Partie 1 — Initialisation

Question 1— Création du projet

npm create vite@latest react-task-manager -- --template react : cette commande lance vite en utilisant la dernière version pour créer un projet ayant pour nom “react-task-manager" avec pour template react (javascript).

cd react-task-manager: pour se déplacer à l’intérieur du dossier créé

Les fichiers à la racine sont index.html, package.json, vite.config.js, le dossier src (coeur de l’application) contenant les fichiers main.jsx, App.jsx, App.css, index.css et le dossier assets.

npm install: télécharge toutes les dépendances à l’intérieur du package.json, crée un dossier node_modules et génère un fichier package-lock.json

npm run dev: lance le projet

Question 2 — Initialisation Git

git init : initialise un dépôt github dans le dossier react-task-manager

git status : Cette commande affiche l’état du dépôt local.

Aucun commit n’a été créé et Git ne suit pas les fichiers, ils sont dans l’espace de travail (Working directory).

Résultat de la commande:

On branch main

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

.gitignore

README.md

eslint.config.js

index.html

package-lock.json

package.json

public/

src/

vite.config.js

git add . : ajoute tous les fichiers du projet à la zone de staging(Staging Area). Il s'agit d'une zone intermédiaire où sont mis les fichiers à inclure dans un prochain commit.Ils sont marques "staged"

git status : Les fichiers sont prêts à être enregistrés dans un commit

Résultat de la commande :

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: .gitignore

new file: README.md

new file: eslint.config.js

new file: index.html

new file: package-lock.json

new file: package.json

new file: public/vite.svg

new file: src/App.css

new file: src/App.jsx

new file: src/assets/react.svg

new file: src/index.css

new file: src/main.jsx

new file: vite.config.js

git commit -m "chore(init): initial vite react project": Crée un commit contenant tous les fichiers ajoutés avec un message décrivant l'action réalisée

git status : Aucun changement n'est en attente, tous les fichiers sont suivis et enregistrés dans l'historique Git (Repository).

Résultat de la commande :

On branch main

nothing to commit, working tree clean

Question 3 — Lien avec GitHub

git branch -M main: cette commande renomme la branche courante main en écrasant toute branche main préexistante

git remote add origin https://github.com//react-task-manager-m1.git : connecte le dépôt local au dépôt distant Github

Origin est par convention le nom du dépôt distant

git status : Le dépôt local n'est pas encore synchronisé avec celui distant

On branch main

nothing to commit, working tree clean

Your branch is based on 'origin/main', but the upstream is not set.

git push -u origin main: cette commande tranfère les commits du dépôt local vers le dépôt distant. Le contenu de la branche main est donc envoyé sur Github

git status: indique qu'il n'y a aucune différence entre les dépôts local et distant. Pas de fichier à committer

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Partie 2 — Travail par fonctionnalité

Question 4 — Branche de structure

git status : le dépôt est propre et synchronisé

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

git checkout -b feature/app-structure : crée la branche feature/app-structure et passe de la branche main à cette dernière

git status: Il n'y a pour le moment aucun changement

On branch feature/app-structure

nothing to commit, working tree clean

Création des fichiers TaskList.jsx et TaskForm.jsx

git status: les fichiers existent mais Git ne les suit pas encore

On branch feature/app-structure

Untracked files:

(use "git add <file>..." to include in what will be committed)

src/components/

nothing added to commit but untracked files present (use "git add" to track)

git add . puis **git status :** Ajout des nouveaux fichiers à la staging Area

On branch feature/app-structure

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: src/components/TaskForm.jsx

new file: src/components/TaskList.jsx

git commit -m "feat(components): add task components structure": crée un commit qui contient les fichiers ajoutés

git status :

On branch feature/app-structure

nothing to commit, working tree clean

git push origin feature/app-structure : envoie la branche feature/app-structure vers le dépôt distant Github

git status : les branches locale et distante sont synchronisées

On branch feature/app-structure

nothing to commit, working tree clean

Une branche feature a utilisée afin d'éviter d'effectuer les changements directement sur la branche main. Cela permet une organisation claire. En cas d'erreur, la branche main reste stable.

Question 5 — Intégration dans main

git checkout main : pour basculer sur la branche main

git status: vérifie le statut de la branche, aucune modification

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

git pull origin main : pour récupérer d'éventuels changements sur la branche main

git status: la branche locale main est synchronisée avec origin/main

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

git merge feature/app-structure: pour fusionner la branche feature/app-structure dans main. Les commits de feature/app-structure sont intégrés dans la branche main

git status: la branche main locale est avance par rapport à la branche main distante.

On branch main

Your branch is ahead of 'origin/main' by 1 commit.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

git push origin main : envoie les nouveaux commits vers le dépôt distant

git status : les branches locales et distantes sont synchronisées

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

On a donc créé une nouvelle brache feature pour le développement d'une nouvelle fonctionnalité. Cette dernière est ensuite poussée vers le dépôt distant. Puis, on se place sur la branche main, on récupère les dernières modifications sur la branche main distante avant de fusionner la branche feature dans main. La branche locale main contenant la nouvelle fonctionnalité est par la suite, poussée vers le dépôt Github.

Partie 3 — Développement fonctionnel

Question 6 — Ajout de tâche

git checkout -b feature/add-task : crée une nouvelle branche pour implémenter la fonctionnalité d'ajout de tâche puis bascule sur la branche crée

git status:

On branch feature/add-task

nothing to commit, working tree clean

Implémentation de la fonctionnalité

git add . : les fichiers modifiés sont ajoutés dans la zone de staging

git status : ils passent à l'état modified

On branch feature/add-task

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: src/App.css

modified: src/App.jsx

modified: src/components/TaskForm.jsx

modified: src/components/TaskList.jsx

git commit -m "feat(tasks) : implement add task" : crée un commit contenant l'implémentation de l'ajout de tâche.

git status : les modifications sont enregistrées localement

On branch feature/add-task

nothing to commit, working tree clean

git push origin feature/add-task: envoie la branche feature/add-task vers Github

git status: les branches locale et distante sont synchronisées

On branch feature/add-task

nothing to commit, working tree clean

git checkout main: retour sur la branche main

git pull origin main : mets à jour la branche locale main

git merge feature/add-task : fusionne les modifications de la branche feature/add-task dans main

git push origin main : pousse la branche locale main sur le dépôt Github

git status :

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Question 7 — Affichage des tâches

git checkout -b feature/display-tasks: crée une nouvelle branche feature/display-tasks et s'y positionne

git status :

On branch feature/display-tasks

nothing to commit, working tree clean

git add . puis git commit -m "feat(tasks): display task list": ajout des fichiers modifiés à la staging area puis création d'un commit contenant la fonctionnalité d'affichage

git push origin feature/display-tasks: pousse la branche locale sur github

git checkout main : bascule sur la branche main

git merge feature/display-tasks : fusionne le contenu de la branche créée dans main

git push origin main: mets à jour la branche main distante avec la nouvelle fonctionnalité

git status: les dépôts lacalet distant sont synchonisés

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

COMPARAISON : le worflow de cette question est identique à celui de la question précédente.

Partie 4 — Correction et refactor

Question 8 — Correction

git checkout -b fix/empty-task-validation : crée et bascule sur la branche fix/empty-task-validation

git status : affiche la branche courante et l'état des fichiers

git add . : ajoute les fichiers modifiés à la zone de staging

git commit -m "fix(tasks): prevent empty task" : enregistre les changements dans un commit

git push origin fix/empty-task-validation: envoie la branche vers le dépôt distant Github

git checkout main: Switch vers la branche main

git merge fix/empty-task-validation : fusionne le contenu de fix/empty-task-validation dans la branche locale main

git push origin main : envoie la branche main modifiée sur le dépôt distant Github

git status : les branches locale et distante sont synchronisées

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Le **préfixe fix** a été utilisé parce qu'il s'agit d'une correction d'un code existant et pas l'implémentation d'une nouvelle fonctionnalité

Question 9 — Refactor

git checkout -b refactor/task-state: crée et bascule sur la branche refactor/task-state

git status:

On branch refactor/task-state

nothing to commit, working tree clean

git add . : ajoute les fichiers modifiés à la staging Area

git commit -m "refactor(tasks): improve state structure" : crée un commit avec les dernières modifications

git push origin refactor/task-state: envoie la branche refactor/task-state vers le dépôt distant Github

git checkout main: bascule sur main

git merge refactor/task-state : fusionne les modifications de la branche refactor dans main

git push origin main : publie le contenu de la branche locale main sur github

git status : les dépôts local et distant sont synchronisés

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Différence entre feat et refactor : le feat est utilisé pour l'ajout d'une nouvelle fonctionnalité. Quant au refactor, c'est pour l'amélioration du code(sa structure ou son organisation)

Partie 5 — Synchronisation

Question 10 — Travail parallèle

git checkout -b feature/delete-task : crée une nouvelle branche nommée feature/delete-task

git status :

On branch feature/delete-task

nothing to commit, working tree clean

git add . : ajoute les fichiers modifiés à la staging Area

git commit -m "feat(tasks): add delete task" : crée un commit avec les dernières modifications sur la branche

git push origin feature/delete-task : pousse la branche locale feature/delete-task sur le dépôt distant github

git status: les branches sont synchronisées

On branch feature/delete-task

Your branch is up to date with 'origin/feature/delete-task'

nothing to commit, working tree clean

Question 11 — Mise à jour avec main

git checkout main : bascule sur main

git pull origin main : mets à jour la branche locale main

git status :

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

git checkout feature/delete-task: bascule sur feature/delete-task

git merge main : fusionne le contenu de main dans feature/delete-task

git status puis **git push origin feature/delete-task :** mets à jour la branche distante avec les derniers commits intégrés depuis main

git checkout main: switch vers la branche main

git merge feature/delete-task : fusionne le contenu de feature/delete-task dans main

git push origin main : pousse les modifications sur le dépôt distant

git status :

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

La **synchronisation** est nécessaire pour éviter les conflits et pour une bonne collaboration.

Git status est un outil clé pour suivre si la branche est **en avance**, **en retard**, ou **synchronisée** avec le dépôt distant.

Partie 6 — Versioning

Question 12 — Tag de version

git status:

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

git tag -a v1.0.0 -m "release: version 1.0.0": crée un tag annoté sur le dernier commit de la branche main

git status: le tag n'a pas affecté le working tree

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

git push origin v1.0.0: envoie le tag vers le dépôt distant

git status: git status reste inchangé

Le tag permet de marquer des versions stables dans l'historique Git, de revenir à une version précise du code