TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Dmitri Kuznetsov 213077IAIB

ÜLIKOOLI AMETLIKE INFOKANALITE KOORMUS JA VALIKULISE TEABE LEVITAMISE TÕHUSUS: INFOÜLEKÜLLUSE VÄHENDAMINE E-KUULUTUSTE TAHVLI KAUDU

Bakalaureusetöö

Juhendaja: Karl-Erik Karu
MSc

Tallinn 2025

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Dmitri Kuznetsov

04.06.2025

Annotatsioon

Lõputöö eesmärk on välja töötada teavitussüsteemi prototüüp, mis aitab vähendada infoüleküllust ülikooli ametlikes infokanalites ning parandada suhtlust programmijuhtide ja tudengite vahel. Süsteem peab võimaldama teavituste sihipärast edastamist, isikupärastamist ning tagasiside kogumist.

Töö eesmärgi saavutamiseks rakendatakse kombineeritud uurimismetoodikat, mis hõlmab probleemanalüüsi, küsitlusuuringuid tudengite ja programmijuhtide hulgas ning olemasoleva protsessi kaardistamist. Saadud sisendi põhjal määratletakse süsteemi funktsionaalsed ja mittefunktsionaalsed nõuded ning töötatakse välja sobiv lahendus.

Töö tulemusena valmib veebipõhine prototüüp, mis võimaldab sihtrühmapõhist teadete haldamist, mallide loomist ja taaskasutust, teadete filtreerimist ning lugemisstatistika jälgimist. Loodud lahendus vastab kasutajate ootustele ja seatud nõuetele ning pakub tugeva lähtekoha süsteemi edasiseks arendamiseks ja integreerimiseks ülikooli infosüsteemiga.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 6 peatükki, 13 joonist.

Abstract

The load on university official information channels and the effectiveness of selective information dissemination: reducing information overload through a digital notification board

The aim of this thesis is to develop a notification system prototype that helps reduce information overload in university communication channels and improves the flow of information between program managers and students. The system is intended to support targeted communication, message personalization, and feedback collection to ensure that important messages are not lost among less relevant ones.

The development process begins with a problem analysis and continues with user surveys among students and program managers, as well as the mapping of current communication processes. Based on the gathered input, functional and non-functional requirements for the system are formulated, ensuring that the solution aligns with actual user needs.

As a result, a web-based prototype is developed, enabling targeted notifications, reusable message templates, filtering and prioritization, and the ability to track which students have read a message. The solution meets the defined goals and offers a solid foundation for future development. In the long term, the system can also be integrated into the university's existing digital platforms to further streamline communication and reduce the need to manage information across multiple tools.

The thesis is in Estonian language and contains 33 pages of text, 6 chapters, 13 figures.

Lühendite ja mõistete sõnastik

AI	*Artificial Intelligence*, tehisintellekt
API	*Application Programming Interface*, rakendusliides
AS-IS	Kirjeldab hetkel toimuvat protsessi
Backend	serveripoolne rakendus
CSS	*Cascading Style Sheets*, kaskaadlaadistik
DOM	*Document Object Model*, dokumendiobjektide mudel
DSR	*Design Science Research*, disainiteaduslik uurimus
ER	*Entity–relationship*, entiteet–seos
Frontend	kasutajaliides
HTML	*HyperText Markup Language*, hüperteksti märgistuskeel
HTTP	*HyperText Transfer Protocol*, hüperteksti edastusprotokoll
JSON	*JavaScript Object Notation*, lihtne andmevahetusvorming
JWT	*JSON Web Token*, JSON-i põhine veebi turvatõend
REST	*Representational State Transfer*, esitusoleku siire
SDLC	*Software Development Life Cycle*, tarkvaraarenduse elutsükkel
SQL	*Structured Query Language*, struktureeritud päringute keel
TL;DR	*Too Long; Didn't Read*, kasutatakse, et anda kiire ülevaade pikemast tekstist
TO-BE	Kirjeldab soovitud protsessi pärast muudatusi või parandusi
URL	*Uniform Resource Locator*, ühtne ressursilokaator
ÕIS	õppeinfosüsteem

Sisukord

1 Sissejuhatus 9
1.1 Probleem 9
1.2 Eesmärk 10
1.3 Töö struktuur 11

2 Metoodika 12
2.1 Objekti kirjeldus 12
2.2 Arendustööriistad 12
2.2.1 Raamistikud 13
2.2.2 Keeled 13
2.2.3 Tehnoloogiad 13
2.3 Arendusmetoodika 14
2.4 Uurimismetoodika 16

3 Nõuded 18
3.1 Alternatiivsed lahendused 18
3.1.1 AI-põhine teabe kokkuvõtmine (AI summarization) 18
3.1.2 TL;DR 18
3.1.3 Discord 19
3.2 Küsitluste tulemused 19
3.2.1 Tudengiküsitluse tulemused 19
3.2.2 Programmijuhtide tulemused 21
3.2.3 Küsitluste analüüs ja järeldused 22
3.3 Olemasolev AS-IS protsess 23
3.4 Funktsionaalsed nõuded 24
3.5 Mittefunktsionaalsed nõuded 24
3.6 Kasutajalood 25

4 Tulemus 26
4.1 TO-BE protsessimudel 26
4.2 Arhitektuur 27
4.2.1 Esitluskiht 27
4.2.2 Rakendusloogika kiht 28
4.2.3 Andmebaasikiht 29
4.2.4 Kihtide seosed 30

4.3 Vaated . . . 30
4.4 Testimine . . . 33
4.4.1 Automatiseeritud funktsionaalsed testid . . . 33
4.4.2 Visuaalne ja manuaalne testimine . . . 34

5 Analüüs ja järeldused . . . 35
5.1 Nõuetele vastavus . . . 35
5.1.1 Funktsionaalsete nõuete täitmine . . . 35
5.1.2 Mittefunktsionaalsete nõuete täitmine . . . 36
5.2 AS-IS ja TO-BE protsesside võrdlus . . . 36
5.3 Hinnang tööle . . . 38
5.4 Edasiarendus . . . 39

6 Kokkuvõte . . . 41

Kasutatud kirjandus . . . 42

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks . . . 44

Lisa 2 – „Lugemata“ vaade . . . 45

Lisa 3 – „Lemmikud“ vaade . . . 46

Lisa 4 – Teate detailvaade tudengi vaates . . . 47

Lisa 5 – Mallide haldamise vaated . . . 48

Lisa 6 – Autentimisvaated . . . 50

Jooniste loetelu

1 DSR mudel [11]. 14
2 Tarkvaraarenduse elutsükkel [10]. 15

3 Kasuliku info osakaal tudengite hinnangul. 19
4 Peamised raskused ametlike e-kirjade lugemisel ja jälgimisel. 20
5 Kasutajate ootused digitaalse teavitustahvli funktsioonidele. 20
6 Programmijuhtide peamised väljakutsed ametlike teadete edastamisel. . . 21
7 Programmijuhtide jaoks olulisemad funktsioonid tõhusaks teavituste edastamiseks. 22

8 Rakenduse arhitektuur. 27
9 Olemi-suhte diagramm. 29
10 „Teated“ vaade. 31
11 Teate detailvaade programmijuhi vaates. 32
12 „Lisa uus teade“ vaade. 32
13 „Mallid“ vaade. 33

1 Sissejuhatus

Digiajastul toimub ülikoolides infovahetus paljude formaalsete kanalite kaudu, millest levinumad on e-post, tudengiportaalide uudistevood ja teadetetahvlid hariduse infosüsteemides. Need on mõeldud üliõpilastele nii ametliku õppetööga seotud teabe (kursuse materjalid, eksamite kuupäevad, õppekorralduse muudatused) kui ka valikuliste teadaannete (praktika- ja tööpakkumised, avalikud loengud, seminarid, huviringid) edastamiseks.

Info digitaalne edastamine toob kaasa võimaluse levitada sõnumeid kiirelt ja suurele auditooriumile, kuid sellega kaasneb paratamatult ka infohulga märkimisväärne kasv. Kuna teateid saadetakse sageli eri allikatest, näiteks kursuste õppejõududelt, osakondadest või ülikooli erinevatest üksustest, suureneb üldine teadete arv järjepidevalt ning võib hakata negatiivselt mõjutama info selgust ja loetavust. Ilma sobivate vahenditeta on kasutajatel keeruline tõhusalt kontrollida neile saabuvate teadete voogu, mis omakorda põhjustab probleeme nii sõnumite edastajatele kui ka vastuvõtjatele.

Sellisest olukorrast tulenevad probleemid, nagu üliõpilaste vähene motivatsioon regulaarselt teavitusi kontrollida ja programmijuhtide raskused sõnumite efektiivsel levitamisel, on muutunud oluliseks kitsaskohaks ning nõuavad süsteemset ja läbimõeldud lahendust.

1.1 Probleem

Praktikas on ilmnenud, et info hulk võib muutuda nii suureks, et õpilastel on raskusi olulise info eristamisega vähem olulisest [1]. Mitmes uuringus teatasid õpilased, et kui enamik teateid ei olnud nende jaoks olulised, vähenes nende motivatsioon pidevalt oma postkasti või uudistevoogu kontrollida [2]. Seetõttu võib tähelepanuta jääda info, mis on akadeemilises plaanis eluliselt tähtis, nagu eksamitele registreerumise ajad, stipendiumitaotluste esitamise tähtajad või kavandatavad muudatused kursuse korralduses.

Olukord on koormav mitte ainult õpilastele. Programmijuhid, kelle kohustuste hulka kuulub üliõpilaste teavitamine ja akadeemiliste otsuste edastamine, kohtavad sageli ebamugavusi, kuna peavad kulutama palju aega teadaannete ettevalmistamisele ja levitamisele, mis sageli kattuvad muude teabeallikatega. Näiteks võib programmijuht soovida saata mõne õppekavaga seotud lühisõnumi, kuid rakendatava süsteemi ühtsuse puudumise tõttu tuleb see uuesti sisestada või mitmesse erinevasse kanalisse kopeerida. See tekitab ülekoormuse olukorra, kus õppijad saavad sama sõnumi mitu korda, kuid erinevas kontekstis või

veidi muudetud kujul, mis omakorda vähendab õigeaegse info märkamise tõenäosust.

Lisaks võib vastuolu tekkida siis, kui programmijuhid püüavad vähendada tarbetuid teadaandeid, kuid õpilased kurdavad endiselt liigse teabe levitamise üle. Selline olukord näitab tervikliku lahenduse puudumist: ühelt poolt puudub personaliseeritud filtreerimine, teisalt puudub tsentraliseeritud infohaldusmehhanism, mis võimaldaks programmijuhtidel mugavalt hallata sõnumite sihtgruppe. Tulemuseks on meilide, loendite või teadetetahvlite ülekoormus, mis vähendab nii sisukate sõnumite tõhusust kui ka kasutajate rahulolu.

Aastate jooksul on ülikoolid katsetanud erinevaid lahendusi, nagu teabe jagamine spetsiaalsetes loendites (nt iga kursuse jaoks üks) või automaatsete sortimisreeglite loomine, kuid paljudel üliõpilastel on endiselt ebamugav oma "ametlikku postkasti" jälgida, kuna nad ei leia alati vajalikku teavet kohe [2]. Samuti tuleb ette olukordi, kus õpilased ei ava oma e-kirju üldse, kuna on varem saanud liiga palju sõnumeid, mis ei ole nende õppekavaga seotud. Seetõttu võidakse ametlikku, kuid olulist teadaannet (näiteks aine ajakava muudatust) täielikult ignoreerida. Seetõttu on vaja parandada ülikooli infokanalite toimimist nii, et tudengid leiaksid vajaliku info kiiremini ning programmijuhid saaksid edastada teateid täpselt neile, kellele need on mõeldud.

1.2 Eesmärk

Käesoleva bakalaureusetöö põhieesmärk on välja töötada lahendus, mis vähendab ülikooli ametlike infokanalite koormust ja tõstab üliõpilaste motivatsiooni oluliste teadete regulaarseks jälgimiseks. Selle üldise eesmärgi saavutamiseks püstitati neli alaeesmärki:

1. Uurida tudengite ja programmijuhtide kogemust senise ametliku infovahetuse korraldamisel kvantitatiivsete ja kvalitatiivsete meetodite abil. Eesmärk on saada ülevaade nii praegustest kitsaskohtadest kui ka kasutajate soovidest süsteemi efektiivsemaks muutmisel.
2. Kujundada lahendus (nt vabatahtlik e-kuulutuste tahvel), mis eraldab või filtreerib valikulist infot ametlikust teabest ning võimaldab kasutajal määrata oma eelistused. See peaks võimaldama üliõpilastel õppimise ajal selgemalt eristada olulisi teadaandeid üldistest kuulutustest ning ühtlasi vähendama programmijuhtide töökoormust, kuna lihtsustab neil teadete sisestamist ja edastamist.
3. Hinnata lahenduse efektiivsust objektiivsemate mõõdikutega. Kuna hetkel puudub filtreerimisvõimalus ja kõik sõnumid on koondatud ühte infomassiivi, siis tuleb esmalt kaardistada olemasolev (AS-IS) protsess koos selle kitsaskohtadega ning seejärel võrrelda potentsiaalse (TO-BE) lahendusega, mis sõnumeid sorteerib ja filtreerib. Seega saab mõõdikute abil selgitada, kas uus süsteem vähendab info

ülekoormust, lihtsustab programmijuhtide haldusprotsessi ja parandab olulise info kättesaadavust.

1.3 Töö struktuur

Sissejuhatusele järgneb peatükk, kus kirjeldatakse töös kasutatud metoodikaid, mida rakendatakse meilide filtreerimise ja sortimise lahenduse loomisel. Samuti tutvustatakse põhitööriistu ja selgitatakse protsessi probleemide kaardistamisest esialgse prototüübi testimiseni.

Kolmandas peatükis on välja toodud nõuded loodavale süsteemile, mille väljatöötamisel on arvestatud nii üliõpilaste kui ka programmijuhtide küsitluste tulemusi. Samuti analüüsitakse olemasolevat lahendust.

Neljas peatükk on pühendatud tulemustele, kus kirjeldatakse süsteemi arhitektuuri ja tuuakse välja testimise tulemused, kas lahendus vähendab info ülekoormust ja lihtsustab programmijuhtide tööd.

Viiendas peatükis analüüsitakse loodud lahenduse efektiivsust ja tehakse järeldused. Samuti annab see ülevaate järgmistest sammudest, mis võivad arendust laiendada või täiustada.

Töö lõpus on kokkuvõte, kus tuuakse välja kõige olulisemad järeldused ja vaadeldakse, kas seatud eesmärgid said täidetud.

2 Metoodika

Käesolevas peatükis antakse ülevaade töö metoodilistest alustest, kirjeldatakse uuritavat objekti, kasutatavaid arendustööriistu ja arendusmetoodikat. Lisaks esitletakse uurimismetoodikat, mis hõlmab nii üliõpilaste kui ka programmijuhtide küsitlusi, et tuvastada praeguse süsteemiga seotud kitsaskohad ja ootused.

2.1 Objekti kirjeldus

Käesoleva töö keskmes on ülikooli ametlike teadete edastamise protsess, mis praegu toimub peamiselt lihtsa e-posti teel. Täpsemalt saavad tudengid kõik teated (teave nende õpingute, praktika- või tööpakkumiste kohta, sündmuste teated) ühes e-posti voos, ilma võimaluseta neid sõnumeid automaatselt sortida või filtreerida. See toob kaasa teabe üleküllluse, kuna olulised sõnumid võivad teadete massis kaduma minna.

Samas puuduvad programmijuhtidel ka tööriistad, mis võimaldaksid teateid kiiresti koostada ja mugavalt üliõpilastele saata. Kui programmijuht saab organisatsioonilt teate (nt kutse seminarile või stipendiumipakkumise), peab ta sageli kirja käsitsi kopeerima ja uuesti tudengite nimekirja või programmi saatma. Selline korduv käsitsi kopeerimine suurendab nii ajakulu kui ka ohtu, et sõnum saadetakse valele adressaadile või ei edastata seda üldse. Kuna süsteemis puuduvad filtreerimise või automaatse sorteerimise funktsioonid, on info üleküllluse vähendamiseks ja oluliste sõnumite nähtavamaks muutmiseks vaja nii õpilaste kui ka programmijuhtide huve arvestavat lahendust.

2.2 Arendustööriistad

Selles peatükis antakse ülevaade vahenditest ja tehnoloogiatest, mida plaanitakse rakenduse väljatöötamisel kasutada. Kuna töö eesmärk on luua tõhus lahendus ülikooli e-kirjade filtreerimiseks ja sorteerimiseks, tuleb valitud tööriistadel toetada nii paindlikku kasutajaliidese (frontend) arendust kui ka suure koormusega serveri- ja andmebaasitoiminguid. Alljärgnevalt käsitletakse esmalt kasutatavaid raamistikke, seejärel programmeerimiskeeli ning lõpuks olulisi tehnoloogiaid, mis kogu süsteemi tervikuks ühendavad.

2.2.1 Raamistikud

Kasutajaliidese (frontend) loomiseks valitakse React.js, mis toetab moodulitel põhinevat arendusstiili. React.js on Facebooki loodud JavaScripti teek, mille abil saab luua komponentidel põhinevaid veebirakendusi [3]. Selle põhiline eelis seisneb virtuaalse DOM (*Document Object Model*) kasutamises, mis muudab rakenduse dünaamiliseks ja võimaldab kiirelt teateid (e-kirju) uuendada ilma kogu lehte uuesti laadimata. Kuna tudengid peaksid rakendust kasutades saama reaalajas filtreerida ja sorteerida suure hulga kirju, on React.js selle jaoks hea valik.

2.2.2 Keeled

HTML (*HyperText Markup Language*) ja CSS (*Cascading Style Sheets*) on veebidisaini aluseks [4], [5]. Need tehnoloogiad võimaldavad luua paindlikku ülesehitust ja kujundust React'i komponentidele, nii et tudengitele on tagatud visuaalselt selge, loogiline ja kiirelt reageeriv liides. HTML struktureerib sisu, samas kui CSS annab võimaluse kujundada stiile, mis aitavad kasutajal infomassi seest vajalikku infot märgata.

Rakenduse põhiloogika kirjutatakse JavaScriptis. JavaScript on üks veebiarenduse põhielemente, mida kasutatakse nii kliendipoolses (frontend) kui ka serveripoolses (backend) arenduses [6]. Kuna projektis on plaanis rakendada nii React.js kui ka Node.js, on ühe keele kasutamine mugav.

2.2.3 Tehnoloogiad

Serveripoolse loogika rakendamiseks valitakse Node.js, mis on JavaScripti-põhine keskkond üritustepõhise arhitektuuriga [7]. Kuna programmijuhtidel võib korraga olla vaja saata hulganisti e-kirju (sh eristada neid kursuse, õppekava või valdkonna kaupa), on Node.js suurepärane lahendus suure hulga samal ajal töötavate päringute haldamiseks. Node.js lihtsustab ka integratsiooni React.js-iga, sest kogu arendusprotsess käib ühes keeles ning kasutatakse sarnaseid töövooge.

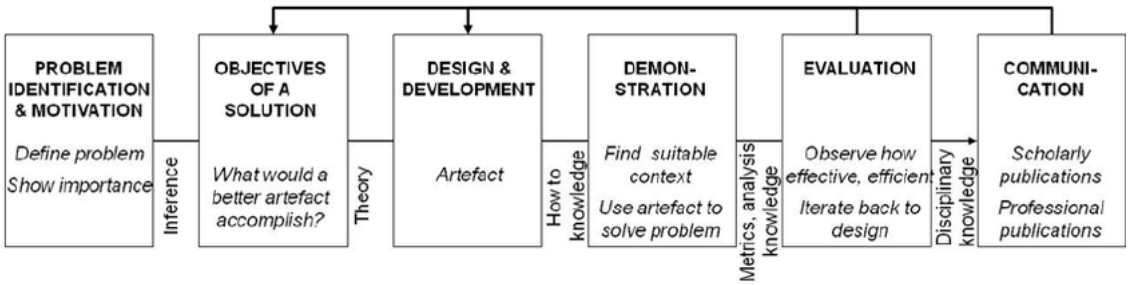
Andmebaasi valikuks on PostgreSQL, mis on tuntud oma kõrge jõudluse, tervikluse tagamise ja laiendamisvõime poolest [8]. Relatsioonimudeli kasutamine võimaldab salvestada struktureeritult nii teavituste metaandmeid kui ka kasutajate eelistusi (nt kohandatud filtrid), tagades ühtlasi andmetervikluse.

2.3 Arendusmetoodika

Käesoleva bakalaureusetöö lahenduse loomisel kasutatakse kombineeritud arendusmetoodikat, mis tugineb DSR (*Design Science Research*) teadusparadigmale ja tarkvaraarenduse elutsüklile (Software Development Life Cycle, SDLC) [9], [10]. Selline lähenemine võimaldab mitte ainult luua lahenduse, mis aitaks vähendada infoüleküllust ülikooli ametlikes infokanalites, vaid hinnata ka selle praktilist väärtust reaalsetes kasutustingimustes ning süstemaatiliselt struktureerida kogu arendusprotsessi.

Arendusprotsessi lähtepunktiks on probleemi täpne määratlemine. Selleks, et mõista, millised raskused kaasnevad ülikooli ametlike infokanalite praeguse toimimisega, kaardistatakse tudengite ja programmijuhtide kogemused ning vajadused. Eesmärk on välja selgitada, kas olemasolev infosüsteem toetab õppetegevust või takistab seda. Probleemi mõistmiseks kasutatakse empiirilisi vahendeid (vt 2.4), mille tulemused on aluseks edasistele arengutele.

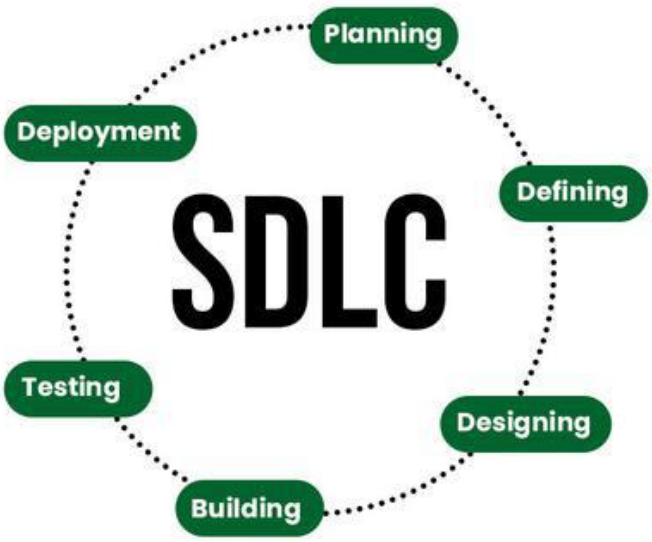
Design Science Research raames keskendub töö lahenduse kui artefakti loomisele, hindamisele ja iteratiivsele täiustamisele (Joonis 1). Artefakti loomisel ei keskenduta ainult tehnilisele arengule, vaid kogu protsessi käsitletakse kui uurimuslikku arendustegevust, mille eesmärk ei ole üksnes toimiva süsteemi loomine, vaid ka teadmiste kogumine ja analüüs selle kohta, kuidas selline lahendus reaalses õpikeskkonnas toimida saab. Iteratiivne lähenemine võimaldab reageerida kasutajate tagasisidele ja järk-järgult täiustada loodavat süsteemi vastavalt tegelikele vajadustele.



Joonis 1. DSR mudel [11].

Töös kasutatav tarkvaraarenduse elutsükkel (SDLC) on jagatud mitmeks järjestikuseks etapiks, millest olulisemad on planeerimine, nõuete kaardistamine, süsteemi projekteerimine ja arendus (Joonis 2). Planeerimisetapis selgitatakse välja probleemi olemus ja seatakse töö eesmärgid. Samuti määratakse kindlaks osapooled, kellele loodav süsteem on suunatud: peamiselt üliõpilased ja programmijuhid, samuti TalTechi infosüsteemide arendusosakond, kellega võib lõplikuks integreerimiseks vaja minna koostööd. Nõuete kaardistamise etapp kirjeldab olemasolevat protsessi (AS-IS), kus teateid ei filtreerita ega isikupärastata ning

programmijuhid peavad paljud teated käsitsi kopeerima ja korduvalt edastama. Seejärel töötatakse välja TO-BE protsess või tulevane lahendus, kus iga kasutaja saab määrata teadetele filtrid ja prioriteedid ning programmijuhid saavad teateid otse süsteemi avaldada, ilma et peaks neid käsitsi kopeerima ja edasi saatma. Siin on määratletud nii funktsionaalsed nõuded (nt kategooriahaldus, sorteerimisvõimalused) kui ka mittefunktsionaalsed nõuded (nt süsteemi jõudlus, kasutusmugavus, juurdepääsetavus).



Joonis 2. Tarkvaraarenduse elutsükkel [10].

Projekteerimisetapis koostatakse süsteemi arhitektuurne skeem, mis määratleb seosed komponentide ja andmevoogude vahel. Kavandatakse kasutajaliides Reacti raamistikus, serveripoolsed teenused Node.js keskkonnas ning PostgreSQL-i baasil luuakse andmemudel, mis võimaldab struktureerida teadete, kasutajaprofiilide ja seadistuste salvestamist. Samuti on plaanis välja töötada API (*Application Programming Interface*) liidesed, mille abil saab süsteem tulevikus integreeruda TalTechi üliõpilasportaaliga.

Kui ülaltoodud sammud on lõpule viidud, liigub protsess edasi arendusfaasi, kus kavandatud lahendus hakatakse samm-sammult tehniliselt teostama. Arendus toimub etapiviisiliselt, järgides iteratiivset mudelit: esmalt juurutatakse olulisemad funktsioonid (nt teadete sisestamine, sorteerimine) ja seejärel lisatakse järk-järgult lisavõimalusi (nt kasutajapõhine filtreerimine, ajastatud teavitused). Arendusega kaasneb pidev testimine ja valideerimine, mille käigus hinnatakse süsteemi sobivust ja vajadusel korrigeeritakse selle funktsionaalsust. Arendusprotsess lõpeb lahenduse võrdlusega AS-IS ja TO-BE mudelite vahel, kus hinnatakse loodud rakenduse mõju tegelikule töövoole, info edastamise kiirust ja kasutajate rahulolu.

2.4 Uurimismetoodika

Iga uurimistöö keskmes on metoodika valik, mis määrab, kuidas andmeid kogutakse, kuidas neid analüüsitakse ja milliseid järeldusi on võimalik teha. Kuna käesoleva töö eesmärgiks on luua lahendus, mis aitaks vähendada infoülekoormust ülikooli ametlikes e-kanalites, on eriti oluline mõista, kuidas sihtrühmad, eelkõige üliõpilased ja programmijuhid, sellesse probleemi suhtuvad, milliseid kitsaskohti nad näevad ja milliseid lahendusi nemad eelistaksid. Seda informatsiooni saab koguda erinevate uurimismeetodite abil, mis kirjanduses on tavaliselt jagatud kahte suurde rühma: kvantitatiivseks ja kvalitatiivseks.

Kvantitatiivne lähenemine põhineb standardiseeritud või kvantifitseeritavatel andmetel ja võimaldab tulemusi üldistada suuremale valimile [12]. Kavandades küsimustikke suletud küsimustega, Likerti skaala küsimustega või tehes eksperimentaalseid mõõtmisi, saab uurija arvulised tulemused, mida saab statistiliste meetoditega analüüsida (nt sagedus- ja protsentjaotused, keskmised, korrelatsioonid). Kvantitatiivne metoodika on eriti kasulik siis, kui sihtpopulatsioon on suur ja eesmärk on määrata probleemi või nähtuse ulatus, näiteks kui paljudel õpilastel on raskusi olulise teabe leidmisega või kui paljud programmijuhid kulutavad olulisel määral aega õppetööga mitteseotud sõnumite edastamisele.

Kvalitatiivset lähenemist iseloomustab andmete kogumise meetod, mis annab üksikjuhtumitele, arusaamadele ja kogemustele põhjalikuma ülevaate. See võib hõlmata poolavatud või täielikult avatud intervjuusid, fookusgruppe, vaatlusi ja dokumendianalüüsi [13]. Kvalitatiivse meetodi eesmärk on tuua esile detailsemaid aspekte, mida ei saa ainuüksi numbritega käsitleda, näiteks miks õpilased teatud filtrivalikuid ei kasuta, mida programmijuhid keskplatvormilt kõige enam vajavad või kuidas kasutajad tõlgendavad "infoülekoormust" omal moel. See meetod pakub sageli rikkalikku kirjeldust, kuid ei pruugi olla suurte proovide jaoks aja ja ressursside osas kõige kuluefektiivsem.

Käesolevas töös otsustati kasutada kvantitatiivset lähenemist, kuna selle põhieesmärk on saada statistiliselt üldistatavat tagasisidet suuremalt hulgalt kasutajatelt – üliõpilastelt ja programmijuhtidelt. Kvantitatiivne meetod võimaldab mõõta, kui levinud on teatud probleemid (nt olulise info märkamata jäämine) või millised funktsioonid tunduvad kõige vajalikumad (nt sortimisvõimalused). Kuna üks töö fookuseid on kindlaks teha, kui palju õpilasi kogeb õppimist takistavat teabe ülekoormust, sobib just arvuliste tulemuste kogumine ja analüüs, et tuvastada probleemi sagedus, jaotus ning ulatus. Kvalitatiivne meetod aitaks küll selgitada individuaalseid kogemusi, kuid töö peamine eesmärk on siiski saada lai ülevaade hetkeolukorrast, olemasoleva süsteemi probleemidest ja sobivatest lahendusvõimalustest, mistõttu on kvantitatiivne lähenemine valitud kõige otstarbekamaks.

Praktikas kasutatakse kvantitatiivset meetodit kahe iseseisva veebiküsitluse kaudu, millest üks on suunatud tudengitele ja teine programmijuhtidele. Tudengitele suunatud küsitlus keskendub esmalt hinnangule olemasoleva infosüsteemi kasutamisele, seejärel tuvastatakse peamised probleemid ning lõpuks uuritakse, milliseid funktsionaalsusi sooviksid vastajad näha võimalikus uues lahenduses. Programmijuhtide küsitlus keskendub eelkõige teadete edastamise tööprotsessile, tuvastab selle peamised kitsaskohad ning kaardistab ootused ja soovid seoses uue süsteemi võimalustega. Küsitluste läbiviimiseks kasutatakse Google Formsi keskkonda, mis valiti eelkõige seetõttu, et see on kergesti ligipääsetav, võimaldab vastuseid mugavalt koondada ning toetab vahetult kvantitatiivset analüüsi.

3 Nõuded

Käesolevas peatükis vaadeldakse varem välja pakutud alternatiivsed lahendused, seejärel esitatakse küsitluse tulemused, mis annavad empiirilise aluse edasiarendatavatele nõuetele. Samuti määratletakse praegune (AS-IS) protsess, milliseid funktsionaalseid ja mittefunktsionaalseid nõudeid tuleks arvesse võtta ning millised kasutajalood kirjeldavad kõige paremini rakenduse tegelikku kasutamist.

3.1 Alternatiivsed lahendused

Selleks, et leevendada infoülekullust ning parandada kommunikatsiooni tõhusust, on kasutusel erinevaid tehnilisi lahendusi. Käesolevas peatükis käsitletakse kolme peamist alternatiivset meetodit: AI (*Artificial Intelligence*) põhist kokkuvõtlikku sisu edastamist, TL;DR (*Too Long; Didn't Read*) tööriista ja Discordi platvormi kasutamist.

3.1.1 AI-põhine teabe kokkuvõtmine (AI summarization)

Üheks uuemaks ja järjest populaarsemaks lahenduseks infoülekülluse vähendamiseks on AI-põhine sisu kokkuvõtmine. See meetod seisneb selles, et tehisintellekt loeb läbi pikad tekstid ning genereerib nendest lühikese ja arusaadava kokkuvõtte. Näiteks Google Gmail on lisanud funktsiooni "Summarize", mis AI-abil võimaldab kasutajatel kiiresti haarata sõnumite põhisisu ilma kogu teksti lugemata [14]. Selline lähenemine aitab märkimisväärselt vähendada info töötlemiseks kuluvat aega ja keskenduda kiiremini olulisemale.

3.1.2 TL;DR

TL;DR on teine levinud lahendus infoülekülluse vältimiseks. Selle põhimõte seisneb selles, et pikade tekstidega on käsitsi või automaatselt loodud lühikokkuvõte kaasas, mis edastab teksti sisu vaid mõne lausega [15]. Seda kasutatakse sageli meilikommunikatsioonis, veebilehtedel ja teistes infovoogudes, kus sisu maht on suur ning inimestel puudub aeg või motivatsioon seda täielikult lugeda. TL;DR-lahendused võivad olla nii AI-põhised kui ka käsitsi koostatud. Näiteks ettevõtted ja ülikoolid võivad oma teadete lõppu lisada TL;DR-lõigu, mis võtab kokku olulise teabe, näiteks kuupäevad, tähtajad ja vajalikud tegevused, mis võimaldab tudengitel ja töötajatel kiiremini ning efektiivsemalt vajalikku infot haarata.

3.1.3 Discord

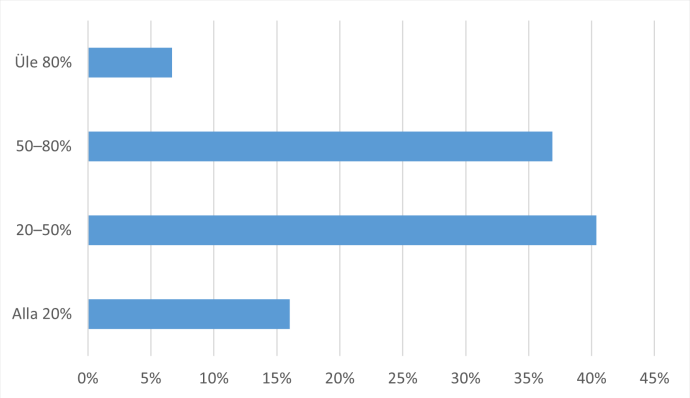
Kolmas alternatiivne lahendus on suhtlusplatvorm Discord, mida kasutatakse aktiivselt näiteks TalTechi informaatika erialal. Discord võimaldab luua erinevaid kanaleid, mille abil saab info temaatiliselt struktureerida ja hoida olulist teavet lihtsamini leitavana [16]. Näiteks informaatika eriala Discord-serveris on loodud kanalid erinevate teemade jaoks, nagu "praktika", "lõputöö", "üldinfo", ning see võimaldab tudengitel valida kanalid, mis neid otseselt puudutavad või huvitavad. Selline kanalite struktureerimine vähendab info üleküllust, sest tudengid saavad valida ja jälgida ainult neile vajalikke kanaleid, vältides liigse ja ebaolulise info hulka. Lisaks võimaldab Discord mugavalt teateid visuaalselt esile tõsta, samuti toetab platvorm kiiret tagasisidet ja otsesuhtlust õppejõudude ja tudengite vahel.

3.2 Küsitluste tulemused

Alljärgnevalt käsitletakse 06.–13. aprillil 2025 toimunud veebiküsitluste tulemusi. Kaks eraldi Google Formsi küsimustikku, üks üliõpilastele ja teine programmijuhtidele, saadeti ülikooli ametliku Outlooki meili kaudu sobivatele adressaatidele. Vastamine oli anonüümne ja võttis aega umbes viis minutit. Kokku oli 374 tudengi ja 20 programmijuhi vastust, mille põhjal koostatakse järgnevad kvantitatiivsed ülevaated ja järeldused.

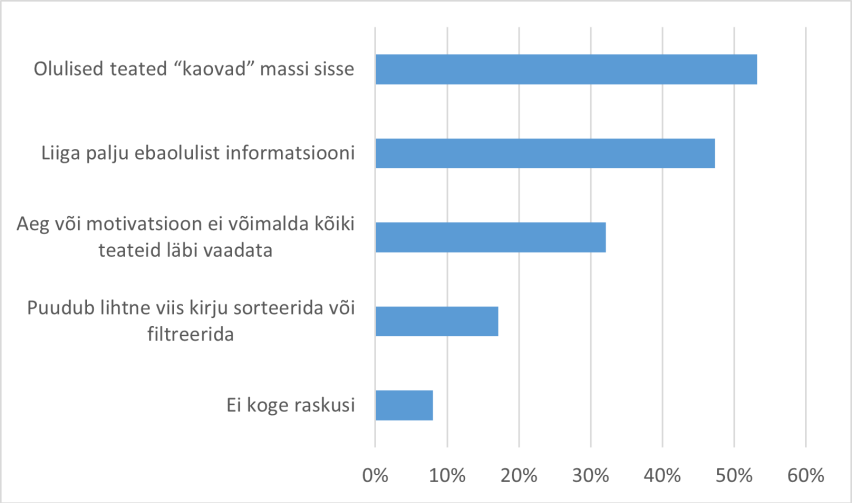
3.2.1 Tudengiküsitluse tulemused

Tudengiküsitluse tulemused kinnitavad, et ülikooli ametlikku meili kasutatakse igapäevaselt – 26,7% vastajatest avab oma postkasti mitu korda päevas ja 37,4% kord päevas. Vaatamata sellele peavad paljud tudengid enamikku neile saabuvaid sõnumeid ebaoluliseks: umbes 40% peab vajalikuks vaid 20–50% sõnumitest ja ligi 16% usub, et vähem kui 20% infost on tõepoolest kasulik (Joonis 3).



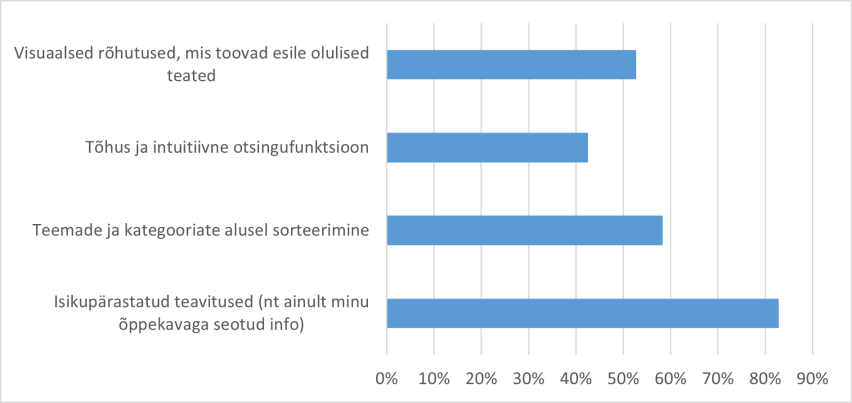
Joonis 3. Kasuliku info osakaal tudengite hinnangul.

Selle tulemusena väheneb tudengite motivatsioon kirju lugeda: vaid viiendik õpilastest tunnistab suurt huvi, samas kui üle poole loeb kirju regulaarselt, kuid kaotab infomüras tähelepanu ning iga neljas üliõpilane hindab oma motivatsiooni e-posti lugeda madalaks. Kõige sagedamini esinevad probleemid on liiga suur hulk ebaolulisi sõnumeid, oluliste teadete "ära kadumine" nende vahele ning sorteerimis- ja filtreerimisvõimaluste puudumine – neid kogeb enam kui viis tudengit kümnest (Joonis 4). Tulemuseks on see, et peaaegu 28% tudengitest on vähemalt korra jäänud eksami-, praktikakonkursi või stipendiumiteatega hiljaks.



Joonis 4. Peamised raskused ametlike e-kirjade lugemisel ja jälgimisel.

Samas näitavad vastused tugevat valmisolekut uut isikupärastatud teavitusplatvormi kasutada: 45,7% õpilastest oleksid nõus seda kasutusele võtta, kui lahendus oleks intuitiivne, ligi 11% kasutaks seda kohe ja 36,4% vastanutest vajaks rohkem teavet. Enim nõutud funktsioonid (Joonis 5) on isikupärastatud teavitused (83% vastajatest), kategooriapõhine sortimine (58%) ja visuaalne esiletõstmine (52%).



Joonis 5. Kasutajate ootused digitaalse teavitustahvli funktsioonidele.

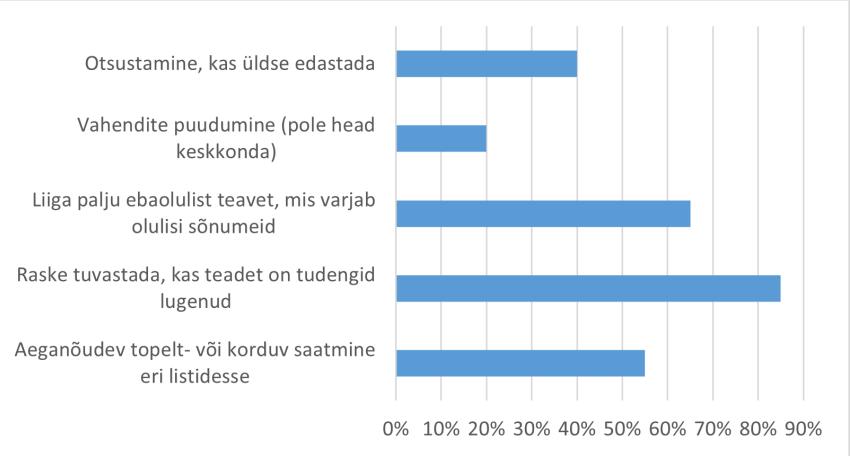
Kokkuvõttes kinnitavad andmed, et praegune süsteem tekitab infoüleküllust, vähendab õpilaste motivatsiooni ja raskendab olulise info kättesaadavust, samas kui personaliseeritud,

filtreeritav teadetetahvel võiks vastata õpilaste selgelt väljendatud vajadustele.

3.2.2 Programmijuhtide tulemused

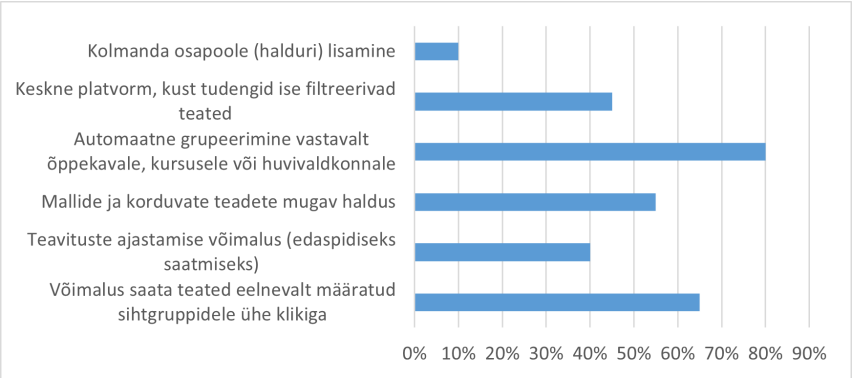
Kuigi 65% programmijuhtidest kulutab teadete muutmisele ja edastamisele alla ühe tunni nädalas, on enamik vastanutest rahulolematud praeguse teavitussüsteemiga. Ainult 15% peab seda väga tõhusaks, samas kui ülejäänud hindavad seda parimal juhul piisavaks, kuid vajaks täiustamist, või üldse ebaefektiivseks.

Üheks keskseks probleemiks peetakse asjaolu, et üliõpilased ei pruugi olulist teavet õigel ajal kätte saada – 90% programmijuhtidest kinnitab, et tähtsad teated jäävad tihti või aeg-ajalt märkamata. Lisaks toob 65% vastanutest välja, et liigse ebaolulise info hulk hajutab tudengite tähelepanu ning raskendab olulise info märkamist. Veelgi enam, 85% programmijuhtidest leiab, et süsteemis puudub tagasiside – ei ole võimalik kindlaks teha, kas tudengid on sõnumi avanud või sellega tutvunud. Samuti on märkimisväärseks probleemiks käsitsi dubleerimine – samu sõnumeid tuleb korduvalt kopeerida ja saata erinevatesse postitusnimekirjadesse või keskkondadesse, mis muudab töö ajamahukaks ja killustatuks (Joonis 6).



Joonis 6. Programmijuhtide peamised väljakutsed ametlike teadete edastamisel.

Uue lahenduse kasutuselevõtu osas väljendavad programmijuhid üldjoontes positiivset hoiakut: 5% oleksid valmis seda kohe kasutama, 45% oleksid nõus seda kasutama juhul, kui platvorm on kasutajasõbralik ja intuitiivne, ning 35% sooviksid enne otsustamist tutvuda süsteemi demoversiooniga. Vaid 15% vastanutest suhtus ideesse skeptiliselt. Kõige olulisemaks peetavateks funktsionaalsusteks (Joonis 7) tõustakse esile võimalus määrata teavituste sihtrühm ühe klõpsuga (65% vastanutest), mallide ja korduvate teadete mugav haldamine (55%) ning automaatne grupeerimine vastavalt õppekavale või kursusele (80%).



Joonis 7. Programmijuhtide jaoks olulisemad funktsioonid tõhusaks teavituste edastamiseks.

Küsitluse tulemused viitavad selgelt sellele, et programmijuhid peavad igapäevast teavitustööd ajamahukaks ja mitte alati piisavalt tõhusaks, mistõttu nähakse suure väärtusena lahendust, mis aitaks vähendada käsitsi tehtavat korduvat tööd ning suunata info paremini õigete adressaatideni.

3.2.3 Küsitluste analüüs ja järeldused

Läbiviidud küsitluste põhjal võib järeldada, et infoüleküllus on õpilaste ja programmijuhtide jaoks reaalne ja igapäevane probleem. Tudengite puhul ilmneb, et kuigi nad kontrollivad regulaarselt oma ametlikku e-posti, vähendab nende motivatsiooni e-kirju lugeda ebaolulise teabe suur hulk. Kui iga teine üliõpilane leiab, et vähem kui pool talle saadetud teadetest pole kasulik, siis pole üllatav, et ligi veerand neist on vähemalt korra jäänud tähtsa teavituse (nt eksami või stipendiumi kohta) märkamisega hiljaks. See kinnitab, et praegune süsteem ei toeta piisavalt teabe sihipärast edastamist ega paku vajalikke vahendeid sõnumite olulisuse eristamiseks.

Kui rääkida programmijuhtidest, siis suur osa neist leiab, et praegune töövoog on ebaefektiivne: infot tuleb dubleerida, tööd tehakse käsitsi ning pole selgust, kas saadetud sõnumid on õpilasteni jõudnud. Samuti peetakse probleemiks infomüra, mis raskendab olulise teabe eristamist ja sellele tähelepanu pööramist. Märkimisväärne on asjaolu, et enamik programmijuhte on valmis kaaluma uue teavitussüsteemi kasutuselevõttu, eeldusel et see on lihtne ja loogiline kasutada. Eelistatud funktsioonide seas nähakse eelkõige vajadust personaliseerimise ja automatiseerimise järele – näiteks võimalus määrata sihtrühmi, lisada teavitustele kategooriaid ning esile tõsta olulist infot visuaalselt.

Tulemused näitavad üheselt, et olemasolev infosüsteem vajab uuendamist, et vähendada infomüra, tõsta olulise info nähtavust ning vähendada manuaalset töökoormust teavituste

edastamisel. Isikupärastatud ja filtreeritav teavitustahvel, mida kasutajad saavad vastavalt oma rollile ja vajadustele kohandada, pakub realistlikku lahendust mõlema sihtrühma poolt esile tõstetud probleemidele.

3.3 Olemasolev AS-IS protsess

Praegu edastatakse ametlikke teateid tudengitele mitme erineva kanali kaudu, kuid kõik sõnumid jõuavad lõpuks üliõpilaste ametlikku Outlooki postkasti. Teateid saadavad programmijuhid, õppejõud või õppekorraldusega seotud töötajad peamiselt kolme kanali kaudu: Outlook, Moodle või ÕIS (õppeinfosüsteem).

Outlookis saadetakse sõnumeid konkreetsetesse meililistidesse, mis võivad olla seotud näiteks kursuste, õppekavade või õppetasemetega. Moodle'i ja ÕISi puhul lisatakse teated vastavalt kursuse teadetetahvlile või saadetakse teavitused otse süsteemi kaudu kindlale sihtrühmale. Mõlemal juhul liigub sõnum lõpuks automaatselt tudengi Outlooki postkasti.

Selline lahendus koondab kõikvõimalikud teated – alates kursuseinfo ja eksamimuudatuste teadetest kuni praktikate ja stipendiumite kuulutusteni – ühte kohta, ilma võimaluseta neid filtreerida või isikupärastada. Tudengi vaates tähendab see, et olulised sõnumid võivad jääda vähem tähtsate teadete varju, kuna süsteem ei toeta sisupõhist sorteerimist ega eelistuste seadmist.

Sõnumi saatmise protsess on iseenesest sirgjooneline: saatja valib sobiva keskkonna (Outlook, Moodle või ÕIS), määrab adressaadid, koostab sõnumi ja kinnitab selle saatmise. Kui kasutatakse Moodle'it või ÕISi, toimetab süsteem sõnumi automaatselt Outlooki, ilma vajaduseta seda käsitsi edasi saata. Tavaliselt valitakse saatmiseks ainult üks kanal korraga.

Kuigi olemasolev süsteem on formaalselt toimiv, on selle tõhusus ja paindlikkus piiratud. Peamisteks kitsaskohtadeks on teavituste isikupärastamise ja filtreerimise puudumine, tagasisidevõimaluste nappus ning infoüleküllus ühes kanalis. Need probleemid loovad olukorra, kus oluline info võib jääda märkamata ja suhtlus muutub ühepoolseks.

Seetõttu on selge vajadus uue teavitussüsteemi järele, mis võimaldaks sõnumeid sihipäraselt hallata, tudengitel teateid sisuliselt filtreerida ning programmijuhtidel saada ülevaadet sõnumite tegelikust jõudmisest ja mõjust. Selline süsteem aitaks parandada info kättesaadavust, vähendada infomüra ning toetada tõhusamat suhtlust õppetegevuse sees.

3.4 Funktsionaalsed nõuded

Uue teavitussüsteemi funktsionaalsed nõuded põhinevad tudengite ja programmijuhtide küsitluste tulemustel ning kajastavad kasutajate praktilisi ootusi süsteemi toimimisele. Eesmärk on parandada olulise info nähtavust, vähendada infomüra ning lihtsustada teavituste koostamise ja jälgimise protsessi. Süsteem peab võimaldama järgmisi funktsionaalsusi:

- Iga tudeng saab määrata, milliseid teemasid ja kategooriaid ta soovib näha, et vähendada ebaolulise info hulka ning parandada olulise teabe kättesaadavust.
- Teated peavad olema automaatselt jaotatud kategooriate ja teemade kaupa ning tähtsad sõnumid (nt eksamite ja stipendiumide tähtajad) peavad olema visuaalselt esile tõstetud.
- Programmijuht saab ühe klõpsuga valida teavituse sihtrühmaks näiteks konkreetse õppekava, kursuse või õppeaasta tudengid.
- Programmijuht peab saama määrata iga teavituse kategooria (nt praktika, eksamid, üritused) ning hinnata sõnumi olulisust prioriteedi kaudu.
- Süsteem peab pakkuma võimalust salvestada teavituste malle, mida saab hiljem korduvate teadete loomiseks kiiresti ja mugavalt kasutada.
- Programmijuhil peab olema võimalus näha, kas tudengid on saadetud teateid avanud, et hinnata sõnumite tõhusust ja vajadusel kohandada edastamisstrateegiat.

3.5 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded määratlevad, kuidas süsteem peab oma funktsioone täitma – need on seotud kasutusmugavuse, töökindluse, laiendatavuse ja tehnilise sobivusega. Kuigi funktsionaalsed nõuded keskenduvad sellele, mida süsteem teeb, siis mittefunktsionaalsed nõuded loovad eeldused selleks, et kasutajakogemus oleks kvaliteetne, süsteemi arendus oleks jätkusuutlik ning lahendus toimiks sujuvalt ka suurema koormuse ja erinevates tehnilistes keskkondades. Alljärgnevalt on välja toodud peamised mittefunktsionaalsed nõuded kavandatavale teavitussüsteemile:

- Süsteem peab olema kasutajasõbralik ja intuitiivne – teavituste koostamine, lugemine ning navigeerimine kategooriate ja filtrite vahel peab olema lihtne ja loogiline nii tudengitele kui programmijuhtidele.
- Kasutajaliides peab olema puhas ja visuaalselt selge, toetades kiiret info leidmist ning vähendades kasutajate koormust.
- Süsteem peab suutma töödelda suurt hulka kasutajaid ja teavitusi ilma märgatavate

viivitusteta ning tagama teadete kiire ja usaldusväärse kohaletoimetamise.
- Lahendus peab vältima info kadumist ja tarbetuid viivitusi, mis võivad mõjutada olulise info edastamist akadeemilistes olukordades.
- Arhitektuur peab olema paindlik ja laiendatav, võimaldades lisada uusi funktsionaalsusi vastavalt kasutajate vajadustele ja tehnoloogilisele arengule, ning toetama integreeritavust ülikooli olemasolevate infosüsteemidega.

3.6 Kasutajalood

Selleks et paremini mõista erinevate kasutajate vajadusi ja ootusi uue teavitustahvli suhtes, on kirjeldatud tüüpilisi kasutusstsenaariume ehk kasutajalugusid. Kasutajalood kirjeldavad konkreetseid olukordi ja ootusi, mida süsteem peaks toetama, lähtudes küsitluste tulemustest ja analüüsist. Need lood aitavad defineerida, millised funktsionaalsused ja töövood on süsteemi arendamisel prioriteetsed.

Üliõpilase vaatest on oluline, et ta saaks kiirelt ja lihtsalt filtreerida oma huvide ja õppekava põhjal just talle olulised teated. Näiteks:

- Tudengina soovin kuvada ainult oma õppekavaga seotud teavitusi, et vältida liigset infomüra ja keskenduda õppimise jaoks olulisele infole.
- Tudengina soovin, et sõnumid oleksid jagatud kategooriatesse, et ma leiaksin kiiresti üles just selle, mis mind huvitab.
- Tudengina soovin, et tähtsad sõnumid oleksid visuaalselt esile tõstetud, et ma kohe märkaksin neid.
- Tudengina soovin ise määrata, millist tüüpi teavitusi ma soovin näha ja millist mitte, et kohandada oma postkasti vastavalt isiklikele vajadustele.

Programmijuhtide vajadused keskenduvad eelkõige teavituste koostamise ja haldamise lihtsustamisele. Näiteks:

- Programmijuhina soovin määrata sõnumite sihtrühma ühe klikiga, et vältida käsitsi aadresside valimist ja kiirendada teadete saatmist.
- Programmijuhina soovin luua ja kasutada sõnumimalle korduvate teadete saatmiseks, et vähendada igapäevast ajakulu.
- Programmijuhina soovin iga sõnumi juures määrata selle prioriteedi ja kategooria, et tudengid saaksid kiiresti aru teavituse olulisusest.
- Programmijuhina soovin jälgida, kes tudengitest on saadetud teavitused avanud ja lugenud, et vajadusel suunata info uuesti.

4 Tulemus

Käesolevas peatükis käsitletakse bakalaureusetöö praktilist tulemust ehk digitaalse teavitustahvli prototüüpi, mis töötati välja selleks, et lahendada eelnevates peatükkides tuvastatud probleemid, eelkõige vähendada infoülekküllust ning parandada info jõudmist üliõpilasteni. On oluline rõhutada, et tegemist on esimese prototüübiga, mitte lõpliku ja täielikult tootmisvalmis lahendusega. Peatükk algab TO-BE protsessimudeli kirjeldusega, selgitades, kuidas kavandatav lahendus võiks muuta olemasolevat protsessi kasutajasõbralikumaks ja tõhusamaks. Edasi kirjeldatakse süsteemi arhitektuuri, sealhulgas backend-, frontend- ja andmebaasikihte, samuti peamisi kasutajaliidese vaateid. Lisaks antakse ülevaade testimisest, kasutatud lähenemisviisidest ja katvuse tasemest, et hinnata rakenduse töökindlust ja kvaliteeti.

4.1 TO-BE protsessimudel

Kavandatud protsessis alustab programmijuht teavituse edastamist sisenedes uude süsteemi, kus avaneb spetsiaalselt selleks loodud vorm teate koostamiseks. Programmijuht alustab teavituse loomist, sisestades kõigepealt teate pealkirja ja sisu. Seejärel määrab ta teavitusele sobiva kategooria (näiteks eksam, praktika või stipendium) ning lisab prioriteedi, mis võimaldab tudengitel olulisemaid sõnumeid lihtsamini märgata ja eristada. Suureks kasutusmugavuseks on võimalus valida sõnumite saatmisel lihtsalt ja mugavalt mitu sihtrühma korraga, vältides niiviisi korduvat ja ajamahukat käsitsitööd sõnumite saatmisel. Lisaks võimaldab süsteem luua ja salvestada teavituste malle, mis lihtsustab ja kiirendab korduvate, standardsete teadete (näiteks regulaarsete eksamikuulutuste või tähtajaliste meeldetuletuste) edastamist.

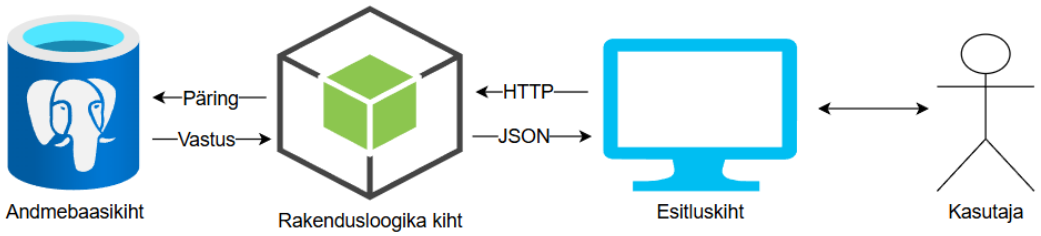
Pärast saatmist jõuab teavitus automaatselt kõigi valitud sihtrühmade tudengiteni, kes näevad seda oma isikupärastatud teavitustahvlil. Üliõpilane saab teateid filtreerida kategooria ja prioriteedi alusel ning kasutada otsingufunktsiooni konkreetsete sõnumite leidmiseks. Süsteem eristab visuaalselt ka avamata teated ning pakub eraldi vaadet ainult lugemata teadete kuvamiseks, mis võimaldab kiiret ligipääsu uuele infole. Kriitilise tähtsusega teated, nagu eksami- või stipendiumitähtajad, on selgelt esile tõstetud, et vältida nende märkamata jäämist. Samuti saab tudeng lisada olulisi teateid lemmikute hulka, et neile vajadusel hõlpsasti uuesti ligi pääseda.

Programmijuhtidele annab süsteem pärast sõnumi saatmist võimaluse jälgida, kui paljud

tudengitest on teate avanud ja läbi lugenud, pakkudes seega väärtuslikku tagasisidet saadetud sõnumite efektiivsuse kohta. See info võimaldab programmijuhtidel vajadusel teha kohandusi suhtlusprotsessis, hinnata infovahetuse tulemuslikkust ning seeläbi suurendada üldist teavitustöö tõhusust.

4.2 Arhitektuur

Kavandatud teavitussüsteem põhineb klassikalisel kolmekihilisel arhitektuuril (Joonis 8), kus süsteemi funktsionaalsus on jaotatud esitluskihiks (frontend), rakendusloogika kihiks (backend) ja andmekihiks (andmebaas). Iga kiht täidab kindlalt määratletud rolli ning suhtleb teiste kihtidega standardiseeritud liideste kaudu. Selline lähenemine tagab süsteemi modulaarse ülesehituse, lihtsustab arendus- ja testimisprotsesse ning võimaldab paindlikult lisada tulevikus uusi funktsionaalsusi või integreerida täiendavaid teenuseid.



Joonis 8. Rakenduse arhitektuur.

4.2.1 Esitluskiht

Esitluskiht on realiseeritud React19 abil, mis võimaldab luua ühe leheküljе rakenduse, kus kogu kasutajaliides on brauseris ja vaadete vahetamine toimub ilma lehe täieliku uuesti laadimiseta. Reacti komponendipõhine arhitektuur võimaldab jagada liidese väikesteks korduvkasutatavateks osadeks, nagu näiteks teadete nimekiri, üksiksõnumi detailvaade või sõnumi loomise vorm, kusjuures iga komponent vastutab oma kitsalt määratletud visuaalse ja funktsionaalse rolli eest.

Andmete otsimine ja salvestamine toimub Redux Toolkiti RTKQuery abil, mis saadab taustal API päringuid, salvestab vastused brauseri vahemällu, jälgib laadimis- ja veaolekuid ning värskendab automaatselt kõiki komponente andmete muutumisel. See tähendab, et pärast sõnumi loomist, muutmist või kustutamist näeb kasutaja kohe viimast olekut.

ReactRouter7 hoolitseb rakenduse sees navigeerimise eest, vastendades URL-id (*Uniform*

Resource Locator) sobivatele komponentidele, nii et näiteks aadress */notifications* kuvab üldvaate kõigist teadetest, */notifications/123* avab konkreetse sõnumi detailid ning */add-notification* annab programmijuhile võimaluse uut teadet sisestada – kõik see toimub sujuvalt üheainsa veebilehe piires.

4.2.2 Rakendusloogika kiht

Rakendusloogika kiht (backend) on loodud Node.js 18 platvormil, kasutades Express 5 raamistikku, mis sobib suurepäraselt REST (*Representational State Transfer*) põhiste API-de arendamiseks. Selle ülesanne on töödelda kasutajaliidesest tulevaid HTTP (*HyperText Transfer Protocol*) päringuid, kontrollida kasutajate autentimist ja autoriseerimist, rakendada äriloogikat ning suhelda PostgreSQL-andmebaasiga, kasutades pg-pool moodulit.

Arhitektuur järgib kihilist ülesehitust, kus päring liigub läbi marsruutide kontrolleritesse, mis haldavad äriloogikat ja teostavad vajalikke andmebaasipäringuid. Näiteks suunab lõpp-punkt */api/notifications* päringu *getAllNotifications* kontrollerisse, kus kontrollitakse kasutaja õigusi ja tagastatakse talle suunatud teated.

Kontrollerid teostavad sisendi valideerimise, rakendavad loogika ja koostavad SQL (*Structured Query Language*) päringud või kutsuvad andmebaasifunktsioone. Vastused tagastatakse JSON (*JavaScript Object Notation*) formaadis.

Autentimine toimub JWT (*JSON Web Token*) abil: pärast sisselogimist genereeritakse kasutajale token, mis säilitatakse kliendipoolel ning lisatakse edaspidistele päringutele päisesse, võimaldades kasutaja usaldusväärset tuvastamist ja volitamist. Turvalisust ja rollipõhist ligipääsu tagavad marsruutide tasemel lisatud vahefunktsioonid, mis kontrollivad JWT-tokeni olemasolu ja kehtivust ning kasutaja rolli.

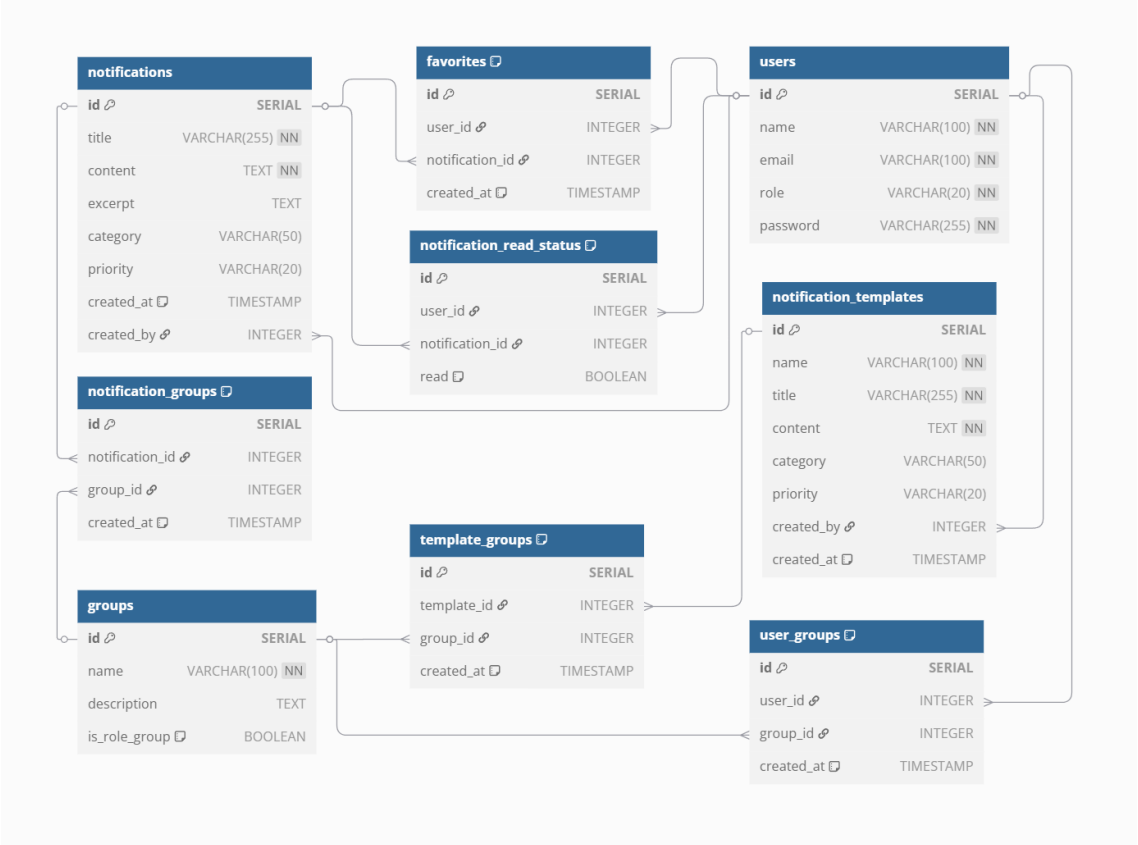
Rakendus kasutab modulaarset arhitektuuri, kus iga äriloogika osa (näiteks teadete, mallide või lemmikute toimingud) on eraldatud iseseisvateks mooduliteks. See lähenemine parandab süsteemi loetavust ja hooldatavust ning võimaldab funktsionaalsust vajadusel hõlpsalt laiendada või muuta, ilma et peaks kogu rakenduse struktuuri ümber töötama.

Kõik API-päringud on varustatud Swaggeri põhise dokumentatsiooniga [17], mis kirjeldab süsteemi lõpp-punkte, sisendparameetreid ja vastuseid. Dokumentatsiooni abil on võimalik selgelt mõista, kuidas teavituste süsteemiga suhelda ning millist infot üks või teine päring tagastab. See toetab süsteemi paremat kasutatavust ja tulevasi liidestusvõimalusi.

4.2.3 Andmebaasikiht

Süsteemi andmed salvestatakse PostgreSQL-i, mille normaliseeritud skeem toetab kõiki teavitustahvli põhifunktsioone ning tagab nii andmete järjepidevuse kui ka skaleeritavuse. Skeemi aluseks on *users* tabel, mis sisaldab andmeid autentimise ja rollide kohta. Kasutajad on paindlikult seotud erinevate *groups* kirjetega, mis võivad tähistada konkreetseid õppekavasid, kursusi või muud sihtrühma loogikat. Teatiste sisu ja metaandmed salvestatakse *notifications* tabelisse, korduvkasutatavad sõnumimallid aga eraldi tabelisse *notification_templates*. Kõik kolm põhientiteeti on üksteisega seotud mitu-mitmele suhete kaudu: *user_groups* seob õpilased ja programmijuhid nende vastavate rühmadega, *notification_groups* määrab, millistele sihtrühmadele konkreetne teatis saadetakse, ja *template_groups* võimaldab malli kohe soovitud rühmadega seostada. Kasutajate interaktsioonid teavitustega kajastatakse kahes täiendavas tabelis: *notification_read_status* olek salvestab, millal ja kas konkreetne kasutaja teatise avas, ning *favorites* võimaldab märkida teated isiklikult oluliseks.

Allolev ER (*Entity–relationship*) diagramm (Joonis 9) visualiseerib tabelite struktuuri ja nendevahelisi seoseid, näidates selgelt, kuidas süsteem toetab isikupärastatud teatisi ja rollipõhist juurdepääsu.



Joonis 9. Olemi-suhte diagramm.

4.2.4 Kihtide seosed

Kogu andmevahetus on ehitatud selgele kliendi-serveri paradigmale, kus React-põhine kasutajaliides täidab esitlusloogika kihi rolli ning kogu äriloogika ja andmete püsivus asuvad serveri ja andmebaasi kihtides. Brauseris tehtud toiming, näiteks uue teate loomine, olemasoleva teate muutmine või lemmikuks märkimine, vallandab RTK Query kaudu HTTP-päringu Node/Express-teenusele. Päring kannab endas JSON-keha ja kui ressurss on kaitstud, siis autoriseerimise päist koos JWT-märgiga. See võimaldab serveril üheselt tuvastada kasutaja identiteedi ja rolli ning rakendada rollipõhist juurdepääsukontrolli.

Serveripoolne marsruut (route) suunab taotluse temaatilisse kontrollerisse. Kontroller kontrollib sisendi formaadi ja õigused, rakendab ärireeglid – näiteks lisab uuele teatele kategooria, prioriteedi ja sihtrühmad ning edastab korralduse PostgreSQL-ile. Andmebaasipäringud teostatakse parameetritega pg-Pooli ühendusbasseini kaudu. Vajaduse korral kasutatakse ühte transaktsiooni, et siduda ühe operatsiooniga teade, sihtrühmad ja lugemisstaatuse kirjed, vältides osalisi salvestusi.

Pärast edukat salvestamist pakib server tulemuse ühtlaseks JSON-objektiks koos metaandmetega (ajatempel, serveri olekukood) ja tagastab selle kliendile. RTK Query peatab vastuse, sünkroniseerib lokaalse vahemälu ja uuendab seotud cache-sildid, mis ajakohastab kõik asjakohased komponendid – näiteks teavituste nimekirja või detailvaate. Tänu sellele pole vajalik dubleerivaid GET-päringuid teha ja kasutaja näeb muudatusi peaaegu reaalajas.

Sama andmevoo loogikat kasutavad ka teised funktsioonid: lugemisstaatuse märkimine käivitab väikese POST-päringu, mis lisab või uuendab kirjet *notification_read_status* tabelis, lemmiku lisamine käib *favorites* tabeli kaudu samal põhimõttel. Veahaldus on samuti kihipõhine: kui kontroller tuvastab sisendivea või õiguse puudumise, saadab ta struktureeritud veateate koos sobiva HTTP-koodiga. RTK Query võtab selle üle ja kuvab kasutajale ühtse veasõnumi, säilitades seeläbi konsistentse kasutuskogemuse. Niisugune kihtide omavaheline selge tööjaotus ja vormindatud andmevahetus võimaldavad süsteemil jääda nii tehniliselt turvaliseks kui ka kasutajale sujuvaks ja reageerivaks.

4.3 Vaated

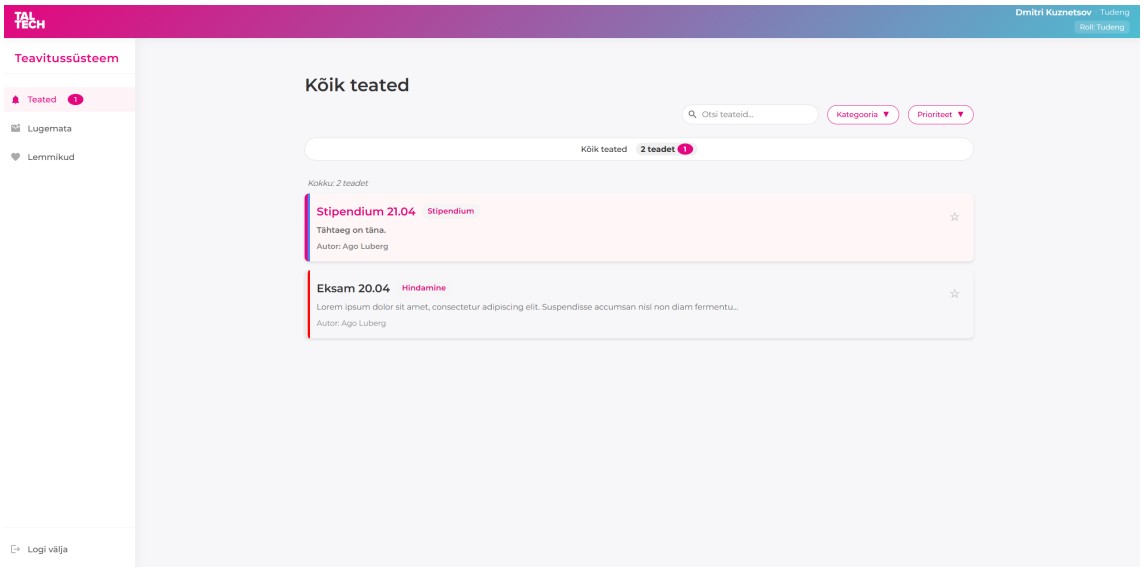
Loodud prototüübis on realiseeritud mitmed olulised kasutajaliidese vaated, mis toetavad süsteemi peamisi kasutusstsenaariume ja pakuvad erinevatele kasutajarollidele (tudeng ja programmijuht) selgeid töövooge. Vaated on üles ehitatud komponentidena, mis tagavad

visuaalse järjepidevuse ja aitavad lihtsustada kasutuskogemust. Disaini loomisel lähtuti TalTechi ametliku stiiliraamatu värvitoonidest [18], mis tagab süsteemi ühtse ja ülikooli visuaalse identiteediga kooskõlas oleva välimuse.

Kõik rakenduse vaated on üles ehitatud *DashboardLayout.jsx* komponendi ümber, mis toimib navigeerimispaneelina. Kasutaja saab liikuda erinevate vaadete vahel, nagu „Teated“ (peavaade), „Lugemata“ ja „Lemmikud“. Kui kasutaja on programmijuht, lisanduvad navigeerimisse ka „Minu teated“, kus kuvatakse tema enda koostatud sõnumid koos võimalusega neid muuta või kustutada, samuti „Lisa uus teade“ ja „Mallid“, kus hallatakse korduvkasutatavaid malliformaate. Ülemises paremas nurgas kuvatakse kasutaja nimi ja roll.

Peavaade „Teated“ on sisuliselt sarnane nii tudengitele kui ka programmijuhtidele – siin kuvatakse kõik kasutajaga seotud teated. Vaade sisaldab filtreerimisvõimalusi kategooria ja prioriteedi alusel, samuti otsinguvälja, mis võimaldab kiiresti leida vajalikke teateid märksõnade põhjal (Joonis 10).

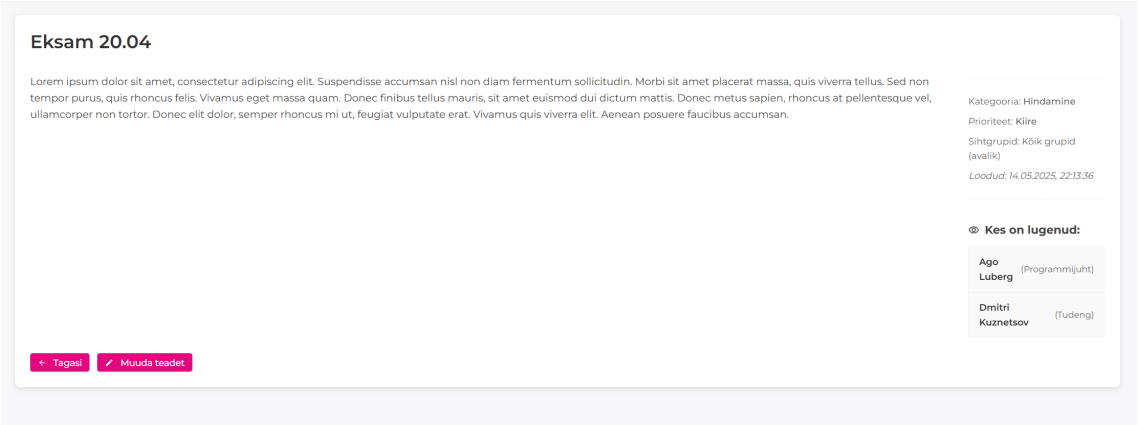
Iga teate juures on näha selle kategooria, prioriteet (märgitud värviga vasakul servas) ning lugemisolek – lugemata teated on visuaalselt eristatud. Paremas servas asub tärn, millega saab teate lisada lemmikute hulka. „Lugemata“ (vt Lisa 2) ja „Lemmikud“ (vt Lisa 3) vaated ei ole iseseisvad vaated, vaid rakendavad globaalset filtrit sellele samale põhivaatele.



Joonis 10. „Teated“ vaade.

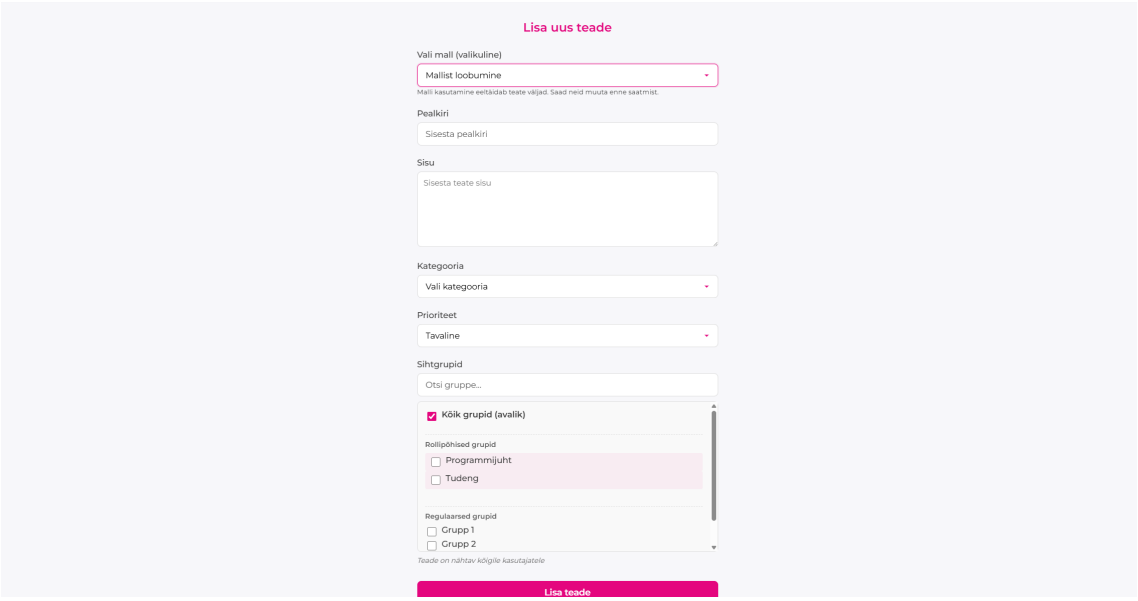
Kui kasutaja vajutab mõnele teatele, avaneb detailvaade, kus kuvatakse pealkiri ja sisu ning paremal servas kategooria, prioriteet ja loomise kuupäev (vt Lisa 4). Kui kasutaja on programmijuht, kuvatakse lisaks ka sihtrühmad, kellele teade saadeti, ja lugemisstatistika, kus on näha, millised tudengid on teadet avanud (Joonis 11). See annab programmijuhile

võimaluse hinnata teavituse tegelikku ulatust ja vajadusel planeerida kordusteavitusi või sisu täiendamist.



Joonis 11. Teate detailvaade programmijuhi vaates.

Vaates „Lisa uus teade“ (Joonis 12) saab programmijuht koostada uue sõnumi, sisestades pealkirja, sisu, kategooria, prioriteedi ning valides sobivad sihtrühmad. Täiendavalt saab kasutaja valida eelnevalt loodud malli, mille sisu ja seadistused kantakse automaatselt väljadele, lihtsustades korduvate teadete koostamist.

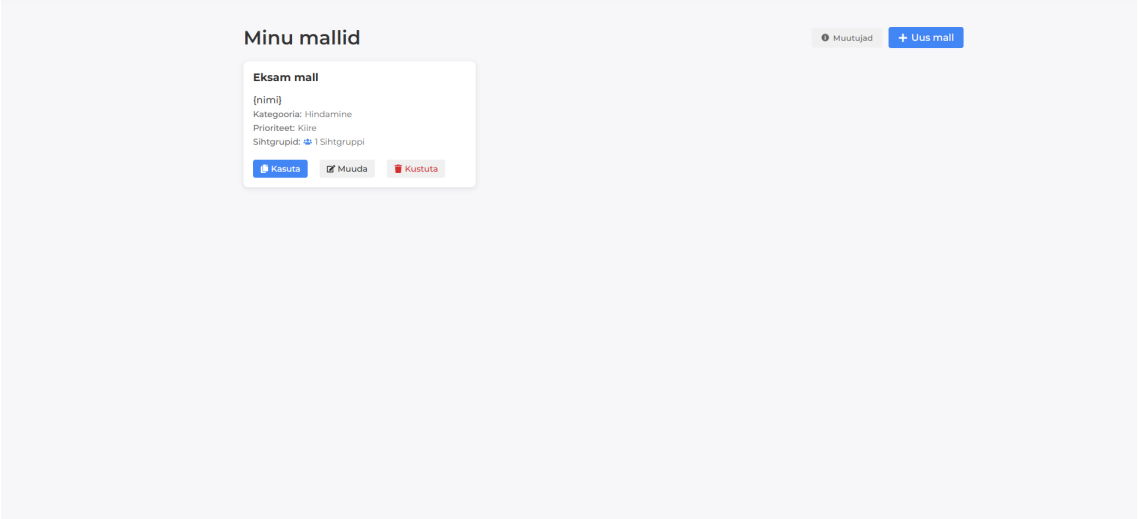


Joonis 12. „Lisa uus teade“ vaade.

Vaade „Mallid“ võimaldab programmijuhil hallata kõiki enda loodud teavitusmalle (Joonis 13). Iga malli saab vaadata, muuta, kustutada või kasutada uue sõnumi koostamisel. Uue malli loomisel sisestatakse selle nimi, pealkiri, sisu, kategooria, prioriteet jms (vt Lisa 5). Mallide sisu saab muuta dünaamiliseks, kasutades muutujaid. Selleks tuleb sõna või fraas kirjutada loogeliste sulgude sisse, näiteks *kuupäev*.

Kui malliga luuakse uus teade, avatakse dialoog, kus küsitakse väärtused kõigile mallis

määratud muutujatele. Kõik sama nimega väljad asendatakse sama väärtusega. Mallide vaates on olemas ka juhend, mis selgitab muutujate kasutamist samm-sammult (vt Lisa 5).



Joonis 13. „Mallid“ vaade.

Lisaks sisulistele vaadetele sisaldab süsteem ka „Sisselogimine“ ja „Registreerimine“ vaateid, mille kaudu saavad kasutajad platvormile ligi (vt Lisa 6). Registreerimisvaade võimaldab luua kasutajakonto, sisestades vajalikud andmed, nagu nimi, unikaalne e-posti aadress, parool, kasutaja roll (tudeng või programmijuht) ning kuuluvus kindlasse gruppi. Kuna prototüüp on suunatud eelkõige testimiseks ja arenduseks, on registreerimisprotsess loodud lihtsustatud kujul, et võimaldada eri kasutajarollide katsetamist ja funktsionaalsuse kontrolli.

Tulevikus on eesmärk asendada see lahendus autentimisega ülikooli keskse sisselogimissüsteemi kaudu (uni-ID), et pakkuda sujuvat, turvalist ja ühtset ligipääsu kõigile kasutajatele. Sisselogimine toimub prototüübis e-posti ja parooli alusel ning vastab levinud turvapõhimõtetele, tagades, et süsteemi kasutavad ainult volitatud isikud.

4.4 Testimine

Rakenduse töökindlust kontrolliti kahes võtmesuunas: automatiseeritud funktsionaalsete testide ning seeria visuaalsete ja käsitsi kontrollide kaudu, et veenduda nii loogika kui ka kasutajaliidese korrektses toimimises.

4.4.1 Automatiseeritud funktsionaalsed testid

Node.js 18 + Express 5 taustateenusele kirjutati Jest-i ja Supertesti abil üksus- ning integratsioonitestid. Supertest laadib rakenduse mällu ning saadab REST-päringuid, kontrollides,

et tagastatavad HTTP-koodid ja JSON-struktuurid vastaksid ootustele. Rollipõhise juurdepääsu kontrollimiseks kasutatakse kolme stsenaariumi: anonüümne kasutaja, tudeng ja programmijuht. PostgreSQL-i päringud täidetakse testrežiimis simuleeritud pg-Pooli ühendusega, mis võimaldab testida päringute loogikat ilma tegelikku andmebaasi muutmata. React-põhist kasutajaliidest testitakse React Testing Library abil, iga komponent renderdatakse virtuaalses DOM-is ning sündmused (klikk, vormi saatmine) peavad käivitama ootuspärased oleku- ja vaatemuutused. RTK Query võrgupäringud asendatakse *MockServiceWorker*-ga, mis tagastab etteantud vastused ja võimaldab testida äärejuhtumeid (nt serveri viga).

Testide katvust hinnatakse automaatselt genereeritud aruande põhjal, mis annab ülevaate rakenduse erinevate komponentide testimise ulatusest. Praeguse seisuga on katvus serveripoolse rakenduse kontrollerite puhul 97,87%, andmebaasikihis 84% ning API marsruutidel 96,89%. See võimaldab usaldusväärselt hinnata, kui hästi on rakenduse äriloogika kaetud automaatsete testidega ning tuvastada võimalikke katmata kohti, mis võivad vajada täiendavat tähelepanu arenduse käigus.

4.4.2 Visuaalne ja manuaalne testimine

Pärast muudatuste tegemist testiti rakendust ka manuaalselt ja visuaalselt. Läbiti peamised kasutajalood – sisselogimine, teate lisamine, filtrite muutmine, lemmikuks märkimine ja väljalogimine, jälgides ühtlasi visuaalse kujunduse järjepidevust. Testimise eesmärk oli veenduda, et kõik funktsioonid töötavad vastavalt kavandatule ning et olemasolev struktuur ei ole muutuste tõttu katki läinud. Erilist tähelepanu pöörati sellele, et rakendus töötaks sujuvalt erinevates seadmetes – kasutajaliides pidi kohanduma ekraanisuuruse järgi ning jääma kasutajasõbralikuks ka mobiilivaates.

5 Analüüs ja järeldused

Käesolevas peatükis analüüsitakse loodud teavitussüsteemi vastavust eelnevalt püstitatud funktsionaalsetele ja mittefunktsionaalsetele nõuetele, võrreldakse olemasolevat ja kavandatud lahendust, hinnatakse töö tulemused ning pakutakse välja võimalikud edasiarenduse suunad.

5.1 Nõuetele vastavus

Lõpliku prototüübi hindamisel kontrolliti selle vastavust eelnevalt määratletud funktsionaalsetele ja mittefunktsionaalsetele nõuetele. Kontrolli eesmärk oli veenduda, et lahendus täidab kõik püstitatud ootused nii süsteemi toimimise, kasutusmugavuse kui ka tehniliste omaduste osas.

5.1.1 Funktsionaalsete nõuete täitmine

Funktsionaalsete nõuete täitmist hinnati lähtuvalt tudengite ja programmijuhtide ootustest. Süsteem pidi võimaldama teavituste isikupärastamist, pakkudes tudengitele võimalust valida huvipakkuvad kategooriad ning filtreerida sisu vastavalt oma vajadustele. Selle eesmärk oli vähendada infomüra ja tuua esile olulised teated, näiteks eksami- või stipendiumiteated. Prototüüp toetab seda kasutusloogikat – liideses on olemas filtreerimis- ja sorteerimisvõimalused ning erineva prioriteediga teated on visuaalselt eristatud, mis lihtsustab olulise info märkamist.

Programmijuhtide jaoks oli oluline, et teavituste edastamine sihtrühmadele toimuks kiiresti ja tõhusalt. Prototüüp võimaldab korraga valida mitu tudengite sihtrühma ühe lihtsa toiminguga, välistades vajaduse sõnumeid käsitsi dubleerida. Teavitustele saab lisada kategooria ja prioriteedi, mis toetab sõnumite struktureerimist ning adressaatide täpset määramist.

Süsteemis on ka mallide haldus – korduvkasutatavaid teavitusi saab salvestada ja vajadusel uuesti rakendada, mis muudab protsessi kiiremaks ja tõhusamaks.

Tagasiside funktsionaalsus võimaldab programmijuhil jälgida, millised tudengid on teavituse avanud. Süsteem logib kõik avamised ja kuvab statistikavaates detailse ülevaate, mis aitab hinnata teavituste tõhusust ja teha vajadusel parendusi sisu või edastamisviisi osas.

Kõigi nimetatud funktsioonide olemasolu ja korrektne toimimine kinnitavad, et prototüüp vastab seatud funktsionaalsetele nõuetele ning toetab nii tudengite kui ka programmijuhtide töövoogu.

5.1.2 Mittefunktsionaalsete nõuete täitmine

Kasutajaliidest testiti põhjalikult nii lauaarvutites kui ka mobiiliseadmete simulaatorites, et hinnata selle selgust, mugavust ja kohanemisvõimet erinevate ekraanisuurustega. Kõik põhifunktsioonid, näiteks teavituse loomine, filtrite muutmine ja lemmikuks märkimine, olid hõlpsasti kasutatavad ühe-kahe kliki kaugusel. Minimalistlik ja visuaalselt puhas kujundus toetab intuitiivset navigeerimist ka väiksematel ekraanidel. Süsteem on loodud kohanduva kasutajaliidesega, mis tagab korrektse toimimise erinevatel ekraanisuurustel.

Jõudluse testimisel ilmnes, et rakendus töötab stabiilselt ja kiiresti – tõrkeid ega aeglustusi ei esinenud. Tänu RTK Query vahemälule ja serveripoolsele pg-Pool ühendushaldusele toimusid päringud sujuvalt, korduvale sisule vastati koheselt lokaalselt, vähendades serverikoormust ja kiirendades laadimist.

Turvalisuse tagavad JWT-põhine autentimine ja rollipõhine autoriseerimine, käsitsi testitud stsenaariumid ei võimaldanud loata ligipääsu. Paroolid salvestatakse turvaliselt bcrypt-räsinguna ja kõik SQL-päringud on parameteriseeritud, vältides süstirünnakuid.

Süsteemi laiendatav arhitektuur tugineb lahtiseotud Reacti komponentidele, Expressi kontrollerimustrile ja normaliseeritud PostgreSQL-skeemile, mis võimaldavad uusi mooduleid lisada ilma olemasolevat äriloogikat muutmata. Moodulipõhine kaustastruktuur lihtsustab süsteemi hooldust ning loob eeldused komponentide eraldamiseks eraldi teenusteks, toetades vajaduse korral mikroteenusepõhist lähenemist.

Kokkuvõttes vastab prototüüp kõigile mittefunktsionaalsetele nõuetele: süsteem on kasutajasõbralik, töökindel ja hästi laiendatav ning tagab sujuva toimimise erinevates seadmetes ja suurema kasutuskoormuse all.

5.2 AS-IS ja TO-BE protsesside võrdlus

Võrreldes praegust (AS-IS) ja kavandatavat (TO-BE) teavituste edastamise protsessi, on võimalik eristada kolme peamist erinevuste valdkonda: töökorraldus, kasutajakogemus ning tagasisidevõimekus.

AS-IS protsessis peab programmijuht koostama iga teate uuesti, isegi juhul, kui tegemist on sisuliselt korduvate teadetega, nagu eksamite või praktikate eelne info. See toob kaasa sisulise dubleerimise ja ajakulu, eriti kui sama teade tuleb sisestada eraldi mitmesse keskkonda. TO-BE lahenduses saab programmijuht luua korduvate teadete jaoks malli, salvestada selle süsteemi ning kasutada seda vajadusel uuesti. See vähendab käsitsitööd, kiirendab teavituste saatmist ja tagab sõnumite vormilise ühtsuse kogu õppeaasta vältel.

Olulisi erinevusi esineb ka kasutajakogemuses. AS-IS lahenduses kuvatakse kõik teated tudengile ilma filtreerimis- või kohandamisvõimaluseta, sõltumata nende sisust või olulisusest. See võib vähendada teadete märgatavust ning raskendada olulise info eristamist üldisest infomürast. TO-BE süsteem võimaldab tudengil isikupärastada teavitustahvlit – teateid saab sorteerida kategooria, prioriteedi, lugemisstaatuse või märksõnade otsingu kaudu. Erineva prioriteediga teadetele rakendatakse visuaalseid rõhutusi ning kasutajal on võimalus märgistada teateid lemmikutena, et neid oleks hiljem lihtsam leida. Lisaks tagab süsteemi responsiivne kujundus võrdselt hea kasutuskogemuse nii lauaarvutis kui ka mobiilseadmes.

Tagasisidevõimekus on TO-BE lahenduses märkimisväärselt paremal tasemel. AS-IS protsess ei võimalda programmijuhil saada teavet selle kohta, kas tudengid on saadetud teadet avanud või sellega tutvunud. Sellest tulenevalt põhinevad järelteavitused oletustel ning puudub mehhanism teavituste tegeliku mõjususe hindamiseks. Kavandatav süsteem pakub reaalajas statistikat teadete avamismäära kohta ning annab programmijuhile võimaluse analüüsida, millised sihtrühmad on info vastu võtnud ja millised mitte. Selle põhjal on võimalik vajaduse korral kohandada sõnumite sisu, ajastust või edastamisviisi, et suurendada nende nähtavust ja mõju. See suurendab suhtluse sihitust ja tõhusust ning toetab andmepõhist juhtimisotsuste tegemist.

Kavandatav teavitussüsteem tähistab olulist sammu edasi võrreldes seniste lahendustega, pakkudes sihtrühmapõhist, paremini hallatavat ja kasutajasõbralikumat infovahetust. Süsteem võimaldab teateid filtreerida, visuaalselt eristada ja jälgida nende mõju, vähendades käsitsitööd ja toetades oluliselt nii tudengite kui ka programmijuhtide infokülluse haldamist.

Siiski tuleb arvestada, et ka kavandataval protsessil on teatud piirangud, mis võivad kasutajamugavust esialgu vähendada. Tudengi vaates tähendab uue süsteemi kasutuselevõtt seda, et osa teateid jääb endiselt Outlooki, mistõttu peab info leidmiseks liigutama mitme süsteemi vahel. Samas võib programmijuhi jaoks uude keskkonda sisenemine ja seal teadete koostamine olla lisakoormus, eriti kui enamik igapäevast suhtlust toimub tavaliselt Outlooki kaudu. Need kasutusmugavuse kitsaskohad ei vähenda uue süsteemi väärtust,

kuid viitavad vajadusele edasiste täiustuste järele. Võimalikud lahendused on käsitletud täpsemalt edasiarenduse alampeatükis (vt 5.4).

5.3 Hinnang tööle

Lõputöö eesmärk oli analüüsida ülikooli ametliku infovahetuse kitsaskohti ja arendada lahendus, mis muudab teavituste edastamise tudengitele sihipärasemaks, isikupärasemaks ja tõhusamaks. Töö käigus viidi läbi süsteemne probleemianalüüs, koguti kasutajate ootusi ja töötati välja prototüüpne lahendus, mis realiseerib uue, kasutajakeskse teavitussüsteemi. Võib öelda, et töö saavutas seatud eesmärgid ning arendatud süsteem vastab nii funktsionaalsetele kui mittefunktsionaalsetele nõuetele.

Töö üheks tugevuseks oli põhjalik eeltöö. Küsitlustele vastas märkimisväärne hulk tudengeid ja programmijuhte, mis võimaldas läbi viia sisuka ja mitmekülgse analüüsi. Sellest tulenevalt oli võimalik formuleerida nõuded, mis lähtuvad otseselt sihtrühmade vajadustest. Süsteemi funktsioonid (näiteks mallide loomine või tagasiside mehhanism) on kõik seotud kasutajate ootustega ning annavad hea aluse rakenduse edasiseks arendamiseks.

Nõuete selge mõistmine võimaldas üles ehitada lahenduse, mis vastas seatud eesmärkidele ning toetas teavituste sihipärast ja kasutajakeskset edastamist. Arenduse käigus kujunes oluliseks tugevuseks hästi läbimõeldud kihiline arhitektuur ja moodulipõhine ülesehitus, mis võimaldas süsteemi osi arendada ja testida isoleeritult. REST-põhine suhtlus kliendi ja serveri vahel tagas selge vastutusjaotuse, RTK Query lihtsustas andmehaldust kliendipoolel ning domeenipõhine äriloogika koos korduskasutatavate komponentidega võimaldab süsteemi funktsionaalsust tulevikus laiendada ilma põhistruktuuri ümber kujundamata. Kasutajaliides on täielikult responsiivne ja kohandub erinevatele ekraanisuurustele, pakkudes sujuvat kasutuskogemust nii arvutis kui mobiilseadmes. Süsteemi stabiilsust ja töökindlust toetavad arendusprotsessi jooksul läbiviidud automaatsed testid ja manuaalsed visuaalsed kontrollid, mis võimaldasid probleemid varakult tuvastada. Lisaväärtusena on kõik süsteemi API-päringud dokumenteeritud Swaggeri abil, mis lihtsustab liidestamist ja toetab tulevast arendust.

Prototüüp ei ole loodud Outlooki ega teiste ametlike kanalite asendamiseks, vaid pigem nende täiendamiseks. Seda võib käsitleda kui sihtrühmapõhist kasutajaliidest, mis aitab struktureerida ja filtreerida tudengitele edastatavat teavet arusaadavamal kujul. Näiteks võiks tulevikus realiseerida lahenduse, kus Outlookist saadetud kirjad suunatakse automaatselt uude süsteemi, kus programmijuht saab need kinnitada, muuta või edasi saata. See looks võimaluse infovoogu lihtsustada ja muuta see kasutajasõbralikumaks.

Tudengi vaates pakub süsteem suurt lisaväärtust: kõik teated asuvad ühes kohas, neid saab otsida, kategoriseerida, filtreerida ja vajadusel salvestada lemmikutesse. See aitab vähendada infomüra ning suurendab olulise teabe kättesaadavust. Samas programmijuhi jaoks ei pruugi uus süsteem hetkel veel otseselt lihtsustada igapäevast töövoogu, kuna senised tööprotsessid põhinevad suuresti Outlooki kasutamisel. Kuni ei ole loodud sobivaid liidestusvõimalusi, võib süsteemi paralleelne kasutamine tunduda lisasammuna. Küll aga loob loodud prototüüp tehnilised eeldused selleks, et tulevikus saaks teavituste koostamise ja edastamise protsess muutuda märksa sujuvamaks ja kasutajasõbralikumaks.

Samas tuli lõputöö käigus ette ka mitmeid kitsaskohti. Kõige olulisemaks puuduseks võib pidada seda, et ajapuuduse tõttu ei jõutud prototüüpi testida tegelike sihtkasutajate – tudengite ja programmijuhtide peal. Selle asemel toimus valideerimine arendaja vaatepunktist ning tehniliste katsete kaudu. Seetõttu puudub lõplik tagasiside kasutajakogemuse kohta, mis on eriti oluline just sellistes infosüsteemides, kus kasutajate harjumused ja ootused mängivad suurt rolli.

Teiseks piiranguks osutus kasutajaliidese kujundusprotsessi ülesehitus. Selle asemel, et töötada disain välja eelnevalt spetsiaalses prototüüpimisvahendis (nt Figma), valmis visuaalne lahendus paralleelselt arendusega. Kujunduslikud otsused sündisid suuresti jooksvalt koodi kirjutamise käigus, mis ei võimaldanud varakult hinnata liidese loogilisust ega kasutatavust. Selle tulemusel tuli mitmeid vaateid arenduse käigus ümber kujundada, et tagada parem kasutusmugavus. Eelnevalt koostatud disainiprototüüp oleks võimaldanud potentsiaalseid probleeme ennetada ning vähendanud ümbertegemiste vajadust lõppfaasis.

Kokkuvõteks töö saavutas püstitatud eesmärgid, pakkudes läbimõeldud ja sihtrühma vajadustest lähtuvat lahendust ülikooli teavitussüsteemi parendamiseks. Vaatamata mõningatele piirangutele kasutajate testimisel ja kujundusprotsessis, on valminud prototüüp funktsionaalne, hästi skaleeritav ning pakub tugeva aluse edasiseks arenduseks ja kasutuselevõtuks.

5.4 Edasiarendus

Kavandatud teavitussüsteem loodi viisil, mis võimaldab selle edasist arendamist nii funktsionaalsuse kui ka integreeritavuse osas. Üheks loogiliseks arengusuunaks on süsteemi liidestamine ülikooli tudengiportaaliga. Selle tulemusel võiks teavitustahvel integreeruda keskkonda, mida tudengid juba igapäevaselt kasutavad, vähendades vajadust liikuda mitme eraldiseisva infosüsteemi vahel.

Liidestusvajadus tuleneb ka TO-BE protsessis ilmnenud kitsaskohtadest. Nimelt peavad

tudengid uut süsteemi kasutama paralleelselt Outlooki ja teiste kanalitega, mis võib tekitada segadust ja vähendada süsteemi igapäevast kasutatavust. Programmijuhi jaoks tähendab eraldi süsteemi kasutamine lisasammu – senise Outlooki-põhise töövoo kõrvale lisandub vajadus logida sisse uude keskkonda ning koostada ja saata seal teateid, mis võib tunduda ajamahukas või ebamugav.

Selle olukorra leevendamiseks võiks tudengite seisukohast kaaluda teavitustahvli integreerimist otse tudengiportaali. Nii oleks kogu oluline info ühes kohas ning uue süsteemi omaksvõtt muutuks loomulikumaks. Programmijuhtide töö lihtsustamiseks võiks loodud süsteemi kasutada Outlooki laiendusena. Näiteks võiks programmijuht jätkata info koostamist Outlookis, kuid saata selle ühe nupuvajutusega edasi uude süsteemi, kus teade saaks vajaliku vormistuse (nt kategooria) ja jõuaks seejärel tudengini juba filtreeritaval ja visualiseeritud kujul. Alternatiivina võiks programmijuht Outlookis saabunud sõnumitele lihtsalt vajutada „Edasta süsteemi“ nupu kaudu, mis suunaks info uude keskkonda, kus programmijuht saaks selle kas koheselt või väikese muudatusega kinnitada ja tudengitele edastada.

Selline mehhanism looks võimaluse ka kolmandate osapoolte (nt programmijuhi abi) ligipääsuks, kelle ülesandeks võiks olla saadetud info süsteemis vormistamine ja edastamine. Protsessi saaks veelgi automatiseerida keelemudeli abil, mis suudab Outlookist pärit teated ise eeltöödelda, eemaldada üleliigse vormistuse ja lisada sobivad kategooriad. Nii jääks programmijuhile vajadusel ainult kinnitamise või väikese kohandamise roll. See muudaks töövoo kergemaks ja vähendaks Outlooki-põhise tööstiili muutmise vajadust.

Oluline on rõhutada, et süsteemi kihiline ja modulaarne arhitektuur, mis eristab selgelt kasutajaliidese, äriloogika ja andmekihi, toetab tulevaste liidestuste ja laienduste elluviimist tehniliselt lihtsalt ja selgelt. Kuna äriloogika on rakenduses jagatud iseseisvateks domeenideks ja API on dokumenteeritud Swaggeri abil, on uute ühenduspunktide loomine ja olemasolevate teenuste laiendamine hästi hallatav. Seega on võimalik lahendust liidestada ülikooli infosüsteemidega ilma olemasoleva süsteemi loogikat ümber kirjutamata.

Lisaks liidestamisele väärivad tähelepanu ka teised tulevikusuunad. Näiteks võiks arendada välja reaalajas toimiv märguannete süsteem, mis kuvab uusi teateid kohe nende avaldamisel. Teine oluline täiendus oleks võimalus lisada teadetega faile – näiteks praktikapakkumiste juures. Täiendavat väärtust looks ka mallifunktsiooni edasiarendus, mis võimaldaks lisaks olemasolevale käsitsikasutusele ajastada automaatselt korduvaid teavitusi kindlateks kuupäevadeks või sündmusteks. Nii saaks programmijuht ette planeerida tähtsate sõnumite ilmumise just õigel ajal. Pikaajalises vaates oleks võimalik arhitektuur jagada mikroteenusteks, mis suurendaks süsteemi töökindlust ja skaleeritavust.

6 Kokkuvõte

Käesoleva töö eesmärgiks oli kavandada ja realiseerida teavitustahvli prototüüp, mis aitaks vähendada info üleküllust üliõpilaste seas ning lihtsustaks programmijuhtide igapäevast suhtlust tudengitega. Analüüsi ja arendustöö aluseks olid läbiviidud kasutajaküsitlused, mille kaudu selgitati välja tudengite ja programmijuhtide peamised vajadused. Nende tulemuste põhjal tuvastati mitmed kitsaskohad olemasolevates lahendustes, nagu infoüleküllus, piiratud võimalused teadete filtreerimiseks ning tagasisideprotsessi puudumine.

Saadud andmete alusel loodi kolmekihiline veebirakendus, mille kasutajaliides põhineb Reacti komponentarhitektuuril, rakendusloogika Node.js ja Expressi platvormil ning andmete püsivust tagab PostgreSQL andmebaas. Olulisteks uuendusteks olid teavitustele kategooriate ja prioriteetide lisamise võimalus koos visuaalse eristusega, mallide loomine ja taaskasutus, lugemisstaatuse jälgimine ning isikupärastatud vaated tudengivaates.

Rakendust testiti nii automatiseeritud testide kui ka manuaalse testimise kaudu. Saavutatud jõudlus ja kasutajaliidese selgus kinnitavad, et lahendus on praktiliselt kasutatav ja vastab enamikele seatud nõuetele. Süsteemi ülesehitus on modulaarne ja võimaldab vajaduse korral funktsionaalsust hõlpsasti laiendada.

Töö tulemusena valmis esimese versiooni prototüüp, mis ei ole veel tootmisvalmis, kuid pakub tugeva aluse edasiseks arenduseks. Süsteemi kasutuselevõtt võimaldaks programmijuhtidel säästa aega, tudengitel paremini navigeerida infohulgas ning kogu ülikoolil saavutada läbipaistvam ja tõhusam infovahetus.

Kuna töö keskendus esmase funktsionaalsuse väljatöötamisele, ei jõudnud autor integreerida loodud prototüüpi tudengiportaaliga. Samuti ei olnud võimalik lahendust testida reaalsete kasutajatega, mistõttu puudub empiiriline tõendus selle kohta, kuidas süsteem praktikas info kättesaadavust või halduskoormust mõjutab. Need piirangud ei vähenda siiski tehtud töö väärtust, vaid osutavad selgelt, millises suunas edasi liikuda. Töö loob tugeva lähtekoha teavitussüsteemi arendamiseks ja ülikooli infovahetuse tulemuslikumaks muutmiseks.

Kasutatud kirjandus

[1] M. J. Eppler and J. Mengis. “The Concept of Information Overload: A Review of Literature from Organization Science, Accounting, Marketing, MIS, and Related Disciplines”. In: *Information Society* (2004), pp. 325–344. DOI: 10.1080/01972240490507974.

[2] D. Bawden and L. Robinson. “The Dark Side of Information: Overload, Anxiety and Other Paradoxes and Pathologies”. In: *Journal of Information Science* (2009), pp. 180–191. DOI: 10.1177/0165551508095781.

[3] *The library for web and native user interfaces*. [Kasutatud: 20.03.2025]. URL: https://react.dev/.

[4] *HTML: HyperText Markup Language*. [Kasutatud: 20.03.2025]. URL: https://developer.mozilla.org/en-US/docs/Web/HTML.

[5] *CSS: Cascading Style Sheets*. [Kasutatud: 20.03.2025]. URL: https://developer.mozilla.org/en-US/docs/Web/CSS.

[6] *JavaScript*. [Kasutatud: 20.03.2025]. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[7] *About Node.js®*. [Kasutatud: 20.03.2025]. URL: https://nodejs.org/en/about.

[8] *PostgreSQL: About*. [Kasutatud: 20.03.2025]. URL: https://www.postgresql.org/about/.

[9] J. vom Brocke, A. Hevner, and A. Maedche. “Introduction to Design Science Research”. In: *Design Science Research*. 2020. DOI: 10.1007/978-3-030-46781-4_1.

[10] *Software Development Life Cycle (SDLC)*. [Kasutatud: 12.04.2025]. URL: https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/.

[11] C. Lawrence, T. Tuunanen, and M. D. Myers. “Extending Design Science Research Methodology for a Multicultural World”. In: *Design Science Research in Information Systems*. 2010. DOI: 10.1007/978-3-642-12113-5_7.

[12] A. Ghanad. *An Overview of Quantitative Research Methods*. 2023. DOI: 10.47191/ijmra/v6-i8-52.

[13] S. J. Agius. “Qualitative research: Its value and applicability”. In: (2013). DOI: 10.1192/pb.bp.113.042770.

[14] J. Romero. *Gmail gets AI-powered "Summarize" feature on iOS and Android, Gemini side panel on the web*. [Kasutatud: 15.04.2025]. URL: https://www.phonearena.com/news/gmail-gets-ai-powered-summarize-feature-on-ios-and-android-gemini-side-panel-on-the-web_id159763.

[15] R. Sheldon. *TL;DR (too long; didn't read)*. [Kasutatud: 15.04.2025]. URL: https://www.techtarget.com/whatis/definition/tldr-TLDR.

[16] *Discord - Group Chat That's All Fun & Games*. [Kasutatud: 15.04.2025]. URL: https://discord.com/.

[17] *API Documentation & Design Tools for Teams*. [Kasutatud: 14.04.2025]. URL: https://swagger.io/.

[18] *TalTech brändi materjalid*. [Kasutatud: 23.04.2025]. URL: https://taltech.ee/brand.

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

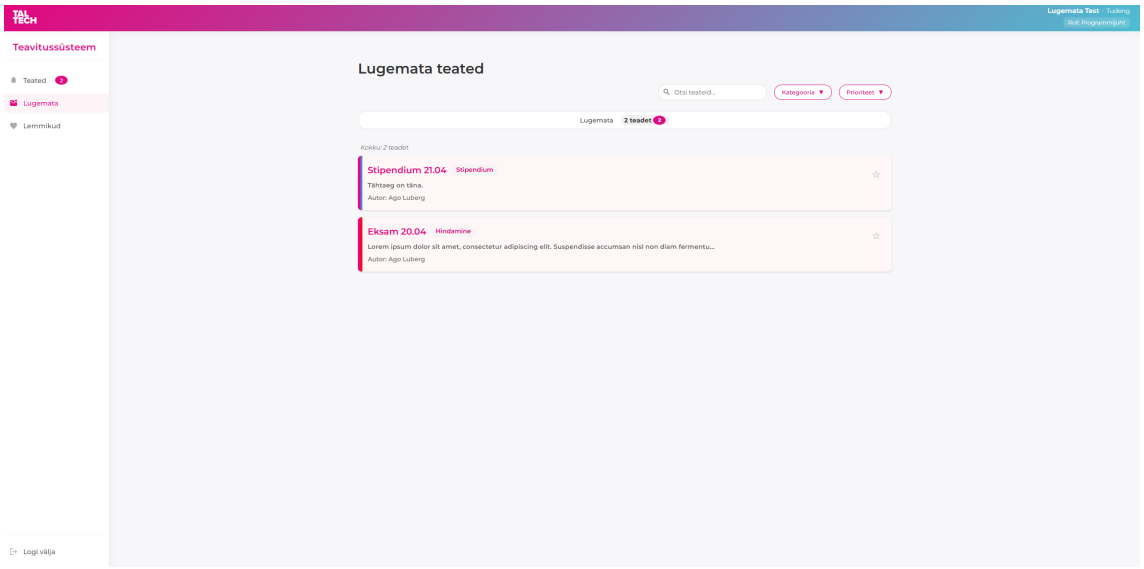
Mina, Dmitri Kuznetsov

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Ülikooli ametlike infokanalite koormus ja valikulise teabe levitamise tõhusus: infoülekülluse vähendamine e-kuulutuste tahvli kaudu", mille juhendaja on Karl-Erik Karu
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

04.06.2025

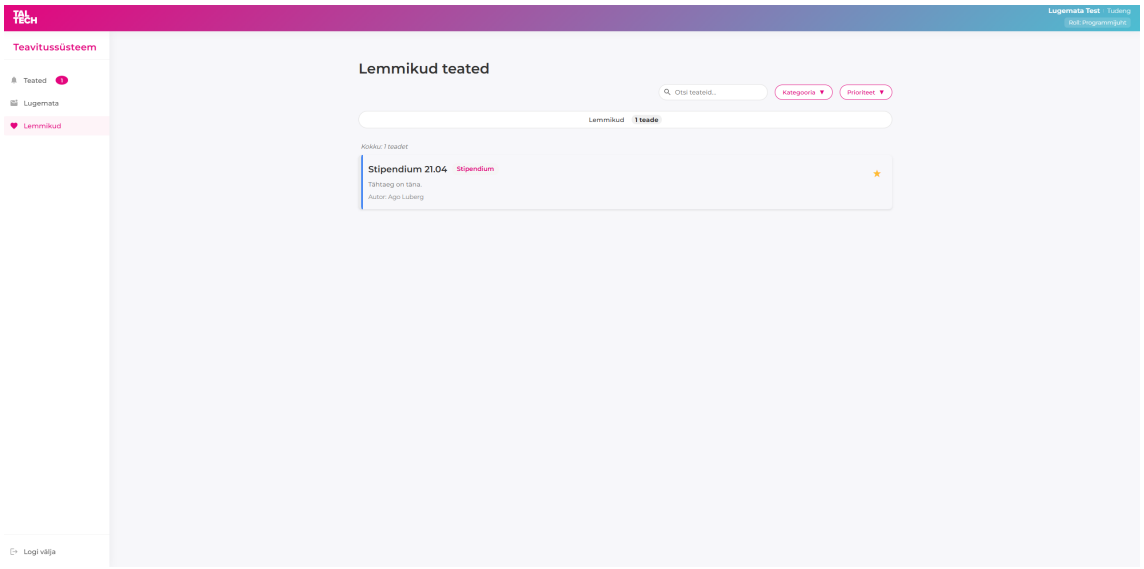
[1] Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või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

Lisa 2 – „Lugemata“ vaade



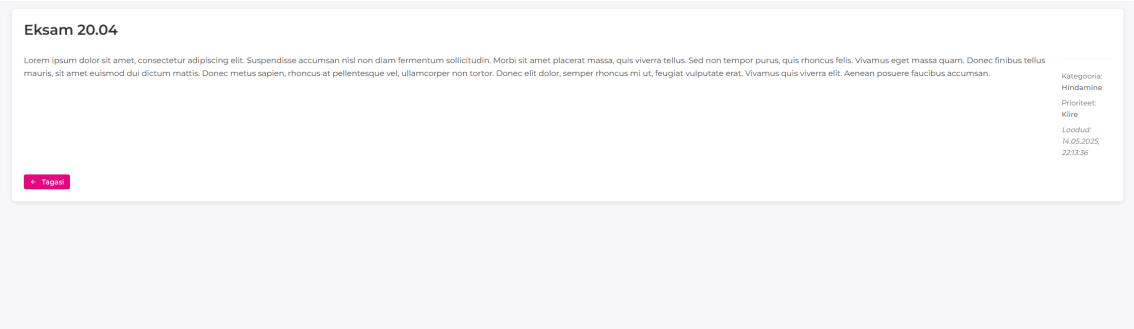
Lugemata teadete vaade.

Lisa 3 – „Lemmikud“ vaade



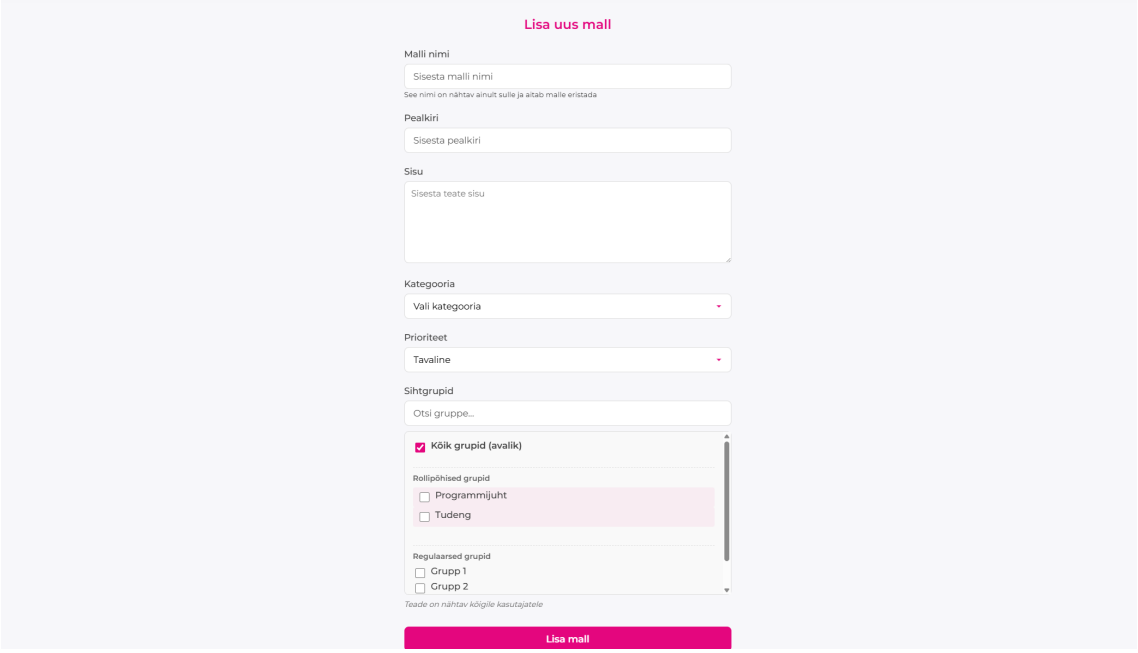
Lemmikute teadete vaade.

Lisa 4 – Teate detailvaade tudengi vaates

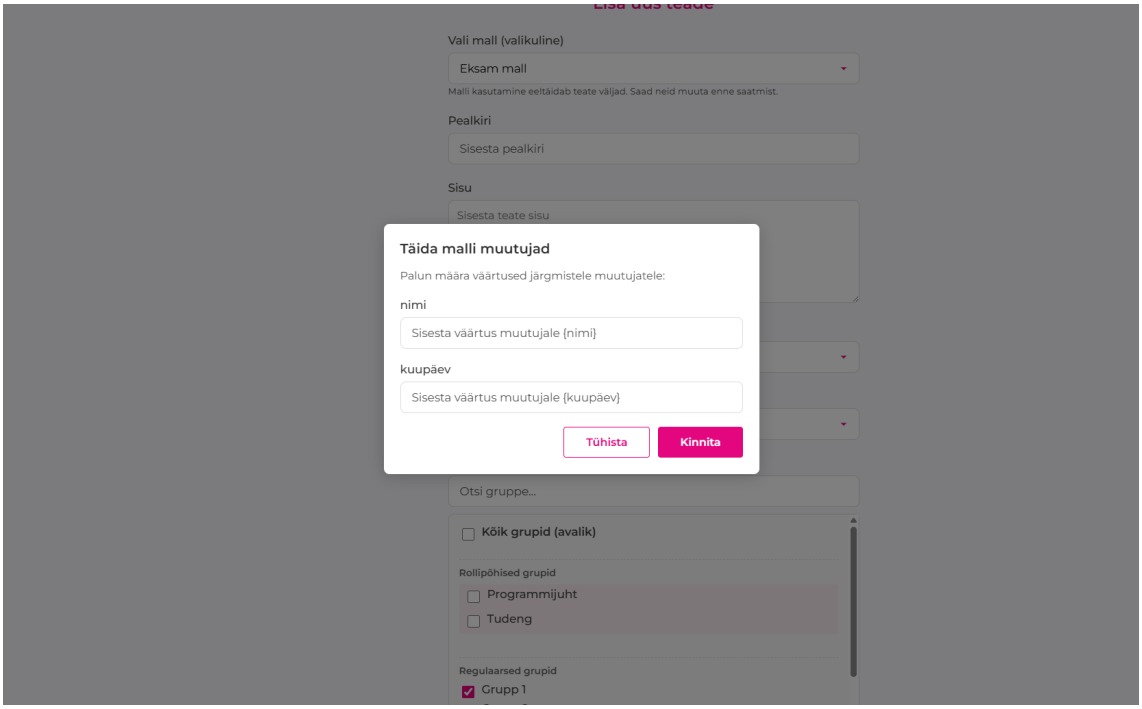


Teate detailvaade tudengi vaates.

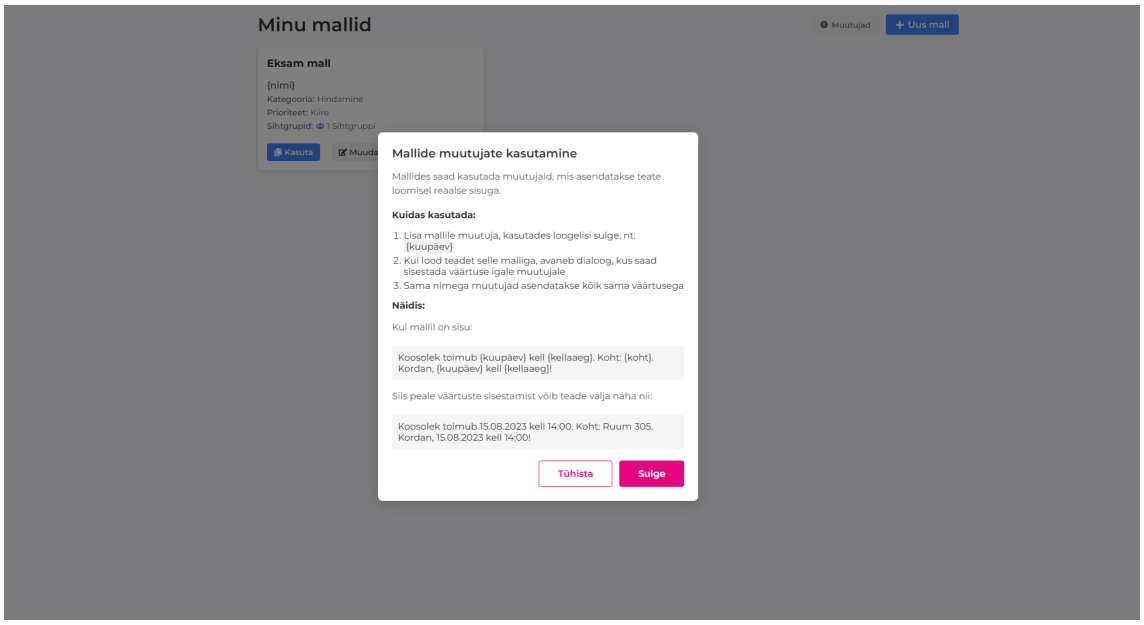
Lisa 5 – Mallide haldamise vaated



Uue malli loomise vaade.

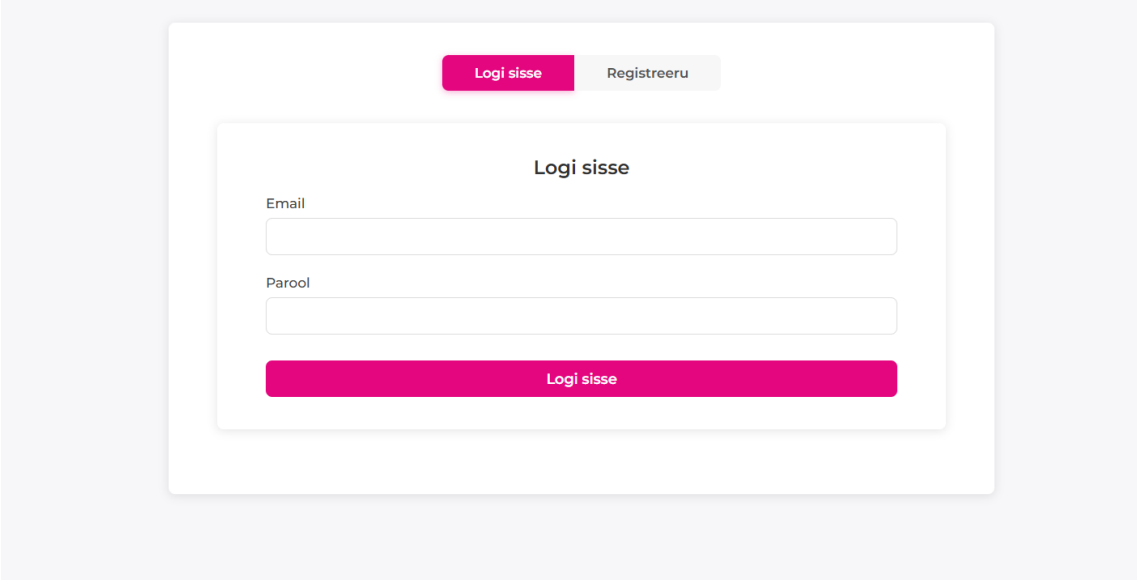


Teadele muutujate lisamine mallist.

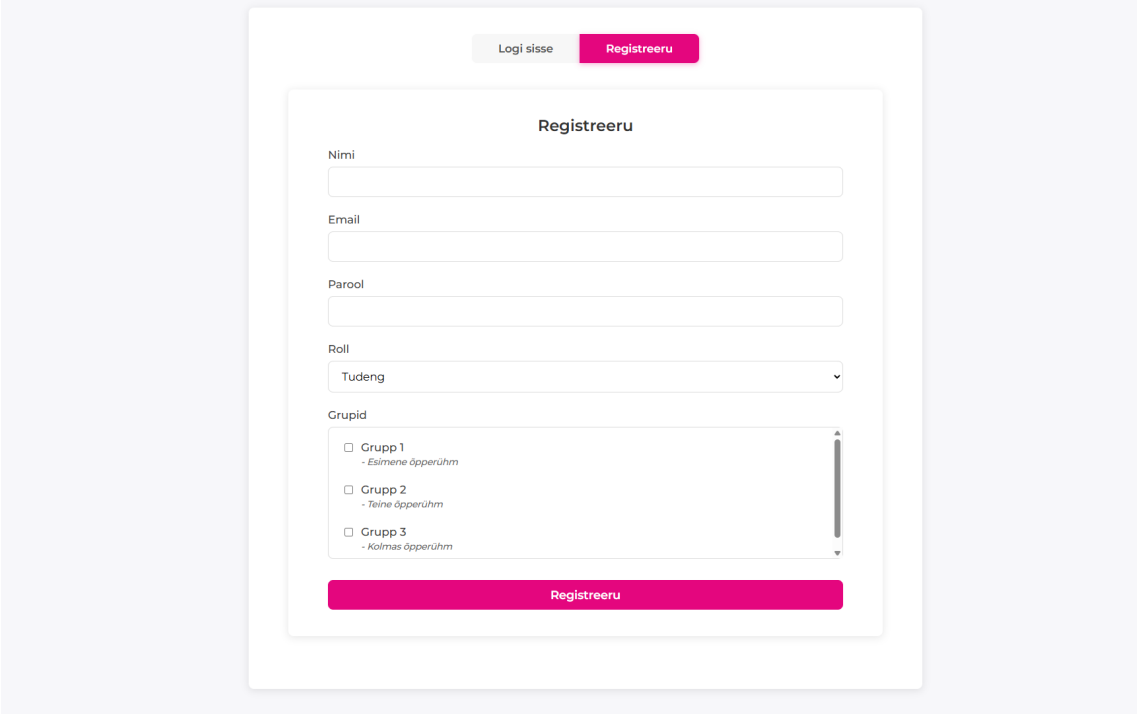


Malli muutujate kasutamisjuhend.

Lisa 6 – Autentimisvaated



„Sisselogimine“ vaade.



„Registreerimine“ vaade.