TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Mattias Tüür 213115IAIB

Sander Pleesi 213070IAIB

Lähtekoodi sarnasust tuvastava süsteemi edasiarendus

Bakalaureusetöö

Juhendaja: Bahdan Yanovich

Bakalaureusekraad

Tallinn 2025

Autorideklaratsioon

Kinnitame, et oleme 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.

Autorid: Mattias Tüür ja Sander Pleesi

03.06.2025

Annotatsioon

Plagiaat võib esineda nii akadeemilistes kirjatöödes kui ka programmikoodis ning on kahetsusväärselt levinud nähtus. Kuna tegemist on rangelt keelatud tegevusega, mida kohtab ka ülikoolides, tuleb tudengite töid regulaarselt kontrollida.

Tallinna Tehnikaülikoolis on selleks loodud veebirakendus, mis aitab tuvastada tudengite kirjutatud lähtekoodide sarnasusi. Selle eesmärk on analüüsida, kui palju plagiaati esineb programmeerimisülesannete lahendamisel ning tuvastada, millised tudengid on võimaliku rikkumisega seotud.

Plagiaadi käsitsi kontrollimine on aeganõudev ja keerukas, eriti kui aines osaleb suur hulk tudengeid. Kõigi lahenduste individuaalne uurimine ja võrdlemine muutub sellisel juhul väga ebamugavaks. Veebirakendus aitab seda protsessi oluliselt kiirendada ja mugavamaks muuta.

Käesoleva lõputöö eesmärk on täiustada rakendust, mille olemasolevate funktsionaalsuste juures esineb mitmeid puudujääke. Plagiaadikontrolliks kasutatakse ainult ühte välisteenust, mis on vahepeal maas ning piirab seetõttu rakenduse kasutust. Samuti on tudengite haldamine ja andmete analüüsimine ebamugav, nõudes käsitsi andmete sirvimist.

Lõputöö käigus kirjeldatakse projekti struktuuri ja uusi funktsionaalsusi. Töö tulemuseks on täiustatud rakendus, mis võimaldab plagiaadikontrolli läbi viia mitme teenuse või programmi abil ning pakub kasutajale paremat ülevaadet kursuse tudengite kohta. Samuti lihtsustab see plagiaadijuhtumite haldamist ja analüüsimist.

Töö tellijaks on õppejõud ja informaatika eriala programmijuht Ago Luberg, kes juhendab mitmeid programmeerimisõppeaineid. Nendes ainetes on oluline kontrollida tudengite lahenduste autentsust ja võimalikku plagiaati.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 48 leheküljel, 7 peatükki, 21 joonist, 1 tabel.

Abstract

Further Development of a System for Detecting Software Similarity

Plagiarism can occur both in academic writing and in source code, and it is unfortunately a widespread phenomenon. Since it is strictly prohibited, including in universities, students' work must be checked regularly.

At Tallinn University of Technology, a web application has been developed to detect similarities in students' source code. Its purpose is to analyze the extent of plagiarism in programming assignments and to identify students who may be involved in potential violations.

Manual plagiarism checking is time-consuming and complex, especially when a large number of students are enrolled in a course. In such cases, investigating all submissions individually becomes highly inconvenient. The web application helps to significantly simplify this process.

The aim of this thesis is to improve the existing application, which currently has several shortcomings. Only one external service is used for plagiarism detection, and since this service is sometimes unavailable, it limits the functionality of the application. In addition, managing students and analyzing the data is inconvenient.

This thesis describes the project structure and introduces new functionalities. The result is an enhanced application, which enables plagiarism detection using multiple services or programs and provides a better overview of students and plagiarism cases.

The client for this thesis is Ago Luberg, a lecturer and program director of Informatics, who teaches multiple programming courses.

The thesis is in Estonian and contains 48 pages of text, 7 chapters, 21 figures, 1 table.

Lühendite ja mõistete sõnastik

AC	Plagiaadituvastuse programm
ACNP	*AntiCutAndPaste*, plagiaadituvastuse programm
AI	*Artificial intelligence*, tehisintellekt
API	*Application Programming Interface*, rakendusliides
Charon	Pistikprogramm
CLI	*Command Line Interface*, käsurea liides
CSS	*Cascading Style Sheets*, veebilehe kujundamise keel
Dolos	Plagiaadituvastuse programm
GitLab	Versioonihalduskeskkond
HTML	*Hypertext Markup Language*, veebilehe märgendamise keel
IDE	*Integrated Development Environment*, arendus keskkond
JAR	*Java Archive*, paketifailivorming
Java	Programmeerimiskeel
JavaScript	Programmeerimiskeel
JDK	*Java Development Kit*, tarkvaraarenduse komplekt
JPlag	Plagiaadituvastuse programm
JSON	*JavaScript Object Notation*, andmevahetusvorming
Moss	*Measure Of Software Similarity*, plagiaadituvastuse programm
ORM	*Object-relational mapping*, objekt-relatsiooniline kaardistamine
OSPC	*Open Software Plagiarism Checker*, plagiaadituvastuse programm
Plaggie	Plagiaadituvastuse programm
Python	Programmeerimiskeel
Sherlock	Plagiaadituvastuse programm
SIM	*The software and text similarity tester*, plagiaadituvastuse programm
SQL	*Structured Query Language*, struktuurpäringukeel
Uni-ID	*University identification*, digitaalne identiteet
URL	*Uniform Resource Locator*, tähistab ressursi asukohta võrgus
ZIP	Tihendatud failiformaat

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Plagiaat on teise autori töö omastamine ilma viitamiseta, esitades selle ekslikult originaalmaterjalina. See on petlik käitumine ja ebaeetiline päris autori suhtes.

See tähendab, et plagieerija ei ole ise pingutanud uute teadmiste omandamiseks. Selle asemel on toime pandud vargus, mis väärib karistust, kuna plagieeritud tööd ei saa ametlikult aktsepteerida.

Plagiaati esineb sageli kirjalike tööde koostamisel, kuid on levinud ka programmeerimises. Plagiaadiks loetakse koodi, mis sarnaneb liigselt kellegi teise loodud koodiga, olgu see siis struktuuri või nimetuste poolest.

Tallinna Tehnikaülikooli informaatika ja teiste IT erialade õppekavades on mitmeid programmeerimisõppeaineid, kus tutvutakse erinevate programmeerimiskeelte ja -tööriistadega. Sellistes õppeainetes on oluline, et kõik deklareerunud tudengid omandaksid teadmised ja oskused iseseisvalt, vältides plagiaati. Seetõttu peavad õppejõud ja abiõppejõud regulaarselt kontrollima tööde originaalsust, mille käsitsi tegemine on ajakulukas ja ebamugav.

Protsessi kiirendamiseks arendati 2019. aastal lõputööna valmis rakendus, millega võimalik programmeerimisülesannetele teha plagiaadikontrolle [1]. Küll aga on rakendusel mitmeid puudujääke, mis muudavad rakenduse kasutuse ebamugavaks.

Käesoleva lõputöö eesmärk on arendada olemasolevat rakendust edasi, muutes selle kasutajasõbralikumaks ja tõhusamaks. Töö tulemusena saavad õppejõud ja abiõppejõud kasutada mitmeid erinevaid teenuseid ja programme plagiaadikontrolliks ning tutvuda visuaalsete andmetega, mis annavad ülevaate tudengite tööde võimalike plagiaadijuhtumite kohta.

1.1 Teema valik

Teema valik tulenes lõputöö tegijate huvist ning soovist panustada õppeainete kvaliteedi parandamisele.

Tudengite tööde kontrollimine on oluline, et saada selge ülevaade nende edasijõudmisest – kui paljud sooritavad õppeaine edukalt ja kui paljudel tekib raskusi. Seetõttu on hädavajalik mugav ja tõhus süsteem plagiaadi kontrollimiseks.

Lõputöö autorid on varasemalt tegutsenud abiõppejõududena õppeainetes "Programmeerimise algkursus" ja "Programmeerimise põhikursus", mis on nende huvi teema vastu veelgi suurendanud. Nende kogemus nii tudengi kui ka õppejõu perspektiivist annab hea ülevaate programmeerimisõppe ülesehitusest, aidates arendada plagiaadikontrolli süsteemi veelgi paremaks.

1.2 Eesmärgid ja oodatav tulemus

Kuna plagiaadikontrolli rakendus loodi juba mitu aastat tagasi ja sellele on pärast valmimist tehtud väga vähe muudatusi, esineb rakendusel mitmeid puudujääke.

Käesoleva lõputöö eesmärgid ja oodatavad tulemused on:

- Parandada olemasolevat plagiaadikontrolli rakendust, muutes selle kasutajatele mugavamaks ja intuitiivsemaks.
- Liidestada juurde teenuseid, millega kasutaja saaks teha plagiaadikontrolli.
- Suurendada plagiaadikontrolli täpsust, et tulemused oleksid täpsemad ja usaldusväärsemad.

1.3 Töökorraldus ja meetodid

Lõputöö autorid kasutasid oma tööprotsessis agiilset metoodikat, rakendades regulaarseid koostööprotokolle. Igapäevaseks suhtluseks kasutati Discordi platvormi, kus arutati töö edenemist ja tehnilisi detaile [2]. Lisaks peeti igal kolmapäeval koosolekuid Microsoft Teamsi keskkonnas, mida juhtis lõputöö juhendaja [3].

Nendel koosolekutel analüüsiti tehtud tööd, tuvastati edasised arendusvajadused ning

koostati järgmise nädala tegevuskava. Pärast iga koosolekut jaotati ülesanded autorite vahel vastavalt koostatud plaanile. Selline lähenemine võimaldas säilitada töö pidevat hoogu ja tagas probleemide õigeaegse tuvastamise.

Lisaks sisemisele töökorraldusele toimus ka regulaarne suhtlus kliendiga, kes andis tagasisidet arendatud lahenduse kohta ning pakkus välja ideid ja funktsionaalsusi, mida võiks rakendusse juurde lisada. Kliendiks oli informaatika õppekava programmijuht ja õppejõud Ago Luberg, kelle sisend aitas suunata arendust vastavalt reaalsele vajadusele ning tagas lahenduse parema kasutatavuse sihtgrupi jaoks.

2 Olemasolevate lahenduste analüüs

Rakenduse edasi arenduse jaoks tuleb eelnevalt tutvuda olemasolevate funktsionaalsustega ning uurida tehnoloogiaid, mis suudavad teha plagiaadi kontrolli. Eesmärk on ära kaardistada rakenduse head küljed ning kitsaskohad, mis vajavad parandamist ja arendamist. Samuti analüüsitakse läbi mitmed erinevad programmid, mis võimaldavad teha plagiaadikontrolle, et nende seast valida välja kõige sobivamad.

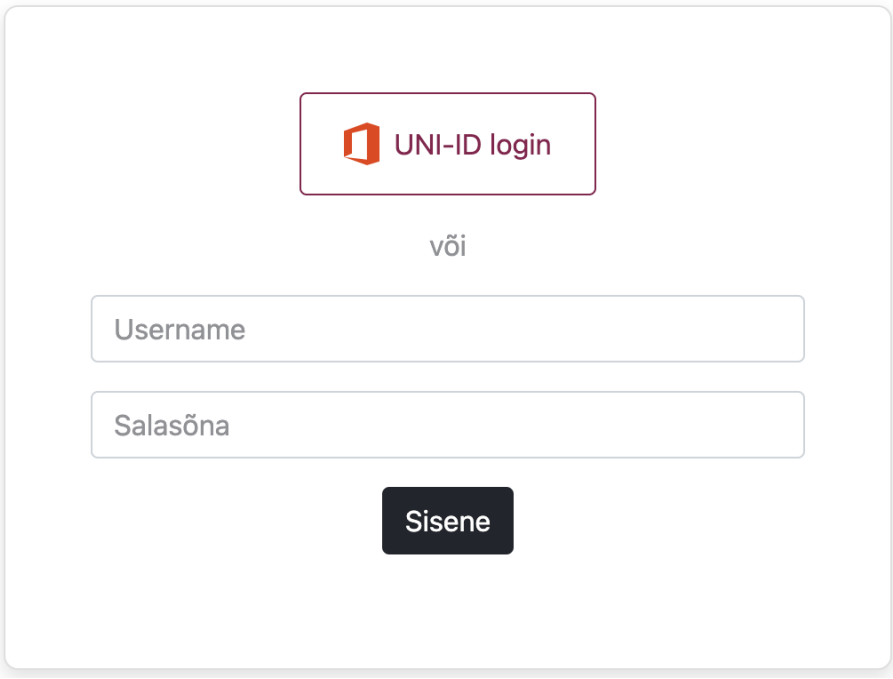
2.1 Olemasolev rakendus

Käesolev alapeatükk annab ülevaate olemasolevast rakendusest, mis loodi 2019. aastal lõputööna [1]. Käiakse läbi kõik funktsionaalsused, mis rakendusel olemas ning mis nende abil kasutajal võimalik teha.

2.1.1 Kasutajad ja sisselogimine

Enne rakenduse veebilehele ligipääsemist peab õppejõud enda kasutajaga sisse logima. Selle jaoks on loodud spetsiaalne sisselogimisvaade koos vormiga, kuhu sisestada kasutajaandmed (Joonis 1).

Sisselogimiseks on kaks võimalust. Esimene võimalus on sisestada kas kasutajanimi või meil koos parooliga. See eeldab, et eelnevalt on õppejõule loodud rakenduse kasutaja administraatori vaates. Alternatiivina saab sisse logida Microsofti autentimise kaudu, vajutades nupule *"UNI-ID login"*. See eeldab, et kasutajal on olemas Microsofti konto, millega võimalik autentida.



Joonis 1. Kasutaja sisselogimisvorm.

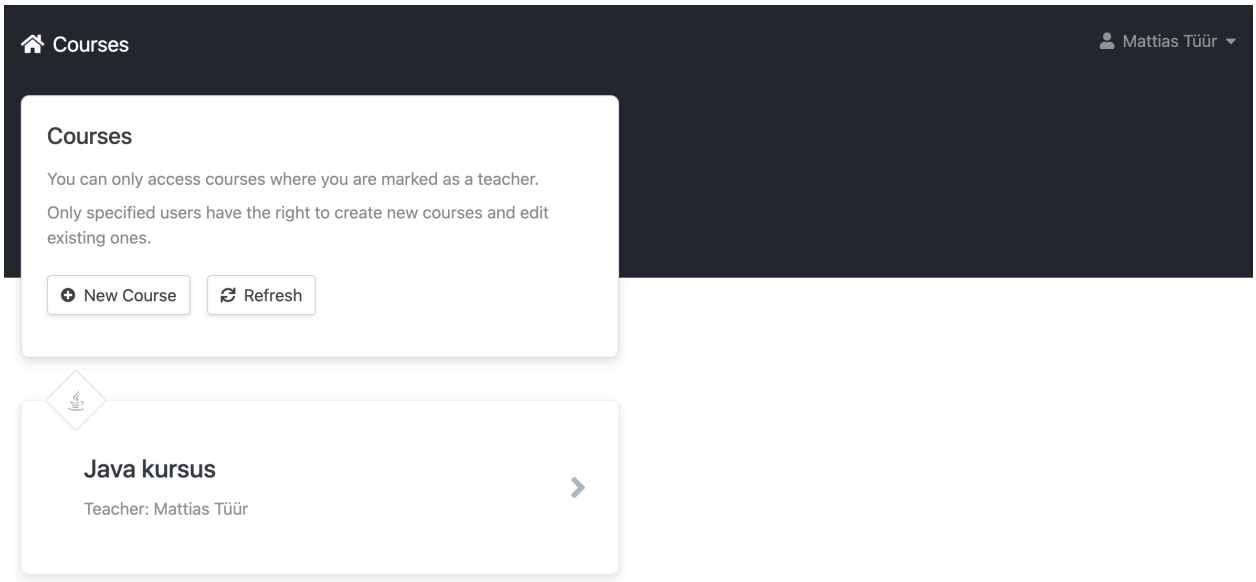
Kui sisestatud andmed on valiidsed ning sisselogimine õnnestub, saab kasutaja ligipääsu rakenduse kõikidele vaadetele ning olenevalt õigustest hakata süsteemis navigeerima. Igas vaates on kasutajal alati võimalus veebilehelt välja logida.

Rakenduses on kolm kasutajarolli: tavakasutaja, haldur ja administraator. Tavakasutaja saab tutvuda ainult nende kursustega, kuhu ta on õpetajana lisatud, kuid tal puudub ligipääs muudele funktsioonidele. Halduril on õigus luua uusi kursuseid ja nende alla ülesandeid ning viia läbi plagiaadikontrolli. Haldurile on nähtavad vaid need kursused, mille ta on ise loonud või kuhu on lisatud õpetajana. Administraatoril on kõik halduri õigused, kuid lisaks saab ta näha ja hallata kõiki rakenduses loodud kursuseid.

Selline rollijaotus võimaldab selgelt eristada kasutajate õiguste tasemeid ning määrata igale kasutajale vastavad ligipääsuvõimalused.

2.1.2 Põhivaade

Põhivaates saab kasutaja hallata kõiki kursusi, mille ta on loonud või kuhu ta on määratud õpetajana. Põhivaate sisuga saab tutvuda joonisel 2.



Joonis 2. Rakenduse põhivaade koos ühe loodud kursusega.

Uue kursuse loomiseks on spetsiaalne nupp, millele vajutades avaneb vorm. See koosneb erinevatest sisendväljadest, millest osa on kohustuslikud ja osa valikulised:

Üldised seaded

- Kursuse nimi (*Name*) – kohustuslik väli, kuhu sisestatakse kursuse nimi. Nimi peab olema unikaalne ja eristuma teistest olemasolevatest kursustest.
- Programmeerimiskeel (*Language*) – kohustuslik väli, mille kaudu valitakse programmeerimiskeel, mida kursuse ülesannetes kasutatakse.
- Charoni identifikaator (*Identifier in Charon*) – valikuline väli, kuhu saab sisestada identifikaatori, mis seob kursuse Charoni süsteemiga.

GitLabi seaded

- Grupp (*Group*) – kohustuslik väli, mis seob kursuse määratud GitLabi grupiga, kust tudengite projektid alla laaditakse.
- Projekti nime regulaaravaldis (*Project name regex*) – valikuline väli, mille abil saab filtreerida projektide nimesid. Kui see väli jääb tühjaks, võetakse arvesse kõik eelnevalt määratud grupi projektid.
- Asukoht (*Location*) – kohustuslik väli, mis määrab, millist tüüpi projekte kloonitakse: kas ainult isiklikke või ka jagatud projekte (*Projects* või *Shared Projects*).
- Faililaiendid (*File extensions*) – kohustuslik väli, kuhu sisestatakse kõik faililaiendid, mida plagiaadikontroll toetab. Sisestada võib mitu laiendit.

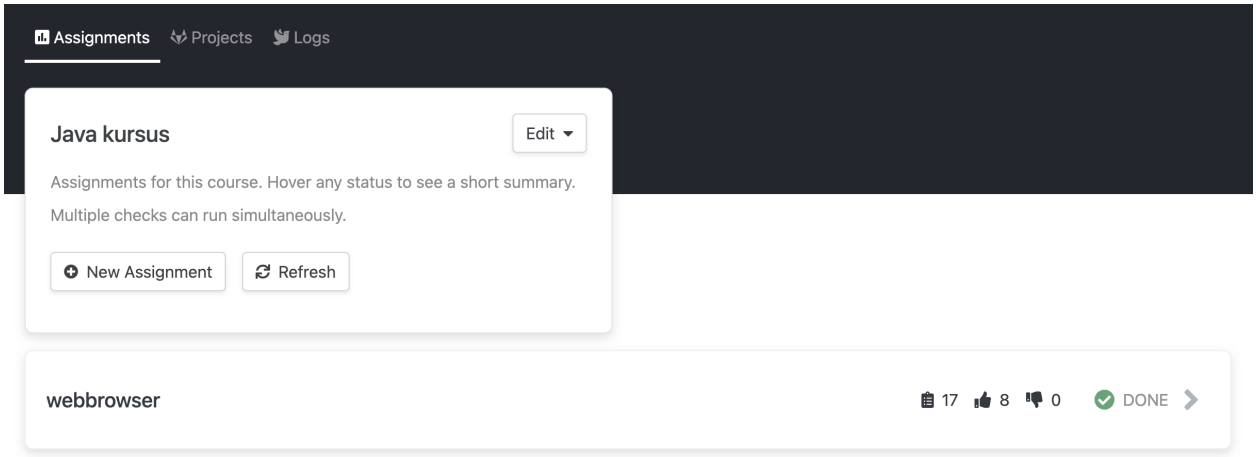
Mossi seaded

- Maksimaalne läbipääsevuste arv enne ignoreerimist (*Maximum number of passes before ignored*) – kohustuslik väli, mis määrab, mitu madala kattuvusprotsendiga vastet kaasatakse tulemustesse enne, kui need edaspidi ignoreeritakse. Vaikeväärtus on 10;
- Kuvatavate kattuvuste arv (*Number of matches shown*) – kohustuslik väli, mis määrab, mitu kattuvust maksimaalselt tulemustes kuvatakse. Vaikeväärtus on 25.

Kui kõik vajalikud andmed on vormi sisestatud, lisatakse uus kursus põhivaatesse. Kursuste andmeid saab uuendada, vajutades nupule *"Refresh"*. Iga kursuse nimel klõpsates suunatakse kasutaja vastava kursuse detailvaatesse.

2.1.3 Kursuse detailvaade

Kursuse detailvaates saab kasutaja hallata kõiki kursusega seotud ülesandeid ning uurida tudengite projektide ja kloonimisprotsessi logisid. Kursuse detailvaate sisuga saab tutvuda joonisel 3.



Joonis 3. Kursuse detailvaade koos ühe loodud ülesandega.

Vaade on jaotatud kolmeks sektsiooniks:

- **Ülesanded** (*Assignments*) – sektsioon, kus kuvatakse kõik kursuse ülesanded, mille jaoks saab läbi viia plagiaadikontrolli.
- **Projektid** (*Projects*) – sektsioon, mille kaudu on võimalik kloonida kõik kursuse raames loodud tudengite projektid. Kui projektid on kloonitud, kuvatakse need

samas vaates. Iga tudengi kohta on olemas link tema GitLabi projektile. Samuti saab vajutada tudengi Uni-ID-l, mis viib kasutaja vastava tudengi profiilivaatesse.

- **Logid** (*Logs*) – sektsioon, kus kuvatakse kloonimisprotsessi logisid, mis võimaldavad jälgida kloonimise edenemist ja näha, milliste tudengite projektid on juba alla laaditud.

Ülesannete sektsioonis on komponent, mille kaudu saab lehte värskendada, et näha viimaseid muudatusi. Samuti on võimalik selle kaudu muuta kursuse seadeid ning lisada või eemaldada kursuse õpetajaid.

Seadete muutmisel avaneb vorm, mis on identne kursuse loomisel täidetud vormiga. Kasutaja saab seal muuta kõiki soovitud andmeid. Vajutades nupule *"Save"*, salvestatakse tehtud muudatused.

Uue ülesande lisamiseks on olemas eraldi nupp, mis avab vastava vormi. Vormis tuleb sisestada järgmised andmed:

- **Ülesande nimi** (*Name*) – kohustuslik väli, mille kaudu määratakse ülesande nimi. Nimi peab olema unikaalne selle kursuse kontekstis.
- **Ülesande tee** (*Assignment path*) – kohustuslik väli, mis määrab ülesandega seotud faili asukoha. Seda kasutatakse ülesannete lahenduste leidmiseks ja töötlemiseks plagiaadikontrolli käigus.
- **Charoni identifikaator** (*Identifier in Charon*) – valikuline väli, mille abil saab ülesande seostada Charoni süsteemiga.

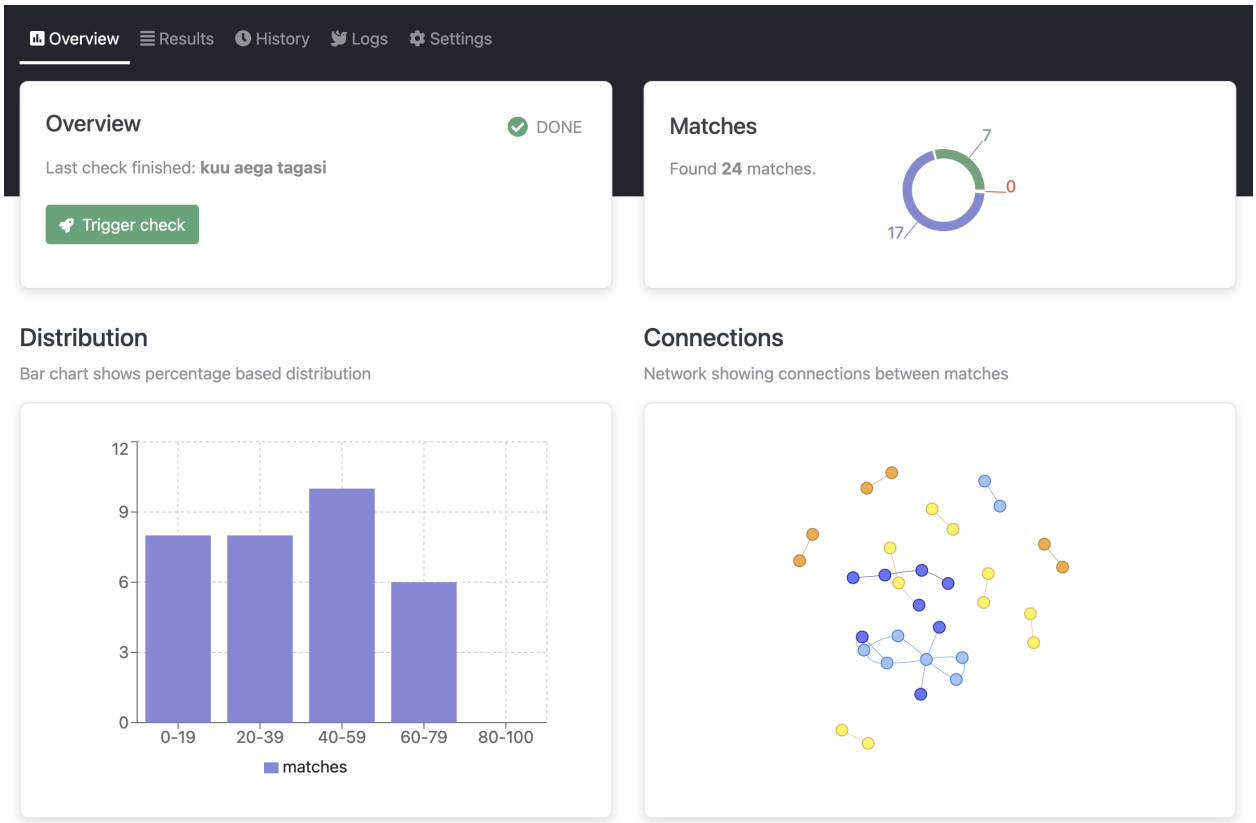
Pärast vormi täitmist lisatakse kursusele uus ülesanne. Iga ülesande juures kuvatakse selle olek: kas tegemist on uue ülesandega, kas plagiaadikontroll on pooleli, lõppenud edukalt või katkestatud veaga. Samuti on ülesande juures näha, mitu kattuvust on kinnitatud, mitu vajab ülevaatamist ja mitu on märgitud plagiaadina.

Vajutades ülesandele, suunatakse kasutaja vastava ülesande detailvaatesse.

2.1.4 Ülesande vaade

Ülesande vaates saab kasutaja tutvuda ülesande detailidega ning vaadata plagiaadikontrollide tulemusi. Vaade on jaotatud viieks sektsiooniks, millele vastavad eraldi komponendid rakenduses:

- **Ülevaade** (*Overview*) – peamine sektsioon (Joonis 4), kus on võimalik käivitada Mossi plagiaadikontroll ning vaadata selle tulemuste visuaalset kokkuvõtet.
- **Tulemused** (*Results*) – sektsioon, kus kuvatakse viimase plagiaadikontrolli tulemused. Tulemusi saab filtreerida kattuvuse staatuse järgi (*"All"*, *"New"*, *"Acceptable"*, *"Plagiarism"*) või otsida tudengite nime alusel.
- **Ajalugu** (*History*) – sektsioon, kus on kuvatud kõik varasemad plagiaadikontrolli tulemused vastava ülesande kohta. Tulemustes navigeerimine toimub kuupäeva alusel.
- **Logid** (*Logs*) – sektsioon, kus on näha plagiaadikontrolli käigus tekkinud logid. Need aitavad jälgida protsessi kulgu ning annavad märku, kas kontroll õnnestus või katkes vea tõttu.
- **Seaded** (*Settings*) – sektsioon, kus saab muuta ülesande konfiguratsiooni, sealhulgas ülesande nime, tee, Mossi seadeid ja faililaiendeid.



Joonis 4. Ülesande vaade plagiaadikontrolli tulemustega.

Praegu on Moss ainus teenus, mille abil saab plagiaadikontrolli rakenduses läbi viia. Kuna Mossi server võib aeg-ajalt olla kättesaamatu, ei ole sellisel juhul võimalik kontrolli teostada, mis piirab rakenduse funktsionaalsust. Seetõttu on oluline leida alternatiivseid

lahendusi, mida saaks vajadusel Mossi asemel või sellele lisaks kasutada. Selleks tuleb analüüsida sobivaid programme ja hinnata nende liidestamise võimalusi rakendusega.

Kui ülesandele on vähemalt üks plagiaadikontroll juba läbi viidud, kuvatakse ülevaate sektsioonis visuaalsed graafikud viimaste tulemuste kohta. Näha on, kui palju kattuvusi tuvastati ning kui suur osa neist on aktsepteeritud, märgitud plagiaadina või ootel ülevaatamist. Lisaks esitatakse tulpdiagramm, mis jaotab kattuvused protsendivahemikesse, ning graafidiagramm, mis näitab tudengite lahenduste omavahelisi kattuvusi.

Ülesande täielikuks kustutamiseks saab kasutada seadete sektsioonis olevat kustutamisvõimalust.

2.1.5 Tudengi profiili vaade

Tudengi profiilivaates kuvatakse detailne teave tudengi kohta, sealhulgas tema nimi, Uni-ID ning kõikide tuvastatud plagiaadijuhtumite koguarv.

2.1.6 Kasutaja seaded

Kasutaja seadete vaates saab kasutaja lisada või muuta oma GitLabi ligipääsu tokeni väärtust, mis võimaldab tal pääseda GitLabi projektidele.

2.2 Plagiaadikontrolli programmid

Järgnevas alapeatükis analüüsitakse erinevaid plagiaadikontrolli programme, et saada ülevaade olemasolevatest lahendustest ja võimalustest, mida saaks lõputöö raames arendatava rakenduse külge liidestada.

Plagiaadikontrolli programmide eesmärk on võrrelda tudengite programmeerimisülesannete lahendusi ning tuvastada võimalikud plagiaadi juhtumid. Erinevad programmid kasutavad erinevaid algoritme ja tehnoloogiaid, et määrata lahenduste sarnasuse protsent ning leida mustreid, mis viitavad kopeerimisele.

2.2.1 Sherlock

Sherlock on plagiaadituvastuse programm, mis võimaldab omavahel võrrelda tekstidokumente ja programmikoodi faile [4].

Programmil puudub oma server, mistõttu tuleb see esmalt alla laadida ja seadistada. See tagab, et programm töötab samas keskkonnas koos projekti teiste teenustega ning on pidevalt kättesaadav. Samuti ei sõltu see välisest serverist, mis võiks Tallinna Tehnikaülikooli serveritest väljaspool töötades ootamatult katkeda.

Sherlock on käsureapõhine rakendus, millel puudub graafiline kasutajaliides. See pakub siiski piisavalt funktsionaalsust vajalike failide analüüsimiseks ja tulemuste kuvamiseks. Kasutaja saab määrata, millist tüüpi faile kontrollida ning kohandada analüüsi seadeid. Seadistustes on võimalik määrata sarnasuse lävendit, mille alusel failid plagiaadina tuvastatakse, täpsusastet ning väljundfaili tüüpi.

Programm on eriti kasulik, kuna suudab võrrelda nii Pythoni kui ka Java koodifaile – mõlemad on laialdaselt kasutusel mitmes Tallinna Tehnikaülikooli õppeaines.

Tegemist on hea alternatiivse teenusega, mida saab kasutada tagavara teenusena.

2.2.2 JPlag

JPlag on plagiaadituvastuse programm, mis on allalaaditav ja toetab mitmeid programmeerimiskeeli. See võimaldab kontrollida Java ja Pythoni koodi ning samuti C++ ja C Sharp, mis on kasutusel mitmetes Tallinna Tehnikaülikooli õppeainetes [5].

JPlagi saab kasutada käsurealiidese (CLI) kaudu, käivitades vastava JAR-faili, või programmiliselt Java API abil. CLI võimaldab JPlagi lihtsalt integreerida erinevatesse automatiseeritud töövoogudesse, samas kui Java API võimaldab plagiaadituvastust lisada otse teistesse Java-põhistesse süsteemidesse.

Programmi tugevuseks on põhjalik koodi analüüs, mis tagastab kahe lahenduse võrdlemise puhul täpsema tulemuse. Samuti pakub rakendus oma poolt välja võimaluse saata plagiaadikontrollile failid, mille sisu mitte arvesse võtta. Niisugustesse failidesse saaks kirja panna ülesande malli, et seda ei arvestataks kattuvusele juurde.

2.2.3 SIM

SIM on plagiaadituvastus programm, mis on võimeline kontrollima plagiaati nii tavalistel tekstidokumentidel kui ka programmikoodi failidel [6].

Programm toetab programmeerimiskeeli nagu näiteks C ja Java. Iga programmeerimiskeele jaoks on eraldi programm, mida tuleb jooksutada, et kontrollida kas spetsiifilise programmeerimiskeelega kirjutatud programmi koodi või teksti dokumenti.

Programmi tugevuseks on plagiaadi kontrolli tase. Iga programm, mis on eraldi loodud, keskendub just ette nähtud programmeerimiskeelele, mistõttu oskab see arvesse võtta süntaksi spetsiifilisi omadusi, mille järgi koodide sarnasusi tuvastada.

Programmi nõrkuseks on programmide arvukus. Kuna iga programmeerimiskeele ja tekstidokumendi kontrollimiseks on eraldi programm, muutub liidestamine selle arvelt tülikamaks, kui soovi teha plagiaadikontrolle erinevate programmeerimiskeeltega kirjutatud lahendustele. Samuti ei toeta programm Pythoni programmerrimiskeelt.

2.2.4 OSPC

OSPC on plagiaadituvastus programm, mis kontrollib omavahel programmikoodi faile ning on võimeline tagastama mitmetes vormides tulemusi [7].

Programm toetab C põhiseid programmeerimiskeeli, nagu Java ja C++, aga on laiendatav ka mõne muu programmeerimiskeele jaoks. See tähendaks, et antud programmi tuleks edasi arendada, et see toetaks ka Pythoni keelt. See osutub selle programmi jaoks suureks nõrkuseks, kuna olemas on muid plagiaaditeenuseid, mis juba toetavad Pythonit ja on selle arvelt lihtsam liidestada.

Programmi tugevusteks on, et sellel on nii graafiline kui ka käsurea liides, mistõttu seda saab mitut moodi kasutada. Samuti on võimalik muuta seadeid, kuidas tarkvara plagiaadikontrolli teeb ning programm on võimeline tuvastama gruppe sarnaste koodilahendustega.

2.2.5 ACNP

ACNP on sarnane plagiaadituvastusprogramm eelnevalt välja toodud programmidega [8].

Programm ei toeta Pythoni programmeerimiskeeli ning on piiratud funktsionaalsustega, kui alla laadida tasuta versioon. Vastasel juhul tuleb tarkvara eest maksta, mis ei ole lõputöö kirjutajate poolt eelistatud. Seda eriti, kui olemas on rakendused, millel vaba tarkvara kõigi funktsionaalsustega.

2.2.6 Dolos

Dolos on plagiaadituvastusprogramm, mis pakub mitmeid metoodikaid, kuidas plagiaadikontrolli teha. Programm toetab kõiki soovitud programmeerimiskeeli, nagu Python, Java ning C++ [9].

Dolost on võimalik kasutada rakendusliidese (API) kaudu või laadides all käsurea programmina. Samuti saab tarkvara seadsitada Docker Compose-i kaudu. Seetõttu tundub, et liidestuses ei ole midagi keerulist ning tarkvara saab mugavalt kasutada plagiaadi kontrolliks.

2.2.7 AC

AC on lokaalselt töötav plagiaadituvastusvahend, mis ei vaja serveriühendust. See toetab mitmeid programmeerimiskeeli, sealhulgas C, Java ja Python [10].

Plagiaadikontrolli teostamiseks kasutab AC erinevaid algoritme, millest peamine on normaliseeritud survekauguse meetod (*Normalized Compression Distance*). See lähenemine on efektiivsem kui traditsioonilised tokenipõhised algoritmid ning sobib hästi mitme programmeerimiskeele korraga töötlemiseks [10].

Programm kuvab tulemusi graafikutena, visualiseerides saadud plagiaadituvastuse andmed. Siiski nõuab AC Java arenduskeskkonna olemasolu, mis muudab selle integreerimise keerukamaks. Erinevalt JPlagist puudub AC-l demonstreeriv veebi-liides, kus oleks võimalik näite tulemusi sirvida ja hinnata.

2.2.8 Plaggie

Plaggie on plagiaadituvastusprogramm, mis toetab ainult Java programmeerimiskeelt, on lokaalselt alla laaditav ning pakub graafilist kasutajaliidest [11].

Kahjuks jääb ainult ühest programmeerimiskeele toetusest väheks, mistõttu ei ole antud tarkvara lõputöö projektile kasulik.

2.3 Lahenduste analüüs ja kokkuvõte

Olemasolev rakendus sisaldab mitmeid funktsionaalsusi, mis on õppejõududele kasulikud. Nende abil saavad nad luua kursuseid ning lisada neile ülesandeid. Iga ülesande puhul on võimalik käivitada plagiaadikontroll ning analüüsida saadud tulemusi lähemalt. Tudengi tasemel on võimalik vaadata individuaalset profiili ja saada ülevaade, kui palju plagiaadijuhte tudengil on esinenud.

Iga plagiaadikontrollist saadud kattuvuse kohta saab õppejõud määrata, kas tegemist on lubatud kattuvuse või plagiaadijuhtumiga. Samuti on võimalik uurida kattuvusi detailvaates, kus tudengite koodilahendused kuvatakse kõrvuti.

Kuigi rakendus toimib üldjoontes hästi, esineb mitmeid kitsaskohti, mille lahendamine muudaks kasutuskogemuse sujuvamaks ja tõhusamaks.

Üheks peamiseks probleemiks on keerukas navigeerimine – tudengi profiilivaatele juurdepääsuks tuleb liikuda läbi kursusevaate ning seejärel tudengite projektide sektsiooni. Lisaks puudub peavaates teave selle kohta, millised plagiaadikontrollid on hetkel aktiivsed, mistõttu peab õppejõud neid eraldi kontrollima. See muudab töövoo vaevaliseks, eriti olukorras, kus soovitakse kiiresti leida ja analüüsida konkreetseid tudengeid, kelle tööd kattuvad plagiaadikontrolli tulemustes.

Tudengi profiilivaade ise sisaldab piiratud hulgal informatsiooni ega paku õppejõule piisavat ülevaadet selle kohta, milliste ülesannete puhul on plagiaat aset leidnud ja kui ulatuslik see on olnud. Kasutajakogemust parandaks visuaalsete graafikute lisamine, mis aitaksid paremini mõista plagiaadi ulatust ning tuvastada, milliste kursusekaaslastega sarnasused esinevad.

Et aidata õppejõul kiiremini otsustada, milliste tudengite profiile lähemalt uurida, oleks otstarbekas lisada uus vaade, mis kuvab üldise statistika, sealhulgas tudengi tuvastatud plagiaatide koguarvu. See muudaks navigeerimise rakenduses oluliselt tõhusamaks.

Praegu saab plagiaadikontrolli teostada ainult ühe semestri kursuse lõikes, mistõttu ei ole võimalik võrrelda töid eri aastakäikude vahel. Sellest tulenevalt võivad mõned plagiaadijuhud jääda avastamata, näiteks juhul, kui tudeng kopeerib varasema aasta lahendust. Võimalus teostada kontrolli mitme kursuse vahel suurendaks plagiaadi tuvastamise täpsust.

Hetkel toetab rakendus vaid üht plagiaadikontrolli teenust – Moss. Kuna Mossi server võib olla aeg-ajalt kättesaamatu, ei ole võimalik uute ülesannete puhul kontrolli läbi viia ning kasutajal jääb üle vaid olemasolevaid tulemusi vaadata. Selle piirangu ületamiseks tuleks rakendusse liidestada täiendavaid plagiaadituvastuse tööriistu, mis võimaldaksid kasutada alternatiivseid allikaid.

Samuti tasub arvestada, et iga plagiaadituvastustööriist võib anda eksitavaid tulemusi. Mitme teenuse paralleelne kasutamine aitaks neid tulemusi omavahel võrrelda ning viia läbi täpsemaid ja usaldusväärsemaid analüüse.

Arvestades, et rakenduse esmakordne kasutamine võib uute kasutajate jaoks olla keerukas, oleks soovitatav lisada põhjalik kasutusjuhend. See oleks abiks õppejõududele ja abiõppejõududele, kellel puudub varasem kogemus rakendusega. Kasutusjuhendi abil saavad nad tutvuda funktsionaalsustega ning alustada süsteemi tõhusat kasutamist juba esimestel kordadel.

Plagiaadituvastusprogrammide valikul lähtuti kolmest peamisest kriteeriumist:

- Milliseid programmeerimiskeeli konkreetne programm toetab.
- Kui keeruline on programmi liidestamine lõputöö raames arendatava rakendusega.
- Kui täpsed ja usaldusväärsed on programmi poolt tagastatud plagiaadikontrolli tulemused.

Arvestades seatud kriteeriume ning analüüsides erinevaid plagiaadituvastusprogramme, otsustati arendatava rakenduse külge liidestada Sherlock ja JPlag. Mõlema tööriista tagastatud tulemused on põhjalikud ja sisukad. Lisaks toetavad mõlemad programmid neid programmeerimiskeeli, mis on laialdaselt kasutusel Tallinna Tehnikaülikooli programmeerimisainetes.

Mõlemal teenusel on ka omad tugevused, mis tegid nad eriti sobivaks: Sherlock võimaldab määrata kattuvuslävendi, mille alusel madalama protsendiga kattuvusi tulemustes ei kajastata. See aitab keskenduda ainult olulisematele kattuvustele ning muudab plagiaadikontrolli protsessi tõhusamaks. JPlag seevastu pakub omaenda veebipõhist kasutajaliidest, mis võimaldab plagiaadikontrolli tulemusi visuaalselt ja detailselt uurida. See on eriti kasulik ülesannete puhul, kus lahendus koosneb mitmest failist – JPlagi vaates on kõik failid selgelt

eristatavad, võimaldades paremini tuvastada kattuvusi eri kohtades.

Tänu nendele eripäradele saavad õppejõud plagiaadikontrolli teostada paindlikumalt ning tulemusi analüüsida tõhusamalt ja mugavamalt.

3 Nõuded edasiarenduseks

Järgmises peatükis esitatakse kõik nõuded, mis tuleb täita olemasoleva projekti edasiseks arendamiseks. Nõuded on jaotatud funktsionaalseteks ja mittefunktsionaalseteks. Nõuete määratlemisel on lähtutud olemasoleva rakenduse ja plagiaadiprogrammide analüüsist ning arvesse on võetud kliendi ootused ja vajadused, et tagada arendustöö eesmärgipärasus.

3.1 Funktsionaalsed nõuded

Funktsionaalsed nõuded määratlevad rakenduse põhifunktsioonid, mis on vajalikud selleks, et kasutajakogemus oleks mugav, sujuv ja eesmärgipärane. Samuti sisaldavad need nõuded funktsionaalsusi, mis on rakenduse üldiseks kasutatavuseks hädavajalikud. Alljärgnevalt on esitatud funktsionaalsed nõuded, mis on olulised olemasoleva süsteemi edasiarendamisel.

1. Tudengi plagiaatide diagrammid ja graafikud
    - visualiseeritakse tudengi poolt sooritatud plagiaadijuhtumid;
    - kuvatakse seosed tudengi plagiaadisoorituste ulatuse ja teiste kursusekaaslaste tulemuste vahel;
2. Tudengitega seotud vaadete täiendused
    - kuvatakse kursuse kõigi tudengite nimekiri koos vastavate plagiaadisoorituste koguarvuga;
    - iga tudengi Uni-ID-d vajutades viiakse vastava profiilivaate juurde;
    - tudengi profiilivaates on detailne ülevaade kõigist sooritatud plagiaadijuhtumitest koos visuaalsete diagrammide ja graafikutega;
3. Plagiaadikontrolli protsesside ülevaade
    - kasutaja saab põhivaates jälgida kõiki pooleliolevaid plagiaadikontrolli protsesse;
    - protsessid on struktureeritud kursuste ja konkreetsete ülesannete alusel;
4. Eelmiste aastate lahenduste arvestus
    - kursuse loomisel on võimalik lisada varasemate aastate projekte;

- mitme aasta kursuste andmeid saab võrrelda plagiaadikontrolli tulemuste põhjal;

5. Plagiaadikontrolli teenused
   - kasutajal on võimalus kontrollida plagiaati mitme erineva teenuse abil;
   - tulemusi saab omavahel võrrelda, kasutades erinevate teenuste väljundeid;
6. Kasutusjuhend
   - rakenduse kasutamiseks on kättesaadav põhjalik kasutusjuhend;
   - juhendis on kõik olulisemad teemad loogiliselt ja selgelt esitatud;

3.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded kirjeldavad tingimusi ja omadusi, mis peavad olema täidetud selleks, et rakenduse funktsionaalsed nõuded toimiksid usaldusväärselt ja tõhusalt. Need nõuded toetavad süsteemi üldist kvaliteeti, kasutatavust ja töökindlust. Alljärgnevalt on esitatud mittefunktsionaalsed nõuded, mis on olulised süsteemi edasiarendamisel.

1. Plagiaadikontrolli teenused peavad töötama taustülesannetena, et võimaldada plagiaadikontrolli protsesside asünkroonset käivitamist ja toimimist
2. Tudengite teabele peab olema lihtne ja kiire ligipääs, et tagada tõhus andmekasutus
3. Kursusega seotud teave peab olema kättesaadav üksnes kasutajatele, kellele on määratud õpetaja roll kursuses
4. Administraatoritel peab olema juurdepääs kõikide kursuste teabele sõltumata nende seotusest konkreetse kursusega

4 Projekti ülesehitus

Rakendus koosneb tagarakendusest (*backend*) ja eesrakendusest (*frontend*), mis suhtlevad omavahel ning talletavad andmeid andmebaasides.

Käesolevas peatükis analüüsitakse rakenduse tehnilist poolt ning käiakse läbi kõik olulised komponendid, mis rakenduse projektist loob ühtse terviku. Tutvustatakse projektis kasutatud tehnoloogiaid, mis tagavad rakenduse toimimise ja võimaldavad selle edasist arendamist.

4.1 Rakenduse tehniline analüüs

Käesolev alapeatükk tutvustab kõiki rakenduse funktsionaalsusi, mille kaudu kasutaja saab eesrakenduses navigeerida ning andmeid saata või pärida.

Rakendus on üles ehitatud modulaarse arhitektuurina, kus erinevad funktsionaalsused on jagatud väiksemateks rakendusteks (mooduliteks). Iga mooduli roll ja vastutus projekti kontekstis on selgelt määratletud, mis muudab kogu süsteemi edasiarendamise struktuursemaks, arusaadavamaks ja lihtsamini hallatavaks.

Iga mooduli toimimist toetavad eraldi kirjutatud testid, mis aitavad tagada rakenduse töökindluse ja vastavuse seatud nõuetele. Samuti on loodud migratsioonifailid, mille abil ehitatakse üles rakenduse andmebaas. Andmebaas salvestab kogu vajaliku teabe, mida kasutajad saavad süsteemis vaadata, muuta või töödelda.

Järgnevalt on iga mooduli funktsionaalsus ja roll süsteemis eraldi selgitatud.

4.1.1 Kasutajad

Kasutajate moodul tegeleb süsteemi kasutajate andmete ja õigustega. Kõik kasutajatega seotud funktsionaalsused on koondatud kataloogi *accounts*.

Andmebaasimudelisse on defineeritud kasutajaobjekt, mis talletab iga kasutaja kohta olulised andmed, nagu kasutajanimi, konto loomise kuupäev ja kasutajarollid. Rollide alusel

määratakse, millised õigused igal kasutajal süsteemis on. Selleks on loodud spetsiaalsed meetodid, mis kontrollivad kasutajaõigusi ja aitavad otsustada, millist infot ning millises mahus konkreetne kasutaja näha võib.

Moodulis on defineeritud mitmed vormid, mille abil saavad kasutajad end süsteemi autentida, parooli vahetada ning hallata muid isikuandmeid. Nende tegevuste teostamiseks on loodud vastavad vaated, mis võimaldavad vormide kuvamist ja töötlevad kasutaja poolt sisestatud andmeid. Et neid vaateid oleks võimalik veebiliidese kaudu kasutada, on määratud ka vastavad URL-aadressid.

Moodul võimaldab hallata kasutajate identiteeti ja ligipääsuõigusi, pakkudes selleks vajalikke vorme, vaateid ning andmestruktuure ühtse ja selge arhitektuuriga.

4.1.2 Charon

Charon on pistikprogramm, mida kasutatakse Tallinna Tehnikaülikooli programmeerimisainetes ülesannete automaatseks testimiseks, hindamiseks ja kaitsmiseks. See suhtleb plagiaadikontrolli rakendusega, et saada vastavad tulemused [12].

Charoniga seotud koodifailid on koondatud kataloogi *charon*, kus on defineeritud kõik vajalikud URL-aadressid ühenduse loomiseks ja andmete vahetamiseks. Lisaks on implementeeritud päringufunktsioonid, mis võimaldavad hankida ning uuendada kursuste ja ülesannetega seotud andmeid ning kuvada tudengite koodide kattuvusi plagiaadi tuvastamiseks.

4.1.3 Kursused

Kursuste moodul haldab kogu olulist teavet kursuste ja nendega seotud ülesannete kohta. Mooduliga seotud failid on koondatud kataloogi *courses*.

Andmebaasis on defineeritud objektid, mis kirjeldavad kursusi, kursuseõpetajaid, ülesandeid ja tudengeid. Nende objektide abil hallatakse infot selle kohta, millised kursused on loodud, millised ülesanded neile kuuluvad, kes on kursusele registreerunud ning millised õpetajad on vastutavad. Õpetajatele on tagatud ligipääs kursuse plagiaadikontrolli tulemustele ja tudengite andmetele.

Iga andmebaasi objekti jaoks on loodud vastavad serialiseerijad, URL-aadressid ja vaated, mille kaudu saab kursusega seotud andmeid küsida, esitada ja töödelda.

4.1.4 Peamine moodul

Peamise mooduli kood asub kataloogis *iapb*.

Selles moodulis paiknevad funktsioonid, mis vastutavad tudengiprojektide kloonimise ja uuendamise eest, et need oleksid valmis saatmiseks plagiaadikontrolli teenusele. Iga plagiaadikontrolli protsessi jaoks on olemas spetsiaalsed funktsioonid, mis haldavad töövoogu ja logivad olulist infot – näiteks millised projektid on kontrolli alla võetud, kas protsess kulges edukalt või esines vigu.

Mossi teenusega suhtlemiseks on loodud eraldi funktsioonid, mis tegelevad projektide edastamisega, tulemuste pärimisega ning võimalike vigade käsitlemisega. Kõik tõrked logitakse kasutajasõbralikult, et anda selget tagasisidet toimunud protsessi kohta.

Plagiaadikontrollide asünkroonseks käivitamiseks on moodulis defineeritud spetsiaalne Celery-klass, mis haldab protsesside järjekorda ja täitmist. Selle abil jaotatakse kõik veebirakenduses toimuvad tööprotsessid optimaalselt, tagades süsteemi sujuva ja tõhusa toimimise.

4.1.5 Microsoft autentimine

Microsofti kasutajate jaoks on loodud eraldi autentimisfunktsioonid, mis asuvad kataloogis *microsoft auth*.

Seal on defineeritud Microsofti kasutaja mudel ning funktsioonid, mis võimaldavad kasutaja andmete kätte saamist ja verifitseerimist. Samuti on sarnased funktsioonid loodud Xboxi kasutajate jaoks.

Andmebaasis on ära defineeritud objektid Microsofti kasutajate ja Xboxi kasutajate jaoks.

4.1.6 Moss

Moss on veebipõhine plagiaadikontrolli teenus, mis suudab võrrelda koodifaile mitmes erinevas programmeerimiskeeles, näiteks Pythonis ja Javas [13].

Selle teenusega seotud rakenduse failid paiknevad kataloogis *moss*. Seal on defineeritud Mossi kliendiklass, kuhu on koondatud kõik vajaminevad andmed teenusega ühenduse loomiseks, sealhulgas toetatud programmeerimiskeeled, mille failidega Moss oskab töötada.

Lisaks on loodud spetsiaalne klass kattuvusandmete haldamiseks, mis esitab teavet koodi kattuvuse ulatuse ning kattuvate ridade kohta. Mossi tulemuste esitamiseks on eraldi klass, mille abil saab detailset infot tuvastatud plagiaadi, kattuvusprotsentide ning kahtluse alla sattunud tudengite kohta.

Oluline on märkida, et Moss on hetkel ainus plagiaadikontrolli teenus, mis on rakendusega integreeritud. Seetõttu ei ole kontrollide läbiviimine alati võimalik, kui teenuse server pole saadaval.

4.1.7 Plagiarism

Plagiaadikontrolli protsesside, tudengite GitLabi projektide ja kontrollitulemuste haldamiseks on loodud spetsiaalne moodul, mille failid asuvad kataloogis *plagiarism*. Moodulis on defineeritud mitmed andmeobjektid, mis toetavad süsteemi tõhusat ja läbipaistvat toimimist.

Kattuvuste andmemudelid

Plagiaadikontrolli tulemuste salvestamiseks ja analüüsimiseks on loodud järgmised objektid:

- kattuvus (*match*) - viitab kahele tudengile, kelle lahenduste vahel leiti kattuvus, ning sisaldab teavet kattuvuse ulatuse kohta protsentides;
- kattuvad read (*match similarity*) - määratleb koodisegmendid, mis kattusid kahe lahenduse vahel;
- kattuvuse staatuse kommentaar (*match status update comment*) - talletab õpetaja lisatud selgitused, kui kattuvuse staatus on ümber määratud (“uus“, “vastuvõetud“ või “plagiaat“).

Plagiaadiprotsessi andmemudelid

Iga plagiaadikontrolli protsess on seotud järgmiste objektidega:

- plagiaadi protsess (*plagiarism run*) - esindab konkreetset kontrolli, määrates ajahetke ja ülesande, mille kohta kontroll käivitati;
- kõik protsessi kattuvused (*plagiarism run matches*) - koondab kõik antud kontrollis

tuvastatud kattuvused;

- protsessi staatus (*plagiarism run status*) - kirjeldab kontrolli olekut (kas see on käimas, lõppenud edukalt või katkestatud veaga).

Projektide ja lahenduste andmemudelid

Plagiaadikontrolli alla kuuluvad tudengite projektid ja lahendused on kirjeldatud järgmiste objektidega:

- tudengi projekt (*project*) - seob konkreetse tudengi konkreetse kursusega;
- projetki GitLabi andmed (*GitLab project*) - sisaldab tehnilisi üksikasju, nagu projekti URL ja muudatuste arv (commits);
- ülesande lahendus (*submission*) - viitab tudengi esitatud koodile, mis saadeti kontrolli.

Kõigi objektide andmete kätte saamiseks ja töötlemiseks on valmis tehtud serialiseeriad, URL-aadressid ja vaated.

Antud moodul võimaldab hallata kogu plagiaadikontrolli elutsüklit – alates projektide kloonimisest kuni lõplike tulemusteni. See seob iga protsessi loogiliselt kursuste ja tudengitega. Tänu sellele on tagatud loogiliselt mõistetav andmestruktuur ja lihtne navigeeritavus kogu plagiaadikontrolli süsteemis.

4.1.8 Seaded

Projektiga seotud konfiguratsioonifailid asuvad kataloogis *settings*. Selles kataloogis on koondatud kõik olulised moodulid, mida rakendus kasutab, et tagada nende korrektne registreerimine ja kasutamine. Sealhulgas on määratletud:

- andmebaasi ühenduse seaded,
- Mossi plagiaadikontrolli teenusega seotud andmed,
- muud vajalikud konfiguratsioonid rakenduse käivitamiseks ja stabiilseks toimimiseks.

See struktuur tagab, et kogu süsteem töötab kooskõlas ning kõik vajalikud komponendid on omavahel korrektselt seotud.

4.2 Tagarakenduse tehnoloogiad

Tagarakenduses on kasutatud paindlikke raamistikke, mis võimaldavad veebirakenduse arhitektuuri mugavat ja tõhusat arendamist. Järgnev alapeatükk räägib nendest lähemalt.

4.2.1 Django

Antud projekti tagarakendus on kirjutatud Django raamistikuga, mis kasutab Pythoni programmeerimiskeelt [14].

Antud raamistik osutus valitud tehnoloogiaks, kuna senine veebirakendus on kirjutatud sellega ning tegemist on mugava raamistikuga, millel on mitmeid eeliseid.

Django on disainitud veebiülesandeid täitma kiiresti ja lihtsasti. Djangot on võimalik kasutada ilma andmebaasita, aga tema üheks suureks tugevuseks on ORM. Tänu sellele on võimalik mugavalt andmebaasi mudel valmis disainida Pythoni koodiga.

Djangol on olemas lihtsasti mõistetav ja loogiline struktuur, millega koodi jaotada failidesse ja kaustadesse ning igas kaustas valmis kirjutada funktsionaalsus, mis täidab kindlat ülesannet projektis. Saab eraldi ära defineerida kasutatavad URL-d, mille suunas päringuid teha ning eraldi failidesse kirjutada funktsioonid, mis päringuid saadavad ja vastu võtavad.

Omalt poolt pakub ta ka välja valmis tehtud administraatorite rakendust, kus rakenduse peakasutajad saavad lisada, muuta ja kustutada objekte. Selle jaoks on tarvis mudel registreerida adminni mooduli koodis.

4.2.2 Celery

Celery on järjekorrapõhine asünkroonne töötlussüsteem, mida rakendus kasutab plagiaadikontrollide planeerimiseks ja käivitamiseks. Lisaks võimaldab see kloonida ja uuendada tudengite projekte. Tänu Celeryle saab mitmeid protsesse samaaegselt käivitada, mis muudab rakenduse töökindlamaks ja tõstab selle kasutusmugavust ning tõhusust.

4.2.3 Pytest

Pytest on Pythoni raamistik testide kirjutamiseks. Seda kasutatakse projekti funktsionaalsuste testimiseks, et veenduda nende toimimisest.

Pytest pakub mugavat viisi testide kirjutamiseks. Funktsiooni testimiseks on põhiliselt vaja kasutada *assert* lauseid, mis võrdlevad funktsiooni poolt valmistatud andmeid oodatud andmetega. Et teste oleks selgem ja arusaadavam lugeda ning et ei tuleks liiga palju duplikaatkoodi, on võimalik kirjutada ka *setup* meetodeid, kus defineeritakse ära mitmed vajalikud muutujad, millega tehakse katsetusi mitmetes testides.

Samuti saab kirjutada ka funktsioone, mis tagastavad andmebaasi objektide näiteid, millega teste läbi tehakse. Siis on kõik vajalikud testi komponendid kokku pandud eraldi funktsioonidesse.

Kõik testid on paigutatud eraldi test kataloogi projekti failide seas, et neid oleks muude failide seast lihtne üles leida ja kätte saada.

4.3 Eesrakenduse tehnoloogiad

Eesrakendus on kirjutatud React raamistikuga, mis pakub oma poolt välja mugavaid viise HTMLi, CSSi ja JavaScripti koodi kirjutamiseks. Järgnev peatükk räägib antud raamistikust lähemalt.

4.3.1 React

React on üks mitmest eesrakenduse liidestest, millega võimalik kirjutada koodi kasutajaliidese disainimiseks.

Kood on jaotatud mitmeteks komponentideks, millest iga üks tagastab oma HTMLi koodi, millele on juurde antud JavaScripti funktsioone ja CSSiga stiili. Komponendid on taaskasutatavad ning neid kombineerides võimalik mitmeid suuremaid lehekülgi kokku panna [15]. Niisugune komponentideks jagamine aitab hoida projekti struktuuri loogilisena ja lihtsam on koodi ja esinevaid vigu üles otsida [1].

Nendest eelistest olenevalt on otsutatud antud raamistikku kasutada projekti eesrakenduse arendamiseks. Lisaks kasutatakse Reacti sellepärast, kuna olemasolev projekt on arendatud antud raamistikuga ning lõputöö kirjutajatel on varasem kogemus tehnoloogiaga.

4.4 Andmebaas

Rakenduse andmeid hoiustatakse andmebaasis PostgreSQL. Peatükk räägib antud andmebaasist lähemalt.

4.4.1 PostgreSQL

PostgreSQL on andmebaasi süsteem, millega võimalik kirjutada komplekseid päringu käskluseid ja ehitada üles andmebaasi tabeleid. Süsteem on avatud lähtekoodiga ning toetab suurel määral SQL-standardi funktsionaalsust ning pakub mitmeid kaasaegseid võimalusi, mille seas välisvõtmed ja uuendatavad vaated [16].

Andmebaas on seotud Djangos ORM-ga loodud mudelitega, mis kirjeldavad ära kogu andmebaasi struktuuri. Nii on andmebaasis ära salvestatud kõik kursused, ülesanded, tudengid, kasutajad ja muud rakendusega seonduvad andmed. Andmebaasi mudeliga on võimalik täpsemalt tutvuda lisas 4.

4.5 Projekti haldus

Plagiaadi kontrolli süsteem koosneb mitmest tehnoloogiast ja teenusest. Seetõttu on tarvis kõik projekti osad omavahel kokku koondada ja hallata neid kui ühe tervikuna. Antud peatükk räägib lähemalt, mis selle jaoks kasutusel on.

4.5.1 Docker

Kogu projekti käivitamiseks ja jooksutamiseks kasutatakse Docker konteinereid. Kokku on kasutusel kuus konteinerit, millest igaüks jooksutab ühte projekti teenust. Näiteks Django konteiner hoiab enda sees projekti tagarakendust ning Node konteiner hoiustab kõik vajalikud teegid, mida esirakendusel on vaja. Nii toimivad teenused turvalisemalt ja ei häiri üksteie protsesse.

4.5.2 GitLab

Projekti arendamise vältel tehakse programmikoodist pidevalt uusi harusid, kuhu juurde kirjutada uut funktsionaalsust või teha parandusi varasemalt valmis kirjutatud koodile. Tehtud muudatused liidetakse pärast projekti põhiharuga. Kogu projekti halduseks on

kasutusel GitLab, kus saab luua uusi projekte, neid kloonida arenduskeskkonda ning luua projektist uusi harusi, kus juba projekti edasi arendada.

GitLab on samuti kasutusel, et kätte saada tudengite õppeaine repositooriumid, mis sisaldavad õppeaine jooksul loodud ülesannete faile, millele plagiaadikontrolli teha. Rakenduses saab tõmmata kursuse kohta kõigi tudengite projektid ning pärast võtta sealt soovitud failid, et need juba edasi saata kas Mossile või mõnele teisele teenusele plagiaadikontrolliks.

5 Tehniline teostus

Arenduse käigus on tehtud muudatusi ja täiendusi nii ees- kui ka tagarakenduses. Samuti on viimistletud andmebaasi objekte, et tagada lisatud funktsionaalsuste korrektne toimimine.

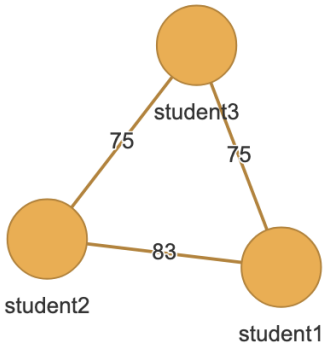
Käesolevas peatükis kirjeldatakse täpsemalt, millised funktsionaalsused on juurde arendatud ning kuidas neid kasutada.

5.1 Tudengi profiili vaate täiendused

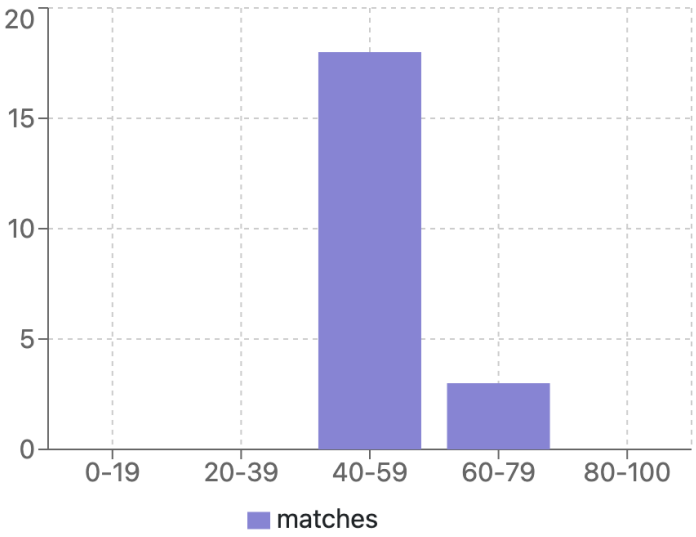
Varasemalt kuvati tudengi profiilivaates ainult tema nimi, Uni-ID ning plagiaadijuhtumite koguarv.

Sellele on nüüd lisatud visuaalseid komponente, mis pakuvad oluliselt detailsemat ülevaadet tudengi plagiaadikäitumisest.

- Tudengitevaheline seosegraafik (Joonis 5) visualiseerib valitud tudengi ja teiste tudengite vahelisi plagiaadiseoseid. Iga graafiku tipp tähistab tudengit (tema Uni-ID järgi), serv näitab lahendustevahelist sarnasuse protsenti ning tipu värvus viitab plagiaadi ulatusele. Nii on võimalik visuaalselt tuvastada sagedamini kattuvusi tekitavaid tudengeid.
- Plagiaadi jaotuse diagramm (Joonis 6) kuvab, kui suur osa tudengi töödest on märgitud plagiaadiks erinevates sarnasuse protsendivahemikes. See võimaldab kiiret hinnangut selle kohta, kui ulatuslikud plagiaadijuhtumid on olnud.
- Plagiaadiks märgitud tööde nimekiri kuvab kõik valitud tudengi tööd, mis on tuvastatud plagiaadina (Joonis 7). Igal real on esitatud kursus, ülesanne, teise tudengi Uni-ID (kelle lahendusega koos kattuvus plagiaadiks märgitud), kattuvate ridade arv ning kattuvuse loomise kuupäev. Kattuvuste ulatus on esitatud erinevates värvitoonides, mis aitavad kiiresti märgata suurema sarnasusega juhtumeid.



Joonis 5. Tudengi kattuvuste seos kursusekaaslastega.



Joonis 6. Tulpdiagramm tudengi kattuvuste jaotamiseks protsendi alusel.

Tööde nimekirjas on mitmed elemendid interaktiivsed:

- Kursuse nimele vajutades suunatakse kasutaja vastava kursuse vaatesse.
- Ülesande nimele vajutades avaneb ülesande tulemuste vaade.
- Teise tudengi Uni-ID-le vajutades avaneb tema profiilivaade.

- Silmaikooniga nupu kaudu avaneb detailne võrdlusvaade mõlema tudengi koodilahendustest.

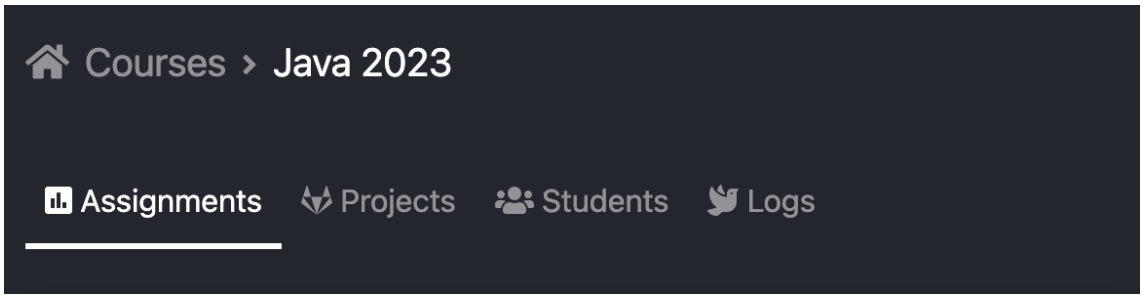
Course name	Assignment	Matched with	Lines matched	View details	Match was made
Java 2023	EX01	Student3	53 lines		12.04.2025, 16:45
Java 2023	EX01	student4	72 lines		13.04.2025, 13:15
Java 2023	EX01	student5	34 lines		14.04.2025, 11:20

Joonis 7. Tudengi kõigi kattuvuste nimekiri.

Kõik need lisandused aitavad paremini navigeerida plagiaadijuhtumite vahel, võimaldavad kiiremat ja täpsemat analüüsi ning annavad selgema ülevaate plagiaadi ulatusest ning tudengite omavahelistest seostest.

5.2 Tudengite vaade

Rakenduses oli algselt olemas nimekiri tudengitest koos nende GitLabi projektidega, kuid see pakkus piiratud infot plagiaadijuhtumite kohta. Selle täiendamiseks loodi uus vaade, kus kuvatakse tudengite plagiaadiga seotud andmed.



Joonis 8. Kursuse ava vaade.

Uues tudengite vaates on iga tudengi kohta näha tema nimi, Uni-ID ning tuvastatud plagiaadijuhtumite koguarv (Joonis 9). Varem oli see info kättesaadav ainult tudengi profiilivaates, mistõttu uus vaade annab märksa parema ülevaate kõigist kursuse tudengitest korraga.

Kui kasutaja vajutab tudengi Uni-ID-l, suunatakse ta vastava tudengi profiilivaatesse, kus on kättesaadav detailsem info tudengi ja tema plagiaadijuhtumite kohta.

Vaade lihtsustab võrdlemist ning aitab kiiremini tuvastada korduvaid plagiaati toime pannud tudengeid.

Tudengite vaate andmed pärinevad Django serveripoolsest vaatest, kus filtreeritakse välja kõik tudengid, kes kuuluvad vastavasse kursusesse. Lisaks arvutatakse iga tudengi kohta plagiaadijuhtumite koguarv, arvestades ainult neid juhtumeid, mis on seotud kursustega, kus vaadeldav kasutaja on õppejõud. See tagab, et õppejõud näeb ainult temaga seotud andmeid. Päringute koostamisel kasutatakse Django päringuliidest (QuerySet API), mis võimaldab mugavalt filtreerida, seostada seotud andmeid ja aggregeerida tulemusi.

Uni-ID	Name	Plagiarism cases
uniid1	Student 1	0
uniid2	Student 2	2
uniid3	Student 3	0

Joonis 9. Nimekiri kõikidest kursuse tudengitest.

5.3 Peavaate täiendus

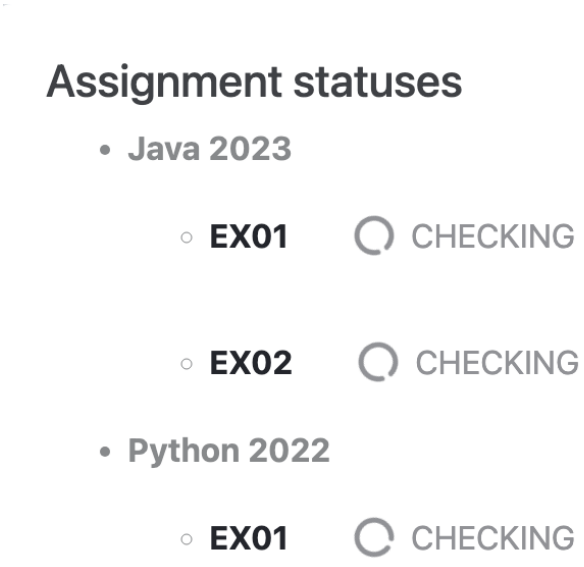
Peavaatesse on lisatud uus sektsioon, mis kuvab ülevaatlikult kõik aktiivsed plagiaadikontrolli protsessid (Joonis 10).

Kui kasutaja käivitab plagiaadikontrolli, lisatakse see protsess automaatselt sellesse sektsiooni ja püsib seal seni, kuni kontroll on lõpule viidud. See võimaldab kasutajal kiiresti näha, mitu protsessi on parasjagu töös ning milliste õppeainete ja ülesannetega need seotud on.

Uue sektsiooni kuvamiseks tehakse päring andmebaasi, kust küsitakse kõik plagiaadikontrolli protsessid, mille olek (*status*) on *CHECKING*. See tagab, et kasutajale kuvatakse ainult parasjagu käimasolevad kontrollid, mitte lõpetatud või ebaõnnestunud protsessid.

Iga protsessi ülesande nime peale vajutades suunatakse kasutaja vastava ülesande vaatesse. Seal saab tutvuda detailsemalt protsessi logidega või vaadata varasemate sama ülesandega

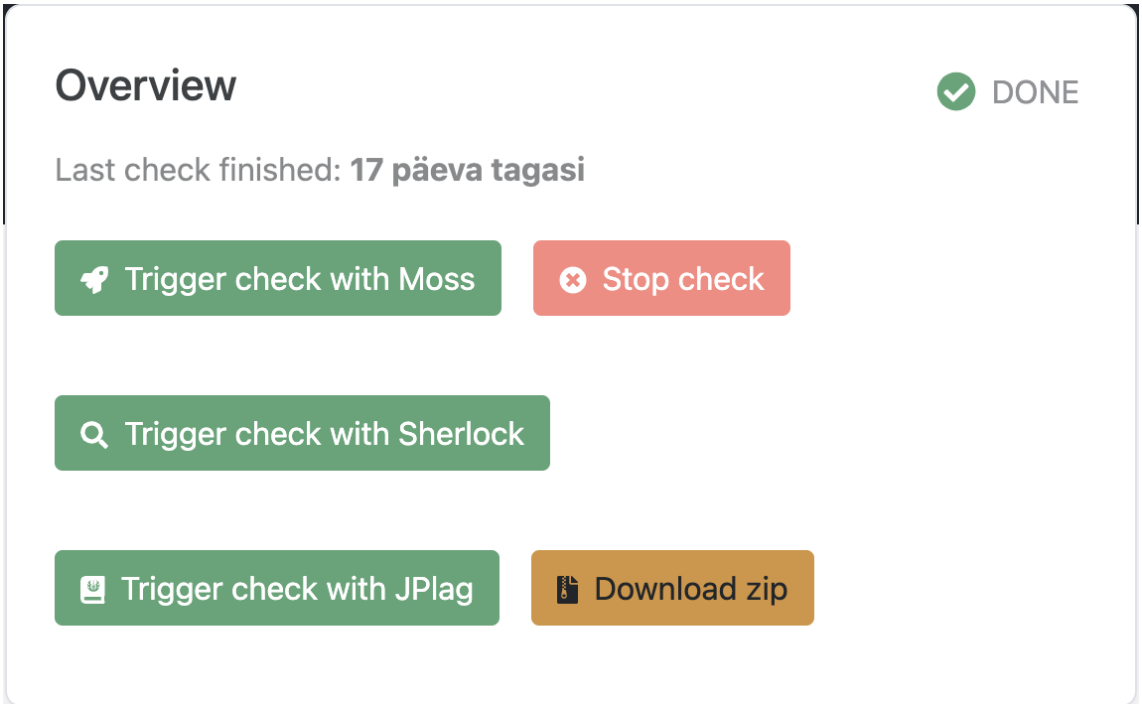
seotud plagiaadikontrollide tulemusi.



Joonis 10. Kõik jooksvad plagiaadikontrolli protsessid.

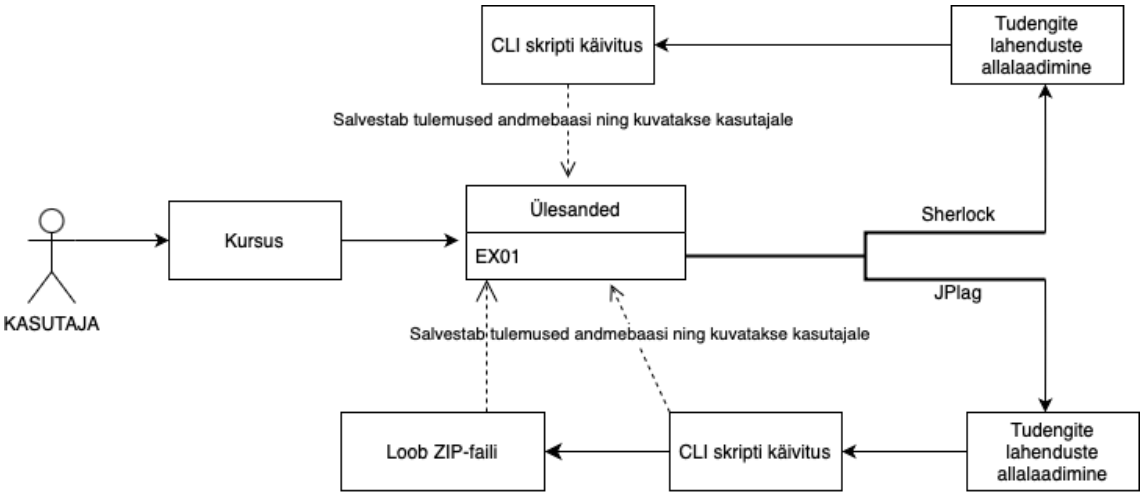
5.4 Uued plagiaadi kontrolli teenused

Varasemalt toetas rakendus plagiaadikontrolli teostamist ainult Mossi teenuse kaudu. Nüüd on sellele lisaks juurde integreeritud kaks uut tööriista – Sherlock ja JPlag – millega saab samuti teostada plagiaadituvastust. Joonisel 11 on võimalik näha ülesande põhivaate komponenti, mille kaudu võimalik käivitada plagiaadikontrolli kolme erineva tehnoloogiaga.



Joonis 11. Ülesande põhivaade, kus võimalik plagiaadi kontrolle käivitada.

Uute tööriistade eeliseks on nende sõltumatus välistest serveritest: need ei vaja päringute saatmiseks ega tulemuste saamiseks väliseid ühendusi. Selle asemel on programmid integreeritud otse rakenduse projekti koosseisu ning töötlevad tudengite koodifaile lokaalselt vastavalt käivitatud käskudele. See tagab, et plagiaadikontrolli on võimalik läbi viia igal ajal, olenemata välistest teenustest või serverite kättesaadavusest. Tööriistade protsessidega on võimalik tutvuda joonisel 12.



Joonis 12. Sherlocki ja JPlagi plagiaadikontrollide tegevusdiagramm.

Erinevate tööriistade kasutamine annab võimaluse läbi viia mitmeid plagiaadikontrolle ning võrrelda tulemusi omavahel. See võimaldab kasutajal hinnata, kui suures ulatuses ühtivad kontrollide tulemused erinevate programmide puhul ning tööriistade spetsiifikast lähtuvalt analüüsida, millistes aspektides lahendused omavahel sarnanevad.

5.4.1 Sherlock

Sherlocki liidestamiseks kasutati käsurealiidest (CLI), mille kaudu on võimalik käivitada plagiaadikontroll ning pärida tulemusi failidest, millel võib olla vabalt valitud laiend. Käesoleva lõputöö raames loetakse Sherlocki tulemused tavalistest tekstifailidest (.txt). Tulemuste failist loetakse, milliste tudengite faile omavahel võrreldi ning mis on failide kattuvuse protsent.

Liidestuse jaoks loodi projekti struktuuri eraldi kataloog nimega *sherlock* teenuse integreerimiseks. Kataloogi lisati ka kompileerimisfail (*Makefile*), mille abil seadistati ja käivitati kogu Sherlocki rakendus.

Sherlocki kasutamisel on võimalik kursuse seadetes määrata mitmeid tingimusi, mis mõjutavad plagiaadikontrolli tulemusi.

Kasutaja saab seadistada maksimaalse arvu kattuvusi, mida tulemustes kuvatakse, ning määrata vaikimisi kattuvuse protsendi – see määrab, kui suur peab sarnasus olema, et kattuvus tulemustesse kaasatakse.

Kontrolli käivitamise hetkel on võimalik vaikimisi protsenti muuta, võimaldades kasutajal kohandada sätteid konkreetse kontrolli vajadustele vastavalt, ilma et peaks iga kord kõiki kursuse seadeid uuesti üle vaatama.

Sherlocki puhul saab kattuvuse protsendi väärtuseks valida vahemikus 20% kuni 100%. Kursuse loomise vormi on võimalik näha lisas 5. Sarnane vorm on rakenduses olemas ka kursuse seadete muutmiseks.

Kattuvuse protsendi määramiseks on andmebaasis muudetud nii ülesannete tabelit kui ka kursuste tabelit. Mõlemasse tabelisse on juurde lisatud uus veerg *threshold* ning on arvu tüüpi (*smallint*). Iga kord kui seadetes muudetakse protsenti, salvestatakse andmebaasis ka uuendused.

Plagiaadikontrolli käivitamisel laaditakse esmalt alla kõik tudengite lahenduste failid. Seejärel käivitatakse Sherlocki käsk, mis arvestab eelnevalt määratud kattuvuslävendit, mis on üles otsitud andmebaasist ülesannete tabelist. Tulemuste faili kirjutatakse vaid need read, mille kattuvusprotsent on võrdne või ületab määratud piiri.

Kui tulemuste fail on loodud, loeb programm sellest tulemused – maksimaalselt nii palju, kui seadetes on määratud lubatud kattuvuste arvuks. Seejärel salvestatakse andmed andmebaasi, et neid saaks hiljem veebilehel kuvada ja analüüsida.

5.4.2 JPlag

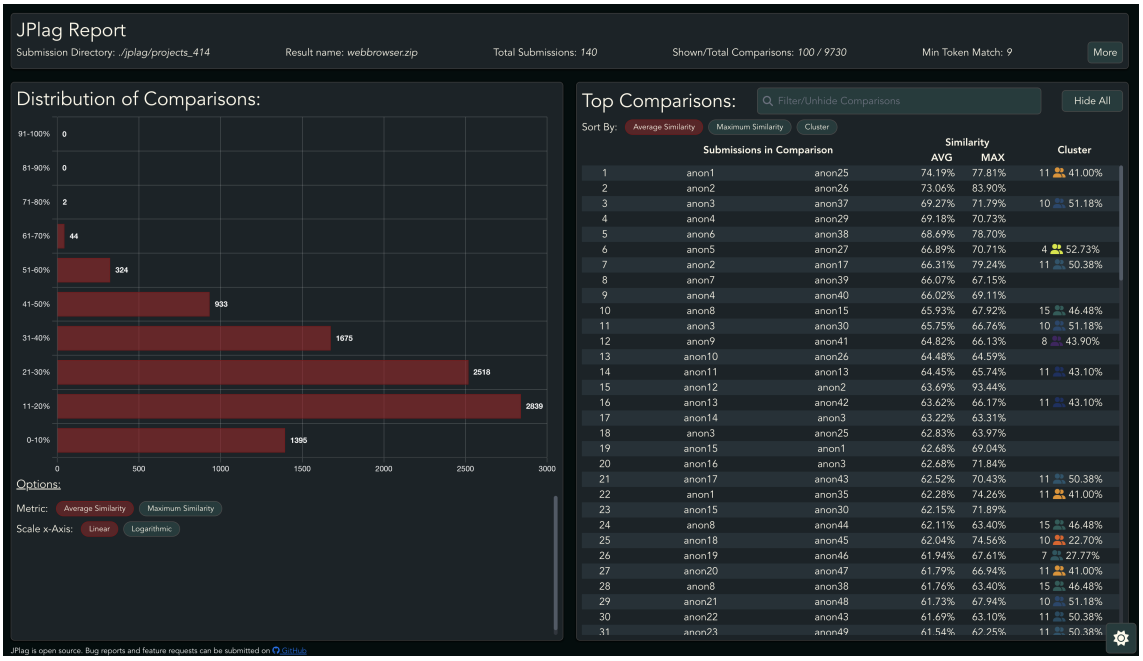
JPlagi kasutamisel on võimalik seadetes määrata, mitu kattuvust maksimaalselt tulemustes kuvatakse.

Kui kontroll JPlagiga on lõpetatud, saab tulemusi vaadata ka JPlagi enda veebiliideses lisaks rakenduse sisemistele tulemustele. Selleks salvestatakse andmebaasi ZIP-fail, mis sisaldab kõiki plagiaadikontrolli käigus genereeritud tulemusi konkreetse ülesande kohta. ZIP-faili saab rakenduse kaudu alla laadida pärast kontrolli lõppu.

Lõputöös kasutati JPlagi käsurealiidestust (CLI), mis võimaldab automatiseeritult käivitada plagiaadikontrolli ning salvestada tulemused ZIP-failina. Enne kontrolli käivitamist tõmmatakse tudengite ülesande lahendused alla Git-repositooriumidest, salvestatakse ajutiselt sobivasse kaustastruktuuri ning arhiveeritakse ka andmebaasi. Seejärel käivitatakse JAR-fail koos vajalike parameetritega, kasutades CLI liidest. Kasutatud JPlagi versioon on 4.3.0, mille käivitamiseks on nõutud JDK 17 olemasolu. See ZIP-fail sisaldab muu hulgas ka JSON-faile, milles on esitatud detailne teave kontrolli tulemuste kohta – näiteks kattuvused, vastendatud failid ning võrdlusstatistika. Lisaks sisaldab JPlag ka koondraportit (lähemalt saab tutvuda lisas 6), mida kasutatakse tulemuste visuaalseks esitamiseks. Need JSON-failid on integreeritud ka rakendusse ja neid kasutatakse tulemuste visualiseerimisel.

Andmebaasis on uuendatud ülesannete tabelit, kuhu lisati uus veerg (*zip_file*), mis on bait massiivi tüüpi (*bytearray*). Sellesse veergu salvestatakse alla laaditava ZIP-faili metaandmed. Iga kord, kui kasutaja soovib alla laadida plagiaadikontrolli tulemuste ZIP-faili, otsitakse vastavad andmed üles just sellest tabelist.

Allalaaditud faili saab üles laadida JPlagi veebivaatesse (Joonis 13), kus avaneb visuaalne ülevaade tulemustest [17]. See vaade erineb lõputöös arendatud rakenduse omast eelkõige sellepoolest, et JPlag esitab iga tudengi ülesande lahenduse failid eraldi, säilitades nende algse struktuuri. See on eriti kasulik ülesannete puhul, kus lahendus koosneb mitmest failist – näiteks eraldi koodifailid ja testid.



Joonis 13. Plagiaadikontrolli tulemused visualiseeritud JPlagi veebivaates.

Lõputöö raames loodud rakenduses kuvatakse kõik ühe tudengi koodifailid koondatult ühesainsas failis, mis muudab vajaliku faili leidmise ja võrdlemise aeganõudvaks ja ebamugavaks. JPlagi vaade muudab selliste juhtumite analüüsi oluliselt selgemaks ja tõhusamaks.

5.5 Kasutusjuhend

Lõputöö raames loodud rakenduse tõhusamaks kasutamiseks on loodud eraldi kasutusjuhendi vaade. See aitab kasutajal kiiresti rakenduse tööpõhimõtetega tutvuda ning sujuvalt kasutamisega alustada, tutvustades kõiki olulisi funktsionaalsusi ja vaateid.

Kasutusjuhend on jaotatud mitmeks leheküljeks, millest igaüks keskendub ühele konkreetsele teemale:

- Rakenduse peavaade – kuidas kursusi luua ja hallata.

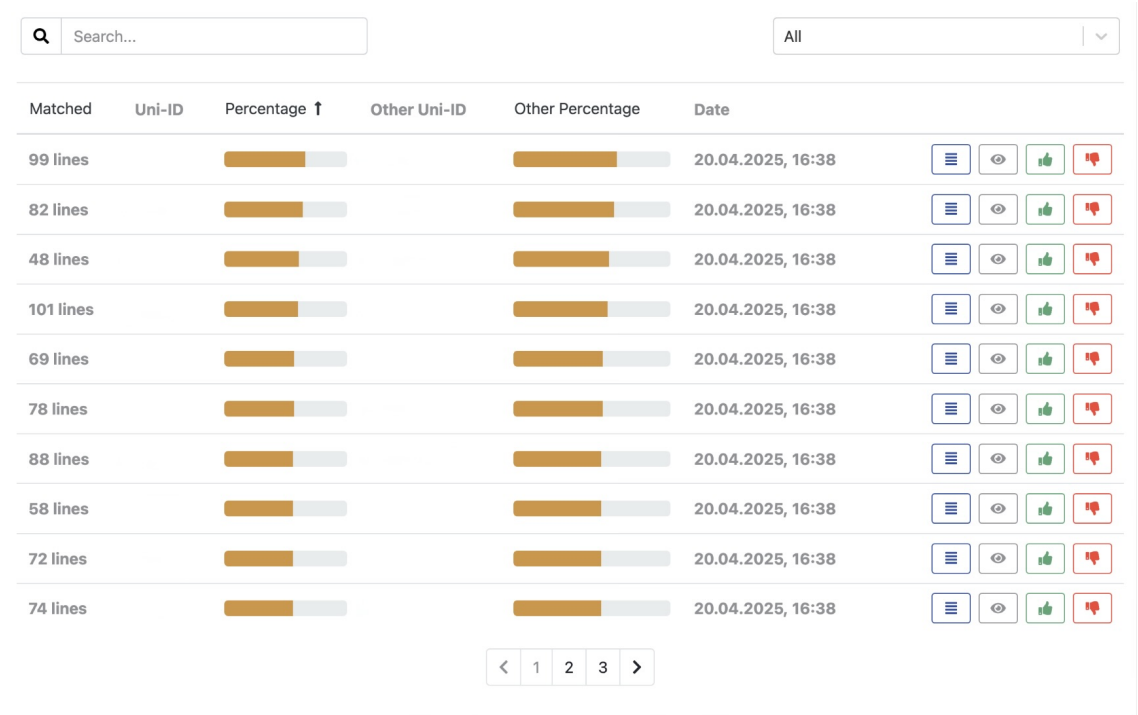
- Kursuse detailvaade – kuidas ülesandeid luua ja hallata.
- Ülesande vaade – kuidas kontrolli tulemusi vaadata ja analüüsida.
- Tudengi profiilivaade – millist infot tudengi kohta on võimalik näha.
- Plagiaadituvastusteenuste erinevused – iga tööriista eripära ja tööpõhimõte.

Kasutusjuhend on eeskätt abiks uutele kasutajatele, kes ei ole rakenduse liidesega varem kokku puutunud. Kuna iga uus kasutajaliides võib esialgu tunduda keeruline, aitab juhend oluliselt vähendada õppimiskõverat ja kiirendab rakenduse omaksvõttu.

5.6 Tulemuste vaate täiendused

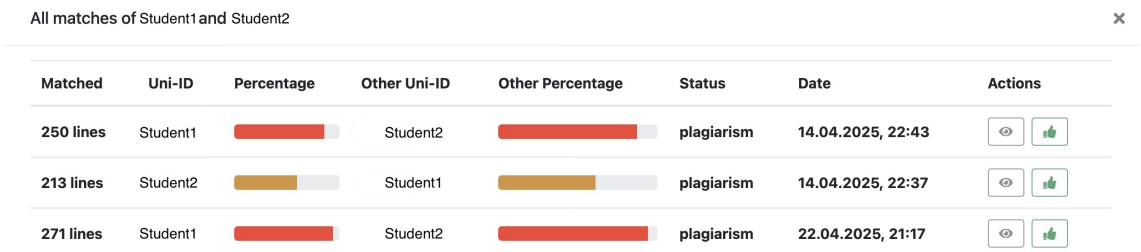
Arendustöö käigus on täiendatud plagiaaditulemuste nimekirja nii tulemuste kui ka ajaloo sektsioonis. Joonisel 14 on kujutatud tulemuste sektsiooni vaade, mis on sarnane ajaloo sektsiooni vaatega. Ajaloo vaates on võimalik kuvada erinevate plagiaadikontrollide tulemusena leitud kattuvuste loendeid.

Ajaloo vaate juurde on samuti lisatud, mis programmiga plagiaadikontrolli tehti. Andmebaasis uuendati selle jaoks plagiaadikontrollide protsessi tabelit (*plagiarism_run*), kuhu lisati juurde uue veerg *service*, mis märgib ära programmi, mis on rakendusele liidestatud ning millega kontroll tehti.



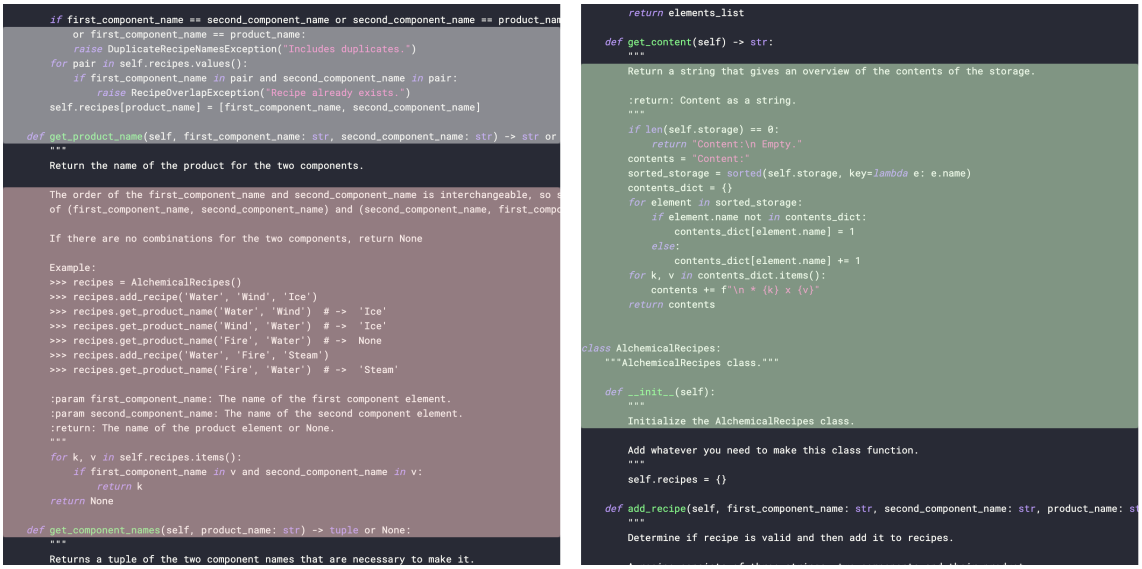
Joonis 14. Nimekiri kõikidest plagiaadikontrolli poolt leitud kattuvustest.

Iga kattuvuse juurde on lisatud uus nupp, mille abil saab avada kõik kattuvused sama tudengipaari kohta (Joonis 15). See on eriti kasulik, kui sama tudengipaari puhul on erinevad plagiaadikontrollid tuvastanud mitu kattuvust sama ülesande juures.



Joonis 15. Kõik plagiaadikontrolli kattuvused sama tudengipaari kohta.

Samuti on täiustatud detailvaadet, kus saab võrrelda kahe tudengi lahendusi (Joonis 16). Kui plagiaadikontroll on tehtud Mossi või JPlagi abil, kuvatakse lahendustes selgelt märgitud kattuvuse segmendid. Igal segmendil on oma värv, mis aitab erinevaid kattuvuskohti hõlpsamalt eristada. Sherlock niisugust funktsionaalsust ei paku.



Joonis 16. Detail vaade tudengite lahenduste võrdlemiseks.

Nende täienduste eesmärk on muuta plagiaaditulemuste analüüsimine kasutajale oluliselt mugavamaks ja vähendada lisatöö vajadust.

5.7 Muud parandused ja täiendused

Rakenduses on tehtud ka mitmeid väiksemaid täiustusi, mis muudavad selle kasutamise oluliselt mugavamaks ja kasutajasõbralikumaks.

Ülesande vaates on ümber kujundatud seadete ala. Varasemalt tuli ülesande kustutamiseks avada eraldi seadete menüü, mis muutis toimingu ebamugavaks ja aeganõudvaks. Nüüd on kustutamise nupp toodud otse seadete salvestamise nupu juurde, et see oleks hõlpsamini leitav ja kiiremini kasutatav.

Lisaks on mitmetes vaadetes tudengite Uni-ID-d muudetud linkideks, mis viivad otse vastava tudengi profiilivaatesse. See parandab rakenduse navigeeritavust ning muudab lihtsamaks plagiaadijuhtumitega tudengite leidmise ja nende olukorra analüüsimise.

Varasemalt sai plagiaadikontrolle teha ainult ühe semestri kursuse piires. Nüüd on aga võimalik lisada mitu kursust, mille tudengite projektitööd alla laadida, ja võrrelda lahendusi omavahel eri aastakäikude lõikes. Selle muudatuse toetamiseks on andmebaasis uuendatud kursuste tabelit – kahe olemasoleva veeru struktuuri on muudetud. Kui varem sisaldas tabel veerge *group_id* (täisarv) ja *group_name* (sõne), siis nüüd on need asendatud veergudega *group_ids* (täisarvude massiiv) ja *group_names* (sõnede massiiv), et võimaldada mitme kursuse seostamist ühe kirje kaudu.

6 Rakenduse testimine ja tulemuste analüüs

Järgnevad alapeatükid annavad ülevaate rakenduse testimisest ning tulemuste analüüsist, mille eesmärk oli hinnata rakenduse kvaliteeti ja kasutusmugavust. Läbivalt käsitletakse kõiki lõputöö käigus sõnastatud nõudeid, tuues välja, millised neist said täidetud ning millised jäid täitmata, koos vastavate selgitustega.

6.1 Valideerimine

Rakenduse tulemuste hindamiseks on läbi viidud valideerimine. Selleks on kasutajad rakendust testinud ning andnud tagasisidet nii rakenduse kvaliteedi kui ka kasutusmugavuse kohta. Lisaks on lõputöö autorid ise katsetanud plagiaadikontrollide käivitamist erinevate tehnoloogiatega, et võrrelda nende kiirust ja tõhusust. Plagiaadikontrollid on läbi viidud 2025. aasta "Programmeerimise põhikursuse" raames lahendatud ülesannetele, mis olid tüüpilised koodilahendused, mitte suuremahulised projektid.

Kasutajate testimiseks on igale testijale ette valmistatud kindel ülesannete komplekt, mida nad pidid rakenduses täitma. Iga ülesande puhul jälgiti sooritamise aega, töövoogu ning võimalike tõrgete esinemist. Samuti vaadati, kui kaua aega kulutas iga teenus plagiaadikontrolli tegemiseks. Kasutajatestide ülesannetega saab tutvuda lisas 2.

Pärast ülesannete sooritamist täitsid testijad küsimustiku, mille kaudu nad andsid tagasisidet oma kogemuse ja rakenduse üldise kasutatavuse kohta. Küsimustik on loodud Google Forms abiga, millega võimalik luua küsimustike, mille juurde lisada näiteks valik- ja avatud vastustega küsimusi [18]. Kogutud andmete põhjal on võimalik hinnata rakenduse mugavust, selgust ja tõhusust ning tuvastada valdkonnad, mida võiks veel täiustada. Küsimustikuga on võimalik tutvuda lisas 3.

Kõigi kursuse ülesannete plagiaadikontrolli käigus on hinnatud, kui kaua kontroll erinevate programmide ja teenustega aega võtab ning millised on saadud tulemused. Analüüsis vaadeldakse keskmisi kattuvusprotsente, et mõista, kui suured on erinevused programmide

tulemuste vahel. Mossi ja JPlagi tulemustes võrreldakse, kuidas kumbki tööriist esitab koodisegmente, mida on tuvastatud plagiaadina.

Plagiaadikontroll viidi läbi kokku 10 erineva ülesandega, millele saadeti hindamiseks 146 lahenduste repositooriumit. Iga kontroll tagastas maksimaalselt 25 kattuvuspaari. Sherlocki puhul seati tingimus, et tulemustes kajastataks ainult need kattuvused, mille protsentuaalne sarnasus oli vähemalt 25%. Mossi puhul seati tingimuseks, et 10 esimest kattuvust, mille kattuvusprotsent väga madal, ignoreeritakse ja ei lisata tulemustesse.

6.2 Tulemused

Rakenduse katsetamises osales üks õppejõud ja kolm abiõppejõudu. Üldiselt said kõik testijad ülesannetega hästi hakkama. Esines vaid üksikuid kohti, kus tekkis kerge segadus või kahtlus. Ülesannete lahendamine toimus kiiresti: keskmiselt kulus ühe ülesande täitmiseks 1–2 minutit. Kõige pikem lahendamisaeg oli 8 minutit, mis oli tingitud Mossi plagiaadikontrolli pikast töötlusajast.

Testijate käitumismustrid jagunesid kaheks: pooled täitsid vormid kiiresti ja käivitasid protsessid vähese kontrolliga, samas kui teised kontrollisid hoolikalt sisestatavaid andmeid ja protseduuride õigsust.

Testimiste käigus jälgiti ka, kui kaua erinevad teenused plagiaadikontrolli teostamiseks aega kulutavad, et saada täpne ülevaade iga teenuse keskmisest töötlusajast.

Plagiaadikontrollid viidi testijate poolt läbi kursuse „Programmeerimise põhikursus” teise ülesandega *EX02WebBrowser* ning tulemused on esitatud tabelis 1.

Tabel 1. Ülesande *EX02WebBrowser* plagiaadikontrolliks kulunud ajad erinevate teenuste poolt.

Testija	**Moss**	**Sherlock**	**JPlag**
1	7 minutit ja 35 sekundit	1 minut ja 13 sekundit	1 minut ja 30 sekundit
2	1 minut ja 15 sekundit	1 minut	1 minut ja 20 sekundit
3	1 minut ja 31 sekundit	1 minut ja 4 sekundit	1 minut ja 10 sekundit
4	1 minut ja 17 sekundit	1 minut ja 2 sekundit	1 minut ja 12 sekundit

Tabelist on näha, et esimesel testijal kulus Mossi teenusega plagiaadikontrolli läbiviimiseks oluliselt rohkem aega kui teistel. Selle põhjuseks võib olla asjaolu, et esmakordsel kont-

rollimisel tuli Mossil töödelda kogu andmestik põhjalikult ning andmed pidid serverisse esmakordselt laekuma. Hilisemate testimiste käigus olid samad andmed juba teenuse jaoks kättesaadavad, mistõttu toimus plagiaadikontroll märgatavalt kiiremini. Üldiselt on aga näha, et plagiaadikontrolliks kulunud ajad olid erinevate testijate vahel pigem sarnased. Kõige kiiremini on kontrolli teinud Sherlock. Moss ja JPlag on kontrolle teinud natuke kauem.

Testijate hinnangul oli rakendust väga lihtne kasutada ning veebilehe navigeerimine oli loogiline ja arusaadav. Enamikul juhtudest rakendus töötas probleemideta ning kasutajaliidese disainiga jäädi üldiselt rahule. Ainsaks esinenud veaks oli olukord, kus tudengivaates puudusid andmed pärast lehe värskendamist.

Kasutajatele meeldis rakenduse:

- selge ja arusaadav kasutajaliides,
- lihtne kasutusloogika,
- võimalus kasutada plagiaadikontrolliks mitut erinevat teenust,
- võimalus analüüsida lahenduste erinevusi ja kattuvusi tudengite vahel.

Välja toodi ka ettepanekuid, mida rakenduses paremaks teha:

- Kattuvuse määramise nupud: Testijatele jäi segaseks, kumb pöidlaikooniga nupp määrab kattuvuse plagiaadiks ja kumb aktsepteeritavaks. Soovitati muuta nupud selgemaks.
- Plagiaadi kontrolli tehnoloogia kuvamine: Soovitati, et kattuvuste nimekirjas kuvataks, millise teenuse kaudu vastav kattuvus tuvastati, et tulemusi oleks lihtsam analüüsida.
- Leheküljeindikaatorid: Soovitati täiustada lehekülgede navigeerimise süsteemi, et oleks selgemalt nähtav, millisel lehel parasjagu ollakse, mitu lehekülge kokku on ning lisada võimalus liikuda otse algusesse või lõppu.

Lisaks pakuti välja uusi funktsionaalsusi, mis muudaksid rakenduse veel kasutajasõbralikumaks:

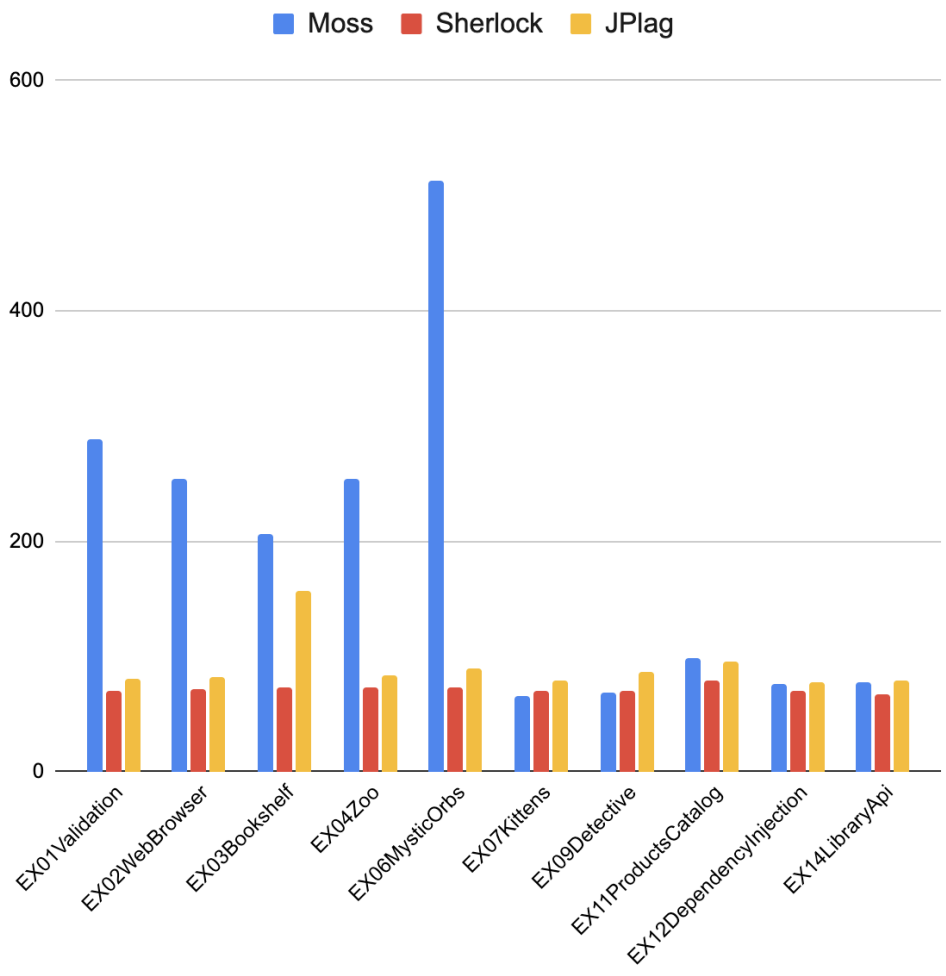
- Osalejate valik plagiaadikontrollis: Võimalus määrata, kas konkreetset osalejat (nt abiõppejõudu) tuleks kontrollimisse kaasata või mitte.

- Charoni liidestuse täiustus: Sooviti, et rakenduse liidestust Charoniga täiendataks, et selle kaudu saaks käivitada plagiaadikontrolle uute programmidega.

Testimise tulemusel võib järeldada, et rakendus on kasutajatele lihtne ja mugav kasutada. Samas ilmnes mitmeid arengukohti, mille parendamine aitaks veelgi suurendada süsteemi kasutajasõbralikkust ja funktsionaalsust.

Lisaks testijate teostatud plagiaadikontrollidele viisid lõputöö autorid iseseisvalt läbi kontrollid kõigi testimiseks valitud ülesannetega. Tulemuste diagrammide tegemiseks on kasutatud Google Sheets programmi [19].

Joonis 17 visualiseerib erinevate programmide ja teenuste plagiaadikontrollide kestvusaegu.



Joonis 17. Plagiaadikontrollide kestvusajad kõigi programmide poolt sekundites.

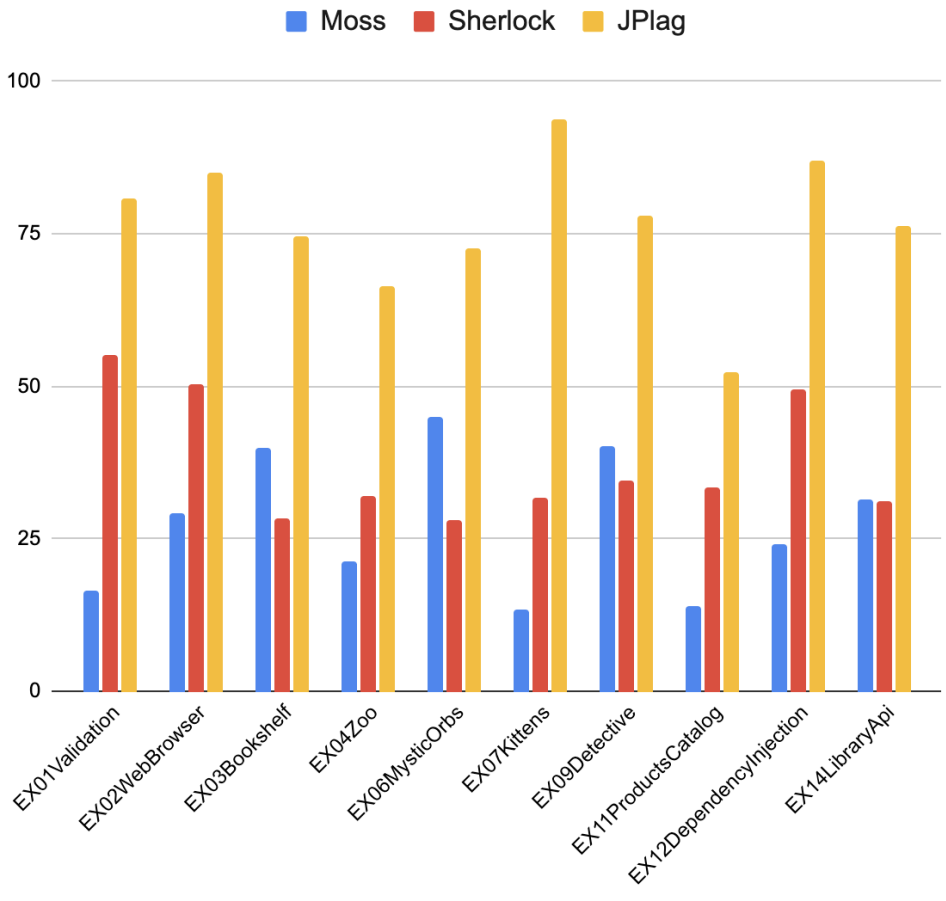
Jooniselt on näha, et esimese viie ülesande puhul kulus kõige rohkem aega Mossil, seejärel

JPlagil ning kõige vähem Sherlockil. Viimase viie ülesande korral olid kõigi programmide sooritusajad ligilähedaselt sarnased.

Mossi maksimaalne sooritusaeg ulatus 8 minuti ja 33 sekundini. JPlagi puhul oli maksimaalne aeg 2 minutit ja 37 sekundit ning Sherlockil 1 minut ja 20 sekundit — need on märgatavalt kiiremad tulemused, mille puhul kasutaja ei pea liiga kaua ootama.

Moss töötleb lahendusi eriti põhjalikult, mis sõltuvalt ülesande mahust võib kaasa tuua märkimisväärselt pikema töötlusaja. Kuigi kõigi programmide tulemused on sisukad ja informatiivsed, suudavad JPlag ja Sherlock teatud juhtudel kontrolli läbi viia märksa lühema ajaga.

Plagiaadikontrollide käigus analüüsiti ka tagastatud tulemusi, mille põhjal arvutati keskmised kattuvusprotsendid iga teenuse ja programmi kohta. Iga protsent näitab, kui suures ulatuses oli tudengite lahendustes keskmiselt tuvastatav potentsiaalne plagiaat. Tulemused on visualiseeritud joonisel 18.



Joonis 18. Tudengite lahenduste keskmised kattuvusprotsendid plagiaadikontrollide põhjal.

Kõige kõrgemaid kattuvusprotsente on tagastanud JPlag, mis arvestab koodi struktuurilisi sarnasusi, isegi kui need vaid osalised. Moss ja Sherlock on andnud vaheldumisi suuremaid ja väiksemaid kattuvusprotsente, mis viitab, et teatud koodiosad jäetakse arvestamata. See näitab, et potensiaalseks plagiaadiks määratakse vaid olulisemad kohad koodis, kus plagiaadi võimalus suurem.

Sellised erinevused tulemustes annavad väärtusliku ülevaate, kuidas iga programm plagiaati erinevalt analüüsib. Kui kasutada tööriistu koos, koguneb laiem valik kattuvusi, mida võimalik edasi uurida ja põhjalikumalt analüüsida.

Tulemusi analüüsiti ka rohkem süvitsi, et võrrelda Mossi ja JPlagi poolt tuvastatud kattuvussegmente. Sellist võrdlust oli võimalik läbi viia vaid juhtudel, kus mõlemad programmid tuvastasid kattuvuse samade tudengipaaride lahendustes.

Mitmete ülesannete ja tudengipaaride põhjal tehtud võrdluses ilmnes, et enamikul juhtudel tuvastas Moss rohkem väiksemaid kattuvussegmente, samas kui JPlag kattis vähem, kuid mahukamaid ja ulatuslikumaid segmente. Sageli kattis JPlag suurema osa koodist, mistõttu olid ka selle programmi tagastatud kattuvusprotsendid kõrgemad.

Siiski leidus ka juhtumeid, kus Mossi ja JPlagi tulemused olid väga sarnased ning mõlemad katsid ligikaudu samas ulatuses koodi. See näitab, et Moss keskendub rohkem koodiosadele, mis võivad viidata tõsisemale plagiaadile, samal ajal kui JPlag tuvastab ulatuslikumalt sarnaseid koodilõike, sõltumata nende plagiaadipotentsiaalist.

Kõik funktsionaalsed nõuded said täidetud. Tudengite jaoks loodi eraldi vaade tudengite nimekirjast, kust on võimalik tutvuda üldise infoga ning liikuda edasi konkreetse tudengi profiilile. Tudengi profiilil on kättesaadavad detailsemad andmed plagiaadijuhtumite kohta, sealhulgas info selle kohta, kui palju plagiaati on tudeng tuvastatult teinud ning milliste kursusekaaslastega need juhtumid seotud on. Plagiaadijuhtumid on visualiseeritud diagrammide ja graafikutena, mis muudab info mõistetavamaks ja analüüsitavamaks.

Kasutaja saab jälgida kõiki plagiaadiprotsesse rakenduse põhivaates ning teostada kontrolli erinevate kursuste tööde lõikes. Plagiaadikontrolli saab läbi viia mitme erineva teenuse ja programmiga, võimaldades täpsemate tulemuste põhjal teostada süvitsi minevaid analüüse. Lisaks on rakendusega kaasas kasutusjuhend, mis katab kõik olulised teemad ja aitab

kasutajal süsteemi kiiresti omandada.

Ka kõik mittefunktsionaalsed nõuded said täidetud. Rakendusse liidestatud plagiaadituvastuse tööriistad töötavad iseseisvate taustülesannetena, mis tagab süsteemi parema jõudluse ja töökindluse. Tudengite andmetele ligipääs on muutunud lihtsaks ja mugavaks, vähendades käsitööd ning kiirendades igapäevatööd. Õpetaja rolliga kasutajatel on ligipääs kõigile neile määratud kursustele ja nendele vastavale infole, samal ajal kui administraatorid saavad hallata kõiki rakenduses loodud kursusi.

6.3 Edasised arendusvõimalused

Käesoleva lõputöö raames on välja töötatud rakendus, mis võimaldab sooritada plagiaadikontrolle mitme erineva tehnoloogia abil ning analüüsida saadud tulemusi ja tudengite lahendusi võimalike plagiaadijuhtumite tuvastamiseks. Kuigi rakendus sisaldab juba mitmeid kasulikke ja kasutajasõbralikke funktsionaalsusi, on sellel ka mitmeid edasisi arendusvõimalusi, mis võimaldaksid süsteemi veelgi laiendada ja olemasolevaid lahendusi täiustada.

Võimalik oleks rakendusse integreerida keelemudelitel põhinevaid plagiaadituvastussüsteeme. Need suudaksid sarnaselt olemasolevatele tehnoloogiatele sooritada plagiaadikontrolli, kuid potentsiaalselt veelgi suurema täpsuse ja põhjalikkusega. Sellised süsteemid võiksid pakkuda kasutajatele paremaid vahendeid tudengite lahenduste analüüsimiseks ning võimalike plagiaadijuhtumite tuvastamiseks.

Arvestada võiks tehisintellekti abil genereeritud lahendusi. Võrdlemine tudengi kirjutatud ja tehisintellekti loodud lahenduste vahel võimaldaks tuvastada ka sellist plagiaadi, kus lahendus pole otseselt kopeeritud mõnelt teiselt tudengilt, vaid loodud AI-toega. Selline lähenemine aitaks tuvastada laiemat plagiaadijuhtumite ringi ning suurendaks ülesannete iseseisva lahendamise nõuet.

Rakenduse edasine arendus võiks hõlmata ka funktsionaalsust, mille abil sooritab süsteem automaatselt kindla ajavahemiku järel plagiaadikontrolle. Selline lahendus võimaldaks määrata kontrollide sagedust ja ajastust ning vähendaks kasutaja käsitööd, kiirendades seeläbi tulemuste kättesaamist ja analüüsi alustamist.

Mitmed rakenduses kasutatavad teegid ja platvormide versioonid on aegunud, mistõttu nende ajakohastamine on vältimatu. Aegunud tehnoloogiad võivad muutuda kokkusobimatuks uuemate tarkvarakomponentidega ning põhjustada süsteemitõrkeid. Kuna rakenduse komponendid on mitme aasta vanused, tuleb uuendamist läbi viia ettevaatlikult, arvestades omavahelisi sõltuvusi ja võimalikke versioonikonflikte.

Täiendusi vajab ka rakenduse kasutusjuhend. Olemasolevat juhendit võiks muuta selgemaks ja struktureeritumaks, et katta rakenduse põhifunktsioonid ja kasutusloogika paremini. Lisaks võiks kasutusjuhendit täiendada instrueeriva videoga, mis annaks kasutajatele visuaalse ülevaate rakenduse tööpõhimõtetest ja julgustaks juhendit aktiivsemalt kasutama.

Välja on käidud soov täiustada rakenduse liidestust õppekeskkonnaga Charon, mida kasutatakse laialdaselt programmeerimisülesannete lahenduste ja kaitsmiste haldamiseks. Täiustatud liidestus võimaldaks Charonis hallatavate ülesannete plagiaadikontrolli viia läbi ka uute tehnoloogiatega, mis lisati lõputöö käigus rakendusele juurde. See lihtsustaks rakenduse kasutamist ning laiendaks selle kasutusvõimalusi õppetöös.

7 Kokkuvõte

Käesoleva lõputöö eesmärk oli täiustada olemasolevat rakendust, mis on loodud tudengite lähtekoodide plagiaadijuhtumite tuvastamiseks. Töö fookus oli olemasolevate funktsionaalsuste edasiarendamisel, et tagada rakenduse stabiilne kasutatavus ning parandada selle kasutajasõbralikkust. Projekti peamisteks kitsaskohtadeks olid üheainsa plagiaadituvastusteenuse kasutamine ja ebamugav navigeerimine rakenduse vaadete vahel.

Eesmärkide saavutamiseks viidi läbi mitmete plagiaadituvastustehnoloogiate ja -teenuste võrdlev analüüs, mille tulemusel valiti välja sobivaimad lahendused ja integreeriti need rakendusse. Samuti testiti olemasolevat funktsionaalsust ning viidi sisse mitmeid parandusi ja täiendusi. Selle tulemusel muutus tudengite andmete leidmine ja analüüsimine kursuste lõikes lihtsamaks ning plagiaadijuhtumite statistika jälgimine tõhusamaks. Rakenduse kasutajaliides kuvab nüüd varasemast rohkem selgitavat teavet, mis toetab paremat kasutajakogemust. Lisaks võimaldab rakendus võrrelda plagiaadituvastuse tulemusi mitme erineva teenuse lõikes, pakkudes põhjalikumat analüüsivõimalust.

Edasiarendatud rakendust testiti kevadsemestril Tallinna Tehnikaülikooli õppeaines "Programmeerimise põhikursus" koostöös õppejõu ja abiõppejõududega. Testimisel viidi plagiaadikontrolle läbi mitme teenuse abil, et hinnata rakenduse töökindlust ja kasutusmugavust. Tagasiside näitas, et rakendus on mugav ja intuitiivne kasutada ning mitme teenuse tugi aitab märgatavalt kaasa tudengite tööde kontrollimisele. Testijate ettepanekute põhjal viidi sisse täiendavad muudatused, mis parandasid rakenduse kasutatavust veelgi. Valminud rakendus saavutas lõputööga seatud eesmärgid ning loodud lahendus osutus edukaks.

Rakendusel on mitmeid potentsiaalseid edasiarendus võimalusi, mille seas tehisintellektil põhinevate mudelite ja Charon pistikprogrammiga liidestamine. Sellised edasiarendused võimaldaksid õppejõududel ja abiõppejõududel rakendust veelgi tõhusamalt kasutada ja sellele mugavamalt ligi pääseda.

Kasutatud kirjandus

[1] Ragnar Rebase. *Lähtekoodi sarnasust tuvastava süsteemi arendamine*. [Accessed: 04-02-2025]. URL: https://digikogu.taltech.ee/et/Item/e1aa3467-61dc-48a3-9f47-34271b47d653.

[2] Discord Inc. *Discord*. [Accessed: 09-05-2025]. URL: https://discord.com/.

[3] Microsoft. *Teams*. [Accessed: 09-05-2025]. URL: https://www.microsoft.com/en-us/microsoft-teams/group-chat-software.

[4] Rob Pike. *The Sherlock Plagiarism Detector*. [Accessed: 13-03-2025]. URL: https://github.com/diogocabral/sherlock?tab=readme-ov-file.

[5] Timur Sağlam. *JPlag - Detecting Source Code Plagiarism*. [Accessed: 13-03-2025]. URL: https://github.com/jplag/jplag.

[6] Dick Grune. *The software and text similarity tester SIM*. [Accessed: 31-03-2025]. URL: https://dickgrune.com/Programs/similarity_tester/.

[7] Arthur Zaczek. *OSPC*. [Accessed: 31-03-2025]. URL: https://github.com/arthurzaczek/OSPC.

[8] Andreas Viklund. *ACNP Software*. [Accessed: 31-03-2025]. URL: http://www.anticutandpaste.com/anticutandpaste/.

[9] Team Dodona. *Dolos*. [Accessed: 31-03-2025]. URL: https://dolos.ugent.be/docs/.

[10] Manuel Freire. *AC*. [Accessed: 31-03-2025]. URL: https://github.com/manuel-freire/ac2?tab=readme-ov-file.

[11] Aleksi Ahtiainen. *Plaggie*. [Accessed: 31-03-2025]. URL: https://www.cs.hut.fi/Software/Plaggie/.

[12] Ago Luberg. *Charon*. [Accessed: 04-02-2025]. URL: https://charon.pages.taltech.ee/doc/index.html.

[13] Alex Aiken. *Moss. A System for Detecting Software Similarity*. [Accessed: 03-02-2025]. URL: https://theory.stanford.edu/~aiken/moss/.

[14] Django Software Foundation. *Django documentation*. [Accessed: 03-02-2025]. URL: https://docs.djangoproject.com/en/5.1/intro/overview/.

[15] Meta Platforms. *React Reference Overview*. [Accessed: 13-03-2025]. URL: https://jplag.github.io/JPlag/.

[16] The PostgreSQL Global Development Group. *PostgreSQL 14.18 Documentation*. [Accessed: 03-02-2025]. URL: https://www.postgresql.org/docs/14/index.html.

[17] Timur Sağlam. *JPlag - Detecting Source Code Plagiarism*. [Accessed: 13-03-2025]. URL: https://jplag.github.io/JPlag/.

[18] Google. *Google Forms*. [Accessed: 09-05-2025]. URL: https://workspace.google.com/intl/et/products/forms/.

[19] Google. *Google Sheets*. [Accessed: 30-05-2025]. URL: https://workspace.google.com/products/sheets/.

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

Meie, Mattias Tüür ja Sander Pleesi

1. Anname Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Lähtekoodi sarnasust tuvastava süsteemi edasiarendus", mille juhendaja on Bahdan Yanovich
    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. Oleme teadlikud, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autoritele.
3. Kinnitame, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

03.06.2025

[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.

Lisa 2 – Valideerimisülesanded

Järgnevalt on esitatud kõik ülesanded, mida rakenduse testijad pidid täitma:

1. Loo uus kursus 2025. aasta "Programmeerimise põhikursuse“ jaoks. Seejärel laadi kursuse raames alla kõik tudengite projektid.
2. Loo kursuse alla uus ülesanne *EX02WebBrowser* ja vii sellel läbi plagiaadikontroll, kasutades Mossi teenust.
3. Muuda sama ülesande seadetes Sherlocki lävendiprotsent 60 protsendi peale ning käivita plagiaadikontroll Sherlocki abil.
4. Tee sama ülesandega plagiaadikontroll JPlagi teenusega ning laadi pärast kontrolli tulemused alla *ZIP* failina.
5. Ava plagiaadikontrolli tulemustest üks kattuvus ja määra selle staatus plagiaadiks. Seejärel vali plagiaadiks määratud kattuvuse esimene tudeng ja liigu tema tudengiprofiili lehele.

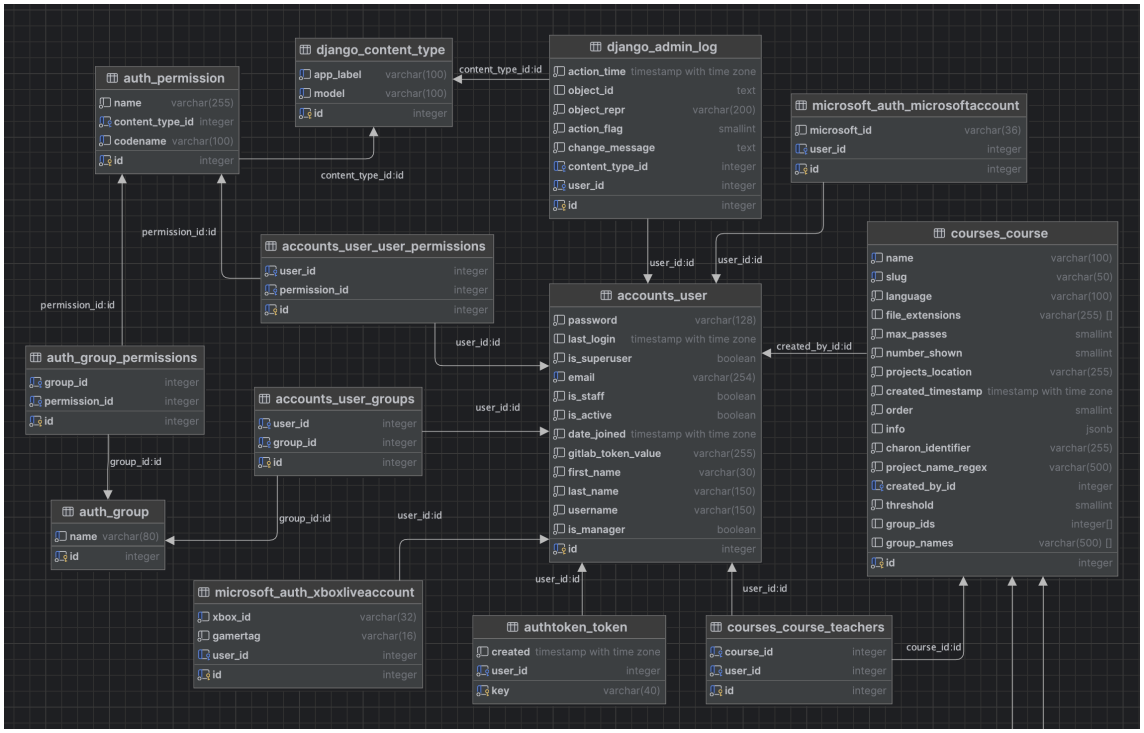
Lisa 3 – Tagasiside küsimustik

Antud vorm on tagasisidestamiseks lõputöö käigus arendatud tarkvara testimiseks.

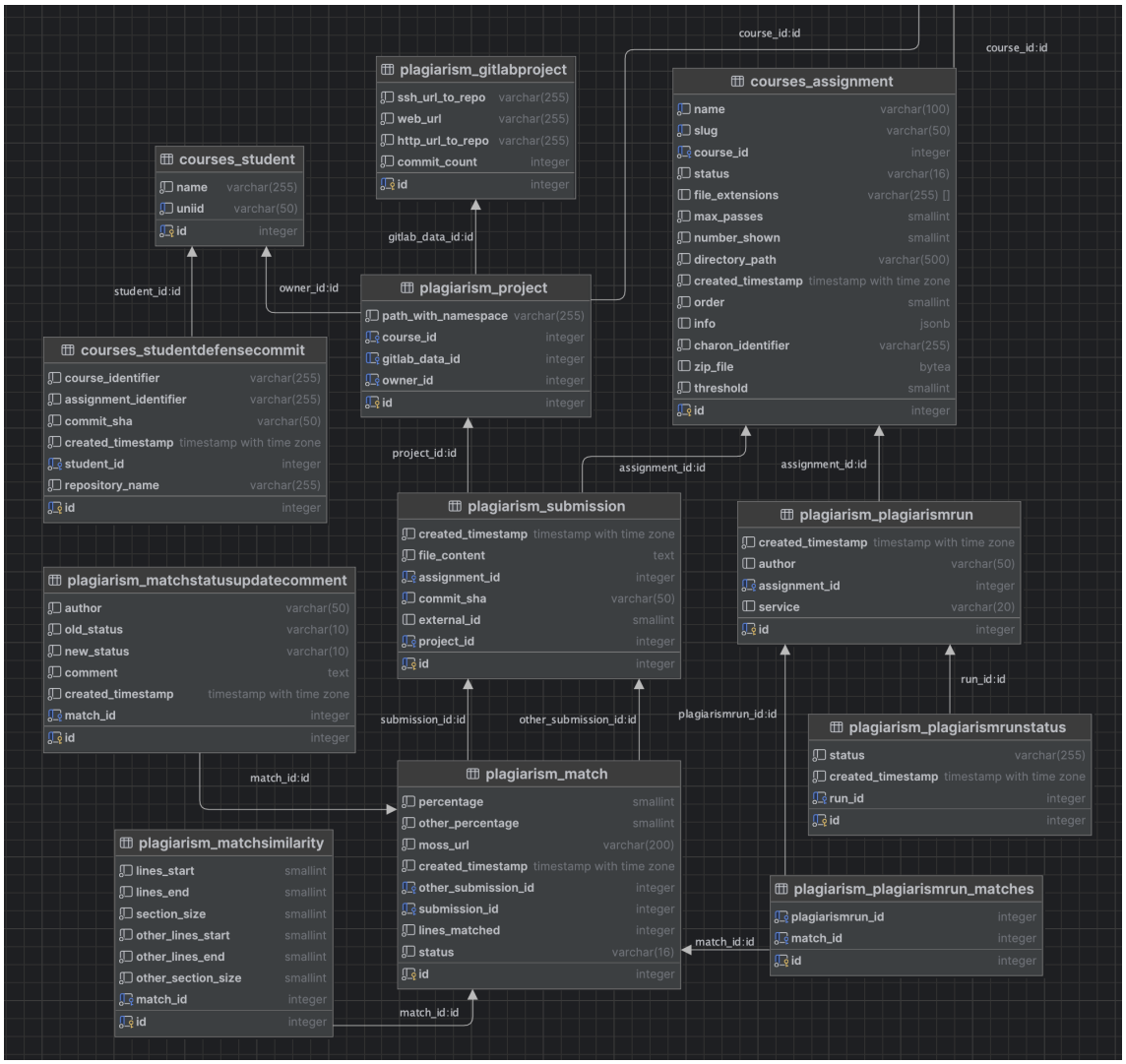
1. Sinu nimi
2. Sinu roll (õppejõud või abiõppejõud)
3. Kui lihtne oli rakendust kasutada?
    - Väga lihtne
    - Pigem lihtne
    - Neutraalne
    - Pigem keeruline
    - Väga keeruline
4. Kui loogiline oli rakenduses navigeerimine?
    - Väga loogiline
    - Pigem loogiline
    - Neutraalne
    - Pigem ebaloogiline
    - Väga ebaloogiline
5. Kaua võttis rakendusest arusaamine aega?
    - Väga vähe
    - Pigem vähem aega
    - Pigem kauem aega
    - Väga kaua
6. Kas kõik funktsionaalsused töötasid või esines tõrkeid?
    - Kõik töötas ilma tõrgeteta
    - 1-2 tõrget esines
    - 3 või rohkem tõrget esines
7. Kui meeldiv oli kasutajaliidese disain?
    - Jäin väga rahule
    - Pigem meeldis

    - Neutraalne
    - Pigem ei meeldinud
    - Üldse ei meeldinud
8. Mis sulle rakenduse juures meeldis?
9. Mis võiks rakenduses paremini olla?

Lisa 4 – Rakenduse andmebaas

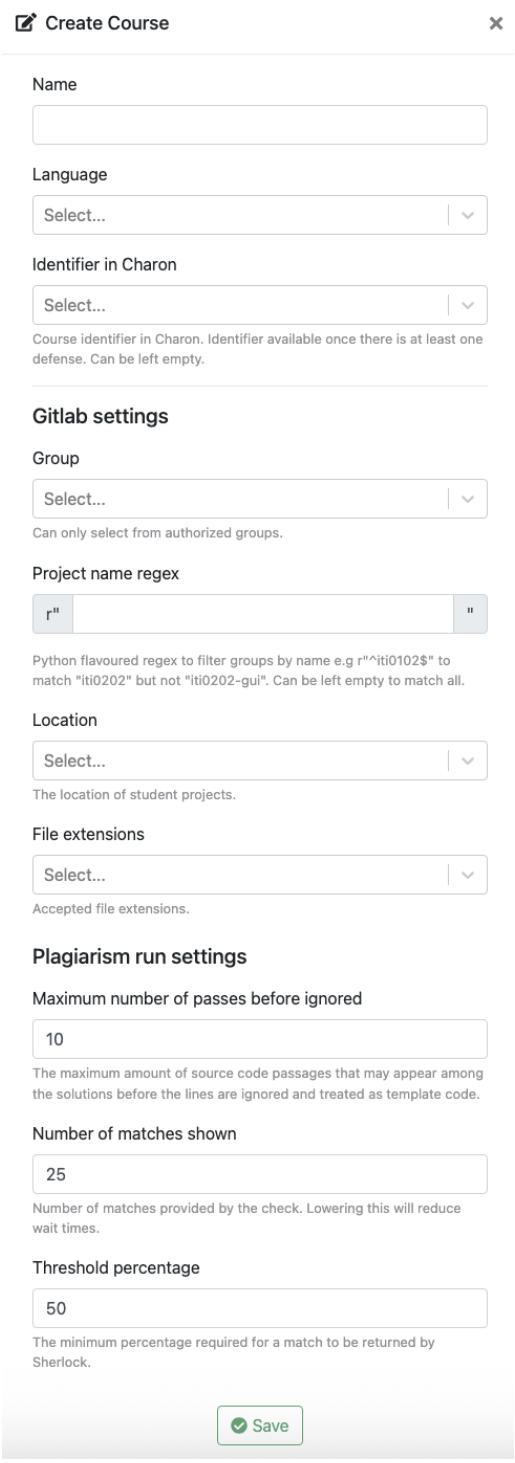


Joonis 19. Rakenduse andmebaasi esimene joonis.



Joonis 20. Rakenduse andmebaasi teine joonis.

Lisa 5 – Kursuse loomise vorm



Joonis 21. Kursuse loomise vorm kättesaadaval rakenduse põhivaates.

Lisa 6 – JPlagi koondraporti JSON fail


```
{
  "jplag_version": {
    "major": 4,
    "minor": 3,
    "patch": 0
  },
  "submission_folder_path": ["./jplag/projects_45"],
  "base_code_folder_path": "",
  "language": "Javac based AST plugin",
  "file_extensions": [".java", ".JAVA"],
  "submission_id_to_display_name": {
    "student_1": "student_1",
    "student_2": "student_2",
    "student_3": "student_3"
  },
  "submission_ids_to_comparison_file_name": {
    "student_1": {
      "student_2": "student_1-student_2.json",
      "student_3": "student_1-student_3.json"
    },
    "student_2": {
      "student_3": "student_2-student_3.json"
    }
  },
  "failed_submission_names": [],
  "excluded_files": [],
  "match_sensitivity": 9,
  "date_of_execution": "09/04/25",
  "execution_time": 1264,
  "metrics": [
    {
      "name": "AVG",
      "distribution": [3, 4, 3, 33, 296, 1563, 2850, 3971, 2383, 219],
```

```
    "topComparisons": [
      {
        "first_submission": "student_1",
        "second_submission": "student_2",
        "similarity": 0.94
      },
      {
        "first_submission": "student_3",
        "second_submission": "student_1",
        "similarity": 0.91
      },
      {
        "first_submission": "student_3",
        "second_submission": "student_2",
        "similarity": 0.85
      }
    ],
  },
  {
    "name": "MAX",
    "distribution": [4, 6, 14, 134, 822, 2238, 3073, 3260, 1647, 127],
    "topComparisons": [
      {
        "first_submission": "student_1",
        "second_submission": "student_2",
        "similarity": 1.0
      },
      {
        "first_submission": "student_3",
        "second_submission": "student_1",
        "similarity": 0.96
      },
      {
        "first_submission": "student_3",
        "second_submission": "student_2",
        "similarity": 0.96
      }
    ],
  }
```

```
  ],
  "clusters": [
    {
      "average_similarity": 0.90,
      "strength": 0.002,
      "members": ["student_1", "student_2", "student_3"]
    }
  ],
  "total_comparisons": 3
}
```