USTHB **2024-2025**

Faculté d'informatique **3ème Année CS**

Cycle Ingéniorat

projet de programmation avancée

THÈME DU PROJET

" Application QCM"

Présenté par :

- **KHETTAOUI NAHLA**
- **CHETOUH AMIRA NARIMANE**
- **DAHMANI NAILA**

Encadré par :

Mr Said MOUHOUN

Introduction

Dans le cadre de notre projet, nous avons développé une application de QCM (Questions à Choix Multiples) pour étudiants en informatique, pour tester et renforcer leurs connaissances dans divers domaines tels que la programmation en Python , la compilation , le développement web et la recherche opérationnelle .

Ce projet vise à satisfaire les exigences suivantes :

1. Fournir une interface graphique interactive avec l'utilisateur .
2. Permet à l'utilisateur de s'inscrire , et sauvegarder ses informations .
3. Offrir aux l'utilisateur la possibilité de s'exercer avec des QCM en mesurant le temps passé pour chaque questions .
4. Permet l'exportation des résultats .
5. Donner un accès à l'historique des scores obtenues .

Implémentation et choix technique

Le programme repose sur les classes suivantes :

- User : pour gérer les informations et l'historique de l'utilisateur .
  fonctions implémentés :
  - initialisation d'un utilisateur avec son nom et un historique vide
  - Ajouter un résultat de QCM avec la date (et l'heure) , score et la catégorie .
  - Affichage de l'historique des QCM passées par l'utilisateur

  choix technologique :
  - ☑ gestion des dates et heures en utilisant la méthode **datatime.now()** pour obtenir la date et l'heure actuelle , **strftime("%Y-%m-%d %H:%M:%S")** pour formater cette date dans un format spécifique , exemple : 2025-01-22 10:34:45 .

- QuizManager : pour gérer les questions et l'exécution du quiz .
  fonctions implémentés :
  - initialisation en chargeant les questions depuis le fichier csv et récupérer l'utilisateur
  - Charger les question dans un dictionnaires depuis le fichier CSV (stockés par catégorie)
  - Charge les données des utilisateurs depuis un fichier JSON.

- ➢ Sauvegarde les utilisateurs et leur historique dans le fichier JSON
- ➢ Gère la connexion d'un utilisateur, crée un nouveau profil si nécessaire.
- ➢ Exécute un QCM pour l'utilisateur et enregistre son score.
- ➢ Exporter les résultats des utilisateurs dans un fichier CSV.

choix technologique :

- ☑ gestion des fichier csv (lecture et écriture) avec **csv.DictReader**
- ☑ manipulation des listes **List** et des dictionnaires **Dict** pour organiser et manipuler les questions, les réponses et l'historique des utilisateurs.
  La structure des questions est un dictionnaire imbriqué avec des catégories comme clés.
- ☑ gérer une erreur de type **FileNotFoundError** si le fichier users.json n'est pas trouvé
- ☑ Le module **`random`** est utilisé pour mélanger les options de réponses pour chaque question via la fonction **`random.shuffle()`.**
- ☑ Exportation des résultats: la méthode **`export_results`** utilise **`csv.writer`** pour enregistrer les résultats des quiz des utilisateurs dans un fichier CSV, avec des colonnes pour le nom d'utilisateur, la date, la catégorie et le score.
- ☐ utilisé pour sauvegarder et récupérer les données des utilisateurs avec JSON

❖ QuizApp : pour implémenter l'interface graphique et l'interaction avec l'utilisateur .
fonctions implémentés :

- ➢ Connexion de l'utilisateur avec gestion de l'historique des scores.
- ➢ Sélection d'une catégorie de QCM et affichage des questions avec plusieurs choix.
- ➢ Chronomètre limitant le temps de réponse.
- ➢ Feedback immédiat après chaque réponse.
- ➢ Stockage des scores et possibilité d'afficher/exporter l'historique.

choix technologique :

- ☑ CustomTkinter(`ctk`) : Améliore l'apparence des interfaces Tkinter classiques.

plus :

- ☑ POO: Utilisation de classes (**`QuizApp`**, **`QuizManager`**, **`User`**) pour une structure modulaire.

Les défis rencontrés

1. Gestion du chronomètre de réinitialiser correctement le temps lorsqu'un utilisateur effectue un deuxième test dans la même session sans conflit avec le précédent
2. Persistance des données chargées et sauvegardées correctement dans l’historique des utilisateurs en JSON
3. Filtrer l'historique pour exporter uniquement l’historique de l'utilisateur ciblé sans inclure d'autres utilisateurs.
4. Mélange des réponses et maintenir l’association entre les choix mélangés et la bonne réponse.
5. Gestion du code source avec GitHub nécessitant un effort d'adaptation.