KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Programų sistemų testavimas (T120B162)

Testavimo planas

Darbą atliko IFF-7/14
Grupės studentai:

Rimvydas Neverauskas

Valentinas Kasteckis

Henrikas Juzuitis

Eligijus Kiudys

Darbą priėmė:

lekt. BARISAS Dominykas

Kaunas, 2020

Turinys

1. Laboratorinio darbo tikslai ir uždaviniai 3
2. Pasirinktas projektas testavimui 3
   2.1. Aprašymas 3
   2.2. Architektūra 3
   2.3. Panaudojimo atvejai....4
   2.4. Funkciniai reikalavimai 4
   2.5. Diegimas 4
3. Testavimo planas 5
   3.1. Vienetų testavimas....5
   3.2. Kvalifikacijos testavimas....6
   3.3. Testavimas nepalankiausiomis klasėmis 6
   3.4. Našumo testavimas 7
4. Testavimo taikymo sritis....8
5. Testavimo strategija....8
6. Testavimo technikos 8
7. Pradiniai reikalavimai testavimui 9
8. Testavimo prioritetai....9
9. Testavimo aplinka....9
10. Testavimo scenarijus 10
   1.1. Sukurti žaidimo sesiją....10
   1.2. Prisijungti prie žaidimo....10
   1.3. Laivo padėjimas....10
   1.4. Šūvis 10
   2.1. Sukurti žaidimą....11
   2.2. Prisijungti prie žaidimo....11
   2.3. Laivų statymas....12
   2.4. Laivų pozicijų pateikimas žaidimui....12
   2.5. Komunikacija tarp žaidėjų....12
   2.6. Šauti šūvį 12
   2.7. Šūvių vizualizavimas....13
   2.8. Laivo nušovimas....13
   2.9. Žaidėjo atsijungimas ir pasidavimas....13
   2.10. Žaidimo pabaiga 14
11. Testavimo rizikos....14
12. Testavimo kalendorius....14

1. Laboratorinio darbo tikslai ir uždaviniai

1.1. Pasirinkti programinę įrangą, kuri bus testuojama ir parengti jos aprašymą.
1.2. Pasirinkti testavimo plano skyrius.
1.3. Testavimo plane pasirinkti aktualius skyrius.
1.4. Testavimo plane aprašyti pasirinktus skyrius.

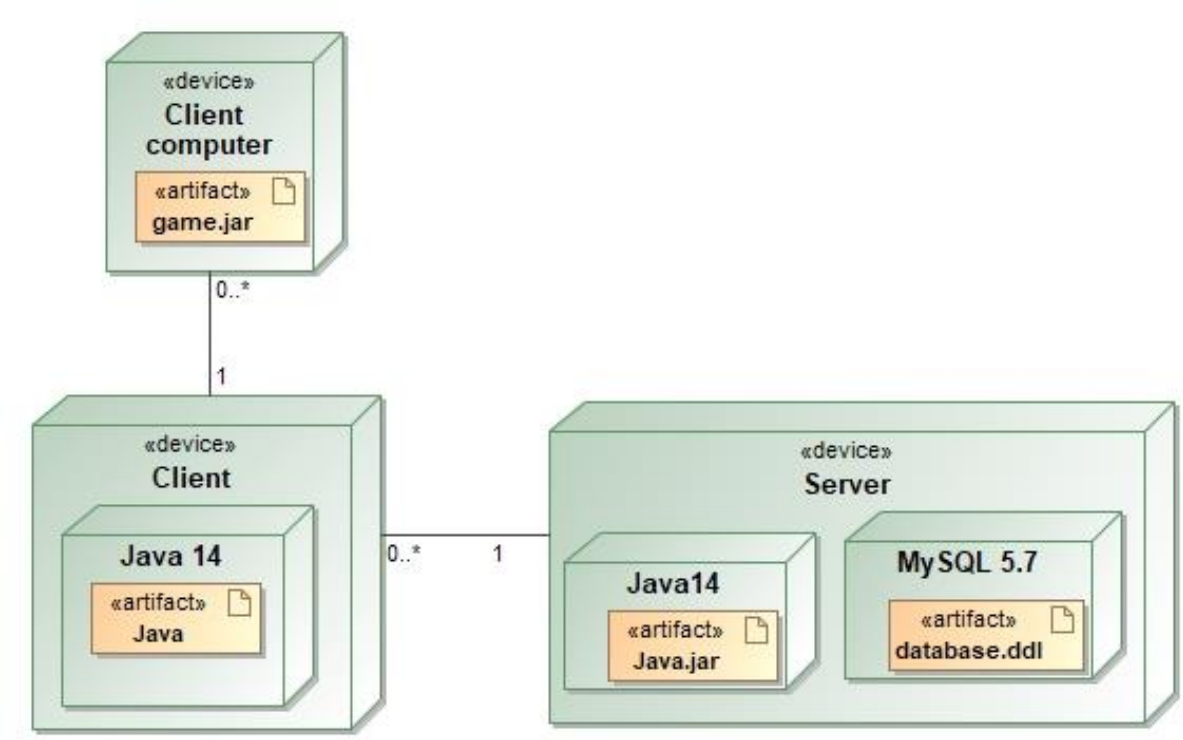
2. Pasirinktas projektas testavimui

2.1. Aprašymas

Žaidimas „Laivų mūšis“. Žaidimas yra išskaidytas į 2 dalis – klientinė bei serverinė dalis. Tiek klientinė, tiek serverinė dalis realizuota su JAVA programavimo kalba. Kliento sąsaja sukurta su Java Swing.

2.2. Architektūra

Pav. 1 architektūros diagrama



2.3. Panaudojimo atvejai

Pav. 2 panaudojimo atveju diagrama

2.4. Funkciniai reikalavimai

- Žaidėjai turės prisijungti.
  - Turi dalyvauti du žaidėjai.
- Žaidėjai turi susidėlioti laivus.
  - Visi laivai turi būti sudėlioti, kitaip žaidimas neprasidės.
- Žaidėjas gali pulti savo priešą.
- Kiekvienas šūvis matomas pas priešininką, bei pas šaulį.
- Apie pilnai nuskandintus laivus bus pranešta.
- Žaidime bus sekami raundai.
- Žaidime bus pranešimai apie pergales.
- Žaidimas baigsis, kai kažkuris žaidėjas nuskandins visus savo priešo laivus.

2.5. Diegimas

Serverinė dalis yra įdiegta ant Azure „Linux“ serverio. Naudojama Ubuntu 18.04 distribucija bei Java 14 versija. Duomenų bazė naudoja MySQL 5.7.

3. Testavimo planas

3.1. Vienetų testavimas

Numatomas tikslas ištestuoti 50% parašyto kodo vienetų testais.

Bus naudojamas **JUnit** testams rašyti, kadangi tiek serverinė, tiek klientinė dalis bus padaryta su **Java** programavimo kalba.

Prioritetas bus skiriamas sudėtingiausia logiką vykdančiomis klasėmis, įvairios DTO (*Data Transfer Object*) klasės nebus testuojamos, kadangi jose nebus vykdoma sudėtinga logika apart *set* bei *get* metodų.

Testavimo technikos:

Su **JUnit** bus testuojamos sudėtingiausių klasių vykdomoji logika. Testuojant serverine dalį, vietoje kreipimosi į duomenų bazę, duomenys bus pakeičiami netikrais (*Mock*), o klientinėje dalyje, kadangi ji vykdys HTTP užklausas į serverinę dalį, grąžinamas atsakas iš serverio (*Response*) taip pat bus pakeičiamas į netikrą.

Kodo padengiamumo skaičiavimas bus apskaičiuojamas naudojantis **JUnit** su **Intelij** programavimo aplinka.

Siekiamas rezultatas yra, jog po kiekvieno naujo kodo papildymo (*commit* į repositoriją), testai būtu visi „žali“.

Vienetų testų tikslas:

Pasiekti tam tikrą testų kiekio lygį, jog kuriant naujus funkcionalumus sistemai, vienetų testai užtikrint, kad tam tikri komponentai veikia teisingai.

Potencialios rizikos ir sprendimas:

Tam tikrų klasių metodai gali būti privatūs. Programuojant bus stengiamasi logiką skaidyti į tam tikrus komponentus (klases) bei juos pernaudoti panaudojant *dependency injection* į pagrindinę klasę, taip metodai testuojamos klasės bus vieši.

Kas testuos programinę įranga ir kada testai bus paleidžiami?

Programuotojus bet kuriuo metu galės paleisti JUnit testus, taip pat po kiekvieno kodo papildymo, bus paleidžiami Unit testai ir grąžinamas įvertinimas su išklotinė ar testai buvo sėkmingai įvykdyti ar ne. Jeigu jie buvo nesėkmingai įvykdyti, bus pateikiama išklotinė kurie testai „nepraėjo“.

Suspendavimo kriterijus

Jeigu darant papildomą funkcionalumą, seni jau parašyti testai nepraeina ir grąžinamos klaidos, reikia juos pataisyti ir tik tada tęsti naujus funkcionalumus.

Testavimo aplinkos bus lokalios, jos gali būti įvairių operacinių sistemų, tai yra:

- MacOS
- Windows 10
- Linux (Ubuntu 20.04 arba Ubuntu 18.04)

Testavimas produkcijos serveryje nebus vykdomas. Taip pat bus realizuota *Continious Integration* dalis su Github Actions, tai yra po kiekvieno kodo papildimo (*Commit*), bus pakuriama virtuali aplinka su Linux operacine sistema bei Ubuntu 20.04 distribucija, bus įrašoma Java 14 versiją bei paleidžiami JUnit testai. Taip pat planuojama pridėti ir statinę kodo analizę.

Dalinė išvada – bus siekiama pasiekti 50% kodo padengiami naudojantis **JUnit** technologija.

3.2. Kvalifikacijos testavimas

Kvalifikacijos testavimas tikrina ar sistemos veikimas užtikrina vartotojo lūkesčius. Šio testavimo tikslas yra įvertinti sistemos atitikimą su iškeltais vartotojo tikslais ir patikrinti jų atitikimą.

- Kvalifikacijos kriterijai
    - Sukūrus žaidimą, vartotojui yra pranešama apie žaidimo sukūrimą ir yra pateikiamas žaidimo sesijos raktas.
    - Prisijungiant prie žaidimo vartotojui yra pateikiamas priešininko slapyvardis, pranešimas apie prisijungimą prie žaidimo ir galimybė pradėti statyti laivus.
    - Laivų statymas žaidime privalo būti pavaizduotas vizualiai vartotojui.
    - Vartotojui pateikiant laivų pozicijas, žaidimas praneša ar pozicijos buvo priimtos arba nepriimtos.
    - Žaidėjai gali komunikuoti tarpusavyje žinučių pagrindu.
    - Apie negalimus šūvius yra pranešama žaidėjui.
    - Vartotojo ir priešininko šūviai yra vizualiai pateikiami abiem žaidėjams.
    - Žaidimas praneša žaidėjams apie nušautus laivus.
    - Žaidimas praneša žaidėjui apie priešininko pasidavimą ir atsijungimą.
    - Žaidimas praneša apie žaidėjo pergalę.

Kvalifikacijos testavimas bus atliekamas rankinių būdų. Šio testo pagalba yra tikimasi išpildyti vartotojo lūkesčius pagal išvardintus kvalifikacijos kriterijus.

3.3. Testavimas nepalankiausiomis sąlygomis

Streso testavimas skirtas patvirtinti programos stabilumo ir patikimumą. Šio testavimo esmė išmatuoti programos atsparumą, bei klaidų apdorojimą ekstremaliomis apkrovos sąlygomis.

Testavimo Įrankiai

Šiam testavimui bus naudojamas atviro kodo įrankis **JMeter** bei rankinis testavimas. JMeter yra atviro kodo **Java** aplikacija skirta atlikti įvairiems testavimams kaip nepalankių sąlygų testavimas, našumo testavimas ir testų analizė.

Tikslas

- Stabilumas – Apkrovos testo metu patikrinti ar aplikacija veikia pagal testavimo scenarijų taip pat bus patikrinta ar neatsiranda klaidų.
- Aprovimas – Patikrinti kiek daugiausia žmonių gali žaisti ir patikrinti kokį limitą pasiekus aplikacija nebeveikia.

Technika

Kadangi sistema yra paremta REST principu, su JMeter turime sukurti gijų grupę kurioje nustatome kiek naudotojų apkraus sistemą. Susidarome testavimo planą JMeter aplikacijoje. Naudosime HTTP Request tipo testą į kurį įvesime sistemos serverio adresą, norimą ištestuoti metodą (GET, PUT, POST, DELETE) su norimais parametrais. Prie šio testo pridedame klausytoją, kuris bus rezultatų medis. Apkrovimo metu bus naudojamas rankinis testavimas. Šis testavimas padės patikrinti kaip aplikacija veikia apkrovimo metu ir kokios problemos tada atsiranda.

Testo rezultatai

- Užklausos siuntimo, gavimo laikas
- Netikėtų klaidų aptikimas
- Serverio lūžio riba

3.4. Našumo testavimas

Našumo testavimas verifikuoja iškeltus sistemos našumo reikalavimus. Taip pat šis testavimas įvertina esamos sistemos galimybes ir reakcijos trukmę. Šio tipo testavimas yra atliekamas kai sistemos funkcionalumas yra stabilus ir ją gali pasiekti keletas žmonių vienu metu.

Įrankis

Šiam testavimui bus naudojamas atviro kodo įrankis **JMeter**. Tai yra **Java** aplikacija, kuri skirta apkrauti sistemos funkcionalumą ir išanalizuoti sistemos našumą šio apkrovimo metu.

Tikslas

Šio testavimo metu bus tikrinamas sistemos:

- Reakcijos laikas – laikas per kurį serveris atsako į kliento užklausą.
- Apkrovimas – kiek vienu metu gali būti žaidimo sesijų.
- Stabilumas – sistemos stabilumas, kai vienu metu vyksta tam tikras žaidimo sesijų kiekis, tam tikrą laiką.

Technika

Kadangi sistema yra paremta **REST** principu, su **JMeter** turime sukurti gijų grupę ir į ją įdėti norimą testą, šiuo atveju **HTTP Request** į kurį įvesime sistemos serverio adresą, bei norimą ištestuoti metodą (GET, PUT, POST, DELETE) su norimais parametrais. Prie šio testo pridedame klausytoją, kuris bus rezultatų medis. Čia matysime visą testavimo užklausos informacija, t.y.:

- Užklausos krovimo laiką
- Prisijungimo laiką prie serverio
- Vėlavimo trukmę
- Klaidų skaičių
- Atsakymo kodą
- Atsakymo žinutę

4. Testavimo taikymo sritis

Testavimas bus atliekamas žaidimo pagrindiems logikos sritims ir užsakovo iškeltiems reikalavimams. Taip pat testavimo pagalba bus įsitikinta, kad programos veikimas atitinka vartotojo veikimo lūkesčius.

- Vienetų testų prioritetas bus skiriamas sudėtingiausia logiką vykdančiomis klasėmis, įvairios DTO (*Data Transfer Object*) klasės nebus testuojamos, kadangi jose nebus vykdoma sudėtinga logika apart *set* bei *get* metodų.

- Kvalifikacijos testai apims pagrindines vartotojo sąsajos funkcijas. Šias funkcijas žaidimo vartotojai naudoja kiekvieną kartą pradedant žaidimą ir pasikartojančiai vykdo žaidimo sesijos metu. Šių testų tikslas yra tikslingai ir sėkmingai užtikrinti žaidėjus apie žaidimo būseną ir jo eigą.

- Našumo testavimas padės nustatyti maksimalų žaidimų sesijų kiekį, kuris nepadarytų įtakos serverio našumui, bei atsako laikams. Taip bus užtikrintas sistemos stabilumas.

5. Testavimo strategija

Pirmiausia bus vykdomi vienetų testai, patikrinti atskiriems sistemos komponentams. Tikslas yra pasiekti tam tikrą testų kiekio lygį, jog kuriant naujus funkcionalumus sistemai, vienetų testai užtikrintų, kad tam tikri komponentai veikia teisingai.

Našumo testuose tikrinsime komunikacijos greičius tarp žaidimo kliento ir serverio, susidarant įvairioms žaidimo būsenoms ir padėtims.

Toliau atliksime nepalankių sąlygų testavimą. Sistemos streso testuose bus patikrinta žaidimų kiekių įtaka komunikacijai, tarp klientinės ir serverines žaidimų dalių. Taip pat ištirsime, kokia įtaka tai daro žaidėjo sesijos veikimui.

Galiausiai bus atliekamas rankinis kvalifikacijos testavimas, patikrinti ar testuojama sistema atitinka iškeltus kvalifikacijos kriterijus vartotojui.

Testavimas produkcijos serveryje nebus vykdomas.

Testavimo aplinkos bus lokalios, jos gali būti įvairių operacinių sistemų, tai yra:

- MacOS
- Windows 10
- Linux (Ubuntu 20.04 arba Ubuntu 18.04)

6. Testavimo technikos

1. Vienetų testavimas:
   Su **JUnit** bus testuojamos sudėtingiausių klasių vykdomoji logika. Testuojant serverine dalį, vietoje kreipimosi į duomenų bazę, duomenys bus pakeičiami netikrais

(*Mock*), o klientinėje dalyje, kadangi ji vykdys HTTP užklausas į serverinę dalį, grąžinamas atsakas iš serverio (*Response*) taip pat bus pakeičiamas į netikrą.

Kodo padengiamumo skaičiavimas bus apskaičiuojamas naudojantis **JUnit** su **Intelij** programavimo aplinka.

Siekiamas rezultatas yra, jog po kiekvieno naujo kodo papildymo (*commit* į repositoriją), testai būtu visi „žali“.

2. Kvalifikacijos testavimas:
   Vartotojo sąsajos testavimas pagal iškeltus kvalifikacijos kriterijus bus atliekamas rankiniu būdu. Testuotojas privalės paleisti žaidimo vartotojo sąsają ir atlikti visus testus aprašytus testavimo scenarijuje, kad įvertintų kvalifikacijos lygį.

3. Testavimas nepalankiausiomis sąlygomis
   Kadangi sistema yra paremta REST principu, su JMeter turime sukurti gijų grupę kurioje nustatome kiek naudotojų apkraus sistemą. Susidarome testavimo planą JMeter aplikacijoje. Naudosime HTTP Request tipo testą į kurį įvesime sistemos serverio adresą, norimą ištestuoti metodą (GET, PUT, POST, DELETE) su norimais parametrais. Prie šio testo pridedame klausytoją, kuris bus rezultatų medis. Apkrovimo metu bus naudojamas rankinis testavimas. Šis testavimas padės patikrinti kaip aplikacija veikia apkrovimo metu ir kokios problemos tada atsiranda.

7. Pradiniai reikalavimai testavimui

- Privalo būti sukurta programinės įrangos specifikacija.
- Veikianti programa.
- Privalo būti sukurta ir veikianti testavimo aplinka.

8. Testavimo prioritetai

- Funkcionalumas – ar programa atlieka visas jai priklausančias funkcijas tinkamai?
- Tinkamumas naudoti – ar programa lengvai suprantama?
- Našumas – ar programa atitinka sutartus našumo kriterijus?

9. Testavimo aplinka

Serveris:

- Linux (ubuntu 18.04)
- 1 GiB of RAM
- 1 vCPU

Klientas:

- MacOS

- Windows 10
- Linux (Ubuntu 20.04 arba Ubuntu 18.04)f

10. Testavimo scenarijus

1. Vienetų testų scenarijus

1.1. Sukurti žaidimo sesiją

1.1.1. Aprašymas: žaidimo sesijos pradžios sukūrimas.

1.1.2. Testavimo žingsniai: įvesti vartotojo slapyvardį ir paspausti sukurti žaidimą mygtuką.

Laukiamas rezultatas
Žaidimo sesija sukurta

1.2. Prisijungti prie žaidimo

1.2.1. Aprašymas: prisijungimas prie sukurto žaidimo.

1.2.2. Prieš sąlygą: turi būti sukurta žaidimo sesija.

1.2.3. Testavimo žingsniai: įvedamas žaidimo kodas ir spaudžiamas prisijungimo mygtukas.

Kodas	Žaidimo sesija egzistuoja	Laukiamas rezultatas
xCa6d4t	Taip	Žaidėjas prijungtas prie žaidimo
xCa6d4s	Ne	Žaidėjas neprijungiamas prie žaidimo

1.3. Laivo padėjimas

1.3.1. Aprašymas: padedamas laivas žaidimo žaidėjo lange.

1.3.2. Prieš sąlygą: turi būti prisijungę du žaidėjai vienoje sesijoje.

1.3.3. Testavimo žingsniai: pasirenkamas laivas kurį norima įdėti ir spaudžiama ant langelio į kurį bus dedamas, kai visi laivai sudėti spaudžiamas pabaigos mygtukas.

Laivo pozicijos	Visi laivai sudėti	Laukiamas rezultatas
Žaidėjo ribose	Taip	Laivai padedami, atnaujinamas žaidimo langas
Už žaidėjo ribų	Taip	Neleidžiama pabaigti laivų dėjimo
Žaidėjo ribose	Ne	Neleidžiama pabaigti laivų dėliojimo

1.4. Šūvis

1.4.1. Aprašymas: šaunama į priešininko lentą.

1.4.2. Prieš sąlygą: turi būti prisijungę du žaidėjai vienoje sesijoje ir sudėlioti visi laivai.

1.4.3. Testavimo žingsniai: pasirenkamas norimas šūvis ir pasirenkama koordinatė į kurią šaunama.

Pataikė	Laukiamas rezultatas

Taip	Langelis nudažomas raudonai
Ne	Langelis nudažomas juodai

2. Kvalifikacijos kriterijų testavimo scenarijus

2.1. Sukurti žaidimą

2.1.1. Aprašymas: Žaidimo sesijos sukūrimas.
2.1.2. Prieš sąlyga: žaidėjas nėra prisijungęs prie kito žaidimo sesijos.
2.1.3. Testavimo žingsniai: Testuotojas nuspaudžia sukurti žaidimą mygtuką.

Paspaudė žaidimo sukūrimo mygtuką	Laukiamas rezultatas
Taip	Konsolėje yra išvedamas pranešimas apie sėkminga žaidimo sesijos sukūrimą ir yra pateikiamas žaidimo sesijos raktas.
Ne	Konsolėje nėra išvedamas pranešimas.

2.2. Prisijungti prie žaidimo

2.2.1. Aprašymas: Žaidėjo prisijungimas prie žaidimo sesijos su slapyvardžiu ir sesijos raktu.
2.2.2. Prieš sąlyga: žaidėjas Nėra prisijungęs prie kito žaidimo sesijos ir yra įvedęs slapyvardį.
2.2.3. Testavimo žingsniai: Testuotojas įveda slapyvardį, žaidimo sesijos raktą ir spaudžia prisijungti prie žaidimo mygtuką.

Slapyvardis	Ar žaidimo sesija egzistuoja	Laukiamas rezultatas
	Taip	Konsolėje yra išvedamas pranešimas apie nesėkminga prisijungimą prie žaidimo, dėl neegzistuojančio slapyvardžio.
	Ne	Konsolėje yra išvedamas pranešimas apie nesėkminga prisijungimą prie žaidimo, dėl neegzistuojančio sesijos rakto.
Tomas	Ne	Konsolėje yra išvedamas pranešimas apie nesėkminga prisijungimą prie žaidimo, dėl neegzistuojančio sesijos rakto.

Tomas	Taip	Konsolėje yra išvedamas pranešimas apie sėkminga prisijungimą prie žaidimo.

2.3. Laivų statymas

2.3.1. Aprašymas: Žaidimo pradžios laivų statymo etapas.

2.3.2. Prieš sąlyga: Žaidėjas yra prisijungęs prie žaidimo sesijos ir žaidimas laukia žaidėjo laivų pozicijų.

2.3.3. Testavimo žingsniai: Testuotojas stato laivus ant žemėlapio.

Leistina laivo pozicija	Laukiamas rezultatas
Taip	Pasirinkta pozicija žemėlapyje yra nuspalvinama žaliai.
Ne	Pasirinkta pozicija žemėlapyje yra nuspalvinama oranžiniai.

2.4. Laivų pozicijų pateikimas žaidimui

2.4.1. Aprašymas: Vartotojo sustatytų laivų pozicijų pateikimas.

2.4.2. Prieš sąlyga: Žaidėjas yra prisijungęs prie žaidimo sesijos, žaidimas laukia žaidėjo laivų pozicijų.

2.4.3. Testavimo žingsniai: Testuotojas paspaudžia pateikti laivų pozicijas mygtuką.

Visi laivai yra pastatyti	Laukiamas rezultatas
Taip	Konsolėje yra išvedamas pranešimas apie sėkmingai pastatytus laivus ir pranešama, kad dabar laukiame kito žaidėjo laivų.
Ne	Konsolėje yra išvedamas pranešimas apie nesėkminga laivų pozicijų pateikimą.

2.5. Komunikacija tarp žaidėjų

2.5.1. Aprašymas: Žaidėjų komunikacija tekstinių žinučių pagrindų.

2.5.2. Prieš sąlyga: Žaidėjas yra prisijungęs prie žaidimo sesijos.

2.5.3. Testavimo žingsniai: Testuotojas įvedą žinutės tekstą ir paspaudžia siųsti žinutę.

Žinutes tekstas	Laukiamas rezultatas
	Konsolėje yra išvedamas pranešimas apie nesėkmingai, dėl tuščios žinutės teksto.
Sveikas	Konsolėje yra išvedama žinutė abiem žaidėjams.

2.6. Šauti šūvį

2.6.1. Aprašymas: Žaidėjas šauna šūvį į pasirinktą žemėlapio poziciją.

2.6.2. Prieš sąlyga: Žaidėjas yra prisijungęs prie žaidimo sesijos ir žaidėjo eilė atlikti veiksmą.
2.6.3. Testavimo žingsniai: Testuotojas paspaudžia žemėlapio poziciją.

Nešautas langelis	Laukiamas rezultatas
Taip	Pasirinktos pozicijos langelis yra nuspalvinamas.
Ne	Konsolėje yra išvedamas pranešimas apie nesėkmingai šūvį.

2.7. Šūvių vizualizavimas

2.7.1. Aprašymas: Žaidėjas šovė šūvį į pasirinktą žemėlapio poziciją.
2.7.2. Prieš sąlyga: Žaidėjas yra prisijungęs prie žaidimo sesijos ir žaidėjo šūvis yra galimas.
2.7.3. Testavimo žingsniai: Testuotojas stebi kaip pasikeičia pasirinktos pozicijos langelis.

Žaidėjas pataikė į laivą	Laukiamas rezultatas
Taip	Pasirinktos pozicijos langelis yra nuspalvinamas mėlyna spalva.
Ne	Pasirinktos pozicijos langelis yra nuspalvinamas raudona spalva.

2.8. Laivo nušovimas

2.8.1. Aprašymas: Nušauto laivo pranešimas žaidėjams
2.8.2. Prieš sąlyga: Žaidėjas yra prisijungęs prie žaidimo sesijos ir žaidėjo šūvis yra galimas.
2.8.3. Testavimo žingsniai: Testuotojas stebi žemėlapio langelius ir konsolę.

Žaidėjas nušovė laivą	Laukiamas rezultatas
Taip	Nušauto laivo pozicijos abiem žaidėjams yra nuspalvinamos violetiniai ir konsolėje yra pranešama apie nušauta laivą.
Ne	Pasirinktos pozicijos langelis yra nuspalvinamas raudona spalva.

2.9. Žaidėjo atsijungimas ir pasidavimas

2.9.1. Aprašymas: Žaidėjo atsijungimas ir pasidavimas žaidimo sesijos metu.
2.9.2. Prieš sąlyga: Priešininkas yra prisijungęs prie žaidimo sesijos ir žaidimas yra nepasibaigęs.
2.9.3. Testavimo žingsniai: Testuotojas atjungia žaidėja arba nuspaudžia pasiduoti mygtuką.

Žaidėjo veiksmas	Laukiamas rezultatas
Žaidėjas pasiduoda	Priešininko konsolėje yra išvedamas pranešimas, kad žaidėjas pasidavė.

Žaidėjas atsijungė	Priešininko konsolėje yra išvedamas pranešimas, kad žaidėjas atsijungė.

2.10. Žaidimo pabaiga

2.10.1. Aprašymas: Žaidimo sesijos pabaiga.

2.10.2. Prieš sąlyga: Priešininkas yra prisijungęs prie žaidimo sesijos ir žaidimo būsena yra pasibaigusi (galutinė stadija).

2.10.3. Testavimo žingsniai: Testuotojas stebi kaip keičiasi konsolės pranešimai.

Žaidimo sesijos pabaigos priežastis	Laukiamas rezultatas
Žaidėjas pasidavė	Konsolėje yra išvedamas pranešimas, kad žaidėjas pasidavė ir priešininkas laimėjo ir žaidimo sesija yra pabaigiama.
Žaidėjas atsijungė	Konsolėje yra išvedamas pranešimas, kad žaidėjas atsijungė ir priešininkas laimėjo ir žaidimo sesija yra pabaigiama.
Žaidėjo visi laivai buvo nušauti	Konsolėje yra paskelbiamas laimėtojas, pralaimėtojas ir žaidimo sesija yra pabaigiama.
Priešininko visi laivai buvo nušauti	Konsolėje yra paskelbiamas laimėtojas, pralaimėtojas ir žaidimo sesija yra pabaigiama.

11. Testavimo rizikos

Rizika	Rizikos aprašymas	Rizikos mažinimas
Programinės įrangos kodo prieinamumo kontrolė	Programinės įrangos kodas gali turėti skirtingus prieinamumo lygius, kaip private, protected ir public.	Programuojant kodo logiką skaidyti į komponentus ir naudoti dependency injections pagrindinėse klasėse.
Serverio gyvavimas	Serverio gyvavimo išlaikymas gali sudaryti trukdžių žaidimo veiklai ir testavimui.	Palaikyti testavimą su realistiniu lokaliu serveriu.

12. Testavimo kalendorius

Testavimo tipas	Pradžia	Pabaiga
Vienetų testai	2020-09-01	2020-10-05
Našumo testavimas	2020-10-06	2020-11-07
Testavimas nepalankiausiomis sąlygomis	2020-11-08	2020-11-25
Kvalifikacijos testavimas	2020-11-26	2020-12-30