MISSION 5 : CHECKLIST – GIT HOOKS (Automatiser des actions)

Objectif : comprendre, créer et tester un hook Git (pre-commit) pour automatiser une action avant un commit.

Étape 1 — Préparation dans le projet

- ⬜ Ouvrir le projet dans VS Code
- ⬜ Vérifier qu'il n'y a aucune modification non committée :

```
git status
```

- ⬜ Faire un commit checkpoint si nécessaire :

```
git add .
git commit -m "Checkpoint avant création d'un hook"
```

- ⬜ Vérifier la présence du dossier des hooks :

```
ls .git/hooks
```

(ou dir .git/hooks sous Windows)

- ⬜ Repérer le fichier pre-commit.sample

Étape 2 — Activer un hook Pre-Commit

- ⬜ Se rendre dans le dossier .git/hooks
- ⬜ Dupliquer le fichier :

```
cp .git/hooks/pre-commit.sample .git/hooks/pre-commit
```

- ⬜ Vérifier que le fichier s'appelle maintenant : **pre-commit** (sans extension)
- ⬜ Rendre le fichier exécutable (si nécessaire) :

```
chmod +x .git/hooks/pre-commit
```

(Sous Windows Git Bash, ça fonctionne aussi)

Étape 3 — Modifier le hook pre-commit

- ⬜ Ouvrir .git/hooks/pre-commit dans VS Code
- ⬜ Effacer son contenu
- ⬜ Ajouter ce script minimal :

```
#!/bin/sh
echo "Vérification pre-commit : aucun mot interdit ne doit apparaître."
if grep -R "interdit" .; then
  echo "Erreur : le mot 'interdit' a été trouvé. Commit bloqué."
  exit 1
fi
exit 0
```

- ⬜ Enregistrer
- ⬜ Vérifier que le hook est bien pris en compte

Étape 4 — Tester le hook (commit bloqué)

- ⬜ Modifier un fichier du projet
- ⬜ Ajouter volontairement le mot "interdit" dedans
- ⬜ Ajouter le fichier :

```
git add fichier
```

- ⬜ Tenter un commit :

```
git commit -m "Test hook bloqué"
```

- ⬜ Observer que le commit est **refusé**
- ⬜ Vérifier que le message du hook s'affiche

Étape 5 — Tester le hook (commit accepté)

- ⬜ Supprimer le mot "interdit"
- ⬜ Ajouter le fichier :

```
git add fichier
```

- ⬜ Faire un commit :

```
git commit -m "Commit accepté après correction"
```

- ⬜ Vérifier que le commit passe correctement
- ⬜ Noter l'expérience dans docs/journal.md

Étape 6 — Créer votre propre hook simple

L'équipe doit créer un hook qui fait une action parmi ces choix (au minimum) :

⬜ **afficher un message personnalisé**

⬜ **refuser les commits trop courts (ex : messages < 5 caractères)**

⬜ **refuser les commits contenant "todo"**

⬜ **vérifier qu'un fichier obligatoire existe (ex : README.md)**

⬜ **ajouter automatiquement la date dans un fichier log**

Dans .git/hooks/pre-commit :

- ⬜ Modifier le script
- ⬜ Tester plusieurs commits pour valider son fonctionnement
- ⬜ Noter le comportement dans docs/journal.md

Étape 7 — Défi final de l'équipe

Réaliser les actions suivantes :

- [ ] ⬜ Le hook bloque un commit volontairement incorrect
- [ ] ⬜ Le hook accepte un commit correct
- [ ] ⬜ Le hook affiche un message personnalisé
- [ ] ⬜ Le groupe explique ce que fait son hook dans docs/journal.md
- [ ] ⬜ Chaque membre a testé le hook
- [ ] ⬜ Le hook a été conservé dans le dépôt
- [ ] ⬜ Le groupe a listé des idées d'automatisation utiles

MISSION VALIDÉE SI :

- ⬜ Le fichier pre-commit existe
- ⬜ Le hook fonctionne et s'exécute
- ⬜ Un commit a été bloqué
- ⬜ Un commit a été accepté
- ⬜ Un hook personnalisé a été créé
- ⬜ Les tests sont notés dans le journal
- ⬜ Le défi final est complété