TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Tarkvaratehnika (100864) õppekava

Eerik Muuli

Graafiliste ülesannete automaatkontroll programmeerimise vaba juurdepääsuga e-kursuste raames

Magistritöö (30 EAP)

Juhendaja: Eno Tõnisson

Tartu 2017

Graafiliste ülesannete automaatkontroll programmeerimise vaba juurdepääsuga e-kursuste raames

Lühikokkuvõte:

Programmeerimisülesannete automaatne kontrollimine on vaba juurdepääsuga e-kursuste ehk MOOCide (Massive Open Online Course) juures hädavajalik suure hulga esitatud lahenduste tõttu. Ülesannete nõuded peavad olema detailselt sõnastatud, et neid oleks võimalik automaatselt kontrollida. Väga täpsed nõuded ülesannetele piiravad jällegi nende loomingulisust. Probleemi leevendamiseks loodi antud lõputöö raames süsteem, mis suudab automaatselt hinnata programmeerimisalaste ülesannete graafilist väljundit, kasutades pildituvastust (image recognition). Süsteemi rakendatakse algajatele mõeldud programmeerimisalaste ülesannete puhul, mille lahenduseks on soovitud objekti graafilise väljundiga programmid. Lahendusprogrammist genereeritud pilti analüüsitakse pildituvastusega, mille tulemuseks on arv, mis näitab tõenäosust, et soovitud objekt oleks pildil. Esitus on arvestatud või mittearvestatud vastavalt eelmainitud tõenäosusele. Valminud süsteemi testiti MOOCi peal, täpsemini 2272 esitatud lahenduse peal. 4.6% tulemustest olid valenegatiivsed ning 0.5% tulemustest valepositiivsed. Kursuse jooksul läbi viidud vaheküsitlusest selgus, et 82.1% vastanute arvates töötas süsteem hästi või väga hästi ning keskmine hinnang süsteemile 5 palli skaalal oli 4.4.

Võtmesõnad:

Automaatne hindamine, automaatkontrollid, MOOC, programmeerimine, pildituvastus, arvutigraafika

CERCS: P175 - Informaatika, süsteemiteooria

Automatic assessment of programming assignments using image recognition

Abstract:

Automatic assessment of programming tasks in MOOCs (Massive Open Online Courses) is essential due to the large number of submissions. However, this often limits the scope of the assignments since task requirements must be strict for the solutions to be automatically gradable, reducing the opportunity for solutions to be creative. In order to alleviate this problem, we introduce a system capable of assessing the graphical output of a solution

program using image recognition. This idea is applied to introductory computer graphics programming tasks which solutions are programs that produce images of a given object on the screen. The image produced by the solution program is analysed using image recognition, resulting in a probability of a given object appearing in the image. The solution is accepted or rejected based on this score. The system was tested in a MOOC on 2,272 solution submissions. The results contained 4.6% cases of false negative and 0.5% cases of false positive grades. A participant survey revealed that the system was perceived to be functioning well or very well by 82.1% of the respondents, with an average rating of 4.4 out of 5.

Keywords:

Automatic assessment, automatic grading, MOOC, programming, image recognition, computer graphics

CERCS: P175 - Informatics, systems theory

Sisukord

Sissejuhatus

Programmeerimisalased vaba juurdepääsuga e-kursused ehk MOOCid (*Massive Open Online Course*) on teiste MOOCide kõrval väga populaarseks saanud. Programmeerimise kursuse keskmes on ülesanded. Oluliseks komponendiks MOOCide ülesannete juures on automaatsed kontrollid, kuna käsitsi hindamine ei ole suure hulga osalejate puhul võimalik. Programmeerimisülesandeid on võimalik automaatselt hinnata, kontrollides programmi koodi ja/või selle väljundit. Enamjaolt on hinnatav väljund teksti kujul. MOOCis „Programmeerimise alused“ on enamik ülesannetest tekstilise väljundiga, kuid on ka mõned ülesanded, mille puhul lahendusprogramm genereerib graafilise väljundi ekraanile. Väljundis on nõutud konkreetsed objektid, näiteks lipp või maja. Varasematel toimumiskordadel kontrolliti graafilise väljundiga ülesannete lahendusi manuaalselt, mis oli aeganõudev töö. Antud probleemi lahenduseks loodi käesoleva lõputöö raames süsteem, mis suudab automaatselt hinnata graafilise väljundiga ülesannete lahendusi.

Käesoleva lõputöö eesmärgiks on luua ning kirjeldada eelpool mainitud süsteemi. Loodud süsteem võtab sisendiks kursusel osaleja esitatud lahenduse ning saadab selle virtuaalmasina liivakasti (*sandbox*) ehk kohta, kus on võimalik ohutult rakendusi teistest eraldatuna käivitada. Liivakastis genereeritakse esitatud lahenduse koodist pildifail. Süsteem saadab seejärel pildi internetis asuvale pildituvastusteenusele, mis tagastab tõenäosuse, et soovitud objekt (näiteks maja) eksisteerib pildil. Esitatud lahendus saab hindeks arvestatud või mittearvestatud vastavalt eelnimetatud tõenäosusele. Lisaks hindab lõputöö loodud süsteemi kasulikkust ning summeerib osalejatelt saadud tagasiside peamised aspektid.

Lõputöö koosneb seitsmest peatükist. Esimeses peatükis antakse ülevaade MOOCide ülesannete hindamisest. Tuuakse välja erinevad hindamismeetodid ning nende eelised ja puudused. Teises peatükis tutvustatakse Tartu Ülikooli arvutiteaduse instituudis toimuvaid MOOCe ning kirjeldatakse detailselt neis esinevaid graafilise väljundiga ülesandeid. Kolmandas peatükis võrreldakse pildituvastusteenuseid ning tutvustatakse kriteeriume, mida võeti sobivaima teenuse valimisel arvesse. Neljandas peatükis antakse ülevaade protsessist, mille käigus lõputöö autor kontrollis käsitsi kõiki graafilise väljundiga esitatud lahendusi. Viiendas peatükis esitatakse statistika automaatselt hinnatud lahenduste kohta ning analüüsitakse kursusel osalejate hulgas läbiviidud tagasisideküsitluse tulemusi. Kuuendas peatükis kirjeldatakse olulisemaid tehnoloogiaid, mida antud lõputöö käigus valminud süsteemi loomisel kasutati ning lisaks antakse detailne kirjeldus loodud süsteemi

kohta. Seitsmendas peatükis analüüsitakse valminud süsteemi potentsiaalseid riske ning nende võimaliku kahju suurust. Käesoleva magistritöö lõpetuseks on kokkuvõte.

1. MOOCide ülesannete hindamisest

Koostöö juhendaja ja juhendatava vahel ning esitatud ülesannetele vastavuses olev tagasiside on peamised tegurid, mis toetavad õppimisprotsessi. Hea tagasiside olulisust ei saa kunagi üle tähtsustada. Algajad programmeerijad vajavad täpset ning personaalset tagasisidet, et aidata vigu mõista ning lahendusi paremaks muuta. Tänu mõistlikule tagasisidele on õppijatel võimalik oma vigu analüüsida ning seeläbi areneda [1].

Antud peatükis antakse ülevaade peamistest hindamismeetoditest, mida kasutatakse erinevate MOOCide ülesannete lahenduste hindamisel. Esile tuuakse osalejate omavaheline tööde hindamine ning automaatkontrollid.

1.1 Osalejate omavaheline tööde hindamine

Esimene MOOCide ülesannete hindamisel ning lahenduste kontrollimisel kasutatav moodus on kursusel osalejate omavaheline tööde kontrollimine (*peer assessment* / *self-assessment*). Seda hindamissüsteemi rakendatakse tihti juhtudel, mil ülesannet saab lahendada mitut moodi ning ei ole kindlat õiget ega vale vastust, näiteks esseede puhul. Majanduslikust vaatepunktist on see kursuse läbiviijatele kasulik, kuna aitab vähendada kursuse korraldajate töömahtu. Lisaks võidavad sellest ka kursusel osalejad, kuna teiste inimeste lahendusi kontrollides kinnistub õpitud materjal paremini. Käesoleva magistritöö raames käsitletud kursustel pole seni eelmainitud hindamisviisi kasutatud [2].

Antud hindamisviisil on ka omad negatiivsed küljed: hindamisprotsess pole piisavalt läbipaistev; kursusel osalejate tagasiside ei pruugi olla täpne ning usaldusväärne; on oht, et kõik tööd ei saa hinnatud ja kursusel osalejad ei usalda kaasosalejate teadmisi ning arusaama materjalist. On äärmiselt oluline, et oleks tagatud hindajate ning hinnatavate anonüümsus, kuna vastasel korral võivad inimsuhted hindamist mõjutada ja kaaslastele ei taheta negatiivset tagasisidet anda. Lisaks ei saa kindel olla, et kõik kursusel osalejad on kursis plagieerimise kontrollimisega. Üheks võimalikuks lahenduseks oleks see, et konkreetset esitatud lahendust hindab mitu inimest ning ainult juhul, kui antud tagasisides on suured erinevused, vaatab esitatud lahenduse üle ka kursuse läbiviija [3]. Selline hindamismeetod peaks moodustama kogu kursuse hindest vaid väikese osa, et olla kindel kursuse läbijate hinde ning teadmiste „õigsuses“ [4]. Antud lõputööga seotud kursustel osalejate omavahelist tööde hindamist ei kasutatud.

1.2 Automaatkontrollid

Teiseks hindamistüübiks on automaatkontrollid. Vreda Pieterse Pretoria Ülikoolist on väitnud, et kuna MOOCide puhul puudub kontakt osalejate ja korraldajate vahel, siis peaks automaatkontroll olema võimalikult kvaliteetne, et asendada traditsioonilist kontaktki juhendajaga. Traditsioonilisel juhul on võimalik juhendajal anda lisakommentaare ülesande kohta ning privaatset tagasisidet. Enamike MOOCide puhul puudub igasugune suhtlus kursusel osalejate ja korraldajate vahel. On oluline, et automaatkontrolle kasutades oleks osalejatel võimalik ülesandeid esitada piiramatu arv kordi, kuna siis on neil võimalik oma esitust vastavalt tagasisidele parandada ning täiustada, mis aitab omakorda õpitud materjali kinnistada. Piiramatu arvu lahenduste juures võib tekkida oht, et kursusel osalejad kasutavad tagasisidet ning automaatkontrolli justkui testi enda lahendusele ning täiustavad oma lahendust vaid selleks, et automaatkontroll nende töö ära hindaks. MOOCide puhul aga ei oma see nii tähtsat rolli, kuna eeldatavasti võtavad sellest osa inimesed, kes soovivad õppida ning end arendada.

On väga oluline, et automaatkontrolli tagasiside oleks asjakohane ning abistav. Selle tagamiseks on mitmeid lisavõimalusi: kirjutada juhend, kuidas automaatkontrolli tulemust interpreteerida ning tänu sellele oma esitust parandada; pakkuda õppijatele foorumi võimalust, et nad saaksid oma muresid jagada ning üksteist aidata; pidevalt uuendada ning täiendada automaatkontrolli ennast parema kvaliteedi ning tagasiside eesmärgil. Korraliku automaatkontrolli koostamine on keerukas protsess ning tuleks meeles pidada, et iga viga kontrollis võib tekitada probleeme. Ühe korraliku automaatkontrolli koostamine on vähemalt sama mahukas kui heade valikvastustega küsimuste väljamõtlemine. Võib isegi tekkida olukord, kus aeg, mis kulus varem käsitsi ülesannete kontrollimisele, kulub nüüd tervenisti automaatkontrollide koostamisele. Ajavõit automaatkontrollide kasutamise juures hakkab tekkima siis, kui samu ülesandeid ning automaatteste kasutatakse korduvalt erinevatel aastatel või erinevatel kursustel [1]. Enamasti kontrollitakse esitatud lahendusprogrammi õigsust selle käivitamisel. Nimelt võrreldakse kursusel osaleja programmi väljundit kursuse läbiviija koostatud lahendusprogrammi väljundiga. Lisaks ülesande õigsuse kontrollimisele on võimalik veel kontrollida lahenduse keerukust, stiili, disaini ning efektiivsust [5].

1.3 Graafilise lahendusega ülesannete automaatkontrollid

Tehniliste ülesannete puhul (nt programmeerimine) on ülesande kirjeldus ülimalt tähtis, et automaatkontrollid töötaks. Kui ülesande kirjeldusse jäävad sisse ebamäärasused, siis on võimalik kursusel osalejatel seda mitut moodi tõlgendada. See omakorda tähendab, et ülesandel võib olla erinevaid lahendusi, kuid automaatkontrollid aktsepteerivad enamasti vaid üht korrektset lahendust [6].

Kursusel osalejate üldine huvi on aga ühe korrektse lahendusviisiga vastuolus – nimelt on programmeerimisalastel kursustel osalejatel suurem huvi graafiliste ülesannete vastu [7], aga just graafilise väljundiga ülesandeid on kõige raskem automaatselt hinnata. Graafiliste ülesannete automaatkontrollide koostamine käib enamasti nii kursuse korraldajatel kui ka kursusel osalejatel üle jõu [8]. Siiski võib välja tuua ühe toimiva lahenduse graafiliste kasutajaliideste automaatseks hindamiseks, mis loodi Brightoni ülikoolis, kasutades JEWL nimelist raamistikku. Nimetatud süsteem võimaldab automaatselt hinnata programmeerimiskeeles Java kirjutatud programme, millel on graafiline kasutajaliides. JEWL on spetsiaalne graafiliste kasutajaliideste raamistik, mis toetab üheaegselt nii programmeerimist kui ka automaatset hindamist. JEWL asendab graafilise kasutajaliidese testraamistikuga, mis interpreteerib testitava programmi instruktsioone. Antud raamistikku kasutades on võimalik kogu kasutajaliidese funktsionaalsust testida, kuid kahjuks pole veel suudetud leida moodust kontrollimaks spetsiifiliste graafiliste objektide olemasolu kasutajaliidesel [7].

Olgugi et viimase kümne aasta jooksul on loodud palju automaatkontrollisüsteeme ning automaatseid tagasisidesüsteeme, toetamaks erinevaid arvutiteaduse kursusi, siis ühtegi koheselt kasutamisvalmis süsteemi graafilise väljundiga programmeerimisülesannete automaatseks kontrollimiseks ei leidu [9]. Brightoni ülikooli kogemust järeldub, et piisava vajaduse, huvi ning oskuste korral on automaatse hindamissüsteemi loomine võimalik [7]. Antud lõputöö käigus valmis süsteem, mis suudab automaatselt hinnata graafilise väljundiga ülesannete lahendusi. Loodud süsteemi on täpsemalt kirjeldatud kuuendas peatükis.

2. Kursustest ning graafilise väljundiga ülesannetest

Käesolev peatükk tutvustab MOOCi „Programmeerimise alused“ ning graafilise väljundiga programmeerimise ülesandeid, mida antud kursusel lahendada tuleb. Programmerimist sissejuhatavaid MOOCe on Tartu Ülikooli arvutiteaduse instituudis korraldatud alates 2014. aastast. Arvutiteaduse instituudis toimub kolm eestikeelset, peamiselt täiskasvanutele mõeldud programmeerimisalast MOOCi: 4-nädalane kursus „Programmeerimisest maalähedaselt“, 8-nädalane kursus „Programmeerimise alused“ ning 8-nädalane kursus „Programmeerimise alused 2“. Viimase aasta jooksul on kursusel „Programmeerimise alused“ osalenud 3835 inimest. Kursus on kokku toimunud neli korda. Antud lõputöös on vaatluse all kursused, mis toimusid 2016. aasta märtsist maini ning 2017. aasta jaanuarist märtsini.

Enamasti tuleb igal nädalal lahendada neli ülesannet ning iganädalane test. Kursuse neljandal nädalal tutvustatakse osalejatele graafika teemat. Graafika teema juures on kursusel osalejatel võimalik valida kolme ülesande vahel, millest lahendada tuleb vähemalt üks, mis tähendab tulemuseks „arvestatud“ saamist. Kõigi ülesannete raames tuleb kursusel osalejatel joonistada pilt kasutades Pythoni teeki nimega Tkinter [10]. Võimalik on valida kolme erineva ülesande vahel: joonistada (a) lipp, (b) liiklusmärk või (c) maja.

Esimene ülesanne lahendamiseks tuleb koostada programm, mis joonistab ekraanile Eesti haldusüksuse lipu. Lipp peab sisaldama vähemalt kolme erinevat värvi või koosnema mingist keerulisest kujundist (vt Joonis 1). On soovituslik valida lipp, kus on midagi korduvat, tsüklilist, et programmeerimisel tsükleid kasutada.

Joonis 1. Kursusel osaleja lahendus lipu ülesandele aastast 2016.

Teise ülesande sooritamiseks tuleb koostada programm, mis joonistab ekraanile liiklusmärgi. Liiklusmärgi valiku osas piiranguid pole. Lisaks ei ole ka ühtegi kohustuslikku elementi, mis peab liiklusmärgil eksisteerima (vt Joonis 2). Ainsaks soovituseks on valida liiklusmärk, milles esineks kordusi ning kus tsüklite kasutamine oleks asjakohane.

Joonis 2. Kursusel osaleja lahendus liiklusmärgi ülesandele aastast 2016.

Kolmanda ülesande lahendamiseks tuleb koostada programm, mis joonistab ekraanile maja. Maja peab koosnema vähemalt kolmest elemendist. Elementide valiku osas piiranguid pole. Mõned näited vabalt valitud elementidest on järgnevad: uks, aken, katus ja korsten (vt Joonis 3).

Joonis 3. Kursusel osaleja lahendus maja ülesandele aastast 2017.

Vajadus graafilise väljundiga ülesannete automaatkontrollidele tuli aastal 2016, kui käsitsi tuli hinnata üle 1200 esitatud lahenduse. See võttis suure hulga kursuse korraldajate ajast. 2016. aastal esitati kõik lahendused, sealhulgas nii kood kui pilt graafilisest väljundist, spetsiaalsesse foorumisse, kus kursusel osalejad said pärast lahenduse esitamist üksteise töid näha. Sellise foorumi kasutamisel esinesid mõned probleemid: kõik kursusel osalejad ei soovinud oma lahendust teistega jagada ning foorum muutus väga aeglaseks pärast seda, kui sajad inimesed olid oma lahenduse sinna esitanud.

3. Pildituvastamise teenust pakkuvate teekide võrdlus

Käesolev peatükk tutvustab pildituvastamise teenust pakkuvaid teeke, selgitab nende eeliseid ning puuduseid ning annab neist võrdleva ülevaate. Selleks, et leida parim võimalik lahendus graafiliste ülesannete automaatseks kontrolliks, tuli kõigepealt mõelda potentsiaalsetele lahendustele, milleks olid kas a) ise pildituvastamise algoritmi kirjutamine või b) olemasoleva pildituvastamise teenuse kasutamine.

Piltidelt objektide tuvastamine on üheks tavalisemaks probleemiks piltidega tegelevate süsteemide juures. Objekte aitavad defineerida nende kuju, värv ning struktuur. Tuvastamist kasutatakse erinevate valdkondade juures ning tuvastada on võimalik näiteks järgmisi asju: sõrmejälgi, nägusid, käsikirja ning kõiksugu konkreetseid objekte. Selleks, et panna süsteemi automaatselt visuaalseid objekte ära tundma, on tarvis algoritme, mille alusel pildituvastus toimib [11].

Pildituvastamise algoritmi kirjutamise teeb keeruliseks asjaolu, et seda tuleb oluliselt vastavalt ülesandele muuta. Seega otsustas autor uurida, kas pildituvastust pakutakse ka teenusena, kuna selliselt oleks võimalik kursusel osalejatele mõeldud ülesandeid kergelt muuta ilma automaatkontrolli ennast oluliselt muutmata. Teekide võrdlemiseks valis autor välja enamlevinud teenusepakkujad pildituvastamise valdkonnas: Clarifai, Google Cloud Vision ja Imagga. Just need kolm teenusepakkujat andsid otsingumootorites enim vasteid ning ei vajanud katsetamiseks rahalist investeeringut.

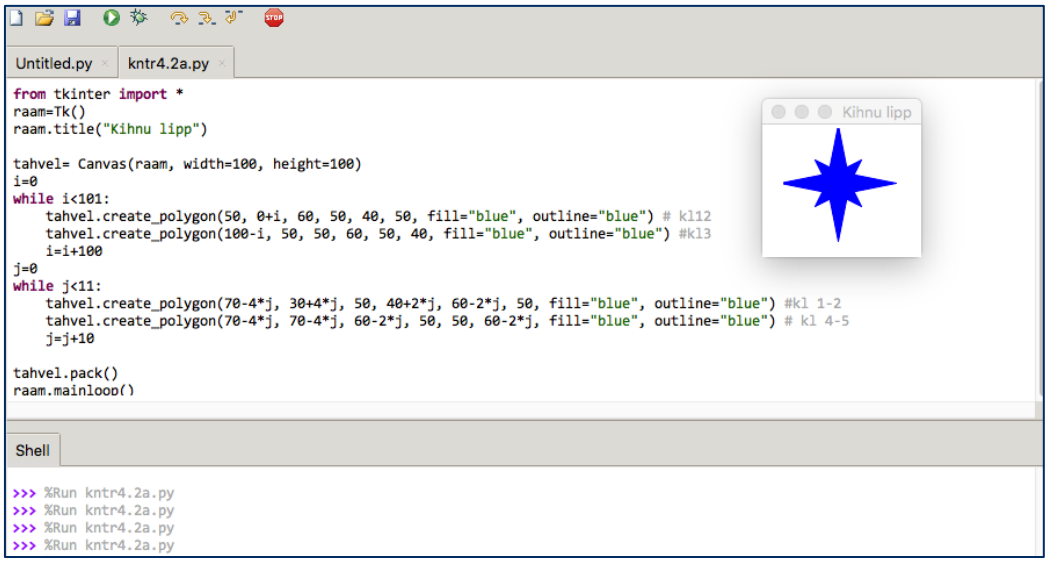
3.1 Sobiva teenusepakkuja valimise kriteeriumid

Selleks, et sobivaim teenusepakkuja valida, oli tarvis välja selgitada teatud parameetrid, mis olid antud lõputöös valminud lahenduse jaoks olulised. Esimeseks parameetriks oli teenuse kiirus ehk üheks päringuks keskmiselt kuluv aeg. Ühte päringusse kuulub üks pilt, selle saatmine teenusepakkujale ning teenusepakkujalt vastuse saamine. Teiseks parameetriks oli teenuse hind, täpsemalt kui kallis on iga tuhande või viie tuhande päringu tegemine. Kolmanda parameetrina hindas autor teenusepakkuja dokumentatsiooni ning rakendusliidest. Hinnati dokumentatsiooni arusaadavust, korrektsust ning teenuse ülesseadmise lihtsust. Lisaks arvestati asjaolu, kas teenusepakkujal oli kasutajatele loodud näidisrakendus näidiskoodiga, mis aitaks kergemini teenust kasutada ning seda paremini mõista. Ühe tegurina hindas autor kasutajatuge: kui kiiresti reageeriti/vastati, kui huvitatud koostööst/abistamisest oldi ning kui hästi oldi kursis enda toote ja selle võimalustega.

Viimane ning antud lõputöös valminud rakenduse jaoks kõige olulisem faktor oli see, kas uuritav teenusepakkuja võimaldab anda tagasisidet konkreetse(te) märksõna(de) kohta, mis teenuse kasutaja üleslaetava pildiga kaasa annab. See tähendas seda, et teenusepakkujale ei saadeta lihtsalt pilti, et sealt tuvastada erinevaid objekte, vaid et teenusepakkujale saadetakse pilt koos konkreetse märksõnaga objektist, mille olemasolu kohta pildil tagasisidet soovitakse.

3.2 Teenuste testimiseks kasutatud testvalim

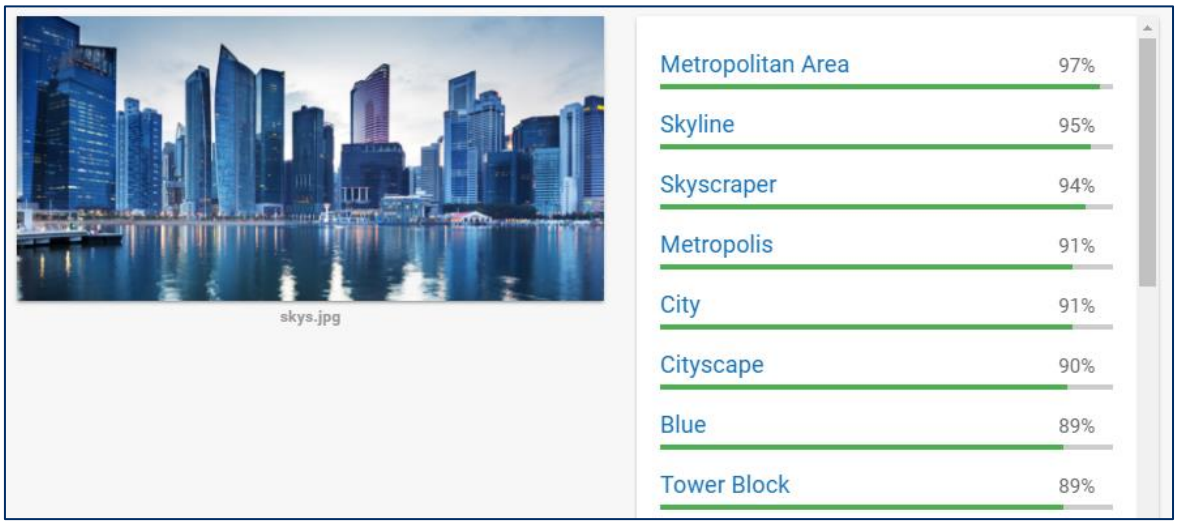
Pärast parameetrite valimist asuti teenuseid katsetama ja võrdlema. Kuna kursuste läbiviijad otsustasid, et suure tõenäosusega kasutatakse antud kursustel ka sel korral samu pildiülesandeid, siis tundus kõige mõistlikum katsetada teenuseid kolme ülesandega, mida eelmisel korral kasutati (Programmeerimise alused (MTAT.TK.012). Testi kuulus 260 liiklusmärgi, 766 lipu ning 150 muud pilti, mille kursusel osalejad olid esitanud, kusjuures muude piltide alla kuulusid enamasti mingi seadme ekraani või kasutajaliidese pildid (nt pesumasina ekraan/kasutajaliides). Lisaks on oluline, et 766 lipu ning 150 muud pilti esitati teenusepakkujatele üheskoos, et avastada võimalikke valepositiivseid tulemusi. Sinna alla kuulusid näiteks pildid, millel tuvastati lipp, kuid tegelikult seda seal kujutatud polnud. Lisaks tuleks kindlasti arvesse võtta asjaolu, et eelmisel aastal tuli ülesannete lahendused laadida foorumisse ning tihtipeale esitasid kursusel osalejad pildi tervest ekraanist, mitte ainult lahendusest, mis raskendab teenusepakkujatel pildilt soovitud objekti tuvastamist (vt joonis 4).



Joonis 4. Pilt kursusel osaleja lahendusest, millel on lisaks graafilisele lahendusele ka kood.

3.3 Google Cloud Vision

Esimeseks teenusepakkujaks valis autor Google Cloud Vision'i. Google Cloud Vision API võimaldab arendajatel mõista pildi sisu, kasutades võimsaid masinõppe mudeleid. Neid mudeleid on võimalik kasutada läbi lihtsate ning mugavate rakendusliideste. Teenus suudab kiiresti tuvastada piltidelt objekte ning jagada neid tuhandetesse kategooriatesse. Lisaks suudab Google Cloud Vision eristada piltidel individuaalseid objekte, nägusid ning võimaldab tuvastada piltidel paiknevaid tähti ja sõnu.



Joonis 5. Google Cloud Vision demo rakendus demonstreerimas pildituvastust.

Vision API teenus paraneb pidevalt, tuvastades üha enam pilte ning uute kategooriate lisamisega paraneb pildituvastamise täpsus. Tänu Google SafeSearch'ile võimaldab Vision API tuvastavada ebasobivaid pilte. Võimalik on ise valida, millist tüüpi pildimaterjali soovitakse eemaldada – näiteks vägivaldseid või pornograafilisi pilte. Kuna Vision API suudab tuvastada inimeste nägudelt erinevaid emotsioone, siis kombineerides seda tavalise pildituvastusega on võimalik teada saada, mida inimene ühest või teisest asjast arvab [12]. Suhtlus kliendi ning teenusepakkuja vahel toimub läbi rakendusliidese, mis tähendab, et tulemused saadetakse kliendile läbi JSON formaadi. Vastuses on märksõnad pildilt leitud objektidest ning nende pildil eksisteerimise tõenäosus, et teenuse kasutaja saaks etteantud pildi või piltide kohta lihtsalt tagasisidet [13].

3.3.1 Cloud Visioni testimise tulemused

Alustuseks testiti teenust 260 pildiga, millel oli kujutatud liiklusmärki. Teenus suutis tuvastada 26 pildilt märgistuse (*„signage"*), 24 pildilt liiklusmärgi (*„traffic sign"*) ning 6

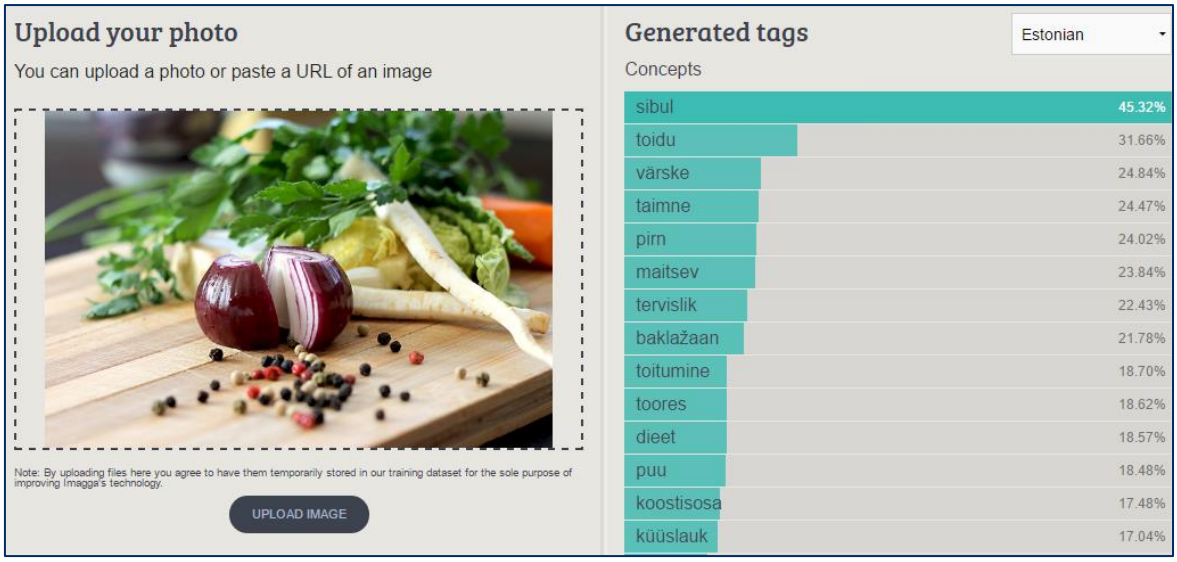
pildilt lipu (*„flag“*). Huvitaval kombel tuvastati rohkem aga järgmisi märksõnu: 71 korral logo, 232 korral kujundit (*„shape“*), 235 korral brändi (*„brand“*) ning kümneid teisi märksõnu. Tulemused (tõenäosused) varieerusid vahemikus 0,5 (miinimum) kuni 0,95 (kõrgeim tulemus). Kuna soovitud märksõnu suudeti tuvastada liiga vähestel kordadel (mille tõenäosus oli suurem kui 0,5), siis ei saa tulemusi piisavaks lugeda.

Teise testi kuulus 600 lipu pilti. Selle testi tulemused andsid huvitava tulemuse. Nimelt tuvastati 260 pildilt märksõna värv (*„color“*) ning paljudelt piltidelt eraldi ka konkreetne värv (nt *„yellow“*). Märksõna lipp (*„flag“*) suudeti kahjuks tuvastada vaid 110 pildil. See-eest märksõna kujund (*„shape“*) esines koguni 494 pildil. Ka lipu piltidelt leiti väga tihti brände (*„brand“*) – koguni 511 korral. Ka siin varieerusid tõenäosused suures vahemikus: 0,5 kuni 0,96.

Kokkuvõtvalt võib öelda, et teenus ei suutnud enamasti piltidelt soovitud objekte tuvastada. Tihti leiti piltidelt sarnaseid objekte, kuid mitte otseselt neid, mida sooviti leida. Positiivse poole pealt võib välja tuua teenuse kiiruse, milleks oli 1,43 sekundit päringu kohta.

3.4 Imagga

Teiseks teenusepakkujaks valis autor Imagga. Imagga on pildituvastamise platvormi teenus üle interneti (PaaS - *Platform as a Service*), mis tähendab, et operatsioonisüsteem ning raamistik on teenusepakkuja poolt kindlaks määratud. Klient saab valida rakenduse, mida soovib etteantud raamistiku sees jooksutada [14]. Teenus pakub ettevõtetele ning arendajatele piltide sildistamise rakendus- ehk programmiliideseid (API), mis võimaldavad neil luua võimsaid rohkete piltidega rakendusi pilves. Nende automaatse sildistamise (*auto-tagging*) tehnoloogia töötab kombineerituna süvaõppest ning konvolutsioonilistest võrkudest, baseerudes väga suurel hulgal etteantud fotodel.



Joonis 6. Imagga veebirakenduse demo lehekülg demonstreerimas pildituvastust.

Jooniselt 6 on näha, et Imagga võimaldab kasutajal valida tuvastatud märksõnade keelt – sellist funktsionaalsust teised teenusepakkujad ei võimaldanud. Automaatse sildistamise teenus suudab koheselt ära tunda üle 2700 objekti ning lisaks tagastada üle 20 000 abstraktse mõiste etteantud pildi kohta [15]. Imagga teenuste hulka kuuluvad piltide automaatne sildistamine, automaatne sorteerimine (*auto-categorization*), automaatne värvide eraldamine (*color extraction*) ja piltide kärpimine (*smart cropping*). Tänu piltide automaatse skaleerimise võimalusele (*auto-scaling*) suudab Imagga teenus väga paljusid päringuid korraga hallata. Nende rakendusliideseid on võimalik kasutada nii eraldi kui ka kombineerituna [16].

3.4.1 Imagga testimise tulemused

Alustuseks testiti teenust 260 pildiga, millel oli kujutatud liiklusmärki. Teenus suutis tuvastada kõigilt antud piltidelt järgmised märksõnad: „liiklusmärk“ ning „sümbol“. 256 pildilt leidis teenus märksõnale „ikoon“ vastavaid tunnuseid. Lisaks leidis teenus piltidelt veel kümneid segadusse ajavaid erinevaid märksõnu. Autori muutis ebakindlaks asjaolu, et kõik tuvastatud märksõnad leidis teenus piltidelt tõenäosusega vahemikus 0,05 kuni 0,45, mida ei saanud rahuldavaks tulemuseks lugeda.

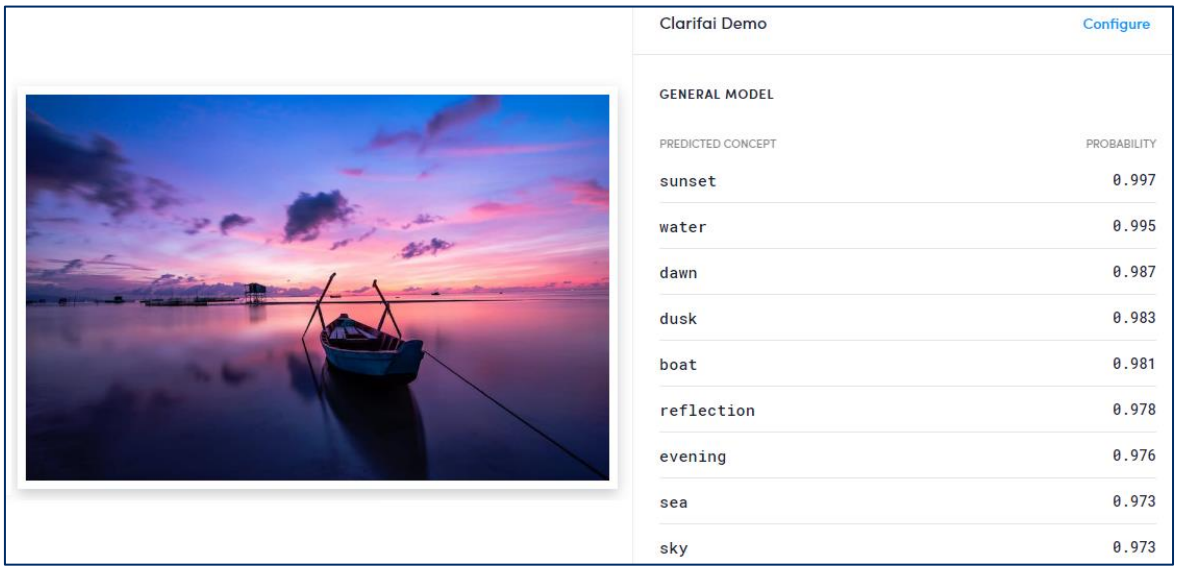
Teise testi kuulus 915 pilti, millest 765 olid lipu pildid ning 150 muud pilti (vt peatükk 3.2). Selle testi tulemused andsid selgema tulemuse. Nimelt 765 lipu pildilt suutis teenus tuvastada vaid 271 lippu (0,35%). See-eest 870 pildilt tuvastati (liiklus)märk (*„sign“*), 858

pildilt sümbol ning 856 pildilt ikoon. Lisaks mainitud objektidele leiti piltidelt veel kümneid märksõnu. Kõigi tuvastatud märksõnade tõenäosus jäi vahemiku 0,05 kuni 0,6.

Kokkuvõtvalt võib öelda, et teenus suutis pildilt tuvastada soovitud objekte, kuid lisaks neile veel kümneid ebavajalikke objekte. Probleemiks oli see, et ebavajalikke objekte ei olnud võimalik soovitud objektidest piisavalt selgelt eristada. Kõigi tuvastatud objektide omavahelised tõenäosused olid liialt sarnased ning üldiselt madalad. Teenuse keskmiseks kiiruseks oli 2,9 sekundit päringu kohta.

3.5 Clarifai

Kolmandaks teenusepakkujaks valis autor Clarifai. Tegemist on tehisintellektiga tegeleva firmaga, mis on edukas just visuaalse tuvastamise (*visual recognition*) valdkonnas. Clarifai lahendab ettevõtete ning arendajate igapäevasituatsioonidest tulenevaid probleeme ning pakuvad neile vastavaid lahendusi, kasutades keerulisi masinõppe süsteeme. Täpsemalt kasutab Clarifai digitaalsetelt piltidelt objektide äratundmiseks konvolutsioonilist võrku [17].



Joonis 7. Clarifai näidisrakendus demonstreerimas pildituvastust.

Clarifai on olnud turuliider visuaalse äratundmise valdkonnas alates 2013. aastast, mil nad võitsid ImageNet konkursil pildituvastamise kategoorias viis esimest kohta. Lisaks pildituvastusele pakutakse ka videotuvastamise teenust ning nende süsteemi on võimalik kasutada läbi rakendusliideste. Rakendusliidese idee on lihtne – klient saadab pildi(d) süsteemile ning süsteem tagastab tulemused. Tulemuseks on pildilt tuvastatud objektid ning tõenäosused iga objekti kohta, et just see objekt leiti pildilt [18].

3.5.1 Clarifai testimise tulemused

Kuna Clarifai võimaldas kontrollida konkreetse märksõna olemasolu etteantud pildilt, siis oli võimalik saada täpsemaid tulemusi teenuse kohta (vt tabel 1). Tabeli vasakpoolses osas on kujutatud tulemused lipu ning muude piltide kohta. Clarifaile saadeti ükshaaval kõik pildid ning paluti neil pildilt ära tunde lipp (*„flag“*) ning anda selle pildil eksisteerimise tõenäosus. Tulemusena leidis Clarifai 566 pildilt lipu tõenäosusega, mis on suurem kui 0,9. See moodustab 61,9 protsenti kõigist selle kategooria esitatud töödest ning 73,9 protsenti kõigist esitatud lipu piltidest. Huvitav on veel märkida, et 146 pilti said tõenäosuseks alla 0,1 protsendi. Need 146 pilti moodustavad aga protsentuaalselt huvitaval kombel 97,3 protsenti piltidest, mis ei olnudki lipu kujutisega.

766 lipu ning 150 muud pilti		260 liiklusmärgi pilti	
Tulemus (*„flag“*)	Jaotus	Tulemus (*„sign“*)	Jaotus
>0.9%	566 (61,9%)	>0.9%	173 (66,5%)
>0.8%	44	>0.8%	44
>0.7%	24	>0.7%	16
>0.6%	17	>0.6%	11
>0.5%	19	>0.5%	11
>0.4%	24	>0.4%	1
>0.3%	23	>0.3%	4
>0.2%	17	<0.3%	0
>0.1%	35		
<0.1%	146 (15,9%)		
Kiirus	1.7 sek / päring (keskmiselt)	Kiirus	1.7 sek / päring (keskmiselt)

Tabel 1. Teenusepakkujate omavaheline võrdlus.

Tabeli paremal pool on võimalik näha ülevaadet tulemustest, mis saadi liiklusmärgi piltide kontrollimisel kasutades Clarifai teenust. Kõik 260 liiklusmärgi pilti saadeti teenusele ning kaasa anti liiklusmärgi (*„sign“*) märksõna. Positiivse tulemusena tagastas Clarifai 173 (66,5%) pildile tõenäosuseks rohkem kui 0,9. Koguni 233 (89,6%) pilti said tulemuseks rohkem kui 0,7. Huvitav on siinjuures asjaolu, et mitte ükski liiklusmärgi pilt ei saanud tulemuseks vähem kui 0,3. Clarifai keskmiseks päringu kiiruseks oli 1,7 sekundit päringu kohta, mis on küll veidi aeglasem kui Google Cloud Visionil, kuid oluliselt kiirem kui Imaggal.

3.6 Teenusepakkujate omavaheline võrdlus

Selleks, et valida parim teenusepakkuja, oli neid tarvis omavahel võrrelda. Võrdlusesse kuulusid mitmed omadused (vt Tabel 2). Esimeseks võrreldavaks omaduseks oli tasuta päringute arv ühes kuus. Kuna oli teada, et varasemalt oli antud kursusel osalenud üle 1000 inimese ning et lahendada oli võimalik kolme ülesannet, siis võis arvestada vähemalt 1000 päringuga. Lisaks oli tarvis jälgida järgnevate päringute hinda (kui tasuta päringud on kasutatud). Parimaks variandiks hinna osas kujunes selgelt Clarifai, kuna pakuvad igakuiselt suurimat tasuta päringute hulka ning maksta tuleb vaid iga järgneva tuhande päringu eest.

Töö autor otsustas ka teenusepakkujate rakendusliidest ning dokumentatsiooni hinnata. Kõige tülikam esialgne seadistus oli Google Cloud Visionil. Nende teenuse ülesseadmiseks oli kõigepealt tarvis nende veebilehel teha suur hulk seadistusi, mis tundusid näiliselt ebavajalikud ning nõudsid palju aega. Imagga puhul oli dokumentatsioon selge ning üheselt mõistetav – probleeme ei tekkinud. Clarifai teenuse dokumentatsioonis esines palju probleeme: seal oli olulistes kohtades trükivigu, loogikavigu ning lausa vigaseid näiteid.

Kõigil kolmel teenusepakkujal eksisteeris näidisrakendus teenuse paremaks tundmaõppimiseks ning lihtsaks katsetamiseks. Nii Google Cloud Visionil kui ka Imaggal olid mugavad näidisrakendused, mis olid lihtsasti kättesaadavad ning kergesti seadistatavad. Clarifai näidisrakendus oli väga aeglane ning see ei töötanud uuemate versioonidega tarkvarast Python. Seetõttu tuli autoril teenuse katsetamiseks luua isiklik katsetamise rakendus.

Tabel 2. Teenusepakkujate omavaheline võrdlus.

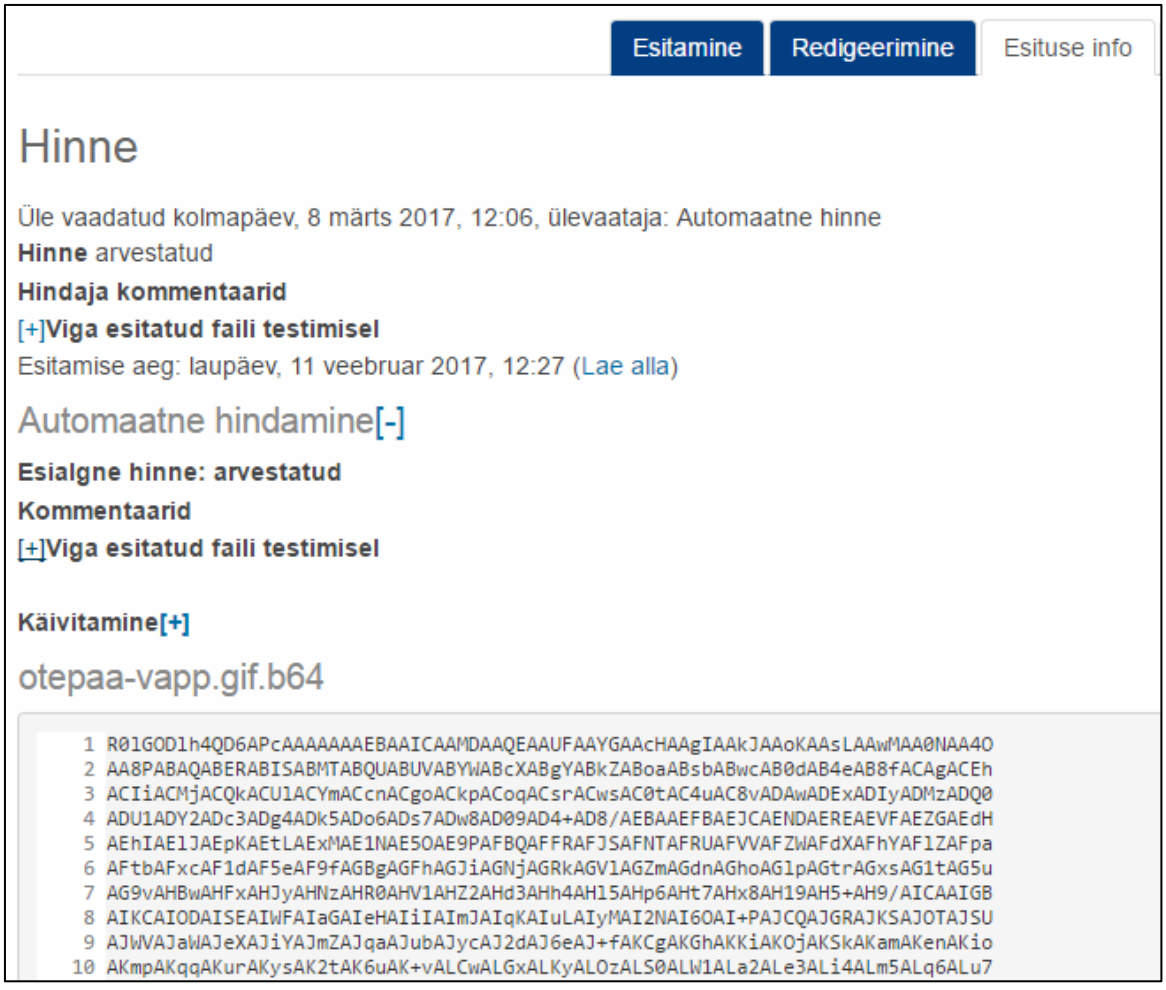
Teenusepakkuja/omadus	Google Cloud Vision	Imagga	Clarifai
Tasuta päringute arv	1000	2000	5000
Edasiste päringute hind	1001-1000000 päringut $1.50/kuu	12000 päringut / 14$ kuu 70000 päringut / 79$ kuu	1000 päringut / 1.20$
API/dokumentatsioon	Esialgne seadistus tülikas, palju näiliselt „ebavajalikke“ asju tuli konfigureerida	Korrektne ning lihtsasti mõistetav	Palju probleeme
Näidisrakendus	Näidiskood olemas, töötas hästi	Näidiskood olemas ning töötas hästi	Näidisrakendus vigane ning aeglane
Kasutajatugi	Vastati nädala jooksul, asjalik	Vastati nädala jooksul, asjalik	Ebapädev, ebastabiilne (kord kiire, kord aeglane)
Pildi analüüs konkreetse(te) märksõna(de) abil	Tugi puudub	Tugi puudub	Tugi olemas

Kuna autoril tuli ühendust võtta kõigi teenusepakkujate kasutajatugedega, siis otsustas ta neid ka võrrelda. Nii Google Cloud Visioni kui ka Imagga puhul oli võimalik konkreetsele küsimusele saada kiire ning konkreetne vastus nädala jooksul. Clarifai kasutajatugi vastas tihti ebatäpselt ning kaua aega (kuni kaks nädalat). Lisaks ei teadnud Clarifai kasutajatugi kohati enda toodet ning ei osanud anda pädevaid soovitusi. Esines olukordi, kus saadud abist polnud mingit kasu või see osutus valeks.

Viimaseks vaatluse all olnud omaduseks oli teenusepakkujate pildi analüüsi võimekus konkreetse(te) märksõna(de) abil (vt peatükk 3.1). Kuna mainitud omadus on antud lõputöö kontekstis kriitilise rolliga, siis ostuski valituks just nimelt Clarifai. Nende kiituseks võib veel mainida, et alates hetkest, mil erinevad probleemid lahenduse leidsid, töötas teenus kiirelt, stabiilselt ning veatult.

4. Esitatud lahenduste läbivaatus

Antud peatükis tutvustatakse lähemalt protsessi, mille käigus töö autor kontrollis käsitsi kursusel osalejate esitatud lahendusi. Autor jälgis jooksvalt esitatud töid nädalal, mil ülesande esitamine osalejatele avati ning aktuaalne oli. Selle nädala jooksul ei tulnud õpilaste ega ka täiskasvanute kursuselt välja ühtegi ülesannetega seotud probleemi ning ükski osaleja ei pöördunud isiklikult kursuse läbiviijate poole. Kuna esitatud töid oli kokku ligi 1500 ning tööde käsitsi läbivaatus toimus ükshaaval, siis algselt tundus mõistlik läbi vaadata ainult mingi hulk esitatud töödest – autor otsustas kontrollida iga kümnendat esitatud tööd. Olles läbi vaadanud umbes kakskümmend tööd, jäi autorile silma esimene viga. Nimelt osaleja, kes oli esitanud kohustusliku Pythoni faili asemel pildifaili, oli saanud hindeks „arvestatud“ ja see oli kriitiline viga.

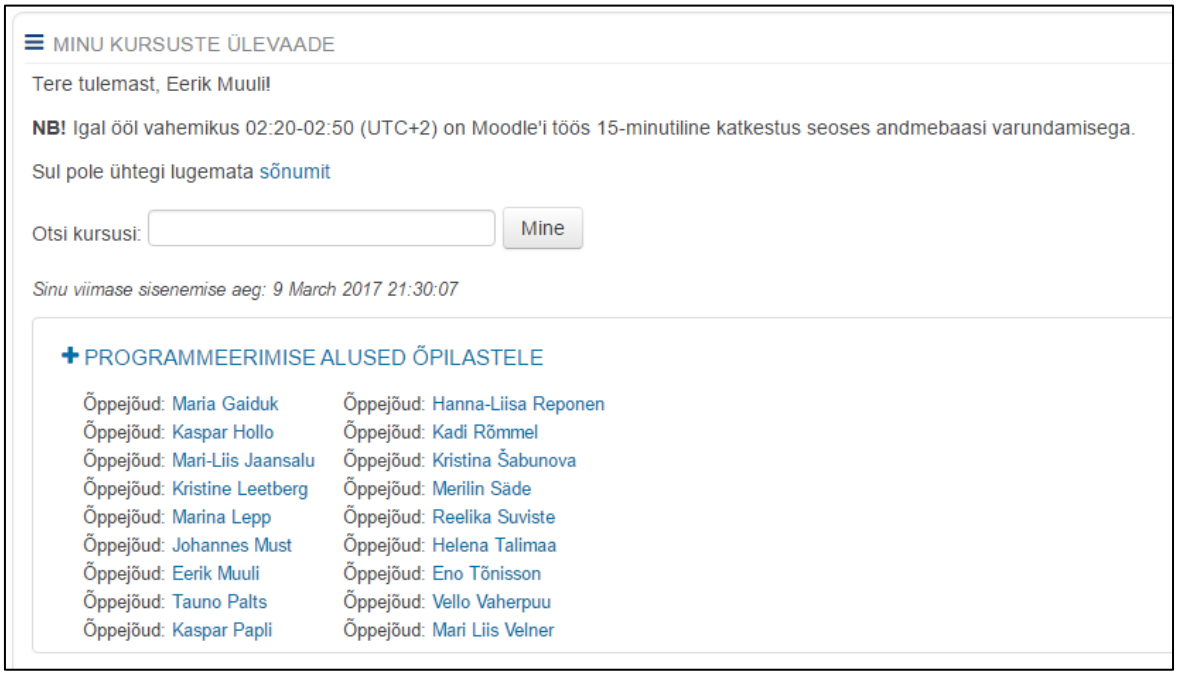


Joonis 8. Automaatkontroll annab vales formaadis esitatud tööle hindeks "arvestatud".

Pärast selle vea leidmist otsustas autor, et oleks tarvis kõik tööd kindluse mõttes üle vaadata. Tuli teha kindlaks, kas ja kui palju on veel selliseid kursustel osalejaid, kes on saanud

ülesannetele ebaõiglaselt positiivse tulemuse. Lisaks tähendas esimese vea leidmine seda, et hindamissüsteemis oli vähemalt üks viga ning autor märkis selle üles, et hiljem vea põhjust analüüsida ning see võimalusel tuleviku jaoks parandada.

Tööde läbivaatamise protsess toimus Moodle'i keskkonnas. Selleks logis autor esmalt õpihaldussüsteemi sisse aadressilt https://moodle.ut.ee/. Kuna käimasolevaid kursusi oli töö kirjutamise hetkel kaks, siis alustas autor tööde ülevaatust kursusest „Programmeerimise alused õpilastele".

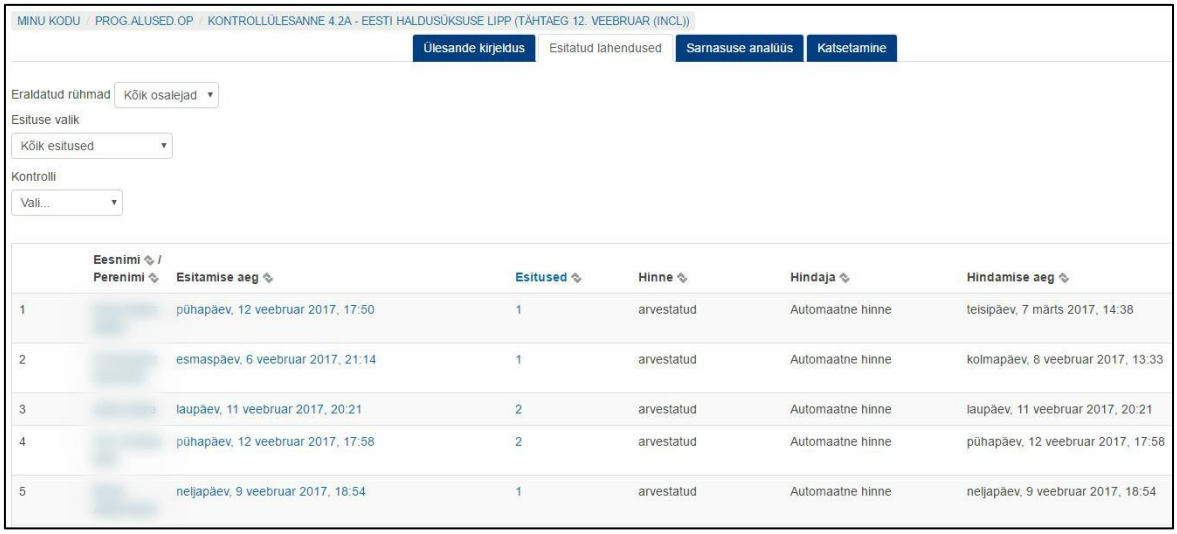


Joonis 9. Sobiva kursuse valimiseks on tarvis selle nimele pealehel olles vajutada.

Kursuse lehelt oli vaja minna vajalike ülesannete juurde. Ülesanded paiknevad kursusel nädalate kaupa, alates kõige varasemast. Graafilised ülesanded toimusid nii õpilaste kui ka täiskasvanute kursuse neljandal nädalal. Arvestuse saamiseks tuli neist positiivsele hindele lahendada vähemalt üks. Üle oli vaja vaadata kõik esitatud tööd, olenemata sellest, kas kursusel osalejal oli juba tulemus arvestatud või mitte.

Joonis 10. Graafilised ülesanded 4.2a, 4.2b ning 4.2c.

Töödele ligipääsemiseks tuli vajutada soovitud ülesandel. Autor alustas ülesandest 4.2a – Eesti haldusüksuse lipp. Järgnevalt valiti „Esitatud lahendused“, mille all olid reas kõik valitud kursusel osalevate inimeste esitatud tööd. Töid on võimalik sorteerida erinevate parameetrite alusel: esitaja eesnimi/perekonnanimi, esitamise aeg, esituste arv, hinne, hindaja ning hindamise aeg. Jooksvalt töid üle vaadates oli mugav sorteerida töid selle järgi, kellel oli veel hinne arvestamata. Kõiki töid üle vaadates ei omanud sorteerimise võimalikkus tähtsust.

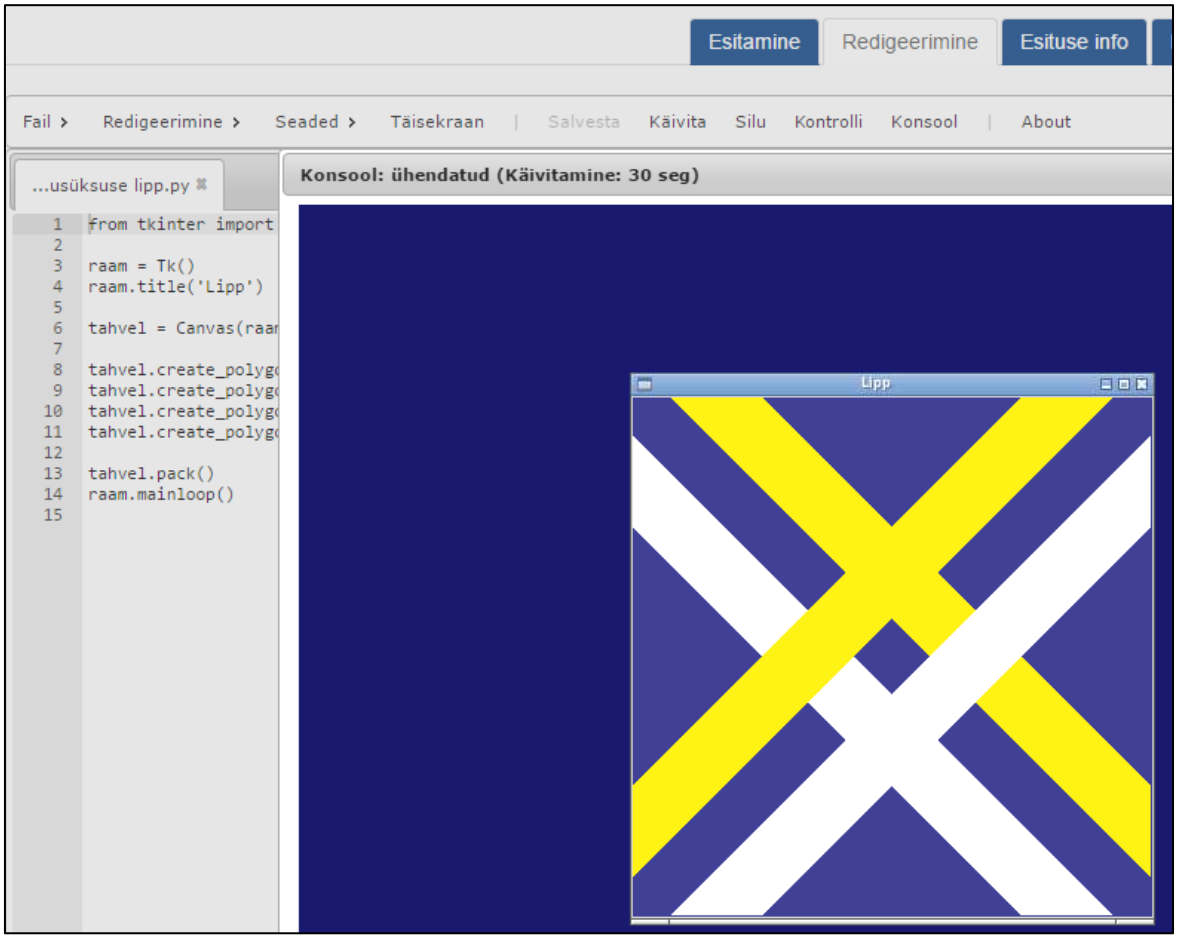


Joonis 11. Esitatud lahenduste alt on võimalik töid sorteerida ning ükshaaval avada.

Konkreetse töö avamiseks on vaja klikkida selle esitamise ajale, seejärel avaneb soovitud töö ning sellega on võimalik teha erinevaid protseduure: uuesti esitada, muuta, käivitada, kontrollida ning hinnata (vt Joonis 8).

Kui esitatud tööl ei olnud ühtegi probleemi (kommentaari veateatega) ning automaatkontroll oli selle ära hinnanud, siis vajutas autor nupule „Redigeerimine“ ning sealt edasi „Käivita“.

Seda toimingut tehes on võimalik visuaalselt näha, kuidas kursusel osaleja esitatud kood virtuaalmasinas käivitatakse ning tulemus ilmub ekraanile. Autor märkis üles tööd, milles esitatud pilt ei vastanud ülesande kirjeldusele. Esines ka juhtusid, kus esitatud töö vastas nõuetele, kuid automaatkontroll ei suutnud lahendusest sobivat pilti tuvastada. Sellistel juhtudel märgiti esitatud lahendus manuaalselt arvestatuks.



Joonis 12. „Käivita“ nupule vajutades on võimalik näha kursusel osaleja esitatud tööd.

5. Tulemused

Käesolevas peatükis antakse ülevaade ülesannete lahendamise statistika kohta ning analüüsitakse kursusel osalejatelt saadud tagasisidet.

5.1 Õpilaste kursus

Õpilastele mõeldud ainele „Programmeerimise alused“ oli kokku registreerinud 275 inimest. Õpilaste kursuse lahenduste juures tuli välja järgmine asjaolu: mitte keegi kursusel osalejatest ei üritanud esitada vale pildi koodi ühegi graafilise ülesande raames.

5.1.1 Lipu ülesanne

Lipu ülesanne oli õpilaste kursusel kõige populaarsem – seda esitati 114 korda. Esitatud töödest said 108 inimest (94,7% kõigist esitustest) automaatse arvestuse ehk kõigi nende kursusel osalejate töödelt suutis automaatkontroll tuvastada korrektse pildi skooriga vähemalt 70%. Ülejäänud kuuele kursusel osalejale (5,3%) andis autor manuaalselt hinde, kuna automaatkontroll ei suutnud esitatud lahendusest piisavalt täpselt pilti tuvastada. Kokku esitati 160 lahendust. Kõige enam esitati üht ülesannet seitse korda ning keskmine esituste arv oli 1,4. Esimese korraga said töö arvestatud 90 õppijat (79%). Ülejäänud pidid arvestuse saamiseks tööd vähemalt kaks korda esitama. Hilisema ülevaatuse käigus selgus, et neli tööd olid saanud automaatkontrolli vea tõttu ebaausa arvestuse. Nendele õppijatele saadeti kiri ning nad esitasid oma tööd uuesti.

5.1.2 Liiklusmärgi ülesanne

Liiklusmärgi ülesannet proovis lahendada 35 kursusel osalejat. Automaatkontroll töötas selle ülesande puhul võrdlemisi hästi. Vaid üks kursusel osaleja 35 õppijast (2,9%) ei saanud automaatset arvestust. Eelmainitud töö hindas autor manuaalselt. Ülejäänud 34 (97,1%) said automaatselt arvestuse. Liiklusmärgi ülesannet esitati kokku 61 korda. Võrreldes lipu ülesandega, kus enim esitusi ühelt inimeselt oli seitse, esitati liiklusmärgi ülesannet kõige rohkem koguni 16 korda. Ka keskmine esituste arv on kõrgem – 1,7. Automaatse hinde sai esimesel korral 28 kursusel osalejat (80%), mis on üsna sarnane lipu ülesandele. Ka liiklusmärgi ülesande automaatkontrollis esines üks viga. Nimelt ka selle ülesande puhul sai automaatkontrolli vea tõttu üks kursusel osaleja valesti arvestuse. Õppijatele anti sellest eksimusest teada ning ta esitas oma töö uuesti.

5.1.3 Maja ülesanne

Kolmest ülesandest pakkus kõige rohkem loomingulisust maja ülesanne, mille lahenduse esitas 62 kursusel osalejat. Nendest 62 õppijast said automaatselt arvestuse 57 kursusel osalejat (91,9%). Ülejäänud viis tööd (8,1%) vaatas autor manuaalselt üle ning märkis need arvestatuks. Maja ülesannet laaditi üles kokku 157 korda ning inimese kohta esitati seda ülesannet keskmiselt kõige rohkem – 2,4 korda. Huvitav maja ülesande juures on asjaolu, et seda esitati maksimaalselt 28 korda ühe inimese poolt, mis on oluliselt rohkem kui eelmise kahe ülesande puhul. Seda ülesannet esitades sai kõige väiksem protsent õppijaid esimese korraga arvestuse, nimelt 42 (67,7%). Ka maja ülesannet esitades leidus kaks tööd, mille korral andis automaatkontroll valesti arvestuse. Mõlemad kursusel osalejad said teavituse ning esitasid lahenduse uuesti.

5.2 Täiskasvanute kursus

Täiskasvanud inimestele mõeldud ainele „Programmeerimise alused“ oli kokku registreerinud 1655 inimest. Kui õpilaste variandis kursusest ei olnud ühtegi katsetust esitada vale pildi koodi, siis täiskasvanute puhul tehti seda kokku seitse korda.

5.2.1 Lipu ülesanne

Ka täiskasvanute seas osutus lipu ülesanne kõige populaarsemaks – seda esitati 695 korda. Esitatud töödest said 673 inimest (96,8% kõigist esitustest) automaatse arvestuse. Ülejäänud 22 osalejat (3,2%) said töö autorilt manuaalselt hinde, kuna automaatkontroll ei suutnud esitatud lahendusest piisavalt täpselt pilti tuvastada. Kokku esitati lipu ülesandele 948 lahendust. Kõige enam esitati üht ülesannet 17 korda ning keskmine esituste arv oli sarnaselt õpilaste kursusele 1,4. Esimese korraga said töö arvestatud 564 osalejat (81,2%). Ülejäänud pidid arvestuse saamiseks tööd vähemalt kaks korda esitama. Hilisema ülevaatuse käigus selgus, et kümme tööd olid saanud automaatkontrolli vea tõttu ebaausa arvestuse. Nendele kursusel osalejatele saadeti kiri ning nad esitasid oma tööd uuesti.

5.2.2 Liiklusmärgi ülesanne

Liiklusmärgi ülesannet proovis lahendada 263 osalejat. Automaatkontroll töötas selle ülesande puhul võrdlemisi hästi. Nagu ka õpilaste kursuse puhul, ei saanud vaid üks õppija 263 kursusel osalejast (0,4%) automaatset arvestust. Eelmainitud töö hindas autor manuaalselt. Ülejäänud 262 (99,6%) said automaatselt arvestuse. Liiklusmärgi ülesannet

esitati kokku 307 korda. Võrreldes lipu ülesandega, kus enim esitusi ühelt inimeselt oli koguni 17, esitati liiklusmärgi ülesannet kõige rohkem vaid 4 korda ühe inimese kohta. Ka keskmine esituste arv on madalam – 1,2. Automaatse hinde sai esimesel korral 234 osalejat (89%), mis on pisut kõrgem lipu ülesandest. Liiklusmärgi ülesande automaatkontrollis esines viis viga. Nimelt ka selle ülesande puhul sai automaatkontrolli vea tõttu viis osalejat valesti arvestuse. Neile anti sellest eksimusest teada ning nad esitasid oma töö uuesti. Huvitav on asjaolu, et seda ülesannet proovis kuus (2,3%) erinevat inimest lahendada esitades lipu ülesande lahendust.

5.2.3 Maja ülesanne

Huvitaval kombel lahendati maja ülesannet täiskasvute kursusel ainult veidi rohkemate osalejate poolt kui liiklusmärgi ülesannet võrreldes õpilaste kursusega. Õpilaste kursusel lahendati maja ülesannet peaaegu kaks korda rohkem kui liiklusmärgi ülesannet. Maja ülesannet lahendas 283 inimest võrreldes 263 liiklusmärgi lahendajaga. Nendest 283 osalejast said automaatselt arvestuse 262 inimest (92,6%). Ülejäänud 21 tööd (7,4%) vaatas autor manuaalselt üle ning märkis need arvestatuks. Maja ülesannet laaditi üles kokku 520 korda ning inimese kohta esitati seda ülesannet keskmiselt kõige rohkem – 1,8 korda. Ka maja ülesande juures on huvitav asjaolu, et seda esitati maksimaalselt 15 korda ühe inimese poolt, mis on oluliselt rohkem kui liiklusmärgi ülesande puhul, aga veidi vähem kui lipu ülesande puhul. Seda ülesannet esitades sai kõige väiksem protsent kursusel osalejaid esimese korraga arvestuse, nimelt 207 (73,1%). Ka maja ülesannet esitades leidus kaks tööd, mille korral andis automaatkontroll valesti arvestuse. Mõlemad kursusel osalejad said teavituse ning esitasid lahenduse uuesti. Lisaks esines ka selle ülesande juures üks töö, mille lahenduseks esitati lipu ülesande lahendus, kuid see ei läbinud automaatkontrolli.

5.3 Tagasisideküsitlus

Kursuse neljandal nädalal viidi läbi küsitlus, mis polnud osalejatele kohustuslik. Küsitlus koosnes 14 küsimusest, millest pooled olid antud lõputööga seotud. Küsimustikule vastas 766 kursusel osalejat.

Tabel 3. Graafilise väljundiga ülesannete keerukus.

Ülesanne/Keerukus	Lipu joonistamine	Liiklusmärgi joonistamine	Maja joonistamine
1 (Liiga lihtne)	7 (1.2%)	3 (1.1%)	9 (3.2%)
2	49 (8.6%)	25 (9.5%)	34 (11.9%)
3	241 (42.3%)	123 (46.6%)	144 (50.7%)
4	244 (42.8%)	101 (38.3%)	86 (30.3%)
5 (Liiga keeruline)	29 (5.1%)	12 (4.5%)	11 (3.9%)
Ei lahendanud ülesannet	196	502	482

Esimese kursusega seotud küsimusega soovis töö autor teada kursusel osalejate arvamust graafilise lahendusega ülesannete keerukuse kohta (vt Tabel 3). Vastajatel oli võimalik anda vastus vahemikus ühest (liiga lihtne) viieni (liiga keeruline). Esimesena tuli tagasisidet anda maja joonistamise ülesande kohta, teisena liiklusmärgi joonistamise ülesande ning kolmandana maja joonistamise ülesande kohta. Üldiselt leidsid kursusel osalejad, et ülesanded olid kas keskmise või keskmisest raskema keerukusega. Lipu joonistamise ülesanne oli kursusel osalejate seas kõige populaarsem, kuid enim osalejaid pidas seda kõige keerulisemaks. Liiklusmärgi ning maja joonistamise ülesanded olid vähem populaarsed, kuid just nende kahe ülesande raskusaste tundus vastajaile sobivaim olevat.

Järgmises küsimuses sooviti kursusel osalejatelt saada üldist tagasisidet graafilise lahendusega ülesannete kohta. Peamised positiivsed aspektid, mis kursusel osalejate vastustest selgusid, olid järgmised: võimalus valida graafiliste ülesannete raskusastet; üksteise tööde võrdlemine selleks ettenähtud foorumis. Kursusel osalejate jaoks oli keeruline ning aeganõudev lahendada graafilisi ülesandeid, kus oli tarvis kasutada geomeetrilisi objekte ning koordinaate. Huvitaval kombel panustasid osad kursusel osalejad graafiliste ülesannete lahendamisele teistest oluliselt rohkem aega, et lahendust pidevalt täiustada ning valmis saada.

766 osalejast 404 (52.7%) leidsid, et graafilise väljundiga ülesanded on kõige huvitavamad (kokku kaheksa ülesande hulgast, mis toimusid kolmanda ning neljanda nädala käigus).

Viimases küsimuses paluti kursusel osalejatel anda tagasisidet graafilise lahendusega ülesannete automaatkontrolli kohta. Täpsemalt paluti teada anda, kui hästi või halvasti loodud süsteem töötab. Vastata oli võimalik skaalal ühest (ei tööta üldse) viieni (töötab väga hästi). Keskmiseks tulemuseks antud küsimusele kujunes 4.4. Kursusel osalejad jäid rahule automaatkontrollide üldise tööga, kuid mõned seisukohad tulid eriti selgelt esile: automaatkontrollimine on kiire ning töötab hästi; mõnedel kursusel osalejatel esines automaatkontrolli kasutades probleeme, kuid nende lahendus sai sellegipoolest hindeks „arvestatud“ – seejuures ei informeeritud tekkinud probleemi osas kursuse läbiviijaid; automaatkontrollist saadud tagasiside on ebatäpne ning ei aita kursusel osalejal oma lahendust eriliselt parandada; kursusel osalejad ei tea detailselt, kuidas automaatkontroll nende töid hindab ning seetõttu puudub neil ettekujutus sellest, mida neilt oodatakse.

5.4 Mida saavad loodud süsteemist kursuse läbiviijad?

Kursusel osales 1828 inimest ning graafilise lahendusega ülesandeid esitati ning hinnati automaatselt 2272 korda. Antud tulemuste hulgas oli 4.6% valenegatiivseid ning 0.5% valepositiivseid hindeid. Selleks, et olla kindel süsteemi töökindluses, vaatas lõputöö autor kõik lahendused ka käsitsi üle. Hinnanguliselt säästeti lõputöö käigus valminud rakendust kasutades 28 tundi kursuse läbiviijate aega. See-eest süsteemi loomine võttis lõputöö autoril vähemalt kaks korda sama kaua. Oluline on märkida asjaolu, et süsteemi kasutegur tuleb välja seda korduvalt erinevate kursuste ning ülesannete raames kasutades.

6. Tehniline teostus

Käesolevas peatükis antakse detailne ülevaade tehnoloogiatest, mida antud lõputöös valminud süsteem kasutas. Lisaks kirjeldatakse süsteemi töötamise protsessi ning kuidas tehnoloogiad koos töötavad.

6.1 Süsteemi loomisel kasutatud tehnoloogiad

6.1.1 Python

Kuna mõlemad kursused, millega antud lõputöö on seotud, õpetavad osalejatele programmeerimiskeelt Python ning kõik teiste ülesandetüüpide automaatkontrollid on kirjutatud selles keeles, otsustas autor ka graafiliste ülesannete automaatkontrollid teha programmeerimiskeeles Python. Seda programmeerimiskeelt peetakse küllaltki lihtsaks, kuna see on dünaamiliste andmetüüpidega keel ehk programmeerija ei pea määrama muutuja tüüpe. Lisaks peetakse seda algajale parimaks keeleks, kuna tänu lihtsale süntaksile on võimalik mõelda probleemile, mitte süntaksile. Python võimaldab kasutada paljusid programmeerimisstiile, näiteks objektorienteeritud, funktsionaalne või protseduuriline programmeerimine. Kasutades Pythonit, on võimalik teha veebirakendusi, teaduslikke arvutusi ning lokaalseid rakendusi keeruliste kasutajaliidestega. Seda keelt on võimalik kasutada kõigi suuremate operatsioonisüsteemidega. Pythoni installeerimisega tuleb vaikimisi kaasa suur hulk teeke, mis võimaldavad paljusid enamlevinud ülesandeid lahendada (nt veebirakenduste loomine, regulaaravaldiste kasutamine, failist lugemine ning kirjutamine jm). Tänu interaktiivsusele on võimalik loodud programme väga kiiresti käivitada ning testida. Automaatne mäluhaldus eemaldab programmeerijalt kohustuse mälu jagada ja vabastada [19][20][21].

6.1.2 Python Grader

Python Grader on moodul, mis tegeleb automaatkontrollidega. See annab kohandatud tagasisidet esitatud ülesannetele. Mooduliga on võimalik luua erinevaid kontrollmeetodeid. Python Grader on oluline, kuna võimaldab hinnata interaktiivseid sisend-väljund tüüpi ülesandeid. Õppija jaoks on tähtis, et ta saab kohest tagasisidet, mis aitab tehtud vea kiirelt leida. Python Grader on installitud virtuaalmasinasse ning selle väljund on JSON tüüpi, mis võimaldab sellel omakorda suhelda Moodle hindamissüsteemiga. Tegemist on ohutu lahendusega, kuna igal kasutusel luuakse liivakast, millel puuduvad õigused väljaspool

iseennast midagi muuta. Seda moodulit kasutatakse mõlemas kursuses, millega antud lõputöö seotud on [22].

6.1.3 Moodle

Moodle (*Modular Object-Oriented Dynamic Learning Environment*) on avatud lähtekoodiga tasuta e-õppe rakendus, mis tagab nii õpetajatele, administraatoritele kui ka õppijatele ühtse turvalise integreeritud süsteemi, milles on võimalik kerge vaevaga luua kohandatud õppimiskeskkondi. 2017. aasta 3. märtsi seisuga on see kasutusel 234 riigis ning selles on üle 11 miljoni kursuse. Moodle'it kasutab üle saja miljoni inimese - kõige populaarsem on rakendus USAs, Hispaanias ning Brasiilias (enim registreerunud kasutajaid). Üle 500 laienduse tagavad selle, et rakendust on võimalik kasutada nii hindamiseks, küsitlusteks, ülesanneteks kui ka sotsiaalseks ning ühiseks õppimiseks. Seda on lihtne installeerida igale arvutile, mis suudab jooksutada PHP-d ning mis toetab SQL-tüüpi andmebaasi. Moodle on platvormist ning operatsioonisüsteemist sõltumatu ning seda on tõlgitud väga paljudesse erinevatesse keeltesse [23][24][25].

6.1.4 GhostScript

Ghostscript on tarkvara, mis pakub järgnevaid teenuseid:

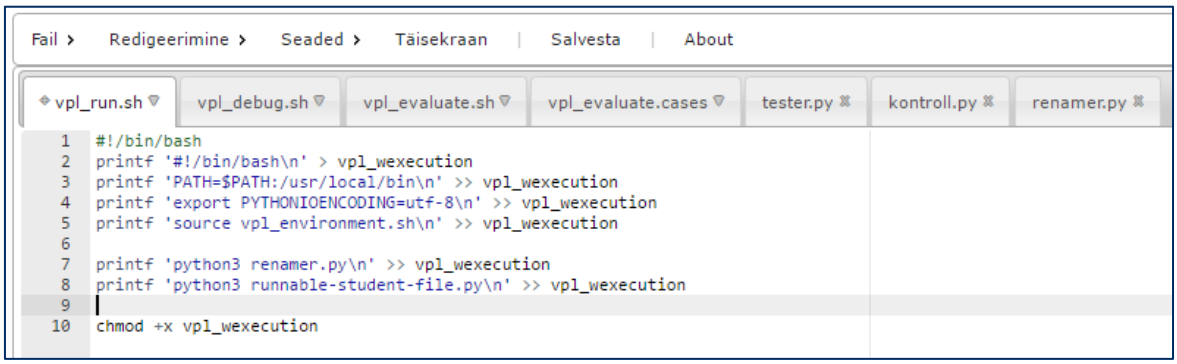
1. PostScript (TM) keele interpreteerimine – sealhulgas võimaldades seda teisendada erinevatesse formaatidesse (nt .JPG). Lisaks võimaldab selles formaadis olevaid faile kuvada ning printida printerites, millel ei ole sisse ehitatud PostScript keele tuge.
2. PDF (*Portable Document Format*) failide interpreteerimine, täiendavalt samad omadused, mis eelmisel.
3. PostScript keeles olevate failide teisendamine PDF failideks ja vastupidi.
4. Programmeerimiskeele C protseduuride (mis rakendavad graafika ja sorteerimise funktsionaalsusi) kasutamine.

Ghostscript on kirjutatud täielikult C keeles, pöörates erilist tähelepanu asjaolule, et see töötaks ilma vigadeta erinevates operatsioonisüsteemides ning virtuaalmasinates [26].

6.1.5 VPL

Tänu VPLile (*Virtual Programming Lab*) on lihtne programmeerimisülesandeid Moodle'is hallata. See võimaldab koodi Moodle'i keskkonnas muuta, käivitada ning hinnata.

Ülesandeid on võimalik üles laadida, alla laadida, nendest koopiaid varundada ning neid taastada. Kursusel osalejatele on võimalik ülesannete kaupa erinevaid õigusi anda, sõltuvalt nende rollist kursusel [27].



Joonis 13. Iga ülesande jaoks on võimalik eraldi määrata, milliseid toiminguid selle kontrollimisel tehakse.

Täiendavalt on võimalik VPL'is ülesannete automaatteste jooksutada. Nii hindamis- kui ka õppimisprotsess muutuvad seega mugavaks ning lihtsaks. Üheks VPL'i funktsiooniks on „Sarnasuse kontroll", mis aitab leida identseid ning sarnaseid lahendusi, et tuvastada osalejaid, kes pole oma ülesandeid ise lahendanud. VPL on tasuta ning selle kood on saadaval GitHub'is [28].

6.1.6 JSON

JSON (*JavaScript Object Notation*) on lihtsustatud andmetalletus ning -vahetusvorming. Seda on inimesel lihtne lugeda ja kirjutada ning samas on masinatel seda lihtne süntaktiliselt analüüsida ning genereerida. JSON põhineb JavaScripti programmeerimiskeele alamhulgal, on tekstivormingus ning seetõttu programmeerimiskeelest sõltumatu.

Joonis 24. Näide JSON failist, mis sisaldab peamist objekti väljadega „nimi", „vanus" ja „lapsed". Väli „lapsed" sisaldab omakorda massiivi kahe objektiga [29].

JSON koosneb kaht tüüpi struktuuridest:

1. Nimi/väärtus paaride kollektsioon.
2. Väärtused, mis on järjestatud. Paljudes keeltes tuntud kui massiiv, vektor, list, loend või jada [30].

6.2 Antud lõputöö käigus valminud süsteem

Käesolevas peatükis antakse täpsem ülevaade sellest, kuidas valmis süsteem 2017. aasta kursuse „Programmeerimise alused“ jaoks, kuidas see töötab ning milliseid tehnoloogiaid selle valmistamisel kasutati.

Graafilise väljundiga ülesannete käsitsi hindamine on muutunud suureks väljakutseks, kuna see on aeganõudev tegevus. Lisaks koodi hindamisele tuleb hinnata ka graafilist väljundit. Tänu masinõppe kiirele arengule on saadaval mitmeid teenusepakkujaid, kes võimaldavad kiiret ning usaldusväärset pildituvastusteenust (vt peatükk 4).

Uue teenuse väljatöötamiseks oli autoril tarvis teha järgmised tegevused:

- analüüsida varem kasutusel olnud hindamismetoodikat;
- koguda kokku eelmisel korral esitatud kursusel osalejate lahendused;
- analüüsida ning valida pildituvastamise teenusepakkuja;
- Rakendad/teostada uus süsteem;
- katsetada uut süsteemi eelnevalt kokku kogutud lahenduste peal.

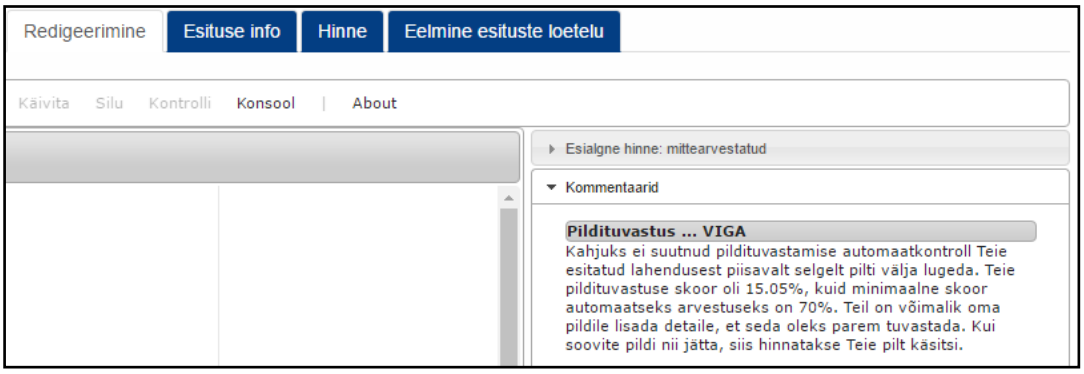
Uue süsteemi loomise protsess algas varem kasutatud hindamismetoodika ning eelmisel kursusel esitatud lahenduste analüüsiga. Esimene probleem, millele tuli leida lahendus, oli tööde esitamiseks kasutusel olnud foorum. Foorumi kasutamisega seotud negatiivsed küljed on mainitud peatükis 3 ning 4. Esitatud lahendusi analüüsides tekkis lõputöö autoril idee genereerida esitatud lahenduse koodist automaatselt pildifail. Sel juhul ei oleks tarvis kursusel osalejal pildifaili kaasa laadida ning poleks ohtu, et pildil on lisaks soovitud väljundile ka muid elemente. Lisaks tagaks see võimaluse koodi kompileerida ning välistaks variandi, et esitatud pilt ning kood pole vastavuses. Enne uue süsteemi rakendamist oli tarvis valida pildituvastamise teenusepakkuja (vt peatükk 4). Pärast sobiva teenusepakkuja väljavalimist oli seda vaja katsetada. Teenuse kiirust, töökindlust ning sobivust katsetati eelmisel kursusel esitatud lahenduste peal.

Graafilise väljundiga ülesanded on kasutusel MOOCis „Programmeerimise alused“. Ülesanded asuvad vabavaralises õpikeskkonnas nimega Moodle (*Modular Object-Oriented Dynamic Learning Environment*) [31]. Programmeerimise ülesanded on loodud ning hinnatud kasutades VPLi (*Virtual Programming Lab*) [32]. VPL võimaldab ülesande loojal defineerida testjuhud iga ülesande jaoks, millel on kasutatud sisendit ning eeldatud väljundit. Lisaks on VPLiga võimalik luua tagasisideteateid vastavalt defineeritud erijuhtudele. Pärast lahenduse esitamist Moodle'isse saadetakse see *VPL execution server*'isse, mis on paigaldatud ühte ülikooli virtuaalmasinasse. Iga kord, kui kursusel osaleja lahendus virtuaalmasinasse saadetakse, luuakse ohutuse tagamiseks virtuaalmasinas uus ajutine liivakast [28].

Kursusel osalejad peavad kasutama graafilise väljundi loomiseks Pythoni teeki nimega Tkinter. Koodi korrektsust kontrollitakse Pythoni abstraktseid süntaksipuid (AST) kasutades [33]. Pärast koodi verifitseerimist nimetatakse esitatud lahendusfail ümber, et vältida probleeme, mis võivad tekkida faili nimes eri sümboleid kasutades. Selleks, et virtuaalmasinas esitatud lahenduskoodiga kõiksugu toiminguid teha (nt käivitada), vajab Tkinteri teek ekraani. Ekraani ühenduse simuleerimiseks virtuaalmasinaga oli tarvis virtuaalmasinasse paigaldada ekraani server nimega Xvfb. Selle tulemusena arvas virtuaalmasina operatsioonisüsteem, et ekraan on ühendatud, kuid tegelikult toimusid kõik graafilised toimingud mälus [34].

Korrektse lahenduse korral oli tarvis esitatud koodi manipuleerida. Täpsemini oli sinna vaja lisada teatud koodilõik, et genereerida esitatud lahendusest pildifail. Pärast lahenduse manipuleerimist oli sellest võimalik tekitada PostScript (.ps) fail. Kasutades tarkvara nimega GhostScript teisendati PostScript fail .JPG või .PNG pildifailiks [26]. Üheks GhostScripti omaduseks ongi nimelt PostScript failide teisendamine rasterpildiks (png, tiff, jpg jne). Pildifail saadeti seejärel Clarifai pildituvastusteenusele, kasutades selleks nende rakendusliidest [18]. Pildifailile lisatakse kaasa ka märksõna, mis esindab konkreetset objekti, mida on vaja antud pildilt tuvastada. Näiteks: kui lahendatud ülesandeks oli maja joonistamine, siis sellisel juhul kaasatakse lahendusest genereeritud pildile märksõna „maja“. Clarifai vastab seejärel, kasutades selleks JSON formaati, milles sisaldub arvuline tõenäosus, et pildiga kaasa saadetud märksõna eksisteerib pildil. Lahendused, mis said tulemuseks kõrgema tõenäosuse kui 0.7, said hindeks „arvestatud“. Antud number valiti kursuse läbiviijate poolt, katsetades süsteemi eelmise aasta lahenduste peal ning tuginedes kogemusele. Tulevikus on võimalik lävendit, millega kursusel osaleja ülesande sooritatud

saab, vastavalt vajadusel jooksvalt muuta, kuid lävendit alla 0.7 langetada ei ole soovituslik, kuna selle tagajärjel kasvab valepositiivsete tulemuste hulk. Kursusel osalejad, kes said esitatud lahenduse tulemuseks madalama skoori kui 0.7, said tagasisideks vastava teate tulemusega ning soovituse esitatud lahendusele lisada detaile või jääda ootama käsitsi hindamist. Kõik tööd, mis ei saanud automaatset arvestust, vaadati kursuse läbiviijate poolt käsitsi üle ning sobivuse korral said ka algselt „mittearvestatud“ lahendused hindeks „arvestatud“. Lahenduste esitamise arv ei olnud piiratud, et kursusel osalejad saaksid soovi korral oma lahendust parandada (vt Joonis 15).



Joonis 15. Kursusel osaleja lahendus, mis ei läbinud automaatset hindamist.

Selleks, et kursusel osaleja esitatud töö saaks hinnatud, on tarvis teha järgnevad toimingud:

1. Kursusel osaleja esitab oma lahenduse (koodi) Moodle kaudu.
2. Lahendus saadetakse VPL Execution Serverisse.
3. Lahendust analüüsitakse Pythoni ASTga.
4. Lahendusfail nimetatakse ümber.
5. Lahendusfaili muudetakse, lisades sinna spetsiifiline kood.
6. Lahendusest genereeritakse PostScript fail.
7. Genereeritud PostScript fail teisendatakse pildifailiks.
8. Pildifail saadetakse koos märksõnaga Clarifai teenusele.
9. Clarifai vastab tõenäosusprotsendiga.
10. Kursusel osaleja saab vastavalt tõenäosusele hindeks kas „arvestatud“ või „mittearvestatud“.

Selleks, et kursusel osalejad saaksid üksteisega oma lahendusi jagada, loodi spetsiaalne foorum. Foorumisse on võimalik üles laadida nii kood kui ka graafiline väljund, mis nende koodist tekib. Foorumisse lahenduse postitamine ei olnud kohustuslik, kuid vajalik selleks, et teiste lahendusi näha.

7. Riskid

Käesoleva lõputöö käigus valminud lahenduse juures esinevad ka teatud riskid. Kuna pildituvastamise jaoks kasutatakse välist teenusepakkujat, siis esimeseks riskiks on see, et teenusepakkuja ei teavita teenuse kasutajaid olulistest muudatustest või ei jõua see info mingil põhjusel teenuse kasutajani. Selline olukord võib tekkida näiteks siis, kui teenusepakkuja otsustab enda API struktuuri muuta. Selline muudatus tähendaks automaatselt seda, et ka antud magistritöös valminud lahendus oleks vaja ümber teha.

Teine risk on samuti seotud teenusepakkujaga. On võimalik, et valitud teenusepakkuja lõpetab töö. Põhjuseid võib olla mitmeid: pankrot, klientide vähene huvi, teenuse vajaduse puudumine või lõppemine. Sellise olukorra tekkimisel oleks tarvis uus teenusepakkuja leida, mis võib aga osutuda keeruliseks, kuna eelnevalt tehtud analüüsi põhjal ei leidunud praegu Clarifaile ühtegi võrdväärset alternatiivi. Uue teenusepakkuja valimine tähendaks omakorda valminud lahenduse ümbertegemist või isegi uuesti tegemist.

Kolmas risk on seotud hetkel kasutusel oleva teenusepakkujaga teenuse hinnaga. Praegu on võimalik kasutada teenust iga kuu 5000 päringu piires tasuta. Piirangu ületamisel tuleb maksta. Kuul, mil teenust kasutati, tehti kokku alla 5000 päringu, mis tähendab, et sel korral oli teenuse kasutamine täiesti tasuta. Kui aga teenusepakkuja peaks otsustama tasuta päringute hulka vähendada või sootuks kaotada, tuleb hakata teenuse eest maksma. Probleem võib ilmneda juba siis, kui valminud lahendus võetakse kasutusele rohkematel kursustel – see tähendab automaatselt rohkem päringuid.

Neljas risk on osaliselt ka lõputöös valminud süsteemi puudujääk. Nimelt ei kontrollita kursusel osalejate lahendustes erinevate värvide kasutamist (nendes ülesannetes, kus nõutud). See tähendab, et osalejad saavad potentsiaalselt hindeks „arvestatud“, kui nad kasutavad vaid kahte värvi: must kujund valgel taustal. Tulevikus on seda riski lihtne vältida täiustades veidi automaatkontrolli ning see aitaks omakorda teha kindlaks, et kursusel osalejad on täitnud kõik ülesannetes esitatud tingimused.

Risk, millele selles lõputöös erilist tähelepanu ei pöörata, on plagiaadid. See on võrdlemisi suur probleem, millega MOOCide raames väga palju ei tegelda. Lahendusi kontrollitakse enamasti automaatselt, mis jätab kursusel osalejatele rohkelt võimalusi süsteemi petta. Üks põhjuseid, miks eelnimetatud probleemiga ei tegelda, on see, et MOOCid on avatud kõigile ning nende eesmärgiks on võimaldada inimestel areneda ning end täiendada. Antud lõputöös käsitletavate kursuste lõpus on võimalik sooritada kontrollitud oludes arvestustöö [35].

Kokkuvõte

Käesoleva lõputöö eesmärgiks oli luua süsteem, mis suudab automaatselt hinnata graafilise väljundiga programmeerimisülesandeid. Süsteem võtab sisendiks kursusel osaleja esitatud lahenduse koodi ning saadab selle virtuaalmasinasse, kus on ajutine liivakast. Liivakastis analüüsitakse lahendust ning muudetakse nii, et sellest saaks genereerida pildifaili. Pildifail saadetakse pildituvastamise teenusepakkujale Clarifaile, mis tagastab vastusena tõenäosuse, et spetsiifiline objekt eksisteerib saadetud pildil. Vastavalt sellele tõenäosusele saab esitatud töö hindeks kas „arvestatud“ või „mittearvestatud“.

Süsteemi muudab mugavaks asjaolu, et esitada tuleb vaid ülesande lahenduskood. Kursusel osalejad jäid automaatkontkontrolli üldise suutlikkusega rahule. Soovi korral on kursusel osalejatel endiselt võimalik oma lahendusi üksteisega jagada selleks ettenähtud foorumis.

Tulevikus on süsteemi kindlasti võimalik täiustada. Hetkel kursusel osalejatele antud tagasiside on väga üldine ning ei aita neid eriliselt konkreetsete probleemide korral. Lisaks oleks võimalik veel täpsemalt kontrollida esitatud lahenduskoodis vajaminevaid elemente ning selle põhjal tagasisidet anda. Logidesse kirjutamist oleks võimalik täiendada ning detailsemaks muuta, et oleks võimalik automaatkontrolli töö kohta paremat statistikat teha ning sealt vigu leida.

Süsteemi kasutati ning testiti programmeerimise MOOCis „Programmeerimise alused“. Kuna kursuse läbiviijad võitsid tänu antud lõputöös valminud süsteemile oluliselt aega ning kursusel osalejatelt saadud tagasiside oli positiivne, siis võib kindlusega väita, et süsteemi kasutatakse ka edaspidi.

Viidatud kirjandus

[1] Pieterse V. Automated assessment of programming assignments. *Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research,* 2013, pp. 45-67.

[2] Wulf J.; Blohm I.; Leimeister J. M.; Brennen W. Massive open online courses. *Business & Information Systems Engineering (BISE),* 2014, vol. 6, no. 2, pp. 111-114.

[3] Doherty I.; Harbutt D.; Sharma N. Designing and Developing a MOOC. *Medical Science Educator,* 2015, vol. 25, no. 2, pp. 177-181.

[4] Yousef A. M. F.; Wahid U.; Chatti M. A.; Schroeder U.; Wosnitza M. The Impact of Rubric-Based Peer Assessment on Feedback Quality in Blended MOOCs. *International Conference on Computer Supported Education,* 2015, pp. 462-485.

[5] Pears A.; et al. A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, 2007, vol. 39, no. 4, pp. 204-223.

[6] Staubitz T.; Klement H.; Renz J.; Teusner R.; Meinel C. Towards practical programming exercises and automated assessment in massive open online courses. *Teaching, Assessment, and Learning for Engineering (TALE)*, 2015, pp. 23-30.

[7] English J. Automated assessment of GUI programs using JEWL. *ACM SIGCSE Bulletin*, 2004, vol. 36, no. 3, pp. 137-141.

[8] Thornton M.; et al. Supporting student-written tests of gui programs. *ACM SIGCSE Bulletin*, 2008, vol. 40, no. 1, pp. 537-541.

[9] Nikander J.; et al. Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. *Informatics in Education-An International Journal*, 2004, vol. 3, no. 2, pp. 267-288.

[10] Graphical User Interfaces with Tk. 2017. https://docs.python.org/3/library/tk.html (16.05.2017)

[11] Salve S. G.; Jondhale K. C. Shape matching and object recognition using shape contexts. *Computer Science and Information Technology (ICCSIT)*, 2010, vol. 9, pp. 471-474.

[12] CLOUD VISION API. 2017. https://cloud.google.com/vision/ (16.05.2017)

[13] Mulfari D.; et al. Using Google Cloud Vision in assistive technology scenarios. *Computers and Communication (ISCC)*, 2016, pp. 214-219.

[14] IaaS, PaaS, SaaS. 2013. http://www.termnet.ee/pilveteenus/iaas-paas-saas (16.05.2017)

[15] Dimiccoli M.; et al. SR-clustering: Semantic regularized clustering for egocentric photo streams segmentation. *Computer Vision and Image Understanding,* 2016.

[16] About Imagga Technologies Ltd. 2017. https://imagga.com/company (16.05.2017)

[17] Coban E. B. Neural Networks and Their Applications.

[18] About Clarifai. 2017. https://clarifai.com/about (16.05.2017)

[19] Learn Python Programming. 2015. https://www.programiz.com/python-programming (16.05.2017)

[20] What is Python? Executive Summary. 2017. https://www.python.org/doc/essays/blurb/ (16.05.2017)
[21] Python Overview. 2016. https://wiki.python.org/moin/BeginnersGuide/Overview (16.05.2017)
[22] Python Grader. 2017. https://github.com/kspar/python-grader (16.05.2017)
[23] What is Moodle? 2017. http://www.lambdasolutions.net/resources/what-is-moodle/ (16.05.2017)
[24] Mis on Moodle? 2010. http://etu.ut.ee/kevad-2010/mis-on-moodle/ (16.05.2017)
[25] About Moodle. 2016. https://docs.moodle.org/32/en/About_Moodle (16.05.2017)
[26] GhostScript. 2016. https://www.ghostscript.com/Ghostscript.html (16.05.2017)
[27] Moodle's new Virtual Programming Lab (VPL) module. 2010. http://www.moodlenews.com/2010/moodles-new-virtual-programming-lab-vpl-module/ (16.05.2017)
[28] VPL General Documentation. 2014. http://vpl.dis.ulpgc.es/index.php/support (16.05.2017)
[29] JSON. 2015. https://et.wikipedia.org/wiki/JSON (16.05.2017)
[30] Introducing JSON. 2015. http://www.json.org/ (16.05.2017)
[31] Dougiamas M.; Taylor P. Moodle: Using learning communities to create an open source course management system. 2003, pp. 171-178.
[32] Rodríguez-del-Pino J. C.; Rubio Royo E.; Hernández Figueroa Z. A Virtual Programming Lab for Moodle with automatic assessment and anti-plagiarism features. *Proceedings of The 2012 International Conference on e-Learning, e-Business, Enterprise In-formation Systems, & e-Government*, 2012.
[33] Abstract Syntax Trees. 2017. https://docs.python.org/3.6/library/ast.html (16.05.2017)
[34] Xvfb Homepage. 2012. https://www.x.org/archive/X11R7.6/doc/man/man1/Xvfb.1.xhtml (16.05.2017)
[35] Programmeerimise alused. 2017. http://www.ut.ee/et/mooc/programmeerimise-alused (16.05.2017)

Lisad

I. Failid

- Githubist on võimalik kätte saada süsteemi loomiseks ning testimiseks vaja läinud faile ning pilte.
    - https://github.com/heavenzeyez1/Magistritoo

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Eerik Muuli**,
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
   Graafiliste ülesannete automaatkontroll programmeerimise vaba juurdepääsuga e-kursuste raames,
   (*lõputöö pealkiri*)

   mille juhendaja on Eno Tõnisson,
   (*juhendaja nimi*)

   1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **18.05.2017**