## **Cahier des Charges: Système de Gestion de Produits et de Vente**

**1. Introduction**

Ce document présente le cahier des charges d'un système de gestion de produits et de vente utilisant l’appareil Famoco PX400. Le système permettra de scanner les produits, de ajuster et générer des reçus, de stocker les données dans une base de données et de les afficher sur une application web.

**2. Fonctionnalités**

**2.1. App Mobile (React Native):**

**Scanner les codes-barres et les QR codes des produits:**

* Intégration d'une bibliothèque de scan performante et fiable.
* Gestion des erreurs de scan et affichage de messages clairs.

**Afficher les informations des produits:**

* Nom, description, prix, quantité en stock, image, etc.
* Possibilité de filtrer et rechercher des produits.
* Affichage des informations de manière claire et concise.

**Modifier les informations des produits avant de générer un reçu:**

* Modification d’information tel que la quantité.
* Ajout de notes ou de commentaires.
* Validation des modifications avant de générer le reçu.

**Générer et imprimer des reçus sur l'imprimante thermique du PX400:**

* Possibilité de choisir les informations à afficher sur le reçu.
* Impression automatique du reçu après confirmation.

**Envoyer les données des reçus au serveur web via une API:**

* Formatage des données en JSON pour une transmission efficace.
* Sécurisation de la communication avec HTTPS.
* Gestion des erreurs de transmission et retransmission automatique.

**2.2. Application Web (Django):**

**Afficher la liste des produits avec leurs informations et QR codes:**

* Tri et filtrage des produits par nom, catégorie, etc.
* Affichage des images des produits.
* Génération de QR codes uniques pour chaque produit.

**Ajouter de nouveaux produits avec des informations et générer des QR codes:**

* Formulaire de saisie des informations du produit (nom, description, prix, quantité, etc.).
* Génération automatique du QR code après la création du produit.
* Validation des données saisies et gestion des erreurs.

**Modifier et supprimer des produits:**

* Formulaire de modification des informations du produit.
* Confirmation avant la suppression d'un produit.

**Afficher la liste des reçus générés:**

* Tri et filtrage des reçus par date, client, etc.
* Affichage des détails de chaque reçu (produits, prix, total, etc.).
* Possibilité de télécharger et d'imprimer les reçus.

**2.3. Serveur Web et API (Django):**

**Recevoir les données JSON de l’appli mobile:**

* Décodage et validation des données reçues.
* Stockage des données dans la base de données SQLite.
* Envoi de messages de confirmation ou d'erreur a l’application mobile.

**Stocker les données des produits et des reçus dans une base de données SQLite:**

* Création de tables et de relations pour une gestion efficace des données.
* Optimisation des requêtes pour une performance optimale.
* Sauvegarde et restauration régulières de la base de données.

**Gérer les utilisateurs et les autorisations d'accès:**

* Création de comptes utilisateurs avec différents niveaux d'accès.
* Authentification et autorisation sécurisées via des tokens.

**Fournir une API RESTful pour:**

* Afficher la liste des produits avec leurs informations et images.
* Ajouter, modifier et supprimer des produits.
* Afficher la liste des reçus avec leurs détails.
* Filtrer et rechercher des produits et des reçus.

**3. Architecture**

**3.1. Application Mobile :**

* React Native
* Bibliothèques de scan de codes-barres (ZXing, Scandit)
* Modules de communication réseau (Axios, Fetch API)

**3.2. Application Web:**

* Django
* Framework d'interface utilisateur (Bootstrap)
* Chargement dynamique des données pour une meilleure performance
* Exportation des données vers des formats CSV et Excel

**3.3. Serveur Web et API:**

* Django
* Framework de base de données SQLite
* Bibliothèques de sécurité et d'authentification (Django REST Framework, Django User model)
* Modules de génération de QR codes (qrcode-python)

**4. Technologies**

* **Mobile App:** React Native, JavaScript
* **Serveur Web et API:** Python, Django, SQLite, RESTful API
* **Application Web:** Django, HTML, CSS, JavaScript
* **Base de données:** SQLite
* **Communication:** JSON
* **Authentification:** JWT (JSON Web Tokens)
* **Sécurité:** HTTPS, gestion des erreurs et des exceptions

**5. Non-Fonctionnel**

**Sécurité:**

* Authentification et autorisation robustes
* Cryptage des données sensibles(mots de passes)

**Convivialité:**

* Interfaces utilisateur intuitives et réactives
* Documentation claire et accessible
* Formation ou support aux utilisateurs

**Évolutivité:**

* Architecture capable de gérer une croissance du nombre de produits et de transactions
* Possibilité d'ajouter de nouvelles fonctionnalités facilement

**6. Conclusion**

Ce cahier des charges fournit un cadre global pour le développement et la mise en œuvre du système de gestion de produits et de vente.