LOG3430 - Méthodes de test et de validation du logiciel

Laboratoire 2

Tests de mutations

Hisham Boulifa - 2085232

Pungtzé-Sy D. Kamdem - 2139333

Gr.: 02

26 février 2024

**Question 1**

Combien de mutants ont été tués ? 117

Combien ont expiré (timed out) ? 0

Combien étaient suspects ? 0

Combien ont survécu ? 237

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Figure 1. Résultats des tests après mutations

**Question 2**

En utilisant les deux rapports, quelle est la couverture d’instruction du fichier avec le plus de mutants survivants ? 31%

Quel est le nom du fichier ? handler\_async.py

Quelle est la couverture d’instruction du fichier avec le moins de mutants survivants ? Étant donné que les fichiers cache\default.py, cache\interface.py, version.py et details.py sont très petits, nous avons identifiés, parmi les plus volumineux, le fichier avec le moins de mutants survivants dont la couverture de code est de 86%

Quel est le nom du fichier ? bogon.py

**Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement**

Figure 2. Rapport HTML généré par Mutmut

Une image contenant texte, nombre, Police, capture d’écran

Description générée automatiquement

Figure 3. Couverture de code

**Question 3**

Trouvez quatre mutants créés à partir de quatre types de mutateurs différents, affichez ces mutants et nommez le type de chaque mutant.

**Question 4**

Créez de nouveaux tests ou corrigez les tests d’ipinfo pour tuer les mutants que vous avez identifiés à la question 3. Quels tests avez-vous modifiés (donnez le chemin des fichiers et le nom des tests) et pourquoi avez-vous fait les modifications que vous avez faites ?

Relancez Mutmut, quels sont les nouveaux résultats ? Vous devez aussi soumettre le repo ipinfo complet, contenant le rapport Mutmut, sous la forme de fichier zip nommé Q4.zip.

Figure 4.

Figure 5. Couverture de code de la question 4

Figure 6. Couverture de code de la question 5

Figure 7. Couverture de code de la question 6 avec la méthode *coverage.py*

Figure 8. Couverture de code de la question 6 avec la méthode *pytest-cov*

Figure 9. Tests ajoutés

Figure 10. Couverture de code de la question 7

Figure 11. Lignes couvertes grâce aux nouveaux tests