

SET-222

Software Operations & Maintenance

Experiment # 02

**Experiment Title**

**Advanced Git and GitHub Operations**

**Assessment of CLO(s): 03**

**Performed on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name:** |  | | |
| **Roll No.** |  | **Group** |  |
| **Semester** |  | **Session** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S. No.** | **Perf. Level**  **Criteria** | **Excellent**  **(2.5)** | **Good**  **(2)** | **Satisfactory**  **(1.5)** | **Needs Improvement**  **(0 ~ 1)** | **Marks Obtained** |
| **1** | Project Execution & Implementation | Fully functional, optimized, and well-structured. | Minor errors, mostly functional. | Some errors, requires guidance. | Major errors, non-functional, or not Performed. |  |
| **2** | Results & Debugging  Or Troubleshooting | Accurate results with effective debugging  Or Troubleshooting. | Mostly correct, some debugging Or Troubleshooting needed. | Partial results, minimal debugging  Or Troubleshooting. | Incorrect results, no debugging Or Troubleshooting, or not attempted. |  |
| **3** | Problem-Solving & Adaptability  (VIVA) | Creative approach, efficiently solves challenges. | Adapts well, minor struggles. | Some adaptability, needs guidance. | Lacks innovation or no innovation, unable to solve problems. |  |
| **4** | Report Quality & Documentation | Clear, structured, with detailed visuals. | Mostly clear, minor gaps. | Some clarity issues, missing details. | Poorly structured, lacks clarity, or not submitted. |  |
| **Total Marks Obtained Out of 10** | | | | | |  |

**Experiment evaluated by**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instructor’s Name** | **Ms. Shagufta Aftab** | | |
| **Date** |  | **Signature** |  |

## Copyright © Department of Engineering & Technology – UIT University Karachi

**Objective:**

* To understand and resolve merge conflicts in Git.
* To learn advanced Git operations like rebasing and cherry-picking.
* To explore Git hooks and automation for streamlined workflows.

### **Theory:**

Git is a distributed version control system that enables collaboration and code tracking. Advanced Git operations help developers manage complex workflows efficiently.

**Key Concepts:**

1. **Merge Conflicts** – Occurs when two branches have conflicting changes.
2. **Git Rebase** – Moves a feature branch to the latest commit of the main branch.
3. **Cherry-Picking** – Selectively apply specific commits from one branch to another.
4. **Git Hooks** – Scripts that trigger before or after Git events (e.g., pre-commit, post-merge).
5. **Git Automation** – Using GitHub Actions or scripts to automate repetitive tasks.

**Lab Exercise:**

**Exercise 1: Resolving Merge Conflicts**

1. Create a repository and initialize Git:

git init advanced-git-lab

cd advanced-git-lab

1. Create and switch to a new branch:

git checkout -b feature-branch

1. Modify a file and commit changes:

echo "Feature update" > file.txt

git add file.txt

git commit -m "Added feature update"

1. Switch back to the main branch and make another change:

git checkout main

echo "Main branch update" > file.txt

git commit -am "Updated file in main branch"

1. Merge feature-branch into main:

git merge feature-branch

6. Resolve conflicts manually, then commit the resolved changes.

**Exercise 2: Git Rebase**

1. Switch to the feature branch:

git checkout feature-branch

1. Rebase onto the latest main branch:

git rebase main

1. If conflicts occur, resolve them and continue:

git rebase --continue

1. Push the rebased branch:

git push origin feature-branch --force

**Exercise 3: Cherry-Picking Commits**

1. Identify a commit hash from git log.
2. Apply a specific commit to another branch:

git cherry-pick <commit-hash>

**Exercise 4: Implementing Git Hooks**

1. Navigate to the hooks directory:

cd .git/hooks

1. Create a pre-commit hook:
2. echo "echo 'Checking code style'" > pre-commit

chmod +x pre-commit

1. Test by making a commit.

**Exercise 5: Automating Git Processes**

1. Set up a GitHub Action to run tests on push.
2. Create a .github/workflows/main.yml file and add:

name: CI Pipeline

on: push

jobs:

test:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Run Tests

run: echo "Running tests..."

3.Commit and push to trigger automation.

### **Assessment & Exercises:**

1. Explain the purpose of Git rebase and how it differs from merge.
2. How can you resolve a merge conflict manually?
3. What is the use of git cherry-pick, and when should it be used?
4. Implement a Git pre-commit hook that prevents committing large files.
5. Configure a GitHub Action workflow for automated code testing.

**Conclusion:**

In this lab, students practiced resolving merge conflicts, rebasing, cherry-picking, and automating Git processes using hooks and GitHub Actions. Mastering these operations enhances collaboration and streamlines software development.