Cahier des Charges : Caisse de Retraite par Répartition avec Épargne

1. Introduction

1.1 Contexte

Ce projet vise à développer une application web pour la gestion d'une Caisse de Retraite par Répartition avec Épargne. L'application doit permettre la gestion des demandes de retraite, le suivi de l'épargne des adhérents, la génération de rapports financiers et la gestion administrative des opérations.

1.2 Objectifs

- Fournir une interface utilisateur intuitive pour les adhérents et les administrateurs.
- Assurer la sécurité des données et la conformité réglementaire.
- Permettre une gestion efficace des fonds de retraite.

2. Description des Modules

2.1 Module Retraites

Fonctionnalité

- Gestion des demandes de retraite des adhérents.

Modèle

- **DemandeRetraite**
  - `profil`: Référence à l'utilisateur (adhérent).
  - `montant_retraite`: Montant demandé.
  - `date_demande`: Date de la demande.
  - `statut`: Statut de la demande (en attente, approuvé, rejeté).

Critères de succès

- L'utilisateur peut soumettre une demande de retraite.
- L'administrateur peut approuver ou rejeter les demandes via une interface dédiée.

2.2 Module Épargne

Fonctionnalité

- Suivi de l'épargne des adhérents.

Modèle

- **Epargne**
  - `profil`: Référence à l'utilisateur (adhérent).
  - `montant`: Montant épargné.
  - `date`: Date de l'épargne.
  - `interets`: Intérêts accumulés.

Critères de succès

- Les utilisateurs peuvent consulter leur solde d'épargne et les intérêts accumulés.
- L'administrateur peut générer des rapports d'épargne.

2.3 Module Rapports Financiers

Fonctionnalité

- Génération de rapports financiers pour les adhérents et la direction.

Modèle

- **RapportFinancier**
  - `date`: Date du rapport.
  - `total_cotisations`: Total des cotisations.
  - `total_retraites`: Total des retraites.
  - `total_epargnes`: Total des épargnes.

Critères de succès

- Les rapports doivent être générés mensuellement et disponibles au téléchargement.
- Les utilisateurs doivent avoir accès à des résumés financiers.

2.4 Module Administration

Fonctionnalité

- Gestion administrative des opérations.

Modèle

- **Audit**
  - `action`: Description de l'action réalisée.
  - `date_action`: Date et heure de l'action.
  - `utilisateur`: Référence à l'utilisateur ayant effectué l'action.

Critères de succès

- Les administrateurs peuvent consulter l'historique des actions.
- Les utilisateurs peuvent être ajoutés ou supprimés par les administrateurs.

2.5 Système de Retraite par Répartition

Fonctionnalité

- Gestion des fonds de retraite par répartition.

Modèle

- **FondsRetraite**
  - `montant_total`: Montant total des fonds.
  - `historique_paiements`: Historique des paiements de retraite.

Critères de succès

- Les paiements de retraite doivent être enregistrés et affichés dans l'historique.
- La gestion des fonds doit être accessible aux administrateurs.

---

3. Pratiques de Développement

3.1 Environnement de Développement

Configuration

- **Python** : Version 3.9 ou supérieure.
- **Django** : Version 4.0 ou supérieure.
- **PostgreSQL** : S'assurer que PostgreSQL est installé et configuré.

Gestion des Dépendances

- Utiliser un fichier `requirements.txt` pour gérer les dépendances du projet.

`pip freeze > requirements.txt` # En étant dans l'environnement virtuel après l'installation de tout les packages.

3.2 Contrôle de Version

- Utiliser Git pour le contrôle de version.
- Créer un dépôt Git sur GitHub ou GitLab pour héberger le code.

3.3 Tests

- Écrire des tests unitaires et d'intégration pour chaque module.
- Utiliser `pytest` ou `unittest` pour exécuter les tests.

3.4. Documentation

- **Documentation Technique** : Maintenir une documentation claire et à jour pour chaque module et les flux de travail.
- **Wiki ou README** : Utiliser un fichier README ou un wiki pour fournir des instructions sur la configuration de l'environnement de développement et le déploiement.