Rayane Rachid Kennaf et Lemar Andar

Projet synthèse

Groupe 01

# Projet synthèse :

# LOCs

Travail présenté à Jean-Christophe Demers

Cégep du Vieux-Montréal

18 mai 2023

Voici LOCs, l'application qui vous expose au gens autour de vous. Discutez, partagez et échangez avec eux. Que ce soient des photos, de la musique ou vos pensées, notre réseau social permettra un utilisateur de parler dans le chat public de sa localisation. De plus, votre identité est anonyme jusqu'au moment d'accepter l'échange d'information avec un autre utilisateur. Dites maintenant adieu à l'ennui et bonjour aux nouvelles connexions avec LOCs.

L'utilisateur rentre dans notre application et peut voir directement le chat public du lieu où il se situe. Si vous êtes dans le Cégep du Vieux-Montréal, Vous avez le droit de communiquer dans la discussion. Au cas où vous changez d'endroit, alors il ne sera plus possible de bavarder dans le chat public du Cégep du Vieux-Montréal. Vous avez accès à n'importe quel chat public autour de vous, mais pour y participer il faut s'y rendre comme dans Pokémon Go. Ces lieux peuvent être des restaurants, des bibliothèques, des festivales, des cinémas, etc. Vous comprenez alors que cette application est fondamentalement basée sur le système de géolocalisation. Pour faciliter la découverte de ces lieux, notre page principale vous affiche les endroits autour de vous. Avec la fonctionnalité de préférence de distance, la page vous affiche donc les localités selon votre choix. De plus, une carte vous permet de rechercher les fils de discussion partout dans le monde. Celle-ci permet aussi de voir en temps réel l'achalandage (moyenne de personnes qui utilise l'application sur place). Ce système vous permettra de joindre et d'éviter les lieux achalandés.

Ceci était la présentation initiale du projet et de ce qu'on avait pensé au début du projet. Quand au résultat final, nous estimons avoir réussi à faire 90% des fonctionnalités que l'on voulait initialement. Voici les choses que l'on peut faire dans l'application : chatter dans des chats public et privées, créer compte contenant un profil. Également, nous pouvons avoir selon la localisation de l'usager une liste de lieu qui sont des chatrooms et nous pouvons déterminer lequel est le proche et surtout de voir si l'usager est dans le lieu ou non, permettant ainsi à ce dernier d'être chattable ou non. L'usager peut également avoir une map des lieux autour de lieux et peut directement accéder au lieu au lieu d'utiliser un rayon de recherche. Il y a un certain filtrage sur les lieux que l'on peut ou non chatter. Le principe de de chattable ou non s'applique également sur la carte. L'usager peut également avoir un aperçu de son profil et peut le modifier en direct, l'usager peut également deloc une autre personne. Ceci est toutes les fonctionnalités que nous avons pu réaliser dans le laps de temps offert. Dans le 10% restants, nous n'avons pas eu le temps de faire en sorte que l'usager puisse voir les profil privées des usagers qu'il a deloc, de pas avoir un système de deloc dynamique auquel deux utilisateurs reçoivent une notification si un des deux usagers veut deloc avec l'autre.

En ce qui a trait au résume du développement de Locs, nous avions eu beaucoup de problème. Tout d'abord, le fait d'avoir utiliser React native nous a aider finalement, mais au début du projet, la compréhension de ce Framework fut difficile, car nous apprenions le langage tout en codant notre projet, rendant ainsi notre code peu élégant. Également, react native favorise davantage le développement d'application en utilisant le paradigme de programmation fonctionnelle au lieu de l'orientée-objet, ce qui rend également notre code peu réutilisable. Également, un design pattern en particulier qui est celui du Builder, à dû être abandonner par non-utilisation du paradigme de l'orientée-objet, donc aucune réelle manière d'implémenter un Builder. De plus, lors de la réalisation de L'UML, nous avions eu l'idée de créer des DAO pour tout ce dont on avait besoin pour l'app (usager, chat,

chatrooms), mais cela a de l'être abandonner, car avec le fait d'utiliser MongoDB Atlas, il fut presque impossible d'implémenter des DAO et fut obliger d'implémenter une api qui gère les insertions, update, delete avec des post/get routes. Les deux sources les plus importantes fut les vidéos tutoriels sur react native ainsi que chatgpt. Le premier nous a permis de comprendre la base de React-native tandis que le second nous a surtout aider à comprendre le code que l'on fessait et surtout debugger le code qu'on avait, car le debugger (javascript) n'est pas très user-friendly. Tant qu'aux évènements, il n'y en a pas eu réellement d'importants. Certes la démo que l'on avait présentée lors de la mi-session n'était pas celle que l'on voulait faire, mais ces des choses qui arrivent. Enfin, je trouve que la session en projet synthèse fut correcte, je pense qu'on avait quand même du temps pour le projet, même s'il fallait mettre beaucoup d'heures sur le projet en dehors des cours.

Voici maintenant toutes les fonctionnalités avec leur indication de s'il fonctionne ou non.

| **Fonctionnalités** | | | | | **Indication** |
|---|---|---|---|---|---|
| Clavardage dans un lieu public | | | | | fonctionelle |
| Clavardage dans un lieu privee (entre 2 personne) | | | | | fonctionelle |
| Calcul de distance entre les lieux/utilisateurs | | | | | fonctionelle |
| Affichage des lieux proche de l'utilisateur | | | | | fonctionelle |
| Login | | | | | fonctionelle |
| Register | | | | | fonctionelle |
| Deloc (affichage des donnees privees) | | | | | semi-fonctionelle |
| Affichage d'une map des lieux | | | | | fonctionelle |
| Gestion des connexions/deconnections | | | | | fonctionelle |
| Modification du profil public/privee | | | | | fonctionelle |
| Onglet aide | | | | | abandonne |
| Personnalisation interface | | | | | abandonne |
| Mise à jour | | | | | abandonne |

En termes d'amélioration possibles, c'est en majeur partie relier au code et surtout comme nous avons structure le code. Je pense que nous aurions dû forcer le fait d'avoir utiliser le paradigme de l'orientée-objet, car avec ce que nous avions appris dans notre cursus, nous aurions eu une codebase beaucoup clean avec très peu de répétition de code. Le problème avec notre code que c'est seulement du code qui permet de faire une faire quelque chose. Nous n'avions pas réellement respecté notre approche initiale que l'on avait penser dans notre UML et sa se remarque dans notre code. En effet, la principale amélioration de notre code réside dans son manque de modularité. Nous avons omis de suivre une approche orientée objet, ce qui aurait permis de structurer notre code de manière plus propre et réutilisable. En utilisant des concepts tels que l'encapsulation, l'héritage et le polymorphisme, nous aurions pu concevoir une architecture plus solide. Également, malgré qu'on ait pu faire la grande majorité des fonctionnalités que l'on voulait faire, notre code est très ''expérimental'' avec plusieurs méthodes. Par exemple : au début du projet, nous avions fait un login et un page register pour tester

notre api et on fessait les vérifications sur le login de façon mauvaise, mais sa marcher donc on la garder. Ce n'est que vers la moitie de la session que l'on sait rendu compte que sa n'allait pas marcher dans toutes les situations, donc nous avions rectifier pour le reste du code, mais pas le début. Ce qui fait en sorte qu'on a surtout un code très irrégulier pour le coup.

Pour l'auto-évaluation individuelle, nous pensons que notre projet est à la base quelque chose d'ambitieux qui aurait être un peu mieux réfléchi pour ainsi mieux démontrer les acquis de notre formation collégiale. Pour ce qui est de la satisfaction de la réalisation du projet, nous pensons être quand même fière de notre projet, car nous ne pensions même pas le réaliser au complet, car on fessait un projet toute en apprenant une technologie nouvelle. Pour notre pourcentage de réalisation, nous mettons à chacun de nous une note de 85-90%, car nous jugions avoir quand même mis beaucoup de temps sur le projet.

Pour l'auto-évaluation en équipe, il faut surtout comprendre comment le projet a été séparer initialement. Lemar s'occuper du Frontend (vue) et moi je fessais le backend (l'api, les fetch, les fonctions de calculs, etc.). Cependant, au fur et à mesure de l'avancement du projet, il est devenu évident que cette répartition n'était pas équilibrée en termes de charge de travail.

En effet, le développement du backend s'est révélé beaucoup plus complexe que prévue. J'ai dû faire face à des défis techniques et à des exigences de fonctionnalités qui ont pris beaucoup de temps à résoudre. En revanche, la partie frontend que Lemar a prise en charge s'est avérée relativement plus simple et moins exigeante en termes de temps et de ressources.

Cette disparité dans la charge de travail a eu un impact négatif sur notre productivité globale. J'ai été submergé par les tâches du backend et cela a entraîné des retards dans la livraison des fonctionnalités et a affecté notre capacité à avancer efficacement dans le projet.

Il est important de noter que la répartition initiale des responsabilités n'était pas intentionnellement déséquilibrée. Dans notre cas, il aurait été préférable de reconnaître plus tôt cette inégalité dans la charge de travail et de trouver des moyens de la rééquilibrer, soit en redistribuant les tâches, soit en apportant une assistance supplémentaire à la partie la plus chargée.

Voici ce qui conclut notre rapport final de LOCs. Malgré tous les soucis, nous sommes quand même fière d'avoir réaliser un projet de cette envergure et c'est pour le coup notre premier réel projet que nous avions coder et penser de A à Z.