Experiment No. : ..... Date: ..... Page No.: .....

Name of the Experiment : .....

Case Study 7: Practical Git Workflow.

a. Demonstrate the process of initializing a Git repository. Create a new file. Stage it, commit the changes and display the commit history using relevant commands.

b. Perform and explain Git branching and merging operations. Include Practical examples and discuss common real-world use cases.

Title of the Experiment:-

Practical Implementation of Git Workflow: Repository Initilization, Branching and Merging

Aim:-

To understand and demonstrate the practical use of Git in software development by initializing a Git repository committing changes, & performing branching and merging operations.

Theory:

Git is a distributed Version Control System that helps developers track changes in source code during development. Git allows individual and team collaboration by managing versions of files and enabling workflows such as branching, merging, and remote pushing.

Experiment No. : ..... Date: ..... Page No.: .....

Name of the Experiment : .....

Key Concepts :

- Repository :- A directory tracked by Git.
- Commit :- A snapshot of your changes.
- Staging :- Preparing files for a commit.
- Branch :- A separate line of development.
- Merge :- Integrating changes from one branch into another.

Benefits of Git :

- Enables parallel development through branches.
- Maintains a complete history of the project.
- Provides collaboration tools for teams and open source.
- Supports rollback to previous versions if bugs are found.

a. Git Repository Initialization, File commit, & History.

Steps with commands Explanations.

Step 1 :- Create a new project folder.
command :- mkdir my-git-demo && cd my-git-demo.

- Set up a working directory.

Step 2 : Initialize a git repository
command :- git init

- Creates an empty .git folder for tracking

Experiment No. : ..... Date: ..... Page No.: .....

Name of the Experiment : .....

Step 3: Create a new file
Command :- echo "Welcome to git" > hello.txt
• Adds content to a file.

Step 4: Check Git Status
Command : git status
• Shows understand or untracked file hello.txt.

Step 5: Stage the file
command :- git add hello.text
• Prepares the file for commit.

Step 6 :- Commit the change
Command :- git commit -m "Initial commit with hello.txt"
• Records the change with a message

Step 7:- Display commit history
Command :- git log
• Shows commit ID, author, date and message.

b. Git Branching and Merging operations

Branching & Merging - Practical Workflow.

Step 1:- Create a Branch
Command :- git branch feature-1
• Creates a new branch for a feature

Step 2:- Switch to the branch.
command :- git Checkout feature-1

Experiment No. : ..... Date: ..... Page No.: .....

Name of the Experiment : .....

- Moves to the feature-1 branch.

Step 3:- Edit the file
Command: echo "this is feature 1" >> hello.txt.
- Adds content to simulate feature work.

Step 4:- Add and commit changes
Command: git add hello.txt + git commit. -m "Add feature 1"
- goes back to the main branch / Save work in. feature branch.

Step 5:- Ad Switch back to main
Command:- git checkout main.
- goes back to the main branch.

Step 6:- Merge feature branch.
Command: git merge feature-1
- Integrates from feature-1 into main.

Common Real-World Use Cases:

Usecase.	Explanation
• Feature Development	Each developer creates a branch per feature
• Bug fixing.	Bugs are fixed on Separate branches and merged after testing
• Code Reviews	Branches are pushed to GitHub & reviewed before merging
• Release Management	Create branches like release./v1.0 for production releases

Experiment No. : ..... Date: ..... Page No.: .....

Name of the Experiment : .....

Conclusion

By practicing Git commands, I understood importance of Version Control in software development. Gits branching and merging features make collaboration & code safety much easier.