Cahier des Charges : Application de Covoiturage

Version : 1.0 **Date** : 19/09/2025 **Objectif** : Développer une application de covoiturage avec tarification dynamique basée sur la distance et les préférences utilisateurs.

---

1. Acteurs du Système

Acteurs
Acteur
Description
Conducteur
Utilisateur proposant un trajet et son véhicule.
Passager
Utilisateur réservant une place dans un trajet.
Administrateur
Gère les utilisateurs, les véhicules, et les litiges.
Système
Gère les calculs de distance, les notifications, et les paiements.

---

2. Fonctionnalités Principales

A. Gestion des Utilisateurs et Profils

Objectif : Permettre aux utilisateurs de s'inscrire, se connecter, et gérer leur profil.

Fonctionnalités - Utilisateurs
Fonctionnalité
Description
Acteurs Concernés
Inscription/Connexion
Création de compte et authentification via token.
Conducteur, Passager
Gestion du profil
Mise à jour des informations personnelles et préférences.
Conducteur, Passager
Gestion des préférences utilisateur
Définition des préférences (fumeur, musique, etc.).
Conducteur, Passager

Tâches techniques associées :

- Créer les APIs pour l'inscription/connexion (lien avec `profiles`, `session`).
- Développer l'interface de gestion du profil (front-end).
- Implémenter la gestion des préférences (table `user_preference`).

B. Gestion des Véhicules

Objectif : Permettre aux conducteurs d'enregistrer et gérer leurs véhicules.

Fonctionnalités - Véhicules
Fonctionnalité
Description
Acteurs Concernés
Ajout d'un véhicule
Enregistrement des détails du véhicule (marque, carburant, etc.).
Conducteur
Mise à jour des informations
Modification des détails du véhicule.
Conducteur
Affichage des véhicules disponibles
Liste des véhicules pour un conducteur.
Conducteur, Passager

Tâches techniques associées :

- Créer les APIs pour ajouter/mettre à jour un véhicule (lien avec `vehicles`, `vehicle_brand`, `fuel`).
- Développer l'interface d'ajout/modification de véhicule (front-end).

C. Gestion des Trajets

Objectif : Permettre aux conducteurs de proposer des trajets (ponctuels ou récurrents) et aux passagers de les réserver.

Fonctionnalités - Trajets
Fonctionnalité
Description
Acteurs Concernés
Création d'un trajet
Définition d'un trajet avec arrêts, prix/km, et véhicule.
Conducteur
Réservation d'une place
Réservation d'une place par un passager.
Passager
Gestion des trajets récurrents
Création de trajets récurrents (ex : tous les lundis).
Conducteur
Calcul du prix
Calcul automatique du prix en fonction de la distance.
Système
Annulation d'un trajet
Annulation par le conducteur ou le passager.
Conducteur, Passager

Tâches techniques associées :

- Implémenter les APIs pour créer un trajet (lien avec `trips`, `routes`, `trip_stops`).
- Développer l'interface de création de trajet (front-end).
- Implémenter le calcul du prix (back-end).
- Créer les APIs pour les trajets récurrents (lien avec `recurring_trip`).

D. Gestion des Réservations et Paiements

Objectif : Permettre aux passagers de réserver des places et de payer.

Fonctionnalités - Réservations
Fonctionnalité
Description
Acteurs Concernés
Réservation d'une place
Sélection d'un trajet et réservation.
Passager
Paiement en ligne
Paiement via carte, PayPal, etc.
Passager
Confirmation de réservation
Validation par le conducteur.
Conducteur
Annulation de réservation
Annulation par le passager ou le conducteur.
Passager, Conducteur

Tâches techniques associées :

- Implémenter les APIs pour les réservations (lien avec `bookings`).
- Intégrer un système de paiement (Stripe, PayPal, etc.).
- Développer l'interface de réservation et de paiement (front-end).

E. Gestion des Avis et Notifications

Objectif : Permettre aux utilisateurs de laisser des avis et recevoir des notifications.

Fonctionnalités - Avis et Notifications
Fonctionnalité
Description
Acteurs Concernés
Laisser un avis
Notation et commentaire après un trajet.
Passager, Conducteur
Recevoir des notifications
Alertes pour les réservations, messages, etc.
Tous
Messagerie interne
Échange de messages entre conducteurs et passagers.
Tous

Tâches techniques associées :

- Implémenter les APIs pour les avis (lien avec `reviews`).
- Développer le système de notifications (lien avec `notification`).
- Créer l'interface de messagerie (front-end).

F. Gestion des Arrêts et Itinéraires

Objectif : Permettre la création et la gestion des arrêts et itinéraires.

Fonctionnalités - Arrêts et Itinéraires
Fonctionnalité
Description
Acteurs Concernés
Ajout d'un arrêt
Définition des arrêts (départ, arrivée, intermédiaires).
Conducteur
Calcul des distances
Calcul automatique des distances entre arrêts.
Système
Affichage des itinéraires
Visualisation des trajets avec arrêts.
Tous

Tâches techniques associées :

- Intégrer une API de cartographie (Google Maps, OpenStreetMap) pour calculer les distances.
- Développer l'interface de gestion des arrêts (front-end).

G. Système de Fidélité (Optionnel)

Objectif : Récompenser les utilisateurs actifs.

Fonctionnalités - Fidélité
Fonctionnalité
Description
Acteurs Concernés
Accumulation de points
Points gagnés après chaque trajet.
Tous
Échange de récompenses
Utilisation des points pour obtenir des avantages.
Tous

Tâches techniques associées :

- Implémenter les APIs pour la gestion des points (lien avec `loyalty_points`).
- Développer l'interface d'échange de récompenses (front-end).

---

3. Répartition des Tâches par Équipe

Voici une suggestion de répartition des tâches pour une équipe de 3-4 personnes (ou pour toi-même sur plusieurs sprints) :

Équipe 1 : Back-end (APIs et Logique Métier)

- Implémenter les APIs pour les utilisateurs, profils, et sessions.
- Développer les APIs pour les véhicules et les trajets.
- Implémenter le calcul des distances et des prix.
- Intégrer le système de paiement.
- Créer les APIs pour les réservations, avis, et notifications.

Équipe 2 : Front-end (Interfaces Utilisateur)

- Développer les interfaces d'inscription/connexion.
- Créer les interfaces de gestion des profils et véhicules.
- Concevoir les interfaces de création de trajets et de réservation.
- Implémenter l'interface de messagerie et de notifications.

Équipe 3 : Base de Données et Intégration

- Finaliser le schéma de la base de données (corrections et optimisations).
- Mettre en place les scripts de migration et de peuplement.
- Intégrer les APIs de cartographie pour le calcul des distances.

Équipe 4 : Tests et Qualité (Optionnel)

- Rédiger les tests unitaires et d'intégration.
- Valider les cas d'usage et les scénarios de bord.

---

4. Diagramme de Flux Principal

Voici un exemple de flux pour la **réservation d'un trajet** :

1. **Passager** : Recherche un trajet (via `trips` et `routes`).
2. **Système** : Affiche les trajets disponibles avec les arrêts (`trip_stops`).
3. **Passager** : Sélectionne un trajet et réserve une place (`bookings`).
4. **Système** : Calcule le prix en fonction de la distance et du `price_per_km`.
5. **Passager** : Paie la réservation (`Payment`).
6. **Conducteur** : Reçoit une notification (`notification`) et confirme la réservation.

---

5. Technologies Recommandées

- **Back-end** :Php
- **Front-end** : Vue.js - TypeScript
- **Base de données** : MySQL.
- **Cartographie** : Google Maps API ou OpenStreetMap.
- **Paiement** : Stripe, PayPal, ou Mangopay.

---

6. Prochaines Étapes

1. **Valider le cahier des charges**
2. **Prioriser les fonctionnalités et les dispatcher**
3. **Planifier les sprints** (ex : 2 semaines par sprint).