Documentation Technique pour le Projet Soigne Moi

1. Réflexions Initiales Technologiques

Applications Développées :

- **Application Web (PHP)** : Utilisée pour la gestion des séjours par les patients, et la gestion des par les admin. Cette application permet l'accès aux données via une interface web et fournit des fonctionnalités d'administration.
- **Application Mobile (Flutter)** : Facilite la consultation des informations sur les patients et la gestion des avis et prescriptions par les médecins via un appareil mobile.
- **Application Bureautique (Python avec Tkinter)** : Application desktop pour une gestion locale des entrées, sorties et les dossiers des patients.

Choix Technologiques :

- **PHP** : Choisi pour sa robustesse dans le développement web et sa facilité d'intégration avec des bases de données MySQL.
- **Flutter** : Sélectionné pour ses capacités de création d'applications mobiles multiplateformes avec une interface utilisateur fluide et performante.
- **Python avec Tkinter** : Utilisé pour son accessibilité et sa simplicité dans le développement d'applications desktop.

2. Configuration de l'Environnement de Travail

1. PHP Web Application :

- **Serveur Web** : XAMPP (inclus Apache, MySQL, et PHP).
- **Répertoire de Développement** : `C:\xampp\htdocs\studi_ecf\soignemoi_spn`
- **Configuration MySQL** : Base de données hébergée localement avec les accès configurés via `phpMyAdmin`.

2. Flutter Mobile Application :

- **Environnement de Développement** : Flutter SDK installé sur Windows.
- **Répertoire de Développement** : `C:\Users\Svetlana Nigay\flutter soigne_moi\flutter_app_s\flutter_app_s`

3. Python Desktop Application :

- **Environnement de Développement** : Python installé avec Tkinter.
- **Exécutable Python** : `C:\Users\Svetlana Nigay\AppData\Local\Microsoft\WindowsApps\python.exe`

3. Explication du Plan de Test

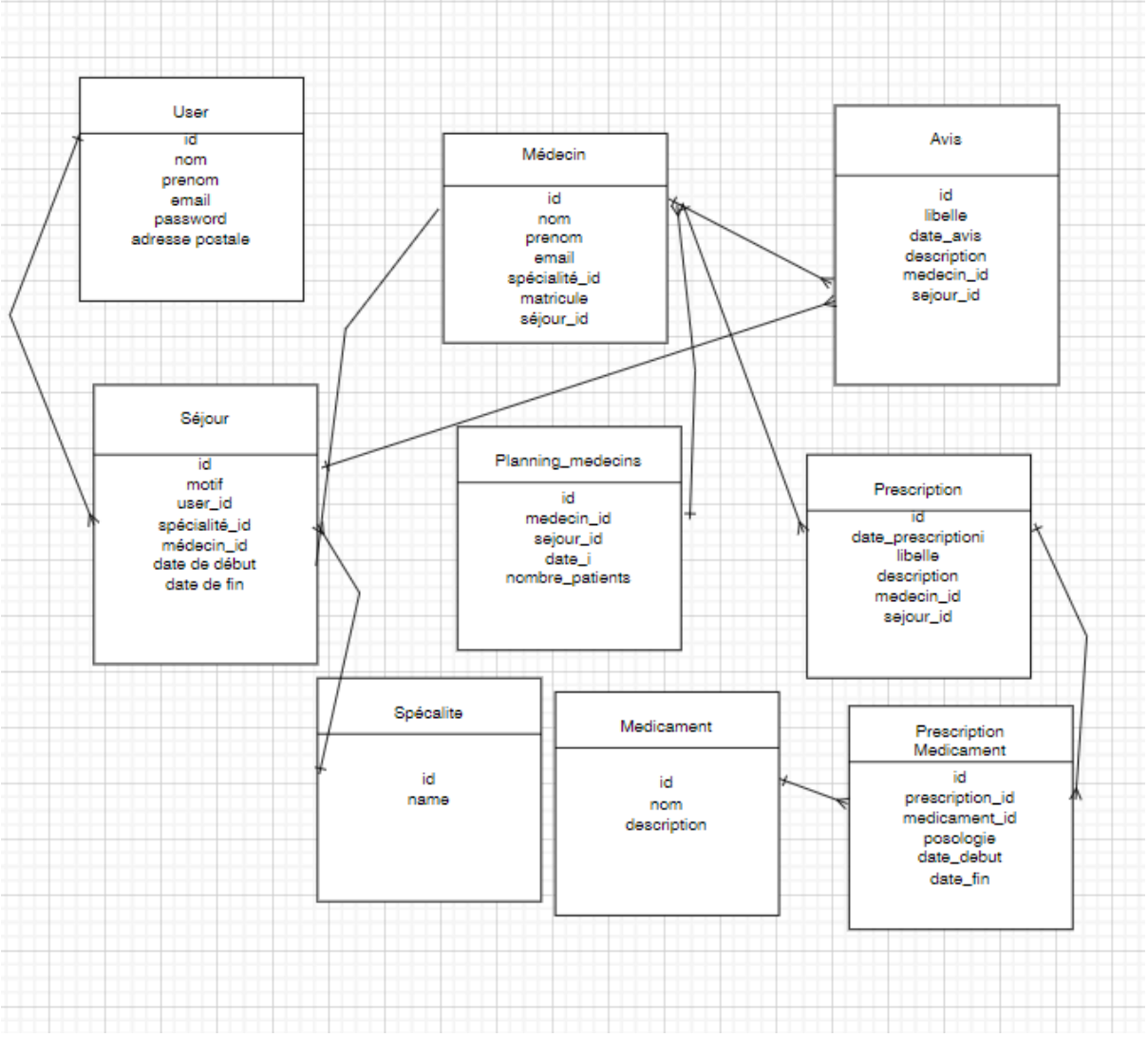
Plan de Test :

- **Tests Manuels** : Après chaque ajout de fonctionnalité, des tests manuels ont été réalisés pour vérifier le bon fonctionnement de chaque composant. Les tests comprenaient des vérifications de l'interface utilisateur, des interactions avec les bases de données, et des scénarios d'utilisation pour s'assurer que les nouvelles fonctionnalités répondent aux exigences spécifiées.

Méthodologie :

1. **Développement** : Ajout de nouvelles fonctionnalités ou corrections de bugs.
2. **Test** : Vérification manuelle de chaque fonctionnalité ajoutée pour confirmer son bon fonctionnement.
3. **Validation** : Passage à la phase suivante du développement uniquement après validation des tests manuels.

Base de Données : MCD



Entités et leurs Attributs

1. **`avis`**
   - **Attributs :** `id`, `libelle`, `date_avis`, `description`, `medecin_id`, `sejour_id`
   - **Description :** Représente un avis donné par un médecin pour un séjour particulier.
2. **`medecin`**
   - **Attributs :** `id`, `nom`, `prenom`, `email`, `matricule`, `specialite_id`
   - **Description :** Représente un médecin, avec ses informations personnelles et son ID de spécialité.
3. **`medicaments`**
   - **Attributs :** `id`, `nom`, `description`
   - **Description :** Représente les médicaments disponibles dans le système.
4. **`planning_medecins`**
   - **Attributs :** `id`, `medecin_id`, `sejour_id`, `date_i`, `nombre_patients`
   - **Description :** Représente la planification des médecins pour un séjour à une date donnée.
5. **`prescriptions`**
   - **Attributs :** `id`, `date_prescription`, `libelle`, `description`, `medecin_id`, `sejour_id`
   - **Description :** Représente une prescription médicale faite pour un séjour par un médecin.
6. **`prescriptions_medicaments`**
   - **Attributs :** `id`, `prescription_id`, `medicament_id`, `posologie`, `date_debut`, `date_fin`
   - **Description :** Représente les médicaments prescrits dans le cadre d'une prescription spécifique, avec les détails de la posologie et les dates de début et de fin.
7. **`sejours`**
   - **Attributs :** `id`, `date_debut`, `date_fin`, `motif`, `specialite_id`, `user_id`, `medecin_id`
   - **Description :** Représente un séjour hospitalier ou une consultation, avec les dates, le motif, la spécialité, le patient, et le médecin associé.
8. **`specialites`**
   - **Attributs :** `id`, `name`
   - **Description :** Représente les spécialités médicales disponibles.
9. **`user`**
   - **Attributs :** `id`, `prenom`, `nom`, `email`, `password`, `adress`, `role`
   - **Description :** Représente un utilisateur du système, qui peut être un patient ou un personnel médical.

Relations entre Entités

1. **`avis`**
   - **Relation avec `medecin` :** Chaque avis est associé à un médecin (clé étrangère `medecin_id`).
   - **Relation avec `sejours` :** Chaque avis est associé à un séjour (clé étrangère `sejour_id`).
2. **`medecin`**
   - **Relation avec `specialites` :** Chaque médecin est associé à une spécialité (clé étrangère `specialite_id`).
3. **`planning_medecins`**
   - **Relation avec `medecin` :** Chaque planification concerne un médecin (clé étrangère `medecin_id`).
   - **Relation avec `sejours` :** Chaque planification concerne un séjour (clé étrangère `sejour_id`).
4. **`prescriptions`**
   - **Relation avec `medecin` :** Chaque prescription est faite par un médecin (clé étrangère `medecin_id`).
   - **Relation avec `sejours` :** Chaque prescription est associée à un séjour (clé étrangère `sejour_id`).
5. **`prescriptions_medicaments`**
   - **Relation avec `prescriptions` :** Chaque enregistrement de médicament est associé à une prescription (clé étrangère `prescription_id`).
   - **Relation avec `medicaments` :** Chaque enregistrement de médicament se réfère à un médicament spécifique (clé étrangère `medicament_id`).
6. **`sejours`**
   - **Relation avec `user` :** Chaque séjour est associé à un patient (clé étrangère `user_id`).
   - **Relation avec `medecin` :** Chaque séjour est associé à un médecin (clé étrangère `medecin_id`).
   - **Relation avec `specialites` :** Chaque séjour est associé à une spécialité médicale (clé étrangère `specialite_id`).

4. Transaction SQL

But de la Requête :

La transaction SQL permet d'assurer que l'insertion d'un nouveau médecin dans la table `medecins` est accompagnée de la création d'un utilisateur associé dans la table `users`. Cela garantit la cohérence des données et l'intégrité des informations dans la base de données.

- alide la transaction, confirmant les modifications apportées aux tables.

Cette méthode garantit que toutes les modifications sont effectuées de manière atomique, c'est-à-dire que soit toutes les modifications sont appliquées, soit aucune ne l'est en cas d'erreur.