**1)-**

1. En tant qu'utilisateur, je veux pouvoir créer un nouveau projet en spécifiant son nom, sa description, et les membres de l'équipe associés.
2. En tant qu'utilisateur, je veux pouvoir ajouter des tâches à un projet, avec des descriptions, des dates d'échéance et des attributions à des membres de l'équipe.
3. En tant qu'utilisateur, je veux pouvoir visualiser un calendrier global avec les dates de début et de fin des projets, ainsi que les tâches associées.
4. En tant qu'utilisateur, je veux pouvoir marquer des jalons importants dans le projet et être notifié lorsque ceux-ci sont atteints.
5. En tant qu'utilisateur, je veux pouvoir suivre le progrès global du projet à l'aide de graphiques et de tableaux de bord.
6. En tant qu'utilisateur, je veux pouvoir collaborer avec les membres de l'équipe en ajoutant des commentaires et en partageant des fichiers directement sur la plateforme.
7. En tant qu'utilisateur, je veux que l'application soit accessible à partir de divers appareils pour une flexibilité maximale.
8. En tant qu'utilisateur, je veux pouvoir exporter des rapports sur l'avancement du projet pour les partager avec d'autres parties prenantes.

**2)-**

1. Gestion de Projet

* Création de projet
* Visualisation du calendrier global

1. Gestion des Tâches

* Ajout de tâches
* Suivi des jalons

1. Collaboration et Accessibilité

* Collaboration entre membres de l'équipe
* Accessibilité multi-appareils

**3)-**

**Sprint 1:**

* Date de début: 15/01/2023
* Date de fin: 28/01/2023
* Objectifs: Création de projet, Ajout de tâches, Collaboration de base

**Sprint 2:**

* Date de début: 01/02/2023
* Date de fin: 14/02/2023
* Objectifs: Visualisation du calendrier global, Suivi des jalons, Accessibilité multi-appareils

**Sprint 3:**

* Date de début: 20/02/2023
* Date de fin: 05/03/2023
* Objectifs: Graphiques de suivi, Exportation de rapports, Finalisation et tests

**4)-** Le Burndown Chart représente graphiquement le travail restant par rapport au temps. Une pente constante vers le bas indique un progrès régulier. Des variations aident à identifier les retards ou les avancements rapides. L'utilisation du Burndown Chart permet de prendre des décisions éclairées, de détecter les problèmes rapidement et d'ajuster le plan si nécessaire pour respecter les délais.

**5)-** Le workflow dans Jira représente le cycle de vie des tâches, de la création à l'achèvement. On peut personnaliser les états, transitions et actions associées pour refléter le processus spécifique d'un projet. Par exemple, on peut ajouter des états comme "En cours de test" et des transitions comme "Envoyer en test". Cela assure que le workflow s'aligne sur les besoins particuliers du développement logiciel.

**6)-** La fonctionnalité de Roadmap dans Jira permet de visualiser et de planifier l'évolution du projet sur une échelle temporelle étendue. Elle aide à aligner les initiatives à long terme, à gérer les dépendances et à communiquer efficacement avec les parties prenantes. La Roadmap offre une vue holistique qui facilite la prise de décisions stratégiques pour le succès du projet à long terme.

**7)-** Pour une répartition équilibrée, utilisez la capacité de visualiser la charge de travail par membre d'équipe dans Jira. En encourageant la collaboration, utilisez des fonctionnalités telles que les commentaires, les mentions et les tableaux de bord partagés. Les outils comme les boards Scrum/Kanban et les rapports d'avancement individuel aident à suivre la progression et à identifier les zones nécessitant un soutien supplémentaire.

|  |  |  |
| --- | --- | --- |
| Question | Réponse | Note |
| **Partie 1 : Shell Commandes 7pts** | | |
| Q1. Se déplacer dans le répertoire parent : Utilisez la commande pour passer du répertoire courant à son répertoire parent | **cd ..** |  |
| Q2. Créer un dossier et se deplacer dans ce Q3. nouveau dossier : Créez un nouveau dossier appelé "ProjetShell" dans le répertoire courant. Puis s’y déplacer | **mkdir ProjetShell**  **cd ProjetShell** |  |
| Q4. Créer un Fichier html : Utilisez une commande pour créer un fichier HTML nommé "index.html" dans "ProjetShell". En Ecrivant le contenu | **echo "<!DOCTYPE html>**  **<html lang='en'>**  **<head>**  **<meta charset='UTF-8'>**  **<meta name='viewport' content='width=device-width, initial-scale=1.0'>**  **<title>Mon Projet Shell</title>**  **</head>**  **<body>**  **<h1>Bienvenue dans mon projet Shell!</h1>**  **<p>Ceci est un fichier HTML créé à l'aide de commandes Shell.</p>**  **</body>**  **</html>" > ProjetShell/index.html** |  |
| Q5. Créer suxessivement les dossiers Styles et Scripts puis se deplacer dans chacun et créer un fichier css simple monStyle.css puis le script main.js | **mkdir ProjetShell/Styles**  **mkdir ProjetShell/Scripts**  **cd ProjetShell/Styles**  **echo "body {**  **background-color: #f4f4f4;**  **color: #333;**  **}" > monStyle.css**  **cd ..**  **cd Scripts**  **echo "console.log('fichier de main.js');" > main.js** |  |
| Q6. Revenir sur le répertoire du projet et lister son contenu | **cd ~/repertoire\_de\_projet/ProjetShell**  **ls** |  |
| Q7. Afficher le contenu du fichier html | **cd ~/repertoire\_de\_projet/ProjetShell**  **cat index.html** |  |
| **Partie 2 : Git et Git Hub -20pts-** | | |
| Q1. Initialiser un repository Git :  Dans le répertoire "ProjetShell", initialisez un nouveau repository Git. Puis Vérifier le statut du repository. | **cd ~/repertoire\_de\_projet/ProjetShell**  **git init**  **git status** |  |
| Q2. Ajouter des fichiers au staging area :  Ajoutez "index.html" au staging area.  Effectuer un commit :  Effectuez un commit avec le message "Initial commit". | **cd ~/repertoire\_de\_projet/ProjetShell**  **git add index.html**  **git commit -m "Initial commit"** |  |
| Q3. Annulez le dernier commit et montrez le status | **cd ~/repertoire\_de\_projet/ProjetShell**  **git reset --soft HEAD^**  **git status** |  |
| Q4. Ajouter tous les elements au staging puis faire un commit et montré le status | **cd ~/repertoire\_de\_projet/ProjetShell**  **git add .**  **git commit -m "Ajout de tous les fichiers"**  **git status** |  |
|  |  |  |
| Q5. Créer une nouvelle branche :  Créez une branche appelée "feature-css".  Changer de branche :  Passez à la branche "feature-css".  Faire des changements sur la nouvelle branche :  Ajoutez du style CSS supplémentaire à "monStyle.css".  Q6. Revenir à la branche principale :  Retournez à la branche principale.  Faire des changements sur la branche principale :  Ajoutez du contenu JavaScript supplémentaire à "script.js".  Et au fichier monStyle.css de manière à créer un conflit de merge | **cd ~/repertoire\_de\_projet/ProjetShell**  **git branch feature-css**  **git checkout feature-css**  **echo "/\* Nouveau style CSS \*/" >> monStyle.css**  **git status**  **cd ~/repertoire\_de\_projet/ProjetShell**  **git checkout master**  **echo "console.log('Changement sur la branche principale.');" >> Scripts/main.js**  **echo "/\* Modification sur la branche principale \*/" >> Styles/monStyle.css**  **git status** |  |
| Q7. Fusionnez la branche "feature-css" dans la branche principale, créant ainsi un conflit de fusion.  Résoudre le conflit de fusion :  Utilisez une commande pour résoudre le conflit de fusion.  Refaire le merge :  Refaites le merge après avoir résolu le conflit. | **cd ~/repertoire\_de\_projet/ProjetShell**  **git merge feature-css**  **git add Styles/monStyle.css**  **git merge --continue** |  |
| Q8. Décrire comment Trouvez l'URL de votre repository GitHub | **Une fois que vous êtes dans le repository, vous verrez l'URL du repository dans la barre d'adresse de votre navigateur. L'URL se terminera généralement par ".git".** |  |
| Q9. Utilisez une commande pour définir le remote repository sur l'URL que vous avez récupérée. | **cd ~/repertoire\_de\_projet/ProjetShell**  **git remote add origin https://github.com/votre-nom utilisateur/votre-repository.git** |  |
| Q10. Poussez les deux branches dans votre gitHub | **cd ~/repertoire\_de\_projet/ProjetShell**  **git push origin master**  **git push origin feature-css** |  |
| Q11. Créer une nouvelle branche sur GitHub :  Créez une nouvelle branche appelée "feature-js" directement sur GitHub.  Faire des changements sur la nouvelle branche :  Ajoutez du code JavaScript supplémentaire à "script.js" sur la branche "feature-js".  (les changements dans les deux branches sur git pour créer un conflit | **cd ~/repertoire\_de\_projet/ProjetShell**  **git checkout master**  **echo "console.log('Modification sur la branche principale.');" >> Scripts/script.js**  **git add Scripts/script.js**  **git commit -m "Modification sur la branche principale"** |  |
| Q12. Fusionnez la branche "feature-js" dans la branche principale sur GitHub, créant ainsi un problème de fusion. (pullRequest) et l’executé |  |  |
| Q13. Résolution des pb de merge puis faire le merge | **git checkout master**  **git merge feature-js**  **git add chemin/vers/les/fichiers/resolus**  **git merge --continue**  **git log**  **git status**  **git push origin master** |  |
| Q14. Dans votre local repository comment connaitre le repository distant et comment le changer (commandes) | **git remote -v**  **git remote set-url origin nouvelle\_url\_du\_repository**  **git remote -v**  **git remote add nom\_du\_remote nouvelle\_url\_du\_repository** |  |
| **Partie 3 : Questions de cours (4pts)** | | |
| Q1. Est il possible d’annuler un push donnez une commande si oui | **git reset --hard HEAD^**  **git push -f origin nom\_de\_la\_branche** |  |
| Q2. Est il possible d’annler un add si oui donnez la commande | **git reset -- chemin/vers/le/fichier**  **git reset** |  |
| Q3. Est il possible de faire annuler un merge si oui donnez la commande | **git reset --hard HEAD^**  **git reset --soft HEAD^** |  |
| Q4. Répondre aux question | Reprendre les réponses des questions 1, 2,3 cette fois-ci sur github |  |