in git bash</u>

↳ git clone <URL>

(v) <u>Local commit</u> → (git checkout -b test)

→ touch file.txt

→ vi file.txt

→ git add file.txt

→ git commit -m "Added file.txt in the cloned repository"

<u>push the changes</u>

→ git push origin main test

<u>check for changes</u>

→ git log --oneline commits

<u>pull the changes</u>

→ git pull origin test

(vi) SHA-1 is important because -

→ it represents the commit hash code of the performed commit.

→ It can be used to only merge that particular commit.

→ SHA-1 is unique for every commit because it is a 40-hexa decimal character hash code and represents how the commit was performed and what were done in it.

(vii) If both members try to commit locally and have to push it, the runner-up will face problems because the commit performed and pushed by the first person are not synced with the runner-up's repository and the runner-up will have to pull the changes first and then push his changes in order to resolve the problem.

(viii)   → git pull origin main

     → git ~~push~~ log --oneline

     → git push origin main

Stretch task

  (a) In the screenshots (in the 22BDO10019_LAB-MST-1.md file)