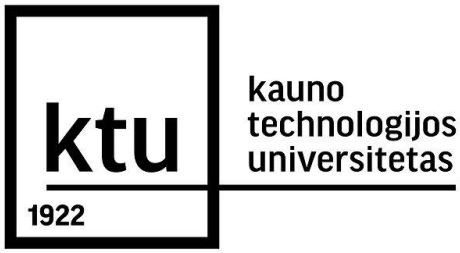


INFORMATIKOS FAKULTETAS

T120B162 Programų sistemų testavimas

1 laboratorinis darbas

Studentas: Mangirdas Kazlauskas, IFF-4/1

Dėstytojas: doc. Šarūnas Packevičius

KAUNAS 2017

Turinys

1. Įvadas

Šiame skyriuje pateikiamas įvadas į laboratorinį darbą bei supažindinama su testuojama sistema.

1.1. Laboratorinio darbo tikslai ir uždaviniai

1) Išsiaiškinti testavimo plano struktūrą;
2) Išsiaiškinti, kokie punktai turi būti išpildyti bei kokie punktai yra svarbiausi, norint sudaryti tinkamą programinės įrangos testavimo planą;
3) Sudaryti testavimo planą norimai ištestuoti sistemai;
4) Paruošti laboratorinio darbo ataskaitą.

1.2. Testuojamos sistemos aprašymas

Sistema „Scatterify“ - elektroninei komercijai skirta SaaS (angl. Software as a service) platforma, kuri leistų verslininkams pardavinėti savo produkciją naudojantis Facebook Messenger aplikacija.

Veikimo principas – pačią kuriamą platformą sudaro keturios dalys: prezentacinis platformos puslapis, internetinė aplikacija, kuria naudosis verslininkai bei tam tikros prekybos bei paslaugų sferos darbuotojai (pvz., kasininkai, padavėjai), Facebook Messenger susirašinėjimo robotas (angl. chatbot) bei aplikacijų programavimo sąsaja (angl. trump. API).

Įmonė, norėdama naudotis šia verslas-verslui (angl. business to business, B2B) skirta platforma, prisijungia prie internetinės aplikacijos, kurioje nurodo savo organizacijos Facebook paskyrą (oficialų Facebook puslapį), priregistruoja organizacijai priklausančias prekybos vietas, sukonfigūruoja Messenger susirašinėjimo roboto parametrus bei, atlikę apmokėjimą, gali iš karto naudotis sistema, kuri bus pasiekiama visiems klientams, besinaudojantiems Facebook Messenger aplikacija. Internetinėje aplikacijoje prekybos vietos realiu laiku mato gautus konkrečius užsakymus, jų statusą. Taip pat aplikacija suteikia galimybę sekti pardavimo vietų ar visos organizacijos bendrą statistiką realiu laiku.
Sistemos funkciniai reikalavimai:
Neregistruotas sistemos naudotojas gali:

1) Peržiūrėti platformos prezentacinį puslapį;
2) Prisijungti prie internetinės aplikacijos naudodamasis Facebook paskyra.

Registruotas sistemos vartotojas gali:

1) Atsijungti nuo internetinės aplikacijos;
2) Prijungti (užregistruoti) organizacijos Facebook paskyrą prie platformos;
3) Keisti organizacijos informaciją:
    a. Redaguoti bendrą organizacijos informaciją;
    b. Pridėti prekybos vietą;
    c. Redaguoti prekybos vietą;
    d. Pašalinti prekybos vietą;

4) Keisti susirašinėjimo roboto nustatymus;
5) Peržiūrėti gautus prekybos vietos užsakymus;
6) Peržiūrėti gautą užsakymą sudarančias prekes;
7) Keisti gauto užsakymo statusą;
8) Keisti gautą užsakymą sudarančios prekės statusą;
9) Peržiūrėti užsakymą pateikusio kliento informaciją;
10) Peržiūrėti organizacijos statistiką įvairiais pjūviais;
11) Peržiūrėti prekybos vietos statistiką įvairiais pjūviais.

1.3. Testuojamos sistemos architektūra

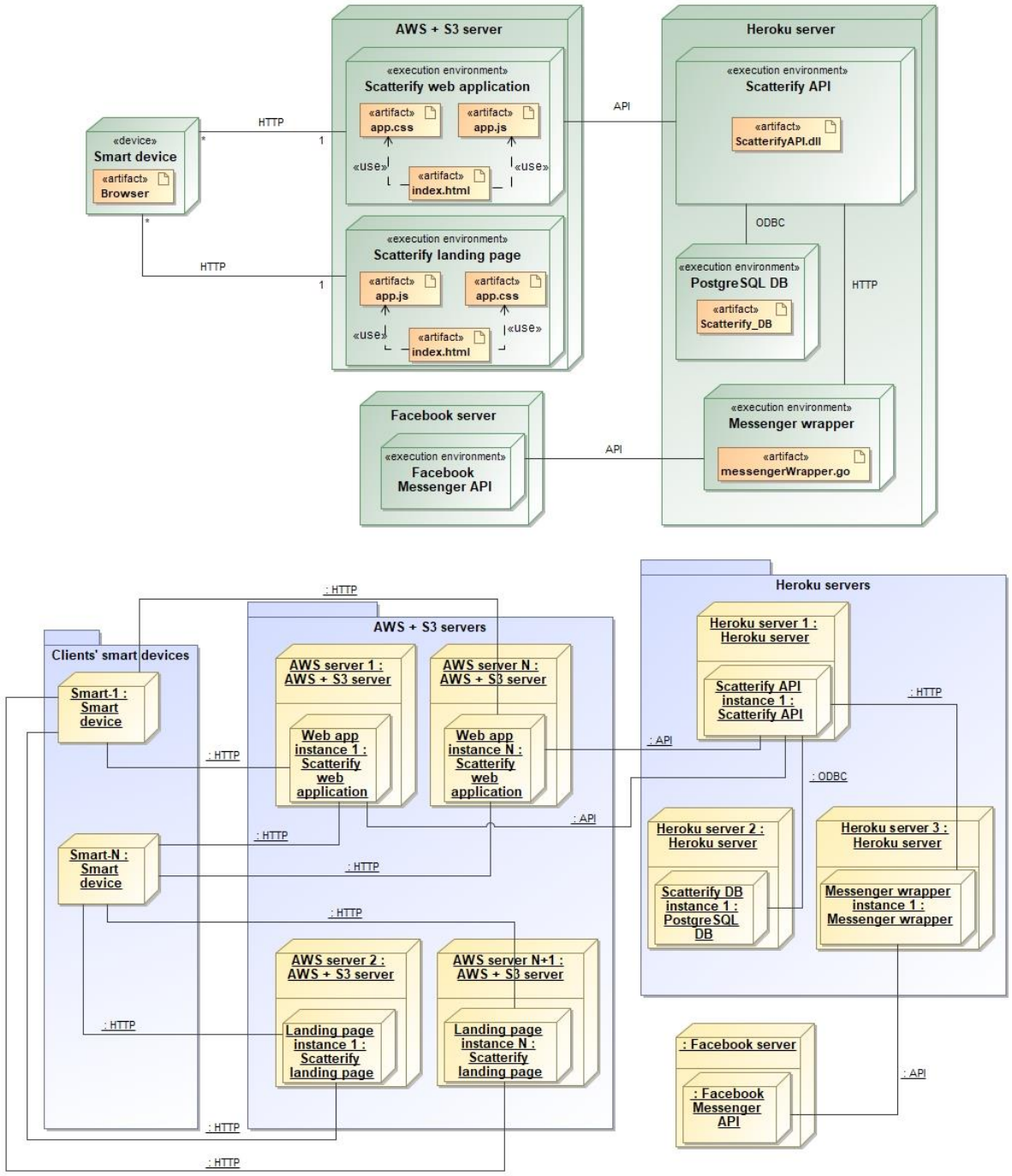
Kliento pusė (ang. Front-End) – sukurta naudojant Vue.js + Vuex + Vue-router + Vuetify + Webpack + Socket.io + axios;

Serverio pusė (angl. Back-End) – sukurta naudojant ASP.NET Core 2, testavimui - xUnit bei kitas bibliotekas, padedančias sistemos API kūrimo procesui (pvz., AutoMapper, Mock ir t.t.);

Pagrindinis servisas – duomenų bazės valdymas, autentifikacija;

Pagalbinis servisas – duomenų gavimui iš Facebook Messenger API, reikiamos užklausos pagrindiniam servisui formatavimui, susirašinėjimo roboto logikos realizavimui.

1 pav. pavaizduota kuriamos sistemos diegimo diagrama. Iš diagramos matosi, kad sistemos failų talpinimui yra naudojami Amazon web services (AWS) serveriai bei Heroku serveriai. Kiekviena sistemos dalis yra diegiama atskirame serveryje. Internetinė aplikacija bei prezentacinis puslapis klientams yra pasiekiama per HTTP protokolą. Kadangi prezentacinis puslapis yra statinis, jis neturi jokios sąsajos su Scatterify API, tačiau internetinės aplikacijos veikimui (pvz., duomenų manipuliavimui su duomenų baze) Scatterify API yra reikalingas ir yra pasiekiamas per aplikacijų programavimo sąsają. Pats Scatterify API vykdo duomenų mainus su duomenų baze- tam naudojama ODBC sąsaja. Viena iš platformos dalių – Facebook Messenger susirašinėjimo roboto veikimas organizacijos Facebook puslapyje. Scatterify API į Facebook Messenger API tiesiogiai užklausų nesiunčia – tai yra daroma bendraujant su Messenger wrapper HTTP protokolu, o Messenger wrapper su Facebook API bendrauja per aplikacijų programavimo sąsają.



1 pav. Sistemos "Scatterify" diegimo diagrama

2. Sistemos testavimo tikslai

1) Užtikrinti, kad sistema veikia taip, kaip nurodyta specifikacijoje, t.y., atitinka tiek funkcinius, tiek nefunkcinius reikalavimus, išpildo visus kokybės užtikrinimo aspektus bei tenkina visus numatytus panaudojimo atvejus taip, kad klientas galėtų naudotis sistema be atsirandančių klaidų ar sutrikimų;
2) Testavimo metu ne tik užtikrinti tai, kad sistema veiktų taip, kaip buvo numatyta specifikacijoje, bet ir identifikuoti esamas naujas ar galimai sistemos plėtros laikotarpiu

atsirasiančias klaidas, apie jas pranešti savo komandai bei jas stengtis ištaisyti iki tol, kol sistema bus pradėta naudotis, ar bent jau numatyti galimas sistemos korekcijas plėtros laikotarpiu, kad surastos naujos klaidos būtų ištaisytos arba jų būtų išvengta.

3. Testavimo prioritetai

Šiame skyriuje pateikiama sistemos testavimo prioritetai bei apibrėžiamas klaidų klasifikavimas pagal jų prioritetus bei svarbą.

3.1. Sistemos testavimo prioritetai

Sistemos testavimo prioritetai yra tokie (nuo svarbiausio iki mažiausiai svarbaus):

1) Pagrindinis sistemos funkcionalumas - funkcijos, kurios yra būtinos, norint išpildyti pagrindinį sistemos veikimo scenarijų;
2) Sistemos funkcionalumas, nesusijęs su pagrindinėmis sistemos funkcijomis ir pagrindiniu sistemos veikimo scenarijumi (pvz., vartotojo prisijungimo duomenų koregavimas, sistemos statistikos funkcionalumas ir pan.);
3) Sistemos naudotojo sąsaja (UI) – tvarkingas bei vientisas sistemos dizainas, sistemos naudojimosi patogumas/paprastumas, patrauklus sistemos grafinių elementų naudojimas;
4) Sistemos greitaveika – kaip ilgai sistema užtrunka apdorodama duomenis, užkraudama sistemos langus, siųsdama ir apdorodama asinchronines užklausas;
5) Saugumas – ar prisijungimas bei naudojimasis sistema yra saugus, ar trečiųjų šalių asmenis gali pasiekti privačius duomenis, ar sistema yra apsaugota nuo kibernetinių atakų (tiek sistemos duomenų vagysčių, tiek bandymo nutraukti sklandų sistemos veikimą).

3.2. Sistemos klaidų prioritetai ir svarba (severity)

Pačios sistemos testavimo prioritetai ir klaidų, atsirandančių sistemos kūrimo/naudojimosi metu prioritetai skirstomi kiek skirtingai. Programinės įrangos testavimo procese sistemos klaidos turi du pagrindinius vertinimo kriterijus:

1) Sistemos klaidos prioritetas – tai dydis, kuris parodo, kokia tvarka turi būti ištaisomos atsiradusios sistemos klaidos (ar klaida turi būti ištaisyta dabar, ar gali kiek palaukti). Prioritetas yra nustatomas remiantis kliento reikalavimais. Klaidos prioritetų kategorijos yra pateiktos 2 lentelėje.

1 lentelė. Sistemos klaidų prioritetų klasifikacija

Klaidos prioritetas	Klaidos prioriteto reikšmė	Klaidos prioriteto apibūdinimas
1	Aukštas (High)	Klaida yra pastebima, tačiau jos ištaisymas gali palaukti (jei yra klaidų su aukštesniu prioritetu).

2	Vidutinis (Medium)	Klaida turi būti pataisyta įprasto plėtros procesu metu. Klaidos ištaisymas gali palaukti iki kitos programinės įrangos sukūrimo versijos.
3	Žemas (Low)	Klaida turi būti ištaisyta kaip tik galima greičiau, nes ji daro didelę įtaką sistemos veikimui – sistema naudotis negalima tol, kol ji nebus ištaisyta.

2) Sistemos klaidos svarba – tai dydis, kuris parodo, kaip stipriai klaida gali paveikti sistemą ir jos veikimą.

2 lentelė. Sistemos klaidų svarbos klasifikacija

Klaidos svarba	Klaidos svarbos reikšmė	Klaidos svarbos apibūdinimas
S1	Critical	Klaida nutraukia visos sistemos arba kai kurių jos dalių darbą ir/ar stipriai paveikia bei sugadina sistemos duomenis. Tam, kad pasiektume norimą rezultatą, sistemoje nėra jokių alternatyvių funkcijų.
S2	Major	Klaida nutraukia visos sistemos arba kai kurių jos dalių darbą ir/ar stipriai paveikia bei sugadina sistemos duomenis, tačiau norimam rezultatui pasiekti sistemoje yra kitų būdų.
S3	Moderate	Klaida, kuri nenutraukia sistemos ar jos dalių darbo, tačiau dėl jos sistema pateikia klaidingus rezultatus.
S4	Minor	Klaida, kuri nenutraukia sistemos darbo ar kitaip neigiamai nepaveikia naudojimosi sistema ir norimi rezultatai gali būti pasiekiami kitais būdais.
S5	Cosmetic	Klaidos, kurio nedaro įtakos sistemos veikimui ar naudojimuisi sistema bei yra susijusios su sistemos išvaizda.

4. Rolės ir atsakomybės

Šiame skyriuje pateikiami testavimo procese dalyvaujantys asmenys bei jų rolės/pareigos.

4.1. Programuotojas (developer)

1) Sistemos kūrimas (funkcijų realizacimas);
2) Sistemos panaudos atvejų, nefunkcinių reikalavimų išpildymas/realizavimas;
3) Įvairių tipų (vienetų, integracinių ir t.t.) testų realizavimas sistemoje;
4) Priėmimo testų palaikymas.

4.2. Kokybės užtikrinimo inžinierius (QA lead – quality assurance lead)

1) Įvairių tipų testų kūrimas, sudarymas, identifikavimas bei apžvalga (pvz., testavimo scenarijų identifikavimas/sudarymas, jų vykdymas/priskyrimas testuotojams, testavimo rezultatų apžvalga);
2) Sistemos panaudos atvejų sudarymas, koregavimas, apžvalga;
3) Sistemos palaikymo koordinavimas (naudotojų grįžtamojo ryšio apžvalga, jų pastebėtų klaidų/galimų patobulinimų struktūrizavimas bei sistemos veikimo ataskaitų sudarymas).

4.3. Testuotojas (tester)

1) Testavimo scenarijų vykdymas remiantis testų scenarijais bei testavimo planu;
2) Naujų klaidų sistemoje identifikavimas;
3) Naudotojo vadovo sudarymas.

4.4. Projekto vadovas (PM – project manager)

1) Skirtingų rolių darbuotojų (programuotojų, kokybės užtikrinimo inžinierių, testuotojų) darbų koordinavimas, remiantis sudarytu testavimo grafiku;
2) Darbų užbaigtumo užtikrinimas (ar atlikti visi numatyti reikiami darbai);
3) Bendravimas su galutiniu klientu;
4) Sistemos specifikacijos sudarymas, jei yra galimybė, bendraujant su galutiniu klientu;
5) Sistemos kūrimo resursų paskirstymas;
6) Komandos informavimas (susirinkimų organizavimas, kliento naujienų/naujų reikalavimų/pakeitimų transliavimas komandai).

5. Prielaidos (būtinos sąlygos testavimui atlikti)

Šiame skyriuje aprašomos būtinos sąlygos, kurios užtikrins sklandų testavimo plano vykdymą.

1) Testuojamos sistemos dokumentacijoje turi būti aprašyti funkciniai ir nefunkciniai reikalavimai;

2) Funkciniai reikalavimai turi būti išreikšti panaudos atvejų diagrama bei veiklos diagramomis;
3) Turi būti numatyta klaidų taisymo strategija (nustatyti klaidų klasifikavimo požymiai – kurios klaidos turi būti taisomos pirmiausia, kaip turi būti skirstomos klaidos bei jų taisymo prioritetai);
4) Turi būti numatyta sistemos testavimo strategija (kokios sistemos vietos, funkcionalumas turi būti testuojamos pirmiausia);
5) Remiantis veiklos diagramomis turi būti sudaryti sistemos testavimo scenarijai;
6) Turi būti paruoštas naudotojų sistemos priėmimo testavimo planas (kas/kokiu metu bus suteikiama testuoti galutiniam naudotojui, kokiu būdu bus gaunamas grįžtamasis ryšys apie sistemos veikimą);
7) Prieš kiekvieną konkretų funkcionalumo testavimą, sistemoje privalo būti realizuotas atitinkamas funkcionalumas;
8) Turi būti įdiegta bei tinkamai sukonfigūruota testavimo aplinka (pvz., testavimo duomenų bazė atskirta nuo realiosios, dirbama su testavimo etapo failais);
9) Turi būti paruošta aplinka sistemos klaidų registravimui, darbų paskirstymui (pvz., Jira, GitHub, Redmine).

6. Testavimo aplinka

Testuojamos sistemos prezentaciniam puslapiui bei sistemos internetinei aplikacijai patalpinti naudojamos AWS (Amazon Web Services) technologijos – AWS Cloudfront (CDN), AWS S3, AWS Route53.

Duomenų bazei patalpinti naudojamas AWS RDS (1024RAM, 1vCPU, 1ECU (1GHZ)).

Testuojamos sistemos serverio pusės daliai (API) patalpinti naudojama jau minėta paslauga AWS Route53 bei AWS Beanstalk (512RAM, 1vCPU, Container, auto-scaling, load-balancing).

Testavimo scenarijai bus atliekami naudojant tokią aparatinę/programinę įrangą (minimalūs reikalavimai):

1) Stacionarus arba nešiojamas kompiuteris;
2) Procesorius: i7 bent 5 kartos ir 4 branduolių, sugebantis palaikyti bent 2.6GHz dažnį;
3) Bent 8 GB operatyviosios atminties (DDR4);
4) Bent 256GB SSD;
5) Microsoft Windows 10 OS;
6) Įrašytos naujausios populiariausių interneto naršyklių (Mozilla Firefox, Google Chrome) versijos.

7. Testavimo technikos

Tam, kad sistema būtų ištestuota tinkamai, neužtenka jai parinkti tik vienos kažkurios rūšies testus, turi būti taikomos skirtingos testavimo technikos, vykdomi skirtingi testavimo tipai. Mūsų testuojamai sistemai bus taikomos tokios testavimo technikos:

1) Vienetų (unit) testai – testai, kurie testuos konkrečias kodo vietas, funkcinius vienetus (pvz., atskiras funkcijas, klasių dalis ir pan.). Šie testai bus rašomi programuotojų ir bus kuriami lygiagrečiai kuriant sistemos funkcijas. Šių testų kūrimui numatyta naudoti xUnit biblioteką;
2) Integravimo (integration) testai – testai, kurie testuoja sistemos funkcijų veikimą, kai sistema bendrauja su kitomis trečiųjų šalių, išorinėmis sistemomis. Integravimo testai taip pat bus sudaromi programuotojų, kuriami kartu su vienetų testais, taip užtikrinant tinkamą sistemos API funkcijų veikimą (pvz., duomenų mainai su duomenų baze). Šių testų kūrimui taip pat numatyta naudoti xUnit, taip pat Mock biblioteką;
3) Performance testai – testai, kurie padės ištestuoti sistemos elgseną didinant duomenų apkrovas, užklausų kiekį, taip ištestuojant sistemos stabilumą, patikimumą, resursų panaudojimą bei padės identifikuoti sistemos spragas, kurių iš esmės negali parodyti kitų rūšių testai. Už šių testų sudarymą atsakingas bus kokybės užtikrinimo inžinierius;
4) Saugumo (security) testavimas – testai, kurie padės išsiaiškinti sistemos saugumo spragas naudotojų autentifikacijos ir autorizacijos procesuose. Testai padės užtikrinti, kad autentifikacijos procesas, prisijungiant prie sistemos, yra saugus, API metodai gali būti pasiekiami tik autentifikuotų bei autorizuotų sistemos naudotojų. Už šiuos testus taip pat atsakingas bus kokybės užtikrinimo inžinierius;
5) Atidavimo (acceptance) testavimas – testavimas, kurio metu bus išsiaiškinta, ar sukurta sistema tenkina naudotojo lūkesčius, ar sistema yra efektyvi ir veiksminga realaus veikimo sąlygomis (pvz., atliekant konkrečius užduočių scenarijus). Už šį testavimą bus atsakingi testuotojai, kurie sudarinės naudotojo vadovą ir pateiks konkrečias užduotis atlikti galutiniams naudotojams. Šio testavimo rezultatų analizę atliks testuotojai koordinuojami kokybės užtikrinimo inžinieriaus. Pats testavimas bus atliekamas du kartus – išpildžius pagrindinį kuriamos sistemos funkcionalumą bei realizavus visą sistemą.

8. Testavimo grafikas

3 lentelėje pateiktas kuriamos sistemos testavimo grafikas numatomomis datomis, kada turi būti vykdomas konkretus testavimo etapas.

3 lentelė. Kuriamos sistemos testavimo grafikas

Testavimo procesas	Nuo	Iki
Unit testai	2017-10-01	2018-02-01
Integration testai	2017-10-01	2018-02-01

Performance testai	2017-12-01	2018-02-15
Security testai	2017-11-01	2018-02-01
Acceptance testai (1 etapas)	2017-12-01	2017-12-15
Acceptance testai (2 etapas)	2018-02-15	2018-03-01

9. Testavimo rizikos

Šiame skyriuje pateikiami galimi keblumai, kurie gali atsirasti ir sutrikdyti normalią testavimo proceso eigą. Galimos testavimo rizikos ir jų sprendimo būdai:

1) Blogai sudarytas testavimo grafikas – gali atsitikti taip, kad testavimo grafike nurodyto testavimo procesų laiko gali neužtekti tinkamam sistemos testavimui. Šiai problemai ištaisyti būtų galima perdėlioti testavimo grafiką prailginant atitinkamų testavimo procesų periodus ar pasamdant naujų resursų į komandą, kad testavimą būtų galima suspėti atlikti per numatytą laiką;
2) Neigiami priėmimo testų rezultatai – galutiniams naudotojams (klientams) gali netikti galutinis sistemos vaizdas, naudotojo sąsaja, scenarijų bei užduočių vykdymas naudojantis sistema, todėl gali tekti perdaryti ne vieną sistemos komponentą ar sistemos išvaizdą. Galimam šios problemos sprendimui tinkamas variantas – glaudus bendradarbiavimas su galutiniu naudotoju, jam vis parodant ir duodant išmėginti sistemos funkcionalumą, kad grįžtamasis ryšys būtų gaunamas kiek galima anksčiau.
3) Darbuotojų kompetencijos stoka – tai glaudžiai siejasi su atliekamo testavimo kokybe bei galutinio testavimo rezultatu, nes daugiau patirties turintys darbuotojai gali išvengti kai kurių klaidų, kurios galbūt prasprūstų pro akis mažiau patirties turintiems darbuotojams. Taip pat mažiau patirties turintys darbuotojai yra linkę mažiau tiksliai įvertinti numatomus darbus bei jų vykdymo trukmę, kas gali daryti didelę įtaką testavimo grafikui. Todėl reikia stengtis, kad komandoje būtų bent po vieną mid-senior lygio programuotoją ar testuotoją, kuris galėtų padėti šių klaidų išvengti.
4) Kritinių klaidų neidentifikavimas – turimos omeny tos klaidos, kurios turi didelę reikšmę sistemos veikimui ir kurios testavimo proceso etape nebuvo surastos. Tokio klaidos dažniausiai išryškėja jau galutiniam naudotojui naudojantis sistema. Todėl reikia stengtis testavimo procesus vykdyti nuosekliai bei siekti kuo didesnio sistemos kodo padengimo testais, taip sumažinant tokių klaidų atsiradimo tikimybę.
5) Sistemų (testavimo aplinkų) trikdžiai – testavimo metu gali atsirasti techninių sistemos trikdžių ar programinės įrangos gedimų, kurie prailgintų numatytą testavimo procesų užbaigimo laikotarpį. Tokias problemas numatyti sunku, tačiau jų tikimybę galima sumažinti naudojantis kuo naujesne aparatine įranga, legalia programine įranga bei pasirinkamus tinkamus paslaugų (pvz., debesų kompiuterijos ar interneto) tiekėjus.

10. Testavimo scenarijai

Šiame skyriuje pateikiami testavimo scenarijai, kurie bus vykdomi testuojant kuriamą sistemą. Testavimo scenarijai aprašyti naudojant Gherkin testų rašymo kalbą.

10.1. Prezentacinio puslapio testavimo scenarijai

10.1.1. Naudotojo navigacija prezentaciniame puslapyje

Feature: Landing page navigation

Scenario: Landing page navigation scroll behaviour

Given a web browser is at the Scatterify landing page

When the user clicks the <item> in navigation menu

Then the page scrolls to the section <section>

Examples:

item	section
About	About us
Pricing plans	Pricing
Pre-register now!	Registration

Scenario: Back to top scroll behaviour

Given a web browser is at the Scatterify landing page

And the user scrolled below the “About us” section

When the user clicks the “Back to top” item in navigation menu

Then the page scrolls to the top of the landing page

10.1.2. Nukreipimas į internetinės aplikacijos registracijos formą beta versijai išmėginti

Feature: Redirecting to beta access registration form

Scenario: “Get started!” button behaviour

Given a web browser is at the Scatterify landing page

When the user clicks the “Get started!” button in Registration section

Then this page: https://scatterify.typeform.com/to/lie3ZJ opens in new browser tab

10.1.3. Prezentacinio puslapio išvaizda mobiliuosiuose įrenginiuose

Feature: Landing page responsiveness

Scenario: Landing page navigation

Given a web browser is at the Scatterify landing page

And the landing page is opened on mobile device

When the user scrolls below the page header section

Then mobile navigation toolbar appears on the top

And back to top button appears on the lower right corner

Scenario: Layout changes

Given a web browser is at the Scatterify landing page

And the landing page is opened on mobile device

When the user is at the “Pricing” section

Then tiles of the pricing plans are shown in one column

10.2. Internetinės aplikacijos testavimo scenarijai

10.2.1. Prisijungimas

Feature: Facebook login

Scenario: Login success

Given the following Facebook users exist:

email	password
scatterify_user@scatterify.com	scatterify_success
scatterify_user2@scatterify.com	scatterify_success
scatterify_user3@scatterify.com	scatterify_success

And a web browser is at the Scatterify application home page

When the user clicks the “Continue with Facebook” button

And the user enters <email> as email

And the user enters <password> as password

Then the user is logged in to the Scatterify web application

Examples:

email	password
scatterify_user@scatterify.com	scatterify_success
scatterify_user2@scatterify.com	scatterify_success
scatterify_user3@scatterify.com	scatterify_success

Scenario: Login failed

Given the following Facebook users exist:

email	password
scatterify_user@scatterify.com	scatterify_success
scatterify_user2@scatterify.com	scatterify_success
scatterify_user3@scatterify.com	scatterify_success

And a web browser is at the Scatterify application home page

When the user clicks the “Continue with Facebook” button

And the user enters <email> as email

And the user enters <password> as password

Then “The password that you’ve entered is incorrect” error message is shown

Examples:

email	password
scatterify_user@scatterify.com	scatterify_failed
scatterify_user2@scatterify.com	scatterify_failed
scatterify_user3@scatterify.com	scatterify_failed

10.2.2. Atsijungimas

Feature: Logging out from the system

Scenario: Log out button behaviour

Given a web browser is at the Scatterify web application

And the user is logged in

When the user clicks the "Log out" button

Then the user is logged out from the system

And the user is redirected to Scatterify web application home/login page

10.2.3. Organizacijos/prekybos vietų langas

Feature: Branches page

Scenario: Worker's branches page general view

Given the user is logged in as role "Worker"

And the user is assigned to at least one branch

When the user is in branches page

Then the user sees all the assigned organizations and branches

Scenario: Owner's branches page general view

Given the user is logged in as role "Owner"

When the user is in branches page

Then the user sees organization customization page

Scenario: Worker's organizations amount on page

Given the user is logged in as role "Worker"

And branches belong to different organizations

And the user is assigned to <amount> branches

When the user is in branches page

Then <amount> organizations are shown

Examples:

amount
1

| 2 |

| 3 |

Scenario: Worker's branches amount on page

Given the user is logged in as role "Worker"

And branches belong to the same organization

And the user is assigned to <amount> branches

When the user is in branches page

Then 1 organization is show with <amount> of branches to choose from

Examples:

| amount |

| 1 |

| 2 |

| 3 |

Scenario: Worker selects branch

Given the user is logged in as role "Worker"

And the user is assigned to at least one branch

When the user is in branches page

And the user selects one of the branches

Then the user is redirected to orders page

And orders of the selected branch are shown

Scenario: Owner's branches page view when organization is created

Given the user is logged in as role "Owner"

And the owner has created the organization before

When the user is in branches page

Then the user sees organization information and created branches

Scenario: Owner's branches page view when organization is not created

Given the user is logged in as role "Owner"

And the owner has not created the organization before

When the user is in branches page

Then "You have not created any organization yet." message is shown

And "Create organization" button is shown

10.2.4. Užsakymų langas

Feature: Orders page

Scenario: Owner can’t see orders page

Given a web browser is at the Scatterify web application

When the user is logged in as role “Owner”

Then orders page is not visible in menu

Scenario: Worker has not selected any branch

Given a web browser is at the Scatterify web application

And the user is logged in as role “Worker”

And the user has not selected any branch in branches page

When the user is trying to access orders page

Then the user is redirected to the branches page

And “Choose branch” message is shown

Scenario: Worker has selected a branch

Given a web browser is at the Scatterify web application

And the user is logged in as role “Worker”

And the user has selected a branch in branches page

When the user is in orders page

Then orders of that branch are shown

Scenario: Branch has active orders

Given a web browser is at the Scatterify web application

And the user is logged in as role “Worker”

And the user has selected a branch in branches page

And branch has <amount> active orders

When the user is in orders page

Then <amount> orders of that branch are shown

Examples:

amount
1
2
3

Scenario: Branch has no active orders

Given a web browser is at the Scatterify web application

And the user is logged in as role “Worker”

And the user has selected a branch in branches page

And the selected branch has no active orders

When the user is in orders page

Then no orders are shown

10.2.5. Naršyklės pranešimai

Feature: Browser notifications

Scenario: Permission for browser notifications

Given a web browser is at the Scatterify web application

When the user logs into the system

And user has not allowed browser notifications yet

Then the browser notification dialog pops up

Scenario: New order browser notification

Given a web browser is at the Scatterify web application

And the user is logged in as “Worker”

When the new order appears in orders page

Then the browser notification with text “New order received” appears

11. Testavimo rezultatai bei testavimo išbaigtumas

Testavimo proceso pabaigoje turi būti pasiekti tokie rezultatai:

1) Užtikrinta, kad testuojama sistema yra stabili naudotis;
2) Paruoštas užbaigtas testavimo planas;
3) Paruoštas užbaigtas sistemos naudotojų vadovas;
4) Paruoštos kiekvieno testavimo grafike nurodyto testavimo proceso ataskaitos;
5) Užbaigta testuojamos sistemos specifikacija (kuri galėjo keistis po naudotojų atidavimo testų, jei buvo atliekami kokie nors specifikacijoje nenumatyti sistemos pokyčiai).

Sistemos testavimas laikomas išbaigtu, kai:

1) Sistemos užsakovas (jei toks yra) ir sistemos kūrimo komanda tarpusavyje susitaria, jog sistemos testavimas ir jo rezultatai tenkina abi šalis;
2) Kai sistemoje nėra likusių užregistruotų jokių didesnės reikšmės klaidų (svarba S3 ir aukštesnė);
3) Visi automatiniai testai grąžina teigiamą rezultatą (testai įvykdomi sėkmingai);
4) Testuojamo kodo padengimas siekia bent 90%;

5) Kai sistemoje atsirandančių klaidų skaičius tendencingai mažėja ir artėja prie 0;
6) Kai sistema savo darbą atlieka taip pat stabiliai, nepriklausomai nuo augančio sistemos naudotojų skaičiaus.

12. Išvados

Laboratorinio darbo metu buvo:

1) Išanalizuota testavimo plano struktūra;
2) Išskirti svarbiausi testavimo planą sudarantys punktai, jie detalizuoti bei pritaikyti testuojamai sistemai;
3) Sudarytas testuojamos sistemos testavimo planas;
4) Paruošta laboratorinio darbo ataskaita.