**Task2:**

Steps:

Setting up an effective Git environment with the right branching strategy for the Spring Pet clinic application to accommodate the needs of 5 Developers, 2 Testers and 2 DevOps Professionals can be a multi-step process.

* Setting Up a Repository:

• After forking the repository from the given URL above [https://github.com/spring-petclinic/spring-framework-petclinic], clone the repository.

2. Clone your forked repository to your local environment using the command,

Git clone https://github.com/praveenDevarapu/spring-petclinic.git

3. Check your current working directory by running the pwd command.

pwd

4. Initialize your Git Repository:

git init

5. After cloning or initializing a Git repository, make sure to change into the appropriate directory using the cd command:

cd spring-framework-petclinic

6. To keep the fork up to date with the original repository, we add a remote called "upstream" to the original repository:

git remote add upstream <https://github.com/spring-petclinic/spring-framework-petclinic.git>

3. Git Environment Branching Strategy for Spring-Pet clinic Application

* Create Main Branch (DEV): This branch is for only production-ready code. It should only contain stable and tested code. Develop Branch is only for ongoing development. Adding new feature Branches will be created under this branch only.
* Feature Branch: Developers create feature branches from develop for developing new features or making changes. Different types of branches created for different types of features.
* Testing Branch (QA): Testers create branches from develop for writing and running test cases. They can be used for finding errors and creating test cases for better deployment.
* Release Branches: After product ready we have to deploy it in a new release branch and can be named as like ‘release/1.0”
* Hotfix Branches: These are used as critical resource branch which is used for updating issues occurred in main branch then merging it into main and develop branch
* Continuous Integration (CI) Branch: Set up a CI/CD pipeline that automatically builds and tests every push to the develop branch.

**WORKFLOW**

Developers:

Checkout a new feature branch for the work:

git checkout -b feature/features

• Develop and commit the code.

• Push the branch to the fork:

git push origin feature/features

• Create a PR to merge the feature branch into main branch

• Here we shall collaborate with other developers via code reviews on PRs.

Testers:

Checkout a new test branch for their test cases:

git checkout -b test/QA

• Create and commit test cases.

• Push the branch to their fork.

git push origin test/QA

• Create a PR to merge the test branch into the main branch.

• Collaborate with other testers via PRs for test case reviews.

DevOps Professionals:

Checkout a new DevOps branch for their automation scripts or infrastructure changes:

git checkout -b devops/scripts

• develop and commit the scripts changes.

• Push the branch to their fork:

git push origin devops/scripts

• Create a PR to merge the DevOps branch into the main branch.

• Collaborate with other DevOps professionals via PRs for automation and infrastructure changes.

With these, we can create a well-organized Git environment with an effective branching strategy that supports the requirement of the client, with collaboration among Developers, Testers, and DevOps professionals while minimizing errors and conflicts of the SpringPetClinic application.

Git repo:

https://github.com/praveenDevarapu/spring-petclinic.git

