UCLouvain

UCLouvain - Sciences Informatiques

APP2 - Groupe B.20 - Rapport
QCM

Groupe :
Enzo ANDRADE ORLETTI
Jad M'RABTI
Viktor PROFIROV
Pedro SIBOMANA

Table des matières :

22 octobre 2024

1 Introduction

Ce projet répond à une demande spécifique d'un client, visant à développer un logiciel permettant de créer, gérer et corriger des QCM de manière automatisée. Conçu pour les élèves et les enseignants, le logiciel a pour objectif de faciliter l'apprentissage et la gestion des examens, tout en réduisant les risques de tricherie. Il s'appuie sur des fichiers textes structurés, garantissant une création et une correction des QCM plus rapides et efficaces.

2 Tâches du projet

2.1 Conception des questionnaires

La première étape du projet consiste à concevoir des questionnaires de type QCM adaptés aux besoins des élèves. Cela implique la formulation de questions pertinentes en fonction du niveau d'étude. L'objectif est de permettre aux élèves de tester leurs connaissances de manière efficace et précise. Les questions doivent être claires, variées et bien structurées, afin d'assurer une évaluation juste et équilibrée.

2.2 Création du PRNG

2.2.1 Tâche INGinious

Après avoir lu le document qui nous a introduit à la thématique des pseudo-générateurs de nombres aléatoires, nous avons testé notre capacité de compréhension et d'application des connaissances acquises à travers celui-ci dans la tâche INGinious. Dans cette tâche, il nous fallait créer des tests garantissant la qualité du PRNG. Étant donné que la plateforme elle-même nous fournissait de petits indices sur les cas que les tests devaient couvrir, nous avons adopté la stratégie suivante : D'abord, nous avons soumis un code vide afin de visualiser toutes les conditions proposées par INGinious, puis nous avons développé le code en traitant chaque condition une à une littéralement. De cette manière, nous avons pu mieux contrôler ce qui répondait déjà aux exigences et ce sur quoi nous devions encore travailler.

2.2.2 Implémentation dans le code

Dès que nous avons réussi la tâche INGinious, nous avions donc le code pour tester le PRNG. Pour le créer, nous avons utilisé la bibliothèque `random`. Nous avons ensuite créé la classe `PRNG`, qui avait comme attributs initiaux une `graine` et un `intervalle`, ainsi que pour méthode la fonction `next_int()`. Cette fonction fait appel à une autre méthode de la bibliothèque `random` qui génère des nombres aléatoires, et dans ce cas, nous avons défini que ces nombres seraient compris entre 1 et la valeur de l'intervalle (non inclus). Une fois la classe créée, nous avons constaté dans notre propre console que deux instances de notre `PRNG` avec des valeurs de `graines` et d'`intervalles` identiques généraient toujours les mêmes séquences, et inversement.

2.3 Mélange aléatoire des réponses

Afin d'éviter les tentatives de tricherie et d'assurer l'unicité de chaque test, l'utilisation d'un PRNG (Pseudo-Random Number Generator ou générateur de nombres pseudo-aléatoires) est primordiale. C'est ici que notre PRNG (générateur de nombres pseudo-aléatoires) entre en jeu. La stratégie que nous avons élaborée est la suivante : créer une copie de la liste originale, mais uniquement avec les options de réponses ; créer une sorte de mémoire sous forme de liste, qui ne contiendra que des zéros

au début ; générer un nombre entier aléatoire compris entre 0 et l'indice de la dernière réponse dans la liste ; récupérer l'élément grâce à l'indice généré ; le PRNG génère un nouvel indice aléatoire qui sera l'adresse de la réponse dans notre mémoire ; après avoir répété ce processus pour toutes les réponses, nous créons une nouvelle liste sans les zéros et uniquement avec les réponses mélangées.

2.4 Définition des modes de cotation

Dans le projet, nous avons implémenté trois modes de correction : gentille, sévère et neutre. Par défaut, la correction gentille et sévère augmentait de 1 point la note de l'utilisateur pour chaque réponse correcte. En cas de réponse incorrecte, en mode gentil, l'utilisateur ne perdait aucun point, contrairement à la correction sévère, qui retirait un point. Cependant, l'objectif principal de cette section doit être le développement du mode de correction neutre, qui a été le seul créé de manière totalement autonome par le groupe, c'est-à-dire que nous devions en définir le fonctionnement. Après une réflexion collective, nous avons eu l'idée de mettre en place un pourcentage de réponses correctes par question et d'incrémenter la note de l'utilisateur en fonction de ce pourcentage. L'explication de cette relation se trouve plus loin dans le rapport, dans la section « 4. Problèmes rencontrés ».

3 Fonctionnement du Groupe

Tout d'abord, le groupe a fait preuve d'efficacité et de sérieux tout au long du projet, ce qui nous a permis d'atteindre rapidement un résultat satisfaisant. Nous avons bénéficié d'une bonne synergie dès le début, ce qui nous a permis d'être rapidement à l'aise. Nous avons pu jongler entre humour et sérieux, favorisant ainsi une bonne harmonie et une atmosphère agréable au sein du groupe. De plus, nous avons également travaillé en dehors des cours pour résoudre les différents problèmes rencontrés. Nous posions nos questions lorsque certains points du code n'étaient pas clairs, évitant ainsi qu'un membre du groupe ne se retrouve perdu ou en retard. Enfin, nous n'avons pour l'instant aucun point faible à signaler concernant le groupe. Aucune faille n'a entravé notre progression durant la réalisation du projet.

4 Problèmes rencontrés

4.1 Malus

Premièrement, nous avons eu l'opportunité de relever un défi intéressant avec la création du malus pour le mode de cotation neutre. Nous avons exploré différentes formules pour élaborer un malus, et finalement, nous avons découvert une formule efficace qui calcule le pourcentage de bonnes réponses par question. Ainsi, plus le pourcentage est élevé, moins l'utilisateur gagnera de points en cas de bonnes réponses, et vice-versa. Cette formule est : $pourcentage = compteur\_bonnes\_reponses \,/\, total\_options$.

4.2 Réponses 'out of range' et/ou type not int

Deuxièmement, il y a eu un problème assez commun : celui du *out of range*. Lorsqu'un utilisateur donnait une réponse qui n'était pas parmi celles proposées, le programme passait directement à la question suivante. Pour y remédier, nous avons numéroté les réponses et créé une condition indiquant

que l'utilisateur recevra un message disant que cette option n'existe pas, et qu'il doit en fournir une qui soit valide. La même situation se présente si la réponse de l'utilisateur n'est pas un nombre entier.

4.3 Option 'Toutes les réponses précédentes'

4.3.1 Sens du mot 'précédentes'

Troisièmement, il fallait gérer l'option *Toutes les réponses précédentes*, qui nous a posé plusieurs problèmes. Après l'insertion de notre PRNG pour le mélange des réponses, cette option était parfois placée au début ou au milieu des choix possibles, ce qui n'a pas de sens. Étant donné que le mot "précédentes" implique qu'il y ait des options antérieures, cette réponse ne peut pas être mélangée avec les autres et doit toujours être imprimée en dernière position. Ainsi, dans la partie du code réservée au mélange aléatoire des réponses, nous avons créé une condition qui vérifie systématiquement si la réponse est *Toutes les réponses précédentes*. Si c'est le cas, elle est placée comme dernier élément en mémoire pour garantir qu'elle soit toujours affichée en dernière position.

4.3.2 'Toutes les réponses' : TRUE => Reste d'options : TRUE

De plus, si cette option était vraie, cela impliquerait que toutes les autres soient également vraies. Si l'utilisateur choisissait cette option, le programme lui donnait la possibilité de sélectionner d'autres réponses, même si cette option couvrait déjà toutes les bonnes réponses. Nous avons résolu ce problème en définissant que l'utilisateur ne pouvait répondre qu'une seule fois et, s'il choisissait 'Toutes les réponses précédentes', il gagnerait 1/1 point. S'il optait pour toute autre réponse, il gagnerait 1 point divisé par le nombre de réponses possibles.

5 Conclusion

Tout comme dans l'APP1, la communication a constitué un élément fondamental au sein de notre groupe dans la réalisation de ce nouveau projet. De nouveaux défis ont émergé au cours de notre parcours, mais grâce à celle-ci, nous les avons surmontés avec une grande efficacité et avons réussi, en fin de compte, à respecter toutes les exigences de notre planning. Étant donné que le code présentait des proportions plus importantes par rapport à celui du premier projet, le regard extérieur de chaque membre du groupe a été essentiel pour identifier les points à améliorer chez les autres, dans les cas où un membre ne les percevait pas jusqu'alors. Cela a permis une mise à jour constante du code jusqu'à ce que nous parvenions à la version finale sous la forme la plus soignée possible.

Grâce à l'APP2, nous avons acquis des compétences en matière de création de classes et d'implémentation de méthodes, d'utilisation de bibliothèques en Python, de définition et d'allocation des fonctions aux emplacements appropriés, et surtout, sur la manière de gérer les listes et de traiter les erreurs.

Enfin, un point important à souligner est la capacité du groupe à relier des concepts et des normes du monde réel à la programmation. À partir de l'exemple cité de l'option "Toutes les réponses précédentes", le groupe a su proposer de nouvelles solutions et démontrer de nouvelles façons d'interpréter la question, sans se limiter à la simple mise en œuvre d'un code dénué de sens et d'applicabilité dans la réalité.