 **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**

1. **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.

1. **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).

1. **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**.

* + 1. **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.
    1. **Base de données** :
* Table **users** : id, username, email, password, created\_at
* Table **tasks** : id, user\_id , title, description, status, created\_at
  1. **Vues :**

Templates HTML/Jinja2 pour l’affichage.

* + 1. **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.
  1. **Contrôleurs (Controllers):**

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

* + 1. **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()

1. **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
   1. **Connexion (Login):**

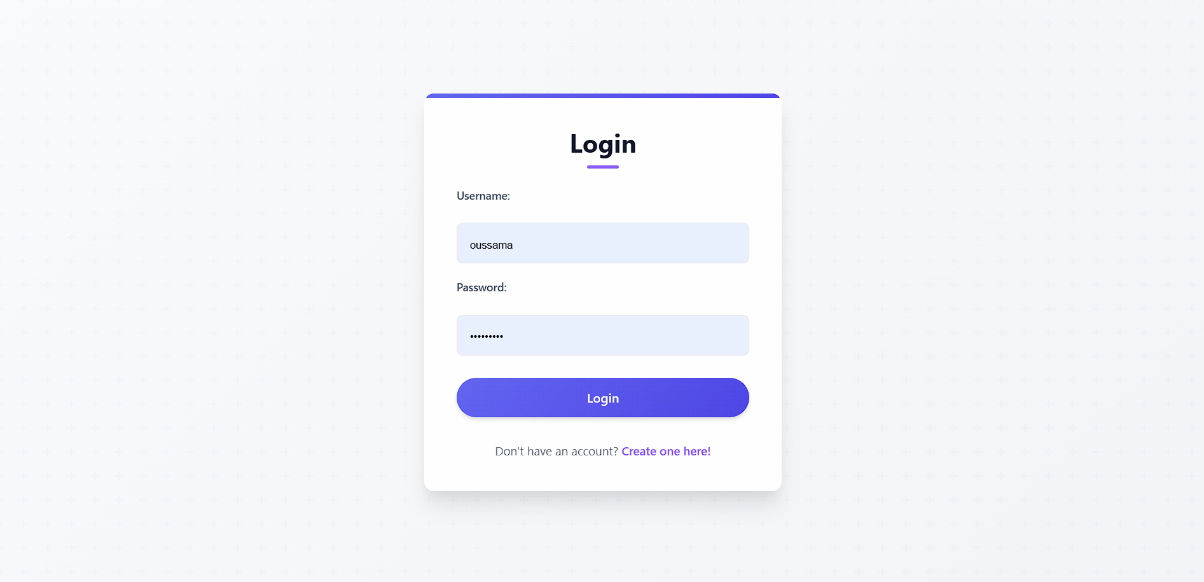
* **Processus :**

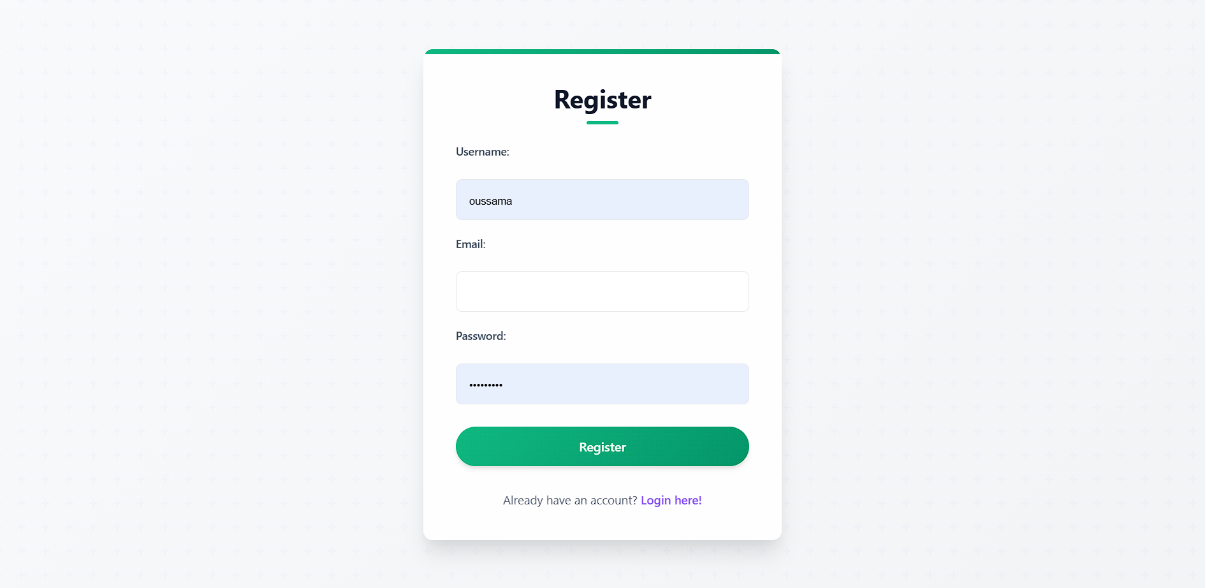
1. L'utilisateur saisit ses credentials
2. **AuthController.handle\_login() :**

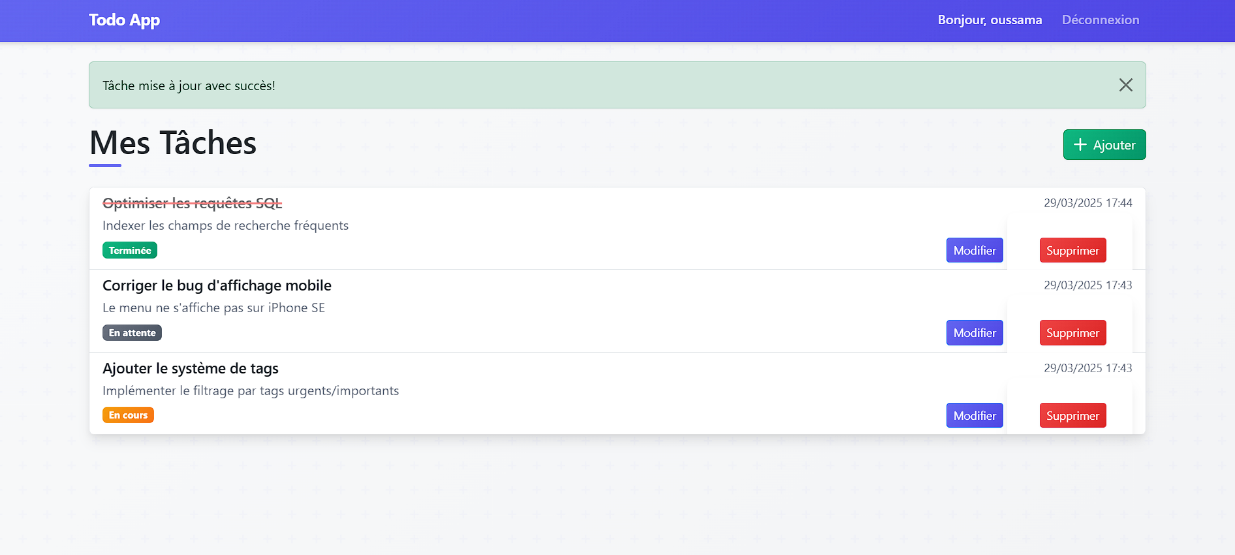
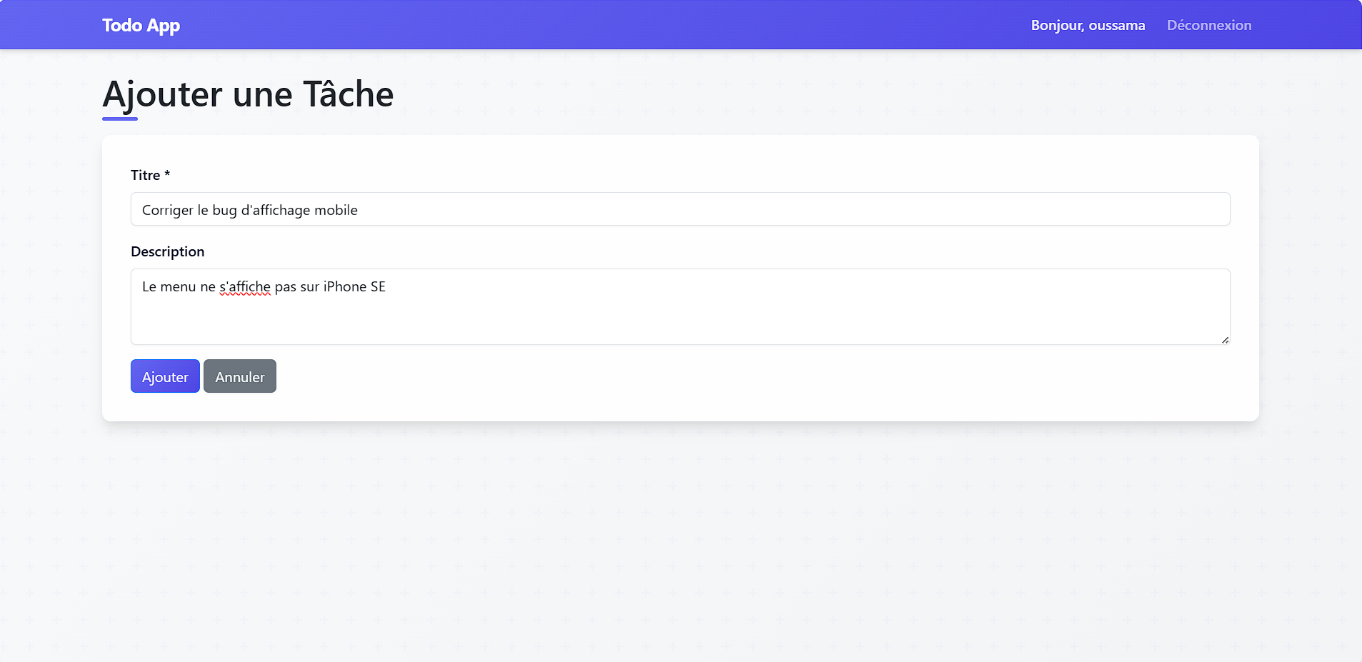
* Vérifie l'existence de l’utilisateur via UserModel.get\_user\_by\_username()
* Compare les mots de passe **en clair** (non sécurisé - critique à corriger)

1. Si valide :

* Crée une **session Flask** avec session['username'] = …
* Redirige vers la page d'accueil
  1. **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()**

1. **Captures d'Écran et Visualisation du Projet:**
   1. **Homepage:**
   2. **Login :**
   3. **register :**

****

* 1. **autre :**