TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Jürgen Tamm 205982IABB

Tehisintellekti rakendamise väljakutsed ettevõtte Build.Works Technologies OÜ näitel

Bakalaureusetöö

Juhendaja: Karl-Erik Karu
MSc

Tallinn 2024

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: Jürgen Tamm

12.05.2024

Annotatsioon

Lõputöö eesmärk on uurida tehisintellekti rakendamise väljakutseid ettevõtte Build.Works Technologies OÜ näitel.

Selleks luuakse lahendus, mis suudab paremini soovitada ehituselementidele sobivaid tooteid, kasutades tehisintellekti. Toodete soovitamise muudavad keerukaks andmete kvaliteet ja suur hulk erinevaid tehniliste nimede ja parameetritega elemente ja tooteid.

Antud probleemi lahendamiseks arendatakse süsteem, mis kasutab suurte keelemudelite (LLM) võimekust, nagu GPT-3 ja GPT-4, et analüüsida ja soovitada sobivaid tooteid. Lahendus saavutati kombineerides olemasolevaid tehnoloogiaid ja andmetöötlusalgoritme.

Projekti tulemusena suudab uus süsteem tõhusamalt ja täpsemalt soovitada tooteid, mis vastavad ehituselementidele, pakkudes ettevõttele ärilisi eeliseid ja parendatud kasutajakogemust.

Koos praktilise lahenduse arendamisega viidi läbi kvalitatiivne uuring AIRE organisatsiooniga, mis keskendus ettevõtete probleemidele tehisintellekti rakendamisel. Sellest uuringust ning lõputööst järeldavast analüüsist tehti ettepanekuid, kuidas ületada tehisintellekti rakendamise väljakutseid.

Lõputöö on kirjutatud Eesti keeles ning sisaldab teksti 58 leheküljel, 6 peatükki, 23 joonist, 3 tabelit.

Abstract

Challenges of implementing artificial intelligence as an example of the company Build.Works Technologies OÜ

The aim of this thesis is to explore the challenges of implementing artificial intelligence using the example of Build.Works Technologies OÜ. For this purpose, a solution is developed that can better recommend suitable products for construction elements using artificial intelligence. The complexity of product recommendations is increased by data quality issues and the large number of different elements and products with technical names and parameters.

To address this problem, a system is developed that utilizes the capabilities of large language models (LLM), such as GPT-3 and GPT-4, to analyze and recommend suitable products. The solution is achieved by combining existing technologies and data processing algorithms.

As a result of the project, the new system can more efficiently and accurately recommend products that match construction elements, providing the company with business advantages and improved user experience.

Alongside the development of the practical solution, a qualitative study was conducted with the AIRE organization, focusing on the challenges companies face when implementing artificial intelligence. From this study and the analysis derived from the thesis, recommendations were made on how to overcome the challenges of implementing artificial intelligence.

The thesis is in Estonian and contains 58 pages of text, 6 chapters, 23 figures, 3 tables.

Lühendite ja mõistete sõnastik

AI	*Artificial Intelligence* ehk tehisintellekt viitab arvutisüsteemide võimele sooritada ülesandeid, mis tavaliselt nõuavad inimintellekti
AIRE	AI and Robotics Estonia (AIRE) on Eesti tehisintellekti ja robootika arenduskeskus
API	Rakendusliides, komplekt rakendustarkvara ehitamiseks ehk *Application Program Interface*
BIM	Ehitusinformatsiooni mudel ehk *Building Information Modeling* on digitaalne esitus ehitusprojekti füüsilistest ja funktsionaalsetest omadustest
CRUD	CRUD viitab neljale põhitoimingule andmebaasis: loomine(Create), lugemine(Read), uuendamine(Update) ja kustutamine(Delete)
ERP	Ettevõtte ressursside planeerimine ehk *Enterprise Resource Planning* on ärijuhtimise tarkvara
Fine-tune	*Fine-tuning* ehk mudeli häälestamine viitab masinõppemudeli täiendavale treenimisele konkreetsete andmete põhjal
GPT	Tehisintellekti keelemudel ehk *Generative pre-trained transformer* kasutab masinõpet ja andmeid, et luua inimese moodi teksti
IDE	Integreeritud arenduskeskkond ehk *Integrated Development Environment* on tarkvararakendus, mis pakub kõiki vajalikke tööriistu tarkvara arendamiseks ühes kohas
JSON	JSON ehk *JavaScript Object Notation* on lihtsustatud andmevahetusvorming, mis põhineb *JavaScript*'i programmeerimiskeele alamhulgal
JSONL	JSON Lines on tekstiformaat, kus iga rida on eraldi JSON objekt
Kanban	*Kanban* on meetod süsteemide töö haldamiseks, mida kasutatakse tihti agiilsel tarkvaraarendamisel
LLM	Suur keelemudel ehk *Large Language Model* on masinõppemudel
OECD	Majanduskoostöö ja arengu organisatsioon ehk *Organisation for Economic Co-operation and Development*

Prompt	*Prompt* on käsk või sisend, mida antakse tehisintellekti mudelile, et saada soovitud vastus või tegevus
RAG	*Retrieve-and-Generate* on meetod, kus mudel otsib vastuseid suurtest andmekogudest ja loob vastuste saadud teabe põhjal
Sandbox	*Sandbox* ehk liivakast on isoleeritud testimiskeskkond
SKP	Sisemajanduse koguprodukt
Stand-up koosolek	*Stand-up* koosolek, tuntud ka kui püstijalakoosolek, on lühike ja efektiivne koosviibimine, kus osalejad seisavad tavaliselt ringis või grupis
XML	Laiendatav märgistuskeel ehk *Exensible Markup Language* mida kasutatakse andmete kirjeldamiseks ja edastamiseks struktureeritud viisil

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Organisatsioonid teadvustavad üha enam nende käsutuses olevate andmete väärtust, ning nüüd vajavad nad tööriistu nende paremaks kasutamiseks [1]. Tehisintellekt (AI) on tõusev trend mida kasutab Eestis 5% ettevõtetest [2]. Eesti tehisintellekti kasutuselevõtu eksperdirühma aruandes leitakse et „tehisintellekti näol on tegemist alles uudse tehnoloogiaga, mis ei tarvitse veel täna anda organisatsioonile otsest majanduslikku efekti, kuna see on alles arenemisjärgus. Tehnoloogia küpsemist ootama jäädes on aga oht jääda arengus maha“ [3]. McKinsey Global Institute'i hinnangul[1] loob tehisintellekti kasutuselevõtt aastaks 2030 kogu maailmas täiendavalt 13 triljonit USA dollarit lisandväärtust, kiirendades SKP kasvu 1,2% võrra aastas [3]. Käesolev töö uurib tehisintellekti rakendavate ettevõtete väljakutsete kohta Build.Works ettevõtte näitel.

1.1 Probleemi kirjeldus

Väikeettevõtete suutelisus omaks võtta uusi tehnoloogiaid piirab nende väike mastaap, piirates nende võimalusi kasu saada arenevast digitaalmajandusest [4, p. 115]. OECD andmetel on väikeettevõtetel mitmeid takistusi tehisintellekti kasutuselevõtmisel: vähene teadlikkus, andmekultuuri puudumine, ümberõppe vajadus, integreerimise pöördumatud kulud ning vajadus ekstra kulutustele hiljem [5, p. 195]. Tehisintellektil on võime tekitada suuri organisatsioonilisi eeliseid (finants-, turundus- ja administratiiveelised) [1], seda arvestades tekib risk, et väikeettevõtted jäävad üha rohkem taha keskmistest ning suurematest ettevõtetest. Build.Works ettevõttes osutus probleemiks ehituselementide ja toodete ebaefektiivne soovitussüsteem, mistõttu arendati antud töö raames tehisintellekti integratsioon, et tagada kasutajatele efektiivne ja kasutajasõbralik lahendus.

1.2 Eesmärgid

Käesolev lõputöö keskendub tehisintellekti tehnoloogiate rakendamise väljakutsete uurimisele Eesti väikeettevõtetes, tuginedes organisatsiooni AIRE poolt kogutud andmetele. Töö raames rakendab autor tehisintellekti väikeettevõttes, analüüsides oma kogemusi ning uurides erinevaid tehnoloogilisi lahendusi ja teenusepakkujaid. Eesmärk

on automatiseerida ettevõtte toodete soovitussüsteemi, tuues välja erinevate lahenduste omadused, kulud ja rakendamise eeldused. Saadud tulemusi võrreldakse AIRE andmete ja olemasoleva kirjandusega.

Alameesmärgid:

1. Praktilise tehisintellekti rakenduse elluviimine ettevõttes, hinnates erinevaid tehnoloogilisi lahendusi.

2. Toodete soovitussüsteemi automatiseerimise võimaluste analüüsimine, kaaludes nii tehnilisi kui majanduslikke aspekte.

3. Tulemuste võrdlemine AIRE andmete ja kirjanduses esitatud teadmistega, et tõhustada tuleviku rakendusi.

4. Tehisintellekti rakenduste integreerimisvõimaluste uurimine ettevõtte operatsioonidesse, mõõtes nende mõju ettevõtte efektiivsusele ja innovatsioonile.

Antud eesmärgid keskenduvad nii antud lahenduse edukale teostamisele, kui ka suurema pildi, ehk tehisintellekti rakendamise üldistele väljakutsetele ning nende ületamisele.

1.3 Töö struktuur

Töö on jagatud kuueks sisuliseks osaks. Esimeseks on sissejuhatus, kus kirjeldatakse probleem ning töö eesmärk koos alameesmärkidega. Teiseks osaks on taust ja metoodika kus räägitakse tehisintellektist ning selle rakendamise väljakutsetest ettevõtetes. Kolmandas osas tutvustatakse ettevõtet, kus lõputöö läbi viidi, kasutatud tööriistu ning tööprotsessi. Neljandas osas selgitatakse kuidas kujunes välja millist keelemudelit kasutatakse töös, kuidas valiti välja AIRE kvalitatiivseks uuringuks, olemasoleva lahenduse puudused ning kaardistatakse uue lahenduse nõuded. Viiendaks on lahenduse tulemuste kirjeldus, kus kirjeldatakse lahenduse arendust. Kuuendas peatükis analüüsitakse lahendust, väljakutseid arendamisel, kas lahendus vastas kriteeriumitele, võimalusi edasi arenduseks ning ettepanekud tehisintellekti rakendamiseks ettevõtetes kasutades AIRE uuringu ning sellest tööst järelduvaid tulemusi. Kuuendas peatükis asub kokkuvõte ning töö lõpus asub kasutatud allikate kirjeldus ning lisad.

2 Taust

Selles peatükis antakse ülevaade tehisintellekti teoreetilisest taustast ja selle rakendamise väljakutsetest. Peatükk hõlmab tehisintellekti definitsiooni ja erinevaid liike, ning selgitab hallutsinatsioonide probleemi. Lisaks käsitletakse tehisintellekti rakendamise väljakutseid väikeettevõtetes.

2.1 Tehisintellekt

Euroopa parlamendi kohaselt on tehisintellekt masin mis ilmutab inimlaadseid võimeid, nagu mõtlemine, õppimine, planeerimine ja loovus [6]. Kuid tehisintellekt alles areneb, rääkimata tunnetest ning teadvusest. Tegu on konkreetseid ülesandeid teostavate rakendustega, mis töö käigus arenevad [3].

„Peamine erinevus klassikalistest tarkvaralahendustest seisneb asjaolus, et kui viimaste puhul täidavad arvutiprogrammid neile programmeerija poolt etteantud käske, siis tehisintellekti algoritmide puhul rangelt etteantud programmiloogika puudub ja tehisintellekt peab õige lahenduseni jõudma erinevate masinõppe algoritmide kaudu“ [3]. Selle tulemusena suudavad masinõppe algoritmid andmete tohutul hulgal töötlemisel luua autonoomseid lahendusi, mis annab tehisintellektile võime mõjuda sõltumatuna.

Tehisintellekti kasutusele võtt võib üllatavalt kombel vastakaid reaktsioone esile tuua. Näiteks 67% eestimaalasi ootab, et oleksime selle võimsa tehnoloogia rakendamisel pigem ettevaatlikud [3]. Ühest küljest pakuvad need esilekerkivad tehnoloogiad ja tehnikad organisatsioonide tuleviku jaoks märkimisväärseid võimalusi. Teisest küljest võitlevad nende praktilised rakendused paljude lahendamata probleemidega nagu puuduvad regulatsioonid, mis tagaks eetilised normid. Mitmed organisatsioonid(IEEE, CERNA) on algatanud arutelusid, et sellised eetilised süsteemid loodaks. Lisaks saavad palju kriitikat tehisnärvivõrke(*deep learning*) kasutavad süsteemid, kuna need on läbipaistmatud, ega näita kuidas algoritmid jõuavad lõpptulemuseni [1].

2.2 AGI ja LLM

Artificial Generative Intelligence (AGI) ehk generatiivset tehisintellekti võib kirjeldada kui masinaid millel on võime mõista või lahendada mis tahes intellektuaalset ülesannet, millega saab inimene hakkama [7]. Suur keelemudel ehk *Large Language Model* (LLM) peetakse üheks generatiivse tehisintellekti tüübiks.

LLM-id on üles ehitatud masinõppele [8]. Neid mudeleid treenitakse tohutul hulgal andmetega sadade miljardite, tihti triljonite parameetritega, mis võtab palju ressursse treenimiseks ja jooksutamiseks [7]. Tänu sellele suudavad need mudelid mõista loomulikku inimkeelt ning kasutada andmeanalüüsi, et ilma struktuurita küsimustele arusaadav ja mõistlik vastus anda. Tabelis 1 on välja toodud avatud lähtekoodiga keelemudelite edetabel, mis põhineb mudelite tehnilistel aruannetel. Värskendatud 2024 märtsis [9].

Tabel 1. LLM võrdlus

Mudel	**Võimekus**	**Konteksti aken *token*'ites**	**1M *token*'i sisendi maksumus**	**1M *token*'i väljundi maksumus**
Claude 3 Opus	84.83%	200K	15$	75$
Gemini 1.5 Pro	80.08%	128K	7$	21$
Gemini Ultra	79.52%	8.192K	X	X
GPT-4	79.45%	128K	10$	30$
Claude 3 Sonnet	76.55%	200K	3$	15$
Gemini Pro	68.28%	32K	0.125$	0.375$
Palm 2-L	65.82%	8K	X	X
GPT-3.5	65.46%	16K	0.5$	0.375$

2.3 Hallutsinatsioonid

Suured ning võimsad keelemudelid suudavad väga hästi luua sidusat ja kontekstuaalset asjakohast teksti, kuid esile kerkib üks kriitiline probleem – hallutsinatsioonid. Hallutsinatsioon on probleem kus mudelid genereerivad usutavat, kuid faktiliselt ebaõiget või kohati mõttetut teavet [10].

Hallutsinatsioone põhjustavad peamiselt mudelite treenitud andmetest, mudeli treenimisest endast ning järelduste tegemise etapis kui mudelit treenitakse. Probleemid hõlmavad halba andmekvaliteeti, vale informatsiooni, eelarvamusi, aegunud teadmisi ning arhitektuurilisi ja strateegilisi puudusi, mis takistavad nõuetekohast treenimist [10].

Töös „Hallucination is Inevitable: An Innate Limitation of Large Language Models“ [10] on järeldatud, et hallutsinatsioone on võimatu kõrvaldada reaalse maailma keelemudelites. See aga ei tähenda, et hallutsinatsioone ei ole võimalik leevendada. Andmetega seotud probleemide korral tuleks tähelepanu pöörata faktidele keskendunud andmekogudele ning automaatsetele andmepuhastus tehnikate väljatöötamisele. Otsingute täpsustamisele või suurendamisele, mis maandab keelemudelile asjakohaseid dokumente, võib aidata vähendada teadmiste lünkasid ja hallutsinatsioone. Tehnikad nagu *ChainOf-Though* ja *Tree-of-Thought* on aidanud teadmiste meenutamise ning põhjendatusega.

2.4 Tehisintellekti rakendamise väljakutsed väikeettevõtetes

Väikeettevõtted seisavad silmitsi mitmete väljakutsetega, kui nad püüavad oma äriprotsessidesse integreerida tehisintellekti tehnoloogiaid. Neist esimene ja peamine on teadmiste puudumine tehisintellekti võimalustest. Teadlikkus sellest, kuidas tehisintellekt võiks neid aidata, on sageli piiratud, mis omakorda piirab nende võimalusi tehisintellekti efektiivseks kasutuselevõtuks. Samuti on oluline märkida, et väikeettevõtetel on tihti piiratud juurdepääs vajalikele organisatsioonilistele protsessidele ja välistele andmetele, mis on tehisintellekti integreerimiseks hädavajalikud [11].

Väikeettevõttel tulevad abiks digitaalsed platvormid mis pakuvad neile võimalusi tehisintellekti rakendamiseks oma äritegevuses. Sellised modulaarsed platvormid võimaldavad ettevõtetel kasutada teenusepakkuja pakutavaid võimalusi vastavalt ettevõtte teadmistele ja ressurssidele [11].

Näiteks võimaldavad tehisintellekti platvormid juurdepääsu keerukatele tehnoloogiatele, nagu loomuliku keele töötlemine ja masinõpe, ilma et neil oleks vaja teha suuri investeeringuid IT-infrastruktuuri.

Ühed sellised kõige laiemaid tehisintellekti vorme, mida pakuvad nüüdseks väga paljud platvormid, on vestluseks mõeldud tehisintellekt, mis leiavad kasutust väga paljudes rakendustes abistamisega, sisu genereerimisega, tõlketeenusena, andmete analüüsis jpm.

3 Metoodika

Antud peatükis antakse ülevaade metoodikast, mida kasutati käesolevas projektis. Lisaks käsitletakse ettevõtte Build.Works tausta ja kasutatud tööriistu ning tehnoloogiaid.

3.1 Ettevõte taust

Build.Works on pilvepõhine ehitustehnoloogia ettevõte, mis pakub tarkvaraplatvormi, aidates ehitus ettevõtetel oma projekte tõhusamalt ja tulemuslikumalt juhtida. Ettevõtte eesmärk on kokku panna BIM tehnoloogia ERP süsteemidega, et siduda moodulehitise disainiprotsess ja ehitusprojekti läbiviimise protsess.

BIM ehk ehitusinformatsioonimudel on digitaalne esitusviis, mis hõlmab ehitusprojekti olulisi andmeid alates geomeetrilisest kujutisest kuni tehniliste spetsifikatsioonide ja ajakavadeni. Tarkvaralahendused võimaldavad projektiosapooltel, nagu arhitektid, insenerid ja ehitajad, koostööd teha ühises digitaalses keskkonnas, parandades suhtlust ja vähendades vigu ehitusprotsessis. BIM mudeli eesmärk on tõsta ehitusprojekti terviklikkust, efektiivsust ja jätkusuutlikkust, aidates kaasa objekti elutsükli tõhusale haldamisele. Eesti keeles võiks BIM-i nimetada kui "ehitusinformatsioonimudelit".

ERP süsteem on tarkvaralahendus, mis ühendab ettevõtte erinevad osakonnad, nagu finants, müük ja tootmine, üheks tervikuks. Selle eesmärk on optimeerida protsesse, parandada andmehaldust ja võimaldada reaalajas juurdepääsu olulistele andmetele. ERP aitab ettevõttel saavutada paremat efektiivsust, suurendada läbipaistvust ning vähendada kulusid.

Platvorm sisaldab tööriistakomplekti projektide kavandamiseks, planeerimiseks ja teostamiseks. Build.Worksi võivad kasutada kasutavad igas suuruses ehitusmeeskonnad, alates väikeettevõtetest kuni suurte ettevõteteni. Meeskonnad pääsevad sellele juurde kõikjalt, kus on interneti-ühendus. Platvorm on ka mobiilisõbralik, nii et meeskonnad saavad seda liikvel olles kasutada.

3.2 Kasutatud tööriistad

3.2.1 Tehisintellekti platvorm

Tehisintellekti rakendamine algab põhjaliku analüüsiga, millist tehisintellekti kasutada vajaliku probleemi lahendamiseks. See hõlmab erinevate tehisintellekti tüüpide, nende võimete, rakendusvaldkondade ja integreerimisküsimuste kaalumist. Analüüsi käigus pöörati peamiselt tähelepanu ühele platvormile - OpenAI. Antud teenusepakkuja esindab laia spektrit võimalusi, alates üldotstarbelisest tehisintellektist kuni spetsialiseeritud ettevõtte taseme lahendusteni. Lisaks uuriti teisi selle valdkonna võimsamaid keelemudeli pakkujate platvorme nagu Google(Gemini), Anthropic(Claude) ja teised platvormid mis kasutavad läbi kasutajaliideste nende keelemudelite API'sid.

OpenAI poolt loodud ChatGPT (*Generative Pre-trained Transformer*) toode sai alguse isoleeritud keelemudelist, millel polnud välismaailmaga seost. Nüüd töötab OpenAI välja ja annab välja integratsioone, mis võimaldavad nende keelemudelitel suhelda teiste rakenduste ja internetiga [7]. Kasutada on võimalik mitmeid mudeleid, alates tekstist ja pildist aru saavatest ning neid genereerivatest mudelitest kuni spetsiifiliste ülesannete lahendamiseks mõeldud mudeliteni. Mudeleid on treenitud erinevates keeltes esineva tekstiga, mis võimaldab masinal mõista ja toota teksti mitmesugustes keeltes. ChatGPT'l on võime õppida interaktsioonide kaudu, iga vestlus ja päring aitab mudelil muutuda täpsemaks, võimaldades paremaid vastuseid ja paremat kasutajakogemust [12].

3.2.2 Arendustööriistad ja tehnoloogiad

Lõputöö praktilises osas kasutati ettevõtte juba olemasolevaid arendusvahendeid ja süsteeme. SAP S/4HANA ettevõtte ressursside planeerimise (ERP) süsteem toimib pilvepõhise lahendusena tagarakenduse jaoks. Arendustööde läbiviimiseks kasutatakse Visual Studio Code tarkvara, milles arendati kasutajaliidest ning rakenduse loogikat. Tagarakenduse loogika arendamiseks on kasutusel SAP *Web Integrated Development Environment* (IDE), mis on spetsiaalselt kohandatud SAP süsteemidega töötamiseks.

Andmete haldamiseks kasutatakse SAP HANA andmebaasi, mis on integreeritud SAP süsteemi infrastruktuuriga. Koodihaldus ja versioonikontroll on organiseeritud läbi BitBucket platvormi.

Arenduses on rakendatud JavaScript programmeerimiskeelt, nii esiliidese kui tagarakenduse loogika kirjutamiseks. Kasutajaliidese arendamiseks kasutatakse XML keelt, mis vastab SAP Fiori disainijuhistele.

Selline tehnoloogiate ja vahendite valik on määratud eelkõige ettevõtte poolt, kelle infrastruktuuri ja süsteemidega lõputöö on integreeritud, et tagada ühtlus olemasolevate tööprotsessidega.

3.2.3 *Prompt engineering*

Prompt engineering ehk juhiste ülesehitamine on kriitiline osa keelemudelite efektiivses rakendamises. See hõlmab endas juhiste koostamist, mis suunavad tehisintellekti soovitud vastuseid andma.

*Prompt'*ide ülesehitamine algab probleemi mõistmisest ja täpsete eesmärkide seadmistest. *Prompt* peab sisaldama vajalikku taustinfot ja täpseid juhiseid või ülesandeid. Edukaks ülesehituseks on tarvis kasutada iteratiivset lähenemist, kus *prompt'e* testitakse ja kohandatakse pidevalt.

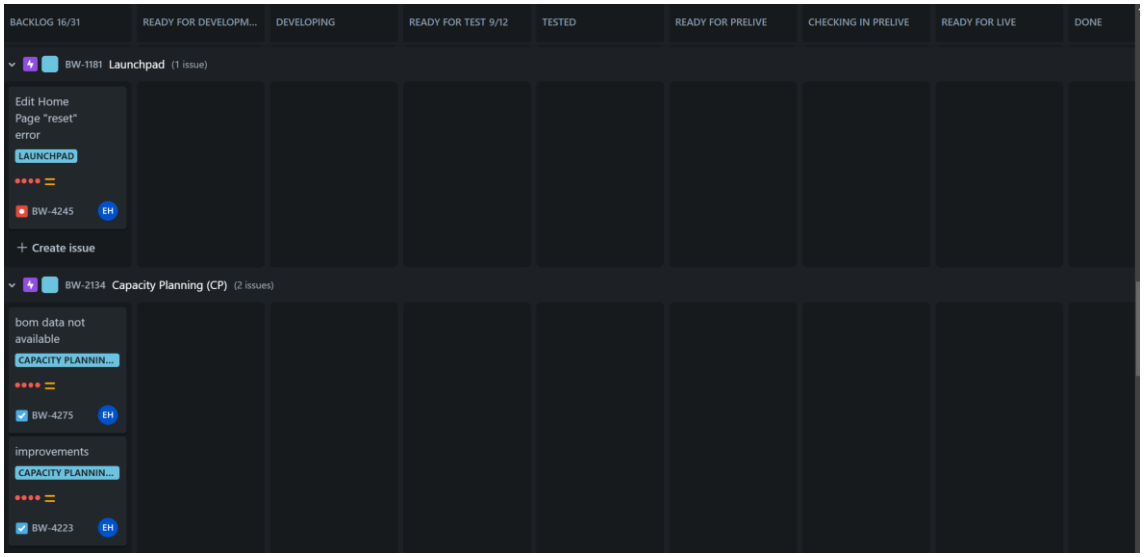
OpenAI dokumentatsioon rõhutab *prompt*'ide mõju mudeli väljundi kvaliteedile [13]. Hästi kujundatud juhis võib märkimisväärselt parandada mudeli jõudlust. *Prompt*'ide täpsustamisel tuleb tähelepanu pöörata kontekstile, millist teavet tuleks ignoreerida, millele keskenduda. Samuti tuleb tähelepanu pöörata vastuste struktuurile ja vormingule, et tulemused oleksid järjepidevad.

3.3 Tööprotsessi kirjeldus

Lõputöö raames kasutatud ettevõttes toimus tarkvaraarendus agiilsete meetodite järgi. Igapäevane töörütm sai alguse hommikustest *stand-up* koosolekutest, kus tiimi liikmed jagasid üksteisega oma tööülesannete progressi. Koosolekutel arutati, millised ülesanded on käsil, millises arengufaasis need on, kui palju aega võib nende lõpetamine veel võtta ning kas esineb takistusi. Joonisel 1 on kuvatud JIRA tabel, kus on tööülesanded ning nende etapid.

Tootesoovitussüsteemi arendamisel andsid aktiivselt nõu teised arendajad ning tootejuht, pakkudes vajalikku tagasisidet ning juhiseid. Selline koostöö ja suunajuhised tootejuhilt

olid arendusprotsessi lahutamatu osa, tagades süsteemi efektiivse ja eesmärgipärase valmimise.



Joonis 1. JIRA *kanban* tabel

4 Nõuded

See peatükk kirjeldab nõudeid, mis seati OpenAI mudelite valikule ja rakendamisele andmete analüüsiks. Peatükis käsitletakse lähemalt AIRE organisatsiooni ning selgitatakse, miks nad valituks osutusid. Lisaks antakse ülevaade olemasolevast lahendusest ja kirjeldatakse nõudeid uue lahenduse jaoks.

4.1 OpenAI mudelid

See peatükk keskendub OpenAI mudelite valikule. Üle käiakse mudelite eripärad, hinnastused, võimekused ja võimalused, et leida kõige sobivamad mudelid ülesande lahendamiseks.

4.1.1 GPT-3.5-turbo-0125

Esimene mudel millega lahendust katsetati oli GPT-3.5-turbo-0125 tema odavuse ja kättesaadavuse pärast. 3.5-turbo oli ainuke valitud mudelitest, mida sai kasutada kontol oleva algse 5$ krediidiga. Teised mudelid ning nende hinnastused on välja toodud tabelis 1.

Tabel 2. OpenAI mudelite maksumused [14]

Mudel	**Sisendi maksumus**	**Väljundi maksumus**	**Treenimise maksumus**
GPT-3.5-turbo-0125	$0.50 / 1M *token'*it	$1.5 / 1M *token'*it	-
GPT-4-0125-preview	$10 / 1M *token'*it	$30 / 1M *token'*it	-
Fine-tuning mudel GPT-3.5-turbo	$3 / 1M *token'*it	$6 / 1M *token'*it	$8.00 / 1M *token'*it

4.1.2 *Fine-tune* mudel

Fine-tuning, ehk mudeli häälestamine, võimaldab kohandada olemasolevat mudelit (antud juhul GPT-3.5-turbo-0125) oma andmete põhjal. See kohandab mudelit konkreetse kasutusjuhtumiga, parandades selle mudeli jõudlust ja täpsust. Mudelile tuli minimaalselt koostada 10 näidet, mille pealt mudel saaks õppida. Näited tuli üles laadida

JSONL formaadis ja seejärel API päringu kaudu treenimiseks üles laadida. Pärast treenimist võib treenitud mudelit kasutada nagu igat teist mudelit.

Fine-tuning mudel pakkus võimalust kohandada mudelit spetsiifiliste andmetega, siiski osutus see lahendus vähem praktiliseks väikese näidete arvu tõttu. Kui andmeid oli vähe või kontekst muutus, ei suutnud mudel pakkuda täpseid ja usaldusväärseid vastuseid. Seetõttu ei osutunud *fine-tuning* mudel lõplikuks valikuks.

4.1.3 GPT-4-0125-preview mudel

GPT-4-0125-preview on kõige hiljem välja tulnud ja võimsaim mudel, mis oli arendamise hetkel eelvaate etapis, kuid kasutusvalmis. See mudel ei ole kättesaadav esialgse krediidiga, vaid selleks tuli ise krediiti juurde kanda. GPT-4-0125 tulemused on palju järjepidevamad ja täpsemad, pakkudes paremat kasutajakogemust ja vähem vigaseid soovitusi. GPT-4-0125 ei andnud „laiskasid“ vastuseid, mida võis täheldada GPT-3.5 mudeli puhul, nagu on mainitud ka OpenAI blogis [15].

Mudelite valikul lähtuti mitmest nõudest, sealhulgas:

- Tõhusus ja täpsus- mudel pidi olema võimeline pakkuma täpseid ja tõhusaid soovitusi.
- Kuluefektiivsus- mudel pidi olema rahaliselt mõistlik, arvestades kasutuse sagedust ja andmete mahtu.
- Kasutusmugavus- mudeli integreerimine pidi olema sujuv ja lihtne, et tagada kiire juurutamine ja vähendada keerukust.
- Skaleeritavus- lahendus pidi olema modulaarne ja hõlpsasti kohandatav tulevaste vajaduste ja võimalike tehnoloogiliste uuenduste jaoks.

Neid kriteeriume silmas pidades valiti lõplikuks lahenduseks kombinatsioon GPT-3.5 ja GPT-4 mudelitest, pakkudes parimat tasakaalu täpsuse, kuluefektiivsuse ja kasutusmugavuse vahel.

4.2 AIRE

AI and Robotics Estonia(AIRE) toetab Eesti tööstusettevõtteid nutikate digilahenduste kasutusele võtmisel tehisintellekti ja robootika vallas [16]. AIRE nõustab, koolitab, hindab ja viib läbi demoprojekte tehisintellekti teemadel. AIRE töötab koos ülikoolide-teadusparkide- ja tööstusega. Tegemist on maatriksorganisatsiooniga kus on laiendatud meeskondades ligi 60 inimest, hõlmates teadlasi ning eksperte ülikoolidest.

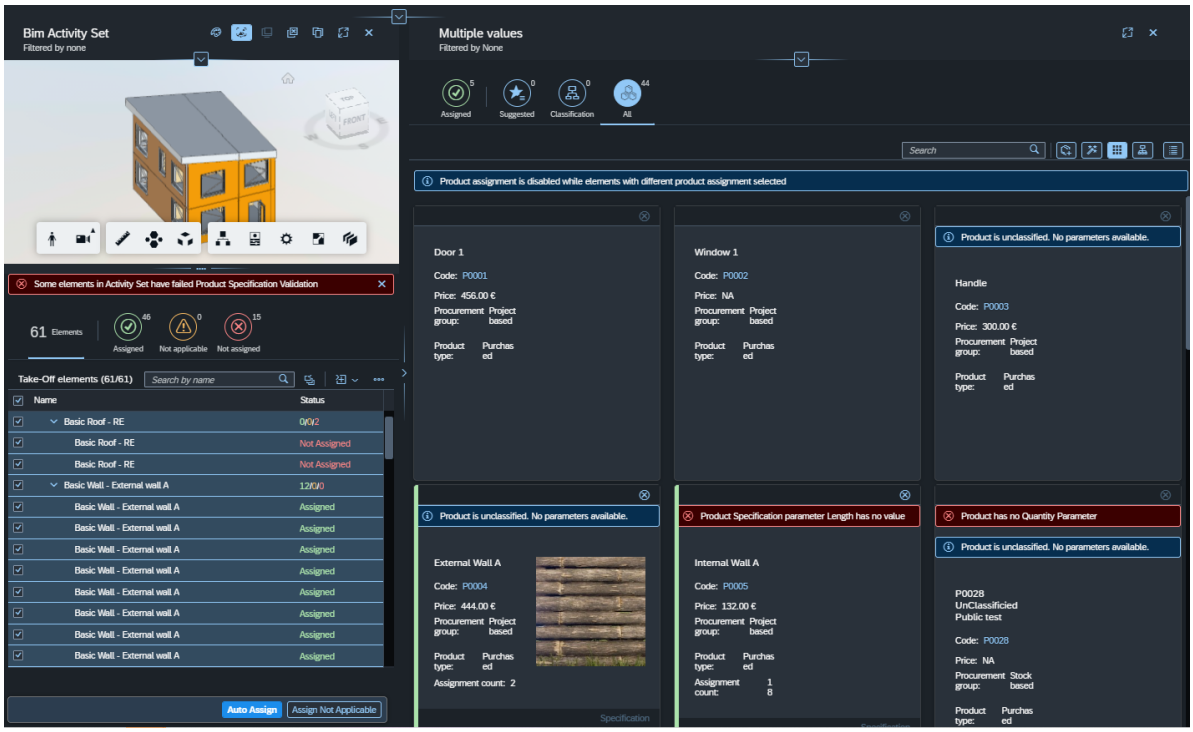
AIRE'ga viidi läbi antud töö raames kvalitatiivne uuring. Uuringu eesmärk oli saada sügavam arusaam probleemidest, millega ettevõtted kokku puutuvad, kui nad tehisintellekti kasutusele tahavad võtta. Nad valiti uuringus osalema, kuna on juhtivad tehisintellekti ja robootika lahenduste asjatundjad, kes nõustavad ja hindavad Eesti ettevõtete vajadusi nende valdkondade osas. 2024 jaanuari seisuga on AIRE koostööd teinud 146 ettevõttega, kellest 32 on mikro ettevõtted, 52 väikesed, 57 keskmised ja 5 suurt kelle käive kokku on umbes 9% kogu tööstuse käibest.

AIRE peamisteks teenusteks on digiküpsuse hindamine, tehisintellekti ja/või robotiseerimise otstarbekuse nõustamine, finantseerimise nõustamine – erakapitali kaasamine/avalikud meetmed, lisaks koolitused väikeste ja keskmise suurusega ettevõtetele tehisintellekti, digitaalsete tehnoloogiate ja robootika kohta ning demo projektide läbiviimine.

Töös on neil üle 30 lahenduse kus otsitakse nutikaid, tehisintellekti komponendiga lahendusi. Valmis lahendused on neil ettevõtetes nagu Milrem, Raja, Yanu, Comodule ja Valdek.

4.3 Olemasolev lahendus

Selles peatükis tutvustatakse olemasolevat lahendust toodete soovitamiseks ja määramiseks elementidele enne tehisintellekti kasutuselevõttu. Joonisel 2 on näha ehitus mudelit, selle elemente(vasakul) ning tooteid(paremal).



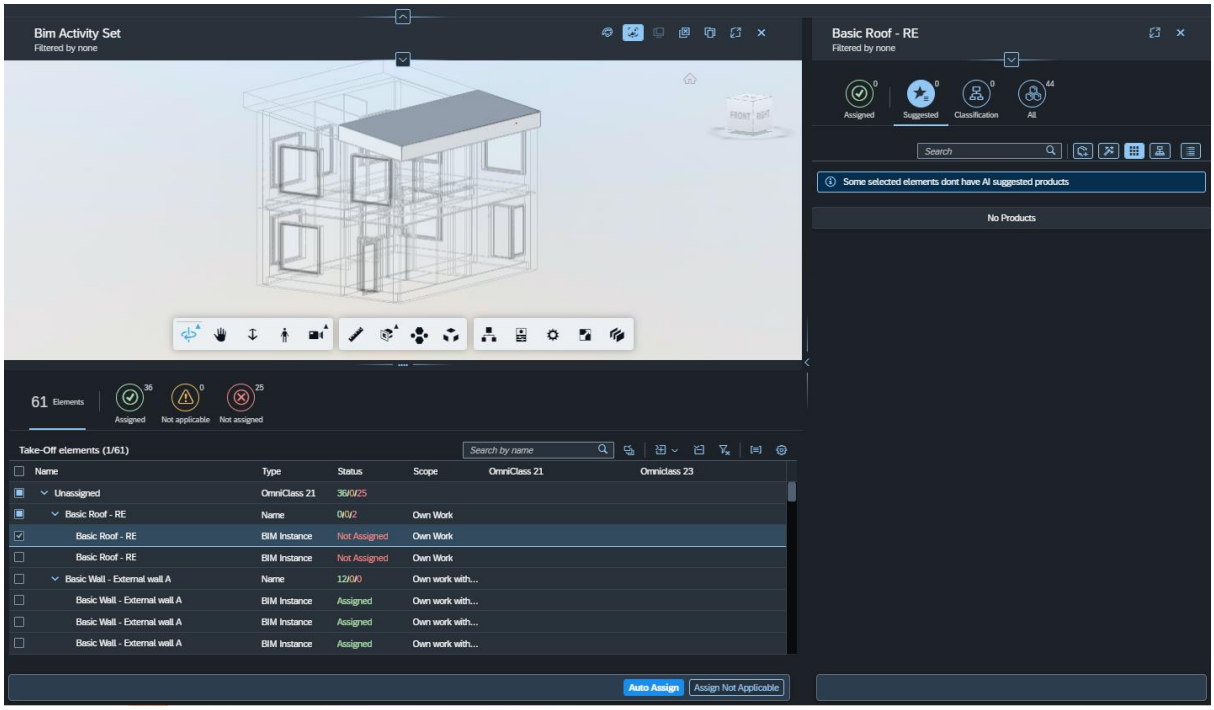
Joonis 2. Elementide ja toodete vaade

4.3.1 Toodete soovitamine

Enne tehisintellekti kasutusele võtmist oli samuti võimalus määrata BIM elementidele tooteid automaatselt. Olemasolev loogika põhineb elementide ja toodete sarnasuste leidmisele ning kasutaja sisestatud filtritele. Esiteks kogutakse kasutaja poolt seadistatud filtrite ja otsingusõnede andmed. Seejärel genereeritakse soovitusi, võttes aluseks antud sisendid ning võrreldes elementide ja toodete tootekoode ning muid parameetreid nende kattuvuse põhjal.

Esmalt vaadatakse kas on võimalik soovitusi teha elemendi tootekoodi põhjal. Kõik elemendid, mida kasutaja valis, käiakse üle läbi loendades erinevate tootekoodide esinemisi. Iga elemendi puhul kontrollitakse, kas selles on vastav tootekood, kui jah, siis suurendatakse vastavat loendurit iga kord, kui tootekood esineb. Pärast seda määratakse kindlaks erinevate tootekoodide arv. Kui loodud massiivis on ainult üks unikaalne tootekood ja kõik valitud elemendid sisaldavad seda tootekoodi, kontrollitakse, kas vastav toode on olemas toodete massiivis. Antud lahenduses on valitud elementidele korraga võimalik ainult ühte tüüpi toodet soovitada, mis sobitub kõikidele elementidele. Siis seadistatakse soovitusobjekt, mis sisaldab kogu vajalikku teavet lõplike soovituste kohta, sealhulgas soovitusfiltri funktsioon, mis on kasutusvalmis toodete filtreerimiseks, samuti sõnumeid mis aitavad kasutajal mõista soovitusprotsessi.

Juhul kui tootekoodi põhjal soovitusi ei leita, otsitakse tootesoovitusi elemendi parameetrite ja klassifikatsioonikoodide põhjal. Esmalt kontrollitakse, kas on võimalik määrata elementidele vastavaid klassifikatsioonikoode. Kui elementide massiiv sisaldab ainult ühte elementi, kasutatakse spetsiifilist funktsiooni, et leida selle elemendi klassifikatsioonikood. Kui elemente on rohkem, kasutatakse kõigi elementide ühiseid klassifikatsioonikoode. Kui on olemas üks konkreetne klassifikatsioonikood, filtreeritakse tooteid, et leida need, mis vastavad antud klassifikatsioonikoodile. See toimub kontrollides toodete klassifikatsiooni koodi. Samuti proovitakse leida kattuvaid elemendi/elementide parameetreid, mis võiksid viidata konkreetsele tootele. Selle funktsionaalsuse lõpus tagastatakse jälle soovitusobjekt, mis sisaldab soovitusfiltrit ning ülejäänud vajalikku teavet soovituste kohta. Lõpuks kuvatakse soovitused *suggested* filtriga tabelisse. Joonisel 3 on näha, et ühtegi toodet ei leitud katuse elemendi jaoks.



Joonis 3. Olemasoleva lahenduse tootesoovitused

Antud lahenduse eesmärgiks oli pakkuda kasutajale elementide jaoks tootesoovitusi, kasutades andmeanalüüsi meetodeid ja kasutajapoolt määratud kriteeriume. See hõlmab elementide parameetrite, klassifikatsioonikoodide ja kasutajapoolt seadistatud filtrite analüüsimist, et soovitada kasutajale tooteid, mida määrata vastavatele elementidele. Vastavad soovitused kuvatakse kasutaja jaoks kasutajaliideses.

4.3.2 Toodete automaatselt määramine

Toodete automaatne määramine hõlmab protsessi, kus süsteem kasutab soovitusfunktsioone toodete määramiseks ning rakendab neid automaatselt kõikidele vastavatele BIM elementidele. Erinevalt manuaalsest valikust, kus kasutaja peab iga elemendi jaoks tooteid käsitsi valima, võimaldab automaatne määramine optimeerida ja kiirendada tootemääramise protsessi, pakkudes kasutajatele tõhusat viisi suure hulga andmete haldamiseks.

Iga elemendi kohta kaardistatakse elemendile sobivad tooted vastavalt soovitusalgoritmile. Seejärel kontrollitakse iga toote puhul, kas see sisaldab peamist koguseparameetrit ning kas elemendi parameetrite hulgas on sama parameeter ja kas sellel on väärtus olemas. Seejärel määratakse toode elemendile.

4.3.3 Olemasoleva lahenduse puudused

Varasem lahendus hõlmas endast piiratud võimalusi leida sobivaid tooteid elementidele. Vähene paindlikkus erinevate projektide, elementide, toodete ärakasutamisel, et leida sobivaid tooteid mida elementidele määrata.

Peamiseks probleemiks kujunes algse lahenduse efektiivsus määrata potentsiaalseid tooteid elementidele, ning soovitused olid ära määratud väga rangete kriteeriumite põhjal, mistõttu ei suutnud süsteem määrata tooteid tõhusalt, tuues vähest väärtust kasutajale potentsiaalsete toodete soovituse näol. Toodete määramine oli suur käsitsi töö, milles sai vähe toetuda automaatsele toodete määramise süsteemile või soovitustele.

Kokkuvõttes, vana süsteemi puudulik automatiseerimine toodete määramisel ei võimaldanud kasutajatel aega kokku hoida, mis tõi kaasa suure manuaalse töökoormuse ja vähendas süsteemi üldist kasulikkust. Tõhusate soovituste puudumine ja paindlikkuse vähesus mõjutasid negatiivselt kasutaja võimet leida ja määrata sobivaid tooteid elementidele. Sellise süsteemi jäikus ja läbipaistvuse puudumine vähendas kasutaja kogemuse kvaliteeti, muutes vajalikuks süsteemi efektiivsemaks muutmist, et suurendada kasutusmugavust ja adaptiivsust eri tüüpi projektide jaoks.

4.4 Nõuded uuele süsteemile

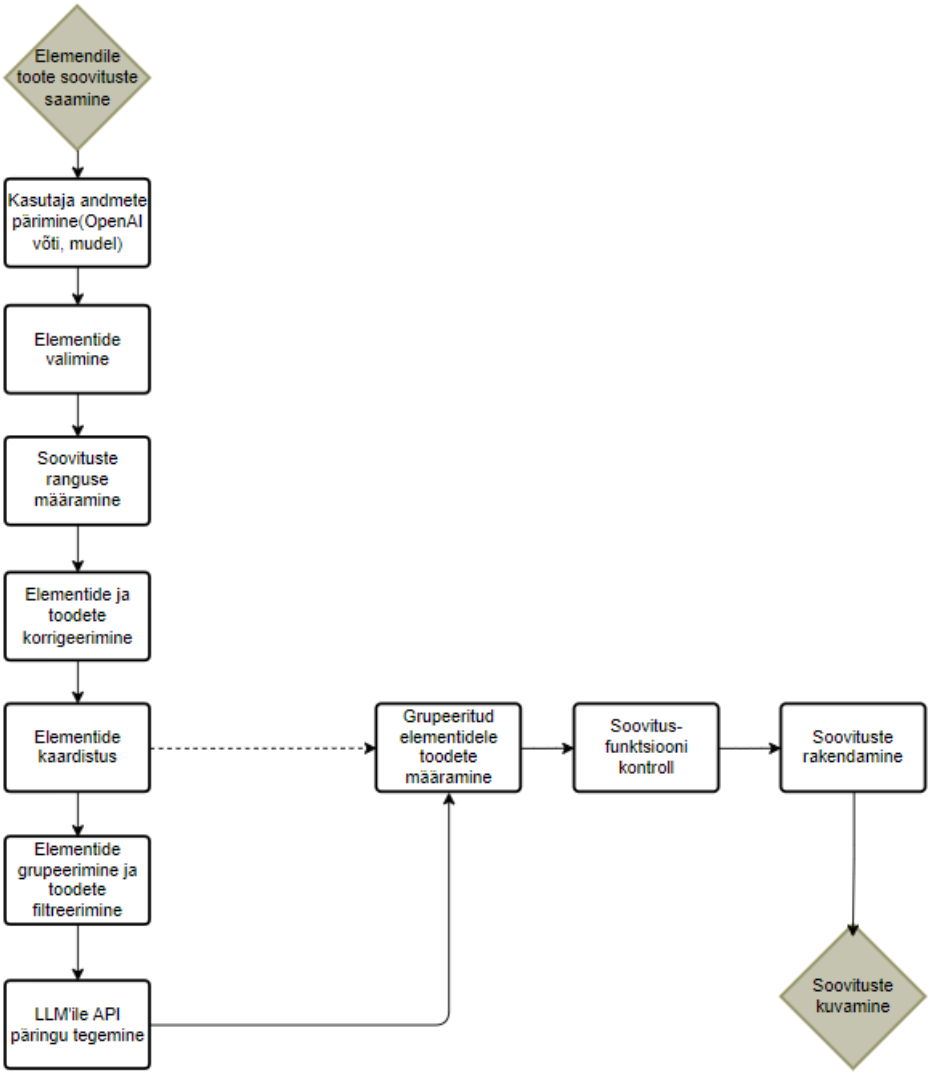
Uue süsteemi kavandamisel olid peamisteks eesmärkideks:

- Elemendile tõhusam vähem kitsendatud toodete soovitamine
- Lahendus ei tohi olla aeglane, kuna see mõjub halvasti kliendi kasutajakogemusele
- Lahenduse maksumus
- Lahenduse integreerimise keerukus

Teisejärgulised eesmärgid:

- Hierarhiline tehisintellekti mudelite kasutamise võimalus, globaalne vs lokaalne
- Lahendus peaks olema aktuaalne, võimalus edasi arendada tulevikus
- Mudeli ja kasutajapõhine maksevõimalus
- Võimalus mudelile kaasa anda kontekst projekti/ülesande näol

Lahenduse kavandamisel seati peamised eesmärgid, mis keskenduvad lahenduse tõhususele, kiirusele, maksumusele ja integreerimise keerukusele. Eesmärk oli arendada süsteem, mis suudaks erinevatele elementidele korraga soovitada sobivaid tooteid, säilitades samas kiire reageerimisaja, et tagada sujuv kasutajakogemus. Joonisel 4. on kuvatud peamine toote soovitamise protsess.



Joonis 4. Tootesoovituste protsess

Teisejärguliste eesmärkide hulka kuulusid tehisintellekti mudelite hierarhiline kasutamine, lahenduse ajakohasus ja tulevikuarenduste võimalus, mudeli- ja kasutajapõhine maksevõimalus ning võimalus anda mudelile konteksti projekti või ülesande näol. Samuti uuriti lahenduse aktuaalsuse säilitamist, uurides erinevaid platvorme ja tööriistu, mis võimaldaksid süsteemi tulevikus kohandada ja täiendada.

5 Tulemused

See peatükk annab põhjaliku ülevaate arendusprotsessist. Alustatakse raamistikust ja andmete korrigeerimisest, et teha esimesed päringud keelemudeli pihta. Seejärel kirjeldatakse, kuidas töödeldakse andmeid enne ja pärast päringu saatmist. Lisaks katsetati võimalusi efektiivsemalt saata suuremahulisi andmeid keelemudelile ning käsitletakse andmete filtreerimist ja grupeerimist, mis aitas lahenduse kallidusega. Viimastes alampeatükkides kirjeldati tehisintellekti poolt soovitatud toodete ärakasutamist automaatses määramissüsteemis ning OpenAI seadete salvestamist ja veahaldust.

5.1 Eeltöö arendusprotsessile

Enne arendamisega alustamist uuriti milline on hetkel olemasolev toodete soovitamise lahendus. Millel põhinevad toodete soovitused, miks praegune süsteem ei toimi nii hästi nagu vaja oleks ning kuidas kasutada ära juba olemasoleva lahenduse algeid, et arendada parem süsteem elementide toodete soovitamiseks. Esmane süsteem oli parajalt keerukas, pakkudes kasutajatele tootesoovitusi, mis põhinesid andmeanalüüsil ja kasutajapoolt määratud kriteeriumidel mis on täpsemalt ära kirjeldatud peatükis 3.2.

Seejärel pühenduti põhjalikule uurimistööle, et tuvastada võimalikud tehnoloogilised lahendused, mis võiksid vastata projekti spetsiifilistele nõuetele. LLM mudeleid katsetati mitme platvormi *sandbox* keskkondades, et näha, millised keelemudelid said antud ülesandega kõige paremini hakkama. Parima tulemuse andsid kõige võimsamad keelemudelid, mis kitsendasid platvormide valikut.

Järgmiseks hinnati integreerimise- ja tehnoloogia keerukust ning võimalusi, mida tehisintellekti pakkujad pakuvad. OpenAI lahendused osutusid selles osas kõige sobivamaks, kirjeldatud peatükis 1. OpenAI valiti välja mitte ainult seetõttu, et see on üks varasemaid ja laialdaselt tunnustatud tehisintellekti platvorme, vaid ka tänu sellele, et selle kohta leidub rohkelt avatud lähtekoodiga materjale. Lisaks pakub OpenAI mitmeid erinevaid lahendusi, mida oli võimalik katsetada, näiteks: mudeli

häälestamine/treenimine oma andmetega, assistendid, mis võimaldavad failidest otsida infot ning mitmed erinevad hinna ja võimekuse järgi klassifitseeritud mudelid.

5.2 Raamistiku püstipanek

Järgnevalt, keskenduti GPT raamistiku kasutajaliidese ja taga rakenduse püstitamisele ning esimese API päringu katsetamisele. See etapp oli kriitiline, et mõista, kuidas süsteemi üles ehitada ja kuidas tehisintellekti mudelid praktikas toimivad. Uue süsteemi nõuetest lähtuvalt tuli keskenduda lahenduse vähesele integratsioonikeerukusele, mistõttu esmalt uuriti OpenAI mudeleid, mis on tuntud oma paindlikkuse poolest. Peamiselt keskenduti GPT-3.5-turbo-0125 ja GPT-4-0125-preview mudelitele, mis võimaldavad analüüsida kasutaja saadetuid andmeid ning juhiseid ning tagastada vastus. Selline vestlus toimub läbi API päringu [17]. Joonisel 5 on kuvatud JSON struktuur selliseks API päringuks:

```
{
    messages: [
      {
        role: "system",
        content: "{Prompt}"„
      },
      {
        role: "user",
        content: "{Data}" },
    ],
    model: "GPT-3.5-turbo-0125"
}
```

Joonis 5. OpenAI mudelite API

5.3 Andmete korrigeerimine

Algsetes mudeli ning toote andmetes oli väga palju infot, mille saatmine tehisintellekti mudelile ei annaks täpsemaid vastuseid ning oleks väga kulukas. OpenAI mudelid on ülesehitatud *token*'ite põhiselt - sisendi/väljundi andmete suurusele, sellepärast oli esimene samm andmete töötlemisel rafineerida ja korrastada saadetavad andmed- eemaldada ebavajalikud parameetrid ning vajaminevad andmed kompaktselt kirjutada.

```
const trimElementData = (oElement) => {
              let sProductCodeValue = null;
              const oElementClassification =
       this.getElementClassificationValueIfValid(oElement,
       iClassificationParameterId, oClassificationItemsByCode);
              const aTrimmedParameters =
       oElement.bomBimParameters.flatMap(oParam => {
                const oFoundParam = oParametersData.Parameters.find(p =>
       p.Id === oParam.ParameterId);
                if (oFoundParam) {
                  if (oFoundParam.Name === 'Product Code') {
                    sProductCodeValue = oParam.Value;
                    return []; // Skip adding to Parameters
                  } else {
                    return [oFoundParam.Id];
                  }
                }
                return [];
              });

              const oTrimmedElement = {
                Id: oElement.Id,
                Name: oElement.Name
              };
              if (sProductCodeValue) {
                oTrimmedElement.ProductCode = sProductCodeValue;
              }
              if (oElementClassification) {
                oTrimmedElement.Class = oElementClassification;
              }
              if (aTrimmedParameters.length > 0) {
                oTrimmedElement.ParamIds = aTrimmedParameters;
              }

              return oTrimmedElement;
            };

            const trimProductData = (oProduct) => {
              let aTrimmedParameters = [];
              const iDefParamId =
       oProduct.DefaultActivityQuantityParameterId;
              if (oProduct.specificationParameters) {
                aTrimmedParameters =
       oProduct.specificationParameters.map(param => (param.parameter.Id));
              }
              const oTrimmedProduct = {
                Id: oProduct.Id,
                Name: oProduct.Name,
                Code: oProduct.ExternalCode
              };
              if (aTrimmedParameters.length > 0) {
                oTrimmedProduct.ParamIds = aTrimmedParameters;
```

```
        }
        if
(oProduct.parameterClassificationItem?.ClassificationItemCode) {
          oTrimmedProduct.Class =
oProduct.parameterClassificationItem.ClassificationItemCode;
        }
        if (iDefParamId) {
          oTrimmedProduct.DefParamid = iDefParamId;
        }
        return oTrimmedProduct;
      };
```

Joonis 6. Andmete korrigeerimine

Pöörata tuli tähelepanu sellele, et saadetav info tehisintellekti mudelile oleks nii väheste tähemärkidega kirjutatud kui võimalik. Selline lähenemine aitab kokku hoida päringute hinnas ning optimeerides mudeli töötlemisaega. Joonisel 6. on välja toodud kood, mis filtreerib sisse vaid vajaminevad parameetrid.

Toote ja elemendi andmeid korrigeeriti ning optimeeriti arenduse vältel pidevalt. Lõplik andmete ehitus on kuvatud Joonisel 7:

```
{
  "ElementData": [
    {
      "Id": "integer",
      "Name": "string",
      "ProductCode": "string",(optional)
      "Params": [string],(optional)
       "Class": "string"(optional)
    }
  ],
  "ProductData": [
    {
      "Id": "integer",
      "Name": "string",
      "Code": "string",
      "Class": "string",(optional)
       "DefParamId": "integer",(optional)
      "Params": [string](optional)
    }
  ]
}
```

Joonis 7. Rafineeritud andmed

Järgmiseks sammuks oli katsetada OpenAI poolt pakutud mudeleid, nende võimekust, sobivust ning hinda antud ülesande kontekstis.

5.4 Andmete töötlemine

Mudeli juhistesse anti juhised millises formaadis ja struktuuriga andmeid tagastada. Neid andmeid tuli salvestada ning töödelda ülejäänud rakenduse kasutusele vastavaks.

Joonisel 8 on kirjeldatud API päringust tulevate andmete formaat:

```
"ElementData": {
  "ElementId": [
    {
      "ProductId": integer,
      "Reason": string,
      "Rating": integer
    }
  ],
}
```

Joonis 8. API päringu vastus

Elementide tootesoovitused on kaardistatud elemendi Id järgi. Sellised objektid sisaldavad toodete massiivi, kus igas tooteobjektis on olemas tema Id, põhjendus miks selline soovitus selle elemendi kohta tehti ning hinnang 1-10, kui hästi toode ja element sobituvad.

```
aElementIds.forEach((oUniqueElementId) => {
            const aAssignedProducts =
oElementsWithProducts[oUniqueElementId];
            const oUniqueElement =
oProductAssignmentData.oAllTrimmedSelectedElements.find(oElement =>
oElement.Id.toString() === oUniqueElementId);
            const aEquivalentElements =
oProductAssignmentData.oAllTrimmedSelectedElements.filter(oElement =>
              this.areElementsEquivalent(oUniqueElement, oElement));

            aEquivalentElements.forEach(oEquivalentElement => {
              const currentProductIds =
oProductAssignmentData.AISuggestedElementIdsWithProducts[oEquivalentEl
ement.Id].map(product => product.Id);
              const uniqueNewProducts =
aAssignedProducts.filter(product =>
!currentProductIds.includes(product.Id));
              if
(oProductAssignmentData.AISuggestedElementIdsWithProducts[oEquivalentE
lement.Id]) {
                oProductAssignmentData.AISuggestedElementIdsWithProduc
ts[oEquivalentElement.Id] = [
                  ...oProductAssignmentData.AISuggestedElementIdsWithP
roducts[oEquivalentElement.Id], ...uniqueNewProducts];
              } else {
                oProductAssignmentData.AISuggestedElementIdsWithProduc
ts[oEquivalentElement.Id] = aAssignedProducts;}
            });
          });
```

Joonis 9. Elementide tootesoovituste salvestamine

Joonis 9. kood tegeleb API päringust tulevate Elemendi Id järgi kaardistatud toodete massiivi objektide salvestamisega. Esmalt määrati teistele valitud samaväärsetele elementidele, mis ei saadetud API päringuga kaasa samad tooted, sellest protsessist on kirjeldatud punktis 5.6. Samuti vaadatakse üle kas API päringust tulevatele elementidele on juba olemas varasemaid tehisintellekti soovitusi, kui on siis lisatakse unikaalsed tooted varasemate soovituste juurde lisaks. Lõpuks salvestatakse kõik soovitused toodete mudeli andmetesse, et neid oleks võimalik edasi kasutada.

Järgnevalt navigeeritakse kasutajaliideses soovitatud toodete vaatesse ning kontrollitakse kas kõikidel valitud elementidel on tehisintellekti poolt tehtud toote soovitusi, joonis 10. Tehisintellekti soovitusfilter arendati nagu varasemad filtrid: soovitusi näidatakse ainult siis kui valitud elementidel on ühiseid tooteid. Juhul kui soovitusi küsiti vaid ühe elemendi kohta, kuvatakse need tootesoovitused. Lõpus tagastatakse soovitusobjekt koos soovitusfiltriga(joonisel 11) mis filtreerib tooteid Id põhiselt.

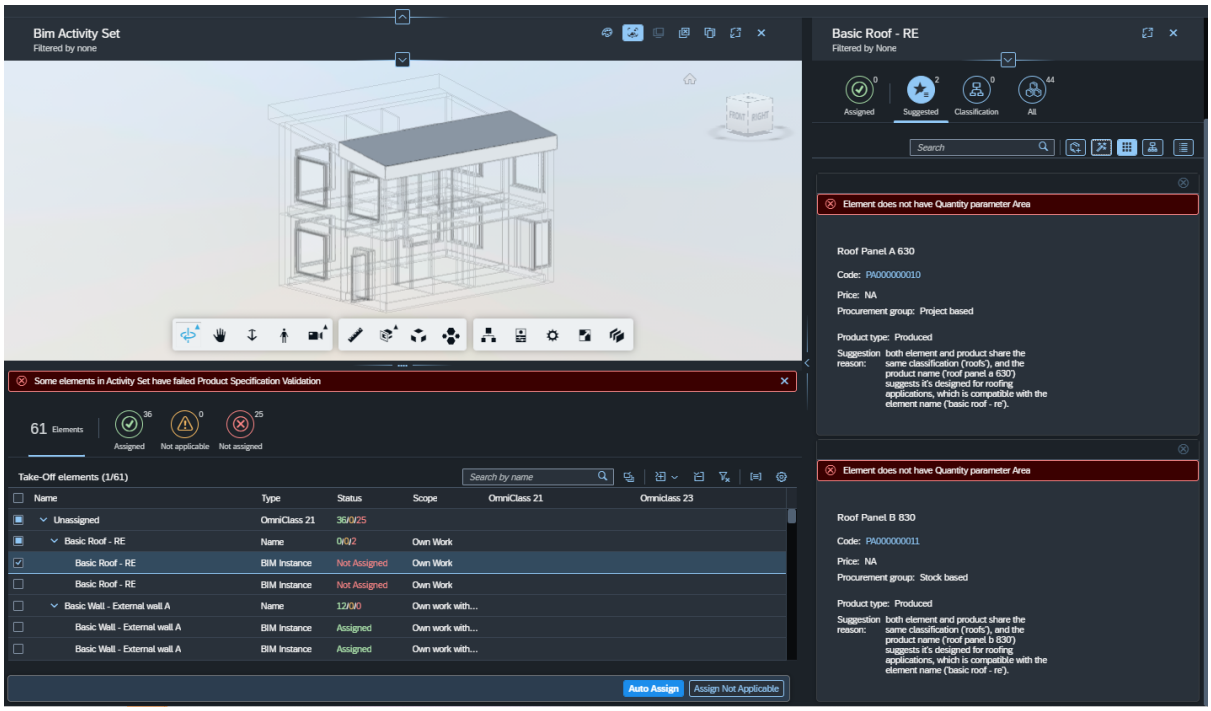
```
    checkIfAIHasSuggested (aElementIds, oProductAssignment) {
      const oAISuggestedElementData =
oProductAssignment.AISuggestedElementIdsWithProducts;
      const oSelectedElementIds =
Object.keys(oAISuggestedElementData).reduce((acc, key) => {
        key = parseInt(key, 10);
        if (aElementIds.includes(key)) {
          acc[key] = oAISuggestedElementData[key];
        }
        return acc;
      }, {});
      let bProductMatch = false;
      let fnProductIdFilter;
      const aProductLists = Object.values(oSelectedElementIds).map(aProducts
=>        aProducts.map(oProduct => oProduct.Id)
      );
      let aCommonProductIds;
      if (aProductLists.length === 1) {
        aCommonProductIds = aProductLists[0];
      } else {
        aCommonProductIds = aProductLists.reduce((a, b) => a.filter(c =>
b.includes(c)));
      }
      if (aCommonProductIds.length) {
        bProductMatch = true;
      }
      fnProductIdFilter = this.productIdFilter.bind(this, aCommonProductIds);
      return {
        suggestionFilter: bProductMatch ? fnProductIdFilter : null,
        matchBySuggestion: bProductMatch,
        informationMessage: bProductMatch ? ('MessageInfoSuggestionByAI') :
null,
        errorMessage: null,
        AISuggestion: true
      };
    }
```

Joonis 10. Tehisintellekti soovitusobjekti loomise funktsioon

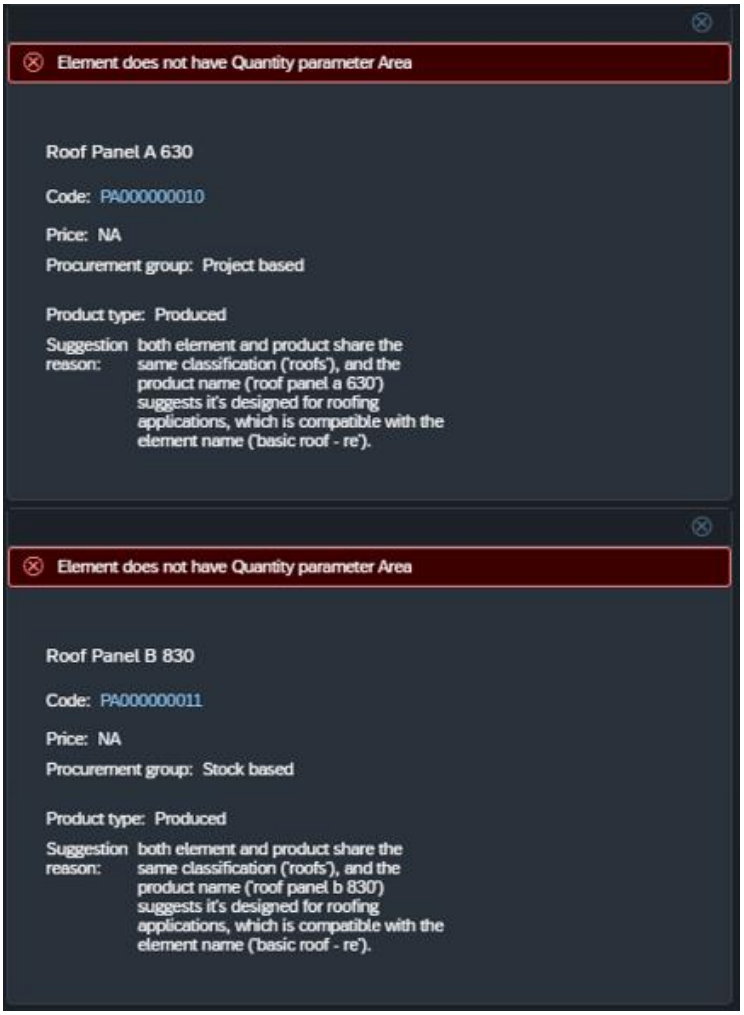
```
productIdFilter: function (aIdsToMatch, oItem) {
      return aIdsToMatch.map(id => String(id)).includes(String(oItem.Id));
    },
```

Joonis 11. Soovitusfiltri funktsioon

Loodud soovitusobjekt tagastatakse ning rakendatakse soovitusobjekti soovitusfiltrifunktsiooni toodete vaates. Peatükis 4.3.1 joonisel 3 on näha, et algne süsteem ei suutnud leida soovitusi antud elemendile. Joonisel 12 ja 13. on kuvatud tehisintellekti toodete soovitused koos nende soovitamise põhjustega.



Joonis 12. GPT-4 mudeli poolt soovitatud tooted



Joonis 13. GPT-4 mudeli soovitatud toodete lähivaade

5.5 Suuremahulised andmed

Pärast esimeste OpenAI mudelite katsetamist hakati otsima võimalusi, kus andmed oleks võimalik eelnevalt üles laadida tehisintellekti pakkuja teenuse keskkonda. See võimaldaks mudelil sama ülesannet täita ilma, et kasutaja peaks iga kord suure hulga andmeid saatma läbi API päringu. Esimeseks loogiliseks variandiks osutus OpenAI poolt arendatud assistentide teenus, mis võimaldab kasutada mudeleid ja tööriistu nagu kooditõlk, funktsioonide väljakutsumine ning failiotsing.

Raamistiku loomise ja esimeste katsetuste käigus selgus, et assistendid, kuigi pakuvad laiemat funktsionaalsust, on kulukad sarnaselt traditsioonilistele GPT-3.5 või GPT-4 API-dele. Tööriist "failiotsing", mis haldas andmefailide kasutamist, tõestas end efektiivsena, kuid mudel vaatas läbi terve faili, mis arvestati andmesisestuskulude hulka, mis omakorda tõi kaasa suurenenud kulud. Neid probleeme täpsemalt kirjeldab peatükk 6.1.2.

Lisaks uuriti teisi platvorme nagu Google Vertex AI, mis suudab suhelda BigQuery'ga SQL päringute kaudu, ja Vectara ning Command-R, mis võimaldab andmeid enne päringuid üles laadida. Kuid kuna kõik need lahendused kasutasid samu hinnastamise põhimõtteid nagu OpenAI assistendid, siis rohkem neid lahendusi ei uuritud.

5.6 Andmete filtreerimine ja grupeerimine

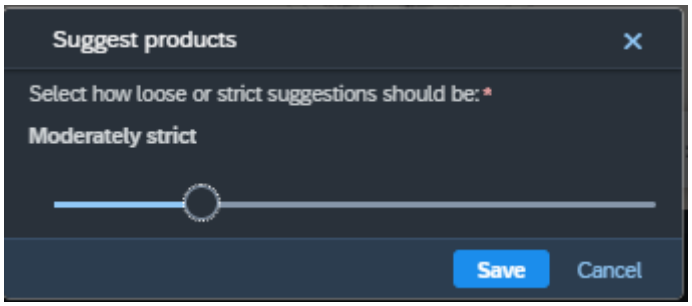
Järgmiseks sammuks oli lahenduse optimeerimine ning kasutaja jaoks teenus soodsamaks teha. Selle lahendamiseks tuli saadetavaid tooteid filtreerida ning samaväärseid elemente grupeerida.

Kasutaja saab määrata kui mitut kriteeriumi tuleb arvestada, mida saab seadistada kasutajaliideses. Minimaalselt saab seadistada, et arvestada tuleb ühte kriteeriumit, et takistada kasutajatel tarbetuid kõiki tooteid saata keelemudelile. Kriteeriumiteks olid:

1) Toote põhiparameeter on elemendi parameetrite seas
2) Toote ja elemendi ja koodid on võrdsed
3) Toote ja elemendi klassifikatsioonid on võrdsed
4) Toote ja elemendi nimes on vähemalt üks kattuv sõna

5) Toote ja elemendi parameetrid kattuvad vähemalt 50%

Antud kriteeriume kuvatakse kasutajale ranguse liugurina, joonis 14, mida liigutades arvestatakse sisse rohkem või vähem kriteeriume mis peavad läbima.



Joonis 14. Ranguse liugur

Selline lahendus tähendas aga seda, et selline kontroll tehti kõikide elementide pihta, mille kohta taheti saada soovitusi tehisintellektilt. Mida rohkem elemente, seda rohkem kattuvaid tooteid. Toodete filtreerimine on kõige tõhusam kui valida üks või mitu samaväärset elementi, elementide koguse ja erinevate omaduste kasvades arvestatakse sisse üha rohkem tooteid mida saata, kasvatades kulukust. Joonisel 15 on välja toodud kood, mis kontrollib neid kriteeriumeid.

```
      const aFilteredTrimmedData = aTrimmedProducts.filter(oProduct =>
{
        return aUniqueElements.some(oElement => {
          function tokenize (sName) {
            return new
Set(sName.toLowerCase().split(/\s+/).filter(sToken => sToken.length >
2));
          }
          const bHasValidDefaultParameter = oProduct.DefParam != null
&& oElement.Params.includes(oProduct.DefParam);
          const bProductCodeMatch = oProduct.Code ===
oElement.ProductCode;

          const sProductTokens = tokenize(oProduct.Name);
          const sElementTokens = tokenize(oElement.Name);
          const aOverlap = [...sProductTokens].filter(sToken =>
sElementTokens.has(sToken)).length;
          const bTokenMatch = aOverlap > 0;
          const bClassificationMatch = oProduct.Class ===
oElement.Class;
          let bParameterOverlap = false;
          if (oProduct.Params) {
            const productParameterSet = new Set(oProduct.Params);
            const elementParameterSet = new Set(oElement.Params);
            const intersectionSet = new
Set([...productParameterSet].filter(x => elementParameterSet.has(x)));
            const iOverlapPercentage = intersectionSet.size /
productParameterSet.size;
            bParameterOverlap = iOverlapPercentage >= 0.5;
          }

          let iTrueConditions = 0;
          if (bHasValidDefaultParameter) iTrueConditions++;
          if (bProductCodeMatch) iTrueConditions++;
          if (bTokenMatch) iTrueConditions++;
          if (bClassificationMatch) iTrueConditions++;
          if (bParameterOverlap) iTrueConditions++;

          const iSliderValue = Kasutaja poolt seadistatud
kriteeriumite rangus.

          return iTrueConditions >= iSliderValue;
        });
      });
```

Joonis 15. Toodete filtreerimine

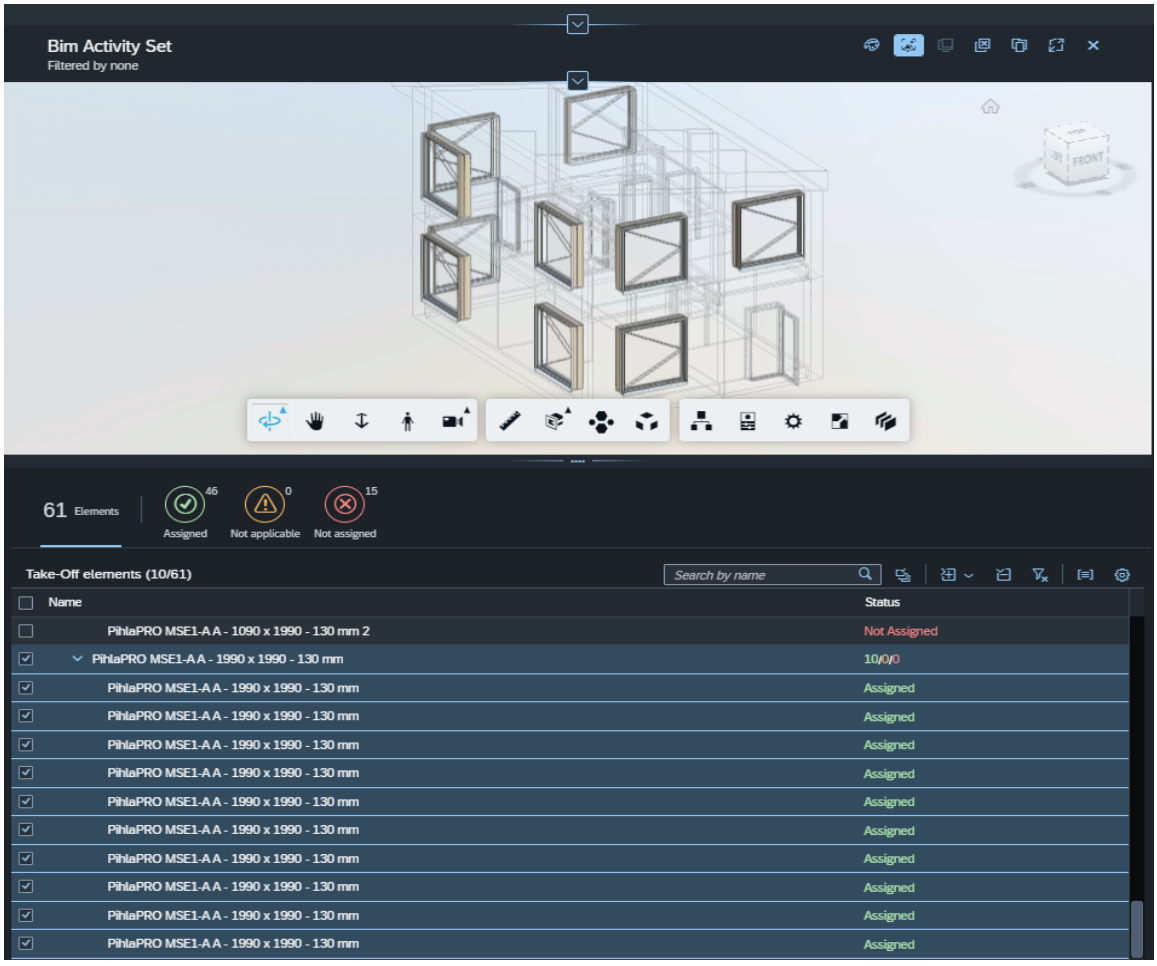
Järgmiseks säästukohaks oli elementide grupeerimine. Tihtipeale päritakse andmeid samaväärsetele elementidele, saades samu tooteid. Selle asemel, et mudelile saata mitu samaväärset elementi, kaardistati enne saatmist kõikide saadetavate elementide Id,

filtreeriti sisse samaväärsetest elementidest vaid üks ning hiljem määrati ülejäänutele samaväärsetele elementidele samad tootesoovitused. Joonisel 16 olev kood kontrollib kas elemendid on samaväärsed.

```
areElementsEquivalent: function (oElement1, oElement2) {
      const { Id: id1, Parameters: param1, ...rest1 } = oElement1;
      const { Id: id2, Parameters: param2, ...rest2 } = oElement2;
      if (param1) param1.sort();
      if (param2) param2.sort();
      const fullRest1 = { ...rest1, Parameter: param1 };
      const fullRest2 = { ...rest2, Parameter: param2 };
      return JSON.stringify(fullRest1) === JSON.stringify(fullRest2);
    },
```

Joonis 16. Elementide samaväärsuse kontrollfunktsioon

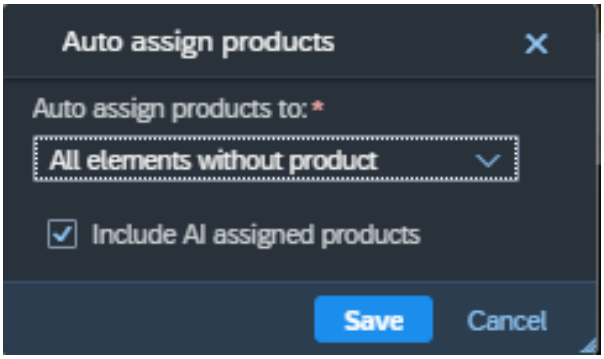
Joonisel 17. on valitud aknad, mis on samaväärsete omadustega. Nendest elementidest saadetakse tehisintellektile vaid üks, millele küsitakse tootesoovitusi ning hiljem määratakse teistele elementidele samad soovitused, vähendades päringu kulu.



Joonis 17. Elementide grupeerimine

5.7 Automaatsel määramisel tehisintellekti soovitatud toodete kaasamine

Olemasolev automaatne määramine kasutas esialgse lahenduse soovitatud tooteid ning kontrollis kas neid on võimalik määrata. Sellesse lahendusse integreeriti ka võimalus kaasata varasemaid tehisintellekti poolt soovitatud tooteid, joonisel 18 olev märkeruut.



Joonis 18. Tehisintellekti soovituste kasutamine automaatsel määramisel

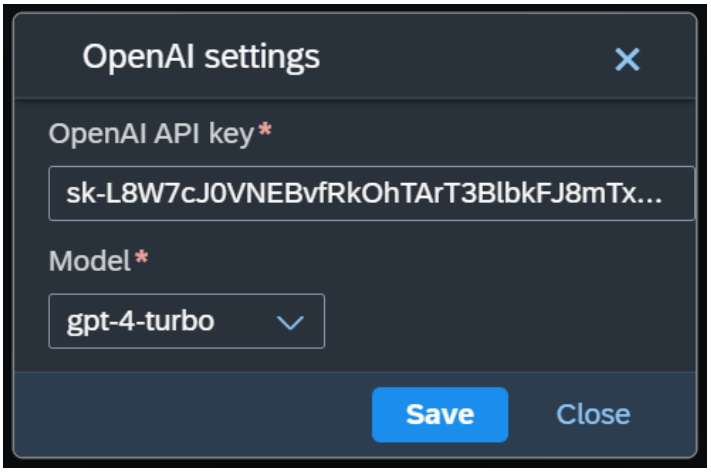
Tehisintellekt soovitas sageli mitut toodet ühe elemendi jaoks, mis tõstatas küsimuse, kuidas valida kõige sobilikum toode automaatseks määramiseks elemendile. Selle probleemi lahendamiseks kasutati samuti tehisintellekti mudelit, mis analüüsis iga toote sobivust elemendile. Mudeli päringule lisati nõue, et iga toote kohta tuleb esitada hinnang skaalal 1-10 (joonis 8), mis peegeldab toote ühilduvust määratud elemendiga. Kasutajaliidese vaates kohandati tulemused nii, et tooted sorteeritakse esmalt hinnangu ja seejärel tootekoodi järgi, tagades, et kõrgeima hinnanguga toode oleks alati esimesena nähtav.

Olemasolev lahendus määras elemendile tooteid vaid juhul, kui soovitatud tooteid oli üks, ehk sellele elemendile filtreeritud toodete massiivist valiti esimene toode. Tehisintellekti soovitatud toodete määramiseks tuli teha kontroll sellesse lahendusse, et kui elemendil on tehisintellekti poolt soovitatud tooteid ja neid on mitu siis tuleb määrata esimene toode neist, ehk kõige kõrgema hinnanguga toode.

5.8 OpenAI seadete salvestamine

Rakenduse arendamisel oli vaja anda kasutajatele võimalus sisestada oma OpenAI API võti ja valida OpenAI teenustega suhtlemiseks eelistatud mudel. Selle nõude täitmiseks

kavandati ja juurutati kasutajaliides, mis võimaldab kasutajatel mugavalt sisestada oma API-võti ja valida mudel saadavalolevate mudelite loendist, joonis 19.



Joonis 19. OpenAI seadete salvestamine

Lisaks arendati nende kasutajaeelistuste tõhusaks haldamiseks tagarakenduses välja CRUD (loomine, lugemine, värskendamine, kustutamine) funktsionaalsus. Töökindluse tagamiseks juurutati süsteemis ka terviklik veahaldus, joonis 20. Veahaldus arvestab kõigi OpenAI API poolt tulevate võimalike veakoodidega, pakkudes kasutajatele selget tagasisidet ja juhiseid võimalike probleemide lahendamiseks. Selline veahaldus tagas selle, et kasutajad saaksid oma OpenAI konfiguratsioonid meie süsteemi sujuvalt integreerida.

```
function handleOpenAIError(e) {
	let errorMessage = "An unexpected error occurred while
processing your request.";
	if (e.status) {
		switch (e.status) {
		case 400:
			errorMessage = "Maximum context length reached.
Consider choosing another model or select less elements.";
			break;
		case 401:
			errorMessage = "Authentication failed. Please check
if your API key is correct.";
			break;
		case 403:
			errorMessage = "Access Denied. You are trying to
access the service from a blocked region.";
			break;
		case 429:
			if (e.message && e.message.includes("rate limit"))
{
				errorMessage = "Rate limit exceeded. Please
slow down your request rate.";
			} else {
				errorMessage = "Quota exceeded. Please check
your plan and billing details.";
			}
			break;
		case 500:
			errorMessage = "Server error. Please try again
later.";
			break;
		case 503:
			errorMessage = "Service unavailable. The server is
currently overloaded. Please try again later.";
			break;
		default:
			errorMessage = `${e.message}`;
		}
	}
	return errorMessage;
}
```

Joonis 20. OpenAI veahaldus

6 Analüüs ja järeldused

Selles peatükis antakse ülevaade arenduse käigus tekkinud väljakutsetest ja saavutatud tulemustest. Peatükis käsitletakse põhjalikult, kuidas erinevad lahendused ja meetodid on mõjutanud projekti kulgu ja lõpptulemusi, ning millised järeldused sellest protsessist on tehtud. Arutletakse nii tehniliste kui ka äriliste aspektide üle, et pakkuda terviklikku ülevaadet projekti edukusest ja võimalustest tulevikus. Samuti käsitletakse tehisintellekti rakendamise olulisust ettevõtetes, tuues välja parimad praktikad ja soovitused tõhusaks integreerimiseks.

6.1 Tulemuste analüüs

Selles peatükis käsitletakse peamisi väljakutseid, millega arenduse käigus silmitsi seisti, ning analüüsitakse saavutatud tulemusi ja nende mõju projektile.

6.1.1 Suurte keelemudelite võimekus

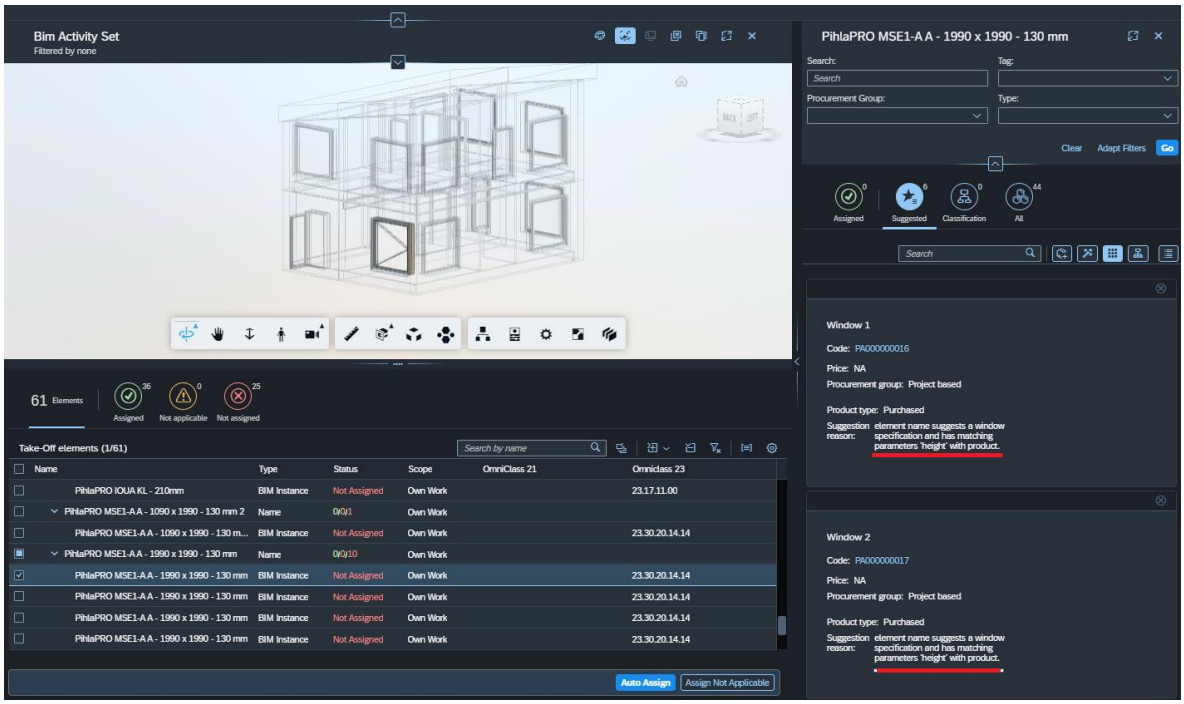
Kõige võimsamad keelemudelid, välja toodud peatükis 2.2 Tabel 1 suudavad keeruliste ülesannetega väga hästi hakkama saada. Antud töö kontekstis üritasid LLM mudelid leida elementidele tootesoovitusi nende nime, klassifikatsiooni, tootekoodi ja parameetrite ühilduvuse ja sobivuse alusel. Tihtipeale ei olnud kõikidel elementidel ning toodetel olemas kõiki ülal nimetatud andmeid ning tuli leida loogilisi seoseid siiski toodete nimede ja/või toodete nimedes sisalduvate dimensioonide kaudu, millega said võimsamad keelemudelid hästi hakkama. Katsetusi viidi läbi mitmete LLM'idega, kasutades *sandbox* keskkondi igal platvormil.

Probleemid kerkisid rohkem esile kui katsetati sama sisendiga nõrgemaid keelemudeleid, nagu ka GPT-3-turbo, mis osutasid antud ülesande jaoks kohati ebapiisavaks.

Erinevaid ettevõtteid kes tegelevad ehitusega on palju, ning kordades rohkem on tooteid abstraktsete ning tehniliste nimede ning koodidega. Selliste andmetega elementide ning toodetega ümber käies on tihti vaja eelnevaid või projekti spetsiifilisi nõudvaid teadmisi, mis teeb toodete soovitamise väga keeruliseks. Näiteks sooviti tootesoovitusi elemendile

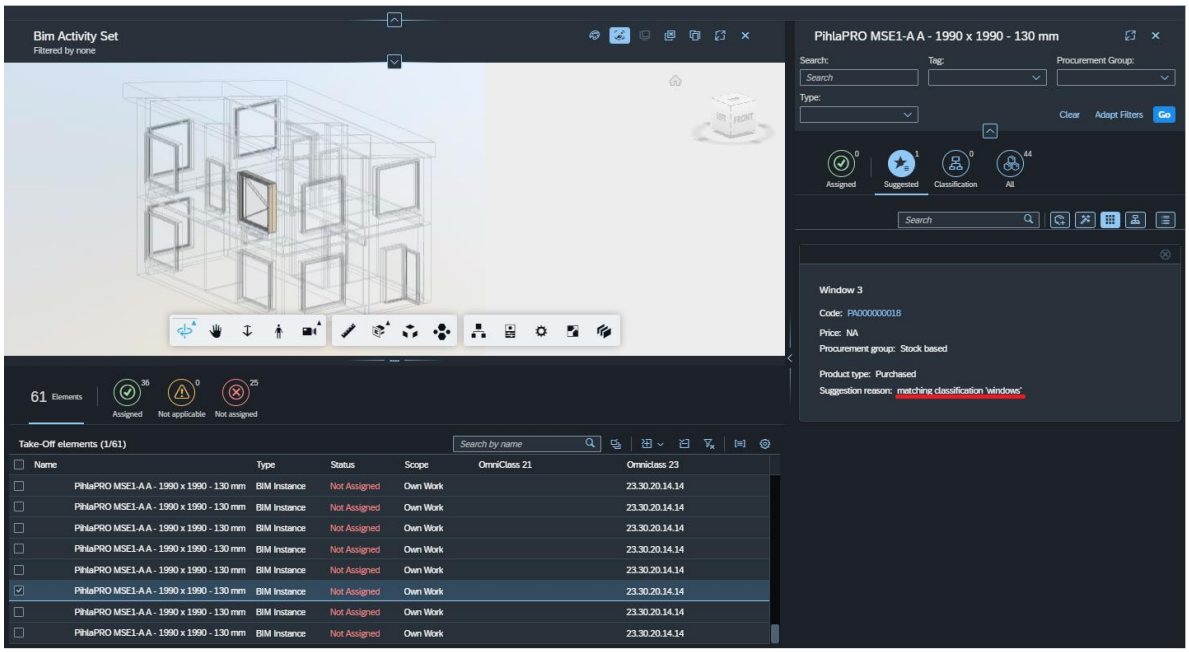
nimega "Name":" PihlaPRO MSE1-A A - 1990 x 1990 - 130 mm" ning parameetriga "Params":[Count,Module,Level,Panel,Height]}]. Tegemist on aknaga, mida kirjeldab tootenimetus PihlaPRO MSE1-A ning ka nimes sisalduvad dimensioonid "1990 x 1990 - 130 mm ", lisaks kaasas olevad parameetrid.

GPT-4-turbo tõi soovituseks 6 erinevat akent, tuues iga soovituse põhjuseks - „Element name suggests a window specification and has matching parameters 'Height' with product“, joonis 21. Mõlemad olid tõesed väited. Vastused olid mitmete kordade peale väga ühtlased ning usaldusväärsed.



Joonis 21. GPT-4 soovitused

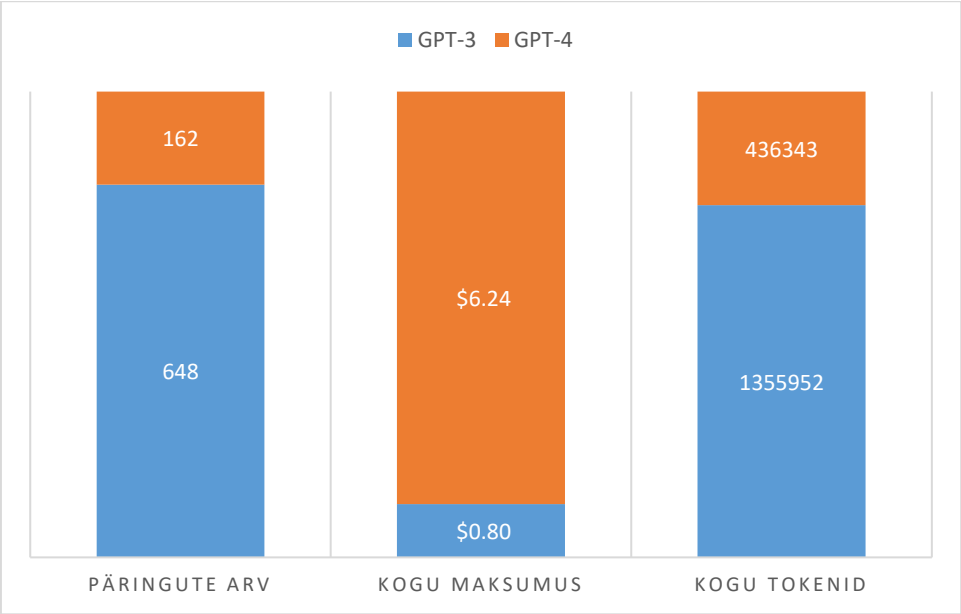
GPT-3-turbo vastused olid aga rohkem ettearvamatud. Mudel soovitas peamiselt ainult ühte akent elemendile, kuid mõtles välja põhjendusi nagu ühtivad toote- või klassifikatsioonikoodid, joonis 22. Nagu näha elemendi andmetest, ei ole antud elemendil üldse tootekoodi ega klassifikatsiooni, ehk antud põhjendused on hallutsineeritud, millest räägiti pikemalt peatükis 2.4.



Joonis 22. GPT-3 soovitused

6.1.2 Lahenduse maksumus

Kokku tehti arenduse käigus 810 päringut tootesoovituste kohta, mis läks maksma 7.04$. GPT-4 mudeli puhul tuli keskmiseks päringu hinnaks 0.038$ ning 1000 *token*'i ehk ligikaudu 750 sõna [14] hinnaks 0.0143$. GPT-3 mudel tuli keskmiseks päringu hinnaks 0.0012$ ning 1000 *token*'i hinnaks 0.0006$. Päringud mudelite järgi on välja toodud joonisel 23.



Joonis 23. Päringud

Projektis kus on 61 elementi ning 44 toodet tuli kõikidele elementidele tootesoovituse saamiseks pärast toodete filtreerimist ja elementide grupeerimist sisendi suuruseks 1806 *token*'it. Ühele tootele soovituse saamiseks tuli keskmiselt sisendi suureks 769 *token*'it. Enne grupeerimist ja filtreerimist oleks sisendi maksumus olnud kõiki elementide puhul olnud 3770 *token*'it ning ühe elemendi puhul 1639 [18].

Analüüsi käigus on näha, et maksumust suudeti üle poole võrra alandada andmetöötlusmeetoditega, kuid selgineb ka, et üksikule elemendile toote soovituste pärimine võrreldes kõikide toodetega on kallis. Ühe elemendi toote soovitused on ligikaudu 40% kõikide elementide tootesoovituse päringust. Erinevate projektide andmetega võivad *token*'ite suhted muutuda.

Antud näited on tehtud kõige minimaalsemate kriteeriumitega, ehk ainult 1 kriteerium. Andmekvaliteedi paranedes ning toodete filtreerimise ranguse tõstes on võimalik lahenduse efektiivsust tõsta.

6.1.3 Andmete kvaliteet

Lahenduse testimise juures tuli probleemiks olemasolevate projektide andmete terviklikkus ja kvaliteet. Saadaval olevate projektide andmed, mille peal lahendust katsetati, võisid olla puudulikud või vigased.

Peamiselt katsetati kahte mudelit, tavalist moodulmaja ning torustike mudeleid, mille kvaliteet oli hea. Mida kvaliteetsemad andmed keelemudelile esitatakse, seda täpsemaid ja asjakohasemaid tootesoovitusi on võimalik saada. Tihtipeale ei ole projektide andmed korrektsed või võivad olla lausa puudulikud. Sellepärast ei suutnud mitmes olukorras algne tootesoovitus süsteem midagi soovitada. Uus lahendus seevastu suutis olla loovam -rakendada olukorra põhiselt erinevaid kriteeriume ning leida loogilisi seoseid toote nimede ja dimensioonide põhjal.

Kokkuvõttes võib öelda, et kuigi süsteem pakub kasutajale rohkem võimalusi tootesoovitusteks, on korrektsete ja terviklike andmete olemasolu peamine, et süsteem efektiivselt toimiks.

6.1.4 Kulu-tõhususe suhe suurte andmekogumitega suurte keelemudelite kasutamisel

Töö üheks suureks osaks oli leida piisavalt võimas keelemudel, mis suudaks keerukaid andmeid töödelda ning oleks kuluefektiivne. Lahenduses tuli saata korduvalt palju tooteandmeid, et keelemudel suudaks võimalikuid täpseid soovitusi tagastada elementidele. Selline lähenemine võib aga olla väga kulukas, kuna ehitusettevõtetel võib olla tuhandeid tooteid ning tihtipeale otsitakse konkreetset ühte või paari toodet soovitavatele elementidele.

Sellise probleemi lahenduseks üritati leida platvormi, kus oleks võimalik andmed eelnevalt üles laadida, et vähendada sisendandmete maksumust. Lisaks oleks selline lahendus abiks ka hierarhilistele andmete kasutamisele, kus kasutajad saavad globaalseid ja projektipõhiseid andmeid eraldi või koos mudelile kontekstiks anda. Antud hierarhilise konteksti loomise eesmärgiks oleks ka võimalus keelemudelil kasutada spetsiifilise projekti detailset ehitusplaani infot, mis aitaks tootesoovitamisega.

Katsetades OpenAI assistentide APIt selgus, et kuigi selline andmete üles laadimise võimalus on olemas, on sellel märgatavad kulud. Keelemudel otsis üleslaetud andmetest vastavad tooted üles, kuid luges terve faili sisu sisendiandmete maksumuseks. Sellest järeldus, et vahet ei ole kas laadida andmed üles või anda need iga kord API päringuga kaasa. Lisaks sellele tuli maksta ka iga päeva eest millal assistendil tööriist „failiotsing“ on lubatud ning assistentidega suhtlev API võis kasutada kõiki eelnevalt küsitud küsimusi ja vastuseid, mis samuti lisandus sisendandmete kulude hulka ning tegid pärimise kallimaks. Tabelis 3 on välja toodud selliste tööriistade maksumused.

Peatükis 5.5 mainitud teised teenusepakkujad kasutasid sarnaseid hinnastamise mudeleid mis OpenAI assistendid.

Tabel 3. OpenAI assistendi tööriistade maksumused [14]

Tööriist	**Maksumus**
Kooditõlk	$0.03 / sessioon
Failiotsing	$0.10 / GB salvestusruumi päeva kohta

6.1.5 Promptid

Esialgsed *prompt'*id andsid arusaama põhiülesandest mida oodati, kuid sellel puudus spetsiifilisus andmestruktuuri, toote soovitamise põhjuste ning väljundi vormingu osas. See probleem nõudis pidevat *prompt'*ide täpsustamist, et parandada soovituste täpsust ja asjakohasust.

Optimeerimine on keerukas, sest peab leidma tasakaalu detailide ja päringu maksumuse vahel; liiga pikk ja detailne *prompt* suurendab kulusid ning võib vahel prioritiseerida liialt teatud osa *prompt'*ist, samas kui lühike ja üldine võib anda soovimatuid tulemusi. Lisaks on andmestruktuuri standardiseerimine ja täpsustamine oluline. Korrektselt määratlemata sisend- ja väljundformaadid võivad viia andmetöötlusvigadeni.

Üks suurim väljakutse on ka tehisintellekti hallutsinatsioonid (peatükk 2.3). See on eriti problemaatiline rakendustes, kus täpsus ja faktipõhisus on kriitilised. Seega peab süsteemi arendamisel arvestama mehhanismidega, mis tuvastavad ja korrigeerivad võimalikke eksitavaid tulemusi, et tagada süsteemi usaldusväärsus ja tõhusus. Antud ülesande kontekstis täpsustati *prompt'*is, et tuleb ka anda hinnang 1-10 sobivuse põhjal, mis aitab sobivamad tooted esile tuua.

Kokkuvõttes nõuab efektiivse soovitussüsteemi loomine pidevat kohandamist ja iteratsiooni, et kohanduda muutuvate nõudmistega ja parandada kasutajakogemust, hoides samal ajal kontrolli all kulusid ja tagades andmete täpsuse. Lõplik *prompt* on välja toodud lisas 2.

6.2 Lahenduse vastavus kriteeriumitele/nõuetele

Arendusprotsessi käigus täideti kõik peamised eesmärgid. Kõige tähtsam eesmärk, mis oli muuta süsteemi efektiivsemaks saavutati. Uus lahendus suudab leida seoseid mida range andmetöötlusmeetoditega tihti ei suudeta teha. GPT-4 mudel mis on keelemudelite võimekuse poolest üks võimsamaid, suudab täpselt ja järjepidevalt tooteid soovitada. GPT-3 mudel, mis on võimekuse poolest nõrgem, ei ole nii täpne oma soovitustega, kuid hea kvaliteediga andmete korral suudab samuti häid soovitusi anda, tehes seda 20x odavamalt kui GPT-4 mudel.

Klientidelt ei saadud lahenduse kohta tagasisidet, kuna lahendus ei ole veel *live*-süsteemis. Küll aga saadi positiivset tagasisidet tootearendusejuhilt, kes tõi esile, et uus lahendus täidab alguses seatud peamisi eesmärke ning teisejärguliste eesmärkide võimalike lahenduste ja puuduste kohta saadi targemaks.

Ärilisest vaatepunktist on süsteem vähendanud manuaalse töökoormuse tõttu kulusid ja parandanud süsteemi efektiivsust, vähendades projektide hilinemise riski. Samuti parandab süsteemi tõhusus ja kiirus klientide kasutajakogemust. Lahenduses rakendati kasutajapõhist maksevõimalust, kus maksumus sõltub mudeli ja kasutaja poolt tarbitud ressurssidest, mis muudab süsteemi paindlikumaks ja kuluefektiivsemaks.

Siiski oli ka eesmärke, mida täielikult või osaliselt ei saavutatud. Hierarhilise tehisintellekti mudelite kasutamine ei osutunud kuluefektiivseks, kuna paljud platvormid, sealhulgas OpenAI assistendid, ei pakkunud piisavalt taskukohaseid lahendusi andmete eelnevaks üleslaadimiseks ja hierarhilise konteksti tõhusaks kasutamiseks. Tänu sellele ei arendatud ka võimalust kasutajatel kaasa anda konteksti projekti/ülesande näol. Lisaks selgus, et andmete kvaliteet ja terviklikkus on lahenduse täpsuse ja usaldusväärsuse seisukohalt kriitilised. Uus lahendus pakub rohkem võimalusi ja paindlikkust, kuid selle efektiivsus sõltub suuresti esitavate andmete kvaliteedist.

6.3 Võimalused edasi arenduseks

Integratsioon tehisintellekti süsteemidega on ettevõtte protsessides kriitilise tähtsusega, et säilitada ettevõtte konkurentsivõime ja uuenduslikkus. Üks võimalus tarkvaralahenduse edasi arendamiseks on võimaldada kasutajatel erinevaid keelemudelite teenuse pakkujaid kasutada. Uued keelemudeli pakkujad kerkivad esile pidevalt, ning konkureerivad omavahel keelemudelite võimekuse ja hinna poolest. Selline lahendus garanteeriks, et kasutajal on võimalik kasutada kõige võimsamat ning odavamat keelemudelit.

Üheks teisejärguliseks eesmärgiks seati 4.4 peatükis arendada võimekus kasutajatel konteksti luua projektide või ülesande dokumentide näol. Kuigi seda eesmärki ei täidetud lahenduse kalliduse pärast, on see siiski väärt lisa, mis võib maandada keelemudeleid spetsiifilisse projekti või ülesande konteksti, muutes lahendust efektiivsemaks.

Lisaks oleks võimalik kaaluda RAG (Retrieve-and-Generate) meetodi ja vektorandmebaaside kasutamist. RAG meetod ühendab tehisintellekti mudelite võimekuse andmebaasidega, et otsida ja hankida relevantset teavet suurest andmekogumist ning kasutada seda teabe genereerimiseks, parandades täpsust ja vähendades hallutsinatsioonide arvu. Vektorandmebaasid võimaldavad efektiivsemalt hallata ja otsida sarnaseid andmeid, kasutades vektorkujutisi, mis parandaks süsteemi üldist jõudlust ja täpsust. Kuigi nende lahenduste integreerimist ei jõutud antud projekti raames lähemalt uurida integreerimiskeerukuse ja ajalise piirangu tõttu, on need siiski perspektiivikad võimalused tuleviku arendusteks.

Kokkuvõtteks on võimalus kasutada antud lahenduse struktuure teistes ettevõtte protsessides, kus on kasutajal liigset käsitsi tööd. Sellised automatiseeritud lahendused tagavad, et kasutajal oleks mugav ning efektiivne kasutada rakendust, tõstes selle kliendisõbralikkust.

6.4 AIRE ettevõtete analüüs

Kvalitatiivse uuringu tulemusel AIRE esindatavaga andis hea ülevaate Eesti ettevõtete üldistest väljakutsetest tehisintellekti rakendamisel. Uuriti AIRE projektide käekäiku, peamiseid ettevõtete väljakutseid, väikeste ja keskmiste/suurte ettevõtete erinevusi väljakutsete näol ning soovitusi ettevõtetele, kes tehisintellekti rakendada soovivad.

AIRE sõnul on vähe ettevõtteid, kes on koheselt valmis tehisintellekti rakendamiseks, nende hinnangul 5-10%. Peamiseks probleemiks on ettevõtete digitaliseeritus, ERP juurutamise pikk aeg ning üleüldiste eeldustehnoloogiate kasutuselevõtt. Paljudes organisatsioonides ei ole töökäsud digitaalsed, majandatakse palju paberil mis raskendab selliste lahenduste kasutuselevõttu. AIRE esimeseks ülesandeks on ettevõtete digiküpsuse hindamine. Alles peale seda on mõistlik hinnata tehisintellekti otstarbekust ning anda neutraalne ülevaade, mis on täna mõistlik ning teha tasuvusanalüüs.

Peamiste väljakutsete kohta toob AIRE veel välja, et töötajate ning oskustööliste leidmine on raske, paberimajandus eksisteerib veel paljudes organisatsioonides, protsessid ja infoliikumine ei ole digitaliseeritud. Suur probleem on üleüldine teadmatus ning tehisintellekti vajaduse hindamise võimekus.

Suurte, väikeste ja keskmiste ettevõtete kohta ei leitud, et oleks suuri erinevusi. Peamiselt on kõik juhtkonna ning ettevõtte kultuuri taga kinni- kas digitaliseerimine on äristrateegiasse sisse kirjutatud. Välja toodi, et suurtel ettevõtetel on vahendeid rohkem, et selliseid lahendusi katsetada ning integreerida, kuid väiksematel on tihtipeale kergem ja kiirem ümber teha neid protsesse. Lisaks on riigi poolt mitmeid meetmeid, mis toetavad digitaliseerimist ettevõtetes [19].

Kokkuvõtteks toob AIRE välja, et tuleb targalt planeerida tehisintellekti kasutuselevõttu. Hinnata otstarbekust, võimekust seda kasutusele võtta. Eesmärk tuleb võtta kasvuks, kui tehisintellekti ei uurita ega rakendata on võimalus, et konkureerivad organisatsioonid teevad seda ning on vaid aja küsimus millal neid konkurentsist välja süüakse. Maha jäämine toimub järk-järgult, kuid lõpp tuleb kiirelt.

6.5 Ettepanekud tehisintellekti rakendamiseks ettevõtetele

Järgnevalt on toodud üldised soovitused ettevõtetele, kuidas tehisintellekti edukalt integreerida, tuginedes AIRE soovitustele ja käesoleva töö järeldustele.

AIRE soovitused:

1. **Otstarbekuse hindamine**: väikeettevõtetel on oluline esmalt hinnata, kas tehisintellekti kasutuselevõtt on nende jaoks otstarbekas ja kas neil on selleks vajalik võimekus. Planeerimine peab olema tark ja läbimõeldud, arvestades ettevõtte ressursse ja vajadusi.

2. **Teadlikkuse tõstmine**: Eestis on olemas vajalik võimekus ja toetus tehisintellekti rakendamiseks, näiteks EAS-i kaudu [19]. Ettevõtted peaksid aktiivselt otsima ja kasutama neid võimalusi, et suurendada oma teadlikkust ja oskusi tehisintellekti valdkonnas.

3. **Lahenduste kaardistamine:** kui on selge, kus tehisintellekti rakendada, tuleks uurida, kas vastav lahendus on juba olemas, koostööd teha teiste ettevõtetega. Kui see pole saadaval, tuleks kaaluda, kas on mõistlik ja tasuv arendada lahendus ise või kasutada olemasolevaid mustreid ja mudeleid, nagu GPT.

4. **Töötajate koolitamine**: ettevõtted peaksid investeerima oma töötajate koolitamisse, et neil oleksid vajalikud oskused ja teadmised tehisintellekti

kasutamiseks. Järelkasvu tagamine on kriitilise tähtsusega, et ettevõte saaks pikaajaliselt tehisintellektist kasu.

5. **Tehnoloogiajuhi roll**: paljudel Eesti ettevõtetel puudub tehnoloogiajuht, kes vastutaks digitaalsete arenduste ja tehisintellekti juurutamise eest. Sellise rolli loomine ja tähtsustamine peaks olema prioriteet, et tagada edukas digitaliseerimine ja tehisintellekti integreerimine.

Tulenevalt töös läbi viidud analüüsist järelduvad järgmised ettepanekud:

1. **Võimekusega kooskõlas lahendused**: väiksematele ettevõtetele on oluline valida tehisintellekti lahendused, mis on kooskõlas nende tehniliste ja finantsiliste võimalustega. Lihtsad ja taskukohased platvormid võivad olla sobivamad esialgseteks rakendusteks [11].

2. **Tark planeerimine**: pool kogu lahenduse arendusest läks asjade testimisele ja võimaluste otsimisele, et leida kõige sobivamad lahendused. Kaardistades, millised protsessid ja valdkonnad saavad tehisintellektist kõige rohkem kasu, saab luua selge tegevuskava.

3. **Modulaarsus ja skaleeritavus**: lahenduste modulaarsus ja skaleeritavus on võtmetähtsusega, et tagada nende pikaajaline kasutamine ja lihtne kohandamine muutuvate vajaduste korral.

4. **Katsetamine ja iteratiivne arendamine**: tehisintellekti lahenduste juurutamine peaks toimuma järk-järgult, alustades pilootprojektidest ja katsetustest. See võimaldab tuvastada ja lahendada võimalikke probleeme varajases staadiumis ning kohandada lahendusi vastavalt saadud tagasisidele. Antud lahendust demoprojektide näol pakub ka AIRE.

Kokkuvõtteks, tehisintellekti edukaks rakendamiseks on vaja hoolikat planeerimist, töötajate koolitamist ja koostööd teiste ettevõtetega. Automaatsete lahenduste kasutamine võib oluliselt parandada ettevõtte efektiivsust ja konkurentsivõimet.

7 Kokkuvõte

Lõputöö eesmärk oli uurida tehisintellekti rakendamise väljakutseid ettevõtte Build.Works Technologies OÜ näitel. Probleemiks osutus ehituselementide ja toodete ebaefektiivne soovitussüsteem. Eesmärgiks oli arendada süsteem, mis kasutab tehisintellekti, et paremini soovitada ehituselementidele sobivaid tooteid.

Metoodika hõlmas suurte keelemudelite (LLM) kasutamist, nagu GPT-3 ja GPT-4, et analüüsida ja soovitada sobivaid tooteid, kombineerides olemasolevaid tehnoloogiaid ja andmetöötlusalgoritme. Lisaks viidi läbi kvalitatiivne uuring AIRE organisatsiooniga, mis keskendus ettevõtete probleemidele tehisintellekti rakendamisel.

Tulemuste põhjal valmis uus süsteem, mis suudab uus süsteem tõhusamalt ja täpsemalt soovitada tooteid, mis vastavad ehituselementidele, pakkudes ettevõttele ärilisi eeliseid ja parendatud kasutajakogemust. Uus lahendus vähendas manuaalse töökoormuse tõttu kulusid ning parandas süsteemi efektiivsust. Samuti selgus, et süsteemi täpsus sõltub suuresti esitavate andmete kvaliteedist.

Järeldustest selgus, et tehisintellekti rakendamine võib oluliselt parandada ettevõtte efektiivsust, kuid selle edukas integreerimine nõuab hoolikat planeerimist ja kvaliteetsete andmete olemasolu. Lisaks on oluline, et ettevõtted investeeriksid töötajate koolitamisse ja koostöösse teiste organisatsioonidega, et suurendada oma teadlikkust ja võimekust tehisintellekti valdkonnas.

Kasutatud kirjandus

[1] S.-L. F. W. S. K. K. J. a. T. W. Wamba-Taguimdje, „Influence of artificial intelligence (AI) on firm performance: the business value of AI-based transformation projects,“ *Business Process Management Journal,* 12 May 2020.

[2] Statistikaamet, „Tehisintellekti tehnoloogiate kasutamine ettevõtetes on tõusutrendis,“ Statistikaamet, 15 September 2023. [Võrgumaterjal]. Available: https://www.stat.ee/et/uudised/tehisintellekti-tehnoloogiate-kasutamine-ettevotetes-tousutrendis. [Kasutatud 12 Märts 2024].

[3] M. j. k. m. Riigikantselei, „Eesti tehisintellekti kasutuselevõtu ekspertrühma aruanne,“ Riigikantselei, Tallinn, 2019.

[4] OECD, „Key issues for digital transformation in the G20,“ OECD, Berlin, 2017.

[5] OECD, „The Digital Transformation of SMEs,“ OECD Publishing, 2021. [Võrgumaterjal]. Available: https://read.oecd-ilibrary.org/industry-and-services/the-digital-transformation-of-smes_bdb9256a-en#page195. [Kasutatud 2 March 2024].

[6] E. Parlament, „Mis on tehisintellekt ja kuidas seda kasutatakse?,“ Euroopa Parlament, Tallinn, 2023.

[7] B. C. R. A. a. S. S. A. F. Mohammad, „LLM/GPT Generative AI and Artificial General Intelligence (AGI): The Next Frontier,“ IEEE Conference Publication, 24 July 2023. [Võrgumaterjal]. Available: https://ieeexplore.ieee.org/document/10487188. [Kasutatud 22 April 2024].

[8] Cloudflare, „What is a large language model (LLM)?,“ Cloudflare, [Võrgumaterjal]. Available: https://www.cloudflare.com/en-gb/learning/ai/what-is-large-language-model/. [Kasutatud 24 April 2024].

[9] Vellum, „LLM Leaderboard,“ Vellum, 2024. [Võrgumaterjal]. Available: https://www.vellum.ai/llm-leaderboard. [Kasutatud 22 April 2024].

[10] S. J. M. K. Ziwei Xu, „Hallucination is Inevitable: An Innate Limitation of Large Language Models,“ Arxiv, 22 January 2024. [Võrgumaterjal]. Available: https://arxiv.org/abs/2401.11817. [Kasutatud 25 April 2024].

[11] C. P. Ruiqi Wei, „Artificial intelligence and SMEs: How can B2B SMEs leverage AI platforms to integrate AI technologies?,“ *Industrial Marketing Management,* November 2022.

[12] P. P. Ray, „ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope,“ *Internet of Things and Cyber-physical Systems,* pp. 121-154, 2023.

[13] OpenAI, „Platform,“ OpenAI, [Võrgumaterjal]. Available: https://platform.openai.com/docs/. [Kasutatud 12 March 2024].

[14] OpenAI, „Pricing,“ OpenAI.

[15] OpenAI, „OpenAI blog,“ OpenAI, 25 January 2024. [Võrgumaterjal]. Available: https://openai.com/blog/new-embedding-models-and-api-updates. [Kasutatud 20 March 2024].

[16] A. -. A. &. R. Estonia, „AIRE,“ AIRE, 6 May 2024. [Võrgumaterjal]. Available: https://aire-edih.eu/. [Kasutatud 8 March 2024].

[17] OpenAI, „OpenAI Node API Library,“ Npm, [Võrgumaterjal]. Available: https://www.npmjs.com/package/openai. [Kasutatud 15 March 2024].

[18] OpenAI, „Tokenizer,“ OpenAI, [Võrgumaterjal]. Available: https://platform.openai.com/tokenizer. [Kasutatud 25 April 2024].

[19] E. j. K. ühendasutus, „Avaleht - EASi ja KredExi ühendasutus,“ EASi Ja KredExi Ühendasutus, 9 May 2024. [Võrgumaterjal]. Available: https://eas.ee/. [Kasutatud 10 May 2024].

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

Mina, Jürgen Tamm

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Tehisintellekti rakendamise väljakutsed ettevõtte Build.Works Technologies OÜ näitel“, 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.

12.05.2024

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 – Lõplik *prompt*

"Assign products to each element. First check if Element.ProductCode matches with Product.Code, if it matches, then thats the correct product, only return that product for that element. If theres no exact match, try to match Classification. Then analyze products and elements focusing on logical compatibilities in names, parameter overlap and dimensions. Include a reason why that product is compatible with that element based on name/classification/product code/parameters overlap/dimensions and a rating from 1-10 how compatible the product is. Return only a JSON formatted like: ElementData: {ElementID1: [{Id: ProductID1, Reason: Reason1, R: Rating1}, {Id: ProductID2, Reason: Reason2, R:Rating2}], }"