TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Tarkvaratehnika õppetool

Flash rakenduse suitsutestide automatiseerimine erinevate platvormide jaoks kasutades SikuliX ja teisi abitööriistu

Bakalaureusetöö

Üliõpilane: Natalja Tamming
Üliõpilaskood: IABB124531
Juhendaja: Jekaterina Tšukrejeva

Tallinn
2016

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

---------- ----------

(*kuupäev*) (*allkiri*)

Annotatsioon

Käesoleva töö eesmärgiks on suitsutestide automatiseerimine arvuti-, veebi- ja mobiilplatvormidel Flash rakenduse jaoks. Selleks kasutatakse Java programmeerimiskeelt, raamistikku SikuliX ja teisi abistavaid tööriistu, millest antakse ülevaade töös.

Suitsutestide automatiseerimine peaks vähendama testimisele kuuluvat aega ja seeläbi kiirendama arendusprotsessi ning selle arvelt parandama toote kvaliteeti.

Projektitöö tulemusena esitletakse komplekti automaattestidest, mis katavad olulist osa automatiseerimist vajavatest suitsutestidest. Testide kohta on koostatud testilood koos selgitustega. Samuti tutvustatakse SikuliX raamistikku ja teda abistavaid tööriistu. Töö tulemusena on järeldatud, et valminud automaattestid ja koostöö SikuliX'i ja teda abistavate tööriistade vahel töötavad vastavalt vajadustele ning on valmis edaspidiseks arendamiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 61 leheküljel, 6 peatükki, 16 joonist, 2 tabelit.

Abstract

The aim of this thesis is to automate smoke test cases for Flash based application on desktop, web and mobile platforms. It is done by using Java programming language, SikuliX framework and other supportive tools, which are introduced in this thesis.

Automation of smoke test cases should decrease time spend for testing and thereby speed up development process and improve product quality.

As a result a set of automated test cases were implemented, which cover a significant part of required automation smoke test cases. Every test case has its own story with explanations. The author also presents introduction to SikuliX framework with its supportive tools. It has been concluded that implemented automation of smoke test cases and cooperation between SikuliX framework with its supportive tools work as expected and ready for further development.

The thesis is in Estonian and contains 61 pages of text, 6 chapters, 16 figures, 2 tables.

Lühendite ja mõistete sõnastik

Agiilne tarkvaraarendus	***Agile Software Development*** Agiilne tarkvaraarendus on ka inkrementaalne mudel, kus tarkava luuakse kiiretes inkrementaalsetes tsüklites nii, et igale tsüklile liidetakse uus väiksem funktsionaalsus [22].
API	***Application Programming Interface*** Rakendusliides ehk programmiliides on reeglistik olemasoleva valmisprogrammiga suhtlemiseks. See võimaldab luua ühenduse erinevate programmide vahel [23].
Clojure	***Clojure*** Funktsionaalne programmeerimiskeel, mis on mõelnud töötamiseks koostöös Javaga [24].
Flash	***Flash*** Tarkvara, mis on mõelnud vektorite animeerimiseks. Algselt kavandatud animatsioonide loomiseks veebilehtedel [25].
HTML	***Hyper Text Markup Language*** See on keel, milles märgendatakse veebilehti [26].
IDE	***Integrated Development Environment*** Tarkvara, mis on mõeldud tarkvara arendajatele ning mis on varustatud vajalike osadega tarkvararakenduste loomiseks [27].
Intellij IDEA	***Intellij IDEA*** Java Integreeritud arenduskeskkond, mis pakub arendajale võimaluse luua ning kompileerida tarkvara ning kasutada emulaatoreid [28].
Java	***Java*** Kõrgetasemeline programmeerimiskeel, mis on mõeldud spetsiaalselt kasutamiseks Interneti hajuskeskkonnas [29].

JNI ***Java Native Interface***
Võimaldab välja kutsuda Javast teistes keeltes realiseeritud programme ja teistes keeltes realiseeritud programmides välja kutsuda Javat [30].

JRuby ***JRuby***
Ruby programmeerimiskeel, mis on disainitud Java platvormil jooksutamiseks [31].

Jython ***Jython***
Python programmeerimiskeel, mis on disainitud Java platvormil jooksutamiseks [32].

Maven ***Maven***
Automaatne ehitamise vahend, mida kasutatakse koos Javaga [33].

MTT ***Multi Table Tournament***
Mitme laua turniir kindla algusajaga.

OpenCV ***Open Source Computer Vision***
Meetodite raamatukogu, eesmärgiga reaalajas arvutipiltide nägemiseks ja töötlemiseks [34].

POM ***Project Object Model***
See on XML fail, mis sisaldab informatsiooni projekti kohta, mida kasutab Maven projekti ehitamiseks [35].

Python ***Python***
Programmeerimiskeel ning samas ka programm, mis interpreteerib Python keeles kirjutatud programme [36].

Ruby ***Ruby***
Integreeritav programmeerimiskeel. Tegu on objektorienteeritud keelega, kõiki andmetüüpe käsitletakse objektidena [37].

Scala ***Scala***
See on uudne ja omapärane programmeerimiskeel, mis ühendab endas mitut programmeerimise paradigmat: funktsionaalprogrammeerimist ja

	objekt-orienteeritud programmeerimist. Samuti paistab silma veel sellega, et jookseb (peamiselt) Java Virtual Machine'il ja on ühilduv Javaga ehk eksisteerivaid Java teeke on võimalik Scalas otseselt kasutada [38].
Scrum	***Scrum*** See on iteratiivne ja kasvava populaarsusega agiilse tarkvara arendamise raamistik.
Skript	***Script*** Käsujada, mida täidetakse ilma kasutajapoolse vahelesegamiseta [39].
SNG	***Sit & Go*** Turniir, millel pole kindlat algusaega. Mäng algab niipea, kui laud on täis.
Tesseract OCR	***Tesseract Optical Character Recognition*** Kõige täpsem tasuta teksti otsimiseks ja konverteerimiseks mõeldud mootor [40].
UI	***User interface*** Üldise mõistena on kasutajaliides vahend, mis võimaldab inimesel suhelda masinaga ehk kõik, mida näeme, kuuleme, tunneme [41].
XML	***Extensible Markup Language*** See on W3C väljatöötatud ja soovitatud standardne üldotstarbeline märgistuskeel, mille eesmärgiks on struktureeritud info jagamine infosüsteemide vahel, eelkõige Interneti veebipõhistes rakendustes [42].

Jooniste nimekiri

Joonis 1. Unibet Poker arvutirakendus. Fuajee leht 15
Joonis 2. Unibet Poker rakendus. Fuajee lehe struktuur 15
Joonis 3. Unibet Poker arvutirakendus. Fuajee lehe vasak menüü 16
Joonis 4. Unibet Poker arvutirakendus. Rahamäng Texas Hold'em laud 17
Joonis 5. Unibet Poker arvutirakendus. Minu Profiil leht 18
Joonis 6. Unibet Poker arvutirakendus. Minu Profiil lehe struktuur 19
Joonis 7. Unibet Poker arvutirakendus. Identiteedi loomise leht 20
Joonis 8. Unibet Poker arvutirakendus. Lauatapeedi valiku leht 20
Joonis 9. Kasutusloo diagramm 21
Joonis 10. SikuliX IDE kasutajaliides 33
Joonis 11. SikuliX Java API logimise näide 33
Joonis 12. SikuliX tööstruktuur [4] 34
Joonis 13. Arvutiplatvormi SikuliX Java API FindFailed näide 36
Joonis 14. TestNG XML dokumendi näide 39
Joonis 15. TestNG testi väljundi näide 39
Joonis 16. Arvuti-, veebi- ja mobiilplatvormide klasside koostöö struktuur 41

Tabelite nimekiri

Tabel 1. Nõuded automaattestide töökeskkonnale.....22
Tabel 2. TestNG annotatsioonid ja atribuudid.....38

Sisukord

1. Sissejuhatus 12
1.1 Taust ja probleem 12
1.2 Ülesande püstitus 12
1.3 Metoodika 13
1.4 Ülevaade tööst 13
2. Testitav rakendus ja nõuded automaattestide töökeskkonnale 14
2.1 Testitava rakenduse lühitutvustus 14
2.2 Testitava rakenduse struktuuri lühikirjeldus 14
2.2.1 Fuajee leht 15
2.2.2 Minu Profiil leht 18
2.3 Testitava rakenduse põhifunktsionaalsus 21
2.3.1 Otstarve 21
2.3.2 Tegutsejad 21
2.3.3 Kasutusloo diagramm 21
2.4 Nõuded automaattestide töökeskkonnale 22
3. Suitsutestimise lühitutvustus ja testilugude kirjeldus 23
3.1 Suitsutestimine ja selle roll projektis 23
3.2 Väljavalitud testilugude kirjeldus 23
3.2.1 Testilood arvutiplatvormi jaoks 24
3.2.2 Testilood veebiplatvormi jaoks 26
3.2.3 Testilood mobiilplatvormi jaoks 28
4. Tööriistade ülevaade 31
4.1 SikuliX tutvustus 32
4.1.1 Kuidas SikuliX töötab 34
4.1.2 Kuidas SikuliX otsib ekraanipilte ekraanil? 35
4.1.3 Kui palju aega kuulutab SikuliX otsimise peale? 36
4.2 Selenium tutvustus 37
4.3 Mobizen tutvustus 37
4.4 TestNG raamistiku tutvustus 37
5. Automaattestide loomine 40

5.1 Klasside koostöö struktuur 40
5.2 Lahendus arvutiplatvormile 41
5.2.1 Esimene testilugu 44
5.3 Lahendus veebiplatvormile 47
5.4 Lahendus mobiilplatvormile 48
5.5 SikuliX Java API tagasiside praktikast 50
5.5.1 Eelised 50
5.5.2 Puudused 51
5.6 Järeldused 51
5.7 Edasised plaanid 52
6. Kokkuvõte 53
Summary 54
Kasutatud kirjandus 56
Lisa 1 61

1. Sissejuhatus

Tänapäeval on tarkvaralahendustel oluline roll ühiskonna muutumises ja kujunemisel. Aina rohkem teenuseid, mida varasemalt teostati klienditeenindaja abiga, tehakse kasutades erinevaid tarkvaralahendusi. Võidavad mõlemad, nii ettevõte kui ka klient. Et selline koostöö toimiks, peaks pakutav tarkvaratoode võimaldama teostada toiminguid vastavalt lubatule. Siin on suureks abiks tarkvara testimine.

Tarkvara testimine on väga oluline valdkond iga tarkvaratoote arendusetapis, mille eesmärgiks on aidata teenusepakkujal viia oma tarkvaratoode pakkumiseks vajalikule kujule. Testimine peab olema võimalikult efektiivne ning sellepärast on õige testimise plaan ja tehnika väga olulised.

Testimise protsessi hoolikas planeerimine annab meile edumaa. Aina rohkem räägitakse testimise protsessi automatiseerimisest, sest just see peaks parandama protsessi efektiivsust. Siin on omad väljakutsed, millega peab kindlasti arvestama. Tehes seda mõõdukalt ja õigesti planeeritult, on võimalik oluliselt parandada toote arendusprotsessi ja kvaliteeti.

1.1 Taust ja probleem

Ettevõttes, kus töötan on kasutusel manuaalne testimine ja tarkvaraarendus toimub vastavalt agiilsele metoodikale. Toote nõuded muutuvad pidevalt ja väljalaskmisperiood on iga kuu tagant.

Töötades tarkvaratestijana rakendusega, mis omab suurt funktsionaalsust, olen seismas silmitsi probleemiga, kus on väga raske jõuda õigeaegselt kõike vastavalt vajadustele testida. Selle tulemusena kannatab toote kvaliteet ja klientide rahuolu langeb. Protsessi parendamise eesmärgil tekkis soov automatiseerida testimist alustades suitsutestidest. Automaattestide loomine oleks siin esimeseks sammuks testimise automatiseerimise raamistiku ülesehitamisel.

1.2 Ülesande püstitus

Käesoleva bakalaureusetöö eesmärgiks on väljavalitud suitsutestide automatiseerimine arvuti-, veebi- ja mobiilplatvormidel Flash rakenduse jaoks kasutades SikuliX raamistikku ja teda abistavaid tööriistu.

1.3 Metoodika

Töö eesmärgi saavutamiseks:

- Analüüsitakse testitavat rakendust ja luuakse kriteeriumid automaattestide töökeskkonnale.
- Valitakse välja 5 kõige sobivamat suitsutesti koos operatsioonisüsteemidega ja brauseriga iga platvormi jaoks.
- Iga väljavalitud suitsutesti tarvis koostatakse testilugu.
- Analüüsitakse SikuliX raamistikku ja teda abistavaid teisi tööriistu.
- Automatiseeritakse 5 väljavalitud suitsutesti iga platvormi jaoks.

1.4 Ülevaade tööst

Töö on jaotatud neljaks osaks. Esimeses osas tutvustatakse testitavat rakendust ja nõudeid automaattestide töökeskkonnale.

Teises osas antakse ülevaade suitsutestimisest ja selle rollist projektis. Esitletakse viieteistkümne väljavalitud suitsutesti testilood automatiseerimiseks.

Kolmandas osas tutvustatakse rakenduse testimise automatiseerimise tehnoloogiaid. Räägitakse lähemalt SikuliX raamistikust ja teda abistavatest tööriistadest.

Neljandas osas esitletakse töö praktilises osas koostatud automaattestide esimese testiloo põhjalikku kirjeldust arvutiplatvormi jaoks. Lahenduse sarnasuse tõttu pole vajadust kõigi automaattestide kirjeldamiseks. Lisaks esitletakse ühte olulist klassi, mis erineb igal platvormil. Osa lõpus räägitakse SikuliX raamistikuga töötamise muljetest. Samuti tuuakse välja järeldused SikuliX raamistiku ja teda abistavate tööriistade kohta. Veel annan ülevaate edasistest plaanidest.

Antud lõputöö koostaja on varasemalt töötanud manuaalse testijana ning automatiseerimisega kokkupuutunud vaid põgusalt. Seetõttu vaadatakse vahendeid ka alustava automatiseerija pilgu läbi.

2. Testitav rakendus ja nõuded automaattestide töökeskkonnale

Käesolevas peatükis tutvustakse testitavat rakendust lähemalt. Esitatakse lühikirjeldus rakenduse struktuurist ja tutvustatakse põhifunktsionaalsust. Samuti lisatakse nõuded automaattestide töökeskkonnale.

2.1 Testitava rakenduse lühitutvustus

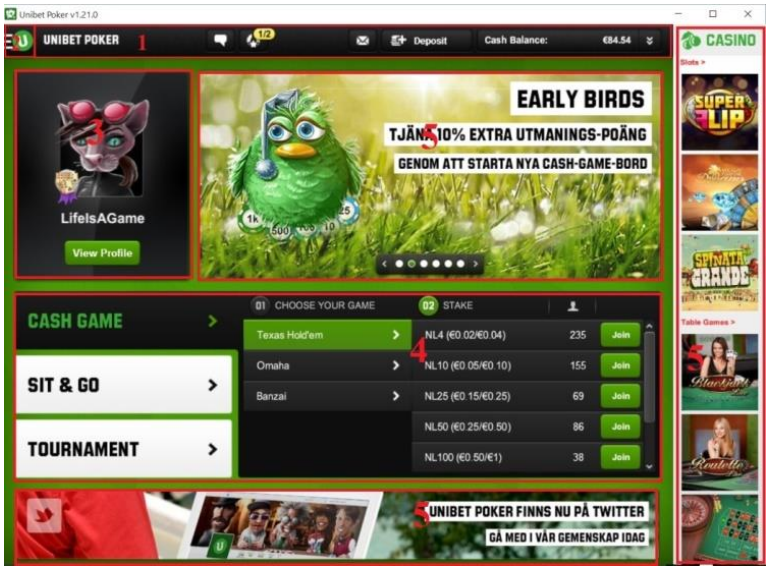
Töökohas arendatakse Unibet Poker rakendust. Projektis kasutatakse Scrum raamistikku ning projekt sai alguse 2013 aasta sügisel. Unibet Poker on Relax Gamitng Ltd. poolt loodud eraldiseisev pokkeritarkvara, mis on tasuta kättesaadav töölauakliendina Windows kui ka OSX operatsioonisüsteemile, brauseripõhise versioonina Windows'is kui ka OSX'is ja tahvelarvutitele Android ja iOS operatsioonisüsteemidele. Tegemist on Flash rakendusega [2]. Rakendus on kättesaadav Unibet'i kodulehelt ja Apple App Store'is.

Unibet Poker võimaldab sul nautida unustamatut elamust ning mängida hulgaliselt mänge teiste mängijate vastu. Siin on olemas suurepärane lojaalsussüsteem, liitumisboonus ja palju muud [3].

2.2 Testitava rakenduse struktuuri lühikirjeldus

Siin tutvustatakse rakenduse kahte peamist lehte - Fuajee (Main Lobby) ja Minu Profiil (My Profile). Esitletakse lehtede näidised koos struktuuri kirjeldusega. Lehtede näidised on välja toodud inglise keeles, sest rakendusel puudub eesti keele võimalus. Lehe struktuur on esitatud eesti keeles. Rakenduse teistest lehtedest räägitakse lühidalt.

2.2.1 Fuajee leht



Joonis 1. Unibet Poker arvutirakendus. Fuajee leht

Joonisel 1 vasakul pool numbriga 3 on tähistatud kasutaja identiteet ’LifeIsAGame’. Üleval pool keskel numbriga 5 on märgitud bannerite koht, kus praegu on käimas ’Early Birds’ kampaania. Paremal nurgas ja kõige alumisel ribal on numbriga 5 tähistatud samuti koht reklaamide jaoks. Bänneri all keskel numbriga 4 on tähistatud koht, kus on võimalik valida ja liituda mänguga.

Joonisel 2 on esitletud rakenduse Fuajee lehe struktuur eesti keeles.

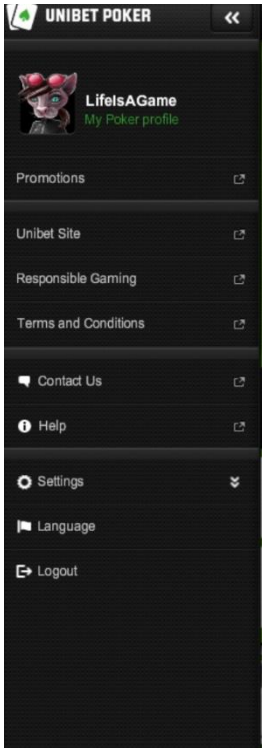
Fuajee leht
- Ülemine riba
  - Teated
  - Missioonid
  - Sõnumid, deposiit ja kontoseis
- Vasak menüü
  - Erinevad lingid
  - Seaded
  - Keelte vahetus
- Identity koht
  - Minu Profiil lehe avamine
- Mänguvalija
  - Rahamängud
  - SNG mängud
  - MTT mängud
- Bännerite koht

Joonis 2. Unibet Poker rakendus. Fuajee lehe struktuur

Fuajee leht koosneb ülemisest ribast, vasakust menüüst, mänguvalijast ja bänneri kohtadest.

Ülemisel ribal joonisel 1 numbriga 1 asuvad teated, missioonid, sõnumid, deposiit ja kontoseis. Sama riba on olemas ka Minu Profiil lehel, millest on lähemalt räägitud alapeatükis 2.2.2.

Vasak menüü avaneb vajutades nuppu, mis on Joonisel 1 märgitud numbriga 2. Vasak menüü on välja toodud joonisel 3 allpool. Sama menüü on olemas ka Minu Profiil lehel, millest on täpsemalt räägitud alapeatükis 2.2.2.

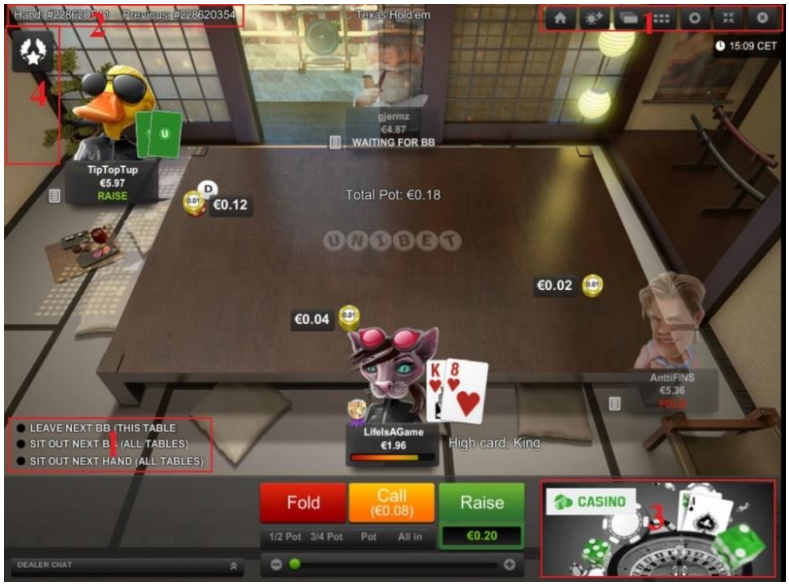


Joonis 3. Unibet Poker arvutirakendus. Fuajee lehe vasak menüü

Menüü koosneb paljudest vajalikest linkidest ja seadetest. Samuti on võimalik vahetada rakenduse keelt.

Mänguvalija asub Fuajee lehe keskel. Siin on võimalik valida kolme erineva mängutüübi vahel. Rahamängudest on võimalik mängida Texas Hold'em, Omaha või Banzai.

Järgnevalt joonisel 4 esitletakse näidet rahamängu Texas Hold'em lauast.



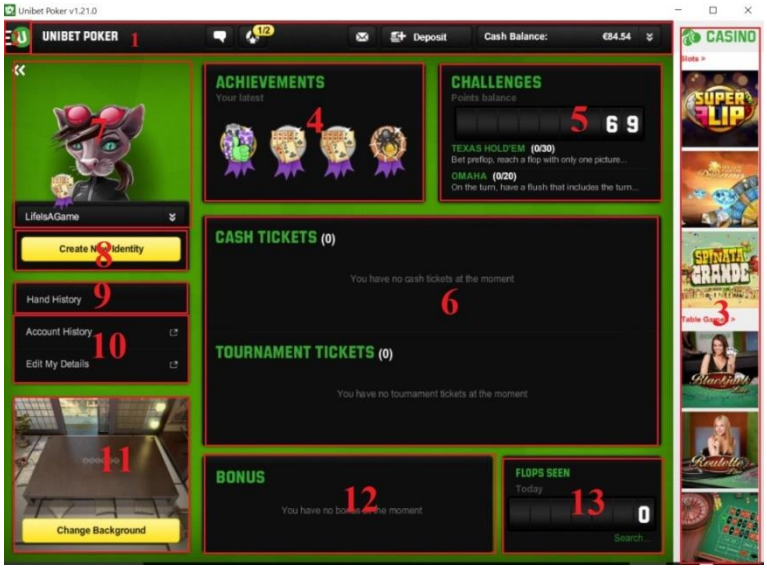
Joonis 4. Unibet Poker arvutirakendus. Rahamäng Texas Hold'em laud

Üleval paremal ja vasakul all numbriga 1 on tähistatud erinevad seadistused alustades laua suuruse muutmisest ja lõpetades raha lisamisega. Üleval vasakul numbriga 2 on tähistatud numbrilised lingid, mis näitavad käimasoleva ja eelmise käe ajalugu. Käe numbrite järgi on võimalik hiljem vaadata mängude ajalugu. Paremal all numbriga 3 on märgitud bänneri koht. Vasakul numbriga 4 on tähistatud missioonide koht. Mängijal on olemas igakuised missioonid, uue mängija missioon ja teised erinevad kampaaniate raames toimuvad missioonid. Täites missioone on võimalik võita auhindu. Näiteks uus pilet või raha.

SNG on turniirid, millel pole kindlat algusaega. Mäng algab niipea, kui laud on täis. Kättesaadavad on kahe või viie inimesega mängud.

MTT mängud ehk mitme laua turniirid. Lai valik erinevaid turniire: freerollid, garanteeritud auhindadega turniirid koos jackpott'idega, Unibet Open ja palju muud.

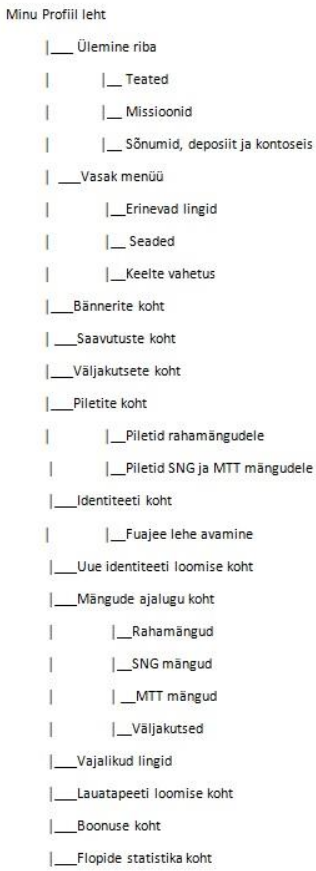
2.2.2 Minu Profiil leht



Joonis 5. Unibet Poker arvutirakendus. Minu Profiil leht

Joonisel 5 vasakul pool numbriga 7 on tähistatud kasutaja identiteet 'LifeIsAGame'. Koht, kus on võimalik luua uus identiteet on märgitud numbriga 8. Kasutaja saavutuste ja väljakutsete leht on markeeritud numbriga 4 ja 5. Olemasolevad piletid mängudele asuvad keskel numbriga 6 tähistatud alas. Mängude ja floppide ajalugu koos teiste vajalike linkidega asuvad vasakul ja keskel numbrite 9, 13 ja 10 juures. Kui kasutajal on boonuseid, siis need asuvad keskel alumisel osal (tähistatud numbriga 12). Lauatapeedi vahetamisvõimalus asub vasakul all (joonisel tähistatud numbriga 11).

Järgnevalt joonisel 6 on esitletud Minu Profiil lehe struktuur.



Joonis 6. Unibet Poker arvutirakendus. Minu Profiil lehe struktuur

Minu Profiil leht koosneb ülemisest ribast, vasakust menüüst, bännerite/ saavutuste/väljakutsete kohast. Samuti on olemas pileti, identiteedi, uue identiteedi loomise, mängude ajaloo, lauatapeedi, boonuste ja floppide koht.

Ülemisel ribal on teated, missioonid, sõnumid, deposiit ja kontoseis. See riba on olemas ka Fuajee lehel, millest on räägitud alapeatükis 2.2.1

Vasak menüü avaneb vajutades Joonisel 3 asuvat numbriga 2 tähistatud nuppu. Vasak menüü on kujutletud joonisel 3 alapeatükis 2.2.1. Sama menüü on olemas ka Fuajee lehel, millest on räägitud alapeatükis 2.2.1

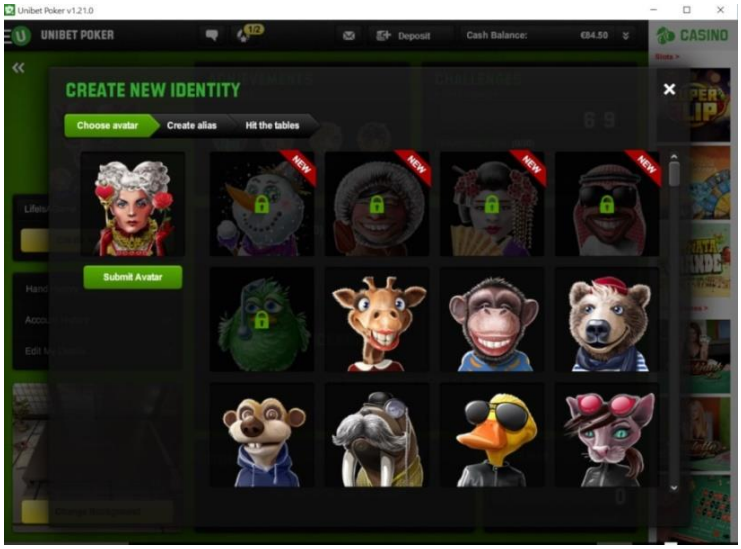
Saavutuste lehel on kasutajal olemas palju erinevaid lahtilukustuvaid saavutusi, mida on võimalik kasutada oma avatari peal.

Väljakutsete leht koos väljakutsetega, mida täites on võimalik saada lisaraha, boonust jpm.

Piletite koht koosneb rahamängude, SNG ja MTT mängude piletitest. Siin on võimalik vaadata olemasolevaid pileteid ja vajadusel ühineda mänguga.

Identiteedi lehel on olemas nupp, mida vajutades avaneb Fuajee leht.

Igal mängijal on võimalik luua uus identiteet koos uue nime ja avatariga. Joonisel 7 on esitatud uue identiteedi loomise leht.

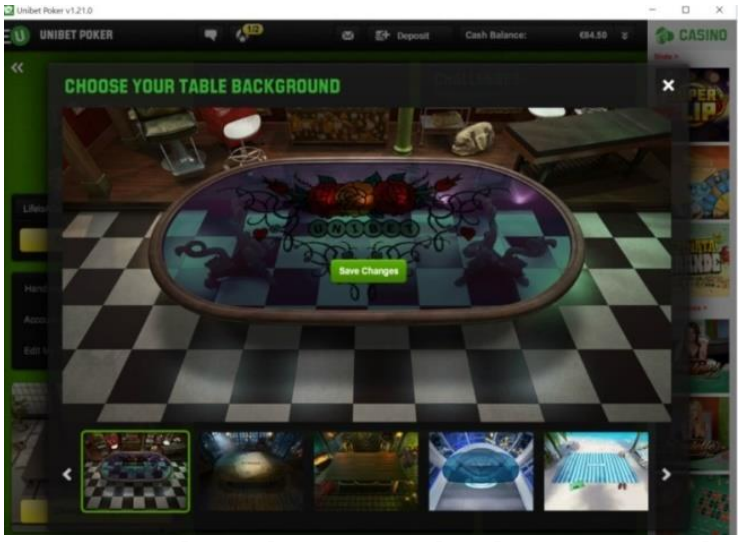


Joonis 7. Unibet Poker arvutirakendus. Identiteedi loomise leht

Siin on võimalik luu endale uus identiteet valides sobiva nime ja pildi. Kokku on võimalik valida 148 avatari vahel.

Mängitud käte ajalugu on võimalik vaadata kõikide mängitud mängude kohta. Samuti on võimalik vaadata tehtud väljakutsete ajalugu.

Mängijal on võimalik muuta lauatausta. Joonisel 8 on esitatud lauatausta muutmise leht.



Joonis 8. Unibet Poker arvutirakendus. Lauatapeedi valiku leht

Hetkel on kättesaadavad 25 erinevat tausta.

2.3 Testitava rakenduse põhifunktsionaalsus

Selles alapeatükis on esitatud testitava rakenduse põhilised funktsionaalsed nõuded koos kasutusloo diagrammiga. Rakenduse mitteolulised funktsionaalsed nõuded, mis ei mõjuta rakendusele kehtestatud reegleid ja rakenduse töötamist on jäetud välja.

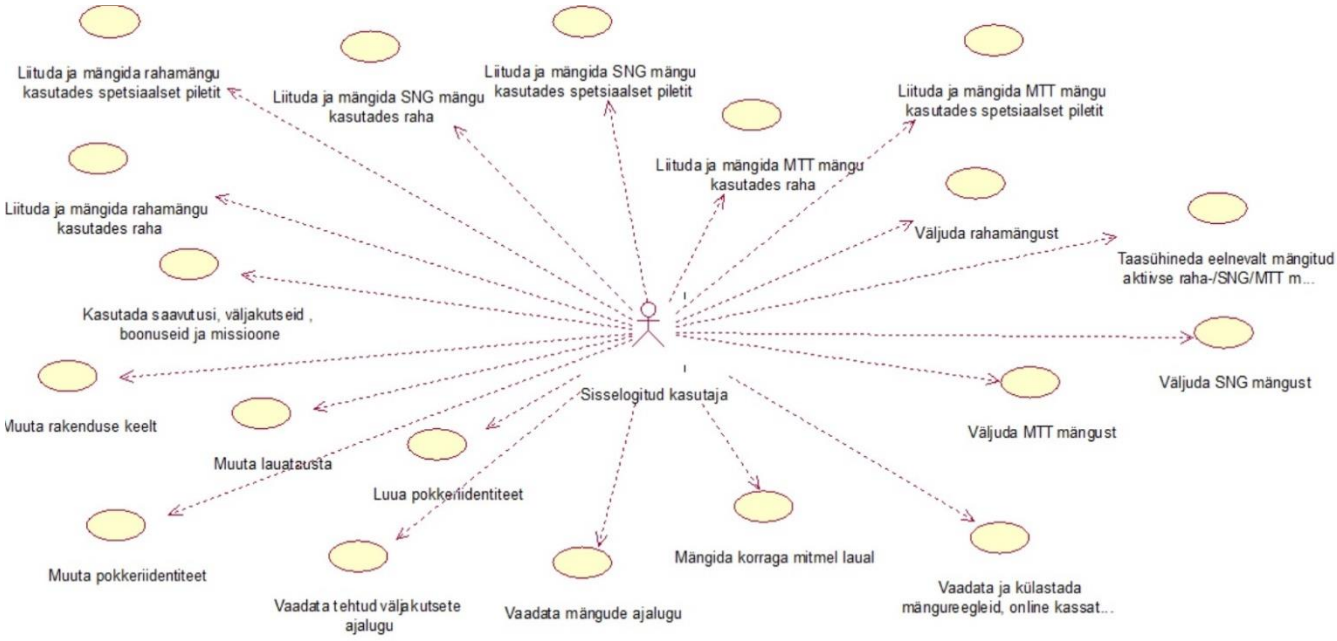
2.3.1 Otstarve

Tutvustada lugejale testitava rakenduse peamisi ja olulisi funktsionaalseid nõudeid.

2.3.2 Tegutsejad

Sisselogitud kasutaja

2.3.3 Kasutusloo diagramm



Joonis 9. Kasutusloo diagramm

Sisselogitud kasutajal peaks olema võimalik :

- liituda ja mängida rahamängu kasutades raha
- liituda ja mängida rahamängu kasutades spetsiaalset piletit
- liituda ja mängida SNG mängu kasutades raha
- liituda ja mängida SNG mängu kasutades spetsiaalset piletit
- liituda ja mängida MTT mängu kasutades raha
- liituda ja mängida MTT mängu kasutades spetsiaalset piletit

- taasühineda eelnevalt mängitud aktiivse raha-/SNG/MTT mängu(de)ga juhul kui rakendus ootamatult lõpetas töötamise
- väljuda rahamängust
- väljuda SNG mängust
- väljuda MTT mängust
- vaadata ja külastada mängureegleid, online kassat ja muid sarnaseid kohti
- mängida korraga mitmel laual
- vaadata mängude ajalugu
- vaadata tehtud väljakutsete ajalugu
- luua pokkeriidentiteet
- muuta pokkeriidentiteet
- muuta lauatausta
- muuta rakenduse keelt
- kasutada saavutusi, väljakutseid, boonuseid ja missioone

Lähtudes testitava tarkvara omadustest, funktsionaalsetest nõuetest ja testimise vajadustest, olen koostanud nõuded automaattestide töökeskkonnale. Nõuded on esitatud järgnevas alapeatükis.

2.4 Nõuded automaattestide töökeskkonnale

Selles alapeatükis esitlen nõudeid loodavale automaattestide töökeskkonnale. Nõuete koostamisel lähtusin testitava tarkvara omadustest ja vajadustest. Samuti olen arvestanud ettevõtte sooviga kasutada vaid tasuta tööriistu ja testija sooviga kirjutada automaatteste Java programmeerimiskeeles.

Tabel 1. Nõuded automaattestide töökeskkonnale

Platvormide toetus	Arvuti, veebi ja mobiil
Operatsioonisüsteemide toetus	Windows 7+, OSX 10.10+, Android 4+ ja iOS 7+
Brauserite toetus	Google Chrome, Firefox, Internet Explorer, Safari
Rakenduste toetus	Funktsionaalne rakendus ja veebirakendus
Flash rakenduse testimise võimalus	Jah
Kasutatavad programmeerimiskeeled	Java
Tööriistade maksumus	Tasuta
Testitava rakenduse lähtekoodile liigipääsu puudumine	Jah

3. Suitsutestimise lühitutvustus ja testilugude kirjeldus

Käesolevas peatükis tutvustakse suitsutestimist ja räägitakse selle rollist projektis. Samuti esitatakse väljavalitud testilugude kirjeldused.

3.1 Suitsutestimine ja selle roll projektis

Suitsutestimine on pinnapealne testimine eesmärgiga kontrollida, et testitava rakenduse põhifunktsioonid töötavad korrektselt. Need testid võiksid olla esimene samm, mida oleks mõistlik automatiseerima hakata [1].

Tavaliselt suitsutestimist tehakse uutele versioonidele ja see võtab umbkaudselt 3-4 tundi aega. Viimase versiooniga viiakse läbi põhjalikum testimine. Probleemiks on see, et tihipeal ei ole võimalik teada, milline versioon on viimane, sest testimise käigus võidakse avastada uusi vigu, mis nõuab taas uue versiooni loomist ning uuesti testimist. Kokkuvõttes, manuaalsele suitsutestimisele kulub väga palju aega ning tekib vajadus automatiseerimise järele.

Suitsutestide komplektist olen välja valinud 5 testilugu iga platvormi jaoks. Väljavalitud testid on oma realiseerimise mõttes erineva keerukusastmega. Samuti need testid esitlevad testitava rakenduse funktsionaalsusest erinevad osad.

Väljavalitud teste automatiseerides on võimalik näidata, et koostöö SikuliX raamistiku ja teda abistavate teiste tööriistade vahel töötab vastavalt planeeritutele. Samuti garanteerib see edu ka ülejäänud suitsutestide automatiseerimiseks.

Kokkuvõtlikult võib öelda, et selline testide arv katab testitava tarkvara põhifunktsionaalsuse erinevad osad ja näitab, et loodud automaattestid ja koostöö SikuliX ning teda abistavate teiste tööriistade vahel töötavad vastavalt vajadustele.

3.2 Väljavalitud testilugude kirjeldus

Järgnevalt esitan väljavalitud suitsutestide jaoks koostatud testilood kolme platvormi tarbeks. Testiloo struktuuri koostamisel ja kirjeldamisel kasutasin Maili Markvardt'i koostatud dokumenti [1].

3.2.1 Testilood arvutiplatvormi jaoks

ID	001
Lühikirjeldus	Testiloo eesmärk on kontrollida, et laua tapeeti on võimalik vahetada
Eeltingimused	Rakendus on avatud, mängija on sisselogitud ja asub fuajees. Rakenduse keel on inglise keel
Sammud	1. Programm ootab ja vajutab nuppu ’View Profile’ 2. Programm ootab ja vajutab nuppu ’Change Background’ 3. Programm ootab 'Choose Your Table Background' teksti ning valib juba etteantud tapeeti 4. Programm ootab ja vajutab ’Save Changes’ nuppu 5. Programm ootab ja paneb kinni ’Change Your Table Background’ lehe vajutades ’X’ 6. Programm kontrollib, et uus tapeet on olemas 7. Programm kontrollib, et Minu Profiili leht on avatud
Oodatav tulemus	Laua tapeet on vahetatud

ID	002
Lühikirjeldus	Testiloo eesmärk on kontrollida, et kasutajal on võimalik enda avatari peale panna saavutus
Eeltingimused	Rakendus on avatud, mängija on sisselogitud ja asub fuajees. Rakenduse keel on inglise keel
Sammud	1. Programm ootab ja vajutab nuppu ’View Profile’ 2. Programm ootab ning vajutab ’Achievements’ teksti peale 3. Programm ootab ’Your achievements ’teksti ning valib juba etteantud saavutuse 4. Programm ootab saavutust avatari peal ning paneb saavutuste lehe kinni vajutades ’X’ 5. Programm ootab saavutust avatari peal Minu Profiil lehel ning vajutab nuppu ’<<’ tagasi fuajeesse 6. Programm otsib saavutust avatari peal 7. Programm kontrollib, et fuajee on avatud

Oodatav tulemus	Saavutus on nähtav kasutaja avatari peal Minu Profiili lehel ja Fuajee lehel

ID	003
Lühikirjeldus	Testiloo eesmärk on kontrollida, et kasutajal on võimalik ühineda ’SNG’ lauaga kasutades spetsiaalset piletit
Eeltingimused	Rakendus on avatud, mängija on sisselogitud ja asub fuajees. Mängijal on olemas spetsiaalne ’SNG’ pilet. Rakenduse keel on inglise keel
Sammud	1. Programm ootab ja vajutab nuppu ’View Profile’ 2. Programm otsib spetsiaalset ’SNG’ piletit ning vajutab selle peale 3. Programm leiab mängu, kuhu ta tahab ühineda ja vajutab ’Join’ nuppu 4. Sisseostu tegemas, Programm vajutab nuppu ’Ticket’ 5. Programm otsib ja leiab, et järjekorra sõnum on olemas
Oodatav tulemus	Kasutaja on ühinenud ’SNG’ mänguga kasutades selleks spetsiaalset piletit

ID	004
Lühikirjeldus	Testiloo eesmärk on kontrollida, et ’SNG’ mäng läheb uuesti käima peale rakenduse kinni panemist
Eeltingimused	Mängija on eelnevalt ühinenud ’SNG’ mänguga ja on järjekorras. Rakenduse keel on inglise keel
Sammud	1. Programm avab rakenduse 2. Programm logib sisse 3. Programm ootab ja kontrollib, et ’SNG’ järjekorra sõnum on olemas
Oodatav tulemus	’SNG’ mäng avaneb uuesti peale rakenduse kinni panemist

ID	005
Lühikirjeldus	Testiloo eesmärk on kontrollida, kas ’SNG’ mängu järjekorrast on võimalik välja minna

Eeltingimused	Mängija on eelnevalt ühinenud ’SNG’ mänguga ja on järjekorras. Rakendus on avatud, mängija on sisselogitud ja SNG laud on avatud. Rakenduse keel on inglise keel
Sammud	1. Programm otsib ja kontrollib, et järjekorra sõnum on olemas 2. Programm otsib ja vajutab ’X’ 3. Programm otsib kinnitamise teadet ja vajutab ’Yes’ 4. Programm ootab väljumise kinnitamise teadet ja vajutab nuppu ’OK’ 5. Programm ootab ja kontrollib, kas Fuajee leht on laetud
Oodatav tulemus	Kasutaja on väljunud ’SNG’ mängu järjekorrast

3.2.2 Testilood veebiplatvormi jaoks

ID	006
Lühikirjeldus	Testiloo eesmärk on kontrollida, et rahamänguga on võimalik liituda kasutades raha
Eeltingimused	Rakendus on avatud, mängija on sisselogitud ja asub fuajees. Rakenduse keel on inglise keel
Sammud	1. Programm ootab ja vajutab nuppu ’Cash Game’ 2. Programm ootab ja vajutab nuppu ’Banzai’ 3. Programm ootab ja kaks korda vajutab nuppu ’NL1’ 4. Programm ootab sisseostu ja vajutab nuppu ’OK’ 5. Programm ootab laua avalehe nuppu ilmumist 6. Programm leiab laua seadistuste nuppu ilmumist
Oodatav tulemus	Kasutaja on liitunud rahamänguga ja rahamängu laud on avatud

ID	007
Lühikirjeldus	Testiloo eesmärk on kontrollida, et kasutaja sisseostu ja fuajee saldod on võrdsed
Eeltingimused	Rakendus on avatud, mängija on sisselogitud ja asub fuajees. Rakenduse keel on inglise keel
Sammud	1. Programm ootab ja leiab sisselogitud kasutaja saldo

	2. Programm jätab meelde sisselogitud kasutaja saldo 3. Programm ootab ja vajutab nuppu ’Cash Game’ 4. Programm ootab ja vajutab nuppu ’Banzai’ 5. Programm ootab ja kaks korda vajutab nuppu ’NL1’ 6. Laua sisseostult Programm ootab ja leiab ’e:’ teksti 7. Programm jätab meelde olemasoleva sisselogitud kasutaja laua saldo 8. Programm võrdleb kasutaja saldosi
Oodatav tulemus	Kasutaja saldod on võrdsed

ID	008
Lühikirjeldus	Testiloo eesmärk on kontrollida, et on võimalik luua uus identiteeti
Eeltingimused	Rakendus on avatud, mängija on sisselogitud ja asub fuajees. Rakenduse keel on inglise keel
Sammud	1. Programm vajutab nuppu ’View Profile’ 2. Programm kontrollib, et ’Achievements’ tekst on olemas ja vajutab nuppu ’Create new identity’ 3. Programm ootab ja vajutab liuguri alguse noole peale 4. Programm vajutab vasaku nuppu hiire peale ning hoiab 10 sekundit 5. Programm vabastab hiire vasaku nupu 6. Programm ootab ja leiab etteantud pildi 7. Programm ootab ja vajutab 'Submit Avatar' nuppu 8. Programm vajutab ’enter alias’ väljasse ning kirjutab ’aliasi’ nime, mis on etteantud 9. Programm ootab ja vajutab ’Submit’ nuppu 10. Programm ootab ja leiab ’Identity created’ teksti 11. Programm vajutab nuppu ’Close’ 12. Programm otsib uue avatari pildi Minu Profiili lehel 13. Programm võrdleb identiteedi nime oodatava tulemusega
Oodatav tulemus	Uus identiteet on loodud

ID	009
Lühikirjeldus	Testiloo eesmärk on kontrollida, et saavutuste filter töötab korrektselt
Eeltingimused	Rakendus on avatud, mängija on sisselogitud ja asub fuajees. Rakenduse keel on inglise keel
Sammud	1. Programm ootab ja vajutab nuppu ’View Profile’ 2. Programm otsib ja vajutab ’Achievements’ teksti peale 3. Programm lohiseb liuguri ülevalt alla 4. Programm otsib ja leiab lukustatud märgi 5. Programm ootab ja vajutab 'all' filtri peale 6. Programm otsib ja vajutab ’Locked’ peale 7. Programm otsib ja leiab lukustatud märgi 8. Programm otsib ja vajutab ’locked’ filtri peale ning valib ’Unlocked’ 9. Programm otsib ja vajutab ’Unlocked’ peale 10. Programm otsib ja ei leiab lukustatud märgi 11. Programm paneb kinni saavutuste lehe vajutades ’X’
Oodatav tulemus	Filter töötab korrektselt

ID	010
Lühikirjeldus	Testiloo eesmärk on kontrollida, et on võimalik avada Minu Profiil leht
Eeltingimused	Rakendus on avatud, mängija on sisselogitud ja asub fuajees. Rakenduse keel on inglise keel
Sammud	1. Programm ootab ja vajutab nuppu ’View Profile’ 2. Programm ootab ’View Profile’ nuppu 3. Programm otsib ’Achievements’ teksti
Oodatav tulemus	Minu Profiil leht on avatud

3.2.3 Testilood mobiilplatvormi jaoks

ID	011
Lühikirjeldus	Testiloo eesmärk on kontrollida, et rahamänguga on võimalik liituda kasutades spetsiaalset piletit

Eeltingimused	Rakendus on avatud, mängija on sisselogitud ja asub fuajees. Rakenduse keel on inglise keel
Sammud	1. Programm ootab ja vajutab kaks korda nuppu ’NL4’ 2. Sisseostul Programm vajutab nuppu ’Ticket’ 3. Programm ootab ’Buy chips (Ticketi)’ teksti 4. Programm vajutab ’OK’ nuppu 5. Programm ootab kodulehe nuppu 6. Programm otsib ’X’ nuppu
Oodatav tulemus	Kasutaja on liitunud rahamänguga ja rahamängu laud on avatud

ID	012
Lühikirjeldus	Testiloo eesmärk on kontrollida, et rahamängu laud läheb uuesti käima peale rakenduse kinni panemist
Eeltingimused	Mängija on eelnevalt ühinenud rahamänguga ja mängib. Rakenduse keel on inglise keel
Sammud	1. Programm avab appi 2. Programm logib sisse 3. Programm ootab kodulehe nuppu 4. Programm otsib ’X’ nuppu
Oodatav tulemus	Rahamängu laud on avatud

ID	013
Lühikirjeldus	Testiloo eesmärk on kontrollida, kas aktiivsest rahamängust on võimalik välja minna
Eeltingimused	Mängija on eelnevalt ühinenud rahamänguga, laud on avatud ja mäng on aktiivne . Rakenduse keel on inglise keel
Sammud	1. Programm ootab kodulehe nuppu 2. Programm otsib ’X’ nuppu 3. Programm ootab ja vajutab ’X’ nuppu peale 4. Programm ootab väljumise kinnitamise teadet ja vajutab nuppu

	’Yes’ 5. Programm ootab ja kontrollib, et Fuajee leht on laetud
Oodatav tulemus	Kasutaja on väljunud rahamängust ja asub fuajees

ID	014
Lühikirjeldus	Testiloo eesmärk on kontrollida, et on võimalik avada ’Full Tournament Schedule’ leht
Eeltingimused	Rakendus on avatud, mängija on sisselogitud ja asub fuajees. Rakenduse keel on inglise keel
Sammud	1. Programm ootab ja vajutab nuppu ’Tournament’ 2. Programm ootab ja vajutab’ View Full Tournament Schedule’ lingi peale 3. Programm otsib mängijate ikooni 4. Programm otsib ’Full tournament schedule’ teksti
Oodatav tulemus	’Full Tournament Schedule’ leht on avatud

ID	015
Lühikirjeldus	Testiloo eesmärk on kontrollida, et on võimalik vahetada rakenduse keel vene keele vastu
Eeltingimused	Rakendus on avatud, mängija on sisselogitud ja asub fuajees. Rakenduse keel on inglise keel.
Sammud	1. Programm ootab ja vajutab nuppu ’U’ 2. Programm ootab ja vajutab ’Language’ nupu peale 3. Programm ootab ja vajutab ’English’ nupu peale 4. Programm ootab ja vajutab ’Русский’ nupu peale 5. Programm ootab ja vajutab ’OK’ nuppu 6. Programm otsib ja leiab ’Игра на деньги’ nupu
Oodatav tulemus	Rakenduse keel on vene keel

4. Tööriistade ülevaade

Käesolevas peatükis tutvustatakse suitsutestide automatiseerimiseks kasutatud tehnoloogiaid. Põhitööriista – SikuliX'i on kirjeldatud põhjalikumalt. Samuti on räägitud tema olemusest, kasutamise võimalustest ja tööpõhimõtetest. Teisi põhitööriista abistavaid tehnoloogiaid (Selenium, Mobizen ja TestNG) on kirjeldatud lühidalt.

SikuliX' põhitööriistaks valimise põhjuseks osutus tema võime töötada Windows ning OSX operatsioonisüsteemidel ja testida kõike, mida on kujutletud arvutiekraanil (kaasarvatud Flash rakendusi).

Selenium on populaarne veebiautomatiseerimise raamistik, mida olen kasutanud veebiplatvormi automaattestide loomisel. Selenium'i töösse valiku põhjuseks osutus tema võime töötada koos peaaegu kõikidega brauseritega. Tema eesmärgiks oli avada testitav rakendus Chrome brauseris. SikuliX on samuti võimeline seda tegema, kuid see võtaks palju rohkem aega võrreldes Selenium'iga.

SikuliX ei ole võimeline töötama reaalsel füüsilisel mobiilsel seadmel ja see oli põhjuseks, miks valiti Mobizen abitööriistaks.

Mobizen rakendus on suurepärane lahendus Android mobiilidele ekraanipildi peegeldamiseks arvutiekraanile. See rakendus oli abiks mobiilplatvormi automaattestide loomisel. Mobizen rakenduse töösse valiku põhjuseks osutus selle võime peegeldada mobiilekraani arvutiekraanile ja võimalus kontrollida mobiiliekraani arvutiekraanil. Lisaks selle suurepärane kiirus, mis on siinkohal samuti väga oluline. Tema eesmärgiks oli peegeldada Android tahvelarvuti ekraani arvutiekraani peale. Tänu sellele oli võimalik testida ja kontrollida testitavat rakendust mobiili peal otse arvutist.

TestNG raamistik oli suurepäraseks vahendiks koodi struktureerimiseks terve raamistiku jaoks. Selle raamistikku töösse valiku põhjuseks oli kasutajasõbralikus ja multifunktsionaalsus. Tänu sellele sain kasutada töös erinevad annotatsioone koos atribuutidega ja luua testikomplekte kasutades XML formaadis faile. Nende abiga sain ma lihtsamal moel määrata, mis järjekorras testid algavad ja kohandada testikomplekte. Need võimalused on suureks abiks automaattestide läbiviimisel.

4.1 SikuliX tutvustus

SikuliX (käesolevas töös kasutan seda nimetust) või Sikuli Script ja Sikuli IDE on avatud lähtekoodiga tarkvara ning see on tasuta kättesaadav allalaadimiseks. SikuliX on võimeline automatiseerima kõike, mida te näete oma ekraani peal kasutades selleks ekraanipilte. UI elementide tuvastamiseks ja kontrollimiseks kasutan OpenCV võimalust**.** See on suurepärane lahendus, kui puudub liigipääs testitava tarkvara koodile [7].

SikuliX on väga võimekas vahend. Ta toetab skriptilisi keeli nagu näiteks Python 2.7 ja Ruby 1.9 ja 2.0. Samuti on võimalik kasutada teda koos Java programmeerimiskeelega ja koos Java skriptimise keeltega (nagu näiteks: Jython, JRuby, Scala, Clojure jpt) [7].

Vahend on samuti võimeline suhtlema arvutihiire ja klaviatuuriga [7].

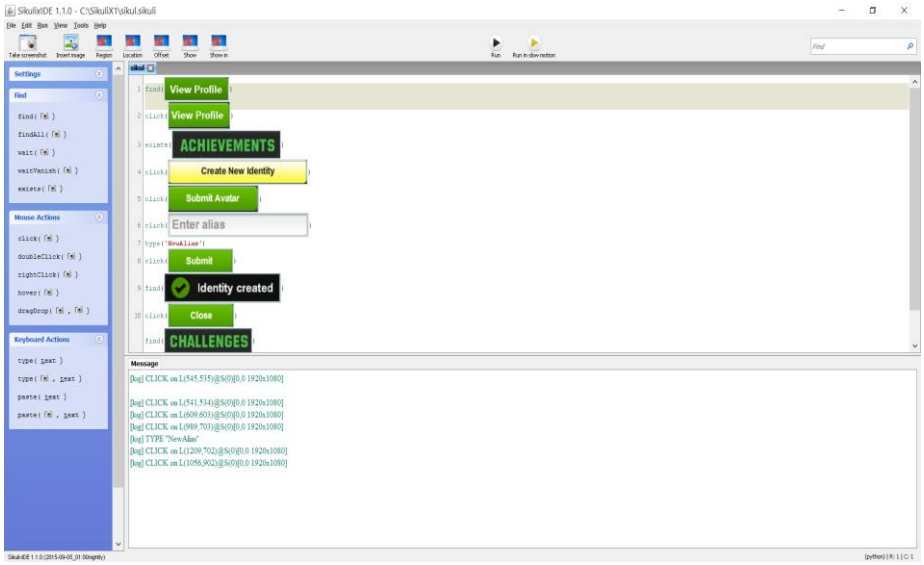
Tänu OCR Tesseractile oskab SikuliX otsida teksti ekraanipiltide seest [7].

SikuliX on Java tarkavara, mis töötab Windows XP+, Mac 10.6+ ja enamustel Linux/Unix süsteemidel [7].

Seda vahendit on võimalik kasutada mitmel viisil. SikuliX (Sikuli) IDE ja SikuliX (Sikuli Script) Java API nimetatakse SikuliX'iks. Sikuli Java API on sarnane SikuliX'iga, kuid veidi erinev oma kasutamise auditooriumilt. Nendest võimalustest on räägitud allpool [7].

SikuliX IDE

SikuliX IDE on tasuta keskkond, kus on võimalik luua skripte lihtsamal moel kasutades selleks ekraanipilte. Testija ei pea oskama programmeerida. Kõik tegevused on kättesaadavad vajutades vajalikku nuppu [8].



Joonis 10. SikuliX IDE kasutajaliides

Vasakul menüüs on tegevused, mida tahetakse teha testitava tarkvaraga. Üleval menüüs on tegevused, mis aitavad üle anda skriptile ekraanipildi, mida on vaja testi sooritamiseks.

SikuliX Java API

SikuliX Java API on ettenähtud kasutamiseks koos Java programmeerimiskeelega või koos mõne Java põhineva programmeerimiskeelega nagu näiteks: Jython, JRuby, Scala, Clojure jne [5].

Projektis on võimalik kasutada importides seda raamatukoguna või maven projektides, importides pom failis [5].

SikuliX logib igat sammu, kuhu ta vajutab. Logimise näide on toodud joonisel 7.

```
[log] App.open [10032:]
[log] CLICK on L(679,370)@S(0)[0,0 1920x1080]
[log]  TYPE "ko2"
[log] CLICK on L(561,473)@S(0)[0,0 1920x1080]
[log] CLICK on L(216,482)@S(0)[0,0 1920x1080]
[log] CLICK on L(215,991)@S(0)[0,0 1920x1080]
[log] CLICK on L(727,917)@S(0)[0,0 1920x1080]
[log] CLICK on L(733,538)@S(0)[0,0 1920x1080]
[log] CLICK on L(1298,193)@S(0)[0,0 1920x1080]
[log] App.close: [10032:Unibet Poker.exe]
```

Joonis 11. SikuliX Java API logimise näide

L tähendab asukohta ja sulgudes on koordinaadid (x,y) kuhu SikuliX on vajutanud. *S* tähendab arvutiekraani. *S(0)* tähendab põhimonitori ehk kui meil on ainult üks monitor. Reeglina see on ka põhimonitor. Kui meil on 2 monitori, siis üks on põhimonitor *S(0)* ja teine

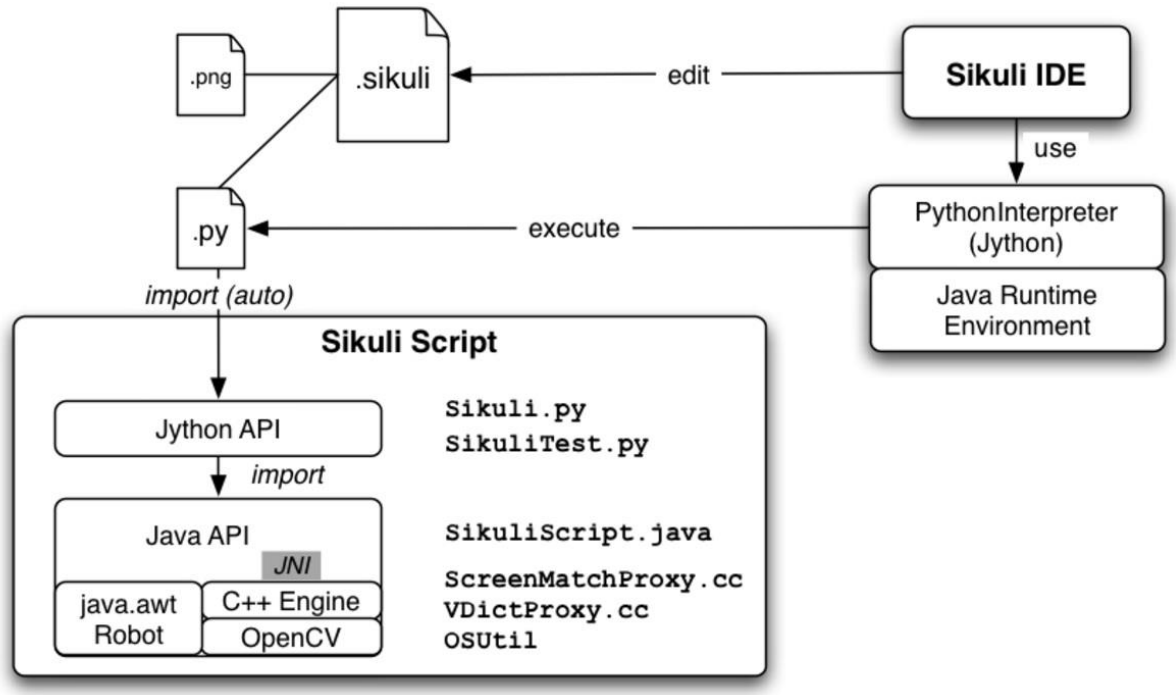
esimene monitor ehk *S(1).* Nurksulgudes on regioon ehk meie antud arvutiekraan, kus test toimub [17, 18, 20].

Sikuli Java API

Sikuli Java API on mõeldud kasutamiseks Java programmeerijate jaoks. See on tehtud selleks, et oleks võimalik kasutada SikuliX võimalusi otse Java programmides, mitte kirjutada Jython skripte [6].

Selles raamatukogus on uuendatud API ja palju uusi võimalusi, mis polnud esialgses SikuliX'is. Näiteks: võimalus sobitada värve, käsitleda sündmuseid ja leida geomeetrilisi mustreid nagu näiteks ristkülikukujulisi nuppe. Kättesaadav importides raamatukoguna ja maven projektides [6].

4.1.1 Kuidas SikuliX töötab



Joonis 12. SikuliX tööstruktuur [4]

SikuliX on Jython ja Java raamatukogu, mis võimaldab koostöö ekraanipiltide ja klaviatuuri ning hiire vahel [4].

SikuliX'i tuum on Java raamatukogu, mis koosneb kahest osast: *java.awt.Robot,* mis vastutab klaviatuuri ja hiire suhtlemise eest ja *C++ mootor,* mis põhineb OpenCV'el, mis otsib antud ekraanipilti ekraanil [4].

C++ mootor on ühendatud Javaga läbi *JNI* ja vajab kompileerimist iga platvormi jaoks [4].

Java raamatukogu tipus on Jyhtoni kiht, mis on tehtud lõppkasutajate jaoks. See on kogum lihtsamatest ja arusaadavatest käskudest [4].

Sikuli skript (.sikuli) on kataloog mis koosneb Python lähtefailist (.py). Ta esindab automatiseerimise töövoogu ning sisaldab ekraanipilte, mida kasutab lähtefail [4].

Skripti salvestamisel võib Sikuli IDE teha lisa HTML faili .sikuli kataloogis. Neid faile saavad kasutajad jagada visuaalse koopiana veebi kaudu [4].

Sikuli IDE käivitab ning muudab Sikuli skripte. Sikuli Skripti käivitamiseks loob Sikuli IDE *org.python.util.PythonInterpreter* ja automaatselt annab paar esimest rida interpriteerijale (Sikuli Jython moodulite importimiseks ja tee panemiseks .sikuli kataloogi). Kui see on tehtud, .py skriptid hakkavad jooksma tänu *PythonInterpreter.execfile()* [4].

4.1.2 Kuidas SikuliX otsib ekraanipilte ekraanil?

SikuliX kasutab OpenCV paketti, et leida üks pilt teise pildi sees või ekraanil ja avastada muudatuse antud regioonil ekraani sees [9].

Koht *matchTemplate()* on see, kus me hakkame otsima eesmärgi pilti. Selleks me teeme ekraanipildi defineeritud regiooni sees kasutades Java Robot klassi. See on nüüd baaspilt, kus hakkavad käima otsingud toimuma. Eesmärgi pilt on loodud ning kättesaadav failist. Mõlemad pildid on konverteeritud OpenCV objektideks [9].

Nüüd me käivitame *matchTemplate()* meetodi. Otsimise käigus me liigume üks piksel korraga alustades ülevalt vasakult ja võrdleme saadud tulemusi, kui head või halvad nad on meie eesmärgi jaoks. Iga asukoha (x,y) jaoks me salvestame tulemuse maatriksis R. Lõpuks saame maatriksi baasipildi tarvis, mis sisaldab iga piksli sarnasust võrreldes eesmärgi pildi iga piksliga (alustades ülevalt vasakult) [9].

Iga piksli tulemus varieerub vahemikus 0.0 kuni 1.0 vahel. Mida madalam väärtus, seda väiksem on sarnasus eesmärgipildiga. Tulemus väärtusega 0.7 - 0.8 viitab kõrgele tõenäosusele, et pilt on seal [9].

Järgmisel sammul me kasutame maksimaalse tulemuse saamiseks teist OpenCV meetodi *minMaxloc()*. Meie eesmärk on leida piksel, mille sarnasus on kõige kõrgem. Vaikimisi on seatud, et kui on leitud piksli tulemus > 0.7, siis see tähendab eesmärgipiksli leidmist. Teised tulemused annavad *FindFailed* erandi ja test ebaõnnestub [9].

Kui pilt on leitud luuakse objekt, mis tähistab teatud regiooni, milles sisaldub pilti [9].

Kui pilt pole leitud lõpetame otsimise või alustame uue otsingu uue pildiga. Seda korratakse kuni pilt on leitud või otsimise aeg (vaikimise 3 sekundit) on lõppenud ning tulemus on *FindFailed* ja test ebaõnnestunud [9].

```
FindFailed: can not find P(C:/Users/Natalja/IdeaProjects/SikuliDesktopWindowsTesting/target/classes/RandomAchievementOnPlayerProfile.png) S:
 0.7 in S(0)[0,0 1920x1080]

  Line 2189, in file Region.java

    at org.sikuli.script.Region.handleFindFailedShowDialog(Region.java:2189)
    at org.sikuli.script.Region.handleFindFailed(Region.java:2134)
    at org.sikuli.script.Region.wait(Region.java:2546)
    at Pages.BasePage.waitAndClick(BasePage.java:61)
    at Pages.MyProfilePage.applyAchievement(MyProfilePage.java:134)
    at MyProfilePageTests.AchievementApplyTest(MyProfilePageTests.java:41) <16 internal calls>
    at org.testng.SuiteRunnerWorker.runSuite(SuiteRunnerWorker.java:52)
    at org.testng.SuiteRunnerWorker.run(SuiteRunnerWorker.java:86) <3 internal calls>
    at org.testng.IDEARemoteTestNG.run(IDEARemoteTestNG.java:72)
    at org.testng.RemoteTestNGStarter.main(RemoteTestNGStarter.java:122) <5 internal calls>
```

Joonis 13. Arvutiplatvormi SikuliX Java API FindFailed näide

Joonisel on näha, kuidas SikuliX käitub, kui ta ei suutnud otsitavat pilti leida. *P* sulgudes on välja toodud pilt, mida ei suudetud leida. S tähendab *similarity* ehk sarnasus, millega SikuliX otsis. Sarnasus on siin 0.7. *S(0)* tähendab põhimonitori ning [0,0 1920x1080] tähistab põhimonitori positsiooni ja selle mõõtmeid [18, 20].

4.1.3 Kui palju aega kuulutab SikuliX otsimise peale?

Mida suurem põhipilt on, seda kauem otsing kestab. Kaasaegsetel süsteemidel suurte monitoridega väikese või keskmise suurusega pildi (kuni 10.000 pikslit) otsimisele kuulub aega umbes 0.5-1.0 sekundit või pisut rohkem. Selleks, et vähendada kuluvat aega peaksime vähendama otsingu regiooni ehk põhipilti. Väikesed pildid (näiteks 10 pikslit regioonides suurusega 1000 pikslit) on leitavad umbes 10 millisekundidega või kiiremini [9].

4.2 Selenium tutvustus

Selenium on populaarne veebiautomatiseerimise raamistik. Hetkel on see jagatud kolmeks osaks: Selenium IDE, Selenium Remote Control ja Selenium WebDriver [10].

- Selenium IDE on Firefox lisana töörist, mis kasutab salvestamise ja taasesitamise tehnikat. See lubab lindistada veebis toimingut ja taasesitada neid ja vajadusel konverteerida paljudesse programeerimiskeeltesse [11].
- Selenium Remote Control on API-liides, mis võimaldab kasutajatel kirjutada teste sobivas programeerimiskeeles ja jooksutada neid Javascripti veebilehitsejas [12].
- Selenium WebDriver on API-liides, mis on Selenium Remote Control edasiarendus. See võimaldab juhtida veebilehitsejat väljavalitud programmeerimiskeeles. WebDriveriga toetab brausereid: Google Chrome, Firefox, Internet Explorer, Opera, Safari htmljunit ja phantomJs ning programmeerimiskeelI: Java, C, Ruby, Python, Javascript [13].

Oma töös kasutasin Selenium WebDriver API-liidest veebiplatvormi automatiseerimiseks BaseTest klassis, mis on kirjeldatud peatükis 5.

4.3 Mobizen tutvustus

Mobizen on tasuta Androidi rakendus, mis lubab peegeldada ja kontrollida mobiili või tahvli ekraani otse arvutist. Rakendus on kättesaadav töölauakliendina ja veebikliendina [14].

Rakendus Mobizen kasutasin mobiiliplatvormi automatiseerimiseks BaseTest klassis, mis on kirjeldatud peatükis 5.

4.4 TestNG raamistiku tutvustus

TestNG on testimise raamistik, mille eesmärgiks on koodi struktureerimiseks võimaluste pakkumine. See raamistik omab palju võimalusi. Näiteks annotatsioonide kasutamine, mis jagab teste osadeks ning võimaldab määrata nendele lisakriteeriumi; andmepõhise testimise toetus; XML faili kasutamine teste konfigureerimiseks; teiste rakenduste toetus (näiteks Eclipse, IDEA, Ant, Maven, Netbeans ja palju muud) [15].

Tabelis 2 tutvustatakse TestNG annotatsioone ja atribuute, mida käesoleva töö raames kasutasin.

Tabel 2. TestNG annotatsioonid ja atribuudid

Annotatsioon	Atribuut	Kasutus
@BeforeMethod		Lisatakse meetodile, mis kirjeldab tegevusi, mida tuleb teha enne iga testklassis oleva meetodi käivitamist.
@AfterMethod		Lisatakse meetodile, mis kirjeldab tegevusi, mida tuleb teha peale iga testklassis olevat meetodit
@Test		Kasutatakse testi samme kirjeldava klassi või meetodi määratlemiseks.
	priority	Kasutatakse testi samme kirjeldava meetodi prioritiseerimiseks. Madala prioriteediga meetodid käivitatakse enne

Näide TestNG raamistiku annotatsioonide ja atribuudi kasutamisest on näha peatükis 5, kus kirjeldan BaseTest klassi ja testi. Ülejäänud näited on kättesaadavad lisas 1.

Testide konfigureerimiseks ja käivitamiseks on võimalik kasutada XMLformaadis faile. Siin on võimalik määrata erinevaid seadistusi ning kohandada testikomplekte vastavalt vajadustele [16].

Käesolevas töös olen kasutanud XML-faili, et kirjeldada testikomplekti (XML-märgend <*suite*>) nime ning sisu. Testikomplekt sisaldab teste (XML-märgend <*test*>). Testi sisuks on klassid (XML-märgend <*class*>). Vaikimisi kutsutakse välja klassis kõik annoteeritud meetodid, kuid on samuti võimalik määrata millised meetodid nimetatud klassist kutsutakse (XML-märgend <*methods*> <*include*>) [15].

Joonisel 14 on toodud näide TestNG XML- failist. Näide on pärit töö praktilisest osast arvutiplatvormi automaattestide testikompleti kirjeldamisest.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="My profile page tests" verbose="2">
  <test name="SikuliDesktopWindowsTesting">
    <classes>
      <class name="MyProfilePageTests">
        <methods>
          <include name="achievementApplyTest"/>
          <include name="changeTableBgTest"/>
        </methods>
      </class>
    </classes>
  </test>
</suite>
```

Joonis 14. TestNG XML dokumendi näide

Märgendi atribuut *verbose* reguleerib, kui suur tagasiside tuleb raamistikult sisseehitatud logimissüsteemi [15]. Joonisel olen seadistanud *verbose* = ’2’. Näide väljundist antud seadistuse puhul on toodud välja Joonisel 15.

```
PASSED: AchievementApplyTest
PASSED: ChangeTableBgTest

===============================================
    SikuliDesktopWindowsTesting
    Tests run: 2, Failures: 0, Skips: 0
===============================================

===============================================
My profile page tests
Total tests run: 2, Failures: 0, Skips: 0
===============================================
```

Joonis 15. TestNG testi väljundi näide

Joonisel üleval pool on kirjeldatud kahe testi meetodid koos tulemustega. Joonise keskel on välja toodud testi nimi koos testide arvu ja tulemusega. Kõige all on välja toodud testikompleti nimi koos testide koguarvuga ja tulemusega.

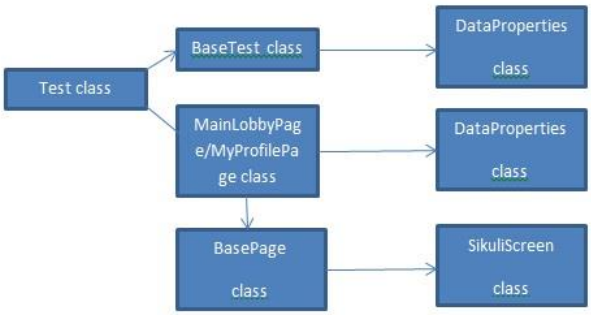
5. Automaattestide loomine

Projekti olemasolevatest suitsutestidest olid välja valitud automatiseerimiseks 5 testi iga platvormi jaoks. Lisaks olid välja valitud arvutiplatvormi jaoks Windows 10 operatsioonisüsteem, veebiplatvormi tarvis Chrome brauser ja mobiilplatvormi jaoks Android 5 operatsioonisüsteem . Mobiiliplatvormi jaoks kasutati Nexus 9 tahvelarvutit. Eelnimetatud 15 testi on automatiseeritud ja lahendus on välja toodud Lisas 1. Eelnimetatud operatsioonisüsteemid, brauser ja tahvelarvuti Nexus 9 olid välja valitud töö realisatsiooni näitamiseks lähtudes testitava tarkvara nõudmistest ja töö optimaalsuse mõttes. Sama lahendus hakkab töötama ka teiste tahvelarvutitega, mis kasutavad operatsioonisüsteeme, mis vastavad automaattestide töökeskkonnale esitatud nõuetele. Sama kehtib ka teiste brauserite ja operatsioonisüsteemide suhtes.

Selles peatükis esitletakse klasside koostöö struktuuri, mis on sarnane kõikide platvormide jaoks. Põhjalikumalt kirjeldatakse ja selgitatakse realisatsiooni ainult esimese testiloo arvutiplatvormi jaoks. Teiste platvormide jaoks testide realisatsioon on sama, ainult pildid, mille järgi SikuliX testib on erinevad. Testide realisatsiooni sarnasuse tõttu pole vajadust kõigi testilugude põhjalikumaks kirjeldamiseks. Töö lisa 1 sisaldab teiste testilugude koodinäited. Samuti kirjeldatakse BaseTest klassi iga platvormi jaoks eraldi. Selle klassi realisatsioon erineb kõikides platvormides. Lahenduse lõpus on väljatoodud tagasiside ja järeldused. Automaattestid on kirjutatud Intellij IDEA tasuta versioonis kasutades SikuliX Java API-liidest ja Seleniumi WebDriveri API-liidest ning Java programmeerimiskeelt.

5.1 Klasside koostöö struktuur

Siin on esitatud klasside koostöö struktuuri kirjeldus kõikide platvormide jaoks.



Joonis 16. Arvuti-, veebi- ja mobiilplatvormide klasside koostöö struktuur

Joonisel 16 välja toodud Test klass tähendab testimise klassi koos testidega. Käivitades testi või testi klassi, Test klass pöördub BaseTest klassi poole. BaseTest klass on mõeldud testitava rakenduse avamiseks ja kinnipanemiseks. BasetTest klass pöördub DataProperties klassi poole. DataProperties klassi abiga on võimalik lihtsamalt moel laadida vajaliku pilti ja/või teksti, kasutades selleks mõeldud klassi meetodeid. Edaspidi test või Test klass pöördub MainLobbyPage ja teiste sarnaste klasside lehtede poole. Need lehed sisaldavad ja esindavad testitava rakenduse vastava lehe kasutajaliidest koos vajalike meetoditega. Need lehed pöörduvad ka DataProperties klassi ja BasePage poole. BasePage on põhiklass teiste testitava rakenduse kasutajaliideseid sisalduvate lehtede jaoks. BasePage klass sisaldab nende klasside ühiselt kasutatavaid meetodeid. BasePage pöördub SikuliScreen poole eesmärgiga saada ligipääs arvutiekraanile. See on vajalik SikuliX'i jaoks.

5.2 Lahendus arvutiplatvormile

Siin on esitatud põhjalik kirjeldus BaseTest, DataProperties ja BasePage klassist. Samuti on välja toodud esimese testiloo lahenduse kirjeldus. Teiste testilugude lahendused on kättesaadavad lisas 1.

Rakenduse avamiseks ning kinnipanemiseks oli loodud klass BaseTest. Siin olen kasutanud SikuliX Java API-liidest, mis on kirjeldatud alapeatükis 4.1, TestNG (kirjeldatud alapeatükis 4.4) ja Java programmeerimiskeelt. Klass on toodud välja allpool:

```
public class BaseTest {
    private App app;

    @BeforeMethod
    public void setUp() throws InterruptedException, FindFailed {
        openDesktopClient();
        LoginPage loginpage = new LoginPage();
        assertTrue(loginpage.isLoginPageLoaded());
        loginpage.loginIn("ko2");
        Thread.sleep(10000);
```

```
    }

    @AfterMethod
    public void tearDown() {
        closeDesktopClient();
    }

    private void openDesktopClient() {
        try {
            App.open(DataProperties.get("poker.path"));
            Thread.sleep(1000);
        } catch (Exception e) {
            fail("Can't open desktop client " + DataProperties.get("poker.path"));
        }
    }

    private void closeDesktopClient() {
        App.close("Unibet Poker");
    }
}
```

Siin on kaks peamist meetodit: setUP() ja tearDown(). Need meetodid sisaldavad teisi vajalikke meetodeid.

@BeforeMethod, *@AfterMethod*, *Assert.assertTrue*, *Assert.assertTrue* on pärit TestNG raamistikust[15]. Rakenduse avamiseks ja kinnipanemiseks kasutan *App.open* ja *App.close* võimalusi, mis tulevad SikuliX Java API'st [5] ning login sisse kasutades enda tehtud klassi LoginPage ja selle meetodeid. Laen vajalikud andmeid kasutades meetodeid DataProperties klassist. Klassi esimene pool on toodud välja allpool:

```
private static Properties PROPERTIES;

    static {
        PROPERTIES = new Properties();
        URL props = ClassLoader.getSystemResource("Data.properties");
        try {
            PROPERTIES.load(props.openStream());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static String getProperty(String key) {
        return PROPERTIES.getProperty(key);
    }

    public static String get(String key) {
        return getProperty(key);
    }
```

DataProperties klass [21] oli kasutatud töö lihtsustamise eesmärgil. Klass on kopeeritud avalikust allikast internetist. Tema põhiline eesmärk oli aidata mugavalt laadida projekti ressurssidest pildid ja andmeid. Terve klass on kättesaadav lisas 1.

Kasutades meetodi get(), laen olemasolevast failist Data.properties vajalikud andmeid.

Järgnevalt on välja toodud põhiklass BasePage. Sellest klassist hakkavad meetodeid pärima teised klassid, mis esindavad testitava rakenduse kasutajaliidese osa. Klass oli loodud töö lihtsustamise eesmärgil. Siin olen kasutanud SikuliX Java API-liidest [5] ja Java programmeerimiskeelt. Klass on toodud välja allpool:

```
public class BasePage {
    protected Region window;

    protected BasePage() {
        window = SikuliScreen.getScreen();
        Settings.OcrTextRead = true;
        Settings.OcrTextSearch = true;
    }

    protected void takeScreenshot() {

window.saveScreenCapture("C:\\Users\\Natalja\\IdeaProjects\\SikuliDesktopWindowsTesting\\sc
reens\\");
    }

    protected void waitAndClick(Pattern pattern) throws FindFailed {
        window.wait(pattern, 20);
        window.click(pattern);
    }

    protected void waitAndClick(Pattern pattern, Pattern pattern1) throws FindFailed {
        window.wait(pattern, 20);
        window.click(pattern1);
    }
}
```

Klassi konstruktoris kutsun välja klassi SikuliScreen ja annan SikuliX'ile arvutiekraani, ehk tööobjektiks saab terve arvutiekraan. SikuliScreen klass on esitatud allpool:

```
public class SikuliScreen {

    private static Screen screen;

    public synchronized static Screen getScreen() {
        if (screen == null) {
            screen = new Screen();
        }
        return screen;
    }
}
```

SikuliScreen klass on loodud arvutiekraani loomiseks. Vajalik SikuliX'iga töötamiseks.

SikuliX klass *Screen* [20] on mõeldud füüsilise monitori esindamiseks, kus SikuliX hakkab tööd tegema.

Töö tegemiseks SikuliX vajab regioonile kohta, kus ta hakkab otsima pilte. Regioon on ristkülikukujuline osa arvutiekraanist, mis on määratud koordinaatidega alates üleval vasakust nurgast (x,y) mis on pikslites nihe alates ülemisest vasakust nurgast, mis on reeglina (0,0) ning laiusega ja kõrgusega (w,h) [18]. Siin me anname regioonile terve arvutiekraani.

Settings.OcrTextRead ja *Settings.OcrTextSearch* lubavad testitava rakenduse sees teksti otsida ja lugeda [18].

Ülejäänud meetodid *wait(), click(), saveScreenCapture()* on ühed põhilisemast SikuliX meetoditest. Sulgudes *pattern*, tähendab pilti, mida me hakkame arvutiekraanil otsima. Sulgudes number tähendab mitu sekundid meetod *wait()* ootab otsitava pildi ilmumist [18, 19].

5.2.1 Esimene testilugu

```
@Test
  public void changeTableBackground() throws FindFailed {
      MainLobbyPage mainlobby = new MainLobbyPage();
      mainlobby.openMyProfilePage();
      MyProfilePage myprofile = new MyProfilePage();
      myprofile.changeTableBackground();
      assertTrue(myprofile.isMyProfilePageLoaded());
  }
```

Üleval toodud test kontrollib, kas on võimalik muuta mängulaua tausta.

Siin on kasutusel kaks klassi: MainLobbyPage ja MyProfilePage. MainLobbyPage klass on toodud välja allpool:

```
public class MainLobbyPage extends BasePage {
    private Pattern viewProfileButton;
    private Pattern gameChooserText;

    public MainLobbyPage() {
        viewProfileButton = new Pattern(DataProperties.path("ViewProfileButton.png"));
        gameChooserText = new Pattern(DataProperties.path("GameChooserText.png"));
    }

    public boolean isLobbyPageLoaded() {
        try {
```

```
            window.wait(gameChooserText.similar((float) 0.7), 25);
            window.find(viewProfileButton.similar((float) 0.7));
            return true;
        } catch (FindFailed e) {
            return false;
        }
    }

    public void openMyProfilePage() throws FindFailed {
        window.wait(viewProfileButton.similar((float) 0.9), 20);
        window.click(viewProfileButton.similar((float) 0.7));
    }
}
```

Klass laiendab BasePage klassi ja esindab testitava rakenduse Fuajee lehe kasutajaliidest. Klass sisaldab lisaks vajalikke meetodeid testide jaoks.

Klassi konstruktoris laetakse pildid, mida hakatakse otsima ja kasutama selles klassis. Pildid laetakse kasutades DataProperties klassi meetodit path(). Need meetodid on välja toodud allpool:

```
  public static String getPathFor(String file) {
        return ClassLoader.getSystemResource(file).getPath().toString()
                .substring(1);
    }

    public static String path(String file) {
        return getPathFor(file);
    }
```

DataProperties klass [21] oli loodud töö lihtsustamise eesmärgil. Klass on kopeeritud internetist avalikust allikast. Tema põhiline eesmärk oli aidata laadida projekti ressurssidest pilte ja andmeid. Terve klass on kättesaadav lisas 1.

SikuliX kasutab sõna *pattern* otsitava pildi tuvastamiseks. Iga otsitava *pattern'i* jaoks on võimalik kasutada sõna *similar*. *Similar* tähendab, kui suure sarnasusega SikuliX otsib. Vaikimisi see on 0.7, kuid on alati võimalik seda vähendada või suurendada. Samuti on võimalik muuta *wait()* meetodi aega spetsiaalselt teatud pildi jaoks kirjutades mitu sekundit SikuliX ootab pildi ilmumist [18, 19] .

Meetodid *wait(), click(), find()* on ühed põhilisemast SikuliX meetoditest[18].

MyProfilePage klass on toodud välja allpool:

```
public MyProfilePage() {
        oneSngTwoSeatJoinIdentifier = new
Pattern(DataProperties.path("OneSngTwoSeatJoinIdentifier.png"));
        oneSngTwoSeatTicket = new Pattern(DataProperties.path("OneSngTwoSeatTicket.png"));
```

```
        sngJoinButtonFromTicketLayout = new
Pattern(DataProperties.path("SngJoinButtonFromTicketLayout.png"));
        ticketBuyinButton = new Pattern(DataProperties.path("JoinWithTicketButton.png"));
        createNewIdButton = new Pattern(DataProperties.path("CreateNewIdButton.png"));
        unibetPokerLogo = new Pattern(DataProperties.path("UnibetPokerLogo.png"));
        changeBgButton = new Pattern(DataProperties.path("ChangeBGButton.png"));
        changeBgLayoutText = new Pattern(DataProperties.path("ChangeBGLayoutText.png"));
        tableBgOnChange = new Pattern(DataProperties.path("TableBGLayout.png"));
        saveBgButton = new Pattern(DataProperties.path("SaveBGButton.png"));
        layoutCloseButton = new Pattern(DataProperties.path("LayoutCloseButton.png"));
        achievementsLayoutText = new
Pattern(DataProperties.path("AchievementsLayoutText.png"));
        yourAchievementsLayoutText = new
Pattern(DataProperties.path("YourAchievementsLayoutText.png"));
        randomAchievement = new Pattern(DataProperties.path("RandomAchievement.png"));
        randomAchievementOnPlayer = new
Pattern(DataProperties.path("RandomAchievementOnPlayer.png"));
        randomAchievementOnPlayerProfile = new
Pattern(DataProperties.path("RandomAchievementOnPlayerProfile.png"));
        backToLobbyButton = new Pattern(DataProperties.path("BackToMainLobbyButton.png"));
        randomAchievementOnPlayerLobby = new
Pattern(DataProperties.path("RandomAchievementOnPlayerLobby.png"));
        changedTableBgMyProfile = new
Pattern(DataProperties.path("ChangedTableBgMyProfile.png"));
    }

    public boolean isMyProfilePageLoaded() {
        try {
            window.wait(unibetPokerLogo.similar((float) 0.7), 25);
            window.find(createNewIdButton.similar((float) 0.7));
            return true;
        } catch (FindFailed e) {
            return false;
        }
    }

    public void joinSngWithTicket() throws FindFailed {
        waitAndClick(oneSngTwoSeatTicket.similar((float) 0.7),
oneSngTwoSeatTicket.similar((float) 0.7));
        waitAndClick(oneSngTwoSeatJoinIdentifier.similar((float) 0.7),
sngJoinButtonFromTicketLayout);
        waitAndClick(ticketBuyinButton.similar((float) 0.7));

    }

    public void changeTableBackground() throws FindFailed {
        waitAndClick(changeBgButton);
        waitAndClick(changeBgLayoutText, tableBgOnChange);
        waitAndClick(saveBgButton);
        waitAndClick(layoutCloseButton);
        assertTrue(isElementPresent(changedTableBgMyProfile));
    }

    public boolean isElementPresent(Pattern pattern) {
        try {
            window.wait(pattern.similar((float) 0.8), 20);
            window.find(pattern.similar((float) 0.8));
            return true;
        } catch (FindFailed e) {
            return false;
        }
    }

    public void applyAchievementOnAvatar() throws FindFailed {
        waitAndClick(achievementsLayoutText);
        waitAndClick(yourAchievementsLayoutText, randomAchievement);
```

```
        waitAndClick(randomAchievementOnPlayer, layoutCloseButton);
        waitAndClick(randomAchievementOnPlayerProfile, backToLobbyButton);
        window.find(randomAchievementOnPlayerLobby);
    }
}
```

Klass laiendab BasePage klassi ja esindab testitava rakenduse Minu Profiil lehe kasutajaliidest. Lisaks sisaldab klass vajalikke meetodeid testide jaoks.

Klassi konstruktoris laetakse pildid, mida hakatakse otsima ja kasutama selles klassis. Pildid laetakse kasutades DataProperties klassi meetodi path(). Meetodid on välja toodud käesoleva töö leheküljel 45. SikuliX kasutab sõna *pattern* otsitava pildi tuvastamiseks. Iga otsitava *pattern'i* jaoks on võimalik kasutada sõna *similar* [19].

Similar tähendab, kui suure sarnasusega SikuliX otsib. Vaikimisi see on 0.7, kuid on alati võimalik seda vähendada või suurendada. Samuti on võimalik muuta *wait()* meetodi aega spetsiaalselt teatud pildi jaoks, kirjutades palju sekundit SikuliX ootab pildi ilmumist [18, 19].

Assert.assertTrue on pärit TestNG raamistikust. Meetod waitAndClick() on pärit BasePage klassist, mida see klass laiendab. See meetod ootab vajaliku pildi ilmumist ekraanile ja vajutab selle peale. Meetodid *wait()*, *find()* on ühed põhilisemast SikuliX meetoditest [18].

5.3 Lahendus veebiplatvormile

Siin on esitatud põhjalik kirjeldus BaseTest klassist. Põhjuseks selle realisatsiooni erinevus teistest platvormidest. Testilugude lahendused on kättesaadavad lisas 1.

Rakenduse avamiseks ja kinnipanemiseks oli loodud põhiklass BaseTest. Siin olen kasutanud Selenium WebDriver API-liidest, mis on kirjeldatud alapeatükis 4.2, SikuliX Java API-liidest, mis on kirjeldatud alapeatükis 4.1, TestNG raamistikku, mis on kirjeldatud alapeatükis 4.4 ja Java programmeerimiskeelt. Klass on toodud välja allpool:

```
public class BaseTest {
    WebDriver driver = null;

    @BeforeMethod
    public void setUp() {
        openWebClient();
    }

    @AfterMethod
    public void tearDown() {
        closeWebClient();
```

```
    }

    public void openWebClient() {
        try {
            driver = new FirefoxDriver();
            driver.get(DataProperties.get("web.path"));
            Thread.sleep(30000);
        } catch (Exception e) {
            fail("Can't opeen web client " + DataProperties.get("web.path"));
        }
    }

    private void closeWebClient() {
        driver.quit();
    }
}
```

Siin on kaks peamist meetodit: setUP() ja tearDown(). Need meetodid sisaldavad teisi vajalikke meetodeid. Klassi struktuuri otsustasin teha samasugusesse nagu arvutiplatvormil BaseTest klass.

@BeforeMethod, *@AfterMethod*, *Assert.assertFail* on pärit TestNG raamistikust [15]. Rakenduse avamiseks ning kinnipanemiseks kasutan Selenium WebDriver API-liidest ja laen andmed kasutades DataProperties klassi, mis on välja toodud käesoleva töö leheküljel 42.

Oma struktuuri poolest on klass sarnane arvutiplatvormiga. Klassi realisatsioon on veidi erinev, mis on tingitud Selenium WebDriver API-liidese kasutamisest.

5.4 Lahendus mobiilplatvormile

Siin on esitatud põhjalik kirjeldus BaseTest klassist. Põhjuseks selle realisatsiooni erinevus teistest platvormidest. Testilugude lahendused on kättesaadavad lisas 1.

Rakenduse avamiseks ning kinnipanemiseks oli loodud põhiklass BaseTest. Siin olen kasutanud Mobizen rakendust, mis on kirjeldatud alapeatükis 4.3, SikuliX Java API-liidest, mis on kirjeldatud alapeatükis 4.1, TestNG raamistikku, mis on kirjeldatud alapeatükis 4.4 ja Java programmeerimiskeelt. Klass on toodud välja allpool:

```
public class BaseTest {
    App app;
    private Region window;
    private Pattern appIcon;
    private Pattern deviceAllAppButton;
    private Pattern pokerCloseButton;
    private Pattern deviceHomeButton;
```

```
    public BaseTest() {
        appIcon = new Pattern(DataProperties.path("AppIcon.png"));
        deviceAllAppButton = new Pattern(DataProperties.path("DeviceAllAppButton.png"));
        pokerCloseButton = new Pattern(DataProperties.path("PokerCloseButton.png"));
        deviceHomeButton = new Pattern(DataProperties.path("DeviceHomeButton.png"));
        window = SikuliScreen.getScreen();
    }

    @BeforeMethod
    public void setUp() throws InterruptedException, FindFailed {
        openMobileClient();
        LoginPage loginpage = new LoginPage();
        assertTrue(loginpage.isLoginPageLoaded());
        loginpage.loginIn("ko2");
        Thread.sleep(10000);
    }

    @AfterMethod
    public void tearDown() throws FindFailed {
        App.focus("Mobizen");
        closeMobileClient();
    }

    private void openMobileClient() throws FindFailed {
        try {
            App.focus("Mobizen");
            Thread.sleep(10000);
            waitAndClick(appIcon.similar((float) 0.9));
            waitAndClick(appIcon.similar((float) 0.9));
            Thread.sleep(10000);
        } catch (Exception e) {
            fail("Can't open mobile android client ");
        }
    }

    private void closeMobileClient() throws FindFailed {
        waitAndClick(deviceAllAppButton.similar((float) 0.9));
        waitAndClick(pokerCloseButton.similar((float) 0.9));
        waitAndClick(deviceHomeButton.similar((float) 0.9));
    }

    private void waitAndClick(Pattern pattern) throws FindFailed {
        window.wait(pattern, 20);
        window.click(pattern);
    }
}
```

Siin on kaks peamist meetodit: setUP() ja tearDown(). Need meetodid sisaldavad teisi vajalikke meetodeid.

Siin olen kasutanud palju SikuliX abi. Klassi konstruktoris laen vajalikud pildid, mida hakatakse kasutama selles klassis. Loon regiooni jaoks arvutiekraani ja teostan toiminguid kasutades SikuliX raamistikku.

SikuliX ei oska töötada reaalsel füüsilisel mobiil seadmel. Töö õnnestumiseks, peegeldan mobiili ekraani arvutiekraani peale kasutades Mobizen rakendust ja SikuliX abil teostan toimingud mobiilil oleva testitava rakendusega.

@BeforeMethod, *@AfterMethod* ja *Assert.assertTrue* on pärit TestNG raamistikust[15].

App.focus tuleb SikuliX Java API võimalustest [5] ning login sisse kasutades enda tehtud klassi LoginPage ja selle meetodeid . Laen andmeid DataProperties klassi kasutamisega, mis on välja toodud käesoleval töö leheküljel 45.

Lisaks olid tehtud abimeetodid waitAndClick(), openMobileClient() ja closeMobileClient(). Esimene meetod ootab pildi ilmumist ekraanile ja siis vajutab pildi peale. Teine meetod avab rakenduse ja kolmas meetod paneb rakenduse kinni. Meetodid *wait()*, *click()* on ühed põhilisemast SikuliX meetoditest [18].

5.5 SikuliX Java API tagasiside praktikast

5.5.1 Eelised

- **Tasuta tarkvara**
  Avatud lähtekoodiga tarkvara on tasuta kättesaadav internetis.
- **Tarkvara töö ulatus**
  Testida on võimalik kõike, mida te näete oma arvutiekraanil. Väga hea lahendus testimiseks, kus puudub ligipääs rakenduse elementidele näiteks Flash rakenduse puhul.
- **Suur funktsionaalsus**
  Väga palju funktsioone ning võimalusi, mida on võimalik kasutada testimise käigus.
  Mõned ekraanipildi otsingu funktsioonid: *find()*, *findAll()*, *exists()*, *wait()*, *waitVanish()*.
  Mõned hiire funktsioonid: *click()*, *doubleClick()*, *rightClick()*, *hover()*, *dragDrop()*, *mouseDown()*, *mouseUp()* [18] .
  Mõned klaviatuuri funktsioonid: *type(text)*, *paste(text)*, *keyDown()*, *keyUp()* ja palju muud [18].
- **Kasutajasõbralikkus**
  Tarkvara on suhteliselt lihtne ning arusaadav. Kasutajal ei pea olema eelnevat programmeerimise oskust.

- **Head informatsiooni allikad**

  Saadaval on palju erinevaid materjale, kus on võimalik õppida tarkvara kasutamisest. Materjalid on koos näidiste ja piltidega.
- **Suur kasutajate hulk ja kogemuste jagamine**

  Tarkvaral on palju kasutajaid ning on alati võimalik küsida abi.
- **Paindlik ja kiire**

  Tarkvaral on palju võimalusi, mida on võimalik ka enda järgi seadistada.

  Näiteks vaikimisi SikuliX otsib ekraanipildi 3 sekundiga enne kui annab vea. Seda on võimalik reguleerida, suurendades või vähendades *setAutoWaitTimeout 'it.*

  SikuliX otsib ekraanipildi kasutades meetodeid, nagu näiteks *Region.find()*. Vaikimisi leidmise sarnasus on siin 0.7, mis tähendab iga piksli jaoks 70% täpsust. Seda on võimalik muuta vähendades või suurendades *Settings.similarity*. Samuti on võimalik muuta sarnasuse väärtust, mis on seotud vaid teatud pildiga [18].
- **Pidev arendus**

  SikuliX on pidevas arenduses. Viimane uuendus oli seitsmendal oktoobril 2015 ning oktoobris hakati arendama SikuliX2 versiooni 2.0.0 [7].

5.5.2 Puudused

- **Ekraanipiltide vajadus**

  Testimine põhineb piltidelt. Pildi puudumine hukutab testi
- **Ekraanipildi kokkulangevuse vajadus**

  Testi õnnestumiseks peab otsitav pilt alati kokku langema etteantud pildiga. Pildi erinevus 1 piksli võrra annab *FindFailed* erandi ja test ebaõnnestub.
- **Ekraanipiltide suur hulk**

  SikuliX peamiseks testimise komponendiks on ekraanipildid ning tavaliselt neid on palju.

5.6 Järeldused

SikuliX on väga võimekas ning paindlik tarkvara koos paljude võimalustega. Selle tarkvara kasutusele võtmist peaks kaaluma, kui soovite testida ja mängida elementidega tuginedes ainult nende visuaalsele kujundile. See on suurepärane lahendus Flash rakendustele. Teiseks peamiseks SikuliX eeliseks on see, et ta töötab nii Windows, Mac kui ka enamustel

Linux/Unix süsteemidel. See teeb võimalikuks testida Flash rakendusi kõigil eelnimetatutel platvormidel.

SikuliX'i on väga lihtne kasutada ja integreerida teiste tööriistadega. Oma töös olen seda kasutanud koos Selenium ning Mobizen rakendusega ja TestNG raamistikuga. Sellest tulenevalt oli mul võimalik testida Flash raskendust nii arvuti-, veebi- kui ka mobiilplatvormil.

SikuliX peamiseks miinuseks on asjaolu, et ta sõltub väga palju ekraanipiltide olemusest ning täpsusest 1 piksel. Brauseri või brauseri kasutajaliidese muutumisel peaks kõik testid ümber kirjutama - vastasel juhul testimine ebaõnnestub.

Selenium on populaarne veebiautomatiseerimisraamistik, mis töötab enamike brauseritega. Lihtne kasutada koos teiste tööriistadega. Oma töös olen kasutanud eelpool nimetatut vaid mängu avamiseks vajalikus brauseriaknas. Seetõttu ei oska ma kõnealust vahendit rohkem kommenteerida.

Mobizen rakendus oli väga heaks abiks mobiiliekraani arvutiekraanile peegeldamiseks ja kontrollimiseks. Rakendust on lihtne kasutada ja ekraanipildi edastamise kiirus oli hea.

TestNG raamistik oli suurepäraseks lahenduseks testide koodi struktureerimiseks. Lihtne kasutada, kasutajasõbralik ja palju võimalusi. Näiteks saab määrata, millises järjekorras testid hakkavad töötama. See on kasulik juhul kui on olemas sõltuvus. Samuti meeldis see, et testide konfigureerimiseks ja käivitamiseks on võimalik kasutada XML formaadis faile. Siin on võimalik määrata väga palju seadistusi ning kohandada testikomplekte vastavalt vajadustele ja palju muud.

Kasutades eelnimetatud vahendeid, olid loodud automaattestid ja koostöö SikuliX ja teda abistavate tööriistade vahel. Automaattestid ja selline koostöö toimib ning on valmis edaspidiseks arendamiseks.

5.7 Edasised plaanid

Töös valminud automaatteste hakatakse kasutama igapäevases arendusprotsessis. Samuti automatiseeritakse ülejäänud projekti suitsutestid kasutades koostööd SikuliX ja teda abistavate tööriistade vahel. Lisaks kavatsetakse edasi arendada koostööd SikuliX ja teda abistavate tööriistade vahel testimise automatiseerimise raamistikuks.

6. Kokkuvõte

Toote kvaliteet on alati olnud inimeste tähelepanu keskpunktis. Tarkvara testimisel siin on üliolu line roll, mis peab olema ka väga hästi täidetud.

Tänapäeval jääb toote keerukuse ning ajaressursi piiratuse tõttu puhtalt manuaalsest testimisest vajaka ning just seetõttu peaks kaaluma testimise protsessi automatiseerimist.

Käesoleva bakalaureusetöö eesmärgiks oli väljavalitud suitsutestide automatiseerimine arvuti-, veebi-, mobiilplatvormidel Flash rakenduse jaoks kasutades SikuliX raamistikku ja seda abistavaid tööriistu.

Eesmärgi täitmiseks teostati järgmised tegevused:

- Tutvustati ja analüüsiti testitavat rakendust. Esitleti rakenduse struktuur, kirjeldati funktsionaalseid nõudeid ja toodi välja kriteeriumid automaattestide töökeskkonnale.
- Valiti välja 5 kõige sobivamat suitsutesti, samuti operatsioonisüsteemid ning brauser iga platvormi tarvis.
- Lühidalt kirjutati suitsutestimisest ja iga väljavalitud suitsutestide jaoks koostati testilood.
- Tutvustati ja analüüsiti SikuliX raamistikku ja teisi abistavaid tööriistu.

Töö tulemusena automatiseeriti 5 väljavalitud suitsutesti arvuti-, veebi- ja mobiilplatvormidele. Arvutiplatvormil kasutati Windows 10 operatsioonisüsteemi, veebiplatvormil Chrome brauserit ja mobiilplatvormil Android 5 operatsioonisüsteemi. Põhjalikumalt kirjeldati esimest testilugu arvutiplatvormi jaoks. Samuti esitleti klasside koostöö struktuuri kõikide platvormide jaoks. Toodi välja senine tagasiside ja muljeid SikuliX kasutamisest ning järeldused kõikide vahendite kasutamisest ja edasised plaanid.

Praktilise osana loodud automaatteste ja koostööd SikuliX ja teda abistavate tööriistade vahel hakatakse kasutama igapäevaselt arendusprotsessis ning edasi arendama automatiseerimise raamistikuks.

Töö raames loodud koostöö SikuliX ja teda abistavate tööriistade vahel sobib ükskõik millise Flash rakenduse testimise automatiseerimiseks, kus kasutajaliidese osad ei vahetu tihti.

Summary

Product quality has always been at the core of everyone's attention. Software testing has the main responsibility here that should be perfectly done.

Nowadays, because of complexity of the products and time limitations only manual testing is not enough, and that is why we need to start thinking about test automation.

The aim of this thesis was to automate smoke test cases for Flash based application on desktop, web and mobile platforms, using SikuliX framework and its supportive tools.

In order to achieve this goal, the following activities were done:

- Application used for testing was introduced and analysed. Its structure, main functional requirements and basic criteria's for automated test cases work environment were presented.
- 5 most suitable smoke test cases, operation systems and browser were chosen for every platform.
- Brief introduction of smoke testing was presented, and test stories were made for every smoke test case.
- SikuliX framework and its supportive tools were presented and analysed.

As a result, 5 chosen smoke test cases were automated on desktop, web and mobile platforms. For desktop platform - Windows 10 operation system, for web platform - Chrome browser and for mobile platform - Android 5 operation system were used. Detailed explanation was done only for first automated test case on desktop platform. Also, class cooperation structure for all platforms was introduced.

At the end of the work, the author gave her feedback and impressions after trying and using SikuliX framework, made conclusion about all used tools and discussed further plans of automated test framework development.

Created in this work automated tests and cooperation between SikuliX and its supportive tools will be used in everyday software development process by the author at her work and will be developed further into automation framework.

Cooperation between SikuliX and its supportive tools, created in this thesis, is suitable for testing of any Flash application, which UI parts do not change often.

Kasutatud kirjandus

[1] Markvardt, M. Tarkvara testimist käsitlev juhendmaterjal. [WWW]

https://www.mkm.ee/sites/default/files/tarkvara_testimise_juhis_-_koopia.doc (03.01.2016)

[2] Unibet koduleht. Pokker Unibetis. [WWW]

https://www.unibet.ee/help/products/poker/poker-at-unibet (03.01.2016)

[3] Pokernews koduleht. Unibet täielik ülevaade ja allalaadimine. [WWW]

http://ee.pokernews.com/unibet-poker-ee (03.01.2016)

[4] SikuliX koduleht. How SikuliX works. [WWW]

http://doc.sikuli.org/devs/system-design.html (03.01.2016)

[5] Github koduleht. Usage in Java programming. [WWW]

https://github.com/RaiMan/SikuliX-2014/wiki/Usage-in-Java-programming (03.01.2016)

[6] Code Google koduleht. Sikuli Java API. [WWW]

https://code.google.com/p/sikuli-api/ (03.01.2016)

[7] SikuliX koduleht. SikuliX framework. [WWW]

http://www.sikulix.com/ (03.01.2016)

[8] QualityLogic koduleht. Sikuli IDE. [WWW]

https://www.qualitylogic.com/Contents/Mobile/Technology/Sikuli-IDE.aspx (03.01.2016)

[9] SikuliX koduleht. How does SikuliX find images on the screen? [WWW]

http://www.sikulix.com/stories/how-does-sikuli-find-images-on-the-screen (03.01.2016)

[10] Seleniumi koduleht. What is Selenium? [WWW]

http://www.seleniumhq.org/ (03.01.2016)

[11] Selenium koduleht. Selenium IDE. [WWW]

http://www.seleniumhq.org/projects/ide/ (03.01.2016)

[12] Selenium koduleht. Selenium WebDriver. [WWW]

http://www.seleniumhq.org/projects/webdriver/ (03.01.2016)

[13] Selenium koduleht. Selenium Remote Control. [WWW]

http://www.seleniumhq.org/projects/remote-control/ (03.01.2016)

[14] Mobizen koduleht. Mobizen. [WWW]

https://www.mobizen.com/ (03.01.2016)

[15] TestNG koduleht. TestNG. [WWW]

http://testng.org/doc/documentation-main.html (03.01.2016)

[16] Devcolibri koduleht. Тестирование с помощью TestNG в Java. [WWW]

http://devcolibri.com/1528 (03.01.2016)

[17] Sikuli koduleht. Location. [WWW]

http://doc.sikuli.org/location.html (03.01.2016)

[18] Sikuli koduleht. Region. [WWW]

http://doc.sikuli.org/region.html (03.01.2016)

[19] Sikuli koduleht. Pattern. [WWW]

http://doc.sikuli.org/pattern.html (03.01.2016)

[20] Sikuli koduleht. Screen. [WWW]

http://sikulix-2014.readthedocs.org/en/latest/screen.html (03.01.2016)

[21] Java-Sikuli-Demo. Dataproperties klass. [WWW]

https://github.com/polusok/Java-Sikuli-Demo/blob/master/src/main/java/com/spotify/utils/DataProperties.java (03.01.2016)

[22] Petuhhov, I. Inkrementaalne arendusmudel. [WWW]

http://www.e-uni.ee/e-kursused/eucip/arendus/1222_inkrementaalne_arendusmudel.html (03.01.2015)

[23] Saar, J. Rakendusliides. [WWW]

https://et.wikipedia.org/wiki/Rakendusliides (03.01.2015)

[24] Cognitect, Inc. Clojure koduleht. [WWW]

http://clojure.com (03.01.2016)

[25] Mischok, S. What is Flash, when and why to use it? [WWW]

http://www.killersites.com/articles/articles_FlashUse.htm (03.01.2016)

[26] W3schools koduleht. What is Html? [WWW]

http://www.w3schools.com/html/html_intro.asp (03.01.2016)

[27] Search Sofware Qualty koduleht. Integrated development environment (IDE) definition.

http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment

(03.01.2016)

[28] Jetbrains koduleht. Intellij IDEA. [WWW]

https://www.jetbrains.com/idea/ (03.01.2016)

[29] E-teatmik. Java. [WWW]

http://www.vallaste.ee/index.htm?Type=UserId&otsing=731 (03.01.2016)

[30] Oracle koduleht. Java Native Interface. [WWW]

http://docs.oracle.com/javase/7/docs/technotes/guides/jni/index.html (03.01.2016)

[31] Github koduleht. About JRuby. [WWW]

https://github.com/jruby/jruby/wiki/AboutJRuby (03.01.2016)

[32] Jython wiki. [WWW]

https://wiki.python.org/jython (03.01.2016)

[33] Apache Maven koduleht. Welcome to Apache Maven. [WWW]

https://maven.apache.org (03.01.2016)

[34] OpeCV koduleht. [WWW]

http://opencv.org/ (03.01.2016)

[35] Apache Maven koduleht. Introduction to the POM. [WWW]

https://maven.apache.org/guides/introduction/introduction-to-the-pom.html (03.01.2016)

[36] Python wiki. Sissejuhatus. [WWW]

http://kuutorvaja.eenet.ee/wiki/Python#Sissejuhatus (03.01.2016)

[37] Ruby koduleht. About Ruby. [WWW]

https://www.ruby-lang.org/en/about (03.01.2016)

[38]Sõnajalg, S.Aktorite mudelil põhinev paralleelprogrammeerimine. [WWW]

http://kodu.ut.ee/~varmo/seminar/sem08S/sonajalg.pdf (03.01.2016)

[39] E-teatmik. Script. [WWW]

http://www.vallaste.ee/index.htm?Type=UserId&otsing=288 (03.01.2016)

[40] Tesseract OCR google leht. [WWW]

https://code.google.com/p/tesseract-ocr/ (03.01.2016)

[41] Search Sofware Quality koduleht. User interface (UI) definition. [WWW]

http://searchsoa.techtarget.com/definition/user-interface (03.01.2016)

[42] W3schools koduleht. What is XML. [WWW]

http://www.w3schools.com/xml/xml_whatis.asp (03.01.2016)

Lisa 1

Testprogrammi lähtekood

1. https://bitbucket.org/
   Kasutajanimi: Testuni10
   Parool: Testtest1

Kõigi kolme projekti struktuurid on sarnased. Esitan arvutiplatvormi lähtekoodi struktuuri: