**Rapport de Projet Python Testing**

**Nom du Projet :** Python\_Testing-9miseenforme  
**Date de Début :** [14-11-2024]  
**Date de Rapport :** [27-11-2024]

**1. Contexte et Objectifs**

Le projet **Python\_Testing-9miseenforme** a pour objectif de développer une application Python qui permet de valider des adresses e-mail et de gérer des clubs. Ce projet vise à mettre en œuvre des tests automatisés pour garantir la robustesse et la fiabilité du code. En suivant les meilleures pratiques de développement, nous visons à atteindre un niveau de couverture de test élevé et à nous assurer que toutes les fonctionnalités fonctionnent comme prévu.

**2. Environnement de Développement**

* **Système d'exploitation :** Windows
* **Environnement virtuel :** Créé à l'aide de venv pour isoler les dépendances du projet.
* **Outils de test :**
  + **Nose2** : Utilisé pour exécuter les tests unitaires et d'intégration.
  + **Pytest** : Moteur de test en arrière-plan pour Nose2.
  + **Locust** : Utilisé pour effectuer des tests de performance.
* **Framework Web :** Flask.

**3. Étapes de Développement**

**3.1. Mise en Place de Flask**

**3.1.1. Installation et Configuration**

* **Installation de Flask :** Flask a été installé via pip dans l'environnement virtuel.

pip install Flask

* **Création de l'application :** Un fichier server.py a été créé pour définir l'application Flask, y compris les fonctions nécessaires.

**3.1.2. Développement des Routes**

Des routes ont été définies dans server.py pour gérer les requêtes des utilisateurs, notamment :

* **Route d'accueil :** Pour afficher une page d'accueil avec des informations sur l'application.
* **Route de validation d'e-mail :** Pour recevoir une adresse e-mail via une requête POST et renvoyer la validation.
* **Route de gestion des clubs :** Pour charger et afficher les informations des clubs.

**3.1.3. Gestion des Erreurs**

Des gestionnaires d'erreurs ont été ajoutés pour capturer les exceptions et renvoyer des messages d'erreur appropriés, garantissant une expérience utilisateur fluide.

**3.2. Développement des Fonctionnalités**

**3.2.1. Validation des E-mails**

Les fonctions de validation des e-mails ont été développées pour traiter différents cas d'utilisation, notamment :

* Test d'une adresse e-mail vide.
* Test d'une adresse e-mail invalide.
* Test d'une adresse e-mail sans le symbole '@'.
* Test d'une adresse e-mail valide.

**3.2.2. Gestion de Clubs**

Des méthodes ont été créées pour gérer les données des clubs, telles que :

* Chargement des éléments de club.
* Vérification de la structure des données de club.

**3.3. Mise en Place des Tests**

**3.3.1. Structure des Tests**

Le répertoire des tests se compose de plusieurs sous-dossiers, chacun ayant une fonction spécifique :

tests/

├── integration\_tests/

├── test\_fonctionnels/

├── test\_unitaires/

├── \_\_pycache\_\_/

├── test\_by\_nose2.py

└── \_\_init\_\_.py

**3.3.2. Tests Unitaires**

Les tests unitaires ont été écrits pour chaque fonctionnalité afin de garantir leur bon fonctionnement.

**3.3.3. Tests d'Intégration**

Des tests d'intégration ont été écrits pour valider les interactions entre les différentes parties de l'application.

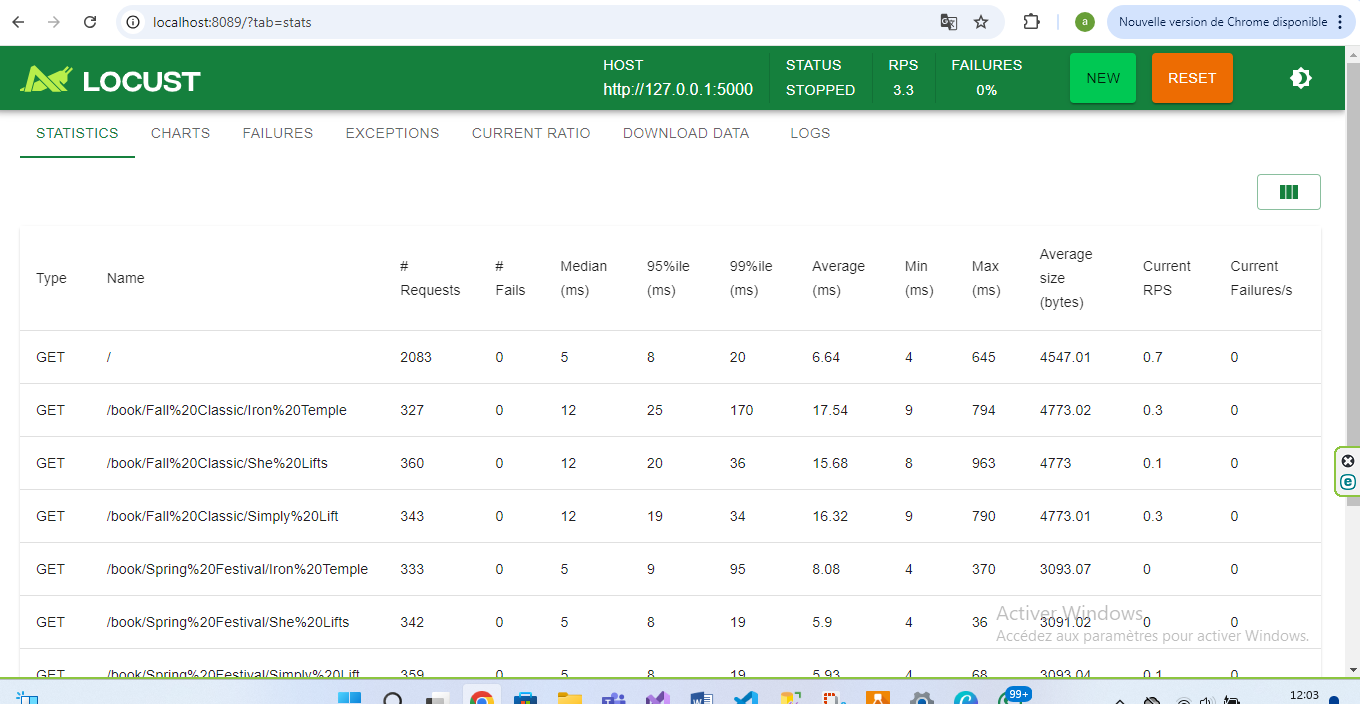
**3.3.4. Tests de Fonctionnalité**

Les tests de fonctionnalité ont été élaborés pour garantir que chaque fonctionnalité de l'application répond aux spécifications définies.

**3.3.5. Tests de Performance**

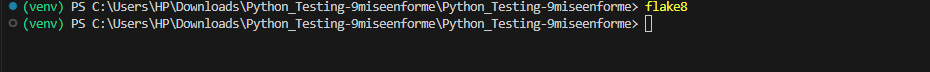
Des tests de performance ont été effectués à l'aide de Locust pour évaluer le temps de réponse et la capacité de l'application à gérer une charge d'utilisateurs simultanés.

**Rapport de Locust**

****

* **Host :** <http://127.0.0.1:5000>
* **Statut :** Stopped
* **RPS :** 3.3
* **Échecs :** 0%
* **Total des requêtes :** 8110
* **Requêtes réussies :** 8110
* **Temps de réponse moyen :** 8.18 ms

**3.4. Utilisation de PEP 8**

****

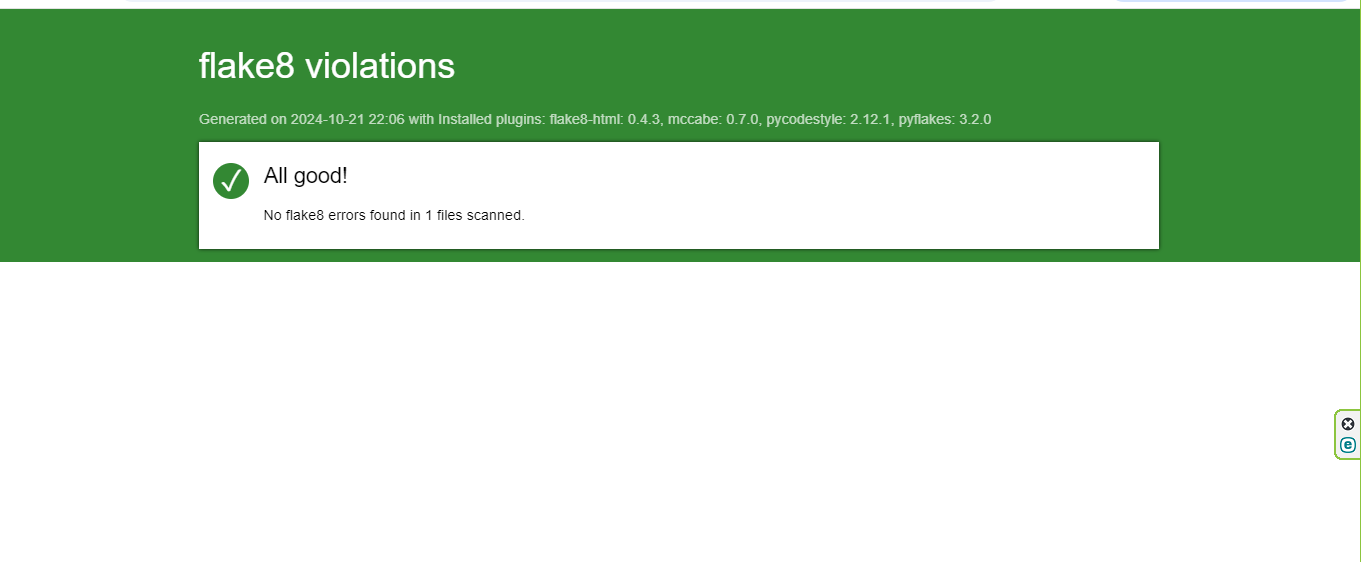
Des efforts ont été faits pour respecter les conventions de style PEP 8, incluant :

* **Indentation :** 4 espaces pour l'indentation.
* **Longueur des lignes :** Limitation à 79 caractères.
* **Documentation :** Ajout de docstrings pour documenter les fonctions et classes.

**Outil utilisé :** Flake8 pour vérifier le respect de ces conventions.

**4. Rapport sur les Violations Flake8**

**Date de Génération :** 2024-10-21 22:06  
**Plugins Installés :**



* **flake8-html :** 0.4.3
* **mccabe :** 0.7.0
* **pycodestyle :** 2.12.1
* **pyflakes :** 3.2.0

**Résumé des Résultats :**

* **État :** ✓ All good!
* **Nombre de Fichiers Scannés :** 1
* **Erreurs Flake8 Trouvées :** Aucune

**5. Exécution des Tests**

**5.1. Commande Exécutée**

Les tests ont été exécutés à l'aide de la commande suivante :

pytest

**5.2. Résultats des Tests**

* **Total des tests exécutés :** 16
* **Tests réussis :** 16
* **Tests échoués :** 0

**5.3. Détails des Tests**

Tous les tests ont réussi, comme indiqué ci-dessous :

tests\integration\_tests\test\_integration.py . [ 6%]

tests\test\_fonctionnels\test\_home\_page.py . [ 12%]

tests\test\_fonctionnels\test\_reservation.py . [ 18%]

tests\test\_unitaires\test\_booking.py ... [ 37%]

tests\test\_unitaires\test\_login.py ... [ 56%]

tests\test\_unitaires\test\_somme.py .... [ 81%]

tests\test\_unitaires\test\_utils.py .. [ 93%]

tests\test\_unitaires\test\_validate\_email.py . [100%]

**6. Conclusion**

Le projet **Python\_Testing-9miseenforme** a été mis en place avec succès, avec des fonctionnalités de validation d'e-mails et de gestion des clubs. Tous les tests unitaires et d'intégration ont été exécutés avec succès, ce qui confirme que toutes les fonctionnalités de l'application fonctionnent comme prévu.