TARTU ÜLIKOOL

Arvutiteaduse instituut
Informaatika õppekava

August Roosi

Veebileht argumentatsioonikaartide loomiseks

Bakalaureusetöö (9 EAP)

Juhendaja:
Vambola Leping

Tartu 2025

Veebileht argumentatsioonikaartide loomiseks

Lühikokkuvõte:

Bakalaureusetöö eesmärk oli välja arendada veebilehepõhine tööriist argumentatsioonipuude loomiseks ning visualiseerimiseks. Veebilehe kujundamisel pöörati tähelepanu veebilehe selgusele ja arusaadavusele. Loodud veebileht on eestikeelne ning selle funktsionaalsus võimaldab lehte kasutada meeskonnatöös mitmel kasutajal samaaegselt. Töös selgitatakse argumentatsioonipuude mõistet, analüüsitakse olemasolevate argumentatsioonipuude tegemise veebilehti ja nende nõrkusi ja tugevusi ning näidatakse, kuidas praktiline osa nende lahenduste arendamisele kaasa aitab. Lisaks eeltoodule kirjeldatakse lõputöö praktilise osa tööprotsessi ning kasutusvõimalusi. Praktilise osa tegemiseks kasutati React teeki ning argumendipuude kuvamiseks React Flow teeki. Andmebaas ehitati üles PostgreSQL-iga ning suhtlus andmebaasi ja veebilehe vahel tagati Django raamistikuga. Töö tulemusena valmis praktiliselt kasutatav lahendus, mille toimivust katsetati testkasutajate poolt.

Võtmesõnad: argumentatsioonipuu, veebirakendus, React-Flow, andmebaas, visualiseerimine

CERCS: P170

Webpage for creation of argument maps

Abstract:

The goal of this bachelor's thesis was to develop a web-based tool for creating and visualizing argument maps. The site was designed with clarity and usability in mind, supports the Estonian language, and allows multiple user collaboration. The thesis explains the concept of argument maps, analyzes existing tools and their strengths and weaknesses, and demonstrates how the developed solution addresses these. The practical part was built using React and React Flow for visualization, with a PostgreSQL database and Django for backend communication. The final product was tested by users and is ready for practical use.

Keywords: argumentation tree, web application, React Flow, database, visualization

CERCS: P170

Sisukord

Sissejuhatus

Veebilehed on väga levinud viis tööriistade levitamiseks internetis. Kui osata otsida, siis leiab rakendusi, mis näiteks aitavad leida tee kohaliku kohvikuni, pakuvad võimalusi pilditöötluseks, võimaldavad tekstifailide redigeerimist ja palju muud. Tööriistu, mis aitavad muuta paremaks arutelusid teiste inimestega või iseendaga, on loodud juba mõnda aega tagasi, kuid need on algselt olnud joonistena paberil, hiljem ka arvutis allalaetavate rakendustena ja nüüd on tekkinud uued lahendused veebilehtedena. Veebilehti, mille põhifookus on hõlbustada ning tõsta arutelu kvaliteeti on üpris vähe, ning kasutajate teadlikkus nende olemasolust on pigem väike. Minu lõputöö eesmärk on uurida parimaid argumentatsioonikaardi loomise võimekusega veebilehti ning proovida nende näitel ehitada uus ja edasiarendustega versioon. Eeskuju võtan Argumentation.io ja Mindmup veebilehtedelt. Esimeses peatükis seletan lahti mõned olulisemad mõisted ja terminid. Teises peatükis kirjeldan täpsemalt argumentatsioonikaardi olulisust, toon näiteid millised veebilehed selle loomist kõige paremini võimaldavad, mis on nende tugevused ja nõrkused ning kuidas minu praktiline osa saab pakkuda nendest inspireerituna sama probleemi lahendamiseks edasiarendusi. Kolmandas peatükis seletan lahti kasutatud tehnoloogiad ning nende valimise tagamaid, kirjeldan rakenduse arhitektuuri ning erinevate koodi struktureerimise valikute põhjendusi. Neljandas peatükis analüüsin ja kirjeldan veebilehe peamist kasutuse olukorda, veebilehe testimist ning veebilehe kõigile kättesaadavaks tegemist. Viiendas peatükis kirjeldan veebilehe kasutust ning kasutajate tagasisidet.

1. Mõisted ja terminid

Argumentatsioon (ingl *argumentation*) ehk väite tõesuse tuletamine.

Tipp (ingl *vertice*) ehk punkt graafil, mis tähistab mingit objekti, minu lõputöös argumenti.

Serv (ingl *edge*) ehk ühendus kahe tipu vahel graafil, see annab edasi mingi kahe tipu vahelise seose või omaduse.

Puu (ingl *tree*) ehk sidus tsükliteta graaf.

Juurtipp (ingl *root node*) ehk kõige kõrgema taseme tipp, millel puudub ülemus. Kõigil teistel tippudel on täpselt üks tipp.

Argumentatsioonikaart (ingl *argumentation map*) ehk argumentatsiooni visuaalne väljendus.

Argumentatsioonipuu (ingl *argumentation tree*) on argumentatsioonikaart, mis on kujult graafiteooria puu. Töös kasutan läbivalt argumentatsioonikaardi ja argumentatsioonipuu mõisteid, sest need terminid on oma sisult väga sarnased.

Mõttekaart (ingl *mindmap*) ehk mõtete või teabe kaardistamine nii, et seotud info mullid on üksteisega ühendatud, kindel kuju puudub.

Loogilised operaatorid (ingl *logical operators*) ehk sümbolid, mis võtavad kaks tõeväärtus argumenti ning annavad tõeväärtus väljundi vastavalt sellele, millise loogilise operaatoriga tegemist on. Kõige tuntumad loogilised operaatorid on JA ning VÕI. Näiteks JA annab tõene väärtuseks, kui mõlemad tema argumendid on tõesed.

Implikatsioon (ingl implication) on loogikas ja matemaatikas kasutatav mõiste, mis viitab järeldusele või tagajärjele, mis tuleneb teatud eeldustest või tingimustest. Lihtsamalt öeldes tähendab see, et kui üks väide on tõene, siis sellest järeldub, et ka teine väide on tõene. Implikatsioon on väär, kui tõesest eeldusest tehakse väär järeldus.

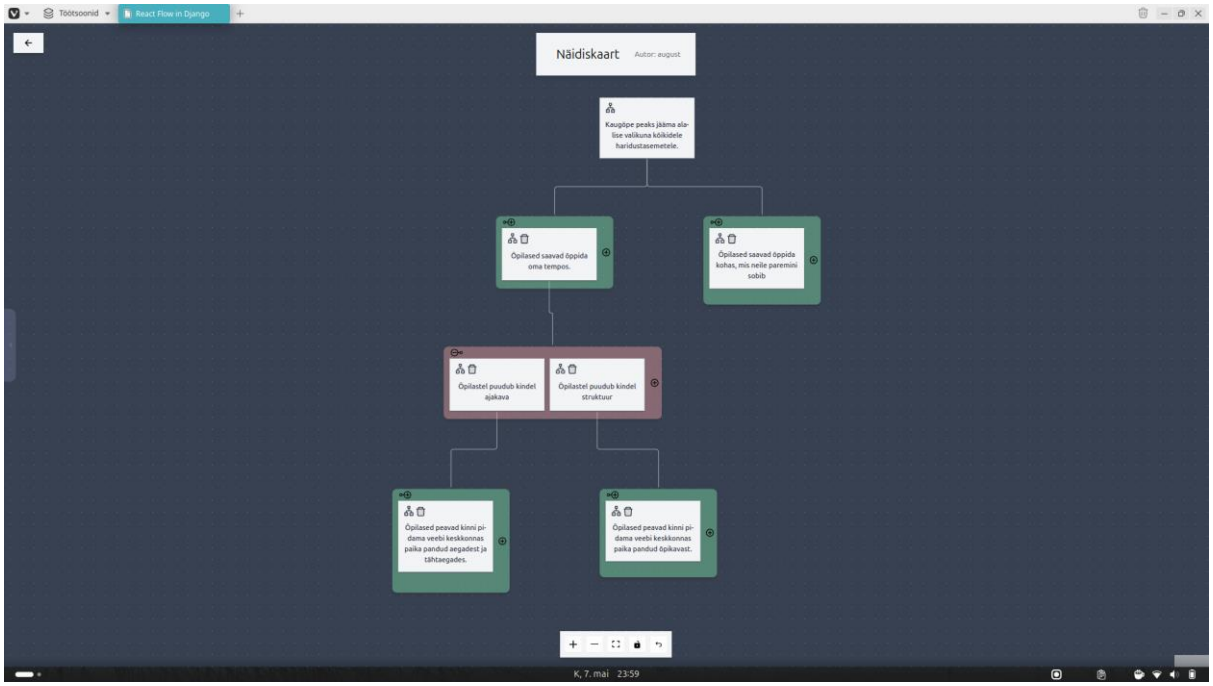
Tagaliides (ingl *backend*) on projekti serveri poolne osa, kus on kood ning vajalikud failid ja kaustad, et hallata andmebaasi ja eesliidese vaheline suhtlus ning andmete äriloogika.

Eesliides (ingl *frontend*) on projekti kasutaja poolne osa, kus on kood ning muud vajalikud failid ja kaustad, et luua kasutaja vaade, selle välimus, ning kõik kasutaja poolne äriloogika.

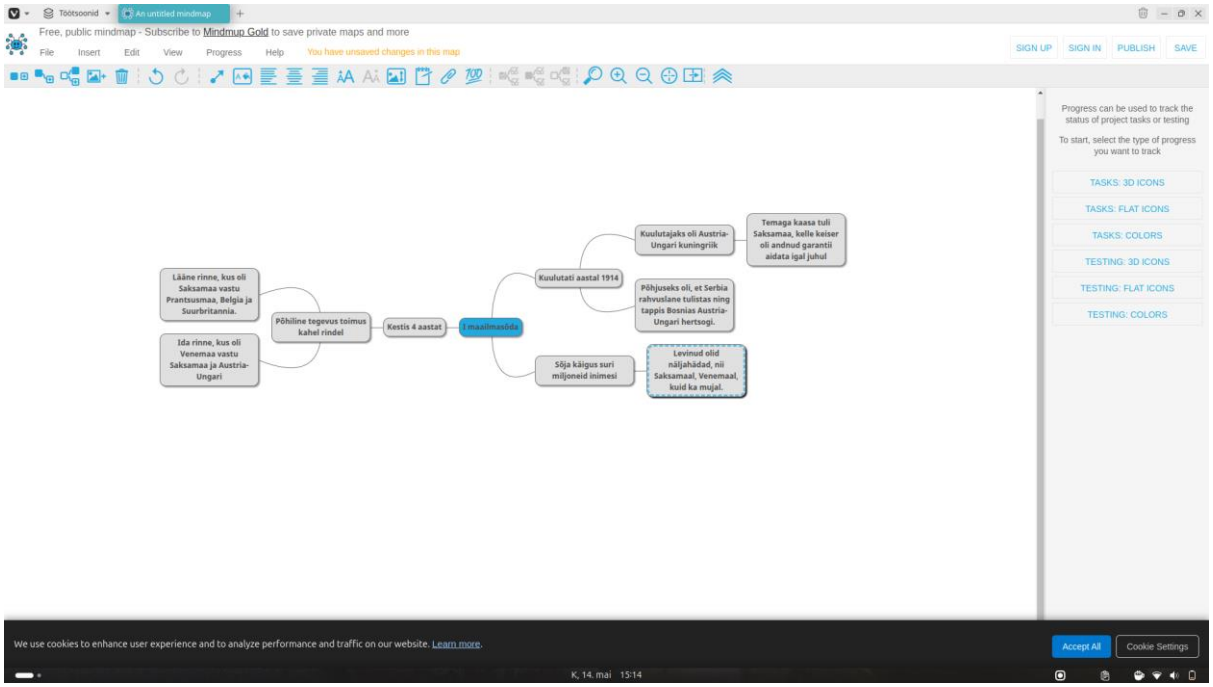
2. Taustauuring

Argumentatsioon, vaidlus või väitlus on mingi väite tõeväärtuse hindamine. Tavaliselt esitatakse selleks algväite kohta poolt- ning vastuargumente. Argumentatsioonikaart on argumentatsioonide visuaalne väljendamine erinevatel viisidel. Lõputöö veebilehe loomisel rakendasin argumentatsioonipuu põhimõtet, kus argumentatsioon kujutatakse graafiteooria puu kujul. Argumentatsiooni on kasulik kujutada puuna, sest siis saab teha järgnevaid visuaalseid eristusi: kasutada erinevaid värve ja kujundeid, teabe esitamise hierarhiat. Cullen jt (2018) leidsid oma uuringus, et argumentatsioonipuude kasutajatel on täheldatud parem kriitiline mõtlemine ning et neil suurenes arusaamine õpitavatest materjalidest võrreldes üliõpilastega, kes argumentatsioonipuid ei kasutanud. Arendan tööriista, mis võimaldaks kasutajatel mugavalt luua argumentatsioonikaarte, lisaks on eesmärk tagada hea ligipääsetavus ning kindla arvamusega juhend loodud tööriista kasutamiseks. Lõputöö jaoks otsustasin teha argumentatsioonikaardi veebilehena, sest siis ei pea kasutaja argumentatsioonikaardi loomiseks alla laadima eraldi tarkvara, vaid saab kasutada tööriista otse oma veebilehitsejas.

Argumendikaart koosneb peamiselt tippudest, mis on mullid, kuhu on tekst sisse kirjutatud ning servadest, mis on jooned erinevate tippude vahel. Tipud võivad olla ka värvilised või erineva kujuga, et mingit tähendust edasi anda. Kui tipp on punane siis see on tavaliselt vastuargument, kui roheline siis on poolt ning halli puhul määramata või mitte selge. Minu praktilise osa argumendikaardi struktuur jaotub peamiselt kolmeks (vt joonis 1). Esiteks mullid on argumendigrupid ja kui argumendigrupp on roheline, siis on ta poolt ning kui on punane või hall siis on tegemist määramata argumendigrupiga. Teiseks, valged kastid on argumendid, kui on argument, millel pole mulli, siis on tegemist algväitega. Lisaks, kui argumente on ühes mullis mitu, siis piisab ühe argumendi ümber lükkamisest, et kogu argumendigrupp oleks kehtetu. Kolmandaks, valged jooned on ühendused argumentide või argumendigruppide vahel. Mõttekaart on nagu argumendipuu aga pole kindlat alati struktuuri, erinevaid tipu liike on vähem ning eesmärk ei ole otseselt millegi tõestamine (vt joonis 2).



Joonis 1. Argumendikaardi näidis. Allikas: autori koostatud



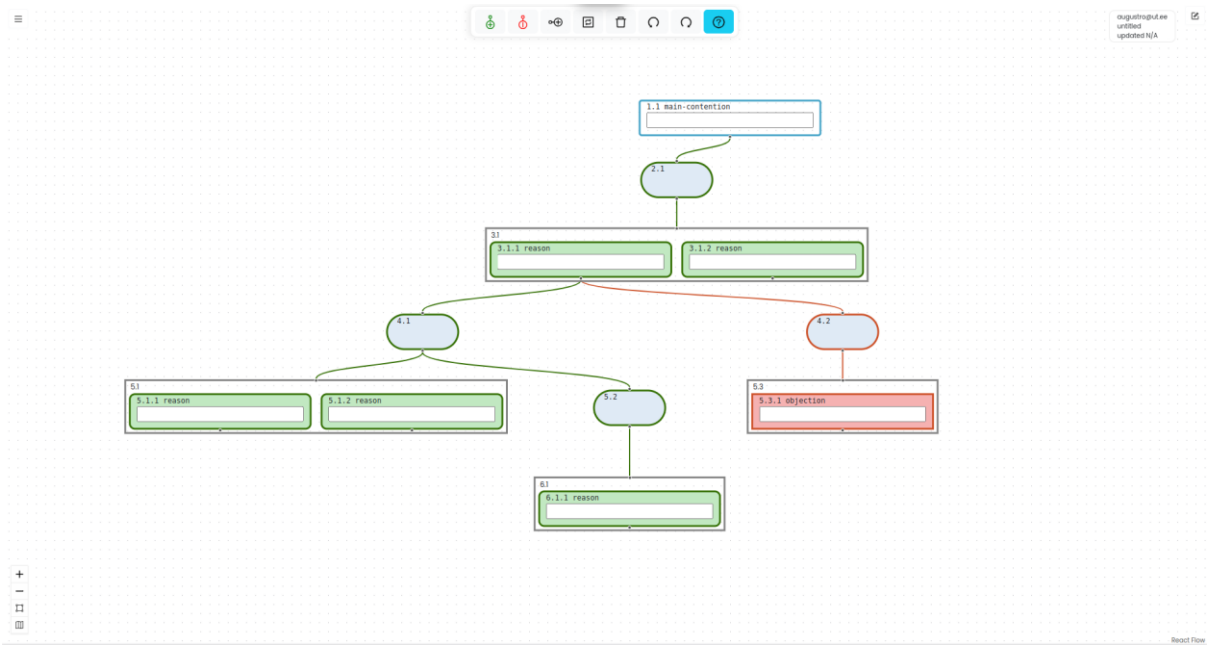
Joonis 2. Mõttekaardi näidis. Allikas: Mindmup, 2025. https://www.mindmup.com.

Veebilehed kus saab teha mõttekaarte on üsna levinud, kuid veebilehti, millega saaks teha argumenatsioonipuid leidub vähem. Selleks et välja arendada tööriist, millega saab struktureeritult kujutada argumentatsioone, uurisin juba olemasolevaid lahendusi. Võrdluseks on võetud järgnevad veebilehed: Argumentation.io (2025) ja MindMup (2025). Analüüsi koostamisel võtan arvesse erinevaid muutujaid, mis saavad mõjutada argumendikaardi

kasulikkust. Davies jt (2019) toovad välja, et eriline tähtsus on argumentatsioonipuude tööriistadel visuaalsel osal, et inimesel oleks võimalikult lihtne hoomata keerulist teavet. Kuna tööriista kasutaja kompetentsi ei saa tagada, siis Rathkopf (2024) juhib tähelepanu, et tööriist peaks aitama suunata kasutajat ning tugevdada tema võimekust teha loogilisi arutelusid. Lisaks mainivad Davies jt (2019), et argumendipuude visualiseerimisel peaks olema rõhk just argumentide loogilistel ühendustel ning järeldustel.

2.1 Argumentation.io

Argumentatio.io on üks väheseid veebilehti, mis keskendub ainult argumentatsiooni visualiseerimise funktsionaalsusele, ilma mõttekaartide loomise võimalust kaasamata. Veebilehel on argumendikaart puu kujul, kuid erinevalt tavalisest saab kahe tipu vahelisele ühendusele argumendi teha. Ehk kui kasutajal on väide ja see on ühendatud argumendiga ning kasutaja nõustub väite ja argumendiga aga ei nõustu, et argumendist järeldub väide, siis saab ta just selle järeldusele vastu vaielda. Olukorras, kus mitu argumenti on koos põhjuseks ühele väitele ning sõltuvad üksteisest, saab need argumendid ühe serva alla viia, näiteks joonis 3, kus mitu argumenti on kõrvuti. Veebilehel on vastu- ja pooltargumendid tähistatud vastavalt punase ja rohelise värviga. Lisaks saab lugeda juhendit ning luua kasutaja, kuid salvestamise ja jagamise funktsionaalsuse jaoks on vaja tasuda liikmemaks, mis on 5 € kuus. Veidi segadust võib tekkida, kui kustutada kuskilt serv või tipp, sest sellega kaovad ka kõik selle alamservad ja -tipud.

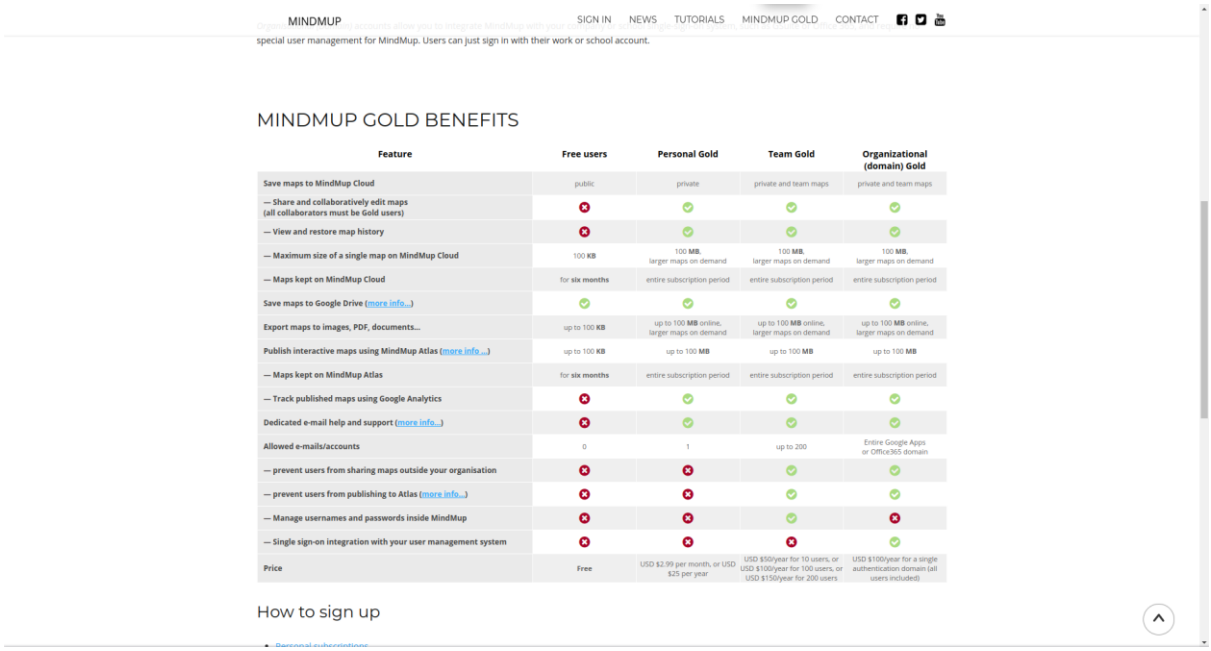


Joonis 3. Veebilehe Argumentation.io argumendipuu loomise kuva. Ovaal siin kaardil tähistab implikatsiooni selgitust. Allikas: Argumentation.io, 2025. https://argumentation.io.

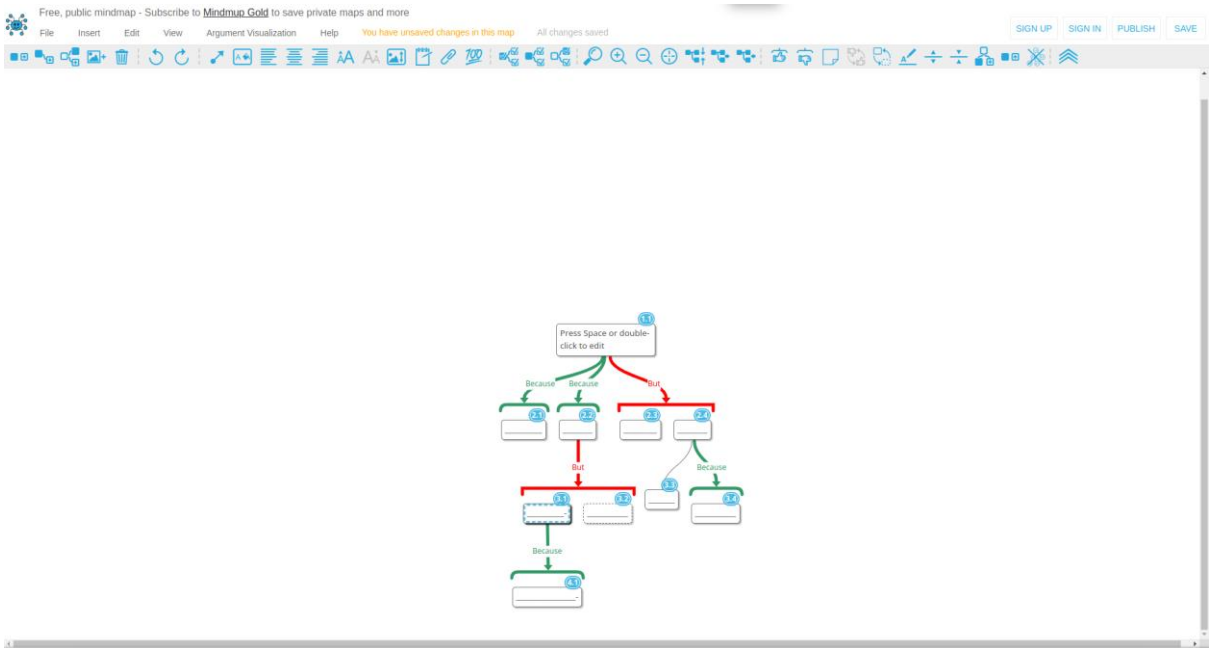
Kasutajaliides on ingliskeelne ning leht on kaasaegse kujundusega. Nupud on selgelt eristatavad ning kergesti mõistetavad isegi lisaselgitusteta (vt joonis 3). Ekraanil kuvatakse korraga ainult vajalik informatsioon. Kui argumente tekib palju, võivad servad mingil määral kaotsi minna, kuid see ei mõjuta oluliselt loetavust. Argumentation.io (2025) on loodud kasutades React Flow'd, mis tähendab, et mingil määral kasutatakse ka Reacti. Andmebaasi majutamiseks kasutas veebileht populaarset teenust Firebase.

2.2 Mindmup

MindMup (2025) leht esitab kasutajale algselt mõttekaardi loomise vaate, kuid kui piisavalt otsida, saab menüüst leida ka argumentatsioonipuu loomise võimaluse. Leht jaotab argumendipuu servadeks ja tippudeks. Kuigi servadele saab lisada teksti, puudub võimalus neile omakorda väiteid lisada. Veebileht pakub mitmeid visuaalseid kohandamisvõimalusi: saab muuta tippude kuju, värvi, suurust ja asukohta, kuid nende valikute tähtsus on kaheldav. Lisaks, kui tippe ja servi tekib liiga palju, saab need kokku pakkida üheks tipuks ning vajadusel uuesti lahti pakkida. Sarnaselt eelnevatele on vastu- ja pooltargumendid vastavalt tähistatud punase ja rohelisega. Samuti on võimalik muuta serva laiust, et edastada argumendi olulisust. Olukorras, kus mitu argumenti on koos põhjuseks ühele väitele, saab need ühe servaga väitega ühendada. Veebileht pakub ka järgmisi teenuseid: salvestamine nii pilve kui ka arvutisse, ajaloo vaatamine ning jagamine (vt joonis 4). Tasuta teenustest on saadaval ainult salvestamine.



Joonis 4. Veebilehe Mindmup.com liikmemaksu valiku kuva. Allikas: Mindmup, 2025. https://www.mindmup.com.



Joonis 5. Veebilehe Mindmup.com argumendipuu loomise kuva. Allikas: Mindmup, 2025. https://www.mindmup.com.

Kasutajaliides on ingliskeelne ning kuigi argumendipuu ise on välimuse poolest selge ja lihtsalt loetav, ei kehti sama ülejäänud lehe kohta. Tööribal on väga palju nuppe, kuid puudub nähtav omadus, mis eristaks tähtsamaid nuppe vähem olulistest (vt joonis 5). Soovitud tulemuse saavutamiseks on kasutajal sageli vaja katsetada erinevaid nuppe ja otsida valikuid

menüüdest. MindMup GitHubis (2019) on näha, et argumentatsioonikaartide loomiseks ei ole kasutatud eraldiseisvat raamistikku, vaid loodud on omaenda raamistik, mis põhineb JavaScriptil ja canvas-elemendil. Muu kohta teave puudub, kuna kood ei ole avalik.

2.3 Lõputöö erinevused olemasolevatest veebilehtedest

Eelnevalt välja toodud veebilehed on parimad, mis pakuvad argumentatsioonipuu loomise võimalust ning MindMup (2025) on olnud kasutusel ka Rathkopf (2024) uurimistöös. Tehnoloogia osas võtan eeskuju Argumentation.io-lt (2025), kasutades React Flow raamistikku. Siiski on ruumi arenguks.

Esiteks luuakse veebileht eestikeelsena, sest samalaadset eestikeelset veebilehte hetkel pole. Teiseks loon argumenteerimise süsteemi, mis kasutab loogilisi operaatoreid, põhinedes matemaatilistel vastetel: "ja", "või" ning implikatsioon. Kolmandaks arendan välja argumendipuu peitmise funktsionaalsuse, mis võimaldab argumendi alampuud kokku pakkida. See lihtsustab ülevaadet, eriti keerukate ja mahukate argumentide korral. Rathkopf (2024) rõhutab, et tõhus argumentatsioonipuu peaks suutma visualiseerida ka suuri ja keerulisi argumente, pakkudes selge ülevaate kõige olulisematest väidetest ja nendevahelistest seostest. See aitab kasutajal keskenduda põhiväidetele isegi tihedamate kaartide korral, vältides ülekoormatust. Neljandaks lisan võimaluse viidata eelnevatele argumentidele uues argumendipuus, kui väitele on juba varem puu loodud. Viiendaks võiks kõigil kasutajatel olla ligipääs ühtsesse kogumisse, kus on välja toodud kõik argumentatsioonikaardid.

Lisaks oleks oluline luua veebilehele samalaadne muudatusteloo funktsioon nagu MindMupis (2025), võimaldades vaadata kaardi loomise ajalugu ja vajadusel taastada eelnev olek. Argumentation.io (2025) võimaldab lisada argumente järelduse servale. Seda funktsiooni tuleks rakendada ka lõputöö veebilehel ning seejuures selgemalt eristada selliseid argumente tavaargumentidest. See aitaks kaasa lehe funktsionaalsuse ja kasutusmugavuse parandamisele ning lahendaks Rathkopfi (2024) probleemi, kus on soov vastu vaielda, mitte argumendi sisule vaid eelduse ja järelduse vahelisele seosele ehk implikatsioonile.

3. Rakenduse arhitektuur ja tehnoloogiad

Praktilise osa tehnoloogiateks valisin Django, React ja PosgreSQL-i. Django valisin seetõttu, et kuigi see nõuab suuremat ülesseadistust ning head arusaama, kuidas selle arhitektuur töötab, toetab see kiiret veebilehe arendust. Paljud mudelid, mida mul vaja läheb, näiteks kasutaja ja administraator, on juba ette loodud, lisaks on mul varasem kogemus Django projektide loomisega.

3.1 Tagaliides

Django struktuur koosneb kolmest põhiosast, mida kokkuvõtvalt nimetatakse MVT-ks (Model–View–Template). Esiteks on *model* ehk mudel, mis kirjeldab rakenduse andmestruktuuri ja loogikat. Mudelid määravad ära, millist teavet rakendus talletab ning kuidas eri andmeüksused omavahel seotud on. Django kasutab neid mudeleid andmebaasis tabelite loomiseks ning nende kaudu andmetega suhtlemiseks. Selle lõputöö raames kasutatakse andmebaasimootorina PostgreSQL-i, mis on võimas ja avatud lähtekoodiga relatsiooniline andmebaas. Teiseks on *view* ehk vaade, mis sisaldab rakenduse äriloogikat – seal defineeritakse, kuidas erinevad andmepäringud ja kasutajategevused töödeldakse. View vastutab ka selle eest, millised andmed kasutajale edastatakse. RESTful API puhul on View see koht, kus hallatakse sissetulevaid HTTP-päringuid (GET, POST, PUT, DELETE) ning tagastatakse vastuseid JSON-formaadis. Kolmandaks on *template* ehk mall, mis kujutab endast HTML-i või muu vormingu põhja, kuhu View kaudu töödeldud andmed sisestatakse. See on seotud kasutajaliidese ehk frontend'iga ning võimaldab dünaamiliste lehtede loomist.

PostgreSQL töötab Django rakenduse all oleva andmepangana. Django ORM (Object-Relational Mapping) võimaldab Pythonis defineeritud mudeleid automaatselt tõlkida SQL-iks ning teha keerukaid päringuid ilma otseselt SQL-i kirjutamata. PostgreSQL on valitud tänu oma stabiilsusele, skaleeritavusele, suurepärasele andmetüüpide toele (nt JSONB, ARRAY, UUID) ja täistekstiotsingu võimalustele. See muudab selle sobivaks ka keerukamate rakenduste jaoks, kus andmemahud ja seoste keerukus kasvavad.

3.1.1 Andmemudelid

Django raamistikus kasutatakse andmemudeli loomiseks ORM-i ehk objekti põhist mudelit, mis tähendab, et ei kirjutata SQL-iga tabelid ning päringud. Objekti põhine mudel tähendab, et mudelid kirjutatakse Pythonis klassidena, ning kõik päringud on muudatused nendel

klassidel. Klasside loomise ning muudatuste päringute loomise teeb Django ise, see on kasulik turvalisuse poolest, sest nii saame vältida levinud *SQL injection* turvaauku ning palju kiirem, sest SQL päringuid ei pea ise looma. Andmemudeli faili klassid määratlevad, kuidas ORM loob tabelid andmebaasis. Andmemudeli failis on viis klassi ning neil on mitmeid välju, kuid korduvaid klassidevahelisi välju pole töös uuesti kirjeldatud.

Esimene mudel on *ArgumentMap*, mis on kujutus argumendikaardi struktuurist. Argumendikaardi mudelil on pealkiri, kirjeldus, autor, muutjad, ning loomise ja viimati muudetud kuupäeva väljad. Kuna Django loob automaatselt kasutaja mudeli, mida kasutatakse, et jälgida kes tegi veebilehel muudatuse.

```
class ArgumentMap(models.Model):
    title = models.CharField(max_length=255)
    description = models.CharField(max_length=255, blank=True)
    author = models.ForeignKey(User, on_delete=models.CASCADE, related_name="argument_trees")
    contributors = models.ManyToManyField(User, related_name="collaborations", blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

Joonis 6. Argumendikaardi mudel. Allikas: autori koostatud

Teiseks on mudel *Argument*, mis on kujutus argumentidest, mida kasutaja saab teha erinevate väidete toetamiseks või salgamiseks. Argumendi mudelil on sisu, ehk kirjeldus argumendist. *Is_root* ehk_juur määrab ära, kas tegemist on argumendikaardi algväitega. *Argument_map* väli määratleb ära, millise argumendikaardi juurargumendiga tegemist on, kui aga *Argument_map* väli on määramata, siis tegemist ei olegi juurargumendiga.

```
class Argument(models.Model):
    content = models.CharField(blank=True, null=True, max_length=1000)
    author = models.ForeignKey(User, on_delete=models.CASCADE, related_name="arguments")
    is_root = models.BooleanField(default=False)
    argument_map = models.ForeignKey(ArgumentMap, on_delete=models.CASCADE, related_name="arguments", null=True, blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

Joonis 7. Argumendi mudel. Allikas: autori koostatud

Kolmandaks mudeliks on *Connection*, ehk serv, mis seob erinevaid tippe. Iga serv on seotud ühe kindla argumendikaardiga ehk väli *argument_map*, ning ühe argumendi ja ühe operaatoriga ehk vastavalt väljad *source_argument* ja *target_operator*. Igal seosel on ka ära märgitud, kas see seos räägib argumendi vastu või poolt ning on määratud väljas *stance*.

```python
class Connection(models.Model):
    author = models.ForeignKey(User, on_delete=models.CASCADE, related_name="connections")
    explanation = models.CharField(max_length=50, null=True)
    argument_map = models.ForeignKey(ArgumentMap, on_delete=models.CASCADE, related_name="connections")
    source_argument = models.ForeignKey(Argument, on_delete=models.CASCADE, related_name="source_connections")
    target_operator = models.ForeignKey(Operator, on_delete=models.CASCADE, related_name="target_connections")

    STANCE_CHOICES = [
        ('against', 'Against'),
        ('for', 'For'),
        ('undefined', 'Undefined'),
    ]
    stance = models.CharField(
        max_length=10,
        choices=STANCE_CHOICES,
        default='undefined',
    )

    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

Joonis 8. Ühenduse mudel. Allikas: autori koostatud

Neljandaks on mudel *Operator*, ehk operaator, mis määrab üldisemalt, kuidas kogum argumente määrab mingi väite tõeväärtuse. Kui argument on eraldiseisev, ehk operaatoris on ainult üks argument, siis teame, et argumendikogum kehtib kui see argument kehtib. Kui aga operaatoriga on seotud mitu argumenti, siis nende vahel on VÕI operaatorid, ehk et kogum argumente kehtiks, peab iga argument kehtima. Operaatoritega seotud argumentide haldamiseks on väli *arguments* ning operaatori tüübi ehk JA või VÕI seisundi määramiseks on väli *operator_type.*

```python
class Operator(models.Model):
    OPERATOR_TYPE_CHOICES = [
        ('AND', 'AND Operator'),
        ('OR', 'OR Operator'),
    ]
    argument_map = models.ForeignKey(ArgumentMap, on_delete=models.CASCADE, related_name="operators")
    arguments = models.ManyToManyField('Argument', related_name="operators", blank=True)
    operator_type = models.CharField(max_length=3, choices=OPERATOR_TYPE_CHOICES, blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

Joonis 9. Operaatori mudel. Allikas: autori koostatud

Viiendaks on mudel *Log*, ehk logi, mis jätab meelde kõik muudatused argumendikaardis ning seob need muudatused kindla kasutaja, argumendikaardi ning muudatuse tüübiga. Väljad *user* ja *argument_map* seob ära muudatuse tegija ning muudatuse koha. Väljad *content_type* ütleb, mis mudeli objekti muudeti, *object_id* ütleb, mis selle mudeli objekti id on ja *target* ühendab need kaks ja annab meile selle objekti, mida muudeti. Väljad *change_type* ja *change_description* on tekstilised kirjeldused muudatusest, kus esimene on programmis muudatuse mõistmiseks ning teine on kasutajasõbralik kirjeldus. Väljad *data_before* ja *data_after* on vastavalt muudatuse tagasi võtmiseks ja tagasi võetud muudatuse taastamiseks.

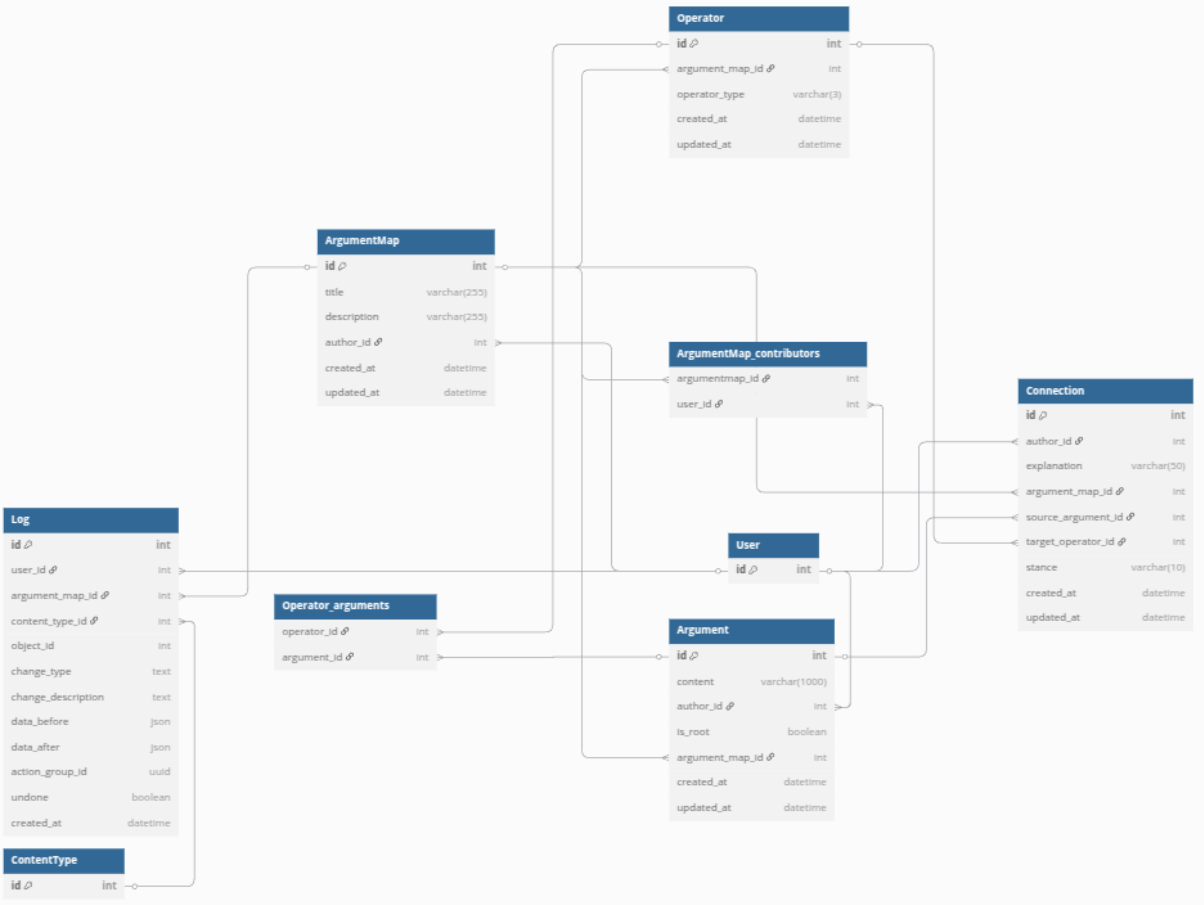
Väljad *undone* ütleb meile, kas muudatus on juba tagasi võetud ning *action_group_id* on identifikaator, mis seob mitu muudatust üheks tervikuks. See on kasulik, sest kui kasutaja kustutab tipu ja see toob endaga kaasa veel servade ja operaatorite muutmise, siis kui tagasi võtta ei muutuks tagasi kõik mainitud muutused vaid ainult kõige viimane, see aga oleks väga segadusse ajav kasutajale ning võib tekitada ettenägematuid probleeme programmis.

```python
class Log(models.Model):
    user = models.ForeignKey(User, on_delete=models.SET_NULL, null=True, blank=True)
    argument_map = models.ForeignKey("ArgumentMap", on_delete=models.CASCADE, related_name="logs")

    content_type = models.ForeignKey(ContentType, on_delete=models.CASCADE)
    object_id = models.PositiveIntegerField()
    target = GenericForeignKey('content_type', 'object_id')

    change_type = models.TextField()
    change_description = models.TextField()
    data_before = models.JSONField()
    data_after = models.JSONField()
    action_group_id = models.UUIDField(default=uuid.uuid4, db_index=True)
    undone = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
```

Joonis 10. Logi mudel. Allikas: autori koostatud



Joonis 11. Mudeli diagramm. Allikas: autori koostatud

Kokkuvõttena, igal argumendikaardil on servad ning üks juurargument. Servad näitavad ära, mis on kindla argumendikaardi operaatorid ja näitavad, kas serv pooldab endaga seotud argumenti või mitte. Operaatorites on kirjas, millised argumendid on vajalikud, ning mis loogiline tehe neid seob. Kõikide argumendipuudes tehtud muudatuste jälgimiseks ning tagasi võtmiseks on logi muudatustest.

3.1.2 Andmevaade

Rakenduses on määratletud URL-ide fail, kus on kirjas kõik võimalikud veebilehe ja API otspunktid, millele saab kasutaja või muu süsteem saata päringuid. Need päringud võivad olla suunatud konkreetsele leheküljele, käivitada muudatusi andmebaasis või lihtsalt pärida andmeid. Iga URL suunatakse kindlasse vaatesse (view), mis analüüsib saabunud päringu sisu ja otsustab, kuidas sellele vastata.

Kui kasutaja külastab mingit veebiaadressi, edastatakse tema HTTP-päring vaatefunktsioonile või klassile, kus toimub kogu seotud äriloogika. Vaade võib teha andmebaasist päringuid, valideerida sisendit, töödelda andmeid või kontrollida kasutaja õigusi. Seejärel valmistab vaade ette vastuse – kas HTML-malli näol (kui tegemist on tavalise veebilehega) või JSON-andmetena, kui tegu on API-päringuga. Näiteks, kui kasutaja soovib näha oma argumentide kaarte, saadetakse päring vastavasse vaatesse, mis hangib andmebaasist kõik selle kasutaja argumendikaardid ja edastab need kas HTML-lehele või API kaudu JSON-formaadis.

Kui tegemist on API-päringuga, kasutatakse Django REST Frameworki puhul spetsiaalset komponenti – *serializer*'it. *Serializer* on tööriist, mis teisendab Django mudelid JSON-andmeteks ja vastupidi. See on oluline, sest meie rakenduses suhtleb eesliides (frontend), mis on loodud Reacti ja TypeScriptiga, Django tagaliidesega JSON-formaadis. Django ORM-is olevad andmed esinevad Python Dictionary kujul ning need tuleb *serializer*-i abil muuta masinloetavasse ja universaalselt kasutatavasse JSON-kujule, et need sobiksid veebiliidese töötlemiseks. Samuti valideerib *serializer* sisendandmed enne, kui neid andmebaasi salvestatakse. Lisaks sellele võivad vaated teha kasutaja õiguste kontrolli (nt kas kasutajal on õigus teatud andmeid muuta), suunata kasutajaid edasi, töödelda vormide sisendeid või hallata veateateid. Django toetab nii funktsioonipõhiseid vaateid kui ka klassipõhiseid vaateid (CBV – Class-Based Views), mis võimaldavad loogikat paremini taas kasutada ja struktureerida. Seega mängivad vaated Django MVT struktuuris võtmerolli: nad vahendavad

andmete liikumist kasutaja, andmebaasi ja kasutajaliidese vahel, olles tihedalt seotud nii mallide (templates) kui ka mudelite (models) toimimisega.

3.1.3 Andmemall

Kolmas osa Django MVT struktuurist on *Template* ehk andmemall. Andmemall on html kujutis veebilehest, kuhu sisestatakse andmed vastavalt vaatele ning kasutajale. Malli algeks tegin base.html, seda kasutame, et kõik teised lehed luua, vältides seeläbi liigset koodi kordumist. Põhilised mallid on home, list_all_argument_maps, list_all_user_maps ja view_argument_map. View_argument_map on eriline, sest see mall on sisendpunktiks React staatilistele failidele ehk need failid, mis visualiseerivad React Flow abil argumendipuusid.

3.2 Eesliides

React-i võtsin ainult seetõttu, et React-Flow teek, mida kasutan argumentatsioonipuude visualiseerimiseks, vajab seda. Postgres on väga laialdaselt soovitatud ning lihtne andmebaas, mida ülesseada ning kasutada. Kuigi Django on piisav, et teha kõik eesliidese arenduse, sellega tegin näiteks kõikide argumendikaartide vaate ja avalehe vaate, oleks tõenäoliselt lihtsam olnud kasutada siis juba kogu eesliidese ulatuses React-i, mis oleks lihtsustanud projekti struktuuri ning väiksem raskus minna argumendikaardi redigeerimise vaate arendamisest, mis on React-is, üle avalehevaate arendamisele, mis on Djangos.

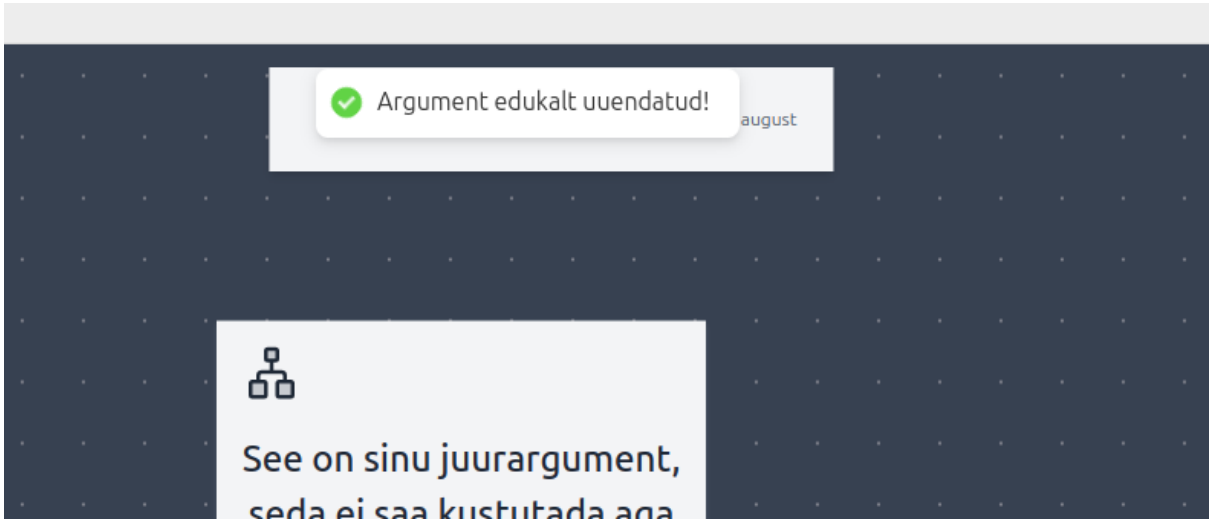
3.2.1 React Flow

Argumentatsioonipuu visualiseerimiseks kasutatakse React Flow teeki, mis põhineb Reactil. Seetõttu on argumentatsioonipuu rakenduses eraldi React projekt, mille staatilised failid ehitatakse Vite abil. Vite-ga loodud staatilised failid serveeritakse Django kaudu andmemalli sisse ning mall läheb omakorda edasi kasutajale. React Flow's on kaks peamist elementi, millega kujutatakse graafi. Esiteks, tipud, mis kujutavad, kas argumente või operaatoreid, nende objektide kirjeldus on eelnevalt andmemudelis välja toodud. Teiseks, servad, mis kujutavad ühendusi nii argumentide kui ka operaatorite ehk siis erinevate tippude vahel. Operaatori ja argumendi tipu jaoks on tehtud eraldi komponendid, mis kujundavad, kuidas need välja näevad, ning mis omadused neil on.

3.2.2 Lisatud teegid

Tippude ja servade haldamiseks kasutatakse Zustand-i *store* ehk andmeladu, mis on kerge ja lihtne olekute haldamise teek. Andmeladu haldab argumendikaardi, tippude, servade ja otsinguvaate olekut erinevate komponentide vahel. Kuna kasutaja jaoks võiks olla

võimalikult lihtne argumendikaarti luua, siis tippude asetsemine toimub automaatselt kasutades Dagre teeki, ning seda rakendab minu poolt tehtud funktisoon applyDagreLayout. Funktsioon tekitab Dagre graafi ning kuna Dagre paneb nende asukohad ise paika, siis saame võtta selle graafi tippude asukohad ning panna selle enda graafile. Dagre aga positsioneerib enda tipud, kas järjekorras, kuidas talle antakse või kui graaaf on paremal poolel suurem, siis pöörab ta selle ümber, eelistades vasakule poole kaldu olemist. Dagre omadus kallutada vasakule tekitab kaks probleemi. Esiteks, kui on grupi tipp kahe lapsega ning lisatakse lapsetipp parempoolsele enne kui vasakpoolsele tipule, siis servad lähevad üksteisega risti, ning servade lugemine on raskendatud. Seetõttu, tuli teha selline süsteem, et servad järjestatakse, nii, et grupis olevate tippude servad lisatakse vasakult paremale. Teine probleem on, et kui kasutaja lisab argumendi siis võib järsku osa või kogu graafi harud pooli vahetada ning kasutajal ei saa seetõttu tekkida tunnetus argumendikaardi arugmentide asetuse osas. Harude vahetumise probleemi lahendamiseni ei jõutud, probleem funktsionaalsust ei mõjuta aga vähendab kasutaja mugavust. Leidsin, et väga tähtis on anda kasutajale ka pidevat tagasisidet sellest, kas tema muudatused olid edukad või mitte, selleks kaasasin projekti teegi react-toast, mis aitab väga lihtsalt tekitada teavitusi eduka või edutu muudatuse korral.

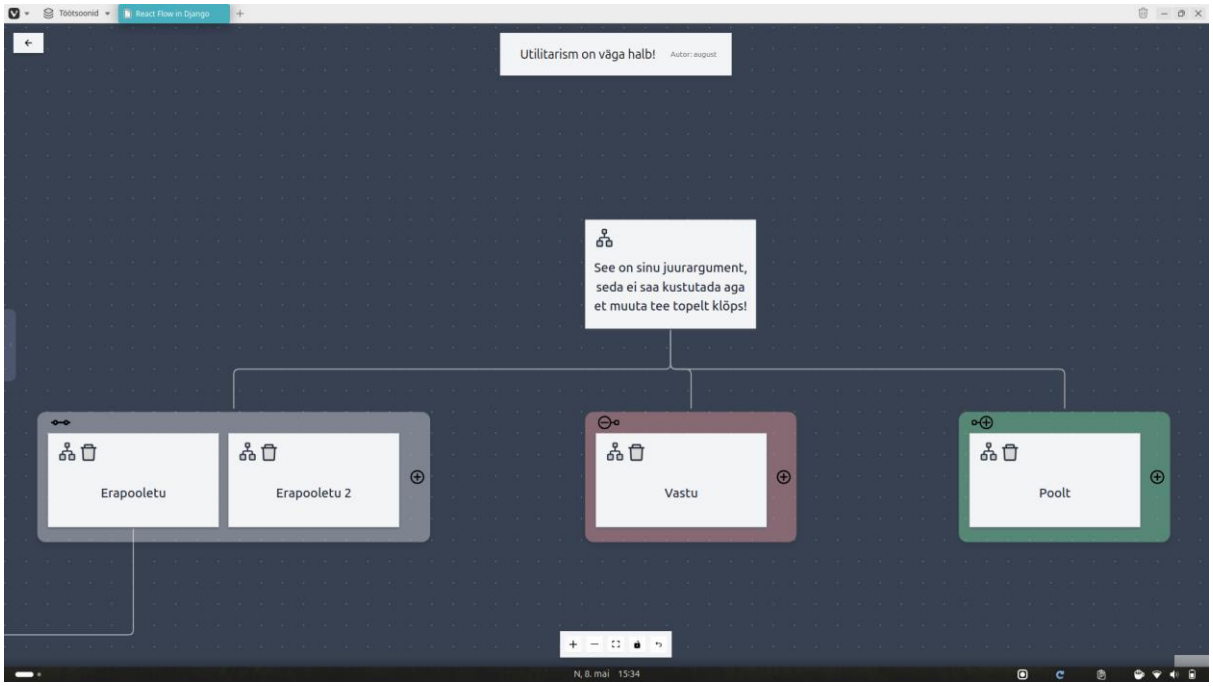


Joonis 12. Kasutajale muudatuste puhul tagasiside andmine. Allikas: autori koostatud

3.2.3 Välimus

CSS-i kirjutamise lihtsustamiseks nii Django andmemallides, kuid ka React-is kasutasin Tailwindcss-i teeki. Välimuse kujundamiseks kaasasin lisaks endale veel inimesi, et saada tagasisidet, mis välimuse juures töötab ning mis on probleemiks. Tegin väikese valimiga küsitluse, milles osales 10 inimest, kus palusin neil hinnata erinevaid ekraanipilte minu rakendusest, kuid ka töös analüüsitud veebilehtede disaine. Märkimisväärne oli, et enamus vaatlejate arvamus nii minu lehekülje algsest disainist, kuid ka Mindmup ja Argumentation.io

lehtedest oli väga negatiivne, enamik hindeid kümne palli skaalal olid viis või vähem. Inimesed eelistasid tumedat tausta ning kumeraid servi, algses versioonis olid ka värvilised nupud, kuid kuna argumendid ise on ka värvilised, siis eemaldasin need, sest tagasisidena öeldi, et värvid läksid liiga kirjuks. Kuna fookus peaks olema argumendi tekstil, siis jätsin need valgeks ning värvilised ümbrused tegin läbipaistvaks, mis pakkus mitu head omadust. Esiteks, saab vältida argumendilt tähelepanu eemale suunamist, kuid on ikka väga lihtne näha, kas argument on poolt, vastu või erapooletu. Teiseks, kui argumendid lähevad pikemaks, siis muutub raskeks näha, kuhu selle argumendi servad lähevad, kuna aga ümbris on läbipaistev, siis serva kogu teekond on nähtav. Praktilise osa üks suurematekst arenguteks osutus argumentatsioonikaardi välimuse arendamine. Välimus on kindlasti tähtis, et kasutajad oleksid tööriista suhtes vastuvõtlikumad ning teema rohkem kutsuv, kuid ka nagu algselt mainiti, peab argumendikaardi disain lihtsustama argumentide ja nende ühenduste hoomamist.



Joonis 13. Lõplik argumentide kujundus. Allikas: autori koostatud

3.3. Kokkuvõte

Rakendus on peamiselt üles ehitatud Django baasil, välja arvatud argumendikaardi visualiseerimine, mille tegin React-iga React Flow teegiga, kuid ka see osa serveeritakse kasutajale läbi Django andmemaalli. Django on struktuurilt MVT, kus andmemudelid

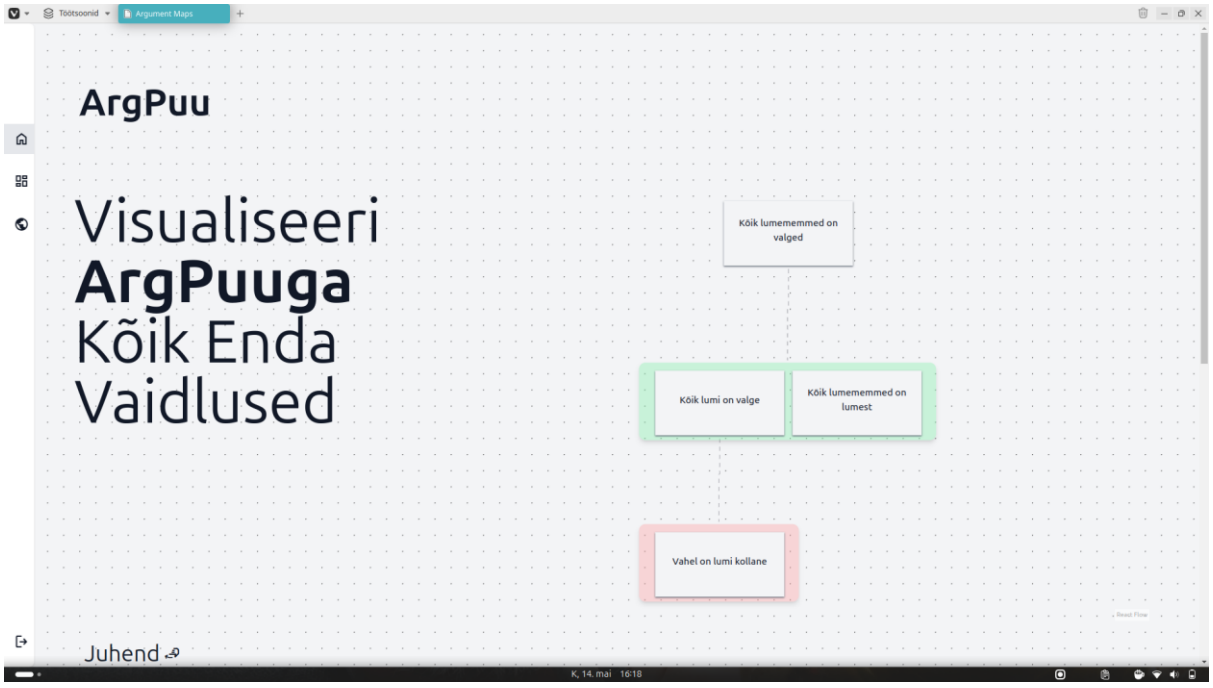
kujutavad andmebaasi tabeleid. Andmevaade vastutab äriloogika eest ning reageerib päringutele. Andmemall paneb paika vaate struktuuri ning andmevaade süstib sellesse andmed, mis seejärel serveeritakse kasutajale. PostgreSQL valik tagab suure jõudluse ja paindlikkuse, samas kui Django ORM lihtsustab andmebaasiga suhtlemist. Kasutasin React-Toast-i, et lihtsustada kasutajale tagasiside andmist. Zustandit, et teha komponentide vahelised andmelaod. Tailwindcss-i, et kujundada rakenduse välimus.

4. Veebileht

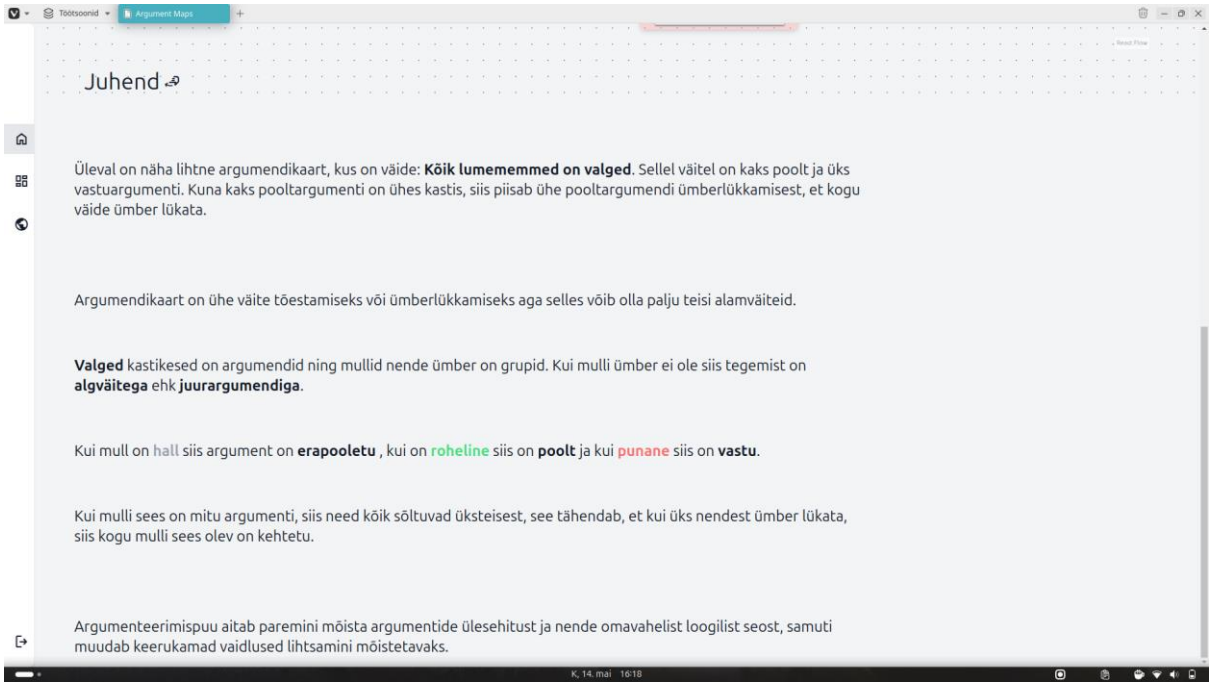
Lõputöö praktilise osana valmis veebileht, kus on 4 peamist vaadet. Esiteks kodu vaade, kus selgitatakse Argpuu eesmärki ning kasutust. Teiseks kasutaja argumendikaartide vaade, kus kasutaja saab hallata enda argumendikaarte. Kolmandaks kõikide kasutajate argumendikaartide vaade, kus on kõik kaardid, mis on autori poolt märgitud kui avalik. Viimaseks on argumendikaardi redigeerimise vaade, kus saab muuta argumendikaardi argumente, servasid ja muud struktuuri.

4.1 Peamise kasutusjuhtumi kirjeldus

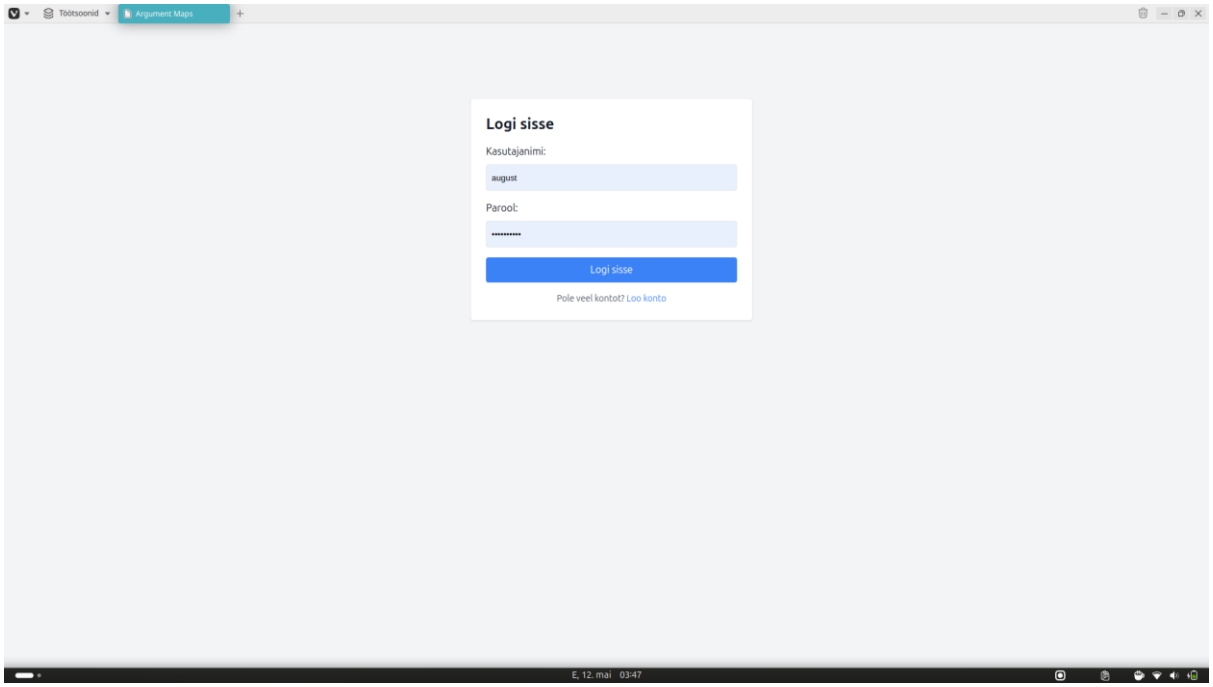
Töö praktiline osa saavutas põhilised eesmärgid. Kasutaja peamine kasutuskäik on selgitatud lahti järgnevalt: kasutaja saab teha endale konto, kasutaja loomise vaates. Kui konto on tehtud, siis näidatakse kasutajale enda argumendikaartide vaadet. See on esmasel käivitamisel tühi ning teavitatkse kasutajat, et ühtegi argumendikaarti ei ole veel loodud. Selle loomiseks on kasutajal sinine nupp pealkirjaga “Tee uus kaart” (vaata joonis 14).



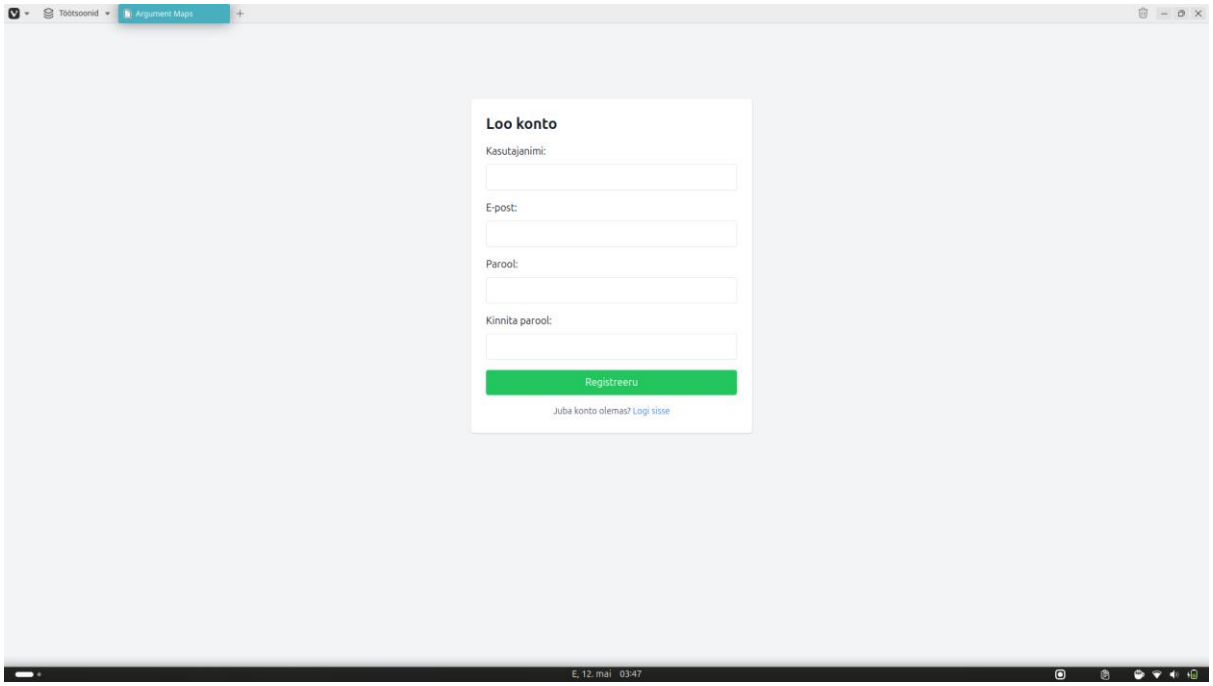
Joonis 14. Avalehe vaade, ehk vaade, mida kasutajale esimesena näidatakse. Allikas: autori koostatud



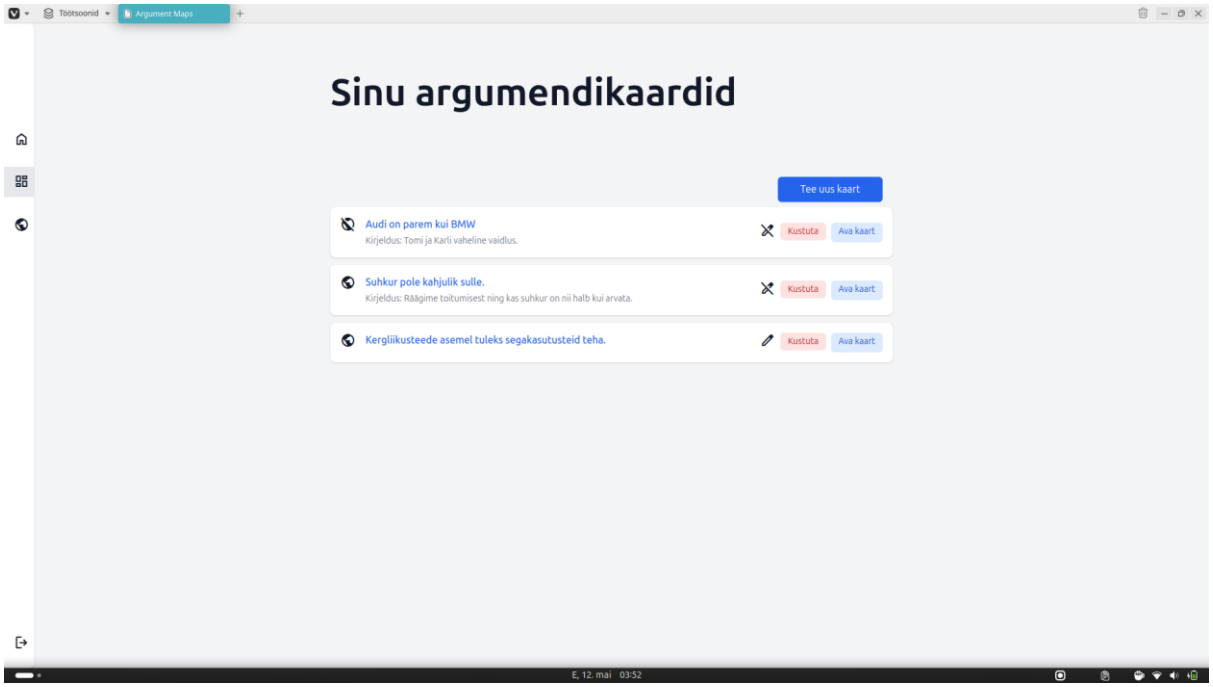
Joonis 15. Avalehe vaate alumises osas olev juhend. Allikas: autori koostatud



Joonis 16. Sisse logimise vaade. Allikas: autori koostatud



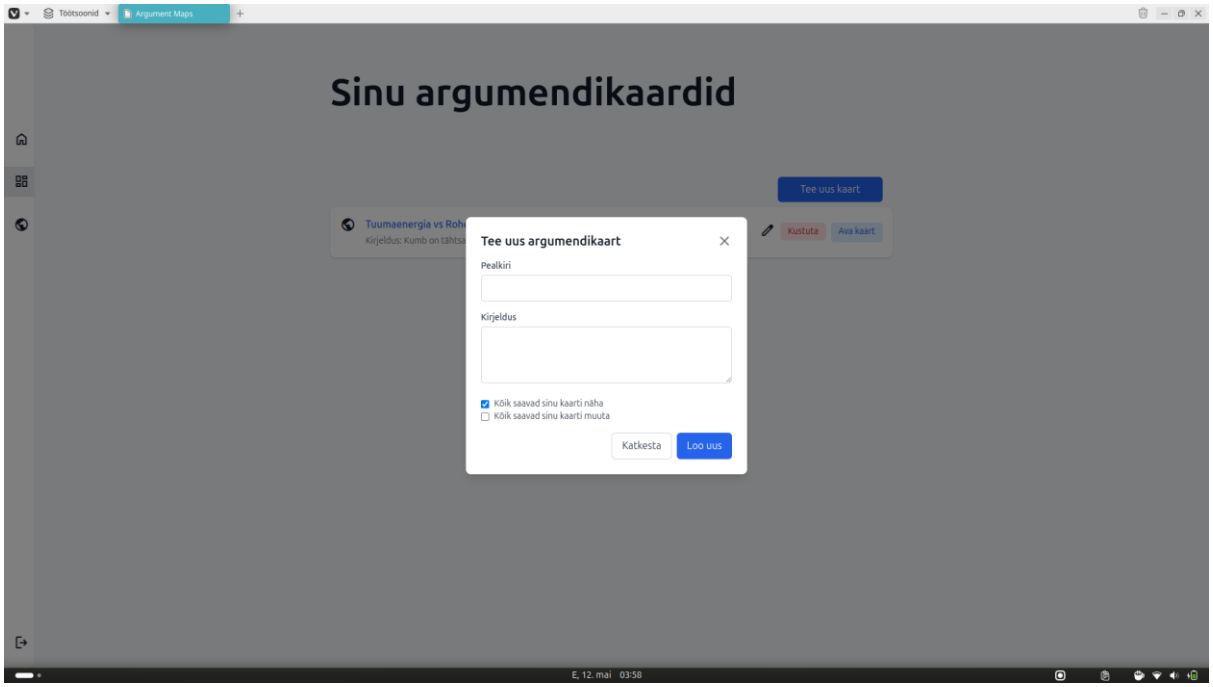
Joonis 17. Konto loomise vaade. Allikas: autori koostatud



Joonis 18. Kasutaja argumendikaartide vaade. Allikas: autori koostatud

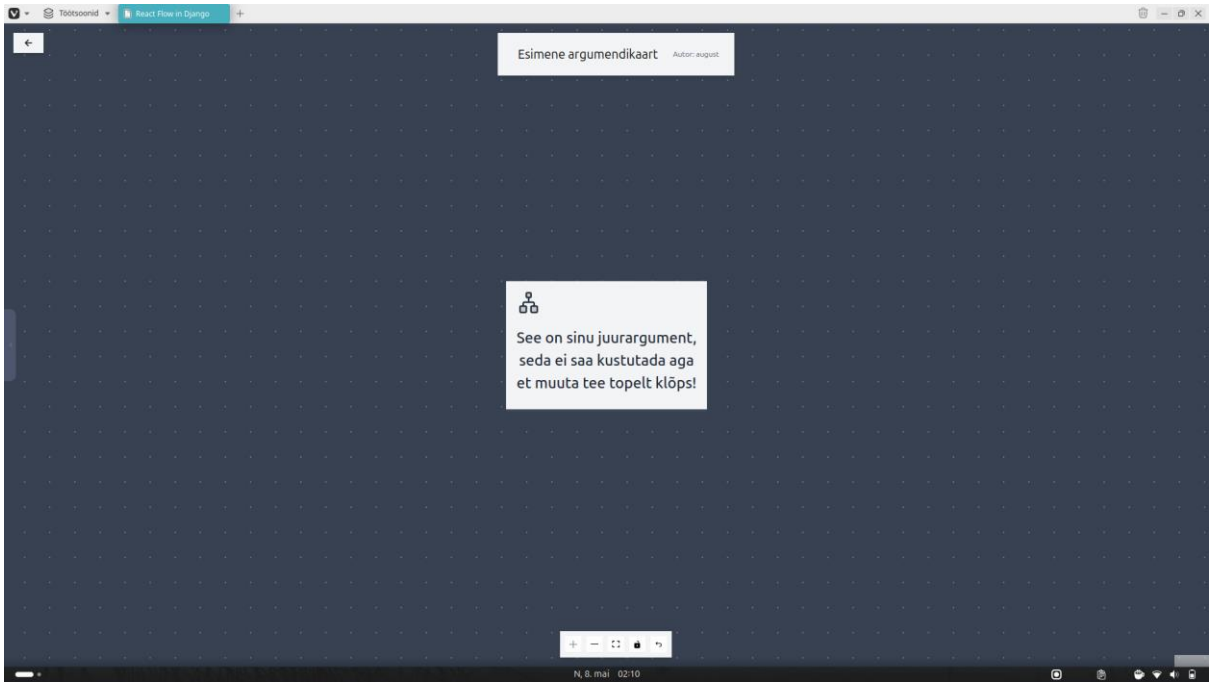
Kui kasutaja vajutab uue kaardi tegemise nuppu, siis näidatakse talle uue argumendikaardi loomise vaadet, kujutatud joonisel 19. Selles vaates tuleb kirja panna lühike pealkiri, kuid on võimalik ka sisestada pikem kirjeldus, mille üle argumendikaardis arutletakse, kelle kohta see käib, mis eesmärgiks on või muu. Lisaks saab kasutaja valida, kas kõigil on ligipääs sellele arutelule, kui jah siis saab kasutaja veel valida, kas kõik saavad seda muuta. Sinise “Loo uus”

nupu vajutamisel tekitatakse uus tühi argumendikaart ning kasutaja viiakse selle redigeerimise vaatesse.

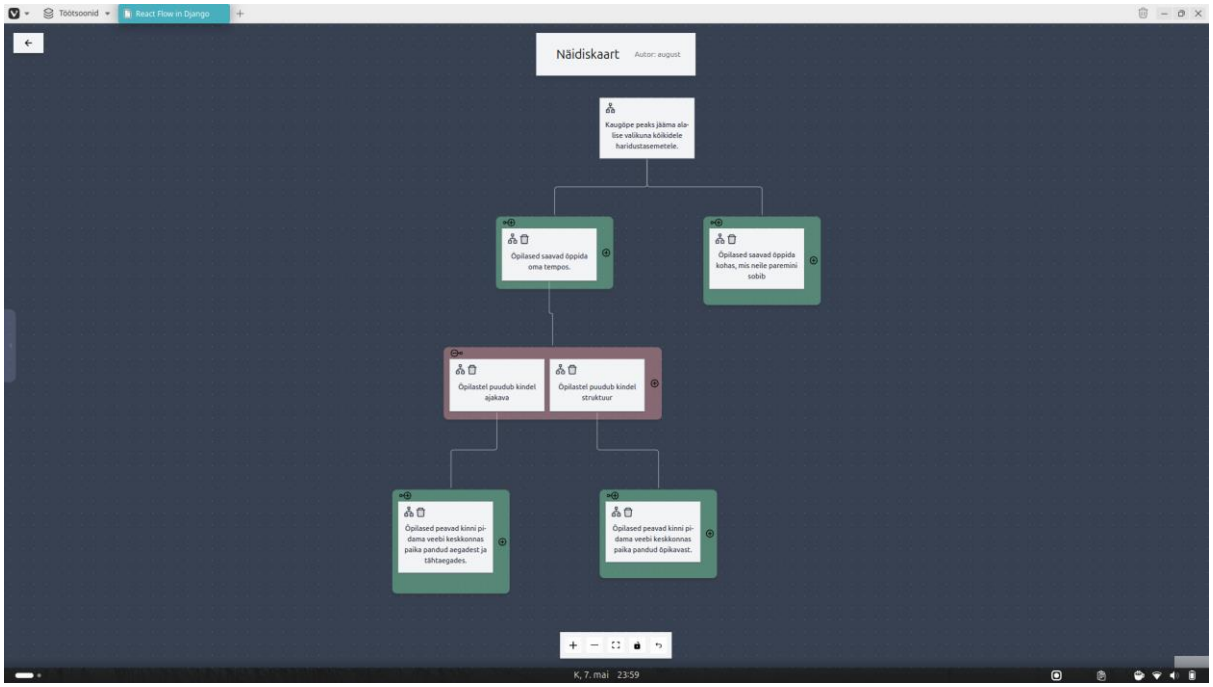


Joonis 19. Argumendikaardi loomise vaade. Allikas: autori koostatud

Argumendikaardi redigeerimise vaates on kasutajale juba ette loodud argument, kuhu on sisestatud juhendtekst, mis soovitab kasutajal luua uus argument või muuta juuraargumendi sisu. Juurargument on eriline argument, mida ei saa kustutada, ega lisada kõrvalargumente. Nüüd on kasutajal järgmised valikud. Ta saab minna tagasi, muuta argumendikaardi pealkirja, või luua uue lapsargumendi. Olukorras, kus kasutaja otsustab veel argumente luua tekib tal graaf, kus saab argumente kustutada, ümber nimetada, muudatusi tagasi võtta ja lisada kõrvalargumente, joonisel 20 on välja toodud näidiskaart.



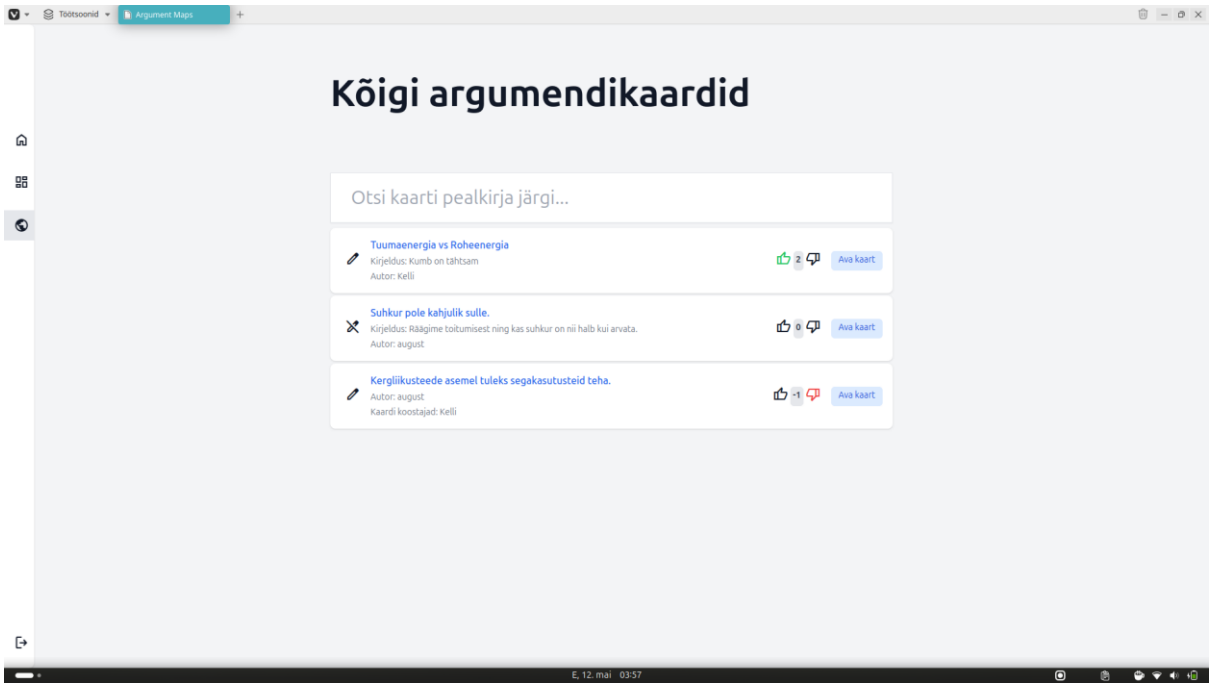
Joonis 20. Argumendikaardi redigeerimise vaade. Allikas: autori koostatud



Joonis 21. Argumendikaardi redigeerimise vaade, näidiskaart. Allikas: autori koostatud

Kõikide kasutajate argumendikaartide vaates saab kasutaja näha erinevaid arutelukaarte, mille puhul looja on märkinud, et kõik saavad seda näha, kujutatud joonisel 22. Argumendikaartide poolt või vastu on võimalik ka hääletada ning poolt- ja vastuhäälte vahe on esitatud numbrina kaardil. Hääletamisega saavad kasutajad avaldada enda arvamust, kui kvaliteetne või õige nende arvates argumendikaardi arutelu on. Iga argumendikaardi juures

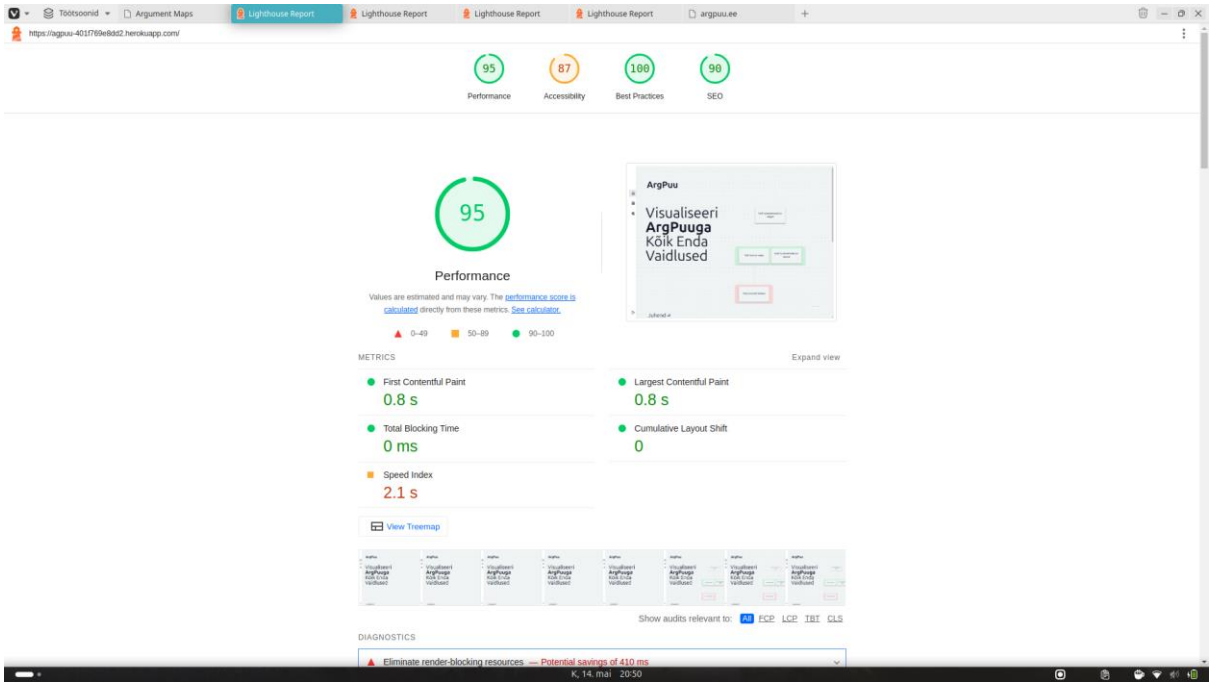
on ka pliiatsi ikoon, kui on lihtsalt pliiats siis tähendab see, et iga inimene või lisada, muuta või eemaldada argumente. Kui aga on maha tõmmatud pliiatsi ikoon, siis saab kaarti muuta ainult selle autor. Välja toodud on ka autor ning muutjad ja kirjeldus kui need leiduvad. Kuna argumendikaarte võib tekkida väga palju, siis sorteerimine on häälte põhjal ning kindla argumendipuu leidmiseks saab kasutada otsingukasti, kus saab kirjelduse, pealkirja või autori järgi sõeluda tulemusi.



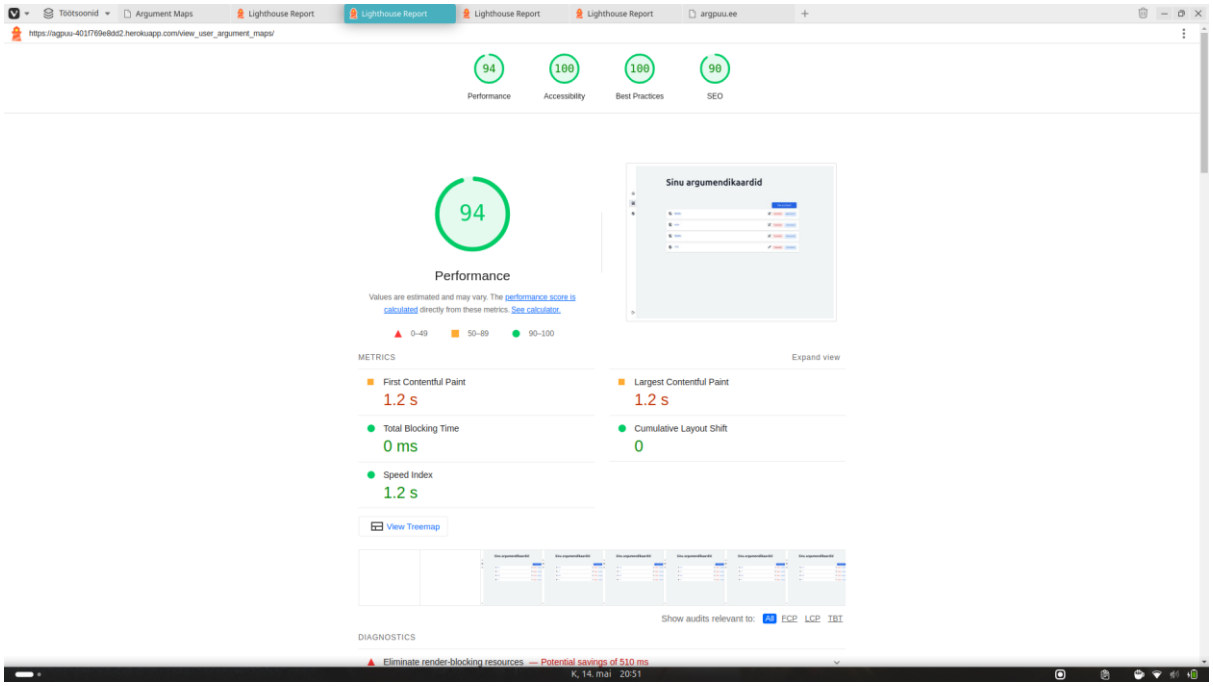
Joonis 22. Kõikide kasutajate argumendikaartide vaade. Allikas: autori koostatud

4.2 Testimine

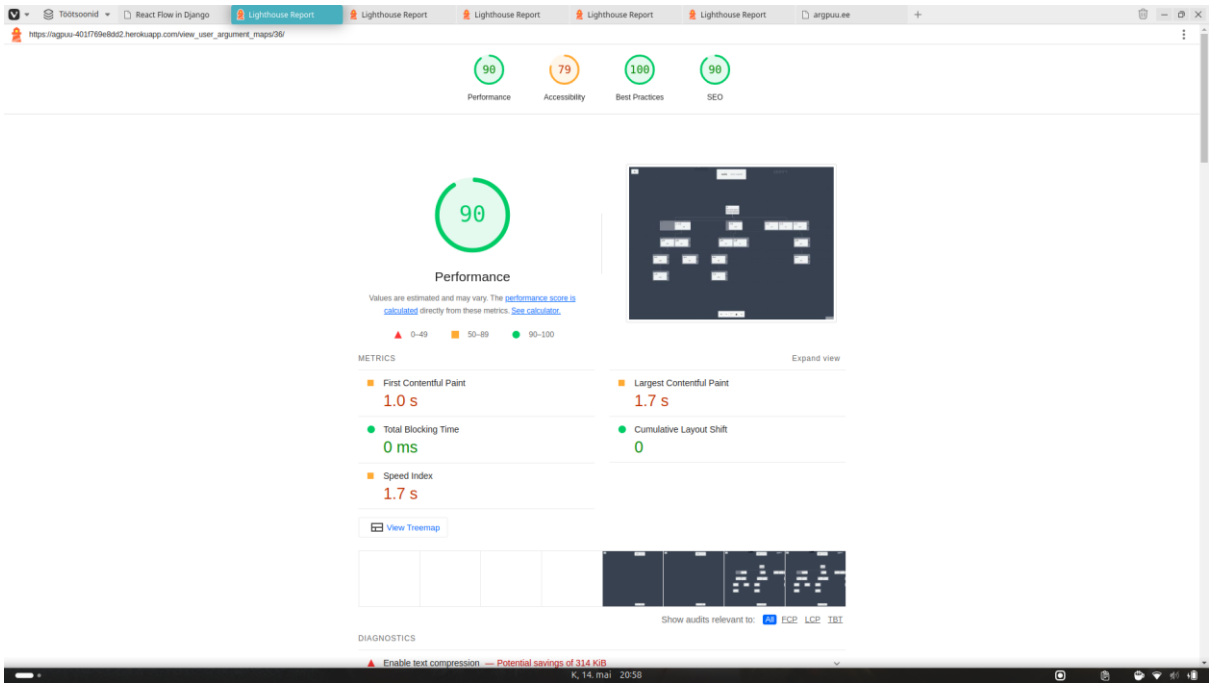
Testimiseks kasutasin Lighthouse-i, see on kasulik tööriist arendajatele, kes soovivad parandada oma veebilehe kiirust, ligipääsetavust, SEO-d ja turvalisust. See aitab leida probleeme ja pakub konkreetseid soovitusi nende parandamiseks. Üldiselt olid tulemused head. Enamus lehetedest laadis kiiresti, kuid argumendikaardi redigeerimise vaate kiirust oleks võimalik siiski kiirendada, et laadimine oleks kohene. Kui lisada argumendikaardile hästi palju argumente, siis see natukene vähendas laadimiskiirust aga see ei olnud märkimisväärne. Lighthouse tõi välja ka paar soovitust, mida laadimise kiirendamiseks. Võiks eemaldada mitte kasutuses oleva JavaScripti koodi. Toodi välja, et kõige suurem aeg kulus argumendi kujutamisele kasutaja poolel, mis oli 1,3 sekundit, samas kui päringu vastusele kulus väga vähe aega, alla poole sekundi.



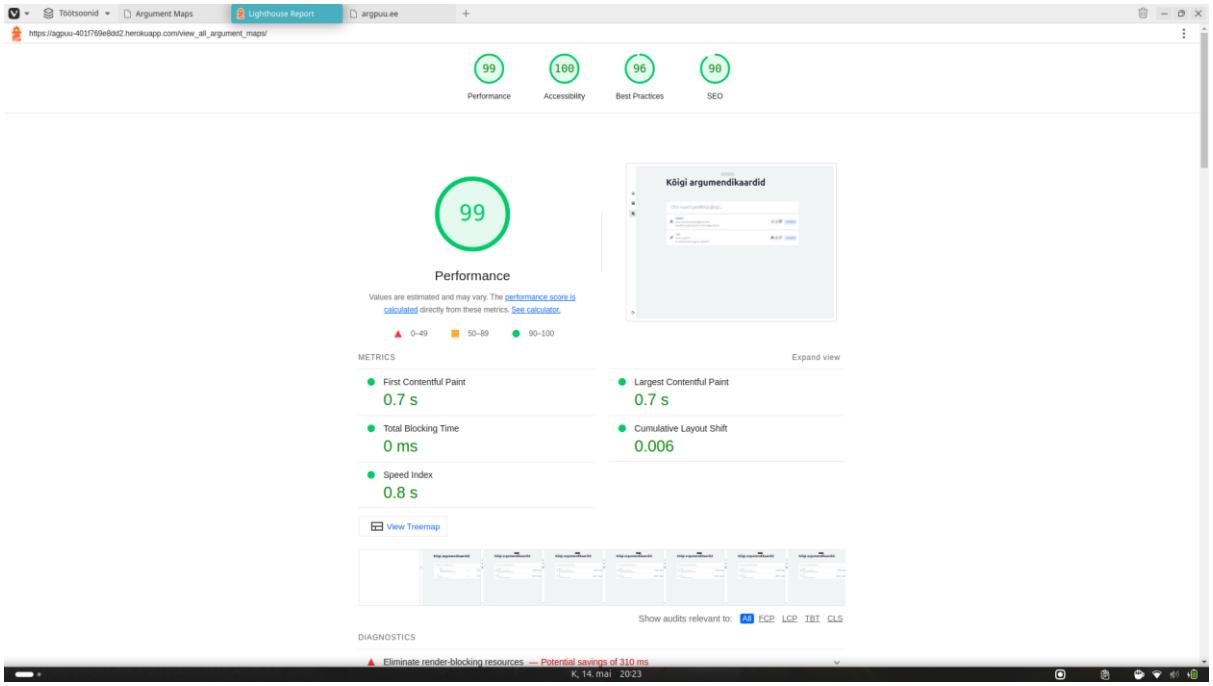
Joonis 23. Lighthouse-i avalehe analüüs. Allikas: autori ekraanitõmmis Google Lighthouse tööriistast (2025)



Joonis 24. Lighthouse-i kasutaja argumendikaartide vaate analüüs. Allikas: autori ekraanitõmmis Google Lighthouse tööriistast (2025)



Joonis 25. Lighthouse-i argumendikaardi redigeerimise vaate analüüs. Allikas: autori ekraanitõmmis Google Lighthouse tööriistast (2025)



Joonis 26. Lighthouse-i kõikide kasutajate argumendikaartide vaate analüüs. Allikas: autori ekraanitõmmis Google Lighthouse tööriistast (2025)

Testimise tulemused olid üldiselt rahuldavad, eriti kuna laadimine toimus piisavalt kiiresti ka suuremate argumendikaartide puhul. Lisaks ei tuvastanud Lighthouse-i analüüsid erilisi probleeme ka teistes meetrikutes.

4.3 Üleslaadimine

Üleslaadimisel võtsin teenuse pakkujaks Heroku, mis on pilvepõhine platvormiteenus (PaaS, *Platform as a Service*), mis võimaldab arendajatel lihtsalt rakendusi hostida, juurutada ja hallata ilma serverite haldamise vajaduseta. Heroku võimaldas üpris kiiresti kogu protsessi käima saada. Tuli luua Herokus Postgres andmebaas ning Django seadetes määrata uus andmebaasi asukoht ja sissepääsuks vajalikud identifikaatorid. Lisaks sellele, tuli luua uus kohalik arendamise keskkond, et üleslaadimine oleks lihtne, kuid säiluks kohalik arendamise võimekus. Veebileht asub aadressil https://www.argpuu.ee ning veebilehe koodi repositooriumi aadress on https://github.com/August-Roosi/Argpuu-Projekt.

5. Tagasiside kasutajatelt ja edasiarenduse võimalused

5.1 Tagasiside kasutajatelt

Tagasiside põhjal oli väga edukas veebilehe kujunduse muudatus, kui lõputöö veebilehe ning töös analüüsitud Argumentation.io ja Mindmup algne hinnang oli keskmiselt alla viie palli kümnest, siis lõpuks oli see üheksa palli kümnest, kus skaala otstes üks ei ole välimuselt kaunis ning kümme on väga kaunis. Kasutajad tõid välja, et avalehel tekkis teatud probleem lehel alla kerimisega, mis tulenes React-Flow omadustest. Soovitustena toodi välja, et võiks olla võimalus mingeid pilte ning graafikuid lisada ja võiks olla automaatne puu haru õigsuse hindamise süsteem. Haru hindamine töötaks nii, kui märkida ära, et ühe haru kõige alumise väitega kasutajad nõustuvad, siis sellest tulenevalt ümber lükatud argumendid muutuvad tuhmiks või muudavad värvi. Kasutajad andsid teada, et argumendikaardi tegemine oli väga lihtne aga üldine kasutajamugavus oli keskmiselt seitse palli, kus skaala otstes üks oli ebamugav ning kümme oli mugav. See tulenes tõenäoliselt eelnevalt mainitud kodulehel kerimise probleemist. Lisaks toodi välja, et kasutajalt ei nõutud kindla raskusastmega parooli, vaid teha sai ka ühe tähelise parooli ning et registreerimisel, kui üks väljadest ei sobi, siis ei sisestata eelnevalt sisestatud väljade sisu vaid tehakse väljad tühjaks. Tagasiside oli väga kasulik ning näitas, et disaini poole pealt oli praktiline osa edukas, kuid et kasutajamugavus vajaks teatud kohendamist ja parandamist.

5.2 Edasine arendus

Praktilise töö käigus valmis väga võimas argumentatsiooni kaartide loomise töörist. Kõige tugevamateks arendusteks oli välimuse ning kasutajate vahelise funktsionaalsuse hõlbustamine. Väga hästi õnnestus süsteemi loomine, kus kasutajad saavad hääletada ning muuta üksteise kaarte. Siiski edasiarenduse võimalusi on mitmeid. Esiteks leidsin seda tööriista arendades, et oleks suur vajadus defineerida ära, mida argumendikaardi lokaalses keskkonnas iga tähtis definitsioon määrab. Tihtipeale on kasutajatel erinev arusaam, mida iga mõiste tähendab ning oleks väga kasulik, kui kõik näeksid ühtselt nende termineid. Teiseks, saaks olla teist liiki argumentatsioonikaart, kaart, kus eesmärk ei oleks kirjeldada vastu ja poolt argumente vaid hoopis kirjeldada mingi väite järelduseni jõudmist, olgu see väide vale või tõene. Selline arutlusvorm oleks palju lihtsam ning kasutajad saaksid iseseisvalt proovida

lahti mõtestada, miks nad mingi väite tõesust usuvad. Kolmandaks, saaks kasutada tehisintellekti, mis aitaks kasutajale vastu ja poolt argumente lisada või luua argumente reaalajas, nii, et kasutaja sisestab järjest väiteid. Selline kasutus saaks olla kasulik samal ajal toimuva kiireloomulise arutelu visualiseerimiseks, näiteks debati puhul. Neljandaks, leian et võiks olla täpsemad juhised või rööpad, mis aitaks kasutajal teha kvaliteetseid argumendikaarte. Viiendaks, saaks the eraldi režiimi või integreerida disaini selline funktsionaalsus, kus kasutaja(d) saavad märkida ära, milliste väidete või argumentidega nad nõustuvad ja ei nõustu ning et selle järgi näeks ära, kas algväide kehtib ning mis loogiline arutluskäik selle järelduseni viis. Lisaks, on hetkel väga suur vastutus kasutajal teada, milline võiks olla hea argumendikaart aga suunis võiks tulla rakenduse poolt, mis juhatab teda struktuurilistel valikutel, näiteks, et, kas väidet saaks osadeks teha. Kasutajate vahelise suhtluse poole pealt veel nii palju, et argumendid võiksid näidata rohkem infot, kes tegi, millal tegi ning võimaldada reaalajas näha teiste muudatusi.

Kokkuvõte

Selle bakalaureusetöö raames analüüsisin olemasolevaid argumentatsioonikaardi loomise veebilehti, tuues välja nende tugevused ja nõrkused ning seejärel püstitasin enda töö eesmärgid. Praktilise osana valmis argumentatsiooni visualiseerimise veebileht. Veebilehe peamised funktsionaalsused on argumentide ja ühenduste loomine, kustutamine ja muutmine, oma teiste kasutajatega, argumendikaartide jagamine, teiste kasutajate argumendikaartide muutmine ning hindamine. Praktilisel osal oli kaks tähtsat arengut võrreldes eelnevatega argumentatsiooni kaartide loomise tööriistadega. Esiteks oli kaardi välimuse parandamine, sest kasutajad olid üldiselt rahulolematud olemasolevate disainidega. Teiseks, argumentatsiooni kaarte saab inimestega jagada, ning lubada neil teha muudatusi. Koostöö võimaldamine oli tähtis, sest see võimaldab inimestel teha koostööd üksteisega ning julgustab elavamat diskussiooni.

Rakenduse tegin suuremas osas Djangoga, kuid argumentatsioonipuu visualiseerimiseks kasutasin React-il põhinevat React Flow-i, mis on loodud graafide tegemiseks. Andmebaasiks valisin PostgreSQL. Edasiarenduseks on mitmeid valikuid, nendest märkimisväärseim on mõistete defineerimine, et kasutajad saaksid sarnaselt aru argumentatsiooni jaoks vajalikest definitsioonidest ja argumentide hindamise režiim, kus lõpuks saaks aru, kas algväide kehtib, ning millise arutluskäiguga sinna jõuti. Tagasisidena kinnitas, et disain oli paranenud, kuid ilmestas, seda, et rakendusel on võimalik veel palju parandusi teha, näiteks toodi välja ebamugavused kasutamisel ning edasiarendused, milleks on eelnevalt mainitud argumentide hindamise režiim. Testimisel leidsin, et veebileht laadis kiiresti ka suuremate argumendipuude korral.

Kasutatud kirjandus

Argumentation.io. https://www.argumentation.io (07.01.2025)

Cullen, S., Fan, J., van der Brugge, E., & Elga, A. (2018). Improving analytical reasoning and argument understanding: a quasi-experimental field study of argument visualization. Npj Science of Learning, 3(1), 1–6. https://doi.org/10.1038/s41539-018-0038-5 (07.01.2025)

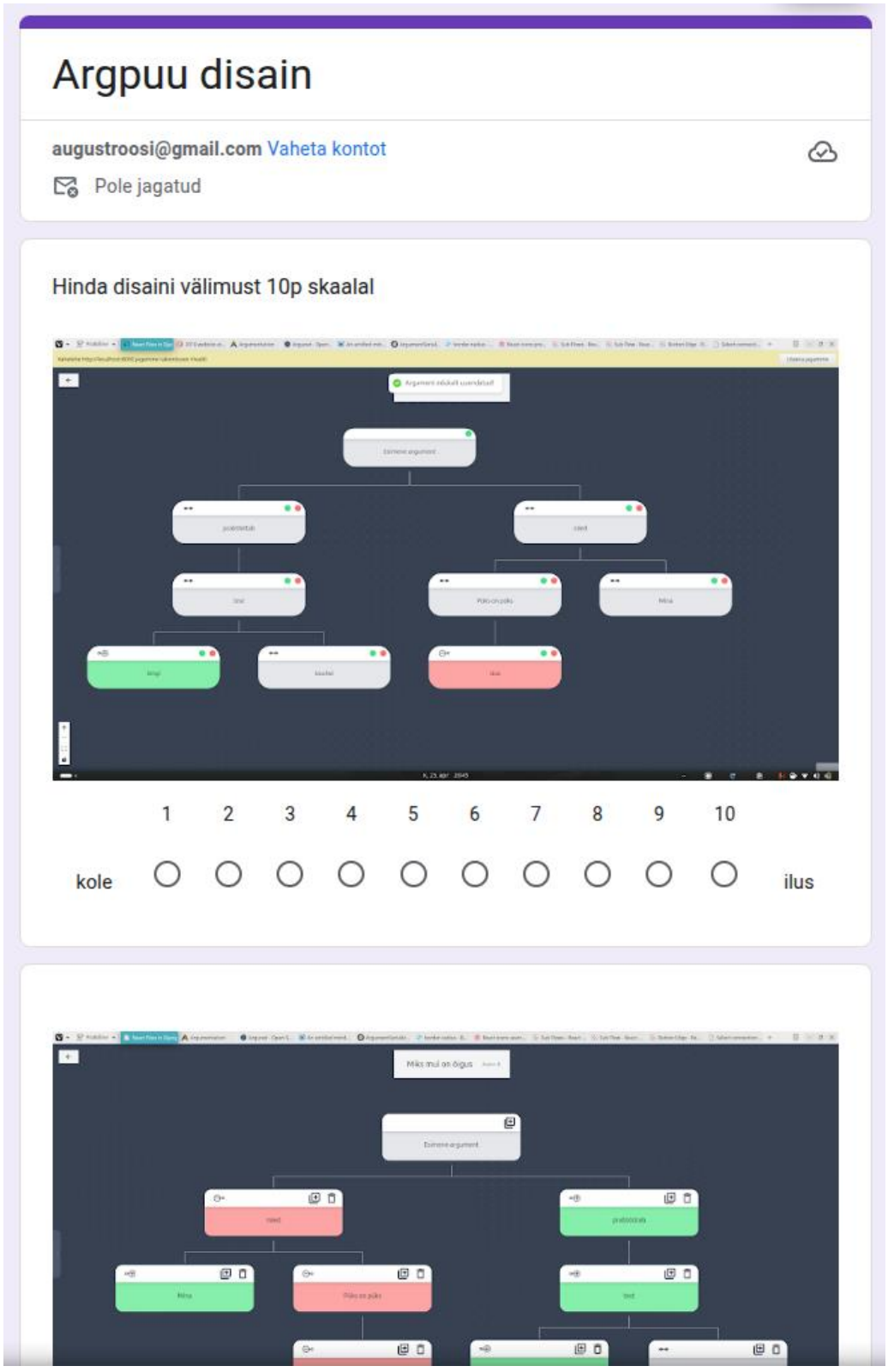
Davies, M., Barnett, A., & Gelder, T. van. (2019). Using Computer-Aided Argument Mapping to Teach Reasoning. Ecampusontario.pressbooks.pub. https://ecampusontario.pressbooks.pub/criticalthinking1234/chapter/introduction/ (07.01.2025)

Mindmup. (2019). GitHub - mindmup/mapjs: JavaScript Mind Map visualisation and management. GitHub. https://github.com/mindmup/mapjs (07.01.2025)

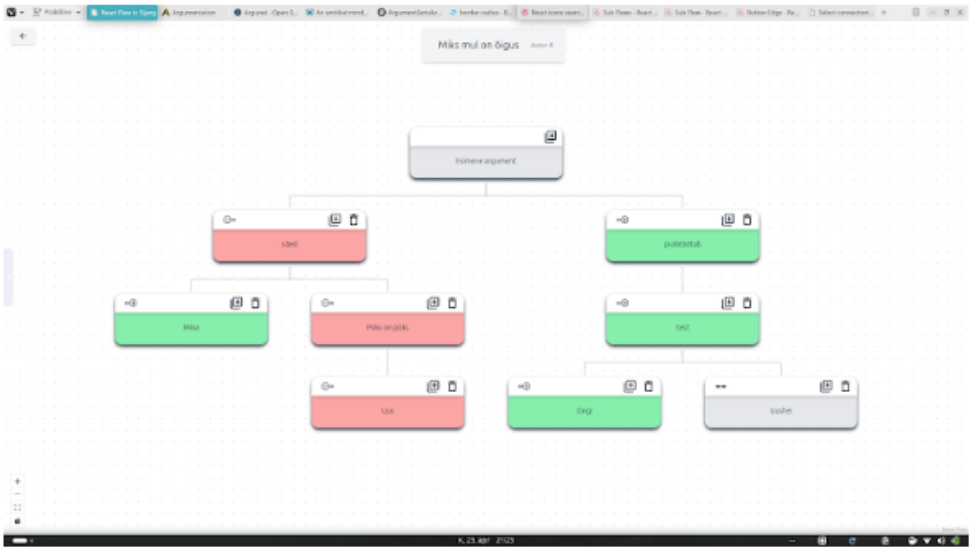
MindMup. https://www.mindmup.com (07.01.2025)

Rathkopf, C. (2024). Some Benefits and Limitations of Modern Argument Map Representation. Argumentation, 38(2), 199–224. https://doi.org/10.1007/s10503-023-09626-5 (07.01.2025)

Lisad 1. Argpuu disaini küsimustik.

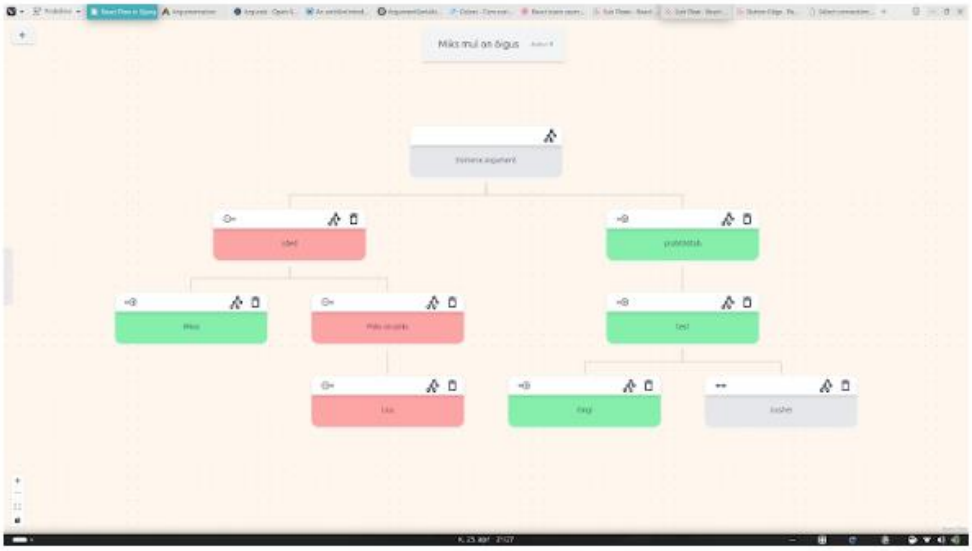


	1	2	3	4	5	6	7	8	9	10	
kole	○	○	○	○	○	○	○	○	○	○	ilus



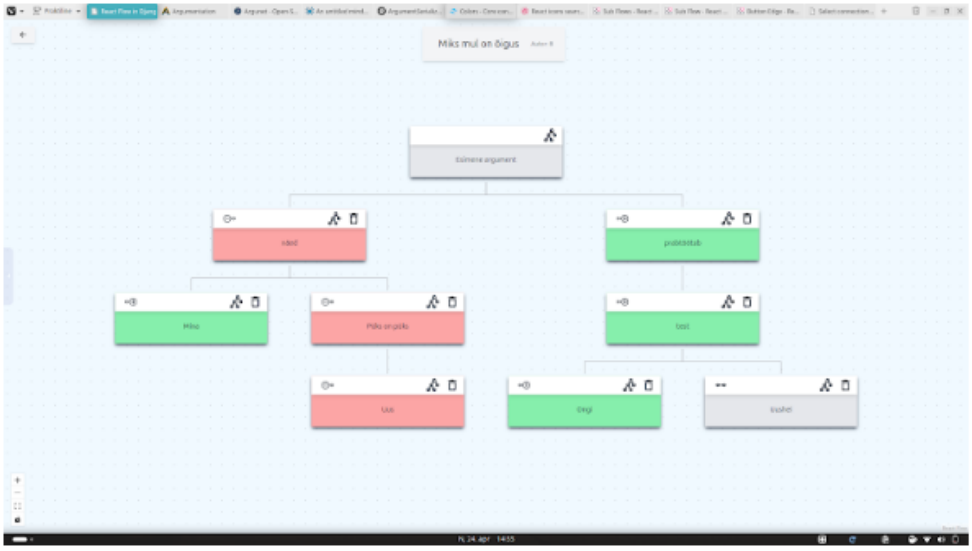
	1	2	3	4	5	6	7	8	9	10	
kole	○	○	○	○	○	○	○	○	○	○	ilus

	1	2	3	4	5	6	7	8	9	10	
kole	○	○	○	○	○	○	○	○	○	○	ilus

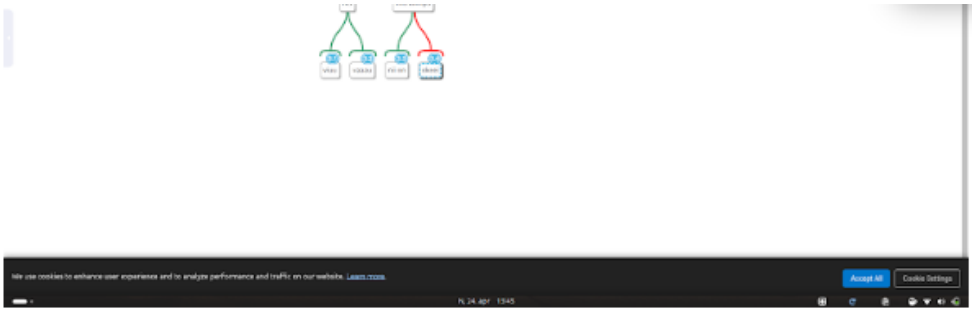


	1	2	3	4	5	6	7	8	9	10	
kole	○	○	○	○	○	○	○	○	○	○	ilus

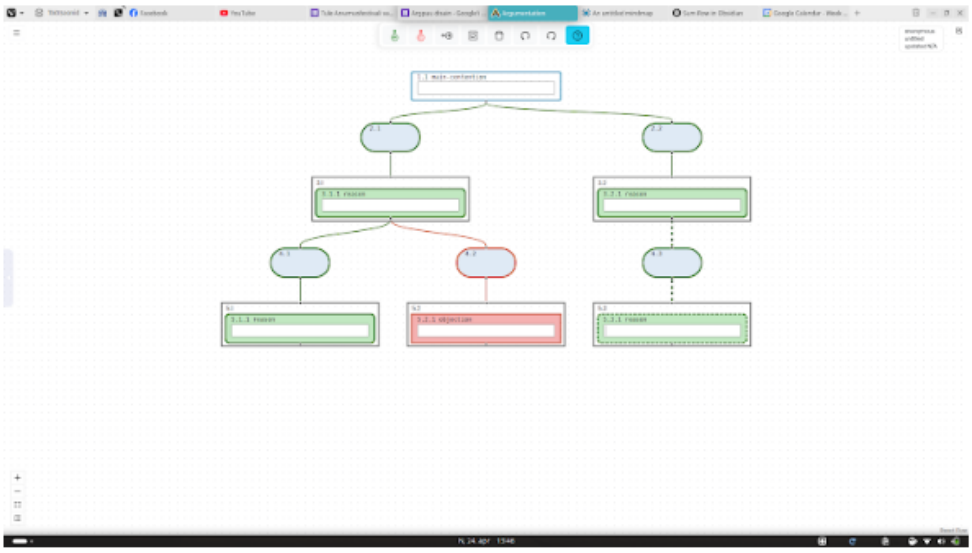
	1	2	3	4	5	6	7	8	9	10	
kole	○	○	○	○	○	○	○	○	○	○	ilus



	1	2	3	4	5	6	7	8	9	10	
kole	○	○	○	○	○	○	○	○	○	○	ilus



	1	2	3	4	5	6	7	8	9	10	
kole	○	○	○	○	○	○	○	○	○	○	ilus



	1	2	3	4	5	6	7	8	9	10	
kole	○	○	○	○	○	○	○	○	○	○	ilus

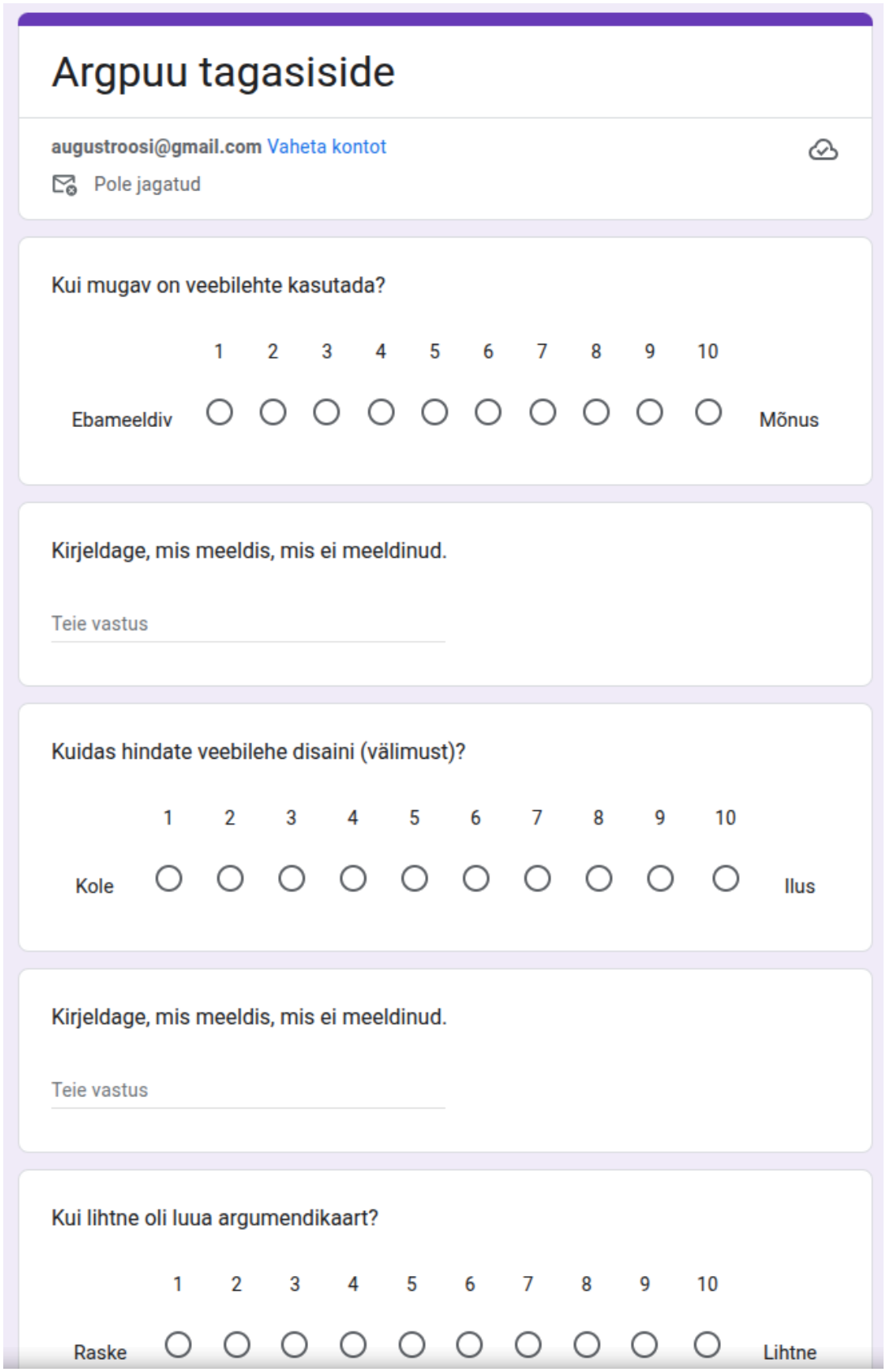
Mis ei meeldinud või meeldis ülemiste piltide juures?

Teie vastus

Saada ära

Tühjenda vorm

Lisad 2. Argpuu tagasiside küsimustik.



Argpuu tagasiside

augustroosi@gmail.com Vaheta kontot

Pole jagatud

Kui mugav on veebilehte kasutada?

	1	2	3	4	5	6	7	8	9	10	
Ebameeldiv	○	○	○	○	○	○	○	○	○	○	Mõnus

Kirjeldage, mis meeldis, mis ei meeldinud.

Teie vastus

Kuidas hindate veebilehe disaini (välimust)?

	1	2	3	4	5	6	7	8	9	10	
Kole	○	○	○	○	○	○	○	○	○	○	Ilus

Kirjeldage, mis meeldis, mis ei meeldinud.

Teie vastus

Kui lihtne oli luua argumendikaart?

	1	2	3	4	5	6	7	8	9	10	
Raske	○	○	○	○	○	○	○	○	○	○	Lihtne

Kirjeldage, mis tegi enda argumendikaardi loomise lihtsaks või keeruliseks.

Teie vastus

Kas midagi jäi veebilehte kasutades segaseks?

Teie vastus

Kas sooviksite, et veebilehel oleks veel mingi funktsionaalsus?

Teie vastus

Selgitage, millal võiks olla argumendikaardil argumendid ühe mulli sees ning millal eraldi mullides.

Teie vastus

Selgitage mida tähistab ühendus kahe argumendi vahel.

Teie vastus

Saada ära

Tühjenda vorm

Litsents

Mina, August Roosi,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose
   "Veebileht argumentatsioonikaartide loomiseks",
   mille juhendaja on Vambola Leping,
   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada Tartu Ülikooli digitaalarhiivi kuni autoriõiguse kehtivuse lõppemiseni;

2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni;

3. olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile;

4. kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

August Roosi
15.05.2025