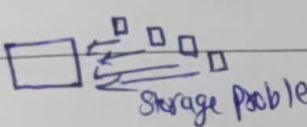
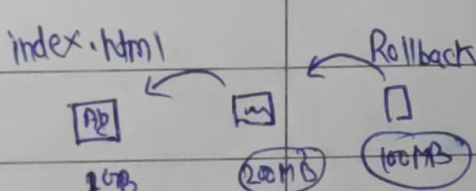
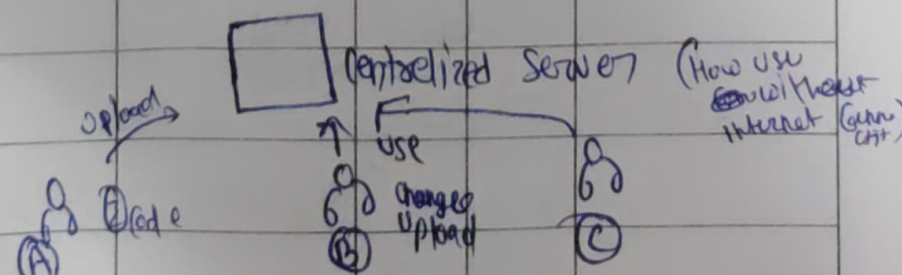


Today's Date: _____

GIT

Sr. No.	Time	Patient Name	Contact No.	SpO2	Temp (°C)	Remarks
1		GIT ⇒ Distributed Version Control System.				
2		requirement (is q) why?				
3		(Developer) Storage problem → Resolve, local version control system develop				
4						
5		local version control system :-				
6		- Storage problem resolve				
7		<u>Ex</u>				
8						
9						
10		<u>Problem's in VCS:-</u>				
11		1. Collaboration (only manual)				
12		2. Data lose (dev. system store)				
13		CVCS VCS solution				
14						
15		CVCS (Centralized version control system) ↓				
16						
17						
18						
19						
20		Problem in CVCS ⇒ 1. Internet Connectivity				

Today's Date:

Sr. No.	Time	Patient Name	Contact No.	SpO2	Temp (°C)	Remarks
1		Solution of DVCS				
2		DVCS (Distributed Version Control System)				
3		Ex				
4		<u>GIT</u>				BitKeeper
5						
6		Code →				Centralized Server
7						
8		Code (Client) L1				Code (Client) L2
9						Code (Client) L3
10						Code (Client) L4
11		GIT → open source				
12		more linux towards				
13						
14		GITHUB				
15		BITBANE				Centralized server / website / Public Cloud / Developer
16		<u>GIT Benefits</u>				
17						
18		1. Work on snapshot				(Point in time backup / image)
19		2. Every operation is local				
20						

Today's Date:

Sr. No.	Time	Patient Name	Contact No.	SpO2	Temp (°C)	Remarks
1		3. Git integrity.				* Git work on checksum value / #value Command → sha → sha256sum let's pass
2		4. Git Generally only add data				Git id (8d532ab2de) # value show (let's pass) check uses field abc (changes in system) check again sha256sum let's pass
3						
4		Introduction				
5		→ Git is a DevOps tool used for Source Code management				
6		→ Free Free open source, work on small to big project				
7		→ Git is example of DVCS.				
8		→ Git tools like :- AWS CodeCommit, Mesoslogal, Helix Core.				
9		→ work on linux/windows/unix				
10		→ platform independent				
11						
12		git --version (check version)				
13						
14		START				
15		1. mkdir NETFLIX				
16		2. cd NETFLIX				
17		3. git status (show git repository / status)				
18		4. git init (for initialize into .git)				
19		↓				
20		ls -al				(.git) (directory)

Sr. No.	Time	Patient Name	Contact No.	SpO2	Temp (°C)	Remarks
1						
2		vim index.css → (hello)		*	<u>Global</u> <u>Developer</u> <u>master</u>	
3		touch index.html		*	↓ → <u>git status</u>	
4		git status		*	<input type="checkbox"/> <u>Refit</u> → [Blog] <u>Feature branch</u>	
5		git add . (staging area)				
6		git status				
7						
8		touch chat.txt				
9		git status				
10		git add chat.txt				
11						
12		(<u>create</u>) git commit -m "my first commit"				
13		(<u>all version</u>) git log				
14						
15						
16		git config user.name "PRINCE NARULA"				
17		git config user.email "surya.narula.32@gmail.com"				
18		→				
19		for configuration of name & email				
20						

Repository → Collection of commit, history of code.

Today's Date:

Sr. No.	Time	Patient Name	Contact No.	SpO2	Temp (°C)	Remarks
1		git config --global user.name				"PRINCE NARULA"
2		git config --global user.email				"surya.narula.32@gmail.com"
3						
4		(Repository) <u>GitHub</u>				(<u>Public</u>) <u>Repository</u> of <u>github</u> , copy history
5		↓				into linux cli.
6		local system				
7						git clone <u>repo link</u>
8						Is (from github)
9						<u>same repository</u>
10						git status
11		* In this only download				git log
12		repo. not uploading, require				
13		id pass.				git clone <u>link</u> <u>my project</u>
14						Set on your local
15		local system (repository)				
16		Push ↓				
17		GitHub				
18						
19		⇒ git remote add <u>origin</u> <u>link</u> (github new repository)				
20		<u>any name</u>				

- git remote
git remote -v

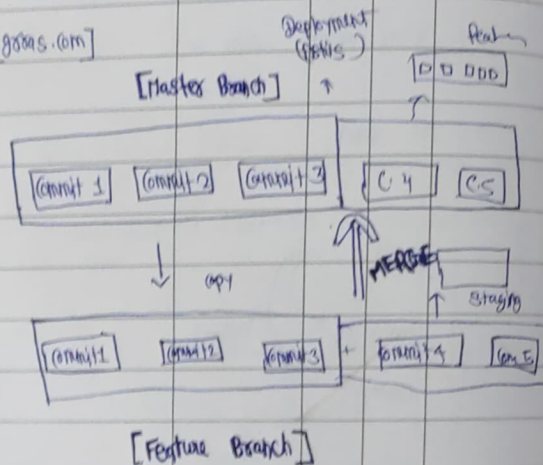
Sr. No.	Time	Patient Name	Contact No.	SpO2	Temp (°C)	Remarks
1		vim .git/Config	[git remote -v]			
2		↓				
3		git push origin master				
4		any name				
5		↓				
6		Username :				
7		Password : token generate				
8		(profile → setting → Developer setting → Personal access token)				
9		→ Generate new token → (Generate token)				
10						
11		Password share →	git config --global credential.helper store			
12		check where password share				
13		vim ~/.gitconfig				
14						
15		<u>RESET :</u>				
16		1. soft	Commit 1 ✓1			
17		2. mixed	Commit 2 ✓2			
18		3. hard	Commit 3 ✓3			
19						
20						

Sr. No.	Time	Patient Name	Contact No.	SpO2	Temp (°C)	Remarks
1		1. git reset --soft HEAD~1				(code)
2		(small project single file)				(commit area remove available in staging, WIP)
3						(code)
4		2. git reset --mixed				(commit, staging area remove available in WIP)
5		(big project, multiple file)				
6						(code)
7		3. git reset --hard				(commit, staging & WIP + remove area)
8						
9		head ~ 1 (final commit) shift				
10		Commit !				
11		Commit !				
12		Commit !				
13						
14		cd /usr/local				
15		↳ first second third				
16		create two log files				
17		first second third abc.log xyz.log				
18		(staging, commit)				
19		Ignore log files & other files add.				
20		vim .gitignore				

abc.log
xyz.log * .log

Today's Date: _____

Sr. No.	Time	Patient Name	Contact No.	SpO2	Temp (°C)	Remarks
1		<u>BRANCHES</u>				
2						
3		BRANCHES IN GIT				
4						
5		Journal (owner)	[github.com]			
6						
7			[Master Branch]			
8		Journal (owner)				
9		Journal (owner)				
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						



git branch (create branch)

git checkout -b krf2 (branch create & checkout from master branch)

- git branch krf2
- git checkout krf2 (switch into branches)

git branch

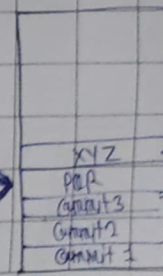
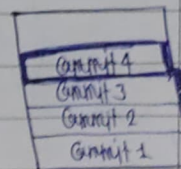
Today's Date: _____

Sr. No.	Time	Patient Name	Contact No.	SpO2	Temp (°C)	Remarks
1		15				
2		(Same data folders show as master branch)				
3		vim file1 add, commit log				
4		vim file2 add, commit log				
5		git log				
6		git checkout master				
7		git log				
8		git checkout krf2 (all ok with krf2 branch) then				
9		git log				
10		git checkout master				
11		git merge krf2 (merging)				
12						
13						
14						
15						
16						
17						
18						
19						
20						

Rebase / Squash

Rebase

[MASTER]



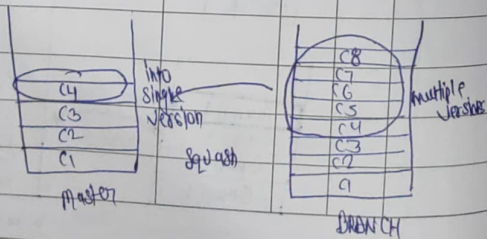
both available after rebase
-> Commit 4 add

rebase

In case of two branching both are not available

Today's Date: _____

Sr. No.	Time	Patient Name	Contact No.	SpO2	Temp (°C)	Remarks
1		git xbase master				
2						
3		After Master ↓ (add new file/folder)		BRANCH (also add some file/folder)		
4			↓	use <u>xbase</u>		
5						
6						
7		git xbase master	[master branch] [BRANCH 1]			
8						
9		git merge BRANCH1	[master branch] (master branch 2/3)			
10			[master branch] Command run			
11						
12						
13						
14		<u>Squash</u>				
15						
16						
17						
18						
19						
20						



Today's Date: _____

Sr. No.	Time	Patient Name	git checkout MASTER	Contact No.	SpO2	Temp (°C)	Remarks
1		git merge --squash BRANCH1					(MASTER BRANCH)
2		↓					(commit [BRANCH] multiple version)
3		(staging area show)					↓
4		git status ↑					[MASTER branch] single commit (version)
5		git commit -m "final Commit By Branch 1"					
6							
7		<u>merge Conflicts</u>					
8							
9							
10		master branch push ✓					
11							
12		BRANCH PUSH ↓					
13		git checkout BRANCH					
14							
15		git push origin BRANCH					
16							
17		In Github (remote server)					
18		merge master & branch ↓					
19							All request
20							then merge together

In Github merge only in remote server not in CLI.

Today's Date: _____

Sr. No.	Time	Patient Name	Contact No.	SpO2	Temp (°C)	Remarks
1		Fetch				When merge in github & you take merge
2		Pull				← on cli, How?
3						
4		1. <u>Fetch</u>				(Fetch + manually merge)
5						
6		git fetch				(Code ^(Fetch) come from github (master branch))
7						[Code Comp (Commit/Version)]
8		git merge origin/master				
9		git log				
10						
11		2. <u>Pull</u>				(Fetch + merge)
12						
13		git pull origin master				
14						
15						
16		<u>Reset & Revert</u>				
17		After Resh Revert must use				
18						git revert Commit1
19						(Commit Revert)
20						Commit 3 (not del) Commit 2 Commit 1

Today's Date: _____

Sr. No.	Time	Patient Name	Contact No.	SpO2	Temp (°C)	Remarks
1		Revert → Commit not del.				
2		Reset → Commit del.				
3						(other)
4		Copy → fork				(In Github repository copy and fork)
5						
6		git send to hashil				
7		git pull				Post then fork in hashil github account
8						
9		Git → Gih hub				(fork)
10						
11		↓				
12		hashil add file in this Repository				(but not change in gaurav)
13		↓				
14		Say to gaurav please add this changes				
15		(Pull request)				
16		(hashil) → contribute				
17		↓				
18		open pull request				
19		then file (pull) request go to gaurav Github				
20		↓				

me	Patient Name	Contact No.	SpO2	Temp (°C)	Remark
	gaugan github :-	refresh <div style="border: 1px solid black; padding: 2px;">Pull req. ①</div> ↑ click then View			
		<div style="border: 1px solid black; padding: 2px;">Confirm merge</div> refresh			
		<div style="border: 1px solid black; border-radius: 50%; padding: 5px; display: inline-block;">merged</div> <u>done</u>			
	<u>hook's</u>				
	<pre> # cd .git/ ↓ ls ↓ hook files (• sample file) cd hooks / ↓ vim post-commit ↓ [git push -u HERO master] ↓ chmod a+x post-commit </pre>				

第 1 页

2

19

→

book files (• sample file)
cd books ↓	

```
vim post-commit
```

↓

```
[git push -u HERO master]
```

↓

chmod a+x post-commit