LAPIN AMK
Lapland University of Applied Sciences

TOIMEKSIANTAJAN SISÄISEN CHATIN ANALYSOINTISOVELLUKSEN KEHITTÄMINEN

Puranen Jenny

Opinnäytetyö

Tietojenkäsittelyn koulutus
Tradenomi (AMK)

2025

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tietojenkäsittelyn koulutus
Tradenomi (AMK)

Tekijä	Jenny Puranen	**Vuosi**	2025
Ohjaaja	Yrjö Koskenniemi		
Toimeksiantaja	Suomalainen keskisuuri yritys		
Työn nimi	Toimeksiantajan sisäisen chatin analysointisovelluksen kehittäminen		
Sivumäärä	62 + 5		

Opinnäytetyön tavoitteena oli kehittää käyttäjäystävällinen ja selkeä Proof of Concept (PoC) -sovellus, joka tukee keskisuuren suomalaisen yrityksen sisäisen chatbotin kehittämistä. Yrityksellä oli tarve analysoida kuukausittaisia chat-lokeja tunnistaakseen chatbotin tietopohjan mahdollisia puutteita sekä henkilöstön koulutustarpeita. Opinnäytetyön pääasiallisena tutkimuskysymyksenä oli, kuinka kehitetään sovellus, jonka avulla voidaan analysoida chatbotin keskusteluista kerättyä dataa liiketoiminnan ja chatbotin kehittämisen tueksi. Tukikysymyksinä selvitettiin, millaisia kysymyksiä ohjautuu eteenpäin asiantuntijalle ja miten toistuvia kehityskohteita voidaan tunnistaa asiantuntijalle ohjatuista kysymyksistä.

Tietoperustassa tarkasteltiin sisäisen viestinnän haasteita, chatbotteja viestinnän tukena sekä datan analysoinnin menetelmiä. Työssä sovellettiin monimenetelmällistä lähestymistapaa, jossa yhdistettiin kvalitatiivisia ja kvantitatiivisia menetelmiä. Kvalitatiivisessa vaiheessa kartoitettiin yrityksen tarpeita haastattelujen avulla, kun taas kvantitatiivisessa analyysissä hyödynnettiin Python-ohjelmointikieltä ja pandas-kirjastoa viestihistorian analysointiin. Sovelluksen toteutus tehtiin Flask web -kehyksellä ja sen selainpohjainen käyttöliittymä suunniteltiin esittämään analyysin tulokset helposti ymmärrettävässä muodossa.

Opinnäytetyön tuloksena syntyi PoC-sovellus, joka analysoi chatbotin keräämää viestihistoriaa ja luokittelee ne avainsanojen perusteella palvelukategorioihin. Sovellus voi auttaa yritystä tunnistamaan keskeiset kehityskohteet ja priorisoimaan kehitystoimenpiteet tehokkaammin. Lisäksi se voi tukea strategista päätöksentekoa tarjoamalla analyysituloksia visuaalisessa muodossa sekä ladattavina raportteina. Kehitetty PoC-sovellus toimii yritykselle konseptin demonstraattorina ja luo vankan perustan tuotantovalmiin ratkaisun jatkokehitykselle.

Avainsanat	analyysi, avainsanat, chatbotit, ohjelmistokehitys, Python, tiedonhallinta
Muita tietoja	Työhön liittyy toimeksiantajalle toimitettu PoC-sovellus sisäisen chatin analysointiin ja sen käyttöönotto-ohjeistus.

LAPIN AMK
Lapland University of Applied Sciences

Abstract of thesis

Degree Programme in Business Information Technology

Bachelor of Business Administration (BBA)

Author	Jenny Puranen	**Year**	2025
Supervisor	Yrjö Koskenniemi		
Commissioned by	A medium-sized Finnish company		
Title	Development of an application for analyzing internal chat for the commissioner		
Number of pages	62 + 5		

The aim of this thesis was to develop a user-friendly and clear Proof of Concept (PoC) application to enhance the functionality of an internal chatbot for a mid-sized Finnish company. The company required a solution to analyze monthly chat logs to identify gaps in the chatbot's knowledge base and assess training needs for staff. The primary research question was: "How can an application be developed to analyze data collected from chatbot conversations to support business operations and improve the chatbot's capabilities?" Supporting questions explored the types of queries forwarded to human experts and how these queries could be analyzed to uncover recurring areas for improvement.

The theoretical framework examined challenges in internal communication, the role of chatbots in supporting communication, and methods for data analysis. A mixed-methods approach combining qualitative and quantitative techniques was employed. Qualitative interviews were conducted to understand the company's needs, while quantitative analysis utilized Python and the pandas library to process and analyze chat history. The application was implemented using the Flask web framework, and its browser-based user interface was designed to present the analysis results in an easily understandable format.

The outcome of this thesis was a PoC application that analyzes chatbot messages and categorizes them into service categories based on keywords. This tool helps the company identify key areas for development and prioritize improvement efforts more effectively. Additionally, it may support strategic decision-making by delivering analysis results in both visual formats and downloadable reports. As a concept demonstration, the PoC application provides a solid foundation for further development into a production-ready solution.

Keywords	analysis, chatbots, information management, keywords Python, software development
Special remarks	The thesis includes a PoC application for analyzing internal chat-logs and its deployment instructions submitted to the commissioner.

SISÄLLYS

1 JOHDANTO

Sisäinen viestintä on keskeinen osa organisaation toimintaa, sillä se varmistaa tiedon kulun ja yhteistyön eri yksiköiden ja työntekijöiden välillä. Tehokas sisäinen viestintä parantaa organisaation tehokkuutta sekä edistää työyhteisön sujuvaa toimintaa. (Tkalac Verčič 2021, 365, 367.) Toimiva sisäinen viestintä paitsi parantaa tiedonkulkua ja vähentää väärinkäsityksiä, myös lisää työntekijöiden sitoutumista ja työhyvinvointia (Luka 2024, 420–421). Heikko sisäinen viestintä voi puolestaan johtaa epäluottamukseen ja jopa heikentää organisaation toimivuutta (Tkalac Verčič 2021, 365). Tämän opinnäytetyön tarkoitus on kehittää helppokäyttöinen sovellus, joka mahdollistaa toimeksiantajan sisäisen chatin paremman ymmärryksen ja tehostaa liiketoiminnan kehittämistä parantamalla sisäistä kommunikaatiota sekä tukemalla päätöksentekoa datan analysoinnin ja raportoinnin avulla.

Digitalisaation myötä monet organisaatiot ovat alkaneet hyödyntää automaatiota ja tekoälypohjaisia ratkaisuja, kuten chatbotteja, tehostamaan viestintää ja tiedonhallintaa (Wang, Lin & Shao 2022, 1). Chatbotit pystyvät käsittelemään suuria määriä tietoa ja vastaamaan nopeasti yleisimpiin kysymyksiin, mikä paitsi vapauttaa henkilöstöresursseja myös nopeuttaa tiedon löytämistä ja parantaa palvelun tehokkuutta. Chatbottien kanssa käydyistä keskusteluista kerätty data voi paljastaa viestinnän ja tietopohjan puutteita, joiden täyttäminen voi tehostaa organisaation toimintaa ja reaktiokykyä. (Huseynov 2023, 56–58.)

Chatbottien toiminta perustuu ennalta määriteltyyn tietopohjaan, joka ohjaa niiden kykyä vastata kysymyksiin ja tuottaa relevantteja vastauksia (Hussain & Ginige 2019, 334). Tästä syystä chatbotit vaativat jatkuvaa seurantaa, jotta niiden tietopohja sekä vastausten laatu ja kattavuus pysyvät ajan tasalla. Ongelmat on hyvä tunnistaa hyvissä ajoin, jotta palvelua voidaan kehittää vastaamaan entistä paremmin työntekijöiden ja organisaation tarpeisiin.

Tämä opinnäytetyö tehdään toimeksiantajayritykselle, jolla on käytössä sisäinen chatbot, mutta ei järjestelmällistä tapaa analysoida chatbotilta asiantuntijalle eskaloituja keskusteluja. Toimeksiantajalla on tarve kehittää yrityksen sisäisen

chatbotin toimivuutta ja varmistaa, että se pystyy vastaamaan tehokkaasti työntekijöiden tarpeisiin. Tämä edellyttää chatbotin tietopohjan kattavuuden arviointia ja mahdollisten puutteiden tunnistamista, jotta yrityksen prosessien tehostaminen ja työn kulun optimointi voidaan varmistaa. Toimeksiantaja haluaa tunnistaa chatbotin kanssa käydyistä keskusteluista ne, jotka on ohjattu asiantuntijalle, ja järjestää ne kategorioittain. Lisäksi yritys haluaa tarkastella näitä tuloksia määrittelemillään aikaväleillä, jotta toistuvat trendit voidaan tunnistaa ja kehitystoimenpiteet suunnitella ja toteuttaa tehokkaasti. Toimeksiantaja toivoo myös, että sovelluksella voidaan luoda raportteja tuloksista myöhempää käyttöä varten.

Opinnäytetyön tavoitteena on luoda helppokäyttöinen ja selkeä Proof of Concept -sovellus, joka analysoi ja tuottaa hyödyllistä tietoa yrityksen sisäisen chatbotin chat-viesteistä. Sovelluksen on tarkoitus toimia konseptin demonstraattorina, josta yritys voi halutessaan jatkokehittää tuotantoon menevän version. Tutkimuskysymys on, kuinka kehitetään sovellus, jonka avulla voidaan analysoida chatbotin keskusteluista kerättyä dataa liiketoiminnan ja chatbotin kehittämisen tueksi. Tukikysymyksiä ovat, millaisia kysymyksiä chatbot ohjaa asiantuntijalle ja miten toistuvia kehityskohteita voidaan tunnistaa asiantuntijalle ohjatuista kysymyksistä.

Opinnäytetyöllä saavutettavat hyödyt voivat olla merkittäviä toimeksiantajalle, sillä sovelluksen avulla voidaan tunnistaa chatbotin tietopohjan puutteet sekä kartoittaa henkilöstön lisäkoulutustarpeet palvelukohtaisesti. Tämä mahdollistaa koulutuksen tarkemman kohdentamisen, mikä voi parantaa työntekijöiden tyytyväisyyttä ja tehokkuutta. Sovellus voi tarjota ajantasaisen analyysin, jonka avulla voidaan seurata chatbotin toimintaa ja kehitystoimenpiteiden vaikutuksia eri aikaväleillä. Organisaatio voi hyötyä tästä konkreettisesti, koska viestintäprosesseja voidaan tehostaa ja resurssien käyttöä optimoida. Lisäksi opinnäytetyön tuloksia voidaan hyödyntää laajemmissa kehitysprojekteissa, joissa keskitytään tekoälyratkaisujen ja digitalisaation edistämiseen organisaation strategisessa kehittämisessä. Digitalisaation tuomat innovaatiot voivat merkittävästi vahvistaa yritysten kykyä sopeutua ja menestyä muuttuvassa liiketoimintaympäristössä (Tsai ym. 2024, 2). Tästä syystä kehittämällä chatbotin toimintaa analysoivan sovelluksen yritys ei pelkästään paranna operatiivista tehokkuuttaan ja päätöksenteon laatua, vaan myös syventää ymmärrystään chatbottien tuottamasta datasta.

Tämä on tärkeää paitsi yksittäisen yrityksen kilpailukyvyn kannalta, myös laajemmin alueellisen kehityksen näkökulmasta, sillä yritysten toiminnan tehostuminen voi heijastua positiivisesti paikalliseen talouteen, työpaikkojen määrään ja osaamisen kehitykseen.

Työn alussa perehdytään aiheen tietopohjaan sekä chatbottien rooliin yrityksen sisäisessä viestinnässä ja päätöksentekoa tuottavan datan lähteenä. Tämän jälkeen käsitellään projektin suunnittelua ja sovelluksen toteutusta. Lopuksi esitellään toteutuksen tulokset, eli millainen sovellus projektin myötä syntyi, sekä arvioidaan lopputulosta eri näkökulmista.

2 ORGANISAATION VIESTINTÄ JA TIEDONHALLINTA

2.1 Sisäinen viestintä ja sen haasteet

Organisaation viestintä jakautuu ulkoiseen ja sisäiseen viestintään (Luka 2024, 421). Keytonin (2017, 506) mukaan viestintä organisaatiossa voi olla verbaalista, kuten puhuttua tai kirjoitettua ja nonverbaalista, kuten eleillä tai muilla visuaalisilla keinoilla välitettyä. Viestintä voi tapahtua suunnitellusti tai spontaanisti ja se voi olla tietoista tai tiedostamatonta (Keyton 2017, 506; Bond 2007, 219). Ulkoinen viestintä käsittää kaiken, mitä organisaatio viestii ulkopuolisille sidosryhmille, kuten asiakkaille, yhteistyökumppaneille, medialle ja muulle yleisölle (Wonneberger & Jacobs 2016, 368). Sisäinen viestintä puolestaan kattaa organisaation sisällä tapahtuvan kommunikaation kaikissa muodoissaan (Kalla 2005, 304). Sisäistä viestintää voidaan toteuttaa esimerkiksi kasvotusten, painettujen julkaisujen, sähköisten viestintäkanavien tai sosiaalisen median kautta (Tkalac Verčič & Špoljarić 2020, 101926).

Sisäinen viestintä on organisaation toiminnan kannalta keskeinen tekijä, joka vaikuttaa suoraan työyhteisön tehokkuuteen ja sujuvuuteen (Tkalac Verčič 2021, 365). Hyvin toimiva sisäinen viestintä varmistaa tiedonkulun ja edistää yhteistyötä eri osastojen ja yksilöiden välillä, mikä parantaa työn laatua ja vahvistaa organisaation kulttuuria (Sinčić Ćorić, Pološki Vokić & Tkalac Verčič 2020, 366–367). Tehokas sisäinen viestintä on keskeinen osa organisaation strategista suunnittelua ja sen puutteet voivat johtaa työntekijöiden sitoutumisen heikkenemiseen sekä korkeampaan henkilöstön vaihtuvuuteen (Hargie, Tourish & Wilson 2002, 4).

Viestinnän haasteet voivat vaikeuttaa organisaation päivittäistä toimintaa ja päätöksentekoa, mikä voi heikentää kokonaistehokkuutta (Sinčić Ćorić ym. 2020, 366). Sisäisen viestinnän yleisiä haasteita ovat esimerkiksi ympäristön häiriöt, kuten melu, viestintäkanavan valintaan liittyvät ongelmat sekä viestien sisällön epäselvyys. Lisäksi tärkeiden viestien riittämätön toisto (redundanssin puute) voi estää viestien tehokasta ymmärtämistä ja omaksumista organisaation sisällä. Viestintää voi hankaloittaa myös muille tuntemattoman ammattikielen käyttö.

Tämä voi johtaa esimerkiksi väärinkäsityksiin ja viivästyttää tiedonkulkua organisaatiossa. (Luka 2024, 420–422.) Sisäisen viestinnän ongelmien ratkaiseminen edellyttää selkeiden viestintäkäytäntöjen kehittämistä ja jatkuvaa parantamista, sillä viestinnän laatu vaikuttaa suoraan yhteistyön sujuvuuteen, työtyytyväisyyteen ja organisaation tavoitteiden saavuttamiseen (Pîrjol & Radomir 2017, 42–44).

2.2 Chatbotit viestinnän tukena

Chatbot on ohjelmisto, joka käyttää tekoälyä simuloidakseen keskustelua ihmisen kanssa (Miklosik, Evans & Qureshi 2021, 106530). Chatbotit voidaan luokitella monella eri tavalla, esimerkiksi käyttötarkoituksen, käyttöympäristön tai teknisen toteutuksen mukaan (Huseynov 2023, 57–61). Teknisen toteutuksen näkökulmasta chatbotit jaetaan yksinkertaisiin sääntöpohjaisiin botteihin sekä kehittyneempiin AI-pohjaisiin botteihin. Sääntöpohjaiset botit toimivat ennalta ohjelmoitujen sääntöjen mukaan, kun taas AI-pohjaiset chatbotit kehittyvät ja paranevat jatkuvien vuorovaikutusten kautta, jolloin ne pystyvät oppimaan ja mukautumaan käyttäjän tarpeisiin. (Miklosik ym. 2021, 106530.) Chatbotit voidaan jakaa sisäisiin, eli yrityksen työntekijöiden käytössä oleviin, sekä ulkoisiin, eli kaikkien saatavilla oleviin chatbotteihin (Huseynov 2023, 56).

Chatbotit ovat mullistaneet organisaatioiden viestintä- ja palveluprosesseja tarjoamalla automaattisen ja reaaliaikaisen ratkaisun niin sisäisiin kuin ulkoisiin vuorovaikutustarpeisiin (Huseynov 2023, 57). Ulkoisissa palveluissa, kuten esimerkiksi organisaation nettisivulla asiakaspalvelubottina, chatbotit tehostavat toimintaa pystymällä vastaamaan monelle asiakkaalle samanaikaisesti tuote- ja palvelukyselyihin ja tätä kautta vähentämään resurssikustannuksia (Adamopoulou & Moussiades 2020, 1). Viime vuosina chatbotteja on alettu hyödyntää yhä enemmän myös sisäisessä viestinnässä. Sisäisessä viestinnässä ne nopeuttavat pääsyä yrityksen tietopohjaan ja automatisoivat työtehtäviä. (Huseynov 2023, 56–58.)

Sergiienko (2024) kuvailee artikkelissaan, kuinka sisäiset chatbotit voivat merkittävästi vahvistaa modernien organisaatioiden viestintää ja tiedonkäsittelyä. Chat-

bottien avulla on mahdollista vähentää manuaalisten tehtävien määrää, nopeuttaa tiedonhankintaa ja parantaa viestinnän tehokkuutta, mikä tekee työskentelystä sujuvampaa ja tukee resurssien tehokasta käyttöä. Huseynov (2023, 57–58) puolestaan korostaa, että sisäisten chatbottien hyödyntäminen lisää organisaation ketteryyttä automatisoimalla rutiinitehtäviä, tarjoamalla työntekijöille ajantasaista tietoa ja vapauttamalla resursseja strategisiin ja monimutkaisempiin tehtäviin. Lisäksi chatbotit tehostavat viestintää, edistävät työntekijöiden sitoutumista ja parantavat tuottavuutta vähentämällä vasteaikoja ja virheitä.

Keskusteludatan tallentaminen ja analysointi on keskeinen osa chatbotin toimintaa, sillä se tarjoaa arvokasta tietoa organisaation toiminnan kehittämiseksi. Chat-keskusteluista muodostuu keskusteluloki, eli chat-loki, johon tallentuvat kaikki käyttäjän ja chatbotin tai asiakaspalvelijan välillä vaihdetut viestit. Chat-loki sisältää viestien sisällön, aikaleimat sekä käyttäjän ja chatbotin väliset vuorovaikutukset. (Khan & Das 2018, 78–79.) Chatbottien keräämä data tarjoaa organisaatioille mahdollisuuden analysoida viestinnän puutteita ja tunnistaa kehityskohteita, mikä parantaa käyttäjäkokemusta ja tukee yrityksen liiketoimintaa. Kerätty data auttaa reagoimaan kehitystarpeisiin nopeammin ja optimoimaan toimintaprosesseja, mikä vahvistaa yrityksen kilpailukykyä. (Huseynov 2023, 58.)

2.3 Datan analysointi

Data-analyysi on prosessi, jossa data järjestetään ja muokataan niin, että sitä voidaan hyödyntää menneiden tapahtumien selittämisessä ja tulevaisuuden ennustamisessa (Cuesta 2013, 7). Tavoitteena on luoda tietoa, joka tukee päätöksentekoa ja auttaa optimoimaan prosesseja organisaation kehittämiseksi (Semanjski 2023, luku 5.3). Data-analyysi voidaan jakaa kolmeen päävaiheeseen. Ensimmäisessä vaiheessa raakadataa kerätään ja tutkitaan ongelman ymmärtämiseksi. Toisessa vaiheessa analyysimenetelmiä kehitetään ja dataa puhdistetaan laadun varmistamiseksi. Kolmannessa eli viimeisessä vaiheessa analyysin tulokset kootaan ja esitetään raporttien ja visualisointien avulla päätöksenteon tueksi. (Stoudt, Vásquez & Martinez 2021, 1.) Datan analysoinnissa käytetään tyypillisesti erilaisia työkaluja ja ohjelmointikieliä, kuten esimerkiksi Pythonia tai R-kieltä (Mckinney 2010, 56). Data voidaan jakaa rakenteelliseen (structured) ja rakenteettomaan (unstructured) dataan. Rakenteellinen data, kuten taulukot ja

tietokannat, on järjestetty selkeästi määriteltyihin luokkiin ja hierarkioihin. Rakenteeton data sen sijaan sisältää esimerkiksi tekstiä, kuvia ja videoita. (Cuesta 2013, 10.)

Yleisesti data-analytiikan tyypit voidaan jakaa kolmeen pääryhmään: deskriptiiviseen, ennakoivaan ja preskriptiiviseen analyysiin. Deskriptiivinen analyysi pyrkii selvittämään mitä tapahtui ja miksi, joka auttaa organisaatioita tarkastelemaan menneitä tapahtumia ja ymmärtämään datan välistä yhteyttä. Ennakoiva analyysi puolestaan pyrkii ennustamaan tulevia tapahtumia ja niiden syitä. Preskriptiivinen analyysi tarjoaa suosituksia ja toimintasuunnitelmia päätöksentekoon selvittämällä mitä jatkossa tulisi tehdä ja miksi. (Lepenioti, Bousdekis, Apostolou & Mentzas 2020, 57.) Näin ollen data-analyysityypin valinta riippuu analysoitavan datan luonteesta ja analyysin tavoitteista.

Tässä opinnäytetyöprojektissa keskitytään chatbot-datan analysointiin, mikä tekee deskriptiivisestä analyysista sopivimman lähestymistavan. Deskriptiivinen analyysi hyödyntää erilaisia menetelmiä liiketoimintadatan käsittelyssä ja jäsentämisessä (Sharma & Bansal, 2018; Shmueli & Koppius, 2011).

Yksi keskeisistä menetelmistä on tekstianalyysi, jonka avulla voidaan saada hyödyllistä tietoa tekstimuotoisista aineistoista (Srinivasa-Desikan 2018, 10). Tämän menetelmän avulla voidaan tunnistaa avainsanoja, teemoja sekä analysoida kielen rakenteita ja tunteita (Arya 2020, 173). Ohjelmointikielistä erityisesti Python erottuu laajalla avoimen lähdekoodin ekosysteemillään, mikä tekee siitä erinomaisen valinnan tekstianalyysiin (Srinivasa-Desikan 2018, 14). Lisäksi Pythonin monipuoliset kirjastot, kuten pandas datan käsittelyyn, tekevät siitä tehokkaan ja joustavan työkalun data-analyysiin (McKinney 2022, luku 1.3).

Tekstimuotoisen aineiston analysoinnin lisäksi ajallisten piirteiden tarkastelu on tärkeää chatbot-datan ymmärtämisessä. Aikasarja-analyysi mahdollistaa ajallisesti jäsennellyn datan tarkastelun ja ilmiöiden kehityksen seurannan ajan kuluessa (Box, Jenkins, Reinsel & Ljung, 2016, 1). Aikasarja-analyysissä voidaan hyödyntää pandas-kirjaston keskeisiä ominaisuuksia, kuten DataFrame-rakennetta ja aikasarjojen käsittelyyn tarkoitettuja toimintoja. Näiden avulla pystytään tunnistamaan viestinnän trendit ja kausivaihtelut, mukaan lukien sesonkiluonteiset muutokset. (Cuesta 2013, 283.)

Datan visualisointi täydentää analyysiprosessia tekemällä monimutkaisesta tiedosta helpommin ymmärrettävää. Visualisoinnit tukevat organisaation päätöksentekoa tarjoamalla selkeitä ja havainnollisia esityksiä analyysin tuloksista, mikä edistää strategisten linjausten suunnittelua ja kehityskohteiden tunnistamista. (Cuesta 2013, 15.) Python tarjoaa tehokkaita työkaluja datan visualisointiin, joista yksi keskeisimmistä on Matplotlib-kirjasto (McKinney 2022, luku 9). Matplotlibin avulla voidaan luoda monenlaisia kaavioita, kuten pylväs-, piirakka- ja aikasarjakaavioita (Matplotlib Documentation 2024).

Yhdistämällä näitä analyysitekniikoita organisaatio saa syvempää ymmärrystä viestinnän kriittisistä ongelmakohdista ja kehittämistarpeista, mikä tukee tiedolla johtamista ja päätöksenteon tehostamista. Tärkeäksi työkaluksi tässä projektissa nousee Pythonin kirjastoista erityisesti pandasin DataFrame-rakenne ja GroupBy-toiminto, joiden avulla voidaan analysoida ja ryhmitellä dataa avainsanojen mukaan. Tämän klusterointimenetelmän avulla voidaan tehokkaasti ryhmitellä samankaltaisia viestejä, mikä helpottaa toistuvien teemojen ja rakenteiden tunnistamista (Srinivasa-Desikan 2018, 15). Visualisointiin käytetään Pythonin Matplotlib-kirjastoa, jonka avulla voidaan havainnollistaa analyysituloksia sovelluksen käyttöliittymässä aikasarjakaaviona ja tukea sitä kautta organisaation päätöksentekoa.

Työn toteuttamisessa on perehdytty tutkimuksiin tietopohjan aiheista sekä datan analysoinnista Python-ohjelmointikielellä. Päälähteenä työssä on käytetty McKinneyn kirjaa *Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter* (kolmas painos, selainversio) ja tukena on hyödynnetty Pythonin sekä sen kirjastojen dokumentaatiota.

3 SOVELLUKSEN TOTEUTUKSEN SUUNITTELU

3.1 Projektin rajaus

Opinnäytetyössä keskitytään Proof of Concept (PoC) -sovelluksen kehittämiseen, eikä tarkoituksena ole luoda tuotantovalmista versiota. Sovelluksen tavoitteena PoC:n mukaisesti on todistaa konseptin toimivuus ja sen keskeiset toiminnallisuudet, kuten analyysin tarkkuus ja käytettävyys. Sovellus toimii demonstraattorina siitä, miten chatbotin keräämää dataa voidaan hyödyntää organisaation kehityksessä, mutta se ei sisällä tuotantoversion monia vaatimuksia, kuten esimerkiksi käyttäjänhallintaa tai tietoturvaa.

Projektiin on rajattu vain PoC:n kannalta keskeisimmät toiminnot: datan lisäys, kategorian valinta, aikavälin valinta, datan analysointi, analyysin visualisoiminen sekä raportin lataaminen. Tämä tarkoittaa, että sovellus keskittyy ainoastaan olennaisiin toimintoihin, joita tarvitaan sen toimivuuden arvioimiseksi ja konseptin toteutettavuuden varmistamiseksi.

Analysoitava data rajoittuu vain niihin chatbot-keskusteluihin, jotka on ohjattu asiantuntijalle toimeksiantajan toiveiden mukaisesti, jotta voidaan tunnistaa tietopohjan puutteet ja kehityskohteet. Viestit kategorisoidaan ennalta määriteltyjen avainsanojen perusteella eri palvelukategorioihin, joista tässä PoC-vaiheessa käytetään nimiä palvelu A ja palvelu B. Tämä lähestymistapa mahdollistaa hyödyllisten analyysien ja raporttien tuottamisen rajatuista palveluista, mikä tukee tarkempaa analyysiä ja jatkokehitystä.

Käyttöliittymän osalta projektiin sisällytetään vain välttämättömät toiminnot, jotka tukevat sovelluksen ydintoimintoja ja tekevät sen käytöstä sujuvaa. Käyttöliittymän määrittely on käsitelty tarkemmin erillisessä luvussa, jossa kuvataan sille asetettuja vaatimuksia. Sovelluksen käyttökieleksi on valittu suomi, sillä suurin osa sen käyttäjistä ja analysoitavista viesteistä on suomenkielisiä. Tämän vuoksi myös analyysissä käytettävät avainsanat ovat suomeksi. Sovelluksen koodit ja tekniset termit on kuitenkin kirjoitettu englanniksi, jotta ne noudattavat kansainvälisiä ohjelmistokehityskäytäntöjä.

Projektista on rajattu ulkopuolelle toimeksiantajan suorittamat käytettävyystestaukset. Sovelluksen testaus toteutetaan kehitysprosessin aikana kehittäjän toimesta, sisältäen lopputestauksen. Laajempi testaus jää toimeksiantajan vastuulle, mikäli sovellus etenee jatkokehitykseen.

3.2 Tutkimus- ja kehittämismenetelmät

Opinnäytetyössä hyödynnetään sekä kvalitatiivisia että kvantitatiivisia tutkimusmenetelmiä, jotta sovelluksen suunnittelussa ja toteutuksessa voidaan tarkastella ongelmia monipuolisesti ja kattavasti. Kvalitatiivinen tutkimusmenetelmä on erityisen keskeinen projektin alkuvaiheessa, jossa kerätään tietoa sidosryhmien kokemuksista ja tarpeista. Haastattelut antavat arvokasta tietoa organisaation viestintäprosessien nykytilasta sekä kehitystarpeista, mikä ohjaa vaatimusten määrittelyä ja sovelluksen suunnittelua.

Kvantitatiivinen tutkimusmenetelmä puolestaan tulee keskeiseksi sovelluksen teknisessä toteutuksessa ja erityisesti chat-lokin datan analysoinnissa. Sovellus analysoi laajaa valmista kvalitatiivista dataa, eli chatbotin viestihistoriaa, kvantitatiivisilla menetelmillä, kuten tilastollisilla analyyseilla. Kvantifiointi, eli kvalitatiivisen aineiston muuttaminen kvantitatiiviseen muotoon, tarkoittaa esimerkiksi teemoihin liittyvien elementtien esiintymistiheyden laskemista (Saaranen-Kauppinen & Puusniekka 2006). Kvantifiointi mahdollistaa toistuvien ongelmien systemaattisen tunnistamisen, laskemisen ja analysoinnin. Kvantitatiivinen analyysi käyttää tilastollisia menetelmiä, jotka mahdollistavat viestien kvantitatiivisen luokittelun ja auttavat tunnistamaan merkityksellisiä trendejä sekä tekemään yleistettäviä johtopäätöksiä kehittämistarpeista (Taherdoost 2022, 57). Kvantitatiivisen analyysin tulokset auttavat kohdentamaan toimenpiteet tarkasti, mikä tehostaa organisaation viestinnän ja palvelujen kehittämistä.

Monimenetelmällinen lähestymistapa tukee sekä sovelluksen käyttäjäystävällisyyttä että sen teknistä toimivuutta. Yhdistämällä kvalitatiivista ja kvantitatiivista analyysia pystytään paitsi ymmärtämään käyttäjien kokemuksia, myös analysoimaan chatbotin teknistä suorituskykyä ja viestintäprosessien tehokkuutta. Monimenetelmällinen kehitystapa antaa laajemman ja uskottavamman ymmärryksen

tutkimusongelmasta (Guével & Absil 2023, 201). Monimenetelmällisen lähestymistavan etuna on sen kyky hyödyntää kummankin lähestymistavan vahvuuksia erityisesti monimutkaisissa tutkimuskonteksteissa (Taherdoost 2022, 55).

3.3 Vaatimusmäärittely

Vaatimusmäärittely kuvaa, mitä sovelluksen tulee tehdä ja millaisia ominaisuuksia sen odotetaan tarjoavan käyttäjille (Metatavu 2023). Sovelluksen vaatimusmäärittely perustuu organisaation esille tuomiin tarpeisiin, jotka selvitettiin tutkimusmenetelmien avulla. Kvalitatiivisissa haastatteluissa kartoitettiin sidosryhmien toiveita ja odotuksia sovelluksen suhteen, minkä perusteella määriteltiin sovelluksen käytettävyys- ja toiminnallisuusvaatimukset. Tavoitteena on tarjota ratkaisu, joka vastaa käyttäjien tarpeisiin selkeästi ja tehokkaasti.

Haastatteluista saatujen tulosten pohjalta sovelluksen käytettävyysvaatimuksiksi määriteltiin selkokielisyys ja helppokäyttöisyys. PoC-sovellus suunnitellaan siten, että kuka tahansa käyttäjä osaa käyttää sitä ilman erityistä teknistä osaamista. Käyttöliittymän tulee olla looginen, selkeä ja visuaalisesti miellyttävä. Sovelluksen komponentit sijoitetaan järkevästi, jotta navigointi on johdonmukaista ja käyttäjä löytää helposti tarvitsemansa toiminnot. Sovelluksen käyttökokemuksen tulee olla sujuva ja intuitiivinen, mikä lisää sen käytettävyyttä ja tehokkuutta.

Ei-toiminnalliset vaatimukset kuvaavat sovelluksen teknisiä ominaisuuksia, jotka tukevat sen laatua ja varmistavat käyttäjäkokemuksen sujuvuuden. Näihin vaatimuksiin valittiin toimeksiantajan tarpeiden perusteella muun muassa suorituskyky ja vakaus. Sovelluksen tulee reagoida nopeasti käyttäjän antamiin syötteisiin, jotta työskentely säilyy tehokkaana ja miellyttävänä. Lisäksi sovelluksen stabiilius pyritään takaamaan siten, että se palautuu mahdollisista virhetilanteista luotettavasti. Vaikka tämä ei ole PoC-vaiheen keskeisimpiä tavoitteita, stabiilius otetaan huomioon sovelluksen suunnittelussa.

Toiminnalliset vaatimukset kuvaavat sovelluksen konkreettisia toimintoja ja prosesseja. Ne määritettiin toimeksiantajan tarpeiden ja sidosryhmien toiveiden perusteella, jotta sovellus tukee tehokkaasti käyttäjien työn kulkuja ja analyysitarpeita. Sovelluksen tulee mahdollistaa toimeksiantajan kuukausittaisten Excel-tiedostojen lisääminen ja niiden sisältämän kvalitatiivisen aineiston analysointi

kvantifiointia hyödyntäen. Kvantitatiivinen analyysi mahdollistaa viestien avainsanojen tunnistamisen, mikä auttaa luokittelemaan viestit palvelukategorioihin. Käyttäjä voi tarkastella näitä tietoja halutulla aikavälillä ja tulostaa raportteja, esimerkiksi tietyn palvelun tai ajanjakson mukaan. Käyttäjä voi myös hakea tietoja hakusanojen avulla, mikä tehostaa tietojen etsimistä ja analysointia eri palveluiden osalta. Sovelluksen toiminnallisuuksien tavoitteena on antaa käyttäjälle joustava ja tehokas tapa tarkastella chatbotin toimintaa ja sen tuottamaa dataa eri näkökulmista sekä tuottaa helposti ymmärrettäviä raportteja analyysin tueksi.

3.4 Toiminnallinen määrittely

Toiminnallinen määrittely kuvaa sovelluksen oleellisimmat toiminnallisuudet ja niiden toteutustavat. Määrittelyn tavoitteena on antaa selkeä kuva sovelluksen keskeisistä toiminnoista ja käyttäjän vuorovaikutuksesta sovelluksen kanssa. Sovelluksen tärkeimmät toiminnallisuudet liittyvät datan lisäämiseen, analysointiin palvelukategorioittain sekä haun tulosten tarkasteluun määritetyllä aikavälillä ja avainsanojen perusteella.

PoC-sovelluksesta toteutetaan pelkistetty ja käyttäjäystävällinen web-sovellus, joka painottaa selkeyttä ja helppokäyttöisyyttä. Web-sovellus valittiin sen tarjoaman joustavuuden vuoksi, sillä se mahdollistaa käyttöliittymän tehokkaan suunnittelun ja toteutuksen. Lisäksi web-pohjainen ratkaisu on kevyt, helposti siirrettävissä eri ympäristöihin ja yhteensopiva monenlaisten järjestelmien kanssa.

Sovelluksen käyttöliittymä on suunniteltu niin, että sen perustoiminnot ovat helposti saavutettavissa ilman turhia elementtejä tai etsimistä. Navigointivalikon avulla käyttäjä voi helposti siirtyä etusivun analyysitoimintojen ja datanhallintasivun välillä ilman ylimääräisiä vaiheita.

Datanhallintasivulla käyttäjä voi lisätä kuukausittaiset chat-lokit Excel-tiedostoina latauspainikkeen kautta. Kun käyttäjä lisää tiedoston, sovellus käsittelee ja analysoi viestit automaattisesti Python-skriptin, eli ennalta määritetyn komentosarjan, avulla. Prosessissa tunnistetaan avainsanat, luokitellaan viestit ennalta määriteltyihin palvelukategorioihin ja tallennetaan analyysin tulokset tietokantaan (kuvio 1). Latauksen jälkeen käyttäjä näkee datanhallintasivun taulukosta onnistuneesti ladatun tiedoston tai virhetilanteessa ilmoituksen epäonnistuneesta lisäyksestä.

Kuvio 1. Datan lisäyksen ja käsittelyn yleiskuva

Sovelluksen keskeisin toiminnallisuus, analyysi, on käytettävissä sovelluksen etusivulla. Kun data on tallennettu tietokantaan, käyttäjä voi siirtyä etusivulle suorittamaan hakuja. Hakutoiminnossa käyttäjä voi valita alasvetovalikoista halutut palvelukategoriat, avainsanan sekä aikavälin, johon sisältyvät alkamis- ja päättymiskuukaudet sekä -vuodet. Kun valinnat on tehty, analyysi käynnistetään analysointipainikkeella.

Sovellus luo käyttäjälle automaattisesti yhteenvedon haun tuloksista, jossa analysoidut viestit on jaoteltu valitun ajanjakson ja palvelukategorioiden mukaan. Tämän jälkeen käyttäjä voi tallentaa haun tulokset raporttina napin painalluksesta omalle tietokoneelleen. Sovelluksen käyttöprosessi on suunniteltu loogiseksi ja vaiheittaiseksi, mikä minimoi virheiden ja tietojen häviämisen riskin.

4 MENETELMÄLLINEN TOTEUTUS

Sovellus kehitettiin tiiviissä yhteistyössä toimeksiantajan kanssa hyödyntäen ketteriä menetelmiä, mikä mahdollisti joustavan ja iteratiivisen kehitysprosessin. Jatkuva yhteistyö varmisti, että työ eteni toimeksiantajan tarpeiden ja odotusten mukaisesti. Palavereja pidettiin yhteensä 6 kappaletta.

4.1 Työkalut ja teknologiat

Sovellus kehitettiin hyödyntäen web-pohjaista Full Stack -ratkaisua, jossa Python-ohjelmointikieltä käytettiin sen monipuolisuuden ja tehokkaiden datankäsittely- ja analysointikirjastojen vuoksi. Esimerkiksi pandas-kirjasto mahdollistaa Excel-tiedostojen helpon lukemisen ja käsittelyn (McKinney 2022, luku 1.3), mikä on olennaista projektin vaatimusten kannalta. Lisäksi Pythonin yksinkertainen syntaksi, eli koodikielen rakenteelliset säännöt, nopeuttaa kehitysprosessia ja tekee koodista helposti luettavaa ja ylläpidettävää.

Sovelluksen backend, eli sen taustatoiminnot, toteutettiin Flaskilla, joka on kevyt ja joustava Python-pohjainen web-kehys. Flask tarjoaa yksinkertaisen ja tehokkaan alustan verkkosovellusten kehittämiseen sekä valmiita työkaluja ja rakenteita, jotka helpottavat sovelluksen toteutusta (Flask Documentation 2024c). Se mahdollistaa työkalut tietokannan ja käyttöliittymän yhdistämiseen, reittien määrittämiseen ja sovelluksen liikenteen hallintaan (GeeksforGeeks 2024a). Lisäksi Flask käyttää Jinja-mallinnusmoottoria, joka mahdollistaa dynaamisten ja turvallisten mallipohjien luomisen sekä tietokannasta haetun datan esittämisen käyttäjäystävällisessä muodossa (Flask Documentation 2024b). Tämä mahdollistaa sovelluksen sisällön mukautumisen käyttäjän toimien ja haettujen tietojen perusteella esimerkiksi näyttämällä kysytyimmät kysymykset suoraan tietokannasta ilman sivuston uudelleenrakentamista.

Tiedon hallintaan käytettiin MySQL-tietokantaa, joka mahdollistaa rakenteellisen datan tallentamisen ja tehokkaat kyselyt analyysia varten. MySQL valittiin sen luotettavuuden ja skaalautuvuuden vuoksi ja se toimi myös sovelluksen suodatus- ja analyysitoimintojen taustalla. MySQL on luotettava ja laajalti käytetty tie-

tokantaratkaisu, joka skaalautuu hyvin suurempiinkin tietomääriin (GeeksforGeeks 2024b). Käyttöliittymä puolestaan rakennettiin HTML- ja CSS-teknologioilla, jotta sovellus olisi selkeä ja käyttäjäystävällinen. HTML määrittelee verkkosivun rakenteen ja sisällön hierarkian, kun taas CSS vastaa sen visuaalisesta ilmeestä, kuten väreistä, fonteista ja asettelusta.

Ohjelmointiympäristönä käytettiin Visual Studio Code -editoria, joka tarjoaa monipuoliset työkalut Python-kehitykseen. Lisäksi se tukee hyvin Flask-sovelluskehystä ja MySQL-tietokantapalvelimen käyttöä laajennusten avulla. Pythonin venv-virtualisointiympäristöä käytettiin eristämään sovelluksen riippuvuudet, mikä helpotti niiden hallintaa ja varmisti kehitysympäristön vakauden. Sovelluksen kehityksessä käytettiin XAMPP:n tarjoamaa paikallista MySQL-palvelinta, joka mahdollisti tietokannan hallinnan ja tietojen testaamisen kehitysympäristössä. Flask-kehityspalvelimella sovellusta voitiin testata reaaliaikaisesti selaimessa, mikä nopeutti kehitysprosessia ja tarjosi välittömän palautteen tehdyistä muutoksista.

Kehitysprosessissa käytettiin Git-versionhallintajärjestelmää, joka mahdollistaa koodin helpon versioinnin. Versionhallinnan avulla varmistettiin, että projektin kaikki kehitysvaiheet ja muutokset voidaan dokumentoida ja palata tarvittaessa vanhoihin versioihin. Versionhallinta tekee kehitysprosessista hallitun ja vähentää virheiden syntymistä (Git Documentation 2024). Kehittämisessä hyödynnetyt työkalut ja teknologiat, eli projektin teknologiapino (kuvio 2) valittiin niiden soveltuvuuden, skaalautuvuuden ja helppokäyttöisyyden perusteella.

Kuvio 2. Sovelluksen teknologiapino

PoC-projektissa on tärkeää, että teknologiat mahdollistavat nopean toteutuksen, testaamisen ja iteroinnin, mikä teki Flaskin, Pythonin ja MySQL:n kaltaisista kevyistä ja joustavista ratkaisuista ihanteellisia. Ne tarjoavat riittävän toiminnallisuuden järjestelmän peruskonseptin todentamiseen, mutta ovat myös helposti laajennettavissa mahdollisessa sovelluksen jatkokehityksessä.

4.2 Datan kerääminen ja testidatan luonti

Analysoitava data tuodaan sovellukseen kuukausittaisista Excel-tiedostoista. Alun perin tavoitteena oli analysoida suurempaa määrää todellista dataa useilta kuukausilta, mutta toimeksiantajalta saatiin lopulta käyttöön vain yhden kuukauden tiedot. Tämä olisi rajoittanut analyysin kattavuutta ja estänyt palvelukategorioiden kysymysten esiintyvyyden tarkastelun pidemmällä aikavälillä. Tämän vuoksi päätettiin luoda testidataa alkuperäisen aineiston pohjalta, joka mahdollisti laajemman analyysin ja sovelluksen eri toimintojen kattavan testauksen.

Testidataa generoitiin Python-skriptillä, jonka avulla simuloitiin viestejä eri kuukausille vuosien 2021–2023 väliltä. Generoitu data pohjautui todellisiin rakenteisiin, mikä tarkoittaa, että jokaisessa tiedostossa on samat sarakeotsikot, kuten message ja sender, kuin alkuperäisessä aineistossa. Tiedostojen nimeämisessä noudatettiin yhtenäistä käytäntöä, sillä toimeksiantajan alkuperäisissä tiedostoissa vuosi ja kuukausi olivat kätevästi mukana tiedoston nimessä. Tämä mahdollisti tiedostojen tehokkaan järjestämisen ja analyysin ajallisten tarkastelujen suorittamisen. Tässä raportissa esitetyt esimerkit on muutettu geneerisemmiksi, jotta toimeksiantajan yksityiskohtaiset tiedot eivät ole tunnistettavissa. Luodut Excel-tiedostot sisälsivät yhteensä 50 kappaletta sekä chatbotin että käyttäjän viestejä.

Botin viestit toimivat keskustelujen aloituspisteinä ja määrittävät, mitkä viestit otetaan mukaan analyysiin. Esimerkiksi viesti "Aktivointiviesti" toimii merkkinä, jonka jälkeen seuraava käyttäjän lähettämä viesti tallennetaan tietokantaan ja sisällytetään analyysiin. Tämä on havainnollistettu taulukossa 1, jossa esitetään, kuinka chatbotin (Bot) viestit ja niitä seuraavat käyttäjän (Human) viestit järjestyvät tiedostossa. Käyttäjän viestit puolestaan generoitiin satunnaisesti valittujen teemojen ja avainsanojen pohjalta. Avainsanat lisättiin osaksi viestien sisältöä ja niiden

käyttö määriteltiin tarkasti testidataa generoivassa Excel-tiedostojen luontiskriptissä. Tämä mahdollisti sovelluksen toiminnallisuuksien testauksen realistisilla esimerkeillä.

Taulukko 1. Esimerkki testidatan rakenteesta Excel-tiedostossa

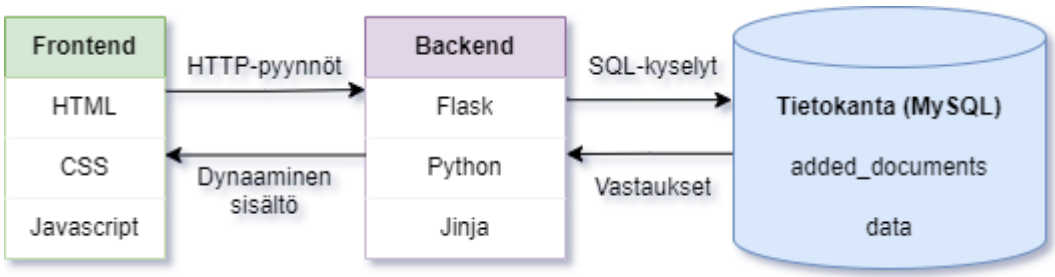
message	sender
Aktivointiviesti	Bot
Moi! Tiedätkö mistä voisi löytyä	Human
Aktivointiviesti	Bot
Moikka! Mietin tuota	Human
Aktivointiviesti	Bot
Hei! Etsin tietoja	Human

Testidatan rakenteen suunnittelussa säilytettiin yhteensopivuus alkuperäisen aineiston kanssa, jotta datan esikäsittelyvaiheet voitiin suorittaa yhtenäisesti sovelluksessa. Tarkemmat tiedot projektin aineistonhallinnasta, kuten säilytys-, käsittely- ja poistokäytännöistä, on esitetty liitteessä 1.

4.3 Sovelluksen arkkitehtuuri

4.3.1 Yleiskuva arkkitehtuurista

Sovellus toteutettiin kolmitasoisena arkkitehtuurina, jossa frontend, backend ja tietokanta muodostavat sovelluksen keskeiset osat (kuvio 3). Käyttäjä kommunikoi sovelluksen frontendin eli käyttöliittymän kanssa, joka on rakennettu HTML:n, CSS:n ja JavaScriptin avulla. Käyttöliittymä lähettää käyttäjän toiminnan perusteella HTTP-pyyntöjä sovelluksen backendille.



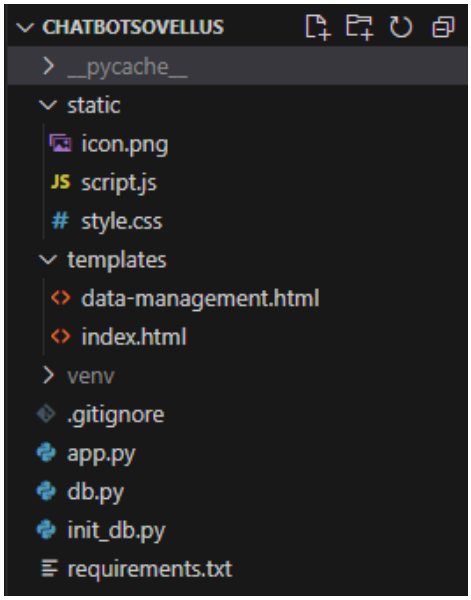
Kuvio 3. Sovelluksen arkkitehtuurin yleiskuva

Backend vastaa sovelluksen toiminnallisista ydinprosesseista, kuten käyttäjäpyyntöjen käsittelystä, SQL-tietokantakyselyjen suorittamisesta ja tiedonhallinnasta. Pythonilla toteutettu Flask-kehys ohjaa käyttäjän tekemät pyynnöt oikeisiin toimintoihin reittien avulla. Reititys yhdistää käyttäjän toimet backendin toiminnallisuuksiin ja palauttaa tulokset käyttöliittymään dynaamisena HTML-sivuna Jinja-mallinnuksen avulla. Reitit on jaettu analyysireitteihin ja datanhallintareitteihin, joiden toiminnallisuus käsitellään tarkemmin luvuissa 4.4 ja 4.5.

Tietokanta on sovelluksen ydin, ja se toimii analysoitavan datan tallennuspaikkana. Tietokanta on toteutettu MySQL-teknologialla, joka mahdollistaa rakenteellisen datan tehokkaan hallinnan ja kyselyjen suorittamisen. Analysoitavat viestit ja tiedostojen metatiedot tallennetaan tietokantaan, mikä varmistaa sujuvan integraation sovelluksen muiden osien kanssa.

4.3.2 Sovelluksen hakemistorakenne

Sovelluksen hakemistorakenne (kuvio 4) on suunniteltu selkeäksi ja modulaariseksi noudattaen Flask-sovelluksille tyypillistä rakennemallia, mikä tukee sovelluksen helppoa ylläpitoa, jatkokehitystä ja skaalautuvuutta. Sovelluksen juurihakemisto sisältää kaikki keskeiset komponentit, kuten backendin ja frontendin toteutukseen tarvittavat tiedostot, staattiset resurssit sekä sovelluksen suorittamiseen liittyvät ympäristöasetukset.



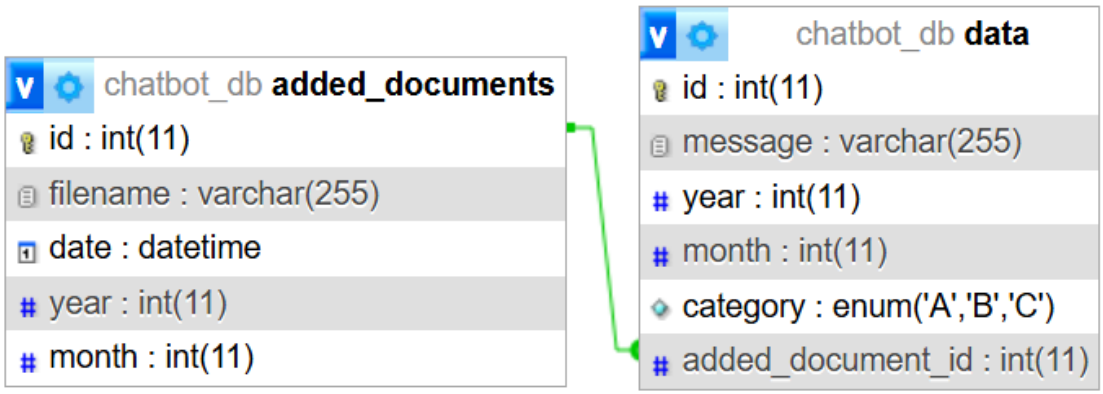
Kuvio 4. Sovelluksen hakemistorakenne

Sovelluksen staattiset resurssit, kuten tyylitiedosto (CSS), JavaScript-tiedosto ja ikoni, joita käyttöliittymä hyödyntää, on tallennettu *static*-kansioon. Näitä tiedostoja käytetään käyttöliittymän ulkoasun ja toiminnallisuuksien luomisessa. Esimerkiksi *style.css* määrittää sovelluksen visuaalisen ulkoasun ja *script.js* sisältää käyttöliittymän toiminnallisuuksia, kuten ponnahdusikkunoiden toiminnallisuudet. T*emplates*-kansio puolestaan sisältää sovelluksen HTML-mallipohjat, jotka muodostavat käyttöliittymän eri näkymät. Sovelluksen pääsivusta eli analyysisivusta vastaa *index.html*, jossa käyttäjä voi suorittaa analyysejä. *Data-management.html* tarjoaa näkymän datanhallintaan liittyviin toimintoihin, kuten tiedostojen lisäykseen ja poistoon.

Tietokantayhteyksien hallinta on keskitetty tiedostoon *db.py*, joka sisältää eriytetyn funktion tietokantayhteyden luomiseksi. Tiedosto *init_db.py* vastaa tietokantataulujen luomisesta ja alustamisesta, mukaan lukien taulujen väliset relaatiosäännöt. Sovelluksen logiikka ja reititykset on keskitetty tiedostoon *app.py*, joka toimii sovelluksen keskeisenä ohjauskeskuksena ja yhdistää tietokannan käyttöliittymän toimintoihin. Sovelluksen riippuvuudet on listattu tiedostossa *requirements.txt*, mikä mahdollistaa kehitysympäristön nopean käyttöönoton. *Gitignore*-tiedosto puolestaan sisältää määrittelyt tiedostoille ja kansioille, joita ei sisällytetä versionhallintaan. Tämä hakemistorakenne varmistaa, että sovelluksen eri osat ovat helposti löydettävissä ja muokattavissa. Esimerkiksi jos käyttöliittymän ulkonäköä halutaan päivittää, muutokset voidaan tehdä *static*- ja *templates*-kansioiden tiedostoihin ilman, että backend-logiikkaan tarvitsee koskea.

4.3.3 Tietokantarakenne

Sovelluksen tietokantarakenne perustuu kahteen keskeiseen tauluun, *added_documents* ja *data* (kuvio 5), jotka on suunniteltu tallentamaan analysoitavan datan metatiedot sekä itse analyysissä käytettävät viestit. Tietokanta on toteutettu MySQL-tietokantajärjestelmällä, joka soveltuu hyvin rakenteellisen datan tallentamiseen ja käsittelyyn. Sen avulla sovelluksessa pystytään suorittamaan tarkkoja kyselyjä ja varmistamaan, että analyysi perustuu luotettavaan dataan.



Kuvio 5. Sovelluksen tietokantarakenne

Added_documents-taulu toimii kaikkien sovellukseen lisättyjen Excel-tiedostojen metatietojen säilytyspaikkana. Jokainen taulun rivi vastaa yksittäistä tiedostoa ja sisältää tiedoston nimen, lisäyspäivämäärän sekä tiedoston nimestä haettavan vuoden ja kuukauden. Näiden tietojen avulla sovellus voi hallita tiedostoja tehokkaasti esimerkiksi mahdollistamalla niiden järjestämisen aikajärjestykseen tai poistamisen. Chat-lokitiedostojen ajankohdan tallentaminen tiedoston nimestä on olennaista analyysitoimintojen kannalta, sillä se mahdollistaa tietyn ajanjakson tarkastelun.

Toinen tärkeä taulu, *data*, sisältää analyysiin sisällytettävät viestit, jotka on käsitelty ja tallennettu Excel-tiedostoista. Jokainen rivi edustaa yksittäistä viestiä ja sisältää kaikki viestiin liittyvät tiedot, kuten viestin sisällön, vuoden ja kuukauden. Taulussa on myös sarake, joka määrittää viestin palvelukategorian, mikä helpottaa analyysia ja viestien ryhmittelyä. Taulu käyttää viiteavaimena (foreign key) *added_document_id*-kenttää, joka viittaa *added_documents*-taulun tiettyyn tiedostoon. *ON DELETE CASCADE* -määritys varmistaa, että tiedoston poistamisen yhteydessä myös siihen liittyvät viestit poistuvat *data*-taulusta, mikä ylläpitää tietokannan eheyttä. Taulujen välinen suhde on toteutettu yksi-moneen-periaatteella, jossa yksi tiedosto voi sisältää useita viestejä.

Tämä rakenne mahdollistaa datan tehokkaan hallinnan ja analysoinnin. Esimerkiksi analyysin aikana sovellus voi hakea vain tietyn tiedoston viestit rajaamalla haun valittuun ajanjaksoon. Tietokantarakenne muodostaa sovelluksen ytimen ja tukee sen keskeisiä toimintoja, kuten datan hallintaa, esikäsittelyä, analysointia sekä tulosten esittämistä.

4.4 Datanhallintasivu: tiedoston hallinta ja prosessointi

Datanhallintasivu on sovelluksen keskeinen työkalu, jonka avulla käyttäjä voi lisätä analyysiin tarvittavaa dataa sekä hallita jo lisättyjä tiedostoja. Tämä sivu tarjoaa työkalut Excel-tiedostojen lisäämiseen, niiden sisällön käsittelyyn ja tietojen tallentamiseen tietokantaan, mikä on välttämätöntä sovelluksen analyysitoimintojen hyödyntämiselle.

4.4.1 Reititys ja tietojen esittäminen käyttöliittymässä

Sovelluksen toiminnallisuudet rakentuvat Flask-kehyksen tarjoaman reititystoiminnallisuuden varaan. Reititys määrittelee, mitä toimintoja sovellus suorittaa käyttäjän vieraillessa tietyssä osoitteessa (Flask Documentation 2024a). Tämä voi sisältää toimintoja, kuten analyysisivun tarkastelun tai tiedoston lisäämisen. Esimerkiksi reitti */data-management* (kuvio 6) ohjaa käyttäjän *data-management.html*-sivulle, joka toimii datanhallinnan käyttöliittymänä.

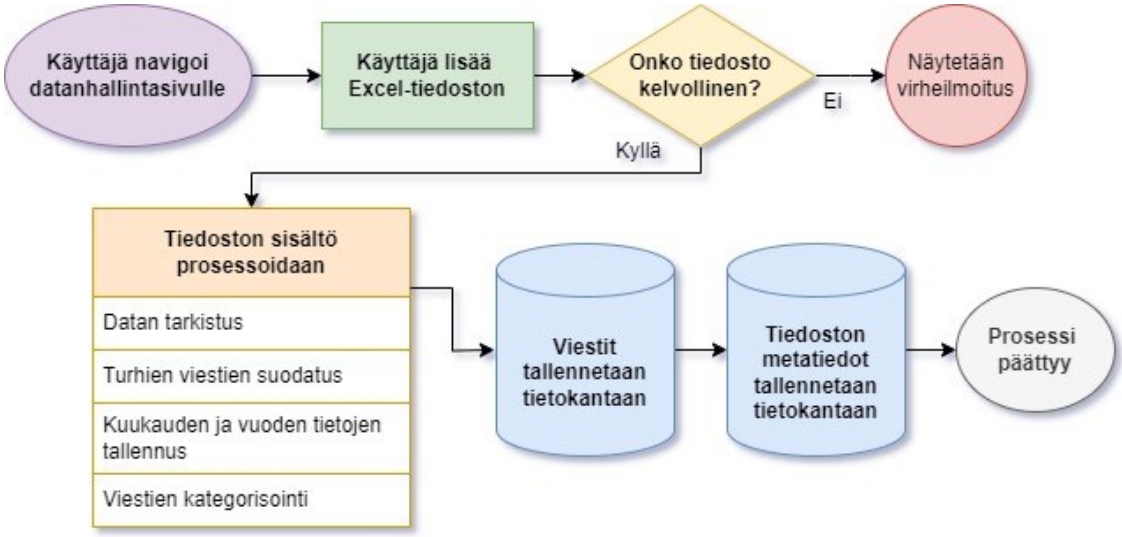
```
@app.route('/data-management')
def data_management():
    connection = create_db_connection()
    cursor = connection.cursor()
    cursor.execute("SELECT id, filename, date FROM added_documents")
    added_documents = cursor.fetchall()
    cursor.close()
    connection.close()
    return render_template('data-management.html', added_documents=added_documents)
```

Kuvio 6. Datanhallintareitti

Reitissä käytetään tietokantayhteyttä *create_db_connection*-funktion avulla, minkä jälkeen SQL-kysely hakee lisättyjen tiedostojen metatiedot eli nimen ja lisäyspäivämäärän tietokannasta. Nämä tiedot esitetään datanhallintasivulla taulukkomuodossa, mikä tarjoaa käyttäjälle selkeän yleiskuvan hallittavasta datasta. Flaskin *render_template*-funktio hyödyntää Jinja-mallinnusta yhdistämällä tiedot *data-management.html*-sivupohjaan, mikä mahdollistaa tietojen dynaamisen esittämisen käyttöliittymän taulukossa.

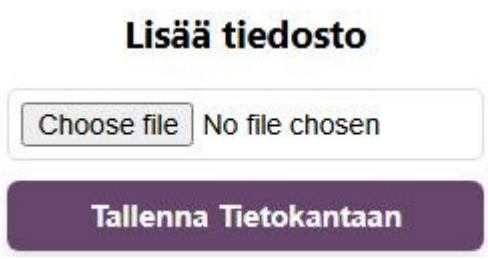
4.4.2 Tiedostojen lisääminen ja prosessointi

Tiedostojen hallinnan ja analyysin tehokkuus perustuu suurelta osin niiden lisäys- ja käsittelyprosessien sujuvuuteen. Kuviossa 7 esitetään prosessi, jossa käyttäjä lataa Excel-tiedoston datanhallintasivulle, minkä jälkeen sovellus hoitaa tiedoston käsittelyn automaattisesti. Tämä prosessi on keskeinen analyysitulosten luotettavuuden ja tarkkuuden kannalta, sillä se varmistaa, että analysoitavaksi päätyy ainoastaan olennaista ja virheetöntä dataa.



Kuvio 7. Tiedoston lisäys- ja käsittelyprosessin vuokaavio

Tiedoston lisääminen tapahtuu datanhallintasivulla, jossa käyttäjä valitsee lisättävän Excel-tiedoston ja lähettää sen palvelimelle HTML-lomakkeen avulla. Lomake sisältää *Choose file* -painikkeen tiedoston valitsemista varten ja *Tallenna Tietokantaan* -painikkeen käsittelyn aloittamiseksi (kuvio 8). Frontendin *script.js* varmistaa, että tiedosto on valittu ennen lomakkeen lähettämistä, ja ongelmatilanteissa se näyttää ponnahdusikkunan, joka ilmoittaa tiedoston lisäyksen epäonnistumisesta.



Kuvio 8. Tiedoston lisääminen datanhallintasivulla

Backendin puolella tiedosto käsitellään */upload*-reitillä, jossa prosessi rakentuu keskeisesti *process_excel*-funktion ympärille. *Process_excel*-funktio löytyy kokonaisuudessaan liitteessä 2. Tämä funktio vastaa tiedoston käsittelyn logiikasta ja sen toiminta voidaan jakaa seuraavaksi käsiteltäviin vaiheisiin.

Ensimmäisessä vaiheessa hyödynnetään *extract_year_and_month_from_filename*-funktiota, jonka avulla tiedostonimestä haetaan analyysin kannalta tärkeät ajalliset tiedot eli vuosi ja kuukausi. Nämä tiedot tallennetaan sekä tiedoston metatietoihin että viestien tietokantaan, ja niitä hyödynnetään aikaväliin perustuvissa analyyseissä. Tämän jälkeen tiedosto muunnetaan Pandasin DataFrame-rakenteeksi (kuvio 9). DataFrame-rakenne mahdollistaa datan tehokkaan tarkastelun ja käsittelyn, sillä se tarjoaa monipuoliset työkalut tietojen järjestämiseen riveittäin ja sarakkeittain (McKinney 2022, luku 5.1). Samalla tarkistetaan, että tiedosto sisältää analyysin kannalta välttämättömät sarakkeet, kuten *message* ja *sender*. Jos sarakkeita puuttuu, prosessi keskeytetään ja käyttäjälle ilmoitetaan virheestä ponnahdusikkunalla. Jos kaikki vaaditut sarakkeet löytyvät, DataFrame-rakenteesta poistetaan tämän jälkeen rivit, joissa nämä tärkeät kentät ovat tyhjiä, jotta analyysiin käytetään vain oleellista dataa.

```
df = pd.read_excel(file)
if 'message' not in df.columns or 'sender' not in df.columns:
    return "Excel-tiedostossa ei ole 'message' tai 'sender' -sarakkeita."
df = df.dropna(subset=['message', 'sender'])
```

Kuvio 9. Excel-tiedoston validointi ja muuntaminen DataFrame-rakenteeseen

Seuraavaksi käsitellään tiedoston sisältämät viestit yksitellen. Jokaisen viestin kohdalla tarkistetaan, onko se tarkoituksenmukainen analyysiin (ei sisällytetä esim. liian lyhyitä tai pelkkiä numeroita sisältäviä viestejä) ja määritetään sen palvelukategoria käyttäen *categorize_message*-funktiota (kuvio 10). Tämä funktio luokittelee viestit avainsanojen perusteella kolmeen pääkategoriaan: A, B tai C (muut).

```
def categorize_message(message):
    message = message.lower()
    if len(message) < 10:
        return None
    if message.isdigit():
        return None
    if message in IGNORED_MESSAGES:
        return None
    if any(keyword in message for keyword in CATEGORY_A_KEYWORDS):
        return 'A'
    elif any(keyword in message for keyword in CATEGORY_B_KEYWORDS):
        return 'B'
    else:
        return 'C'
```

Kuvio 10. Viestien kategorisointifunktio

Avainsanojen rooli prosessissa on keskeinen, sillä niiden avulla sovellus tunnistaa viestien sisällön ja ohjaa analyysin kohdentumista oikeisiin palvelukategorioihin. Sovelluksessa määritellyt avainsanalistat (*CATEGORY_A_KEYWORDS* ja *CATEGORY_B_KEYWORDS*), jotka toimeksiantaja on laatinut analyysin tarpeita varten, toimivat viestien luokittelun perustana. Viestit, jotka eivät sisällä mitään määriteltyjä avainsanoja, ohjataan automaattisesti yleiseen kategoriaan C, mikä mahdollistaa myös poikkeavien viestien käsittelyn järjestelmällisesti. Lisäksi *IGNORED_MESSAGES*-listan avulla analyysistä poistetaan sellaiset viestit, jotka eivät ole analyysin kannalta merkityksellisiä. Näin varmistetaan, että analyysiin päätyy vain oleellista ja käsiteltävissä olevaa dataa. Kuviossa 11 on esitetty, kuinka analyysiin hyväksytyt viestit ja niiden metatiedot, kuten uusi palvelukategoria, vuosi ja kuukausi, tallennetaan tietokannan tauluun *data*.

```
cursor.execute("""
    INSERT INTO data (year, month, category, added_document_id, message)
    VALUES (%s, %s, %s, %s, %s)
""", (year, month, category, added_document_id, message))
```

Kuvio 11. Analysoidun datan tallentaminen tietokantaan

Kun analyysiin hyväksytyt viestit on tallennettu tietokantaan, sovellus on valmis hyödyntämään näitä tietoja muissa toiminnallisuuksissa. Lisätty tiedosto tallentuu tietokannan *added_documents*-tauluun ja käyttäjälle ilmoitetaan onnistuneesta tiedoston lisäyksestä ponnahdusikkunan avulla.

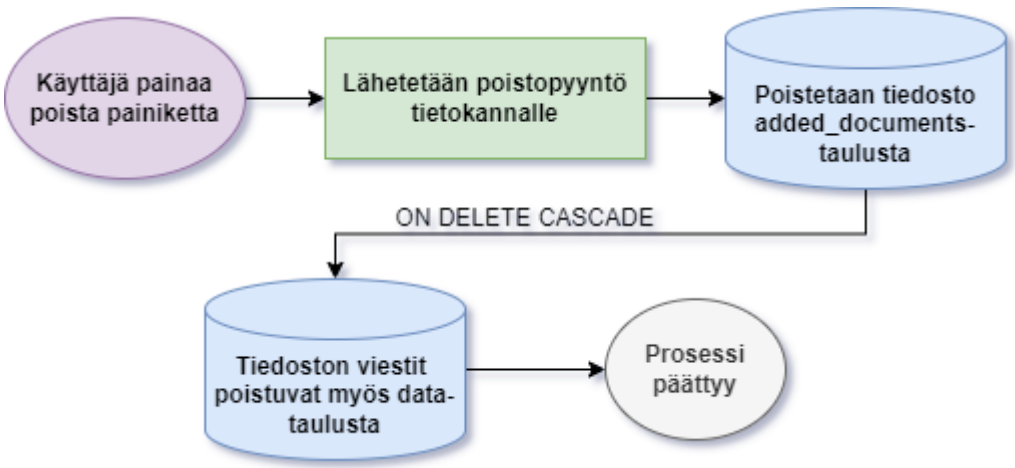
4.4.3 Tiedostojen poistaminen

Lisätyt tiedostot tallentuvat tietokannan *added_documents*-tauluun, josta niiden tiedot haetaan datanhallintasivun taulukkoon (kuvio 12) automaattisesti. Taulukossa esitetään tiedoston nimi, lisäysajankohta sekä *Poista*-painike, joka mahdollistaa tiedoston helpon poistamisen.

Tiedostonimi	Lisätty	Poista tiedosto
keskustelut helmikuu 2022.xlsx	2024-11-13 14:37:50	Poista
keskustelut huhtikuu 2022.xlsx	2024-11-13 14:37:55	Poista
keskustelut kesäkuu 2021.xlsx	2024-11-13 14:37:59	Poista

Kuvio 12. Datanhallintasivun taulukkonäkymä

Kuviossa 13 esitetään tiedoston poistoprosessin vaiheet. Prosessi käynnistyy, kun käyttäjä painaa *Poista*-painiketta, minkä seurauksena tietokantaan lähetetään poistopyyntö. Tämä poistaa tiedoston *added_documents*-taulusta. Tiedoston poistaminen *added_documents*-taulusta poistaa automaattisesti myös siihen liittyvät viestit *data*-taulusta *ON DELETE CASCADE* -ominaisuuden avulla. *ON DELETE CASCADE* -mekanismi varmistaa, että tietokantaan ei jää viittauksia poistettuihin tiedostoihin, mikä ylläpitää tietokannan eheyttä (GeeksforGeeks 2021).



Kuvio 13. Tiedoston poistoprosessi

Kun käyttäjä painaa *Poista*-painiketta, sovellus käyttää JavaScript-funktiota *confirmDelete()* varmistusponnahdusikkunan avaamiseen ennen poistamista. Vahvistuksen jälkeen poistotoiminto suoritetaan */delete-document*-reitiltä. Reitti (kuvio 14) vastaanottaa poistettavan tiedoston yksilöivän tunnisteen (*id*), jonka perusteella poistoprosessi käynnistyy.

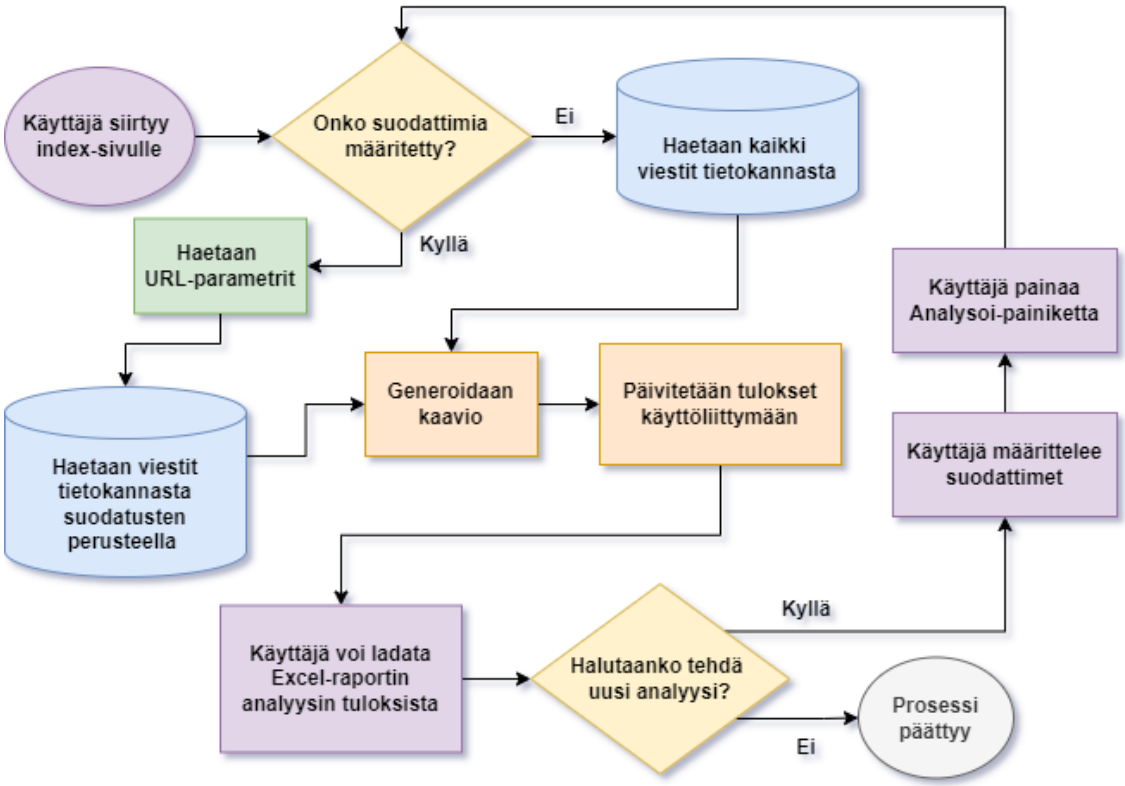
```
@app.route('/delete-document/<int:document_id>', methods=['POST'])
def delete_document(document_id):
    connection = create_db_connection()
    cursor = connection.cursor()
    cursor.execute("DELETE FROM added_documents WHERE id = %s", (document_id,))
    connection.commit()
    cursor.close()
    connection.close()
    return redirect(url_for('data_management'))
```

Kuvio 14. Tiedoston poistokoodi

Poiston jälkeen Jinja päivittää automaattisesti datanhallintasivun taulukon, mikä pitää käyttöliittymän ajan tasalla ilman, että sivua tarvitsee erikseen ladata uudelleen. Datanhallintasivun toiminnallisuudet takaavat, että analyysiprosessiin käytettävä data on aina ajantasaista ja helposti hallittavissa. Tämä mahdollistaa tehokkaan ja tarkasti kohdennetun analyysin.

4.5 Analyysisivu: datan suodatus ja analyysitulokset

Analyysisivu toimii sovelluksen pääsivuna, joka mahdollistaa chatbotin viestihistorian analysoinnin ja tulosten tarkastelun käyttäjän määrittelemien kriteerien perusteella. Näitä kriteerejä ovat esimerkiksi palvelukategoria, ajanjakso ja avainsanat. Sivu yhdistää suodatuslomakkeen, visuaalisen kaavion sekä taulukkomuotoisen esityksen ja tarjoaa monipuolisen tavan tehdä analyysejä ja tarkastella analyysituloksia. Analyysisivun toiminnot on havainnollistettu kuviossa 15, jossa kuvataan prosessin vaiheet käyttäjän siirtymisestä *index*-sivulta aina tulosten visualisointiin ja Excel-tiedoston lataamiseen saakka.



Kuvio 15. Analyysisivun toiminnallisuuksien vuokaavio

Tässä luvussa käsitellään tarkemmin kuviossa 15 esitettyjä vaiheita, mukaan lukien suodatuslomakkeen toiminta (ks. luku 4.5.1), tietokantakyselyt (ks. luku 4.5.2), tulosten visualisointi (ks. luku 4.5.3) sekä Excel-raportin luominen (ks. luku 4.5.4). Analyysisivun koodi, eli *index*-reitti, on kokonaisuudessaan liitteessä 3.

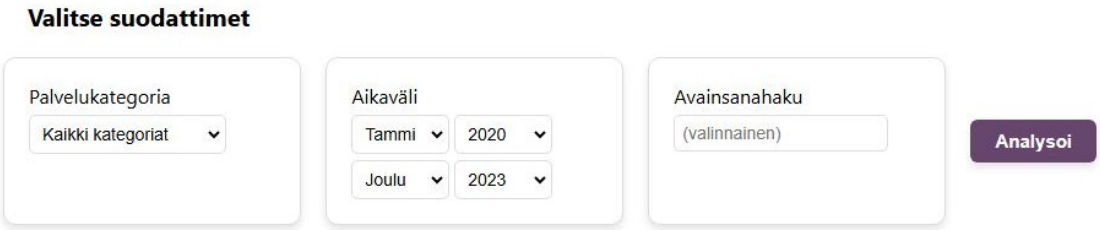
4.5.1 Index-reitti ja suodatuslomakkeen toiminta

Analyysisivun toiminta käynnistyy, kun käyttäjä siirtyy sovelluksen *index*-reitille. Tämä reitti käsittelee suodatuslomakkeelta saadut parametrit, kuten palvelukategorian, ajanjakson ja avainsanan, jotka välitetään URL-parametreina (kuvio 16) sovelluksen backendille. *Index*-reitin tehtävänä on hakea tietokannasta suodatettu data ja ohjata se käyttöliittymään analyysin visualisoimista varten. Mikäli suodattimia ei ole määritelty, sovellus käyttää oletusarvoja. Esimerkiksi kategorian kohdalla oletusarvo on *all*, mikä tarkoittaa, että hakuun otetaan kaikki kategoriat.

```
/?category=all&start_month=01&start_year=2020&end_month=12&end_year=2023&keyword=moi
```

Kuvio 16. Suodatuslomakkeen URL-parametrit

Analyysisivulla sijaitseva haun suodatuslomake (kuvio 17) muodostuu kolmesta osasta: palvelukategoriasta, aikavälistä ja avainsanahausta. Käyttäjä voi valita haluamansa palvelukategorian pudotusvalikosta, määrittää aikavälin asetukset ja tarkentaa halutessaan analyysiä tiettyyn termiin avainsanahaun avulla.



Kuvio 17. Sovelluksen suodatusvalinnat

Kun suodatusvalinnat on tehty, käyttäjä voi käynnistää analyysin *Analysoi*-painikkeella, joka lähettää tiedot sovelluksen backendille jatkokäsittelyä varten. Sovelluksen suodatuslomake hyödyntää Jinja-mallinnusta, mikä varmistaa, että käyttäjän tekemät valinnat pysyvät näkyvissä lomakkeella myös suodatuksen suorittamisen jälkeen. Suodatuslomake toimii käyttäjän keskeisenä työkaluna analyysin tarkkuuden ja laajuuden määrittämisessä. Sen kautta annetut tiedot ohjaavat analyysin taustalla tapahtuvaa SQL-käsittelyä ja visualisointia, mikä mahdollistaa käyttäjälle tarpeisiin mukautuvan analyysin.

4.5.2 SQL-kyselyt ja datan käsittely

Sovelluksen backend vastaanottaa suodatuslomakkeelta URL-parametrit ja käyttää Flaskin *request.args* -ominaisuutta niiden lukemiseen ja käsittelyyn. Flaskin ominaisuus *request.args* mahdollistaa URL-kyselyparametrien lukemisen avain-arvo-pareina, jolloin parametrit voidaan käsitellä helposti sovelluksen backendissä (GeeksforGeeks 2023). Saadut arvot ohjaavat dynaamisesti muodostettavia SQL-kyselyitä (kuvio 18), joiden avulla käyttäjän määrittämät suodattimet rajataan tarkasti. Esimerkiksi aikavälin valinta lisää kyselyyn ajanjaksoa rajaavat ehdot, kun taas avainsanan puuttuessa kysely kattaa kaikki viestit ilman tekstisuodatusta.

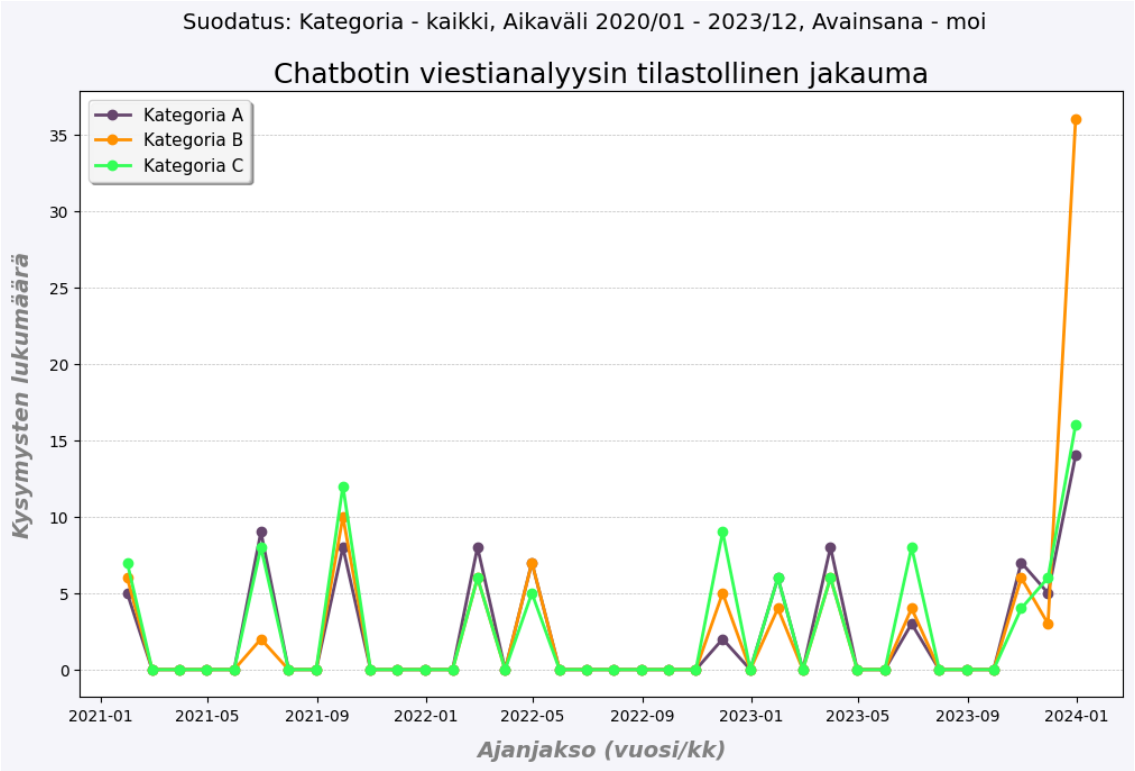
```
time_series_query = """
    SELECT year, month, category, COUNT(*) as count
    FROM data
    WHERE 1=1
"""
filters = []
if start_year and start_month:
    time_series_query += " AND (year > %s OR (year = %s AND month >= %s))"
    filters.extend([start_year, start_year, start_month])
```

Kuvio 18. Ote dynaamisesta SQL-kyselystä analyysidatan suodattamiseen

Kyselyiden joustavuus perustuu *if*-ehtoihin, jotka mukauttavat SQL-lauseita käyttäjän tekemien valintojen perusteella. SQL-kyselyillä haettu data käsitellään backendissä ennen sen siirtämistä käyttöliittymään. Tiedot muunnetaan pandas-kirjaston avulla DataFrame-rakenteiksi, mikä mahdollistaa tiedon jatkokäsittelyn ennen visualisointia. Käsitelty ja muokattu data muodostaa perustan analyysisivun toiminnalle, jossa eri SQL-haut tuottavat tarvittavat datakokonaisuudet. Analyysisivun toiminta perustuu kolmen dynaamisen SQL-haun (*time_series_query*, *category_count_query* ja *filtered_messages_query*) palauttamaan kolmeen datakokonaisuuteen: aikasarjadataan, kategoriakohtaisiin viestimääriin ja yksittäisiin suodatettuihin viesteihin.

4.5.3 Tulosten visualisointi

SQL-kyselyiden palauttamat datakokonaisuudet toimivat perustana analyysitulosten visualisoinnille. Aikasarjadata luo perustan visuaaliselle aikasarjakaaviolle (kuvio 19), joka havainnollistaa viestimäärien kehitystä ajallisesti ja kategorioittain. Kategoriakohtaiset viestimäärät esitetään tiiviinä yhteenvetona analyysin tuloksista ja yksittäiset viestit näytetään taulukkomuodossa yksityiskohtaisempaa tarkastelua varten.



Kuvio 19. Aikasarjakaavio haun tuloksista

Aikasarjakaavio luodaan sovelluksessa *create_time_series_chart*-funktiolla, joka on keskeinen osa analyysisivun toiminnallisuutta ja löytyy kokonaisuudessaan liitteessä 4. *Index*-reitti kutsuu tätä funktiota aina, kun käyttäjä avaa analyysisivun tai tekee uuden suodatuksen, varmistaen, että kaavio päivittyy dynaamisesti käyttäjän valintojen perusteella. SQL-kyselyn palauttama data (*time_series_data*) välitetään *create_time_series_chart*-funktiolle, jossa ensimmäisessä vaiheessa data muunnetaan pandas DataFrame-rakenteeksi (kuvio 20). Tämä DataFrame sisältää kaavion luomiseen tarvittavat sarakkeet, kuten vuosiluvut, kuukaudet, kategoriat ja viestimäärät. Pandasin DataFrame tarjoaa erinomaisen pohjan kaavioiden luomiselle, sillä se kokoaa datan rakenteelliseen muotoon, joka soveltuu suoraan visuaalisten esitysten rakentamiseen (McKinney 2022, luku 9).

```
df = pd.DataFrame(data, columns=['Year', 'Month', 'Category', 'Count'])
```

Kuvio 20. Kaavion DataFrame-rakennekoodi

Aikasarjakaavion tietojen järjestämiseksi DataFrameen lisätään sarake, jossa vuosi ja kuukausi yhdistetään *Date*-sarakkeeksi (kuvio 21). Tämä sarake toimii

indeksinä, jonka avulla tiedot lajitellaan aikajärjestykseen. Pandasin sisäänrakennettu päivämääräindeksointityökalu tukee tehokasta datan järjestämistä aikajärjestyksessä (pandas Documentation 2024b).

```
df['Date'] = pd.to_datetime(df['Year'].astype(str) + '-' +
                            df['Month'].astype(str), format='%Y-%m')
df = df.set_index('Date')
```

Kuvio 21. Kaavion aikasarjatietojen muotoilu- ja indeksointikoodi

Seuraavaksi DataFrame-rakenteesta muodostetaan pivot-taulukko (kuvio 22), jossa tiedot ryhmitellään kategorioiden mukaan. Pivot-taulukot tarjoavat tehokkaan välineen tietojen ryhmittelyyn ja tiivistämiseen, erityisesti silloin, kun data halutaan esittää selkeästi useiden kategorioiden välillä (McKinney 2022, luku 8.3). Näin kysymysten lukumäärät voidaan jaotella palvelukategorioittain ja esittää jokaiselle ajanjaksolle erikseen, mikä parantaa datan visualisointia ja analysointia. Puuttuvat arvot täydennetään käyttämällä *fill_value=0*-parametria, joka korvaa pivot-taulukon tyhjät solut nollilla. Tämä varmistaa, että laskentatoiminnot toimivat ilman virheitä myös silloin kun tietoa puuttuu yksittäisiltä riveiltä tai sarakkeilta.

```
df_pivot = df.pivot_table(index=df.index, columns='Category',
                          values='Count', fill_value=0)
df_pivot = df_pivot.resample('ME').sum()
```

Kuvio 22. Kaavion pivot-taulukon luomis- ja uudelleenryhmittelykoodi

Pivot-taulukkoa muokataan ryhmittelemällä tiedot kuukausittaisiksi yhteenvedoiksi *resample('ME').sum()*-metodia käyttäen. Resample-metodi on erityisesti aikasarjoille suunnattu työkalu, joka mahdollistaa tietojen ryhmittelyn valitun aikajakson mukaisesti (Pandas Documentation 2024a). Toisin kuin *fill_value=0*, resample luo kokonaan uusia aikavälejä, jos tietoa puuttuu tietyiltä ajanjaksoilta ja asettaa niiden arvoksi 0. Tämä varmistaa, että kaikki ajanjaksot ovat edustettuina, vaikka tiettynä kuukautena ei olisi tallennettu lainkaan dataa. Tämä tekee aikasarjasta jatkuvan ja eheän. Lopputuloksena kaavion tiedot esitetään selkeästi kuukausitasolla, mikä helpottaa analysointia ja parantaa tulkittavuutta.

Kun data on käsitelty, sen visualisointi toteutetaan Pythonin Matplotlib-kirjaston avulla kahdessa päävaiheessa: kaavion piirtäminen ja ulkoasun viimeistely. Matplotlib on yleinen työkalu datan visualisointiin ja sen mukautettavat ominaisuudet, kuten värikartat ja kaavion muotoilut, tekevät siitä monipuolisen työkalun julkaisukelpoisten kaavioiden luomiseen (McKinney 2022, luku 9).

Ensimmäisessä vaiheessa kaavioon lisätään jokaisen haussa esiintyvän kategorian (kategoria A, kategoria B ja kategoria C) tiedot erottuvin värein. Käytettävät värit määritellään etukäteen värikartassa (*color_map*), joka yhdistää kunkin kategorian tiettyyn väriin. Kategorioiden tiedot noudetaan pivot-taulukosta, jonka rakenne mahdollistaa kunkin kategorian esittämisen erillisenä aikasarjana. Tämä lähestymistapa helpottaa eri kategorioiden välisten trendien ja vaihteluiden vertailua. Kaavion luomisen yhteydessä jokaiselle kategoriapisteelle lisätään viivatyypit ja merkintäpisteet (*marker*), jotka helpottavat kaavion lukemista (kuvio 23). Tämä prosessi varmistaa, että kaavio kattaa kaikki analyysin kannalta olennaiset ajanjaksot ja esittää trendit selkeästi käyttäjälle.

```
for col in df_pivot.columns:
    color = color_map.get(col, '#000000')
    plt.plot(df_pivot.index, df_pivot[col], label=f'Kategoria {col}',
             linewidth=2, linestyle='-', marker='o', color=color)
```

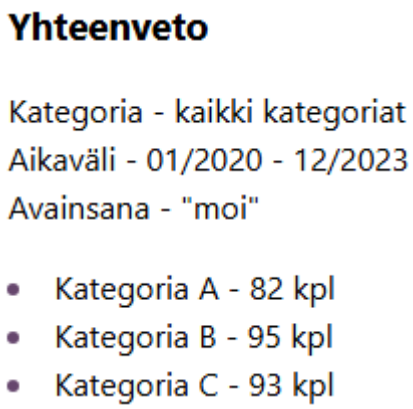
Kuvio 23. Kaavion tietojen visualisointikoodi

Toisessa vaiheessa kaavion ulkoasu viimeistellään lisäämällä tarpeelliset tekstit. Kaavioon lisätään otsikot ja kaavion yläpuolelle sijoitetaan käyttäjän suodatusvalintoihin perustuva metateksti, joka sisältää tiedot valitusta aikavälistä, kategoriasta ja avainsanasta (kuvio 24). Lopputuloksena on selkeä ja helppolukuinen aikasarjakaavio, joka tukee chatbotin viestianalyysin tulkintaa. Tämän jälkeen kaavio tallennetaan sovelluksen hakemistoon polulle *static/aikasarjakaavio.png*.

```
plt.title("Chatbotin viestianalyysin tilastollinen jakauma", fontsize=18)
plt.xlabel("Ajanjakso (vuosi/kk)")
plt.ylabel("Kysymysten lukumäärä")
filter_info = f"Suodatus: Kategoria - {category}, Aikaväli {date_range},
                Avainsana - {keyword}"
plt.figtext(0.5, 0.96, filter_info, ha="center", fontsize=14, wrap=True)
```

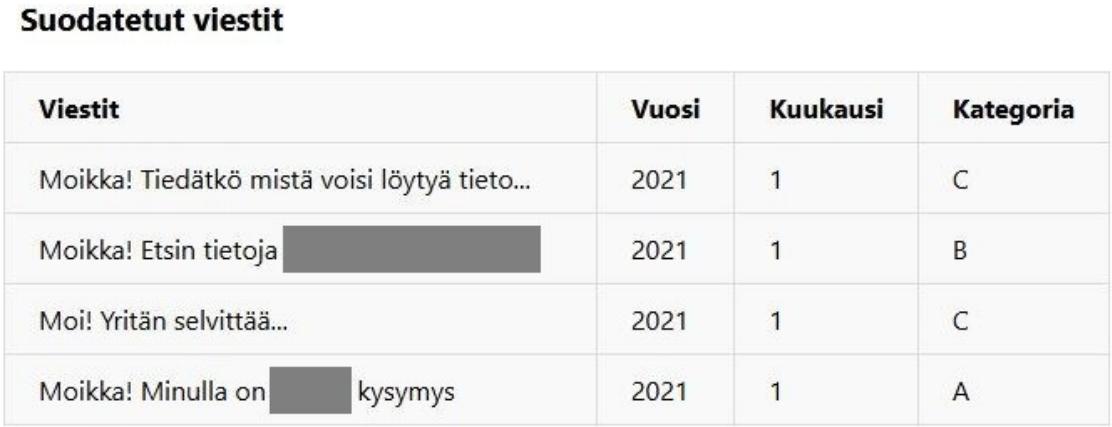
Kaavio 24. Kaavion tekstien lisäyskoodi

Aikasarjakaavion lisäksi analyysisivulla esitetään tiivis yhteenveto haun tuloksen kategoriakohtaisista viestimääristä (kuvio 25). Tämä yhteenveto sisältää haun suodatustiedot, kuten valitun aikavälin, kategorian ja mahdolliset avainsanat. Lisäksi kategoriakohtaiset viestimäärät esitetään listana käyttäen *category_count_query*-kyselyn palauttamaa *category_count_data*-tietorakennetta. Tiedot esitetään Jinja-mallinnuksen avulla, joka luo dynaamisesti HTML-sisällön ja sijoittaa kullekin kategorialle ja sen viestimäärälle erillisen listaelementin (*li*) *index.html*-tiedostoon ennen sen lähettämistä selaimeen. Mikäli tuloksia ei ole, yhteenveto näyttää käyttäjälle ilmoituksen, ettei hakuehdoilla löytynyt dataa.



Kuvio 25. Yhteenveto analyysin tuloksista

Kolmas analyysin keskeinen datakokonaisuus koostuu yksittäisten viestien tarkastelusta taulukkomuodossa. Tämä näkymä täydentää analyysin tulosta tarjoamalla yksityiskohtaisia tietoja suodatetuista viesteistä, kuten viestin sisällön, ajankohdan sekä palvelukategorian (kuvio 26).



Kuvio 26. Analyysisivun taulukkonäkymä

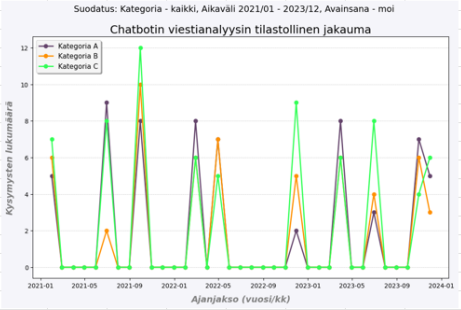
Myös taulukossa (*table*) hyödynnetään Jinja-mallinnusta, joka saa datansa (*filtered_messages*) backendin SQL-kyselystä *filtered_messages_query*. Jinja mahdollistaa suodatustulosten reaaliaikaisen päivityksen aina, kun käyttäjä muuttaa suodatuskriteerejä. Näin taulukon ja yhteenvedon tiedot pysyvät jatkuvasti synkronoituina.

Aikasarjakaavio ja yhteenveto tarjoavat kokonaisvaltaisen yleiskuvan analyysin tuloksista, kun taas taulukko syventää ymmärrystä tarjoamalla yksityiskohtaisempaa dataa. Tämä kokonaisuus tarjoaa analyysin tulokset selkeästi ymmärrettävässä muodossa ja tukee tiedon syvällisempää tarkastelua.

4.5.4 Tulosten vienti Excel-tiedostona

Kun analyysi on suoritettu, käyttäjä voi ladata analyysin tulokset sisältävän Excel-tiedoston analyysisivulta klikkaamalla yhteenvedon ja aikasarjakaavion alapuolella sijaitsevaa *Vie Tiedot Exceliin* -nappia. Tämä ominaisuus helpottaa analyysitulosten jakamista ja hyödyntämistä esimerkiksi organisaation sisäisessä viestinnässä tai päätöksenteon tukena. Ladattava raportti tuo lisäarvoa esittämällä analyysin tulokset tiiviissä ja helposti käsiteltävässä muodossa. Excel-tiedosto (kuvio 27) sisältää valitut suodatustiedot, yhteenvedon viestimääristä, analyysin aikasarjakaavion sekä suodatetut viestit.

	A	B	C	D
1	**Suodatus**	**Arvo**		
2	Kategoria	Kaikki kategoriat		
3	Aikaväli	01/2021 - 12/2023		
4	Avainsana	moi		
5				
6	**Kategoria**	**Määrä**		
7	A	62		
8	B	55		
9	C	71		
10				
11	**Viestit**	**Vuosi**	**Kuukausi**	**Kategoria**
12	Moi! Etsin tietoja	2021	1	B
13	Moi! Minulla on kysymys	2021	1	A
14	Moikka! Etsin tietoja	2021	1	B
15	Moi! Yritän selvittää...	2021	1	C
16	Moi! Mietin tuota...	2021	1	C
17	Moikka! Tiedätkö mistä voisi löytyä tieto...	2021	1	C
18	Moikka! Etsin tietoja	2021	1	B
19	Moi! Minulla on kysymys	2021	1	A
20	Moikka! Tiedätkö mistä voisi löytyä tieto...	2021	1	C
21	Moikka! Minulla on	2021	1	A



Kuvio 27. Analyysin tulokset ladatussa Excel-raportissa

Kun käyttäjä klikkaa latauspainiketta, Excel-raportin luonti käynnistyy Flask-sovelluksen *download_summary*-reitillä. Aivan kuten *index*-reitti, myös tämä reitti hyödyntää SQL-kyselyitä analyysitietojen noutamiseen tietokannasta ja käyttää Pythonin pandas-kirjastoa datan käsittelyyn sekä DataFrame-rakenteiden

luomiseen. Suodatustiedot, kategoriakohtaiset yhteenvedot ja yksittäiset viestit muunnetaan pandas DataFrame-muotoon (kuvio 28). Tämä vaihe varmistaa, että kaikki tiedot ovat yhtenäisessä ja jäsennellyssä muodossa raportin tuottamista varten.

```
filters_df = pd.DataFrame({
    "Suodatus": ["Kategoria", "Aikaväli", "Avainsana"],
    "Arvo": [category if category != 'all' else "Kaikki kategoriat", date_range,
             keyword if keyword else "Ei avainsanaa"]})
summary_df = pd.DataFrame(summary_data, columns=["Kategoria", "Määrä"])
messages_df = pd.DataFrame(filtered_messages, columns=["Viestit", "Vuosi",
                                                       "Kuukausi", "Kategoria"])
```

Kuvio 28. Excel-raportin DataFrame-rakenteiden luonti

Raportin kirjoittaminen toteutetaan hyödyntäen *io.BytesIO*-objektia. *Io.BytesIO* on osa Pythonin io-moduulia ja tarjoaa työkalun tiedon käsittelyyn suoraan muistissa ilman välitiedostoja (Python Documentation 2024). Tiedot tallennetaan Excel-tiedostoon hyödyntämällä pandas-kirjaston ExcelWriter-ominaisuutta. Tiedoston teknisestä rakenteesta ja sisällön luomisesta vastaa XlsxWriter, joka toimii kirjoitusprosessin moottorina (*engine*). XlsxWriter tarjoaa monipuoliset muokkausominaisuudet, sujuvan yhteensopivuuden DataFrame-rakenteiden kanssa sekä mahdollisuuden lisätä PNG-muotoisia kaavioita Excel-tiedostoon (XlsxWriter Documentation 2024).

Kaikki tiedot sijoitetaan *Yhteenveto*-nimiselle välilehdelle, jossa suodatustiedot, kategoriakohtaiset yhteenvedot ja yksittäiset viestit järjestetään loogisesti erillisiin osioihin (kuvio 29). Lisäksi *len*-funktiota käytetään määrittämään, mille riveille eri tietokokonaisuudet sijoitetaan, jotta niiden keskinäinen järjestys säilyy johdonmukaisena.

```
output = io.BytesIO()
with ExcelWriter(output, engine='xlsxwriter') as writer:
    filters_df.to_excel(writer, index=False, sheet_name='Yhteenveto', startrow=0)
    summary_df.to_excel(writer, index=False, sheet_name='Yhteenveto',
                        startrow=len(filters_df) + 2)
    messages_df.to_excel(writer, index=False, sheet_name='Yhteenveto',
                         startrow=len(filters_df) + len(summary_df) + 4)
```

Kuvio 29. Tietojen kirjoittaminen Excel-raporttiin XlsxWriterin avulla

Excel-tiedostoon lisätään myös analyysissä tuotettu aikasarjakaavio, joka on tallennettu *static*-kansioon PNG-muodossa. Tämä toteutetaan käyttämällä XlsxWriter-kirjaston *insert_image*-metodia, jolla määritellään kaavion sijainti taulukossa (*F2*), skaalataan kaavio puolet pienemmäksi ja tarkennetaan sijaintia tarvittavilla parametreilla (kuvio 30).

```
chart_path = os.path.join('static', 'aikasarjakaavio.png')
worksheet.insert_image('F2', chart_path, {'x_offset': 15, 'y_offset': 10,
                                          'x_scale': 0.5, 'y_scale': 0.5})
```

Kuvio 30. Aikasarjakaavion lisääminen Excel-raporttiin

Sarakkeiden leveydet ja tekstin asettelu optimoidaan raportin luettavuuden, sisällön selkeyden ja visuaalisen laadun parantamiseksi. Tähän hyödynnetään XlsxWriter-kirjaston muotoiluparametreja, kuten tekstin keskittämistä ja rivien leveyden säätämistä (kuvio 31).

```
for col_num in range(len(filters_df.columns)):
    worksheet.set_column(col_num, col_num, 20, centered_format)
for col_num in range(len(summary_df.columns)):
    worksheet.set_column(col_num, col_num, 15, centered_format)
```

Kuvio 31. Excel-raportin sarakkeiden- ja tekstienmuotoilu

Lopuksi käyttäjälle palautetaan Excel-muotoinen tiedosto HTTP-vastauksena hyödyntäen Flask-sovelluskehyksen *make_response*-metodia. Tämä mahdollistaa tiedoston lataamisen suoraan sovelluksesta käyttäjän tietokoneelle.

4.6 Sovelluksen testaus

Testauksen tavoitteena oli varmistaa, että sovellus toimii odotusten mukaisesti ja täyttää toimeksiantajan tarpeet ja vaatimukset. Koska kyseessä oli PoC-sovellus ja se suunniteltiin mahdollisimman pelkistetyksi ja yksinkertaiseksi, sen toimintojen testaaminen oli suhteellisen suoraviivaista.

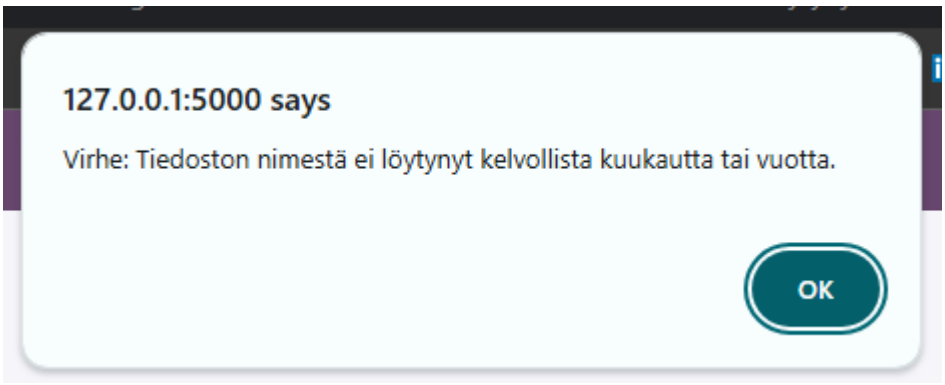
Sovelluksen testaaminen oli keskeinen osa kehitysprosessia ja sitä toteutettiin jatkuvasti kehityksen aikana hyödyntäen Flaskin kehityspalvelinta. Tämä mahdollisti virheiden nopean havaitsemisen ja korjaamisen jo kehitystyön alkuvai-

heista lähtien. PoC-sovelluksen luonteen vuoksi käyttäjätestaukset rajattiin kehitysprosessin ulkopuolelle. Testaaminen keskittyi sen sijaan sovelluksen tekniseen toimivuuteen ja perusominaisuuksien testaamiseen, jotta varmistettiin, että sovellus vastaa toimeksiantajan odotuksia ja todentaa idean toimivuuden.

Yksi keskeisistä osa-alueista oli tietokannan taulujen ja toiminnallisuuksien huolellinen testaaminen. Tietokannan toimivuutta testattiin tietojen lisäämisellä, poistamisella sekä tietokantahakujen avulla. Näiden toimenpiteiden avulla varmistettiin tietokantaan tallennetun datan tarkkuuden ja luotettavuuden säilyttäminen, eli datan eheys. Jinja-mallinnuksen avulla toteutetut dynaamiset taulut helpottivat merkittävästi tulosten ajantasaista tarkastelua. Taulukoissa esitetyn datan oikeellisuus ja tarkkuus varmistettiin vertaamalla sitä MySQL:n phpMyAdmin-ohjauspaneelin tuloksiin. Tietokannan toimintojen luotettavuuden varmistaminen oli kriittistä, sillä sovelluksen toiminta perustuu vahvasti oikeelliseen ja ajantasaiseen dataan.

Käyttöliittymän osalta testaus keskittyi erityisesti sivujen responsiivisuuden sekä lomakkeiden ja suodattimien toimivuuden varmistamiseen. Käyttöliittymän testauksessa otettiin huomioon eri selainten yhteensopivuus; päätestaus suoritettiin Google Chromella, mutta toimivuutta testattiin myös Microsoft Edgellä ja Firefoxilla. Kaavion ja taulukoiden toimivuutta testattiin tarkistamalla, että ne päivittyvät oikein käyttäjän tekemien valintojen perusteella.

Sovelluksen valmistuttua sen toimivuutta testattiin erilaisilla käyttöskenaarioilla, jotka simuloivat todellisia käyttötilanteita. Näihin testitapauksiin kuului muun muassa virheellisten tai puutteellisten Excel-tiedostojen lisääminen. Sovellus suoriutui näistä tilanteista hyvin, sillä se hylkäsi vääränlaiset tiedostot ja esitti käyttäjälle ponnahdusikkunan, jossa kerrottiin virheestä (kuvio 32). Lisäksi testattiin hakutoimintojen tarkkuutta ja suorituskykyä. Näin varmistettiin, että hakutulokset vastaavat tarkasti käyttäjän määrittämiä kriteerejä.



Kuvio 32. Virheilmoitus vääränlaista Excel-tiedostoa lisättäessä

Testauksessa huomioitiin myös tilanteet, joissa Excel-tiedostoissa oli puutteellisia tai vääränmuotoisia tietoja. Sovellus osasi suodattaa pois analyysistä tarpeettomat viestit, kuten esimerkiksi liian lyhyet viestit tai pelkkiä numeroita sisältävät viestit. Tämä varmistaa, että sovellus osaa erotella analyysin kannalta tärkeät viestit tarpeettomista.

Testausprosessissa huomioitiin myös sovelluksen tekniset riippuvuudet. Sovellus asennettiin testaamista varten uuteen ympäristöön, noudattaen toimeksiantajalle laadittua käyttöönotto-ohjeistusta. Haluttiin varmistaa, että Flask-kehityspalvelin, Pythonin virtuaaliympäristö ja tarvittavat kolmannen osapuolen kirjastot, kuten Matplotlib ja pandas, toimivat odotetusti. Tämä asennustestaus varmisti, että sovellus on helposti käyttöönotettavissa myös muissa ympäristöissä.

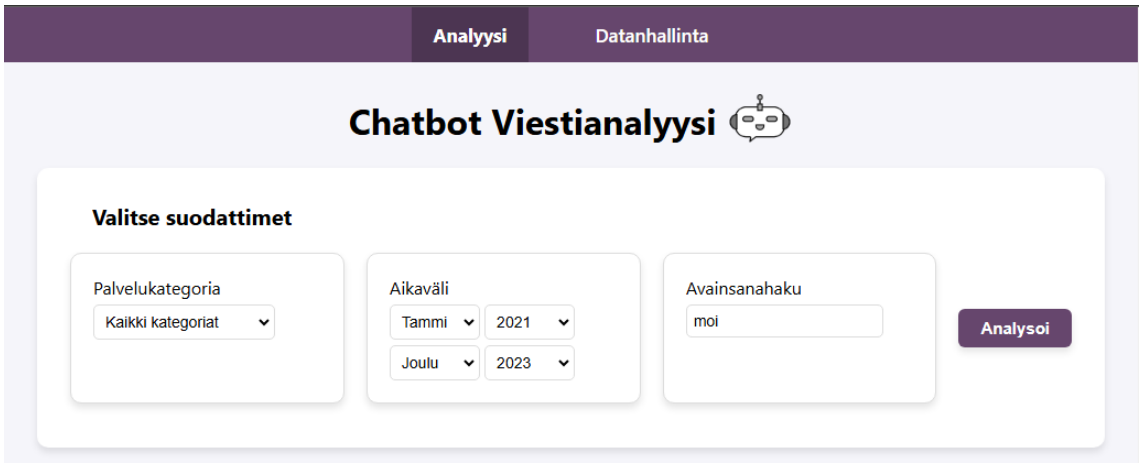
Testausprosessin ansiosta PoC-sovelluksen perustoiminnot todettiin toimiviksi. Reaaliaikainen testaus ja monipuoliset testit varmistivat, että sovellus täyttää vaatimukset ja on valmis esitettäväksi toimeksiantajalle.

5 SAAVUTETTU TUOTOS

5.1 Valmis sovellus

Sovellus koostuu kahdesta pääosiosta: analyysisivusta ja datanhallintasivusta. Näiden sivujen avulla käyttäjä voi vaivattomasti lisätä ja poistaa Excel-tiedostoja, suorittaa analyyseja sekä tarkastella analyysin tuloksia. Sovelluksen intuitiivinen käyttöliittymä ja automaattiset prosessit, kuten datan analysointi tiedostojen lisäyksen yhteydessä, tekevät sen käytöstä sujuvaa ja aikaa säästävää. Tämä tukee tehokasta tiedonhallintaa ja nopeuttaa analyysiprosessia.

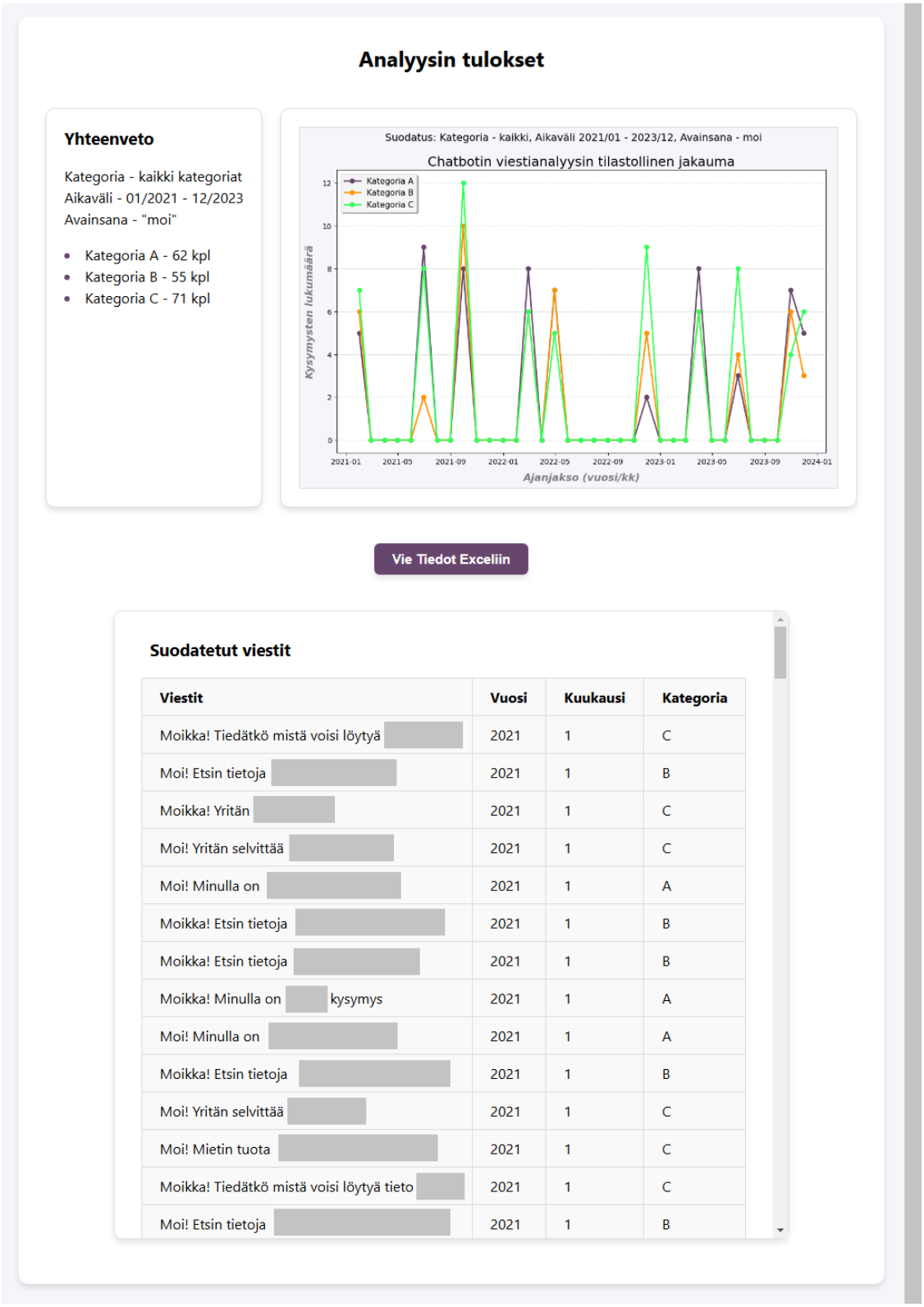
Analyysisivu (kuvio 33) tarjoaa käyttäjälle monipuoliset työkalut viestihistorian tarkasteluun ja analysointiin. Navigointipalkki sijaitsee analyysisivun yläosassa ja tarjoaa käyttäjälle helpon pääsyn sovelluksen eri osioihin. Navigointipalkin alapuolella sijaitsee suodatuslomake, jonka avulla käyttäjä voi suorittaa analyysejä. Sen avulla käyttäjä voi rajata dataa tehokkaasti eri hakuehtojen, kuten kategorian, aikavälin tai avainsanan, perusteella. Tämä mahdollistaa nopean pääsyn juuri siihen tietoon, joka vastaa käyttäjän tarpeita.



Kuvio 33. Analyysisivun yläosa

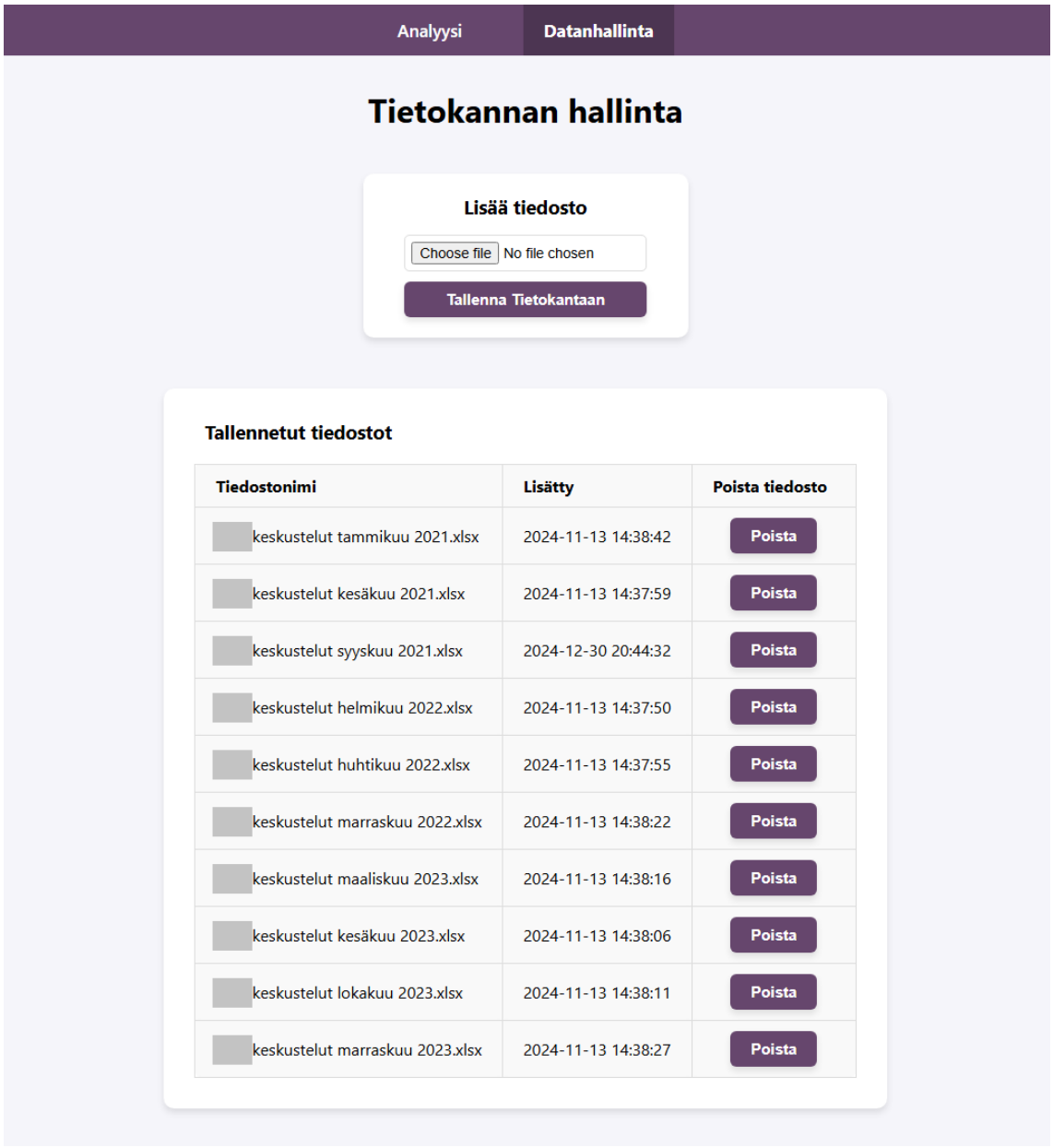
Analyysisivun alaosassa (kuvio 34) tulokset esitetään visuaalisesti ja taulukkomuodossa, mikä selkeyttää tiedon tarkastelua. Aikasarjakaavio havainnollistaa viestimäärien kehityksen eri kategorioissa valitulla aikavälillä ja yhteenveto-osio kokoaa kategoriakohtaiset viestimäärät tiiviiksi esitykseksi. Suodatetut viestit ryh-

mitellään taulukkoon kategorian, vuoden ja kuukauden mukaan, mikä mahdollistaa tarkemman analyysin. Lisäksi analyysitulokset voidaan siirtää Excel-raporttiin jatkokäsittelyä varten yhdellä napin painalluksella.



Kuvio 34. Analyysisivun alaosa

Datanhallintasivu (kuvio 35) puolestaan keskittyy tiedostojen hallintaan ja tietokannan ylläpitoon. Käyttäjä voi lisätä uusia viestidataa sisältäviä Excel-tiedostoja järjestelmään, minkä jälkeen data analysoidaan automaattisesti ja järjestellään valmiiksi analyysia varten. Tarpeettomat tiedostot voidaan poistaa järjestelmästä, jolloin myös niihin liittyvät viestit poistuvat tietokannasta. Näiden toimintojen avulla käyttäjä pystyy ylläpitämään järjestelmän ajantasaisuutta ja eheyttä.



Kuvio 35. Datanhallintasivu

Kokonaisuutena sovellus ratkaisee toimeksiantajan keskeisen ongelman: se tarjoaa työkalun, jolla voidaan analysoida sisäisen chatbotin viestejä. Sen selkeä

rakenne ja automatisoidut toiminnot tekevät siitä helposti omaksuttavan eri käyttäjäryhmille.

5.2 Sovelluksen analyysin tulosten hyödynnettävyys

Sovelluksen analyysitoiminnot tarjoavat kattavan näkymän chatbotin viestihistorian keskeisiin teemoihin ja trendeihin avainsanojen perusteella. Näiden työkalujen avulla käyttäjä voi selvittää, mitkä kysymykset ohjautuvat asiantuntijoiden käsiteltäviksi ja tätä kautta tunnistaa kehityskohteita chatbotin toiminnassa.

Analyysin keskiössä on aikasarjakaavio, joka havainnollistaa viestimäärien kehitystä eri kategorioissa valitulla aikavälillä. Kaavio tuo esiin merkittäviä trendejä, kuten viestien kausivaihteluita ja tiettyjen kategorioiden ylikuormitusta, mikä voi viitata chatbotin tietopohjan puutteisiin tai uusiin nouseviin aiheisiin. Lisäksi kaavion selkeät visuaaliset elementit helpottavat datan tulkintaa.

Yhteenveto täydentää kaavion tarjoamaa yleiskuvaa tiivistämällä analyysitulokset helposti ymmärrettävään muotoon. Kategoriakohtaiset viestimäärät auttavat hahmottamaan nopeasti mitkä aihealueet kuormittavat asiantuntijoita eniten. Tämä mahdollistaa resurssien kohdentamisen tehokkaasti ja voi toimia pohjana chatbotin kehittämiselle.

Taulukko tuo analyysiin yksityiskohtaisuutta esittämällä haun tuloksen viestit yksitellen, jolloin käyttäjä voi tarkastella, mitkä viestit ovat ohjautuneet asiantuntijan käsittelyyn. Tämä mahdollistaa toistuviin teemoihin ja ongelmiin paneutumisen syvällisemmin sekä auttaa selvittämään, mitä tietoja chatbotin tietopohjasta saattaa puuttua. Lisäksi taulukkomuotoinen esitys tarjoaa selkeän tavan hahmottaa laajempaa kontekstia.

Analyysin tulokset voidaan ladata myös Excel-raporttina. Raportointiominaisuus tarjoaa selkeän ja helposti jaettavan tavan hyödyntää analyysidataa esimerkiksi sisäisessä viestinnässä tai kehitystoimenpiteiden suunnittelussa.

Analyysi auttaa kartoittamaan chatbotin kehitystarpeet ja priorisoimaan niitä perustellusti, mikä voi vähentää manuaalisen työn tarvetta ja parantaa chatbotin ky-

kyä vastata työntekijöiden kysymyksiin itsenäisesti ja tehokkaasti. Lisäksi analyysi voi tarjota arvokasta tietoa henkilöstön koulutustarpeista. Tunnistamalla toistuvaa manuaalista tukea vaativat aiheet voidaan järjestää kohdennettua koulutusta, joka vahvistaa osaamista ja parantaa samalla työn laatua ja työntekijöiden tyytyväisyyttä.

Näiden lisäksi analyysin tulokset voivat tukea sekä operatiivista että strategista päätöksentekoa. Data toimii jatkuvan, datalähtöisen oppimisen ja kehittämisen perustana, mahdollistaen tehokkaan reagoinnin muuttuviin tilanteisiin. Sovellus ei rajoitu pelkästään reaaliaikaisen tilannekuvan tarjoamiseen, vaan se voi toimia myös ennakoivan kehityksen työkaluna, edistäen organisaation tavoitteiden saavuttamista.

6 POHDINTA

6.1 Tuotoksen reflektointi

Opinnäytetyön tavoitteena oli kehittää helppokäyttöinen Proof of Concept -sovellus, joka tukee toimeksiantajan sisäisen chatbotin toiminnan analysointia ja kehittämistä. Tutkimuskysymyksenä oli, kuinka voidaan kehittää sovellus, joka analysoi chatbotin keskusteluista kerättyä dataa liiketoiminnan ja chatbotin kehittämisen tueksi. Lisäksi tarkasteltiin, millaisia kysymyksiä chatbot ohjaa asiantuntijalle ja miten toistuvia kehityskohteita voidaan tunnistaa näistä viesteistä. Nämä kysymykset ohjasivat kehitysprosessia, keskittyen erityisesti asiantuntijoille ohjautuvien kysymysten tunnistamiseen ja toistuvien teemojen analysointiin.

Kehitetty sovellus tukee organisaation sisäistä viestintää tarjoamalla systemaattisen tavan analysoida chatbotin viestihistoriaa. Kuten Tkalac Verčić (2021, 365) toteaa, tehokas sisäinen viestintä on keskeinen tekijä organisaation toiminnan sujuvuudelle ja tiedonkululle. Sovelluksen avulla voidaan tunnistaa tiedonkulun haasteita sekä selvittää, missä palveluissa tarvitaan lisäkoulutusta tai chatbotin tietopohjan laajentamista. Tämä vastaa Lukan (2024, 420–422) esittämiin viestinnän haasteisiin, kuten tiedon epäselvyyteen ja redundanssin puutteeseen.

Chatbotit ovat jo aiemmin osoittautuneet hyödyllisiksi organisaatioiden viestinnässä, sillä ne nopeuttavat tiedonsaantia ja vähentävät manuaalisen työn määrää (Huseynov 2023, 56–58). Tässä työssä kehitetty sovellus vahvistaa näitä etuja analysoimalla chatbotin viestihistoriaa ja tunnistamalla, mitkä aiheet vaativat lisähuomiota. Chatbotin keskustelulokien analysointi perustuu deskriptiiviseen analyysiin, joka soveltuu hyvin menneiden tapahtumien ymmärtämiseen ja niiden perusteella tehtäviin päätöksiin (Lepenioti ym. 2020, 57).

Sovellus tarjoaa konkreettisia työkaluja chatbotin keskusteluhistorian viestien analysointiin. Se mahdollistaa asiantuntijoille ohjautuneiden viestien tarkastelun avainsanojen perusteella sekä viestien järjestämisen kategorioittain. Aikasarjakaavio havainnollistaa viestimäärien kehitystä ajallisesti eri kategorioissa, mikä helpottaa trendien ja kausittaisten vaihteluiden tunnistamista. Suodatusmahdollisuudet auttavat käyttäjää keskittymään tiettyihin aihealueisiin tarkemmin. Tulok-

set osoittavat, että sovellus voi auttaa tunnistamaan chatbotin tietopohjan puutteita ja tarjota arvokasta tietoa siitä, missä aiheissa henkilöstö tarvitsee lisäkoulutusta. Lisäksi raportointiominaisuus mahdollistaa analyysidatan hyödyntämisen organisaation päätöksenteossa ja chatbotin kehittämisessä.

Kehitysprosessi pohjautui huolellisesti laadittuun vaatimusmäärittelyyn, jossa painotettiin erityisesti käytettävyyttä ja toiminnallisia tarpeita. Toiminnallisessa määrittelyssä kuvattiin, miten nämä vaatimukset toteutetaan sovelluksen teknisissä ratkaisuissa. Projektin selkeä rajaus vain PoC-vaiheen olennaisiin toimintoihin, kuten datan lisäämiseen, analyysiin ja raportointiin, mahdollisti keskittymisen sovelluksen ydinominaisuuksiin ja konseptin vahvistamiseen. Kehitystyössä yhdistettiin sekä kvalitatiivisia että kvantitatiivisia menetelmiä. Haastattelut syvensivät käyttäjävaatimusten ymmärrystä ja ohjasivat vaatimusten määrittelyä, kun taas chat-lokin datan analyysi toi prosessiin tarkkuutta ja järjestelmällisyyttä. Tämä monimenetelmällinen lähestymistapa varmisti, että sovellus vastasi sekä teknisiin että toiminnallisiin tavoitteisiin.

Yhteistyö toimeksiantajan kanssa oli keskeistä kehitysprosessin onnistumiselle. Projektin aikana pidettiin kuusi suunnittelu- ja seurantapalaveria, joissa tarkennettiin tavoitteita, rajattiin kehitystyön laajuutta ja varmistettiin, että sovellus vastaa käyttäjien tarpeisiin. Kehitysprosessi eteni ketterästi ja iteratiivisesti, mikä mahdollisti saadun palautteen nopean hyödyntämisen seuraavissa vaiheissa. Toimeksiantajan ehdotusten perusteella suodatustoimintoja ja raportointiominaisuuksia mukautettiin paremmin organisaation tarpeisiin. Tämän lisäksi, kun kävi ilmi, että toimeksiantajalta oli saatavilla vain yhden kuukauden chatbot-data, ryhdyttiin välittömästi kartoittamaan realistisen testidatan luomista sovelluksen jatkokehityksen tueksi.

Teknologiavalinnat tukivat projektin onnistumista. Pandas-kirjasto mahdollisti tehokkaan datan analysoinnin ja käsittelyn, Flask-kehitysalusta tarjosi joustavan pohjan sovelluksen rakentamiseen ja Matplotlib toimi keskeisenä työkaluna visuaalisten analyysitulosten luomiseen. Näiden teknologioiden yhteensopivuus ja toimivuus mahdollistivat sen, että sovellus vastasi sille asetettuja tavoitteita.

Vaikka sovellus täyttää sille asetetut keskeiset vaatimukset, kehityskohteita on edelleen. Näitä kehitysalueita on käsitelty tarkemmin luvussa 6.2, jossa esitetään

ehdotuksia sovelluksen parantamiseksi tulevaisuudessa. Kokonaisuudessaan kehitysprosessin onnistumista tukivat selkeästi määritellyt rajaukset, monipuolinen tutkimusote sekä yhteistyö toimeksiantajan kanssa. Kehitetty PoC-sovellus toimii demonstraattorina siitä, miten chatbotin tuottamaa dataa voitaisiin hyödyntää organisaation päätöksenteossa ja kehittämisessä. Lisäksi se tarjoaa arvokasta tietoa chatbotin toiminnasta ja organisaation viestinnän kehityskohteista, mikä tukee tiedolla johtamista ja jatkuvaa kehittämistä.

6.2 Sovelluksen jatkokehitys

Toimeksiantajan kehitysprosessissa ideointi- ja kokeiluvaihe ovat keskeisessä roolissa ennen tuotantokelpoisen version kehittämisen aloittamista. Projektin aikana toteutettu PoC-sovellus toimii tärkeänä apuvälineenä tässä prosessissa, sillä sen avulla voidaan arvioida ratkaisun tarpeellisuutta ja toteutuskelpoisuutta. Sovellus antaa organisaatiolle mahdollisuuden arvioida sen hyödyllisyyttä ennen merkittävien resurssien sitomista tuotantokehitykseen. Lisäksi PoC-sovellus auttaa jatkokehityksen suuntaviivojen määrittelemisessä ja realistisen budjetin laatimisessa.

Jatkokehityksen näkökulmasta sovelluksen käyttöliittymä on yksi keskeisistä kehityskohteista. Vaikka nykyinen käyttöliittymä vastaa käytettävyysvaatimuksia, sen visuaalista ilmettä ja käyttäjäystävällisyyttä voitaisiin parantaa. Selkeämpi ja houkuttelevampi käyttöliittymä voisi lisätä sovelluksen käytettävyyttä. Käyttöliittymän kehityksessä tulisi kiinnittää huomiota myös saavutettavuuteen, kuten värisokeuden ja muiden esteettömyystarpeiden huomioimiseen.

Tarkistustoiminnot, kuten tiedostotyypin validointi ja saman tiedoston tuonnin estäminen, voisivat merkittävästi parantaa sovelluksen virhetilanteiden hallintaa. App.py-tiedoston funktiot ja reititykset olisi hyvä eriyttää omiin moduuleihinsa, mikä selkeyttäisi koodirakennetta ja parantaisi sen ylläpidettävyyttä. Tämä modulaarisuus tekisi myös uusien ominaisuuksien lisäämisestä helpompaa ja mahdollistaisi sovelluksen skaalautumisen laajempiin tarpeisiin. Excel-raporttien hallintaa puolestaan voisi tehostaa lisäämällä tiedostojen nimeämiseen logiikkaa, kuten automaattisesti generoitavan päivämäärän tai käyttäjän valitsemien suodattimien sisällyttämisen tiedoston nimeen.

Palvelukategorioiden ja suodatusmahdollisuuksien laajentaminen voisi olla tärkeä kehitysalue, sillä se parantaisi analyysin kattavuutta ja lisäisi sovelluksen käyttöarvoa. Laajennettavuuden näkökulmasta sovellukseen voitaisiin lisätä tuki avainsanoille eri kielillä, mikä mahdollistaisi sen käytön monikielisissä ympäristöissä. Lisäksi sovellusta voitaisiin kehittää tukemaan muita organisaation toimintoja, kuten esimerkiksi HR-prosesseja, laajentaen sen hyödyntämismahdollisuuksia eri osastoilla.

Tietoturvan parantaminen on kriittinen osa jatkokehitystä erityisesti, jos sovellusta aiotaan hyödyntää laajemmin organisaation toiminnassa. Käyttäjien autentikoinnin ja datan salauksen toteuttaminen ovat keskeisiä toimenpiteitä, jotka varmistavat sovelluksen turvallisuuden ja säännöstenmukaisuuden. Epäluotettavan tiedon syöttämisen estäminen ja virheiden seurantaa varten lisättävä lokitusjärjestelmä vahvistaisivat sovelluksen luotettavuutta ja ylläpidettävyyttä. Tietoturvariskien hallintaa voitaisiin tehostaa sisällyttämällä sovelluksen suunnitteluun myös kattava riskiarviointi.

Skaalautuvuuden tukeminen on myös olennainen osa-alue sovelluksen kasvua ajatellen. Pilvipohjaisen ratkaisun hyödyntäminen mahdollistaisi suurten datamäärien käsittelyn ilman suorituskykyongelmia ja varmistaisi, että sovellus pystyisi tukemaan organisaation kasvavia tarpeita pitkällä aikavälillä.

Kaiken kaikkiaan sovelluksen jatkokehitys vaatii huomiota käytettävyyden, tietoturvan, skaalautuvuuden ja ylläpidettävyyden osa-alueilla. Näiden kehityskohteiden toteuttaminen mahdollistaisi sovelluksen muuttamisen tuotantokelpoiseksi työkaluksi, joka tukisi organisaation liiketoimintaa tehokkaasti ja pitkäjänteisesti.

6.3 Eettiset lähtökohdat

Tutkimuseettisen neuvottelukunnan (TENK 2024) mukaan hyvän tieteellisen käytännön (HTK) keskeisiä tutkimuseettisiä periaatteita ovat luotettavuus, rehellisyys, arvostus ja vastuunkanto. Näitä periaatteita noudatettiin opinnäytetyössä tarkasti. Luotettavuus varmistettiin dokumentoimalla tutkimusprosessi huolellisesti ja varmistamalla sovelluksen analyysiprosessin läpinäkyvyys, jotta tulokset ovat toistettavissa. Rehellisyys toteutui esittämällä tutkimuksen tulokset objektii-

visesti ja vääristelemättä. Arvostus ilmeni toimeksiantajan henkilöstön yksityisyyden suojeluna sekä heidän näkökulmiensa ja toiveidensa huomioimisena tutkimusprosessin aikana. Lisäksi se näkyi Python-ohjelmointikielen ja sen kirjastojen vastuullisessa hyödyntämisessä sekä aiempien tutkimusten ja alan kirjallisuuden asianmukaisessa tunnustamisessa viittausten avulla. Vastuunkanto näkyi siinä, että tutkimuksen kaikissa vaiheissa noudatettiin tietosuojakäytäntöjä ja eettisiä ohjeistuksia.

Datan keruussa ja analysoinnissa noudatettiin EU:n yleisen tietosuoja-asetuksen (GDPR) periaatteita, kuten datan oikeellisuutta, läpinäkyvyyttä ja rekisteröityjen oikeuksien kunnioittamista, jotka asetuksessa määritellään (Tietosuojalaki 2018/1050). Koska tutkimuksessa käsiteltiin sensitiivistä dataa, sen turvallinen käsittely, yksityisyyden suoja ja luottamuksellisuus varmistettiin tietosuojaperiaatteiden mukaisesti. Tietoturvan takaamiseksi toimeksiantajan datan käyttö rajattiin vain tutkimuksen kannalta välttämättömään laajuuteen. Kaikki toimeksiantajan datan käsittely suoritettiin paikallisilla palvelimilla, eikä tietoja tallennettu pysyvästi analyysin ulkopuolisiin järjestelmiin. Tietoja hyödynnettiin ainoastaan sovelluksen kehittämiseen ja testaamiseen.

Luottamuksellisuuden varmistamiseksi toimeksiantajan nimi sekä heidän palveluidensa nimet on poistettu julkisista raporteista ja esityksistä. Palveluihin viitattiin nimillä "palvelu A" ja "palvelu B", ja raportin esimerkeissä sekä koodipätkissä esiintyvät avainsanat sensuroitiin. Haastateltuja työntekijöitä kuvattiin yleisluontoisesti, jotta heidän henkilöllisyytensä ei kävisi ilmi. Lisäksi kaikki analyysiin liittyvät päätökset dokumentoitiin, jotta prosessi säilyi läpinäkyvänä.

Ennen opinnäytetyön aloittamista laadittiin virallinen opinnäytetyösopimus toimeksiantajayrityksen kanssa, jossa sovittiin yksityiskohtaisesti aineiston käsittelystä, tietosuojasta ja muista eettisistä kysymyksistä. Sopimuksessa määriteltiin myös opinnäytetyön raportin julkistamisen ehdot sekä ei-julkistettavat osat, kuten sovellus, sen lähdekoodi ja käyttöönotto-ohjeistus. Lisäksi toimeksiantajalle tarjottiin mahdollisuus tarkastaa opinnäytetyön raportti ennen sen viimeistelyä, jotta varmistettiin, että se vastasi heidän odotuksiaan ja vaatimuksiaan. Näillä toimenpiteillä varmistettiin, että datan käyttö ja tietojen analysointi tapahtuivat täysin

luottamuksellisesti ja että opinnäytetyön tulokset tuotettiin eettisesti kestävällä tavalla.

6.4 Luotettavuuden tarkastelu

Opinnäytetyön luotettavuus perustui huolellisesti valittuihin tutkimusmenetelmiin, datan laatuun sekä analyysin järjestelmälliseen toteutukseen. Keskeistä oli, että käytetyt menetelmät soveltuivat tutkimuskysymysten ratkaisemiseen. Monimenetelmällinen lähestymistapa, jossa yhdistettiin sekä kvalitatiivisia että kvantitatiivisia menetelmiä, varmisti tutkimuksen monipuolisuuden ja kattavuuden. Åkerblad ja Seppänen-Järvelä (2023, luku 9) korostavat, että monimenetelmällisessä tutkimuksessa kvalitatiivisen ja kvantitatiivisen analyysin tulee olla johdonmukaista ja toisiaan täydentävää. Tätä periaatetta sovellettiin tässä tutkimuksessa yhdistämällä toimeksiantajan edustajien haastatteluista kerätty kvalitatiivinen tieto chatbotin viestidatan kvantitatiiviseen analyysiin, mikä tarjosi kattavan näkemyksen chatbotin suorituskyvystä ja kehitystarpeista.

Haastatteluaineiston perusteella määriteltiin analyysityökalun keskeiset vaatimukset. Näiden pohjalta kehitettiin PoC-sovellus, joka mahdollistaa chatbotin viestidatan systemaattisen tarkastelun ja analysoinnin kvantifioinnin avulla. Alun perin kvalitatiivinen viestidata koostui työntekijöiden asiantuntijoille esittämistä kysymyksistä, jotka analysoitiin avainsanojen perusteella ja luokiteltiin palvelukategorioihin. Kvantifiointivaiheessa laskettiin viestien esiintyvyydet eri kategorioissa, mikä mahdollisti chatbotin toiminnan kattavan arvioinnin ja kehityskohteiden tunnistamisen. Kvantifioituja tuloksia hyödynnettiin esimerkiksi aikasarjakaavioissa, jonka avulla voitiin visualisoida toistuvien kysymysten määrän kehitystä eri ajanjaksoina.

Tuomi ja Sarajärvi (2018, luku 6) tarkastelevat kvalitatiivisen tutkimuksen luotettavuutta ja korostavat, että sen määrittely ei ole yksiselitteistä, sillä se pohjautuu erilaisiin tutkimusperinteisiin ja näkökulmiin. He kuitenkin painottavat, että aineiston huolellinen käsittely sekä tutkimusprosessin läpinäkyvyys ja johdonmukaisuus ovat keskeisiä asioita luotettavuuden varmistamiseksi. Tässä tutkimuksessa nämä periaatteet toteutettiin dokumentoimalla käyttäjähaastatteluiden tulkinnat tarkasti ja varmistamalla aineiston käsittelyn järjestelmällisyys. Chatbotin

viestihistoria analysoitiin systemaattisesti ja avainsanojen sekä viestisisältöjen tulkinta tehtiin yhtenäisten kriteerien mukaisesti, mikä lisäsi tutkimuksen uskottavuutta ja siirrettävyyttä.

Bryman ja Bell (2015, luku 6) korostavat, että kvantitatiivisen tutkimuksen luotettavuuden keskeisiä kriteerejä ovat reliabiliteetti ja validiteetti. Reliabiliteetti tarkoittaa mittaustulosten toistettavuutta, mikä tässä tutkimuksessa varmistettiin hyödyntämällä automatisoituja analyysiprosesseja. Validiteetti puolestaan viittaa siihen, kuinka hyvin mittaus todella vastaa tutkittavaa ilmiötä. Tässä tutkimuksessa validiteettia vahvistettiin vertaamalla analyysin tuloksia chatbotin viestihistoriaan sovelluksen testauksen yhteydessä.

Datan käsittelyn laatu oli myös keskeinen tekijä tutkimuksen luotettavuudessa. Chatbotin viestihistoriasta koostuva aineisto edellytti huolellista esikäsittelyä ja puhdistamista analyysin tarkkuuden varmistamiseksi. Tutkimusprosessin läpinäkyvyys toteutettiin dokumentoimalla kaikki vaiheet kattavasti, mukaan lukien projektin rajaus, käytetyt menetelmät ja mahdolliset rajoitteet, kuten analyysiin käytettävissä ollut yhden kuukauden chatbot-data. Tämä rajoite vaikutti analyysin kattavuuteen ja toistettavuuteen, mutta sitä pyrittiin kompensoimaan luomalla realistista testidataa kehitysvaiheen tueksi. Lisäksi aineiston puhdistusprosessissa poistettiin harhaanjohtavat tai tarpeettomat tiedot, mikä vähensi virhemahdollisuuksia. Analyysi toteutettiin Pythonin laajasti käytetyillä ja hyväksytyillä kirjastoilla, kuten pandas ja Matplotlib, jotka tukevat analyysin tarkkuutta ja toistettavuutta.

Kaikki edellä mainitut tekijät varmistivat, että opinnäytetyö tuotti luotettavia ja käytännönläheisiä tuloksia, jotka voivat olla hyödyllisiä sekä toimeksiantajalle että laajemmalle tutkimusyhteisölle. Toimeksiantaja sai käyttöönsä PoC-sovelluksen, joka toimii pohjana tuotantovalmiin version suunnittelulle ja kehittämiselle. Tutkimusyhteisölle työ tarjoaa esimerkin siitä, kuinka deskriptiivistä analyysia voidaan soveltaa chatbot-datan käsittelyyn ja analysointiin. Tutkimuksessa käytetyt menetelmät ja analyysityökalut tukevat myös muiden vastaavien chatbot-analyysihankkeiden toteuttamista ja voivat toimia pohjana tuleville projekteille, joissa chattibottien keskustelulokeja analysoidaan organisaation tiedonkulun ja viestintäprosessien kehittämiseksi.

6.5 Oman oppimisen pohdinta

Opinnäytetyöprosessi tarjosi ainutlaatuisen tilaisuuden oppia monipuolisesti uusia taitoja ja perehtyä niin tutkimuksen tekemiseen kuin siihen, mitä kaikkea toimeksiantajalle tehtävässä projektissa on huomioitava. Projektin laajuus tarjosi sekä haastetta että palkitsevia kokemuksia. Alku oli haasteellinen projektin laajuuden ja sen asettamien vaatimusten vuoksi, mutta olen tyytyväinen siihen, kuinka pystyin systemaattisesti ja määrätietoisesti edistämään projektia haasteista huolimatta. Ketterä kehitysprojekti toimeksiantajan kanssa teki projektista erityisen opettavaisen. Oli myös innostavaa päästä osaksi toimeksiantajan kehitysprosessia ja nähdä, miten oma työpanos voisi tukea organisaation toimintaa.

Teknisellä puolella opin uutena Pythonin Flask-webkehyksen toimintaa sekä laajempaa Python-kirjastojen, kuten pandasin ja Matplotlibin, hyödyntämistä projektin tarpeisiin. Data-analyysin osalta opin näkemään datan arvon aivan uudella tavalla. Projektin aikana koen päässeeni aiheeseen kunnolla sisälle ja opin hahmottamaan, kuinka keskeistä datan analysointi ja visualisointi on päätöksenteon tukemisessa. Koen kuitenkin, että aiheen parissa on vielä paljon opittavaa.

Projektin aikana jouduin sopeutumaan erilaisiin haasteisiin, kuten aikataulumuutoksiin, sekä ylimääräiseen työmäärään, joka syntyi testidatan luomisesta. Näiden tilanteiden myötä opin priorisoimaan työtehtäviä ja suunnittelemaan aikatauluja joustavammin. Kehittämisen varaa on edelleen siinä, että oppisin välttämään liiallista yksityiskohtiin takertumista ja löytämään tasapainon käytännöllisyyden ja laadukkaan lopputuloksen välillä.

Opinnäytetyöprosessi ei kehittänyt ainoastaan teknistä osaamistani, vaan vahvisti myös projektinhallintataitojani ja kykyä mukautua uusiin tilanteisiin. Opin arvostamaan joustavan suunnittelun ja iteratiivisen kehittämisen merkitystä, jotka olivat keskeisiä työn onnistumisessa. Kaiken kaikkiaan koen, että opinnäytetyö antoi arvokasta oppia ja lisäsi itseluottamustani haastavien ja monipuolisten projektien toteuttamisessa tulevaisuudessa.

LÄHTEET

Adamopoulou, E. & Moussiades, L. 2020. Chatbots: History, technology, and applications. Machine Learning with Applications, Vol 2 (2020), 100006. Viitattu 23.10.2024 https://doi.org/10.1016/J.MLWA.2020.100006.

Arya, A. 2020. An overview of textual analysis as a research method for cultural studies. Indexed Journal with IC, Vol 6 (2020). Viitattu 5.11.2024 https://www.ijirmf.com/wp-content/uploads/IJIRMF202003030.pdf.

Åkerblad, L. & Seppänen-Järvelä, R. 2023. Monimenetelmällinen tutkimus: opas suunnitteluun ja toteutukseen. Helsinki: Gaudeamus.

Bond, D. 2007. Reflections on Conscious Communication: A Critical Process for Transitional Societies. Teoksessa K. A. April & M. Shockley (toim.) Diversity. London: Palgrave Macmillan, 219–225. Viitattu 25.12.2024 https://doi.org/10.1057/9780230627529_14.

Box, G., Jenkins, G., Reinsel, G., & Ljung, G. 2016. Time series analysis: Forecasting and control, 5. painos. Hoboken, NJ: John Wiley & Sons.

Bryman, A. & Bell, E. 2015. Business Research Methods. 4. painos. Oxford: Oxford University Press.

Cuesta, H. 2013. Practical data analysis: Transform, model, and visualize your data through hands-on projects, developed in open source tools. Birmingham: Packt Pub.

Flask Documentation 2024a. Quickstart - routing. Viitattu 8.12.2024 https://flask.palletsprojects.com/en/stable/quickstart/.

Flask Documentation 2024b. Templates - Jinja. Viitattu 5.12.2024 https://flask.palletsprojects.com/en/stable/templating/.

Flask Documentation 2024c. Welcome to Flask. Viitattu 1.12.2024 https://flask.palletsprojects.com/en/stable/.

GeeksforGeeks 2021. MySQL – ON DELETE CASCADE Constraint. Viitattu 10.12.2024 https://www.geeksforgeeks.org/mysql-on-delete-cascade-constraint/.

GeeksforGeeks 2023. Using Request Args for a Variable URL in Flask. Viitattu 10.12.2024 https://www.geeksforgeeks.org/using-request-args-for-a-variable-url-in-flask/.

GeeksforGeeks 2024a. Flask tutorial. Viitattu 21.12.2024 https://www.geeksforgeeks.org/flask-tutorial/.

GeeksforGeeks 2024b. MySQL database scalability. Viitattu 21.12.2024 https://www.geeksforgeeks.org/mysql-database-scalability/.

Git Documentation 2024. About version control. Viitattu 11.12.2024 https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control.

Guével, M. R. & Absil, G. 2023. Using mixed methods to evaluate complex interventions: From research questions to knowledge transferability. Teoksessa M. R. Guével & G. Absil (toim.) Global Handbook of Health Promotion Research, Vol 3 Cham: Springer, 201–213. Viitattu 2.10.2024 https://doi.org/10.1007/978-3-031-20401-2_17.

Hargie, O., Tourish, D. & Wilson, N. 2002. Communication audits and the effects of increased information: A follow-up study. Journal of Business Communication, Vol 39 Nro 4, 414–436. Viitattu 25.12.2024 https://doi.org/10.1177/002194360203900402.

Huseynov, F. 2023. Chatbots in digital marketing: Enhanced customer experience and reduced customer service costs. Teoksessa A. S. Munna, M. S. Shaikh & B. U. Kazi (toim.) Contemporary Approaches of Digital Marketing and the Role of Machine Intelligence. Hershey, PA: IGI Global, 46–72. Viitattu 18.10.2024 https://doi.org/10.4018/978-1-6684-7735-9.CH003.

Hussain, S. & Ginige, A. 2019. Extending a Conventional Chatbot Knowledge Base to External Knowledge Source and Introducing User-Based Sessions for Diabetes Education. Teoksessa M. Elkhodr (toim.) Enabling Technologies and Architectures for Next-Generation Networking Capabilities. Hershey, PA: IGI Global, 333–343. Viitattu 20.11.2024 https://doi.org/10.4018/978-1-5225-6023-4.CH015.

Kalla, H. 2005. Integrated internal communications: A multidisciplinary perspective. Corporate Communications: An International Journal, Vol 10 Nro 4, 302–314. Viitattu 10.11.2024 https://doi.org/10.1108/13563280510630106.

Keyton, J. 2017. Communication in Organizations. Annual Review of Organizational Psychology and Organizational Behavior, Vol 4 (2017), 501–526. Viitattu 18.10.2024 https://doi.org/10.1146/annurev-orgpsych-032516-113341.

Khan, R. & Das, A. 2018. Build Better Chatbots. Bangalore: Apress. Viitattu 17.11.2024 https://doi.org/10.1007/978-1-4842-3111-1.

Lepenioti, K., Bousdekis, A., Apostolou, D. & Mentzas, G. 2020. Prescriptive analytics: Literature review and research challenges. International Journal of Information Management, Vol 50 (2020), 57–70. Viitattu 2.10.2024 https://doi.org/10.1016/J.IJINFOMGT.2019.04.003.

Luka, I. 2024. Solving Internal Communication Challenges in Tourism and Hospitality Enterprises in Three Northern European Countries. Journal of Education Culture and Society, Vol 15 Nro 2 (2024), 419–438. Viitattu 9.12.2024 https://doi.org/10.15503/JECS2024.2.419.438.

Matplotlib Documentation 2024. Plot types. Viitattu 30.12.2024 https://matplotlib.org/stable/plot_types/index.html

McKinney, W. 2010. Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference (SciPy 2010), 56–61. Viitattu 11.10.2023 https://doi.org/10.25080/Majora-92bf1922-00a.

McKinney, W. 2022. Python for data analysis: Data wrangling with pandas, NumPy, and Jupyter. 3. painos. Sebastopol, CA: O'Reilly Media. Viitattu 25.12.2024 https://wesmckinney.com/book/.

Metatavu 2023. Vaatimusmäärittely: Mitä se tarkoittaa ja miksi se on tärkeä osa projektin onnistumista? Viitattu 19.9.2024 https://metatavu.fi/vaatimusmaarittely-mita-se-tarkoittaa-ja-miksi-se-on-tarkea-osa-projektin-onnistumista/.

Miklosik, A., Evans, N. & Qureshi, A. 2021. The use of chatbots in digital business transformation: A systematic literature review. IEEE Access, Vol 9 (2021), 106530–106539. Viitattu 5.10.2024 https://doi.org/10.1109/ACCESS.2021.3100885.

pandas Documentation 2024a. pandas.DataFrame.resample. Viitattu 10.12.2024 https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.resample.html.

pandas Documentation 2024b. Time series / date functionality. Viitattu 14.12.2024 https://pandas.pydata.org/docs/user_guide/timeseries.html.

Pîrjol, F. & Radomir, L. 2017. The role of internal communication on the efficiency of the activity in an organization. Business Excellence and Management, Vol 7 Nro 2 (2017), 27–45. Viitattu 1.2.2025.

Python Documentation 2024. io — Core tools for working with streams. Viitattu 13.12.2024 https://docs.python.org/3/library/io.html.

Saaranen-Kauppinen, A. & Puusniekka, A. 2006. 7.3.3 Kvantifiointi. KvaliMOTV – Menetelmäopetuksen tietovaranto. Tampere: Yhteiskuntatieteellinen tietoarkisto. Viitattu 20.12.2024 https://www.fsd.tuni.fi/menetelmaopetus/kvali/L7_3_3.html.

Semanjski, I. 2023. Data analytics. Teoksessa I. C. Semanjski (toim.) Smart Urban Mobility. Amsterdam: Elsevier, 121–170. Viitattu 27.11.2024 https://doi.org/10.1016/B978-0-12-820717-8.00008-7.

Sergiienko, B. 2024. Internal Chatbots: 7 Proven Use Cases & Real Examples. Master of Code Global 23.8.2024. Viitattu 13.12.2024 https://masterofcode.com/blog/internal-chatbot.

Sinčić Ćorić, D., Pološki Vokić, N. & Tkalac Verčič, A. 2020. Does good internal communication enhance life satisfaction? Journal of Communication Management, Vol 24 Nro 4 (2020), 363–376. Viitattu 9.10.2024 https://doi.org/10.1108/JCOM-11-2019-0146.

Srinivasa-Desikan, B. 2018. Natural language processing and computational linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras. Birmingham: Packt Publishing.

Stoudt, S., Vásquez, V. & Martinez, C. 2021. Principles for data analysis workflows. PLOS Computational Biology, Vol 17 Nro 3, e1008770. Viitattu 3.11.2024 https://doi.org/10.1371/JOURNAL.PCBI.1008770.

Taherdoost, H. 2022. What are different research approaches? Comprehensive review of qualitative, quantitative, and mixed method research, their applications, types, and limitations. Journal of Management Science & Engineering Research, Vol 5 (2022), 53–63. Viitattu 16.10.2024 https://doi.org/10.30564/jmser.v5i1.4538.

TENK 2024. Hyvä tieteellinen käytäntö (HTK). Tutkimuseettinen neuvottelukunta 22.4.2024. Viitattu 3.2.2025 https://tenk.fi/fi/hyva-tieteellinen-kaytanto-htk.

Tietosuojalaki 5.12.2018/1050. Viitattu 29.12.2024 https://www.finlex.fi/fi/laki/ajantasa/2018/20181050.

Tkalac Verčič, A. & Špoljarić, A. 2020. Managing internal communication: How the choice of channels affects internal communication satisfaction. Public Relations Review, Vol 46 Nro 3 (2020), 101926. Viitattu 1.2.2025. https://doi.org/10.1016/j.pubrev.2020.101926.

Tkalac Verčič, A. 2021. The impact of employee engagement, organisational support and employer branding on internal communication satisfaction. Public Relations Review, Vol 47 Nro 1 (2021), 102009. Viitattu 18.10.2024 https://doi.org/10.1016/J.PUBREV.2021.102009.

Tsai, W.-H., Chen, J.-S., Soomro, R., Memon, S., Dahri, N., Al-Rahmi, W., Aldriwish, K., Salameh, A., Al-Adwan, A. & Saleem, A. 2024. The Adoption of Digital Technologies by Small and Medium-Sized Enterprises for Sustainability and Value Creation in Pakistan: The Application of a Two-Staged Hybrid SEM-ANN Approach. Sustainability, Vol 16 Nro 17 (2024), 7351. Viitattu 7.10.2024 https://doi.org/10.3390/SU16177351.

Tuomi, J. & Sarajärvi, A. 2018. Laadullinen tutkimus ja sisällönanalyysi. Uudistettu laitos. Helsinki: Tammi.

Wang, X., Lin, X. & Shao, B. 2022. How does artificial intelligence create business agility? Evidence from chatbots. International Journal of Information Management, Vol 66 (2022), 102535. Viitattu 10.10.2024 https://doi.org/10.1016/J.IJINFOMGT.2022.102535.

Wonneberger, A. & Jacobs, S. 2016. Mass Media Orientation and External Communication Strategies: Exploring Organisational Differences. International Journal of Strategic Communication, Vol 10 Nro 5 (2016), 368–386. Viitattu 26.11.2024 https://doi.org/10.1080/1553118X.2016.1204613.

XlsxWriter Documentation 2024. Creating Excel files with Python and XlsxWriter. Viitattu 13.12.2024 https://xlsxwriter.readthedocs.io/index.html.

LIITTEET

Liite 1. Aineistonhallintasuunnitelma

Opinnäytetyön tutkimusaineisto koostui haastatteluilla kerätystä kvalitatiivisesta aineistosta, toimeksiantajan toimittamista avainsanalistoista sekä Excel-tiedostosta, joka sisälsi chatbotin keskusteluhistorian yhden kuukauden ajalta. Haastattelut toteutettiin pääasiassa Teamsin välityksellä ja niiden sisältö dokumentoitiin Word-tiedostoihin jatkoanalyysiä varten.

Koska toimeksiantajalta saatiin käyttöön vain yksi Excel-tiedosto, kehitys- ja testausvaiheessa luotiin käsin lisätestidataa. Tämä testidata koostui geneerisistä viesteistä, jotka sisälsivät avainsanoja sovelluksen toiminnallisuuden varmistamiseksi. Kaikki tiedostot käsiteltiin huolellisesti tietoturvaperiaatteita noudattaen.

Sovelluksen kehitys ja testaus suoritettiin paikallisesti opinnäytetyö tekijän henkilökohtaisella tietokoneella. Toimeksiantajan dataa tai testidataa ei tallennettu pilvipohjaisiin ympäristöihin. Järjestelmätestaus toteutettiin erillisellä henkilökohtaisella tietokoneella käyttäen ainoastaan testidataa, joka poistettiin tietokoneelta testauksen päätyttyä. Sovelluksen lähdekoodin versionhallinta toteutettiin yksityisessä Git-tietovarastossa, jonka käyttöoikeus oli rajattu yksinomaan opinnäytetyön tekijälle.

Opinnäytetyösopimuksen lisäksi toimeksiantajan kanssa solmittiin salassapitosopimus, joka korosti luottamuksellisten tietojen käsittelyä ja turvallisuutta. Sopimus velvoitti luovuttamaan kaikki toimeksiantajaa tai sen asiakkaita koskevat tiedostot takaisin toimeksiantajalle tai hävittämään ne projektin loputtua.

Projektin päätyttyä toimeksiantajalle luovutettiin sovelluksen lähdekoodi sekä käyttöönotto-ohjeistus. Lopuksi kaikki tutkimusaineistoon liittyvät tiedostot, kuten avainsanalistat, lähdekoodi, toimeksiantajan Excel-tiedosto, testidata ja niiden luontiskriptit, poistettiin huolellisesti toimeksiantajan ohjeiden mukaisesti. Näillä toimenpiteillä varmistettiin aineistonhallinnan tarkkuus, turvallisuus ja tietosuojaperiaatteiden noudattaminen koko projektin ajan.

Liite 2. *Process_excel*-funktio

```
def process_excel(file, filename):
    year, month = extract_year_and_month_from_filename(filename)
    if not year or not month:
        return "Tiedoston nimestä ei löytynyt kelvollista kuukautta tai vuotta."
    try:
        df = pd.read_excel(file)
        if 'message' not in df.columns or 'sender' not in df.columns:
            return "Excel-tiedostossa ei ole 'message' tai 'sender' -sarakkeita."
        df = df.dropna(subset=['message', 'sender'])
        process_next_human_message = False
        connection = create_db_connection()
        cursor = connection.cursor()
        cursor.execute("""
            INSERT INTO added_documents (filename, date, year, month)
            VALUES (%s, NOW(), %s, %s)
        """, (filename, year, month))
        added_document_id = cursor.lastrowid
        for _, row in df.iterrows():
            message = row['message']
            sender = row['sender']
            if message.startswith("Aktivointiviesti"):
                process_next_human_message = True
            elif process_next_human_message and sender == "Human":
                category = categorize_message(message)
                if category:
                    cursor.execute("""
                        INSERT INTO data (year, month, category, added_document_id, message)
                        VALUES (%s, %s, %s, %s, %s)
                    """, (year, month, category, added_document_id, message))
                process_next_human_message = False
        connection.commit()
        cursor.close()
        connection.close()
    except Exception as e:
        return f"Virhe: {str(e)}"
    return None
```

Liite 3 1(2). *Index*-reitti

```python
@app.route('/')
def index():
    connection = create_db_connection()
    cursor = connection.cursor()
    category = request.args.get('category', 'all')
    start_year = request.args.get('start_year', '2020')
    start_month = request.args.get('start_month', '01')
    end_year = request.args.get('end_year', '2023')
    end_month = request.args.get('end_month', '12')
    keyword = request.args.get('keyword', '')

    filters = []
    time_series_query = """
        SELECT year, month, category, COUNT(*) as count
        FROM data
        WHERE 1=1
    """
    category_count_query = """
        SELECT category, COUNT(*) as count
        FROM data
        WHERE 1=1
    """
    filtered_messages_query = """
        SELECT message, year, month, category
        FROM data
        WHERE 1=1
    """
    if start_year and start_month:
        time_series_query += " AND (year > %s OR (year = %s AND month >= %s))"
        category_count_query += " AND (year > %s OR (year = %s AND month >= %s))"
        filtered_messages_query += " AND (year > %s OR (year = %s AND month >= %s))"
        filters.extend([start_year, start_year, start_month])
    if end_year and end_month:
        time_series_query += " AND (year < %s OR (year = %s AND month <= %s))"
        category_count_query += " AND (year < %s OR (year = %s AND month <= %s))"
        filtered_messages_query += " AND (year < %s OR (year = %s AND month <= %s))"
        filters.extend([end_year, end_year, end_month])

    if keyword:
        time_series_query += " AND message LIKE %s"
        category_count_query += " AND message LIKE %s"
        filtered_messages_query += " AND message LIKE %s"
        filters.append(f"%{keyword}%")
```

Liite 3 2(2). *Index*-reitti

```
if category != 'all':
    time_series_query += " AND category = %s"
    category_count_query += " AND category = %s"
    filtered_messages_query += " AND category = %s"
    filters.append(category)

time_series_query += " GROUP BY year, month, category ORDER BY year, month"
category_count_query += " GROUP BY category"
filtered_messages_query += " ORDER BY year, month"
cursor.execute(time_series_query, filters)
time_series_data = cursor.fetchall()
cursor.execute(category_count_query, filters)
category_count_data = cursor.fetchall()
cursor.execute(filtered_messages_query, filters)
filtered_messages = cursor.fetchall()
cursor.close()
connection.close()

create_time_series_chart(time_series_data, category, start_year, start_month,
                         end_year, end_month, keyword)
chart_url = url_for('static', filename='aikasarjakaavio.png')

return render_template(
    'index.html',
    chart_url=chart_url,
    category=category,
    start_year=start_year,
    start_month=start_month,
    end_year=end_year,
    end_month=end_month,
    keyword=keyword,
    filtered_messages=filtered_messages,
    time_series_data=time_series_data,
    category_count_data=category_count_data
)
```

Liite 4. *Create_time_series_chart*-funktio

```
def create_time_series_chart(data, category, start_year, start_month,
                             end_year, end_month, keyword):
    df = pd.DataFrame(data, columns=['Year', 'Month', 'Category', 'Count'])
    df['Date'] = pd.to_datetime(df['Year'].astype(str) + '-' +
                                df['Month'].astype(str), format='%Y-%m')
    df = df.set_index('Date')
    df_pivot = df.pivot_table(index=df.index, columns='Category',
                              values='Count', fill_value=0)
    df_pivot = df_pivot.resample('ME').sum()
    color_map = {'A': '#66466d','B': '#ff9100','C': '#33FF57'}
    plt.figure(figsize=(12, 7), facecolor="#f5f5fa")
    for col in df_pivot.columns:
        color = color_map.get(col, '#000000')
        plt.plot(df_pivot.index, df_pivot[col], label=f'Kategoria {col}',
                 linewidth=2, linestyle='-', marker='o', color=color)
    plt.grid(axis='y', color='gray', linestyle='--', linewidth=0.5, alpha=0.5)
    plt.title("Chatbotin viestianalyysin tilastollinen jakauma", fontsize=18)
    plt.xlabel("Ajanjakso (vuosi/kk)")
    plt.ylabel("Kysymysten lukumäärä")
    plt.legend(title="", loc="upper left", frameon=True, framealpha=0.9, shadow=True, fontsize=11)
    if not category or category == "all":
        category = "kaikki"
    if not start_year or not start_month or not end_year or not end_month:
        date_range = "ei määritetty"
    else:
        date_range = f"{start_year}/{start_month} - {end_year}/{end_month}"
    if not keyword:
        keyword = "ei määritetty"
    filter_info = f"Suodatus: Kategoria - {category}, Aikaväli {date_range}, Avainsana - {keyword}"
    plt.figtext(0.5, 0.96, filter_info, ha="center", fontsize=14, wrap=True)
    chart_path = os.path.join('static', 'aikasarjakaavio.png')
    plt.savefig(chart_path, bbox_inches="tight")
    plt.close()
```