TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Kris-Sten Pallas 223200IAIB

Karl Agasild 222507IAIB

GitLabi projektide metaandmete analüüsi platvormi Cognate tudengivaate loomine ja kasutusmugavuse täiustamine

Bakalaureusetöö

Juhendaja: Ago Luberg

PhD

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: Kris-Sten Pallas ja Karl Agasild

04.06.2025

Annotatsioon

Bakalaureusetöö „GitLabi projektide metaandmete analüüsi platvormi Cognate tudengivaate loomine ja kasutusmugavuse täiustamine“ keskendub Tallinna Tehnikaülikooli tarkvaraprojektide haldusplatvormi Cognate'i edasiarendamisele. Töö peamisteks eesmärkideks on parandada platvormi üldist kasutusmugavust, lahendada andmete täpsusega seotud probleeme, viia lõpule eesrakenduse migratsioon React-tehnoloogiale, automatiseerida tudengite hindamist ning luua uus funktsionaalsus – tudengivaade.

Arendustööde käigus täiustati andmemudelit duplikaatandmete vältimiseks ja projektide korrektseks haldamiseks mitmes grupis. Uuendati grupihalduse loogikat, lisades gruppide eemaldamise ja turvalisema liikmete haldamise. Eesrakendus viidi üle Reactile, mis parandas ühilduvust TalTechi disainisüsteemiga ja lihtsustas hooldust. Automatiseeriti tudengite projektihalduse hindamist, mis vähendas manuaalset tööd ligikaudu 50%. Loodi uus tudengivaade, mis võimaldab üliõpilastel jälgida oma tööpanust reaalajas, toetades eneseanalüüsi. See hõlmas esi- ja tagarakenduse arendusi, sh projektipõhist ligipääsukontrolli.

Tulemusena on Cognate'i platvorm muutunud stabiilsemaks, kasutajasõbralikumaks ja funktsionaalsemaks. Uus tudengivaade pakub tudengitele otsest kasu, samas kui üldised täiustused parandavad õppejõudude kasutuskogemust. Tehtud muudatused on saanud positiivset tagasisidet ja suurendavad platvormi väärtust õppetöös.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 44 leheküljel, 7 peatükki, 20 joonist, 2 tabelit.

Abstract

Development of the student view and usability enhancement for the GitLab projects metadata analysis platform Cognate

The Bachelor's thesis focuses on the further development of the Cognate platform used at Tallinn University of Technology for managing software team projects. The main objectives of the work is to enhance the overall usability of the platform, resolve issues related to data accuracy, complete the migration of the front-end application to React technology, automate student assessment and create a new functionality – a student view.

During the development, the data model was improved to prevent data duplication and to correctly manage projects belonging to multiple groups. Group management logic was updated by adding group deletion capabilities and more secure member management. The front-end application was migrated to React, which improved compatibility with TalTech's design system and simplified maintenance. The project management assessment for students was automated, resulting in about a 50% decrease in manual work. A new student view was created, enabling students to monitor their work contributions in real-time, thereby supporting self-analysis. This involved developments in both front-end and back-end, including project-specific access control.

As a result, the Cognate platform has become more stable, user-friendly, and functional. The new student view provides direct benefits to students, while general improvements enhance the user experience for instructors. The implemented changes have received positive feedback and increased the platform's value in the educational process.

The thesis is in Estonian and contains 44 pages of text, 7 chapters, 20 figures, 2 tables.

Lühendite ja mõistete sõnastik

404-tõrge	Veebiteade, mis viitab sellele, et soovitud lehte ei leitud (*404 Not Found Error*)
AI	Tehisintellekt; inimlaadset otsustus- või õppimisvõimet jäljendav arvutisüsteem (*Artificial Intelligence*)
API	Rakendusliides, mis võimaldab erinevatel süsteemidel või teenustel omavahel suhelda (*Application Programming Interface*)
Bandit	Python-koodi turvaprobleemide tuvastamise tööriist (*Bandit*)
CI/CD	Pidev integratsioon ja pidev tarne (*Continuous Integration and Continuous Delivery*)
Commit	Git'i versioonihaldussüsteemi toiming, mis salvestab hetkeseisundi muudatused hoidlasse; salvestuspunkt koodi ajaloos (*Commit*)
CORS	Turvamehhanism, mis kontrollib, kas veebirakendus tohib päringuid teha teistesse domeenidesse (*Cross-Origin Resource Sharing*)
Cron	Ajakava alusel taustategevusi käivitav süsteem Linuxis (*Cron Scheduler*)
CSS	Stiililehtede keel (*Cascading Style Sheets*)
CSV	Komaga eraldatud väärtuste fail (*Comma-Separated Values*)
D3.js	Dünaamiliste andmevisualisatsioonide teek, mis võimaldab luua interaktiivseid SVG-graafikuid (*Data-Driven Documents*)
Discord	Reaalajas suhtlusplatvorm, mida kasutatakse sageli meeskonnatöö ja arenduskoosolekute pidamiseks (*Discord*)
Django	Pythoni veebiraamistik (*Django*)
Docker	Platvorm tarkvara konteineripõhiseks käitamiseks ja haldamiseks (*Docker*)
Docker Compose	Docker-konteinerite komplektide orkestreerimist võimaldav tööriist (*Docker Compose*)
Dockerfile	Fail, mis kirjeldab konteineri loomise juhiseid (*Dockerfile*)
Eesrakendus	Kliendipoolne rakendus, mis töötab brauseris (*Front-end*)
ESLint	JavaScripti ja TypeScripti koodi staatilise analüüsi tööriist, mis aitab leida probleeme ja jõustada koodistiili (*ESLint*)
Git	Versioonihalduse süsteem, mida kasutatakse koodimuudatuste jälgimiseks ja haldamiseks (*Git*)
GitLab	Tarkvaraarenduse koostööplatvorm (*GitLab*)

HTTP	Hüperteksti edastusprotokoll; protokoll, mida kasutatakse veebis andmete edastamiseks (*HyperText Transfer Protocol*)
ID	Unikaalne tunnus, mida kasutatakse üksuste eristamiseks süsteemis (*Identifier*)
IDE	Integreeritud programmeerimiskeskkond (*Integrated Development Environment*)
Image (Docker)	Konteineri aluseks olev failisüsteemi kujutis koos sõltuvustega (*Docker Image*)
Jest	JavaScripti testiraamistik, mida kasutatakse React-komponentide testimiseks (*Jest*)
JSON	JavaScripti objektnotatsioon; kergekaaluline andmevahetusvorming, mida on lihtne inimestel lugeda ja kirjutada ning masinatel parsida ja genereerida (*JavaScript Object Notation*)
Kasutajakogemus	Kasutaja subjektiivne kogemus ja rahulolu süsteemi kasutamisel (*User Experience*)
Kasutajaliides	Inimese ja tarkvara vaheline suhtluskiht (*User Interface*)
Lint	Automatiseeritud tööriist, mis analüüsib lähtekoodi võimalike probleemide leidmiseks (*Linter*)
LLM	Suur keelemudel, ulatuslikul tekstikorpusel treenitud tehisnärvivõrk (*Large Language Model*)
Live	Toodangukeskkond lõppkasutajatele (*Live Environment*)
Microsoft OAuth	Microsofti autentimisprotokoll, mida kasutatakse kolmanda osapoole sisselogimiseks (*OAuth 2.0 by Microsoft*)
Moodle	Õpihaldussüsteem, mida kasutatakse e-õppe korraldamiseks ja haldamiseks (*Modular Object-Oriented Dynamic Learning Environment*)
NPM	JavaScripti teekide ja tööriistade haldur (*Node Package Manager*)
OpenAI	Tehisintellekti uurimis- ja rakendusettevõte, tuntud oma suurte keelemudelite (nagu GPT) poolest (*OpenAI*)
ORM	Objekt-relatsiooniline peegeldus; võimaldab programmeerijal andmebaasiga suhelda objektide kaudu, otseselt SQL-päringuid kirjutamata (*Object-Relational Mapping*)
Pilet	GitLabis registreeritud tööülesanne või probleemipüstitus, mida kasutatakse projektis konkreetse tegevuse planeerimiseks, jälgimiseks või dokumenteerimiseks (*Issue*)
Pipeline	Andmetöötluse etappidest koosnev jada (*Pipeline*)
PostgreSQL	Relatsiooniline andmebaasi haldamise süsteem (*PostgreSQL*)
Prospector	Python-koodi kvaliteedi kontrollimise tööriist, mis koondab mitmeid analüsaatoreid (*Prospector*)

Pylint	Pythoni koodi staatilise analüüsi tööriist, mis kontrollib vigu, jõustab koodistiili ja pakub koodi parendusettepanekuid (*Pylint*)
Python3	Üldotstarbeline interpreteeritav programmeerimiskeel (*Python 3*)
React	JavaScripti teek kasutajaliideste loomiseks (*React*)
React Testing Library	React-komponentide käitumise testimise tööriist, mis keskendub kasutajavaate kontrollimisele (*React Testing Library*)
REST	Veebirakenduse arhitektuuristiil (*Representational State Transfer*)
Skript	Juhiste kogum, mida arvuti täidab automaatselt (*Script*)
Sprint	Agiilse arenduse ajaühik, mille jooksul realiseeritakse konkreetne hulk ülesandeid (*Sprint*)
SQL	Struktureeritud päringukeel; standardne programmeerimiskeel relatsiooniliste andmebaaside haldamiseks ja nendes olevate andmetega manipuleerimiseks (*Structured Query Language*)
Tagarakendus	Serveripoolne teenus, mis serveerib andmeid kasutajaliidesele (*Back-end*)
Tailwind	CSS-klassidel põhinev kujundusraamistik, mis võimaldab kiiresti stiilida kasutajaliidest (*Tailwind CSS*)
TypeScript	Programmeerimiskeel, mis on JavaScripti ülemhulk, lisades staatilise tüübikontrolli ja muid täiustusi (*TypeScript*)
URL	Veebiaadress, mis määrab ressursi asukoha internetis (*Uniform Resource Locator*)
Vue3/Vue	Progressiivne JavaScripti raamistik üheleheliste rakenduste loomiseks (*Vue.js 3*)
Välearendus	Arendusmeetod, mis keskendub kiirele prototüüpimisele ja reageerimisele muutustele (*Agile Development*)
XML	Laiendatav märgistuskeel; märgistuskeel, mis defineerib reeglid dokumentide kodeerimiseks vormingus, mis on nii inimloetav kui ka masinloetav (*Extensible Markup Language*)

Sisukord

Jooniste loetelu

Tabelite loetelu

1. Sissejuhatus

1.1 Taust

Cognate on Tallinna Tehnikaülikooli IT-teaduskonnas 2022. aastal algatatud tarkvaraplatvorm, mille eesmärk on vähendada õppejõudude ajakulu meeskonnaprojektide hindamisel ja ühtlustada hindamiskriteeriume, automatiseerides GitLabi [1] hoidlatest kogutavate metaandmete analüüsi ning visualiseerimist. Platvormi arendus on kulgenud kolmes järjestikuses bakalaureusetöös:

- 2022 — „GitLabi projektide metaandmete analüüdi ökosüsteem“ [2]. Loodi esimene töötav prototüüp: Django-põhine [3] REST-tagarakendus, Vue [4] + D3.js [5] eesrakendus ning API-d GitLabi andmete sissetoomiseks ja visualiseerimiseks.
- 2023 — „Cognate automatiseerimine ja uue kasutajaliidese loomine“ [6]. Täiendati automatiseerimist ja loodi intuitiivsem kasutajaliides: eesrakendus viidi TypeScript + Vue 3 peale, parandati kriitilised vead ning lisati uusi automaatseid andmetöötlus funktsioone.
- 2024 — „GitLabi projektide metaandmete analüüsi ökosüsteemi Cognate'i kasutajamugavuse parandamine ja arendustöö lihtsustamine“ [7]. Fookus kasutajamugavusel ja arenduse sujuvusel: käsitsi konfigureerimise maht vähenes, hinnete eksport Moodle'isse lisandus, loodi Microsoft OAuth logimise võimalus ning tagarakendus refaktoreeriti (päringute optimeerimine, logimine, testid).

Cognate on kujunenud TalTechi tarkvaraarenduse projektiainete lahutamatuks tööriistaks, pakkudes õppejõududele ühtset vaadet projektide edenemisele. Samas on nii senised arendusetapid kui ka kursuste käigus kogutud tagasiside toonud päevavalgele arenguvõimalused, millele käesolev töö järgnevates peatükkides keskendub.

1.2 Probleem

Rakendus Cognate on loodud selleks, et lihtsustada abiõppejõudude tööd meeskonnaprojektide jälgimisel ja tagasisidestamisel, kuid mitmed tehnilised ja disainilised kitsaskohad piiravad selle tegelikku kasutegurit.

Süsteem on siiamaani kuvanud projektide andmeid ebatäpselt ning mõned funktsionaalsused ei tööta ootuspäraselt.

Kasutajaliides on küll stiililt kooskõlas TalTechi disainisüsteemiga [8], ent realiseeritud Vue3-raamistiku [4] abil. Kuna TalTechi ametlik stiiliraamat ja komponentide teek on React-põhine [9], takistab erinev raamistik täielikku vastavust ülikooli kasutajaliidese standarditele ning raskendab tulevasi hooldus- ja arendusprotsesse.

Kuigi õppejõud ja abiõppejõud saavad vaadelda GitLabist kogutud tööpanuse andmeid, näiteks ajakulu või piletite arvu, peab hindamine toimuma suures osas ikka käsitsi. See tähendab, et igat projekti ja seotud pileteid tuleb üksikasjalikult ja eraldi sirvida ning hinnata, mis on ajamahukas ja ebaühtlane protsess, eriti suuremate meeskondade puhul. See manuaalne töökoormus takistab kiiret ja objektiivset tagasisidestamist.

Lisaks on rakenduses senini realiseerimata mehhanismid, mis võimaldaksid tudengitel saada jooksvalt personaalset ja motiveerivat tagasisidet — mistõttu Cognate'i otsene kasutegur õppijatele jääb täna oluliselt alla selle potentsiaalile.

1.3 Arendusmetoodika

Arendusprotsess tugineb paindlikule, välearenduse [10] töökorraldusele, mille keskmes on regulaarne suhtlus ja pidev iteratsioon. Iga nädal toimub juhendajaga Discordi [11] platvormi vahendusel struktureeritud koosolek, kus vaadatakse üle eelmise sprindi tulemused, seatakse uued eesmärgid ning täpsustatakse tehnilisi küsimusi.

Kliendiga kohtub meeskond harvemini, kuid näost-näkku, et valideerida olulisemaid teetähiseid ja koguda põhjalikumat tagasisidet lõppkasutajate vaatenurgast.

Kõik arendustunnid, ülesanded ja kommentaarid logitakse GitLabi hoidla vastavatesse piletitesse, mis võimaldab läbipaistvalt hinnata töömahtu ning planeerida järgnevaid

iteratsioone.

Igapäevaselt korraldavad tiimiliikmed arutelusid ja koodimissessioone: päevased ühiskoodimised lahendavad kriitilisi takistusi ning õhtused refleksioonid aitavad fikseerida õppetunnid.

2. Rakenduse ülevaade ja arendusvajadused

Cognate on aktiivses kasutuses meeskonnaprojektide õppeainetes, kus see aitab abiõppejõududel jälgida tudengite tööpanust ja projekti kulgu. Kasutajad saavad vaadelda andmeid, mis on kogutud GitLabi hoidlatest, ning hinnata nende põhjal individuaalset ja grupipõhist panust.

Süsteemi senise kasutamise käigus on projekti klient ning rakenduse lõppkasutajad – peamiselt abiõppejõud – välja toonud mitmeid kitsaskohti ja puudujääke, mis takistavad Cognate'i täielikku potentsiaali. Järgnevates alajaotustes on koondatud peamised valdkonnad, mis vajavad tehnilist ümberkujundamist, täiendamist või paremat kasutuskogemust.

2.1 Gruppide haldus

Grupid või projektigrupid on Cognate'i keskne üksus, mis esindab tavaliselt ühte konkreetset õppeainet (näiteks „Tarkvaraarenduse projekt“ või „Erialatutvustus“). Iga grupp sisaldab kohandatavat hindamisstruktuuri ehk hindamispuud, mille alusel hinnatakse gruppi kuuluvate projektide kvaliteeti ja tudengite panust. Projektid tuuakse GitLabist automaatselt süsteemi peale grupi loomist, võimaldades õppejõududel hallata hinnatavaid üksusi ainepõhiselt. Grupp määrab ka, kellel on õigus vastava aine projektidele ligi pääseda, neid muuta ja hinnata.

Cognate'is toimub kogu ligipääsude, projektide ja hindamiskriteeriumite haldamine grupi kontekstis, mistõttu on korrektne ja turvaline grupihalduse süsteemi usaldusväärse toimimise eelduseks.

Kasutajad on juhtinud tähelepanu, et grupihalduse funktsionaalsus on olnud piiratud: gruppe ei saa mugavalt eemaldada, olemasolevatesse gruppidesse uute kasutajate lisamine on ebaloogiline ja kohmakas ning turvariskina on välja toodud, et ka piiratud õigustega kasutajad näevad GitLabi andmepäringuteks kasutatavat privaatset ligipääsuluba.

2.2 Andmete õigsus

Rakendus kuvab mitmetes vaadetes ekslikke andmeid, näiteks vale koodiridade arv või ebatäpne ajakulu, mis seab ohtu hindamise objektiivsuse. Puudub lihtne viis andmete täpsust valideerida.

2.3 TalTechi eesrakenduste ühtsustamine

Klient on rõhutanud vajadust viia Cognate kooskõlla ülikooli disainisüsteemiga. Vue-põhine eesrakendus raskendab React-tehnoloogial põhinevate ametlike komponentide kasutust ning suurendab hoolduskoormust. Üleminek Reactile võimaldaks TalTechi disainisüsteemi komponentide sujuvat integreerimist läbi NPM [12] ja CI/CD [13] protsessi.

2.4 Puudulik kasu tudengitele

Hetkel puudub tudengitel selge ja pidev ülevaade oma tööpanusest ning edenemisest projekti jooksul. Töö tulemuslikkus ja tagasiside jõuab nendeni alles sprindi lõpus. Kliendi hinnangul võiks tudengitele pakkuda sarnast statistilist vaadet nagu õppejõududele, mis toetaks iseseisvat õppimist ja eneseanalüüsi.

2.5 Hindamise automatiseerimise vajadus

Üks projekti kliendi ja lõppkasutajate poolt tõstatatud arenguvajadus on hinnete ja tagasiside suurem automatiseerimine. Hetkel toimub hindamine valdavalt käsitsi – õppejõud vaatavad tudengite tööpanust GitLabis ja sisestavad punktid süsteemi vastavalt oma subjektiivsele hinnangule. See lähenemine on ajamahukas.

Kuna Cognate juba kogub GitLabist piletite sisu, logitud töötunde ja muud metaandmestikku, on süsteemil potentsiaal pakkuda poolautomaatset hindamist, mis põhineb:

- tehnilistel kriteeriumitel, nagu piletite olemasolu, hinnangute ja tööaja lisamine;
- sisu kvaliteedi hindamisel, näiteks pileti pealkirja ja kirjelduse selgus.

Sellise analüüsikihi lisamine võimaldaks anda õppejõule objektiivsemat sisendit hindamiseks ning tudengile personaalsemat ja kiiremat tagasisidet. See toetaks ka eesmärki muuta hindamisprotsess läbipaistvamaks ja järjepidevamaks.

3. Tehnoloogiline lahendus

Käesolevas peatükis kirjeldatakse Cognate'i arenduse käigus tehtud tehnoloogilisi valikuid, muudatusi ning nende põhjuseid. Tuuakse välja varasemad kitsaskohad ja analüüsitakse, millised lahendused võimaldasid parandada rakenduse töökindlust, kasutusmugavust ja vastavust ülikooli kasutajaliidese standarditele. Peatüki alajaotustes käsitletakse eraldi kasutatud tehnoloogiate analüüsi ning CI/CD töövoogude ülesehitust.

3.1 Kasutatud tehnoloogiad

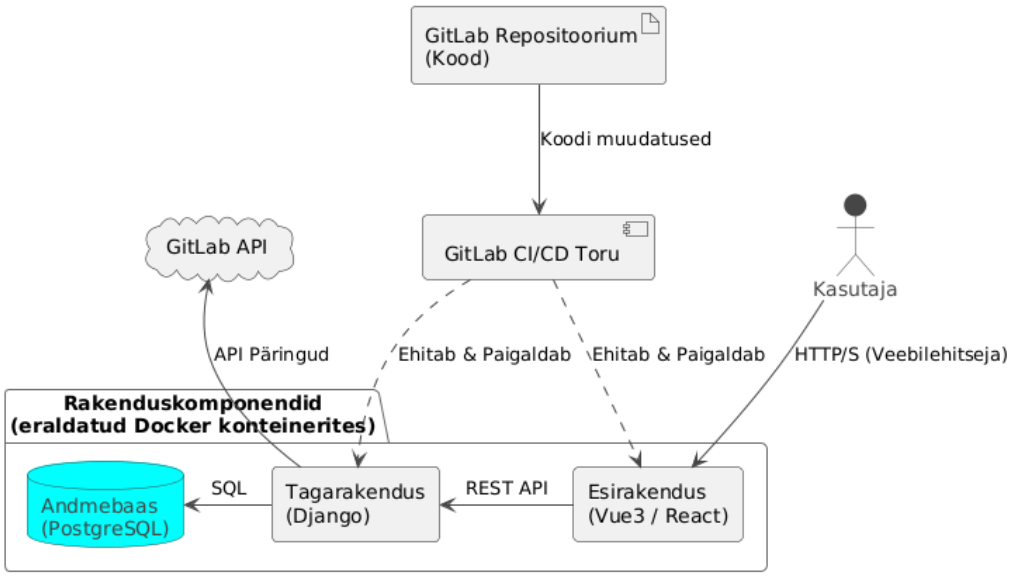
Cognate'i süsteem koosneb kahest peamisest komponendist: tagarakendusest, mis on ehitatud Django veebiraamistiku peale ning kasutab PostgreSQL [14] andmebaasi, ja eesrakendusest, mis oli algselt loodud Vue 3 raamistikus.

Vue valik varasemates arenduse etappides põhines tiimiliikmete eelnevatel kogemustel. Praktikas osutus see aga piiravaks, kuna TalTechi ametlikud kasutajaliidese komponendid on loodud Reacti jaoks. Vue kasutamine tähendas, et ametlikke disainisüsteemi komponente tuli kas manuaalselt üles ehitada või asendada sarnaste analoogidega, mis viis visuaalsete vastuoludeni ning tõstis hoolduskoormust.

Reacti kasutuselevõtt võimaldab paremat kooskõla TalTechi disainiraamistikuga ja lihtsamat komponentide haldust läbi NPM-i. Reacti kasuks räägivad ka kogukonna toetus, tööriistade küpsus ning lai dokumentatsioon. Vue raamistikust migreerides loodi ka ühtsed haagid (ingl k. *hook*) päringute ja autentimise jaoks, mis vähendas dubleeritud loogikat ning parandas hooldatavust.

Andmebaasina jäädi kindlaks PostgreSQL-ile – valik, mis on ennast varasemates iteratsioonides tõestanud stabiilsuse, jõudluse ja SQL-standarditele vastavusega. Kuna kogu rakendus on tihedalt seotud GitLabi API-ga [15], tuli luua ka robustne mehhanism GitLabist imporditavate andmete valideerimiseks ja struktureerimiseks.

Joonisel 1 on kujutatud Cognate'i süsteemi arhitektuurne ülesehitus. Rakendus koosneb kolmest põhilisest komponendist: PostgreSQL andmebaas, Django-põhine tagarakendus ja React-põhine eesrakendus. Kõik komponendid on konteineriseeritud ja juurutatakse GitLabi CI/CD töövoo kaudu. Süsteemi iga muudatus GitLabis käivitab automaatse lintimise, testimise ja paigalduse valitud keskkonda. Kasutaja pääseb eesrakendusele ligi veebibrauseri kaudu.



Joonis 1. Cognate'i süsteemi ülesehitus: CI/CD toru, rakenduskomponendid ja konteineripõhine paigutus.

3.2 Pideva integreerimise ja tarnimise protsessid (CI/CD)

Cognate'i tarnevoog on üles ehitatud GitLab CI [16] + Docker [17] kombinatsioonile, mis võimaldab iga koodimuudatust automaatselt lintida [18], testida, pakendada konteineriks ja juurutada (ingl k. *deploy*) õigesse keskkonda. Praktikas tähendab see, et arendajate töö liigub ühest git harust teise ilma käsitsi serverisse sisenemata; ühtlasi logib GitLab iga sammu, mis lihtsustab tagasipööramist ja auditeerimist.

Rakendust hoitakse kahes selgelt eristatud keskkonnas – arendus ja *live*. Keskkondade käitumist juhivad keskkonnamuutujad (`ENV`), mida Docker Compose annab konteineritele edasi. Näiteks `ENV=staging` laadib `.env.dev` faili, kus on arendusbaasi URL-id, lubatud CORS-domeenid ja silumisrežiim; `ENV=production` aga lülitab silumise välja, seab rangemad turvasätted ja viitab toodangukeskkonna andmebaasile. Nii saab sama koodibaas jooksutada erineva konfiguratsiooniga ilma ridasid ümber kirjutamata.

Konveieri (ingl. k *pipeline*) etapid on järgmised:

- Lintimine – Koodi kvaliteeti ja stiili kontrollitakse automaatselt: eesrakenduses teegi ESLintiga [19] ning tagarakenduses teekide Prospectori [20] ja Banditiga [21]. Vigade ilmnemisel konveier peatub.
- Ehitamine – Docker Compose'i ja Dockerfile'i abil ehitatakse tagarakendusele uus Docker *image*. See hõlmab Pythoni ja süsteemsete sõltuvuste paigaldamist, rakenduse koodi ning *cron*-tööde konfiguratsiooni lisamist konteinerisse.
- Testimine – Eesrakenduse komponente testitakse Jesti [22] ja React Testing Library [23] abil, tagarakendust aga Django sisseehitatud test-jooksutajaga (ingl k. *runner*). Testide ebaõnnestumine peatab edasise juurutamise.
- Juurutamine – Docker Compose peatab vanad konteinerid, käivitab andmebaasi migratsioonid ning seejärel uued, värskelt ehitatud rakenduse- ja andmebaasikonteinerid vastavas keskkonnas (arendus või *live*). Lõpuks aktiveeritakse rakendusesisesed *cron*-tööd. Põhiharu *live*-juurutus nõuab manuaalset kinnitust.

Selline CI/CD-korraldus vähendab käsitsi sekkumist, tagab konfigureerimisvigade varajase tabamise ning teeb väljalasked jälgitavaks – iga keskkonna konteineri juurde talletub täpselt see koodiversioon ja konfiguratsioon, millega juurutus tehti.

4. Realisatsioon

Selles peatükis tuuakse detailselt välja arendustööd, mis viidi läbi Cognate'i rakenduse täiustamiseks, tuginedes eelnevates peatükkides esile toodud kitsaskohtadele ja arendusvajadustele. Töö käigus pöörati tähelepanu eelkõige nende funktsionaalsuste parandamisele, mis olid olulised projekti kliendile ja lõppkasutajatele ehk abiõppejõududele ning mille parendamine aitas suurendada süsteemi töökindlust, kasutusmugavust ja väärtust õppetöös.

Iga alajaotus keskendub kindlale probleemivaldkonnale ning sisaldab:

- probleemi või vajakajäämise lühikirjeldust;
- selle aluseks olevat tehnilist või arhitektuurilist olukorda;
- tehtud muudatusi, mille eesmärk oli probleemi lahendada;
- lühikest refleksiooni uue lahenduse toimivuse või mõjutatud kasutajakogemuse kohta.

Peatükis käsitletakse nii eesrakenduse kui ka tagarakenduse muudatusi: näiteks grupihalduse loogika uuendamist, andmete täpsuse parandamist, eesrakenduse migreerimist React-platvormile ja uute tudengivaadete loomist. Lisaks selgitatakse, kuidas muudatused testiti ja kuidas need versioonihalduse kaudu kasutajateni jõudsid.

4.1 Gruppide haldus

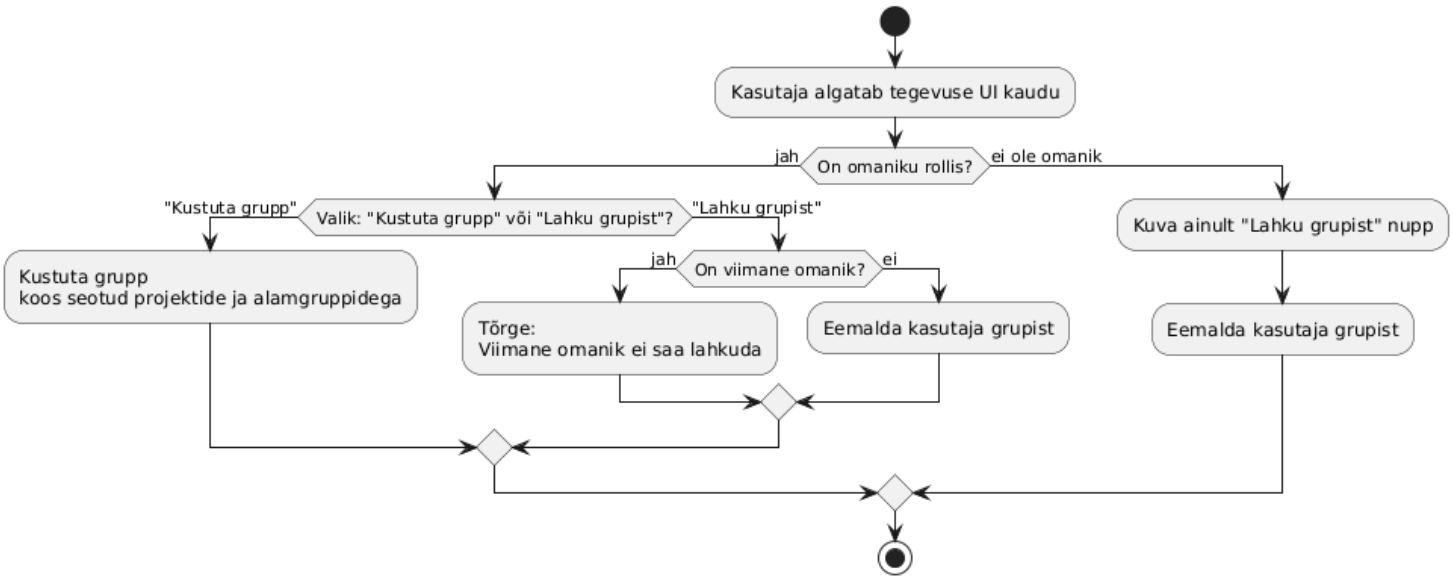
Korralik gruppide haldus on oluline nii süsteemi turvalisuse, andmete korrektsuse kui ka kasutusmugavuse seisukohalt. Arenduse käigus ilmnes mitmeid kitsaskohti, mis takistasid sujuvat grupihaldust – näiteks puudus varasemalt võimalus gruppe kustutada, ligipääs õigustele oli ebapiisavalt piiratud ning liikmete lisamine nõudis mitmeid eraldi samme. Käesolevas peatükis käsitleme tehtud muudatusi ja täiustusi, mille eesmärk oli lahendada need probleemid ja parandada üldist halduse loogikat.

4.1.1 Gruppide eemaldamine

Cognate'i varasemas versioonis ei olnud võimalik projektigruppe süsteemist eemaldada. See tähendas, et ekslikult loodud või testimiseks lisatud grupid jäid süsteemi alles, muutes andmestiku segaseks ja raskendades navigeerimist. Kustutamine sai toimuda ainult andmebaasi kaudu käsitsi, mis eeldas tehnilisi teadmisi ja oli riskantne.

Selle lahendamiseks loodi uue loogikaga API funktsionaalsus `DeleteProjectGroup`, mis võimaldab projektigrupi eemaldamist otse kasutajaliidesest. Kui päringu algatab kasutaja, kellel on grupis omanikuõigused, ning ta on viimane omanik, kustutatakse andmebaasist nii grupp kui ka sellega seotud projektid või alamgrupid. Kui grupis on veel teisi omanikke, saab ka omanik lihtsalt grupist lahkuda.

Grupist lahkumise toetamiseks loodi eraldi API funktsionaalsus `LeaveProjectGroup`, mis võimaldab igal kasutajal vabatahtlikult grupist eemalduda. See kasutab sisuliselt kasutajalt rolli eemaldamise loogikat, kuid sisaldab ka täiendavat kontrolli, mis takistab viimase omaniku lahkumist (`ErrorCode.CANNOT_REMOVE_LAST_OWNER`). Nii tagatakse, et igal projektigrupil on alati vähemalt üks omanikuõigustes kasutaja ning säilib grupi haldamise võimekus. Joonisel 2 on kujutatud grupist lahkumise ja kustutamise otsustusloogika.



Joonis 2. Grupist lahkumise ja kustutamise loogiline töövoog kasutaja rolli alusel.

Selle loogika toetamiseks lisati ka eesrakendusse vastavad nupud: omaniku rollis kasutaja näeb mõlemat – nii grupist lahkumise kui ka kustutamise nuppu. Kõik teised kasutajad

näevad ainult grupist lahkumise võimalust. Selline lähenemine tagab, et grupi kustutamine toimub vaid teadlikult ning ei ole ainus viis omanikel grupist väljumiseks.

4.1.2 Grupi sätete ohukohad

Projekti klient ja rakenduse kasutajad juhtisid tähelepanu turvariskile, mille puhul kõik grupiliikmed said näha GitLabi privaatset juurdepääsuluba, mida kasutati projektide andmete sünkroniseerimiseks. See tähendas, et ka piiratud õigustega kasutajatel oli ligipääs tundlikule teabele. Samuti võisid ka kasutajad muuta kogu grupi sätteid nagu nimi, grupi identifikaator, kirjeldus ja muu selline.

Turvalisuse ja halduse selguse suurendamiseks rakendati rangem rollipõhine juurdepääsukontroll. Selleks kaitsti kõik vastavad API otspunktid, mis käsitlevad grupi objektide muutmist (peamiselt `PUT` ja `PATCH` meetodid), ning ligipääs nendele piiratud ainult kasutajatele, kellel on grupis omaniku roll.

Kuigi osa muudatusi, näiteks GitLabi andmete sünkroniseerimise käivitamine, on lubatud ka administraatoritele — on kriitiliste sätete muutmise funktsionaalsus piiratud ainult omanikele. Samuti eemaldati privaatse juurdepääsuloa nähtavus kasutajaliideses, et vältida selle väärkasutust.

4.1.3 Liikmete gruppi lisamine

Kasutajad tõid esile, et liikmete lisamine gruppidesse oli varasemalt mitmeastmeline, ajamahukas ja vigadele avatud. Uuele kasutajale määrati esmalt vaikimisi tühi roll, mis ei võimaldanud ligipääsu projektidele, mille tõttu kuvati talle 404-tõrge. Seejärel pidi grupihaldur määrama eraldi rolli, nagu „administraator“, et kasutaja saaks projektidele ligi, ning ka sellisel juhul ligipääs ei töötanud.

Selle töövoo parandamiseks muudeti API loogikat vaates `ManageGroupInvitationView`, kus nüüd saab kutse loomisel kohe määrata kasutajale sobiva rolli. Rolli valik valideeritakse – lubatud on vaid süsteemis defineeritud rollid, ning kontrollitakse, et kutsuv kasutaja ei saaks määrata endast kõrgemat rolli (näiteks administraator ei saa kutsuda omanikku). Kui kutse on juba olemas, uut ei looda. Kui kutse vastu võetakse, tegeleb `process_group_invitation` loogika selle realiseerimisega: lisatakse vastav `UserProjectGroup`

kirje andmebaasi ja kustutatakse kutse. Eesrakenduses lisati selleks eraldi rippmenüü, kus kutsuja saab valida, milline roll määrata. Nii muutus lisamisprotsess ühetaoliseks ja turvaliseks, välistades eelnevalt esinenud vead ja liigse käsitöö.

4.2 Andmete õigsus

Cognate'i usaldusväärne toimimine eeldab, et rakendus kuvab GitLabi põhjal kogutud andmed korrektselt, et tugineda nendele hindamisel ja projekti analüüsimisel ning tagasisidestamisel. Kasutajad tõid välja mitmeid juhtumeid, kus andmed – eriti koodiridade arv ja ajakulu – olid valed või eksitavad. Selle aluseks olid mitmed arhitektuursed piirangud ja andmemudeli puudujäägid, mida järgnevalt käsitletakse.

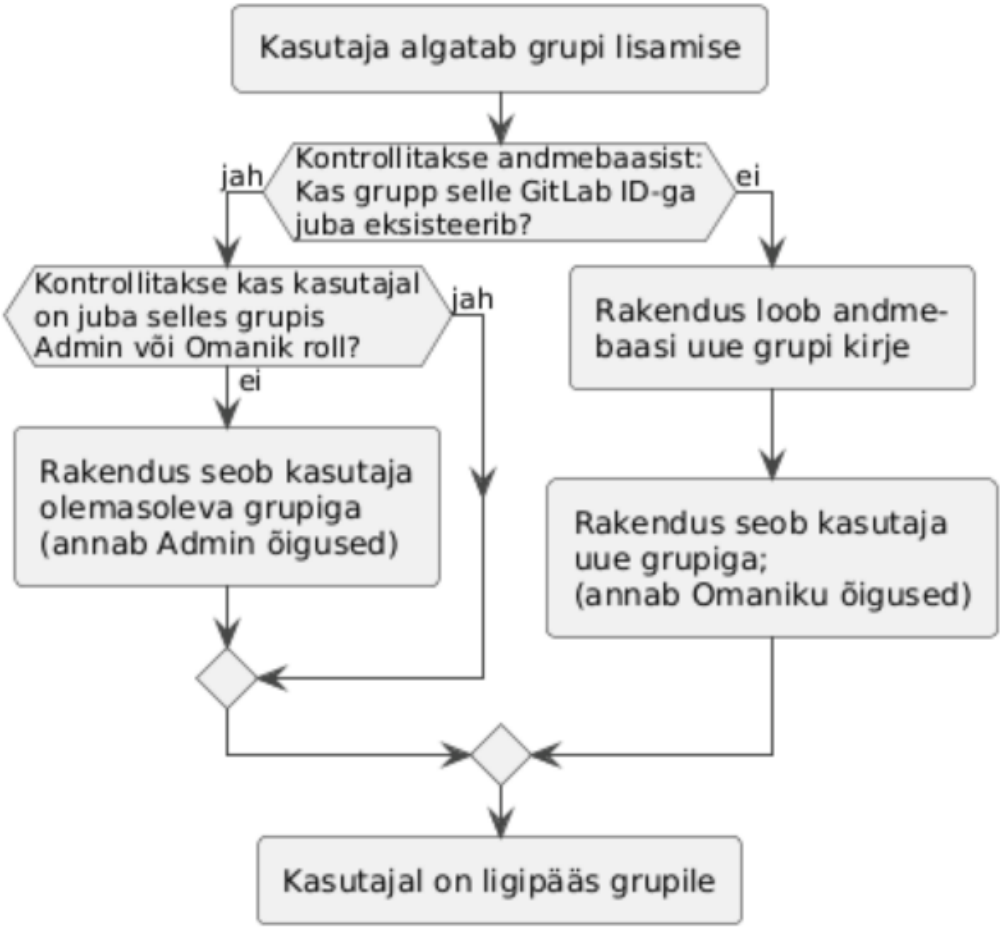
4.2.1 Koodiridade ja aja vale kuvamine

Cognate'i varasem andmemudel ei piiranud, mitu korda sama GitLabi projekt või grupp võis süsteemi sisestatud olla. Mudelites `Project` ja `ProjectGroup` puudusid unikaalsuskitsendused vastavalt `gitlab_id` ja `group_id` väljadele. See tähendas, et kasutajatel oli võimalik luua mitu instantsi ühest ja samast projektist või isegi terved koopiad samast grupist. Selle tulemusena salvestati sama projekti kohta mitu andmekomplekti, mistõttu kogunes andmebaasi duplikaatandmeid (nt `Commit`, `Issue`, `TimeSpent` kirjed), mis andmepäringutes mitmekordselt arvesse võeti ja eesrakenduses eksitavalt kuvati.

Mõnel juhul esines selgeid anomaaliaid, näiteks negatiivne ajakulu, mis oli kasutajaliideses kergesti märgatav ja viitas ilmselgele veale. Seevastu olukordades, kus rakenduses kuvatud ajakulu erines GitLabis logitud väärtusest vaid mõne tunni võrra, oli viga visuaalselt raskesti märgatav ning võis jääda hindajale täiesti märkamatuks. Kuna andmete täpsust ei olnud võimalik süsteemi kaudu kontrollida, pidi abiõppejõud hindamise objektiivsuse tagamiseks kõik projekti GitLabi pileteid käsitsi läbi vaatama ja iga tudengi logitud ajad eraldi kokku liitma. Selline lähenemine oli ajamahukas ja muutis automatiseeritud statistika praktilise kasutamise ebausaldusväärseks.

Cognate'i kasutajatega peetud arutelude ja tagasiside põhjal selgus, et praktiline vajadus luua ühest ja samast GitLabi grupist või projektist mitu instantsi Cognate'i keskkonda puudub. Kasutajate hinnangul piisab täielikult ühest keskseks tõeallikaks olevast grupi- või projektiesindusest.

Probleemi lahendamiseks kehtestati süsteemis unikaalsusnõuded. Mudelile `Project` lisati `gitlab_id` väli, millele rakendati unikaalsusloogika rakenduse tasandil. Mudelile `ProjectGroup` lisati samuti `group_id` väli. Nüüd, kui kasutaja püüab API kaudu (`ProjectGroupView` `POST` meetodiga) lisada gruppi, mille `group_id` on juba süsteemis olemas, tuvastatakse see ning duplikaadi loomise asemel antakse kasutajale automaatselt administraatori õigused juba eksisteerivale grupile (kui tal neid juba pole). Sarnaselt, projektide sünkroniseerimisel GitLabist (nt funktsioonis `_ensure_project_repo_and_bridge_exists`) kontrollitakse kõigepealt, kas projekt vastava `gitlab_id`-ga on juba olemas. Kui on, siis kasutatakse olemasolevat `Project` objekti, vältides duplikaatprojekti loomist. Grupi lisamise vooskeem, mis hõlmab nii uue grupi loomist kui ka olemasolevale juurdepääsu andmist, on esitatud joonisel 3.



Joonis 3. Grupi lisamise loogika Cognate'is.

Loodud lahenduse tulemusena välistab Cognate'i andmemudel nüüd olukorrad, kus süsteemi lisatakse kogemata mitu identset gruppi või projekti, mis varasemalt põhjustas

ebausaldusväärsete andmete tekkimist. Kuna duplikaatkirjete salvestamine on arhitektuurselt välistatud, ei teki enam olukordi, kus sama projekti tegevused summeeritakse mitmekordselt või kuvatakse eksitavalt vales kontekstis. Muudatuse kaudse mõjuna on vähenenud ka andmete hulk, mida süsteem peab töötlema, ning ajakuluga seotud päringute jõudlus on paranenud, kuna dubleeritud andmeid ei salvestata ega kaasata arvutustesse.

4.2.2 Projekt mitmes grupis

Ajakulu ja koodiridade valesti kuvamise probleemi lahendamise käigus ilmnes uus, senini käsitlemata juhtum: olukorrad, kus projekte on mitmesse gruppi jagatud ning üks ja sama GitLabi projekt esineb ainsa tõe allikana mitmes erinevas GitLabi grupis. Praktikas tähendab see, et näiteks kahel erineval õppeainel või kursusel võib olla kasutusel sama arendusprojekt, mille tudengid töötavad ühisel hoidlal, kuid õppetöö ja hindamise kontekst on erinev.

Varasema andmemudeli korral, kus `Project` oli otse seotud ühe `ProjectGroup`-iga (`project.project_group` võõrvõti), oleks mõlema grupi lisamisel Cognate'i süsteemi (juhul kui projektide `gitlab_id` unikaalsust ei jõustataks) loodud eraldi instantsid samast projektist. See põhjustanuks dubleeritud andmete tekkimist.

Kuna eelnevas etapis rakendatud muudatused kehtestasid projekti `gitlab_id` alusel unikaalsusnõude rakenduse loogikas, ei olnud enam võimalik sama projekti teist korda süsteemi lisada – ka juhul, kui see kuulus paralleelselt mitmesse GitLabi gruppi. Kuigi see hoidis ära dubleeritud andmete tekke, takistas see samas ka legitiimseid olukordi, kus sama projekt on seotud mitme erineva grupiga ja vajab kontekstipõhist hindamist.

Probleem tuli lahendada viisil, mis välistab dubleeritud andmete tekkimise, ent samas võimaldab ühel ja samal GitLabi projektil kuuluda mitmesse erinevasse gruppi ning tagab, et projekti liikmeid saab hinnata grupipõhiselt sõltuvalt kontekstist.

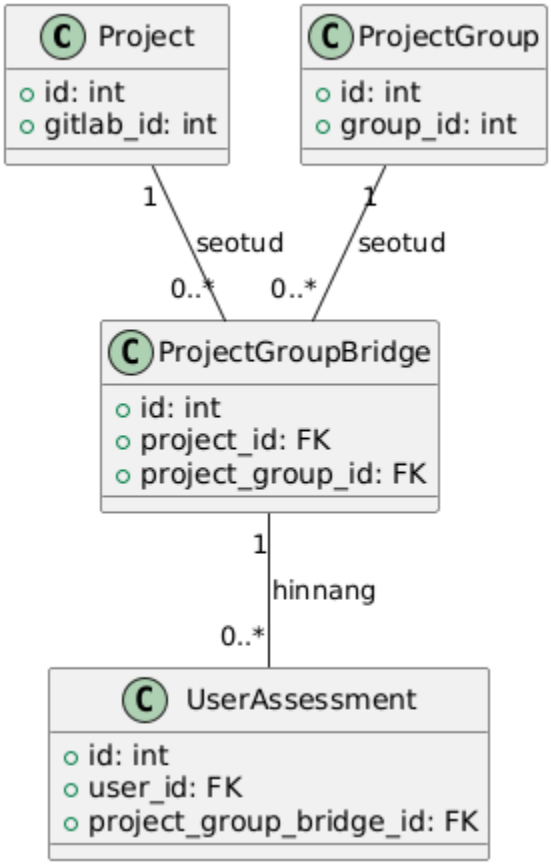
Selleks loodi andmebaasi täiendav seostabel `ProjectGroupBridge`. See mudel implementeerib `ManyToManyField` seose `Project` ja `ProjectGroup` vahel, võimaldades ühel projektil olla seotud mitme grupiga ja ühel grupil sisaldada mitut projekti. Antud struktuur määratleb, milliste Cognate'i gruppide alla konkreetne GitLabi projekt kuulub. Kui süsteemi lisatakse uus grupp, mis sisaldab juba varem olemasolevat projekti (tuvastatud `gitlab_id`

kaudu), ei looda enam eraldi `Project` kirjet – selle asemel lisatakse `ProjectGroupBridge` tabelisse uus seos olemasoleva projekti ja uue grupi vahel.

Järgmisena viidi läbi andmemudeli muudatused hindamissüsteemis. Kõige olulisem muudatus oli `UserAssessment` mudelile `project_group_bridge` võõrvõtme lisamine. See võimaldab siduda tudengi hinde konkreetse projekti ja grupi kontekstiga. Samuti uuendati `UserAssessment` mudeli `unique_together` kitsendust, et see sisaldaks ka `project_group_bridge` välja, tagades, et iga tudengi hinne kategooria kohta on unikaalne kindla projekti-grupi kombinatsiooni piires.

Muudatustega kaasnesid ka vastavad täiendused tagarakenduse päringuloogikasse (funktsioonis `get_milestone_data_for_project` ja funktsioonis `__get_projects_data`), mis nüüd arvestavad `bridge_id`-ga, et andmeid töödelda ja kuvada grupipõhiselt – isegi juhul, kui mitu gruppi jagavad sama projekti.

Selline lähenemine võimaldab ühe GitLabi projekti kontekstis anda hinnanguid mitmes Cognate'i grupis sõltumatult, toetades paindlikku hindamist ning hoides samal ajal ära andmete dubleerimise ja struktuursete vastuolude teket. Joonisel 4 on kujutatud uue andmemudeli seosed projektide, gruppide ja hinnete vahel.



Joonis 4. Projektide ja gruppide seos ProjectGroupBridge kaudu ning selle mõju hindamisele.

4.3 Muudatuste avalikustamine

Projekti klient tõi välja, et varasemates Cognate'i arendusprotsessides on esinenud probleeme puuduliku versioonihalduse ja muudatuste läbipaistvusega. Kuigi arendusi ja parandusi on eelnevalt testitud, ei ole nende *live*-keskkonda viimisega kaasnenud piisavat dokumentatsiooni ega selget ülevaadet tehtud muudatustest. See on tekitanud raskusi süsteemi haldamisel ja arendusloogika mõistmisel, eriti uute arendajate või huvigruppide kaasamisel.

Sellest lähtuvalt otsustasime kogu arendustöö käigus kogunenud muudatused koondada ühtseks väljalaskeks, mis on versiooninumbriga tähistatud ning mille kohta on koostatud ülevaatlik kirjeldus. See lähenemine võimaldab paremat jälgitavust, lihtsustab testimist ning toetab süsteemi järjepidevat hooldust ja laiendamist.

4.3.1 Väljalase arenduskeskkonda

Kõik tehtud muudatused testiti esmalt lokaalselt ja seejärel arenduskeskkonnas, kuhu need väljastati koondatud kujul eraldi väljalaskena. Arenduskeskkonnas viidi läbi funktsionaalne- ja regressioonitestimine, et veenduda, et kõik funktsionaalsused töötavad koostoimes ja muudatused ei põhjusta tagasilööke olemasolevates töövoogudes. Dokumentatsiooni tasandil loodi muudatuste fail, kuhu kirjeldati versioonis tehtud täiendused, parandused ja olulised tehnilised otsused.

4.3.2 Väljalase *live*-keskkonda

Pärast edukat testimist arenduskeskkonnas viidi muudatused üle *live*-keskkonda. Enne väljalaset otsustati olemasolev andmebaas tühjendada ning juurutada ümberkujundatud skeem koos uut tüüpi seostetabeliga, mis lahendab projekti- ja grupivahelised seosed. See tagas andmete sisemise loogika ühtsuse ja välistas võimalikud konfliktid varasemate, arhitektuuriliselt ebajärjekindlate kirjete vahel. Väljalase teostati kontrollitult CI/CD konveieri (inglise k. *pipeline*) kaudu, mille iga etapp oli dokumenteeritud ja vajadusel kordustööna teostatav. Süsteemile määrati versiooninumber ning muudatuste sisu dokumenteeriti nii tehnilisel kui kasutajakesksel tasandil.

Oluline kaalutlus oli otsus mitte kirjutada andmete migratsiooniskripti. Kuna Cognate'i süsteem ei salvesta lokaalselt unikaalseid andmeid, vaid tugineb suuresti GitLabist päritud metaandmetele ja statistikale, ning loodud hinnangud eksporditakse CSV faili, ei olnud vajadust säilitada varasemat andmestikku. Arvestades, et andmebaasis oli kujunenud ulatuslik dubleerunud gruppide ja projektide kogum, peeti otstarbekaks andmestik täielikult nullida ja rakendus puhtalt üles seada. Et vältida andmekao riski, tehti enne andmebaasi tühjendamist täiskoopiad kõigist andmetest, mis võimaldanuks vajadusel taastamist või tagantjärele kontrollimist.

4.4 Eesrakenduse migreerimine

Cognate'i eesrakendus oli peale viimast lõputööd Vue3-raamistiku peal ehitatud, kuid TalTechi ametlik disainisüsteem ja komponenditeek on React-põhised. See lahkheli põhjustas stiililiste vastuolude kõrval ka hoolduskoormuse kasvu: igas uuemas projekti- või ülikoolipoolses uuenduses tuli Vue-spetsiifilisi ülekatteid pidevalt käsitsi kohandada. Ot-

sus langetati, et ühtlustada kasutajaliidese tehnoloogiline baas ning vähendada tulevasi ümberehituskulusid.

4.4.1 Keskkonna seadistamine

Migreerimisprotsess oli järkjärguline. Väiksema mahuga lõputööga seoses tegeles meist eraldi tiim React-rakenduse algseadistamisega: nad lõid projektistruktuuri, rakendasid esmase sisselogimis- ja autentimismooduli ning panid paika ka algse profiilivaate. Samuti projekteeriti uus CI/CD-jooksutaja, mis ehitas koodi ja viis muudatused testikeskkonda peale manuaalset kinnitust.

4.4.2 Olemasoleva refaktoreerimine ning lisakonfiguratsioon

Kuigi React rakenduse põhistruktuur koos esmaste autentimiste, kasutajaprofiili ja CI/CD lahendusega oli toimiv, keskenduti järgnevalt koodi kvaliteedi tõstmisele. Analüüsi fookuses olid kriitilised teenused andmepäringuks ja autentimise haldamiseks, mis hoolimata oma funktsionaalsusest sisaldasid korduvat koodi, ebaühtlast veakäsitlust ning lokaliseerimata kasutajateavitusi.

Refaktoreerimisel konsolideeriti korduv loogika, standardiseeriti veakäsitlus ning loodi ühtne tõlke- ja teavituskiht. Tulemusena vähenes teenuste koodimaht ligikaudu 40%, päringute teostamine lihtsustus märkimisväärselt ning veateated jõuavad kasutajani ühtses stiilis ja kasutaja valitud keeles.

Täiendavalt korrastati vaateid loogilise ülesehituse järgi ning seadistati keskkonnaspetsiifilised konfiguratsioonid, mis võimaldavad sama koodi kasutada nii arendus- kui ka tootmiskeskkonnas. Optimeerimise tulemusena on uute funktsionaalsuste lisamine oluliselt lihtsustunud, kuna suur osa taristuloogikast on nüüd kapseldatud taaskasutatavate komponentidena.

Lisaks loodi spetsiaalne teavituskiht, mis kasutab TalTechi stiiliraamatu `toast()` komponenti ja kapseldab selle ühtse abifunktsioonina. Funktsioon toetab kolme tüüpi visuaalseid teavitusi – `success`, `error` ja `warning` – ning võimaldab määrata kuvamisaja ja teemavärvid (nt tume režiim).

Veateavitused pärinevad üldjuhul teenusekihis `useRequests` toimuvast veahaldusest. Seal

käsitletakse erinevaid HTTP staatusekoode ja teisendatakse need lokaliseeritud sõnumiteks, mida kasutajale näidatakse. Kasutusloogika kokkuvõte on esitatud tabelis 1.

Staatuskood	**Tähendus**	**Kuvatud teavitus**
401	Autentimata kasutaja	Suunatakse sisselogimisele; `error` toast
403	Puuduvad õigused	“Ligipääs keelatud”; `error` toast
404	Ressurssi ei leitud	“Ei leitud”; `error` toast
422	Äriloogikaviga	Lokaliseeritud selgitus; `warning` toast
5xx	Serveri viga	“Serveri viga”; `error` toast

Tabel 1. Veateavituste loogika ja tüübid.

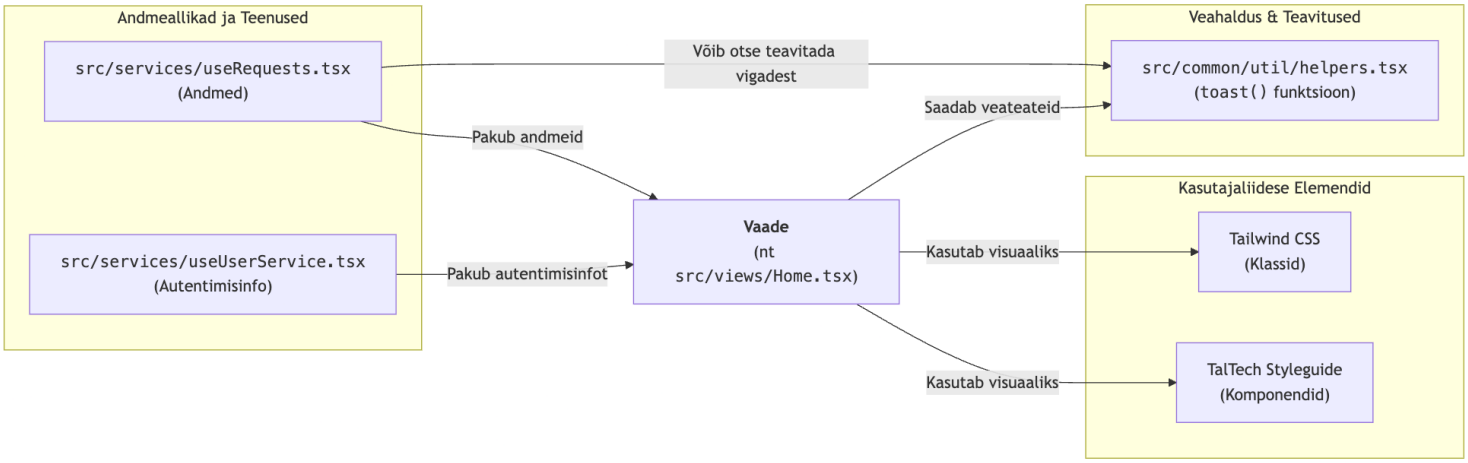
Spetsiifiliste äriloogika vigade puhul kasutatakse abifunktsiooni `translateGeneralBusinessLogicError()`, mis tõlgib tagarakendusest saadetud koodipõhised vead inimesele arusaadavaks sõnumiks. Funktsioon töötab koostöös lokaliseerimiskihiga, võimaldades sama koodi puhul kuvada korrektset sõnumit nii eesti kui inglise keeles. See lähenemine tagab kasutajale parema arusaamise probleemist ja toetab rahvusvahelist kasutust.

4.4.3 Vaadete implementeerimine

Pärast teenuste ja keskkonna ühtlustamist alustati vaadete loomist. Vaated implementeeriti järjekorras, mis võimaldas esmalt katsetada uut mustrit väiksema keerukusega ning seejärel taaskasutada lahendusi suuremate ja keerukamate vaadete puhul. Kujunduses lähtuti TalTechi stiiliraamatust, et säilitada visuaalne ühtsus kogu rakenduses. Lisaks pöörati tähelepanu rahvusvahelistamisele, et kasutajaliides toetaks mitmekeelset kasutust. Selleks kasutati `useTranslation` haaki ja lokaliseeritud sõnastikke, mis võimaldavad tõlkida kõik vaadetes kasutatavad tekstilised ressursid vastavalt kasutaja keelevalikule (kas inglise või eesti keel).

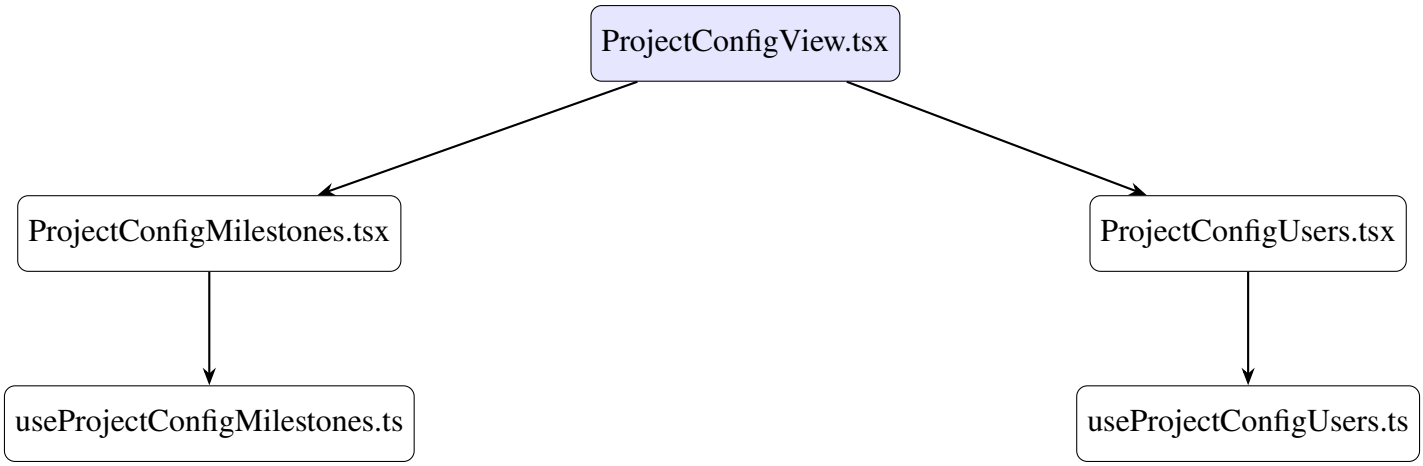
Kõik vaated järgivad sarnast arhitektuurset mustrit: andmed hangitakse päringuhalduri `useRequests` kaudu, kasutajainfo ja õigused jõuavad rakenduseni autentimisteenuse `useUserService` vahendusel, visuaal põhineb TalTechi komponentidel ja Tailwind [24] klassidel ning veateated jõuavad kasutajani standardiseeritud teavituste kaudu.

Selline struktuur võimaldab selgelt eristada vaate loogikat, visuaalkomponente ning äriloogikat käsitlevaid funktsioone, mis tagab kiire arenduse, koodi korduvkasutatavuse ning ühtse kasutuskogemuse. Vaadete ülesehituse loogika on näha joonisel 5.



Joonis 5. React-vaate arhitektuurne ülesehitus: andmed, autentimisinfo, veateavitused ja visuaalkomponendid.

Lisaks üldisele arhitektuurilisele mustrile on igal vaatel ka sisemine struktuur, mis jaguneb komponentideks ja nende poolt kasutatavateks haakideks (ingl k. *hooks*). Näiteks `ProjectConfigView.tsx` ehk projektisätete vaade kasutab kahte alamkomponenti, millest kumbki rakendab oma loogikahaaki. Nendevaheline sõltuvussuhe on kujutatud joonisel 6.



Joonis 6. ProjectConfigView vaate komponentide ja haakide vahelised sõltuvused.

Sellise arhitektuurse struktuuri alusel realiseeriti rakenduses mitmed vaated, millest igaüks täidab spetsiifilist funktsiooni ja tugineb samadele tehnilistele põhimõtetele. Järgnevalt on toodud ülevaade olulisematest vaadetest ning nende rakendusloogikast.

Registreerimisvaade

Kasutajakonto loomise funktsionaalsus oli esimene vaade, mis valmis, kuna see pani proovile autentimisvood ja tõlgitavate veateadete süsteemi. Varasem Vue-põhine vorm säilitas vaid väljade loogika; kogu valideerimine ja veateadete käsitlus on viidud teenuse tasandile ning sõnumid esitatakse kasutajale tõlgitaval kujul, vältides jäigalt kodeeritud ingliskeelset teksti.

Vaade põhineb Reacti funktsionaalsel komponendil ja kasutab haake `useState` vormiväljade ja veateadete haldamiseks ning `useNavigate`, et suunata kasutaja pärast registreerimist sisselogimislehele. Vorm kasutab TalTech stiiliraamatu komponente nagu `Form`, `Input`, `TTNewButton`, `ButtonGroup`. Registreerimistaotlused tehakse `useUserService.register` haagi abil.

Gruppide ning grupisätete vaated

Kasutajate grupid viidi uude tabelipõhisesse vaatesse, kus andmete kuvamine ja interaktsioon põhinevad `GroupList` komponendil. Vaade kasutab komponente nagu `TTNewCard`, `Input`, `Typography`, `Tooltip` . Kasutajaliidese funktsionaalsus hõlmab kliendipoolset sorteerimist, tekstipõhist otsingut ning alakomponente uute gruppide loomiseks (`GroupAdd`) ja kutsete haldamiseks (`GroupInvites`).

Grupisätete haldamiseks kasutatav `GroupConfigView` võimaldab muuta grupi üldandmeid, kasutajarolle ning projektide seoseid. Vaade on jagatud vahelehtedeks, mille nähtavus sõltub kasutaja rollist ja mille vahel liigutakse lokaalselt React-põhiste haakide `useState` ja `useMemo` abil. Rollipõhine juurdepääs tagatakse kohandatud `useRoles` haagi kaudu.

Projektide vahelehe kaudu on kasutajatel võimalik hallata grupiga seotud GitLabi projekte. Seal saab lisada projektide seoseid manuaalselt ning vajadusel algatada projektide sünkroniseerimine GitLabist, et andmed oleksid ajakohased. See funktsionaalsus võimaldab käsitsi sekkumist juhtudel, kui automaatne sünkroniseerimine on puudulik või vajatakse täiendavaid eriprojekte.

Ümberjärjestamise funktsionaalsus — näiteks projektide või kasutajate visuaalne lohistamine — on realiseeritud kohandatud komponentidega `DraggableBoard` ja `DraggableItem`, mis kasutavad `react-beautiful-dnd` teeki. Need komponendid toetavad nii hiirega kui ka puuteplaadiga kasutust ning tagavad kasutajasõbraliku ja dünaamilise konfiguratsiooni.

Projektide vaade

Projektide vaade kuvab kõik aktiivsed projektid konkreetse grupi sees. Vaade on üles ehitatud `ProjectsView.tsx` komponendina, mis kasutab `ProjectList` alamkomponenti andmete kuvamiseks. Visuaalse loogika aluseks on komponendid (`TTNewCard`, `Typography`, `Tooltip`) ning otsinguväli projektide filtreerimiseks.

Projektisisese info sünkroniseerimiseks GitLabi andmetega kasutab vaade spetsiaalset `useProjectRefresh` haaki, mis haldab API-päringute protsessi ja progressi. Värskendamise ajal kuvatakse kasutajale edenemisriba (`ProgressBar`), mis annab visuaalse indikatsiooni sünkroniseerimise edenemisest.

Projektiloendi filtreerimine toimub lokaalselt brauseri tasandil, kasutades viitega viivitust (*debounce*) — see vähendab tarbetuid API-päringuid ja tagab sujuvama kasutuskogemuse. Iga projekt on klikitav ja viib kasutaja edasi vastava projekti detailvaatesse.

Vaade tugineb URL-i parameetrile `groupId`, mis määrab, millise grupi projekte kuvatakse. Navigatsioon, autentimine ja veateavitused toimivad vastavalt üldisele arhitektuursele mustrile.

Projekti ning projektisätete vaated

Projekti detailvaade (`ProjectDetailView.tsx`) võimaldab õppejõududel ja administraatoritel analüüsida konkreetse projekti statistikat ja edenemist. Vaade tugineb URL-i parameetritele (`groupId`, `projectId`) ning laadib andmed vastava projektiga seotud tegevuste kohta, kasutades spetsiaalset loogikahaaki `useProjectDetails`. Andmete hulka kuuluvad näiteks meeskonnaliikmete panus, ajakulu piletite lõikes, osalus piletitel ja kogutud punktid.

Vaade koosneb mitmest alamkomponendist, sealhulgas:

- `ProjectOverallStats` – kuvab projekti üldised statistilised näitajad;
- `ProjectTeamMembers` – tabel ja diagramm arendajate panusest (kasutades Recharts teeki);
- `ProjectMilestonesSection` – kuvab tudengite seisu sprindipõhiselt ning võimaldab eksportida punkte;
- `ProjectIssueLog` – kuvab ajakulu logisid;
- `ProjectAnalyticsCharts` – kuvab ajapõhiseid graafikuid (päevapõhine ja kumulatiivne). Näitab andmeid koos ajakulu logidega vastavad valitud ajavahemikule.

Projektisätete vaade (`ProjectConfigView.tsx`) võimaldab seadistada projektiga seotud rolle, meeskonnaliikmeid ja sprinte. Vaade jaguneb vahekaartideks (`ProjectConfigUsers`, `ProjectConfigMilestones`), mille nähtavus sõltub kasutaja rollist. Mõlemad komponendid kasutavad visuaalset lohistusliidest (*drag-and-drop*), mis on realiseeritud `react-beautiful-dnd` teegi ja kohandatud komponentide abil. Sprindid on vaja lohistada vastavalt hindamispuu poolt genereeritud lahtritesse, et süsteem teaks, mis projektipõhine sprint kuulub millise genereeritud malli sprindi alla.

Hindamispuu

Hindamispuu vaate (`AssessmentView.tsx`) eesmärk on võimaldada õppejõududel ja juhendajatel määratleda hindamisstruktuur konkreetse grupi raames. Vaate keskmes on interaktiivne puukujuline skeem, kus iga sõlm esindab hindamiskriteeriumit, kategooriat või sprinti.

Vaade kasutab mitmeid haake ja alakomponente, millest olulisemad on:

- `useAssessmentTree` – laadib ja haldab hindamisskeemi struktuuri, võimaldab

skeemi uuendamist;

- `useNodeOperations` – pakub sõlmede lisamise, muutmise, kustutamise ja kloonimise funktsionaalsust;
- `AssessmentTreeChart` – põhikomponent, mis visualiseerib puu kujulise skeemi D3 teegi abil;
- `NodeModals` ja `NodeForm` – hüpikaknad sõlmede muutmiseks ja lisamiseks;
- `NodeActionsPanel` – kuvatakse valitud sõlme kohta täpsem info ja tegevusnupud.

Struktuur võimaldab lisada sprinte koos ajavahemikega (algus- ja lõppkuupäev), määrata punktipiiranguid ning grupeerida hindamiselemente kategooriate kaupa. Kõik tegevused on kasutajaliideses hallatud akendega, mis loovad loogilise ja sujuva kasutuskogemuse.

Lisaks sisaldab vaade komponenti `TemplateGenerator`, mis võimaldab luua hindamismalle valitud kuupäeva seisuga. See toetab tagasisidestamise protsessi, võimaldades õppejõududel hinnata tudengeid süsteemselt ja dokumenteeritult.

Muuseas on vaate ülesehitamiseks kasutatud komponente nagu `TTNewCard`, `Typography`, `Icon` ning kasutab ka `react-datepicker` moodulit kuupäevade sisestamiseks.

Hindamisvaate tehniliseks keerukuseks oli puu rekursiivne renderdamine, eriti suurte hindamisstruktuuride puhul. Selleks rakendati lazy-load lähenemist: sõlmede alamharud laetakse ja renderdatakse ainult siis, kui kasutaja need avab. See tagab vaate jõudluse ja stabiilsuse ka keerukate skeemide puhul.

Punktide sisestamise vaade

Kui hindamispuu vaade võimaldab määratleda hindamisstruktuuri ja kriteeriumid, siis punktide sisestamise vaade (`ProjectMilestoneGradingView.tsx`) võimaldab õppejõududel või juhendajatel anda tudengitele punkte konkreetse sprindi lõikes. Vaade töötab koostoimes hindamispuu struktuuriga ja kasutab samade kategooriate alusel punktide sisestamiseks dünaamiliselt genereeritud vorme.

Kasutaja saab iga tudengi kohta määrata punktid erinevates kriteeriumites. Punktide sisestamine toimub spetsiaalse komponendi `StudentAssessmentForm` kaudu, mis toetab nii klaviatuuri kui ka hiire- ja kerimisinteraktsioone (nt hiirerattaga väärtuse muutmine). Iga sisestusvälja puhul jälgitakse, et väärtus ei ületaks kriteeriumis määratud maksimumi – seda valideeritakse kohapeal ning tagarakenduse kaudu.

Hindamistulemustest annab visuaalse ülevaate `MilestoneRadarChart` komponent (radardiagramm), mis kasutab `Recharts` teeki. Diagramm kuvab meeskonna punktide jaotuse erinevate hindamiskriteeriumite lõikes ning uueneb reaalajas koos sisestatud andmetega. Iga telg radardiagrammil tähistab üht hindamiskategooriat ning telje pikkus vastab saavutatud punktisummale.

Diagrammi väärtused on vastavalt andmetele normaliseeritud, et tagada õiglane võrreldavus erinevate kategooriate vahel – arvesse võetakse nii igas kategoorias määratud maksimaalseid punkte kui ka tiimis olevate tudengite arvu. See võimaldab esitada proportsionaalse ja skaleeritud ülevaate meeskonna üldisest sooritusest, sõltumata tiimi suurusest või hindamiskriteeriumite mahulisest varieeruvusest.

Põhilised kasutatud haagid ja komponendid:

- `useMilestoneGradingData` – laadib hinnatavad andmed, tagastab tudengite hindamisinfo, võimaldab punktide salvestamist ja lõpphinnangu esitlust;
- `StudentAssessmentForm` – sisendvorm, mis toetab dünaamilisi sisestusvälju iga hindamiskriteeriumi kohta;
- `MilestoneRadarChart` – meeskonnapõhine visualiseering hindamistulemustest.

Vaade kasutab `TTNewCard`, `Loader`, `Alert`, `Typography`, `Icon`, `TTNewButton` komponente stiiliraamatust ning võimaldab kasutajatel hinnata kogu meeskonda sujuvalt ja läbipaistvalt.

Olulisemad tehnilised väljakutsed vaadete implementeerimisel:

- TalTechi stiiliraamatu komponentide kasutamine, mille dokumentatsioon osutus kohati ebajärjekindlaks.
- Recharts teegi integreerimine ja kujunduse kohandamine vastavalt TalTechi visuaalile, sealhulgas tumeda/heleda teema tugi.
- Hindamispuu rekursiivne renderdamine.
- Kohandatud kujunduse saavutamine vastavalt soovitud disainile, mis nõudis mitmete komponentide täiendamist ja ühilduvuse tagamist.

4.5 Tudengivaate loomine

Cognate'i varasem versioon oli suunatud eelkõige abiõppejõududele, võimaldades neil hinnata meeskonnaprojektides osalevate tudengite tööpanust, tööjaotust ja arengut. Tudengitele endile aga puudus selge ülevaade nende tööpanusest ning see info jõudis nendeni alles arendussprintide lõpus õppejõult saadud punktide ja kommentaaride kaudu.

Projekti klient tõi korduvalt esile vajaduse parandada tudengite kasutajakogemust, suurendada läbipaistvust ja toetada eneseanalüüsi. Lõppkasutajatelt kogutud tagasiside näitas, et tudengitel on raske hinnata, kui aktiivselt nad projekti panustavad võrreldes oma meeskonnakaaslastega või millised tegevused loovad enim väärtust. Samuti toodi välja, et jooksva tagasiside puudumine võib mõjutada motivatsiooni ja takistada arengukohtade varast märkamist.

Sellest lähtuvalt otsustati arendustöö käigus luua Cognate'i uus tudengivaade, mille eesmärk on pakkuda üliõpilastele reaalajas ülevaadet oma tööpanusest – sarnasel kujul nagu see on kättesaadav abiõppejõududele. Uus vaade võimaldab tudengitel jälgida näiteks:

- logitud töötunde ja nende jaotust ajas;
- koodiridade muutust sprintide kaupa;
- sprintide kaupa visuaalset tagasisidet töömahust ja aktiivsusest.

Tudengivaate lisamise eesmärk ei ole mitte ainult anda paremat ülevaadet, vaid ka motiveerida ja toetada iseseisvat õppimist, enesehindamist ning meeskonnasisese töö tasakaalustamist.

4.5.1 Tudengivaate planeerimine

Esialgu kavandati tudengitele täiesti uut vaadet, mis keskenduks nende individuaalsele panusele ja pakuks selget tagasisidet. Planeerimise käigus ilmnes aga, et süsteemis eksisteeris juba osaline funktsionaalsus, mis sarnanes soovitud tudengivaatega, kuid sellel oli mitmeid puudujääke, mis ei vastanud kliendi nõuetele:

- Ligipääs: Tudengid ei saanud vaatele iseseisvalt ligi. Selleks oleks pidanud abiõppejõud igale tudengile manuaalselt vastavates projektides spetsiifilised rollid määrama.
- Andmete ulatus: Olemasolev lahendus tagastas päringutes ka teiste meeskonnaliikme-

te individuaalsed hinded ja abiõppejõudude kommentaarid, mis ei tohiks tudengile nähtavad olla.

- Funktsionaalsus: Tudengitele oli ekslikult võimaldatud hinnete eksportimise funktsioon.

Nende leidude põhjal otsustati olemasolev lahendus ümber ehitada, mitte luua täiesti uut vaadet nullist. Kliendiga kooskõlastatud kohandatud eesmärgid olid järgmised:

- Automaatne ligipääs: Tudengid peavad saama oma projektide vaatele ligi automaatselt, ilma abiõppejõu sekkumiseta, tuginedes nende olemasolevale projektiliikmelisusele GitLabis.
- Privaatsus ja andmete piiramine: Tagarakendus tohib tudengile tagastada ainult tema isikliku statistika ja projekti üldised koondandmed. Teiste meeskonnaliikmete individuaalsed punktid, hinnangud või kommentaarid peavad olema välistatud.
- Kohandatud funktsionaalsus: Eemaldada tudengite jaoks ebavajalikud funktsioonid, nagu hinnete eksportimine.

See lähenemine võimaldas ära kasutada olemasolevat koodibaasi, keskendudes selle täiustamisele ja turvalisemaks muutmisele vastavalt tudengi vajadustele.

4.6 Tudengivaate tagarakendus

Olemasoleva tudengitele suunatud funktsionaalsuse kohandamine nõudis mitmeid muudatusi Cognate'i tagarakenduses, et tagada andmete korrektne piiramine ja automaatne ligipääs.

4.6.1 Automaatne rollide määramine ja projektipõhine ligipääs

Selleks, et tudengid saaksid automaatselt ligipääsu oma projektidele, täiendati süsteemi GitLabist andmete sünkroniseerimise loogikat. Funktsioonis `_sync_project_members_from_gitlab` leitakse projekti liikmed GitLabist. Iga liikme kohta, kes on GitLabis vähemalt arendaja (inlg k. *developer*) tasemel (GitLabi `access_level >= 30`), luuakse või uuendatakse Cognate'i süsteemis `UserProject` kirjed. Igale sellisele kasutajale määratakse automaatselt nii tudengi roll (`UserProject.Roles.STUDENT`) kui ka liikme roll (`UserProject.Roles.MEMBER`). See tagab, et tudengitel on koheselt ligipääs oma

projektide vaatele „Minu projektid“ all, ilma et abiõppejõud peaksid käsitsi õigusi määrama.

4.6.2 Andmete filtreerimine ja turvalisus API tasemel

Et tagada andmete korrektne selektiivsus ja privaatsus, loodi tudengivaate jaoks uued spetsiaalsed REST-põhised API otspunktid:

- `MyProjectDetailView`: Tagastab projekti üldandmed. Kasutab funktsiooni `find_project_metadata`.
- `MyProjectMilestonesStudentView`: Tagastab tudengispetsiifilised andmed sprintide kohta. Kasutab funktsiooni `find_my_project_assessment_milestones_for_student`. See funktsioon omakorda kutsub üldisemat funktsiooni `find_project_assessment_milestones`, kuid seejärel filtreerib tulemusi: teiste tudengite punktid (`user_points`) seatakse väärtusele `None` ja sprindi `assessed` staatus arvutatakse ümber, lähtudes sellest, kas sisseloginud tudengil on antud sprindis punkte.
- `MyProjectDevelopersStudentView`: Tagastab andmed sisseloginud tudengi panuse kohta (read, aeg, punktid). Kasutab funktsiooni `find_my_project_developer_data_for_student`. See funktsioon kutsub üldist `find_project_developers`, kuid muudab vastust nii, et ainult sisseloginud tudengi `total_points` on nähtavad; teiste tudengite punktid varjatakse (`None`).

Kõik need tudengispetsiifilised *use case* funktsioonid võtavad arvesse ka `group_id` parameetrit, et leida õige `ProjectGroupBridge` ID ja tagada, et kuvatavad andmed (eriti hinded) on kontekstipõhised selle grupi jaoks, mille kaudu tudeng projekti vaatab.

Need API otspunktid on disainitud tagastama ainult tudengile vajalikku ja lubatud informatsiooni:

- Rollipõhine filtreerimine: Dekoraator `@student_has_access_to_project` tagab, et ainult vastava rolliga tudengid pääsevad andmetele ligi.
- Andmete piiramine: Tagastatakse ainult päringu teinud tudengi isiklik panus (logitud tunnid, koodimuudatused, tema punktid) ja projekti üldine, anonüümne koondstatistika.
- Privaatsus: Teiste meeskonnaliikmete individuaalsed tulemused (nt täpsed punktid)

on välistatud või varjatud.

- Efektiivsus: Otspunktid on optimeeritud tagastama vaid vajalikke andmevälju, et vähendada andmemahtu ja parandada eesrakenduse jõudlust.

Selle uue API otspunktide komplekti kasutuselevõtt on keskne osa tudengivaate turvalisuse ja kasutatavuse tagamisel, kuna see võimaldab serveripoolselt kontrollida andmete voogu ja vältida tundliku info lekkimist eesrakendusse.

4.7 Tudengivaate eesrakendus

Tudengivaate kasutajaliidese kohandamisel keskenduti olemasoleva „Minu projektid“ vaate täiustamisele, et see vastaks tudengi vajadustele. Peamine eesmärk oli pakkuda visuaalselt lihtsasti mõistetavat ja sisuliselt kasulikku ülevaadet tudengi enda tööpanusest projektis. Säilitati varasemalt kavandatud visuaalsed elemendid, mis annavad tagasisidet tudengi panuse, ajakasutuse ja aktiivsuse kohta arendussprintide lõikes.

Peamised komponendid ja muudatused tudengivaates:

- Projekti üldstatistika: Komponent `ProjectOverallStats` kuvab projekti üldised näitajad. Tudengivaates on see identne õppejõu vaatega, kuna see ei sisalda personaalset infot.
- Meeskonnaliikmete panus: Komponent `ProjectTeamMembers` kuvab tabeli ja diagrammi arendajate panusest. Tudengivaate jaoks on API vastus kohandatud nii, et teiste tudengite täpsed punktid on varjatud (tagastatakse ‘null‘), kuid sisseloginud tudeng näeb oma punkte. Graafikul kuvatakse kõigi liikmete ajakulu.
- Sprintide ülevaade: Komponent `ProjectMilestonesSection` kuvab tudengite seisu sprindipõhiselt. Tudengivaates on eemaldatud hinnete eksportimise nupp ja võimalus otse hindeid muuta. API tagastab ainult sisseloginud tudengi punktid, teiste omad on varjatud.
- Ajakulu logid ja analüütika: `ProjectIssueLog` ja `ProjectAnalyticsCharts` komponendid kuvavad vastavalt ajakulu logisid ja ajapõhiseid graafikuid. Tudengivaates hangitakse andmed `fetchScopedTimeSpent` funktsiooniga, mis teeb päringu üldisele `/projects/<projectId>/timespent/` otspunktile, kuid filtreerib andmeid eesrakenduses, et kuvada ainult sisseloginud tudengiga seotud logisid või summeeri-

tud andmeid.

- Ligipääs: Tudeng näeb projekte, mille liige ta GitLabis on ja mis on Cognate'i sünkroniseeritud, ilma et peaks eraldi ligipääsu taotlema. Navigatsioon piirdub tema enda projektidega; grupivaadet ega teiste projektide detailset statistikat, mis ei ole seotud tema otsese osalusega, talle ei kuvata.

Eesrakendus, mis on loodud Reacti abil ja kasutab TalTechi disainisüsteemi komponente, integreerib tudengile suunatud funktsionaalsuse sujuvalt olemasolevasse „Minu projektid" sektsiooni. Tudeng näeb projekte, mille liige ta GitLabis on ja mis on Cognate'i sünkroniseeritud, ilma et peaks eraldi ligipääsu taotlema või projekti ID-sid sisestama.

4.8 Gitlab piletite analüüs

Tudengiprojektide kvaliteedi hindamisel on oluline analüüsida, kuidas meeskonnad GitLabi pileteid (ingl k. *issues*) kirjutavad. Piletite korralik loomine, ajakava järgimine ja sisuline selgus on olulised näitajad projektide juhtimise ning meeskonnatöö taseme hindamiseks.

Selleks loodi Cognate'i platvormile spetsiaalne analüüsifunktsionaalsus, mis aitab õppejõududel kiiremini hinnata piletite vastavust seatud reeglitele ning saada sisulist tagasisidet tehisintellekti abil. Lahendus koosneb kahest osast: tagarakendusest, mis teostab tegeliku analüüsi, kasutades nii andmebaasipõhiseid kontrolle kui ka suunatud LLM-põhist hindamist, ja eesrakendusest, mis kuvab analüüsi tulemusi kasutajale.

4.8.1 Tagarakendus

Funktsionaalsus toetab kahte peamist analüüsiliiki: reeglipõhine ja AI-põhine. Kogu loogika on struktureeritud vastavalt teenusepõhimõtetele ja asub `app.usecase.issue_analysis` moodulis.

Reeglipõhine analüüs

Reeglipõhist analüüsi teostab funktsioon `analyze_issues_for_milestone`, mille ülesanne on:

- laadida vastav projekt ja sprint ning kontrollida, et nad oleksid seotud;
- koguda kõik sprinti kuuluvad piletid;
- käivitada kontrollid igale piletile (`issue_db_checks.py`): kas on hinnang, kirjeldus,

vastutaja (ingl k. *assignee*), aega logitud jne;

- koostada raport: kontrollide läbimisprotsent, erinevad kriteeriumid, probleemsed piletid.

Kontrollifunktsioonid on lihtsad Django ORM päringud, mis tagastavad sõnastiku kujul tulemuse koos põhjendusega. Näiteks `check_issue_has_estimate` tagastab `{"passed": True/False, "message": "..."}`.

Tehisintellektil põhinev analüüs

AI-analüüsi teostab funktsioon `analyze_milestone_ai`, mis:

- kogub sprinti kuuluvad piletid;
- valmistab need ette LLM-i jaoks (kasutades XML-laadset märgendust `format_issue_for_xml`);
- koostab detailse prompti `ANALYSIS_PROMPT_TEMPLATE` alusel;
- saadab selle GPT mudelile (`issue_ai_checks.py`);
- parsib tagastatud JSON-vastuse ja valideerib selle Pydantic mudelitega (`MilestoneAnalysis`, `CriterionEvaluation`).

Kui vastuseid pole või ilmneb viga (nt puuduv API võti või LLM ei tagasta kehtivat JSONi), tagastatakse frontendi jaoks selge veateade.

Järgnevas tabelis 2 on esitatud kõik süsteemis kasutatavad piletite analüüsikriteeriumid koos analüüsitüübiga. Kriteeriumid on jagatud kahte gruppi – reeglipõhised kontrollid, mis põhinevad andmemudeli otsingu- ja valideerimisloogikal, ning AI-põhised kriteeriumid, mille hindamine toimub OpenAI [25] mudeli kaudu.

Kriteerium	Analüüsi tüüp
Pilet on seotud sprindiga	Reeglipõhine
Ajahinnang on määratud	Reeglipõhine
Tööaeg on logitud (kui suletud)	Reeglipõhine
Piletile on määratud vastutaja	Reeglipõhine
Piletil on olemas kirjeldus	Reeglipõhine
Pileti sulgeja erineb vastutajast	Reeglipõhine
Pealkiri on selge	AI-põhine
Kirjeldus on sisukas	AI-põhine
Tööülesande ulatus on määratletud	AI-põhine
Tegevused on konkreetsed	AI-põhine
Valmiduse kriteeriumid on esitatud	AI-põhine

Tabel 2. Piletite analüüsikriteeriumid ja nende tüübid.

API-vaated ja marsruudid

- `/projects/<id>/issue_analysis/` – kutsub välja reeglipõhise analüüsi;
- `/projects/<id>/issue_ai_analysis/` – kutsub välja AI-põhise analüüsi.

Mõlemad vaated valideerivad sisendid, teostavad vajaliku analüüsi ja tagastavad frontendi ootustele vastava struktuuriga JSON-vastuse.

Tehnilised võtmekohad ja arhitektuurilised valikud

Cognate'i piletianalüüsi funktsionaalsus on üles ehitatud modulaarselt, võimaldades selgelt eristada erinevaid kontrolliloogikaid ning hõlpsasti neid laiendada või asendada.

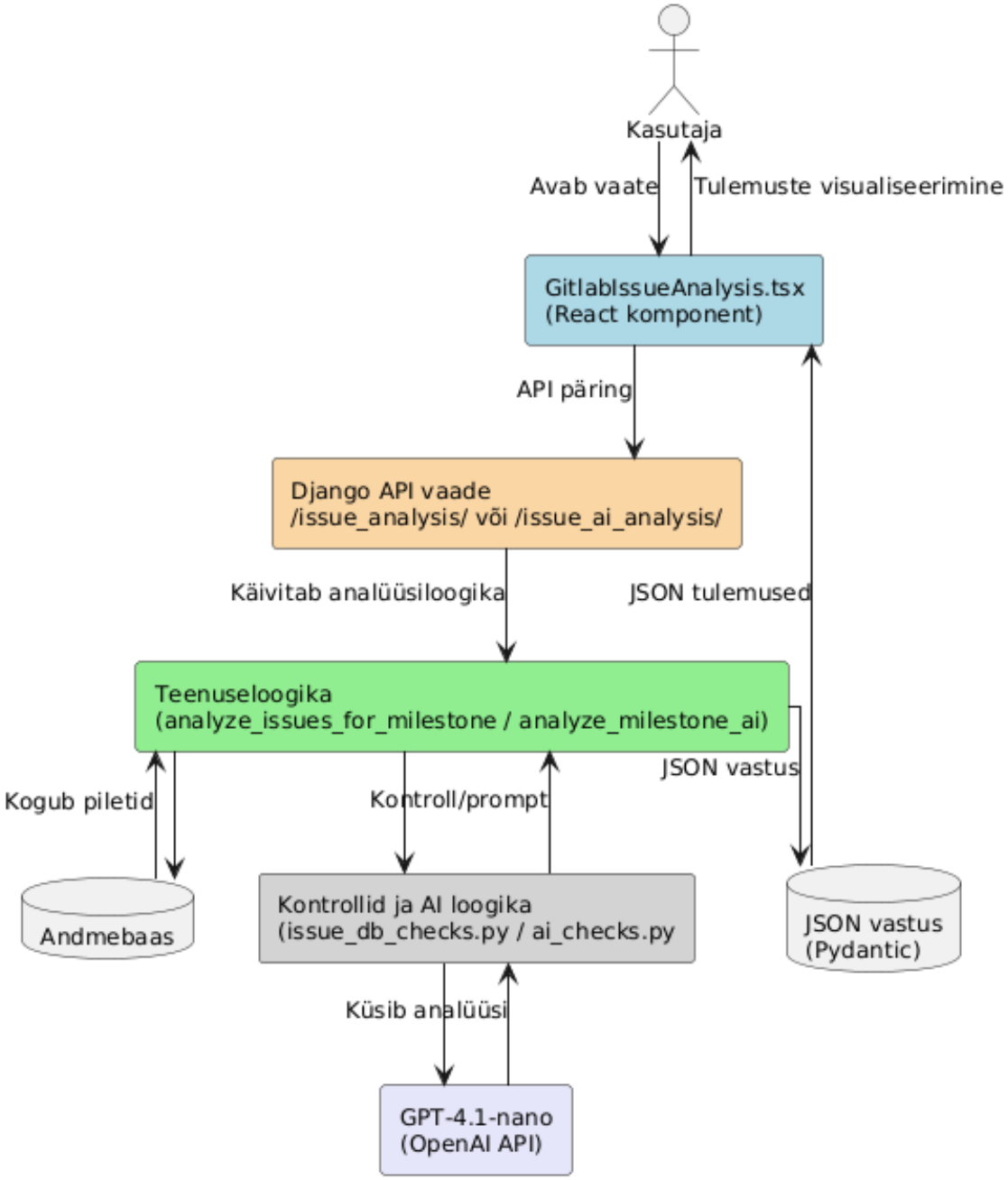
- **Andmete eraldatus:** Reeglipõhine ja AI-põhine analüüs on realiseeritud eraldi failides (`issue_db_checks.py` ja `issue_ai_checks.py`), mis tagab selguse ja isoleerituse.
- **Struktureeritud väljund:** AI analüüsi tulemused valideeritakse `Pydantic` mudelite abil. Mudelid `CriterionEvaluation` ja `MilestoneAnalysis` (`analysis_criteria_structured_output.py`) määravad väljundi täpse struktuuri ja aitavad vältida vigaseid vastuseid.
- **Mudelivalik:** Analüüsis kasutatakse `gpt-4.1-nano` mudelit, mis pakub sobiva tasakaalu kulu ja väljundi struktuurse kvaliteedi vahel. Kuna analüüs ei nõua keerukat

semantilist mõistmist, on väiksem mudel piisav.

- **Skaleeritavus:** Kulude ja jõudluse optimeerimiseks on võimalus AI analüüs ainult osale piletitest teha — valim moodustatakse juhuslikult ning selle suurust saab kohandada.
- **Kohandatavus:** Uute reeglipõhiste kontrollide lisamine toimub eraldi funktsioonidena `issue_db_checks.py` failis. Samuti on võimalik muuta AI analüüsi kriteeriume ning prompti ilma muudatusi tegemata teenuseloogikas.

Selline arhitektuur tagab, et süsteemi saab tulevikus laiendada, täiustada või ümber seadistada minimaalse koodimahu ja riskiga.

Et paremini illustreerida piletite analüüsi üldist töövoogu ning komponentidevahelist koostoimet, on joonisel 7 kujutatud kogu protsess: alates kasutajapoolse vaate aktiveerimisest kuni struktureeritud tulemuste visualiseerimiseni. Skeem rõhutab süsteemi kihilist ülesehitust ja selget vastutuse jaotust frontendi, API, teenuseloogika ja tehisintellekti vahel.



Joonis 7. Piletite analüüsi vaate töövoog: kasutaja, eesrakendus, API, loogika ja tehisintellekti koostoime.

4.8.2 Eesrakendus

Piletite analüüsi vaade realiseeriti vastavalt peatükis 4.4.3 kirjeldatud arhitektuurile. Vaade kasutab TalTechi disainisüsteemi komponente, päringute haldamiseks `useRequests`-põhist haaki ning järgib vaatekomponendi ja alamkomponentide mustrit, kus äriloogika on kapseldatud kohandatud haakidesse. Selline ülesehitus võimaldab moodulit isoleeritult testida ning kergesti laiendada, näiteks uute kontrollitüüpide või AI kriteeriumite lisamisel.

Komponent `GitlabIssueAnalysis.tsx` esindab modaalaknas avanevat analüüsivaadet, mida kasutatakse `ProjectMilestonesSection` kontekstis. Kogu kontrollide ja kriteeriumite sisu saadakse serverist struktureeritud JSON-kujul, ning eesrakendus tegeleb ainult andmete visualiseerimise ja tõlgendamisega kasutajale.

Vaade pakub kahte tüüpi analüüsi:

- **Reeglipõhine analüüs**, mis kuvab kontrollide tulemused iga pileti kohta, näiteks kas hinnang puudub, pilet pole seotud sprindiga või vastutaja on määramata;
- **Tehisintellektil põhinev analüüs**, mis kuvab AI poolt genereeritud struktureeritud tagasisidet, sh hinnanguid pealkirjade, kirjelduste ja ulatuse kohta.

Vaade põhineb Reacti funktsionaalsel komponendil ning kasutab Taltechi komponente nagu `Modal`, `TTNewCard`, `TTNewButton`, `Icon`, `Loader` ja `Typography`. Andmete laadimine ja olekute haldus toimub haagi `useGitlabIssueAnalysis` kaudu, mis korraldab kaks eraldi API-päringut: reeglipõhise ja AI-põhise analüüsi jaoks.

Kasutajaliidese funktsionaalsus sisaldab:

- kahte vahekaarti analüüside eristamiseks;
- laadimisolekute ja vigade käsitlemist;
- dünaamilist kuvamist vastavalt valitud vahekaardile;
- automaatset avamist, kui vanemkomponent suurendab `triggerAnalysis` väärtust.

Tehisintellekti analüüsi vaates kuvatakse piletite koguarv, analüüsitud piletite arv, kriteeriumipõhised hinnangud (nt pealkirja selgus, kirjelduse kvaliteet), tugevused, parendusvaldkonnad ja konkreetsed soovitused. Reeglipõhises vaates kuvatakse kontrollide läbivuse kokkuvõte, piletid, mis vajavad tähelepanu, ning detailid, miks kontroll ebaõnnestus.

Haak `useTranslation` tagab, et kõik kasutajaliidese tekstid (sealhulgas kontrollide nimed ja kirjeldused) oleksid lokaliseeritud, kuid kuluefektiivsuse nimel jäeti AI analüüs vaid inglisekeelseks. Visuaalselt eristuvad tulemused ikoonide ja värvide kaudu, võimaldades kasutajal kiirelt hinnata piletite kvaliteeti.

5. Analüüs

Selles peatükis keskendutakse tehtud arendustööde tulemuste ja nende mõju analüüsile. Vaadeldakse, kuidas Cognate'i platvormi täiustused on mõjutanud kasutusmugavust ja andmete täpsust ning kuidas uus tudengivaade on rakendust täiendanud. Lisaks tuuakse välja projekti kliendi, abiõppejõudude ja tudengite tagasiside, mis peegeldab lahenduse praktilist väärtust.

5.1 Tulemused

Käesoleva bakalaureusetöö peamisteks tulemusteks on GitLabi projektide metaandmete analüüsi platvormi Cognate'i oluliselt täiustatud kasutusmugavus ning uue tudengivaate loomine ja implementeerimine. Arendustööde käigus lahendati mitmeid varasemaid kitsaskohti süsteemi funktsionaalsuses, andmete täpsuses ning tehnilises arhitektuuris.

Olulisemad saavutatud tulemused on järgmised:

- **Grupihalduse täiustamine:** Parandati gruppide haldamise funktsionaalsust, lisades võimaluse gruppe eemaldada kasutajaliidese kaudu. Samuti tõsteti turvalisust, piirates tundliku info (nt GitLabi ligipääsuluba) nähtavust ning rakendades rollipõhist ligipääsu grupisätete muutmisele. Liikmete gruppi lisamise protsess muudeti intuitiivsemaks, võimaldades rolli määramist kohe lisamise hetkel.
- **Andmete õigsuse tagamine:** Lahendati kriitiline probleem seoses andmete dubleerimisega, mis varasemalt põhjustas eksitavat statistikat koodiridade ja ajakulu kohta. Rakendati unikaalsusnõuded GitLabi grupi- ja projektiidentifikaatoritele ning loodi andmemudel, mis korrektselt käsitleb olukordi, kus üks projekt kuulub mitmesse gruppi. Selle tulemusena on Cognate'i poolt kuvatavad andmed usaldusväärsemad, mis on hindamisprotsessi objektiivsuse seisukohalt võtmetähtsusega. Andmebaasi struktuuri muudatused ja duplikaatkirjete vältimine parandasid ka päringute jõudlust.
- **Eesrakenduse migreerimine React tehnoloogiale:** Viidi lõpule Cognate'i eesra-

kenduse üleviimine Vue platvormilt Reactile. See tagab parema ühilduvuse TalTechi disainisüsteemiga, lihtsustab edasist hooldust ja arendust ning võimaldab kasutada ülikooli ametlikke kasutajaliidese komponente. Migreerimise käigus refaktoreeriti olulisi teenuseid, standardiseeriti veakäsitlus ja loodi React-põhised ning täiustatud vaated (sealhulgas registreerimine, gruppide haldus, projekti detailvaated koos graafikutega, hindamine).

- **Tudengivaate täiustamine:** Varasemates versioonides eksisteeris küll tudengitele mõeldud vaade, kuid see ei vastanud kliendi nõuetele ega olnud praktikas kasutatav (õigusi ei saanud iseseisvalt taotleda, kuvatavad andmed olid piiratud ning ligipääs ulatus ka teiste tudengite hinnanguteni). Käesoleva töö käigus viidi see vaade üle TalTechi React-põhisele disainiraamistikule, parandati kuvavead ja lisati interaktiivsed diagrammid, mis näitavad tudengi logitud töötunde, koodimuudatusi ja teisi muudatusi sprintide lõikes. Tagarakendusse lisandus projektipõhine ligipääsuloogika ning spetsiaalne API otspunkt, mis tagastab tudengile ainult temaga seotud andmed — see suurendas privaatsust ja võimaldas tudengitel ligipääsu iseseisvalt aktiveerida ilma õppejõu sekkumiseta.
- **GitLabi piletite analüüsi funktsionaalsuse loomine:** Arendati välja uus moodul GitLabi piletite kvaliteedi automaatseks hindamiseks. Funktsionaalsus hõlmab nii reeglipõhiseid kontrolle (näiteks pileti metaandmete, nagu hinnangu või vastutaja olemasolu) kui ka tehisintellektil põhinevat sisuanalüüsi (hinnates pileti pealkirja ja kirjelduse selgust ning ulatust). See annab õppejõududele tööriista tudengite projektijuhtimise oskuste objektiivsemaks hindamiseks ning tudengitele endile vahetumat ja detailsemat tagasisidet oma töö kvaliteedi kohta.

Nende arenduste tulemusel on Cognate'i platvorm muutunud stabiilsemaks, kasutajasõbralikumaks ning pakub senisest laiemat väärtust nii õppejõududele kui ka tudengitele, toetades efektiivsemalt meeskonnaprojektide haldamist ja hindamist Tallinna Tehnikaülikoolis. Kliendi ja abiõppejõudude tagasiside kinnitab nende parenduste positiivset mõju rakenduse igapäevasele kasutusele.

5.2 Kliendi hinnang

Bakalaureusetöö raames viidi läbi arendustöid, mille eesmärk oli täiustada projektide analüüsimise rakendust, mida kasutatakse õppeainetes nagu “Erialatutvustus” ja “Tarkvaraarenduse projekt”. Lõputöö käigus parandati olemasoleva rakenduse kasutuskogemust ning täiendati seda funktsionaalsustega. Arendustööde käigus keskenduti muu hulgas aja kuvamise, punktide määramise ning kasutajate lisamise loogika parandamisele.

Kommunikatsioon arendustiimiga oli pidev ning töö käigus arutati kliendiga regulaarselt erinevaid funktsionaalsusi ning tiim võttis arvesse kliendipoolset tagasisidet. Hea näide sellest on olukord, kus rakenduse üks kriitilisemaid komponente, aja kuvamine, ei töötanud ootuspäraselt. Kuna aja kuvamine on abiõppejõudude jaoks üks olulisemaid näitajaid, mille põhjal tudengeid hinnatakse, oli see probleem kriitiline. Õnneks sai see mure lahendatud, mistõttu muutus rakendus taas ka praktiliselt kasutatavaks.

Parandused, mis rakendusele tehti, on toonud kaasa selge kasutusmugavuse kasvu. Lehe erinevad funktsionaalsused töötavad nüüd oluliselt kiiremini, sealhulgas: hindamise loogika ja kasutajaliides on muutunud intuitiivsemaks, punktide määramine on sujuvam ja selgem, projektide ja kasutajate lisamine süsteemi on oluliselt lihtsustatud. Samuti lisati visuaalseid vihjeid, mis annavad kasutajale paremat tagasisidet ning muudavad kasutajakogemuse märgatavalt intuitiivsemaks ja sujuvamaks.

Eesrakenduse migreerimine Vue3-lt Reactile ning TalTechi ametliku stiiliraamatu komponentide kasutuselevõtt oli rakenduse tulevikuks oluline samm. See tagab, et rakendus näeb välja ja käitub samamoodi nagu teised ülikooli infosüsteemid, vähendab visuaalsete lahkhelide riski ja muudab tulevased hooldus- ning laiendustööd märksa kuluefektiivsemaks: uusi kasutajaliidese komponente või disainiuuendusi saab versiooniuuendustena lihtsalt NPM-i kaudu sisse tuua ilma neid käsitsi ülekirjutamata.

Tudengivaade, mis töö käigus loodi, pakub nüüd tudengitele paremat ülevaadet nende projektide seisust ning võimaldab neil lihtsamini jälgida oma edusamme. Tudengivaade suurendab läbipaistvust ja toetab enesehindamist. Vaade võimaldab tudengitel reaalajas jälgida näiteks: logitud töötunde, punkte ja koodiridade muutust sprintide kaupa. Kokkuvõttes toetab tudengivaade iseseisvat õppimist ning aitab tasakaalustada meeskonnasisest tööjaotust.

Samavõrd olulise uuendusena tõi klient esile GitLabi piletite analüüsi mooduli lisandumise. See funktsionaalsus, mis kombineerib reeglipõhiseid kontrolle tehisintellekti toetatud sisuanalüüsiga, on kliendi hinnangul väärtuslik tööriist õppejõududele, et kiiremini ja ühtlasemalt hinnata tudengite loodud piletite kvaliteeti. Ühtlasi aitab see tudengitel paremini mõista projektijuhtimise heade tavade olemust ja saada konkreetset tagasisidet oma tööle, mis toetab õppeprotsessi.

Autorid näitasid head arusaamist rakenduse toimimisest ning arvestasid kliendi vajadustega. Lõputöö raames tehtud arendustööde tulemusel on Cogante muutunud märgatavalt kasutajasõbralikumaks ning toetab hindamisprotsesse tõhusamalt.

5.3 Abiõppejõudude tagasiside

Pärast 2025. aasta uuendusi koguti Lisas 2 välja toodud küsimustikuga tagasisidet kümnelt abiõppejõult, kes olid Cognate'i platvormi kasutanud nii enne kui ka pärast tehtud muudatusi. Küsitluse tulemused näitasid märkimisväärset paranemist kasutajakogemuses ja üldises rahulolus.

Enne uuendusi hindasid abiõppejõud oma üldist rahulolu platvormiga keskmiselt hindega 2,0 (skaalal 1–5, kus 1 on „väga rahulolematu“ ja 5 „väga rahul“). Enne uuendusi tõid kümnest vastanud abiõppejõust peamiste probleemidena esile (vastajate arv sulgudes):

- gruppidesse liikmete lisamise ebamugavus (8);
- kuvatava ajakulu ebatäpsus (8);
- sprintide sobitamise ebamugavus (7);
- punktide sisestamise kohmakus (6);
- projektidest tudengite puudumine (4);
- kuvatava muudetud koodiridade arvu ebatäpsus (4);

Eraldi toodi välja ka:

- laadimisanimatsioonide puudulikkus (2).
- hindamispuu koostamise probleemid (1);

Platvormi visuaalse poolega koges probleeme 8 abiõppejõudu (2 sageli, 6 mõnikord), samas kui 2 kogesid seda harva. Gruppide haldamise lihtsust hinnati enne uuendusi keskmiselt

hindega 2,2, projektide hindamise mugavust hindega 2,4 ning sprintide sobitamise mugavust hindega 2,2.

Pärast uuendusi hindasid abiõppejõud oma üldist rahulolu platvormiga keskmiselt hindega 4,5, mis näitab olulist positiivset nihet. Kümnest vastanust märkisid paranemist järgmistes valdkondades (vastajate arv sulgudes):

- gruppidesse liikmete lisamise ebamugavus (8);
- kuvatava ajakulu õigsus (8);
- sprintide sobitamise ebamugavus (8);
- kuvatava muudetud koodiridade arvu ebatäpsus (6);
- hinnete panemise ebamugavus (6);
- projektidest tudengite puudumine (4);

Eraldi toodi välja ka:

- kasutajakogemuse üldine paranemine (1);
- parem tagasiside lehelt toimingute kohta (nt laadimisindikaatorid) (1).

Kõik 10 vastanut leidsid, et platvormi visuaalne pool on uuendatud versioonis oluliselt (6 vastanut) või mõnevõrra (4 vastanut) paremini kooskõlas ülikooli teiste süsteemidega. Gruppide haldamise lihtsust hinnati nüüd keskmiselt hindega 4,2, projektide hindamise mugavus sai keskmiseks hindeks 4,6 ning eriti positiivselt hinnati sprintide sobitamise mugavust, mille keskmine hinne oli 4,7.

Uuendused on mõjutanud positiivselt ka abiõppejõudude tööefektiivsust. Enne uuendusi kulus 8 abiõppejõul kümnest sprindi hindamiseks ligikaudu 30 minutit ja 2 abiõppejõul 45 minutit (keskmine aeg 33 minutit). Pärast uuendusi kulub 2 abiõppejõul hindamiseks 15 minutit ja 8 abiõppejõul 30 minutit (keskmine aeg 27 minutit). See tähendab keskmist ajalist võitu 6 minutit sprindi kohta ehk keskmise hindamisaja vähenemist umbes 18%.

Uutest funktsionaalsustest pidasid abiõppejõud piletite ja projektihalduse analüüsimise võimalust väga kasulikuks, andes sellele keskmiseks hindeks 4,4. Tudengivaate kasulikkust üliõpilaste eneseanalüüsi ja motivatsiooni toetamisel hinnati keskmiselt hindega 4,3.

5.4 Tudengite tagasiside

Uurimaks tudengite arvamust Cognate'i uue tudengivaate ja analüüsifunktsioonide kohta, koguti Lisas 4 välja toodud küsimustikuga tagasisidet neljateistkümnelt tudengilt.

Kõik 14 küsitletud tudengit leidsid, et nad oleksid soovinud Cognate'i tudengivaadet kasutada varasemate ainete nagu „Erialatutvustus" ja „Tarkvaraarenduse projekt" raames. Loodud tudengivaate kasulikkust oma töö jälgimisel ja analüüsimisel hindasid tudengid keskmiselt hindega 4,3 (skaalal 1–5, kus 1 on „mitte väga kasulik" ja 5 „väga kasulik"). Tudengivaate mõistmise ja kasutamise lihtsust hindasid tudengid keskmiselt hindega 4,1 (skaalal 1–5, kus 1 on „väga keeruline" ja 5 „väga lihtne").

Pileti analüüsi funktsionaalsuse kohta antud avatud vastustes tõid tudengid korduvalt esile selle kasulikkust vigade ja ununenud tegevuste (nt ajakulu prognoosi, pileti sidumine) märkamisel enne hindamist: „ei pea abiõppejõudu tüütama vaid näeb kohe ise ära, kui kuskil on mingi asi määramata," „piletite tegemisel ununevad vahepeal tähtsad tegevused ära." Mitmed leidsid, et see aitab hoida aega kokku ja mainib tähtsamaid aspekte, mida meeles pidada. Üldiselt peeti seda heaks ja huvitavaks lisandiks.

Tehisintellektil põhineva analüüsi kohta arvasid tudengid, et see on samuti abistav ja „näitab detailsemalt, mida muutma peab." Mitmed tõid välja, et see on mugav viis saada „second opinion" ja analüüsida kõiki pileteid korraga. Eriti meeldisid tudengitele tehisintellekti analüüsi osad „Tugevused", „Parandused" ja „Soovitused", mis annaksid „lihtsat, kuid vajalikku infot". Kokkuvõtvalt leiti, et funktsionaalsus annab tudengivaatele lisandväärtust.

6. Edasiarenduste võimalused

Kuigi rakendus täidab edukalt oma praegust eesmärki tudengiprojektide haldamisel ja hindamisel, on mitmeid potentsiaalseid suundi, kuidas süsteemi tulevikus edasi arendada ja kasutusvõimalusi laiendada.

Üheks oluliseks arenguvõimaluseks oleks võimaldada tudengil liidestada Cognate'iga ka oma teisi GitLabi hoidlaid. Hetkel keskendub süsteem peamiselt konkreetsete projektigruppide raames toimuvatele arendustele, mis põhinevad agiilsel sprintide loogikal. Kui rakenduse projektipõhine struktuur teha paindlikumaks ja vähem sprintide-keskseks, avaneks võimalus pakkuda ülevaadet ka väiksemamahulistest või individuaalsetest töödest, mis ei järgi klassikalist iteratiivset arendusmudelit. See muudaks rakenduse kasulikuks ka nende kursuste kontekstis, kus ei kasutata meeskonnatööd ega agiilseid praktikaid.

Teiseks võiks Cognate'i laiendada ka lõputööde haldamise ja analüüsimise suunas. Lõputöö protsess hõlmab sageli iseseisvat, kuid ajaliselt hajutatud arendust, mille puhul oleks väärtuslik näha töö edenemist versioonihalduse tasandil. Selline funktsionaalsus võimaldaks õppejõududel või juhendajatel paremini jälgida tudengi tööprotsessi, hinnata panust ja anda struktureeritud tagasisidet.

7. Kokkuvõte

Käesolev bakalaureusetöö keskendus Tallinna Tehnikaülikooli tarkvaraprojektide haldusplatvormi Cognate'i edasiarendamisele, sihiga parandada selle kasutusmugavust ning luua uus, tudengitele suunatud funktsionaalsus. Töö ajendiks olid platvormi senises kasutuses ilmnenud kitsaskohad, sealhulgas ebatäpsused andmete kuvamisel, piiratud grupihalduse võimalused, eesrakenduse tehnoloogiline lahknevus ülikooli standarditest ning tudengitele suunatud otsese tagasisidemehhanismi puudumine.

Töö eesmärkide saavutamiseks viidi läbi mitmeid olulisi arendustöid. Esiteks tegeleti andmete õigsuse probleemiga, mille lahendamiseks struktureeriti andmemudelit ümber, et vältida duplikaatkirjete teket ja korrektselt hallata projekte, mis kuuluvad mitmesse gruppi. Teiseks parandati grupihalduse funktsionaalsust, lisades gruppide eemaldamise võimaluse, täiustades turvalisust seoses tundliku API pääsmega ning lihtsustades liikmete lisamist. Kolmandaks viidi lõpule eesrakenduse migratsioon Vue platvormilt Reactile, mis tagab parema ühilduvuse TalTechi disainisüsteemiga ning hõlbustab tulevast hooldust. See hõlmas olemasolevate teenuste refaktoreerimist ja mitmete kasutajaliidese vaadete uuendamist või loomist. Neljandaks loodi uus funktsionaalsus GitLabi piletite analüüsimiseks, mis pakub õppejõududele nii reeglipõhist kui ka tehisintellekti toetatud tagasisidet tudengite projektijuhtimise kvaliteedi kohta.

Töö peamisteks tehnilisteks väljakutseteks ja seega ka kõige keerukamateks osadeks osutusid mitmed aspektid. Esiteks, uute tehnoloogiate omandamine ja rakendamine, millega autoritel varasem kokkupuude puudus, nõudis märkimisväärset süvenemist ja kohanemist. Teiseks oli keskse tähtsusega olemasoleva tagarakenduse põhjalik ja süsteemne tundmaõppimine, et seda tõhusalt parandada ning edasi arendada. Kolmandaks kujunes oluliseks ja nõudlikuks ülesandeks eesrakenduse migreerimise protsessi hoolikas planeerimine ja tehniline implementeerimine.

Käesoleva bakalaureusetöö raames tehtud arenduste tulemusena on Cognate'i platvorm

muutunud oluliselt töökindlamaks, kasutajasõbralikumaks ja funktsionaalsemaks. Parandatud andmekvaliteet, täiustatud haldusvahendid ning Reactil põhinev kaasaegne eesrakendus loovad parema kasutuskogemuse õppejõududele. Uus tudengivaade laiendab platvormi kasutegurit otse üliõpilastele, edendades läbipaistvust ja iseseisvat õppimist. Tehtud arendused on saanud positiivset tagasisidet nii projekti kliendilt kui ka rakenduse kasutajatelt, kinnitades töö edukust ja praktilist väärtust õppetöö kontekstis.

Kasutatud kirjandus

[1] GitLab. URL: https://gitlab.cs.taltech.ee.

[2] Mart Kaasik ja Kristjan Kõiv. *GitLab Project Metadata Analysis Ecosystem*. Supervisor: Ago Luberg, PhD. 2022.

[3] Django Software Foundation. *Django Documentation*. [Accessed: 16-11-2024]. URL: https://docs.djangoproject.com/en/4.2/.

[4] Evan You et al. *Vue.js Documentation*. [Accessed: 16-11-2024]. URL: https://vuejs.org/guide/introduction.html.

[5] Bostock, Mike ja D3.js Contributors. *What is D3?* [Accessed: 12-05-2025]. 2025. URL: https://d3js.org/what-is-d3.

[6] Laura Anni, Marcus Pruks ja Henri Helisma. *Automating Cognate and Creating a New User Interface*. Supervisor: Ago Luberg, PhD. 2023.

[7] Kristjan Marcus Sulaoja ja Andreas Saarep. *GitLabi projektide metaandmete analüüsi ökosüsteemi Cognate'i kasutajamugavuse parandamine ja arendustöö lihtsustamine*. Supervisor: Ago Luberg, PhD. 2024.

[8] Tallinn University of Technology. *TalTech'i digitaalne stiiliraamat*. [Accessed: 30-01-2025]. 2025. URL: https://storybook.taltech.ee/?path=/docs/docs-intro--docs.

[9] Meta Open Source. *React Documentation*. [Accessed: 16-11-2024]. URL: https://react.dev/learn.

[10] Agile Alliance. *Manifesto for Agile Software Development*. [Accessed: 18-05-2025]. 2001. URL: https://agilemanifesto.org/.

[11] Discord. URL: https://discord.com.

[12] NPM. *npm Documentation*. [Accessed: 18-05-2025]. 2024. URL: https://docs.npmjs.com/.

[13] Red Hat. *What is CI/CD?* [Accessed: 05-05-2024]. URL: https://www.redhat.com/en/topics/devops/what-is-ci-cd.

[14] The PostgreSQL Global Development Group. *PostgreSQL 14 Documentation*. [Accessed: 18-05-2025]. 2021. URL: https://www.postgresql.org/docs/14/.

[15] GitLab. *GitLab'i API dokumentatsioon*. [Accessed: 16-11-2024]. 2024. URL: https://docs.gitlab.com/ee/api/rest/.

[16] GitLab. *Get started with GitLab CI/CD*. [Accessed: 05-05-2024]. URL: https://docs.gitlab.com/ee/ci/.

[17] Docker. *Docker Documentation*. [Accessed: 18-05-2025]. 2025. URL: https://docs.docker.com/.

[18] *Pylint User Guide - What is Pylint?* [Accessed: 18-05-2025]. URL: https://pylint.pycqa.org/en/latest/user_guide/index.html#what-is-pylint.

[19] OpenJS Foundation. *ESLint Documentation*. [Accessed: 19-05-2025]. 2025. URL: https://eslint.org/docs/latest/.

[20] Prospector Developers. *Prospector Documentation*. [Accessed: 19-05-2025]. 2025. URL: https://prospector.landscape.io/en/latest/.

[21] OpenStack Security Project. *Bandit Documentation*. [Accessed: 19-05-2025]. 2025. URL: https://bandit.readthedocs.io/en/latest/.

[22] Meta Open Source. *Jest Documentation*. [Accessed: 19-05-2025]. 2025. URL: https://jestjs.io/docs.

[23] Testing Library Authors. *React Testing Library Documentation*. [Accessed: 19-05-2025]. 2025. URL: https://testing-library.com/docs/react-testing-library/intro.

[24] Tailwind Labs. *Tailwind CSS Documentation*. [Accessed: 19-05-2025]. 2025. URL: https://tailwindcss.com/docs.

[25] OpenAI. *OpenAI API Documentation*. [Accessed: 23-05-2025]. 2025. URL: https://platform.openai.com/docs.

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

Meie, Kris-Sten Pallas ja Karl Agasild

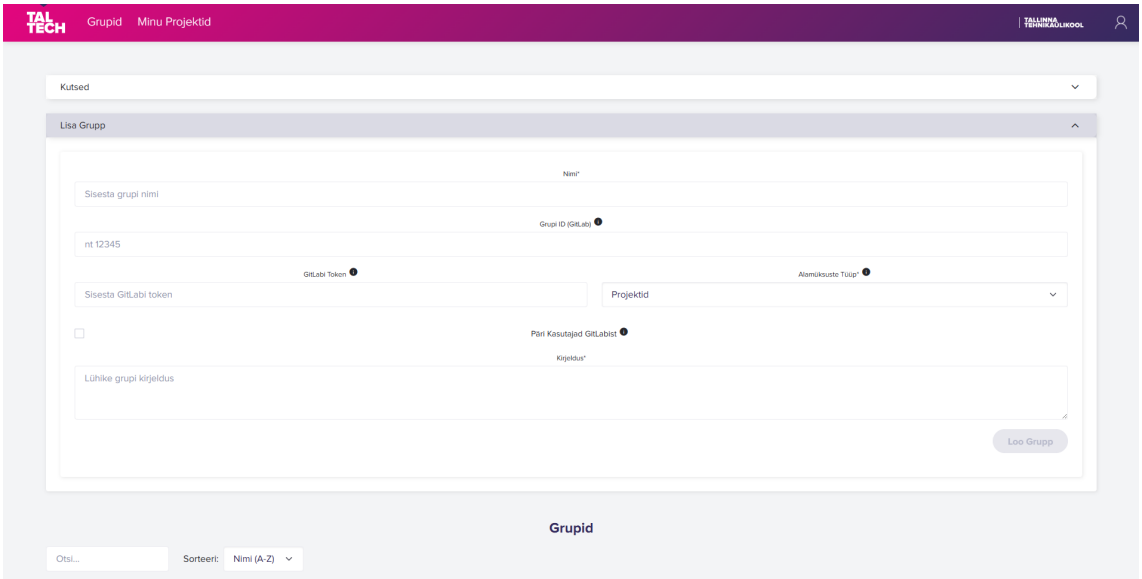
1. Anname Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "GitLabi projektide metaandmete analüüsi platvormi Cognate tudengivaate loomine ja kasutusmugavuse täiustamine", mille juhendaja on Ago Luberg
    1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, 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.

04.06.2025

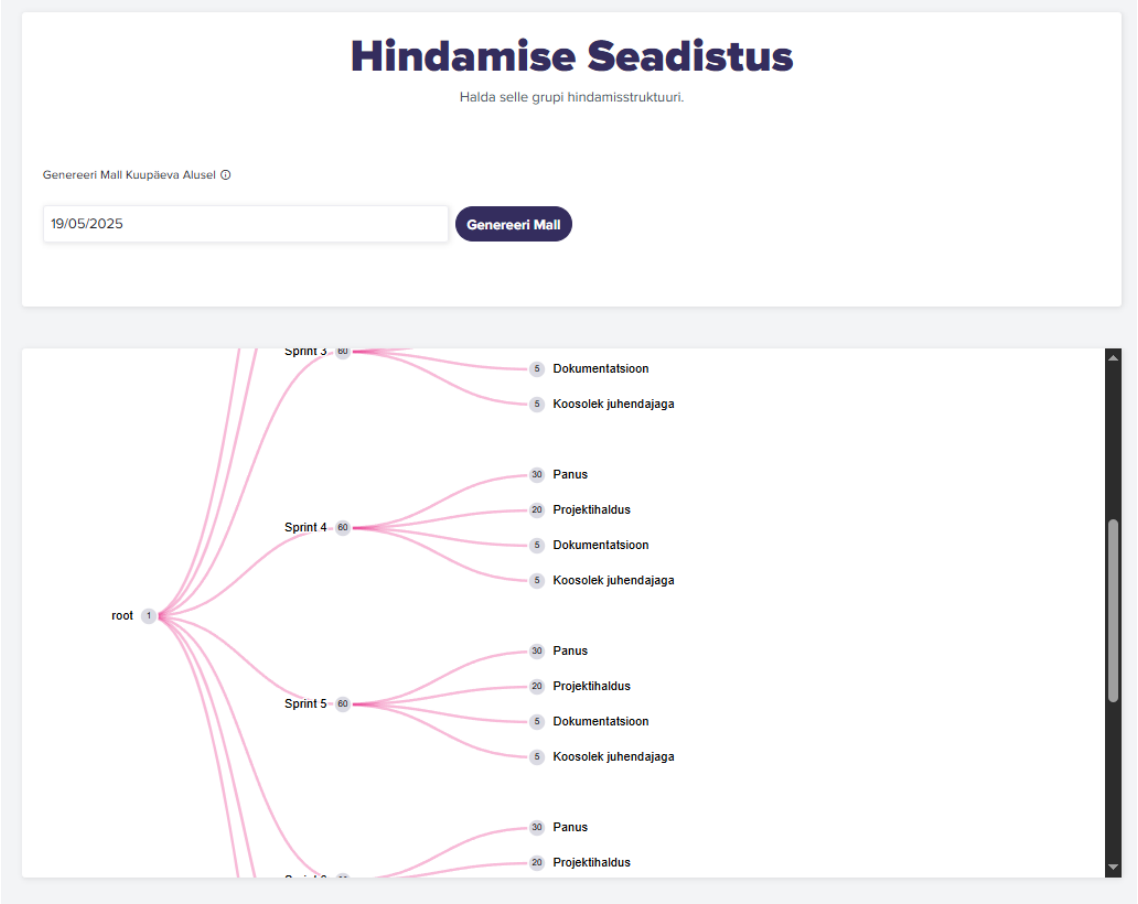
alumine

[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.

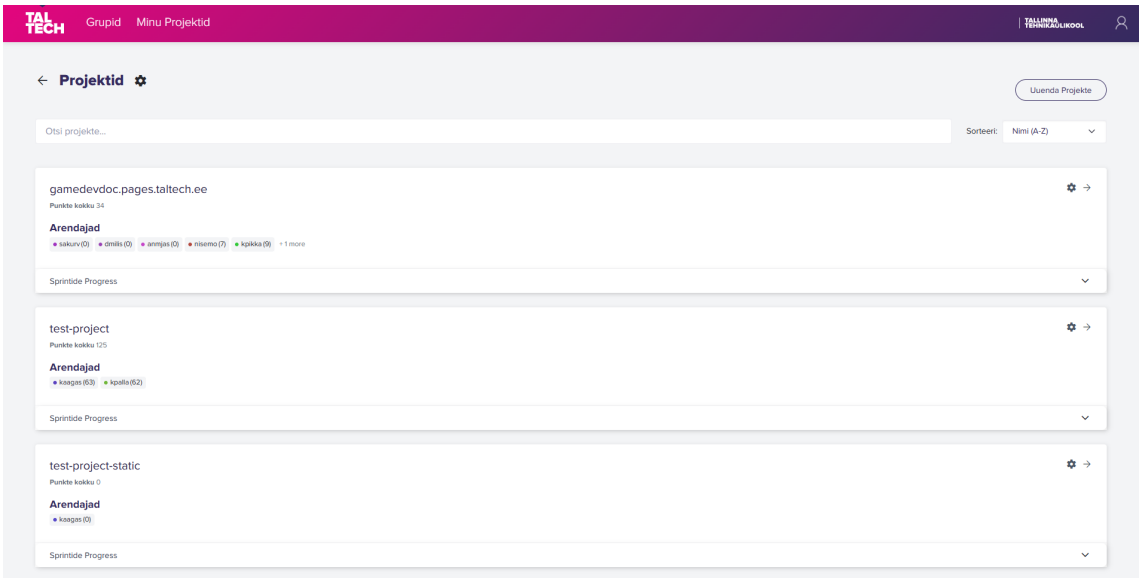
Lisa 2 – Cognate’i peamised vaated



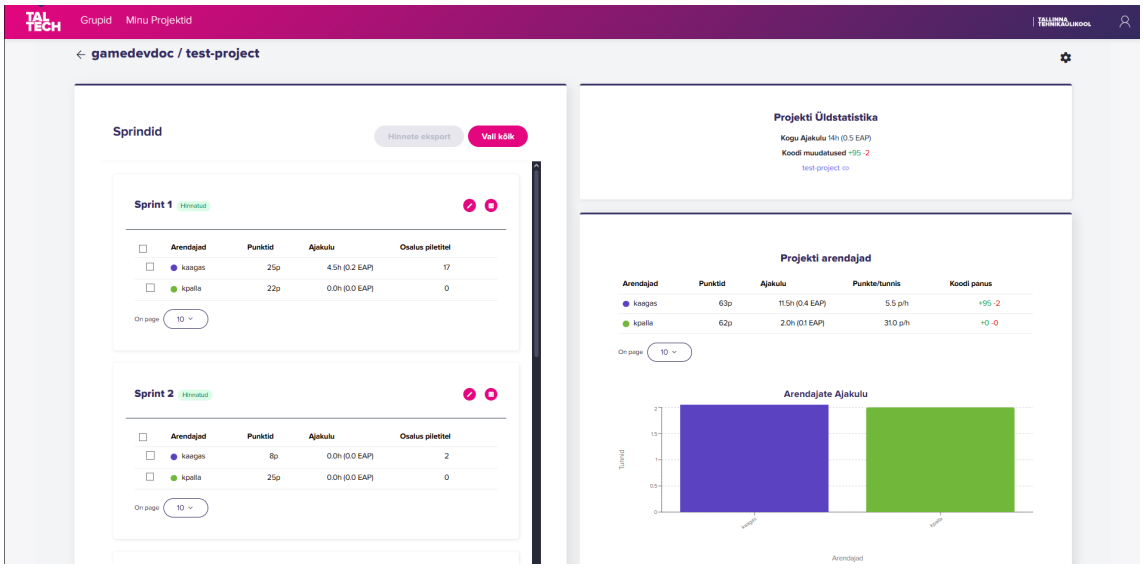
Joonis 8. Grupi lisamise vaade Cognate’is.



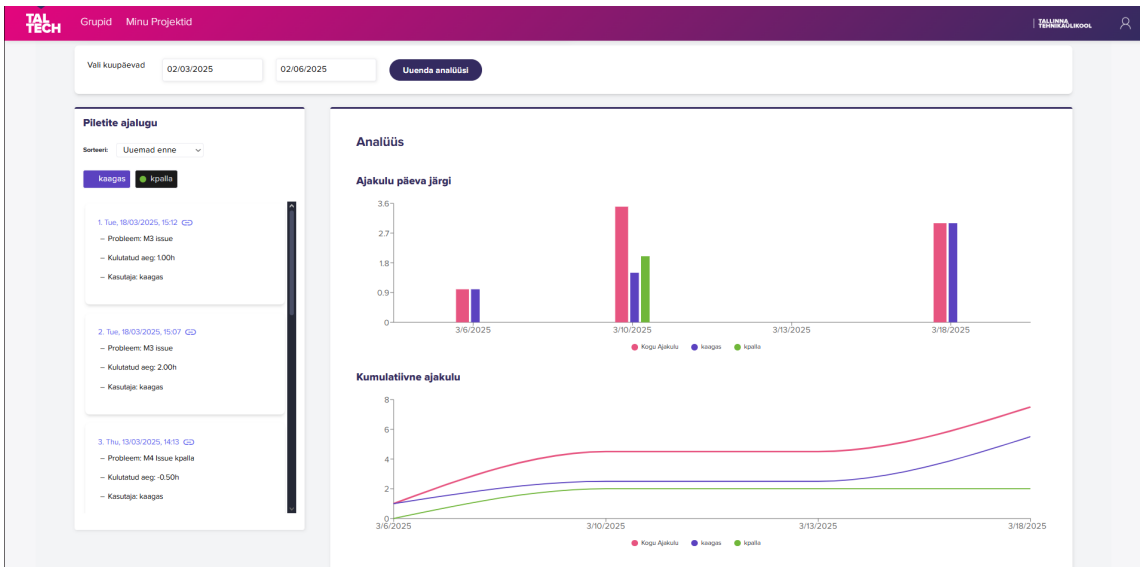
Joonis 9. Hindamispuu vaade Cognate'is.



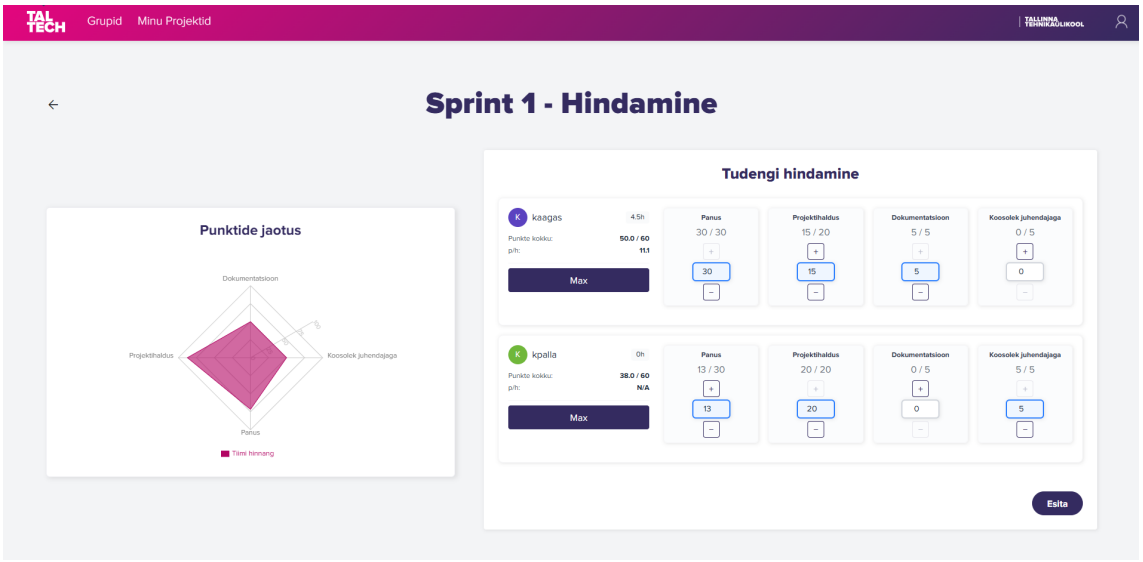
Joonis 10. Projektide üldvaade Cognate'is.



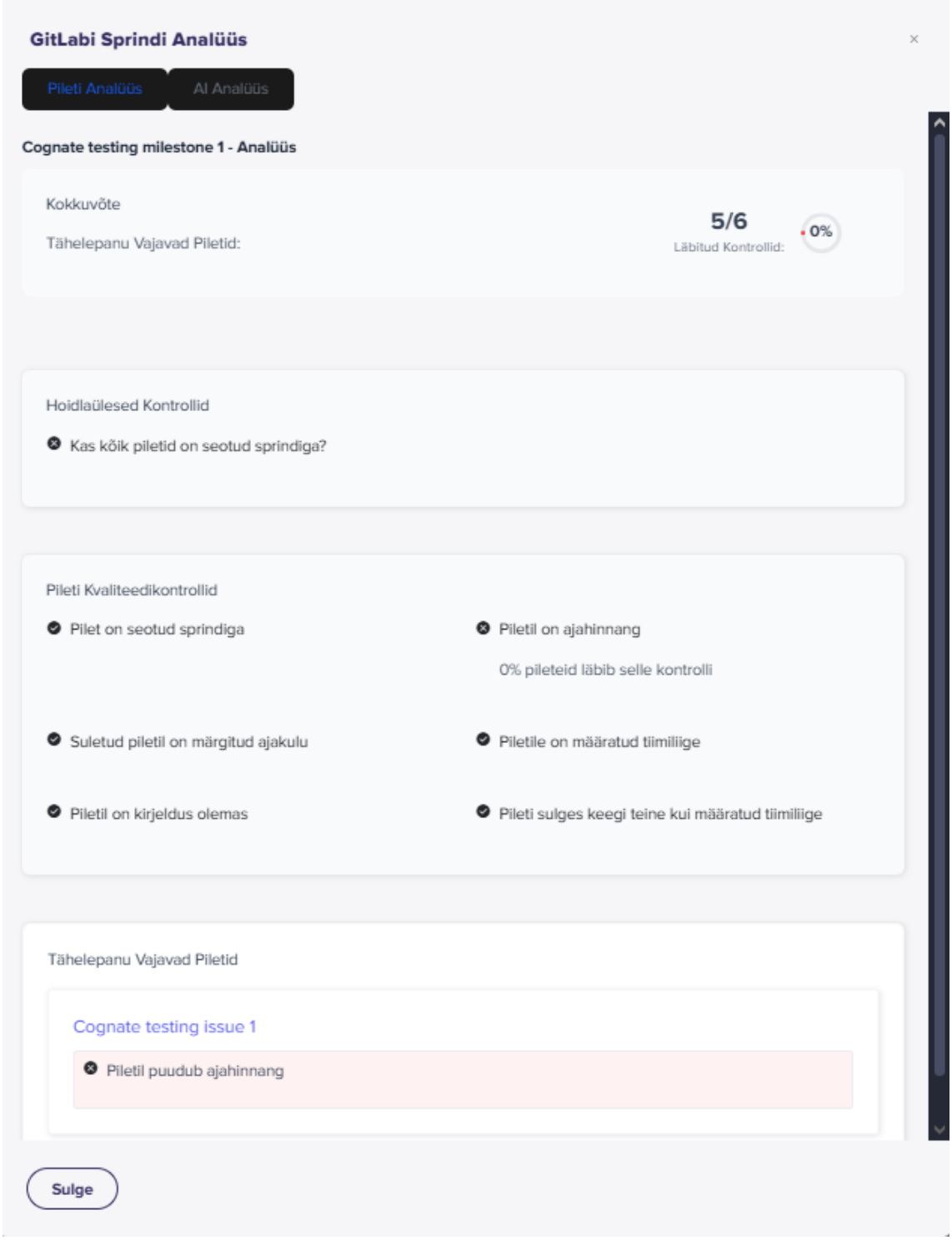
Joonis 11. Õppejõu projekti vaate ülemine osa Cognate'is.



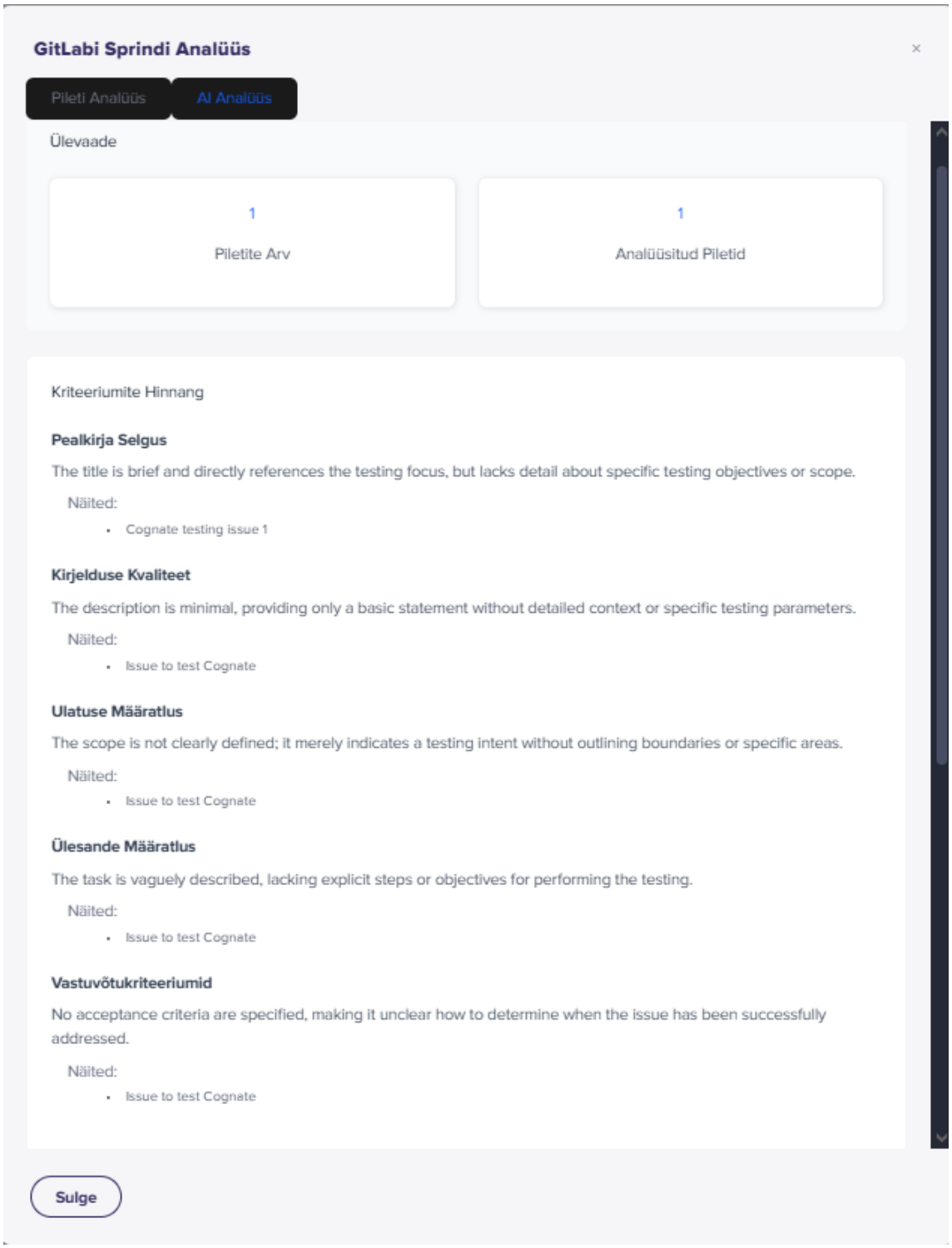
Joonis 12. Õppejõu projekti vaate alumine osa Cognate'is.



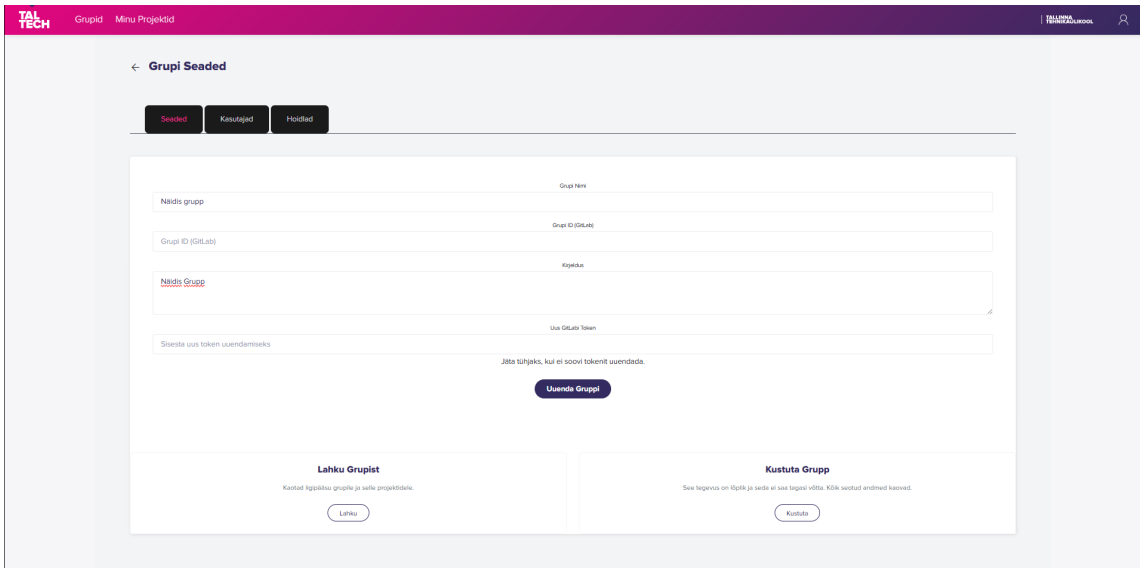
Joonis 13. Projekti hindamise vaade Cognate'is.



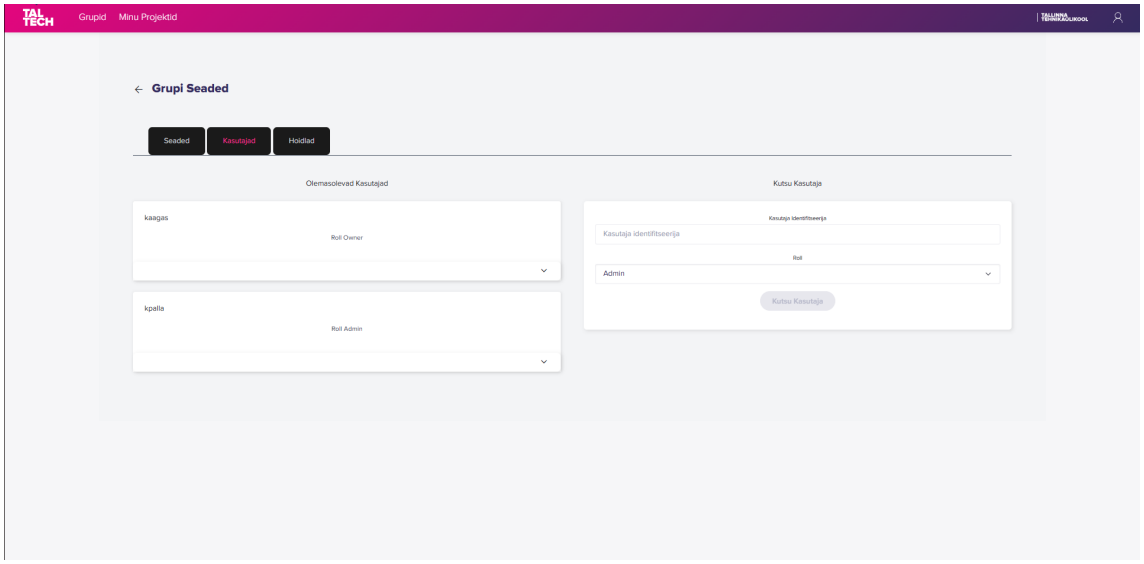
Joonis 14. Pileti analüüsi vaade Cognate'is.



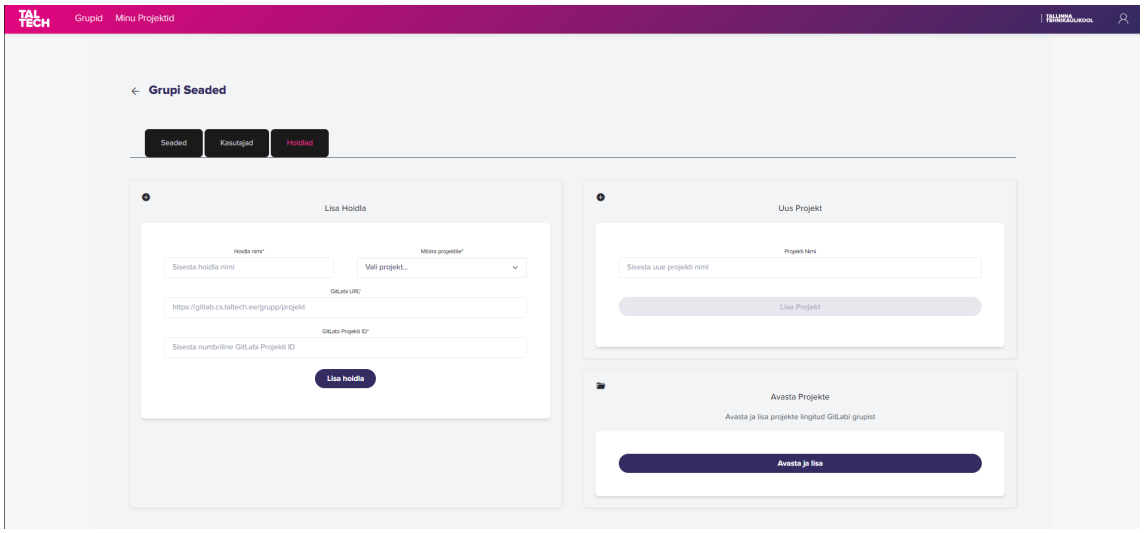
Joonis 15. Pileti sisu konteksti analüüsi vaade Cognate'is.



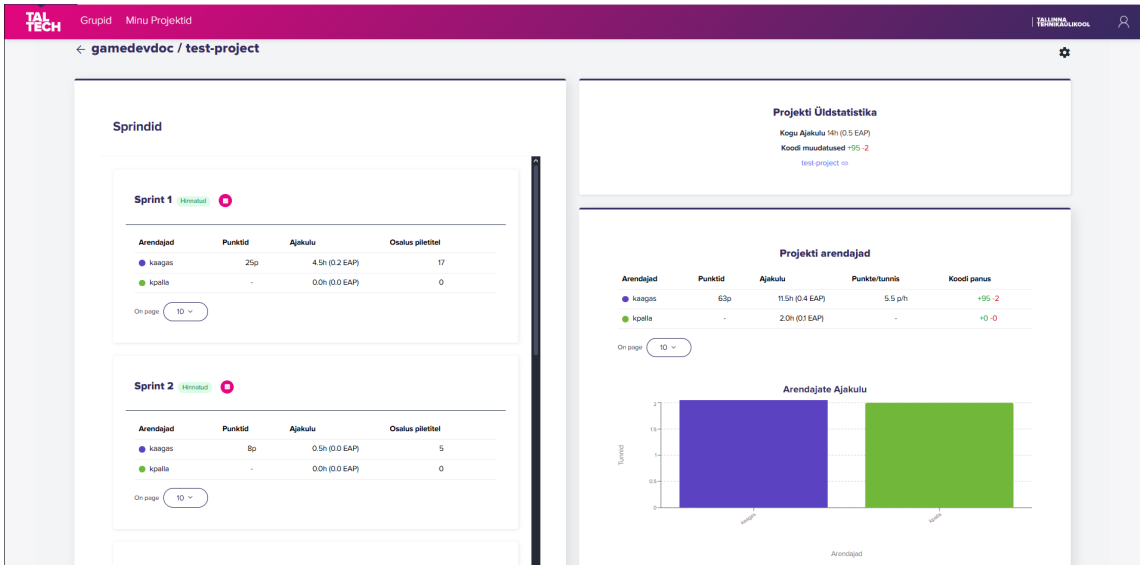
Joonis 16. Grupi seadete vaade Cognate'is.



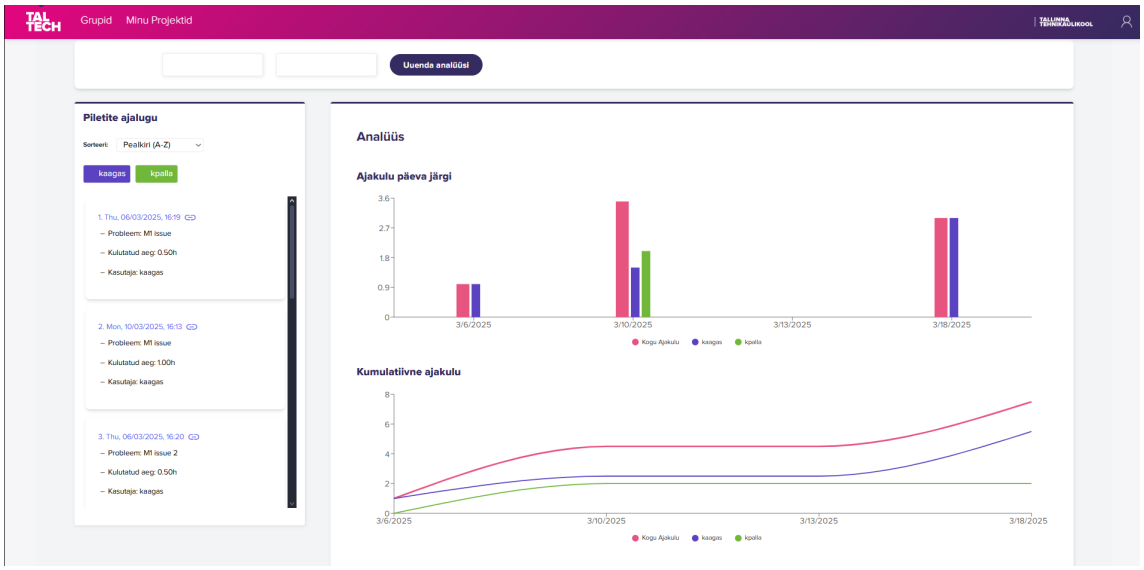
Joonis 17. Grupi kasutajate haldamise vaade Cognate'is.



Joonis 18. Grupi hoidlate haldamise vaade Cognate'is.



Joonis 19. Tudengivaate ülemine osa Cognate'is.



Joonis 20. Tudengivaate alumine osa Cognate'is.

Lisa 3 – Cognate'i küsitlus abiõppejõududele

Tere! Meie oleme informaatika 3. kursuse tudengid Kris-Sten Pallas ja Karl Agasild. Bakalaureusetöö käigus tegime Cognate'i rakendusele uue React-põhise eesrakenduse ning lisasime uusi funktsionaalsusi.

Antud küsimustiku abil soovime kaardistada, kuidas on Cognate'i kasutajakogemus muutunud ning mida arvatakse uutest Cognate'i funktsionaalsustest.

Cognate enne 2025.a uuendusi

*Palun mõelge vastates Cognate'i versioonile, mida kasutasite **enne** 2025. aasta kevaisi uuendusi.*

1. Hinnake oma üldist rahulolu Cognate platvormiga *enne* 2025.a uuendusi.

	1	2	3	4	5	
Väga rahulolematu	○	○	○	○	○	Väga rahul

2. Millised olid peamised probleemid või kitsaskohad, mida kogesite Cognate'i kasutamisel *enne* 2025.a uuendusi?

- ☐ Gruppidesse liikmete lisamine on ebamugav/segadust tekitav
- ☐ Projektidest on tudengid puudu
- ☐ Kuvatud ajakulu on vale
- ☐ Kuvatud muudetud koodiridade arv on vale
- ☐ Sprintide sobitamine on ebamugav
- ☐ Punktide panemine on ebamugav
- ☐ Other: _____

3. Kas kogesite varasemas versioonis probleeme platvormi visuaalse poolega (nt komponentide ebaühtlus, erinevus ülikooli teistest süsteemidest)?

- ○ Jah, sageli
- ○ Jah, mõnikord
- ○ Harva
- ○ Ei kogenud

4. Hinnake gruppide haldamise (lisamine, liikmete haldus, sätete muutmine) lihtsust *enne* 2025.a uuendusi.

	1	2	3	4	5	
Väga keeruline	○	○	○	○	○	Väga lihtne

5. Hinnake projektide hindamise mugavust *enne* 2025.a uuendusi.

	1	2	3	4	5	
Väga ebamugav	○	○	○	○	○	Väga mugav

6. Hinnake projekti sprintide sobitamise mugavust *enne* 2025.a uuendusi.

	1	2	3	4	5	
Väga ebamugav	○	○	○	○	○	Väga mugav

Cognate pärast 2025.a uuendusi

*Palun mõelge vastates Cognate'i **praegusele**, uuendatud versioonile.*

7. Hinnake oma üldist rahulolu Cognate platvormiga *pärast* 2025.a uuendusi.

	1	2	3	4	5	
Väga rahulolematu	○	○	○	○	○	Väga rahul

8. Millised peamised probleemid või kitsaskohad Cognate platvormis on paranenud *pärast* 2025.a uuendusi?

- ☐ Gruppidesse liikmete lisamine on ebamugav/segadust tekitav
- ☐ Projektidest on tudengid puudu
- ☐ Kuvatud ajakulu on vale
- ☐ Kuvatud muudetud koodiridade arv on vale
- ☐ Sprintide sobitamine on ebamugav
- ☐ Hinnete panemine on ebamugav
- ☐ Other: _____

9. Kas olete uuendatud versioonis märganud, et platvormi visuaalne pool (kujundus, komponentide ühtsus) on paremini kooskõlas ülikooli teiste süsteemidega?

- ○ Jah, oluliselt parem
- ○ Jah, mõnevõrra parem
- ○ Ei ole märganud erinevust
- ○ Ei, on halvem
- ○ Ei oska öelda

10. Hinnake gruppide haldamise (lisamine, liikmete haldus, sätete muutmine) lihtsust *pärast* 2025.a uuendusi.

	1	2	3	4	5	
Väga keeruline	○	○	○	○	○	Väga lihtne

11. Hinnake projektide hindamise mugavust *pärast* 2025.a uuendusi.

	1	2	3	4	5	
Väga ebamugav	○	○	○	○	○	Väga mugav

12. Hinnake projekti sprintide sobitamise mugavust *pärast* 2025.a uuendusi.

	1	2	3	4	5	
Väga ebamugav	○	○	○	○	○	Väga mugav

Uued funktsionaalsused

13. Kui kaua aega minutites kulus teil ühe sprindi hindamiseks *enne* 2025.a uuendusi?

Vastus: _____

14. Kui kaua aega minutites kulus teil ühe sprindi hindamiseks *pärast* 2025.a uuendusi?

Vastus: _____

15. Kui kasulikuks peate piletite ja projektihalduse analüüsimise funktsionaalsust?

	1	2	3	4	5	
Üldse mitte kasulikuks	○	○	○	○	○	Väga kasulikuks

16. Kui kasulikuks peate tudengivaadet üliõpilaste eneseanalüüsi ja motivatsiooni toetamisel?

	1	2	3	4	5	
Üldse mitte kasulik	○	○	○	○	○	Väga kasulik

Lisa 4 – Cognate'i küsitlus tudengitele

Tere! Meie oleme informaatika 3. kursuse tudengid Kris-Sten Pallas ja Karl Agasild. Bakalaureusetöö käigus tegime Cognate'i rakendusele uue React põhise eesrakenduse ning lisasime uusi funktsionaalsusi.

Antud küsimustiku abil soovime kaardistada, milline väärtus on tudengi vaatel tudengitele.

1. Kas oleksite soovinud kasutada Cognate'i tudengi vaadet Erialatutvustuse ja Tarkvaraarenduse projekti ainete raames?

- ○ Jah
- ○ Ei

2. Kui kasulikuks peate Cognate'i tudengi vaadet?

	1	2	3	4	5	
Mitte väga kasulikuks	○	○	○	○	○	Väga kasulikuks

3. Kui lihtne oli mõista ja kasutada tudengi vaadet?

	1	2	3	4	5	
Väga keeruline	○	○	○	○	○	Väga lihtne

4. Mida arvate Pileti Analüüsist?

Vastus: _____ _____ _____

5. Mida arvate AI Analüüsist?

Vastus: _____ _____ _____