People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

PROJECT REPORT:

QCM For CS Students

Team :

· CHABATI Rayan
· AMMOUCHE Abderraouf
· NAILI Walid
· BOUZNAD Riyad Mostefa

Program: Engineer, 3rd year

Specialization: Information Security

Supervised by: Mr MOUHOUN SAID

Academic Year: 2024 /2025

Table of Contents

I. **Introduction**

II. **Choix Techniques**

III. **Fonctionnalités Implémentées**

IV. Défis Rencontrés et Solutions

V. **Conclusion et Améliorations Possibles**

Introduction

Objectif

Ce projet avait pour but de développer une application Python permettant aux étudiants de répondre à des QCM, d'évaluer leurs performances et de suivre leur historique. L'application devait être conviviale et capable de gérer plusieurs utilisateurs.

Contexte

Dans le cadre de l'apprentissage, cette application devait aussi intégrer des notions avancées comme la gestion des fichiers, la validation des entrées et l'implémentation d'un chronomètre pour en apprendre d'avantage sur le langage Python.

Le projet est implémenté en Python.

Methodologie

Le chef de groupe a créé un repo, il nous a invité en tant que contributeurs et chaque membre a créé sa propre branche afin de commit et push dans cette dernière. Après avoir push nos fonctions assignées, nous les avons regroupées dans un seul fichier (situé dans main branch).

Choix Techniques

Structures de données

Les éléments clés suivants ont été organisés dans des structures appropriées :

- **Questions et réponses** : Dictionnaire contenant les questions, options et réponses correctes.
- **Utilisateurs et historique des QCM** : Listes de dictionnaires stockées dans un fichier JSON.

Format de stockage

Le format JSON a été utilisé pour son accessibilité, sa compatibilité avec Python et sa capacité à gérer des structures de données imbriquées. Contrairement au format CSV, qui est mieux adapté pour des données tabulaires simples, JSON permet de représenter facilement des relations complexes comme des listes de participations associées à chaque utilisateur. Deux fichiers principaux ont été gérés :

- participants.json : Enregistre les informations des utilisateurs.
- Qcm.json : Contient les questions du QCM.

Langage et bibliothèques

- **Langage** : Python, pour sa simplicité et ses nombreuses bibliothèques.
- **Modules** :
    - threading : Implémentation des chronomètres.
    - datetime : Gestion des dates des participations.
    - json : Lecture et écriture des fichiers JSON.

Fonctionnalités Implémentées

Gestion des utilisateurs

- Identification d'un utilisateur existant ou création d'un nouveau profil.
- Affichage de l'historique des participations (dates et scores).

QCM

- Chargement des questions depuis un fichier JSON.
- Affichage des questions avec options.
- Validation des réponses et calcul des scores.
- Feedback sur chaque question avec indication des bonnes réponses en cas d'erreur.

Gestion du temps

Deux modes de chronométrage :

1. **Par question** : Limitation du temps de réponse à chaque question.
2. **Global** : Temps total alloué pour l'ensemble du test.

Sauvegarde et historique

- Les scores et dates de participation sont sauvegardés dans le fichier participants.json.
- L'historique complet est affiché lors de la connexion de l'utilisateur.

Défis Rencontrés et Solutions

Collaboration avec Git

Problème : Travailler en équipe a requis la création et la gestion de branches Git, ainsi que le merge des modifications pour éviter les conflits.

Solution : Mise en place d'une stratégie Git claire, comprenant la création de branches par fonctionnalité, des commits réguliers et l'utilisation d'outils comme les pull requests pour faciliter les revues de code.

Validation des entrées

Problème : Gérer les entrées invalides sans interrompre l'exécution.

Solution : Implémentation de boucles de validation et gestion des exceptions.

Gestion des fichiers JSON

Problème : Assurer l'intégrité des données lors de l'écriture simultanée par plusieurs fonctions.
Solution : Utilisation de vérifications conditionnelles pour charger ou sauvegarder en toute sécurité.

Synchronisation des timers

Problème : Empêcher les conflits entre les chronomètres global et par question.

Solution : Ajout de conditions pour vérifier l'état du temps global avant de continuer.

Conclusion et Améliorations Possibles

Résumé des Fonctionnalités

L'application répond à toutes les exigences de base :

- Gestion des utilisateurs et historique.
- Questions, réponses, et évaluation.
- Chronométrage et feedback.
- **Catégories de questions** : Permettre de sélectionner une thématique spécifique (Python, Réseaux, Algorithmes, DB, Animes).
- **Exportation des données** : Ajout d'un mode d'exportation des QCM réalisés, des résultats et des scores au format texte.

Améliorations Futures

- **Interface utilisateur** : Développement d'une interface graphique.