**GitHub Commands**

**First Steps**

echo "# Angular-Develop" >> README.md

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin https://github.com/KamalSaru/Angular-Develop.git

git push -u origin main

**Push File**

1. Go file directory---(eg OneDrive\Desktop\IntelliJ Java\FullStackProject\TicketBookingSystemBack)
2. git init
3. git add . (for all file) and git add (file name for single)
4. git commit -m "first commit"
5. git branch -M main (change branch master or main)
6. git remote add origin + (<https://github.com/KamalSaru/FullStackProject.git>)
7. git pull (before push)
8. git push -u origin main/master

**Pull File**

1. git clone------downloads an existing Git repository to your local computer

**Push After Upadate**

1. Go to the directiory file(eg. OneDrive/Desktop/IntelliJ Java/FullStackProject/TicketBookingSystemBack) on gitbash or terminal
2. Write code or change in project first
3. Check status on gitbas (It’s show you any changes)
4. git diff+ (paste modified link)

Eg. src/app/pages/customer-list/customer-list.component.html

1. git stash (If you don’t want push or hide your change)
2. Git stash pop (Unhide or see your changes)
3. Git add . (For all change push)
4. Git add +(Sourc file dir)--- for one change push

Eg. src/main/java/com/diyo/TicketBookingSystem/entity/AdminLogin.java

1. Git commt –m ‘commit’---(Give meaningful comments)
2. Git status (It should remove the red line)
3. Git pull (Before push because we need original)
4. Git push –u origin main/master
5. Check on Github and we can see commit

**Merge Branch**

* Work all the different one branch like ---**webpage**
* Push all the update on same branch.
* Git checkout to the Main Branch
* Git merge **branch name**
* Git commt –m ‘commit’---(Give meaningful comments)
* Git status (It should remove the red line)
* Git pull (Before push because we need original)
* Git push –u origin main/master
* Check on Github and we can see commit

**Conflict Fix & Merge**

* Right click on Project File on Intellij/Working place
* Go to Intelij/Or other tools
* Right click on it
* Go to resolve file conflict
* Drag all changes to the project

**Remove untracked files**

* Git clean –n -n flag is used to perform dry run
* Git clean –nd -nd only list the files, to list down the folders use
* Git clean –f -f flag is used to remove untracked files
* Git clean - -fd -fd flag is used to remove untracked files and folders
* Git clean - -fx -fx flag is used to remove untracked and ignored files
* Git clean --x
* Git reset - -hard --Remove modified file on git

**Clone Repository/Zip Download File**

* Go to Location to save file in git
* Git clone **Repository** (provided from the company)

Eg. git clone <https://github.com/KamalSaru/TicketBookingSystemFront.git>

**Don’t’ Want To Update Or Small Change**

* Git checkout file source
* git checkout src/main/java/com/example/resturantcurd/Menu.java
* OR you can do Undo from working place

**UseFul Git Command**

* Git branch
* Git branch –a ( pull all branch)
* Git checkout userprofile
* Git branch main (go first to main for merge)
* Git merge userprofile
* git checkout -b homepage (create branch from git bash, -b make branch and switch)
* git pull origin **webpage**(If you create from website pull the branch first)
* git checkout **webpage** (If you create branch from github webside)
* git branch **homepage** **main** (create new branch under main branch)
* git push –u origin **homepage** (push branch then check on website)
* git push origin homepage (After create push to remote)
* git branch -D develop (Delete branch, under line is branch name)
* git pull ---before merge
* git branch checkout-----to main—go before main or need to merge branch
* git merge **develop**  ---(merge develop branch to main)

**Most common/frequently used git commands:**

1. git init-----create new git repository in the exact file directory location
2. pwd --------------for check working directory
3. git clone------downloads an existing Git repository to your local computer
4. git status----displays the state of the working directory and the staging area
5. git diff----takes two input data sets and outputs the changes between them.
6. git add----adds a change in the working directory to the staging area
7. git remote add----create a new connection record to a remote repository
8. git commit -m ‘ ’ ---message for upadate
9. git push-----upload local repository content to a remote repository
10. git log------exploring a repository's history
11. git branch-----pointer to a snapshot of your changes, branch list
12. git branch –a------git branch directory
13. git branch webpage-----create new branch
14. git stash---temporarily shelves (or stashes) changes you've made to your working file copy
15. git stash pop----helps us to remove or throw away the latest or the topmost stash
16. git stash apply---- emporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on
17. git pull------ used to fetch and download content from a remote repository and immediately update the local repository to match that content
18. git restore –staged----- the file will only be removed from the Staging Area - but its actual modifications will remain untouched
19. git checkout-------- lets you navigate between the branches created by git branch
20. git switch----- allows you to switch your current HEAD branch
21. git merge----- lets you take the independent lines of development created by git branch and integrate them into a single branch
22. git checkout –b---Creating branche
23. git cherry-pick---- the act of picking a commit from a branch and applying it to another. git cherry-pick can be useful for undoing changes.