UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIEN

Faculté d'Informatique

RAPPORT PROJET TP DU MODULE : PROGRAMMATION AVANCEE

Sujet :

Développement d'une Application de QCM pour Étudiants en Informatique

Réalisé par :

1- BOUHRAMA IMENE
2- BOUZEGZI YASMINE
3-DJAIDJA MERIEM
4-HAMOUCHE SARAH

1-INTRODUCTION :

Le programme développé est un QCM interactif écrit en Python. Il a pour objectif principal de permettre aux utilisateurs de tester leurs connaissances sur différents sujets, tels que Python, les algorithmes et les réseaux. Le programme combine plusieurs fonctionnalités, notamment la gestion des utilisateurs, le stockage des résultats, et l'interaction avec un fichier JSON pour le contenu des questions.

Grâce à une interface textuelle conviviale, chaque utilisateur peut créer un profil, choisir une catégorie de questions, répondre au QCM tout en respectant une limite de temps pour chaque question, puis consulter son score. Les résultats sont également sauvegardés dans des fichiers CSV, offrant une traçabilité et un historique des performances.

2-Les composants principaux du projet

1. Gestion des utilisateurs

- Les informations des utilisateurs (nom, historique des scores et des dates) sont stockées dans un fichier JSON nommé utilisateurs.json.
- Chaque utilisateur dispose également d'un fichier CSV individuel pour l'enregistrement détaillé de ses scores.

2. Gestion des questions

- Les questions du quiz sont stockées dans un fichier JSON nommé questions.json.
- Les questions sont organisées en **catégories**, chaque catégorie contenant une liste de questions, leurs options, et la réponse correcte.

3. Système de qcm interactif

- L'utilisateur choisit une catégorie, puis répond à une série de questions avec une limite de 10 secondes pour chaque question.
- Le score est calculé en fonction des réponses correctes et des délais respectés.
- Les réponses correctes sont affichées pour chaque question incorrecte.

4. Gestion des scores et historique

- Les scores obtenus par un utilisateur sont sauvegardés avec la date et l'heure.
- Les utilisateurs peuvent consulter leurs scores enregistrés via un menu interactif.

3-Description des fonctions utilisées :

1-La fonction recup_utilisateur() :

Objectif : Récupérer les données des utilisateurs enregistrées dans fichier JSON utilisateurs.json.

Fonctionnement :

- Vérifie si le fichier utilisateurs.json existe.
- Si le fichier existe, il le charge et retourne son contenu sous forme de dictionnaire Python.
- Si le fichier n'existe pas, il retourne un dictionnaire vide.

2-La fonction sauvegarde_utilisateur(utilisateur) :

Objectif : Sauvegarder les données des utilisateurs dans le fichier utilisateurs.json.

Fonctionnement :

- Convertit l'objet Python utilisateur en JSON et l'écrit dans le fichier utilisateurs.json.
- Si le fichier n'existe pas, il est créé.

3-La fonction recup_creer_user() :

Objectif : Gérer la connexion ou la création d'un nouvel utilisateur.

Fonctionnement :

- Demande à l'utilisateur de saisir son nom d'utilisateur.
- Si l'utilisateur existe déjà dans le fichier, il affiche son historique (date et score).
- Sinon, il crée un nouveau profil pour l'utilisateur avec un historique vide, puis l'enregistre

4-La fonction recup_questions() :

Objectif : Charger les questions à partir du fichier questions.json.

Fonctionnement :

- Vérifie si le fichier questions.json existe.
- Si le fichier existe, il charge les questions.
- Si le fichier n'existe pas, un message d'erreur est affiché.

5-La fonction selection_categorie(questions) :

Objectif : Permettre à l'utilisateur de choisir une catégorie parmi celles disponibles.

Fonctionnement :

- Affiche les catégories des questions.

- Demande à l'utilisateur de choisir une catégorie en entrant un numéro.
- Valide l'entrée et retourne la catégorie sélectionnée.

6-La fonction convertir_time_en_seconde(time) :

Objectif :Convertir une heure en secondes pour faciliter la gestion du chronomètre.

Fonctionnement :

- Prend une instance datetime et retourne le temps total en secondes (heures × 3600 + minutes × 60 + secondes).

7-La fonction afficher_score(username) :

Objectif : Afficher les scores de l'utilisateur depuis son fichier CSV.

Fonctionnement :

- Lit le fichier CSV associé au nom d'utilisateur.
- Affiche chaque ligne du fichier, correspondant à un score enregistré.

8-La fonction remplir_csv(username, Score, Date) :

Objectif : Enregistrer les résultats du quiz dans un fichier CSV spécifique à l'utilisateur.

Fonctionnement :

- Ouvre ou crée un fichier CSV nommé avec le nom d'utilisateur.
- Si le fichier est vide, écrit les en-têtes (Utilisateur, Score, Date).
- Ajoute une ligne contenant le nom de l'utilisateur, son score et la date.

9-La fonction jouer_quiz(questions, categorie) :

Objectif : Faire jouer l'utilisateur au quiz et calculer son score.

Fonctionnement :

- Affiche les questions de la catégorie sélectionnée avec des options.
- Mesure le temps de réponse de l'utilisateur. Si la réponse dépasse 10 secondes, elle n'est pas prise en compte.
- Calcule le score en fonction des bonnes réponses.

10-La fonction mise_a_jour_historique(utilisateur, username, score) :

Objectif : Mettre à jour l'historique de l'utilisateur après un quiz.

Fonctionnement :

- Ajoute un nouvel enregistrement (date et score) dans l'historique de l'utilisateur.
- Sauvegarde les données mises à jour dans le fichier JSON utilisateurs.json.

11- La fonction principale main() :

Objectif : Point d'entrée principal du programme.

Fonctionnement :

- Gère la connexion ou l'inscription de l'utilisateur.
- Charge les questions disponibles.
- Permet à l'utilisateur de choisir une catégorie et de jouer au quiz.
- Sauvegarde les résultats dans un fichier CSV et met à jour l'historique JSON.
- Offre la possibilité de consulter le score après le quiz.

4-Conclusion :

Ce programme de QCM interactif combine efficacement la gestion des utilisateurs, la structuration des questions, et un système de quiz chronométré. Il permet à chaque utilisateur de s'immerger dans une expérience d'apprentissage personnalisée, tout en suivant ses progrès grâce à des historiques détaillés enregistrés en fichiers JSON et CSV. Grâce à sa conception modulaire, chaque composant est indépendant et facile à comprendre, facilitant ainsi l'ajout de nouvelles fonctionnalités ou catégories de questions. Ce projet représente un outil pratique pour tester ses connaissances tout en améliorant ses compétences de programmation Python.