Text

Description automatically generated with medium confidence

LOG3430 -Méthodes de test et de validation du logiciel

TP1 – Tests unitaires

Groupe 1

Équipe 7

Hugo Lachieze-Rey (1934177)

Dimitry Kamga (1898357)

Remis à :

Hanane Ikhelef

Hiver 2022

Table des matières

[1. Tests unitaires 3](#_Toc94984678)

[a. Couverture avant les tests 3](#_Toc94984679)

[b. Couvertures après les tests 3](#_Toc94984680)

[c. Ajout de tests custom 3](#_Toc94984681)

[2. Tests de flots de données 4](#_Toc94984682)

[a. Ajout de la fonction compute\_user\_trust 4](#_Toc94984683)

[b. Graphe CFG 5](#_Toc94984684)

[c. Tableaux des nœuds 6](#_Toc94984685)

[d. Cas de tests 8](#_Toc94984686)

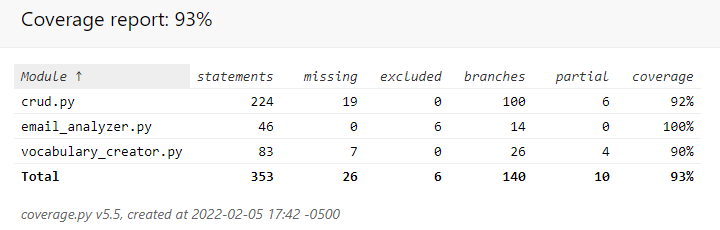
# Tests unitaires

## Couverture avant les tests

Table

Description automatically generated with medium confidence

## Couvertures après les tests



## Ajout de tests custom

Nous avons ajouté des tests custom afin d’augmenter notre *coverage* pour chacun des fichiers demandés :

**Email Analyzer**

def test\_spam\_ham\_body\_prob\_Returns\_expected\_probability\_ham(self, mock\_load\_dict):

def test\_subject\_spam\_ham\_prob\_Returns\_expected\_probability\_ham(self, mock\_load\_dict):

**Vocabulary Creator**

def test\_write\_data\_to\_vocab\_file\_returns\_false(self):

def test\_write\_data\_to\_vocab\_file\_return\_true(self):

def test\_clean\_text\_should\_return\_text (self, mock\_clean\_text):

**Crud**

def test\_add\_new\_user\_return\_false\_on\_email\_unicity\_and\_format(

self,mock\_read\_groups\_file,mock\_read\_users\_file

):

def test\_get\_new\_user\_id\_when\_adding\_correct\_user(

self, mock\_read\_users\_file):

def test\_get\_new\_group\_id\_when\_adding\_correct\_group(

self, mock\_read\_groups\_file):

def test\_add\_new\_group\_fail\_name\_in\_groups\_lookup(

self, mock\_read\_groups\_file, mock\_read\_users\_file

):

def test\_add\_new\_group\_Passes\_correct\_values(

self, mock\_get\_user\_id, mock\_read\_groups\_file, mock\_read\_users\_file

):

def test\_update\_users\_Passes\_correct\_and\_invalid\_is\_Date\_of\_last\_seen\_message\_data\_to\_modify\_users\_file\_when\_field(

self, mock\_read\_users\_file, mock\_modify\_users\_file

):

def test\_update\_users\_Passes\_correct\_and\_invalid\_is\_Date\_of\_first\_seen\_message\_data\_to\_modify\_users\_file\_when\_field(

self, mock\_read\_users\_file, mock\_modify\_users\_file

):

def test\_update\_users\_Passes\_correct\_and\_invalid\_Trust\_data\_to\_modify\_users(

self, mock\_read\_users\_file, mock\_modify\_users\_file

):

def test\_update\_users\_Passes\_correct\_and\_invalid\_SpamN\_data\_to\_modify\_users(

self, mock\_read\_users\_file, mock\_modify\_users\_file

):

def test\_update\_users\_Passes\_correct\_Groups\_modify\_user\_file(

self, mock\_read\_users\_file, mock\_modify\_users\_file, mock\_read\_groups\_file

):

def test\_update\_users\_return\_False\_on\_invalid\_user\_Groups(

self, mock\_read\_users\_file, mock\_modify\_users\_file, mock\_read\_groups\_file

):

def test\_update\_groups\_Passes\_correct\_and\_invalid\_field\_Trust\_data\_to\_modify\_groups\_file(

self, mock\_read\_groups\_file, mock\_modify\_groups\_file

):

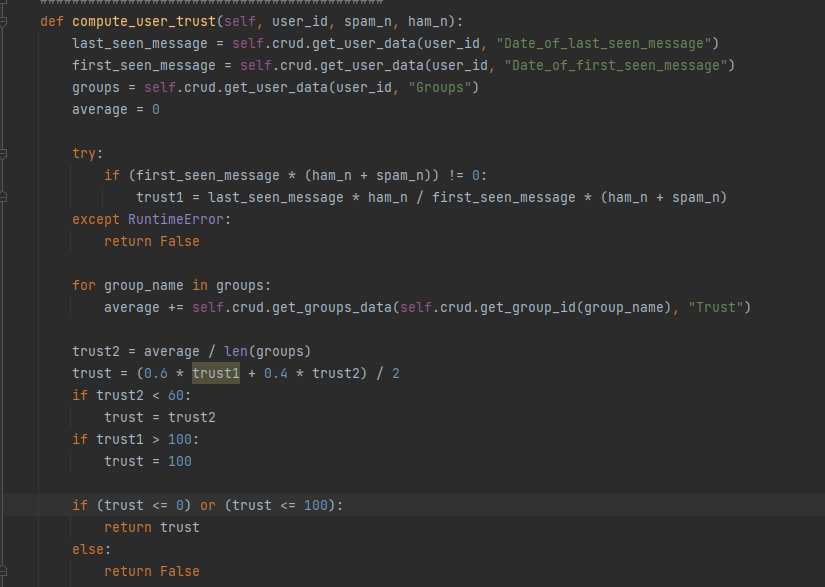
def test\_update\_groups\_Passes\_correct\_on\_invalid\_List\_of\_members(

self,mock\_read\_users\_file,mock\_read\_groups\_file,mock\_modify\_groups\_file,

):

# Tests de flots de données

## Ajout de la fonction compute\_user\_trust



## Graphe CFG

Diagram, engineering drawing

Description automatically generated

## Tableaux des nœuds

|  |  |  |  |
| --- | --- | --- | --- |
| Nœud | DEF | C-USE | P-USE |
| user\_id | 1 | 1 |  |
| spam\_n | 1 | 2, 3 | 2 |
| ham\_n | 1 | 2, 3 | 2 |
| last\_seen\_message | 1 | 3 |  |
| first\_seen\_message | 1 | 2, 3 |  |
| trust1 | 3 | 6 | 9 |
| average | 1, 5 | 6 |  |
| groups | 1 | 6 | 4 |
| group\_name | 4 | 5 | 4 |
| trust2 | 6 | 6 | 7 |
| trust | 6, 8, 10 | 12 | 11 |

## Cas de tests

**ALL-DEFinition coverage**

Un jeu de tests T satisfait le critère « all-Defs » pour le programme P pour chaque variable v, il doit au moins avoir un chemin « definition-clear » pour chaque nœud de définition de v.

|  |  |
| --- | --- |
| Variable | Defs |
| user\_id | DEF(user\_id, 1) |
| spam\_n | DEF(spam\_n, 1) |
| ham\_n | DEF(ham\_n, 1) |
| last\_seen\_message | DEF(last\_seen\_message, 1) |
| first\_seen\_message | DEF(first\_seen\_message, 1) |
| trust1 | DEF(trust1, 3) |
| average | DEF(average, 1)  DEF(average, 5) |
| groups | DEF(groups, 1) |
| group\_name | DEF(group\_name, 4) |
| trust2 | DEF(trust2, 6) |
| trust | DEF(trust, 6)  DEF(trust, 8)  DEF(trust, 10) |

Jeu de test : d1 = <{ len(groups) > 0, trust2 > 60, trust1 < 100, trust > 0, {false}}>

**All C-USE Coverage**

« All C-Use coverage » est une couverture par un jeu de tests T qui contient un DC-PATH(v, n, m) pour chaque DEF(v, n) qui mène vers tous les C-USE(v, n) et menant à quelques P-USE(v, n) couvrant les deux branches de chaque P-USE.

|  |  |
| --- | --- |
| Variable | Chemin |
| user\_id | Path\_ALL\_C\_USE\_user\_id :{1,1} avec un C-USE (user\_id, 1) |
| spam\_n | Path\_ALL\_C\_USE\_spam\_n:{1,2} avec un C-USE (spam\_n, 2) |
| ham\_n | Path\_ALL\_C\_USE\_ham\_n:{1,2} avec un C-USE (ham\_n, 2) |
| last\_seen\_message | Path\_ALL\_C\_USE\_last\_seen\_message:{1,2,3} avec un C-USE (last\_seen\_message, 3) |
| first\_seen\_message | Path\_ALL\_C\_USE\_first\_seen\_message:{1,2} avec un C-USE (first\_seen\_message, 2)  Path\_ALL\_C\_USE\_first\_seen\_message:(1,2,3) avec un C-USE (first\_seen\_message, 3) |
| trust1 | Path\_ALL\_C\_USE\_trust1:(3,4,6) avec un C-USE (trust1, 6)  Path\_ALL\_C\_USE\_trust1:(3,4,5,4,6,7,9) avec un C-USE (trust1, 9) |
| average | Path\_ALL\_C\_USE\_average:(1,2,3,4,6) avec un C-USE (average, 6) |
| groups | Path\_ALL\_C\_USE\_groups :(1,2,3,4,6) avec un C-USE (groups, 6) |
| group\_name | Path\_ALL\_C\_USE\_group\_name :(4,5) avec un C-USE (group\_name, 5) |
| trust2 | Path\_ALL\_C\_USE\_trust2:(6,7) avec un C-USE (trust2, 7) |
| trust | Path\_ALL\_C\_USE\_trust:(6,7,9,11,12) avec un C-USE (trust, 12) |

Jeu de tests

d1 = <{groups.length > 0, , trust2 > 60, trust1 < 100, 0 <= trust <= 100, {trust}}>

d2 = <{groups.length > 0, trust2 < 60, trust1 < 100, {false}}>

d3 = <{groups.length > 0, trust2 > 60, trust1 > 100, trust < 0, {false}}>

d4 = <{groups.length > 0, trust2 > 60, trust1 < 100, trust > 100, {false}}>

**All P-USE Coverage**

Si nous avons une variable v dans un noeud n où cette variable est utilisée comme prédicat, il peut être écrit *P-USE(v,n).* Pour que notre suite de tests respecte le critère *ALL-P-Use*, il doit y avoir au moins un chemin *DEF-CLEAR* qui part de la définition d'une variable *x* vers toutes les utilisations de *x* comme prédicat.

|  |  |
| --- | --- |
| Variable | Chemin |
| user\_id | Path\_ALL\_P\_USE\_user\_id :{1} |
| spam\_n | Path\_ALL\_P\_USE\_spam\_n:{1,2} avec un P-USE (spam\_n, 3) |
| ham\_n | Path\_ALL\_P\_USE\_ham\_n:{1,2} avec un P-USE (ham\_n, 2) |
| last\_seen\_message | Path\_ALL\_P\_USE\_last\_seen\_message:{1} |
| first\_seen\_message | Path\_ALL\_P\_USE\_first\_seen\_message:{1,2} avec un P-USE (first\_seen\_message, 2) |
| trust1 | Path\_ALL\_P\_USE\_trust1:{3,4,6,7,9} avec un p-USE (trust1, 9)  Path\_ALL\_P\_USE\_trust1:{3,4,5,4,6,7,9} avec un p-USE (trust1, 9) |
| average | Path\_ALL\_P\_USE\_average:(1,2,3,4,6) avec un C-USE (average, 6) |
| groups | Path\_ALL\_P\_USE\_groups :{1} |
| group\_name | Path\_ALL\_P\_USE\_group\_name :{4} |
| trust2 | Path\_ALL\_P\_USE\_trust2:{6,7} avec un P-USE (trust2, 7) |
| trust | Path\_ALL\_P\_USE\_trust:{6,7,9,11} avec un P-USE (trust, 11) |

Jeux de tests

d1 = <{groups.length > 0, , trust2 > 60, trust1 < 100, 0 <= trust <= 100, {trust}}>

d2 = <{groups.length > 0, trust2 < 60, trust1 < 100, {false}}>

d3 = <{groups.length > 0, trust2 > 60, trust1 < 100, {false}}>

**All USE Coverage**

Un jeu de tests “T” satisfait le critère “all-Uses” si pour chaque variable “v” d’un programme “p”, il y a au moins un chemin de “definition-clear” en commençant par le noeud de définition jusqu’à chaque noeud d’utilisation de v.

|  |  |
| --- | --- |
| Variable | Chemin |
| user\_id | Path\_ALL\_USE\_user\_id :{1} |
| spam\_n | Path\_ALL\_USE\_spam\_n:{1,2,3} |
| ham\_n | Path\_ALL\_USE\_ham\_n:{1,2,3} |
| last\_seen\_message | Path\_ALL\_USE\_last\_seen\_message:{1,2,3} |
| first\_seen\_message | Path\_ALL\_USE\_first\_seen\_message:{1,2,3} |
| trust1 | Path\_ALL\_USE\_trust1:{3,4,6,9} |
| average | Path\_ALL\_USE\_average:{1,2,3,4,6} |
| groups | Path\_ALL\_USE\_groups :{1,2,3,4,6} |
| group\_name | Path\_ALL\_USE\_group\_name :{4,5} |
| trust2 | Path\_ALL\_USE\_trust2:{6,7,8} |
| trust | Path\_ALL\_USE\_trust:{6,7,9,11,12} |

Jeux de tests

d1 = <{groups.length > 0, , trust2 > 60, trust1 < 100, 0 <= trust <= 100, {trust}}>

d2 = <{groups.length > 0, trust2 < 60, trust1 < 100, {false}}>

d3 = <{groups.length > 0, trust2 > 60, trust1 > 100, trust < 0, {false}}>

d4 = <{groups.length > 0, trust2 > 60, trust1 < 100, trust > 100, {false}}>

**Critère le plus strict**

Après avoir fait les couvertures du All-DEF, All C-USE, All P-USE et All USE, il est possible de remarquer que le critère le plus restrictif c’est le critère de couverture All USE.

- All-DEFS est moins strict car il faut au moins un DC-Path pour chaque DEF

- C-USE est moins strict car il ne traverse pas les P-USES

- P-USE est moins strict car il ne traverse pas les C-USES