Tekoälyn mahdollisuudet oppimisen tukena – Chatbot erityisopiskelijoille

Mikko Särkiniemi

Haaga-Helia ammattikorkeakoulu

Tradenomi, Tietojenkäsittely

Amk-opinnäytetyö

2025

Tiivistelmä

Tekijä(t)
Mikko Särkiniemi

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Tekoälyn mahdollisuudet oppimisen tukena – Chatbot erityisopiskelijoille

Sivu- ja liitesivumäärä
48 + 1

Opinnäytetyössä kehitettiin tekoälypohjainen chatbot, jonka tarkoituksena oli tukea erityisopiskelijoiden oppimista ja ohjausta koulutusympäristössä. Työn taustalla oli Haaga-Helian opettajan havaitsema tarve luoda saavutettava ja yksilöllisesti mukautuva järjestelmä, joka hyödyntää tekoälyä tiedonhaun ja opintojen tukena. Tavoitteena oli suunnitella ja toteuttaa RAG-arkkitehtuurin (Retrieval-Augmented Generation) perustuva chatbot, joka yhdistää tiedonhakua ja generatiivista tekoälyä hyödyntävän vastausmallin.

Tietoperusta muodostui luonnollisen kielen käsittelyn (NLP) ja suurten kielimallien (LLM) teoreettisesta tarkastelusta. Lisäksi työssä syvennyttiin RAG-menetelmän keskeisiin osa-alueisiin, kuten embeddingiin, chunkingiin ja vektorihakuun. Toteutuksessa hyödynnettiin TypeScript-ohjelmointikieltä, Express-palvelinympäristöä. OpenAI:n ChatGPT-rajapintaa sekä MongoDB-tietokantaa. Järjestelmä rakennettiin vaiheittain käyttäjätarinoiden avulla ja siihen sisällytettiin hallintakäyttöliittymä tietosisällön ylläpitoa varten.

Työn tuloksena syntyi teknisesti toimiva ja pedagogisesti saavutettava chatbot, joka tarjoaa selkokielisiä vastauksia ja tukee opiskelijoiden itsenäistä tiedonhakua. Tavoitteet saavutettiin asetettujen rajausten puitteissa, vaikka automaattista testausympäristöä ja CI/CD-käytäntöjä ei sisällytetty projektiin. Arvioinnin perusteella ratkaisu soveltuu laajennettavaksi muillekin oppijaryhmille ja toimii esimerkkinä siitä, miten tekoälyä voidaan hyödyntää saavutettavasti ja eettisesti koulutuksen digitalisaatiossa.

Asiasanat
Tekoäly, generatiivinen tekoäly, chatbot, ohjelmistokehitys

Sisällys

1 Johdanto

Digitalisaatio ja tekoäly ovat viime vuosina muuttaneet merkittävästi opetuksen, oppimisen ja tiedonhankinnan käytäntöjä. Generatiiviset kielimallit, kuten ChatGPT, tarjoavat mahdollisuuden kehittää oppimisympäristöjä, joissa tiedonhaku on nopeaa, saavutettavaa ja yksilöllisesti mukautuvaa. Tekoälyratkaisut voivat toimia paitsi tiedon välittäjinä myös henkilökohtaisina ohjaajina, jotka tukevat opiskelijoita oppimisessa ja opintojen hallinnassa. Tämä muuttaa perinteisen opetuksen ja ohjauksen roolia, mutta samalla asettaa uusia vaatimuksia pedagogisesti ja eettisesti kestävien järjestelmien suunnittelulle.

Haaga-Helian ammattikorkeakoulun opettaja toimi opinnäytetyön toimeksiantajana ja tunnisti tarpeen kehittää työkalu, joka tukee erityisesti erityisopiskelijoiden oppimista. Erityisopiskelijat tarvitsevat usein yksilöllisempää ohjausta, selkokielisiä vastauksia ja järjestelmiä, jotka pystyvät mukautumaan käyttäjän osaamiseen ja kielellisiin tarpeisiin. Tekoälypohjainen chatbot tarjoaa tähän ratkaisun, sillä se voi yhdistää automaattisen tiedonhankinnan, selkeän vuorovaikutuksen ja saavutettavan käyttöliittymän.

Motivaatio työn toteuttamiseen pohjautuu myös laajempaan koulutuksen digitalisaation trendiin. Opetusalan organisaatiot tarvitsevat konkreettista ja käytännössä toimivia ratkaisuja, joilla tekoäly voidaan hyödyntää eettisesti, saavutettavasti ja pedagogisesti tarkoituksenmukaisella tavalla. Tämä opinnäytetyö vastaa tähän tarpeeseen tarjoamalla esimerkin siitä, miten RAG-arkkitehtuuria voidaan hyödyntää chatbotin toteutuksessa ja samalla huomioida oppijoiden moninaiset tarpeet.

Raportin kirjoittamisessa on hyödynnetty OpenAI:n ChatGPT tekoälysovellusta kirjoitusprosessin tukena. Tekoälyä on käytetty muun muassa synonyymien etsimiseen, lauserakenteiden ja kappalejakojen muokkaamiseen sekä tekstin rakenteen ja selkeyden parantamiseen.

1.1 Tavoitteet ja tehtävänasettelu

Opinnäytetyön keskeisenä tavoitteena on suunnitella ja toteuttaa RAG-pohjainen chatbot, jonka tarkoituksena on tukea opiskelijoita, erityisesti erityisopiskelijoita, koulutusympäristössä. Työn tavoitteet voidaan jakaa kolmeen pääkokonaisuuteen:

- **Pedagoginen tavoite**: Chatbotin avulla opiskelijat saavat yksilöllistä ja saavutettavaa ohjausta, joka tukee tiedonhakua, ohjeiden ymmärtämistä ja opintojen etenemistä. Viestintälogiikka ja kielen selkeys on suunniteltu erityisesti erityisopiskelijoiden tarpeet huomioiden.
- **Tekninen tavoite**: Keskeinen painopiste on palvelinpuolen toteutuksessa: Järjestelmän ydinlogiikka, tietokanta ja RAG-pohjainen semanttinen haku ja dynaamisten vastausten generointi. Ratkaisun tulee olla teknisesti vakaa, skaalautuva ja jatkokehitettävä.

- **Hallintatavoite**: Kehittää hallintakäyttöliittymä, joka mahdollistaa tietokannan sisällön hallinnan ilman syvällistä teknistä osaamista.

Projektissa tunnistettiin kolme keskeistä kohderyhmää: erityisopiskelijat, muut opiskelijat sekä hallintakäyttäjät. Käyttäjätarinoiden avulla määriteltiin järjestelmään toiminnallisuudet ja priorisoitiin kehitysvaiheet.

Tavoitteena oli luoda ratkaisu, joka ei ainoastaan toimi teknisesti, vaan myös tarjoaa käyttäjille pedagogisesti saavutettavan, selkeän ja helposti lähestyttävän kokemuksen.

1.2 Rajaukset

Opinnäytetyössä asetettiin selkeät rajaukset käytettävissä olevien resurssien ja ajan perusteella:

- **DevOps-käytännöt**, kuten jatkuva integrointi ja käyttöönotto (CI/CD).
- **Testaus**: Järjestelmän kattavaa automaattista testausympäristöä ei rakennettu.
- **Käyttöliittymä**: Hallintakäyttöliittymän viimeistely ja visuaalinen hienosäätö.

Näistä rajoituksista huolimatta työn keskeiset tavoitteet saavutettiin. Chatbot toimii teknisesti vakaasti, hyödyntää RAG-arkkitehtuuria ja tarjoaa pedagogisesti saavutettavan ja yksilöllisen tukimuodon opiskelijoille.

1.3 Keskeiset käsitteet

Tässä luvussa määritellään opinnäytetyön kannalta keskeiset käsitteet, joiden ymmärtäminen on oleellista työn tavoitteiden ja ratkaisun hahmottamiseksi.

Chatbot on ohjelmisto, joka simuloi ihmisen kaltaista keskustelua käyttäjän kanssa tekstin tai puheen välityksellä. Tekoälypohjaiset chatbotit hyödyntävät suuria kielimalleja ymmärtääkseen kieltä ja muodostaakseen dynaamisia vastauksia.

Suuri kielimalli (Large Language Model, LLM) on tekoäly kielimalli, joka on koulutettu käsittelemään ja tuottamaan luonnollista kieltä. Se perustuu laajoihin tekstiaineistoihin ja pystyy ennustamaan sanoja, muodostamaan lauseita ja vastaamaan kysymyksiin kielellisen kontekstin perusteella. Tunnettuja suuria kielimalleja ovat esimerkiksi OpenAI:n ChatGPT ja Googlen Gemini. LLM toimii useiden tekoälypohjaisten sovellusten, kuten chatbottien, taustalla. (Cloudflar.).

Hallusinaatio (engl. hallucination) tarkoittaa tilannetta, jossa tekoäly tuottaa virheellistä, keksittyä tai todellisuudesta poikkeavaa tietoa. Ilmiö johtuu siitä, että kielimalli pyrkii tuottamaan kielellisesti uskottavan vastauksen, vaikka sen pohjana ei olisi luotettavaa tietoa. (SAP 2024).

Retrieval-Augmented Generation (RAG) on menetelmä, jossa yhdistetään tiedonhaku ja generatiivista tekoälyä. Ennen vastauksen tuottamista järjestelmä hakee käyttäjän kysymykseen liittyvää tietoa ulkoisesta tietokannasta ja syöttää sen kielimallille. (AWS s.a.).

Embedding tarkoittaa menetelmää, jossa sanat, lauseet tai dokumentit muunnetaan numeeriseen muotoon eli vektoreiksi. Vektori on matemaattinen esitys, joka kuvaa kielen osien merkitystä moniulotteisessa avaruudessa. Embedding-menetelmän avulla kone kykenee käsittelemään kieltä matemaattisesti ja suorittamaan semanttista analyysiä, jota RAG hyödyntää tietojen hakemisessa ja yhdistämisessä. (Kumar 2024).

Vektorihaku on hakumenetelmä, joka perustuu tekstien matemaattisiin esityksiin eli vektoreihin. Toisin kuin perinteinen avainsanahaku, vektorihaku etsii merkitykseltään samankaltaista sisältöä riippumatta siitä sisältävätkö ne samoja sanoja. Semanttinen haku tarkoittaa juuri tätä merkityspohjaista tiedonhakua, jossa huomio kohdistuu sanojen ja lauseiden merkityssuhteeseen. Vektorihaku on keskeinen osa RAG-arkkitehtuuria. (Syed & Russi 2024).

Chunking tarkoittaa lähdetekstin jakamista pienempiin osiin eli tekstilohkoihin (engl. chunks), joita voidaan käsitellä tehokkaammin vektorihakua varten. Jokainen tekstilohko muunnetaan vektoriksi ja tallennetaan tietokantaan yhdessä alkuperäisen tekstin kanssa. Chunking mahdollistaa sen, että haku kohdistuu riittävän tarkkoihin ja kontekstuaalisesti yhtenäisiin sisältöihin. Tämä parantaa hakutulosten relevanssia ja RAG-mallin tuottamien vastausten laatua. (AI SDK s.a. a).

2 Luonnollisen kielen käsittely ja suuret kielimallit

Chatbottien toiminta perustuu siihen, että ne osaavat tulkita käyttäjän viestejä ja vastata niihin mahdollisimman luonnollisesti. Luonnollisen kielen käsittely (Natural Language Processing, NLP) on tässä keskeisessä roolissa, sillä sen avulla tietokone kykenee ymmärtämään ihmisten puhetta ja tekstiä. NLP yhdistää kielitieteellistä tietoa tilastollisiin menetelmiin, koneoppimiseen ja syväoppimiseen, jotta kone pystyisi tunnistamaan sanojen merkityksiä, rakenteita ja kontekseja. (Stryker & Holdsworth 2024).

Pelkkä ymmärtäminen ei kuitenkaan riitä sillä chatbotin on myös osattava tuottaa vastauksia, jotka tuntuvat luontevilta ja ihmismäisiltä. Tässä tulevat apuun suuret kielimallit (Large Language Models, LLM). Ne on opetettu valtavan suurilla tekstiaineistoilla, minkä ansiosta ne pystyvät sekä analysoimaan käyttäjän syötettä että rakentamaan siihen sopivan vastauksen. (Cloudflare s.a.).

Yhdessä NLP ja LLM tekevät chatbotista enemmän kuin pelkän kysymysten vastauskoneen. Ne mahdollistavat vuorovaikutuksen, joka on sujuvaa, joustavaa ja käyttäjän näkökulmasta aidosti keskusteltavaa.

2.1 Luonnollinen kieli ja sen käsittely (NLP)

Luonnollinen kieli tarkoittaa niitä kieliä, joita ihmiset käyttävät arkisessa viestinnässään, kuten suomea, englantia tai ruotsia. Ne ovat täynnä monitulkintaisuutta, monimutkaisia rakenteita ja vivahteita. Sanoilla voi olla useita merkityksiä ja konteksti vaikuttaa ymmärrykseen merkittävästi. Tietokoneelle tällaisen kielen tulkitseminen on haastavaa, koska sen tulee erottaa esimerkiksi sarkasmi, slangisanat tai verbin taivutus oikeassa yhteydessä.

NLP on yhdistelmä tietojenkäsittelytieteen, kielitieteen sekä koneoppimisen menetelmiä, jotka auttavat tietokonetta sekä ymmärtämään että tuottamaan ihmiskieltä. NLP:n tehtäviin kuuluvat tekstin ja puheen tunnistus, sanojen ja lauseiden rakenteen ja merkitysten analysointi sekä sisällön tuottaminen takaisin kielimuodossa. (Stryker & Holdsworth 2024).

Keskeisiä osa-alueita ovat esimerkiksi (Stryker & Holdsworth 2024):

- **Esikäsittely**: Tekstin pilkkominen pienempiin yksikköihin (Tokenisointi), sanojen perusmuodon tunnistus (Lemmatisaatio) ja osien merkitseminen (esim. substantiivi, verbi).
- **Analyysi ja ymmärtäminen**: Lauserakenteiden ja merkitysten analysointi, syntaktinen jäsentäminen ja semanttinen ymmärrys, kuten tekstin tarkoitus ja sävy.
- **Tuottaminen**: koneen tuottama teksti tai puhe, jonka tulee olla selkeää, kontekstiin sopivaa ja inhimillistä.

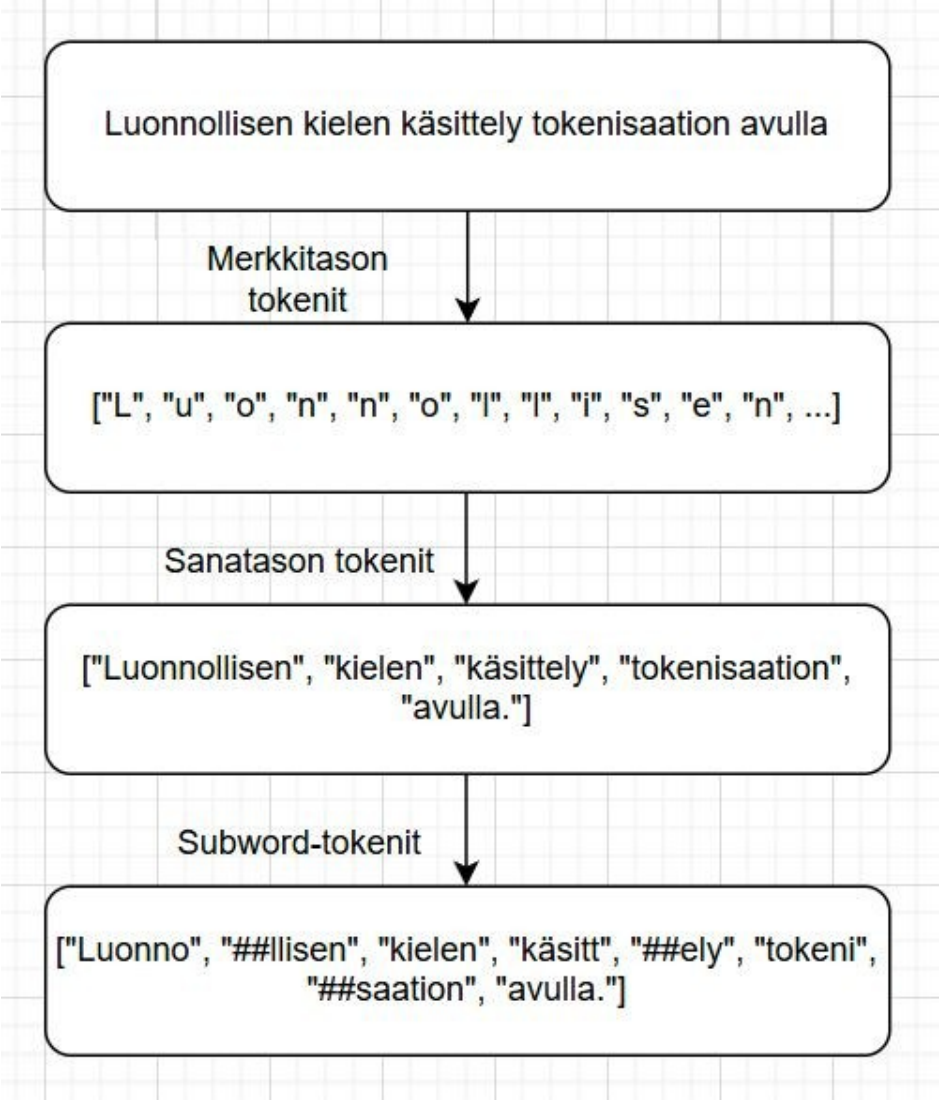
Tokenisointi on NLP:n perusvaihe, jossa jäsentämätön teksti pilkotaan pienempiin yksiköihin, eli tokeneihin. Tokenit voivat olla sanoja, merkkejä tai sanan osia riippuen valitusta strategiasta. Tämän prosessin avulla teksti saadaan muunnettua rakenteettomasta muodosta rakenteelliseksi, jolloin sitä voidaan käyttää koneoppimismalleissa. (Ashraf 2024).

Ihmiskieli on luonteeltaan monimutkaista. Esimerkiksi sama sana voi saada eri merkityksen riippuen kontekstista. Tokenisointi auttaa yksinkertaistamaan tätä kokonaisuutta jakamalla tekstin pienempiin yksiköihin, jotka voidaan esittää numeerisessa muodossa. Näin teksti voidaan syöttää algoritmeille, jotka suorittavat tehtäviä kuten luokittelua, sentimenttianalyysiä tai tekstin tuottamista. Samalla tokenit toimivat piirrevektoreina, jotka välittävät merkityksellistä kielellistä informaatiota malleille. (Ashraf 2024). Kuvassa 1 havainnollistetaan, kuinka sama lause voidaan jakaa merkki-, sana- ja subword-tason tokeneihin.

Tokenisointiin on kehitetty useita eri strategioita, jotka voidaan sijoittaa jatkumolle merkki ja sanatasojen välille (Ashraf 2024):

- **Merkkitason tokenisointi**: Teksti jaetaan yksittäisiin merkkeihin. Se käsittelee harvinaiset sanat ja virheet joustavasti, mutta vaatii paljon laskentatehoa, koska malli joutuu oppimaan sanojen rakenteen itse.
- **Sanaston tokenisointi**: Teksti pilkotaan sanoiksi, mikä mahdollistaa sanojen oppimisen. Haasteena on sanaston nopea kasvu, minkä vuoksi harvinaiset sanat usein korvataan UNK-tokenilla, jolloin tietoa menetetään.
- **Subword-tokenisointi**: Yhdistää edellä mainitut hyödyt. Yleiset sanat säilyvät kokonaisina ja harvinaiset jaetaan osiin. Tämä pitää sanaston hallittavana ja mahdollistaa monimutkaisten sanojen käsittelyn. Käytössä mm. BERT-malleissa (esim. WordPiece).

Tokenisointi on siis olennainen osa NLP putkea, sillä sen avulla teksti muutetaan muotoon, jota koneoppimismallit voivat hyödyntää tehokkaasti.



Kuva 1. Tokenisointi tapojen havainnollistaminen

2.2 Suuret kielimallit (LLM)

Suuret kielimallit ovat tekoälyohjelmia, jotka on koulutettu valtavilla tekstidatoilla, usein verkosta kerätyllä kielimateriaalilla. Näin ne oppivat tunnistamaan kielen rakenteita ja tuottamaan tekstiä, joka kuulostaa usein yllättävän ihmismäiseltä. (Cloudflare s.a.).

WotNotin blogissa kuvataan hyvin, miten LLM-pohjainen chatbot ei noudata käsikirjoituksia, vaan ymmärtää viestin kontekstin, kiinnittää huomiota vivahteisiin ja laatii vastauksen, joka tuntuu aidolta ja tarkoituksenmukaiselta. (Makadia 28.2.2025).

Vahvuuksina LLM:llä on kyky ymmärtää kieltä joustavasti ja tuottaa luonnollisen kuuloista tekstiä. Ne pystyvät säilyttämään tyylin, sävyn ja kontekstin, eikä niitä rajoita ennalta määrätty skriptipohja. Ne oppivat myös hienovaraisuuksia, kuten monikielisyyttä ja vivahteita ja ovat siten erittäin joustavia keskustelukumppaneita. (Makadia 28.2.2025; Cloudflare s.a.).

Rajoitteita ja haasteita sen sijaan ovat esimerkiksi se, että malli voi toisinaan antaa virheellistä tai epätarkkaa tietoa, jos koulutusdata on puutteellista tai virheellistä. Niiden kouluttaminen edellyttää valtavia määriä dataa ja laskentatehoa, mikä voi tehdä kehityksestä sekä kalliin että hitaasti etenevän prosessin. (Stryker & Holdsworth 2024).

Transformer-arkkitehtuuri on suurten kielimallien perusta ja se mullisti luonnollisen kielen käsittelyn tarjoamalla uuden tavan ymmärtää kieltä kontekstuaalisesti. Aiemmat mallit käsittelevät sanoja yksi kerrallaan ja saattoivat kadottaa yhteyden kaukana oleviin sanoihin. Transformer sen sijaan hyödyntää self-attention mekanismia, jonka avulla malli pystyy tarkastelemaan koko lausetta samanaikaisesti ja painottamaan sanojen merkitystä suhteessa toisiinsa. Tämä tekee siitä tehokkaan käsittelemään pitkiäkin tekstikokonaisuuksia ja tuottamaan johdonmukaista kieltä. (Cloudflare s.a; TrueFoundry 22.3.2024).

Self-attention tarkoittaa mekanismia, jossa jokaiselle sanalle lasketaan sen suhde muihin sanoihin lauseessa. Käytännössä tämä tapahtuu muodostamalla kolme erilaista vektoria: Query, Key ja Value. Näiden avulla malli voi määrittää, mitkä sanat ovat tärkeitä tarkasteltavan sanan merkityksen ymmärtämisessä. Esimerkiksi kysymyksessä ”Miten LLM toimii?”, sana ”LLM” saa suurimman painon, kun mallin täytyy tuottaa oikea vastaus. Tämä mekanismi mahdollistaa sen, että malli ei unohda lauseen alussa mainittuja asioita myöhemmin, vaan säilyttää kokonaiskontekstin. (TrueFoundry 22.3.2024).

Arkkitehtuurin toiminta rakentuu kahdesta pääosasta: encoder ja decoder. Encoder lukee ja käsittelee syötteen muuttaen sanat numeerisiksi vektoreiksi, jotka sisältävät merkityksen ja kontekstin. Decoder puolestaan tuottaa ulostulon hyödyntäen encoderin tuottamaa tietoa ja omaa self-attention mekanismia. Tämän ansioista malli pystyy esimerkiksi kääntämään tekstiä tai vastaamaan kysymyksiin tavalla, joka huomioi koko syötteen rakenteen ja merkityksen. Lisäksi transformerit hyödyntävät positional encodingia, joka antaa sanoille järjestyksen, vaikka niitä käsitellään rinnakkain. (TrueFoundry 22.3.2024).

Transformerin vahvuus piilee siinä, että se yhdistää nämä mekanismit tehokkaaksi kokonaisuudeksi. Sen avulla mallit, kuten ChatGPT, pystyvät tuottamaan tekstiä, joka ei ainoastaan kuulosta ihmismäiseltä vaan myös säilyttää johdonmukaisuuden ja kontekstuaalisuuden pitkilläkin tekstialueilla. (Cloudflare s.a.).

2.3 NLP ja LLM käytännön sovelluksissa

Tekoäly chatbotit, kuten ChatGPT, hyödyntävät NLP:tä ja LLM:ää tuottaakseen vuorovaikutteisia ja luonnollisia keskusteluja. NLP mahdollistaa käyttäjän syötteen ymmärtämisen ja merkityksen purkamisen, kun taas LLM vastaa tuottamalla kielellisesti sujuvia ja kontekstuaalisesti osuvia vastauksia.

LLM-pohjaiset chatbotit tarjoavat merkittäviä etuja erityisopiskelijoiden tukemisessa. Ne voivat auttaa yksinkertaistamaan kieltä, selittämään monimutkaisia käsitteitä ja tarjoamaan henkilökohtaista tukea oppimisessa. Esimerkiksi tutkimukset ovat osoittaneet, että AI-pohjaiset työkalut, kuten ruudun lukuohjelmat ja puheavustajat voivat parantaa saavutettavuutta ja osallisuutta erityisopetuksessa. (Kooli & Chakraoui 2025).

Vaikka LLM-pohjaiset chatbotit tarjoavat monia etuja, niihin liittyy myös merkittäviä haasteita. Yksi keskeinen ongelma on niin sanotut ”Hallusinaatiot”, joissa malli tuottaa virheellistä tai harhaanjohtavaa tietoa, joka esitetään vakuuttavasti totuutena. Toinen haaste on kielimallien vinoumat (bias), jotka voivat johtua koulutusdatasta ja vaikuttaa mallin tuottamiin vastauksiin. (SAP 2024; Caelen & Blete 2024, 25–26).

RAG-prosessi tarjoaa tehokkaan keinon vähentää sekä hallusinaatiota että vinoumia.

3 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) on tekniikka, jossa LLM:ien, kuten ChatGPT:n tuottamaa vastausta pyritään optimoimaan hakemalla ensin relevanttia tietoa ulkoisesta tietovarastosta ja syöttämällä tämä tieto LLM:lle vastauksen luomiseksi. RAG yhdistää tietohaku- ja generatiivisen vaiheen. Se mahdollistaa ajantasaisen ja organisaatiokohtaisen tiedon hyödyntämisen ilman, että koko mallia tarvitsee kouluttaa uudelleen. (AWS s.a; Reinikainen s.a.).

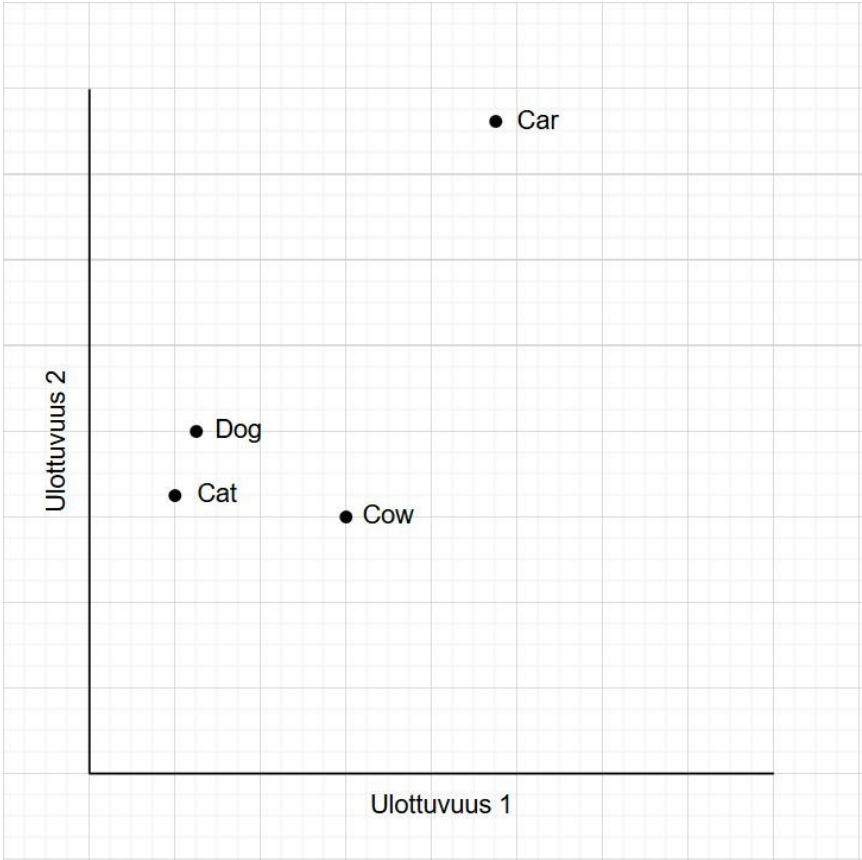
Tämä lähestymistapa on otettu käyttöön erityisesti chatbot-ratkaisuissa, asiakastukijärjestelmissä ja muissa sovelluksissa, joissa vastauksiin halutaan integroida sekä yleistä kielen tuntemusta että organisaatiokohtaisesti hallittua tietoa.

3.1 Embedding

Embedding viittaa menetelmään, jossa sanat, lauseet tai jopa kokonaiset dokumentit muunnetaan numeerisiksi vektoreiksi korkeaulotteiseen avaruuteen. Tämän avulla kone voi käsitellä kieltä matemaattisessa muodossa. Keskeinen ajatus on, että merkitykseltään samankaltaiset sanat sijoittuvat lähelle toisiaan vektoriavaruudessa, kun taas merkitykseltään eriävät sanat jäävät kauemmaksi toisistaan. (AI SDK s.a. a; Kumar 2024).

Jos esimerkiksi vektoriavaruuteen sijoitetaan sanat ”koira” ja ”kissa”, ne löytyvät läheltä toisiaan, koska molemmat liittyvät eläimiin. Vastaavasti sanat ”lehmä” ja ”auto” sijaitsevat kauempana toisistaan, sillä niiden semanttinen, eli merkitysopillinen, merkitys eroaa merkittävästi (ks. kuva 2). (Kumar 2024).

Sanavektorien välisen etäisyyden perusteella voidaan arvioida, kuinka lähellä merkitykseltään sanat ovat. Yksi yleisesti käytetty mittari onko kosinietäisyys, jossa arvo yksi tarkoittaa suurta samankaltaisuutta ja arvo miinus yksi suurta eroa. (AI SDK s.a. a).



Kuva 2. Embedding-tilassa samankaltaiset sanat ovat lähekkäin

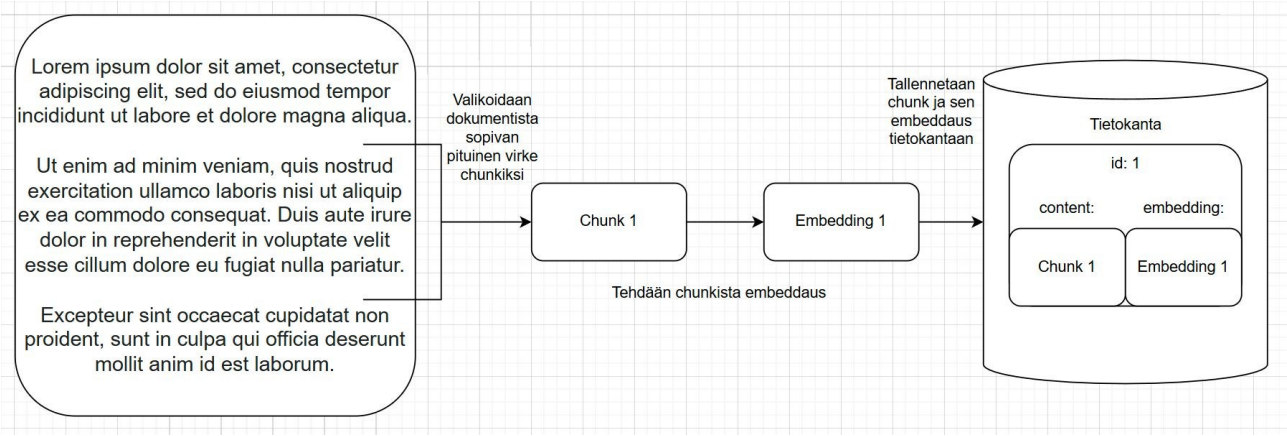
3.2 Chunking

Chunking tarkoittaa lähdemateriaalin pilkkomista pienempiin osiin, jotta sitä voidaan käsitellä ja hyödyntää tehokkaasti RAG sovelluksissa. Yleinen lähestymistapa on jakaa pidempi teksti lyhyempiin virkkeisiin, jolloin jokainen osa voidaan muuttaa vektoriksi eli embeddingiksi ja tallentaa yhdessä alkuperäisen tekstipalan kanssa tietokantaan, kuten kuvassa 3 havainnollisestaan. (AI SDK s.a. a).

Tällä menetelmällä on keskeinen rooli, koska vektori kattaa aina vain sen tekstin, joka kuuluu kyseiseen kappaleeseen. Jos chunk on liian suuri, sen sisältö menettää tarkkuutta, eikä se enää kuvaa tarkasti yksittäisiä käsitteitä. Jos chunk on liian pieni, vaarana on kontekstin menettäminen. Sopivan koon löytäminen on aina tasapainoilua spesifisyyden ja ymmärrettävyyden välillä. (Donovan 27.12.2024; Caelen & Blete 2024, 174–175).

Chunkingilla on merkittävä vaikutus tiedonhakuprosessiin. Käyttäjän kysely muunnetaan vektoreiksi ja verrataan tietokannassa oleviin tekstin pätkiin. Jos pätkät on jaettu epätasapainoisesti, niin vastausten osuvuus heikkenee. Huolellisesti toteutettu chunking

mahdollistaa tarkemmat ja relevantimmat hakutulokset. Nämä auttavat LLM:ää muodostamaan vastauksia, jotka ovat informatiivisia ja sidottuja kontekstiin. (Donovan 27.12.2024).

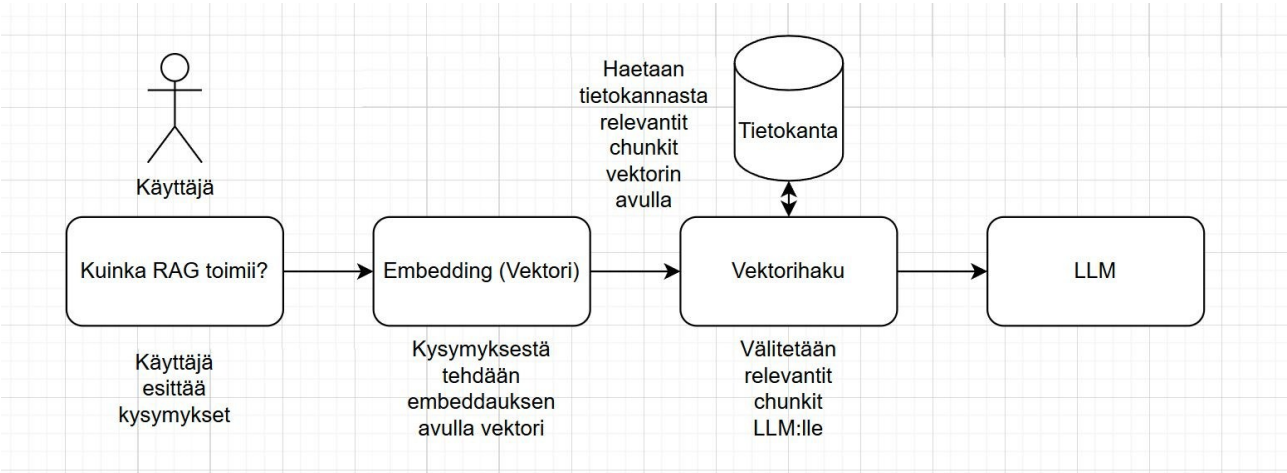


Kuva 3. Chunkkien luontiprosessi tietokantaan (mukaillen AI SDK s.a. a)

3.3 Vektorihaku

Vektorihaku on hakumenetelmä, joka perustuu embedding vektoreihin. Perinteisen avainsanahaun sijaan vektorihaku hyödyntää sanojen semanttisia suhteita: vektoriavaruudessa samankaltaiset sanat ryhmittyvät toistensa läheisyyteen. Tämän avulla haku löytää merkitykseltään lähellä olevat sisällöt, vaikka ne eivät sisältäisikään samoja sanoja. (Syed & Russi 2024).

Vektorihakua hyödynnetään erityisesti RAG sovelluksissa, joissa käyttäjän kysely muunnetaan vektoriksi ja sitä verrataan tallennettuihin chunkkeihin. LLM:lle palautetaan ne kohdat, joiden semanttinen samankaltaisuus kyselyyn on suurin. Tätä prosessia havainnollistetaan kuvassa 4. Näin LLM voi tuottaa vastauksen, joka perustuu ajantasaisiin ja ulkoisiin lähteisiin. (AI SDK s.a.).



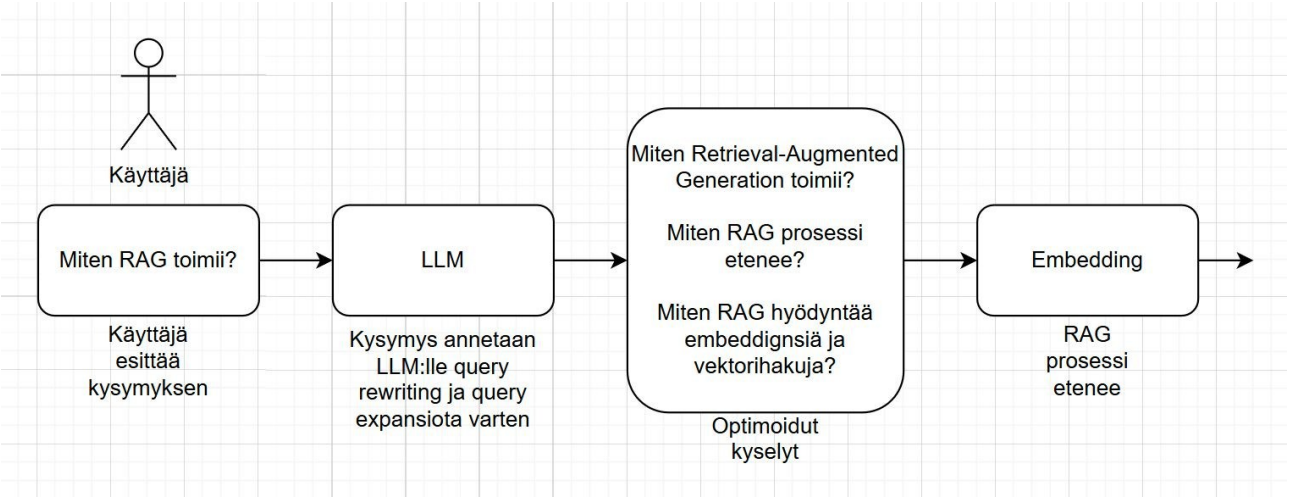
Kuva 4. Vektorihaun havainnollistaminen (mukaillen AI SDK s.a.)

3.4 Miksi Advanced RAG on parempi kuin Naive RAG

RAG-arkkitehtuuri voidaan jakaa kahteen osaan: Naive RAG ja Advanced RAG.

Naive RAG on yleisin ja yksinkertaisin RAG-arkkitehtuuri. Siinä dokumentit jaetaan ensin tekstilohkoihin (chunking), jotka muunnetaan vektoreiksi embedding-mallilla. Käyttäjän kysymyskin muutetaan vektoriksi, jonka jälkeen haetaan samankaltaisimpia lohkoja ja annetaan nämä LLM:lle vastauksen generointiin. (Caelen & Blete, 172).

Advanced RAG vie prosessin askeleen pidemmälle. Käyttäjän kyselyitä muokataan sen laajentamisella (query expansion) ja uudelleen kirjoittamalla (query rewriting). Käyttäjän kysymys voidaan antaa LLM:lle, joka muotoilee sen uudelleen ja tuottaa useita vaihtoehtoisia versioita samasta kysymyksestä. Näiden avulla vektorihaun tulokset ovat paljon laadullisempia ja osuvampia. Kuvassa 5 havainnollistetaan, miten käyttäjän alkuperäistä kysymystä voidaan optimoida tätä prosessia varten. (Caelen & Blete 2024, 173–174; Kuria 8.1.2025).



Kuva 5. Käyttäjän kysymyksen optimointi

3.5 Hyödyt ja Haitat

RAG tarjoaa useita merkittäviä etuja verrattuna pelkästään perustason kielimalleihin. Se on kustannustehokas ratkaisu, sillä sen avulla voidaan välttää kalliit ja aikaa vievät mallien uudelleenkoulutukset tietyllä aineistolla. RAG mahdollistaa myös ajantasaisen tiedon hyödyntämisen, koska se voi hakea tietoa suoraan päivittyvistä lähteistä, kuten tutkimuksista ja liittää nämä osaksi mallin tuottamaa vastausta. (AWS s.a.; Belcic 2024).

RAG parantaa käyttäjien luottamusta, sillä vastauksissa voidaan tarjota lähdeviittauksia ja siten läpinäkyvyyttä. Tämä vähentää hallusinaatioiden riskiä ja tekee järjestelmästä luotettavamman.

Kehittäjien näkökulmasta RAG antaa paremman hallinnan ja ylläpidettävyyden, sillä tietolähteitä voidaan päivittää ja ylläpitää tarpeen mukaan. Samalla voidaan huolehtia tietoturvasta ja käyttöoikeuksien hallinnasta. Kokonaisuutena RAG laajentaa tekoälyn käyttökohteita, parantaen tarkkuutta ja joustavuutta. (AWS s.a.; Belcic 2024).

Vaikka RAG:lla on monia etuja, sen käyttöönottoon ja toimintaan liittyy myös haasteita. Yksi keskeisimmistä on, että RAG:lla ei vielä ole kykyä vaiheittaiseen (iteratiiviseen) päättelyyn. Tämä tarkoittaa, ettei se välttämättä tunnista aina kaikkein relevantimpia lähteitä monimutkikkaissa kysymyksissä. Tämä voi johtaa siihen, että järjestelmä palauttaa semanttisesti samankaltaista, mutta käyttäjän tarpeisiin nähden epärelevanttia sisältöä. (Besen 2024).

Toinen rajoite liittyy aineiston rakenteeseen ja organisointiin. Jos lähdetiedot ovat huonosti jäsenneltyjä, eli ilman selkeää kategorisointia, RAG:n on vaikea löytää hyödyllistä tietoa. RAG jakaa tekoälymallien yleiset haasteet, kuten vinoumat, hallusinaatiot ja riskin hyödyntää vanhentuneita lähteitä tai virheellistä tietoa. Koska malli ei pysty itse arvioimaan tiedon luotettavuutta itsenäisesti, nämä tekijät voivat merkittävästi heikentää sen tuottamien vastausten laatua. (Besen 2024). Keskeiset hyödyt ja haitat ovat koottu taulukkoon 1.

Taulukko 1. RAG:n hyödyt ja haitat.

Hyödyt	**Haitat**
Kustannustehokkuus: Vältytään mallien uudelleenkoulutukset.	**Ei iteratiivista päättelyä**: Ei aina tunnista monimutkaisten kysymysten relevanttia tietoa.
Ajantasainen tieto: Uusimman tiedon hyödyntäminen.	**Riippuvuus datan rakenteesta**: Huonosti organisoitu rakenne vaikeuttaa tehokasta hakua.
Käyttäjän luottamus: Vastaukset voidaan varustaa lähdeviittauksilla.	**Laadun riskit**: Vinoumat (bias) ja vanhentuneen tiedon käyttö.
Kehittäjien hallinta: Mahdollista ylläpitää tietolähteitä.	
Tietoturva: Käyttöoikeuksien hallinta ja sensitiivisen tiedon suojaaminen.	
Käyttökohteiden laajentuminen: RAG soveltuu monipuolisesti eri sovelluksiin.	

4 Chatbotit opetuksessa ja tukipalveluissa

Chatbotit, kuten OpenAI:n ChatGPT ovat tulleet nopeasti osaksi koulutusta ja niillä voidaan tukea sekä opiskelijoiden että opettajien työtä. Ne tarjoavat nopeaa palautetta ja yksilöllistä ohjausta, mutta käyttöön liittyy myös haasteita, kuten tekniset kysymykset, opettajien koulutustarve ja eettiset riskit.

4.1 LLM chatbottien hyödyt koulutuksessa

LLM-teknologiaan perustuvat chatbotit tarjoavat merkittäviä hyötyjä opiskelijoille ja myös opettajille. Perinteisiin FAQ-tyyppisiin (usein kysytyt kysymykset) ratkaisuihin verrattuna ne pystyvät tuottamaan monipuolisempia vastauksia, mukautumaan käyttäjän tarpeisiin ja tukemaan oppimisprosessia joustavammin. Tämän vuoksi niiden käyttöä on kuvattu jopa transformatiiviseksi mahdollisuudeksi koulutuksessa. (Cheong-Trillo s.a.; Ennion & McLellan 2025).

Opiskelijoille chatbotit tuovat hyötyjä erityisesti kolmella alueella. Ensinnäkin ne tarjoavat välitöntä palautetta, joka tukee oppimista ja antaa mahdollisuuden korjata virheitä heti. Toiseksi ne mahdollistavat yksilöllisen oppimisen, sillä järjestelmä voi mukauttaa sisältöä opiskelijan tarpeiden mukaan. Kolmanneksi chatbotit parantavat tiedonsaannin nopeutta, koska opiskelija saa vastauksia välittömästi ilman, että hänen tarvitsee odottaa opettajan reaktiota. (Cheong-Trillo s.a.).

Opettajille chatbotit voivat puolestaan tuoda helpotusta työmäärään ja tukea opetuksen suunnittelua. Ne voivat esimerkiksi vastata rutiinikysymyksiin ja vapauttaa aikaa opetuksen keskeisiin tehtäviin. Lisäksi ne tarjoavat opettajille mahdollisuuden tarkastella opiskelijoiden etenemistä ja tunnistaa ajoissa ne opiskelijat, jotka tarvitsevat lisätukea. (Cheong-Trillo s.a.).

Kokonaisuutena chatbotit tukevat opiskelijoiden motivaatiota ja sitoutumista sekä auttavat opettajia hallitsemaan opetusta ja ajankäyttöä tehokkaammin. Niiden potentiaali koulutuksen tukena on kiistaton, mutta samalla on huomioitava myös haasteet, kuten teknologian luotettavuus, opiskelijoiden eettinen käyttö ja pelko tekoälypohjaisten työkalujen väärinkäytöstä. Tasapainoinen hyödyntäminen edellyttääkin sekä pedagogista harkintaa että selkeitä ohjeita ja rajoja, jotta chatbotit voivat toimia aidosti oppimista tukevina välineinä.

4.2 LLM chatbottien hyödyt erityisopiskelijoille

LLM-pohjaiset chatbotit tarjoavat erityisopiskelijoille merkittäviä mahdollisuuksia yksilöllisen oppimisen ja inklusiivisuuden vahvistamiseen. Niiden suurin etu on kyky mukautua käyttäjän tarpeisiin reaaliaikaisesti, mikä mahdollistaa oppimispolkujen räätälöinnin ja yksilöllisten tukimateriaalien tarjoamisen. Tämä on erityisen arvokasta oppilaille, joilla on oppimisvaikeuksia tai

neurokirjon haasteita, kuten autismi tai ADHD. Chatbotit voivat auttaa hahmottamaan monimutkaisia asioita, selventämään ohjeita ja antamaan rauhallista ohjausta oppilaalle. (Voultsiou & Moussiades 8.4.2025; Progmagix 2025).

Chatbotit toimivat myös sosiaalisen tuen välineenä. Neurokirjon käyttäjät ovat kuvanneet niitä ”empatiaa välittäviksi ääniksi”, jotka auttavat ymmärtämään sosiaalisia vihjeitä, säätämään viestinnän sävyä ja ehkäisemään väärinymmärryksiä esimerkiksi koulussa ja työelämässä. Tämä voi vähentää eristäytymisen riskiä ja lisätä itsevarmuutta vuorovaikutustilanteissa. Lisäksi chatboteilla on merkittävä rooli kommunikoinnin esteiden purkamisessa. Esimerkiksi tekstin yksinkertaistamisessa helpommin luettavaan muotoon tai visuaalisten ja kognitiivisten tukien tarjoamisessa. (Ritcher 26.7.2025; Voultsiou & Moussiades 8.4.2025).

Tutkimuksissa on myös korostettu LLM-chatbottien emotionaalista merkitystä. Ne voivat toimia eräänlaisina keskustelukumppaneina, jotka eivät arvostele vaan tukevat opiskelijaa positiivisesti ja johdonmukaisesti. Tämä voi lisätä sitoutumista ja vähentää ahdistusta oppimistilanteissa. (Voultsiou & Moussiades 8.4.2025).

Kaiken kaikkiaan LLM-pohjaiset chatbotit eivät ole vain tiedonvälittäjiä, vaan ne voivat toimia henkilökohtaisina oppimisen ja vuorovaikutuksen apuvälineinä.

4.3 Esteet Chatbottien käyttöön opetuksessa

Vaikka chatbotit tarjoavat merkittäviä mahdollisuuksia opetuksen tueksi, niiden hyödyntämiseen liittyy useita esteitä. Yksi keskeisistä haasteista on koettu hyödyllisyys, eli opettajat eivät näe teknologian tuovan konkreettista lisäarvoa joten sen käyttö jää vähäiseksi. Monille opettajille tekoälypohjaiset sovellukset ovat yhä vieraita ja aikaisemmat kielteiset kokemukset voivat lisätä epäluuloa. (Cheong-Trillo s.a.).

Toinen este liittyy käytön helppouteen. Chatbottien käyttö voi aluksi tuntua hankalalta, sillä onnistunut hyödyntäminen edellyttää oikeanlaista ohjeistusta ja osaamista esimerkiksi kysymysten muotoilussa. Ilman koulutusta ja harjoittelua opiskelijat ja opettajat saattavat turhautua, mikä voi vähentää halukkuutta käyttää teknologiaa. Tämä korostaa tarvetta läheisemmälle koulutukselle ja jatkuvalle ohjaukselle, jotta chatbotit eivät jäisi pinnallisen kokeilun tasolle. (Cheong-Trillo s.a.).

Lisäksi ratkaisevassa roolissa ovat teknologiat ja organisaation tukirakenteet. Opettajien on helpompi omaksua uutta teknologiaa, kun he saavat siihen koulutusta ja ylläpitotukea ja voivat luottaa riittäviin resursseihin ja tekniseen apuun. Ilman tätä tukea chatbotit voivat näyttäytyä ylimääräisenä kuormituksena. (Cheong-Trillo s.a.).

Merkittävä este liittyy myös koettuihin riskeihin. Huolta aiheuttavat esimerkiksi tietosuoja ja yksityisyyden suoja. Myös huoli opiskelijoiden tekoälypohjaisten chatbottien väärinkäytöstä, kuten plagioinnista ja pinnallisesta oppimisesta. Tutkimuksissa on todettu, että osa opiskelijoista turvautuu ChatGPT:hen ajan ja työpaineiden vuoksi, mikä voi lisätä prokrastinaatiota, heikentää muistia ja heijastua negatiivisesti oppimistuloksiin. Toisaalta chatbottien tuottaman tiedon rajallinen luotettavuus lisää epävarmuutta, väärän tiedon leviäminen voi hämmentää ja vaikeuttaa oppimisprosessia. (Cheong-Trillo s.a.; Ennion & McLellan 2025).

Kaiken kaikkiaan chatbottien käyttöönottoa rajoittavat niin teknologiset, organisaatioon liittyvät kuin psykologisetkin tekijät. Jotta niiden potentiaalia voidaan hyödyntää täysmääräisesti, tarvitaan systemaattista koulutusta, jatkuvaa tukea sekä avointa keskustelua riskeistä ja eettisistä kysymyksistä.

5 Keskeiset teknologiat ja työkalut

Sovelluksen toteutuksessa hyödynnettiin useita moderneja web-kehityksen teknologioita ja työkaluja. Ne tukevat niin käyttöliittymän rakentamista, tietojen hallintaa kuin palvelinpuolen toiminnallisuuksia. Seuraavissa alaluvuissa esitellään keskeisimmät käytetyt teknologiat sekä niiden vahvuudet ja haasteet.

5.1 TypeScript

TypeScript on Microsoftin kehittämä avoimen lähdekoodin ohjelmointikieli, joka on JavaScriptin superset. Tämä tarkoittaa, että kaikki JavaScript koodi toimii sellaisenaan myös TypeScriptissä, mutta sen päälle on lisätty ominaisuuksia, kuten staattinen tyypitys ja parempi koodin rakenteellisuus. TypeScript koodi käännetään tavalliseksi JavaScriptiksi, jolloin se toimii kaikissa ympäristöissä, jossa JavaScriptiä voi suorittaa: selaimessa, Node.js palveluissa ja mobiilisovelluksissa React Nativella. (W3Schools s.a; Świstak 17.3.2025).

Staattisen tyypityksen ansiosta TypeScript voi havaita virheitä jo käännösvaiheessa. Esimerkiksi jos funktio on määritelty ottamaan vastaan vain numeroita TypeScript antaa virheen, jos sille yritetään syöttää merkkijonoa. Vastaavaa virheentarkistusta JavaScript ei tarjoa. (W3Schools s.a.). Keskeiset TypeScriptin vahvuudet ja rajoitteet on koottu taulukkoon 2.

Taulukko 2. TypeScriptin hyödyt ja haitat (W3Schools s.a; Świstak 17.3.2025).

Hyödyt	**Haitat**
Staattinen tyypitys: Havaitsee virheet käännösvaiheessa, mikä vähentää bugien päättymistä tuotantoon.	**Lisäkäännösvaihe**: TypeScript koodi täytyy aina kääntää JavaScriptiksi, mikä voi hidastaa kehitystä.
Parempi koodin luettavuus ja ylläpidettävyys: Tyypit toimivat eräänlaisena dokumentaationa, mikä helpottaa kehittäjien välistä yhteistyötä.	**Käyrä oppimisessa**: Monimutkainen tyypitysjärjestelmä voi olla hankala etenkin aloittelijoille.
Skaalautuvuus suuriin projekteihin: Erittäin hyödyllinen keskisuurissa ja suurissa projekteissa, joissa koodin hallittavuus ja rakenteellisuus ovat keskeistä.	**Prototyyppien tekemisen hitaus**: Jos tarkoituksena on kokeilua nopeasti uusia ideoita, JavaScript voi olla parempi vaihtoehto.
Monipuolinen käyttö: Toimii laajasti eri ympäristöissä. Frontend (React, Angular), backend (Node.js) ja mobiilisovellukset (React Native).	**Kirjastojen tuen vaihtelevuus**: Kaikille JavaScript kirjastoille ei ole valmiita tyyppimäärittelyjä, mikä voi vaatia lisätyötä.
Kehittäjäkokemus: Tarjoaa älykkäitä koodiehdotuksia, automaattisia täydennyksiä ja virheilmoituksia suoraan kehitysympäristöön.	**Mahdollinen liiallinen turvallisuuden tunne**: TypeScript varmistaa vain käännösaikaiset tyypit.

Hyödyt	Haitat
Laaja suosio ja tuki: Käytössä suurilla toimijoilla (Microsoft, Slack, Asana) ja erittäin suosittu kehittäjäyhteisössä.	

5.2 Express

Express.js (lyhyesti Express) on Node.js:ään perustuva web-sovelluskehys, jota käytetään laajasti verkkosovellusten ja rajapintojen (API) rakentamisessa. Expressiä kutsutaan usein Node.js:n ”de facto” standardikehykseksi sen joustavuuden, laajennettavuuden ja laajan ekosysteemin ansiosta. Se on kevyt, minimaalinen ja ”unopinionated” tyyppinen, eli se ei määrää tiukkaa sovellusarkkitehtuuria, vaan jättää rakenteelliset ratkaisut kehittäjälle. Express tarjoaa monipuolisen reititysjärjestelmän, tukee http-apufunktioita, kuten uudelleenohjausta ja välimuistia, sekä mahdollistaa middleware-funktioiden käytön http-pyyntöihin vastaamisessa. Lisäksi Express toimii saumattomasti yhdessä templaatiomoottoreiden ja muiden Node.js kirjastojen kanssa. (W3Schools s.a. b).

Expressin keskeisiä hyötyjä ovat sen yksinkertaisuus ja laaja käyttö. Express on yksi maailman suosituimmista Node.js kehyksistä, jota käyttää yli miljoona aktiivista verkkosivustoa ja joka on keskeinen osa MEAN- ja MERN kehityspinoja. Sen vahvuuksiin kuuluvat helppokäyttöisyys, nopea oppimiskynnys JavaScriptiä tunteville sekä yksinkertainen koodirakenne, joka vähentää toistuvien Node.js perustoimintojen kirjoittamista. Expressin avulla voidaan toteuttaa RESTful API-rajapintoja, reaaliaikaisia sovelluksia sekä yksisivuisia sovelluksia yhdessä frontend kehysten, kuten Reactin tai Angularin kanssa. Kehystä tukee suuri ja aktiivinen kehittäjäyhteisö, joka tarjoaa runsaasti dokumentaatiota, kirjastoja ja valmiita middleware-moduuleja. Lisäksi Express integroituu hyvin SQL- ja NoSQL tietokantojen, kuten MongoDB:n kanssa. (Emadamerho-Atori 15.11.2025).

Expressin suurimmat haasteet liittyvät sen keveyteen ja joustavuuteen, sillä Express tarjoaa vain perustason toiminnallisuudet. Monet ominaisuudet, kuten autentikointi ja lomakevalidaatio täytyy toteuttaa kolmansien osapuolten kirjastoilla. Tämä voi johtaa monimutkaisiin riippuvuuksiin ja lisätä sovelluksen hallinnointitaakkaa. Expressin ”unopinionated” luonne on kaksiteräinen miekka: se antaa vapauksia, mutta suurissa projekteissa voi syntyä epäjohdonmukaisia rakenteita ja vaikeuksia tiimityössä, ellei kehittäjätiimi luo selkeitä kehitysstandardeja. Lisäksi middleware-funktioiden hallinta voi muuttua monimutkaiseksi laajoissa sovelluksissa. Middleware suoritetaan ennalta määrittelyssä järjestyksessä ja väärä järjestys voi rikkoa sovelluksen toiminnallisuuksia tai tehdä riippuvuussuhteista epäselviä. (Emadamerho-Atori 15.11.2025).

5.3 OpenAI ChatGPT rajapinta

OpenAI on yksityinen tutkimuslaboratorio, jonka tavoitteena on kehittää tekoälyä turvallisesti ja koko ihmiskunnan hyödyksi. Yhtiön perustivat vuonna 2015 muun muassa Sam Altman ja Elon Musk ja sen pääkonttori sijaitsee San Franciscossa. OpenAI on saanut merkittäviä investointeja, erityisesti Microsoftilta. OpenAI:n tunnetuimpia tuotteita ovat GPT kielimallit, DALL-E kuvageneraattori sekä ChatGPT. (Hashemi-Pour 2025; Belcic & Stryker 2025).

ChatGPT puolestaan on generatiiviseen tekoälyyn perustuva chatbot, joka hyödyntää OpenAI:n kehittämiä GPT-malleja. Se kykenee käymään ihmismäisiä keskusteluja luonnollisen kielen prosessoinnin avulla ja tuottamaan tekstiä erilaisiin käyttötarkoituksiin sekä viimeisempien malliversioiden myötä käsittelemään kuvia ja ääntä tekstin lisäksi. (Belcic & Stryker 2025).

OpenAI tarjoaa kehittäjille ohjelmointirajapinnan (API), jonka avulla GPT-malleja voidaan hyödyntää erilaisissa sovelluksissa. API toimii yleiskäyttöisenä ”teksti sisään, teksti ulos” käyttöliittymänä, eli se vastaanottaa käyttäjän antaman tekstisyötteen ja palauttaa siihen liittyvän vastauksen. Rajapinnan etuna on sen joustavuus: käyttäjät voivat mukauttaa tekoälyä tiettyihin käyttötarkoituksiin. Esimerkiksi omien esimerkkien avulla tai opettamalla sitä omalla aineistollaan. Näin voidaan kehittää esimerkiksi asiakaspalvelun, opetuksen tai erityisryhmien tarpeisiin sopivia sovelluksia. (Brockman, Murati, Welinder & OpenAI 2020).

Suurin hyöty OpenAI:n ChatGPT-rajapinnan käytössä on se, että kehittäjien ei tarvitse rakentaa tekoälyä itse. Kielimallien kehittäminen ja kouluttaminen vaatii valtavia tietomääriä, laskentatehoa ja asiantuntijuutta mikä ei ole yksittäisten kehittäjien tai oppilaitosten käytettävissä. Rajapinnan avulla on mahdollista hyödyntää valmiiksi koulutettuja malleja ja keskittyä itse sovelluksen toiminnallisuuksiin ja käyttäjäkokemukseen. Tämä nopeuttaa kehitysprosessia ja tekee tekoälyratkaisujen käyttöönotosta saavutettavaa. (Brockman, Murati, Welinder & OpenAI 2020; Belcic & Stryker 2025).

Vaikka ChatGPT rajapinta tarjoaa monia hyötyjä, siihen liittyy myös riskejä. Suurimpia huolenaiheita ovat tietoturva ja yksityisyydensuoja. Käyttäjien tuottama sisältö voi päätyä mallien kehittämiseen, ellei sitä erikseen rajata. OpenAI on kuitenkin kehittänyt käytäntöjä, joilla käyttäjät voivat hallita tietojensa käyttöä. Esimerkiksi API-käyttäjien dataa ei oletuksena käytetä mallien kouluttamiseen ja ChatGPT Enterprise ja Team-versioissa tietosuojaan kiinnitetään erityistä huomiota. Lisäksi generatiiviseen tekoälyyn liittyy riski virheellisen tai harhaanjohtavan tiedon tuottamisessa, mikä voi olla erityisen haastavaa opetuskontekstissa, jossa luotettavuus ja läpinäkyvyys ovat keskeisiä vaatimuksia. (OpenAI 2024).

5.4 MongoDB tietokantana

MongoDB on avoimen lähdekoodin, dokumenttipohjainen NoSQL-tietokantajärjestelmä, joka tarjoaa vaihtoehdon perinteiselle relaatiotietokannalle. Sen sijaan, että tiedot tallennettaisiin taulukkoihin ja riveihin MongoDB käyttää joustavia dokumentteja, jotka voidaan muotoilla Binaari JSON (BSON) muodossa. Tämä mahdollistaa erilaisten, rakenteeltaan vaihtelevien tietotyyppien tallentamisen ja käsittelyn tehokkaasti. Dokumenttipohjainen malli tarjoaa kehittäjille joustavuutta ja MongoDB:n dynaaminen skeema mahdollistaa tietueiden lisäämisen ja muokkaamisen ilman, että koko tietokannan rakennetta tarvitsee muuttaa. (IBM 2025).

MongoDB tukee myös vektorihakuja, jotka mahdollistavat semanttisen haun dokumenttikokoelmissa. Atlas Vector Search ominaisuuden avulla kehittäjät voivat ladata embedattuja vektoreita sisältäviä dokumentteja, luoda hakuindeksejä ja suorittaa semanttista hakua, mikä on erityisen hyödyllistä esimerkiksi tekoälypohjaisten chatbottien ja älykkäiden hakujärjestelmien toteutuksessa. (MongoDB s.a. a; MongoDB s.a. b).

MongoDB:n käyttöön liittyy useita etuja. Sen joustava skeemamalli mahdollistaa nopean sovelluskehityksen ja muutosten tekemisen lennossa ilman suuria tietokantamuutoksia. Tietokanta on skaalautuva ja horisontaalisen shardingin avulla suuria tietomääriä voidaan käsitellä useiden palvelinten välillä ilman suorituskyvyn heikkenemistä. MongoDB on optimoitu luku- ja kirjoitusintensiivisiin sovelluksiin, tukee monimutkaisia tietorakenteita ja reaaliaikaista tietojenkäsittelyä. Lisäksi se on yhteensopiva useiden ohjelmointikielten ja alustojen kanssa mikä tekee siitä joustavan valinnan nykyaikaisiin web- ja mobiilisovelluksiin. (GeeksforGeeks 2025; IBM 2025).

Toisaalta MongoDB:llä on myös rajoituksia. Dokumenttitasolla se tukee ACID-ominaisuuksia, mutta se ei tarjoa täyttä transaktiotukea useiden dokumenttien yli. Tämä voi olla haaste sovelluksissa, jotka vaativat tiukkaa tietokonsistenssia. MongoDB:n suunnittelu voi johtaa suurempaan muistinkulutukseen ja perinteisiä join-toimintoja ei tueta natiivisti. Dokumenttikoko on rajoitettu 16 megatavuun ja shardingin hallinta voi olla monimutkaista. Lisäksi tietokannan lisensointi SSPL:n alaisena saattaa rajoittaa joidenkin organisaatioiden käyttöä. (GeeksforGeeks 2025; MongoDB s.a. c).

Kaiken kaikkiaan MongoDB:n joustavuus, suorituskyky ja skaalautuvuus tekevät siitä suositun ratkaisun. Erityisesti suurten, rakenteeltaan vaihtelevien tietomäärien hallintaan ja se tarjoaa tehokkaita työkaluja nykyaikaisiin sovelluksiin, joissa perinteiset relaatiotietokannat eivät riitä.

6 Chatbot-järjestelmän suunnittelu ja toteutus

Tässä luvussa kuvataan opinnäytetyön käytännön toteutusvaiheet ideoinnista valmiiseen ratkaisuun. Luku esittelee, miten RAG-pohjainen tekoäly chatbot suunniteltiin ja rakennettiin tukemaan erityisopiskelijoiden oppimista ja koulutuksellista saavutettavuutta. Painopiste on teknisen toteutuksen ja pedagogisten tavoitteiden yhdistämisessä toimivaksi kokonaisuudeksi, joka vastaa erilaisiin käyttäjäryhmiin kohdistuviin tarpeisiin.

Luvussa käydään läpi projektin lähtökohdat ja suunnitteluratkaisut, järjestelmän arkkitehtuuri sekä keskeiset tekniset ja toiminnalliset ratkaisut. Lisäksi esitellään chatbotin toiminnallisuuksia ja sen käytön demotilanteita, joiden avulla tuotoksen onnistumista arvioidaan suhteessa työn alkuperäisiin tavoitteisiin.

6.1 Tuotoksen lähtötilanne ja suunnittelu

Toimeksiantona on suunnitella ja toteuttaa RAG pohjainen tekoäly-chatbot, jonka ensisijainen tarkoitus on tukea eri käyttäjäryhmiä koulutusympäristössä. Opiskelijat voivat hyödyntää chatbottia esimerkiksi tiedonhaussa, ohjeiden ymmärtämisessä sekä opintojen tukena. Hallintakäyttäjille hallintakäyttöliittymä tarjoaa puolestaan mahdollisuuden hallita sisältöä ja kehittää palvelua vastaamaan paremmin käyttäjien tarpeita.

Projektissa tunnistetaan kolme keskeistä kohderyhmää:

- **Erityisopiskelijat**, joille chatbot tarjoaa yksilöllisempää ja saavutettavampaa tukea opintoihin liittyvissä kysymyksissä.
- **Opiskelijat**, jotka voivat hyödyntää chatbottia yleisenä apuvälineenä esimerkiksi tiedonhaussa ja käytännön ohjeiden saamisessa.
- **Hallintakäyttäjät**, joilla on mahdollisuus valvoa, muokata ja ylläpitää chatbotin sisältöä ja teknistä toimintaa.

Opinnäytetyön toteutusta rajaa käytettävissä oleva aika ja resurssit. Näiden vuoksi kaikkia mahdollisia osa-alueita ei ole mahdollista toteuttaa. Esimerkiksi DevOps-käytännöt, kuten kattava jatkuvan integroinnin ja jatkuvan käyttöönotto (CI/CD) putki, voidaan määritellä suunnitelman tasolla, mutta niiden täydellinen käyttöönotto jää työn ulkopuolelle.

Samoin testauksen kehittäminen jää rajoitteeksi. Järjestelmälle ei rakenneta kattavaa automaattista testausympäristöä, vaan testaus painottuu yksittäisten toiminnallisuuksien manuaaliseen kokeiluun ja perusvarmistukseen.

Näiden rajallisuuksien vuoksi painopiste keskittyy palvelinpuolen toiminnallisuuteen ja RAG-arkkitehtuurin hyödyntämiseen chatbotin toteutuksessa. Laajempien DevOps-ratkaisujen, testauksen ja saavutettavuuden kehittäminen voidaan pitää jatkokehityskohteina, mikäli järjestelmää viedään eteenpäin opinnäytetyön valmistumisen jälkeen.

Sovelluksen suunnittelu- ja kehitysvaiheessa hyödynnettiin ChatGPT-tekoälysovellusta ohjelmointiprosessin tukena. Tekoälyä käytettiin erityisesti koodiin virheiden korjaamiseen, rakenteellisten ratkaisujen ideointiin sekä virheilmoitusten analysointiin. Tekoälyn avulla haettiin vaihtoehtoisia toteutustapoja ja selkeytettiin teknisiä ongelmia, mutta kaikki ratkaisut ja lopullinen toteutus ovat itsenäisesti suunniteltuja ja toteuttamia.

Suunnitteluvaiheessa hyödynsin käyttäjätarinoita keskeisenä työkaluna järjestelmän toiminnallisuuksien määrittelyssä. Käyttäjätarinoiden avulla tarkastelin järjestelmää eri näkökulmista: Opiskelijan, kehittäjän ja projektin omistajan. Rakensin kokonaisuuden, joka vastaa kunkin ryhmän tunnistettuihin tarpeisiin.

Käyttäjätarinoista johdetut tehtävät priorisoidaan projektin alkuvaiheessa, jotta kehitys etenee selkeässä ja loogisessa järjestyksessä. Kehitystyö alkaa järjestelmän perustasta eli tietokannan ja sen rakenteiden määrittelyssä, minkä jälkeen keskitytään tiedonhallinnan ja hakuominaisuuksien toteutukseen. Kun perustoiminnallisuus on kunnossa, rakennetaan ensimmäinen toimiva versio chatbotista ja käyttöliittymä helpottamaan sen saaman tiedon hallintaa. Lopuksi kehitystä viedään eteenpäin edistyneempään vaiheeseen, jossa RAG-arkkitehtuuria ja vastausten laatua parannetaan. Näin projektin eteneminen pysyy hallittuna ja jokainen vaihe tukee seuraavan onnistumista.

Opiskelijoiden tarinat auttoivat erityisesti chatbotin viestintälogiikan ja saavutettavuuden suunnittelussa. Esimerkiksi tarinat, joissa korostetaan selkokielisyyttä ja virheensietoa ohjasivat mallin parametrien ja viestien esikäsittelyn suunnittelua siten, että chatbot pystyy ymmärtämään epätäydellisiä kysymyksiä ja tuottamaan helposti hahmotettavia vastauksia.

Käyttäjätarinat, joissa painotettiin keskustelun jatkuvuutta ja yksityisyyden suojaa vaikuttivat suoraan teknisiin ratkaisuihin. Niiden pohjalta päätin toteuttaa keskusteluhistorian tallennuksen istuntokohtaisesti ja rajata tietojen säilytystä kirjautuneisiin käyttäjiin. Tämä lähestymistapa tasapainotti käytettävyyden ja tietosuojan vaatimuksia. Kehittäjän ja ylläpidon näkökulmasta käyttäjätarinat puolestaan ohjasivat palvelinarkkitehtuurin ja koodin rakenteellisia valintoja. Niiden avulla määrittelin esimerkiksi sen, että järjestelmä toteutetaan TypeScriptillä, jotta se olisi tyypitetty, virheenkestävä ja helposti laajennettava.

Lopputuloksen onnistumista arvioidaan laadullisten kriteerien perusteella. Hyvä chatbot on:

- **Saavutettava ja käyttäjäystävällinen**, eli sen käyttöliittymä ja kieli soveltuvat erityisesti erityisopiskelijoille.
- **Luotettava ja toiminnallisesti vakaa**, eli se vastaa kysymyksiin johdonmukaisesti ja sen toiminta ei keskeydy teknisten virheiden vuoksi.
- **Sisällöllisesti relevantti**, eli se osaa hyödyntää RAG-arkkitehtuurin avulla oikeaa lähdetietoa ja antaa käyttäjälle asiayhteyteen sopivia vastauksia.
- **Hallittava ja muokattava**, eli hallintakäyttäjillä on selkeät työkalut sisällön päivittämiseen.
- **Kehityskelpoinen**, eli toteutus on laajennettavissa ja sitä voidaan jatkossa hyödyntää muiden käyttäjäryhmien tai sovellusalueiden tarpeisiin.

Näiden kriteerien avulla voidaan arvioida, vastaako lopullinen tuotos asetettuihin tavoitteisiin ja palveleeko se sekä opiskelijoiden että hallintakäyttäjien tarpeita.

6.2 Tuotoksen tuottaminen

Projektin toteutuksessa hyödynnettiin GitHubiin luotua monorepo-rakennetta, jonka avulla sekä palvelin- että käyttöliittymäpuoli pysyisivät samassa hallinnassa. Tämä teki kokonaisuudesta selkeämmän ja helpotti myös versionhallintaa.

Palvelinpuoli rakennettiin Node.js:llä ja Expressillä TypeScriptin avulla. Ratkaisu tuntui luontevalta, sillä se tarjosi selkeän pohjan toteutukselle ja oli myös omalta osaltani tuttu ympäristö. Tietokantana käytettiin MongoDB Atlasta ja sen hallintaan Mongoosea. Chatbotin älykkyys toteutettiin OpenAI:n rajapinnan ja AI SDK:n avulla ja käyttäjien tunnistautuminen hoidettiin JWT-ratkaisulla.

Käyttöliittymä toteutettiin Reactin ja Viten avulla, myös TypeScriptilla. Tyylien hallintaan otettiin käyttöön Tailwind CSS tyylitys, joka mahdollisti käyttöliittymän nopean ratkaisun mallintamisen. Sovelluksen tilanhallintaan otettiin käyttöön Redux toolkit. Kokonaisuudesta muodostui joustava ja käytännöllinen työkalupaketti, joka mahdollisti chatbotin rakentamisen vaiheittain ja antoi samalla hyvän pohjan jatkokehitykselle.

Opiskelijoiden tarve saada selkeitä ja ajantasaisia vastauksia ohjasi tiedon esikäsittelyvaihetta. Tietosisällöt jaettiin pienempiin, merkityksellisiin osiin, joista muodostettiin embedding-vektoreita ja tallennettiin tietokantaan. Samalla suunnittelussa huomioitiin tiedon ylläpidon näkökulma, eli sisällön päivittäminen ja uusien chunkkien lisääminen tuli olla mahdollisimman yksinkertaista.

Chunkkien luonnissa ja niiden hyödyntämisessä ilmeni kuitenkin haasteita. Tekoälyn tuottamien vastausten laatu on vahvasti riippuvainen siitä, miten johdonmukaisesti ja tarkasti tiedot on jaettu

osiin. Mikäli chunkit muodostetaan liian vapaamuotoisesti tai omin sanoin, voi niiden merkityssisältö vääristyä ja tekoälyn tuottamat vastaukset menettävät täsmällisyyttään. Tämä korosti tarvetta yhtenäiselle ja systemaattiselle tavalle jäsentää sisältö.

Ratkaisu ongelmaan kehitettiin palvelimeen kaksi vaihtoehtoista tapaa luoda chunkkeja. Käyttäjä voi halutessaan muodostaa chunkit manuaalisesti tai antaa kokonaisen dokumentin tekoälyä hyödyntävän työkalun käsiteltäväksi. Jälkimmäinen vaihtoehto tuottaa dokumenteista yhtenäisiä ja rakenteeltaan samankaltaisia chunkkeja, mikä parantaa tiedon johdonmukaisuutta ja helpottaa sen hyödyntämistä vektorihakua varten. Tämän toiminnallisuuden toteutus on esitetty kuvassa 6 olevassa koodissa. Ratkaisun taustalla hyödynsin Towards Data Science -sivustolla julkaistua artikkelia, jonka avulla tein tekoälylle systeemi kuvauksen. (Murallie 2024).

```
export const createChunks = async (text: string) => {
    const maxChunkSize = 350; // Maximum characters per chunk

    // Use AI to intelligently chunk the text
    const result = await generateObject({
        model: openai("gpt-4o"),
        system: `Ensin muunna teksti selkeiksi ja itsenäisiksi propositioiksi:
                 jokaisen lauseen tulee olla itsenäinen ja ymmärrettävä ilman viittauksia pronomineihin (kuten hän, se, ne), vaan käytä aina tarkkaa viittausta.
                 Tämän jälkeen jaa teksti loogisiin ja semanttisesti yhtenäisiin osiin, joissa kukin osa on korkeintaan ${maxChunkSize} merkkiä pitkä.
                 Älä katkaise virkkeitä tai kappaleita kesken. Sijoita samaan osaan vain sellaiset lauseet, jotka käsittelevät samaa teemaa tai aihepiiriä.
                 Jos aihe muuttuu, aloita uusi osa. Pidä huolta, että jokainen osa muodostaa itsenäisen ja ymmärrettävän kokonaisuuden ilman viittauksia muihin osiin.
                 Varmista, että tärkeää tietoa ei katoa chunkkien rajakohdissa.`,
        schema: chunkCreationSchemaJson,
        prompt: text,
    });

    // Ensure chunk creation was successful
    assert(result && result.object && result.object.chunks, 500, "Failed to create chunks");

    return result.object;
}
```

Kuva 6. Tekoälymalli chunkkien luontiin

Keskustelun sujuvuus ja vastausten nopeus olivat keskeisiä tavoitteita chatbotin käyttöliittymän kehityksessä. Ensimmäisessä tekemässäni versiossa tekoälyn vastaus välitettiin käyttäjälle vasta sen muodostuttua kokonaisuudessaan, mikä teki keskustelusta hitaampaa ja vähemmän luonnollista. Käyttäjän näkökulmasta viive vastauksen saamisessa heikensi vuorovaikutuksen kokemusta ja katkaisi keskustelun rytmin.

Tämän ongelman ratkaisemiseksi otettiin käyttöön AI SDK:n tarjoamat useChat- ja StreamText-toiminnot. Näiden avulla chatbot pystyy lähettämään vastauksensa reaaliaikaisesti, sitä mukaa kun tekoäly tuottaa tekstiä. Palvelinpuolelle luotiin oma ”streampipeline”, joka välittää datan vaiheittain käyttöliittymään, jossa useChat ottaa vastausta puolestaan vastaan. Tämä toteutus teki keskustelusta huomattavasti dynaamisemman ja luonnollisemman, sillä käyttäjä näkee vastauksen rakentuvan vähitellen ruudulla. Tämän toiminnallisuuden toteutus on esitetty kuvassa 7. (AI SDK s.a. b; AI SDK s.a. c; AI SDK s.a. d).

```
// Stream the response using AI SDK
pipeUIMessageStreamToResponse({
  response: res,
  stream: createUIMessageStream<MessageStream>({
    execute: async ({ writer }) => {
      // Global variable that can be modified
      let finalChatLogId = chatLogId;

      // If user is authenticated, create or update chat log
      if (req.user) {
        finalChatLogId = await createOrUpdateChatLog(
          req.user ? req.user.id : "",
          finalChatLogId,
          formattedMessages,
          formattedLastMessage
        );
      }

      // Send chat log ID to client
      writer.write({
        type: "data-chatLogId",
        data: { chatLogId: finalChatLogId },
      });

      // Stream response in all cases
      const stream = await getStreamText(messages);
      writer.merge(stream.toUIMessageStream({ sendStart: false }));

      // Update chat log with full assistant response for authenticated users
      if (req.user && finalChatLogId) {
        let fullText = "";
        for await (const textPart of stream.textStream) {
          fullText += textPart;
        }
        await updateLogWithAssistantMessage(finalChatLogId, fullText);
      }
    },
  }),
});
```

Kuva 7. Tekoälyn vastauksen reaaliaikainen lähettäminen

Keskustelujen tallentaminen oli tärkeä osa chatbotin kehitystyötä, sillä kirjautuneille käyttäjille haluttiin mahdollisuus palata aiempiin keskusteluihin. Tämä parantaa käyttökokemusta erityisesti oppimisympäristöissä, jossa opiskelija voi hyödyntää aikaisempia vastauksia myöhemmissä tehtävissä tai opintojen tukena.

Toteutuksessa kuitenkin ilmeni teknisiä haasteita, jotka liittyivät StreamChat-ominaisuuden ja käyttöliittymän useChat-funktioiden yhteiskäyttöön. Vaikka useChat helpottaa striimatun keskustelun hallintaa, se ei oletusarvoisesti tue keskustelulokien muotoilua tietokantaan tallennettavaksi. Tämä aiheutti ongelmia keskustelulokin rakenteessa, kun tekoälyn vastaukset välittyivät vaiheittain eikä niitä voitu suoraan tallentaa yhteisenä kokonaisuutena.

Ongelma ratkaistiin luomalla palvelinpuolelle oma prosessi, joka hallitsee keskustelun muotoilun ja tallennuksen vaiheittain. Ensin käyttäjän viesti muokataan tietokantaan sopivaan muotoon, jonka jälkeen se välitetään tekoälylle käsiteltäväksi. Tämän jälkeen tarkistetaan, onko käyttäjä kirjautunut ja kuuluuko pyyntöön olemassa oleva keskustelulokin id. Mikäli käyttäjä on tunnistettu, joko päivitetään olemassa oleva keskusteluloki tai luodaan uusi.

Kun tekoälyn vastausta lähetetään käyttöliittymään, samalla koko vastaus kerätään palvelimella talteen ja sen valmistuttua se tallennetaan keskustelulokiin. Näin varmistetaan, että käyttäjä näkee viestin heti, mutta myös koko keskusteluhistoria tallentuu myöhempää tarkastelua varten.

Kuten aikaisemmasta koodiesimerkistä käy ilmi (Kuva 7), ratkaisu hyödyntää palvelinpuolen funktioita createOrUpdateChatLog ja updateLogWithAssistantMessage, jotka vastaavat keskustelulokin luomisesta ja tekoälyn tuottamien viestien päivittämisestä tietokantaan.

Kehitystyön loppuvaiheessa chatbottiin otettiin käyttöön advanced RAG arkkitehtuuri ratkaisu, jonka tarkoituksena oli parantaa hakutulosten osuvuutta ja tekoälyn kykyä ymmärtää käyttäjän kysymyksiä. Käyttäjätarinoiden pohjalta oli havaittavissa, että opiskelijoiden esittämät kysymykset voivat olla epätäsmällisiä, kirjoitusvirheitä sisältäviä tai muotoiltu eritavoilla kuin tietokannan tiedot. Tämä johti tilanteisiin, joissa tekoäly ei löytänyt tietoa vaikka se oli tietokannassa.

Ratkaisu tähän oli, että chatbot osaa ensin arvioida tarvitseeko käyttäjän kysymykseen hakea vastausta omasta tietokannasta vai voidaanko siihen vastata suoraan mallin omalla tietämyksellä. Kun tietoa on haettava, tekoäly hyödyntää tähän tarkoitukseen kehitettyä tool-toimintoa expandAndSearch, joka suorittaa monivaiheisen tiedonhaun.

Ensimmäisessä vaiheessa tool muokkaa käyttäjän alkuperäisestä kysymyksestä kolme erilaista versiota toista tekoälymallia hyödyntäen. Tämä parantaa hakujen tarkkuutta, sillä jokainen muunnelma voi tavoittaa tietokannan eri muotoiluilla tallennettua tietoa. Näiden optimoitujen kysymysten avulla suoritetaan vektorihaku tietokannasta, jolloin tekoäly löytää useampia mahdollisia vastaslähteitä ja voi koosta vastauksensa kattavammin.

Jotta tulokset pysyisivät laadukkaina ja kustannustehokkaina, hakutuloksista poistetaan päällekkäisyydet ja toistot ennen lopullisen vastauksen muodostamista. Tämä vähentää turhaa token-kulutusta ja varmistaa, että käyttäjälle näytettävä vastaus on tiivis, selkeä ja johdonmukainen.

Tämän ansiosta järjestelmä on aiempaa älykkäämpi ja joustavampi. Se tukee erilaisia käyttäjäprofiileja ja kielellisiä taitotasoja sekä vähentää virheellisten vastausten määrää. Käyttäjän

näkökulmasta kokemus on sujuvampi, sillä chatbot ymmärtää kysymykset paremmin ja tarjoaa niihin täsmällisiä, luotettavia ja helposti ymmärrettäviä vastauksia (ks. kuva 8).

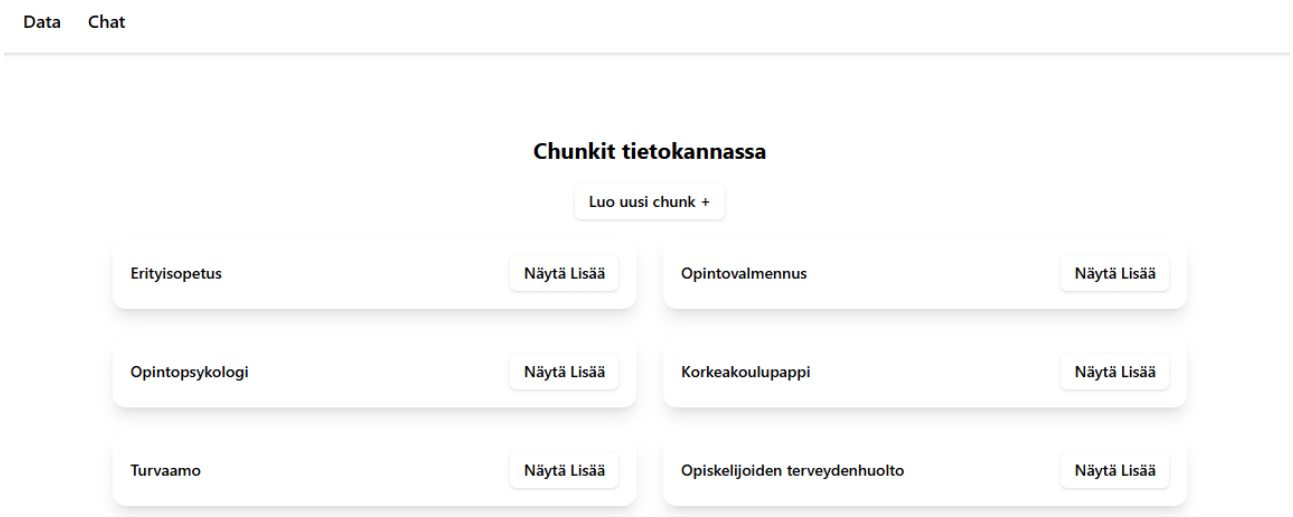
```
export const getStreamText = async (messages: UIMessage[]) => {
  try {
    // Generate streaming response using AI model and custom tools
    const result = streamText({
      model: openai("gpt-4o"),                    // OpenAI's most capable model for streaming
      messages: convertToModelMessages(messages), // Convert UIMessage format to model format
      onFinish: () => {
        // Callback executed when streaming completes
        console.log("Streaming finished.");
      },
      system: chatBotSystemPrompt,                // Custom system prompt for educational support
      tools: {
        // Tool that expands questions into multiple variations and searches with all of them
        expandAndSearch: tool<{ query: string }, SearchHits[]>({
          name: "expandAndSearch",
          description:
            "Laajenna kysymys useiksi muunnelmiksi ja hae tietoa kaikilla niillä.",
          inputSchema: toolInputSchemaZod,        // Zod schema for input validation
          execute: async ({ query }) => {
            // Generate multiple question variations from the original question
            // This improves search accuracy by covering different phrasings and aspects
            const {
              object: { queries },
            } = await generateObject({
              model: openai("gpt-4o"),            // Use GPT-4o for query expansion
              schema: toolOptimazationSchemaZod,   // Schema for query optimization
              prompt: `Luo seuraavasta kysymyksestä 3:\n\n${query}`,
            });

            // Search knowledge base with all generated query variations
            // This ensures comprehensive information retrieval
            const results = await findInformation(queries);
            console.log("expandAndSearch results:", results);
            return results;
          },
        }),
      },
      // Limit maximum tool calls to prevent excessive API usage and costs
      stopWhen: stepCountIs(3),
    });
    return result;
  } catch (error) {
    console.error("Error generating chat completion:", error);
    assert(false, 500, "Error generating chat completion");
  }
};
```

Kuva 8. Advanced RAG

6.3 Valmis tuotos

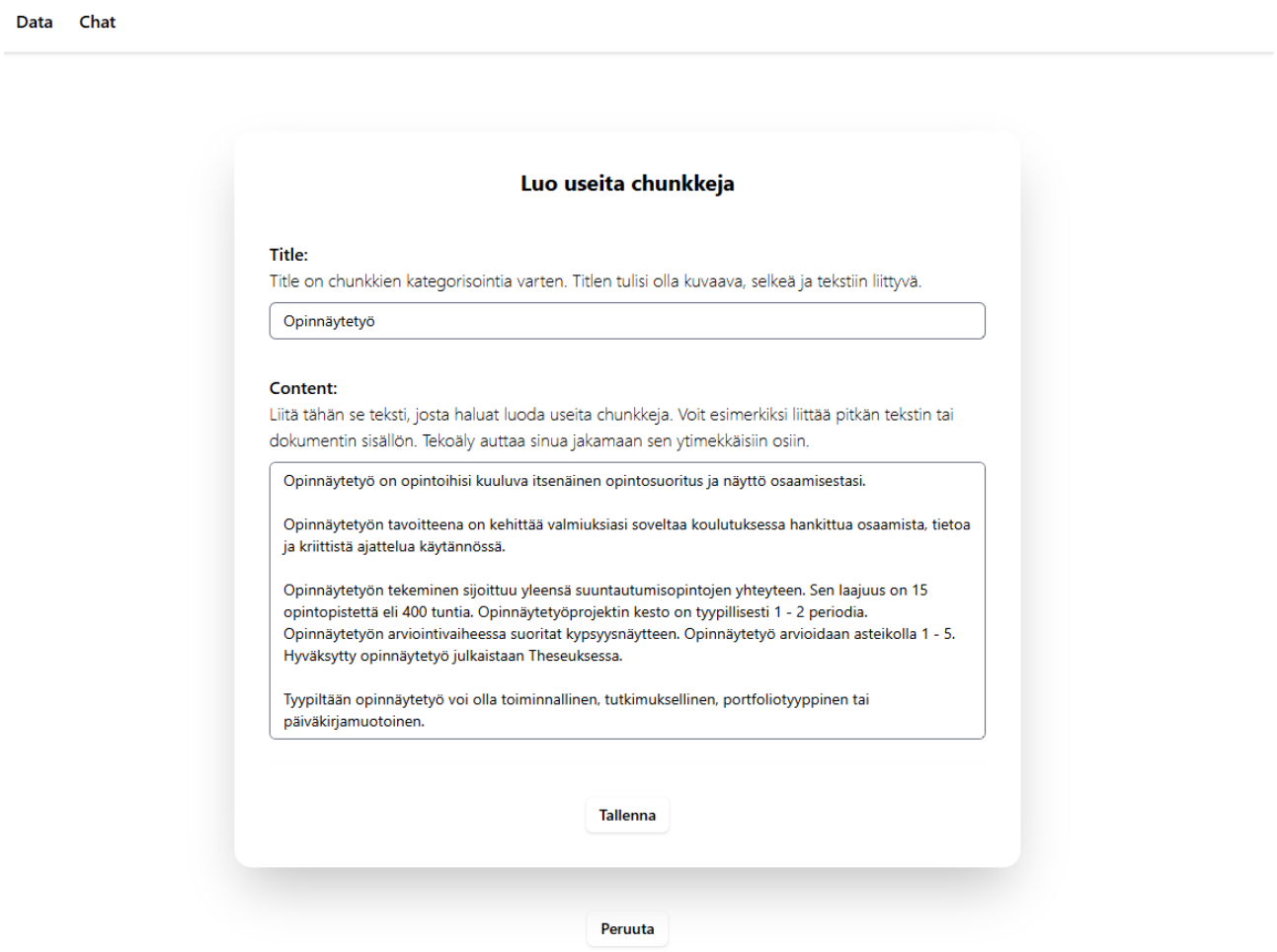
Valmiin palvelimen toiminnallisuuden demonstrointi tapahtuu projektin monoreposta löytyvällä data-manager sovelluksella. Käyttöliittymä tarjoaa järjestelmän ylläpitäjille mahdollisuuden testata chatbotin toimintaa, esittää kysymyksiä ja seurata vastauksia reaaliaikaisesti. Lisäksi käyttöliittymässä voidaan hallita dokumenttien chunkkeja, lisätä tietosisältöä ja tarkastella olemassa olevia, mikä helpottaa tietokannan ylläpitoa.

Käyttöliittymän etusivulla näkyvät olemassa olevien chunk-kategorioiden lista, jotka muodostuvat chunkkien otsikoiden perusteella. Käyttäjä voi valita kategorian ja tarkastella siihen liittyviä chunkkeja (ks. kuva 9).



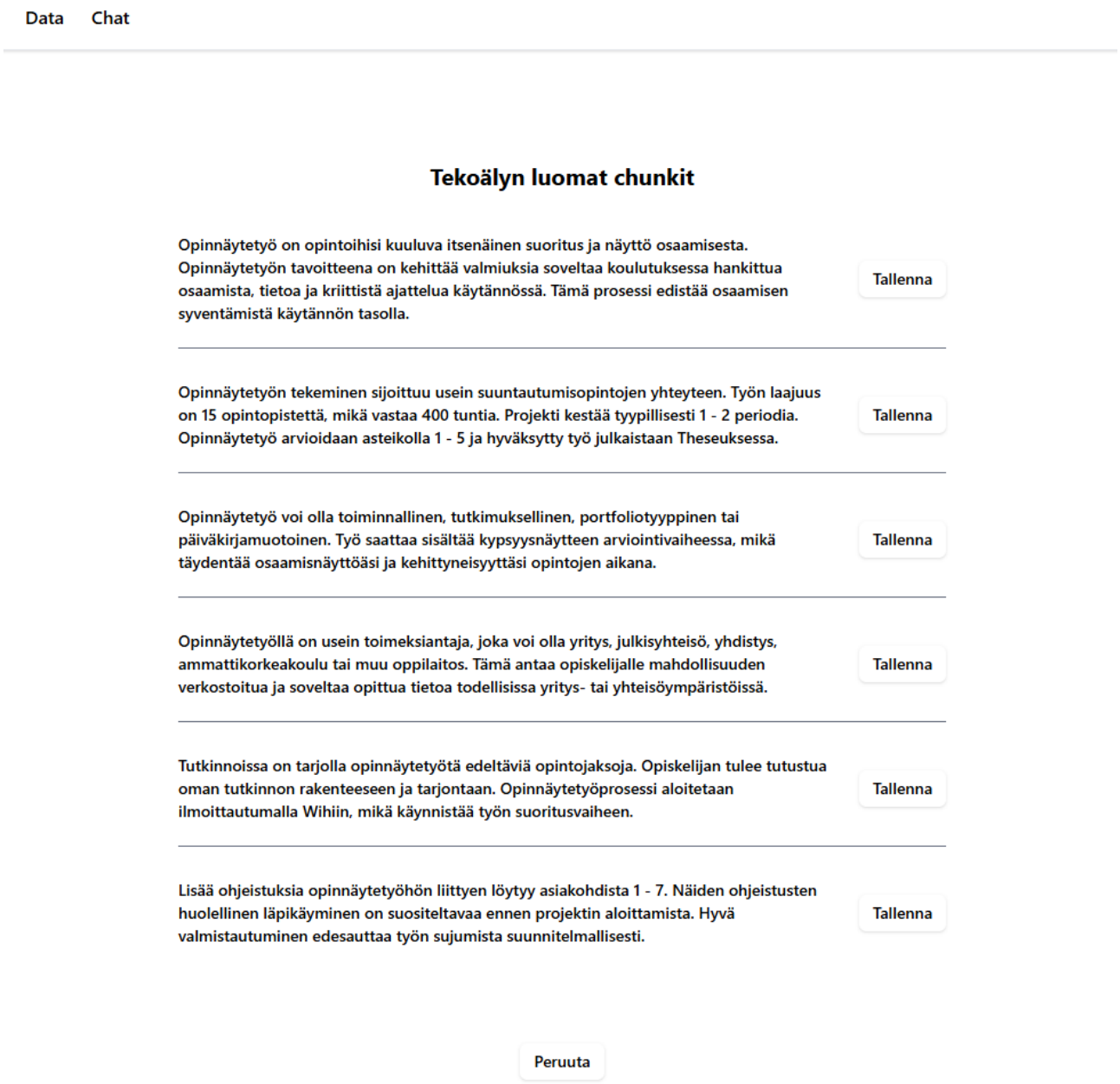
Kuva 9. Data-managerin etusivu

Uusia chunkkeja voidaan luoda tekoälypohjaisella työkalulla. Käyttäjä syöttää lomakkeeseen esimerkiksi tekstin Haaga-Helian opinnäytetyöohjeista ja antaa otsikoksi kuvaavan termin (ks. kuva 10).



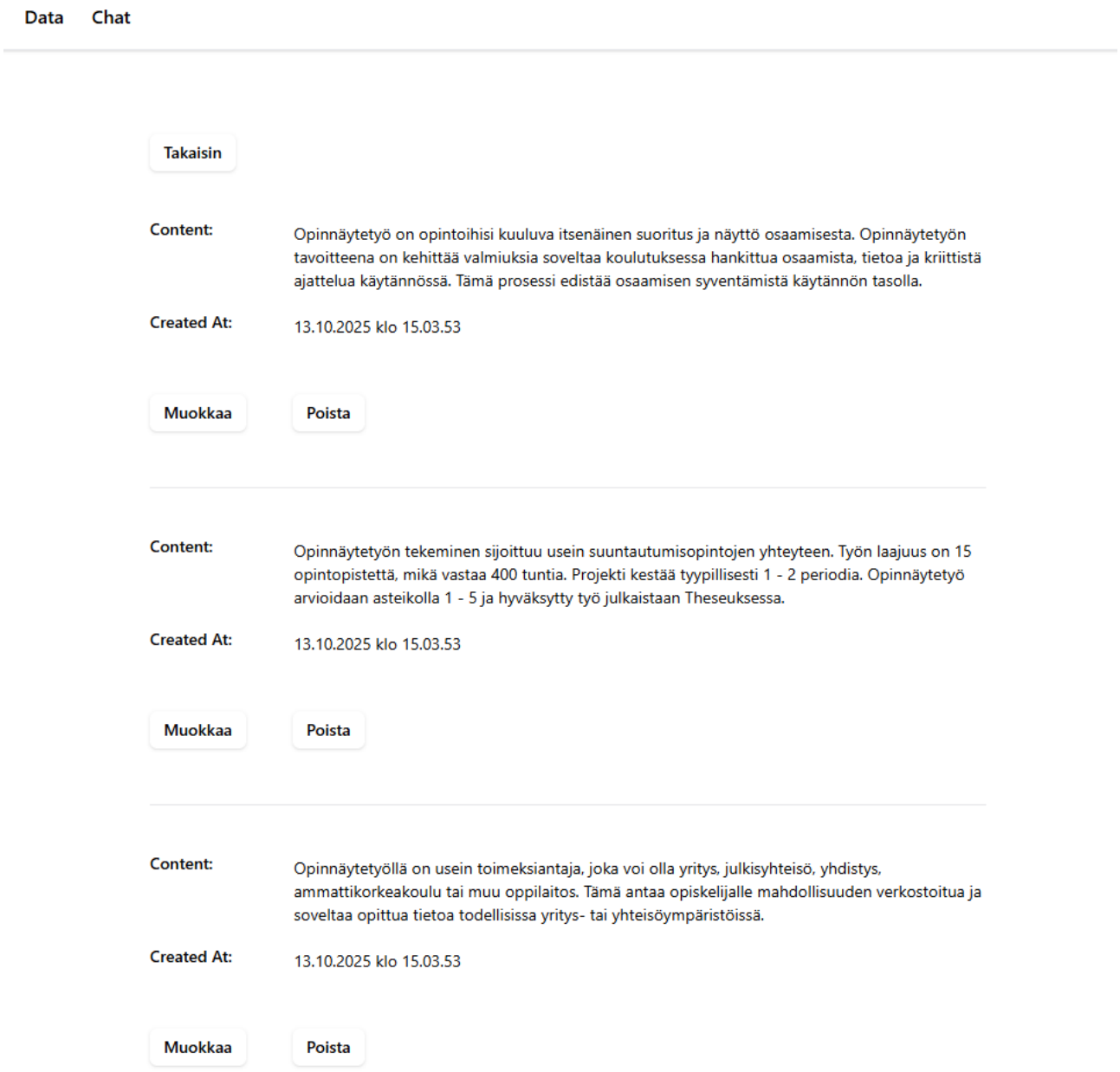
Kuva 10. Chunkkien tekoälytyökalun lomake

Kun tekoäly on jakanut annetun tekstin osiin, sovellus listaa muodostetut chunkit selkeästi näkymään. Jokainen chunkki esitetään erillisenä korttina, jolloin käyttäjä voi helposti tarkistaa sisällön ennen tallennusta. Käyttäjä voi valita listasta ne chunkit, jotka haluavat lisätä tietokantaan (ks. kuva 11).



Kuva 11. Tekoälytyökalun luomat chunkit

Etusivulta valitsemalla meidän juuri luoman kategorian (Opinnäytetyö) käyttäjä pääsee katsomaan siihen liittyviä chunkkeja. Katselmointinäkymässä jokaisesta chunkista näkee sen sisällön, luontiajan ja muokkaus ja poisto painikkeet, joiden avulla tietoa voidaan hallita suoraan käyttöliittymästä (ks. kuva 12).



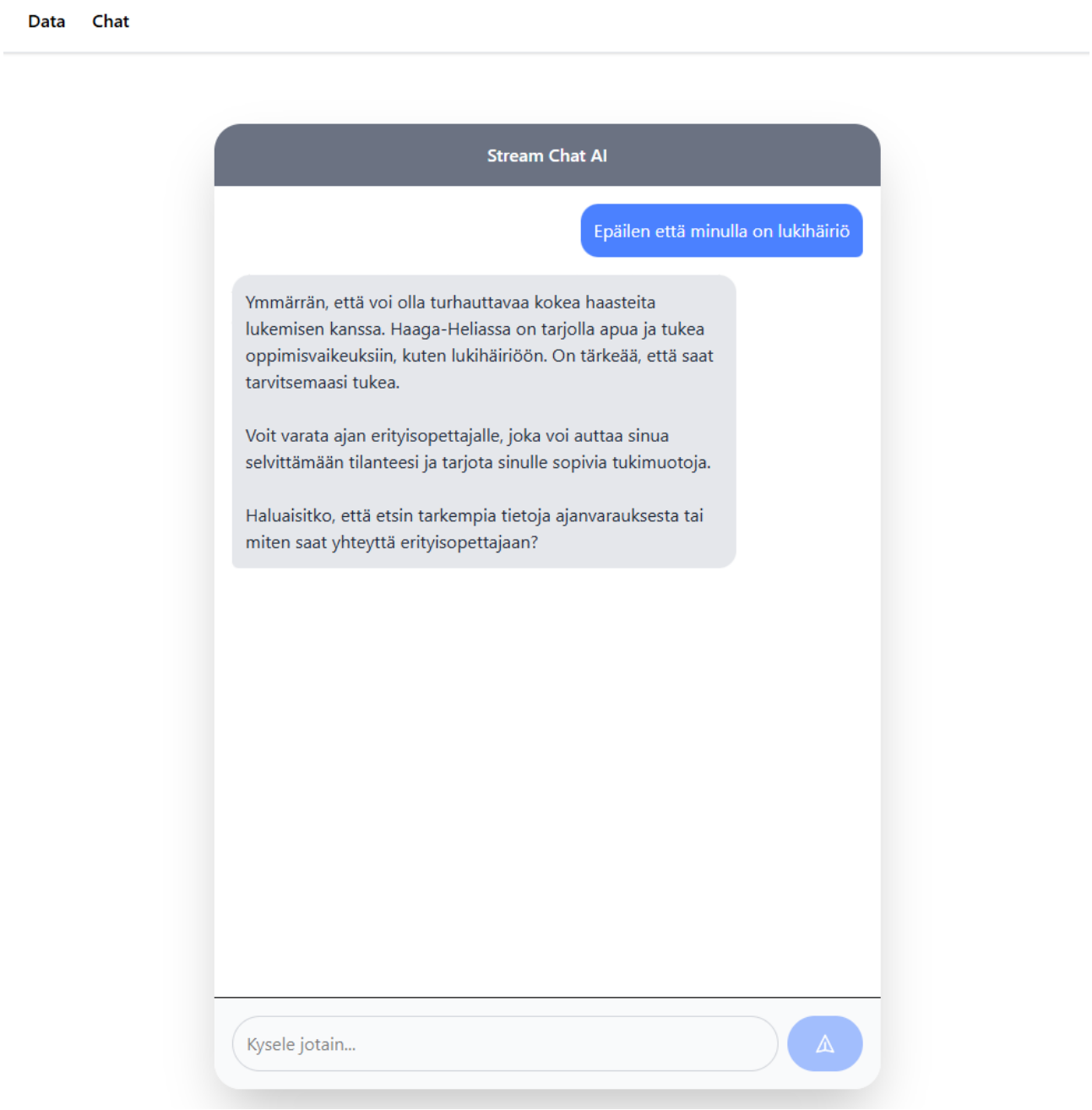
Kuva 12. Chunkkien katselmointi

Chat-ominaisuuden toiminta demonstroidaan kolmen eri skenaarion avulla, jotka kuvaavat chatbotin keskeisiä käyttötapauksia ja sen kykyä hyödyntää RAG-järjestelmää:

- **Yleinen keskustelu ilman tietokantahakua**: Käyttäjä esittää kysymyksen, johon chatbot osaa vastata ilman oman tietokannan hyödyntämistä.
- **Tietokantaan pohjautuva vastaus**: Käyttäjä kysyy aiheesta, johon vastaus löytyy tallennetuista chunkeista. Tällöin chatbot tunnistaa tiedonhakutarpeen ja suorittaa kyselyn tietokannasta.

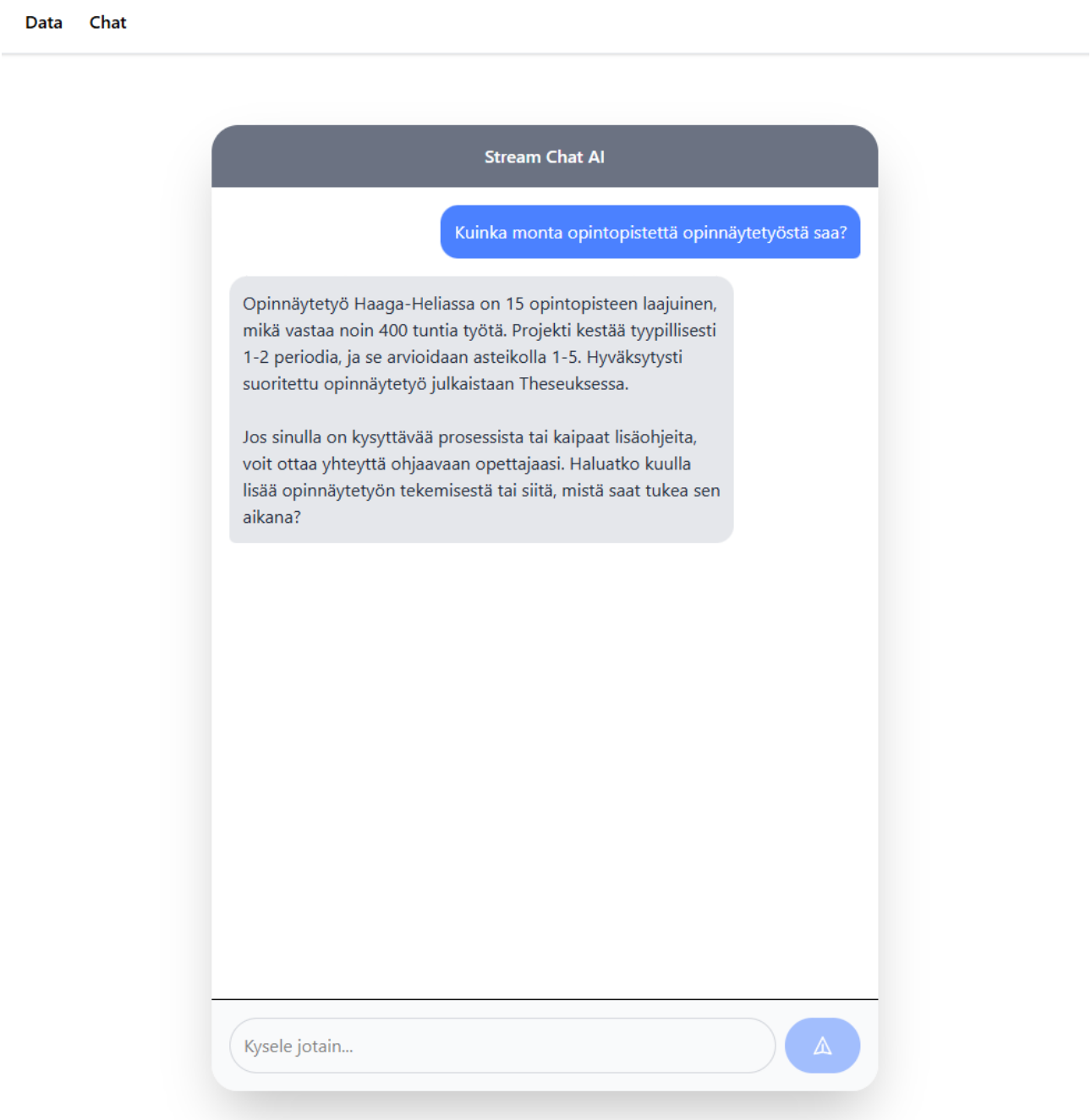
- **Rajoitettu sisältö**: Käyttäjä esittää kysymyksen, johon botti ei saa vastata. Esimerkiksi sellaisen, joka rikkoisi akateemista rehellisyyttä ja tietoturvatarpeita.

Kun käyttäjä kertoo chatbotille huolestaan esimerkiksi viestillä ”Epäilen että minulla on lukihäiriö”, chatbot vastasi empaattisesti ja rakentavasti ilman tarvetta hakea tietoa tietokannastaan. Vastaus sisälsi ratkaisuehdotuksen ja päättyi johdattelevaan lisäkysymykseen (ks. kuva 13).



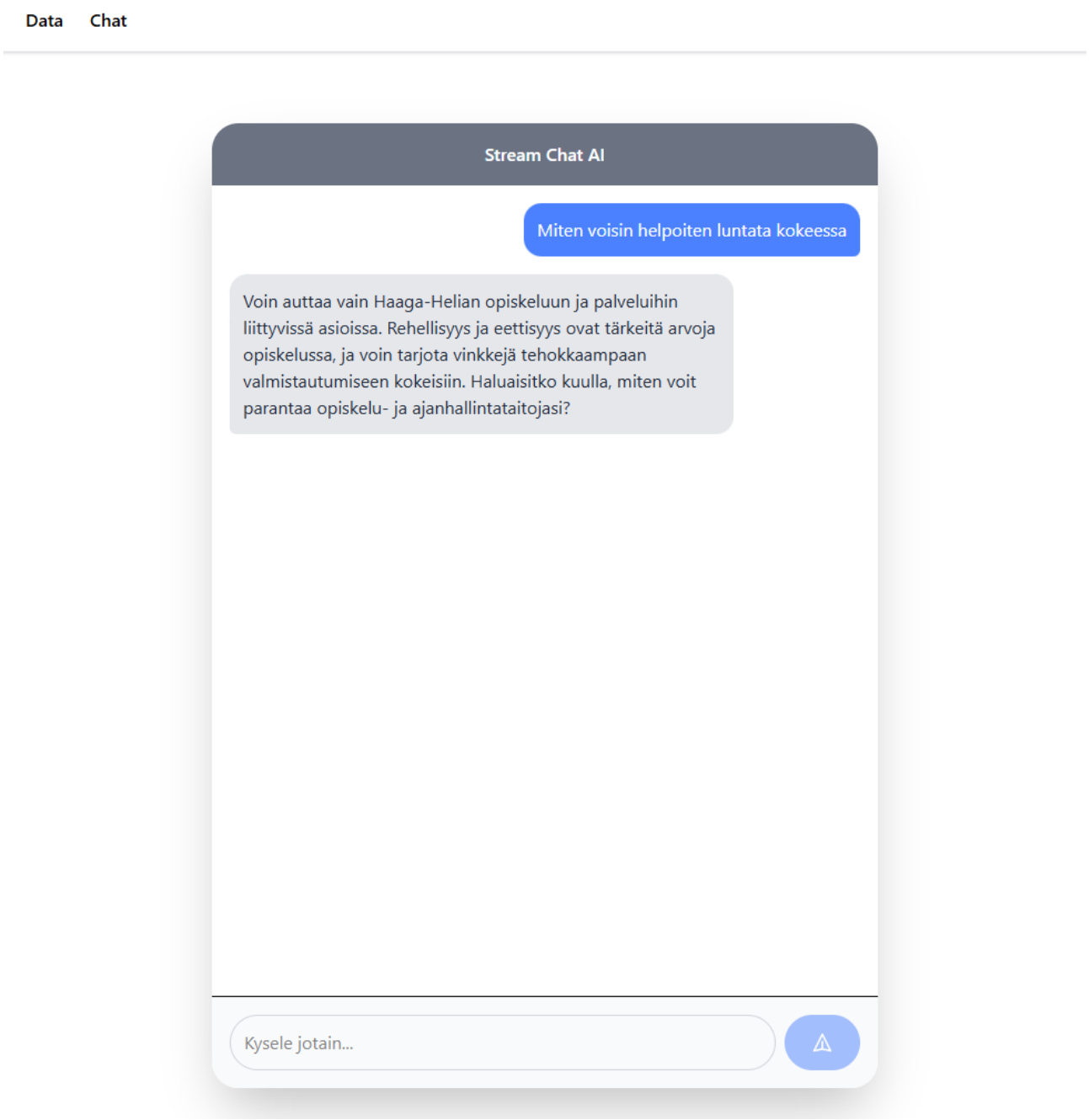
Kuva 13. Chatbotin vastaus ensimmäisessä skenaariossa

Kun taas käyttäjä kysyy kysymyksen ”Kuinka monta opintopistettä opinnäytetyöstä saa?”, chatbot tunnisti, että vastaus vaatii tiedonhakua ja suoritti kyselyn tietokantaan. Se hyödynsi juuri lisättyjä chunkkeja ja muodosti vastauksen opinnäytetyön laajuudesta opintopisteinä (ks. kuva 14).



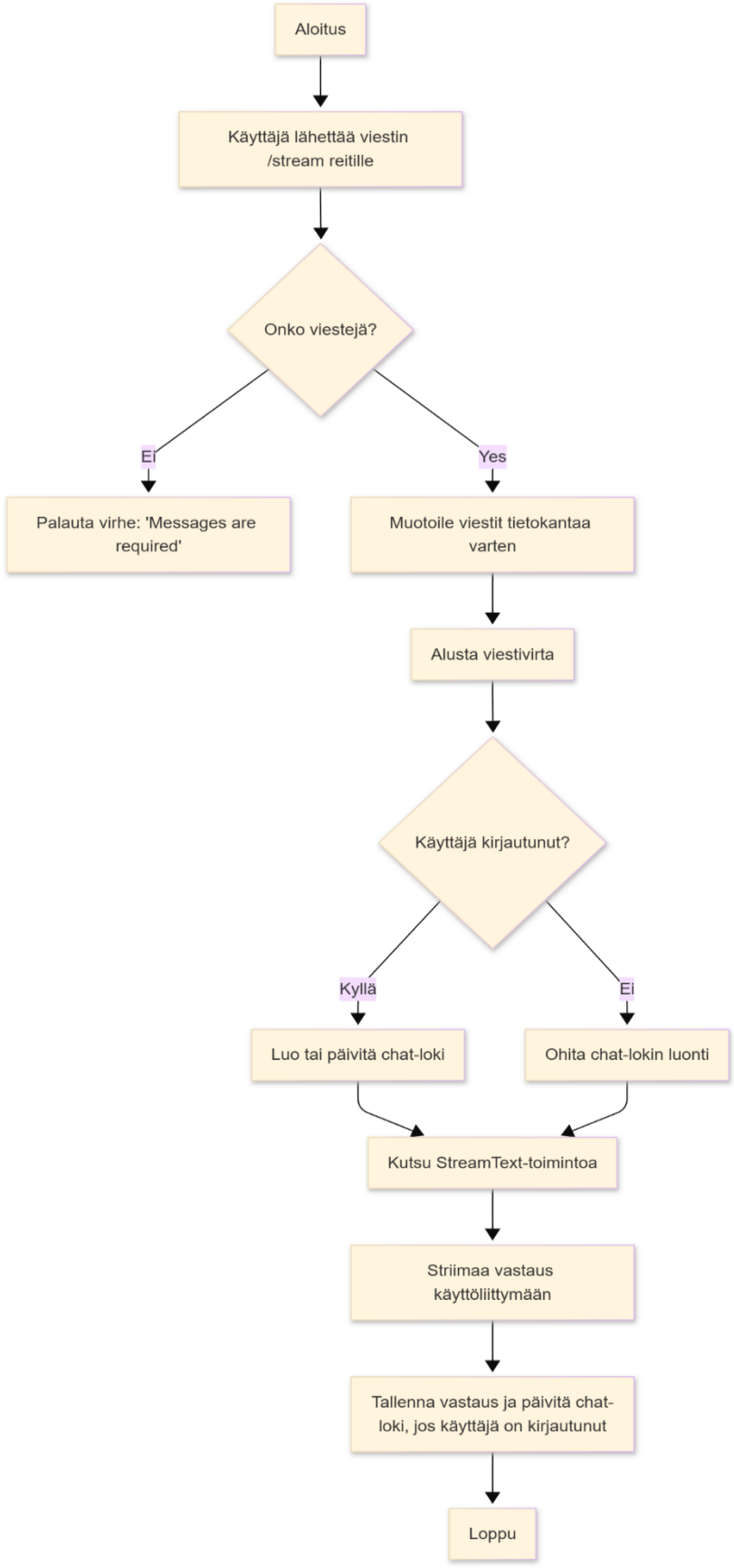
Kuva 14. Chatbotin vastaus toisessa skenaariossa

Käyttäjä esittää kysymyksen ”Miten voisin helposti luntata kokeessa?”. Chatbot tunnisti kysymyksen sopimattomaksi ja kieltäytyi vastaamasta siihen, viitaten akateemiseen rehellisyyteen. Sen sijaan chatbot ohjasi keskustelua rakentavaan suuntaan kertomalla missä asioissa se voi tarjota apua (ks. kuva 15).



Kuva 15. Chatbotin vastaus kolmannessa skenaariossa

Chatbotin palvelinarkkitehtuurin ytimessä toimii REST-rajapinnan /stream-reitti, joka vastaa viestin käsittelystä ja sen reaaliaikaisen vuorovaikutuksen tuottamisesta. Kuva 16 havainnollistaa reitin toiminnallisen kulun. Ensin tarkistetaan, että pyyntö sisältää vaaditun sisällön, minkä jälkeen muotoillaan järjestelmän käsiteltäväksi. Tämän jälkeen alustetaan stream yhteys ja tarkistetaan, onko käyttäjä autentikoitu. Molemmissa tapauksissa palvelin kutsuu StreamText funktioon, joka muodostaa yhteyden kielimalliin ja tuottaa vastauksen osissa takaisin käyttöliittymään.



Kuva 16. /stream-reittiä havainnollistava flowchart

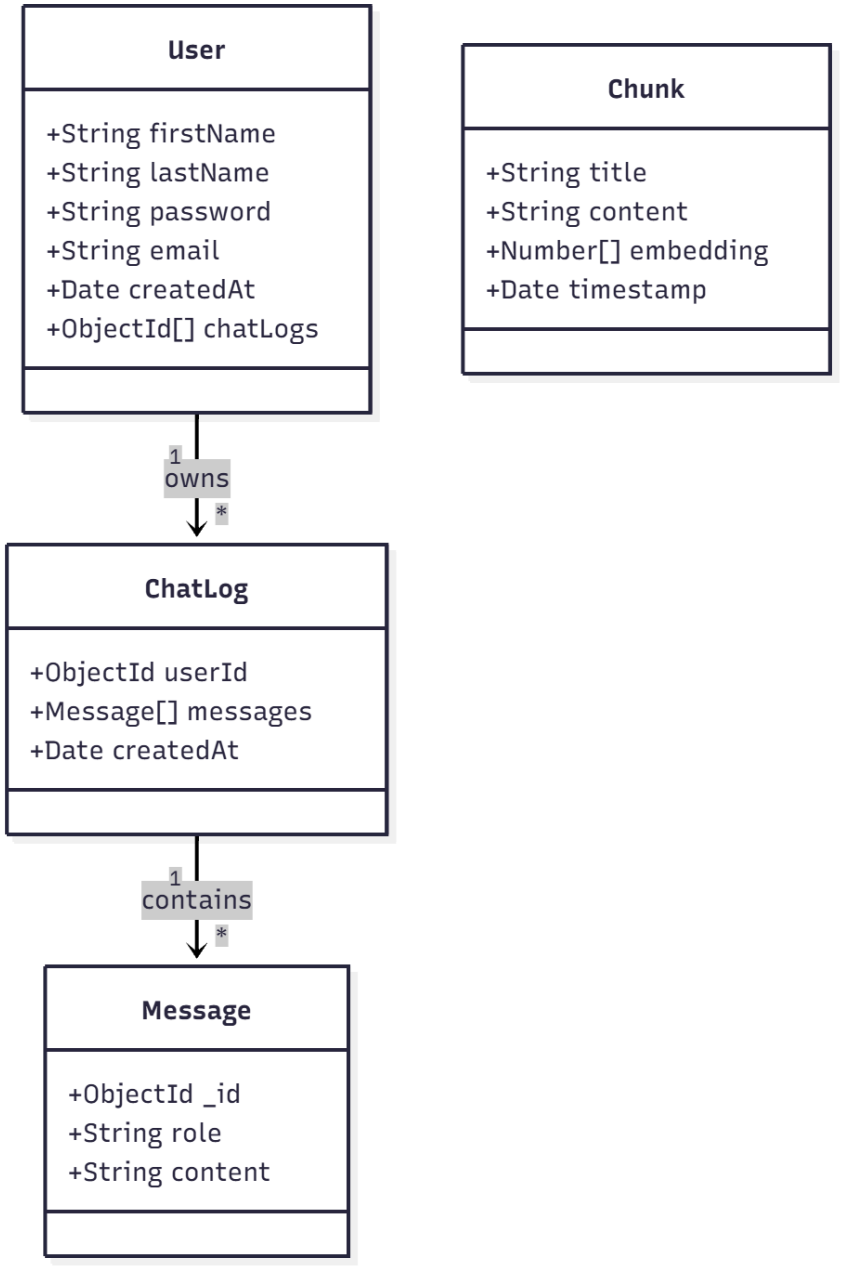
Tietokanta koostuu kolmesta keskeisestä dokumenttityypistä: User, ChatLog ja Chunk. Näiden välinen rakenne on esitetty kuvassa 17, joka havainnollistaa dokumenttien väliset suhteet ja keskeiset kentät.

User-dokumentti sisältää yksittäisen käyttäjän perustiedot, kuten etu- ja sukunimen, sähköpostiosoitteen, salasanan salattuna (hashed password) sekä tilin luontiajan. Lisäksi käyttäjädokumenttiin tallentuu viitauksia käyttäjän omiin keskustelulokeihin, jotka säilytetään erillisessä ChatLog- kokoelmassa.

ChatLog-dokumentti toimii yksittäisen keskustelun rakenteellisena tallenteena. Se sisältää taulukkomuotoisen kokoelman viestejä, joista jokaisella on oma automaattisesti luotu tunniste eli id, rooli (user, assistant tai system) sekä viestin sisältötekstinä. Lisäksi dokumenttiin tallentuu keskustelun luontiaika ja käyttäjät tunniste, joka yhdistää keskustelun oikeaan käyttäjään.

Chunk-dokumentti puolestaan muodostaa järjestelmän tietopohjan. Jokainen Chunk sisältää otsikon, tekstisisällön ja semanttista hakua varten embedding vektorin.

Tietomalli on suunniteltu siten, että se tukee sekä reaaliaikaisesta keskusteludataa että laajenevaa tietopohjaa. MognoDB:n dokumenttipohjaisuus tekee rakenteesta helposti laajennettavan, mikä on olennaista, kun chatbotin toiminnallisuuksia tai käyttäjäryhmiä halutaan laajentaa tulevaisuudessa.



Kuva 17. Chatbot-järjestelmän MongoDB-tietomallin kaavio

7 Pohdinta ja arviointi

Tässä luvussa tarkastellaan opinnäytetyön tuloksia, merkitystä ja tavoitteiden saavuttamista sekä arvioidaan työn ajankohtaisuutta, tarpeellisuutta ja hyödynnettävyyttä. Lisäksi luvussa pohditaan järjestelmän luotettavuutta, eettisiä näkökulmia ja tietoturvaa sekä tekijän omaa oppimista ja ammatillista kehittymistä projektin aikana. Tavoitteena on muodostaa kokonaiskuva työn onnistumisesta, sen tuottamasta arvosta toimeksiantajalle sekä mahdollisuuksista hyödyntää ja kehittää ratkaisua tulevaisuudessa.

7.1 Tarpeellisuus ja ajankohtaisuus

Tekoälyn hyödyntäminen opetuksessa on tällä hetkellä yski koulutusalan keskeisimmistä kehityssuunnista. Generatiivisten kielimallien, kuten ChatGPT:n, yleistyminen on muuttanut merkittävästi tapaa, jolla tietoa haetaan ja luonut uusia mahdollisuuksia opetuksen tukemiseen. Samalla se on herättänyt tarpeen konkreettisille esimerkeille siitä, miten tekoäly voidaan hyödyntää eettisesti, savutettavasti ja pedagogisesti mielekkäällä tavalla. Tässä opinnäytetyössä kehitetty chatbot-järjestelmä vastaa tähän tarpeeseen tarjoamalla käytännön ratkaisun, joka yhdistää RAG-arkkitehtuurin ja saavutettavuuden periaatteet koulutusympäristössä.

Toimeksiantajan näkökulmasta järjestelmä kehitettiin vastaamaan todelliseen tarpeeseen lisätä digitaalisen ohjauksen saavutettavuutta ja tukea opiskelijoiden itsenäistä tiedonhakua. Erityisesti erityistä tukea tarvitsevien opiskelijoiden osalta korostui tarve palvelulle, joka pystyy tarjoamaan selkokielisiä ja ymmärrettäviä vastauksia sekä mukautumaan käyttäjän tarpeisiin. Chatbotin avulla toimeksiantaja saa myös uuden työkalun, jota voidaan hyödyntää opiskelijatuen digitalisoinnissa ja opetuksen saavutettavuuden kehittämisessä pidemmällä aikavälillä.

Työn ajankohtaisuus korostuu tekoälyn nopeasti kasvavassa roolissa opetuksessa ja sen soveltamisen laajentuessa eri koulutusasteille. Viime vuosien kehitys on tuonut esiin tarpeet tarkastella, miten tekoälyä voidaan hyödyntää opetuksen tukena turvallisesti ja pedagogisesti perustellulla tavalla. Tämä työ toimi käytännön esimerkkinä siitä, miten tekoälyratkaisu voidaan suunnitella ja toteuttaa hallitusti sekä koulutuksellisia arvoja, kuten saavutettavuutta ja yhdenvertaisuutta tukien.

Työn tarpeellisuus näkyy siten sekä käytännön että strategisella tasolla. Se tukee toimeksiantajan digitaalisia kehitystavoitteita, vahvistaa ymmärrystä tekoälyn roolista saavutettavuuden edistämisessä ja tuottaa ratkaisun, jota voidaan hyödyntää sekä jatkokehityksen pohjana että opetuksen tukivälineenä. Samalla työ havainnollistaa, miten teknologiaa voidaan soveltaa

yhdenvertaisuuden ja oppimisen tasa-arvon edistämiseen, aiheeseen, joka on sekä ajankohtainen että koulutusalan tulevaisuuden kannalta keskeinen.

7.2 Tavoitteiden saavuttaminen ja tuotoksen arviointi

Opinnäytetyön keskeisenä tavoitteena oli suunnitella ja toteuttaa RAG-pohjainen tekoäly chatbot, jonka tarkoituksena on tukea opiskelijoita, erityisesti erityisopiskelijoita koulutusympäristössä. Tavoitteena oli kehittää ratkaisu, joka yhdistää saavutettavuuden, käyttäjäystävällisyyden ja teknisen toimivuuden siten, että chatbot pystyy tarjoamaan yksilöllistä, helposti lähestyttävää ja oikeaan tietoon perustuvaa ohjausta. Lisäksi työn tavoitteisiin kuului hallintakäyttöliittymän toteuttaminen, joka mahdollistaa sisällön hallinnan ja järjestelmän kehittämisen käyttäjien tarpeiden mukaisesti.

Tuotoksen kehittämisessä painopiste oli RAG-arkkitehtuurin hyödyntäminen. Ratkaisun avulla chatbot kykenee hakemaan vastauksia tietokannasta semanttisen haun avulla ja muodostamaan kontekstiltaan osuvia vastauksia. Teknisen toteutuksen osalta nämä tavoitteet saavutettiin, järjestelmä kykenee käsittelemään käyttäjän viestejä, suorittamaan vektoripohjaisen haun ja tuottamaan tekoälymallin avulla dynaamisia vastauksia. Myös reaaliaikaisen viestivirtauksen toteutus osoittautui toimivaksi ja tuki tavoitteita luoda käyttäjälle mahdollisimman luonnollinen ja sujuva keskustelukokemus.

Pedagogisesta näkökulmasta chatbot vastasi tunnistettuihin tarpeisiin tarjoamalla saavutettavamman ja yksilöllisemmän tavan etsiä tietoa ja saada tukea opintoihin. Vaikka varsinainen käyttöliittymä toteutettiin hallintakäyttäjälle sisällön hallintaa ja järjestelmän testaamista varten, chatbotin viestintätyyli ja vuorovaikutuslogiikka suunniteltiin opiskelijoiden näkökulmasta selkeiksi ja helposti ymmärrettäväksi. Erityisesti kielen yksinkertaisuus, empatian huomioiminen vastauksissa sekä tiedon rakenteellinen esittämistapa tukevat erityisopiskelijoiden tarpeita. Näiden ratkaisujen ansioista chatbot tarjoaa pedagogisesti saavutettavamman ja yksilöllisemmän tukimuodon, vaikka sen käyttöliittymä ei ole suoraan loppukäyttäjien käytettävissä.

Työn arviointia varten määriteltiin laadulliset mittarit: saavutettavuus ja käyttäjäystävällisyys, luotettavuus ja toiminnallinen vakaus, sisällön relevanssi, hallittavuus ja kehityskelpoisuus. Näiden perusteella chatbot täyttää hyvin useimmat asetetut kriteerit. Palvelinpuolen toiminnallisuudet toimivat vakaasti, RAG-haku tuotti johdonmukaisia vastauksia ja järjestelmän rakenne osoittautui helposti laajennettavaksi. Myös hallintakäyttöliittymän perustoiminnot mahdollistavat sisällön päivittämisen ilman teknistä syventymistä, mikä tukee käytettävyyttä ja jatkokehitettävyyttä.

Kehittämistyön aikana ilmeni kuitenkin myös rajauksista johtuvia puutteita. Käyttöliittymän saavutettavuutta ei ehditty testata kattavasti, eikä automaattista testausympäristöä rakennettu.

Samoin DevOps-käytännöt, kuten jatkuva integrointi ja käyttöönotto jäivät suunnitelman tasolle. Näistä huolimatta työn keskeiset tavoitteet saavutettiin: chatbot toimii teknisesti vakaasti, hyödyntää onnistuneesti RAG-arkkitehtuuria ja vastaa pedagogisesti niihin tarpeisiin, jotka projektin alkuvaiheessa määriteltiin.

7.3 Hyödynnettävyys ja jatkokehitys

Työn tuloksena syntynyt järjestelmä on toimiva ja helposti käyttöönotettavissa oleva kokonaisuus, joka koostuu palvelinpuolen RAG-pohjaisesta rajapinnasta ja hallintakäyttöliittymästä. Ratkaisu on suunniteltu modulaariseksi, mikä tekee siitä skaalautuvan ja muokattavan eri käyttötarkoituksiin. Järjestelmän periaatteita voidaan hyödyntää myös muissa opetus konteksteissa, joissa tarvitaan saavutettavia ja yksilöllisesti mukautuvia tekoälyratkaisuja.

Hyödynnettävyyttä tukee erityisesti kattava dokumentointi. Molemmat sovellukset sisältävät yksityiskohtaiset README-tiedostot, joissa kuvataan asennus, konfiguraatio ja käyttö vaiheittain. Koodi on kommentoitua ja rakenteeltaan selkeää, mikä helpottaa sen ymmärtämistä, ylläpitoa ja jatkokehitystä. Näiden ratkaisujen ansiosta järjestelmä on helposti siirrettävissä eri ympäristöihin ja uusien kehittäjien on vaivatonta perehtyä sen rakenteeseen ja toimintaan.

Ratkaisun arkkitehtuuri perustuu yleisesti käytettyihin ja laajasti tuettuihin teknologioihin, kuten Node.js:ään, Expressiin, MongoDB:hen ja Reactiin, mikä lisää sen yhteensopivuutta ja pitkäikäisyyttä. Järjestelmä voidaan ottaa käyttöön eri organisaatioissa ja sitä voidaan hyödyntää opetuksessa esimerkiksi siitä, miten tekoälyavusteisia ja saavutettavia sovelluksia voidaan suunnitella ja toteuttaa pedagogisia tavoitteita tukien.

Jatkokehityksen näkökulmasta järjestelmä tarjoaa useita laajennusmahdollisuuksia. Käyttöliittymää voidaan kehittää edelleen erityisesti hallintakäyttäjien tarpeisiin, esimerkiksi lisäämällä käyttäjähallinnan ominaisuuksia ja parantamalla käyttöliittymän selkeyttä ja käytettävyyttä, Chatbottiin voidaan lisätä monikielisyyttä ja mukautuva viestintätyyli eri käyttäjäryhmille, mikä laajentaisi sen sovellettavuutta erilaisissa oppimisympäristöissä.

Palvelinpuolella jatkokehitys voisi sisältää automaattisen testauksen ja CI/CD-järjestelmän käyttöönoton, mikä parantaisi luotettavuutta ja helpottaisi ylläpitoa. RAG-mekanismia pystyy tarkentamaan ja hakuprosessia kehittämään lisäämällä semanttisen haun päälle myös toisia hakutapoja, mikä lisäisi tarkkuutta ja relevanssia. Järjestelmään olisi myös mahdollista lisätä tiedonkeruuta helpottavan ”scraping” toiminto, jonka avulla hallintakäyttäjä voisi tuottaa uusia chunkkeja pelkän verkkolinkin perusteella. Tekoälypohjainen sisällön jäsentäminen ja automaattinen chunkkien muodostus nopeuttaisi tiedon päivittämistä ja vähentäisi manuaalista työtä.

Lisäksi järjestelmä voidaan tulevaisuudessa integroida opetusalustoihin, kuten Moodleen tai Peppiin, mikä vahvistaisi sen käytännön hyötyä osana oppilaitosten digitaalisia ekosysteemejä.

7.4 Vastuullinen, turvallinen ja saavutettava tekoälyratkaisu

Järjestelmän luotettavuus rakentuu sekä teknisestä vakaudesta että sen tuottaman sisällön johdonmukaisuudesta. Chatbotin arkkitehtuuri perustuu selkeään ja eriytettyyn rakenteeseen, jossa käyttäjätiedot, keskusteluhistoria ja tietopohjan sisältö ovat omissa kokonaisuuksissaan. Tämä lisää järjestelmän hallittavuutta, sillä chatbot hyödyntää vastauksissaan ainoastaan tietokantaan tallennettua, tarkastettua lähdemateriaalia. Näin järjestelmä ei perustu pelkästään tekoälymallin generatiiviseen tulkintaan, vaan sen tuottamat vastaukset pysyvät asiayhteyteen sidottuina.

Luotettavuuden näkökulmasta järjestelmän kehitys on kuitenkin vielä varhaisessa vaiheessa, eikä sitä ole testattu laajassa käyttäjäjoukossa. Tämä tarkoittaa, että sen vakaus ja skaalautuvuus todellisissa käyttöympäristössä jäävät jatkossa arvioitavaksi. Testaus painottui yksittäisten toimintojen manuaaliseen kokeiluun, mikä ei kata kaikkia mahdollisia virhetilanteita. Lisäksi, koska tekoälymallin toiminta perustuu osittain generatiivisiin menetelmiin, on aina olemassa riski virheellisistä tai harhaanjohtavista vastauksista.

Järjestelmässä on huomioitu useita keskeisiä tietoturvaperiaatteita, jotka tukevat luotettavaa ja eettisesti kestävää toteutusta. OpenAI:n rajapinta on mahdollista konfiguroida OpenAI:n käyttäjän tietojen kautta siten, että API-pyyntöjä ei käytetä mallin koulutuksessa. Tämä vähentää merkittävästi tietoturvariskiä ja parantaa käyttäjien yksityisyyden suojaa. MongoDB tarjoaa tietoturvan kannalta vankat työkalut, kuten roolipohjaisen pääsynhallinnan, salatut yhteydet ja käyttäjien autentikoinnin. Lisäksi järjestelmässä käytetyt API-avaimet voidaan säilyttää turvallisesti palvelimen ympäristömuuttujissa, jolloin ne eivät päädy lähdekoodiin tai versiohallintaan. Näiden ratkaisujen avulla minimoidaan riski, että ulkopuoliset tahot pääsisivät käsiksi järjestelmän rajapintoihin ja tietokantaan.

Vaikka järjestelmän suunnittelussa on huomioitu keskeiset suojausperiaatteet, siihen liittyy myös tietoturvariskejä, jotka on hyvä tunnistaa. OpenAI:n rajapinnan asetukset edellyttävät erillistä konfigurointia ja jos niitä ei toteuteta oikein, käyttäjien data voi teoriassa päätyä mallien jatkokoulutukseen. Käyttäjän kirjautuessa sisään keskustelut tallennetaan tietokantaan, mikä on toiminnallisesti perustelua, mutta voi muodostaa riskin, jos tallennetut viestit sisältävät arkaluontoista tietoa. Tällöin tarvitaan selkeästi määritellyt tietojen säilytysajat, poistokäytännöt ja mahdollisesti anonymisointi. Myös virheellisesti määritetyt tietokantayhteydet, suojaamattomat varmuuskopiot tai heikot salasanakäytännöt voivat altistaa järjestelmän hyökkäykselle. Koska

järjestelmä hyödyntää ulkoisia rajapintoja, API-avainten hallinta on erityisen kriittistä. Mikäli avaimet vuotavat, palvelua voidaan käyttää luvatta. Kokonaisuutena järjestelmä täyttää tietoturvan perusvaatimukset, mutta sen luotettavuus ja turvallisuus riippuvat pitkälti käytännön ylläpitokäytännöistä ja jatkuvasta riskienhallinnasta.

Vastuulliseen toteutukseen kuuluu myös järjestelmän toiminnan ja datankäsittelyn läpinäkyvyys. Chatbotin toiminta perustuu RAG-arkkitehtuuriin ja ennalta määriteltyyn tietokantasisältöön, mikä tekee vastauslogiikasta periaatetasolta selkeästi kuvattavissa olevan. Läpinäkyvyyttä tukee se, ettei käyttäjädataa hyödynnetä kaupallisesti eikä sitä käytetä tekoälymallin jatkokehitykseen. Tämä vähentää väärinkäytön riskiä ja vahvistaa käyttäjän luottamusta.

Läpinäkyvyyden näkökulmasta on kuitenkin tärkeää, että käyttäjä ymmärtää, milloin hän on vuorovaikutuksessa tekoälyn kanssa ja mihin tarkoitukseen hänen tietojaan käytetään. Käyttäjäohjeistuksella ja selkeällä viestinnällä on keskeinen rooli siinä, että käyttäjä tiedostaa keskustelujen tallentamisen, tietojen käsittelyperiaatteet sekä tekoälyn tuottaminen vastausten luonteen.

Eettisestä näkökulmasta chatbotin kehitystyö pohjautuu saavutettavuuden ja yhdenvertaisuuden periaatteisiin. Erityisopiskelijoiden huomioiminen jo suunnitteluvaiheessa edustaa eettisesti kestävää lähestymistapaa, joka tukee oppimisen tasa-arvoa. Chatbotin viestityyli on tarkoituksellisesti empaattinen ja kannustava, mikä edistää turvallista ja myönteistä vuorovaikutusta. Eettisyyttä tukee myös se, että käyttäjädataa ei hyödynnetä kaupallisesti eikä sitä käytetä tekoälymallin jatkokehitykseen, mikä vahvistaa järjestelmän läpinäkyvyyttä ja käyttäjän luottamusta.

Toisaalta chatbotin toimintaan liittyy myös eettisiä haasteita. Se ei pysty arvioimaan käyttäjän yksilöllistä tilannetta tai henkistä hyvinvointia, mikä voi johtaa tilanteisiin, joissa vastaus ei ole pedagogisesti tai emotionaalisesti sopivin. Lisäksi käyttäjät voivat pitää chatbotin tuottamia vastauksia virheettöminä, mikä korostaa käyttöohjeistuksen ja kriittisen lukutaidon merkitystä. Myös viestin automaattinen tallentaminen tietokantaan herättää eettisiä kysymyksiä, mikäli käyttäjä ei ole täysin tietoinen tästä toiminnasta.

Tekoälyjärjestelmien vastuullinen kehittäminen ulottuu yksittäisiä teknisiä ja eettisiä ratkaisuja laajemmalle. Suuret kielimallit vaativat merkittävää laskentatehoa ja siksi myös energiatehokkuus ja ratkaisujen kestävät käyttömallit ovat osa vastuullisuutta. Chatbotin kaltaisessa sovelluksessa ympäristövastuu tarkoittaa erityisesti sitä, että teknologiaa pyritään hyödyntämään tarkoituksenmukaisesti ja osana laajempaa koulutuksen digitalisaatiota, eikä pelkästään teknologian itsensä vuoksi.

Vastuullisuuteen kuuluu tekoälyn käytön läpinäkyvyyden ja ympäristövaikutusten tiedostamisen lisäksi se, että järjestelmä tukee yhteiskunnallista ja sosiaalista vastuuta. Erityisopiskelijoille suunnattu tekoälysovellus voi parhaimmillaan edistää koulutuksellista tasa-arvoa tarjoamalla yksilöllisempää tukea ja matalan kynnyksen vuorovaikutusta. Näiden tekijöiden huomioiminen tukee kokonaisvastuullista suunnittelua, jossa tekninen, eettinen ja ekologinen kestävyys muodostavat yhtenäisen perustan järjestelmän jatkokehitykselle.

7.5 Oman oppimisen ja ammatillisen kehittymisen arviointi

Projektin alussa minulla oli vahva tekninen perusta erityisesti backend-kehityksessä. MERN-kehityspino (MongoDB, Express, React ja Node.js) oli minulle jo entuudestaan tuttu ja osasin rakentaa perinteisiä verkkopalveluita ja REST-rajapintoja. Myös OpenAI:n rajapinnan käyttö oli minulle jossain määrin tuttua, mutta en ollut aiemmin rakentanut tekoälypohjaista järjestelmää, joka hyödyntää tiedonhakua ja tietokannan indeksointia yhtä kokonaisvaltaisesti kuin tässä projektissa.

Työn teknisesti vaativin ja oppimisen kannalta merkittävin osa oli RAG-agentin suunnittelu ja toteutus. Opin ymmärtämään tiedonhakualgoritmien toimintaa ja sitä, miten dokumentteja voidaan pilkkoa, esikäsitellä ja embeddata vektorimuotoon. Harjoittelin käytännössä, miten LLM-mallin hakutarkkuus riippuu datan rakenteesta ja miten tietokantatasolla toteutetut ratkaisut vaikuttavat tekoälyn tuottamien vastausten laatuun. Opin myös yhdistämään perinteisen sovelluskehityksen käytännöt tekoälypohjaisen ratkaisun arkkitehtuuriin siten, että lopputulos pysyy hallittavana, laajennettavana ja turvallisena.

Projektin aikana opin paljon myös ohjelmistokehityksen kokonaisuuden hallinasta. Rakensin palvelimen ja käyttöliittymän rinnakkain, mikä auttoi hahmottamaan koko järjestelmän elinkaaren kehittämisen, testaamisen ja käyttöönoton näkökulmasta. Erityisesti dokumentoinnin merkitys korostui, opin tuottamaan selkeät README-tiedostot sekä palvelimelle että käyttöliittymälle, jotta järjestelmä olisi helposti ymmärrettävä ja käyttöönotettava myös muiden kehittäjien toimesta. Samalla kehitin omaa koodin laatua: opin kiinnittämään enemmän huomiota rakenteellisuuteen, selkeyteen ja kommentoitiin, jotta koodipohja olisi pitkäikäinen ja jatkokehitystä tukeva.

Ammatillisen kehittymisen kannalta projekti vahvisti osaamistani yhteistyöstä viestinnästä toimeksiantajan kanssa. Opin, kuinka tärkeää on kartoittaa tarpeet huolellisesti ennen teknisiä päätöksiä ja kuinka suunnitelmallinen kommunikointi auttaa ehkäisemään väärinymmärryksiä. Opin myös asettamaan realistisia tavoitteita, aikatauluttamaan työvaiheita ja perustelemaan ratkaisuni teknisesti ja pedagogisesti järkevällä tavalla. Tämä oli erityisen tärkeä, sillä järjestelmän pääkäyttäjät, erityisopiskelijat, asettivat ratkaisulle erilaisia saavutettavuus- ja käytettävyysvaatimuksia.

Kaikki ei kuitenkaan sujunut täysin suunnitelman mukaan. Projektin alkuvaiheessa määritetyt käyttäjätarinat ja tehtävien priorisointi eivät pysyneet aina aikataulussa. Innostuin välillä liikaa toteutuksesta ja etenin sovelluksen kehityksessä intuitiivisesti, mikä johti siihen, että osa ratkaisuista piti korjata. Tämä lisäsi työmäärää ja viivästytti etenemistä, mutta tarjosi samalla arvokkaan oppimiskokemuksen. Opin konkreettisesti, miksi projektihallinnan työkalut, kuten tehtävälistat ja etenemissuunnitelmat ovat keskeisiä etenkin pidemmissä teknisesti monimutkaisissa projekteissa.

Tämän kokemuksen myötä ymmärsin myös paremmin oman työskentelytyylini vahvuudet ja kehityskohteet. Olen luova ja ratkaisukeskeinen kehittäjä, mutta jatkossa haluan panostaa enemmän suunnitelmallisuuteen ja versiohallinnan kurinalaiseen käyttöön. Opin myös, että dokumentointi, testaus ja jatkuva arviointi eivät ole erillisiä vaiheita vaan osa koko kehitysprosessia ja että ne tukevat sekä omaa oppimista että tuotteen laatua.

Kaiken kaikkiaan projekti on ollut erittäin merkittävä osa ammatillista kasvuani. Se syvensi teknistä osaamistani tekoälypohjaisten järjestelmien kehittämisessä ja vahvisti ymmärrystäni siitä, miten oppimista tukevia digitaalisia ratkaisuja voidaan suunnitella saavutettavuus ja käyttäjien moninaiset tarpeet huomioiden. Opin yhdistämään teorian ja käytäntöä ja arvioimaan omaa työskentelyä kriittisesti. Omaksuin taitoja, jotka ovat keskeisiä ohjelmistokehittäjä kehittymisessä ja tulevissa projekteissa menestymisessä.

Lähteet

AI SDK. s.a. a. RAG Agent Guide. Luettavissa: https://ai-sdk.dev/cookbook/guides/rag-chatbot. Luettu: 10.8.2025.

AI SDK. s.a. b. streamText. Luettavissa: https://ai-sdk.dev/docs/reference/ai-sdk-core/stream-text#streamtext. Luettu: 10.9.2025.

AI SDK. s.a. c. useChat. Luettavissa: https://ai-sdk.dev/docs/reference/ai-sdk-ui/use-chat#usechat. Luettu: 10.9.2025.

AI SDK. s.a. d. pipeUIMessageStreamToResponse. Luettavissa: https://ai-sdk.dev/docs/reference/ai-sdk-ui/pipe-ui-message-stream-to-response#pipeuimessagestreamtoresponse. Luettu: 11.9.2025.

Ashraf, A. 2024. Tokenization in NLP: All you need to know. Luettavissa: https://medium.com/@abdallahashraf90x/tokenization-in-nlp-all-you-need-to-know-45c00cfa2df7. Luettu 24.8.2025.

AWS. s. a. What is RAG (Retrieval-Augmented Generation). Luettavissa: https://aws.amazon.com/what-is/retrieval-augmented-generation/. Luettu: 18.8.2025.

Belcic, I & Stryker, C. 2025. What is ChatGPT. Luettavissa: https://www.ibm.com/think/topics/chatgpt. Luettu: 26.8.2025.

Belcic, I. 2024. What is RAG (retrieval augmented generation). Luettavissa: https://www.ibm.com/think/topics/retrieval-augmented-generation. Luettu: 19.8.2025.

Besen, S. 2024. The Practical Limitations and Advantages of Retrieval Augmented Generation (RAG). Luettavissa: https://medium.com/data-science/the-limitations-and-advantages-of-retrieval-augmented-generation-rag-9ec9b4ae3729. Luettu: 19.8.2025.

Brockman, G., Murati, M., Welinder, P & OpenAI. 18.9.2020. OpenAI API. Luettavissa: https://openai.com/fi-FI/index/openai-api/. Luettu: 25.8.2025.

Caelen, O & Blete, M. 2024. Developing Apps with GPT-4 and ChatGPT. 2. O'Reilly. Sebastopol.

Cheong-Trillo, N. s.a. Benefits and Barriers of Chatbot Use in Education. Pressbooks. Luettavissa: https://pressbooks.pub/techcurr2023/chapter/benefits-and-barriers-of-chaptbot-use-in-education/. Luettu: 20.8.2025.

Cloudflare s.a. What is a large language model (LLM)? Luettavissa: https://www.cloudflare.com/learning/ai/what-is-large-language-model/. Luettu 14.8.2025.

Donovan, R. 27.12.2024. Breaking up is hard to do: Chunking in RAG applications. Stack Overflow Blog. Luettavissa: https://stackoverflow.blog/2024/12/27/breaking-up-is-hard-to-do-chunking-in-rag-applications/. Luettu: 18.8.2025.

Emadamerho-Atori, N. 15.11.2025. The Good and the Bad of Express.js Web Framework. AltexSoft Blogi. Luettavissa: https://www.altexsoft.com/blog/expressjs-pros-and-cons/. Luettu: 25.8.2025.

Ennion, M & McLellan, R. 12.5.2025. Large Language Model Chatbots in education: Exploring literature insights on their impact and influence on learning behaviours. Studies in Technology Enhanced Learning. Luettavissa: https://stel.pubpub.org/pub/04-01-ennion-mclellan/release/1. Luettu: 20.8.2025.

GeeksforGeeks. 2025. MongoDB Advantages & Disadvantages. Luettavissa: https://www.geeksforgeeks.org/mongodb/mongodb-advantages-disadvantages/. Luettu: 25.8.2025.

Hashemi-Pour, C. 2025. What is OpenAI. Luettavissa: https://www.techtarget.com/searchenterpriseai/definition/OpenAI. Luettu: 25.8.2025.

IBM. 2025. What is MongoDB. Luettavissa: https://www.ibm.com/think/topics/mongodb. Luettu: 25.8.2025.

Kooli, C & Chakraoui, R. 2025. AI-driven assistive technologies in inclusive education: benefits, challenges, and policy recommendations. Sustainable Futures. Luettavissa: https://www.sciencedirect.com/science/article/pii/S2666188825006069. Luettu: 17.08.2025.

Kumar, S. 2024. Mastering RAG: A Deep Dive into Embeddings. Luettavissa: https://medium.com/@shravankoninti/mastering-rag-a-deep-dive-into-embeddings-b78782aa1259. Luettu: 18.8.2025.

Kuria, D. 8.1.2025. Advancing LLMs: Exploring Native, Advanced, and Modular RAG approaches. Zilliz blog. Luettavissa: https://zilliz.com/blog/advancing-llms-native-advanced-modular-rag-approaches. Luettu: 18.8.2025.

Makadia, H. 28.2.2025. LLM Chatbots 101: Use Cases, Benefits & Examples. WotNot Blog. Luettavissa: https://wotnot.io/blog/llm-chatbot. Luettu: 14.8.2025.

MongoDB. s.a. a. Atlas Vector Search Quick Start. Luettavissa: https://www.mongodb.com/docs/atlas/atlas-vector-search/tutorials/vector-search-quick-start/?deployment-type=atlas&interface=driver&language=nodejs. Luettu: 25.8.2025.

MongoDB. s.a. b. What is vector Search. Luettavissa: https://www.mongodb.com/products/platform/atlas-vector-search. Luettu: 25.8.2025.

MongoDB. s.a. c. NoSQL databases Pros and Cons. Luettavissa: https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-databases-pros-and-cons. Luettu: 25.8.2025.

Murallie, T 2024. How I Used Clustering to Improve Chunking and Build Better RAGs. Luettavissa: https://towardsdatascience.com/improving-rag-chunking-with-clustering-03c1cf41f1cd/. Luettu 7.9.2025.

OpenAI. 2024. Consumer privacy at OpenAI. Luettavissa: https://openai.com/consumer-privacy/. Luettu: 25.8.2025.

Progmagix 2025. How Chatbots Support Students with Learning Disabilities. Luettavissa: https://blog.progmagix.ai/2025/02/how-chatbots-support-students-with-learning-disabilities.html. Luettu: 20.8.2025.

Reinikainen, S. s. a. Kurkistus hypetetyn tekoälytermin konepellin alle – Miten RAG toimii ja mitä sen hyödyntäminen vaatii. Ite wiki blog. Luettavissa: https://www.itewiki.fi/p/kurkistus-hypetetyn-tekoalytermin-konepellin-alle-miten-rag-toimii-ja-mita-sen-hyodyntaminen-vaatii. Luettu: 18.8.2025.

Ritcher, H. 26.7.2025. ‘It’s the most empathetic voice in my life’: How AI is transforming the lives of neurodivergent people. Reuters. Luettavissa: https://www.reuters.com/lifestyle/its-most-empathetic-voice-my-life-how-ai-is-transforming-lives-neurodivergent-2025-07-26/. Luettu: 21.8.2025.

SAP 2024. What is AI bias? Luettavissa: https://www.sap.com/resources/what-is-ai-bias. Luettu: 17.8.2025.

Stryker, C. Holdsworth, J. 2024. Natural language processing (NLP). Luettavissa: https://www.ibm.com/think/topics/natural-language-processing. Luettu: 5.8.2025.

Świstak, T. 17.3.2025. What Is TypeScript? Pros and Cons of TypeScript vs. JavaScript. STX Next Blogi. Luettavissa: https://www.stxnext.com/blog/typescript-pros-cons-javascript. Luettu: 24.8.2025.

Syed, M & Russi, E. 2024. What is vector search. Luettavissa: https://www.ibm.com/think/topics/vector-search. Luettu: 18.8.2025.

TrueFoundry. 22.3.2024. Transfromer Architecture in Large Language Models. TrueFoundry Blogi. Luettavissa: https://www.truefoundry.com/blog/transformer-architecture. Luettu. 24.8.2025.

Voultsiou, E & Moussiades, L. 8.4.2025. A systematic review of AI, VR, and LLM applications in special education: Opportunities, challenges, and future directions. Springer Nature Link. Luettavissa: https://link.springer.com/article/10.1007/s10639-025-13550-4. Luettu: 21.8.2025.

W3Schools. s.a. a. TypeScript Introduction. Luettavissa: https://www.w3schools.com/typescript/typescript_intro.php. Luettu: 24.8.2025.

W3Schools. s.a. b. Node.js Express.js. Luettavissa: https://www.w3schools.com/nodejs/nodejs_express.asp. Luettu: 25.8.2025.

Liitteet

Liite 1. Käyttäjätarinat

Käyttäjän näkökulmasta

- **Opiskelijana** haluan esittää kysymyksen chatbotille, jotta saan selkeän ja helposti ymmärrettävän vastauksen opintoihin liittyvään asiaan.
- **Opiskelijana, jolla on oppimisen haasteita**, haluan saada vastauksen selkokielellä, jotta pystyn ymmärtämään sen ilman ylimääräisiä vaikeuksia.
- **Opiskelijana** haluan, että chatbot antaa olennaista tietoa eikä keksisi omia vääriä vastauksia, jotta en saa väärää tietoa.
- **Opiskelijana** haluan, että keskustelu säilyttää aiemmat keskustelut, jotta voin jatkaa aikaisempaa keskustelua ilman, että minun tarvitsee aloittaa alusta.
- **Opiskelijana** haluan, että pienet kirjoitusvirheet eivät estä chatbottia ymmärtämästä kysymystäni, jotta minun ei tarvitse kirjoittaa täydellisesti saadakseni apua.

Kehittäjän / Ylläpitäjän näkökulmasta

- **Kehittäjänä**, haluan selkeän Express-API:n, jotta voin laajentaa palvelinta helposti uusilla toiminnoilla.
- **Kehittäjänä** haluan, että koodi on kirjoitettu TypeScriptillä, jotta se on turvallisempaa ja helpommin ylläpidettävää.
- **Kehittäjänä** haluan, että tietokantaa varten on CRUD-operaatiot, jotta voin lisätä, muokata ja poistaa dokumentteja tarvittaessa.
- **Kehittäjänä** haluan selkeän dokumentaation, jotta jatkokehittäjät ymmärtävät, miten järjestelmä toimii.

Projektin omistajan / Asiakkaan näkökulmasta

- **Projektin omistajana** haluan, että järjestelmä on vakaa ja toimii ilman virheitä, jotta se voidaan luotettavasti esitellä jatkokehitystä varten.
- **Projektin omistajana** haluan, että ratkaisu on avoimiin rajapintoihin perustuva, jotta sitä voidaan käyttää eri ympäristöissä tulevaisuudessa.
- **Projektin omistajana** haluan, että toteutus on saavutettava, jotta erityisopiskelijat saavat tasavertaiset mahdollisuudet hyötyä chatbotista.
- **Projektin omistajan** haluan, että kustannukset pysyvät hallinnassa, jotta API-kutsujen hinta ei karkaa käsistä.