School: ..... Campus: .....

Academic Year: ..... Subject Name: ..... Subject Code: .....

Semester: ..... Program: ..... Branch: ..... Specialization: .....

Date: .....

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiement : Team Dev – Git and Collaboration in Projects

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

1. Aim

To understand **Git version control system**, its collaborative features, and implement team-based project development using **GitHub/GitLab**, including branching, merging, and resolving conflicts.

2. Theory

Git:

Git is a distributed version control system (DVCS) used to track changes in source code during software development. It allows multiple developers to collaborate efficiently.

Key Concepts:

1. **Repository (Repo):** Central storage for a project's files and history.
2. **Clone:** Copying a remote repository to a local machine.
3. **Commit:** Saving changes to the local repository with a message.
4. **Branch:** Parallel version of a repository for independent development.
5. **Merge:** Integrating changes from one branch into another.
6. **Pull & Push:** Pull updates from remote repository; push local commits to remote.
7. **Conflict Resolution:** Handling differences when multiple changes overlap.
8. **Fork & Pull Request:** Collaborating on public projects via personal copies and merge requests.

Benefits of Git in Team Development:

- Tracks history of every file.
- Enables **parallel development** using branches.
- Facilitates **collaboration and code review**.
- Provides **backup and recovery** mechanisms.
- Reduces risk of overwriting changes in multi-developer projects.

* Softwares used

1. **Git** – Version control system
2. **GitHub / GitLab / Bitbucket** – Online repository hosting
3. **VS Code / Sublime / Atom** – Code editor
4. **Git Bash / Terminal / CMD** – Command line interface
5. **Web Browser** – For accessing repositories and collaboration

**As applicable according to the experiment.*
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)

Step 1: Setting up Git

```
# Install Git (if not already)
sudo apt-get install git    # Linux
brew install git            # macOS
```

Step 2: Configure Git

```
git config --global user.name "Your Name"
git config --global user.email "yourname@example.com"
```

Step 3: Initialize a Repository

```
mkdir TeamProject
cd TeamProject
git init
```

Step 4: Clone a Remote Repository

```
# Create a new feature branch
git checkout -b feature/login
```

Step 6: Make Changes and Commit

```
# Edit files using code editor
git add .
git commit -m "Added login functionality"
```

Step 7: Push Changes to Remote

```
git push origin feature/login
```

Step 8: Merge Branch

```
# Switch to main branch
git checkout main

# Merge feature branch
git merge feature/login
```

Step 9: Resolve Conflicts (if any)

```
git add <resolved-file>
git commit -m "Resolved merge conflict"
```

Step 10: Pull Latest Changes

```
git pull origin main
```

Step 11: Collaboration via Pull Request (GitHub)

1. Push your branch to GitHub.
2. Create a **Pull Request (PR)** to merge changes into main.
3. Team members review, comment, and approve PR before merging.

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Observations:

Action	Command / Feature	Result / Observation
Repository creation	git init	Local Git repo initialized
Clone repo	git clone <URL>	Local copy of remote repo
Branch creation	git checkout -b <branch>	New branch created for feature work
Commit changes	git add . && git commit -m "msg"	Changes saved in local repo
Push to remote	git push origin <branch>	Changes uploaded to GitHub
Merge branches	git merge <branch>	Feature integrated into main branch
Conflict resolution	Manual edits + git add	Conflicting files resolved
Pull latest changes	git pull origin main	Local repo updated with remote changes
Pull request workflow	GitHub PR	Team collaboration and review completed

*Conclusion

- **Git is essential** for team-based development, enabling multiple developers to work simultaneously.
- Branching and merging allow safe experimentation without affecting main codebase.
- Pull requests and code reviews improve code quality and collaboration.
- Version control ensures history tracking, backup, and easy conflict resolution in projects.
- Using Git improves **productivity, coordination, and project management** in team projects.

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Student:

Name :

Regn. No. :

Signature of the Faculty:

**As applicable according to the experiment. Two sheets per experiment (10-20) to be used.*