1 INFORMATIONS GENERALES

Élève :	Nom :	Prénom :
Élève :	Nom :	Prénom :
Lieu de travail :	ETML / Avenue de Valmont 28b, 1010 Lausanne	
Client Test	Nom :	Prénom :
Client DevOps	Nom :	Prénom :
Dates de réalisation :	2ème trimestre	
Temps total :	~64 périodes (32 P_Test 450 + 32 P_DevOps 324)	

2 PROCÉDURE

- Tous les apprentis réalisent le projet sur la base d'un cahier des charges.
- Les apprentis travaillent par groupe de deux élèves.
- Le cahier des charges est présenté, commenté et discuté en classe.
- Les apprentis sont entièrement responsables de la sécurité et sauvegarde de leurs données.
- En cas de problèmes graves, les apprentis avertissent le client au plus vite.
- Les apprentis ont la possibilité d'obtenir de l'aide externe, mais ils doivent le mentionner.
- Les informations utiles à l'évaluation de ce projet sont disponibles au chapitre 8.

3 TITRE

Todo App

4 SUJET

Mise en place d'une stratégie de tests, élaboration de tests unitaires et E2E et déploiement CI/CD avec GitHub Action.

5 MATÉRIEL ET LOGICIEL À DISPOSITION

- Un PC ETML
- Accès à Internet
- Journal de travail (selon les modalités de votre enseignant)
- Un accès SSH sur une instance Ubuntu
- Un compte GitHub
- Visual Studio Code

6 PRÉREQUIS

- Modules de programmation de base
- Modules de bases de données de base
- ICT-450 en cours
- ICT-324 en cours

7 CAHIER DES CHARGES

7.1 Gestion de projet

1. La planification est à faire selon les instructions spécifiques de votre chef de projet.
2. Un journal de travail devra être rendu. L'outil que vous utilisez est libre, mais les caractéristiques suivantes doivent être respectées :
   - La structure et la présentation sont claires et soignées.
   - Les sources, les fichiers, les répertoires, les commits, et autres sources d'informations concernées par le journal sont référencés.
   - L'état et les durées des tâches mentionnées sont précisés.
   - Toutes les activités planifiées, les aides extérieures, ainsi que les imprévus et les heures supplémentaires y sont mentionnés.
   - Les succès et les échecs sont mentionnés.
   - Le travail journalier et son appréciation critique, ainsi que les réflexions y figurent.

7.2 Qualité

1. Réaliser un programme informatique de qualité
   - Organisé (namespace, classes, commit log,…)
   - Compacté (pas de copié/collé,…)
   - Optimisé (utilisation de structures adaptées)
   - **Testé (tests unitaires et E2E)**
   - Commenté
   - Complet (code, script DB, maquettes PDF, éxécutable, …)
2. Prouver que vous êtes digne de confiance lorsqu'on vous confie un projet
   - Journal de travail à jour
   - Pro-activité
     - **Poser des questions** au client
     - Faire des démonstrations
3. Repo **privé** sur github avec le nom : **cicd-todo-app** (partagé avec vos enseignants) utilisant un **.gitignore** concluant.

7.3 Fonctionnalités requises (du point de vue client)

L'application Todo fonctionne avec un frontend **Vue.js** et un backend **Node.js/Express** qui fournit une API utilisée par le frontend.

La persistance des données de l'API est assurée par une base de données MySQL.

7.3.1 Spécificité de Test

Classification des bugs

Après avoir mis en place l'application fournie, vous devez la tester manuellement, en tant qu'utilisateur :

- Création de compte
- Login
- Gestion des todo
- Modification du profil
- Logout
- Mode clair / sombre

Pour donner suite à ces tests utilisateurs manuels, le rapport devra contenir une liste de bugs que vous devrez classifier par sévérité / gravité. Il vous est demandé d'attribuer une priorité pour chacun des bugs.

Les tests automatisés que vous allez mettre en place devront faire ressortir les mêmes bugs dans un premier temps.

Si vous modifier le code pour corriger un bug (et donc faire passer les tests correspondants), ceci doit être clairement indiqué dans les commit de votre GIT ainsi que dans le rapport (corrections apportées). Les bugs ayant une priorité « haute » devront être corrigés en premier.

Stratégie et plans de test

L'objectif est d'élaborer une stratégie de test cohérente.

Le document de rendu du projet devra contenir :

- La stratégie de test globale au projet
    - Les objectifs de la stratégie de test
    - Le périmètre des tests
    - Les types de tests et les outils utilisés
    - Les environnements de test (objectif, contenu, accès)
    - La configuration des environnements (serveur, DB, navigateur, déploiement)
    - Les risques et mesures d'atténuation
- Les plans de test (un par fonctionnalité) pour les fonctionnalités suivantes :
    - Création du profil (sign up)
    - Authentification (login / logout)
    - Gestion du profil
    - Ajout d'un Todo
    - Gestion des Todo
- Chaque plan de test doit contenir :
    - Périmètre du test
    - Versions des éléments (application, outils, framework)
    - Les données de test si applicable
    - Les types de tests
        - Unitaires
        - E2E
- Les rapports de tests associés

Tests unitaires et E2E

a. Mettre en place les **tests unitaires** sur le backend
   i. en utilisant *jest* et *supertest*
   ii. test de routes en place
      1. authentification
      2. utilisateurs
      3. todo

b. Mettre en place les **tests E2E** du frontend
   i. en utilisant Cypress
   ii. testé sur Edge, Chrome et Firefox
   iii. test des fonctionnalités suivantes *(tous les tests qui semblent nécessaires pas uniquement un test valide)*
      1. sign up
      2. login / logout
      3. gestion du profil
      4. ajout d'un todo
      5. gestion des todo
      6. navigation (todos, à propos, profil, mode clair / sombre)

c. Rapport de code coverage backend et frontend

7.3.2 Spécificités CI/CD

Le workflow CI/CD sera mis en place avec GitHub Action

a. Le déploiement de test doit se déclencher lors de chaque « push » sur la branche principale « main ».
b. Un déploiement de production peut être déclenché manuellement.
c. Le actions sur le frontend et sur le backend doivent inclure :
    i. l'**audit** des package pour éviter toutes vulnérabilités des packages de dépendances
    ii. le **linting** pour éviter des erreurs potentielles dans le code.
    iii. Les **tests** (frontend avec les 3 explorateurs fournit par Cypress)
d. Le workflow sera optimisé pour exécuter certaines tâches en parallèle et utilisera des techniques de mise en cache pour optimiser le temps de déploiement.
e. Les déploiements de l'application « staging » et de « production » seront effectués sur un serveur dédier.

f. Optionnel :
    i. Ajouter un workflow spécifique pour anticiper les conflits de fusion dans la branche principale « main », qui s'exécute lorsqu'une pull request (PR) est ouverte ou mise à jour, afin de vérifier si la PR peut être fusionnée sans conflit avec « main »
    ii. Ajouter un workflow spécifique pour un assignement initial des issues à un utilisateur pour le triage
    iii. Le code coverage des tests E2E du frontend – avec rapport sur codecov.io
    iv. Le code coverage des tests unitaires du backend – avec rapport sur codecov.io
    v. Ajout de badges (statut GitHub Action et % code coverage avec codecov.io) au README.md

7.4 Livrables

1. Une **release** GitHub

Pour le 450 – P_UX

2. Un rapport au format PDF concernant les spécificités de test contenant :
    a. Introduction
    b. Planification initiale
    c. Tests manuels et classification des bugs
    d. Stratégie et plans de tests
    e. Description des tests backend effectués
    f. Description des tests E2E effectués
    g. Résultats de la campagne de test (unitaire et E2E)
    h. Corrections apportées au code
    i. Rapport de tests (statut, conclusion)
    j. Chapitre explicatif de l'usage fait de l'IA dans ce projet
    k. Conclusion

Pour le 324 – P_DevOps

3. Planification et suivit de projet type KanBan faite sur GitHub Project.
4. Un rapport au format PDF concernant les spécificités CI/CD contenant :
    a. Introduction
    b. Description des actions nécessaires (lint, audit, test, run, build) pour faire fonctionner le backend et le frontend incluant les spécificités selon les environnements de développement et de production
    c. Description de toutes les étapes du workflow CI/CD (backend/frontend)
    d. Le schéma (diagramme de flux) initial du workflow complet (backend/frontend) exprimant les conditions et dépendances
    e. Rapports de déploiement (statut, conclusion)
    f. Le schémas CI/CD de GitHub Action mit en place (capture d'écran)
    g. Chapitre explicatif de l'usage fait de l'IA dans ce projet
    h. Conclusion

5. Journal de travail individuel (selon modalité de votre enseignant)

8 Évaluation

1. Auto-évaluation challengée par le client basé sur des éléments observables.
2. Pour la partie technique des tests : Deux questions techniques posées oralement par l'enseignant lors de la dernière séquence du projet.
3. Pour la partie technique du workflow de déploiement : Deux questions techniques posées oralement par l'enseignant lors de la dernière séquence du projet.
4. Le recours à des outils en ligne d'intelligence artificielle (ex. : Chat GPT) est autorisé mais ne peut servir que **d'inspiration à la réalisation**. Chaque développeur doit être à tout moment en mesure d'expliquer le code de manière précise et convaincante.
   En cas d'abus, l'évaluation du projet en tiendra compte.