EX .NO:

DATE:

Create a local repository to Launch the Git bash console.

Aim:

To create a local Git repository and launch the Git Bash console.

Algorithm:

1. Open Git Bash by right-clicking inside the desired folder and selecting "Git Bash Here".
2. Initialize a local Git repository.
3. Add files to the staging area.
4. Commit the changes with a meaningful message.
5. View commit history to verify changes.

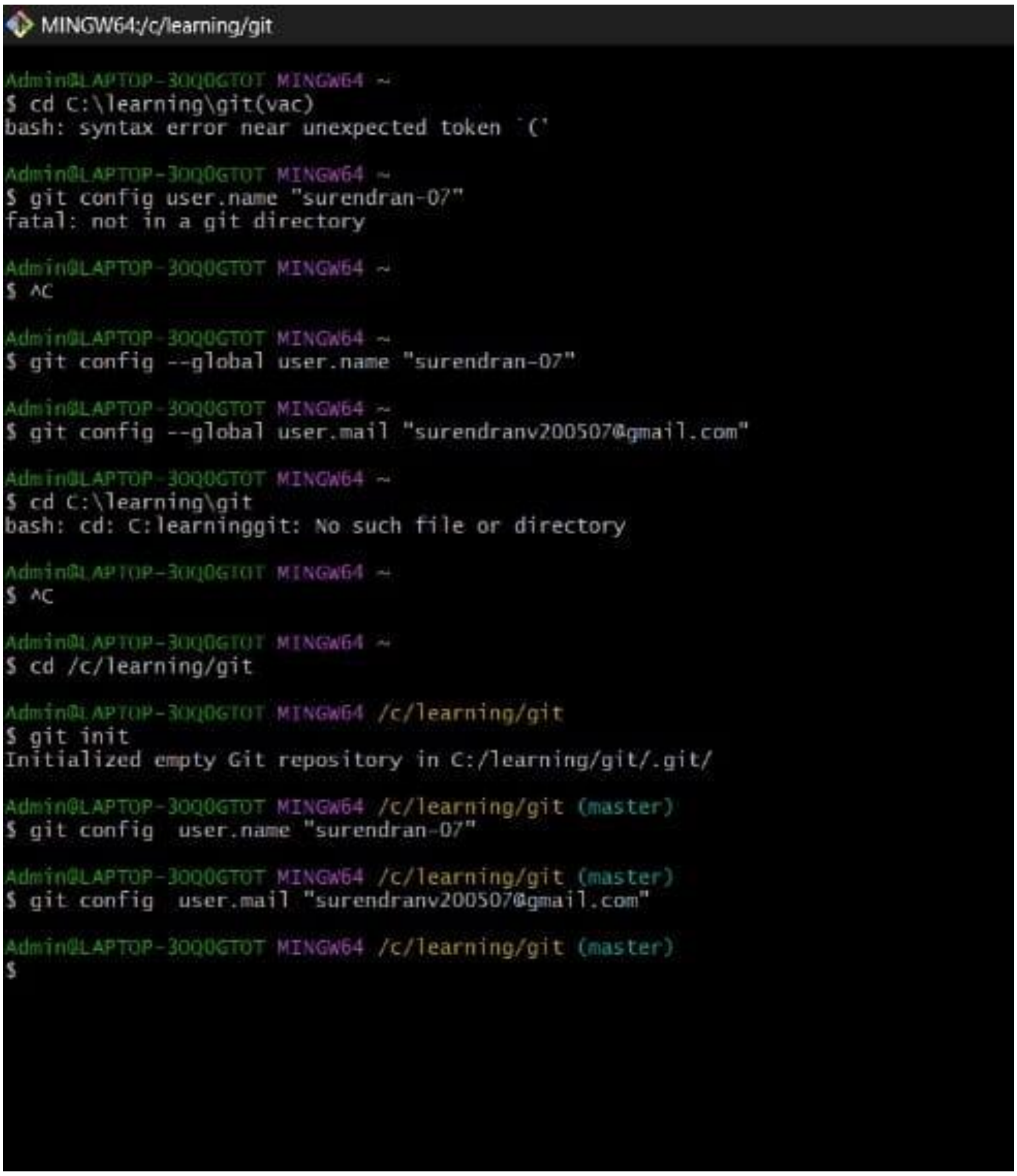
Commands:

```
git init
git add  .
git commit -m "Initial commit"
git log
```

EX .NO:

DATE:

Output:



```
MINGW64:/c/learning/git

Admin@LAPTOP-3OQ0GTOT MINGW64 ~
$ cd C:\learning\git(vac)
bash: syntax error near unexpected token `('

Admin@LAPTOP-3OQ0GTOT MINGW64 ~
$ git config user.name "surendran-07"
fatal: not in a git directory

Admin@LAPTOP-3OQ0GTOT MINGW64 ~
$ ^C

Admin@LAPTOP-3OQ0GTOT MINGW64 ~
$ git config --global user.name "surendran-07"

Admin@LAPTOP-3OQ0GTOT MINGW64 ~
$ git config --global user.mail "surendranv200507@gmail.com"

Admin@LAPTOP-3OQ0GTOT MINGW64 ~
$ cd C:\learning\git
bash: cd: C:learninggit: No such file or directory

Admin@LAPTOP-3OQ0GTOT MINGW64 ~
$ ^C

Admin@LAPTOP-3OQ0GTOT MINGW64 ~
$ cd /c/learning/git

Admin@LAPTOP-3OQ0GTOT MINGW64 /c/learning/git
$ git init
Initialized empty Git repository in C:/learning/git/.git/

Admin@LAPTOP-3OQ0GTOT MINGW64 /c/learning/git (master)
$ git config  user.name "surendran-07"

Admin@LAPTOP-3OQ0GTOT MINGW64 /c/learning/git (master)
$ git config  user.mail "surendranv200507@gmail.com"

Admin@LAPTOP-3OQ0GTOT MINGW64 /c/learning/git (master)
$
```

Result:

Thus we Successfully created a local repository to Launch the Git bash console.

EX .NO:

DATE:

Create a new directory and use init command for a local workflow in Git.

Aim:

To create a new directory and initialize a local Git repository for workflow management.

Algorithm:

1. Open Git Bash.
2. Navigate to the location where the new directory should be created using the cd command.
3. Create a new directory using the mkdir command and move into it.
4. dfdInitialize a local Git repository inside the directory.
5. Check the repository status to confirm initialization.

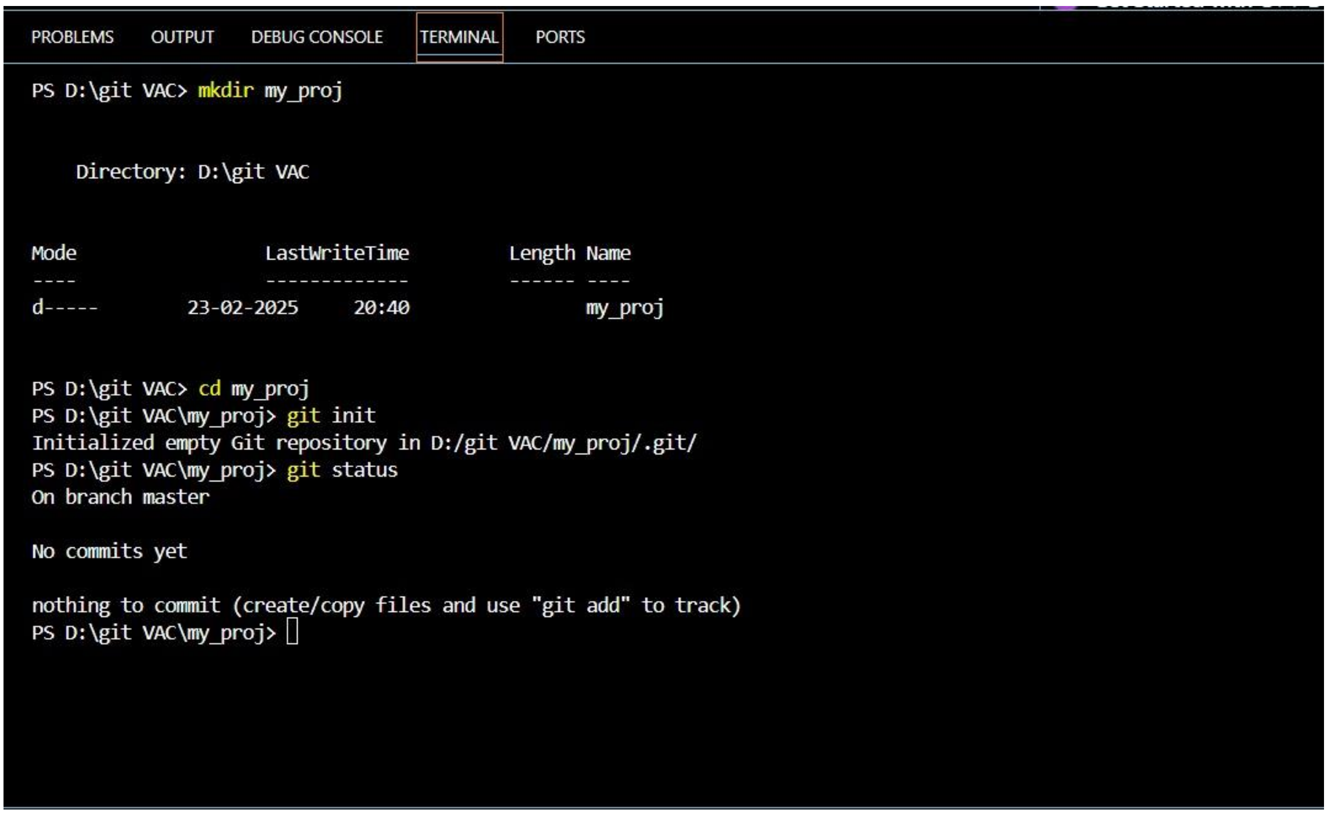
Commands:

```
cd path/to/desired/location

mkdir my_project

cd my_project

git init

git status
```

EX .NO:

DATE:

Output:



```
PS D:\git VAC> mkdir my_proj


    Directory: D:\git VAC


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        23-02-2025     20:40                my_proj


PS D:\git VAC> cd my_proj
PS D:\git VAC\my_proj> git init
Initialized empty Git repository in D:/git VAC/my_proj/.git/
PS D:\git VAC\my_proj> git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
PS D:\git VAC\my_proj> 
```

Result:

Thus we successfully created a new directory and used init command for a local workflow in Git.

EX .NO:

DATE:

Perform a remote workflow operation between two users using Git.

Aim:

To perform a remote workflow operation between two users using Git.

Algorithm:

1. User 1: Create a GitHub repository and push the initial code.
2. User 2: Clone the repository and make changes.
3. User 2: Commit and push the changes to the remote repository.
4. User 1: Pull the latest changes.

Commands:

User 1: Initialize and Push to GitHub

```
git init
git remote add origin https://github.com/user/repo.git
git add .
git commit -m "Initial commit"
git push -u origin main
```

User 2: Clone and Push Changes

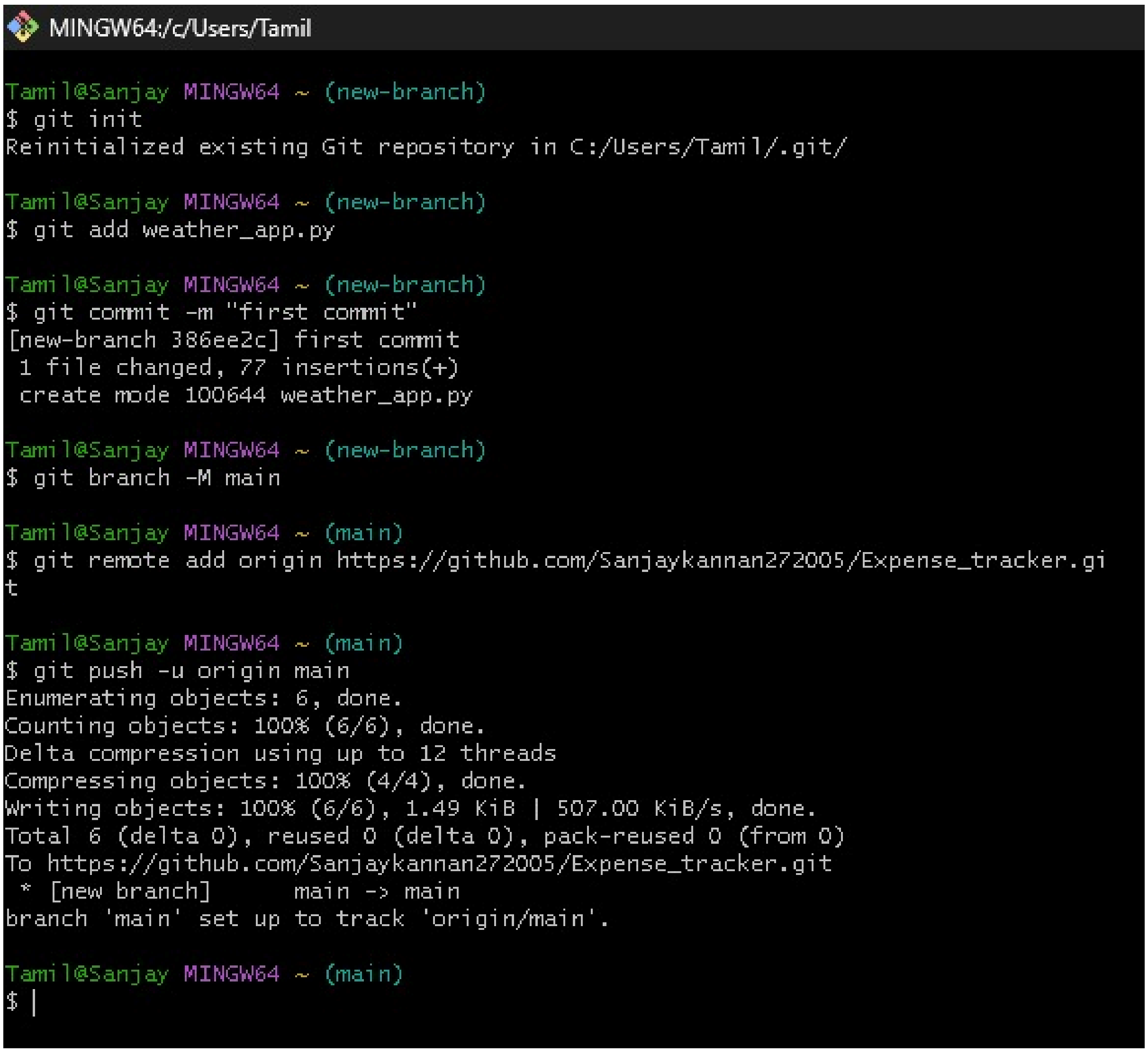
```
git clone https://github.com/user/repo.git
cd repo
git add .
git commit -m "Updated by User 2"
git push origin main
```

User 1: Pull Changes git pull origin main

EX .NO:

DATE:

Output:



MINGW64:/c/Users/Tamil

```
Tamil@Sanjay MINGW64 ~ (new-branch)
$ git init
Reinitialized existing Git repository in C:/Users/Tamil/.git/

Tamil@Sanjay MINGW64 ~ (new-branch)
$ git add weather_app.py

Tamil@Sanjay MINGW64 ~ (new-branch)
$ git commit -m "first commit"
[new-branch 386ee2c] first commit
 1 file changed, 77 insertions(+)
 create mode 100644 weather_app.py

Tamil@Sanjay MINGW64 ~ (new-branch)
$ git branch -M main

Tamil@Sanjay MINGW64 ~ (main)
$ git remote add origin https://github.com/Sanjaykannan272005/Expense_tracker.gi
t

Tamil@Sanjay MINGW64 ~ (main)
$ git push -u origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 1.49 KiB | 507.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Sanjaykannan272005/Expense_tracker.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

Tamil@Sanjay MINGW64 ~ (main)
$ |
```

Result:

Thus we successfully Performed a remote workflow operation between two users using Git.

EX .NO:

DATE:

Create a program to work with local branch using Git

Aim:

To work with a local branch using Git.

Algorithm:

1. Initialize a Git repository.
2. Create and switch to a new branch.
3. Make changes and commit them.
4. Switch back to the main branch.
5. Merge the branch into the main branch.
6. Delete the branch if not needed.

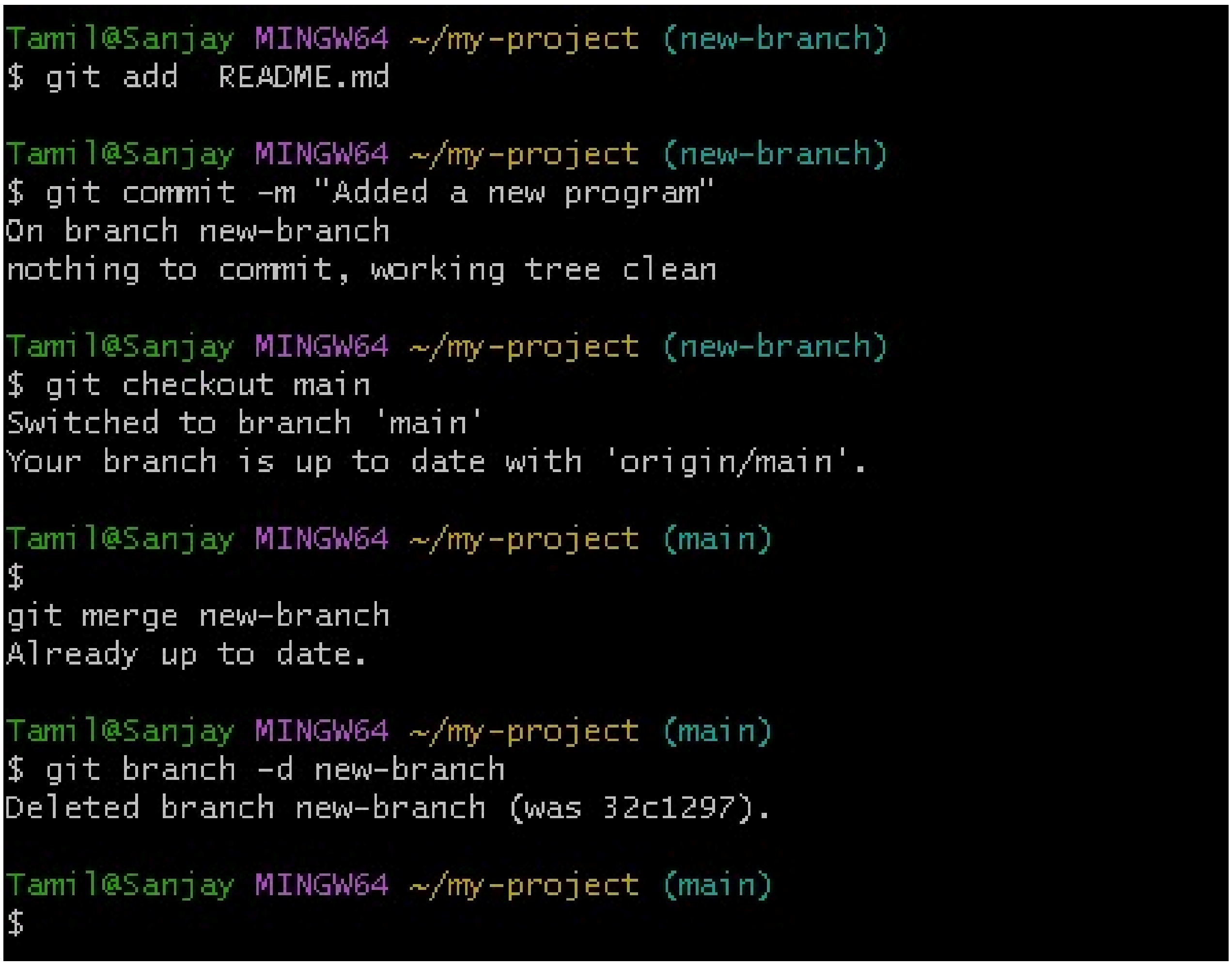
Commands:

```
git init
git checkout -b new-branch
echo "This is sample ” > sample.txt
git add .
git commit -m "Added a new program"
git checkout main
git merge new-branch
git branch -d new-branch
```

EX .NO:

DATE:

Output:



```
Tamil@Sanjay MINGW64 ~/my-project (new-branch)
$ git add  README.md

Tamil@Sanjay MINGW64 ~/my-project (new-branch)
$ git commit -m "Added a new program"
On branch new-branch
nothing to commit, working tree clean

Tamil@Sanjay MINGW64 ~/my-project (new-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Tamil@Sanjay MINGW64 ~/my-project (main)
$
git merge new-branch
Already up to date.

Tamil@Sanjay MINGW64 ~/my-project (main)
$ git branch -d new-branch
Deleted branch new-branch (was 32c1297).

Tamil@Sanjay MINGW64 ~/my-project (main)
$
```

Result:

Thus we successfully Created a program to work with local branch using Git.

EX .NO:

DATE:

Create a program to perform Branching, Merging and Conflict tasks in Git

Aim:

To perform branching, merging, and resolve conflicts in Git.

Algorithm:

1. Initialize a Git repository.
2. Create and switch to a new branch.
3. Modify a file and commit changes.
4. Switch back to the main branch and modify the same file.
5. Merge the branch into the main branch.Resolve conflicts if they occur and commit the final changes.

Commands:

```
git init
git checkout -b new-branch
echo "New change" > file.txt
git add .
git commit -m "Change from new branch"
git checkout main
echo "Main branch change" > file.txt
git add .
git commit -m "Change from main branch"
git merge new-branch
```

EX .NO:

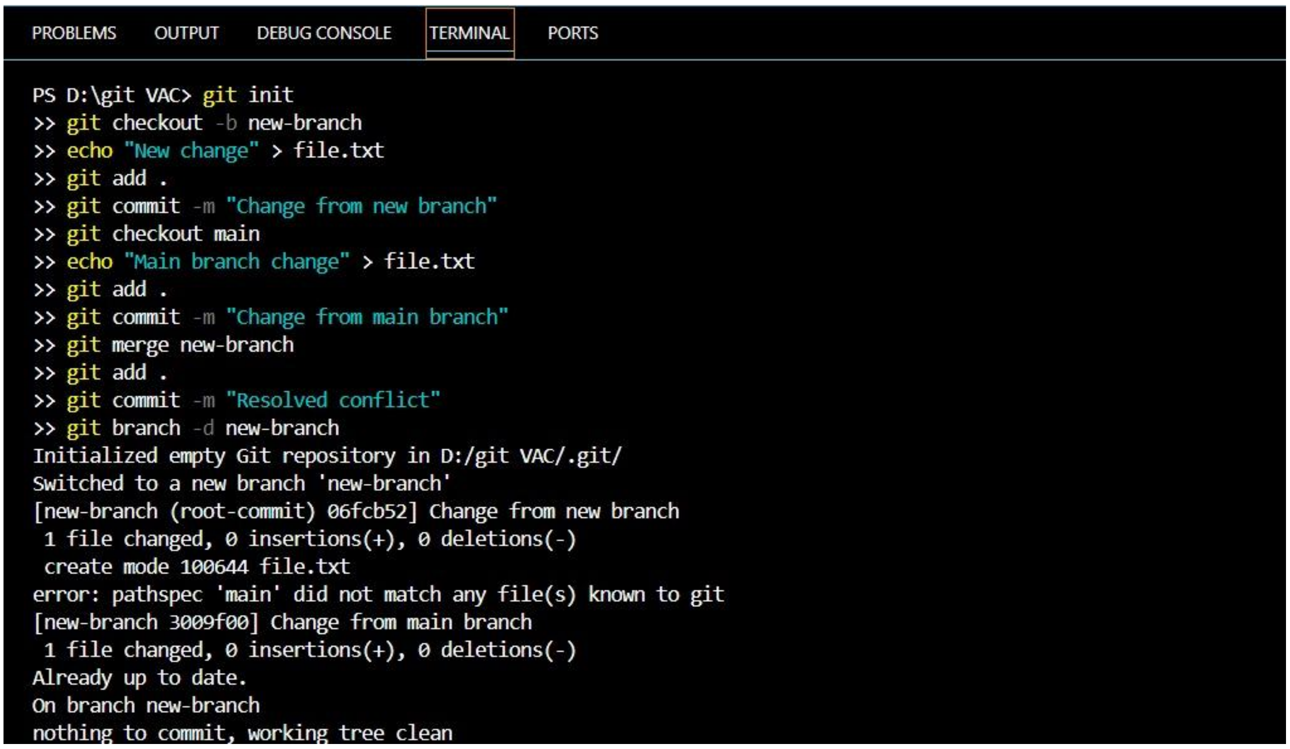
DATE:

git add.

git commit -m "Resolved conflict"

git branch -d new-branch

Output:



```
PS D:\git VAC> git init
>> git checkout -b new-branch
>> echo "New change" > file.txt
>> git add .
>> git commit -m "Change from new branch"
>> git checkout main
>> echo "Main branch change" > file.txt
>> git add .
>> git commit -m "Change from main branch"
>> git merge new-branch
>> git add .
>> git commit -m "Resolved conflict"
>> git branch -d new-branch
Initialized empty Git repository in D:/git VAC/.git/
Switched to a new branch 'new-branch'
[new-branch (root-commit) 06fcb52] Change from new branch
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file.txt
error: pathspec 'main' did not match any file(s) known to git
[new-branch 3009f00] Change from main branch
 1 file changed, 0 insertions(+), 0 deletions(-)
Already up to date.
On branch new-branch
nothing to commit, working tree clean
```

Result:

Thus we successfully Created a program to perform Branching, Merging and Conflict tasks in Git.

EX .NO:

DATE:

Create a program to change history with amend using Git Commands

Aim:

To modify the last commit using the git commit --amend command in Git.

Algorithm:

1. Initialize a Git repository.
2. Create a file, add content, and commit.
3. Modify the file and amend the last commit.
4. Verify the commit history.

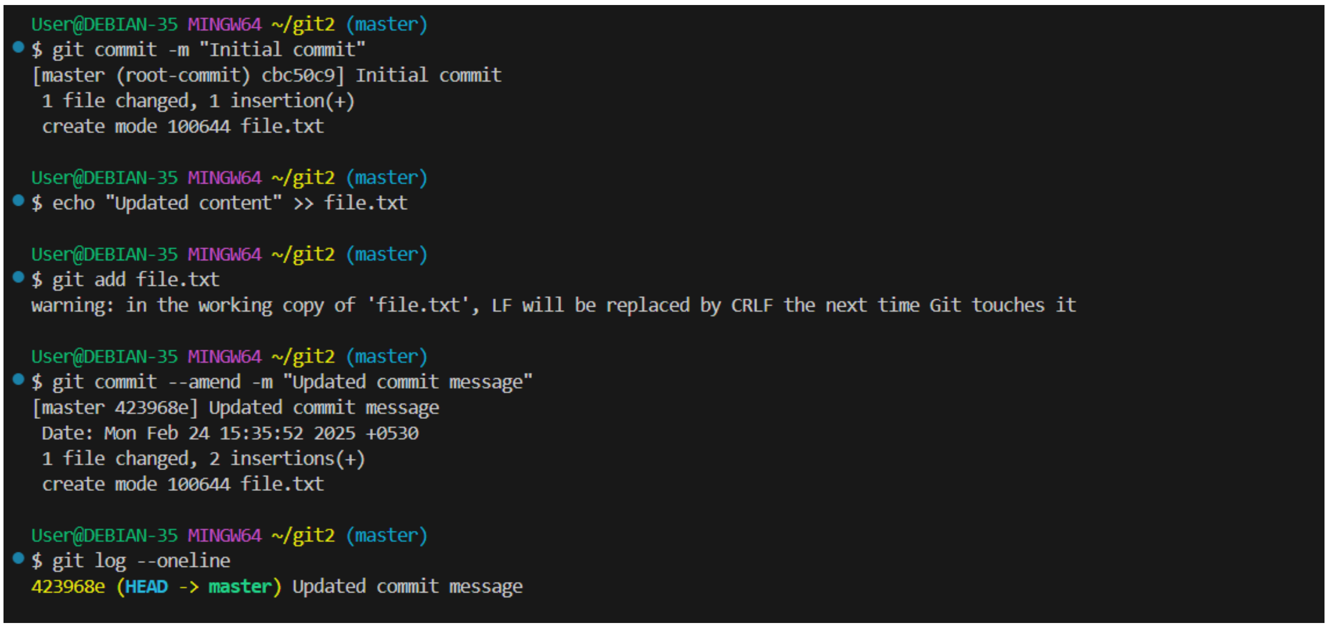
Commands:

```
git init
echo "Initial content" > file.txt
git add file.txt
git commit -m "Initial commit"
echo "Updated content" >> file.txt
git add file.txt
git commit --amend -m "Updated commit message"
git log --oneline
```

EX .NO:

DATE:

Output:



```
User@DEBIAN-35 MINGW64 ~/git2 (master)
$ git commit -m "Initial commit"
[master (root-commit) cbc50c9] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt

User@DEBIAN-35 MINGW64 ~/git2 (master)
$ echo "Updated content" >> file.txt

User@DEBIAN-35 MINGW64 ~/git2 (master)
$ git add file.txt
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next time Git touches it

User@DEBIAN-35 MINGW64 ~/git2 (master)
$ git commit --amend -m "Updated commit message"
[master 423968e] Updated commit message
 Date: Mon Feb 24 15:35:52 2025 +0530
 1 file changed, 2 insertions(+)
 create mode 100644 file.txt

User@DEBIAN-35 MINGW64 ~/git2 (master)
$ git log --oneline
423968e (HEAD -> master) Updated commit message
```

Result:

Thus we successfully created a program to change history with amend using Git Command.

EX .NO:

DATE:

Create a script to perform local branch rebasing in Git

Aim:

To perform local branch rebasing in Git.

Algorithm:

1. Initialize a Git repository.
2. Create and switch to a new branch.
3. Make changes and commit them.
4. Switch back to the main branch and make changes.
5. Rebase the new branch onto the main branch.

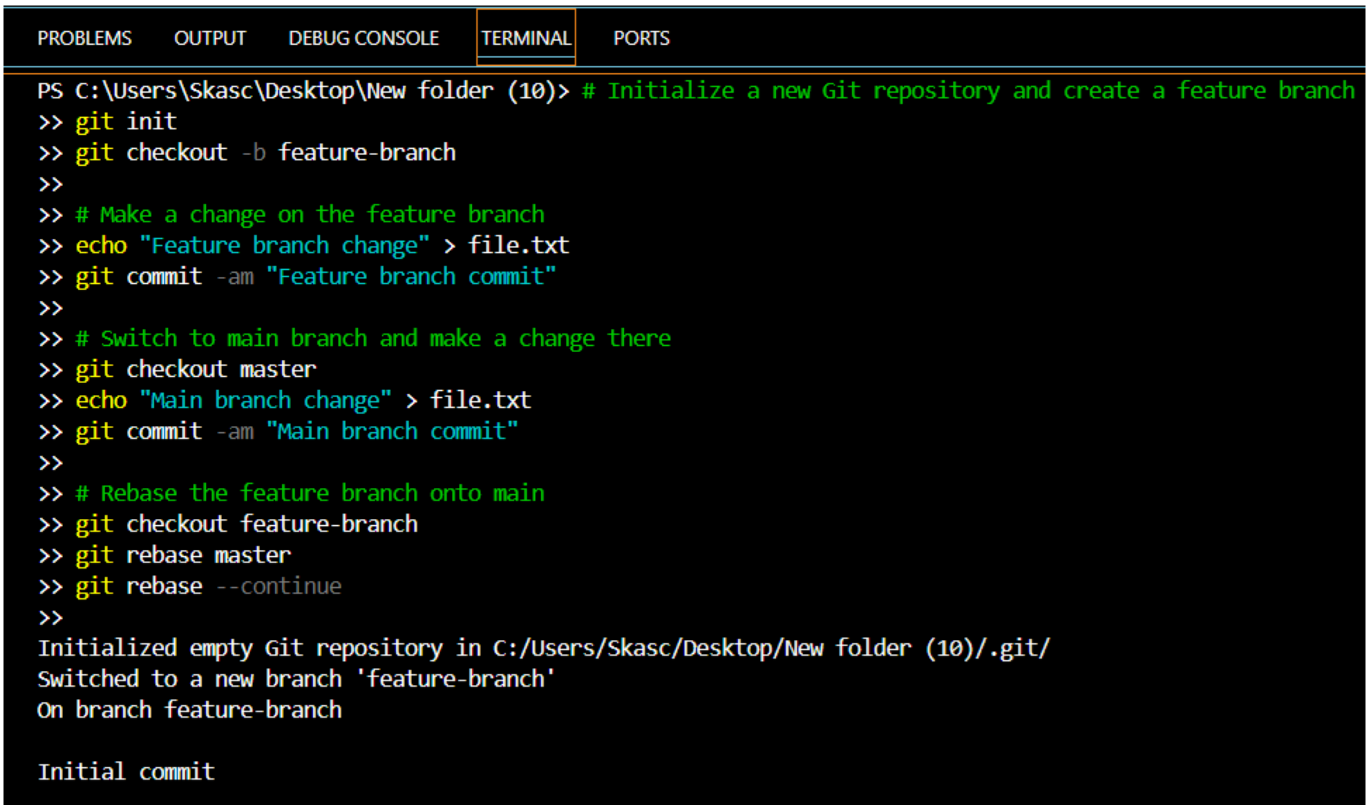
Commands:

```
git init
git checkout -b feature-branch
echo "Feature branch change" > file.txt
git add .
git commit -m "Feature branch commit"
git checkout main
echo "Main branch change" > file.txt
git add .
git commit -m "Main branch commit"
git checkout feature-branch
git rebase main
git add .
git rebase --continue
```

EX .NO:

DATE:

Output:



```
PS C:\Users\Skasc\Desktop\New folder (10)> # Initialize a new Git repository and create a feature branch
>> git init
>> git checkout -b feature-branch
>>
>> # Make a change on the feature branch
>> echo "Feature branch change" > file.txt
>> git commit -am "Feature branch commit"
>>
>> # Switch to main branch and make a change there
>> git checkout master
>> echo "Main branch change" > file.txt
>> git commit -am "Main branch commit"
>>
>> # Rebase the feature branch onto main
>> git checkout feature-branch
>> git rebase master
>> git rebase --continue
>>
Initialized empty Git repository in C:/Users/Skasc/Desktop/New folder (10)/.git/
Switched to a new branch 'feature-branch'
On branch feature-branch

Initial commit
```

Result:

Thus we successfully created a script to perform local branch rebasing in Git.

EX .NO:

DATE:

Create a rough copy of a repository using Git Fork Command

Aim:

To create a rough copy of a repository using the Git fork command.

Algorithm:

1. Open GitHub and navigate to the repository you want to fork.
2. Click the Fork button in the top-right corner.
3. Wait for GitHub to create a copy of the repository in your account.
4. Clone the forked repository to your local system.
5. Navigate into the cloned repository directory.

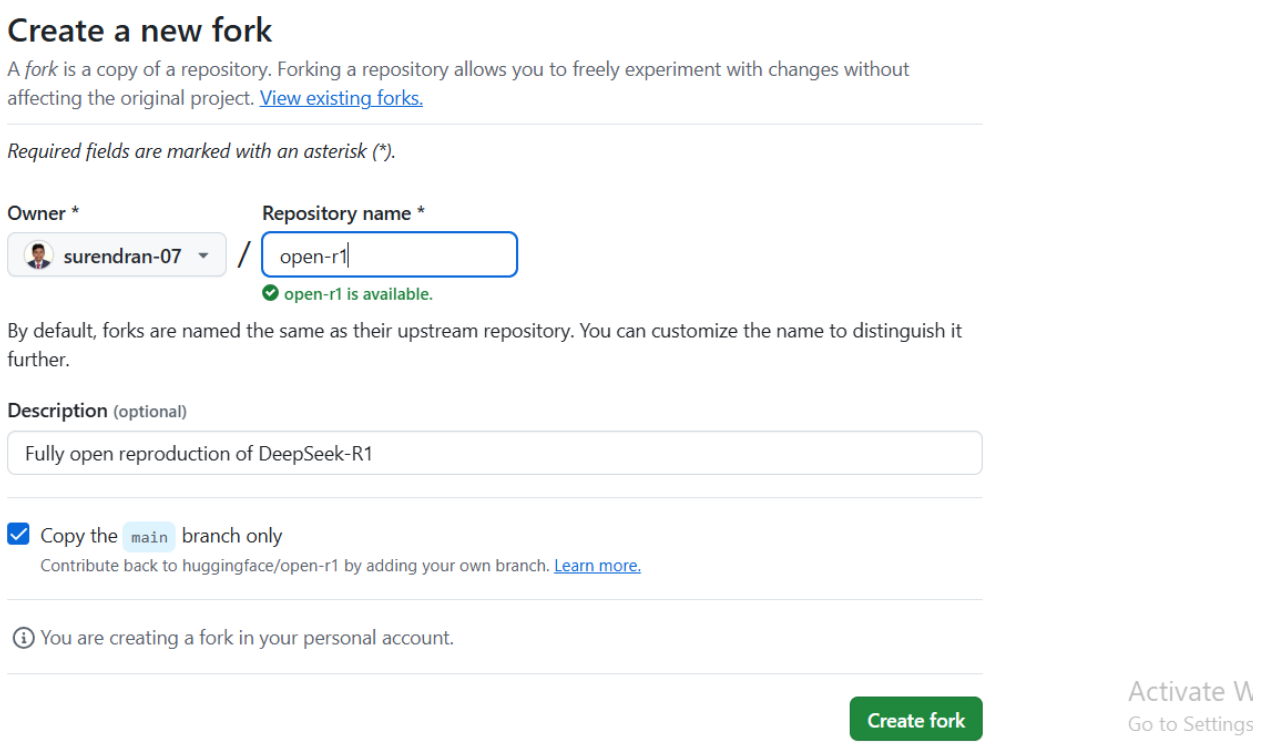
Commands:

```
git clone https://github.com/your-username/forked-repo.git
cd forked-repo
```

EX .NO:

DATE:

Output:



Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. View existing forks.

Required fields are marked with an asterisk (*).

Owner * surendran-07 / Repository name * open-r1

open-r1 is available.

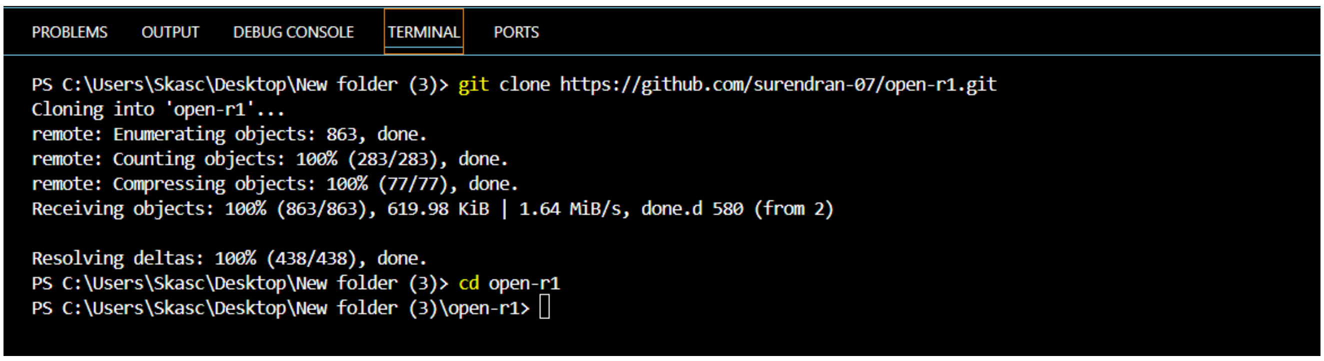
By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional): Fully open reproduction of DeepSeek-R1

[✓] Copy the `main` branch only

Contribute back to huggingface/open-r1 by adding your own branch. Learn more.

You are creating a fork in your personal account.



```
PS C:\Users\Skasc\Desktop\New folder (3)> git clone https://github.com/surendran-07/open-r1.git
Cloning into 'open-r1'...
remote: Enumerating objects: 863, done.
remote: Counting objects: 100% (283/283), done.
remote: Compressing objects: 100% (77/77), done.
Receiving objects: 100% (863/863), 619.98 KiB | 1.64 MiB/s, done.d 580 (from 2)

Resolving deltas: 100% (438/438), done.
PS C:\Users\Skasc\Desktop\New folder (3)> cd open-r1
PS C:\Users\Skasc\Desktop\New folder (3)\open-r1>
```

Result:

Thus we successfully created a rough copy of a repository using Git Fork Command.

EX .NO:

DATE:

Create a program to perform Git Remote Branching with Github.

Aim:

To perform Git remote branching with GitHub.

Algorithm:

1. Clone a GitHub repository.
2. Create and switch to a new branch.
3. Commit changes.
4. Push the new branch to GitHub.

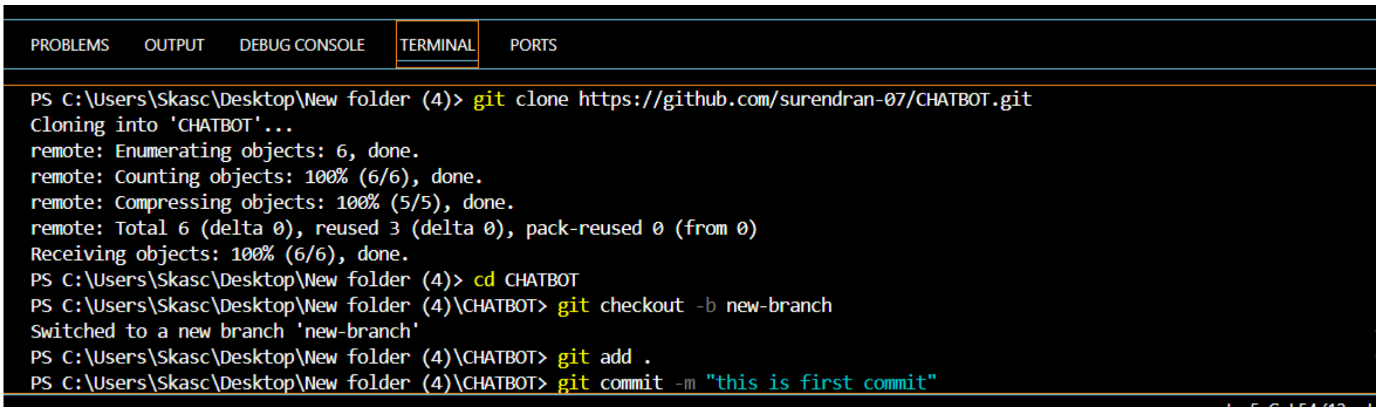
Commands:

```
git clone https://github.com/your-username/repository.git

cd repository

git checkout -b new-branch

git add .

git commit -m "New branch commit"

git push origin new-branch
```

EX .NO:

DATE:

Output:



```
PS C:\Users\Skasc\Desktop\New folder (4)> git clone https://github.com/surendran-07/CHATBOT.git
Cloning into 'CHATBOT'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
PS C:\Users\Skasc\Desktop\New folder (4)> cd CHATBOT
PS C:\Users\Skasc\Desktop\New folder (4)\CHATBOT> git checkout -b new-branch
Switched to a new branch 'new-branch'
PS C:\Users\Skasc\Desktop\New folder (4)\CHATBOT> git add .
PS C:\Users\Skasc\Desktop\New folder (4)\CHATBOT> git commit -m "this is first commit"
```

Result:

Thus we successfully created a program and performed Git Remote Branching with Github.

EX .NO:

DATE:

Create a copy of a specific repository or branch within a repository using Git Clone Command

Aim:

To create a copy of a specific repository or branch within a repository using the git clone command.

Algorithm:

1. Open Git Bash or a terminal.
2. Clone the entire repository from GitHub.
3. Navigate into the cloned repository directory.
4. List all available branches in the repository.
5. Switch to a specific branch if required.
6. Clone only a specific branch if you do not need the entire repository.

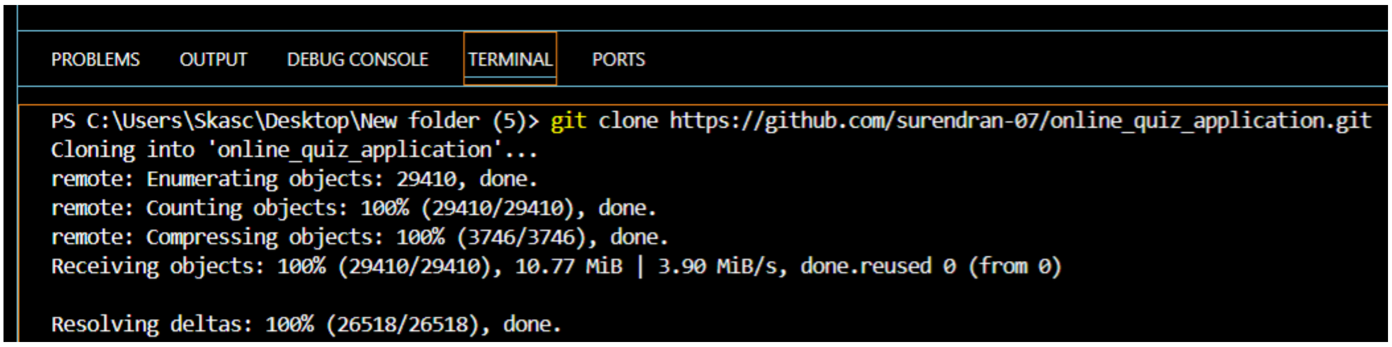
Commands:

```
git clone https://github.com/user/repository.git
cd repository

git clone -b branch-name https://github.com/user/repository.git
cd repository
```

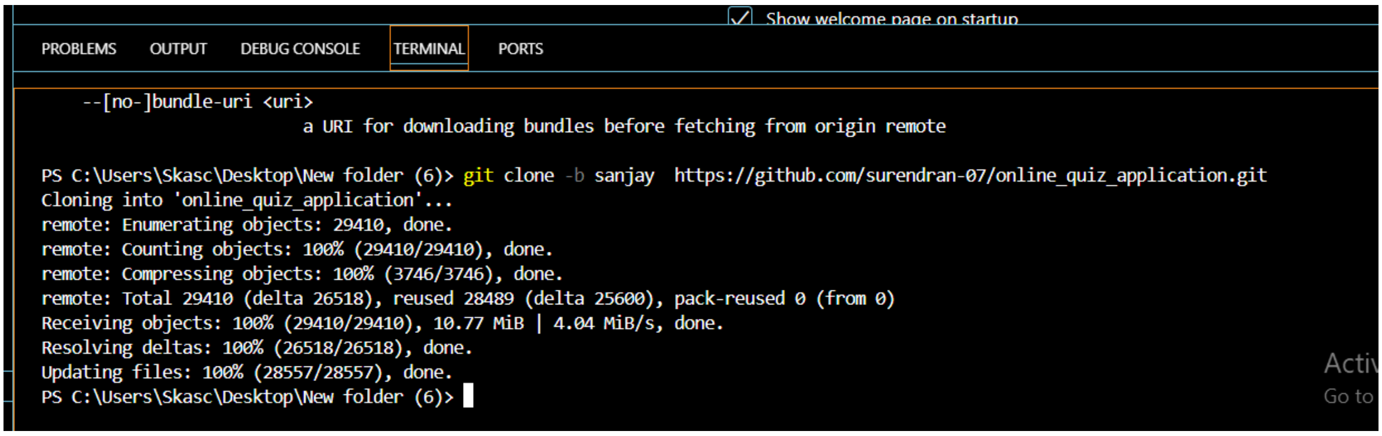
EX .NO:

DATE:

Output:



```
PS C:\Users\Skasc\Desktop\New folder (5)> git clone https://github.com/surendran-07/online_quiz_application.git
Cloning into 'online_quiz_application'...
remote: Enumerating objects: 29410, done.
remote: Counting objects: 100% (29410/29410), done.
remote: Compressing objects: 100% (3746/3746), done.
Receiving objects: 100% (29410/29410), 10.77 MiB | 3.90 MiB/s, done.reused 0 (from 0)

Resolving deltas: 100% (26518/26518), done.
```



```
    --[no-]bundle-uri <uri>
                          a URI for downloading bundles before fetching from origin remote

PS C:\Users\Skasc\Desktop\New folder (6)> git clone -b sanjay  https://github.com/surendran-07/online_quiz_application.git
Cloning into 'online_quiz_application'...
remote: Enumerating objects: 29410, done.
remote: Counting objects: 100% (29410/29410), done.
remote: Compressing objects: 100% (3746/3746), done.
remote: Total 29410 (delta 26518), reused 28489 (delta 25600), pack-reused 0 (from 0)
Receiving objects: 100% (29410/29410), 10.77 MiB | 4.04 MiB/s, done.
Resolving deltas: 100% (26518/26518), done.
Updating files: 100% (28557/28557), done.
PS C:\Users\Skasc\Desktop\New folder (6)>
```

Result:

Thus we successfully created a copy of a specific repository or branch within a repository using Git Clone Command.