Centurion University
Shaping Lives... Empowering Communities...

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**

- Initialize Repository – Create a Git repo locally (git init) or clone an existing one (git clone).
- Create Branches – Each developer creates their own branch for features/bug fixes.
- Stage & Commit Changes – Developers make changes, then run git add . and git commit -m
- "message".
- Push to Remote – Upload changes to a shared remote repository (git push).
- Pull Updates – Regularly pull (git pull) to stay synced with the team's work.
- Merge/PR – Open a Pull Request (or Merge Request) to integrate feature branches into the main branch.
- Code Review – Team reviews, suggests changes, and approves the PR.
- Merge to Main – After approval, merge into the main/master branch.
- Resolve Conflicts – If multiple people change the same code, resolve conflicts manually before merging.
- Deploy/Release – Final tested code is released from the main branch.

Apparatus/Software Used:

1. MetaMask Wallet
2. Remix IDE.
3. MS Word.
4. Brave for researching.

T Implementation Phase: Final Output (no error) Applied and Action Learning

- Team creates a central repository (GitHub/GitLab).
- Developers clone it into their local systems.
- Each member works on separate branches (e.g., feature-login, bugfix-db).
- Developers push their branches to the remote repo.
- Pull Requests are created → Reviewed → Merged.
- The main branch always contains stable and updated code.
- Continuous Integration (CI) can run automated tests after merges.
- Final output: a well-maintained, collaborative, and version-controlled project.

Observation Table:

- Git enables seamless collaboration across distributed teams.
- Branching strategy avoids overwriting and ensures stable production code.
- Version control allows rollback if bugs appear in new updates.
- Merge conflicts highlight overlapping work, requiring coordination.
- Collaboration platforms (GitHub/GitLab) improve transparency and productivity.
- Git workflow is essential for team projects, hackathons, and open-source contributions.

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: