Prototype Django transactionnel

Objectif

Ce travail vise à vous préparer en vue du projet de session, en vous permettant d'explorer par vous-mêmes les mécanismes fondamentaux de Django. L'application que vous réaliserez doit démontrer votre capacité à :

- Structurer un projet Django selon le modèle **MVT (Model–View–Template)**.
- Manipuler une base de données relationnelle via l'ORM intégré.
- Mettre en place des routes, des formulaires, et une interface utilisateur fonctionnelle.

Consignes générales

- Travail individuel.
- Sujet libre (ex. : plateforme de réservation, bibliothèque, gestion de tournois, recettes, sondages, suivi de tâches, mini-blog, etc.).
- Vous devez concevoir une application **transactionnelle** utilisant une base SQLite, avec **au moins 3 ou 4 modèles** reliés entre eux.

Exigences obligatoires

1. Structure du projet

- Respecter l'architecture **MVT**.
- Utiliser un ***layout* de base (base.html)** et étendre vos pages avec le mécanisme de *template inheritance*.
- Gérer les fichiers statiques (**CSS, JS, images**) via {% static '...' %}.
- Intégrer **Bootstrap** (ou équivalent) pour le style et le système de « grid ».

2. Routes et navigation

- Créer plusieurs routes nommées avec {% url '...' %} (avec et sans arguments).
- Gérer des pages statiques (ex. : page d'accueil ou « À propos »).
- Gérer des pages dynamiques (contenu tiré de la BD).

3. Base de données et modèles

- Minimum **3 modèles reliés** entre eux (1-N ou N-N).
- Appliquer les **migrations** correctement.
- Prévoir des données de départ (fixtures ou *createsuperuser* + quelques entrées).

4. Formulaires et transactions

- Formulaire **simple** (un modèle seul).
- Formulaire **complexe** (avec relation : menu déroulant, cases à cocher, etc.).
- Opérations **CRUD complètes** (ajouter, lister, modifier, supprimer).
- Valider correctement les champs (ex. champs obligatoires, formats, unicité).

5. Interface et convivialité

- Navigation claire (barre de menu, liens entre pages).
- Utilisation de messages de feedback (ex. messages de succès/erreur avec *django.contrib.messages*).
- Affichage conditionnel (ex. : afficher « Aucun résultat » si une liste est vide).

Exigences optionnelles (pour aller plus loin)

- Module **admin** :
  - Personnaliser l'affichage (*list_display, search_fields, list_filter*).
  - Gérer le téléversement d'images (config MEDIA + champ *ImageField*).

- Améliorations *front-end* :
  - Messages de confirmation (JavaScript / Bootstrap modal).
  - Pagination (via *Paginator*).

- Expérimenter avec un **ManyToManyField** ou une relation intermédiaire.

- Authentification :
  - Utiliser le **système d'utilisateurs de Django** (connexion, déconnexion, enregistrement).
  - Restreindre certaines actions aux utilisateurs connectés (ex. : création d'objets).

Vous devez remettre sur Léa :

- **Votre code source** (sans votre dossier « .venv »).
  - N'oubliez pas de générer votre fichier **requirements.txt**!

Travail terminé