Nom et prénom : Mohamed MAHROUCH

Cahier de charges minimales :

Système de gestion d'abonnement pour les employés OCP

1. Contexte du projet

OCP souhaite mettre en place un système d'abonnement digitalisé pour ses employés, leur permettant d'accéder à différents services internes à l'aide de crédits alloués : **boissons**, **impressions**, **réservation de salles de réunion**, **accès à des jeux, etc**.

2. Objectifs

- ❒ Offrir une interface intuitive permettant à chaque membre d'utiliser son abonnement.
- ❒ Gérer différents types d'abonnement (Standard, Premium…).
- ❒ Suivre et contrôler les crédits consommés.
- ❒ Faciliter la réservation de services internes.
- ❒ Permettre à l'administration de gérer les services et les droits d'accès.

3. Fonctionnalités principales

a) Côté utilisateur (employé)

- Authentification / inscription
- Visualisation de l'abonnement actif et des crédits restants
- Liste des services disponibles : Boissons, Impressions, Salles de réunion, Jeux
- Utilisation d'un service (avec déduction automatique de crédits)
- Historique d'utilisation
- Réservation (avec planning)

b) Côté administrateur

- Gestion des utilisateurs et de leurs abonnements
- Définition des crédits par abonnement
- Gestion des services disponibles
- Planning global des réservations
- Statistiques d'usage par service / utilisateur

4. Architecture technique

Technologies utilisées :

- ✓ Frontend : Angular
- ✓ Backend : Spring Boot (REST API)
- ✓ Base de données : MongoDB
- ✓ Authentification : JWT + Spring Security

5. Modèle de données

Ce modèle est conçu pour être à la fois flexible et robuste, en utilisant des références entre les collections pour assurer la cohérence des données.

Collection 1 : `users` :

Objectif : Stocker les informations sur les employés (utilisateurs) et les administrateurs.

Nom du Champ	Type	Description
`_id`	ObjectId	Identifiant unique généré par MongoDB.
`firstName`	String	Prénom de l'employé.
`lastName`	String	Nom de famille de l'employé.
`email`	String	Adresse e-mail professionnelle (doit être **unique**). Sert de login.
`password`	String	Mot de passe de l'utilisateur (**hashé** pour la sécurité).
`role`	String	Définit les permissions. Valeurs possibles : `'EMPLOYEE'`, `'ADMIN'`.
`subscriptionId`	ObjectId	Référence à l'identifiant `_id` du document dans la collection `subscriptions`.
`credits`	Number	Solde de crédits actuel de l'utilisateur.
`createdAt`	Date	Date de création du compte utilisateur.

Collection 2 : `subscriptions`

Objectif : Définir les différents types d'abonnements disponibles (Standard, Premium, etc.).

Nom du Champ	Type	Description
`_id`	ObjectId	Identifiant unique généré par MongoDB.
`name`	String	Nom de l'abonnement (ex: "Standard", "Premium"). Doit être **unique**.
`initialCredits`	Number	Nombre de crédits alloués par défaut pour cet abonnement (par mois/période).
`description`	String	Description des avantages de l'abonnement.

Collection 3 : `service_categories`

Objectif : Organiser et regrouper les services de manière logique.

Nom du Champ	Type	Description
`_id`	ObjectId	Identifiant unique généré par MongoDB.
`name`	String	Nom de la catégorie (ex: "Boissons", "Impressions"). Doit être **unique**.
`description`	String	Description optionnelle de la catégorie.
`icon`	String	Nom d'une icône pour l'affichage (ex: "fa-coffee"). Optionnel.

Collection 4 : `services`

Objectif : Lister chaque service individuel que les employés peuvent consommer.

Nom du Champ	Type	Description
`_id`	ObjectId	Identifiant unique généré par MongoDB.
`name`	String	Nom du service (ex: "Café Expresso", "Impression A4 N&B").
`description`	String	Description détaillée du service.
`costInCredits`	Number	Le coût en crédits pour utiliser ce service.
`categoryId`	ObjectId	Référence à l'identifiant `_id` du document dans la collection `service_categories`.
`requiresBooking`	Boolean	`true` si le service nécessite une réservation (ex: salle), `false` sinon.

Collection 5 : `usage_history`

Objectif : Enregistrer chaque transaction de consommation de crédit (pour les services sans réservation).

Nom du Champ	Type	Description
`_id`	ObjectId	Identifiant unique généré par MongoDB.
`userId`	ObjectId	Référence à l'utilisateur (`users`) qui a consommé le service.
`serviceId`	ObjectId	Référence au service (`services`) qui a été consommé.
`creditsDebited`	Number	Le nombre exact de crédits débités lors de cette transaction.
`usageDate`	Date	Date et heure exactes de la consommation.

Collection 6: `bookings`

Objectif : Gérer l'historique et le planning des services qui nécessitent une réservation.

Nom du Champ	Type	Description
`_id`	ObjectId	Identifiant unique généré par MongoDB.
`userId`	ObjectId	Référence à l'utilisateur (`users`) qui a fait la réservation.
`serviceId`	ObjectId	Référence au service (`services`) qui est réservé.
`startTime`	Date	Date et heure de début de la réservation.
`endTime`	Date	Date et heure de fin de la réservation.
`status`	String	Statut de la réservation. Valeurs : `'CONFIRMED'`, `'CANCELLED'`.

6. Pages du frontend (Angular) :

- Page de login / inscription
- Dashboard utilisateur
- Liste des services
- Page de réservation (avec calendrier)
- Historique d'utilisation
- Interface admin (gestion utilisateurs, services, statistiques)

7. Contraintes :

- Le système doit être sécurisé (authentification, rôles).
- Interface doit être responsive.
- Les données doivent être persistées dans MongoDB.
- Prévoir une architecture RESTful claire.

8. Evolutions futures : Intelligence Artificielle , Mobile et Technologies Avancées :

- Génération de **QR code** pour l'accès rapide à certains services.
- **Notifications** (mail ou système) en cas de réservation ou consommation.
- Intégration avec **badge RFID** (dans une version hardware).
- **Réservation vocale** : permettre aux membres de réserver un service (salle , besoin , etc)
- **Chatboot intelligent** : offrir une assistance interactive pour répondre aux questions et guider l'utilisateur.
- **Analyse intelligente des besoins** : analyser l'historique des membres afin de recommander des services plus demandées.
- **Application mobile** : proposer une version mobile du système (Android/iOS) pour permettre aux membres d'accéder aux services à tout moment , avec les memes fonctionnalités que sur la version web