Testplan

Edvin Gibro
Bacilika Glansholm
Viktor Holta
Simon Karlsson
Jessica Kjellin
Max Randow
Erik Simonson
Jakob Söderström

13 april 2024

Version 1.0

Projektidentitet

Hemsida: https://github.com/OSSQA-PUM

Kund: Ola Angelsmark, Advenica AB
E-post: ola.angelsmark@advenica.com

Handledare: Eric Ekström
E-post: eric.ekstrom@liu.se

Kursansvarig: Kristian Sandahl
E-post: kristian.sandahl@liu.se

Projektdeltagare

Namn	**Ansvar**	**E-post**
Edvin Gibro	Teamledare	edvgi966@student.liu.se
Bacilika Glansholm	Arkitekt, Vice teamledare	bacgl188@student.liu.se
Viktor Holta	Testledare	vikho305@student.liu.se
Simon Karlsson	Analysansvarig	simka157@student.liu.se
Jessica Kjellin	Kvalitetssamordnare	jeskj559@student.liu.se
Max Randow	Dokumentansvarig	maxra518@student.liu.se
Erik Simonson	Konfigurationsansvarig	erisi409@student.liu.se
Jakob Söderström	Utvecklingsledare	jakso277@student.liu.se

INNEHÅLL

DOKUMENTHISTORIK

Version	Datum	Utförda ändringar	Utförda av	Granskad
0.1	2024-02-19	Första utkast	VH	MR, JK
0.2	2024-02-22	Skrev fler testfall och flyttade dess sektion till slutet av dokumentet	VH	SK, EG
0.3	2024-03-01	Skrev tabell för testspårbarhet för alla testfall	VH	BG, SK
0.4	2024-03-05	Korrigerade testobjekt och funktionalitet som kommer testas så att de stämmer överens med arkitekturplan, kravspecifikation, och testfall	VH	SK, JK
0.5	2024-03-07	Uppdaterade planerade aktiviteter och uppgifter, som nu beskriver generering av testloggar och skapande av testrapporter mer utförligt	VH	EG, SK
0.6	2024-03-08	Utökning av definitioner, bokstavsordning av definitioner, tillägg av brödtext, tillägg av tabelltexter, och finskrivning	VH	EG, SK
1.0	2024-04-12	Justeringar baserat på kommentarer från oppponeringsgrupp och handledare	VH	SK, ES

1 INTRODUKTION

Detta dokument innehåller information om testning av systemet OSSQA, vilket inkluderar vilka tester som ska utföras, hur de ska utföras, och vad det innebär. Dokumentet följer standarden IEEE Std 829-2008 [1], med anpassningar för att vara enlig med andra dokument och underlätta planeringen av tester.

1.1 Syfte

Syftet med dokumentet är att förse projektgruppen med den information som krävs för att kunna testa OSSQA på ett fullständigt sätt. De genomförda testerna ska säkerställa att kraven specificerade i kravspecifikationen uppfylls [2].

1.2 Definitioner

- **API** - Förkortning av *application programming interface*. Ett gränssnitt för att möjliggöra interaktion mellan två separata datorprogram eller bibliotek.
- **Black-box-testning** - En typ av testning där den interna strukturen av koden inte känns till.
- **CLI** - Förkortning av *command-line interface*. Ett gränssnitt för att interagera med datorprogram via en kommandotolk.
- **Frontend** - Det visuella gränssnittet som användaren använder för att interagera med systemet.
- **GitHub** - En webbplattform för att lagra och hantera repos.
- **GUI** - Förkortning av *graphical user interface*. Ett gränssnitt för att interagera med datorprogram via grafiska element.
- **Pytest** - Ett ramverk för att testa Python-kod.
- **Python** - Ett dynamiskt typat programmeringsspråk för utveckling av mjukvara.
- **Repo** - Från engelskans *repository*. Ett Git-repo är en katalog eller lagringsenhet där alla filer, historik, och konfiguration för ett projekt sparas.
- **REST** - Förkortning av *representational state transfer*. En mängd arkitektoniska begränsningar som innefattar, bland annat, tillståndslös kommunikation mellan en server och en klient.
- **SBOM** - Förkortning av *Software Bill of Materials*. En innehållsdeklaration för en mjukvara som definierar samtliga programbibliotek som används.
- **SUS** - Förkortning av *System Usability Score*. En etablerad enkät för att mäta användbarhet av ett system.

2 TESTSPECIFIKATION

Detta avsnitt beskriver vilka tester som ska utföras, hur de ska utföras, och vad de testar.

2.1 Systemöversikt

OSSQA består av sex moduler, som ska testas för att säkerställa att systemet fungerar korrekt. Två av dessa moduler agerar som gränssnitt mellan de resterande modulerna. För mer information se arkitektplanen [3].

Följande lista innehåller modulerna som agerar som gränssnitt:

- *Frontend*-gränssnitt
- *REST-API*

Följande lista innehåller de resterande modulerna:

- *CLI*
- *GUI*
- Analysator
- Databas

2.2 Tillvägagångssätt

Tre nivåer av tester ska utföras för att testa systemet: enhetstester, integrationstester, och systemtester. Samt ska flera metriker samlas in när dessa tester utförs.

Alla tester ska utgå från black-box-testning, vilket är en metod där testerna endast tar hänsyn till indatan och utdatan av funktionerna de testar. Alltså anses testerna godkända om utdatan av funktionerna matchar den förväntade utdatan.

Enhets- och integrationstesterna ska använda sig av Pytest, vilket är ett ramverk för att testa Python-kod [4]. Tester i Pytest är skrivna som vanliga *Python*-funktioner, vilket gör det väldigt enkelt att implementera.

Alla tester ska automatiseras via *GitHub* Actions, vilket är ett verktyg för att automatiskt exekvera kommandon i ett GitHub-*repo*.

Fun-coverage är ett verktyg för att beräkna funktionstäckning av ett Python-program [5].

2.2.1 *Enhetstester*

Enhetstester säkerställer att de fundamentala funktionerna i systemet fungerar korrekt. Detta syftar på funktioner som inte kallar på några andra funktioner, med undantag för funktion som tillhandahålls av diverse bibliotek.

Följande tabell definierar testfall för enhetstester, samt deras kopplingar till kraven i kravspecifikationen [2]:

Tabell 1: Testfall för enhetstester

ID	Beskrivning	Indata	Förväntad utdata	Beroenden	Krav	Modul
TF1	Analys av ett enstaka beroende	Länk till GitHub-repo	Bedömning utifrån OpenSSF Scorecards områden [6]	Inga	4	Analysator
TF2	Sammanställning av övergripande bedömning	Lista av beroendebedömningar	Bedömning framställd från listan	Inga	6	Analysator
TF3	Parsning av en *SBOM*	Sökväg till SBOM	Lista med beroendena som specificerades i SBOM:en	Inga	2	Analysator
TF4	Skapande av bedömning	Repoversion och dess bedömning	En bedömning skapas i databasen	Inga	7	Databas
TF5	Hämtning av bedömning	ID av bedömning	Den sparade bedömningen	TF4	8	Databas
TF6	Parsning av argument	En sträng med argument	Ett objekt med argument av rätt datatyp	Inga	1, 3	CLI

2.2.2 *Integrationstester*

Integrationstester säkerställer att sammankopplingen av de fundamentala funktionerna fungerar korrekt. Detta syftar på funktioner som kallar på de fundamentala funktionerna.

Följande tabell definierar testfall för integrationstester, samt deras kopplingar till kraven i kravspecifikationen [2]:

Tabell 2: Testfall för integrationstester

ID	Beskrivning	Indata	Förväntad utdata	Beroenden	Krav
TF7	Analys av en SBOM	Sökväg till SBOM	Bedömning framställd från SBOM:en och bedömning sparad i databasen	Inga	1, 2, 3, 4, 6, 7, 9
TF8	Hämtning av översikt av föregående bedömningar	Ingen	En lista av sparade bedömningar	TF7	5
TF9	Hämtning av specifik bedömning	ID av bedömningen	Den sparade bedömningen	TF7, TF8	5

2.2.3 *Systemtester*

Systemtester säkerställer att hela systemet fungerar korrekt. Detta syftar på att systemet behandlas som en enhet, med alla sina fundamentala funktioner sammankopplade.

Följande tabell definierar testfall för systemtester, samt deras kopplingar till kraven i kravspecifikationen [2]:

Tabell 3: Testfall för systemtester

ID	Beskrivning	Indata	Förväntad utdata	Beroenden	Krav
TF10	Automatiserad analys via GitHub Actions	Sökväg till SBOM	Bedömning framställd från SBOM:en	Inga	1, 2, 3, 4, 6, 7, 9, 14

2.2.4 *Metriker*

Följande tabell definierar metriker som ska samlas in, samt deras koppling till kraven i kravspecifikationen [2]:

Tabell 4: Metriker

Beskrivning	Insamlingsmetod	Godkännandekriterium	Krav
Andel godkända testkörningar	Ska beräknas från flera körningar av testfall TF10 i tabell 3	$\geq 90\%$	16
Andel funktionstäckning	Python-paketet fun-coverage, som räknar på vilka funktioner som har använts, ska köras via GitHub Actions	$\geq 60\%$	17
Antal poäng enligt *SUS* [7]	En enkät skickas till kunden som de sedan besvarar	≥ 75 poäng	18
Exekveringstid för analys av en SBOM med 200 beroenden	Tiden ska beräknas i samband med testfall TF7 i tabell 2	≤ 5 timmar	19

2.3 Leveranser

Under systemets utveckling ska följande dokument framställas och uppdateras:

- Testplanen, som beskrivs i avsnitt 1.
- Testloggarna, som innehåller information om varje test som utförs, vilket inkluderar ID av berörda testfall, datum och tid, exekveringstid, filnamn av testmodul, resultat, och eventuella felmeddelanden.
- Testrapporterna, som innehåller information om testloggarna, metriker som har samlats in, avvikelser från testplanen, åtgärder, och bedömning av systemets status.

3 TESTFÖRVALTNING

Detta avsnitt beskriver vad som ska ske under en testperiod och definierar riktlinjer för hur teamet ska utföra testning.

3.1 Veckovis testning

Testning ska ske veckovis. Detta betyder att alla övergripande beslut om testning ska utgå från den föregående veckans testrapport. Samt betyder det att alla testloggar som genereras ska kompileras till en veckolig testrapport.

I början av veckan ska den föregående veckans testrapport granskas för att ta beslut om vad som behöver göras angående testning. Under veckans gång ska testloggar genereras automatiskt via GitHub Actions. I slutet av veckan ska de genererade testloggarna kompileras till en testrapport.

3.2 Testprocess

Detta delavsnitt beskriver när och hur teamet ska utföra uppgifter angående testning.

3.2.1 *Skrivning av tester*

Nya tester ska skrivas när nya funktioner implementeras och gamla tester ska uppdateras när indatan eller utdatan av gamla funktioner ändras, eftersom testerna utgår från black-box-testning som beskrivet i avsnitt 2.2.

3.2.2 *Exekvering av tester*

Tester ska exekveras regelbundet under utveckling av systemet, för att eventuella problem ska upptäckas tidigt.

3.2.3 *Hantering av misslyckade tester*

Tester som misslyckas ska granskas och hanteras så tidigt som möjligt, för att undvika eventuella konflikter av fortsatt utveckling.

3.3 Inlärning

Teamet ska gå på en workshop för att lära sig om testprocessen och testperioden. Denna workshop är preliminär och kan komma att ändras.

REFERENSER

[1] IEEE Computer Society, “Ieee standard for software and system test documentation,” https://ieeexplore.ieee.org/document/4578383, 2008.

[2] PUM14, “Kravspecifikation,” https://github.com/OSSQA-PUM/OSSQA/blob/main/documents/TDDD96_Kravspecifikation.pdf, 2024.

[3] ——, “Arkitektplan,” https://github.com/OSSQA-PUM/OSSQA/blob/main/documents/TDDD96_Arkitektplan.pdf, 2024.

[4] ”pytest: helps you write better programs.” Pytest.org. Hämtad: 2024-03-07. [Online]. Tillgänglig: https://docs.pytest.org/en/8.0.x/.

[5] ”fun-coverage 0.2.0.” Python Software Foundation. Hämtad: 2024-04-12. [Online]. Tillgänglig: https://pypi.org/project/fun-coverage/.

[6] ”Build better security habits, one test at a time.” Open Source Security Foundation. Hämtad: 2024-04-11. [Online]. Tillgänglig: https://securityscorecards.dev/.

[7] J. R. Lewis, “The system usability scale: past, present, and future,” *International Journal of Human–Computer Interaction*, vol. 34, no. 7, pp. 577–590, 2018.