Experiment No. 5:

Implementing a Collaborative Workflow in a Team Project and Publishing Documentation using GitHub Pages

Aim:

To implement collaborative workflows in a team-based software project using GitHub and publish structured project documentation using GitHub Pages.

Objectives:

- ✓ To understand and apply collaborative development using Git and GitHub.
- ✓ To practice creating and managing branches following GitHub Flow.
- ✓ To write semantic and structured commit messages.
- ✓ To review, comment on, and merge pull requests in a team setting.
- ✓ To create technical documentation and publish it using GitHub Pages.

Required Tools:

- Git (installed locally)
- GitHub account (with a team repository)
- Code editor (e.g., VS Code)
- Internet browser for GitHub Pages deployment

Procedure:

Practical Conduction: Step-by-Step with Examples

Part A: Collaborative Workflow using GitHub Flow
Example Project: team-todo-app
A simple team project to build a **To-Do List Web App**.

Step 1: Create a GitHub Repository

- One team member creates a repository:
  https://github.com/username/team-todo-app
- Add team members as collaborators (in repo settings).

Step 2: Clone the Repository Locally
Each member runs:

git clone https://github.com/username/team-todo-app.gitcd team-todo-app

Step 3: Create a New Branch for Your Feature

git checkout -b feature-add-task

This follows the **GitHub Flow**:
Main always has working code → Create branch → Work → PR → Merge

Step 4: Make Changes in Your Branch
Edit/add project files, e.g.:
app.js
javascript

```
function addTask(task) {
   console.log("Task added:", task);
}
```

Step 5: Commit with a Proper Message

```
git add app.js
git commit -m "feat: add basic task adding functionality"
```

✅ **Best Practice**:

Use semantic commit messages:
feat:, fix:, docs:, style:, test:, refactor:

Step 6: Push to Remote Branch

```
git push origin feature-add-task
```

Step 7: Create a Pull Request (PR)

- ✓ Go to the repository on GitHub.
- ✓ GitHub shows a **Compare & Pull Request** banner.
- ✓ Add a title and description.
- ✓ Click **Create Pull Request**.

Step 8: Review and Merge PR

Other team members can:

- Review the code.
- Leave comments.
- Approve or request changes.

Once approved, click **Merge pull request**.

Step 9: Sync Everyone's Code

- git checkout main
- git pull origin main

Part B: Publishing Project Documentation using GitHub Pages

Step 1: Create a docs/ Folder

mkdir docs
Create docs/index.md:
markdown

Team To-Do App

Welcome to the Team Project!
Features- Add tasks- Mark tasks as complete- Remove tasks

Step 2: Commit and Push Docs

git add docs/index.md
git commit -m "docs: added initial documentation"
git push origin main

Step 3: Enable GitHub Pages

- Go to your repository on GitHub.
- Settings → Pages
- Source: main branch → /docs folder
- Click **Save**

GitHub Pages URL will be generated, e.g.:

https://username.github.io/team-todo-app/

Additional Concepts (Theory)

Collaborative Workflows:

Git Flow	GitHub Flow
Uses multiple branches like develop, release, hotfix	Simple: Create feature branches, merge to main
Best for big teams and long-running projects	Best for modern, continuous deployment teams

Best Practices for Commit Messages

Rule	Example
Start with a verb	fix: resolve bug in task deletion
Use lowercase	refactor: simplify task manager
Keep it short and meaningful	docs: update README

Using GitHub for Documentation

Tool	Purpose
README.md	Project overview
Wikis	Internal or detailed documentation
GitHub Pages	Public site for docs, demos

Case Study Example
Repository: https://github.com/tensorflow/tensorflow
Thousands of contributors
Extensive documentation in README, Wiki, and GitHub Pages
Great use of branching, releases, and CI/CD

Final Outputs You Should Have:

Task	Output
Collaborative feature added	Merged PR with code
Commit history	Proper messages per best practices
Documentation site	Live at GitHub Pages URL
Team collaboration	Branching + Reviews + Merging

Expected Outcome:

By the end of this experiment, students will be able to:

- ✓ Understand and implement Git/GitHub collaborative workflows.
- ✓ Collaborate in a team using branching strategies like GitHub Flow.
- ✓ Write proper commit messages and manage repositories professionally.
- ✓ Create and publish project documentation using GitHub Pages.
- ✓ Explore case studies of successful open-source projects.

Conclusion:

This experiment provided hands-on experience in team-based software development using GitHub. Students learned how to efficiently collaborate using branching strategies, semantic commits, pull requests, and reviews. Additionally, by deploying documentation using GitHub Pages, they understood the importance of maintaining clear, accessible project documentation for users and contributors. These skills are essential for modern software engineering practices, open-source contributions, and DevOps workflows.

NOTE:

Practical procedure

Part A: Collaborative Workflow using GitHub Flow

1. **Repository Creation**

   One team member creates a repository:

   `https://github.com/username/team-todo-app`

   Other members are added as collaborators.

2. **Cloning Repository**

```bash
git clone https://github.com/username/team-todo-app.git
cd team-todo-app
```

3. **Creating a Feature Branch**

```bash
git checkout -b feature-add-task
```

4. **Making Code Changes**

   Example modification in `app.js`:

```javascript
function addTask(task) {
    console.log("Task added:", task);
}
```

5. **Committing Changes**

```bash
git add app.js
git commit -m "feat: add basic task adding functionality"
```

6. **Pushing Changes to GitHub**

```bash
git push origin feature-add-task
```

7. **Creating and Managing Pull Requests**
   - Navigate to repository → *Compare & Pull Request*
   - Add description and title → *Create Pull Request*
8. **Code Review and Merge**
   - Team members review and comment
   - Once approved, click *Merge Pull Request*
9. **Sync Main Branch**

```bash
git checkout main
git pull origin main
```

Part B: Documentation with GitHub Pages

1. **Create a `docs/` Folder**

```bash
mkdir docs
```

Inside `docs/index.md`:

```markdown
# Team To-Do App

Welcome to our collaborative project!

## Features:
- Add tasks
- Mark tasks as complete
- Delete tasks
```

2. **Commit and Push Documentation**

```bash
git add docs/index.md
git commit -m "docs: added project documentation"
git push origin main
```

3. **Enable GitHub Pages**

- Go to *Settings → Pages*
- Source: `main` branch, `/docs` folder
- Save → Access URL like: `https://username.github.io/team-todo-app/`