**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

****

**INFORMATIKOS FAKULTETAS**

**PROGRAMŲ SISTEMŲ TESTAVIMAS**

**Studentas: Lukas Gužauskas**

**Dėstytojas: Dominykas Barisas**

**KAUNAS 2018-2019**

# Įvadas

Tai dokumentas apibūdinantis programinės įrangos testo planą pagal Simple Merge sistemą. Visa programinės įrangos testavimo strategija sudaryta iš šių testavimų būdų ir atliekama tvarka:

* Unit Testing – visi programinės įrangos komponentai yra išbandyti.
* Static Testing – taip vadinama paleisti testavimą. Tai yra programinės įrangos testavimo forma, kurioje faktinė programa ar programa nėra naudojama. Vietoj to, šis bandymo metodas reikalauja, kad programuotojai rankiniu būdu nuskaitytų savo kodą, kad surastų bet kokias klaidas. Statinis testavimas yra "White Box" testavimo etapas.
* Saugumo testavimas tikrina sistemų ir jų duomenų konfidencialumą, vientisumą ir prieinamumą. Neigiamas testavimas.
* GUI testavimas (UI testavimas). Programinės įrangos GUI testavimas siekiant užtikrinti funkcionalumą, kada vartotojas paleidžia ar automatinis paleidimas.

# Testavimo apimtis

GUI testavimas sudaryta iš:

* Simple Merge sistema
* Panaudojimo atvejų modeliai yra įkelti, redaguoti, išsaugoti, palyginti, sujungti, kopijuoti į kairę ir kopijuoti į dešinę.

Šio testavimo tikslas yra nustatyti, ar gerai įdiegta programinė įranga ir atitinka funkcinius reikalavimus. Testavimo metu surenkami testavimo duomenys ir rezultatai, kurie reikalingi programinės įrangos priežiūros etape.

# Testavimo strategija

Unit ir saugumo testavimo darbai turi būti užbaigti prieš pradedant GUI testavimo etapą. GUI testavimas nukreiptas į programinės įrangos testavimą vartotojo perspektyvai, siekiant užtikrinti, kaip ji gali naudoti testavimo scenarijų ir atitinka nustatytus kokybės reikalavimus.

Papildoma testavimo veikla bus įtraukta: Unit Testing, GUI Testing, Security Testing, Static testing.

# Pradinės sąlygos

Šios užduotys turi būti atliktos prieš pradedant testavimo veiksmus:

* Turi programos įrangos specifikaciją, naudojimo panaudojimo atvejų modelius.
* Programos įrangos įdiegimas.
* Nustatyti tyrimo metu nustatytų problemų nustatymo procedūrą.
* Nustatyti naudojimo atvejų rinkinį priimant bandymus, siekiant išbandyti visas programinės įrangos funkcijas.
* Nustatyti testavimų aplinką.
* Paskirti testavimo išteklius.
* Apibrėžti testavimo priėmimo standartus.

# Testavimo prioritetai

Toliau testavimo veiksmai yra išvardyti aukščiausią prioritetą:

* Funkcija – ar visos apibrėžtos programinės įrangos funkcijos veikia kaip tikėtasi?
* Naudojimas – ar tai programinė įranga yra naudotojui „draugiškai“?
* Veikla – ar programinė įranga atitinka suderintus atlikimo kriterijus?

# Testavimo technikos

Naudojami šie testavimo technikai:

* Testavimo scenarijus – scenarijus panaudojimo atvejis.
* Įvestas duomens failas – testuotojas patikrins įvesti duomens failą testavimo metu.
* Naudojamas checklist – veiksmas skirta įvertinti sistemų naudojimo paprastumą.
* Sprendimų teksto laukas – naudoj amos kiekvieno panaudoto įvesto rezultatų nustatymui.

# Rolė ir atsakomybė

* Kokybės užtikrinimas – žmogus atsakomybė dėl testavimo procesų planavimą ir jo vykdymą.
* Testuotojas – atlieka testavimo veiksmus ir apibrėžta testavimo planą.
* Testavimo ištekliai – užtikrinti tinkamą testavimų išteklių planavimą ir valdymą.
* Ataskaita apie pažangos testavimą.

# Rezultatai

Po testavimo turi būti pateikti rezultatus:

* Testavimo planas – šis dokumentas su visais pakeista atliktais testavimo proceso metu.
* Pakeitimo užklausos – dokumentas, apibūdinantis programinės įrangos pakeitimus, atsiradusius dėl pakeistų reikalavimų arba nustatytų defektų testavimo metu.
* Galutinis dokumento patvirtinimas, kad sistema atitinka visus funkcinius ir kokybės reikalavimus.

# Testavimo aplinka

Programinės įrangos testavimo metu, programinės ir techninės įrangos konfigūracija turi būti:

Pagrindinis kompiuteris su tokia konfigūracija:

* AMD Radeon Graphics 2.7 GHz, 8GB RAM, 512GB SSD,
* Microsoft Windows 10 Home
* Mozilla Firefox 62.0.2

GUI (Vartotojo sąsaja) testing įranga su tokia konfigūracija:

* Eclipse JAVA IDE
* Java Swing GUI with AssertJ Swing biblioteka arba Jemmy biblioteka
* Automatinis ar rankinis testavimas

Security testing įranga su tokia konfigūracija:

* Eclipse JAVA IDE
* Vega tools

# Testavimo scenarijai

Šie testavimo scenarijai yra skirti patikrinti kiekvieną programinės įrangos funkciją. Scenarijų dalis surinkta panaudojimo atvejų scenarijus. Šios dalys nėra įtrauktos į šį dokumentą, o tos dalys yra susijusios.

Kiekvienas testavimo scenarijai susidaryta:

1. Apibūdinimas – šioje dalyje surinktas apibūdinimas apie naudojimo scenarijus.
2. Pradiniai duomenys – paprastai tai pradinis duomenų failas.
3. Testavimo žingsnis – veiksmai, kuriuos bandytojas turi atlikti testavimo metu.
4. Testavimo atvejai – Testavimo metu įvesti duomenys ir vykdomos programinės įrangos reakcija.

Test script 01.01 Load

Apibūdinimas: Testavimas patikrinti duomenų failo įkrovimą į GUI langą iš darbalaukio.

Pradiniai duomenys: Pasiimti duomenų failą iš darbalaukio.

Testavimo žingsniai:

1. Spausti „Load“ stačiakampio mygtuką.
2. Pasirodo langas, pasirinkti teksto failą.
3. Spausti „Ok“.
4. Tekstiniame lauke rodomas tekstinis failo aprašymas.

Testavimo atvejai:

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Expected result | Comments | |
| Enter valid text files. | Successful load | |  |
| Enter invalid text files which isn’t supported file format. | The system reports error message | |  |
| Enter valid text files which is a blank text file. | Successful load | |  |

Test script 01.02 Edit

Apibūdinimas: Testavimas išbandyti sistemos funkciją, kuri gali atrakinti teksto.

Pradiniai duomenys: Modifikuotas duomenų failas.

Testavimo žingsniai:

1. Spausti „Edit“ stačiakampį mygtuką.
2. Sistema atrakinta teksto lauką.
3. Modifikuoti keletą simbolių ar eilutę.
4. Dar kartą spausti „Edit“ ir sistema vėl užrakinta.

Testavimo atvejai:

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Expected result | | Comments |
| After modified text area, click Edit button | Successful edit |  | |

Test script 01.03 Save

Apibūdinimas: Testavimas patikrinti duomenų failo išsaugojimą į darbalaukį.

Pradiniai duomenys: Duomenys išsaugoti į darbalaukį.

Testavimo žingsniai:

1. Spausti „Save“ stačiakampį mygtuką.
2. Pasirodo langelis, kuris nori išsaugoti failą.
3. Spausti „Ok“.
4. Tekstinis failas jau išsaugota.

Testavimo atvejai:

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Expected result | | Comments |
| After uploaded or modified the file and click Save as button | Successful save |  | |
| Enter an invalid name of the file | Show up a pop-up which is reporting an input error |  | |
| Save as the file name is the same as the original file | Successful rewrite the previous file |  | |
| Not upload the file into the panel and attempt clicking button | Show up a pop-up which is reporting an input error |  | |

Test script 01.04 Compare

Apibūdinimas: Testavimas išbandyti palyginimo tarp 2 tekstinių failų funkciją.

Pradiniai duomenys: 2 duomenų failai.

Testavimo žingsniai:

1. Spausti „Compare“ stačiakampį mygtuką.
2. Sistema vykdo palyginimą tarp 2 tekstinių failų.
3. Pasirodo 2 teksto laukai.
4. Jei yra skirtumas tarp 2 tekstinių failų, tad nurodoma žyma: žalia ar raudona.

Testavimo atvejai:

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Expected result | | Comments |
| 2 different of panels between left and right | Successful compare and display highlighted with color |  | |
| Click Compare button without uploaded 2 files | None information is displayed |  | |
| The left panel is a blank text, other panel has some lines | The result will make up non-highlighted text on the right panel |  | |

Test script 01.05 Copy to Left

Apibūdinimas: Testavimas patikrinta dešiniame lauke tekstą, kuris turi keletą eilučių ir pasirinkta eilutė bus perkeliama į kairįjį lauką.

Pradiniai duomenys: 2 duomenų failai.

Testavimo žingsniai:

1. Spausti „Copy to Left“ stačiakampį mygtuką.
2. Teksto žymėjimas pereina prie kitos eilutės kairiajame tekstiniame lauke.
3. Pasirodo kairiajame teksto lauke.

Testavimo atvejai:

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Expected result | | Comments |
| Select the lines that are different, then clicking button | Successful copy to left which will show up in the left panel |  | |
| The panel on the right contents blank text | To delete lines in the left panel |  | |

Test script 01.06 Copy to Right

Apibūdinimas: Testavimas patikrinta kairinėje teksto lauke, kuris turi keletą eilutes ir pasirinkta eilutę bus dešiniajame skydelyje.

Pradiniai duomenys: 2 duomens failai.

Testavimo žingsnis:

1. Spausti „Copy to Right“ stačiakampio mygtuką.
2. Sistema įgyvendinti, kad žymėjimas pereina į kitos eilutės dešinę teksto lauką.
3. Pasirodyti dešiniajame teksto lauke.

Testavimo atvejai:

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Expected result | | Comments |
| Select the lines that are different, then clicking button | Successful copy to left which will show up in the left panel |  | |
| The panel on the right contents blank text | To delete lines in the left panel |  | |

# Testavimo valdymas

Testavimas turi būti valdomas, kad būtų laikomasi nustatytų taisyklių, kurios apima:

1. Testavimo vykdymas.
2. Testavimo rezultatų pranešimas.
3. Defektų stebėjimas ir tvirtinimas.
4. Testavimo aplinkos konfigūravimo valdymą.

# Testavimo tvarkaraščiai

|  |  |  |
| --- | --- | --- |
| Testing task | Start | Deadline |
| Unit Testing |  |  |
| Static Testing |  |  |
| GUI Testing |  |  |
| Security Testing |  |  |

Susidaryta iš šių keturių testavimo būtų detaliau:

Unit Testing - visi Simple Merge sistema yra išbandyti patikrintų komponentus. Kad programinė įranga pradėta daryti su pirmu testavimu, tai visada būtų Unit Testing įrankis.

Static Testing – tai programinės įrangos testavimo metodika, kurioje programinė įranga yra išbandyta be kodo vykdymo. Jame yra 2 padalytos:

* Peržiūra – paprastai naudojama rasti ir pašalinti klaidas ar dviprasmybes dokumentuose, pvz., reikalavimų, testavimo atvejų ir pan.
* Statinė analizė – parašyti kodas yra analizuojamas struktūriniams defektams, dėl kurių gali atsirasti defektų.
* Statinės analizės įrankis:
  + Kintamasis su neapibrėžta verte.
  + Nesuderinama sąsaja tarp modulių ir komponentų.
  + Kintamieji, kurie yra paskelbti, bet niekada nenaudojami.
  + Nepasiekiamas kodas.
  + Programavimo standartų ir apsaugos pažeidimai.

GUI (UI/UX) Testing – ekranų tikrinimas – meniu, mygtukai ir visi tipai – įrankių juosta, dialogo langeliai. Pas mūsų sistema yra naudojama dialogo langeliai.

* Pateikta checklist užtikrins išsamų GUI testavimą.
  + Patikrinkite visus GUI elementus (dydis, padėtis, plotis).
  + Patikrinkite ar galite atlikti numatytą programos funkciją naudojant GUI.
  + Patikrinkite ar nėra atskirų skirsnių ekranuose.
  + Patikrinkite ar naudojamas šriftas ar tekstas yra suprantamas.
  + Patikrinkite ar vaizdai yra gerai aiškus.
* GUI testavimo metodas:
  + Rankinis testavimas
  + Įrašykite ir pakartokite
  + Modeliuojamas testavimas

Saugumo testavimas - tai programinės įrangos testavimo variantas, kuris yra aparatinės įrangos, programinės įrangos, aplinkos informacijos saugumo užtikrinimo procesas. Rezultatai padeda informacijos saugumo duomens failams lyginti informacinių sistemų apsaugą nuo tokių grėsmių, netikro failo formato ir pažeidžiamumo. Taip pat yra apie visas galimas sistemos, galinčias sukelti informacijos praradimą duomens failams.

Testavimo planas turėtų apimti:

* Su saugumu susiję testavimo atvejai ar scenarijai
* Testavimo duomenys, susiję su saugumo tyrimais
* Testavimo įrankiai, reikalingi saugumo tyrimams
* Analizuojant įvairius testavimų rezultatus iš įvairių saugumo priemonių

# Testavimo rizikos

Galima rizika, kuri gali turėti įtakos testavimo metu:

|  |  |  |
| --- | --- | --- |
| Risk | Description | Mitigation |
| Vartotojas nespėja dalyvauti testavimo procese. | Vartotojas visada daug dirbantis savo reikalais ir neturi laiko atsakyti į klausimus testuotojui ar apžvelgti programinę įrangą. | Užtikrinkite, kad mūsų konsultantams per nurodytą laiką būtų skiriamas vienas vartotojas kartą per savaitę 5 valandoms. |
| Nepriklausomos testavimo aplinkos nebuvimas ir neprieinamumas | Sistema kažkada gali nepažinti kokios nors programinės įrangos testavimo aplinkos. Tuomet gali įvykti klaida. | Atsižvelgiant į tai, kad nėra aplinkos, tvarkaraštis atnaujinamas ir testavimo vykdymas bus atidėtas. |
| Taikymo sritis nėra visiškai apibrėžta | Testavimo planas turi visiškai apibrėžtas funkcijas, testavimo metu testuotojas gali panaudoti savo idėjas, kurias vėliau reikia panaikinti. | Taikymo sritis nėra aiškiai apibrėžta, tačiau funkcijų pakeitimai dar nėra galutinai parengti arba nuolat keičiasi. |