Cégep de Sainte-Foy
Développement mobile et objets connectés – 420-W50-SF

EFCS
Jeu de mémoire

Préparé par
Daniel Huot et Karine Filiatreault

1 Résumé

Ce travail vise à réaliser un **jeu de mémoire connecté** combinant un montage IoT (ESP32 Arduino) et une **application mobile React Native** interagissant via **Firebase Firestore**. L'objectif est de produire un jeu où :

- l'ESP32 gère la logique du jeu de mémoire et enregistre les scores dans *Firestore*;
- une application mobile permet de consulter les joueurs, envoyer des défis, et afficher un classement dynamique.

2 Conditions de réalisation

Valeur de la note finale	Contexte	Nombre de remises	Durée
30%	Équipe de 2	1	3 semaines

3 Tâches à réaliser pour l'application mobile

1) L'équipe créera d'abord un projet sur *Firebase* avec authentification courriel/mot de passe pour la partie mobile, anonyme pour la partie IoT.
2) L'enregistrement dans l'application mobile permettra de saisir un courriel, mot de passe, pseudonyme, prénom, nom de famille et avatar (en *localStorage*).
3) L'authentification mobile créera un utilisateur « applicatif » sur le *Firestore* qui contiendra au minimum les informations suivantes : `userId` (celui obtenu de *Firebase*), pseudonyme, nom et prénom.
4) À l'instar des *pokemons*, on aura une icône de maison (accessible en tout temps dans l'application) qui listera les utilisateurs inscrits dans le système (nom et prénom), il sera cependant impossible de défier un utilisateur lorsque non authentifié.
5) Lorsque authentifié, le menu avatar dans le coin supérieur droit montrera 5 choix :
    a. La liste des défis reçus d'un autre utilisateur (tab)
    b. La liste des défis complétés (tab)
    c. Le tableau des meneurs (tab)
    d. Le profil de l'utilisateur connecté (tab)
    e. Déconnexion

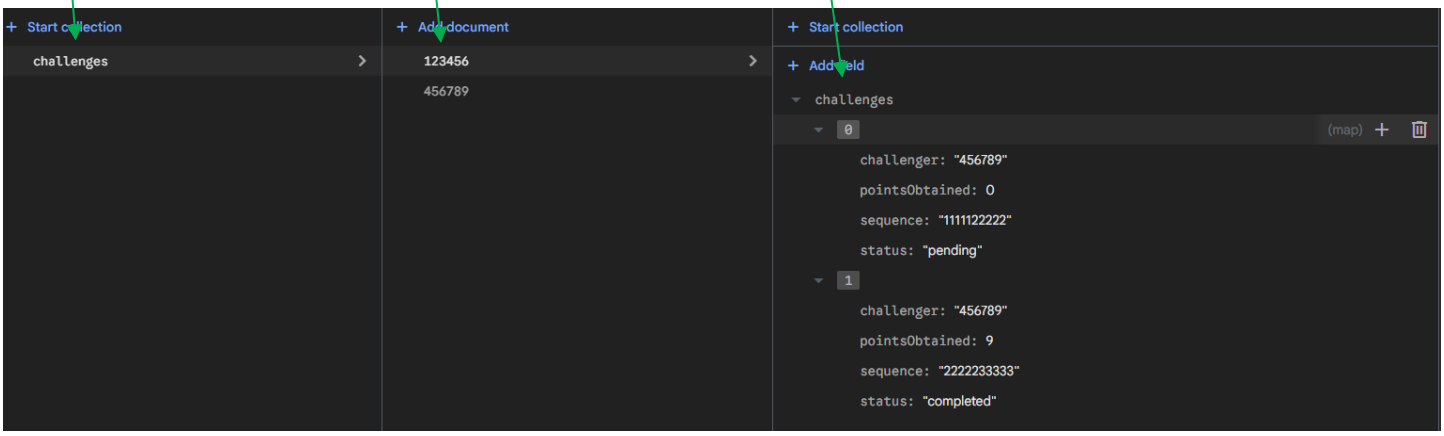
3.1 Création d'un défi

Un défi sera créé selon la structure suivante dans le *Firestore* (**il est impératif de reproduire cette structure sans quoi les défis ne pourront être récupérés sur le montage IoT**) :

La collection s'appelle `challenges`

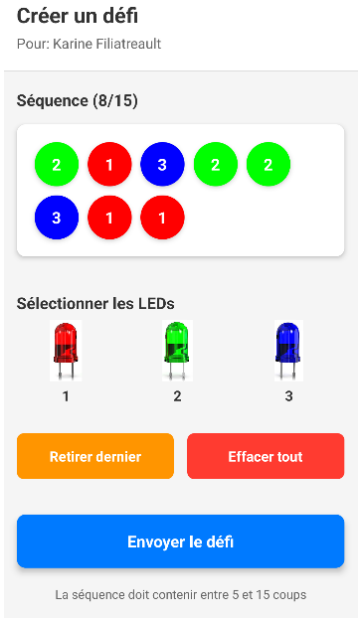
Le nom du document est le `userId` que l'on veut défier

Le *field* s'appelle `challenges`, c'est un array. Il contient des maps qui représentent chacun un défi (l'utilisateur qui a créé le défi, le nombre de points obtenus, la séquence que l'adversaire devra jouer, le statut du défi (pending/completed))



Dans la page d'accueil, une fois authentifié, il sera donc possible de défier un utilisateur inscrit dans le système. La page de création de défi permettra de saisir une séquence entre 5 et 15 coups de longueur, séquence pour laquelle on indiquera quel *led* (1, 2 ou 3) s'allumera à chaque coup.

Vous devez faire un écran étoffé et convivial pour cette page. Voici un exemple, les couleurs sont à titre indicatif, l'important étant l'enregistrement de la séquence avec les positions (voir image ci-haut) :



3.2 Page des défis reçus

Cette page (sous forme de *tab*) affichera la liste des défis que vous avez reçus. Il va de soi qu'elle n'affichera pas la séquence ! Elle affichera les informations suivantes :

a. Le pseudonyme de celui qui vous l'a envoyé
b. La longueur de la séquence que vous devrez réaliser
c. Le nombre de points potentiels que le défi pourrait vous rapporter et le nombre de points potentiels que vous pourriez infliger (déduire) à votre adversaire (*Voir système de pointage plus loin dans l'énoncé)

3.3 Page des défis complétés

Cette page (sous forme de *tab*) affichera la liste des défis que vous aurez complétés. Pour chaque défi (que l'on pourra nommer Défi 1, Défi 2, etc.) on affichera :

a. Le nombre de coups réussis et le nombre de coups demandés
b. Les points obtenus/perdus
c. Les points obtenus/perdus de l'adversaire

3.4 Page du tableau des meneurs

Cette page (sous forme de *tab*) affichera les utilisateurs et leurs points en partant du meneur. La technique utilisée pour stocker ou calculer les points est libre à vous. Il est toutefois à noter que l'on pourra défier directement un utilisateur depuis cette vue.

3.5 Page du profil utilisateur

Cette page (sous forme de *tab*) affichera le profil complet de l'utilisateur :

a. Pseudonyme
b. Prénom et nom
c. Avatar (avec possibilité de le changer)
d. Nombre de points courant
e. Position courante dans le classement
f. Section pour changer le mot de passe

3.6 Déconnexion de l'utilisateur

Cette action déconnecte l'utilisateur et retourne à la page d'authentification / création de compte

4 Tâches à réaliser pour la partie IoT

Matériel obligatoire au montage :

1) Microcontrôleur Esp32
2) 3 *leds* distincts et leurs résistances
3) 3 boutons poussoirs
4) Le joystick
5) L'écran LCD

Important : Tous les membres de l'équipe devront effectuer le montage IoT afin de se défier mutuellement dans la vidéo de démonstration.

4.1 Modification du jeu de mémoire

Le jeu de mémoire du TP2 devra être modifié afin de jouer une séquence d'un trait et ensuite demander à l'utilisateur de confirmer celle-ci d'un seul coup (avec *feedback* visuel). Prévoir un défilement normal de la séquence et un défilement rapide. On gardera toujours le nombre de coup réussis et on saura quand la partie se termine (suite à une erreur ou suite à la réalisation complète de la séquence).

4.2 Récupération des défis

Copiez la librairie « FirestoreChallenges » fournie avec l'énoncé dans le répertoire « lib ». Cette librairie permettra d'interagir avec la structure Firestore illustrée auparavant dans l'énoncé. Ouvrez-la pour voir les méthodes disponibles.

```
public:
  FirestoreChallenges(FirebaseData* firebaseData, String projId);

  bool loadChallenges(String playerId);
  int getCount();
  FirestoreChallenge getChallenge(int index);
  bool updatePoints(int index, int newPoints);
  void printAll();
```

Il va de soi que si la structure n'est pas adéquatement reproduite, cette librairie ne fonctionnera pas !

Dans votre code, voici comment créer une instance du *manager*, celui qui vous donnera accès aux méthodes de la librairie (remplacez avec votre projectId *Firebase*) :

```
FirebaseData fbdo;
FirestoreChallenges challengeManager(&fbdo, "__VOTRE_PROJECT_ID_FIREBASE__");
```

La librairie récupère un maximum de 5 défis non complétés à l'aide de la méthode **`loadChallenges`** qui nécessite le `id` de l'utilisateur. Chaque défi reçu est stocké dans la `struct` **`FirestoreChallenge`**. La méthode **`loadChallenges`** crée en privé un tableau de 5 **`FirestoreChallenge`** et vous pouvez obtenir les défis avec le **`getChallenge`**.

L'écran LCD affichera les défis non complétés (max 5), on pourra les nommer Défi1, Défi2, … mais ils devront contenir l'information de la longueur de la séquence demandée.

4.3 Réalisation d'un défi

Cliquer sur un défi nous amène à un écran permettant de le jouer en mode expert ou normal, ceci impactera la vitesse de défilement de la séquence et le nombre de points gagnés/infligés.

Lorsque que le défi commence :

a. L'écran LCD affiche « défi en cours »
b. Les *leds* clignotent 3 fois pour annoncer le début de la séquence.
c. Le montage IoT défile la séquence au complet et attend par la suite la confirmation de celle-ci avec les boutons poussoirs (*feedback* visuel inclus)

Lorsque le défi se termine (échec ou la séquence a été reproduite au complet) :

a. Tout s'éteint sur le montage (pas de clignotement)
b. Les points obtenus et le statut du défi sont mis à jour sur le *Firestore* (inclus dans la librairie)
c. L'écran LCD affiche « partie est terminée, défi échoué (ou relevé) », le nombre de points gagnés, le nombre de points infligés et une option de retour au menu

De retour au menu, on montrera à nouveau un maximum de 5 défis non réalisés.

4.4 Système de pointage

Vous devez mettre en place un système de pointage compétitif et réaliste pour lequel le mode (expert/normal) aura une influence, tout comme la longueur de la séquence. Dans tous les cas, il sera possible de perdre/gagner des points pour les 2 joueurs :

Exemple limite 1 : Défier un adversaire avec une longue séquence qu'il réussira en mode expert pourrait vous faire perdre beaucoup de points et en faire gagner beaucoup à l'adversaire.

Exemple limite 2 : Défier un adversaire avec une courte séquence qu'il ratera en partant en mode normal pourrait vous valoir beaucoup de points et lui en faire perdre beaucoup.

Remise

En plus du code de l'application qui devra être remis sur LEA, les étudiant(e)s remettront une vidéo, filmée adéquatement et narrée de façon claire et audible, qui présentera les aspects suivants de leur projet. Avant celle-ci, les étudiant(e)s effaceront tous les documents de leur base de données :

La vidéo doit inclure (*voir note pour un étudiant seul) :

1) Démonstration que le *Firestore* est vide.
2) L'inscription des 2 joueurs sur l'application mobile.
3) L'apparition des 2 joueurs dans la page d'accueil avec les boutons de défis présents (car authentifié) et les *tabs* dans l'application.
4) Un des 2 joueurs se déconnecte et montre que les boutons de défis sont absents et que les *tabs* ne sont pas accessibles, il se reconnecte ensuite.
5) Les 2 joueurs montrent que leurs listes de défis reçus, défis complétés sont vides dans l'application mobile, ils montrent aussi que les joueurs ont 0 point dans le tableau des meneurs.
6) L'utilisateur A envoie un 2 défi à l'utilisateur B. L'utilisateur B démontre les 2 défis dans sa page des défis à réaliser de l'application mobile et que sa liste des défis réalisés est toujours vide.
7) On montre l'état du *Firestore*.
8) L'utilisateur B accède son IoT, il joue un des défis en mode normal. L'écran LCD affiche que le défi est en cours.
9) Le joueur réalise le défi (il doit réussir le défi), l'écran montre que le défi est relevé ainsi que les points qui ont été attribués et perdus.
10) Retour au menu LCD, 1 défi restant est affiché.
11) On démontre que le défi est complété sur l'application mobile avec les données du défi.
12) On démontre qu'il reste 1 défi dans la liste des défis à compléter.
13) On montre l'état du tableau des meneurs.
14) L'utilisateur B défie l'utilisateur A une seule fois.
15) On montre l'état du *Firestore*.
16) L'utilisateur A reprend les étapes 8) à 13) en mode expert (le LCD affichera cette fois qu'il n'y a pas de défi actif au retour). Le joueur doit perdre.
17) Les 2 utilisateurs montrent leurs profils respectifs.
18) L'utilisateur B effectue un changement de mot de passe, se reconnecte et refait un tour rapide des *tabs* de l'application.
19) L'utilisateur A change son avatar, se reconnecte et montre son écran de profil.

Finalement, les étudiants commenteront les parties clés de leur code (IoT et mobile), notamment la logique d'affaire cruciale au fonctionnement du jeu, réserver un 3 minutes minimum au total sur cet aspect. On ajoutera à ce 3 minutes une description détaillée et claire du système de pointage mis en place. La qualité de la description du code et du système de pointage sera fortement prise en compte dans la correction.

Note : Si la vidéo est trop volumineuse, un lien de partage vers celle-ci sera inclus dans le haut du fichier main.cpp

(*) Si un étudiant est seul, il devra se créer 2 utilisateurs dans l'application mobile afin d'incarner les utilisateurs A et B. Il ne devra pas oublier de changer le `userId` de son montage afin de montrer les défis des 2 utilisateurs.

Collaboration et plagiat

Une évaluation sommative **ne peut en aucun temps** être le produit d'une **collaboration** ou de **partage** de code entre des personnes ou équipes différentes. **Toute collaboration, partage de code, ou autre interaction sera sanctionnée selon la clause 6.1.12 du plan de cours :**

> Tout acte de plagiat, de tricherie et de fraude sera sanctionné. Constitue notamment un plagiat, une tricherie ou une fraude tout acte de faire passer pour sien un travail, ou toute autre production constituant une évaluation, réalisé en tout ou en partie par une autre personne ou par une **intelligence artificielle**; reproduire en tout ou en partie le travail d'une autre personne ou un travail généré par une **intelligence artificielle**, qu'il s'agisse d'un document imprimé, audiovisuel ou numérique, sans y faire expressément référence.
>
> *Extrait de : Politique d'évaluation des apprentissages du collège*

De plus, vous devez obligatoirement **citer vos sources** si vous vous inspirez de code trouvé sur internet ou provenant de Chat GPT ou autres. Pour les **IA**, indiquez le ***prompt*** utilisé pour interroger l'IA.

L'essentiel de votre travail ne doit pas provenir de l'IA.

Si de tels comportements sont observés, le professeur se réserve le droit de mettre fin à l'évaluation des étudiants fautifs en cours de réalisation et d'attribuer la note 0.

Modalités de remise

Remettez votre projet sur LÉA et la vidéo, dans la section travaux, à l'intérieur d'une archive *Zip*.

⚠ Important

Supprimez les fichiers et dossiers inutiles, c'est-à-dire :

- Cet énoncé et la grille d'évaluation.
- Si la vidéo est trop volumineuse, incluez son lien de partage dans haut du fichier main.cpp

Une pénalité de 15 % est appliqué en cas de retard de moins d'une journée. Au-delà de ce délai, le travail est refusé et la note de 0 % est automatiquement attribuée.

Note : Un projet qui ne compile pas se méritera la note de zéro (0).

Évaluation

Vous trouverez dans le fichier ci-joint la grille d'évaluation utilisée pour la correction de ce travail pratique.