Cahier des Charges

Introduction et Contexte

À la suite du cours de Design Science du Professeur Jean-Henry Morin que j'ai suivi le semestre passé, j'ai rendu un projet de recherche intitulé « RideQuest : a proposal for a mobile application to push people to travel using a ride sharing and public transportation system ». Le but de ce projet a été de penser une application mobile qui permettrait à une personne d'atteindre sa destination à l'aide de co-voiturage et des transports publics en une course. J'ai donc rendu une proposition d'un système qui permettrait en partie l'implémentation d'une telle application. J'ai donc dans l'idée de continuer cette proposition pendant Projet Transverse 1 en sélectionnant le projet n°10 (« Système de réservation de taxi ») car celui-ci permet un accomplissement partiel de mon objectif final, qui est de proposer une application entièrement implémentée pour mon projet de Bachelor.

Objectif du projet

L'objectif du projet est donc de créer le système permettant la réservation d'une voiture mise à disposition par le conducteur sur cette application (pour le moment fictive). Le conducteur doit pouvoir conduire le passager à une plateforme de transport public qui pourrait ensuite l'amener à la destination définie, de le rejoindre à la station convenue pour ensuite le conduire à la destination renseignée, ou de le conduire de l'origine à la destination si aucune option de transport en commun ne peut être trouvé ou si le passager a choisi de ne pas utiliser ce type de transport.

Description des exigences

Exigences fonctionnelles

Le système doit permettre aux utilisateurs de :

- Créer un compte (nom, prénom, ville, âge (pas de personnes de moins de x ans) (x est à voir selon les normes implémentées par les différents pays)) ;
- S'identifier comme conducteur ou passager ;
- (conducteur) Renseigner les informations de leur véhicule (nombre de sièges, marque, couleur, plaques,...) et leur position ;
- (passager) Renseigner leur position/point d'origine, leur destination, la date et l'heure souhaitées de la course ;
- (passager) Renseigner s'iel veut voyager seulement en voiture ou faire une partie en transport public dans la mesure du possible ;
- Annuler le voyage pour n'importe quelle raison ;
- Signaler l'autre personne au besoin ;
- Poster un avis sur l'autre personne ;
- Ne pas suivre l'itinéraire reçue s'iels ne le veulent pas ;
- (chauffeur) Accepter une course si ce.tte.x dernier.ère.x est intéressé.e.x.

Exigences non-fonctionnelles

Le système doit :

- Informer les conducteurs se situant dans un rayon de x KM (x est a décidé plus tard) qu'une nouvelle demande a été partagée par un passager ;
- Avertir le passager qu'un conducteur a accepté la course ;
- Effacer la demande de l'interface des autres conducteurs dès qu'un conducteur l'a accepté ;

- Ne pas accepter deux conducteurs pour une même course ;
- Marquer un conducteur comme « disponible » ou « non disponible » ;
- Marquer un passager comme « libre » (en attente d'un match), « en attente » (en attente du conducteur), « en course », « en transport public » et « idle » (pas de course demandée) ;
- Ne pas envoyer la demande de course à un conducteur qui est marqué comme « non disponible » ;
- Marquer un passager comme « en attente » et le conducteur comme « non disponible » dès que ce dernier a accepté la course ;
- Définir la destination du conducteur comme l'endroit où iel doit déposer le passager ;
- (Si le passager a répondu qu'iel voulait faire une partie de la course en transport public) chercher et lister les lignes de transport public pouvant amener le passager à sa destination se trouvant dans un rayon de x KM (x sera défini plus tard dans le projet) > Trouver celle qui permet d'arriver à la destination le plus rapidement possible en prenant en compte le temps de transport par voiture et l'horaire de la ligne > envoyer l'information au conducteur et au passager ;
- (Si le passager a répondu qu'iel voulait faire toute la course en voiture ou si aucun transport en commun n'a été trouvé) Trouver le chemin le plus rapide allant de l'origine à la destination ;
- (Si aucun transport en commun n'a été trouvé) prévenir le passager et lui demander s'il souhaite faire toute la course en voiture ;
- Si la course a été invalidée, demander une brève description de la raison ;
- Accepter à la fois une même personne dans la base de données des conducteurs ainsi que dans celle des passagers (une personne peut en effet remplir les deux rôles) mais pas en même temps.

Description générale / Architecture fonctionnelle

Pour ce que projet soit opérationnel, il faudra dans un premier temps créer la base de données qui contiendra différentes données, telles que :

- Données des conducteurs : numéro du conducteur, nom, prénom, âge, adresse électronique, mot de passe, position GPS, rating, nombre de courses entreprises, nombre de courses annulées, nombre de kilomètres parcourus, marque de la voiture, numéro de plaques, nombre de sièges, état (« (non) disponible »)
- Données des passagers : numéro du passager, nom, prénom, âge, adresse électronique, mot de passe, position GPS, rating, nombre de courses demandées, nombre de courses annulées, état (« libre/en attente/en course/en transport public/arrivé à destination »)
- Données des courses : numéro de la course, annulée ou non, si annulée donner la raison, heure de départ, heure d'arrivée temps, kilomètres, avis donné au passager, avis donné au conducteur

Ces données seront simulées à l'aide de Mockaroo.

Il faudra ensuite construire un algorithme de sélection qui analysera les positions GPS des conducteurs et du passager pour envoyer la demande aux conducteurs se situant dans le périmètre de x KM. La demande devra être supprimée une fois qu'un conducteur l'a accepté.

Un algorithme d'itinéraire devra être implémenté pour répondre aux demandes du passager (itinéraire voiture, transport en commun).

Une front-end rudimentaire devra être pensée et permettre aux utilisateurs de s'enregistrer/se connecter, d'entrer leur information, … (tout cela devra être envoyé à la base de données). Il sera demandé à l'utilisateur de choisir quel rôle celui-ci souhaite remplir, et en fonction de la réponse, afficher une front-end personnalisée:

- Conducteur : afficher la demande d'un passager avec ses informations générales de la personne et de la course et une map indiquant son emplacement, demander si ce dernier accepte la course, montrer l'itinéraire de la course ;
- Passager : afficher une fenêtre qui demande l'origine, la destination, l'heure et la date de la course voulue, afficher que la recherche d'un conducteur est en cours, afficher une map avec le tracé de la course et les positions du conducteur et passager matched, afficher les informations sur la course.

Une fois que ce dernier a été sélectionné et a accepté la course, une map avec le point d'origine, la destination, la position des deux personnes et le temps et nombre de kilomètres restant sera affichée sur leurs écrans. Il leur sera proposé de noter l'autre personne à la fin de la course. Un bouton « signalé » sera disponible tout au long de celle-ci si un problème survient.

Une implémentation de l'API de Google Maps et l'API des TPG ou encore la Swiss Mobility API des CFF sera nécessaire pour pouvoir définir l'itinéraire à suivre.

Parties prenantes

Deux groupes d'utilisateurs peuvent être nommés : les conducteurs (chauffeurs) et les passagers (clients). Les conducteurs offrent leurs services tandis que les passagers en demandent.

Livrables attendus

Les livrables attendus seront probablement une interface web implémenté grâce à Flask et HTML5 ou un programme ainsi qu'un document PDF contenant le rapport.

Notes

Je ne souhaite pas calculer le prix de la course, sachant qu'une partie non négligeable de mon projet était de trouver une solution pour que les passagers n'aient pas à payer tout en incitant les conducteurs à proposer leurs services. A voir avec vous s'il serait possible de ne pas le faire.

Je ne sais pas si un algorithme d'itinéraire serait long a implémenté et peut être pas nécessaire. A voir avec vous.