Rapport de Projet : Application Cabinet Médical (Oussama Trad, Hamed Slim)

1. Documentation Utilisateur

Instructions pour les Patients

1. **Inscription et Connexion** :
    - Accédez à la page d'inscription via le menu latéral (S'inscrire) ou la page de connexion (Se connecter).
    - Pour l'inscription, fournissez votre email, mot de passe, prénom, nom, numéro de téléphone, date de naissance, adresse et genre.
    - Vous pouvez également vous connecter via Google ou Facebook en sélectionnant les options correspondantes.
    - Une fois connecté, vous serez redirigé vers l'accueil patient (/tabs/accueil).
2. **Prendre un Rendez-vous** :
    - Depuis l'accueil, naviguez vers Rendez-vous dans le menu ou les onglets.
    - Sélectionnez un médecin dans la liste des médecins disponibles (Tous les médecins).
    - Choisissez une date et un créneau horaire disponible (8h-18h, par tranches de 30 minutes, hors week-end).
    - Indiquez le motif de la consultation (facultatif) et joignez un document si nécessaire.
    - Validez pour envoyer une demande de rendez-vous au médecin.
3. **Messagerie** :
    - Accédez à la messagerie via le menu (Messagerie).
    - Consultez vos conversations existantes ou démarrez une nouvelle conversation avec un médecin ou un patient.
4. **Notifications** :
    - Les notifications apparaissent dans un menu déroulant accessible via l'icône de cloche.
5. **Déconnexion** :
    - Cliquez sur Se déconnecter dans le menu pour quitter la session.

Instructions pour les Médecins

1. **Connexion** :

- Les médecins se connectent avec leur email et mot de passe via la page de connexion.
- Une fois connecté, vous êtes redirigé vers l'accueil médecin (/tabs-medecin/accueil-medecin).

2. **Gestion des Rendez-vous** :
   - Dans Accueil Médecin, consultez les demandes de rendez-vous (rendezVousDemandes).
   - Acceptez ou refusez les demandes via les boutons correspondants.
3. **Messagerie** :
   - Utilisez la messagerie pour communiquer avec les patients.
4. **Disponibilités** :
   - Dans Mes disponibilités, définissez vos horaires de travail par jour (8h-18h, tranches de 30 minutes).
   - Les créneaux occupés ou hors disponibilité ne sont pas proposés aux patients.
5. **Notifications** :
   - Recevez des notifications
6. **Déconnexion** :
   - Utilisez l'option Se déconnecter dans le menu.

2. Documentation Technique

Choix Techniques

1. **Backend** :
   - **Framework** : Flask (Python) pour sa légèreté et sa facilité de configuration.
   - **Base de Données** :
     - **MongoDB** pour les données des utilisateurs, médecins, messages
   - **Authentification** :
     - JWT (JSON Web Tokens) pour sécuriser les routes protégées.
     - Bcrypt pour le hachage des mots de passe.
   - **CORS** : Configuré pour permettre les requêtes cross-origin depuis les ports Ionic locaux (8100, 8101, 8102, 8104).
2. **Frontend** :

- **Framework** : Angular avec Ionic pour une application mobile hybride performante.
- **Structure** : Architecture modulaire avec des composants réutilisables (ex. TabsComponent, HeaderComponent).
- **Routage** : Angular Router avec des guards (AuthGuard, PatientGuard, MedecinGuard) pour sécuriser l'accès aux pages.
- **Styles** : SCSS pour des styles modulaires et Ionic pour une UI native-like.

3. **Communication Client-Serveur** :
   - API RESTful avec des endpoints comme /api/register, /api/rendezvous, /api/chat/messages.
   - Utilisation de HttpClient Angular pour les requêtes HTTP avec gestion des erreurs via catchError.
4. **Initialisation des Données** :
   - Service DbInitService pour peupler la base de données avec des données de test depuis un fichier JSON (db-init.json).

Guide d'Installation et d'Exécution

Prérequis

- **Node.js** (v16 ou supérieur)
- **Python** (v3.8 ou supérieur)
- **MongoDB** (local ou Atlas)
- **Ionic CLI** (npm install -g @ionic/cli)

Installation

1. **Backend** :
2. # Cloner le dépôt
3. git clone <url-du-dépôt>
4. cd backend
5. # Installer les dépendances

   Frontend :
6. cd frontend
7. npm install

Exécution

1. **Démarrer MongoDB** :
2. mongod
3. **Lancer le Backend** :
4. cd flask-auth
5. python app.py

Le serveur Flask sera disponible sur http://localhost:5000.

6. **Lancer le Frontend** :
7. cd frontend
8. ionic serve

L'application sera accessible.

9. **Initialisation de la Base de Données** :
    - L'application appelle automatiquement DbInitService au démarrage pour charger les données de test depuis assets/db-init.json.

3. Rapport de Projet : Défis et Solutions

Défi 1 : Navigation entre les Pages

Problème :

- La navigation entre les pages était complexe en raison de la structure à deux rôles (patient et médecin) avec des interfaces distinctes (tabs pour patients, tabs-medecin pour médecins).
- Les guards d'authentification (AuthGuard, PatientGuard, MedecinGuard) causaient des redirections incorrectes si le rôle ou le token n'était pas correctement défini.
- Le menu latéral devait s'adapter dynamiquement au rôle de l'utilisateur.

Solutions :

- **Routage Modulaire** : Utilisation d'un AppRoutingModule avec des routes imbriquées pour les tabs et tabs-medecin. Les guards vérifient le token JWT et le rôle stocké dans localStorage.
- **Menu Dynamique** : Le composant AppComponent utilise des *ngIf pour afficher les options de menu en fonction de isLoggedIn et role. Les méthodes comme goToAccueil() et goToAccueilMedecin() redirigent vers les routes appropriées.

- **Gestion des Redirections** : Ajout de redirections par défaut (/login pour les utilisateurs non connectés) et gestion des erreurs 403/401 via des intercepteurs HTTP dans Angular.
- **Tests** : Tests unitaires (AppComponent.spec.ts) pour vérifier la création du composant et la logique de navigation.

Défi 2 : Création d'une Base de Données Convenable

Problème :

- Les relations entre collections (ex. messages et users et medecins) devaient être cohérentes pour éviter les incohérences.

Solutions :

- **Schéma Hybride** :
    - MongoDB : Collections users, Medecins, et messages pour stocker les profils, rendez-vous, et conversations. Les documents JSON permettent une évolution facile du schéma.
- **Initialisation Automatisée** :
    - Service DbInitService charge un fichier JSON (db-init.json) contenant des médecins, patients, et rendez-vous prédéfinis.
    - Simulation de connexion pour créer des rendez-vous en tant qu'utilisateur, puis confirmation en tant que médecin via des requêtes API séquentielles.
- **Synchronisation des Données** :
    - Lors de la création d'un rendez-vous, mise à jour simultanée des collections users.rendezVousFuturs et medecins.rendezVousDemandes avec des données cohérentes.
    - Utilisation de $push et $pull dans MongoDB pour gérer les tableaux de rendez-vous.
- **Validation** :
    - Contrôles stricts dans les endpoints (ex. vérification des créneaux horaires, conflits de rendez-vous, format de date/heure).
    - Gestion des erreurs avec des messages clairs (ex. "Ce créneau est déjà réservé").