université abdelmalek essaâdi

Faculté des Sciences et Techniques de Tanger

Développement web avancé Back end (Python) :

Rapport de Projet de Fin de Module : Application de Gestion de Tâches

Encadré par :

Othman bakkali

Réalisé Par : Oussama Allouch

I. Introduction :

Ce projet consiste en une application web de gestion de tâches (Todo List) développée avec Flask, suivant une architecture **MVC** (Modèle-Vue-Contrôleur) pour une structure claire et maintenable. L'application permet aux utilisateurs de **créer**, **modifier**, **supprimer** et organiser leurs tâches quotidiennes, avec un système d'authentification sécurisé via des **sessions Flask.**

L'objectif est de proposer une solution simple et sécurisée pour la prise de notes en ligne, en utilisant Flask pour assurer une gestion efficace des sessions et des bases de données.

II. Objectifs du projet :

- Implémenter un système d'authentification sécurisé (inscription, connexion, déconnexion) avec gestion de sessions.
- Afficher le nom de l'utilisateur connecté dans l'interface.
- Permettre aux utilisateurs de gérer leurs tâches :
  - Créer une nouvelle tâche avec titre et description.
  - Modifier le contenu et le statut d'une tâche existante.
  - Supprimer définitivement une tâche.
- Afficher la liste des tâches triées par date de création.
- Permettre le filtrage des tâches par statut (à faire, en cours, terminée).

III. Architecture du projet :

Ce projet suit une architecture **MVC** (Modèle-Vue-Contrôleur) pour une séparation claire des responsabilités. Voici la structure détaillée :

1. Modèles (Models):

Gèrent les données et les interactions avec la base de données **MySQL**.

a. Fichiers principaux :

- **user_model.py :**
  - get_user_by_username(username) → Récupère un utilisateur par son login.
  - create_user(username, email, password) → Ajoute un nouvel utilisateur.
- **task_model.py :**
  - create_task(user_id, title, description) → Ajoute une tâche.
  - get_user_tasks(user_id) → Liste les tâches d'un utilisateur.
  - update_task(task_id, new_data) → Modifie une tâche.
  - delete_task(task_id) → Supprime une tâche.

b. Base de données :

- Table **users** : id, username, email, password, created_at
- Table **tasks** : id, user_id , title, description, status, created_at

2. Vues :

Templates HTML/Jinja2 pour l'affichage.

a. Structure des templates :

- **base.html** → Layout principal (navbar, messages flash).
- **auth/**
  - login.html → Formulaire de connexion.
  - register.html → Formulaire d'inscription.
- **tasks/**
  - tasks.html → Liste des tâches (CRUD).

- add_task.html → Formulaire d'ajout.
- edit_task.html → Formulaire de modification.

3. Contrôleurs (Controllers):

Gèrent la **logique métier** et les routes Flask

a. Fichiers principaux :

- **auth_controller.py**
  - handle_login() → Vérifie les identifiants et crée une session.
  - handle_register() → Enregistre un nouvel utilisateur.
  - handle_logout() → Détruit la session.
- **task_controller.py**
  - show_tasks() → Affiche les tâches de l'utilisateur connecté.
  - add_task() → Ajoute une tâche à la base.
  - edit_task() → Modifie une tâche existante.
  - delete_task() → Supprime une tâche.

Routes (exemple) :

```
auth_bp = Blueprint('auth', __name__)
@auth_bp.route('/login', methods=['GET', 'POST'])
   def login():
      return auth_controller.handle_login()
```

IV. Fonctionnalités principales :

1. Inscription (Register):

- **Processus :**

1. L'utilisateur remplit le formulaire (username, email, password)
2. **AuthController.handle_register()** appelle **UserModel.create_user()**
3. Les infos sont stockées en base de données
4. Redirection vers la page de login

2. Connexion (Login):

- **Processus :**

1. L'utilisateur saisit ses credentials
2. **AuthController.handle_login() :**
   - Vérifie l'existence de l'utilisateur via UserModel.get_user_by_username()
   - Compare les mots de passe **en clair** (non sécurisé - critique à corriger)
3. Si valide :
   - Crée une **session Flask** avec session['username'] = …
   - Redirige vers la page d'accueil

3. Gestion des Session:

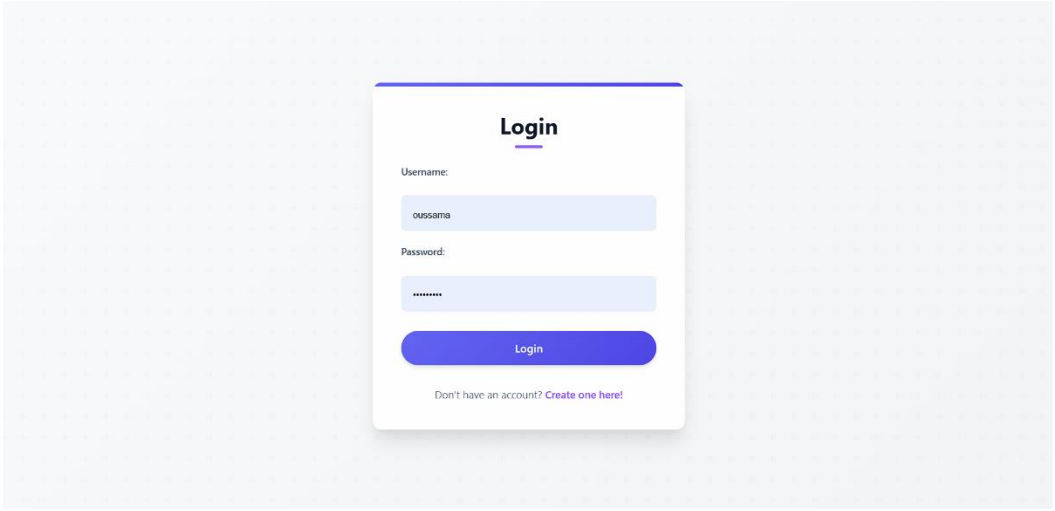
- **Mécanisme :**

- La session Flask est un **cookie sécurisé** côté client
- Toutes les pages protégées vérifient **if 'username' in session**
- La déconnexion (**handle_logout()**) supprime la session avec **session.pop()**

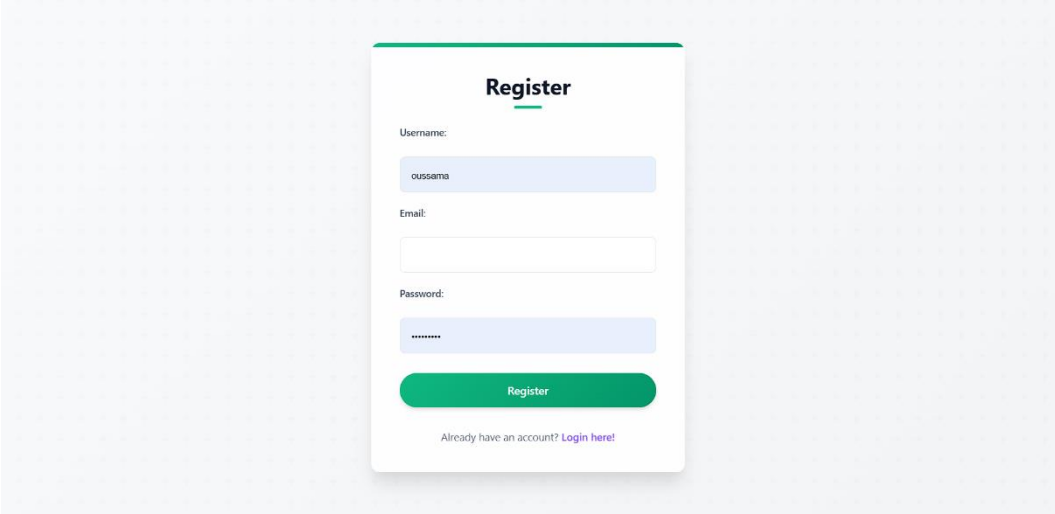
V. Captures d'Écran et Visualisation du Projet:

1. Homepage:

2. Login :

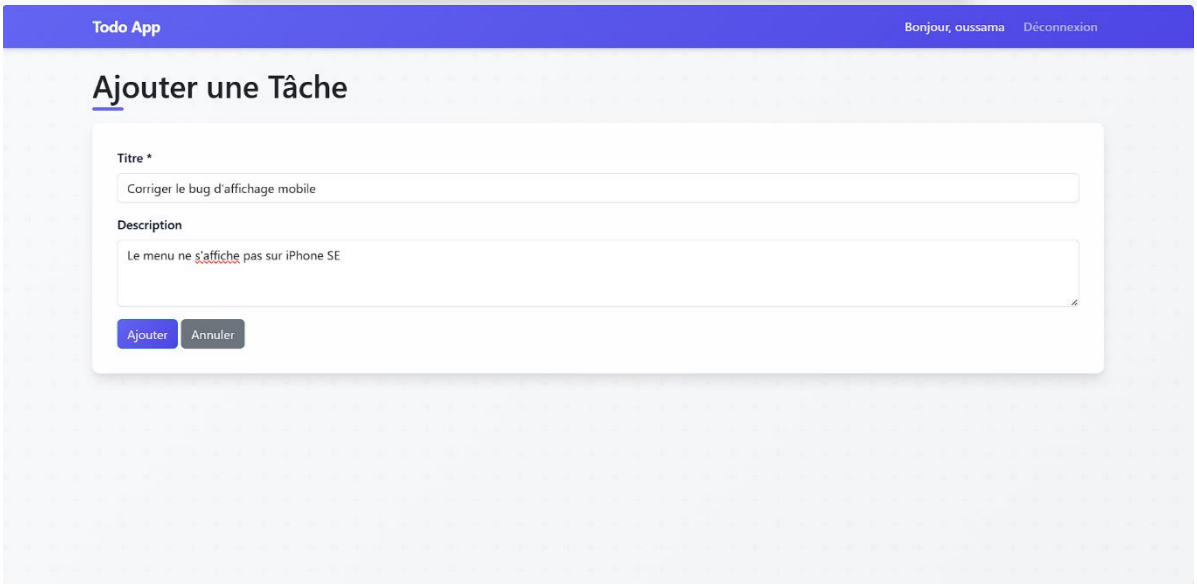


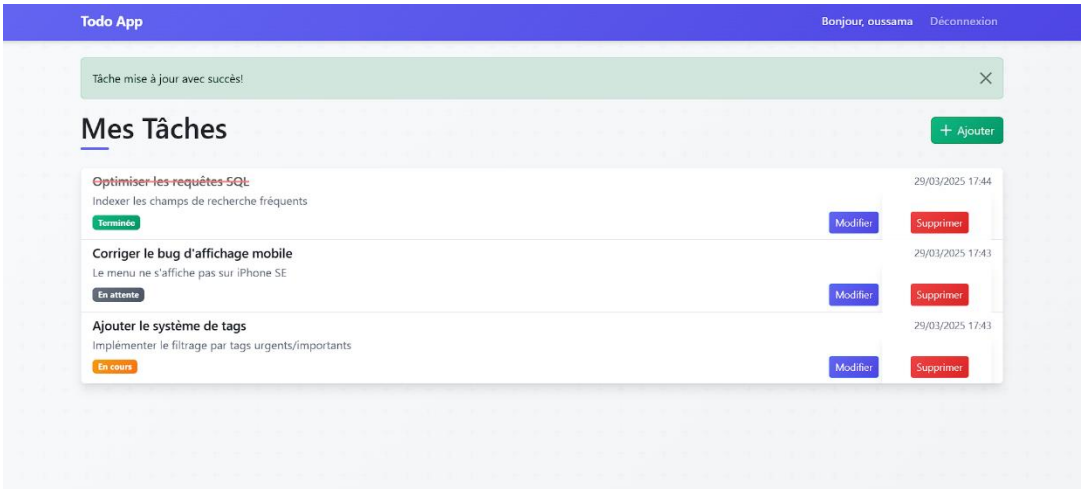
3. register :



4. autre :


Gestionnaire de Tâches
Votre application personnelle de liste de choses à faire
Bienvenue, oussama !
Vous êtes connecté(e) et prêt(e) à gérer vos tâches.
Que souhaitez-vous faire ?
→ Voir mes tâches
→ Ajouter une nouvelle tâche
Déconnexion


Todo App
Bonjour, oussama
Déconnexion
Ajouter une Tâche
Titre *
Corriger le bug d'affichage mobile
Description
Le menu ne s'affiche pas sur iPhone SE
Ajouter
Annuler


Todo App
Bonjour, oussama
Déconnexion
Tâche mise à jour avec succès!
Mes Tâches
+ Ajouter
Optimiser les requêtes SQL
29/03/2025 17:44
Indexer les champs de recherche fréquents
Terminée
Modifier
Supprimer
Corriger le bug d'affichage mobile
29/03/2025 17:43
Le menu ne s'affiche pas sur iPhone SE
En attente
Modifier
Supprimer
Ajouter le système de tags
29/03/2025 17:43
Implémenter le filtrage par tags urgents/importants
En cours
Modifier
Supprimer