WEEK 8

Mandatory Hands-On

**GIT and Version Control**

**.gitignore and its functionality**

1. What is .gitignore in Git?

* Definition: A .gitignore file tells Git which files or folders to ignore (not track) in a repository.
* Purpose: Prevents unwanted files—like temporary build files, logs, OS-specific files, or personal configs—from being pushed to the remote repository.
* Example: You don’t want .env (with passwords) or node\_modules/ (big dependency folder) to be uploaded to GitHub.

2. How .gitignore works

* Git checks the .gitignore file to see if a file/folder matches any pattern inside it.
* If it matches, Git does not track it.
* Important: If a file was already tracked before being added to .gitignore, you must remove it from Git tracking first.

3. Creating a .gitignore file

1. In your Git repo, create a file named:
2. Add patterns for the files/folders you want to ignore.

4. Examples of .gitignore patterns

| Pattern | Meaning |
| --- | --- |
| node\_modules/ | Ignore the entire folder named node\_modules |
| \*.log | Ignore all .log files |
| secret.env | Ignore a single file named secret.env |
| \*.tmp | Ignore all files ending with .tmp |
| build/ | Ignore the build output folder |
| !important.log | Do NOT ignore this specific file, even if it matches other rules |

5. Ignoring files already tracked by Git

If a file is already committed and you later add it to .gitignore, it will still be tracked until you remove it from tracking:

git rm --cached filename

Example:

git rm --cached secret.env

6. Sample .gitignore for a Node.js project

# Dependencies

node\_modules/

# Logs

\*.log

# Environment variables

.env

# Build output

dist/

# OS-specific files

.DS\_Store

Thumbs.db

**Sample Output:**



