2016

SEF01

Bryan Fernand, Michael Drent, Thijs van den Boogaard, Joyce Lam, Jonathan Ratnavel, Alex Abdalla

1-6-2016

Testdocumentatie Appkiller

****

# Voorwoord

In dit verslag wordt uitgelegd wat het doel is van testen, wat het verwachtte eindresultaat is van het testen en welke verschillende testsoorten er zijn. Bij het maken van dit verslag hebben wij het werk verspreid over de hele projectgroep zodat het werk zo eerlijk mogelijk over de groep verspreid is.

Ik wil graag alle leden van SEF01 bedanken voor het goed communiceren en samenwerken aan het testdocumentatie verslag. Ook willen wij graag Eric Nieuwenhuis bedanken voor zijn coaching tijdens het project. Dit verslag is onderdeel van Project PAD, waarvoor dit projectteam de Appkiller maakt.

Inhoud

[Voorwoord 1](#_Toc453154539)

[Inleiding 3](#_Toc453154540)

[1. Huidige situatie 4](#_Toc453154541)

[1.1. Achtergrond opdrachtgever 4](#_Toc453154542)

[1.2. Het Doel van testen 4](#_Toc453154543)

[1.3. Verwachtte eindresultaat 4](#_Toc453154544)

[2. Testdraaiboek 5](#_Toc453154545)

[3. Testresultaten 5](#_Toc453154546)

# Inleiding

Dit verslag is gemaakt om te kijken naar de huidige situatie op het gebied van applicatie uitzetters, wat het doel is van testen, wat het verwachtte eindresultaat zal zijn en hoe wij het hebben getest. Hierdoor is er een beter overzicht van wat er verwacht wordt tijdens het testen en zijn de testresultaten terug te vinden.

Dit verslag is onderdeel van Project PAD. Er wordt gewerkt in dienst van het SEFlab, een instelling die zich bezig houdt met het zuiniger maken van software. In dit project wordt er gewerkt aan een applicatie genaamd “Appkiller”. Het doel van deze applicatie is het mogelijk maken van het uitzetten van applicaties die van te voren door de standaard configuratie zijn bepaald of door een eigen configuratie , zodat de computer minder energie verbruikt.

In hoofdstuk 1 wordt aangegeven wat het doel is van testen, in hoofdstuk 2 wordt het verwachtte eindresultaat weergeven en in hoofdstuk 3 worden de verschillende testsoorten hun testcases in kaart gebracht.

# Huidige situatie

## Achtergrond opdrachtgever

SEFlab is een instelling die zich inzet voor een lager energieverbruik voor software. Dit doet zij door hardware te bestuderen en te kijken hoe de software op de hardware energiezuiniger kan worden geprogrammeerd. SEFlab realiseert dit soort projecten door nauw samen te werken met software specialisten van de Software Improvement Group (SIG), Elektrische ingenieurs van de HVA en vele andere partners. De huidige doelen van het SEFLab zijn:

* Een model dat datacenters en IT afdelingen helpt met het voorspellen en verminderen van hun energie serverconsumptie.
* Het maken van richtlijnen en instrumenten voor software developers om te helpen met groene coderingsprincipes.
* Het maken van een energielabel voor software applicaties om eindgebruikers te helpen het kiezen van aankopen op basis van energie-efficiëntie

## Het Doel van testen

Het doel van testen is het op de proef stellen van de applicatie. Er wordt niet alleen gekeken of alle functies in de applicatie zonder enige complicatie werken, maar er wordt ook gekeken of alle functies wel volledig in de applicatie aanwezig zijn. Zonder testen is de kans groot dat na de uitbrengfase de gebruiker op talloze bugs en foutmeldingen stuit. Hierdoor kan het voorkomen dat de applicatie vrijwel niet bruikbaar is en de applicatie na de uitbrengfase alsnog verbeterd moet worden.

De applicatie zal niet alleen worden getest door het team SEF01, maar ook door de gebruikers. Zo kan er goed getest worden op gebruiksgemak en kan de eindgebruiker een mening geven over de vormgeving van de applicatie. Dit wordt meestal gedaan in de alpha of beta-fase van een applicatie.

## Verwachtte eindresultaat

Er wordt verwacht dat de applicatie na de testfase geen complicaties met zich zal meebrengen. Er wordt verwacht dat na de testfase alle functionele eisen en non-functionele eisen goed zijn verwerkt in de applicatie en er geen bugs in het systeem aanwezig zijn. Door het te laten testen door een groep eindgebruikers verwachten wij dat alle aanwezige bugs en chrash oorzaken worde gevonden en worden verwijdert door het projectteam.

# Testdraaiboek

*In dit hoofdstuk wordt weergeven welke stappen worden uitgevoerd tijdens het testen.*

In het testdraaiboek worden alle stappen van testen uitgelegd. Dit wordt gedaan zodat de programmeurs die gaan testen goed weten hoe het programma moet werken en welke stappen ze moeten doorgaan. Hieronder de opsomming van de teststappen die worden doorlopen:

* Ten eerste wordt het test ID weergeven. Dit wordt gedaan zodat de testcases niet door elkaar gehaald kunnen worden omdat elke test een uniek ID heeft.
* Ten tweede wordt de naam van de test weergeven. Zo is het voor iedereen duidelijk wat er getest gaat worden.
* Ten derde wordt er een korte descriptie gegeven van de testnaam. Dit wordt gedaan zodat de test nog meer verduidelijkt wordt.
* Ten vierde wordt er aangegeven welke use-cases gerelateerd zijn aan de test-case. Zo is het duidelijker voor de programmeur hoe dat deeltje functionaliteit eruit hoort te zien.
* Ten vijfde worden alle stappen weergeven die de programmeurs succesvol moeten ondergaan.
* Als laatste worden de verwachte resultaten gedocumenteerd. Hierdoor weten de programmeurs goed hoe het systeem op de stappen moet reageren.

# Testresultaten

*Hier worden alle testcases met de stappen doorgeven. Alle functionaliteiten van het systeem worden in de testcases weergeven en getest. De programmeurs moeten de testcases volledig doorlopen tijdens het testen en de structuur van de test cases aanhouden.*

**Test 1.**

|  |  |
| --- | --- |
| **Test Case ID** | Kill 0.1 |
| **Test Name** | Testing Kill |
| **Description** | Kills standard processes. |
| **Related Use Cases** | Use case 1 |
| **Steps** | 1. Open application 2. Click ‘Kill’ 3. Click ‘OK’ |
| **Expected result (post-conditions)** | Processes are killed and a screen pops-up: “[*iexplorer.exe …. Jusched.exe*] are killed”. |

**Test 2.**

|  |  |
| --- | --- |
| **Test Case ID** | Kill 0.2 |
| **Test Name** | Kill - Kill own list |
| **Description** | Processes in selected list are killed. |
| **Related Use Cases** | 1, 2, 3 |
| **Steps** | 1. Open application 2. Click ‘Processes’ or ‘Kill’(if ‘Kill’, go to 4.) 3. Select process 4. Click ‘Kill’ 5. Click ‘Unkill processes’ 6. Type name in empty bos 7. Click ‘in database’ 8. Click ‘Kill’ 9. Choose name of list in drop-down menu 10. Click ‘Kill eigen lijst’ |
| **Expected result (post-conditions)** | Processes are saved in a list and a screen pops-up: “Processen in [*name of list*] are killed”. |

P**rocesses**

**Test 1.**

|  |  |
| --- | --- |
| **Test Case ID** | processes 0.1 |
| **Test Name** | Processes - Kill |
| **Description** | Kills selected processes |
| **Related Use Cases** | 2 |
| **Steps** | 1. Open application 2. Click ‘Processes’ 3. Select process 4. Click ‘Kill’ |
| **Expected result (post-conditions)** | Selected processes are killed and a screen pops-up: “[*name selected*] is killed”. |

**Test 2.**

|  |  |
| --- | --- |
| **Test Case ID** | processes 0.2 |
| **Test Name** | processes - Refresh |
| **Description** | Refreshes list |
| **Related Use Cases** | 2 |
| **Steps** | 1. Open application 2. Click ‘Processes’ 3. Click ‘Refresh’ |
| **Expected result (post-conditions)** | Processlist is refreshed |

**Test 3.**

|  |  |
| --- | --- |
| **Test Case ID** | processes 0.3 |
| **Test Name** | processes - Sorting ‘Name’ |
| **Description** | Sorts list by ‘Name’ |
| **Related Use Cases** | 2 |
| **Steps** | 1. Open application 2. Click ‘Processes’ 3. Click ‘Name’ |
| **Expected result (post-conditions)** | Processlist sorted by ‘Name’ |

**Test 4.**

|  |  |
| --- | --- |
| **Test Case ID** | processes 0.4 |
| **Test Name** | processes - Sorting ‘PID’ |
| **Description** | Sorts list by ‘PID’ |
| **Related Use Cases** | 2 |
| **Steps** | 1. Open application 2. Click ‘Processes’ 3. Click ‘PID’ |
| **Expected result (post-conditions)** | Processlist sorted by ‘PID’. |

**Test 5.**

|  |  |
| --- | --- |
| **Test Case ID** | processes 0.5 |
| **Test Name** | processes - Sorting ‘Runtime’ |
| **Description** | Sorts list by ‘Runtime’. |
| **Related Use Cases** | 2 |
| **Steps** | 1. Open application 2. Click ‘Processes’ 3. Click ‘Runtime’ |
| **Expected result (post-conditions)** | Processlist sorted by  ‘Runtime’ |

**Test 6.**

|  |  |
| --- | --- |
| **Test Case ID** | processes 0.6 |
| **Test Name** | processes - Sorting ‘SessionID’. |
| **Description** | Sorts list by ‘SessionID’. |
| **Related Use Cases** | 2 |
| **Steps** | 1. Open application 2. Click ‘Processes’ 3. Click ‘SessionID’ |
| **Expected result (post-conditions)** | Processlist sorted by ‘SessionID’. |

**Test 7.**

|  |  |
| --- | --- |
| **Test Case ID** | processes 0.7 |
| **Test Name** | processes - Sorting ‘Memory’ |
| **Description** | Sorts list by ‘Memory’. |
| **Related Use Cases** | 2 |
| **Steps** | 1. Open application 2. Click ‘Processes’ 3. Click ‘Memory’. |
| **Expected result (post-conditions)** | Processlist sorted by ‘Memory’. |

**Unkill Processes**

**Test 1.**

|  |  |
| --- | --- |
| **Test Case ID** | Unkill Processes 0.1 |
| **Test Name** | Unkill Processes - Undo |
| **Description** | Unkills processes |
| **Related Use Cases** | 1, 2, 3 |
| **Steps** | 1. Open application 2. Click ‘Kill’ 3. Click ‘Unkill processes’ 4. Select process 5. Click ‘Undo’ |
| **Expected result (post-conditions)** | Selected processes are killed and a screen pops-up: “[*selected process*] is working again”. |

**Test 2.**

|  |  |
| --- | --- |
| **Test Case ID** | Unkill Processes 0.2 |
| **Test Name** | Unkill Processes - Undo 2 |
| **Description** | Unkills processes |
| **Related Use Cases** | 1, 2, 3 |
| **Steps** | 1. Open application 2. Click ‘Processes’ 3. Select process 4. Click ‘Kill’ 5. Click ‘Unkill processes’ 6. Select process 7. Click ‘Undo’ |
| **Expected result (post-conditions)** | Selected processes unkilled |

**Test 3.**

|  |  |
| --- | --- |
| **Test Case ID** | Unkill Processes 0.3 |
| **Test Name** | Unkill Processes - Delete selected |
| **Description** | Delete killed processes from Unkill processes(which means not unkilling the processes). |
| **Related Use Cases** | 1, 2, 3 |
| **Steps** | 1. Open application 2. Click ‘Processes’ 3. Select process 4. Click ‘Kill’ 5. Click ‘Unkill processes’ 6. Select process 7. Click ‘Delete selected’ |
| **Expected result (post-conditions)** | Selected processes are deleted. |

**Test 4.**

|  |  |
| --- | --- |
| **Test Case ID** | Unkill Processes 0.4 |
| **Test Name** | Unkill Processes - Save own kill-list |
| **Description** | Killed processes in Unkill-list are saved into a list for future usage. |
| **Related Use Cases** | 1, 2, 3, 4 |
| **Steps** | 1. Open application 2. Click ‘Processes’ or ‘Kill’(if ‘Kill’, go to 4.) 3. Select process 4. Click ‘Kill’ 5. Click ‘Unkill processes’ 6. Type name in empty bos 7. Click ‘in database’ |
| **Expected result (post-conditions)** | Processes are saved in a list and a screen pops-up: “Lijst”[*List name]* is toegevoegd aan database”. |