TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Karina Kiris 193484IADB

Ettevõttesiseste tegevuste automatiseerimislahenduse arendus

Bakalaureusetöö

Juhendaja: Maili Markvardt
Magistrikraad

Tallinn 2024

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.

Autor: Karina Kiris

04.01.2024

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua automatiseerimislahenduse Rootsi *Bolt Food*'i osakonna hetkel käsitsi tehtavate ettevõttesiseste tegevuste jaoks. Lahendus on suunatud ettevõtte sisekasutusele, mille eesmärk on protsessidele kuluva aja säästmine ja inimeste töökoormuse vähendamine.

Viidi läbi põhjalik analüüs valitud osakonna protsesside kohta, sealhulgas uue kulleri liitumisprotsess, kulleri profiili muutmine, kulleri profiili oleku muutmine, restorani kirjelduse lisamine, restorani eeldatava valmistusaja muutmine ja restoranidele tunnuste lisamine. Analüüsi alusel määrati loodava rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded. Nõuete põhjal koostati tehniline analüüs automatiseerimislahenduse loomiseks, mis hõlmab endas tehnoloogiate valikut.

Lõpptulemusena õnnestus luua Rootsi *Bolt Food*'i jaoks automatiseerimise tarkvaralahendus, mis võimaldab mugavalt hallata kullerite andmeid, lisada uusi kullereid, muuta restoranide oodatud valmistamise aega ning kirjeldusi. Lisaks integreeriti ka spetsiaalsed funktsioonid vigade tuvastamiseks ning nende kuvamiseks kasutajale.

Üldiselt rõhutab käesolev lõputöö, kuidas tehnoloogia võimaldab Rootsi *Bolt Food*'i meeskonnal optimeerida oma sisetoiminguid ning tõsta töö efektiivsust. Tulevikus võib tarkvaralahendust edasi arendada ning laiendada, et see kataks veelgi enam valdkondi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 35 leheküljel, 6 peatükki, 16 joonist, 13 tabelit.

Abstract

Development of Automations for Internal Operations

Automation has become an essential component of modern business operations, offering a range of benefits from increased efficiency, accuracy to cost reduction. The main goal of this thesis is to develop automation for internal operations in Swedish Bolt Food in order to reduce time spent on each task. The problem addressed in this thesis is that the operational team is facing repetitive manual tasks on a daily basis that are taking up a lot of time and effort.

In this thesis, explore the various internal operations that can be automated, identify the benefits of automating these operations, and develop automation application for the identified operations. The research methodology involved a comprehensive literature review, case study analysis, and the development and testing of automation scripts.

As the results of this research, the author developed automation scripts which proved to be effective in reducing manual tasks, increasing speed and accuracy, and improving overall operational efficiency. Using the software created as part of this thesis, the following processes were automated - adding new couriers, changing courier data, changing the expected cooking time of the restaurants and adding restaurant descriptions.

Overall, this thesis highlights the importance of automation in modern business operations and provides a practical example of its application in a real-world organizational context.

The thesis is in Estonian and contains 35 pages of text, 6 chapters, 16 figures, 13 tables.

Lühendite ja mõistete sõnastik

API	Application Programming Interface, rakendusliides
CSV	Comma-Separated Values file format
HTML	Hypertext Markup Language, veebilehtede loomiseks mõeldud märgistuskeel
HTTP	Hypertext Transfer Protocol, andmevahetusprotokoll dokumentide ülekandmiseks Internetis
JSON	JavaScript Object Notation, standard andmevahetuseks JavaScripti objektide kujul
NaN	Not a Number
OOP	Objektorienteeritud programmeerimine
Selenium RC	Selenium Remote Control, server käskude vastuvõtmiseks

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tänapäeval on oluline rõhutada automatiseerimise tähtsust äritegevuses. Automatiseerimine mitte ainult ei vähenda tegevuskulusid, vaid suurendab ka tõhusust, täpsust ja lõpuks parandab organisatsiooni üldist tulemuslikkust.

Käesoleva lõputöö eesmärk on arendada automatiseerimislahenduse Rootsi *Bolt Food*'i osakonna sisemiste toimingute optimeerimiseks. Selle töö raames loodud tarkvara abil automatiseeritakse sisemised protsessid, vähendades nende täitmisele tavapäraselt kuluvat tööaega.

Teoreetilises osas uuritakse Rootsi *Bolt Food*'i erinevaid sisemisi käsitsi tehtavaid toiminguid – uue kulleri liitumisprotsessi, kulleri profiili muutmist, kulleri profiili oleku muutmist, restorani kirjelduse lisamist, restorani eeldatava valmistusaja muutmist ja restoranidele tunnuste lisamist, mida saab rakenduse abil automatiseerida. Samuti selgitatakse välja nende toimingute automatiseerimise eelised ja kuidas see võib osakonna tulemuslikkust parandada. Analüüsi osas määratakse rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded automatiseerimislahenduse arendamiseks. Analüüsitakse lahenduse tehnoloogia valiku võimalusi. Praktilises osas töötatakse tuvastatud toimingute jaoks välja automatiseerimislahenduse.

Antud bakalaureusetöö koosneb kuuest osast. Esimeses peatükis tutvustatakse lõputöö teemat, tuuakse välja eesmärk ja probleem. Teises peatükis tehakse püstitatud probleemi analüüsi. Kolmandas peatükis vaadatakse läbi asjakohane kirjandus automatiseerimise, valitud tehnoloogiate ja rakenduste arhitektuuri kohta. Neljandas peatükis tutvustatakse väljatöötatud automatiseerimise rakendust, koodi testimise käiku, rakendust toetavat dokumentatsiooni. Viiendas peatükis käsitletakse tulemusi ja tehakse nende põhjal järeldused samuti soovitusi tulevaste uuringute jaoks. Kuuendas peatükis on käesoleva töö kokkuvõte.

1.1 Taust

Käesoleva bakalaureusetöö ülesanne on ettevõttesiseste tegevuste automatiseerimislahenduse väljatöötamine *Bolt Operations* OÜ jaoks. Teema valik tuleneb aktuaalsest probleemist, mida autor Rootsi *Bolt Food*'i tiimis töötades täheldanud on.

Igapäevaselt puutub autor kokku osakonna sisetoimingutega seotud käsitsi tehtavate ülesannetega. Need manuaalselt teostatavad ülesanded hõlmavad andmete sisestamist, aruannete loomist, klienditoe pakkumist ja muid haldustoiminguid. See võtab palju aega ja sisaldab endas mitmeid korduvaid tegevusi. Need toimingud mitte ainult ei nõua märkimisväärset ajakulu, vaid sisaldavad ka mitmeid korduvaid ülesandeid, luues seeläbi potentsiaali inimlike vigade tekkeks. Selle taustal tekib vajadus automatiseerimise järele, kuna see kiirendab protsesse ja vähendab vigade tekke riski, tagades seeläbi tõhusama ja täpsema töö. Seetõttu autor soovis mõned toimingud automatiseerida, et eemaldada olemasolevat probleemi.

1.2 Probleem

Rootsi *Bolt Food*'i osakonna operatiiv meeskonnal on suur väljakutse, kuna nad tegelevad suuremahulise käsitsi tööga. Praegused manuaalsed tegevused nõuavad märkimisväärset aega ja täpsust, mis soodustab vigade tekkimist ja teenuste osutamise viivitust. Korduvad käsitsi tehtavad toimingud mitte ainult ei vähenda meeskonna produktiivsust, vaid tekitavad ka töötajates stressi, kuna protsess on monotoonne ja aeganõudev. Suure andmemahu käsitsi haldamine mõjutab oluliselt meeskonna produktiivsust, osakonna efektiivsust, misomakord suurendab vigade tekke riski.

Ülalmainitud põhjal avaldub vajadus automatiseerimise järele. Sobivate tööriistade rakendamine pakub võimalust muuta sisemised toimingud sujuvamaks, vähendades töötajate koormust, suurendades operatiivset tõhusust ning minimeerides erinevaid riske. See parandab üldist tõhusust ja loob ka võimaluse keskenduda keerukamatele ülesannetele, mis nõuavad inimintellekti ja loovust. Sel moel muutub osakonna töö ka innovaatilisemaks ja konkurentsivõimelisemaks, kuna töötajate ressursid võivad olla kasutatud muudel eesmärkidel rakenduse töötamise ajal.

1.3 Lähtetingimused

Lõputöö probleemi lahendamisel tuleb lähtuda ettevõttes hetkel olemasolevatest protsessidest. Andmete kogumise ja töötlemise protsess peab olema kooskõlas juba olemasolevate süsteemide ja platvormidega, tagades sujuva integreeritavuse ning minimeerides võimalikke vastuolusid või topelttööd.

1. Andmed peavad olema võetud *Google Sheet'*ist.
2. Rakendus peab teostama toiminguid siseveebis *admin-panel*.
3. Rakendus peab olema turvaline (võõradele isikutele juurdepääs piiratud), kuna käideldavad andmed võivad sisaldada tundlikku teavet.
4. Rakendusel peab olema dokumentatsioon, et tagada rakenduse kasutamiseks ja arendamiseks läbipaistev ning jätkusuutlik põhi, rakenduse arendamiseks ka tulevikus.

1.4 Eesmärk ja ülesande püstitus

Käesoleva bakalaureusetöö peamine eesmärk on arendada Rootsi *Bolt Food*'i osakonnale automatiseerimislahenduse, mis suudaks tõhusalt ja täpselt täita hetkel käsitsi tehtavaid ülesandeid, vähendades sellega oluliselt töötajate koormust. See lahendus on suunatud meeskonna sisekasutuseks, eelkõige autori ja tema 12-liikmelise meeskonna jaoks.

Esmane eesmärk on teostada põhjalik analüüs, et tuvastada sisetegevused, mida on võimalik rakenduse abil automatiseerida. Samuti on oluline kaaluda, millised protsessid vajavad automatiseerimist kõige rohkem ning välja töötada nende toimingute jaoks sobiv automatiseerimise rakendus.

Antud lõputöö tulemused võivad välja selgitada potentsiaalsed automatiseerimise valdkonnad ning demonstreerida, kuidas automatiseerimine võib vähendada töötajate töökoormust, suurendada efektiivsust ja täpsust ning parandada organisatsiooni üldist tulemuslikkust. Samuti anda praktilise näite selle kohta, kuidas automatiseerimise tööriistu saab kasutada reaalse organisatsiooni kontekstis ja see võiks olla juhisena

teistele ettevõtetele, mis soovivad oma sisemistes toimingutes automatiseerimise tööriistu kasutusele võtta.

Eesmärkidest lähtuvalt on püstitatud järgmised ülesanded:

1. Kirjeldada osakonnas olemasolevaid siseprotsesse.
2. Uurida, millised käsitsi tehtavad toimingud on võimalik automatiseerida ning millised nendest vajavad automatiseerimist kõige rohkem.
3. Valida sobiv tehnoloogia, tööriistad ja mustrid rakenduse kirjutamiseks arvestades süsteemi eripära.
4. Arendada ja testida rakendust siseprotsesse automatiseerimiseks.

1.5 Metoodika

Lahenduste väljatöötamise käigus alustatakse eelnevalt sarnaste probleemide lahenduste uurimisega. See hõlmab põhjalikku kirjanduse ülevaadet, kus analüüsitakse varasemat teadustööd automatiseerimise ja sellega seotud rakenduste valdkonnas. Kirjanduse ülevaade loob lõputööle teoreetilise raamistiku ja võimaldab tuvastada potentsiaalseid valdkondi siseprotsesside automatiseerimiseks.

Järgmise sammuna viiakse läbi juhtumianalüüs Rootsi *Bolt Food*'i sisemiste toimingute kohta, keskendudes konkreetsetele valdkondadele, mis võiksid automatiseerimisest kõige rohkem kasu saada. See analüüs hõlmab praeguste protsesside üksikasjalikku jälgimist ja andmete kogumist seotud ülesannete kohta. Analüüs aitab mõista osakonna vajadusi ja prioriteete rakenduse arendamisel juhtides tähelepanu sellele, millised protsessid vajavad kõige kiiremat ja tõhusamat automatiseerimist.

Lahenduse analüüsi osas määratakse funktsionaalsed ja mittefunktsionaalsed nõuded. Nõuete määramise järel analüüsitakse erinevaid tehnoloogiaid ja automatiseerimis võimalusi. Seejärel põhjendatakse valikuid arvestades samuti ka autori eelnevat kogemust erinevate tehnoloogiatega, platvormi nõudeid ja võimalikke edasiarendusi.

Seejärel viiakse läbi arendusetapp, kus kirjeldatud analüüsi tulemusena välja töötatud lahendused realiseeritakse. Rakendus luuakse sobiva programmeerimiskeele abil, võttes arvesse tuvastatud ülesannete keerukust ja spetsiifikat. Lahenduse valmimisel järgneb

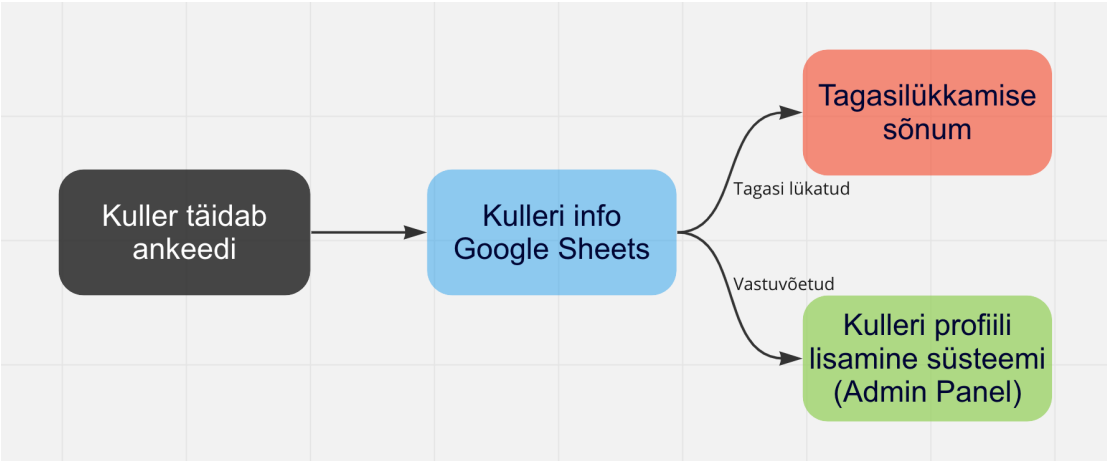
hoolikas rakenduse testimine, kus mõõdetakse rakenduse tõhusust ja vastavust seatud nõuetele. Saadud tulemusi analüüsitakse, hinnatakse lahenduse sobivust ning sõnastatakse järeldused, mis annavad ülevaate Rootsi *Bolt Food*'i sisemiste protsesside võimalikest automatiseerimise eelistest.

2 Probleemi analüüs

Antud bakalaureusetöö probleemi põhjalikumaks tutvustamiseks tuleb omada ettekujutust Rootsi *Bolt Food*'i operatiiv meeskonna mõnedest käsitsi tehtavatest sisetegevustest suure andmete hulgaga – uue kulleri liitumisprotsess, kulleri profiili muutmine, kulleri profiili oleku muutmine, restorani kirjelduse lisamine, restorani eeldatava valmistusaja muutmine ja restoranide tunnuste lisamine, millest räägivad järgnevad alampeatükid.

2.1 Kulleri liitumisprotsess

Kulleri liitumisprotsess (Joonis 1.) hõlmab tavaliselt mitut etappi selleks, et tagada kulleri töö kandidaadi sobivus ning kandidaadil olemasolev vajalik teave ja tööriistad tööülesannete täitmiseks. See on ärikriitiline protsess, kuna see määrab, kas kullerite arv vastab tellimuse nõudlusele.



Joonis 1. Kulleri lisamise mudel

Järgnevalt on toodud tüüpiline uue kulleri liitumisprotsess:

1. Esimene samm on taotlusvormi täitmine, isiklikku teave lisamine.
2. Pärast taotluse vormi täitmist, saabub kulleri info *Google Sheets*'i ning meeskond kontrollib seda.
3. Kui esitatud taotlus on vale, saab kuller tagasilükkamise kirja.

Kui esitatud taotlus on korrektne, minnakse edasi järgmiste etappidega.

4. Kui kuller on läbinud eelmise etapi, antakse talle vajalik varustus – kohaletoimetamiskott.
5. Pärast seda, kui kuller on liitumisprotsessi lõpetanud, luuakse tema profiil süsteemisisesel lehel *admin-panel*. See profiil sisaldab kulleri isikuandmeid ja muud asjakohast teavet.

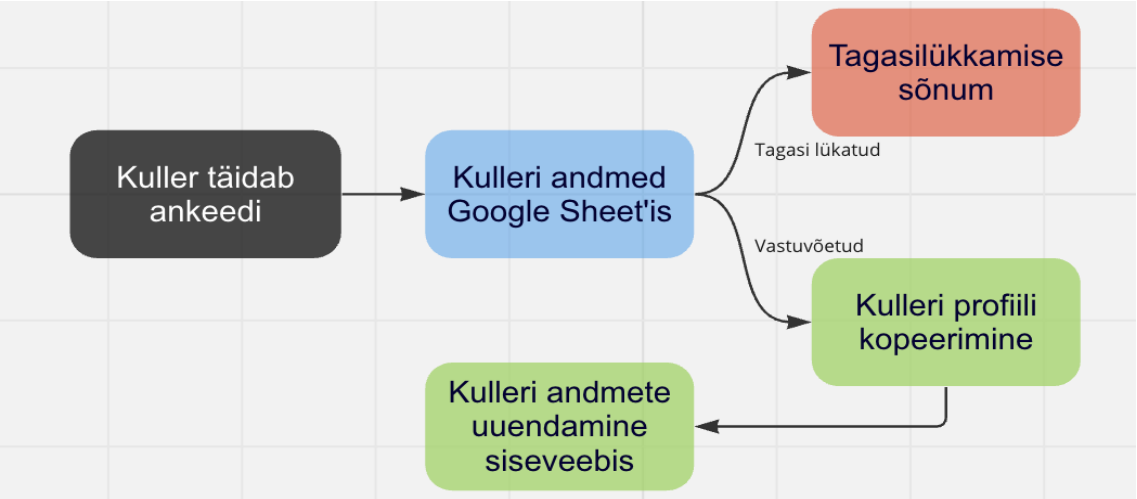
Alltoodud tabel (Tabel 1.) sisaldab andmeid kolme nädala kohta – lisandunud kullerite arvu ja nende aktiveerimiseks kuluva aja kohta. Iga uue profiili loomine võtab umbes üheksa minutit aega.

Tabel 1. Kullerite aktiveerimine

Nädal	Kullerite arv	Aeg(tundides)
Nädal 1	46	6.9t
Nädal 2	39	5.85t
Nädal 3	52	7.8t

2.2 Kulleri profiili muutmine

Kulleritel on võimalus taotleda ettevõtte vahetamist – pöörduda koostööpartneri poole või vahetada oma isikliku ettevõtte vastu. Selle muudatuse rakendamiseks tuleb nende profiil kopeerida ja andmeid uuendada (Joonis 2.).



Joonis 2. Kulleri profiili kopeerimine ja andmete uuendamine

Järgnevalt on toodud kulleri profiili uuendamise protsess:

1. Kuller täidab taotlusvormi.
2. Pärast taotluse vormi täitmist, saabub kulleri info *Google Sheets*'i ning meeskond kontrollib selle üle.
3. Kui esitatud taotlus on vale, saab kuller tagasilükkamise kirja.
   Kui esitatud taotlus on korrektne, minnakse edasi järgmiste etappide juurde.
4. Kulleri ID-d tuleb ükshaaval kopeerida ja kleepida need *admin-panel*'isse.
5. Järgmisena peab avama iga kulleri profiili ja vajutama "*Copy*" nupule selleks, et kustutada vananenud andmed.
6. Järgnevalt tuleb täita kõik vajalikud lahtrid kulleri poolt saadetud andmetega.
7. Viimase etapina oleks vaja iga kord vajutada "*Save*" nupule.

Alltoodud tabel (Tabel 2.) sisaldab andmeid kolme nädala kohta - muudetud profiilide arvu ja nende muudatustele kuluva aja kohta.

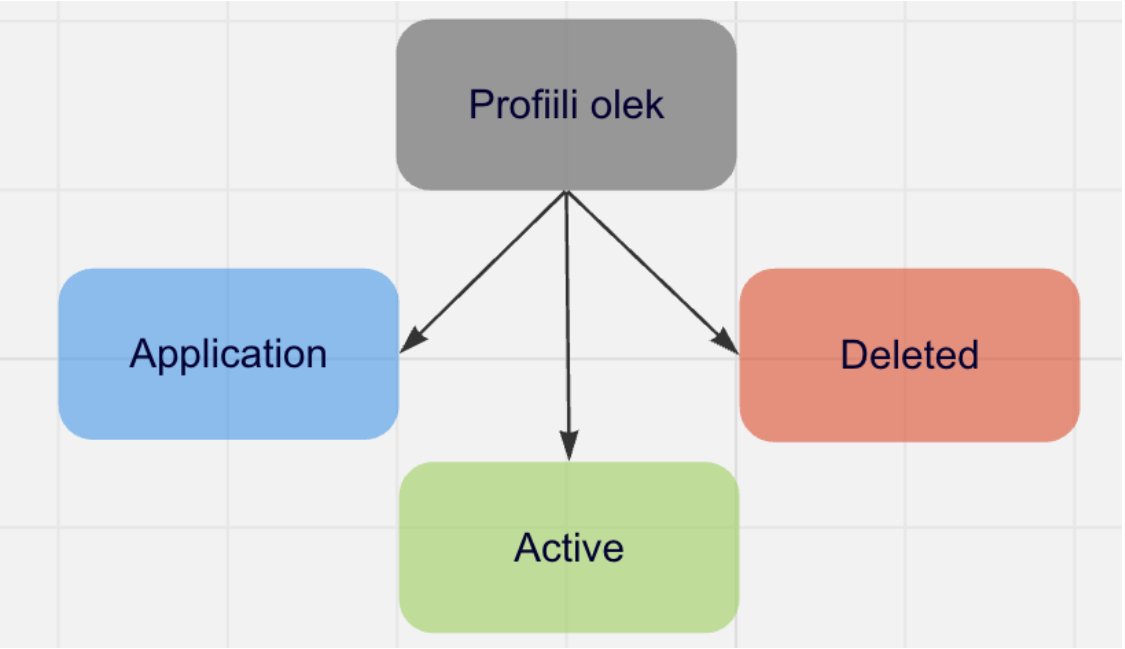
Tabel 2. Kullerite profiili muutmine

Nädal	**Muudatuste arv**	**Aeg(tundides)**
Nädal 1	17	2.7t
Nädal 2	22	3.6t
Nädal 3	14	2.3t

2.3 Kulleri profiili oleku muutmine

Iga kulleri profiil saab olla kolmes erinevas olekus – *application, active, deleted*. Meeskond töötab peamiselt kahega nendest. Esimene olek on "aktiivne", mis tähendab, et kulleril on platvormil olemas aktiivne konto. Teine olek on "kustutatud", mis näitab, et kulleri konto on platvormil peatatud. Aeg ajalt peab meeskond läbi vaatama kulleri profiilid, kes pole *Bolt* platvormi pikalt kasutanud. Antud juhtudel tuleb need kullerid platvormilt ära kustutada ehk teiste sõnadega tuleb muuta nende profiili olekut.

Joonisel 3. on illustreeritud skeem kulleri profiili olekutest.



Joonis 3. Kulleri profiili olekud

Protsessi etapid on järgmised:

1. *Looker* andmebaasi aruande väljavõtmine koos kõikide andmetega, sealt eraldatakse kulleri ID-d.
2. Kulleri ID-d tuleb ükshaaval kopeerida ja kleepida need *admin-panel*'isse.
3. Iga kopeeritud ID jaoks peab avama kulleri profiili ja muutma selle oleku "*Deleted*" peale.
4. Viimase etapina oleks vaja iga kord vajutada "*Save*" nupule.

Alltoodud tabel (Tabel 3.) sisaldab andmeid kolme nädala kohta – muudetud profiilide oleku arvu ja nende muudatustele kuluva aja kohta.

Tabel 3. Kullerite profiili oleku muutmine

Nädal	**Muudatuste arv**	**Aeg(tundides)**
Nädal 1	78	4t
Nädal 2	31	1.55t
Nädal 3	0	0

2.4 Restorani tunnuste lisamine

Restorani tunnuste - '*trait*' jaoks on palju erinevaid kasutusjuhtumeid ning samuti on erinevate tunnuste arv suur. Nende abil süsteem mõistab, kas teatud sätteid tuleks konkreetses restoranis rakendada või mitte. Näiteks, valitud restoran osaleb kampaanias ja õige tunnuse abil rakendatakse allahindlust kogu menüüle või tasuta kohaletoimetamisele. Mõnel juhul võib restoranide arv, millele tuleb omadus lisada või eemaldada, ulatuda kuni tuhandeni. Sellisel juhul hõlmab antud protsess palju aega, teades, et ühe restorani jaoks tunnuse lisamine võtab umbes 3. minutit aega.

Üldiselt näevad läbitehtavad etapid järgmisena:

1. Restorani ID-d tuleb ükshaaval kopeerida ning kleepida need *admin-panel*'isse.
2. Iga kopeeritud ID jaoks peab avama restorani lehe, leidma korrektse tunnuse ja muutma selle tunnuse vastavalt olukorrale.
3. Viimase etapina tuleks iga kord vajutada "*Save*" nupule.

Alltoodud tabel (Tabel 4.) sisaldab andmeid kolme nädala kohta – restoranide arvu (kus lisati tunnus) ja nende lisamiseks kuluva aja kohta.

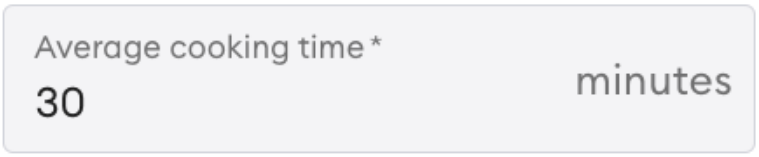
Tabel 4. Restorani tunnuse lisamine

Nädal	**Restoranide arv**	**Aeg(tundides)**
Nädal 1	0	0
Nädal 2	786	39t
Nädal 3	0	0

2.5 Restorani toidu valmistusaeg

Iga platvormi restorani jaoks on määratud ennetatav toiduvalmistamise aeg, mida näidatakse klientidele rakenduses. Võib esineda erinevaid stsenaariume, kus restorani ennetavat valmistusaega (Joonis 4.) tuleb kliendipoolses rakenduses muuta.

Kõige sagedamini juhtub nii, et restoran jääb ettenähtud ajast toidu valmistamisega hiljaks. See mõjutab kliente otseselt, kuna põhjustab tellimuste hilinemist.



Joonis 4. Restorani toidu valmistusaeg

Protsessi etapid on järgmised:

1. Restorani ID-d tuleb ükshaaval kopeerida ning kleepida need *admin-panel*'isse.
2. Iga kopeeritud ID jaoks peab avama restorani lehe ja muutma selle ennetavat valmistusaega kas kauemaks või vähemaks vastavalt olukorrale.
3. Viimase etapina tuleks iga kord vajutada "*Save*" nupule.

Alltoodud tabel (Tabel 5.) sisaldab andmeid kolme nädala kohta – restoranide arvu, kus muudeti toiduvalmistamise aeg ja nende muudatustele kuluva aja kohta.

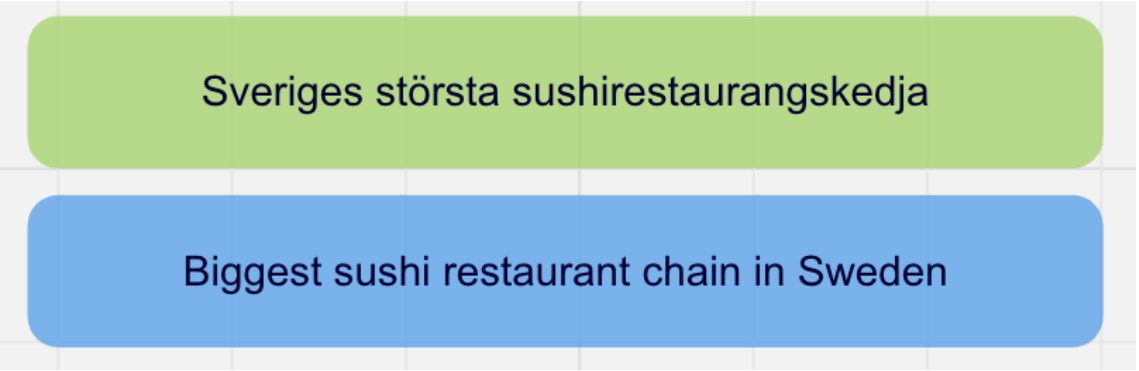
Tabel 5. Restorani toidu valmistusaja muutmine

Nädal	**Restoranide arv**	**Aeg(tundides)**
Nädal 1	37	1.9t
Nädal 2	29	1.6t
Nädal 3	33	1.7t

2.6 Restorani kirjeldus

Restorani kirjelduse olemasolu kohalevedu platvormil on uskumatult oluline. Eelkõige väiksematele ettevõtetele, et oma kontseptsiooni välja tuua ja konkurentidest eristada. Teisest küljest tahavad suured restoraniketid, et nende kaubamärk oleks igal pool samasuguselt esindatud.

Rootsi turul on kasutuses restoranide kirjeldused kahes keeles - rootsi ja inglise keeles (Joonis 5.).



Joonis 5. Restorani kirjeldus

Protsessi etapid on järgmised:

1. Restorani ID-d tuleb ükshaaval kopeerida ning kleepida need *admin-panel*'isse.
2. Iga kopeeritud ID jaoks peab avama restorani lehe ning lisama kirjelduse kas ühes või kahes keeles.
3. Viimase etapina tuleks iga kord vajutada "*Save*" nupule

Alltoodud tabel (Tabel 6.) sisaldab andmeid kolme nädala kohta – restoranide arvu, kus muudeti/lisati kirjeldus ja nende muudatustele kuluva aja kohta. Olukorras, kui restoranil oli kirjeldus juba olemas, tuleb see kõigepealt kustutada ja alles siis lisada uuendatud kirjeldust. Antud juhul võtab kirjelduste lisamine rohkem aega kui tavaliselt.

Tabel 6. Restorani kirjeldus

Nädal	**Restoranide arv**	**Aeg(tundides)**
Nädal 1	26	2.2t
Nädal 2	15	1.3t
Nädal 3	41	3.4t

2.7 Analüüsi kokkuvõtte

Kokkuvõtteks, selles peatükis analüüsiti erinevaid käsitsi tehtavaid sisemisi protsesse, mis on seotud Rootsi *Bolt Food*'i platvormiga. Need ei ole kõik toimingud, millega meeskond tegeleb, aga autor otsustas, et valitud protsessid on korduvad ja kõige ajakulukamad.

Tegevused hõlmasid uue kulleri liitumisprotsessi, kulleri profiili muutmist, kulleri profiili oleku muutmist, restorani kirjelduse lisamist, restorani eeldatava valmistusaja muutmist ja restoranidele tunnuste lisamist. Analüüsist selgus, et need kuus sisemist protsessi on üldiselt sarnase ülesehitusega, korduvad ja mitte väga keerulised. Näiteks uue kulleri lisamine või kulleri andmete muutmine hõlmab vormi täitmist põhiandmetega, nagu nimi, kontaktteave ja muud tüüpi informatsioon. Samamoodi hõlmab restorani tunnuste lisamine või restorani eeldatava valmistusaja muutmine välja värskendamist uue väärtusega.

Kuigi need protsessid võivad tunduda lihtsad ja arusaadavad, tehtavate tehingute suur hulk võib muuta need käsitsi tehes aeganõudvaks. Lisaks võivad need käsitsi tehtavad protsessid siduda väärtuslikke ressursse, mida saaks paremini kasutada strateegiliste ülesannete jaoks.

Kolme nädala jooksul analüüsiti kuut käsitsi tehtavaid tegevust sisemises süsteemis (Tabel 1., Tabel 2., Tabel 3., Tabel 4., Tabel 5., Tabel 6.). Selle analüüsi eesmärk oli saada ülevaade, kui sageli neid toiminguid tehakse ning määrata kindlaks, kui palju aega need igal nädalal võtavad:

1. Uute kullerite lisamine: see protsess toimub tihti (umbes 40 korda nädalas) ja on suhteliselt lihtne, kuid see on ka platvormi tõhusa toimimise seisukohalt kriitilise tähtsusega. Selle protsessi automatiseerimine võib oluliselt parandada töö efektiivsust ja säästa aega. Hetkel kulub iga uue kulleri profiili loomiseks umbes 9 minutit aega, seega ka keskmiselt 7 tundi nädalas. (Tabel 1.)
2. Kulleri andmete muutmine: see protsess toimub samuti tihti (umbes 17 korda nädalas) ja võib olla keeruline, eriti kui värskendamiseks on mitu välja. Teisest

küljest on see protsess väga sarnane tavalise kulleri profiili loomisega. Selle protsessi automatiseerimine võib säästa aega. Hetkel kulub iga kulleri profiili muutmiseks umbes 9 minutit aega, seega keskmiselt 3 tundi nädalas. (Tabel 2.)

3. Kulleri profiili oleku muutmine: see protsess toimub mitte nii sageli kui eelmised. Analüüsitud kolme nädala jooksul tegevuste arv oli järgmine – 78, 31, 0. Need andmed näitavad, et profiili oleku muudatuste hulk ei ole ühtlane. Hetkel kulub iga kulleri profiili oleku muutmiseks umbes 3 minutit aega, seega keskmiselt 2 tundi nädalas. (Tabel 3.)
4. Restorani tunnuse lisamine: kolmest nädalast vaid ühel nädalal pidid meeskonnaliikmed restorani tunnuseid muutma, samal ajal oli restoranide arv väga suur – 768. Töökoormus jagunes kolme samaaegselt muudatusi tegeva inimese vahel. Vastasel juhul oleks selle ülesande täitmiseks kulunud terve töönädal ühe inimese poolt, sest arvutused näitavad, et selleks kuluks umbes 39 tundi. (Tabel 4.)
5. Restorani eeldatava valmistusaja muutmine: tellimuse õigeaegne kohaletoimetamine on kliendi rahulolu jaoks oluline. Analüüsi raames selgus, et toiduvalmistamisaja muutmine toimub umbes 33 korda nädalas ning hetkel sellele kulub keskmiselt 2 tundi nädalas. Selle protsessi automatiseerimine võib vähendada hilinenud tellimuste hulka. (Tabel 5.)
6. Restoranide kirjelduste lisamine: restoranide kirjeldused on olulised klientide usalduse tekitamiseks ja üldise kasutuskogemuse parandamiseks. Kogutud andmed näitavad, et restoranide kirjeldusi lisatakse/muudetakse umbes 29 korda nädalas ja hetkel sellele kuulub keskmiselt 2,5 tundi nädalas. (Tabel 6.)

Analüüsi tulemuste põhjal otsustas autor, millised tegevused tuleb automatiseerida, et vähendada käsitsitööd ja seeläbi suurendada töö efektiivsust. Kahest eelnevalt loetletud sisemisest tegevusest ei olnud mõnel nädalal üldse tegevust tehtud, seetõttu otsustas autor mitte luua automatiseerimist nende kahe ülesande jaoks - kulleri profiili oleku muutmine ja restorani tunnuse lisamine. Otsustati jätkata nelja tegevusega, kuna need olid teostatud korduvalt igal nädalal - uute kullerite lisamine, kulleri andmete muutmine, restorani eeldatava valmistusaja muutmine, restoranide kirjelduste lisamine.

Seetõttu nende nelja valitud sisemiste protsesside automatiseerimise abil võib oluliselt vähendada töötajate töökoormust, parandada tehtavate tegevuste kiirust ja täpsust, vähendades seeläbi vigu, parandades klientide rahulolu ja suurendades üldist ettevõtte töö efektiivsust. Meeskonna väärtuslikke ressursse saab paremini kasutada strateegiliste ülesannete jaoks.

3 Lahenduse analüüs

Antud peatükis määratakse lõputöö skoobis loodava rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded. Lisaks sellele uuritakse erinevaid lähenemisviise, mida valitud protsesside automatiseerimiseks. Lahenduse analüüsi eesmärk on anda ülevaade pakutud lahendusest ja näidata, kuidas see lahendab Peatükis 2. välja toodud väljakutseid. Selles peatükis käsitletakse pakutud lahenduse tehnilisi aspekte, mis on automatiseerimisrakenduse väljatöötamise oluline samm. Kaalutakse erinevaid lähenemisviise ning valitakse kõige sobivamad tööriistad ja metoodikad.

3.1 Nõuded lahendusele

Enne tarkvaraarendusprotsessi algust on ülioluline määratleda nõuded, et määrata kindlaks funktsioonid, millele rakendus peab vastama. Selle protsessi käigus jõuab tarkvara, mis sisaldab vajalikke funktsioone, valmimiseni. Alustuseks on arendusnõuete määratlustel kaks peamist kategooriat, milleks on funktsionaalsed ja mittefunktsionaalsed nõuded [1]. Tabelis 7. on esitatud funktsionaalsed ja mittefunktsionaalsed nõuded.

Funktsionaalsed nõuded määravad kindlaks, mida tarkvara peab tegema. [1] Sellel on oluline roll kasutajate süsteemiga suhtlemise määramisel. Funktsionaalsete nõuete õige tuvastamine kiirendaks arendusprotsessi. Iga funktsioon aitab kaasa süsteemi üldisele kasutatavusele.

Mittefunktsionaalseid nõudeid defineeritakse kvaliteediatribuutidena – see on kirjeldus, kui hästi tarkvarasüsteem oma funktsioone täidab. Sisuliselt funktsionaalsetele nõuetele, mis keskenduvad sellele, mida süsteem teeb ja mittefunktsionaalsed nõuded keskenduvad sellele, kui hästi see toimib. Mittefunktsionaalsed nõuded tagavad, et süsteem vastab sellistele nõuetele nagu jõudlus ja turvalisus. [2]

Tabelis 7. on näidatud vastavalt automatiseerimislahenduse funktsionaalsed ja mittefunktsionaalsed nõuded.

Tabel 7. Automatiseerimislahenduse nõuded

Funktsionaalsed nõuded	**Mittefunktsionaalsed nõuded**
F1. Töötaja peab saama end autentida ehk tõendada.	MF1. Rakendus peab olema kergesti hooldatav ja kohandatav, et see sobiks sisemiste protsesside tulevaste muudatustega.
F2. Andmed peab hankima *Google Sheet*'ist.	MF2. Rakendusel peab olema kasutajasõbralik liides ja seda peab olema lihtne kasutada ettevõtte mittetehnilistele töötajatele.
F3. Töötaja peab suutma lisada uusi kullereid siseveebi rakenduse abil.	MF3. Lahendus peab olema skaleeritav, suutma hakkama saada suure hulga tehingutega ilma süsteemi aeglustamata või kokkujooksmata.
F4. Töötaja peab suutma muuta kulleri andmeid siseveebis rakenduse abil.	
F5. Töötaja peab suutma muuta restorani eeldatava valmistusaega siseveebis rakenduse abil.	
F6. Töötaja peab suutma lisada restoranide kirjeldusi siseveebi rakenduse abil.	

Funktsionaalne nõue (F1) eeldab, et töötaja peab suutma oma isikut autentida, rõhutades rakenduse turvameetmeid, et välistada volitamata isikute juurdepääs sisesüsteemile. Funktsionaalsed nõuded (F3, F4, F5, F6) pärinevad 2. peatükist, kus neid toiminguid kirjeldati täpsemalt. Mittefunktsionaalse nõue (MF2) täituvust on võimalik katsetada

juba rakenduse testimise etapil, andes kolleegidele proovida rakendust kasutada, tulemusena peaks rakendus olema mittetehnilisele töötajale arusaadav. Mittefunktsionaalne nõue (MF3) ehk rakenduse skaleeritavus - peatükis 2. on kirjeldatud mitu tundi nädalas kulub iga tegevuse peale. Automatiseerimise rakenduse arenduse tulemusena võtaks iga tegevus praeguse kestusega võrreldes ligikaudu poole vähem aega.

1. Uute kullerite lisamine: Hetkel võtab antud tegevus 7 tundi nädalas - Pärast peaks võtma 3,5 tundi aega.
2. Kulleri andmete muutmine: Hetkel võtab antud tegevus 3 tundi nädalas - Pärast peaks võtma 1,5 tundi aega.
3. Restorani eeldatava valmistusaja muutmine: Hetkel võtab antud tegevus 2 tundi nädalas - Pärast peaks võtma 1 tunni aega.
4. Restorani kirjelduse lisamine: Hetkel võtab antud tegevus 2,5 tundi nädalas - Pärast peaks võtma 1,25 tundi aega.

Neid nõudeid täites võib rakendus oluliselt tõsta sisemiste protsesside efektiivsust, mille tulemuseks on töötajate töökoormuse vähenemine vabastades neid käsitsitööst. Samuti tagab kasu jätkusuutlikkust hästi dokumenteeritud ja hooldatud lahendus, et seda saaks realiseerida pikemas perspektiivis.

3.2 Tehnoloogia valik

Tänapäeval on mitmeid erinevaid tehnoloogiaid rakenduste arendamiseks, millel kõigil on omad eelised ja puudused. Seetõttu võib parima valiku tegemine osutuda üsna keeruliseks ülesandeks. Kvaliteetne tehnoloogiline alus on see, mis suudab kiiremini skaleeruda kui inimeste arv, kes tegelevad tarkvara hooldamisega. [3] Rakenduse arendamiseks sobiva tööriista valimine on kriitiline otsus, tehnoloogia valikul tuleb lähtuda rakenduse nõuetest ja arhitektuurist. Lõputöö skoop seab ajalised piirangud, mistõttu on mõistlik eelistada juba varasemalt tundma õpitud tehnoloogiad, vältides vajadust nullist alustada ja seeläbi vähendades juurde õppimisele kuluvat aega. Lisaks sellele on oluline kaaluda valitud tehnoloogia eeliseid ja puudusi, skaleeritavust, turvalisust ning selle üldist populaarsust tarkvaraarenduse maailmas. Autor analüüsib

kahte populaarset programmeerimiskeelt automatiseerimislahenduse arendamiseks, milleks on *C#* ja *Python*, võrdlus on toodud allolevas Tabelis 8.

Python on mitmeotstarbeline programmeerimiskeel, mida saab kasutada nii tarkvaraarenduseks, andmetöötluseks kui ka süsteemihalduseks. [4] *Python* on kõrgetasemeline keel, mis tähendab, et see pakub programmeerijale kõrgetasemelisi konstruktsioone ja abstraktsioone. *Python'*i süntaks on sarnane loomuliku keelega ja seda on lihtne õppida isegi algajatel programmeerijatel. Tavaliselt valitakse *Python*'it esimese programmeerimiskeelena, mida ülikoolides õpetatakse. *Python*'it saab kasutada nii skriptimiskeelena – see võimaldab käivitada järjestikku programme ka objektorienteeritud keelena, mis annab võimalus kasutada objektidena ja klassidena defineeritud funktsioone ja meetodeid. [4] *Python* on populaarne oma lihtsuse, ulatusliku teekide toe ja loetavuse tõttu, millepärast on see programmi arendamiseks eelistatud valik. [5] See on eriti kasulik, kuna seda kasutatakse ka laialdaselt automaattestimiseks.

Teise võimalusena on *C#* programmeerimiskeel. *C#* on *Microsoft'*i poolt aastal 2000 välja töötatud staatiliselt tüübitud objektorienteeritud keel, mille abil saab arendada rakendusi mitmesugustele seadmetele, sealhulgas veebiplatvormidele, mobiilseadmetele, arvutitele ja televiisoritele [6]. *C#* põhifunktsioonide hulka kuuluvad objektorienteeritud programmeerimine (OOP), automaatne mäluhaldus prügikogumise kaudu ja komponendipõhise programmeerimise tugi. C# kompileeritud keel, mis tagab kiire ja tõhusa rakenduse arendamisele tugeva aluse. Stackoverflow 2022. aasta uuringu kohaselt kasutas ligikaudu 28% arendajatest just C# programmeerimiskeelt [7]. See sobib eriti hästi *Windows'*i rakenduste, *ASP.NET*-i kasutavate veebirakenduste ja ettevõtte tasemel lahenduste arendamiseks.[8] *C#* keel on saavutanud usalduse paljude ettevõtete seas, eriti nendes, mis nõuavad kõrget jõudlust ja turvalisust, alates pangandusest kuni suurte e-kaubanduse platvormideni, kus töödeldakse ulatuslikke päringuid ja tehinguid [6].

Selles osas uurib autor nende kahe keele eeliseid ja puudusi. Keeltevaheline võrdlus on oluline etapp, kuna selle tulemusel toovad esile nende keelte eripärad, mis võivad mõjutada otseselt arendusprotsessi ja lõpptulemust. Antud võrdlus määraks lõputöö skoobis rakenduse arendamiseks sobivaima taustakeele.

Tabel 8. toob välja *Python*'i ja *C#* kõrgetasemelist võrdlust erinevate aspektide lõikes.

Tabel 8. Python ja C# programmeerimiskeelte võrdlus [9]

Aspekt	***Python***	***C#***
Tippimissüsteem	Dünaamiline tippimine (tõlgendatud keel)	Staatiline tippimine (koostatud keel)
Kasutusjuhtumid	Mitmekülgne – kasutatakse veebiarenduses, andmeteaduses jne	Kasutatakse peamiselt *Windows*'i rakenduste jaoks, *ASP.NET* veebiarenduseks
Süntaks	Puhas ja loetav süntaks, lihtne ka algajatele	Süntaks on sarnane *Java*'ga ja *C++*-ga, olles tuttav nende keelte arendajatele
Kogukonna tugi	Suur ja aktiivne kogukond	Tugev kogukonna tugi, eriti *Microsoft*'i tehnoloogiate jaoks
Platvormi sõltumatus	Väga kaasaskantav – töötab erinevatel platvormidel	Kasutatakse peamiselt *Microsoft*'i ökosüsteemis, kuid platvormiülene tugi kasvab
Ökosüsteem	Laialdased teegid ja raamistikud (nt *Django*, *Flask*)	Rikkalik ökosüsteem, eriti *Windows*'i arendamiseks (*ASP.NET*, *WPF*)
Esitus	Üldiselt aeglasem kui kompileeritud keeled	Üldiselt kiirem jõudlus tänu kompileerimisele

Python ja *C#* on kaks võimsat programmeerimiskeelt, mis sobivad erinevatele ökosüsteemidele ja millel on erinevad tugevused. *Python* on tuntud oma lihtsuse, loetavuse ja mitmekülgsuse poolest, mistõttu on see populaarne valik ülesannete teostamiseks, mis ulatuvad veebiarendusest andmeteaduseni. Teisest küljest on *C# Microsoft'*i välja töötatud keel, mis paistab silma *Windows'*i rakenduste loomisel, *ASP.NET*-iga veebiarendusel ja ettevõtte tasemel lahendustel. [10] Antud lõputöö kontekstis on ei ole *C#* arendus aktuaalne, kuna tabelis 8. juhitakse tähelepanu sellele, et

see programmeerimiskeel on peamiselt mõeldud just *Windows'*i rakenduste loomise jaoks.

Autor otsustas kasutada *Python'*it põhinedes antud programmeerimiskeelele süntaksi puhtusele ja suhtelisele lihtsusele, mis suurendab loetavust ja arenduse kiirust. Suur aktiivne kogukond pakub programmeerimiskeelele eksklusiivset tuge. *Python* on skaleeritav programmeerimiskeel, mis võimaldab arendada nii väikseid kui ka suuri rakendusi. Selle rakenduse üks eeliseid on see, et *Python'*i saab integreerida teiste süsteemide ja keeltega. [11] Veel üks kriteerium tarkvara valikus on tarkvara avatus ehk peab olema *Open-Source* ja kättesaadav. Antud kriteeriumid võimaldavad soovi korral leida internetist võimalikult palju erinevaid abimaterjale. Samuti on teada, et valitud tarkvara on piisavalt paindlik, mitmekülgne ja korduvate toimingute automatiseerimise võimeline.

3.2.1 Raamistik

Järgmise etapi jaoks valib autor raamistiku, mis ühtib harmooniliselt *Python'*i programmeerimiskeelega. Selles kontekstis valik langeb *Selenium'*ile, kuna antud raamistik pakub veebi automatiseerimise jaoks kohandatud tööriistade ja teekide komplekti. [12]

Selenium on raamistik, mis võimaldab veebilehel teostada operatsioone arendaja soovitud viisil. Üks oluline kriteerium oli selle tehnoloogia paindlikkus, mis võimaldab seda kasutada praktiliselt igal veebilehel, erinevalt näiteks *API WebRequest'*idest. *Selenium* kasutamine on lihtne kirjutamisel võrreldes teiste sarnaste tehnoloogiatega ning sobib hästi väikeste rakenduste loomiseks. [13] *Selenium'*i peamine kasutusala on veebirakenduste automaattestimine. *Selenium* võimaldab testide lindistamist ja taasesitamist, kasutades *Firefox'*ile lisatud *Selenium IDE*'t, ilma et oleks vaja testskripti keelt tundma õppida. Lisaks brauseripõhisele testide loomisele pakub *Selenium* spetsiifilist testkeelt nimega *Selenese*, mis võimaldab skripte kirjutada erinevates programmeerimiskeeltes nagu *C#, Java, Python, PHP, Perl* ja *Ruby* ning neid seejärel käivitada erinevates veebibrauserites. [14]

Selenium koosneb mitmest erinevast komponendist:

1. *Selenium IDE* on terviklik integreeritud arenduskeskkond, mis on mõeldud *Selenium'*i testide loomiseks ja on rakendatud *Firefox'*i laiendusena. *Selenium IDE* abil on võimalik salvestada kasutajapoolseid tekstisisestusi, nupuvajutusi ja linkide avamisi ning lisaks võib testjuhtumisse lisada ka laiendusepoolsed käsud, näiteks teksti või elemendi salvestamise. See annab võimaluse koostada teste ilma testskripti keelt valdamata ning on eriti kasulik inimestele, kes soovivad kiiresti ja lihtsalt testida veebilehe funktsionaalsust. [14]
2. *Selenium WebDriver* on *Selenium Remote Control* järeltulija, mis võimaldab vastu võtta *Selenese* või kliendi *API* käske ja saadab need brauserisse. [14]
3. *Selenium Remote Control* on server, mis võimaldab läbi viia erinevates keeltes kirjutatud *Seleniumi* teste ka teistes brauserites. Kliendipõhised draiverid on olemas enamlevinud programmeerimiskeeltele, nagu *Python, Java, Perl , Ruby, PHP* ja *.NET*. [14]
4. *Selenium Grid* on server, mis võimaldab paralleelselt või erinevates virtuaalmasinates käivitada teste erinevate brauseri versioonide ja seadistustega. [14]

Tabel 9. sisaldab erinevusi *Selenium IDE, RC* ja *Webdriver'*i vahel:

Tabel 9. Seleniumi raamistiku komponentide võrdlus [15]

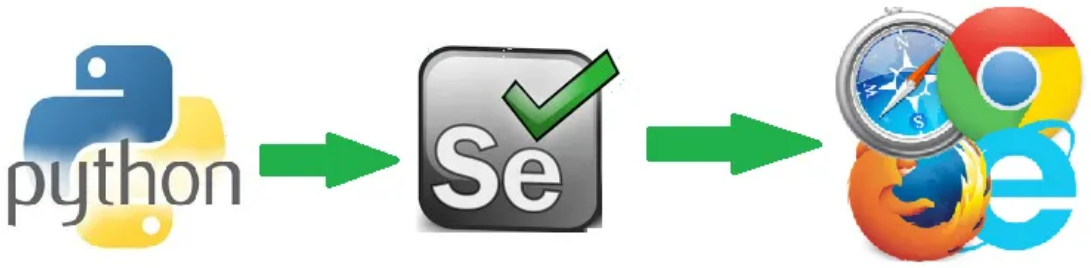
Funktsionaalsus	**Selenium IDE**	**Selenium Remote Control**	**Selenium WebDriver**
Brauser	Seda saab kasutada ainult *Mozilla Firefox*'is	Seda saab kasutada enamiku brauserite jaoks	Seda saab kasutada enamiku brauserite jaoks, sealhulgas peata režiimis
Jõudlus	Kiire, kuna see suhtleb otse brauseriga	Ei suhtle otse brauseriga, veebidraiveriga võrreldes aeglasem	Kiire, kuna see suhtleb otse brauseriga
Objektorienteeritud	Pole objektorienteeritud	Vähesel määral objektorienteeritud *API*	Puhtalt objektorienteeritud *API*
Server	Ei vaja serverit	Vajab serverit	Ei vaja serverit

Funktsionaalsus	**Selenium IDE**	**Selenium Remote Control**	**Selenium WebDriver**
Brauser	Seda saab kasutada ainult *Mozilla Firefox*'is	Seda saab kasutada enamiku brauserite jaoks	Seda saab kasutada enamiku brauserite jaoks, sealhulgas peata režiimis
Jõudlus	Kiire, kuna see suhtleb otse brauseriga	Ei suhtle otse brauseriga, veebidraiveriga võrreldes aeglasem	Kiire, kuna see suhtleb otse brauseriga
Arhitektuur	Tuum on *JavaScript*'i baasil	Tuum on *JavaScript*'i baasil	Suhtleb brauseriga
Kasutamine	Kasutajaliides, mille kaudu saab skripte luua.	Väike ja lihtne *API*	Sisaldab *API*-d ja seda toetavad erinevad keeled

Pärast eelnevat *Selenium IDE, Selenium Remote Control* ja *Selenium WebDriver'*i põhjalikku analüüsi on autor jõudnud strateegilise otsuseni – võrreldes iga komponendi eristavaid tugevusi ja võimalusi, järeldab autor, et *Selenium WebDriver* oleks optimaalne valik automatiseerimislahenduse arendamiseks. *Selenium IDE* ei sobi sellepärast, et antud raamistik on kooskõlas ainult *Mozilla Firefox* veebibrauseriga, mis piirab kasutatavust. *Selenium Remote Control* samuti ei sobi, kuna raamistik ei suhtle otse brauseriga ning on veebidraiveriga võrreldes aeglasem. *Selenium WebDriver* aga ühtib otseselt projekti eesmärkidega (tööülesannete kiirem täitmine) tänu oma tugevatele funktsioonidele, ühilduvusele erinevate programmeerimiskeeltega keskendudes brauseri automatiseerimisele. Üks *Selenium WebDriver'*i kasutamise eeliseid rakenduste arendamiseks on selle võime automatiseerida korduvaid toiminguid, nagu andmete sisestamine, vormide esitamine. See on eriti kasulik antud lõputöö raames – sisemiste toimingute puhul, kus paljud platvormi haldamisega seotud protsessid võivad olla korduvad ja käsitsi tehes aeganõudvad.

Selenium Webdriver on avatud lähtekoodiga testimisraamistik, mida kasutatakse laialdaselt veebirakenduste automatiseerimiseks. See toetab mitmesuguseid programmeerimiskeeli, nagu – *Java, C#, Python, Ruby, Perl, PHP, JavaScript*, mis lubab arendajal kasutada oma lemmiktööriistu. *Selenium WebDriver'*i suurimaks

eeliseks on võime toetada mitmeid erinevaid veebilehitsejaid – *Google Chrome, Internet Explorer, Firefox, Safari, Opera, HtmlUnit* ja *phantomjs*. [16]



Joonis 6. Automatiseerimine *Python*'ni ja *Selenium*'iga veebis [17]

Selenium WebDriver sobib hästi samuti ka mobiilsete seadmete tarkvaralahenduste testimiseks. Erinevalt teistest testimisvahenditest puudub *Selenium WebDriver*'il kasutajaliides, seega tuleb seda kasutada programmeerimise keskkonnas, sest ka vigade haldamine toimub seal. *Selenium WebDriver*'it saab paigaldada oma projekti teekide täiendamise kaudu programmeerimise keskkonnas. *Selenium WebDriver'*i kaks olulisemat klassi on *WebDriver* ja *WebElement*. *WebDriver*'i objekt esindab brauseri draiverit ning juhib brauseri käitumist. *WebElement*'i objekt esindab veebilehel olevat elementi. [16]

4 Lahenduse realiseerimine

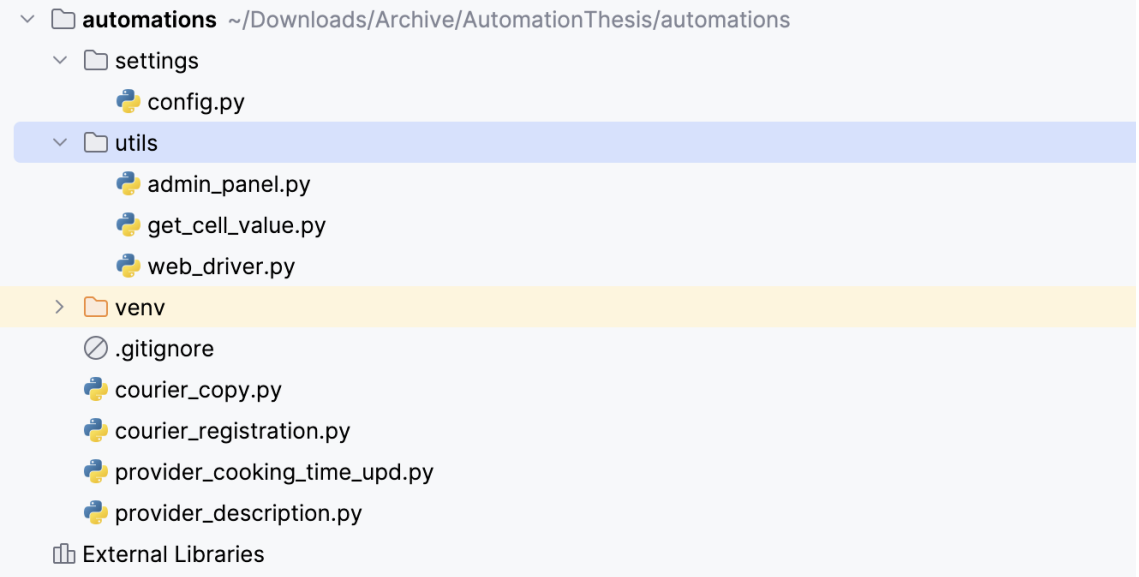
Selles peatükis keskendub autor eelmises peatükis pakutud lahenduse rakendamisele ning selle nõuetele. Eesmärk on automatiseerida käsitsi teostatavad toimingud, mis peatükis 2. tuvastati sisemise süsteemi jaoks vajalikuks. Eelmises peatükis valitud tehnoloogiat kasutatakse valitud toimingute jaoks automatiseeritud skriptide loomiseks. Automatiseerimislahendust luuakse nelja tegevuse jaoks - uute kullerite lisamine, kulleri andmete muutmine, restorani eeldatava valmistusaja muutmine, restoranide kirjelduste lisamine. Käesolevas peatükis keskendutakse süsteemi väljatöötamise tehnilistesse üksikasjadesse ning arutatakse protsessi käigus kasutatud tööriistade ja tehnoloogiate kohta. Peatüki peamine eesmärk on anda põhjalik ülevaade lahenduse arendusprotsessist.

4.1 Projekti struktuur

Koodibaas on struktureeritud erinevatesse kaustadesse. Järgnevalt on kirjeldatud lähemalt projekte sisaldavate kaustade sisu:

- *Settings* kaust sisaldab konfiguratsioonifaili, kus on salvestatud mitmed muutujad, näiteks brauseri tüüp, *JSON* faili andmed *Google'*i kontoga ühenduseks, andmeallika link ja *admin-panel* konto seaded.
- *Util* kaust sisaldab utiliitfunktsioone, mida kasutatakse mitmesuguste ülesannete jaoks, näiteks suhtlemine siseveebiga ning failide *get_cell_value.py* ja *web_driver.py* kasutamiseks, mille funktsionaalsus on kirjeldatud peatükkides 4.3.1 ning 4.3.2.
- Peamised koodifailid, mis käivitavad automatiseerimisprotsesse on nimetatud vastavalt nende funktsioonidele (nt. "*courier_registration.py*" sisaldab koodi uute kullerite lisamise automatiseerimiseks). Iga koodifail on üles ehitatud sarnaselt: peamine funktsioon käivitatakse teatud ajavahemikul ning seejärel toimub vajaliku töölehe lugemine *Google Sheets*'is ja selle põhjal toimub automatiseerimise protsess.

Kaustade struktuuri kuvatõmmis on välja toodud joonisel 7.



Joonis 7. Projekti struktuur

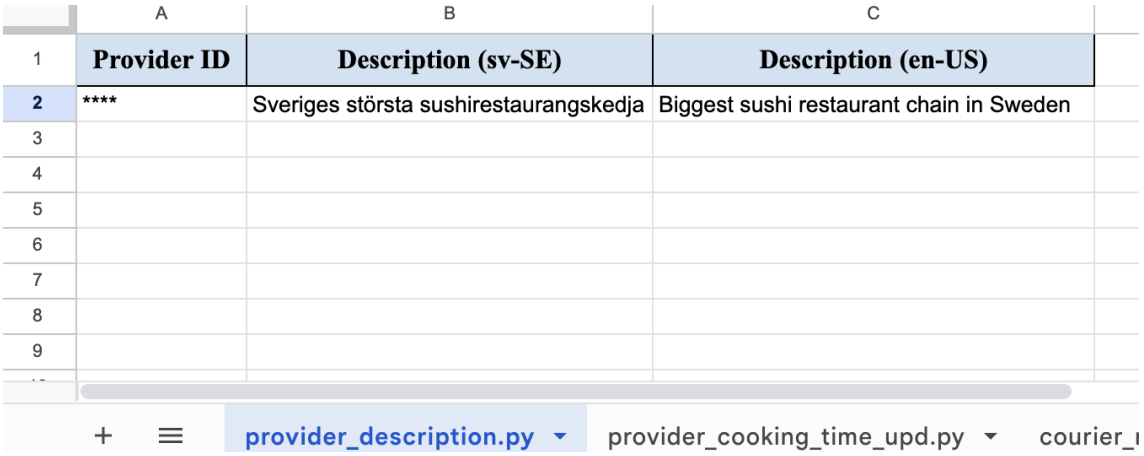
4.2 Rakenduse andmeallikad

Selle lõputöö raames välja töötatud sisesüsteemi automatiseerimisrakendus on loodud suhtlema otseselt *Google Sheets*'iga kui andmeallikaga. *Google Sheets* on pilvepõhine arvutustabeli tarkvara, mis võimaldab kasutajatel võrgus arvutustabeleid luua, muuta ja jagada. See on osa *Google Drive*'i tootlikkuse tööriistade komplektist, mis sisaldab veel selliseid rakendusi nagu *Google Docs, Slides* ja *Forms*. [18] Tarkvara pakub palju funktsioone, sealhulgas diagramme ja graafikuid, tingimuslikku vormindamist, andmete valideerimist, liigendtabeleid jms. Võib öelda, et funktsioonid on samad nagu ka teistes arvutustabeli rakendustes, näiteks *Microsoft Excel*. *Google Sheets*'i kasutamise üks peamisi eeliseid on see, et sellele pääseb ligi kõikjalt, kus on internetiühendus, mistõttu on see paindlik ja mugav tööriist andmete haldamiseks ja analüüsimiseks. Samuti pakub see reaalajas värskendusi, et mitu kasutajat saavad sama dokumendiga samaaegselt töötada.

Skriptid suudavad hankida lehtedelt asjakohaseid andmeid ja kasutada seda erinevate toimingute tegemiseks siseveebis *admin-panel*. *Google Sheets*'i dokumendi lehtede

nimetamine on seadistatud nii, et see vastab otseselt rakenduse sees olevatele automatiseerimisfailide nimedele (Joonis 7.). Samuti iga *Google Sheets*'i lehe esimene rida on kasutusel, et määratleda, milliseid *HTML*-i elemente *admin-panel* rakenduses tuleks vastavate andmetega uuendada. See tähendab, kui lehtede nimesid muudetakse, tekitab see skriptide ja andmeallika vahel ebakõla ning takistab koodide korrektset töötamist. Seetõttu on automatiseeritud toimingute õige funktsionaalsuse tagamiseks on oluline säilitada samad nimetused.

1. Restorani kirjelduse andmeallikas (Joonis 8.). Lehe nimetus on *provider_description.py*, mis vastab ka faili nimetusele (Joonis 7.). Esimene veerg '*Provider ID*' – selle numbri järgi leiab programm õige restorani lehekülje. Veerud '*Description (sv-SE)*' ja '*Description (en-US)*' vastavad siseveebilehe lahtritele, kuhu rakendus sisestab restorani kirjelduse teksti rootsi ja englise keeltes.



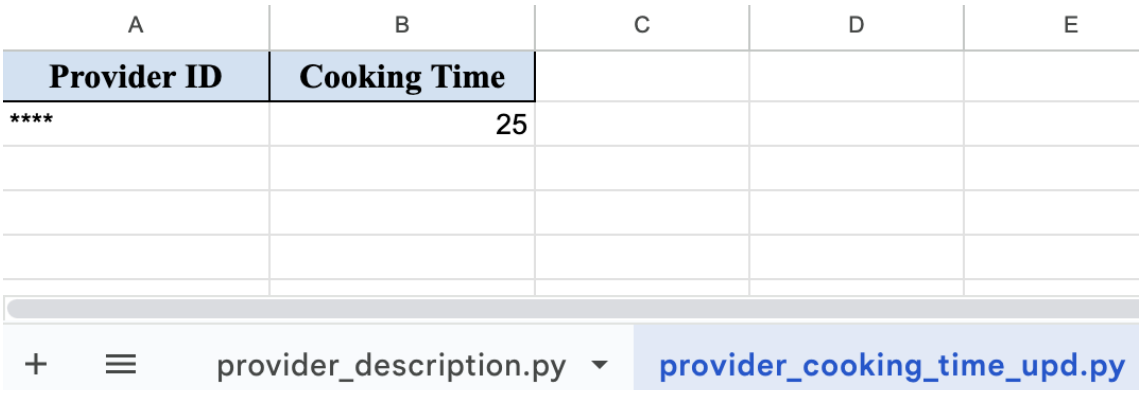
Joonis 8. Restorani kirjelduse allikas

2. Kulleri lisamine – lehe nimetus on *courier_registration.py*, mis vastab ka faili nimetusele (Joonis 7.). Esimese rea lahtrite nimetused vastavad siseveebilehe lahtritele, kuhu rakendus sisestab kõik vajalikud kullerite andmed. Need lahtrid peaksid sisaldama isikuandmeid – eesnimi, perekonnanimi, isikukood, e-posti aadress, pangaandmed, elukoha aadress.

3. Kulleri profiili muutmine – lehe nimetus on *courier_copy.py,* mis vastab ka faili nimetusele (Joonis 7.). Esimene veerg on '*courier_id*' – selle numbri järgi leiab programm õige kulleri profiili, mida tuleb kopeerida. Esimese rea lahtrite

nimetused vastavad siseveebilehe lahtritele, kuhu rakendus sisestab kõik vajalikud kullerite andmed. Need lahtrid peaksid sisaldama isikuandmeid – eesnimi, perekonnanimi, isikukood, e-posti aadress, pangaandmed, elukoha aadress.

4. Restorani valmistusaja muutmine (Joonis 9.) Lehe nimetus on *provider_cooking_time_upd.py*, mis vastab ka faili nimetusele (Joonis 7.). Esimene veerg '*Provider ID*' – selle numbri järgi leiab programm õige restorani lehekülje. Veerg '*Cooking time*' vastab siseveebilehe lahtritele (Joonis 4.), kuhu rakendus sisestab uuendatud toidu valmistusaja.



Joonis 9. Restorani valmistusaja allikas

4.3 Arendus

Antud peatükk keskendub selle lõputöö skoobis kirjeldatud rakenduse arendamisele. Rakendus on loodud kasutades *Pytho*n programmeerimiskeelt ja *Selenium Webdriver* raamistikku. Integreeritud arenduskeskkonnana (IDE) kasutati *JetBrains PyCharm* arendustarkvara. *PyCharm* on populaarne integreeritud arenduskeskkond (IDE) *Python'*i programmeerimiskeele jaoks, mida pakub *JetBrains*. [11] *PyCharm'*i peamine eelis on selles, et see võimaldab arendajal kirjutada ja käivitada *Python'*i koodi kõikidel suurematel platvormidel. *PyCharm* pakub kasutajasõbralikku liidest. Lisaks sellele pakub *PyCharm* võimsaid funktsioone nagu – tüübi tuvastamine, automaatne täitmine, refaktoreerimine, testimine ja silumine, mis muudavad *Python'*i koodi kirjutamise lihtsaks ja tõhusaks. Samuti saab *PyCharm*'is kasutada erinevaid väliseid pluginaid ja

lisasid, et veelgi laiendada funktsionaalsust. [11] Järgnevad alampeatükid toovad välja mõned programmikoodi funktsioonid.

4.3.1 Põhikomponendid

Peamised koodifailid *courier_copy.py, courier_registration.py, provider_description.py,* ja *provider_cooking_time_upd.py* on mõeldud erinevate restoranide ja kulleritega seotud ülesannete automatiseerimiseks *admin-panel* süsteemis. Need failid sooritavad *Selenium*'i abil soovitud automatiseerimise saavutamiseks mitu sammu:

1. *Google Sheets*'i lehe lugemine (Joonis 10.);

```
creds = ServiceAccountCredentials.from_json_keyfile_dict(js_dump,
scope)  # Initialise credentials for GSheet API
sheetname = os.path.basename(__file__)  # Get name of the current
script and use it to find a list with same name in Gsheet file
client = gspread.authorize(creds)  # Connect to API
spreadsheet = client.open_by_url(doc_url)  # Open spreadsheet
database = spreadsheet.worksheet(sheetname)  # Open the needed list
df = pd.DataFrame(database.get_all_records()).fillna('')  # Read data
for script
print('Spreadsheet data from', sheetname, 'list has been read.')
```

Joonis 10. Google Sheets'i lugemine

2. Veebibrauseri käivitamine;
3. Navigeerimine soovitud veebilehele veebibrauseris;
4. Vajalike *HTML* elementide otsimine veebilehelt;
5. Leitud *HTML* elementidega soovitud toimingute tegemine (nt teksti sisestamine ja nupu vajutamine);
6. Tulemuste kontrollimine (nt konkreetse *HTML* elemendi olemasolu);
7. Tulemuste väljastamine kasutajale (konsoolile);
8. Veebibrauseri sulgemine.

4.3.2 *Selenium Webdriver*'i haldus

Joonisel 11. on toodud kood, mis seadistab *Chrome* interneti brauserit kasutades *Selenium WebDriver'*it. Esmalt imporditakse vajalikud moodulid *Chrome*'i brauseri kasutamiseks *Selenium'*iga. Seejärel loob see *Service* objekti, kasutades *ChromeDriverManager'*it, et installida viimane *ChromeDriver'*i versioon, mis on vajalik *Selenium'*i jaoks, et suhelda *Chrome* brauseriga.
Lõpuks algatab uue *Chrome*'i eksemplari *Service* objektiga, mida saab kasutada veebilehtedel navigeerimiseks, elementidega suhtlemiseks ja teiste toimingute tegemiseks *Chrome* brauseris.

```
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver import Chrome

driver =
Chrome(service=Service(ChromeDriverManager().install()))
```

Joonis 11. Seleniumi allalaadimine

Lisades selle koodilõigu otse rakendusse, ei pea kasutajad *Selenium'*i oma arvutisse käsitsi alla laadima ja paigaldama. Ilma selle automaatse integreerimiseta peaksid kasutajad iseseisvalt *Selenium'*i paketi hankima, mis muutub iga järgneva värskendusega eriti ebamugavaks.

4.3.3 Abimeetod

Joonisel 12. on toodud kood, mida kasutatakse kullerikontaktide telefoninumbrite vormindamiseks. Sisesüsteem nõuab, et telefoninumbrid algaksid sümboliga "+", kuid kullerite poolt edastatud numbritel ei pruugi see sümbol alati olla. Seetõttu kontrollib see kood, kas antud rea telefoninumbri lahter sisaldab kehtivat telefoninumbrit (mitte-*NaN*-i *float*) ja kui antud telefoninumber ei alga sümboliga "+", lisab kood sümboli numbri algusesse.

```
import math

def get_cell_value(row, cell_name):
    val = getattr(row, cell_name)

    if isinstance(val, float) and not
math.isnan(val):
        val = int(val)

    if cell_name == 'partner_phone' or cell_name ==
'partner_private_phone' and not
str(val).startswith("+"):
        val = '+' + str(val)

    return val
```

Joonis 12. Telefoninumbri formateerimine

4.4 Rakenduse turvalisus

Programmi turvalisuse tagamiseks on oluline vältida volitamata juurdepääsu ja andmeleke riske. Siseveeb *admin-panel* on kujundatud turvameetmetega, mis tagavad selle poolt töödeldavate andmete kaitset. Üks nendest meetmetest on juurdepääsu piiramine ainult ettevõtte sisestele töötajatele.

Admin-panel lehele pääsemiseks on vajalik konkreetne sertifikaat, mis on seotud töötaja arvutiga. See sertifikaat ei saa olla üle kantav teisele arvutile ning iga töötajale on määratud eraldi sertifikaat. See tähendab, et ainult volitatud töötajad saavad juurdepääsu siseveebi lehele ja teostada selles toiminguid.

Rakenduse käivitamisel, enne soovitud veebilehele sisenemist, peab kasutaja sertifikaadi kinnitama. See autentimisprotsess on oluline kontrollpunkt, tagades turvalise ja usaldusväärse ühenduse enne sihitud lehele sisenemise lubamist. Kui sertifikaati ei kinnitata, keelatakse kasutajal juurdepääs lehele. Veebileht kuvab *HTTP*-tõrge, mis annab märku olekust '*404 Not Found*'. See tõrge ilmneb siis, kui server ei suuda soovitud lehte või ressurssi leida. Sisuliselt on tõrge '*404 Not Found*' serveri viis teatada, et kasutaja määratud *URL* ei vasta serveris olevale sisule. [19].

4.5 Vigadehaldus

Juhtudel, kui sisestatud andmed on valed või kui konkreetne toiming ebaõnnestub, teavitab programm kasutajat veast ja täpsustab, milliste kullerite või restoranide puhul toimingut ei tehtud (Joonis 13.).

```
  except:
        print(" [FAILED]")
        details = admin_panel.collect_page_errors() or
(extra_step_info + [script_step + " error"])
        couriers_not_registered.append((get_cell_value(row,
"partner_phone"), details))

print("Phone numbers of not registered couriers:")
    for failed_item in couriers_not_registered:
        print(failed_item[0])
        print("\n".join(failed_item[1]))
        print()
```

Joonis 13. Kulleri registreerimise viga

Näiteks, kui sisselogimisega on probleeme, teavitab programm kasutajat veast. Samamoodi, kui sisestatud andmetega on probleeme, määrab programm, millised väljad sisaldavad valeandmeid. Juhtudel, kui toiming ebaõnnestub, näiteks kulleri profiili värskendamisel, teavitab programm kasutajat konkreetsest kullerist, kelle puhul toiming ebaõnnestus.

4.6 Testimine

Testimine on tarkvaraarenduse oluline komponent. Testimine tagab, et rakendused töötavad korralikult kõigis stsenaariumites. Testimise tulemus kinnitab, et rakendus vastab etteantud nõuetele. [20]

1. Üksuste testimine: Testimisprotsess algas üksuse testimisega, mille käigus kontrolliti üksikute funktsioonide ja moodulite õigsust. Üksuste testimine on oluline probleemide tuvastamiseks ja parandamiseks põhitasandil, tagades rakenduse koodibaasi terviklikkuse.

2. Läbivtestimine: Rakenduse jõudluse hindamiseks kogu töövoo jooksul viidi läbi põhjalik testimine. See hõlmas täieliku automatiseerimisprotsessi testimist alates toimingute algatamisest kuni lõpptulemuste kontrollimiseni. Läbivtestimine aitab tuvastada võimalikke probleeme, mis võivad tekkida, kui erinevad komponendid üksteisega suhtlevad.
3. Andmepõhine testimine: Autor rakendas andmepõhise testimise, et hinnata rakenduse töökindlust erinevatel olukordadel. See hõlmas erinevate sisendandmetega testide läbiviimist, et tagada rakenduse täpne ja usaldusväärne reageerimine erinevatele tingimustele.

4.7 Rakenduse kasutamist toetavate dokumentide koostamine

Selles peatükis käsitleme rakenduse ettevalmistamiseks vajalikke samme. See hõlmab kogu vajaliku tarkvara installimist ja konfigureerimist, samuti vajalike sisendandmete seadistamist. Eesmärk on anda selge samm-sammuline juhend, mis aitab autori kolleege rakenduse ettevalmistamisel, mida saab kasutada edaspidiseks kasutamiseks viitena.

1. *Python*'i paigaldus: Laadi alla ja jooksuta *Python'*i paigalduspakett selleks, et seda arvutisse paigalda.
2. *PyCharm*'i paigaldus: Laadi alla *Pycharm* Mac-arvutile (*Community* versioon; tasuta, avatud lähtekoodiga). Jooksuta paigalduspakett ja paigalda arvutisse.
3. Navigeeri jaotisesse '*Python Packages*'. Otsi '*pandas*' ja paigalda arvutisse. Tee läbi sama protseduur '*selenium*', '*gspread*', '*webdriver-manager*' ja '*oauth2client'* jaoks.
4. Andmeallikate ettevalmistamine: Loo koopia pakutud *Google Sheets*'i mallist.
    - Andmebaas peab sisaldama kõiki välju, mida vajate partneri ja kulleriprofiilides sisestamiseks (tekstikastid, märkeruudud, rippmenüüd).
    - Lehtede nimetused peavad olema samad, mis automatiseerimis failide nimed.

    Pärast andmefaili loomist jagage see etteantud meili aadressiga. See on süsteemi konto, mida kasutatakse juurdepääsuks Spreadsheedi andmetele API kaudu.
5. Koodide allalaadimine ja seadistamine: Laadi alla kaust koos koodiga ja paki lahti arhiiv. Ava *PyCharm*, ülemises tööriistaribal vajuta '*File*' → '*Open*', vali

lahtipakitud kaust ja vajuta ‘*Open*’. Ava ‘*config.py*’ ja uuenda seal oma kasutajanimi ja parool *admin-panel* süsteemi jaoks. Samas failis ühenda ‘spreadsheet’ oma *Google Sheets* andmebaasiga.

6. Programmi käivitamiseks valige fail, mida soovite käivitada, ja valige ‘*Run 'failinimi.py*'’. Pärast brauseri avanemist peate vajutama oma süsteemi sertifikaadil nuppu ‘Ok’.

5 Tulemused

Käesolevas peatükis antakse ülevaade antud lõputöö tulemustest. Antud töö eesmärk oli luua automatiseerimislahenduse, mis aitaks kaasa Rootsi *Bolt Food'*i sisemiste protsesside toimimise kiirusele. Arenduse tulemusena on alguses seatud põhieesmärk täidetud ning rakendus täidab kõiki analüüsis määratud nõudeid. Lahenduse analüüsis määratud funktsionaalsed ja mittefunktsionaalsed nõuded said arendusprotsessi käigus täidetud. Loodud tarkvara abil automatiseeriti järgmised toimingud – kullerite lisamine, kulleri andmete muutmine, restorani eeldatava valmistusaja muutmine, restoranide kirjelduste lisamine.

5.1 Tulemuste analüüs

Peatükis 2 analüüsiti erinevaid kullerite ja restoranide haldamisega seotud protsesse läbi siseveebi *admin-panel*. Analüüs hõlmas tehtud taotluste arvu ja igale protsessile kulutatud aja jälgimist. Selles peatükis käsitletakse põhjalikumalt automatiseerimise tulemusel saadud tulemusi ja tehakse kogutud andmete põhjal järeldusi.

1. Uute kullerite lisamine: käsitsi tehes kulub iga uue kulleri profiili loomiseks umbes 9 minutit aega, seega keskmiselt 7 tundi nädalas. Peale protsessi automatiseerimist iga uue kulleri profiili loomiseks kulub umbes 2 minutit aega. (Tabel 10.) Seega automatiseerimise tulemusena muutus ühe kulleri aktiveerimise protsess 4.5 korda kiiremaks.

Tabel 10. Kullerite aktiveerimine - tulemused

Nädal	**Kullerite arv**	**Aeg(tundides)**
Nädal 4	33	1.1t
Nädal 5	41	1.36t
Nädal 6	29	58 min

2. Kulleri andmete muutmine: käsitsi tehes kulub iga kulleri profiili muutmiseks umbes 9 minutit aega, seega keskmiselt 3 tundi nädalas. Peale protsessi

automatiseerimist iga kulleri profiili muutmiseks kulub samuti umbes 2 minutit aega. (Tabel 11.) Seega automatiseerimise tulemusena muutus ühe kulleri profiili muutmise protsess 4.5 korda kiiremaks.

Tabel 11. Kullerite profiili muutmine - tulemused

Nädal	**Kullerite arv**	**Aeg(minutites)**
Nädal 4	19	38 min
Nädal 5	16	32 min
Nädal 6	22	44 min

3. Restorani toidu eeldatava valmistusaja muutmine: analüüsi raames selgus, et toiduvalmistamisaja muutmine toimub umbes 33 korda nädalas ning käsitsi tehes sellele kulub keskmiselt 2 tundi nädalas. Peale protsessi automatiseerimist kulub eeldatava valmistusaja muutmisele alla 20 minutit nädalas, mis on rohkem kui kaks korda kiirem. (Tabel 12.)

Tabel 12. Restorani valmistusaja muutmine - tulemused

Nädal	**Restoranide arv**	**Aeg(minutites)**
Nädal 4	37	15 min
Nädal 5	34	13,5 min
Nädal 6	45	18 min

4. Restoranide kirjelduste lisamine: käsitsi tehes restoranide kirjelduste lisamisele/muutmisele kuulub keskmiselt 2,5 tundi nädalas. Peale protsessi automatiseerimist kulub selle protsessile umbes 1 tund aega nädalas, mis on rohkem kui kaks korda kiirem kui varem. (Tabel 13.) Juhul kui restoranil oli juba kirjeldus olemas see tuleb eelnevalt kustutada, mis pikendab aega 10-15 sekundi võrra.

Tabel 13. Restoranide kirjelduse lisamine - tulemused

Nädal	Restoranide arv	Aeg(minutites)
Nädal 4	30	57 min
Nädal 5	27	51 min
Nädal 6	28	53 min

Mittefunktsionaalne nõue (MF3) ehk rakenduse skaleeritavus eeldas, et automatiseerimislahenduse arenduse tulemusena võtaks iga tegevus ligikaudu poole vähem aega võrreldes endise kestusega. Tulemuste analüüsi põhjal selgus, et antud nõue on täidetud järgmiselt.

1. Uute kullerite lisamine: Eelnevalt võttis see tegevus 7 tundi nädalas - Automatiseerimislahenduse abil võtab uute kullerite lisamine 1,5 tundi aega, mis on umbes 4 korda kiirem kui varem.
2. Kulleri andmete muutmine: Eelnevalt võttis see tegevus 3 tundi nädalas - Automatiseerimislahenduse abil võtab kullerite andmete muutmine keskmiselt 40 minutit aega, mis on umbes 4 korda kiirem kui varem.
3. Restorani toidu eeldatava valmistusaja muutmine: Eelnevalt võttis see tegevus 2 tundi nädalas - Automatiseerimislahenduse abil võtab restorani toidu eeldatava valmistusaja muutmine 20 minutit aega, mis on umbes 6 korda kiirem kui varem.
4. Restorani kirjelduse lisamine: Eelnevalt võttis see tegevus 2,5 tundi nädalas - Automatiseerimislahenduse abil võtab restorani kirjelduse lisamine 1 tunni aega, mis on umbes 2,5 korda kiirem kui varem.

Tulemuste analüüsi põhjal selgus, et valitud nelja tegevuste automatiseerimine vähendas töötajate töökoormust ning tehtavatele tegevustele kuluvat aega ja suurendades üldist efektiivsust. Suureks eeliseks osutus töötajate ressursside kasutamine muudel eesmärkidel rakenduse töötamise ajal.

5.2 Projekti tulevik

Lõputöö raames valminud automatiseerimise rakendus on edukalt välja töötatud ja ettevõttes rakendatud. Järgmiseks sammuks oleks kaaluda, kas programmi saab laiendada ja täiustada edasiste valdkondade ja funktsioonide katmiseks. Praegune programmi versioon on mõeldud ainult autori meeskonnaliikmetele, kuid kui tulemused osutuvad rahuldavateks, võib programmi soovitada ka teiste riikide meeskondadele.

6 Kokkuvõte

Käesoleva lõputöö eesmärgiks oli esiteks analüüsida Rootsi *Bolt Food'*i osakonna operatiiv meeskonna sisemisi käsitsi tehtavad toiminguid. Käsitsi tehtavad ülesanded on monotoonsed, aeganõudvad ja suure andmemahuga, põhjustades vigu ja viivitusi teenuste osutamisel. Teiseks lõputöö projekti eesmärgiks oli töötada välja automatiseerimislahenduse osakonna ettevõttesiseste tegevuste jaoks, aitaks kaasa protsesside toimimise kiirusele.

Lahenduse väljatöötamisel esmalt analüüsiti probleemi olemust. Kirjeldati erinevaid käsitsi tehtavaid sisemisi protsesse, mis on seotud Rootsi *Bolt Food'*i platvormiga ning toodi välja, mida saab automatiseerida. Analüüs hõlmas tehtud taotluste arvu ja igale protsessile kulutatud aja jälgimist. Analüüsi alusel määrati loodava rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded. Nõuete põhjal koostati tehniline analüüs automatiseerimislahenduse loomiseks, mis hõlmab endas tehnoloogiate valikut. Lõputöö praktilises osas arendati analüüsile ja nõuetele tuginedes automatiseerimislahendus.

Selle töö raames loodud tarkvara abil automatiseeriti järgmised protsessid – uute kullerite lisamine, kulleri andmete muutmine, restorani eeldatava küpsetusaja muutmine, restoranide kirjelduste lisamine, mis vähendades sellele tavapäraselt kuluvat tööaega. Programmi kasutamisel on mitmeid eeliseid võrreldes käsitsitööga. Näiteks, võimaldab programm kiiremini ja täpsemalt andmeid töödelda ning vähendab vigade tekke tõenäosust. Samuti on oluline märkida, et programmi turvalisus on tagatud mitmete meetmetega, näiteks piiratud juurdepääs sertifikaadiga töötajatele. Automatiseerimise tulemusel muutus valitud protsessidele kuluv aeg kaks kuni kuus korda vähemaks.

Kokkuvõttes võib öelda, et töötulemused on rahuldavad ning automatiseerimislahendus on oluline abivahend Rootsi *Bolt Food'*i osakonna teenuste haldamiseks. Tulevikus võib programmiga laiendada funktsionaalsust ning pakkuda seda ka teistele ettevõte meeskondadele kasutamiseks.

Kasutatud kirjandus

[1] K. E. Wiegers and J. Beatty, Software Requirements. Microsoft Press, 2013.

[2] Al-Saqqa, S. (2020) Agile Software Development: Methodologies and Trends. The University of Jordan, https://pdfs.semanticscholar.org/2fef/154748093288894dbd0b98db1b9b54731c71.pdf Kasutatud 10.04.2023.

[3] Y. Brikman, “Choosing a Tech Stack,” in Hello, Startup, United States: O'Reilly Media, Incorporated, 2015

[4] “Python Documentation.” *Python.org*, 2022, https://www.python.org/doc/. Kasutatud 15.04.2023.

[5] Martelli, A., Ravenscroft, A.M., & Holden, S. (2023) Python in a Nutshell. (20-60). https://books.google.ee/books?hl=en&lr=&id=2WSmEAAAQBAJ&oi=fnd&pg=PT28&dq=Python&ots=oUr0HXQx48&sig=P_iPFPNoTQyry1INoek91VGbweQ&redir_esc=y#v=onepage&q=Python&f=false

[6] Jason Hales, Almantas Karpavicius, and Mateus Viegas, The C# Workshop. Packt Publishing, 2022.

[7] Stackoverflow, “2022 Developer Survey,” 2022. Kättesaadav: https://survey.stackoverflow.co/2022/. Kasutatud 09.04.2023.

[8] “C# documentation.” Microsoft Learn, https://learn.microsoft.com/en-us/dotnet/csharp/. Kasutatud 29.09.2023.

[9] Protasiewicz, Jakub. “Comparing Python vs. C Sharp: A Comprehensive Guide to Choosing the Right Programming Language.” Netguru, 2 October 2023, https://www.netguru.com/blog/python-vs-c-sharp. Kasutatud 30.09.2023.

[10] Shankar, Ramya. “C# vs Python: Head to Head Comparison [Updated].” Hackr.io, 11 August 2023, https://hackr.io/blog/c-sharp-vs-python. Kasutatud 30.09.2023.

[11] “PyCharm: the Python IDE for Professional Developers by JetBrains.” *JetBrains*, https://www.jetbrains.com/pycharm/. Kasutatud 14.04.2023.

[12] S. Nyamathulla , Dr. P. Ratnababu , Nazma Sultana Shaik, Bhagya Lakshmi. N 2021. A Review on Selenium Web Driver with Python. Annals of the Romanian Society for Cell Biology. (Jun. 2021), 16760–16768.

[13] Jain, Vaibhav, and Dr. Kumar Rajnish. “Comparative Study of Software Automation Testing Tools: OpenScript and Selenium.” *"International Journal of Engineering*

Research and Applications, vol. 8, no. 2, 2018, pp. 29-33. *IJERA*, https://www.ijera.com/. Kasutatud 12.04. 2023.

[14] "What is Selenium?" *Selenium WebDriver*, https://www.selenium.dev/. Kasutatud 12.04. 2023.

[15] Bhattacharjee, Debomita. "Difference between selenium IDE RC WebDriver." *Tutorialspoint*, 28 August 2020, https://www.tutorialspoint.com/difference-between-selenium-ide-rc-and-webdriver#. Kasutatud 12.04.2023.

[16] "The Selenium Browser Automation Project." *Selenium*, 2 March 2023, https://www.selenium.dev/documentation/. Kasutatud 12.04.2023.

[17] Bourdeau, Stephanie. "Get Started with Selenium WebDriver in Under 5 Minutes." Medium, 19 November 2019, https://medium.com/swlh/get-started-with-selenium-webdriver-in-under-5-minutes-f9b91e2e9539. Kasutatud 02.10.2023.

[18] "Google Sheets: Online Spreadsheet Editor." Google, https://www.google.com/sheets/about/. Kasutatud 18.04.2023.

[19] "e-Teatmik: IT ja sidetehnika seletav sõnaraamat." http://vallaste.ee/. Kasutatud 21.04.2023.

[20] Whiting, E., & Datta, S. (2022). Design and Development of a Technology-Agnostic NFR Testing Framework: Introducing the framework and discussing the future of load testing in Agile software development. ACM Digital Library. https://dl.acm.org/doi/abs/10.1145/3520084.3520092

[21] Gundecha, Unmesh. *Selenium Testing Tools Cookbook*. Packt Publishing Ltd, 2012. Kasutatud 27.03.2023.

[22] Minh, Quan. "Implementing test automation with Selenium WebDriver." *Theseus*, 2018, https://www.theseus.fi/bitstream/handle/10024/153831/Dao_Quan.pdf. Kasutatud 05.04.2023.

[23] "Selenium Tutorial - Switch Career from Manual To Automation." *QTPselenium*, 3 March 2021, https://www.qtpselenium.com/selenium-tutorial. Kasutatud 17.04.2023.

[24] "Selenium WebDriver For Creating Scrips – Selenium Tutorial #8." *Software Testing Help*, 25 March 2023, https://www.softwaretestinghelp.com/selenium-webdriver-selenium-tutorial-8/. Kasutatud 17.04.2023.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Karina Kiris

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Ettevõttesiseste tegevuste automatiseerimislahenduse arendus" , mille juhendaja on Maili Markvardt.
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

04.01.2024

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.