CÉGEP STE-FOY – HIVER 2020
PROGRAMMATION MOBILE II – 420-V60-SF

Énoncé du Travail Pratique 3

3 février 2020

Préparé par
Benjamin Lemelin

1 Résumé

Créez une application multiplateforme permettant d'apprendre l'alphabet d'une langue (au choix, mais il est proposé d'utiliser les « Hiraganas »). Il y a deux modes : « Apprendre » et « Entrainement ».

2 Conditions de réalisation

Valeur de la note finale	Type	Durée	Nombre de remises
20 %	Individuel	2 semaines	1

3 Spécifications

3.1 Projet Android Studio

Importez le projet fourni et placez-le dans un dépôt Git. Donnez en accès à votre professeur.

3.2 Interface utilisateur

Apprendre

Entrainement

Entrainement (Erreur)

Il n'y a qu'une seule route (comprendre activité). Le mode « Apprendre » est affiché au lancement. Dans ce mode, l'usager se fait présenter les caractères de l'alphabet étranger (aussi appelés graphèmes) avec leur traduction. C'est une liste déroulante sous forme de grille de cartes.

Les onglets du bas permettent de passer du mode « Apprendre » au mode « Entrainement » et vice-versa. Il doit aussi être possible de passer d'un mode à l'autre en glissant l'interface de gauche à droite (et vice-versa). Aucune donnée ne doit être perdue lors du passage d'un mode à une autre.

Le mode « Entrainement » présente un caractère aléatoire de l'alphabet étranger et demande à l'usager de l'identifier parmi trois options. S'il fait le bon choix, un autre caractère lui est présenté. S'il fait le mauvais choix, ce dernier est affiché en rouge et le bouton correspondant est désactivé.

3.3 Données de l'application

L'application doit contenir elle-même ses données. Il n'y a donc pas de service web. Dans le cas des « Hiraganas », les caractères ainsi que leur traduction vous sont déjà fournis. Si vous ne vous sentez pas à l'aise avec cet alphabet, libre à vous d'en utiliser un autre et de fournir, vous-même, la traduction.

3.4 Spécifications techniques

Description
L'usage de « Flutter » est obligatoire.
Il doit être possible de passer d'un mode à un autre sans perte de données. Autrement dit, l'état de chaque mode doit être conservé en tout temps.

3.5 Autres spécifications

Description
Le code doit être en Dart.
L'interface de l'application doit supporter l'orientation portrait et paysage.
L'interface de l'application doit supporter l'anglais et le français.

4 Modalités de remise

Remettre sur LÉA un fichier texte incluant :

1. L'adresse vers le dépôt Git
2. Les matricules de chaque membre de l'équipe

Une seule équivaut à une pénalité de 15 %. Au-delà de ce délai, le travail la note « 0 » est attribuée.

5 Évaluation

Éléments
Fonctionnalités (incluant, mais sans s'y limiter) : • En mode « Apprendre », il est possible de consulter l'alphabet avec sa traduction. • Le mode « Entrainement » montre un caractère aléatoire. • Le mode « Entrainement » montre 3 choix de réponse différents. • Le mode « Entrainement » passe à un autre caractère si l'usager fait le bon choix. • Dans le cas d'une mauvaise réponse en mode « Entrainement », le choix est désactivé et mis en rouge. • L'un des 3 choix du mode « Entrainement » est valide.
Interface utilisateur (incluant, mais sans s'y limiter) : • Création d'interface (méthodes « build ») propres. • Visuel de l'interface utilisateur propre et stable. Interface disponible en anglais et en français.
Multiplateforme (incluant, mais sans s'y limiter) : • Utilisation correcte de « Flutter » pour le multiplateforme. • Utilisation des « StatelessWidget » et des « StatefulWidget » dans les cas appropriés. • Séparation correcte de l'application en plusieurs petits Widgets.
Qualité générale de l'application mobile (incluant, mais sans s'y limiter) : • Changement d'orientation et d'application supporté, sans perte de données.
Qualité générale du code (incluant, mais sans s'y limiter) : • Logique bien pensée, juste, et « non patché ». • Découpage adéquat du code en classes, méthodes et packages. • Architecture adéquate au type d'application développée. • Séparation raisonnable des responsabilités entre les classes. • Respects des standards du langage de programmation. • Nommage clair et sans ambigüité des éléments. • Utilisation de constantes, lorsque nécessaires. • Commentaires compensant uniquement le manque d'expressivité du code. • Respect des bonnes pratiques de programmation générales. • Code propre, sans résidus et facilement lisible. • Aucune erreur ni avertissement à la compilation.
Qualité générale du travail (incluant, mais sans s'y limiter) : • Configuration de projet fonctionnelle. • Respect des consignes de remise. • Aucun fichier inutile remis avec le projet.

La qualité de la langue française fait partie de l'évaluation. Chaque faute de français retire 0,5 % à la note finale jusqu'à concurrence de 20 %.