**1. Fonctionnalités principales :**

* **Gestion des clients :** Ajout, modification, suppression de clients.
* **Gestion des factures :** Création, affichage des factures, statut (payée, non payée), et total des factures par client.
* **Gestion des paiements :** Suivi des paiements effectués, et lien avec les factures.
* **Gestion des dépenses :** Enregistrement des dépenses professionnelles, lien avec les factures ou les paiements.
* **Paramètres de l'entreprise :** Informations générales sur l'entreprise (nom, adresse, TVA, etc.), et génération de rapports ou statistiques.

**2. Structure des pages et des vues :**

**Page d'accueil (Dashboard) :**

* **But :** Afficher un résumé rapide de l'état de l'entreprise.
* **Contenu à afficher :**
  + **Total des clients** : Afficher le nombre total de clients enregistrés.
  + **Total des factures** : Nombre total de factures émises, avec un indicateur du montant total.
  + **Total des dépenses** : Montant total des dépenses enregistrées.
  + **Total des paiements reçus** : Montant total des paiements effectués.

Ce dashboard doit être mis à jour en temps réel pour donner une vue d'ensemble instantanée des finances et de la gestion.

**Page des clients :**

* **But :** Gérer les clients.
* **Contenu à afficher :**
  + Liste des clients avec les informations de base : nom, prénom, email, téléphone.
  + Fonctionnalité de recherche pour filtrer les clients.
  + Options pour ajouter, modifier ou supprimer un client.
  + Lien vers les factures associées à chaque client.

**Page des factures :**

* **But :** Gérer les factures.
* **Contenu à afficher :**
  + Liste des factures avec les informations importantes : montant, statut, date d'émission, client lié.
  + Possibilité de créer une nouvelle facture, de l’associer à un client, et de marquer une facture comme payée.
  + Option pour visualiser ou télécharger les factures sous forme PDF (si nécessaire).

**Page des paiements :**

* **But :** Suivre les paiements effectués.
* **Contenu à afficher :**
  + Liste des paiements avec le montant payé, la date de paiement, et le client associé.
  + Option pour lier un paiement à une facture spécifique.
  + Calcul du solde restant à recevoir pour chaque facture.

**Page des dépenses :**

* **But :** Suivre les dépenses professionnelles.
* **Contenu à afficher :**
  + Liste des dépenses avec montant, date, catégorie (par exemple, matériel, marketing, fournitures, etc.).
  + Option pour ajouter, modifier, ou supprimer une dépense.

**Page des paramètres de l'entreprise :**

* **But :** Mettre à jour les informations générales de l'entreprise.
* **Contenu à afficher :**
  + Champs pour modifier les paramètres de l'entreprise : nom, adresse, type de TVA, logo.
  + Statistiques et rapports sur les finances de l'entreprise (par exemple, total des ventes sur le mois, total des dépenses, solde net).

**3. Interaction avec l'API :**

* L'application **front-end** en Vue.js consommera les données de l'API Django via des requêtes HTTP (ex : avec Axios). Par exemple :
  + Récupérer les données des clients à afficher sur la page des clients.
  + Envoyer de nouvelles factures ou paiements via des requêtes POST.
  + Calculer les totaux sur le dashboard en récupérant les informations pertinentes des API.

**4. Exemple de flux d'une fonctionnalité :**

Prenons l'exemple d'une nouvelle **facture** :

1. **Création de facture** : L'utilisateur entre les informations de la facture via un formulaire Vue.js.
2. **Enregistrement en base** : Lorsque la facture est enregistrée, une requête POST est envoyée à l'API Django pour enregistrer la facture dans la base de données.
3. **Mise à jour du dashboard** : Après l'enregistrement, le dashboard récupère de nouveau les totaux via une requête GET à l'API pour afficher les dernières données, comme le total des factures.

**Répartition des responsabilités :**

* **Backend (Django)** : Fournir des API RESTful pour récupérer et envoyer les données.
  + Gérer la logique métier (par exemple, calcul des totaux de paiements, factures, etc.).
  + Authentification et sécurité (par exemple, JWT ou sessions pour la gestion des utilisateurs).
* **Frontend (Vue.js)** : Afficher les données et interagir avec l'utilisateur.
  + Afficher les données dynamiquement via des composants Vue.js.
  + Gérer les interactions avec l'utilisateur (ajouter des clients, payer des factures, etc.).

**5. Conclusion :**

Ton application devrait donc centraliser la gestion des clients, factures, paiements, dépenses, et paramètres de l'entreprise, tout en offrant une vue d'ensemble claire sur un **dashboard**. Chaque page (clients, factures, paiements, etc.) sera accessible via des routes distinctes, et l'interface sera mise à jour en temps réel via des appels API à Django.