

Python Testing : Rapport de Test et Validation du Projet

VRY AKA JOSEPH

## Introduction

Ce rapport présente une analyse complète des tests effectués sur le projet. Les tests incluent les tests unitaires, les tests d'intégration et les tests de performance avec Locust. L'objectif est d'assurer la qualité, la fiabilité et les performances optimales de l'application dans diverses conditions d'utilisation.

## Tests Unitaires

Les tests unitaires vérifient le fonctionnement isolé de chaque fonction pour garantir leur fiabilité individuelle. Ils permettent de s'assurer que chaque fonction produit les résultats attendus avec des données valides et gère correctement les cas d'erreurs.

Scénarios de test unitaire :

* - Chargement des données des clubs avec `loadClubs` pour vérifier leur disponibilité.
* - Chargement des données des compétitions avec `loadCompetitions` pour vérifier leur disponibilité et la gestion des compétitions terminées.
* - Test de l'authentification et de la redirection via `showSummary` pour les emails valides et invalides.
* - Validation des réservations et de la gestion des points pour des demandes conformes et excessives.

## Tests d'Intégration

Les tests d'intégration permettent de vérifier que les différents composants du système fonctionnent bien ensemble. Ils simulent des interactions complètes entre plusieurs parties du code pour assurer une cohérence et une interaction correcte entre les composants.

Scénarios de test d'intégration :

* - Vérification de la connexion et de la déconnexion avec des emails valides et invalides.
* - Validation de la réservation d'un nombre raisonnable de places pour chaque compétition.
* - Simulation de la réservation avec des points insuffisants et vérification des messages d'erreur.
* - Tentative de réservation pour des compétitions déjà terminées afin de vérifier les restrictions.

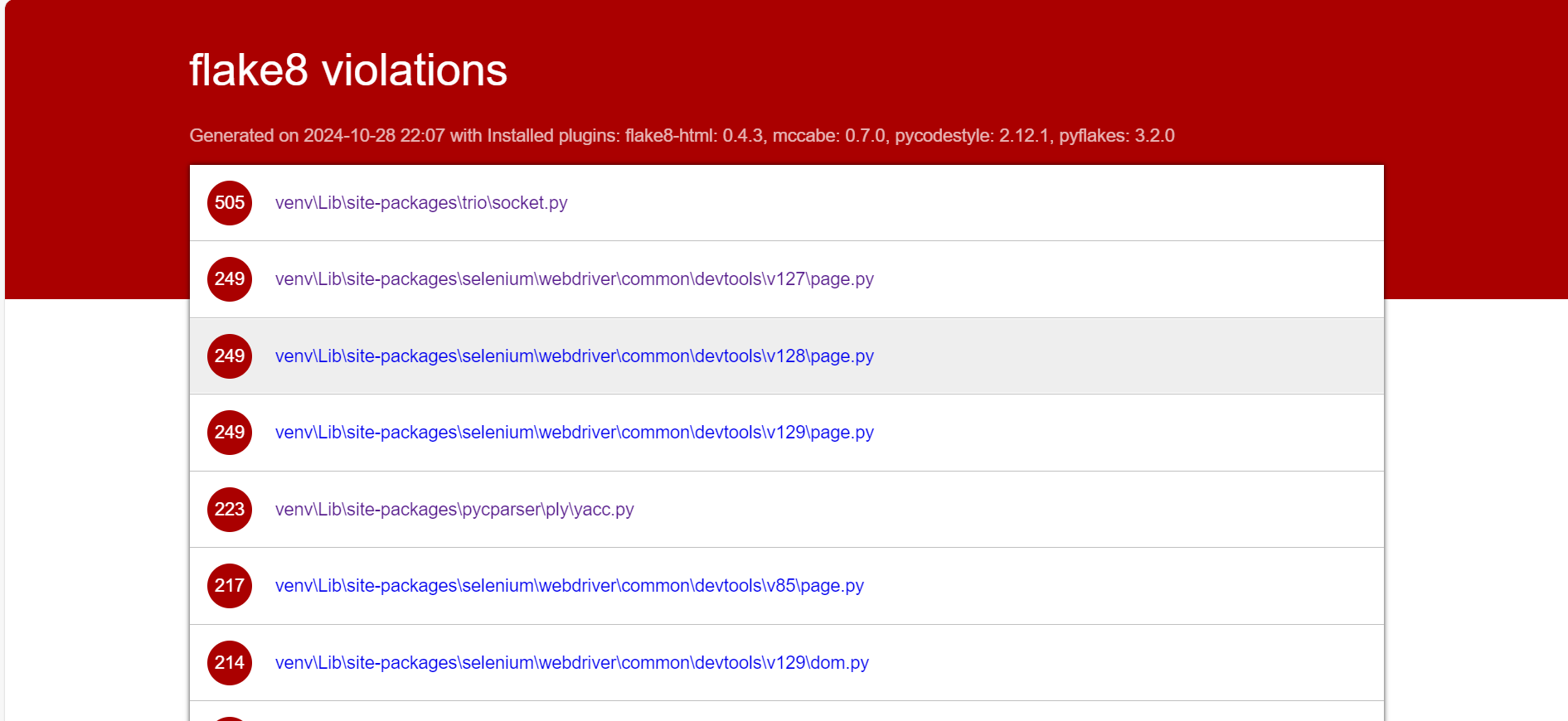
## Tests de Performance avec Locust

Les tests de performance, réalisés avec Locust, évaluent la capacité de l'application à gérer un grand nombre d'utilisateurs simultanés. Ils permettent de déterminer la robustesse et la réactivité de l'application sous une charge importante.

Scénarios de test de performance avec Locust :

* - Accès simultané à la page d'accueil pour tester le comportement en charge.
* - Simulation de connexions avec un email valide pour évaluer la réactivité du serveur.
* - Test de déconnexion sous charge pour vérifier la stabilité et la fluidité de l'expérience utilisateur.

## RAPPORT FLAKE8



Le rapport Flake8 montre un grand nombre de violations de style dans des fichiers Python situés dans le dossier `venv\Lib\site-packages`, qui fait partie de mon environnement virtuel. Ces fichiers appartiennent à des bibliothèques externes et ne font normalement pas partie du code source de mon projet, donc ils ne devraient pas être analysés pour des erreurs de style.

Les nombres à côté des fichiers (comme "505" ou "249") représentent le nombre de violations dans chaque fichier. Par exemple, `venv\Lib\site-packages\trio\socket.py` contient 505 violations, et plusieurs fichiers sous `selenium\webdriver` en montrent 249.

En résumé, la majorité des violations proviennent des bibliothèques installées dans l'environnement virtuel, ce qui complique l’analyse et m'empêche de me concentrer sur les erreurs de style dans mon propre code.

## Résultats Attendues

Les résultats attendus pour chaque type de test sont les suivants :

1. 1. Tests Unitaires : Chaque fonction doit retourner les résultats escomptés avec des entrées valides et gérer les erreurs de manière appropriée pour des entrées non valides.
2. 2. Tests d'Intégration : Les composants doivent fonctionner en harmonie, sans erreur, lors des interactions entre eux.
3. 3. Tests de Performance : L'application doit maintenir des temps de réponse acceptables et supporter une charge d'utilisateurs simultanés sans dégradations significatives.

## Conclusion

Ce rapport démontre que les tests mis en place couvrent efficacement les fonctionnalités essentielles du projet. Les tests unitaires valident le bon fonctionnement de chaque composant isolé, les tests d'intégration confirment l'interaction correcte entre les composants, et les tests de performance montrent la robustesse de l'application sous une forte charge. Il est recommandé de continuer à exécuter ces tests régulièrement, notamment après chaque mise à jour, pour garantir une qualité constante.