Elli Jukarainen

ROBOT FRAMEWORKIN KÄYTTÖÖN-OTTO JA KÄYTTÖLIITTYMÄTESTIEN TEKEMINEN BROWSER-KIRJASTOLLA

Opinnäytetyö

Liiketalouden ammattikorkeakoulututkinto

Tietojenkäsittelyn koulutus

2023

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Tradenomi (AMK)
Tekijä/Tekijät	Elli Jukarainen
Työn nimi	Robot Frameworkin käyttöönotto ja käyttöliittymätestien tekeminen Browser-kirjastolla
Toimeksiantaja	-
Vuosi	2023
Sivut	35 sivua
Työn ohjaaja	Marjo Puikkonen

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli kehittää ohjeet Robot Frameworkin käyttöönottoa varten kokoamalla eri työkalujen asennusohjeet sekä teoriaa Robot Frameworkin perustoiminnallisuudesta ja syntaksista Toinen tavoite oli tehdä esimerkkitestejä käyttöliittymätestauksesta Browser-kirjastoa hyödyntäen. Työn tarkoituksena oli tuottaa lukijalle ohjeet Robot Frameworkin käyttämiseen testiautomaatiossa sekä tarjota yksinkertaisia esimerkkitestejä, joita voi matalalla kynnyksellä kokeilla soveltaa omiin käyttöliittymätesteihin.

Toteutuksessa on käytetty apuna aiheesta löytyvää teoriatietoa ja hyödynnetty myös omaa aikaisempaa osaamista aiheeseen liittyen. Työn teoriaosassa käsitellään ohjelmistotestausta yleisesti ja syvennytään testauksen eri tasoihin sekä testiautomaatioon ja käyttöliittymätestaukseen. Asennusohjeiden sekä hyvien testitapausten luomiseen liittyvän teorian kokoamisessa hyödynnettiin kyseisten teknologioiden virallista dokumentaatiota ja nettisivuja. Laajemmin testauksesta kertovissa teoriaosuuksissa hyödynnettiin kirjallisuutta ja erilaisia blogikirjoituksia aiheesta. Tässä työssä testauskohteiksi on valittu osana sovellusohjelmoinnin opintojaksoa tehty React-sovellus sekä yksi nettisivu. Tehdyt automaatiotestit on esitelty kuvina ohjelmakoodista ja niitä avataan tarkemmin tekstimuodossa.

Opinnäytetyön tuloksena syntyi ohjeet tarvittavien työkalujen asentamiseen vaihe vaiheelta sekä ohjeet asennuksien onnistumisen tarkistamiseen. Esimerkkitestit on rajattu elementtien etsimiseen verkkosivulta sekä sisäänkirjautumisen testaamiseen Robot Frameworkilla. Opinnäytetyössä on myös koottuna hyödyllistä teoriatietoa testaamisesta sekä testiautomaatiosta.

Asiasanat: Robot Framework, testiautomaatio, käyttöliittymätestaus, ohjelmistotestaus

South-Eastern Finland University of Applied Sciences

Degree title	Bachelor of Business Administration
Author (authors)	Elli Jukarainen
Thesis title	Introduction to Robot Framework and how to create UI test cases using Browser Library
Commissioned by	-
Time	2023
Pages	35 pages
Supervisor	Marjo Puikkonen

ABSTRACT

The objective of this thesis was to introduce the test automation tool Robot Framework and to produce instructions for setup and to create simple user interface automation tests using Robot Framework's Browser Library. The goal in this thesis was to explain the basic functionality of Robot Framework and how the syntax worked.

Software testing in general and test automation were explained in the theoretical section of this thesis. Theory was gathered from literature, the official documentation and blog posts. Setup instructions based on the official documentation of the frameworks from the web pages. Setup instructions were made step by step and introduced with screenshots. The tested user interfaces were self-made React application and one web page. User interface tests were presented with screenshots from the code editor and explained in the text sections.

The study resulted in guidelines for starting to use Robot Framework: for installing the necessary tools step by step and for testing the success of the installation. The test examples were defined to only include locating elements on the web page and testing login. This thesis gathered theory from making good test cases with Robot Framework.

Keywords: Robot Framework, test automation, user interface testing, software testing

SISÄLLYS

1 JOHDANTO 5

2 KESKEISET KÄSITTEET 6

2.1 Ohjelmistotestaus 6

2.2 Testitapaus 8

2.3 Testiautomaatio ja Robot Framework 9

2.4 Käyttöliittymätestaus ja Browser-kirjasto 10

3 TYÖKALUJEN ASENTAMINEN 11

3.1 Python 11

3.2 Robot Framework 13

3.3 Selaimen ajurit 15

3.4 Node.js 18

3.5 Browser-kirjasto 19

3.6 Koodieditori 20

4 TESTIEN TEKEMINEN 21

4.1 Testattavat käyttöliittymät 22

4.2 Elementtien etsiminen 24

4.3 Sisäänkirjautumisen testaus 30

5 YHTEENVETO 32

LÄHTEET 34

1 JOHDANTO

Tämän opinnäytetyön aiheena on Robot Frameworkin käyttöönotto ja testitapausten kirjoittaminen. Valitsin tämän aiheen, koska olen harjoittelun ja töiden kautta kiinnostunut siitä ja haluan syventää omaa osaamistani. Testitapaukset rajattiin käyttöliittymätesteihin, sillä olen itse lähinnä tehnyt rajapintatestausta Robot Frameworkilla. Opin enemmän tätä työtä tehdessäni, kun testien aiheena oli käyttöliittymätestit, ne voivat myös olla lukijoille tutumpia ja näin ollen madaltaa kynnystä kokeilla testien tekemistä itse.

Tällä työllä ei ole toimeksiantajaa ja valitsin aiheen itse. Tähän päätökseen vaikutti henkilökohtainen aikatauluni. Tämä aihe on kuitenkin yleishyödyllinen lukijoille, jotka haluavat aloittaa Robot Frameworkin käytön ja työn tekeminen oli itselleni oppimiskokemus. Testiautomaatio on ajankohtainen aihe ja tämä opinnäytetyö kokoaa yksiin kansiin Robot Frameworkin ja Browser-kirjaston käyttöönottoon tarvittavat asennusohjeet ja avaa asennuksen vaiheita tarkemmin lukijalle.

Opinnäytetyön aiheen keskiössä on testiautomaatioon käytettävän avoimen lähdekoodin viitekehys Robot Frameworkin käyttöönotto ja hyvien testitapausten luominen sillä. Testitapausten luonnissa keskitytään käyttöliittymätestien tekemiseen ja selaintestaukseen käytettävä kirjasto on Browser-kirjasto. Tämä opinnäytetyö vastaa kahteen tutkimuskysymykseen:

1. Kuinka ottaa Robot Framework käyttöön?
2. Miten kirjoittaa hyviä testitapauksia käyttöliittymiin Robot Frameworkilla?

Opinnäytetyön tuloksena syntyi ohjeellinen tuotos, jonka avulla lukija voi asentaa itselleen Robot Frameworkin Windows 10-käyttöjärjestelmällä ja siihen tarvittavat muut ohjelmistot, kuten Pythonin ja selaintestaukseen tarvittavat ajurit ja kirjastot. Tämä opinnäytetyö auttaa lukijaa myös pääsemään alkuun testiautomaation teossa, sillä opinnäytetyössä ohjeistetaan esimerkki käyttöliittymätestien tekeminen Robot Frameworkilla. Testitapausten luonti ohjeissa keskity-

tään tekemään testejä hyvien käytäntöjen mukaisesti ja lukija oppii mm. avainsanojen monikäyttöisyyden merkitystä sekä miten testejä ajetaan Robot Frameworkilla. Tämä työ antaa lukijalle myös valmiuksia tutkia erilaisia Robot Frameworkin tuottamia lokeja testiajoista. Hyvien testitapausten luomisessa käytän etsimääni teoriatietoa sekä omaa tietämystäni aiheesta.

Reflectorin (2022) mukaan testiautomaatio on ollut tunteita herättävä aihe testauksessa jo 10–15 vuoden ajan. Osassa yrityksiä testiautomaatio on otettu osaksi jokapäiväistä työarkea ja toisissa sitä ei tehdä lainkaan. Artikkelissa on listattu neljä yleisintä syytä testiautomaation aloittamisen ongelmiin, jotka ovat: testiautomaation aloittaminen kesken kehittämisen, liian kompleksinen ympäristö testiautomaatioon, testiautomaation tekemisen kustannukset ja niistä syntyvä hidas tuotto sekä resurssien puute. Artikkelin mukaan testiautomaatiolla saadaan testaukseen lisää tehokkuutta, tarkkuutta ja kattavuutta. Testiautomaatiolla voidaan mahdollistaa nopeampi julkaisu sekä palautteen saaminen. Kuitenkin testiautomaation lisäksi tarvitaan jonkin verran ihmisen tekemää manuaalista testausta. Tämän opinnäytetyön tavoitteena on, että sen lukija voi matalalla kynnyksellä alkaa tekemään testiautomaatiota Robot Frameworkilla ja saa ohjeet alkuun pääsemiseen sekä lisätiedon hakemiseen. Tavoitteena on myös ohjeistaa lukijaa tekemään hyviä testitapauksia Robot Frameworkilla ymmärtämällä sen syntaksin ja käyttötarkoituksen.

2 KESKEISET KÄSITTEET

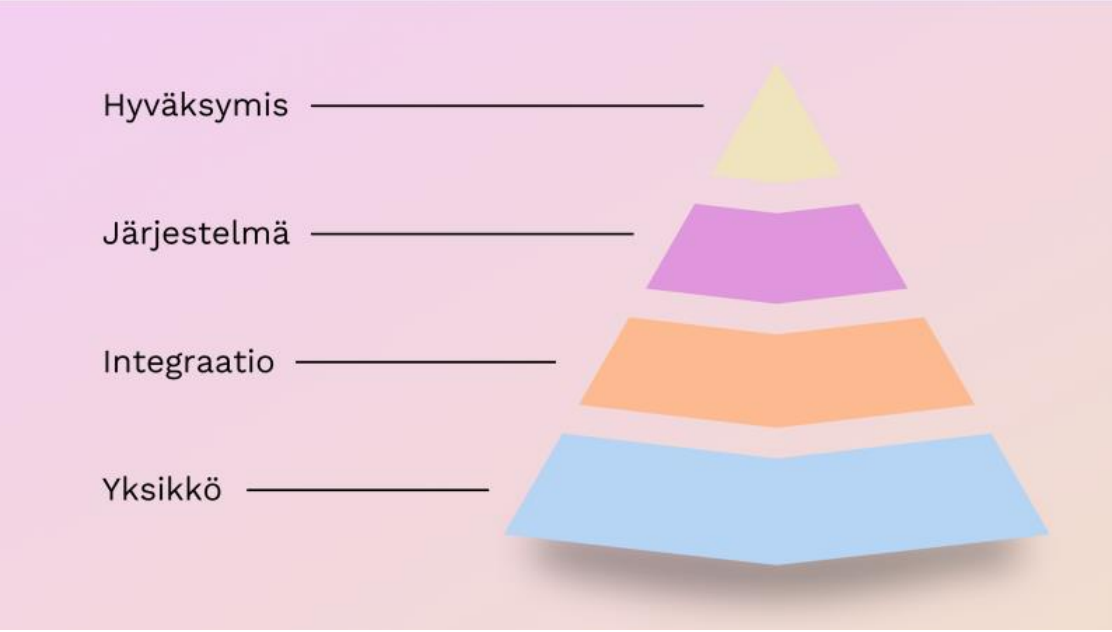
Tässä osiossa esitellään aihealueeseen liittyviä käsitteitä. Esiteltävät käsitteet liittyvät ohjelmistotestaukseen sekä testiautomaation työkaluihin, joita käytetään tämän opinnäytetyön toteutuksessa. Aihe on rajattu käyttöliittymätesteihin, joten se on myös olennainen osa tämän opinnäytetyön keskeisiä käsitteitä.

2.1 Ohjelmistotestaus

Ohjelmistotestausta tehdään, jotta voidaan varmistaa, että toteutettava tuote on sellainen kuin on toivottu ja sen ominaisuudet toimivat tarkoituksenmukaisesti. Ohjelmistotestauksen tehtävänä on tiivistettynä varmistaa, että tehdään oikeaa tuotetta ja tuote on tehty oikein. Testauksella pyritään tunnistamaan ne

osat, joissa tehty tuotos eroaa määrittelyistä. Ohjelmistotestaus eroaa kokonaisuutena hyvinkin paljon ohjelmoimisesta. Testaajan työ on monipuolista, siihen voi kuulua koodin kirjoittamista, dokumentaation tekemistä tai jopa koekäyttäjien haastattelua. Ohjelmistotestausta tehdään myös hyvin eri tavoilla eri yrityksissä, riippuen tuotteista ja yrityksen käytännöistä. Ohjelmistotestaus on hyvin tärkeä osa ohjelmistotuotantoa kannattavuuden kannalta. Ohjelmiston virheen korjaaminen suunnittelu- tai kehitysvaiheessa maksaa 1–2 prosenttia siitä, mitä se maksaa julkaisun jälkeen tehtynä. (Kasurinen 2017, 9.)

Ohjelmistotestauksen eri tasoja ovat yksikkötestaus, integraatiotestaus, järjestelmätestaus ja hyväksymistestaus (kuva 1). Yksikkötestauksessa keskitytään esimerkiksi jonkin koodin osan testaamiseen. Sillä varmistetaan ohjelman pienten osien toimivuus. Yksikkötestit ovat usein tehty kehittäjän toimesta kehitystyön ohessa testiautomaatiolla. Integraatiotestauksella taas testataan sitä, miten järjestelmän eri komponentit keskustelevat keskenään ja toimiiko niiden välinen yhteistyö. Systeemi- tai järjestelmätestauksella katetaan koko järjestelmä ja sen eri käyttötarkoitusten variaatioiden testaus. (VALA.)



Kuva 1. Ohjelmistotestauksen tasot (VALA 2022).

Edellä mainittujen ohjelmistotestauksen tasojen lisäksi ohjelmistotestaus jakaantuu myös testauksen eri osa-alueisiin. Ohjelmistotestauksen osa-alueita voidaan jaotella toiminnalliseen testaukseen, ei-toiminnalliseen testaukseen,

regressiotestaukseen, käytettävyystestaukseen ja tutkivaan testaukseen. (VALA.)

Toiminnallinen testaus eroaa ei-toiminnallisesta testauksesta siten, että toiminnallisessa testauksessa varmistetaan, että ohjelmisto toimii kuten on toivottu ja täyttää määritellyt vaatimukset loppukäyttäjän näkökulmasta, kun taas ei-toiminnallisessa testauksessa tutkitaan esim. suorituskykyä tai vikasietoisuutta. Regressiotestausta käytetään kehityksen aikana tapahtuvien virheiden löytymiseen. Sillä tarkistetaan, etteivät kehityksessä tehdyt muutokset ole rikkoneet jo toimivia osia. Käytettävyystestauksella tutkitaan nimensä mukaan ohjelmiston käytettävyyttä mm. helppokäyttöisyyden, intuitiivisuuden, loogisuuden, värien ja kontrastien näkökulmasta. Hyväksymistestauksen tekee usein asiakas, jolle ohjelmisto toimitetaan. Asiakas tai asiakkaan edustaja tarkistaa lähes valmiin tai valmiin tuotteen ja tutkii vastaako se vaatimuksia. Viimeisenä tasona mainittu tutkiva testaus taas edellyttää vähemmän suunnitelmallisuutta testauksessa, tämä testaus perustuu testaajan ammattitaitoon ja kokemukseen. Tutkivan testauksen aikana voi esimerkiksi pitää päiväkirjaa tehdyistä asioista ja havainnoista ja testauksen painopisteet on sovittu etukäteen. Tätä testausta tehdään monesta eri näkökulmasta ja se kattaa epätavallisetkin lähestymistavat. (VALA.)

2.2 Testitapaus

Testitapauksilla tarkoitetaan skenaarioita, joilla tutkitaan sovelluksen toimivuutta tietynlaisissa toiminnoissa ja olosuhteissa. Testitapaus siis sisältää erilaisia suoritettavia toimintoja, joilla varmistetaan ohjelmiston toiminnallisuus. Testitapauksia käytetään niin manuaalitesteissä kuin automaatiotesteissäkin. Testitapaus sisältää testin eri vaiheet ja sille voidaan tarvittaessa kehittää esi- ja jälkiolosuhteita tapauskohtaisesti. Testiskenaario voi olla mikä tahansa toiminnallisuus, joka voidaan testata ja testitapauksella testataan tämä tietty toiminnallisuus. (Visure s.a.)

Testitapaus sisältää kuvauksen syötteestä ja toiminnasta sekä odotetun tuloksen. Sillä kuvataan, miten voidaan validoida jokin tietty toiminnallisuus tai toiminto. Testitapauksen tulisi sisältää testille jokin tunniste, mitä testataan, oletukset, tarvittaessa testidata, suoritettavat testistepit, odotettu tulos, testissä

syntynyt tulos (suorituksen jälkeen), tieto menikö testi läpi vai ei sekä tarvittaessa kommentteja. (Software Testing Help s.a.)

Käyttöliittymien testauksessa testitapauksia voivat olla mm. navigaation testaus, käyttäjätunnuksen ja salasanan syöttäminen, kirjaudu-napin painaminen sekä käyttäjätunnuksen näkyminen sisäänkirjautumisen jälkeen sivulla. Erilaisilla testitapauksilla varmistetaan, että käyttäjä pystyy käyttämään sovellusta siten kuin on tarkoitettu. Käyttöliittymätestauksessa voidaan eri testitapauksilla tarkastella myös mm. fontin kokoa ja väriä sekä eri elementtejä, jotka muodostavat yhdessä käyttöliittymän. (Deshpande 2023.)

2.3 Testiautomaatio ja Robot Framework

Testiautomaatiolla tarkoitetaan sellaista testaustoiminnan muotoa, jossa ohjelman testaamiseen käytetään automaatiotyövälineitä testien tekemiseen. Tavoitteena on automatisoida toistuvasti tehtäviä testitapauksia. Tällä mahdollistetaan testaajien resurssien vapauttaminen, jos käytössä on esimerkiksi daily build, jossa tehdään joka yö uusi käännös. Automaatiotestit voidaan myös ajastaa ajettavaksi tiettyyn vuorokauden aikaan esim. yöllä ja näin ollen kehittäjät näkevät heti aamulla, onko testeissä löytynyt korjausta vaativia ongelmia. Automatisoidussa käyttöliittymätestauksessa suoritetaan sarja toimenpiteitä ja tarkistetaan, että lopputulos vastaa määrittelyn mukaisia hyväksymisehtoja. Testiautomaation kehittämistä voidaan verrata itse järjestelmän kehittämistyöhön vaativuudeltaan. (Kasurinen 2017, 49.)

Testiautomaatiolla on tarkoitus vähentää manuaalisesti tehtävän regressiotestauksen määrää. Vaikka testiautomaation rakentaminen saattaa olla hidasta ja sen ylläpito voi olla työlästä, on se usein toistettavissa testeissä kuitenkin manuaalitestausta kustannustehokkaampaa. Tyypillisiä testaustyyppejä automatisoinnille ovat savutestit, komponenttitestit sekä integraatiotestit. Automatisoiduilla regressiotesteillä varmistetaan, että aiemmin toimineet ohjelmiston osat eivät mene rikki kehitysprosessin aikana. (Kasurinen 2017, 50.)

Tämän työn yksi keskeisimmistä käsitteistä on Robot Framework, sillä sitä käytetään testiautomaation tekemiseen tässä työssä. Robot Framework on geneerinen avoimen lähdekoodin avainsanapohjainen viitekehys, jota voidaan

testiautomaation lisäksi käyttää myös ohjelmistorobotiikkaan (RPA). Robot Frameworkia tukee Robot Framework Foundation ja se on laajasti käytössä ohjelmistoalalla ja alan johtavissa yrityksissä. Robot Framework on ilmainen ja helppo käyttää, sekä siinä on helposti ymmärrettävä syntaksi ja avainsanat ovat ihmislukuisia. (Robot Framework UA s.a.)

Robot Framework on sen suosion takia olennainen osa ohjelmistotestaajan taitoja. Se on käytössä etenkin hyväksymistestauksessa ja hyväksymistestausvetoisessa kehityksessä. Alun perin Robot Frameworkin kehitys alkoi Pekka Klärckin diplomityöstä vuonna 2005 ja avoimen lähdekoodin versio julkaistiin vuonna 2008. (Kupila 2022.) Tässä opinnäytetyössä keskitytään käyttöliittymien testaamiseen. Robot Frameworkia voi kuitenkin käyttää monipuolisesti myös muihin tarkoituksiin. Se tarjoaa omia ”built-in” -kirjastoja, joita voi hyödyntää mm. tietokantojen sekä rajapintojen testaamiseen (Belton 2023).

2.4 Käyttöliittymätestaus ja Browser-kirjasto

Käyttöliittymä on käyttöjärjestelmän, ohjelman tai laitteen osa, jonka avulla käyttäjä syöttää ja vastaanottaa tietoa. Käyttöliittymiä on olemassa merkkipohjaisia ja graafisia käyttöliittymiä. (Helsingin yliopisto s.a.) Tässä työssä keskitytään graafisiin käyttöliittymiin, koska se on useimmissa järjestelmissä käytössä.

Käyttöliittymien testaamisella saadaan tietoa siitä, miten ohjelmisto toimii loppukäyttäjän näkökulmasta. Käyttöliittymätestauksessa voidaan keskittyä pääpiirteittäin käytettävyyteen, saavutettavuuteen, johdonmukaisuuteen ja yhteensopivuuteen. Testauksella varmistetaan, miten ohjelmisto toimii hiiren ja näppäimistön kanssa ja toimivatko halutut interaktiiviset graafiset elementit esim. kuvat, tekstit, napit ja menut. Käyttöliittymätestauksella tutkitaan mm. pystyykö käyttäjä selaamaan ohjelmistoa ilman ongelmia tai ohjelmiston kaatumista, sekä onko käyttöliittymä helppo ymmärtää ja käyttää. Vaikka ohjelmiston muut testit menisivät läpi ja se olisi määrittelyjen mukainen, se on hyödytön, jos käyttäjä ei voi käyttää sitä. (Hruska 2019.)

Automatisoiduilla käyttöliittymätesteillä voidaan saavuttaa monia hyötyjä. Käyttöliittymätestien automatisoiminen säästää aikaa, antaa nopeammin palautetta ja parantaa testien tarkkuutta. Testejä voi suorittaa useille komponenteille samanaikaisesti. Lisäksi palautteen saa heti testien ajon jälkeen, joka on huomattavasti nopeampaa kuin manuaalisesti tehty käyttöliittymätestaus. Testitulosten dokumentointia helpottaa automaatiotesteistä saatavat raportit. (BrillMindz 2022.)

Käyttöliittymätestausta voi tehdä Robot Frameworkilla hyödyntäen Selenium-kirjastoa tai tähän työhön valitsemaani Browser-kirjastoa. Valitsin Browser-kirjaston vanhemman Selenium-kirjaston sijaan, koska se on nopea ja luotettava sekä kehitetty 2020-vuodelle. Browser-kirjasto pohjautuu PlayWrightiin. Browser-kirjasto tarjoaa avainsanoja selaimella tehtäviin käyttöliittymätesteihin, joita hyödynnetään tässä opinnäytetyössä. Tämän kirjaston käyttämistä varten käyttäjän tulee asentaa myös Python ja Node.js (Browser Library s.a.).

3 TYÖKALUJEN ASENTAMINEN

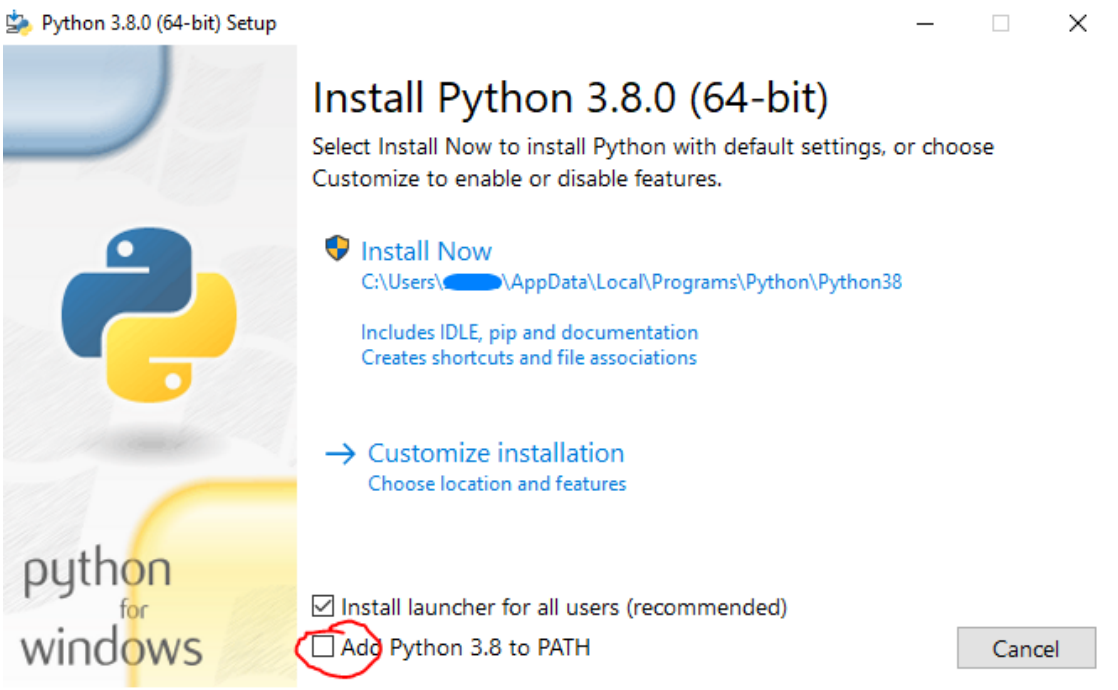
Tässä osiossa esitellään testiautomaation ja käyttöliittymätestien tekemiseen tarvittavien työkalujen asentaminen Windows 10-järjestelmälle. Asentaminen havainnollistetaan tekstin ja kuvien avulla. Kaikki asennukseen tarvittavat ladattavat tiedostot ja työkalujen omat ohjeet löytyvät internetistä hakemalla työkalun nimellä. Asennettavan työkalun asennusohjeiden jälkeen on myös ohjeet asennuksen onnistumisen tarkistamiseen.

3.1 Python

Robot Framework pohjautuu Pythoniin ja näin ollen se vaatii toimiakseen Pythonin asentamisen. Pythonin asentamista varten tulee netistä etsiä Pythonin nettisivut esimerkiksi kirjoittamalla Google-hakuun ”Python”. Sivulta valitaan omalle käyttöjärjestelmälle sopiva asennuspaketti ja ladataan se. Kuvassa 2 nähtävillä Pythonin virallisen asennussivun näkymä.

Kuva 2. Kuva Download Python (python.org s.a.).

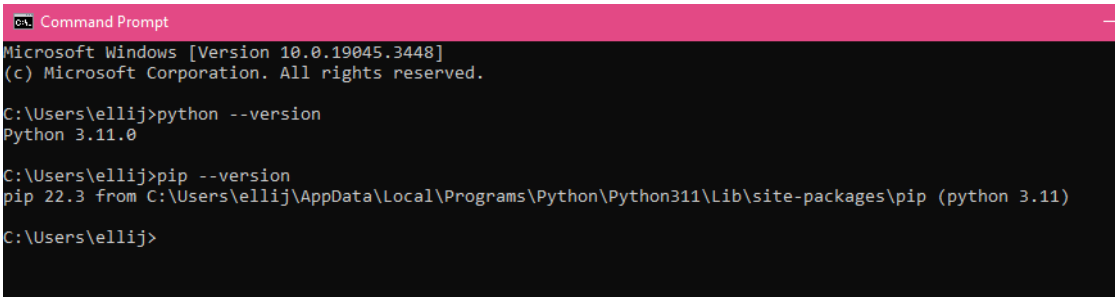
Keltaisesta ’’Download Python x.xx.x” -napista voi ladata asennustiedoston. Tämän jälkeen tulee suorittaa tietokoneelle ladattu asennustiedosto. Pythonin asennuksessa tulee valita ’’Add Python x.x to PATH”, näin ollen Python skriptejä voi ajaa ja pääset käsiksi Python paketteihin mistä vain hakemistosta komentopäätteellä. Pythonin lisäys PATH:iin onnistuu asennuksen yhteydessä valitsemalla kuvassa 3 näkyvän punaisella ympyröidyn valintaruudun.



Kuva 3. Kuva Using Python on Windows (Python setup and Usage s.a.).

Valittavissa on kustomoitu asennus tai nopeampi ”Install Now” -toiminto. Lisää tietoa Pythonin asennuksesta Windowsille löytyy verkosta Pythonin ohjeista, jotka löytyvät Pythonin nettisivuilta ”Documentation” kohdan alta.

Asennuksen onnistumisen voi tarkistaa avaamalla tietokoneen komentokehotteen kirjoittamalla tietokoneen hakuun *Command Prompt* ja sen jälkeen suorittamalla komennon *python --version* kuvassa 4 näkyvällä tavalla. Komento kirjoitetaan komentokehotteelle ja painetaan sen jälkeen enteriä.

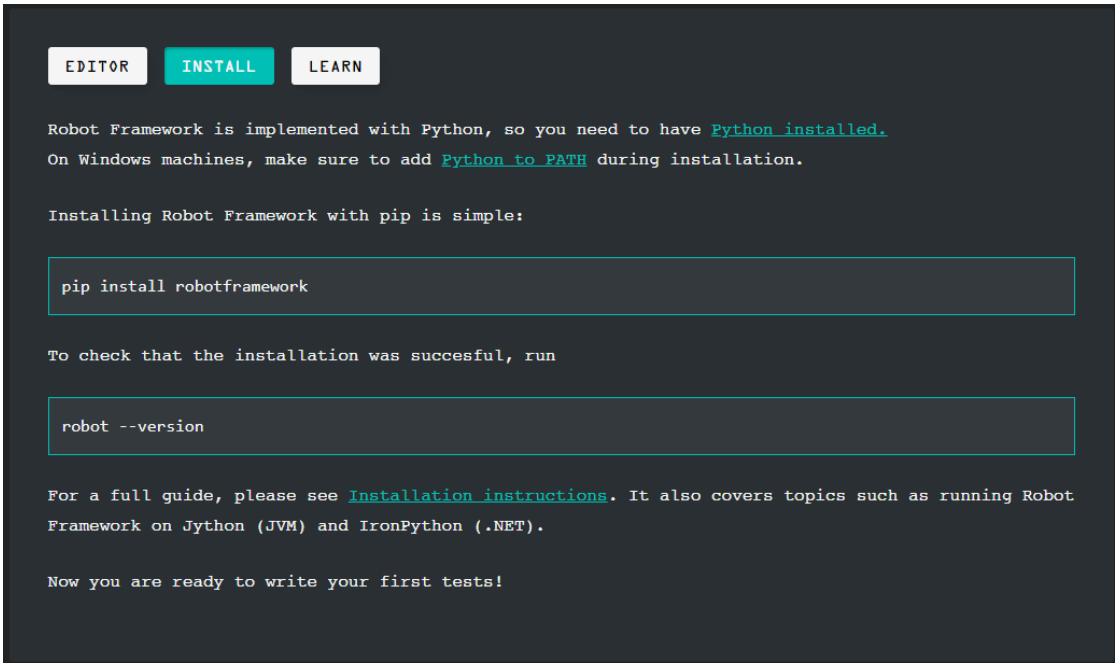


Kuva 4. Kuva Pythonin version tarkistaminen.

Vaihtoehtoisesti komentokehotteeseen voi kirjoittaa *pip --version*. Pip on Pythonin pakettienhallintajärjestelmä, joka asentuu Python asennuksen yhteydessä. Kuvassa 4 näkyy kuva omalta tietokoneeltani, käytössä oleva versioni Pythonista on 3.11.0 ja Pipistä 22.3. Pythonin nettisivuilta voi tarkistaa versioiden tiedot ja tarvittaessa päivittää Pythonin ja Pipin.

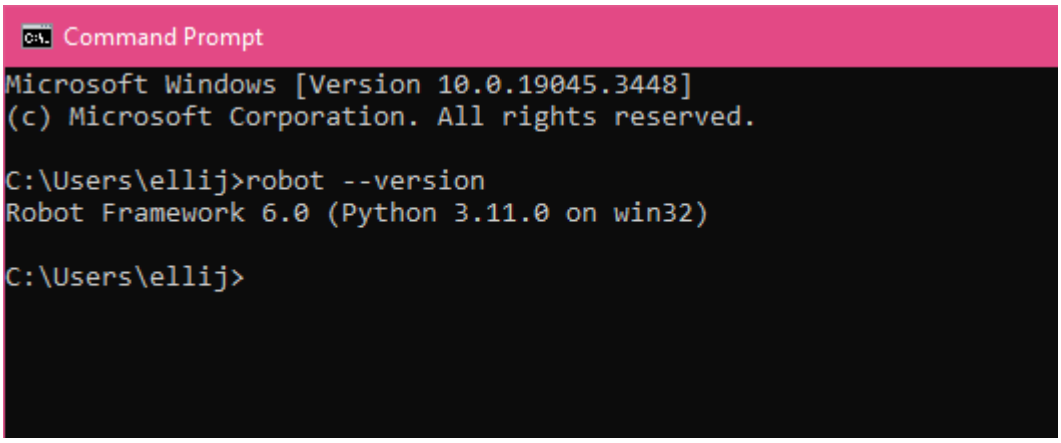
3.2 Robot Framework

Kun Python on asennettu ja asennuksen onnistuminen tarkistettu esimerkiksi version tarkistamisella, voi seuraavaksi asentaa Robot Frameworkin. Robot Frameworkilla on englanninkieliset asennusohjeet omilla sivuillaan, jotka löytyvät hakemalla Googlesta ”Robot Framework”. Sivujen Getting Started-kohdasta valitsemalla Install-napin, tulee näkyviin kuvan 5 mukaiset asennusohjeet.



Kuva 5. Kuva Robot Frameworkin asentaminen (Robot Framework UA s.a.).

Robot Frameworkin voi asentaa käyttämällä Pythonin Pip-pakettienhallintajärjestelmää. Robot Frameworkin voi siis asentaa kirjoittamalla komentokehotteelle *pip install robotframework* ja painamalla enteriä. Tämän jälkeen tulee asennuksen onnistuminen tarkistaa kuvan 6 mukaisesti.



Kuva 6. Kuva Robot Frameworkin version tarkistaminen.

Robot Frameworkin asennuksen voi tarkistaa kirjoittamalla komentokehotteelle *robot --version*. Käytössäni on tällä hetkellä Robot Frameworkin versio 6.0. Lisätietoja Robot Frameworkin versioista löytyy GitHubista.

3.3 Selaimen ajurit

Jotta voidaan testata websivuja käyttöliittymätesteillä, tulee tietokoneelle olla asennettuna selaimen ajurit. Itse käytän esimerkkitesteissä Chromea selaimena. Chromen selainajurit löytyvät kirjoittamalla Googlen hakuun ”chrome webdrivers” ja ChromeDriver-sivusto ohjaa käyttäjän oikeaan paikkaan riippuen Chromen versiosta (Kuva 7). Oman Chrome version voi tarkistaa selaimen asetuksista ”Tietoja Chromesta” tai ”About Chrome” -kohdasta. Jos Chrome versio on vanhempi kuin ajureiden asennussivulla näkyvä versio, tulee ensin päivittää Chrome selaimen versio ajan tasalle.

Channel	Version	Revision	Status
Stable	117.0.5938.149	r1181205	✅
Beta	119.0.6045.9	r1204232	✅
Dev	118.0.5993.3	r1192594	✅
Dev (upcoming)	119.0.6034.6	r1201974	❌
Canary	120.0.6051.2	r1206341	✅
Canary (upcoming)	120.0.6053.0	r1206788	❌

Kuva 7. Kuva selainversioista (chrome-for-testing s.a.).

Kuvassa 7 näkyy, että sivuilta löytyy eri versioita selainajureista. ”Stable” versio on käytössä oleva Chromen versio ja muut ovat sitä varten, että voidaan testata esimerkiksi tulevaa versiota etukäteen ja varmistaa, että sovellus toimii tulevalla versiolla eivätkä muutokset riko sovellusta. Kun klikkaa ”Stable”-kohdasta, sivusto ohjaa uudelle listalle (kuva 8).

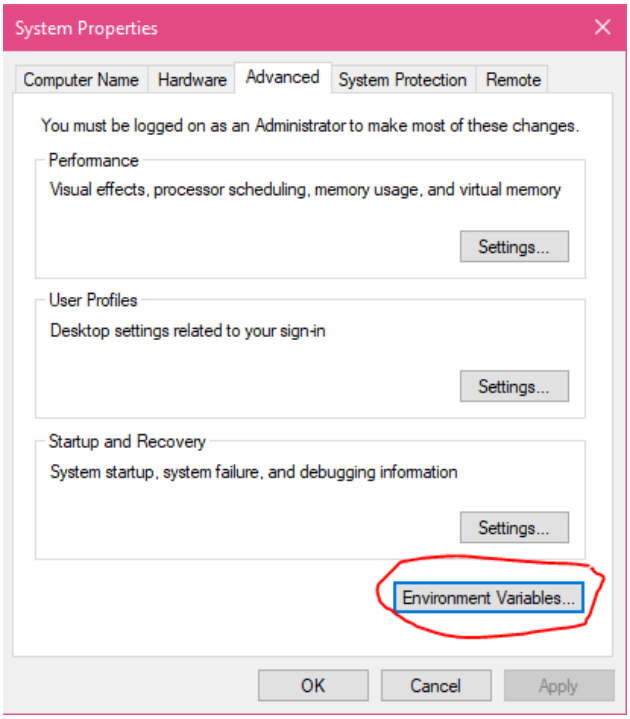
Binary	Platform	URL	HTTP status
chrome	linux64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/117.0.5938.149/linux64/chrome-linux64.zip	200
chrome	mac-arm64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/117.0.5938.149/mac-arm64/chrome-mac-arm64.zip	200
chrome	mac-x64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/117.0.5938.149/mac-x64/chrome-mac-x64.zip	200
chrome	win32	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/117.0.5938.149/win32/chrome-win32.zip	200
chrome	win64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/117.0.5938.149/win64/chrome-win64.zip	200
chromedriver	linux64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/117.0.5938.149/linux64/chromedriver-linux64.zip	200
chromedriver	mac-arm64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/117.0.5938.149/mac-arm64/chromedriver-mac-arm64.zip	200
chromedriver	mac-x64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/117.0.5938.149/mac-x64/chromedriver-mac-x64.zip	200
chromedriver	win32	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/117.0.5938.149/win32/chromedriver-win32.zip	200
chromedriver	win64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/117.0.5938.149/win64/chromedriver-win64.zip	200

Kuva 8. Kuva selainajureista (chrome-for-testing s.a.).

Tältä listalta tulee etsiä oma käyttöjärjestelmä, joka näkyy ”Platform” -kohdassa. Valitsin ladattavaksi win64. Kun omalle käyttöjärjestelmälle sopiva versio on löytynyt listalta, tulee maalata kuvassa näkyvä URL kuvan 8 mukaisesti ja avata se uudella välilehdellä. Näin ajureiden lataus käynnistyy.

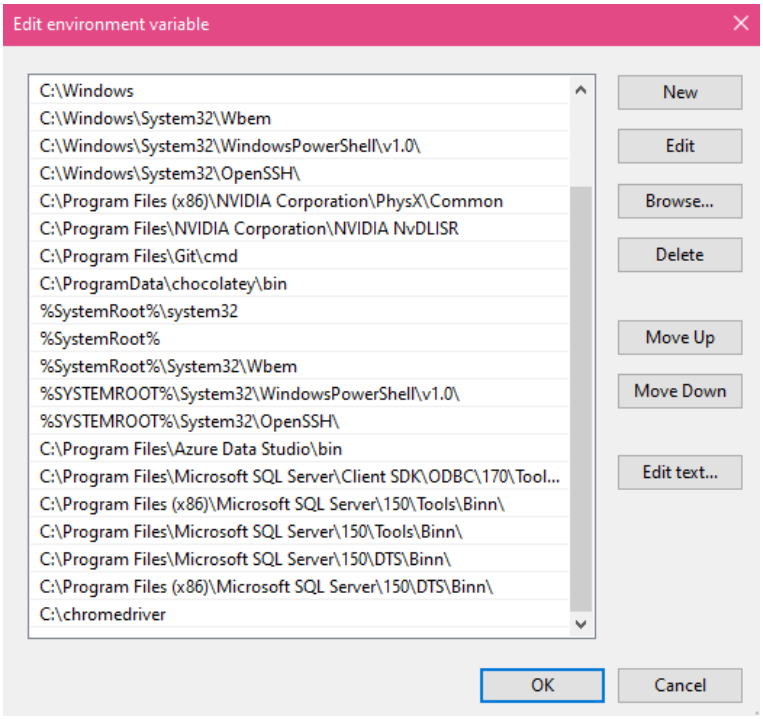
Tämän jälkeen ladatun kansion voi purkaa omalle tietokoneelle. Olen kohdannut todella paljon ongelmia aikaisemmin yrittäessäni saada selainajurit toimimaan asianmukaisesti. On olemassa monia tapoja ottaa ne käyttöön Robot Frameworkilla, mutta käyn tässä läpi oman toimivaksi todetun tapani, jolla ei tarvitse ottaa ajureita käyttöön testien ohjelmakoodissa vaan Robot Framework saa ne käyttöön muualta.

Jotta Robot Framework löytää Windowsissa selainajurit, tulee ne asettaa PATH:iin. Olen laittanut puretusta ajureiden kansiosta chromedriver-nimisen tiedoston Windows C-asemalle tekemääni chromedriver-nimiseen kansioon. Tämän jälkeen olen mennyt asettamaan sen ympäristö muuttujiin. Windowsin hakukenttään voi kirjoittaa *Edit the system environment variables* ja avata sen (kuva 9).



Kuva 9. Kuva Windowsin system properties.

Edellisen kappaleen ohjeita seuraamalla pitäisi seuraavaksi avautua kuva 9 mukainen näkymä tietokoneelle. Kuvassa on ympyröity punaisella kohta, jota tuee seuraavaksi painaa, tämän jälkeen tietokoneelle avautuu environment variables, eli ympäristömuuttujat. Tässä kohtaa on näkyvissä kaksi allekkaista listaa, alemmasta ”System variables”-listasta tulee etsiä Path-niminen muuttuja, klikata sitä ja painaa ”Edit…” -nappia, jonka jälkeen avautuu ympäristömuuttujien muokkausnäkymä (kuva 10).

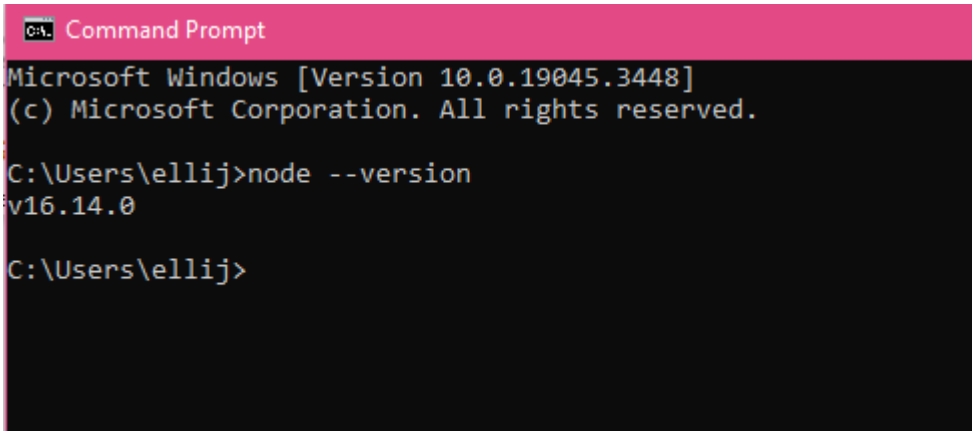


Kuva 10. Kuva Windows Edit environment variable.

Alimmaisena kuvan 10 listalla näkyy C:\chromedriver-kansio, johon olen ladatut selainajurit sijoittanut. Ajurien sijainnin pääsee lisäämään listaan painamalla ”New” -nappia ja kirjoittamalla kansion sijainnin riville. Tämän jälkeen tulee painaa ”OK” -nappia ja tehdä samoin muille ikkunoille, jotka ovat taustalla jääneet auki. Tietokone on hyvä vielä käynnistää uudestaan näiden muutosten jälkeen, jotta ne astuvat voimaan.

3.4 Node.js

Seuraavassa kappaleessa esiteltävää Browser-kirjaston asennusta varten tarvitaan Node.js asennettuna tietokoneelle. Node.js:n asennukseen tarvittavat tiedot löytyvät kirjoittamalla Googleen ”Node.js”. Sivuilta tulee valita omalle käyttöjärjestelmälle sopiva asennuspaketti ja ladata se tietokoneelle. Sivustolla voi valita kahdesta vaihtoehdosta, joko LTS (long time support) eli suositeltu suurimmalle osalle käyttäjistä tai Current joka on uusin versio, eikä välttämättä stabiili. Valitsisin tässä tapauksessa LTS asennuksen. Tämän jälkeen tulee suorittaa ladattu asennustiedosto ja tarkistaa asentamisen onnistuminen (kuva 11).



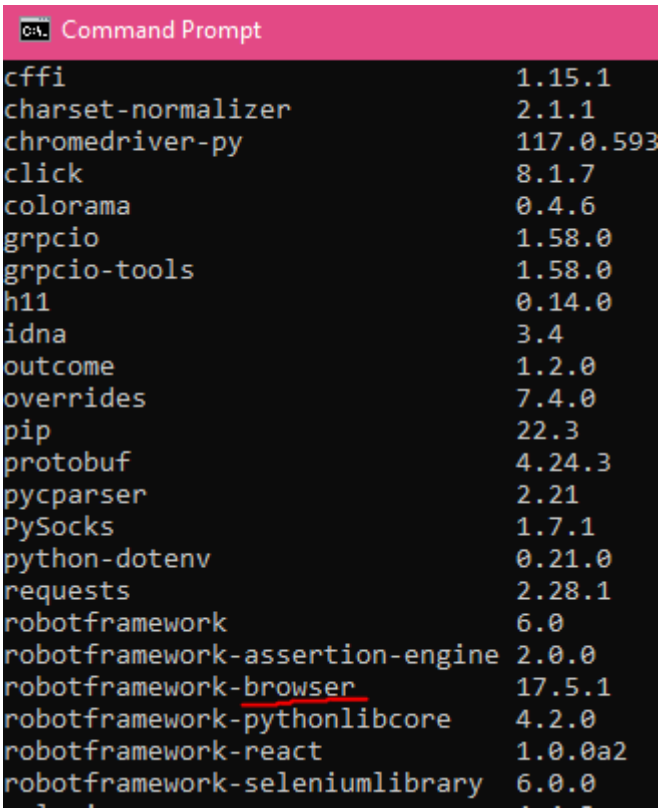
Kuva 11. Kuva Node.js version tarkistaminen.

Node.js asennuksen voi tarkistaa aikaisemmista asennuksista tuttuun tapaan kirjoittamalla omalle komentokehotteelle komento *node --version*. Vastauksena tulee nykyinen Node.js:n asennettu versio, käytössä oleva versioni on v16.14.0.

3.5 Browser-kirjasto

Selaintestejä varten tässä työssä käytetään apuna Browser-kirjastoa, joka tarjoaa avainsanoja testejä varten. Node.js asennuksen jälkeen voidaan asentaa tietokoneelle Browser-kirjasto. Sen asennusohjeet on kerrottu Browser-kirjaston sivuilla, jotka löytyvät hakemalla Googlesta ”Browser library”.

Asennus voidaan tehdä kahdella eri tavalla. Toisessa selainbinäärit sisältyvät kirjastoon ja toisessa niitä käsitellään kirjaston ulkopuolella. Tässä tapauksessa valitaan asennus selainbinäärien kanssa. Asennus suoritetaan kirjoittamalla komentokehotteelle *pip install robotframework-browser*. Tämän jälkeen tulee suorittaa asennuksen toinen kohta, jossa komentokehotteelle kirjoitetaan *rfbrowser init*. Asennetut kirjastot saa näkyviin esimerkiksi kirjoittamalla komentokehotteelle *pip list*, jonka tuloksena tulee Python pakettienhallinnan lista, jossa tulisi näkyä Browser-kirjasto (kuva 12).

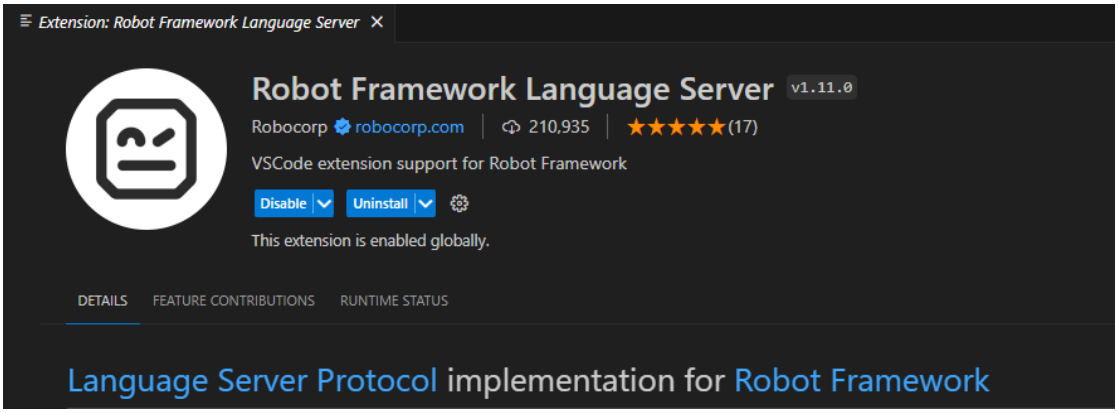


Kuva 12. Kuva robotframework-browser version tarkistaminen.

Käyttämäni Browser-kirjaston versio on 17.5.1. Listalla näkyy myös muita asennuksia, esimerkiksi Selenium-kirjasto, jolla voi myös toteuttaa käyttöliittymätestejä Robot Frameworkia käyttäen.

3.6 Koodieditori

Robot Framework testien kirjoittamiseen tarvitaan jokin koodieditori. Olen itse käyttänyt tähän Visual Studio Codea. Robot Frameworkin kanssa voi käyttää myös jotain toista editoria, mutta tässä opinnäytetyössä ohjeistetaan Visual Studio Coden asentaminen ja käyttö. Visual Studio Coden voi ladata sen omilta nettisivuilta, jotka löytyvät Googlesta ja suorittaa asennuksen. Visual Studio Codessa on laajennoksia (extensions), joita voi hyödyntää Robot Frameworkin kanssa (Kuva 13).



Kuva 13. Kuva Robot Framework Language Server.

Suosittelen asentamaan Visual Studio Coden extensions kohdasta Robot Framework Language Server -laajennoksen. Laajennos helpottaa Robot Frameworkillä työskentelyä. Visual Studio Codessa kannattaa muuttaa muutamia asetuksia, jotta tämän laajennoksen käyttäminen olisi sujuvampaa. Asetuksia pääsee muuttamaan menemällä Visual Studio Coden asetuksiin ja etsimällä sieltä Extensions-kohdan ja sen alta Robot Framework Language Server Configuration. Jotta voi käyttää automaattista koodin katselmointia, tulee valita *Robot > Lint: Enabled*, *Robot > Lint: Unused Keyword* ja *Robot > Lint > Robocop: Enabled*. Muutosten jälkeen tulee Visual Studio Code käynnistää uudestaan. Näitä käyttämällä saa tueksi Robot Frameworkin koodianalysaattorin, joka mm. ilmoittaa virheistä tai ehdottaa parannusehdotuksia.

4 TESTIEN TEKEMINEN

Robot Frameworkilla tehtyjen testien päätavoitteina on olla helposti ymmärrettäviä, helposti ylläpidettäviä ja nopeasti suoritettavia. Testitapausten, testisuitejen, avainsanojen ja muuttujien nimeämiseen on tärkeää kiinnittää huomiota. Hyvä nimi tässä yhteydessä on selkeä ja helposti ymmärrettävä, johdonmukainen sekä sen tulisi kertoa mitä tehdään sen sijaan, että se kertoisi miten tehdään. Hyvin nimettyjen testitapausten, jotka on tehty käyttäen hyvin nimettyjä avainsanoja lisäksi ei tulisi tarvita ylimääräistä dokumentaatiota. Yleiskäyttöiset avainsanat on kuitenkin hyvä dokumentoida toiminnallisuuden kannalta. (Klärck 2014.)

Testeissä käytettävien avainsanojen tulisi olla mahdollisimman kuvaavia, jotta voidaan vähentää esimeriksi kommentoinnin tarvetta. Esimerkiksi avainsana ”Open browser”, kertoo mitä se tekee, joten se ei tarvitse lisäkseen selittävää kommenttia selaimen avaamisesta. Avainsanoissa tulee käyttää muuttujia kovakoodauksen sijaan tarpeen mukaan, niiden ylikäyttö ei ole suositeltavaa. (Klärck 2014.)

Samassa testisuitessa olevien testien tulisi liittyä toisiinsa. Tätä voi pohtia testien alustuksen ja/tai testin päättyessä tehtävän ”teardown”:in kautta. (Klärck 2019.) Testien alustuksessa voidaan esimerkiksi avata selain tietylle sivulle tai muodostaa yhteys johonkin palveluun tai vaikka tietokantaan, alustus suoritetaan ennen testien ajoa. Jos tarvitaan siivousta testin jälkeen, se tapahtuu teardown-osiossa, jotta voidaan varmistaa sen suorittaminen. (Klärck 2019.) Testien siivoamiseen voi kuulua esimerkiksi tietokantatesteissä tallennetun datan tyhjentäminen tietokannasta tai selaintesteissä selaimen sulkeminen. Browser-kirjasto osaa sulkea selaimen automaattisesti, mutta jos käyttää Selenium-kirjastoa, tulee selaimen sulkeminen laittaa testien jälkeen teardown-osioon. Normaalisti yhdessä testisuitessa maksimi testien määränä pidetään kymmentä testiä, pois lukien data-driven tyyppiset testit. Testien tulisi olla itsenäisiä ja niiden mahdollinen initialisointi tulisi tehdä käyttäen setup- tai teardown-osioita. (Klärck 2019.)

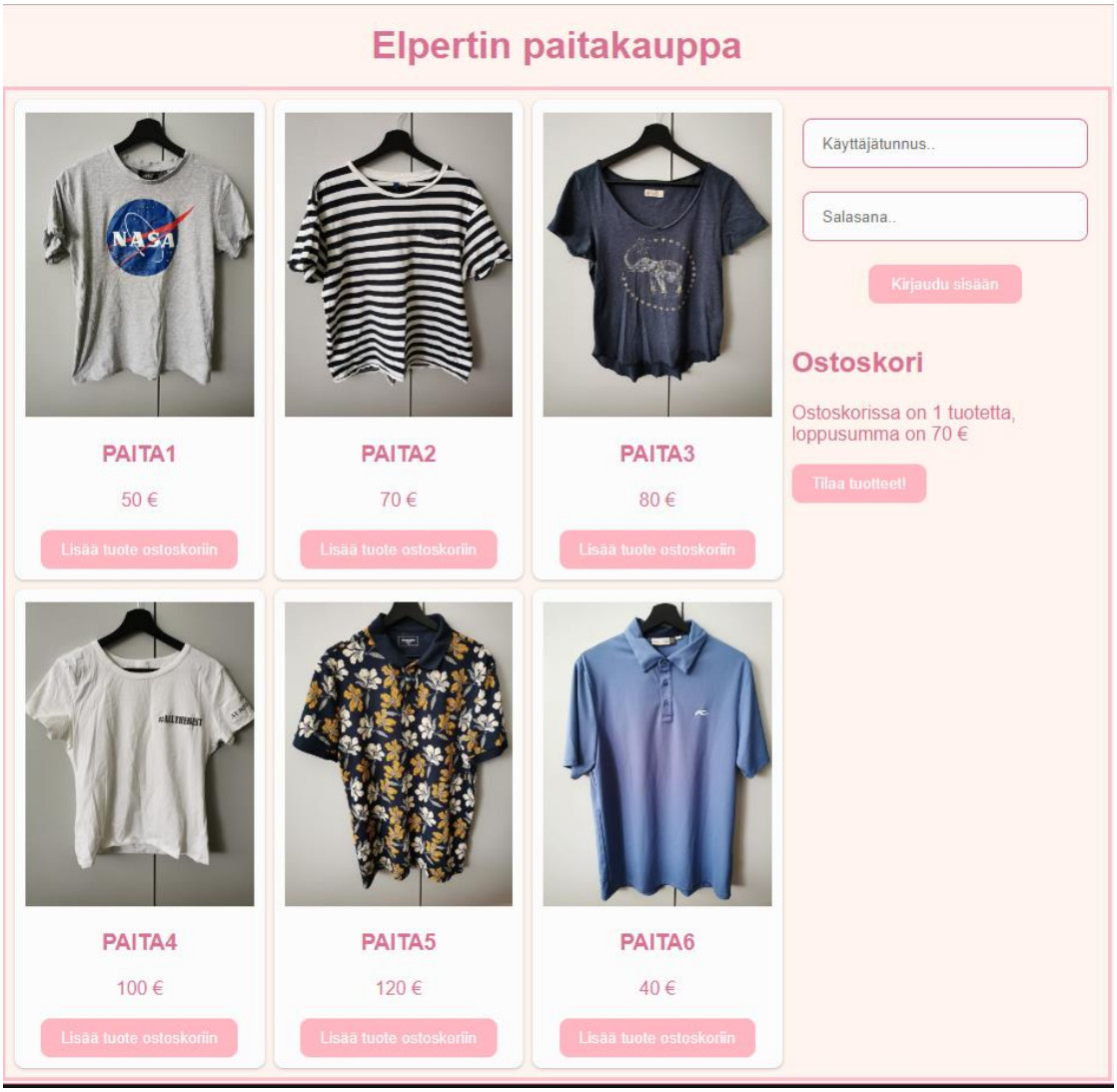
Yhden testin tulisi testata yhtä asiaa. Testattava asia voi olla pieni, jossa testataan jotain osaa ominaisuudesta tai iso esimerkiksi end-to-end testit. Testejä

on kahdenlaisia, workflow testit ja data-driven testit. (Klärck 2019.) Tässä opinnäytetyössä tehdään testejä workflow tyylillä. Workflow testeissä on tyypillisesti tarvittaessa esiehto suiten setupissa, jokin toiminto ja sen jälkeen tuloksen validointi.

Robot Frameworkilla on hyvä user guide, jossa on selitetty kaikki sen toiminnallisuudet, lisäksi eri kirjastoille löytyy hyvin dokumentaatiota netistä. Usein avainsanoja käytetään mm. BuiltIn kirjastosta ja tässä työssä myös selaintestaukseen Browser-kirjastosta. Lisää tietoa kirjastoista löytyy Robot Frameworkin sivuilta kohdasta Docs ja Standard Libraries, BuiltIn kirjastolle on oma linkki samassa paikassa. Python ohjelmointikielellä voi myös tehdä omia kirjastoja käytettäväksi testeissä.

4.1 Testattavat käyttöliittymät

Tämän opinnäytetyön tarkoituksena on havainnollistaa esimerkkitestein, kuinka testata käyttöliittymää Robot Frameworkilla. Tämän takia esiteltävät testit ovat yksinkertaisia. Valitsin yhdeksi testauskohteeksi sovellusohjelmoinnin opintojaksolla tekemäni React-sovelluksen (kuva 14). Sovellus on paitakauppa, jossa on tuotteita myynnissä ja ne voi lisätä ostoskoriin. Lisäksi sovellukseen voi kirjautua testitunnuksilla ja tällöin paidoista saa alennusta. Virheellisellä käyttäjätunnuksella ja salasanalla ei pääse kirjautumaan, vaan käyttäjälle tulee virheilmoitus näkyviin. Ostoskorissa näkyy tuotteiden loppusumma ja siitä voi jatkaa tuotteiden tilaukseen, joka johtaa vain ilmoitukseen ”jatketaan tilaukseen..”. Sovelluksessa voi myös kirjautua ulos.



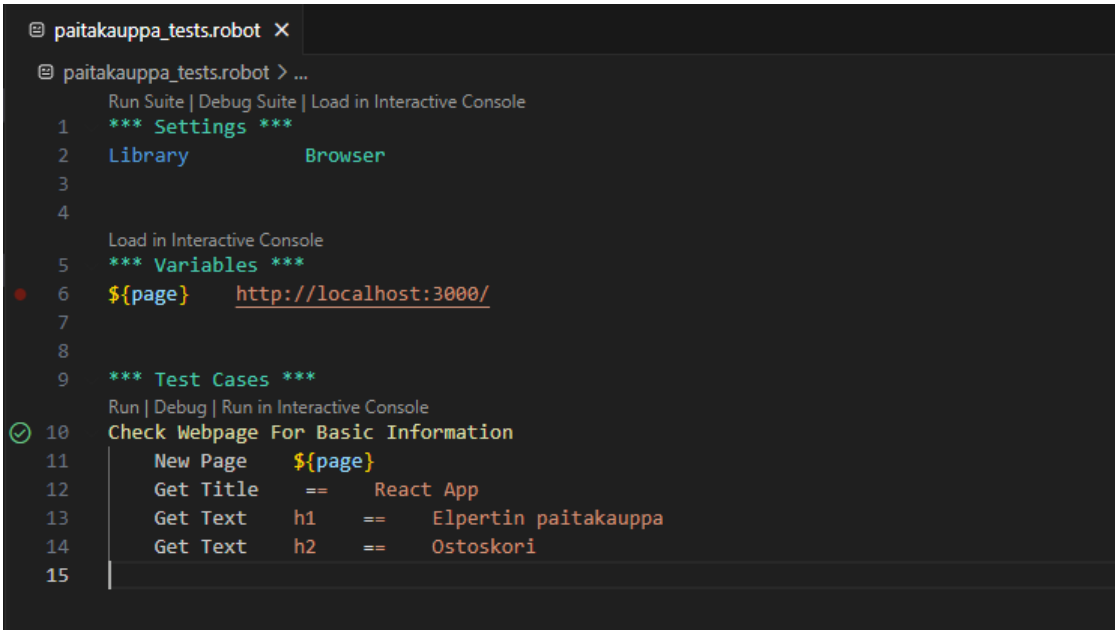
Kuva 14. Kuva testattavasta sovelluksesta.

Kuvassa 14 on näkymä paitakaupan käyttöliittymästä. Kuvassa ei olla kirjauduttu sisään ja ostoskoriin on lisätty yksi tuote. Sovellus koostuu yhdestä päänäkymästä, eikä siinä esimerkiksi ole erillistä navigaatiota, vaan komponentit näytetään yhdellä sivulla.

Valitsin toiseksi testauksen kohteeksi Xamkin nettisivut, jotta testien havainnollistaminen olisi monipuolisempaa. Xamkin nettisivut ovat paitakauppaa laajemmat sisällöltään ja toiminnallisuudeltaan. Erilaisia elementtejä on suhteessa paitakauppa sovellukseen paljon enemmän ja sivulla voi liikkua eri näkymien välillä.

4.2 Elementtien etsiminen

Luodaan ensin testi, jossa tarkistetaan peruselementtejä sovelluksesta. Testien tekeminen aloitetaan luomalla tiedosto Visual Studio Codeen. Tein tiedoston nimeltä paitakauppa_tests.robot, Robot Frameworkilla ajettavat testitiedostot ovat .robot päätteisiä (kuva 15). Näiden lisäksi voi tehdä esimerkiksi .resource tiedostoja, joihin voi tehdä mm. omia avainsanoja ja tuoda ne testitiedostoihin käytettäväksi Settings-osiossa.

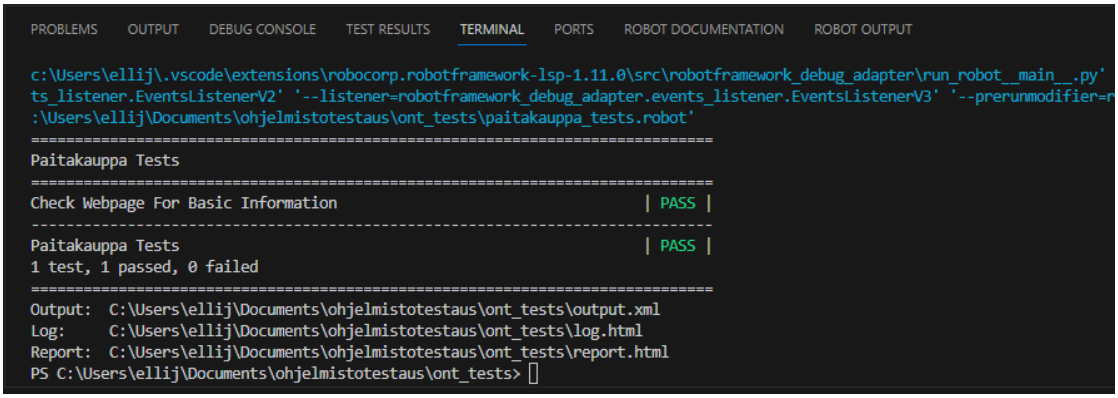


Kuva 15. Kuva esimerkkitestistä Check Webpage For Basic Information.

Robot Framework testin rakenteessa on ensimmäisenä Settings-osio, sen jälkeen Variables-osio ja sitten Test Cases-osio. Robot Frameworkin syntaksissa näihin laitetaan ympärille molemmin puolin kolme * -merkkiä ja väleinä käytetään neljää välilyöntiä tai tabulaattorin painallusta. Settings-osioon olen tuonut käytettävän Browser-kirjaston. Tässä esimerkissä on Variables-osioon laitettu havainnollistamista varten muuttujaan testissä käytettävä osoite. Sivu avataan tässä yhdessä testissä vain kerran, joten oikeasti tuo on tarpeeton tässä yhteydessä. Test Cases-osiossa on ensimmäisenä testin nimi “Check Webpage For Basic Information”. Tämän jälkeen on Browser-kirjaston avainsana ”New Page” joka käyttää muuttujaa ”${page}”, jolle on asetettu arvoksi tässä tapauksessa localhost:3000, koska React-sovellus löytyy sieltä. Tämän jälkeen on kolme erilaista tarkistusta sovellukselle, jotka on toteutettu Browser-kirjaston avainsanoilla ”Get Title” ja ”Get Text”. Näissä tarkistetaan, että

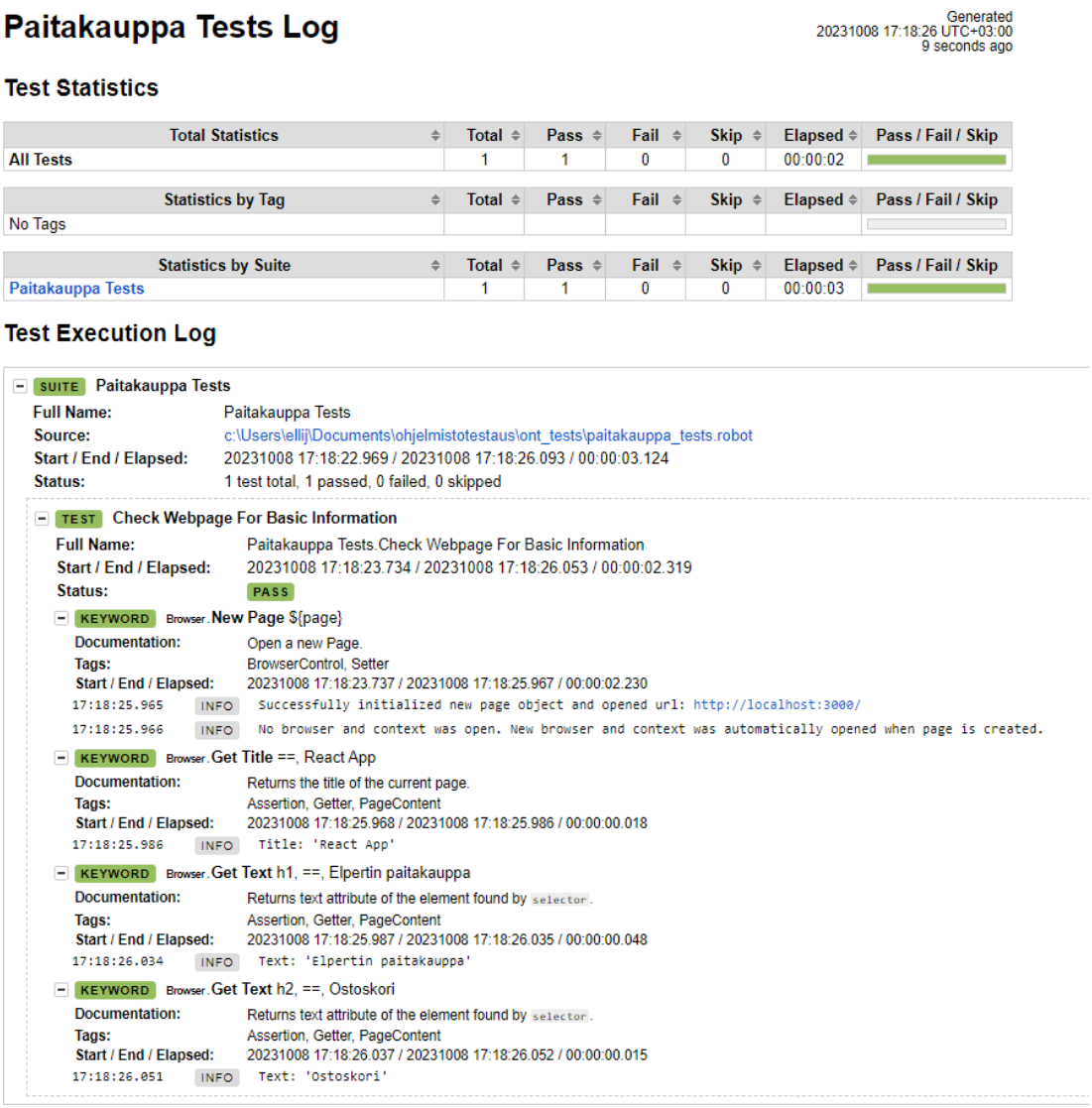
ne ovat yhtä kuin avainsanaan kirjoitetut arvot. Tekstit etsitään tässä käyttäen h1 ja h2 html-tageja elementtien löytämiseen.

Testi voidaan ajaa klikkaamalla testitapauksen nimen yläpuolella olevaa run-painiketta. Tämän jälkeen testi suoritetaan ja koodieditorin terminaaliin tulee näkyviin testin suoritus. Kaikki testisuiten testit voi ajaa myös esimerkiksi siirtymällä Visual Studio Coden terminaalissa kansioon, jossa testitiedosto sijaitsee ja kirjoittamalla terminaaliin komennoksi robot ja sen lisäksi testitiedoston nimen tiedostopääte mukaanlukien ja painamalla enter. Robot Frameworkiin voi myös tehdä launch-tiedostoja testien ajamisen hallinnoimiseen. Kuvassa 16 on kuva terminaalista ensimmäisen testin ajamisen jälkeen.



Kuva 16. Kuva Check Webpage For Basic Information testin ajosta terminaalista.

Terminaalissa näkyy testin nimi ja testin suorituksen tulos PASS eli testi on mennyt läpi. Tästä pääsee myös avaamaan Robot Frameworkin tekemän Login klikkaamalla *Ctrl*-näppäin pohjassa Log: jälkeen näkyvää html osoitetta. Myös muita raportteja voi käyttää, mutta omasta mielestäni Log-näkymä on selkein testien yksityiskohtaisen suorituksen tarkasteluun (kuva 17).



Paitakauppa Tests Log

Generated
20231008 17:18:26 UTC+03:00
9 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	1	1	0	0	00:00:02	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
No Tags						

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Paitakauppa Tests	1	1	0	0	00:00:03	

Test Execution Log

SUITE Paitakauppa Tests
Full Name: Paitakauppa Tests
Source: c:\Users\ellij\Documents\ohjelmistotestaus\ont_tests\paitakauppa_tests.robot
Start / End / Elapsed: 20231008 17:18:22.969 / 20231008 17:18:26.093 / 00:00:03.124
Status: 1 test total, 1 passed, 0 failed, 0 skipped

TEST Check Webpage For Basic Information
Full Name: Paitakauppa Tests.Check Webpage For Basic Information
Start / End / Elapsed: 20231008 17:18:23.734 / 20231008 17:18:26.053 / 00:00:02.319
Status: PASS

KEYWORD Browser . New Page ${page}
Documentation: Open a new Page.
Tags: BrowserControl, Setter
Start / End / Elapsed: 20231008 17:18:23.737 / 20231008 17:18:25.967 / 00:00:02.230
17:18:25.965 INFO Successfully initialized new page object and opened url: http://localhost:3000/
17:18:25.966 INFO No browser and context was open. New browser and context was automatically opened when page is created.

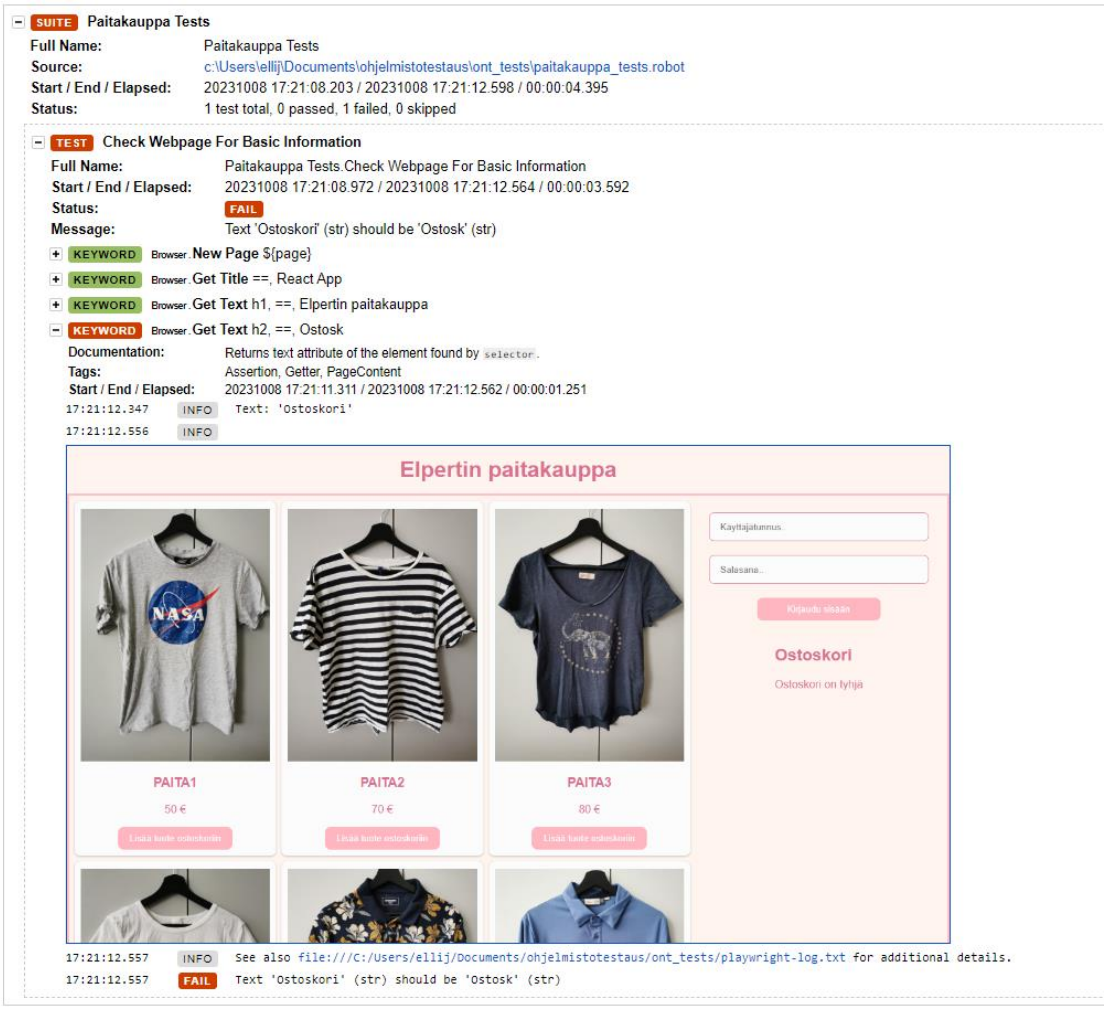
KEYWORD Browser . Get Title ==, React App
Documentation: Returns the title of the current page.
Tags: Assertion, Getter, PageContent
Start / End / Elapsed: 20231008 17:18:25.968 / 20231008 17:18:25.986 / 00:00:00.018
17:18:25.986 INFO Title: 'React App'

KEYWORD Browser . Get Text h1, ==, Elpertin paitakauppa
Documentation: Returns text attribute of the element found by selector.
Tags: Assertion, Getter, PageContent
Start / End / Elapsed: 20231008 17:18:25.987 / 20231008 17:18:26.035 / 00:00:00.048
17:18:26.034 INFO Text: 'Elpertin paitakauppa'

KEYWORD Browser . Get Text h2, ==, Ostoskori
Documentation: Returns text attribute of the element found by selector.
Tags: Assertion, Getter, PageContent
Start / End / Elapsed: 20231008 17:18:26.037 / 20231008 17:18:26.052 / 00:00:00.015
17:18:26.051 INFO Text: 'Ostoskori'

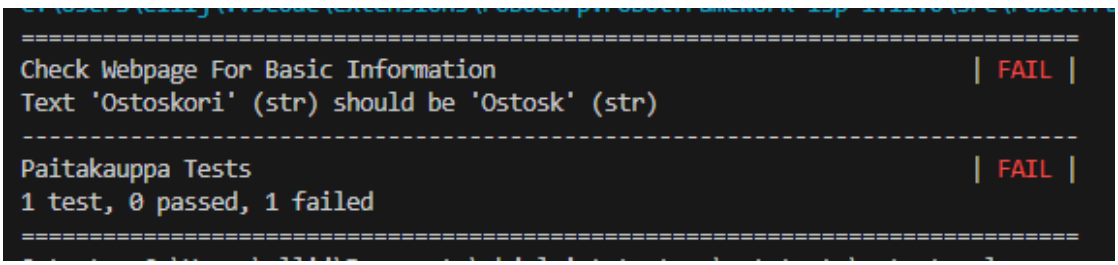
Kuva 17. Kuva Check Webpage For Basic Information testin Logista.

Testin logi aukeaa selaimeen ja siinä näkyy kaikki ajetut testit, sekä avainsanat, joita testissä käytetään. Aluksi nämä ovat kiinni, mutta kuvassa 17 olen avannut kaikki +-painikkeesta ja niissä näkyy tarkempia tietoja. Jos testi epäonnistuu ja avaa tämän login, se avautuu suoraan epäonnistuneeseen kohtaan (kuva 18).



Kuva 18. Kuva Check Webpage For Basic Information testin virhelogista.

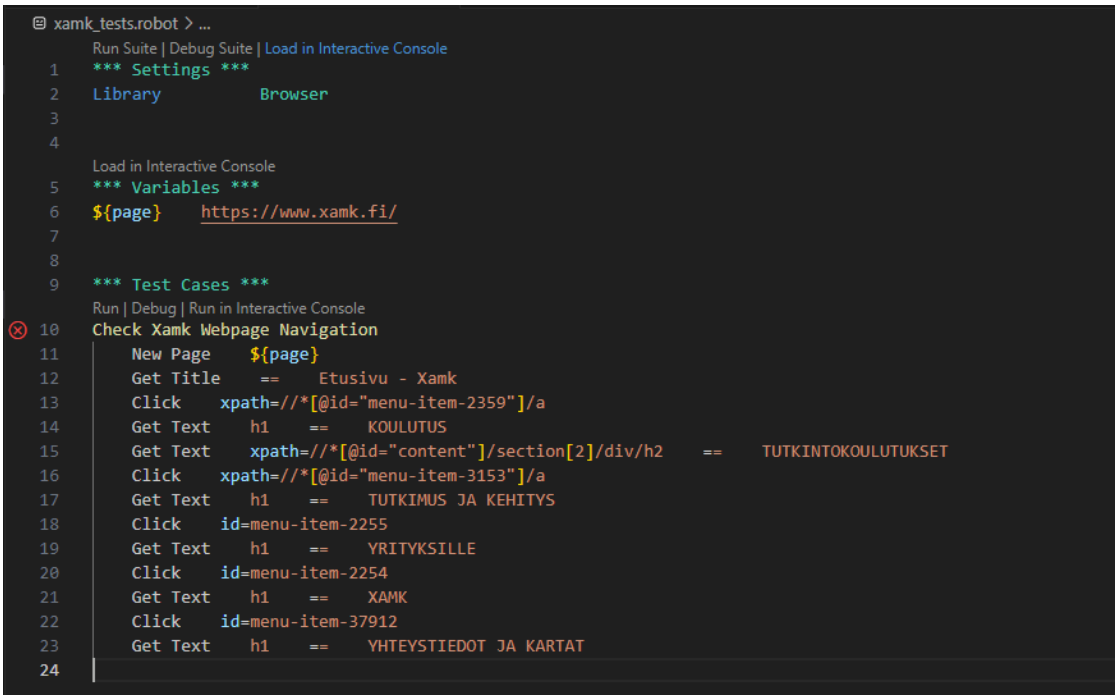
Yleensä epäonnistuneen testin logissa näkyy myös jokin lisätieto siitä, miksi testi ei mennyt läpi. Tässä tapauksessa muutin testiä niin, että h2 otsikon sisällön pitäisi olla ”Ostosk” vaikka se on ”Ostoskori” ja testin logissa näkyy ”Text ’Ostoskori’ (str) should be ’Ostosk’ (str)”. Käyttöliittymätestien epäonnistuessa Robot Framework ottaa myös kuvakaappauksen näkymästä, jossa epäonnistuminen tapahtui ja se näkyy tällä logilla. Virheellinen testin suoritus näkyy koodieditorin terminaalissa kuvan 19 näköisenä.



Kuva 19. Kuva Check Webpage For Basic Information testin virhe terminaalissa.

Terminaalissa on myös nähtävillä tarkempi tieto siitä, mikä testissä meni pieleen. Jos tässä testisuitessa olisi useampi testi, ne näkyisivät allekkain terminaalissa. Osa testeistä voi mennä läpi ja osa epäonnistua. Testin epäonnistuminen kuitenkin keskeyttää suorituksen ja sen jälkeen olevia testejä ei enää ajeta.

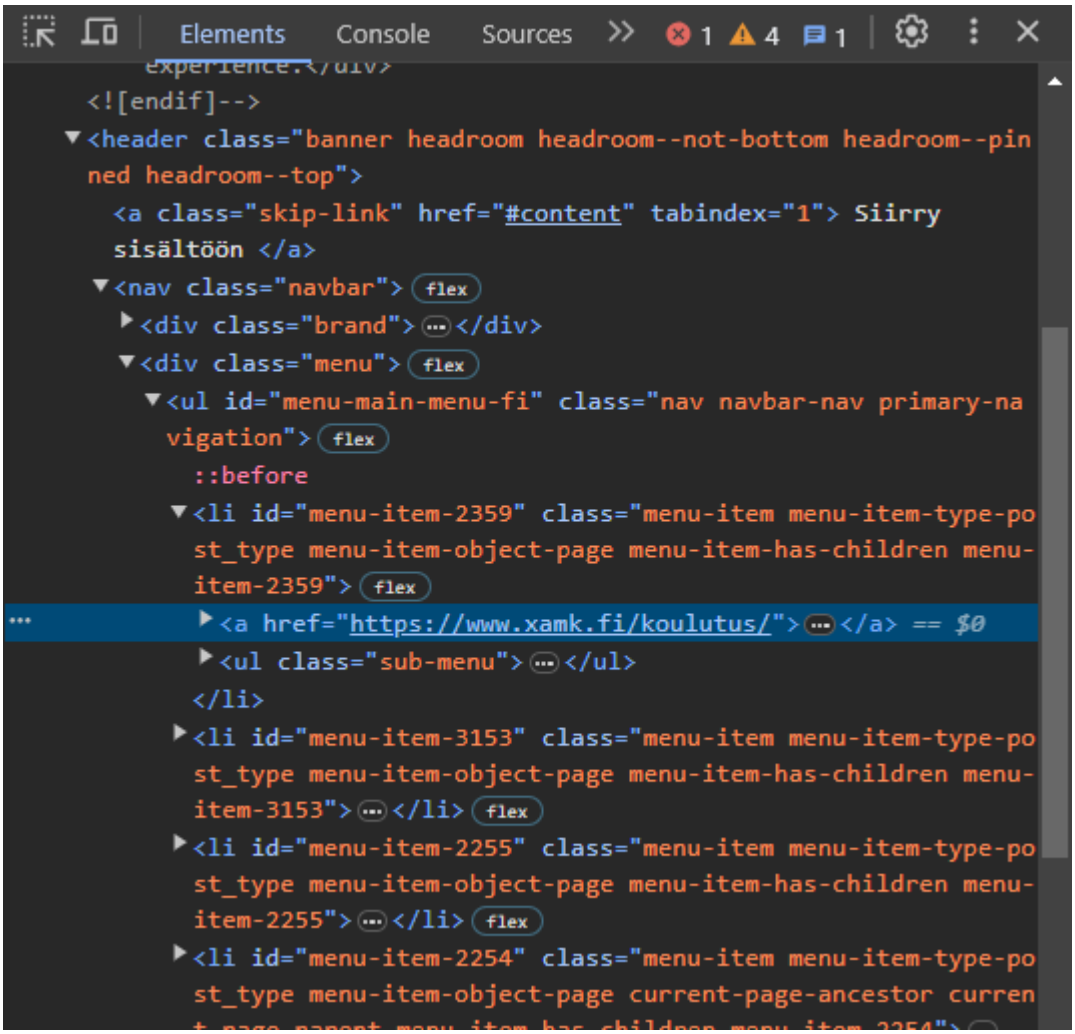
Toisessa testauskohteessa valitsin etsittäviin elementteihin navigaation sisällön. Testi tarkistaa, että Xamkin sivujen navigaatiosta löytyy tietyt elementit ja niitä klikkaamalla pääsee sivuille, joihin navigaation linkit vievät. Tein toisen testitiedoston nimeltä xamk_tests.robot (kuva 20).



Kuva 20. Kuva Check Xamk Webpage Navigation testistä.

Samaan tapaan kuin edellisessä testissä, tässä avataan myös ensin sivu osoitteeseen, jossa testit halutaan suorittaa. Tämän jälkeen on sivun titlen tarkistus. Browser-kirjaston avainsanoilla voi etsiä elementtejä websivustolta monella eri tavalla. Tässä on käytetty elementin xpathia sen löytämiseen muutamassa kohdassa koodiriveillä 13, 15 ja 16. Testi etsii navigaation elementin ja klikkaa sitä, sekä sen jälkeen tarkistaa, että sivulta löytyy tietty otsikko eli navigaatio on ohjannut oikeaan paikkaan. Xpathin lisäksi elementtejä etsitään niiden id:n perusteella koodiriveillä 18, 20 ja 22.

Sivuston elementtejä voi tarkastella Chromella painamalla hiiren oikeaa näppäintä elementin päällä ja valitsemalla sen jälkeen Inspect, joka avaa Chromen developer tools-näkymän (kuva 21). Vaihtoehtoisesti voi avata developer tools-näkymän, joka tapahtuu Chromella näppäinyhdistelmällä *Ctrl + Shift + I* ja etsiä sieltä halutun elementin.



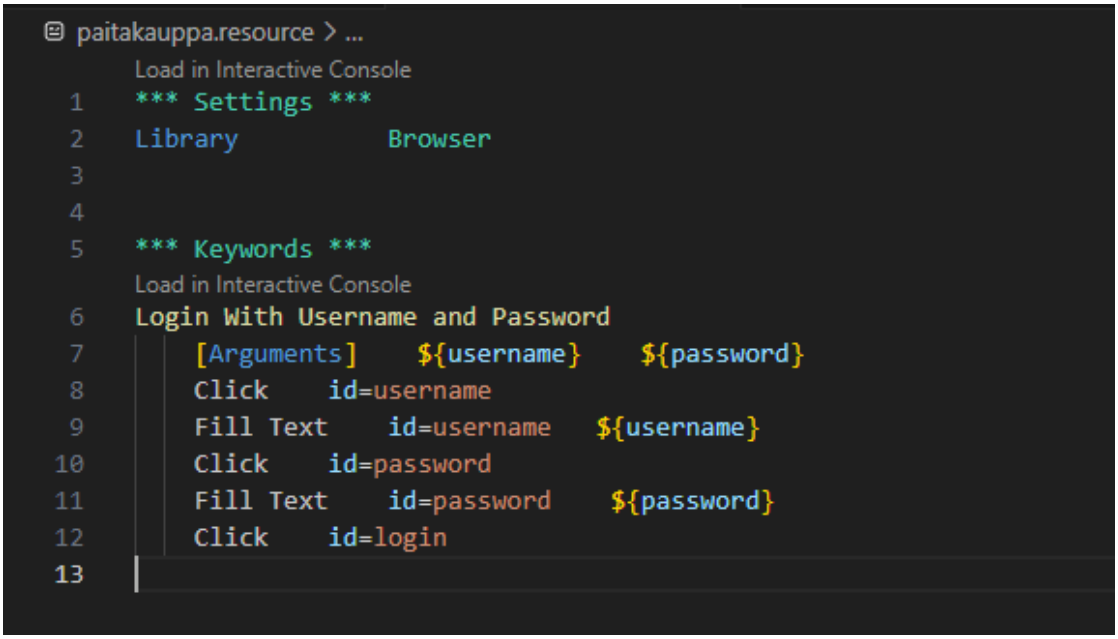
Kuva 21. Kuva Chromen developer tools näkymästä.

Kun Inspect on valittu tietystä elementistä niin developer tools maalaa sen kohdan, jossa tämä elementti on. Tietyn elementin Xpathin saa painamalla elementtiä kuvassa 21 näkyvästä maalatusta kohdasta hiiren oikealla ja valitsemalla Copy ja sen jälkeen copy Xpath. Xpathin käyttö ei ole täysin ongelmatonta, sillä jos sivuston html-rakenteeseen tehdään muutoksia niin elementin Xpath voi muuttua siinä samalla. Testauksen kannalta on hyvä viitata elementin id:seen, mutta aina elementteihin ei ole id:tä laitettu ja tällöin voi joutua käyttämään esimerkiksi Xpathia. Tietyn elementin id:n näkee myös developer

toolsista tarkastelemalla elementtiä. Kuvassa 21 esimerkiksi näkyy listan elementeille omat id:t. Browser-kirjaston eri avainsanojen dokumentaation löytää netistä ja siellä on kerrottu mm. millä eri selektoreilla ”Click” voi etsiä elementtejä.

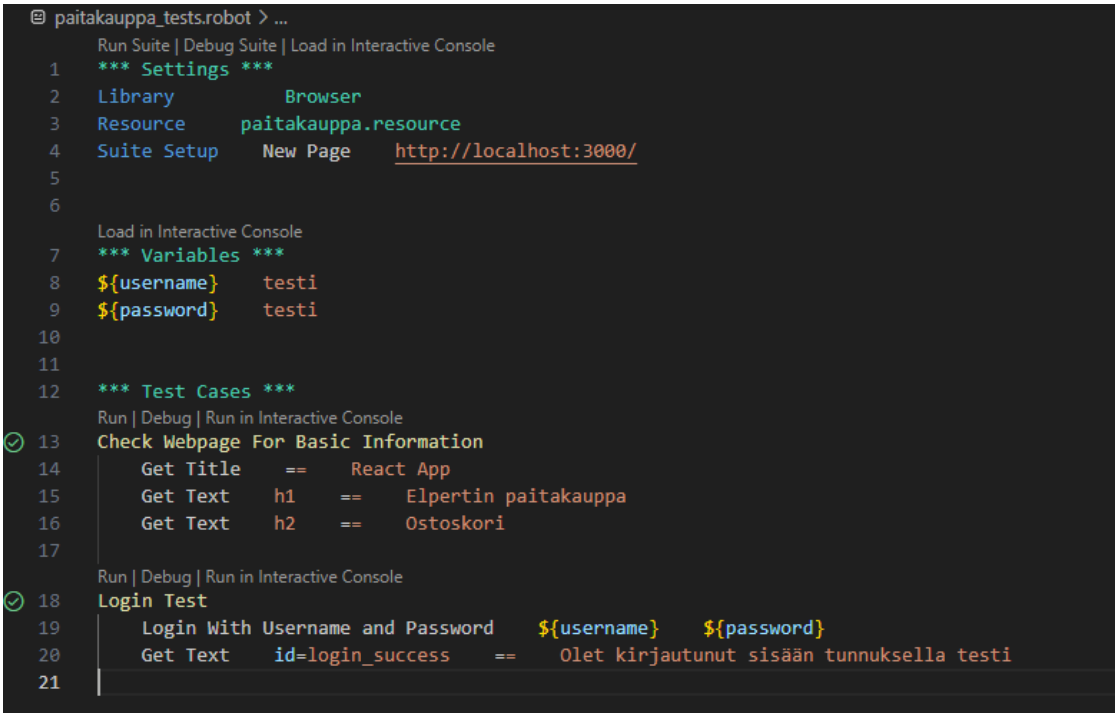
4.3 Sisäänkirjautumisen testaus

Paitakauppasovellukseen voi kirjautua testikäyttäjällä sisään. Tein seuraavaksi testin tätä kirjautumista varten. Robot Frameworkissa voi tehdä itse avainsanoja resurssitiedostoon ja kutsua niitä testissä. Toteutin sisäänkirjautumisen tekemällä sille oman avainsanan erilliseen paitakauppa.resource tiedostoon (kuva 22).



Kuva 22. Kuva avainsanasta Login With Username and Password.

Kirjautumiseen käytettävä avainsana ”Login With Username and Password” sijaitsee erillisessä resurssitiedostossa. Resurssitiedostoon on tuotu settings-osioon käytettävä Browser-kirjasto. ”Keywords” eli avainsanat kohdassa on kirjautumiseen käytettävä avainsana, joka saa argumenteiksi käyttäjätunnuksen ja salasanan, joilla kirjaudutaan sisään. Tämän jälkeen tein uuden testin samaan tiedostoon kuin ensimmäisenkin. Kuvassa 23 näkyy, että uuden testin nimi on ”Login Test”.



Kuva 23. Kuva paitakauppa_tests.robot tiedoston testeistä.

Tässä testisuitessa on nyt kaksi testiä samaan sovellukseen, joten siirsin localhost sivun avaamisen settings-osioon suite setupiksi. Tämä suoritetaan ennen testien ajamista. Settings osiossa otetaan käyttöön tarvittava resurssitiedosto rivillä kolme. Muuttujissa on testaamiseen käytettävä käyttäjätunnus ja salasana. Testitapaus ”Login Test” käyttää resurssitiedostossa olevaa avainsanaa ”Login With Username and Password” ja antaa sille sen vaatimat argumentit eli käyttäjätunnuksen ja salasanan, jotka tulevat variables-osiosta. Kirjautumisen jälkeen sen onnistuminen tarkistetaan etsimällä id:n perusteella elementti, jossa on teksti onnistuneesta sisäänkirjautumisesta. Tämä tarkistus on testitapauksessa eikä avainsanassa, koska näin tehdessä avainsanaa voi hyödyntää esimerkiksi epäonnistuneen kirjautumisen testaamiseen. Samasta syystä avainsana saa argumentteina tunnuksen ja salasanan, eikä niitä ole kovakoodattu avainsanaan. Tein testit onnistuneelle sisäänkirjautumiselle, sekä epäonnistuneelle (kuva 24).

```
Run | Debug | Run in Interactive Console
Login Should Be Successfull
    Login With Username and Password    ${username}    ${password}
    Get Text    id=login_success    ==    Olet kirjautunut sisään tunnuksella testi

Run | Debug | Run in Interactive Console
Login Should Fail
    Login With Username and Password    test    test
    Get Text    id=login_error    ==    Väärä käyttäjätunnus tai salasana

```

Kuva 24. Kuva paitakauppa_tests.robot tiedoston Login testeistä.

Testit on nimetty kuvaavasti ja käyttävät samaa avainsanaa resurssitiedostosta sisäänkirjautumiseen. Argumentteina annetaan ”Login Should Fail” -testissä avainsanalle väärä käyttäjätunnus ja salasana ja tarkistetaan virheviesti etsien elementti id:n perusteella, samaan tapaan kuin onnistunut kirjautuminen tarkistetaan.

5 YHTEENVETO

Tämän opinnäytetyön tekeminen opetti itselle käyttöliittymätestauksen perusteita ja laajensi omaa tietämystä Robot Frameworkin käyttämisestä. Asennusohjeista tuli mielestäni selkeät ja havainnollistavat. Robot Frameworkia varten tulee asentaa monia eri työkaluja, joten koen että selkeistä asennusohjeista on lukijoille hyötyä. Ohjeita seuraamalla Robot Frameworkin käyttöönoton pitäisi sujua ongelmitta. Opin myös paljon yleistä tietoa testauksesta teoriaosuutta tehdessäni.

Tämä työ vastaa asettamiini tutkimuskysymyksiin. Työtä tehdessä esimerkkitestien ja teorian rajaus asetti haasteita. Tuntui, että Robot Frameworkin hyvien testitapausten ja yleiskäyttöisten avainsanojen teoriaa olisi löytynyt todella paljon. Haasteena oli poimia teoriasta juuri tähän työhön olennaiset tiedot. Teoriatietoa löytyi monesta eri lähteestä ja tämä opinnäytetyö on lukijalle hyödyllinen, sillä se kokoaa paljon teoriatietoa yhteen paikkaan. Käyttöliittymätesteistä olisi voinut tehdä enemmän esimerkkejä, mutta pyrin ottamaan tähän työhön olennaisimmat ja selkeimmin havainnollistettavissa olevat asiat. Lisäksi tämä työ tarjoaa lukijalle neuvoja tiedon etsimiseen testien laajentamista ajatellen.

Jatkossa voin hyödyntää oppimaani teoriatietoa työelämässä ja laajentaa entisestään osaamistani testaukseen liittyen. Aikaisemman taustani rajapintatestauksessa huomioiden nyt on helpompi lähteä kokeilemaan itsekin laajemmin käyttöliittymien testausta. Jatkokehitysehdotuksina on testien laajentaminen monipuolisempiin ja monivaiheisempiin käyttöliittymätesteihin sekä myös muiden kuin käyttöliittymätestien tekeminen.

LÄHTEET

Belton, J. 2023. Mastering Test Automation with Robot Framework. Blogi. Päivitetty 20.3.2023. Saatavissa: https://medium.com/@joelbelton/mastering-test-automation-with-robot-framework-9f3d3a6bd9fd [viitattu 30.9.2023]

BrillMinz Tech 2022. What are The Advantages of Automated UI Testing. Blogi. Päivitetty 25.3.2022. Saatavissa: https://medium.com/@brillworkz/what-are-the-advantages-of-automated-ui-testing-aa4f54107d59 [viitattu 30.10.2023]

Browser Library s.a. Introduction. WWW-dokumentti. Saatavissa: https://robot-framework-browser.org/ [viitattu 2.10.2023]

Deshpande P. 2023. Understanding what is UI Test Cases (with Examples). WWW-dokumentti. Päivitetty 21.4.2023. Saatavissa: https://www.browserstack.com/guide/what-is-ui-test-cases [viitattu 2.10.2023]

Helsingin yliopisto s.a. Opiskelijan digitaidot. WWW-dokumentti. Saatavissa: https://blogs.helsinki.fi/opiskelijan-digitaidot/1-tietokoneen-kayton-perusteet/1-1-tietokoneen-toimintaperiaate/kayttojarjestelma-ja-kayttoliittyma/ [viitattu 6.10.2023]

Hruska W. 2019. Why UI testing is important? Blogi. Päivitetty 5.9.2019. Saatavissa: https://hruskawilliam.medium.com/why-ui-testing-is-important-80ae086924e6 [viitattu 6.10.2023]

Kasurinen J. 2017. Ohjelmistotestauksen käsikirja. Saatavissa: https://payhip.com/JPKasurinen [viitattu 2.10.2023]

Klärck, P. 2014. Robot Framework Dos and Don'ts. Diaesitys. Päivitetty 1.9.2014. Saatavissa: https://www.slideshare.net/pekkaklarck/robot-framework-dos-and-donts [viitattu 8.10.2023]

Klärck, P. 2019. HowToWriteGoodTestCases. WWW-dokumentti. Saatavissa: https://github.com/robotframework/HowToWriteGoodTestCases/blob/694b2d666048093e294ef123d56c52a9defb4a1a/HowToWriteGoodTestCases.rst [viitattu 8.10.2023]

Kupila, T. 2022. Robot Framework – toimiva avoimen lähdekoodin ratkaisu. Blogi. Päivitetty 13.6.2022. Saatavissa: https://www.tieturi.fi/blogi/robot-framework-toimiva-avoimen-lahdekoodin-ratkaisu/ [viitattu 30.9.2023]

Python s.a. Download the latest version for Windows. WWW-dokumentti. Saatavissa: https://www.python.org/downloads/ [viitattu 8.10.2023]

Python s.a. Using Python on Windows. WWW-dokumentti. Saatavissa: https://docs.python.org/3/using/windows.html [viitattu 8.10.2023]

Reflector. 2022. Testiautomaatio varmistaa onnistuneen kehittämisen. WWW-dokumentti. Päivitetty 12.9.2022. Saatavissa: https://reflector.fi/testiautomaatio-varmistaa-onnistuneen-kehittamisen/ [viitattu 2.10.2023]

Robot Framework UA s.a. Introduction. WWW-dokumentti. Saatavissa: https://robotframework.org/ [viitattu 30.9.2023]

Software Testing Help. How to Write Test Cases: The Ultimate Guide With Examples. Päivitetty 17.7.2023. Saatavissa: https://www.softwaretesting-help.com/how-to-write-effective-test-cases-test-cases-procedures-and-definitions/ [viitattu 6.10.2023]

VALA s.a. Ohjelmistotestaus ja laadunvarmistus. WWW-dokumentti. Saatavissa: https://www.valagroup.com/fi/palvelut/ohjelmiston-laadunvarmistus/ohjelmistotestaus-ja-laadunvarmistus/ [viitattu 2.10.2023]

VALA. 2022. Mitä on ohjelmistotestaus ja mitä hyötyä siitä on? Ohjelmistotestauksen tasot. WWW-dokumentti. Saatavissa: https://www.valagroup.com/fi/blogi/mita-on-ohjelmistotestaus-ja-mita-hyotya-siita-on/ [viitattu 2.11.2023]

Visure s.a. Mitä ovat testitapaukset? Kuinka kirjoittaa ohjelmistoon liittyviä testitapauksia? WWW-dokumentti. Saatavissa: https://visuresolutions.com/fi/what-are-test-cases-how-to-write-software-related-test-cases/ [viitattu 6.10.2023]