🚀 Git in 5 Days – Real-World Project Based Assignment

Project Name:

Flipkart-Bank Payment Platform

You are part of a DevOps + Development team building a banking & payment platform for Flipkart.

Every day, the system grows.

Every day, you will face **real problems** engineers face in companies.

🗓️ DAY 1 – You Join the Company (Git Foundations)

🎯 Scenario

You joined Flipkart-Bank as a Junior Developer.

Your manager asked you to create the initial codebase and track it properly.

Tasks

1️⃣ Install Git

2️⃣ Configure identity

```
git config --global user.name "YourName"
git config --global user.email "your@email.com"
git config --list
```

3️⃣ Create project folder:

FlipkartBankApp

Inside it create:

login.txt
accounts.txt

Add:

login.txt → User login module
accounts.txt → Bank accounts module

4️⃣ Initialize Git

```
git init
git status
```

5️⃣ Add only login module:

```
git add login.txt
git commit -m "Login module created"
```

6️⃣ Add accounts module:

git add accounts.txt
git commit -m "Accounts module added"

7️⃣ View history:

git log
git show <commit-id>

8️⃣ View internal Git object:

git cat-file -p <commit-id>

🗓️ DAY 2 – Production Mistakes & Recovery

🎯 Scenario

Your team released wrong login logic to production.

You must fix it without deleting history.

1️⃣ Modify login.txt with wrong data

2️⃣ Commit mistake:

git commit -am "Wrong login logic"

3️⃣ Undo it safely:

git reset --soft HEAD~1

Fix file → recommit

4️⃣ Another wrong commit

Delete it completely:

git reset --hard HEAD~1

5️⃣ Make 3 small commits

Compress into one clean commit:

git rebase -i HEAD~3

6️⃣ Recover deleted commit:

git reflog
git reset --hard <hash>

🗓️ DAY 3 – Feature Teams & Parallel Development

🎯 Scenario

Flipkart-Bank now needs multiple features developed at the same time.

Create branches:

git branch dev
git branch feature-upi
git branch feature-creditcard

Switch:

git checkout feature-upi

Add UPI code → commit

Switch:

git checkout feature-creditcard

Add card code → commit

Merge into dev:

git checkout dev
git merge feature-upi
git merge feature-creditcard

Merge to main:

git checkout main
git merge dev

🗓️ DAY 4 – GitHub, Pull Requests & Rollbacks

🎯 Scenario

Code must be pushed to GitHub for team review.

1️⃣ Create GitHub repo

2️⃣ Connect:

git remote add origin <url>
git push -u origin main

3️⃣ Create new branch and push:

git checkout -b feature-otp
git push origin feature-otp

Create Pull Request → merge

4️⃣ Bug introduced

Undo safely:

git revert <commit-id>
git push

🗓️ DAY 5 – Conflicts, Forks & Production Release

🎯 Scenario

Two engineers edited same file at same time.

1️⃣ Pull latest:

git pull

2️⃣ Resolve conflict

3️⃣ Commit fix:

git add .
git commit -m "Resolved merge conflict"

4️⃣ Release:

git tag v1.0
git push origin v1.0

5️⃣ Emergency hotfix:

git checkout -b hotfix
git commit
git checkout main
git merge hotfix

```
git checkout dev
git merge hotfix
```