

Master's Programme in Computer, Communication and Information Sciences

Large Language Model Agent as Insurance Law Assistant

Amir Ingher

© 2024

This work is licensed under a [Creative Commons](https://creativecommons.org/licenses/by-nc-sa/4.0/) “Attribution-NonCommercial-ShareAlike 4.0 International” license.



Author Amir Ingher

Title Large Language Model Agent as Insurance Law Assistant

Degree programme Computer, Communication and Information Sciences

Major Computer Science

Supervisor Senior University Lecturer Arto Hellas

Advisor Ms Elina Eriksson (MBA)

Date 30.09.2024

Number of pages 73

Language English

Abstract

Traffic insurance law can be a complex domain for ordinary individuals to grasp, thus affecting its accessibility. Although the traffic insurance law itself and precedents are publicly available, it can be challenging to navigate through this information in order to understand one's rights for compensation in the event of a traffic accident.

Recent advancements in Large Language Models (LLM) have provided an accessible way for people to ask curated questions, that the LLM then answers. Thus, the thesis aims to utilize LLM and surrounding technologies to improve the accessibility of traffic insurance law for everyday users trying to navigate this complex legal domain. The research was conducted as a project for an early-stage startup.

The thesis developed an agent that employs Retrieval-Augmented Generation (RAG) techniques with LLM. RAG retrieves relevant information from custom-selected legal documents and uses the information to answer users' questions. The agent is encapsulated within a web application which provides an accessible way for the user to communicate with the agent.

The results were evaluated by a legal expert in a human-feedback manner. The expert indicated that the system could generate satisfactory responses to experts' questions. By employing RAG techniques the system mitigated hallucination which is commonly related to LLMs, therefore improving the quality of the answer. However, the research also found that the document retrieval was suboptimal, thereby needing more work to improve the retrieval process. The study highlights the potential of using LLMs and RAG techniques to make traffic insurance law more accessible to common people. Furthermore, the research suggests that improving the retrieval mechanism and possibly employing a multi-agent system could further improve the contextual understanding of the system and thus aid individuals in navigating legal processes more effectively. Hence, future research will focus on optimizing retrieval processes and exploring the use of multiple agents.

Keywords large language model, retrieval-augmented generation, agent, embedchain, traffic insurance law

Tekijä Amir Ingher

Työn nimi Kielimalliagentti vakuutuslakiasiantuntijana

Koulutusohjelma Tieto-, tietoliikenne- ja informaatiotekniikan maisteriohjelma

Pääaine Tietotekniikka

Työn valvoja Senior University Lecturer Arto Hellas

Työn ohjaaja Elina Eriksson

Päivämäärä 30.09.2024

Sivumäärä 73

Kieli englanti

Tiivistelmä

Liikennevakuutuslaki voi olla vaikea aihealue käsittää tavallisille ihmisille, mikä vaikuttaa sen saavutettavuuteen. Vaikka itse liikennevakuutuslaki ja ennakkotapaukset ovat julkisia, voi näiden tulkitseminen ja oikein ymmärtäminen liikenneonnettomuuden sattuessa osoittautua haasteeksi. Suurten kielimallien (LLM) viimeaikainen kehitys on tarjonnut ihmisille helppokäyttöisen tavan esittää kuratoituja kysymyksiä, joihin kielimalli vastaa. Näin ollen tutkielman tavoitteena on hyödyntää kielimallia ja sen ympärille kehitettyjä teknologioita liikennevakuutuslain saavutettavuuden parantamiseksi jokapäiväisille käyttäjille, jotka yrittävät navigoida tällä oikeusallalla. Tutkimus suoritettiin projektina varhaisen vaiheen startup-yritykselle.

Tutkielmassa kehitettiin agentti, joka käyttää Retrieval-Augmented Generation (RAG) -tekniikoita yhdessä kielimallin (GPT-4 Turbo) kanssa. RAG hakee asiankuuluvaa tietoa valikoiduista oikeudellisista asiakirjoista ja käyttää sitä vastatakseen käyttäjien kysymyksiin. Agentti on sisällytetty web-sovellukseen, joka tarjoaa käyttäjälle helppokäyttöisen tavan kommunikoida agentin kanssa.

Tulokset arvioi oikeudellinen asiantuntija, eli agentin vastaukset vahvistettiin ihmispalautteella. Palautteessaan asiantuntija totesi, että järjestelmä pystyi tuottamaan tyydyttäviä vastauksia asiantuntijan esittämiin kysymyksiin. RAG-tekniikoita hyödyntämällä järjestelmä vähensi hallusinaatioita, jotka ovat yleisiä kielimalleja käyttäessä, näin ollen parantaen vastauksien laatua. Tulokset indikoivat myös, että dokumenttihaku ei ollut optimaalinen, täten hakuprosessia tulee vielä työstää.

Tutkimus korostaa LLM- ja RAG-tekniikoiden hyödyntämisen potentiaalia liikennevakuutuslain saavutettavuuden parantamisessa. Lisäksi tutkimus viittaa siihen, että hakumekanismin parantaminen ja mahdollisesti moniagenttijärjestelmän käyttöönotto voisi edelleen parantaa järjestelmän kontekstuaalista ymmärrystä ja siten auttaa yksilöitä oikeudellisissa prosesseissa vieläkin tehokkaammin. Tästä syystä tuleva tutkimus keskittyy hakuprosessien optimointiin ja moniagenttijärjestelmän käytön tutkimiseen.

Avainsanat kielimalli, RAG, agentti, embedchain, liikennevakuutuslaki

Preface

I want to thank Senior University Lecturer Arto Hellas for guiding me through the thesis process and suggesting key technologies for achieving the artefact.

I would like to thank Elina for suggesting this important topic and for being my advisor. Also, thanks to the team for collectively working for a better tomorrow.

Finally, I would like to thank my family for the support and love, my friends for keeping me sane and alive, Loose for hosting a suitable working milieu, Ms.Weasel for advices and my daughter for existing in my life.

Helsinki, 30.09.2024

Amir Ingher

Contents

Abstract	3
Abstract (in Finnish)	4
Preface	5
Contents	6
Abbreviations and Acronyms	8
1 Introduction	9
1.1 Motivation and Problem Statement	9
1.2 Thesis objective and Scope	9
1.3 Thesis structure	10
2 Background	11
2.1 Large Language Models and Transformer Architecture	11
2.1.1 Transformer	11
2.1.2 Training LLMs	13
2.1.3 Available Large Language Models	13
2.2 Influencing Large Language Model Outputs	15
2.2.1 Prompt Engineering	15
2.2.2 Retrieval Augmented Generation	16
2.2.3 Agent and Multi-Agent System	19
2.3 Law Assistance Systems	21
2.3.1 LawSampo and Semantic Retrieval	21
2.3.2 Large Language Models and Retrieval-Augmented Generation in Law Assistance Systems	22
3 Design Science Research Methodology	23
4 Problem Explication and Requirement Analysis	25
4.1 Definition of Requirements	25
4.2 Functional Requirements	26
4.3 Non-Functional Requirements	28
5 Design and Development	30
5.1 Design and Develop Artefact	30
5.2 Imagine and Brainstorm	30
5.3 Assess and Select	31
5.4 Sketch and Build	31
5.4.1 Walk-Through	32
5.4.2 Sketching the Prototype	32
5.5 Justify and Reflect	33

5.6	Design Decisions	34
5.6.1	Design of Agent (Req 1.1 - 1.5, Req 5.2)	34
5.6.2	Multi-Agent System (Req 2.1 - 2.6)	35
5.6.3	Web Application Architecture (Req 3.1 - 4.3, Req 5.1, Req 5.3)	35
5.7	Agent as a Service	36
5.7.1	Configuration of the Parameters	38
5.7.2	Prompt Construction	41
6	Application Demonstration	44
6.1	Demonstrate Artefact	44
6.2	Application Overview	44
7	Evaluation	47
7.1	Evaluate Artefact	47
7.2	Evaluation of Agent by Human-Feedback	47
7.2.1	Agent with Parametric and Non-Parametric Memory (GPT-4 turbo)	48
7.2.2	Agent with Parametric Memory (GPT-4 turbo)	49
7.3	General Evaluation of Open-Source Models	49
7.4	Reliability of Human-Feedback	50
8	Discussion	51
8.1	Generalization	51
8.2	Resource limitations	51
8.3	Related Works	51
8.4	Future Development	52
8.4.1	Scalability of the Agent	52
8.4.2	Multi-Agent System as a Service	53
8.4.3	Improving the RAG	53
8.4.4	Self-Hosted over Third-Party LLM	54
9	Conclusion	55
A	Appendix	60

Abbreviations and Acronyms

Abbreviations

CoT	Chain-of-Thought
DSRM	Design Science Research Methodology
GPT	General Pre-Trained Transformer
LDA	Latent Dirichlet Allocation
LLM	Large Language Model
LOP	Legal Process Outsourcers
LSTM	Long Short-Term Memory
MAS	Multi-Agent System
MIPS	Maximum Inner Product Search
MMLU	Measuring Massive Multitask Language Understanding
MoE	Mixture-of-Experts
RAG	Retrieval-Augmented Generation
RALM	Retrieval-Augmented Language Modeling
RNN	Recurrent Neural Networks
VRAM	Video random-access memory

1 Introduction

Recently there have been huge developments in Natural Language Processing technology, where the most popularly known technology is the transformer. A massive amount of natural language data fed to the transformer model has given rise to Large Language Models (LLM) capable of generating text on top of a given context [1]. This innovation has the potential to further develop accessibility in many different fields, where one of the fields is the legal industry. In this paper, the focus is on further developing accessibility in the field of law. In the following section further arguments for why this particular topic was picked will be laid out.

1.1 Motivation and Problem Statement

The motivation of this paper is to further the knowledge and accessibility of law. As law in general terms would be a vast topic to tackle, the focus lies in traffic insurance law. The research presented in this paper is conducted as a project for a startup aiming to make insurance law and the general process behind it accessible to clients who have a hard time understanding their rights to compensation when a traffic accident occurs. The Internet is full of open data about insurance cases and given verdicts. Furthermore, the law that determines the rules for individual cases in Finland (liikennevakuutuslaki) is publicly available for example at Finlex ¹. However, acquiring relevant information about individuals' rights for compensation is a difficult task that would require often an expert from the (traffic insurance) legal field. With the help of an expert from the insurance legal domain, publicly available legal documents, and modern technologies, the goal is to build an artefact that guides the user through complex processes and helps them recognize the compensations that they are entitled to based on the context that they, the client, have given for the agent.

1.2 Thesis objective and Scope

The objectives of this thesis are to design, develop, and evaluate an artefact that uses Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) techniques to provide accessible legal support for ordinary people within the domain of traffic insurance law. The specific objectives are as follows:

Legal Support via LLM Agent: Develop an intelligent agent based on LLMs to assist individuals in traffic insurance law.

Reliable Document Retrieval: Implement a retrieval system that fetches relevant legal documents and uses them to generate factually correct responses to user questions.

¹<https://finlex.fi/>

Suitable LLM Models: Evaluate various LLM models to identify a model that works in terms of understanding the given context and can generate satisfactory responses in Finnish.

User-Friendly Interface: Design a web application interface, to ensure broad accessibility by allowing different users to interact with the LLM agent.

Multi-Agent Systems in Legal Context: Explore the option of using multiple agents to enhance the reliability and accuracy of the provided legal support.

Evaluation: The developed artefact will undergo human evaluation by a legal expert who will assess the quality and reliability of the responses generated by the system.

The scope of this thesis is defined as follows:

Domain: The project focuses exclusively on Finnish traffic insurance law.

Technological Framework: This project utilizes Large Language Models and Retrieval-Augmented Generation techniques. The primary model under evaluation will be GPT-4 Turbo.

1.3 Thesis structure

This thesis is structured followingly: Chapter 2 examines pre-existing literature on Large Language Models and surrounding technologies, and how these technologies have been utilized in different studies. The chapter continues to explain the recent technological progress within the legal sector. Thus, providing a comprehensive foundation for the study. Chapter 3 Presents the Design Science Research Methodology, therefore showcasing how the research is conducted. Chapter 4 defines the research problem, explains why it was chosen, and provides an understanding of the existing problem and research gap. It also presents the project's requirements, the process for gathering and identifying them, and the criteria for these requirements. Chapter 5 provides insight into the design and development process of building the artefact, demonstrating how the decisions surrounding development and designs are procedural and evolve during the project. Chapter 6 showcases the architecture in detail and demonstrates the usability of the artefact. Chapter 7 evaluates the artefact and its ability to solve the chosen research problem. Chapter 8 discusses the thesis, including its limitations and future research. Finally, Chapter 9 concludes the thesis with a summary of the research, key findings, and their implications.

2 Background

2.1 Large Language Models and Transformer Architecture

Large language models are language models that are trained with large amounts of text data, usually with sequence data [1]. There exist many models that manipulate given text data, but recently the most popularized model has been the transformer architecture. The model can be used for many purposes e.g. translating a given language to another language, however, the generative feature of the transformer has gained attraction [2]. The following subsection unveils the key components behind the architecture and the training process for achieving the ability to generate text.

2.1.1 Transformer

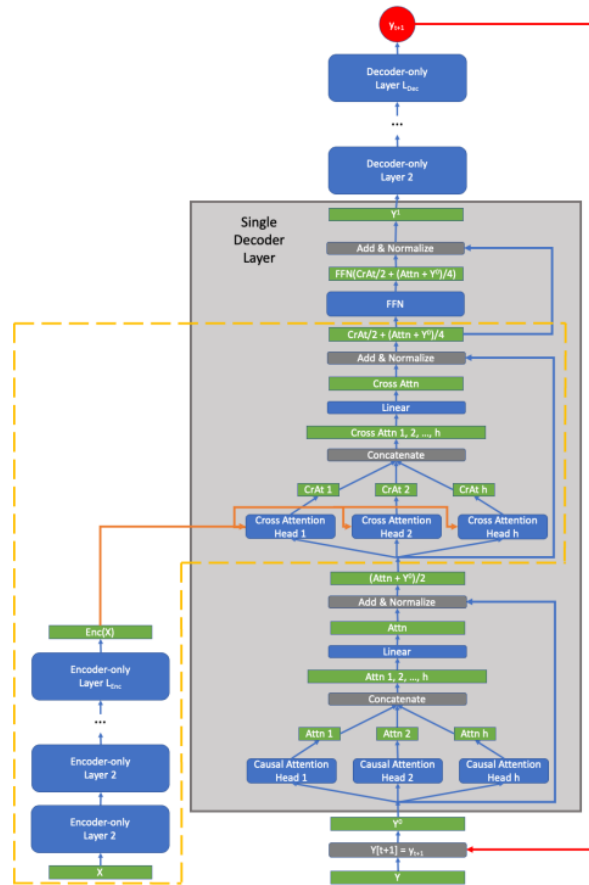


Figure 1: Decoder-only architecture. Everything else is decoder-only architecture except for the parts outlined with yellow. [2].

Transformers offer parallelization and attention mechanisms for self-learning [2]. The interest in this paper lies in the generative features in which the decoder takes input, for instance, a sentence such as “sky is” and then outputs the most likely word

e.g. “blue”. Thus, the type of language model is decoder-only architecture (Figure 1). OpenAI’s GPT series are based on this architecture. The auto-regressive process is repeated until a suitable ending is reached which is informed with the end token. Often characters in sentences are divided into tokens that are strings that appear in the input, however for simplicity, this paper will address them as words. Transformer architecture consists of multiple components, which are further explained below.

Self-Attention Mechanism The Self-Attention Mechanism is responsible for mapping complex patterns and dependencies within a given text [3, 1]. This way the relation of different words in the input is highlighted to give more weight to the word that is relative to the context of the input.

Add and Norm The Add & Norm are used for addition and normalization for the outputs of Masked Multi-Head Attention and Feed-Forward Network [3].

Embedding Input embedding maps the given sentence to float vectors in higher dimensional vector space [3, 2].

Positional Encoding Since the fed text is not processed as sequences of data, but instead mapped as separate vectors, the positional encoder is presented to mark the position of the word in a sentence [3, 2].

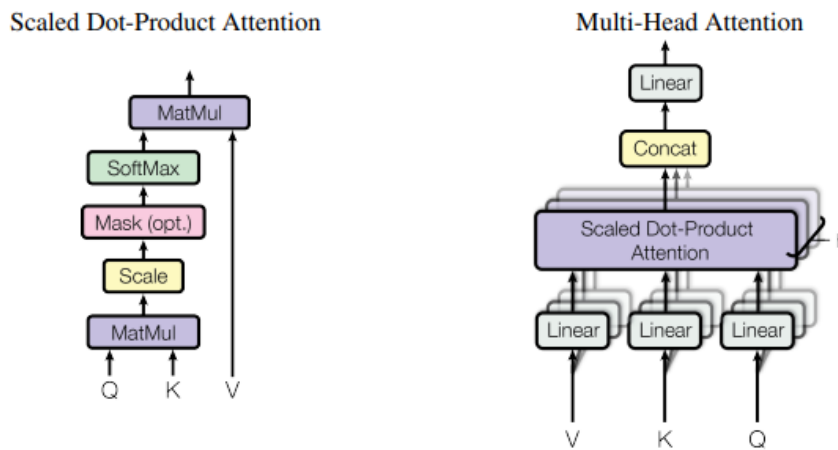


Figure 2: Graph of the dot-product attention (left) and multi-head attention (right) [3].

Scaled Dot-Product Attention Scaled Dot-Product Attention takes embedded vectors and calculates the vector values with weight matrices using a compatibility function [3]. The mentioned method produces query matrix Q and key matrix K, from there, the dot product is calculated between Q and K, followed by normalization, thus the keys attend to the query. Similarly, the given output is produced with matrix

V. The calculation is further visualized in Figure 2. This way the dot product between query and key highlights the relevance of the words and acts as a weight matrix for the value matrix which produces probability for what should be added in the final matrix or “head”.

Masked Multi-Head Attention By using multiple heads, the model may learn many distinct ways to interpret the context [3, 1]. This is called “Multi-Head Attention” and it is done by calculating many heads in parallel and combining the end results of the heads. The “masked” or casual refers to attentioning which converts the numbers to minus infinity so that the words can’t refer to future words. This way it does not gain future information, thus forcing the model to predict the next word from the previous context.

Feed-Forward Network From the Masked Multi-Head Attention, the produced matrix is given as an input for the feed-forward network after addition and normalization [3, 2]. The Feed-Forward network is fully connected and has one hidden layer.

Layers A single layer contains all of the mentioned components. However, when multiple layers are stacked together the model can learn progressively more abstract and complex representations [2]. Each layer builds upon the representations learned by the previous ones, thus enabling the model to understand deeply nested and intricate patterns.

2.1.2 Training LLMs

Pre-training The pre-training phase often is the initial stage of the training. In this process, the model is fed a large amount of text data to learn the structures and relevance of the given text [4]. This is done by tokenizing the text and trying to predict the next token, which generates suitable text in a human-like manner.

Fine-tuning Model fine-tuning refines pre-trained language models by further optimizing the model’s weight parameters contingent on sampled datasets [4]. The process would ideally modify parameters so that it learns the new dataset and stores it in the parametric memory. In this way, new concepts can be learned. The concept can vary depending on the human set goals, for example, if the model should assist in programming tasks the tuning process might involve enhancing the model’s understanding of underrepresented programming languages by translating existing data from well-represented languages or generating synthetic data to improve capabilities like documentation and explanation in coding tasks.

2.1.3 Available Large Language Models

There exists a large collection of large language models which are continuously evolving and published for users to try. However, this subsection covers some of the popular

large language models, both commercial and open-source models. Furthermore, their performance is measured in a variety of different benchmarks, however, this paper will be addressing and presenting a few of the selected benchmarks in the following table 1.

These benchmarks are Measuring Massive Multitask Language Understanding generally known as MMLU and Finnish language performance. Both of these benchmarks offer a way to evaluate language models. MMLU consists of 57 different academic tasks ranging from law, computer sciences, mathematics, and more [5]. It evaluates the model by offering questions with four options that the model should pick from. Finnish language performance is evaluated by inserting a brief Finnish phrase and later evaluating the generated answer with regard to grammatical correctness, coherence, and employ of specialized word stock [6].

Benchmarks		
Models	MMLU	Finnish Performance
GPT-4 Turbo	86,5	X
Llama 33b	57.8	53.36
Llama-3-8B-instruct	29,6	X
FinGPT-8B	X	49,69
Mistral-7B-Instruct	23,06	X

Table 1: Showcasing MMLU and Finnish text generation benchmarks. The benchmark models consist of GPT-4 Turbo ², Llama 33B [7], Llama-3 8B³, FinGPT 8B [6], Mistral 7B⁴.

The selected benchmarks were chosen as they were relevant to the topic. Some of the values are missing due to missing benchmark evaluation from external sources. All of the models mentioned utilize transformer architecture.

Third Party Large Language Models A highly popularized model is OpenAI’s GPT-4-Turbo which is a transformer model. The model in question has performed well in various benchmarks. OpenAI offers a platform where one can run queries for the model via their user interface. Also, users are able to perform more technical implementation by using their API, every request is however charged.

The model scores highest in the MMLU benchmark which is expected due to its large parameter count and resources, e.g. training data size, that have been aiding its development. There was no data for Finnish text generation estimation. However, GPT4 was used to evaluate Finnish language performance in the paper of Poro 34B and the Blessing of Multilinguality [6], therefore which can give insight into the efficiency of the model in regard to Finnish language performance.

Open-source Large Language Models Even though third-party language models offer highly capable language models with huge parameter counts and resource-rich environments, there are openly available models that are currently able to get similar results as third-party solutions like GPT series. These models are called open-source

large language models. In table 1 we list several open-source models and look at their performance. The chosen models are Mistral 7B, Llama 3 8B, and FinGPT 7B. Mistral and Llama are instructional versions, this means that they are fine-tuned to understand prompted instructions.

The benefit of the open-source models is that they are free and private options, compared to third-party vendors. However, the selected model's parameter size is smaller compared to GPT4-Turbo, so they provide a lower need for resources if driven in their own environment. This impacts the benchmark results in the table 1. Where Llama scored highest in MMLU and in second was Mistral. For Finnish language performance, there was no available data to estimate these two models. However, Llama-33B had a slightly higher score compared to FinGPT. Although, the Llama-33B had over four times more parameters than the originally compared Llama-8B.

Quantization Quantization is a fundamental process in the optimization of neural networks, particularly when the resources in the environment are restricted, for example, Video Random-Access Memory (VRAM) size [8]. One primary method of quantization is uniform quantization where the continuous set of real-valued inputs, such as activations or weights, are mapped to a fixed set of integer values. This process involves a scaling factor to adjust the range of real values and a rounding operation to map them to integers. This method provides efficient processing and lower memory requirements for the model.

2.2 Influencing Large Language Model Outputs

To get the LLM to behave in a certain way, it needs to be further refined so that the generated output matches the wanted outcome. This area of influenced text generation is ever-evolving and new methods are continuously introduced. However, this part will introduce several key methods that have been shown to increase the end result of an LLM output.

2.2.1 Prompt Engineering

One way to affect the model's output resource-wise is to use prompt engineering [9]. This should however not be undermined - it takes a lot of prompting and testing effort to find an optimal prompt that yields the model to act according to the user's will in a given task (prompting is demonstrated in Figure 3). Prompt engineering involves designing specific input prompts to guide the language model toward generating the desired outputs. This process typically includes defining roles, giving clear tasks or instructions, and giving contextual information. One of the methods in prompt engineering is "few-shot learning", which involves providing a few examples within the prompts to help the model understand the desired format. This can enhance the adaptability even with minimal data. This approach has proven to be more efficient than zero-shot and one-shot techniques for achieving higher accuracy.

One of the prompting methods is called Chain-of-Thought (CoT). This approach in prompting enhances the reasoning abilities of LLMs by generating a series of

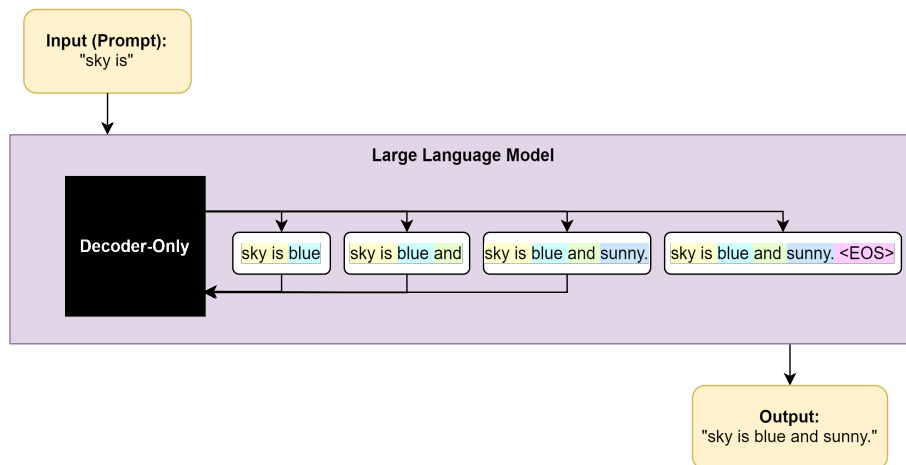


Figure 3: The figure shows an input sentence that is autoregressively generated into probable output.

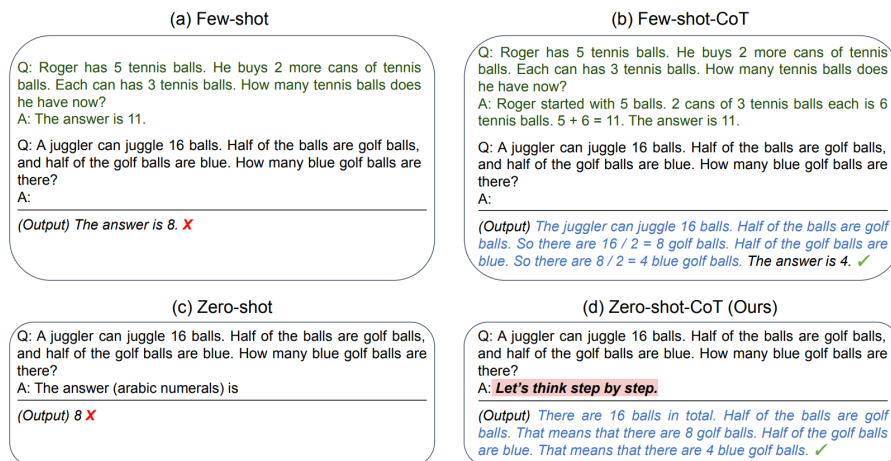


Figure 4: The figure presents zero-shot, few-shot, few-shot-CoT, and Zero-shot-CoT prompting methods. The original image is from the following paper [10].

intermediate reasoning steps that lead to the final output [11]. One exemplary prompt could be "Let's think step by step" which forces the model to break the question into sub-tasks that the model will then try to solve one by one. Zero-shot, few-shot, and CoT are visualized in Figure 4.

2.2.2 Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) is a powerful method for generating more reliable text [12]. It combines non-parametric and parametric memory, where parametric memory stands for the stored memory in model weights and non-parametric relies on the stored knowledge. RAG can mitigate hallucinations and it enables precise and accurate knowledge access mechanisms. This is achieved by storing documents in a vector database where relevant information is retrieved based on the query and

augmented with the query and context. Hence the generative model is capable of answering the question from the given context.

Due to RAG benefits, it has gained popularity in dissimilar fields like programming and finance. For example in programming, RAG has been used to help students in introductory coding courses which were designed to assist in assignments, but not to give straight answers [13]. RAG is used to confine the answering to specific subjects within the homework. Research conducted on 455 students found that it has impacted their grades positively and it has answered 93.5% to the question “good” based on the inquiry directed at students.

Another way the RAG technique has been utilized in funding. More particularly, funding in Italy where chatbot application guides users to seek suitable public procurement for funding their company [14]. RAG is implemented for retrieving summarized information about potential funding that the user can investigate and brainstorm. Later, RAG is used for retrieving more specific information about the selected tenders. This has sped up the interference process for correct document retrieval. These factors indicate that the RAG technique provides certain reliability for divergent domains in assisting tasks where specific knowledge plays a vital role.

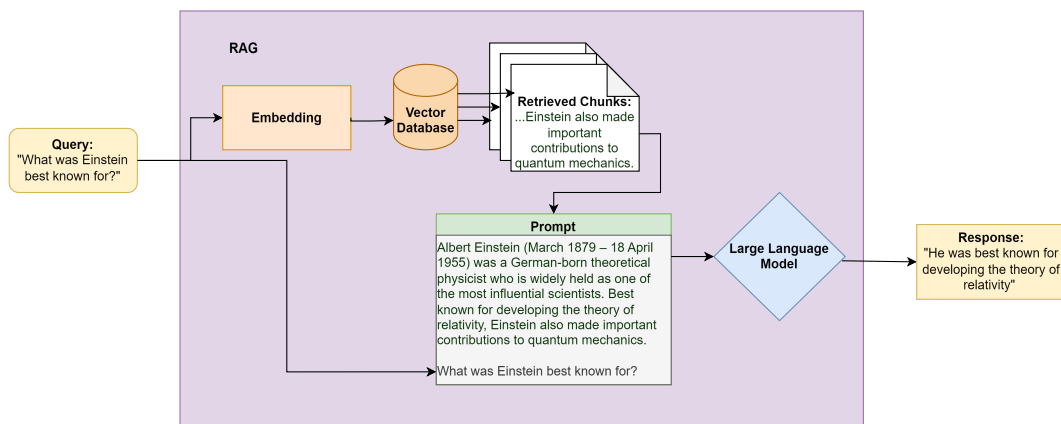


Figure 5: Retrieval-augmented generation (RAG) showcasing scenario where the user asks for information about Albert Einstein which is embedded, retrieved, and concatenated to the final prompt as a context for the LLM to answer from it.

In practice, the mechanism can be divided into the following parts (also visualized in figure 5):

Embedding The given documents e.g. Wikipedia pages are sliced into parts, for instance, a fixed number of characters called chunks [15]. The chunks are then turned into vectors via embedding. The embedding captures the similarity of the sentences in high-dimensional vector space. The relation between vectors can be learned with word2vec, BERT or GPT [16, 17, 3].

Retriever The second part is the retriever functionality that fetches the stored documents from the vector database by using the query [12]. By utilizing methods

like cosine similarity where the value of the angle determines the relevance it enables an efficient way to search within the vector space. Furthermore, one can also employ Maximum Inner Product Search (MIPS) for the same comparing task, where the MIPS is sensitive to angle and vector magnitude. Embedding is used for context embedding and query embedding. Where context is embedded in a vector for storing in a vector database and query embedding aims to impact the retrieved result so that the values from context are related to the query. The top K method is taken advantage of for presenting K amount of similar embedding where the K stands for given integer that decides the amount of documents that should be retrieved.

Construction of final prompt When the results from the retriever have been fetched based on the query, the results and the question are integrated into the prompt. This can be referred to as In-Context RALM [18]. The model receives a set of information followed by a question about the topic. This can be exemplified as follows:

Mistral AI is a French company selling artificial intelligence (AI) products. It was founded in April 2023 by previous employees of Meta Platforms and Google DeepMind.[1] The company raised €385 million in October 2023,[2] and in December 2023, it was valued at more than 2 billion

What is the value of Mistral?"

Mistral AI was evaluated to be 2 billion in December 2023

The first paragraph acts as the retrieved context and the second one as the question. Third paragraphs are generated from the given context and question.

Evaluation Metrics The evaluation methods for RAG were divided into these three classes faithfulness, context relevance, and answer relevance, in the Retrieval Augmented Generation Assessment (RAGAS) framework [19]. Where faithfulness aims to measure how grounded the generated answer is to the given context. This is achieved by collecting a set of statements from answers by using LLM. The process continues to compare the statements against the context via LLM which gives verdicts if the statement is relevant or not. The faithfulness score is then calculated by dividing the supported verdicts with the amount of statements. Answer relevance is calculated by first generating potential questions for the given answer with the help of LLM and then taking cosine similarity between the original question and the generated questions within embedding space. These cosine values are then averaged, thus giving the answer relevance score. Context relevance is attained by extracting a subset of sentences from context through LLM. These sentences should be critical for answering the question. Finally, the context relevance is then divided by the amount of extracted sentences with the total count of sentences within the context.

2.2.3 Agent and Multi-Agent System

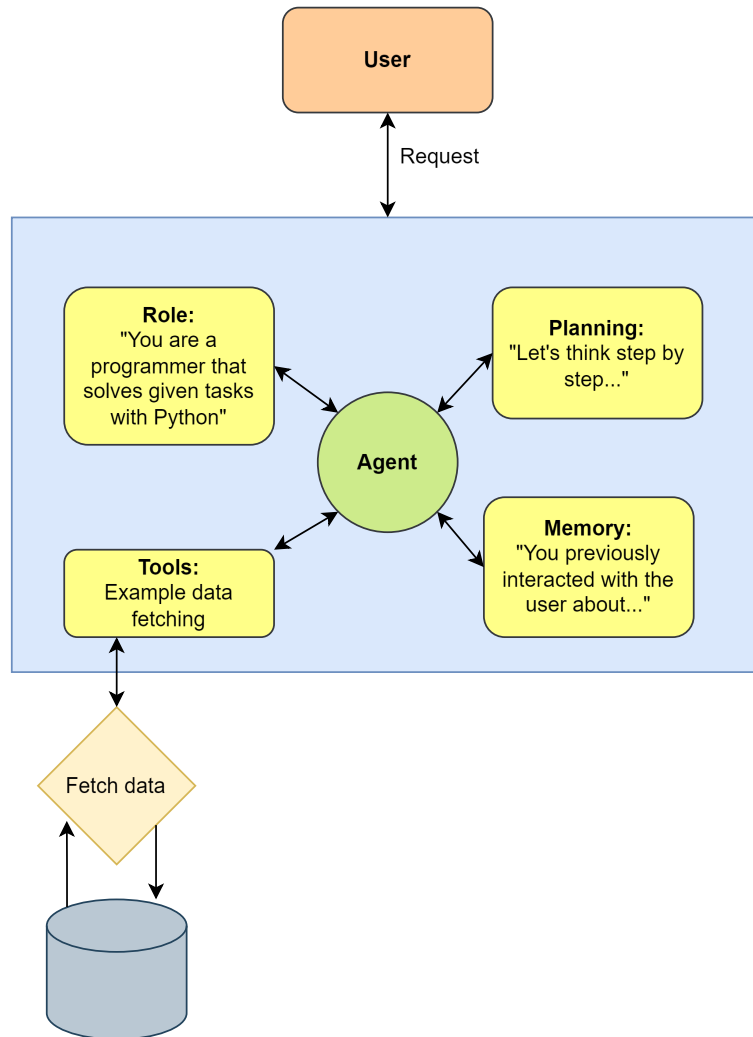


Figure 6: The figure demonstrates a series of features that the agent can withhold. Where the most recognized are role which specifies how the agent should behave, planning which lets the LLM think for example in a sub-task manner (CoT), memory stream for storing summarized versions of conversations, and tools that enable the agent to execute different actions like fetching data from data storage.

An agent often refers to an entity that reacts to different environmental changes by executing some form of an action in reinforcement learning. However, in the context of LLM, the agent interacts with the conversational manners in the environment [20]. These interactions may happen with humans or with other agents (Figure 6). The agent might obtain several features like "Reflexion" where it uses verbal reinforcement instead of weight updating like in typical reinforcement learning scenarios. In the process, the agent reflects on feedback on previous tasks. Another feature that might be used is "memory stream" where summarized versions of the conversations are stored and act as a memory for future decision-making. Agents may also have unique

roles set for them to solve a specific task. Prompt engineering can also be utilized, for example, CoT, where a given task is further divided into smaller sub-tasks.

A multi-agent system (MAS) is a setup where multiple agents interact with each other to pursue a designed goal (Figure 7). This way the general task is divided into sub-tasks that the agents try to solve. The agents have their own roles that are seen as necessary when solving these chores. The task processes can include collective problem-solving through arranged workflows where each agent processes given information and passes it to another [21]. Another form of collective interaction is "multi-turn dialogue" and "multi-agent debate", where conversations between agents are iteratively carried out to refine solutions. In the multi-agent debate, the agents question each other, thus having to reflect and improve their answers after the other agent has pointed out from this questioning process [22].

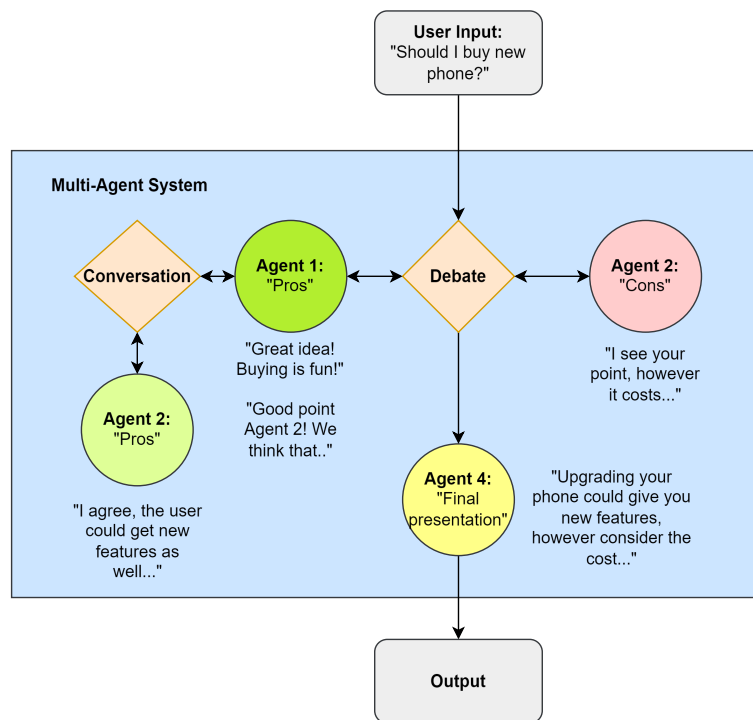


Figure 7: The figure demonstrates an imaginary situation where a user requires pros and cons from a multi-agent system to buy a new phone. The system includes conversational agents which reasons for buying a phone and one agent who debates against it. The final agent presents both arguments of the debate.

The paper Communicative Agents for Software Development [23], introduces a "multi-agent collaboration" framework for software development. It utilizes the waterfall model for collectively producing code. The paper presents a variety of different roles like programmer, tester, and CTO to name a few. The research is one of many examples of how MAS can be used to mimic the organizational level of structures and processes.

2.3 Law Assistance Systems

In this subsection, we research the concept of semantic retrieving and explore the specific roles and impacts of large language models within legal text analysis. Finally, we turn our focus to practical application by examining the utilization of Retrieval-Augmented Generation (RAG)-optimized LLMs in the context of housing dispute resolution in Finland.

2.3.1 LawSampo and Semantic Retrieval

In 1997 the Ministry of Justice of Finland started publishing legislation and case laws as HTML to Finlex Data Bank [24]. Research for improving the “machine-readable legal information” was started in 2012 [24]. Semantic Web standards and Linked Data technologies were proposed for a more auspicious approach. Two versions of the Semantic Finlex utilizing Linked Data were published in 2014 and 2017 [25, 26]. Both of the versions included combined laws, judgments of the Supreme Court, and judgments of the Supreme Administrative Court. The later version (2017) incorporates EU standards for ELI European Legislation Identifier⁵, ECLI European Case Law Identifier⁶, and functionality for automatically updating new legislations. The Semantic Finlex service utilizes the 7-star model⁷ which is an improvement of the 5-star model presented by Berners-Lee⁸. REST API is also provided for easier implementation.

The demonstrative application was built on top of the Semantic Finlex service [24]. This application is called LawSampo, which is a semantic portal. It provides several functionalities, statutes, case law, and case law search. Statutes link opens a view for faceted search, where one can filter statutes on any facet categories. These categories can be document type, year, statute type, or EU directives. A similar approach is used in case law for searching court decisions where the judge’s name and keywords are used as facets. In the case law search the user can use a document or text for retrieving similar judgements. Few methods were used for retrieving similar cases for example TF-IDF, Latent Dirichlet Allocation (LDA), Word2Vec, and Doc2vec [27, 28, 29, 30].

The retrieval methods are further studied in the context of the Supreme Court and Supreme Administrative Court’s judgments [31]. These consist of 13,000 cases published by the Ministry of Justice of Finland. A conclusion for the best-performing individual model against TF-IDF, LDA and Word2Vec was Doc2Vec which is a neural network based model. These methods transform legal texts into vectors, thereby capturing semantic similarity by using cosine similarity [32] for more accurate document retrieval. One motivation for better information retrieval is for a person with a received sentence to gain knowledge about similar cases so that the person can compare its legitimacy.

⁵<https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:C:2012:325:0003:0011:EN:PDF>

⁶[https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52011XG0429\(01\)](https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52011XG0429(01))

⁷www.ldf.fi

⁸<https://www.w3.org/DesignIssues/LinkedData.html>

2.3.2 Large Language Models and Retrieval-Augmented Generation in Law Assistance Systems

Large language models such as GPT-3.5 and GPT-4 have shown promising advancements in legal text annotation [33]. By using zero-shot prompting these models have the potential to reduce the cost compared to manual annotation or supervised machine learning. The study points out that the state-of-the-art LLM advances the democratization of legal services and is recommended for scholars, educators, and legal professionals to leverage it.

GPT4 has performed well against Junior Lawyers and Legal Process Outsourcers (LOPs) in determining legal issues from contracts with an F-score [34] of 87,1%, where LOP succeeded the LLM with an 87,4% and Junior Lawyers managed to take third place with score of 86% [35]. In the location of legal issues within contracts, a similar trend could be observed where LLM took second place when competing against LOP. However, when compared in terms of speed the GPT4 outperforms humans with an average time of 4.7 minutes per document whereas Senior Lawyer achieved 43.46 minutes.

When RAG is fused with LLM it has proven to increase the performance and mitigate hallucination. However, when presenting an application for resolving housing disputes in Finland, GPT-3.5 Turbo combined with RAG optimization did not show promising results when a created assistant was evaluated by a legal expert [36]. It lacked reliability and accuracy when generating answers. Also, RAG had little impact when compared against non-optimized models. Nevertheless, using GPT-4 or higher series might offer improvements in the results [37].

As supplementary functionalities like Mixture-of-Experts (MoE) [38] and multi-agent systems are augmented with LLM and RAG they show enhancement in reliability and accuracy [39]. Chatlaw scored higher than GPT-4 in terms of understanding the legal text, remembering notations, and the capability to include legal knowledge in real scenarios. Multi-agent collaboration agents were given unique roles like Legal Assistant, Legal Researcher, Senior Lawyer, and Legal Editor which operated in configured manners to improve reliability. Legal Researcher employed with RAG aims to alleviate the effect of out-of-date legal documents by fetching and evaluating relevant documents from domain databases and the internet.

These studies showcase potential obstacles, but also improvements regarding LLM and RAG usage in the legal domain.

3 Design Science Research Methodology

This chapter will focus on the research methodology, that is the different techniques, methods, and processes used in this study. Simply, this chapter aims to present how the research is conducted. This chapter therefore presents the Design Science Research Methodology or DSRM, which is the chosen methodology.

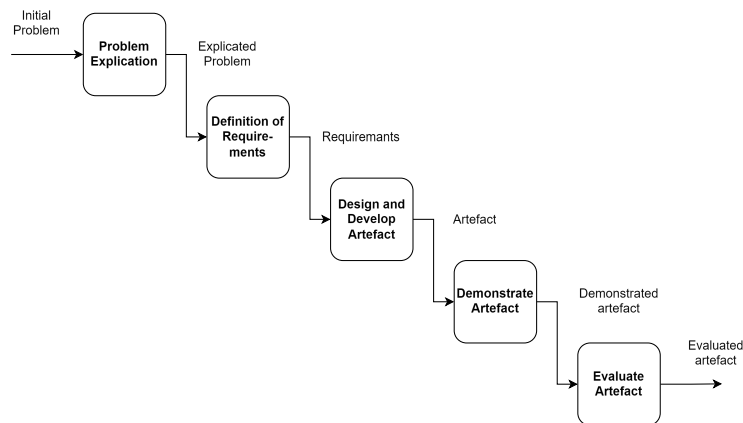


Figure 8: Overview of the method framework for design science research inspired from [40].

The Design Science Research Methodology provides a framework for managing research and guiding the systematic creation and evaluation of innovative artefacts to solve problems or improve existing conditions [40]. DSRM distinguishes itself from traditional empirical research by its action-oriented focus on practical utility and implementation. The methodology is structured into five key activities:

Problem Explication This step involves looking closely at a real-world issue and analyzing it. The problem should be clearly defined and shown to be important for some field of practice, not just locally but globally. Where local practice stands for “a practice in which just one single individual, group, or organization engages” and global practice means “a community of local practices, and to the research community” [40]. Furthermore, it’s important to figure out what’s causing the problem.

Requirement Definition In this step, we list out the requirements for the solution and the artefact.

Design and Development In this activity the artefact is designed and built to tackle the problem.

Demonstration Once the artefact is crafted, it enters the testing phase. The artefact is set in a real situation or in a simulated one to see if it actually works. This step introduces a prototype to prove it’s feasible.

Evaluation Finally, the provided artefact is evaluated. In this step, the goal is to examine whether all of the criteria set in earlier steps are met and to what extent the artefact actually solves the original problem. This step enables better understanding and possible further development.

By systematically progressing through these five main steps, the DSRM framework ensures a comprehensive and coherent approach to design science research, ultimately leading to the development of a solution to the problem at hand.

4 Problem Explication and Requirement Analysis

By following the DSRM framework the first step is to identify the problem before starting to create any sort of design [40]. This chapter will focus on explicating the core problem as well as listing the main requirements.

As stated before, the objective of this thesis is to, by utilizing LLM and RAG, build an agent that provides accessible services to aid and guide traffic accident victims. The research problem was identified when I was approached by a person who had been in a traffic accident and was dissatisfied with the compensation provided by the insurance company. The traffic accident victim in question had extensive connections within the legal network who were available for consultation. This reach-out to these pre-existing connections prompted a long legal discussion about what kind and size of compensation insurance companies' customers are entitled to. After a long process of debating back and forth with the insurance company, a fair agreement was reached.

The problem was therefore selected based on a real-life occurrence and it was clear from the start, that the research would be rather practical than theoretical since the goal was to create an application that would work as a practical solution for this real-world problem.

The key element in reaching a desired outcome was the victim's already established connections within the legal community. After identifying this component, it was established that the problem everyday people have is that they lack these kinds of legal connections that are able to provide free, low-threshold legal counsel.

After this initial hypothesis, we consulted an expert who verified that it is no rare occurrence that, when involved in traffic accidents people often lack the knowledge, network and capital to get legal support. The expert mentioned that this trend can be observed in the social media space, for example, Facebook, where people create groups that act as communes, where they share their similar experiences with insurance companies and give peer support and advice to each other.

4.1 Definition of Requirements

DSRM presents a concept called Define Requirements, which aims to understand the listed problems and connect the gap between problem and solution [40]. Another way to think about it is to ask the following question:

“What artefact can be a solution for the explicated problem and which requirements for this artefact are important to the stakeholders?”

After identifying the core problem, the lack of accessible, low threshold legal support in traffic insurance cases, it was clear that the requirements needed to reflect this issue. A team, that later turned into a startup, was assembled. The team consisted of the founder who approached with the initial abstract concept, three software developers, and one traffic insurance law expert. Based on different areas of expertise, the team considered what requirements this artefact would need in order to be able to create a

preliminary solution. Software developers sketched technical requirements, which the founder and traffic insurance expert verified.

An artefact is described as a human-made object, intended to solve problems [40]. Thus the requirements presented in this chapter were chosen with the intent of building an artefact capable of solving this chosen real-world issue. The discussed and researched initial requirements can be divided into the following two requirement categories:

- Functional requirements and
- Non-functional requirements

The functional requirements refer to the functionality of the artefact which depends on the stated problem and needs of the stakeholders. They are in many cases a particular selection of requirements. The non-functional requirements are more generic requirements and can also be formulated from goals e.g. the solution should increase profit by 10%.

In the following sub-chapters, the chosen requirements will be presented, starting with the functional requirements and moving on to the non-functional requirements.

4.2 Functional Requirements

ID	Component	Requirement
Req 1.1	Agent	Should utilize a language model capable of generating and understanding the Finnish language.
Req 1.2	Agent	The sources that the model references should be up to date and chosen with field experts.
Req 1.3	Agent	API framework shall handle the communication between agent and client.
Req 1.4	Agent	The agent shall utilize RAG techniques to enhance the reliability of the generated answers.
Req 1.5	Agent	Delay of replay from the agent should preferably be as minimal as possible.

Req 2.1	Multi-Agent Service	The agents shall be able to communicate with each other.
Req 2.2	Multi-Agent Service	Communication shall mimic the real-life juridic processes.
Req 2.3	Multi-Agent Service	The process duration should be as minimal as possible, but it should not impact the quality of the replay.
Req 2.4	Multi-Agent Service	The end result shall be sent via chat or email.
Req 2.5	Multi-Agent Service	The communication and end result shall be evaluated by a field expert.
Req 2.6	Multi-Agent Service	The multi-agent system should include traffic insurance advocates and judges.
Req 3.1	User Interface (UI)	Provided solution must be a web service capable of reaching a wide audience.
Req 3.2	User Interface (UI)	Interfaces should provide well thought and understandable access by providing a simple and official-looking design interface.
Req 3.3	User Interface (UI)	UI should enable a general logging and registration logic.
Req 3.4	User Interface (UI)	UI should provide account management.
Req 3.5	User Interface (UI)	The possibility to purchase services shall be included in the interface.
Req 3.6	User Interface (UI)	Communication with the agent ought to be intuitive like a chat mechanism.
Req 3.7	User Interface (UI)	Removing and copying conversations shall be available features in UI.

Req 3.8	User Interface (UI)	A quick test for guiding the customer in different insurance scenarios shall be included.
Req 4.1	Backend	The system shall act as a link between agents and users.
Req 4.2	Backend	Critical information should be stored, like email address, password, and credit balance. Also, communication between the agents shall be stored for the user to later recall them.
Req 4.3	Backend	The backend shall validate the traffic moving through it, for instance agent request length, authenticity of the user, and that the user has paid for the session with the agent.

Table 2: Table showcases functional requirements. It contains an ID that helps identify the requirement, a Component that describes which technical system the requirement is meant to, and a Requirement that is the set requirement for the artefact.

4.3 Non-Functional Requirements

ID	Goal	Requirement
Req 5.1	Availability	The service and agent should be available via different devices. In addition, the downtime should be minimized both from the service and agent side.
Req 5.2	Security	Ideally, the agent would be self-hosted, thus offering better control of the data, but not at the expense of reliability. Also, the service must be hosted in an environment that offers contemporary security and programming should follow best practices both in frontend and backend.

Req 5.3	Scalability	Both the service and the LLM agent should be in an environment that is modern and easily scalable.
---------	-------------	--

Table 3: Table showcases non-functional requirements. It contains an ID that helps identify the requirement and a Requirement that is the set non-functional requirement for the artefact.

Based on conversations within the team and research conducted in social media space, there is a demand for legal assistance within the domain of traffic insurance law. This support should be an affordable and trustworthy way to guide people within the legal space. Large language models and RAG techniques have shown promising capabilities when it comes to answering questions which rely heavily on a predetermined knowledge base. Thus, the capabilities of LLM and RAG are further explored within the traffic insurance legal space.

In the next chapter, the thesis will discuss the designing and developing of the artefact by respecting the requirements.

5 Design and Development

This chapter focuses on the process of generating the artefact based on the requirements that were set in the previous chapter. The procedure follows design and development practices to reach a solution for the underlying problem. The chapter kickstarts the project from a brainstorming session with the team members, where initial ideas for tackling the problem are produced. After, a common idea of the solution has been achieved the final design decisions are implemented as well as evaluated, and finally, the introduction of the initial architecture with its essential features and functionalities is presented.

5.1 Design and Develop Artefact

The activities when designing and developing the artefact consist of four sub-activities which are:

- Imagine and Brainstorm
- Assess and Select
- Sketch and Build
- Justify and Reflect [40]

The first activity "Imagine and Brainstorm" is where creative new concepts are created or existing ones are improved. The "Assess and Select" activity challenges these generated ideas from the Imagine and Brainstorm phase. It aims to improve and thin out the produced concepts. This is done via convergent thinking, for example, an analytical procedure for choosing out of possible concepts. After ideas have been assessed and chosen, the sub-activity continues to the "Sketch and Build" step. The phase strives to generate a plan which steers the creation of the artefact. The final sub-activity is "Justify and Reflect" where the designer reasons and justifies the design decisions they made and produces a design rationale. Documentation of the design rationale provides important insights and warning for future designers who might further develop the project. The four sub-activities mentioned above will be presented in greater detail in the following subchapters.

5.2 Imagine and Brainstorm

Imagine and Brainstorm is the starting step for the design and development where the team collectively creates potential ideas for the problem. The artefact was divided into two parts, one which mainly focused on the application and the other which focused on agent(s) design. During the brainstorming session, an initial architecture was crafted for the full system. The requirements for the agent were further specified for example, what law sources it should use. For the user interaction with the agent, the team thought that inspiration should be taken from already established products that offer

generative AIs, like OpenAI. However, for the multi-agent systems functionality, an email service would send the final processed decision to the user. A list of potential models was formed at the session. The requirement for the model was the capability to interpret Finnish. The group's insurance experts agreed to produce scenarios for the agent to answer and then evaluate the answers given by the agent. This way the developers are guided through the process of agent development.

5.3 Assess and Select

In the Assess and Select phase, designers should assess the solutions that were produced in the Imagine and Brainstorm sub-activity and select one or more as the ground for further design and development. When conducting the Assess and Select process with the team, the initial plan, that was developed during the Imagine and Brainstorm sub-activity, did not change radically. There were however some minor changes. Firstly, our model selection changed the focus to use GPT-4 Turbo and its embedding service, because this would provide a quicker way to achieve proof of concept. Also, it is due to the difficulties that a local model would potentially create, like strong hallucinations. This model selection was finalized when assessing the Finnish language capabilities of different models. The second feature selection was payment functionality, where the team decided to follow payment based on client requests. The reasoning for this was the simplicity of implementing it. The MAS litigation depended on the agents that the team could potentially create and test. Therefore the focus was more on the agent development than on the MAS design. Furthermore, the MAS would need more data about the processes and real-life cases. Otherwise, the chosen data sources, UI design, and systems architecture would stay as initially planned.

Bounded rationality stands for a decision-making process where decision-makers conclude the selection process and pick a solution adequate enough for solving the research problem [40]. Essentially by adopting this mentality, the goal is to be cost and time-efficient and select a satisfactory solution instead of seeking the most optimal one. The team followed this practice when making the assessments and selections. There would have been plenty of options to evaluate, however, the project needed to be launched so that set objectives would be reached and the overall concept could be better estimated.

5.4 Sketch and Build

This subsection focuses on explaining the Sketch and Build sub-activity, which is used to create a blueprint of the artefact out of the selected ideas from previous sub-activities. The sketch helps the designers to realize the overview of the artefact with its functionality and structure.

The sketching was started by forming a drawing (Figure 9) out of the high-level architecture from the brainstorming session. The architecture shows three services that are connected by APIs and one connected to a database. The presented design will act as a guiding point for the developers and will therefore further evolve when the development process advances.

SIPE Architecture

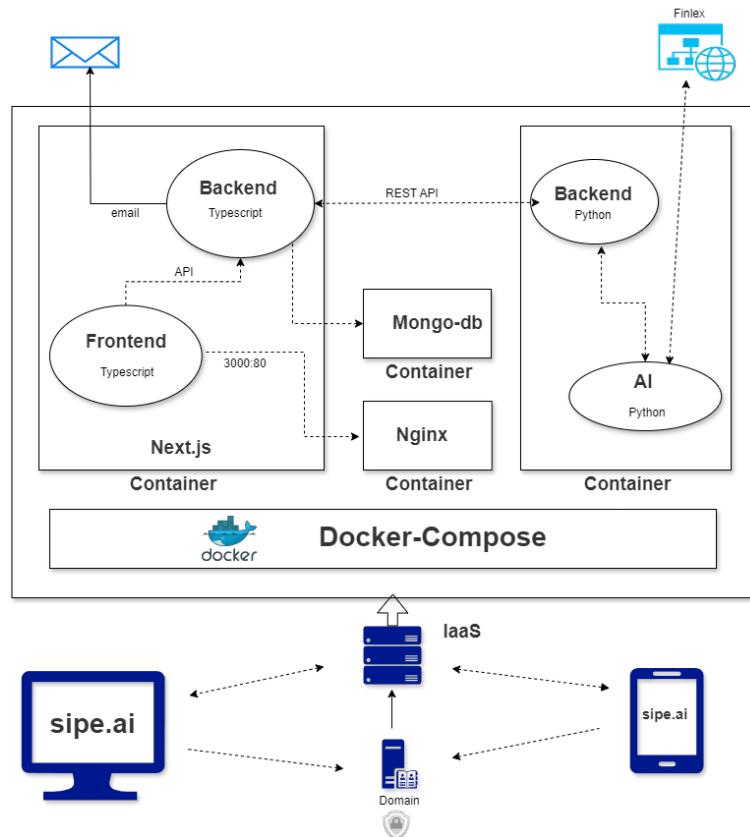


Figure 9: Sketch of the initial architecture

5.4.1 Walk-Through

As a team, when working on the project we practiced the Walk-Through method during the development process [40]. This method acts as a type of peer review, where one of the team members will lead for example a brainstorming or design session whilst other team members, like designers and other stakeholders, listen and have the option to give feedback throughout the process. Other team members thus have the chance to question the decisions or suggest better alternatives.

5.4.2 Sketching the Prototype

As previously described in 5.3 the Assess and Select phase consisted of mapping prior knowledge, research, and conducting minimal testing, whereas in this phase a blueprint was made, from which the prototype was built. The construction of the prototype consisted of the following steps.

Model Selection: The initial model will be selected based on the capability of understanding the Finnish language because the service strongly relies on Finnish law, which serves mainly the Finnish population. The promising

candidate for the model is GPT-4 Turbo, which is already usable via API. It is a very large model trained with huge amounts of data, and from prior experiments, it is capable of understanding the Finnish language. Other promising candidates that have been selected for evaluation are Mistral 7B, Llama 3 8B, Fingpt3, and Viking 7B. These options would, due to their relatively small size, offer a possibility for a self-hosted environment. The mentioned options are open-source models and can be scaled to larger ones if necessary. This solution would respect the non-functional requirement security (Req 5.2) by not releasing the data to third-party vendors. In this way, we can ensure data safety. Nevertheless, for the initial blueprint, the third-party model GPT-4 Turbo fulfills the proof of concept.

Capability of RAG: There were several tools for improving the model's knowledge by implementing RAG. For the sketching phase, two options were selected. The first option for RAG implementation is the LangChain library and the other is Embedchain.

Designing Agent: The agent should utilize one of the selected libraries and answer questions according to the given instructions. Every agent should have their own unique knowledge base and be a specialist in one certain task. Also, the agent should have memories of the conversations that it has disguised with other agents. This feature is important when developing the MAS. The chosen language will be Python and the API framework Django.

Utilizing Multi-Agent System: The MAS trial should follow real-life trial processes and use available sources when making decisions. These sources should contain law, existing processes, and judgments. Every agent should have their own assigned task in the overall setup. The chosen language is Python.

Frontend: The overall design for the interaction should use general web application functionalities like login/registration, account management, and a way to purchase the services provided. Also, it should provide chat as a mechanism to communicate with the expert agents. Inspiration for the chat should be taken from OpenAI. Interaction with the MAS should be done via a form that is presented in the web application where the client sends the overall case for processing and gets the reply via email. Another functionality that frontend offers is quick tests. The chosen language is JavaScript and the framework Next.js.

Backend: The backend will act as a middle manager that manages the user functionalities, conversations, payments, validation, and communication with the Django API.

5.5 Justify and Reflect

Justify and Reflect is the last sub-activity of Design and Develop Artefact which aims to justify the design decisions. The selection of the decisions is covered in the next

subsection where we present the prototype by describing the functionalities and their importance.

5.6 Design Decisions

Now that the requirements have been designed and partly created. In the Sketch and Build section, the thesis went through the development process of the requirements. However, the design decisions focus on the reasoning and justifications of the produced functionalities. The following subsections open more of the decisions made after building the functional requirements.

5.6.1 Design of Agent (Req 1.1 - 1.5, Req 5.2)

Selected Large Language Model (Req 1.1, Req 5.2) The agent applies GPT-4 Turbo from OpenAI to generate answers. The reasoning for this was that the model was able to communicate in Finnish and answer questions in an instruction-based manner. As mentioned previously, other language models were selected for evaluation and were therefore tested on how well they were able to generate Finnish text. These models were Mistral 7B from Mistral AI, Llama 3 8B from Meta, gpt3-finnish (gpt3-finnish-small and gpt3-finnish-medium) from TurkuNLP group and Viking 7B from Silo AI collaboration with TurkuNLP group. Mistral was the only model that did not have any Finnish language capabilities. However, this was predictable since it was trained to generate French, German, Spanish, and Italian. Llama3 however was able to produce the Finnish language despite its small size. The TurkuNLP group's Finnish language was a little bit broken when tested with simple questions, but these models were smaller than the other models to which they were compared. The potential model for further refinement could have been Viking 7B⁹. The model was able to produce the Finnish language and the development focus was to generate Nordic languages. However, it would still need fine-tuning. Nevertheless, the self-hosted option did not make it to the artefact but is still prototyped together with Llama 3.

Data sources (Req 1.2) The agent needs data that covers laws and processes associated with traffic insurance. These data sources can be found from different sites. The team needed an expert who would list all of the relevant sites he or she uses when working with different traffic insurance cases. Based on conversations with the expert a few main sites (Finlex¹⁰ and Liipo¹¹). were picked for the application to bring into service. These sites contain valuable information about the law and general knowledge about these procedures. It is worth mentioning that other sites were also used, but due to trade secrets - the company aims to be profitable and compete in the market - these other sites will not be disclosed.

⁹<https://huggingface.co/LumiOpen/Viking-7B>

¹⁰<https://finlex.fi/fi/>

¹¹<https://www.liipo.fi/>

Structure of the Agent (Req 1.3 - 1.4) The agent is created with Python language and the main tool that the agent utilizes is embedchain, which enables usage of RAG and communication with OpenAI's LLM models. Embedchain was selected over LangChain because it offers a relevant collection of features that is made easier to implement. The embedchain library utilizes LangChain libraries and reduces the number of lines necessary to run the agent. The structure of the agent includes the capability to fetch data when an agent object is constructed. The data that is to be fetched is described in a JSON file which contains all the necessary hyperlinks and data types e.g. PDF or HTML. The agent uses a customly configured YAML file which manages the functionality of embedchain. The agent has one method that is used to query answers using embedchain's query function. The communication with the agent is done via API powered by Django.

Missing Functionalities (Req 1.5) One agent was selected to serve as an expert because it provided acceptable answers at a tolerable speed. The chosen agent is an insurance advocate. We saw that instead of offering multiple different agents specialized in one subject we would instead offer one which is specialized in traffic insurance law. There were several reasons for the decision. Firstly, it is easier to focus on the quality of one agent, and also the initial goal was mainly focused on a law assistant who would be helping with traffic insurance cases. Another reason was the lack of real-world experts who could evaluate the other agents.

5.6.2 Multi-Agent System (Req 2.1 - 2.6)

The team decided to postpone the idea of having MAS processes surrounding insurance cases, due to the fact that having a mixture of agents takes a lot of time since they follow the judicial processes. This in turn would impact the wait time and consume much more resources because multiple agents would have to communicate with each other. Operating in these conditions would be challenging for a company in its initial state. Furthermore, as mentioned in the previous subsection, the lack of testers when using different expert agents would just generate scenarios that cannot be verified. Another obstacle was the lack of necessary data. To be able to process insurance law cases, one would need data about these cases. Based on the expert, this data would contain doctors' evaluation of the injuries, law references, and the final judgment. It was difficult to find these related cases even though the law itself is available to everyone. LawSampo had some events related to traffic insurance cases in storage, but they also lacked the doctor's evaluation of the patient.

5.6.3 Web Application Architecture (Req 3.1 - 4.3, Req 5.1, Req 5.3)

The web application delivers accessibility to any device that has a browser installed. It contains four services, one of which is the Next.js¹² server that runs the UI and works as a backend for managing all the necessary functionalities for the application.

¹²<https://nextjs.org/>

Nginx¹³ is used as a proxy service, and finally an agent API for communicating with the agent expert that serves via Django¹⁴. All of them are built and executed on top of docker¹⁵ and hosted in Google cloud¹⁶ virtual environments. The requirements Req 3.1 to Req 3.8 were achieved however, they will be further demonstrated in chapter 6.

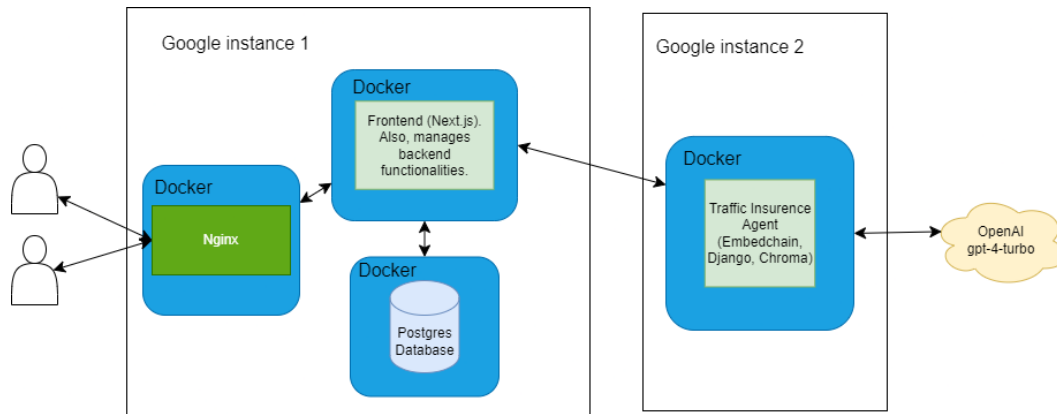


Figure 10: High-level architecture of the overall web application.

Furthermore, the services are hosted in two instances which are interconnected. Building the agent API reintegrates the given data sources to the vector database and uses OpenAI’s embedding service while doing the build, thus charging every time the agent API is built. Because frontend and backend are updated more often it made sense to separate these two services to their own environments. This way the build time and service cost can be reduced. Also, it enables us to scale the environment if needed.

The backend service manages the information requests that are passed to agent API by validating the request body, content length, user authenticity, and request cost compared to user balance. Also, Postgres¹⁷ relational database is used for storing user and conversation data. The structure of the database is presented in Figure 11. With the structure, the backend service can authenticate registered users and gain them access to the application. Transactions are also managed. The user can buy digital credit and use it for the chat service as mentioned previously. The backend keeps track of these transactions.

5.7 Agent as a Service

The agent serves as an expert and the expert uses decided data sources that have been verified by an actual expert as previously mentioned in this chapter.

The agent API service uses embedchain¹⁸ which in return calls the OpenAI’s GPT-4

¹³<https://nginx.org/en/>

¹⁴<https://www.djangoproject.com/>

¹⁵<https://www.docker.com/>

¹⁶<https://cloud.google.com/>

¹⁷<https://www.postgresql.org/>

¹⁸<https://embedchain.ai/>

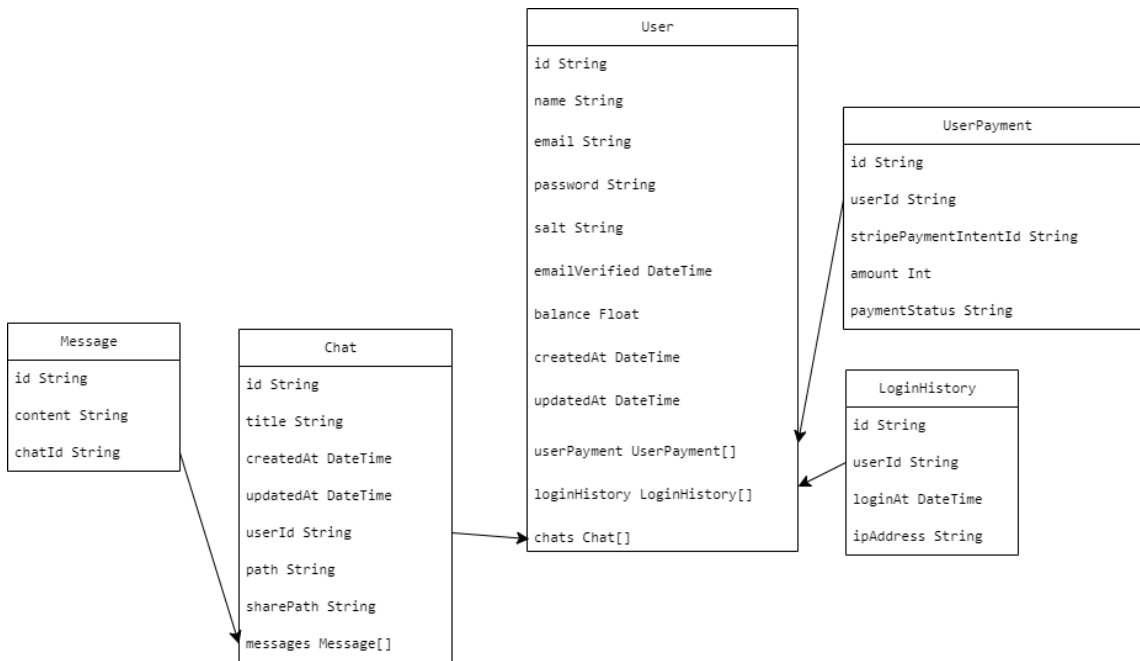


Figure 11: Database relation of the essential tables.

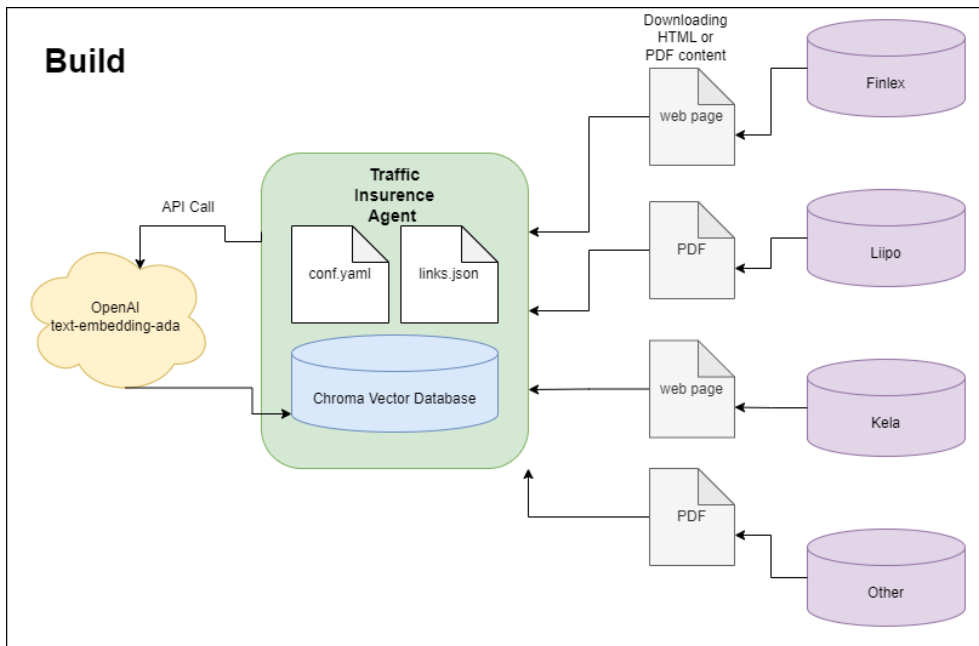


Figure 12: Agent in building phase.

Turbo via REST when the service is operating. However, when build is executed it fetches the relevant documents from multiple different sources as visualized in Figure 12. These documents are converted to vectors using an embedding service from OpenAI, as mentioned previously. Afterwards, the vectors are stored in a Chromin¹⁹

¹⁹<https://www.trychroma.com/>

vector database. All of the mentioned processes are achieved by using the embedchain.

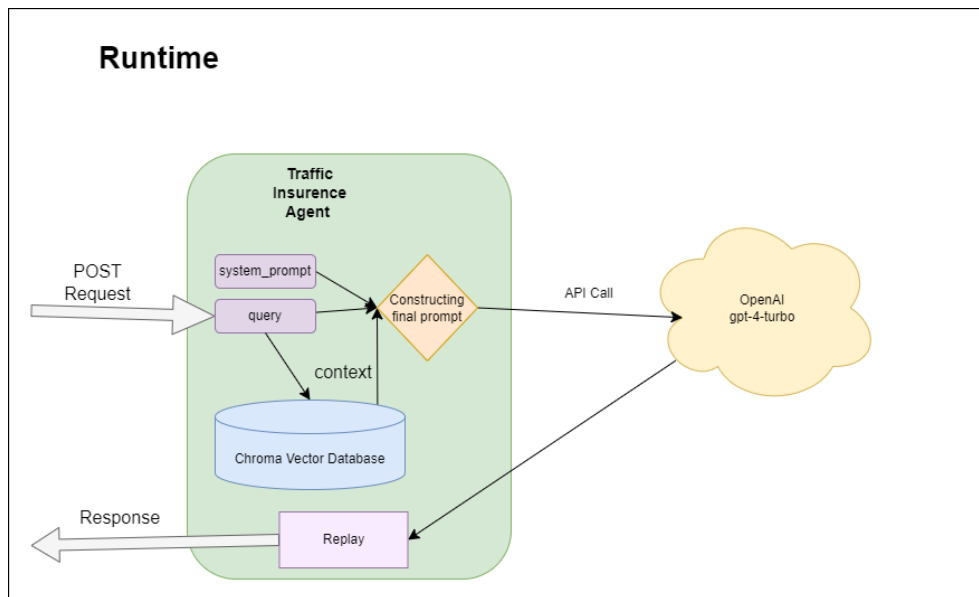


Figure 13: Agent in runtime

When the client interacts with the agent, the agent uses the relevant chunks from the documents. These chunks contain information about multiple laws and processes which are relevant when the agent makes suggestions for possible solutions for the questions related to different traffic accident scenarios. Thus it is important to understand the mechanism behind the agent service, hence in the following sub-sections, we will take a deep dive into the agent’s configuration and prompt construction.

5.7.1 Configuration of the Parameters

Embedchain offers easy usage when it comes to RAG. In the simplest way, one could just use three lines of code (assuming that only one source is used). However, the ready-made configuration was written in English and thus it only communicated in that language even when a Finnish prompt was given. For that reason, we decided to create a custom configuration.

Embedchain’s configure file (YAML).

```

1 app:
2   config:
3     name: 'sosiaaliturva-asiantuntia'
4
5
6 llm:
7   provider: openai
8   config:
9     model: 'gpt-4-turbo-preview'
  
```

```

10     temperature: 0.5
11     max_tokens: 1000
12     top_p: 1
13     stream: false
14     api_key: 'API-KEY'
15     prompt: |
16
17         Konteksti: $context
18
19         Kysymys: $query
20
21         Vastaus:
22
23     system_prompt: |
24
25         Olet sosiaaliturva-asiantuntija, jonka
26         erityisosaaminen on pitkäaikaissairaiden
27         JA vammaisten henkilöiden oikeudet.
28         Avustat sosiaaliturva-asioissa lyhyellä,
29         helposti ymmärretyllä vastauksella.
30
31 vectordb:
32     provider: chroma
33     config:
34         collection_name: 'sosiaaliturva-asiantuntia'
35         dir: db
36         allow_reset: true
37
38 embedder:
39     provider: openai
40     config:
41         model: 'text-embedding-ada-002'
42         api_key: 'API-KEY'
43
44
45 chunker:
46     chunk_size: 2000
47     chunk_overlap: 100
48     length_function: 'len'
49     min_chunk_size: 101
50
51 #cache:
52 #     similarity_evaluation:
53 #         strategy: distance
54 #         max_distance: 1.0

```

```
55 # config:
56 #     similarity_threshold: 0.8
57 #     auto_flush: 50
58
59
```

1. “App” section contains metadata that the user can specify. Also, some utilities can be activated like log level. However, only the name of the agent is specified.
2. The “llm” component is used to manage the large language models. One can select a model supplier via the "provider" element. There are multiple different options for provided models e.g. Mistral AI, NVIDIA AI, or Ollama if the model is hosted locally, to mention a few. For demonstration the selected provider was OpenAI.
3. Next part under “llm” is “config”, which has multiple sub-elements. The first one of the sub-elements is the model selection which is expressed in the “model” element and the chosen model is GPT-4 Turbo. Second, is the hyperparameter “temperature”. The value of the temperature has been set to 0.5. This enables the control of the randomness in the model’s output. The higher the temperature value is the more likely the model will hallucinate. The “max_tokens” limits the amount of tokens that are used in the response. At the current state, it has been set to 1000. For impacting the diversity of the words one can use “top_p”, which is set to 1, so that the model uses a more diverse set of words instead of replying with the most probable word. The “stream” parameter controls whether responses are sent in real-time or after full generation. For the time being it is set to false, however, there is a version where the project uses stream for more user-friendly interaction. The “prompt” parameter provides a type of form that is used when communicating with the model. The “system_prompt” sets the tone or role for the agent. Lastly, the “api_key” is essential for authorizing and managing access to the API, ensuring secure and monitored use of the service.
4. Section “vectordb” includes the “provider” and “config” elements that manage the operations for the vector database. ChromaDB²⁰ has been selected as the default vector database that the application uses. Data collection stores all the related data sources for usage of one agent. The application only utilizes a collection named “sosiaaliturva-asiantuntija”. The directory where the database is generated is “db”. The “allow_reset” element indicates whether the resetting of the vector database is allowed or not.
5. The “embedder” section details information about the embedding model the application uses, managed similarly through “provider” and “config” elements. OpenAI is the designated provider for the embedder. The model being utilized

²⁰<https://www.trychroma.com/>

is 'text-embedding-ada-002'. An API key is required to access this model, due to safety reasons the value is not inserted in the "api_key" element.

6. The "chunker" section outlines parameters that govern how text is divided into chunks for processing by the language model. The following sub-elements of chunker are: "chunk_size", "chunk_overlap", "length_function" and "min_chunk_size". It specifies that each chunk of text sent to the model has a maximum size of 2 000 characters, with an overlap of 100 characters between them. The length of each chunk is calculated using the "len" function. Additionally, each chunk must be at least 150 characters long, which ensures the chunk size is less than 2 000 characters while also being greater than the 100-character overlap.
7. "Cache" part was added but not used. Instead it was commented out. Reasons for this were that whenever the questions were similar to each other, instead of processing the given question normally, it responded with the same replay that was previously processed or it mixed original context with another context. For example, the model addressed "Paavo" as "Pirkko" because their cases were similar. Due to security reasons the application cant allow sensitive data to be swapped between users, therefore this functionality was removed.

5.7.2 Prompt Construction

One of the most impactful phases in the project is the construction of the prompt. The configuration of the prompt contains two parameters "system_prompt" and "prompt" which when constructed can be viewed as follows:

Parameter	Value
system_prompt	"Olet sosiaaliturva-asiantuntija, jonka erityisosaaminen on pitkäaikaissairaiden JA vammaisten henkilöiden oikeudet. Avustat sosiaaliturva-asioissa lyhyellä, helposti ymmärrettävällä vastauksella."

prompt (context)	<p>Konteksti: "35 Pysyvän haitan korvaus voidaan kuitenkin maksaa jatkuvana haittaluokassa 20, kun kysymyksessä on pysyvästi tajuton vammautunut tai pysyvästi laitoshoidossa oleva vaikean aivovamman saanut vanhus (liikennevahinkolautakunnan kiertokirjeet 5/1988 ja 14/1999). Tämän lisäksi korvaus voidaan maksaa jatkuvana henkilölle, joka on vaikean aivovamman seurauksena pysyvästi laitoshoidossa ja jonka sekä kommunikoinnissa että liikuntakyvyssä on pysyviä, vaikea -asteisia rajoitteita. Ennakkokorvaukset Vakuutuslaitoksen on aloitettava korvausasian selvittäminen viipymättä ja suoritettava korvaus tai ilmoitettava, ettei korvausta suoriteta, joutuisasti ja viimeistään kuukauden kuluttua siitä, kun se on saanut vakuutuslain 69 §:ssä tarkoitetut asiakirjat ja tiedot ja muun riittävän selvityksen. Vaikka korvauksen määrä ei olisi riidaton, vakuutuslaitos on kuitenkin velvollinen suorittamaan korvauksen riidattoman osan. Vakuutuslaitoksen ennakkokorvauspäätöksessä tulee yksilöidä, mihin haittoihin ja vammoihin korvaus perustuu ja minkä suuruiseksi haittaluokka on arvioitu. Lopullisena korvauksena suoritetaan ennakkokorvauksena suoritettua haittaluokan ja lopullisen haittaluokan erotus lopullisen korvauksen suoritusvuoden tasossa. Jos ennakkokorvaus on poikkeuksellisesti maksettu yksilöimättä haittaluokkaa, ennakkokorvaus on vähennettävä nimellismääräisenä lopullisesta korvauksesta. 4.1.6 Liikennevahinko sattunut 1.1.2006 tai sen jälkeen Korvaus yli 18- vuotiailla henkilöillä Korvaus määritetään arvioimalla ensin vammoista aiheutuva haittaluokka. Sen jälkeen määritetään haittaluokkaa vastaava peruskorvaus. Kun peruskorvauksen määrä kerrotaan ikäkertoimella, saadaan varsinainen maksettava korvaus. Jokainen eletty ikävuosi 18 vuoden jälkeen pienentää korvausmäärää 75 vuoden ikään saakka. Peruskorvauksen määrä lasketaan 18- vuotiaan henkilön perusmäärää lähtökohdaksi pitäen. Korvausmäärä lasketaan siten, että vahinkoa kärsineen ikänä pidetään hänen vahingon sattumishetken ikäänsä. . . "</p>
------------------	---

prompt (query)	<p>Kysymys: "15-vuotias koululainen, Tuukka ajoi mopollaan ojaan ja sai onnettomuudessa vaikean aivovamman. Tuukka käytti kypärää mutta se meni onnettomuudessa rikki. Tuukka oli sairaalassa 6 kuukautta, jonka jälkeen kotiutui. Hän tarvitsi apua kaikissa päivittäisissä toiminnoissa. Hän ei voinut palata kouluun. Tuukka todettiin täysin työkyvyttömäksi. Mitä liikennevaluutuslain ja vammaispalvelulain mukaisia oikeuksia Tuukalla on?"</p> <p>Vastaus:</p>
----------------	--

Table 4: Prompt construction

When the “system_prompt” and “prompt” are given in Finnish the model understands that its replay should also be in Finnish. This was the main reason why custom configuration was taken in use in the first place.

The first quoted sentence describes the role of the agent. This value is always included as a starting point for any conversation. With the sentence the agent acts more like an expert who has expertise from traffic insurance cases.

When the identity of the agent has been defined via “system_prompt”, it’s time to include the “prompt” parameter which can be observed from the second quoted sentence. Two variables are introduced which are “context” and “query”. From the constructed prompt the “context” variable is presented after the Finnish word “Konteksti:” and “query” after “Kysymys:”. Where “context” is the retrieved information and “query” is the follow up question which has been sent from the client’s side. In the constructed prompt the variables are replaced with actual query sent by clients and context retrieved from the vector database. However, the context is a shortened version of the originally retrieved content because otherwise it would take too much space. Finally, the prompt ends with “Vastaus:”, the word can be translated to “answer” which gives an indication to the Large Language Model to generate the answer.

6 Application Demonstration

In this chapter, the thesis takes a deeper look at the artefact whose functionalities are demonstrated in the following sub-section. The demonstration includes the Demonstrate artefact practice from DSMR. Furthermore, the functionalities consist of a general overview of the user interface features and agent interaction via UI. This chapter also presents the latest state of the overall architecture.

6.1 Demonstrate Artefact

The fourth step of the DSRM framework involves The Demonstrate artefact, which displays the usability of the artefact in real-life scenarios [40], thus showing the usefulness of the solution. The Demonstrate artefact tries to answer the following question:

“How can the developed artefact be used to address the explicated problem in one case?”

This framework can be divided into two sub-activities which are picking and producing a case where the artefact is used and applying the application to the selected case with documented end result. The adopted example can be fictitious, well-documented, real-life, or a combination of all mentioned cases. These activities can be held as a frail way of evaluation, hence it is important that the case where the artefact is used is well defined.

6.2 Application Overview

This section will briefly present the user interface design. The application was initially built separately by the team in their local environment following the walk-through collaboration. After interaction was achieved the application development moved to Google Cloud testing ground for further development. The overview takes place in the testing ground because it is the most developed version of what has been collectively accomplished.

There are four views for interaction. First is login and register which enables the user to create an account or login to one. Login and registration follow the widely adapted practice where the user creates an account by entering his or her email address to which a verification link is then sent. After verification, the user selects a password that will be used when logging in (Figure 19).

The second view is the landing page (Figure 18) which offers a few buttons: A quick test for verifying that the user has rights for compensation and access to chat, which is the main utility of the service. All of the views, except the login and registration page, contain a header on top. The header includes account management and log-out options and it also displays how much balance the user has in its digital wallet. Account management holds information about the user and a function for purchasing more credit to the digital wallet.

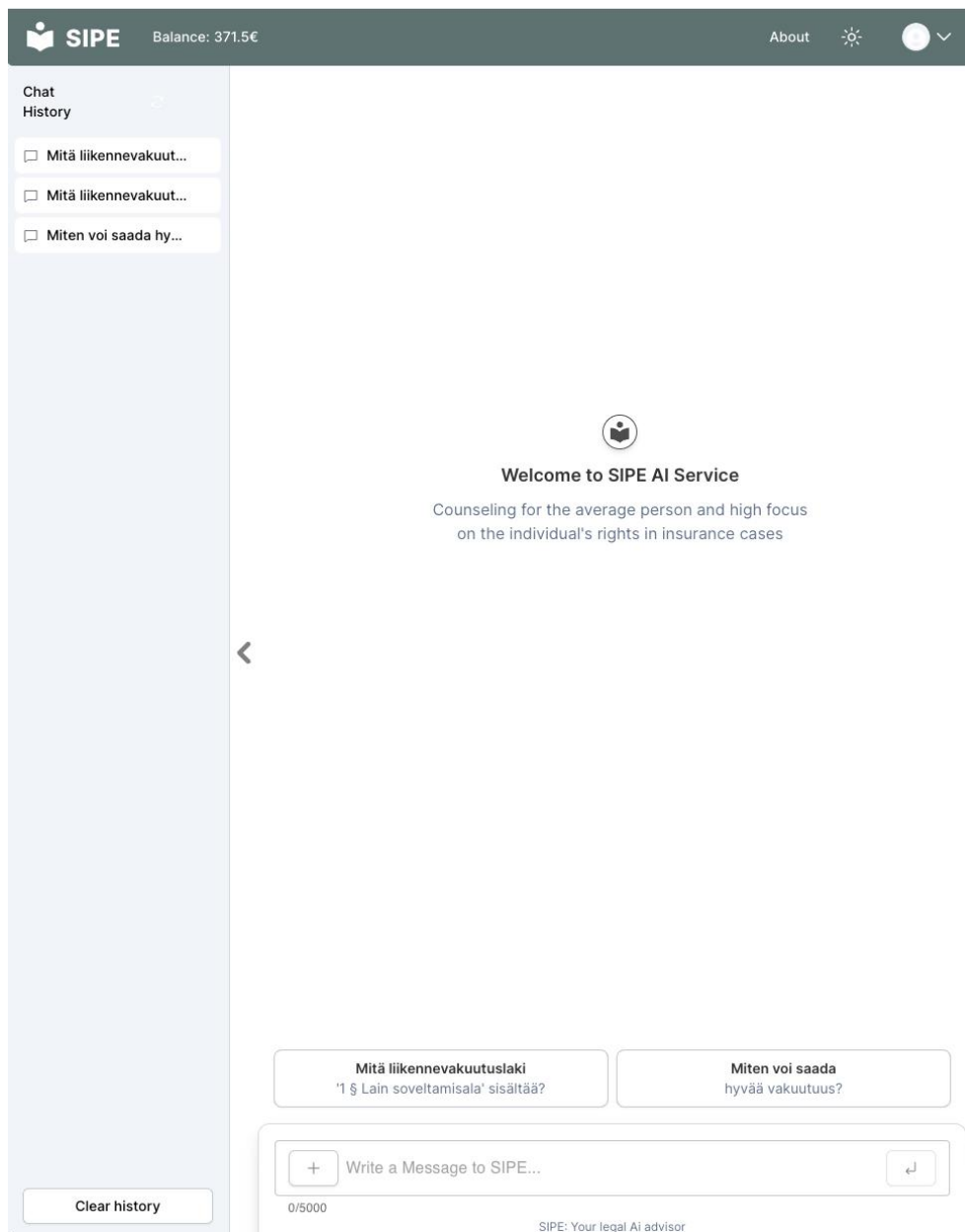


Figure 14: User interface of the chat functionality in the web application.

The third view is a quick test page (Figure 20) which can be accessed via the landing page. The test is used to provide basic knowledge about who to contact if certain conditions are met. Furthermore, it validates the expert agent's usefulness. This way the potential customer knows if it should buy the service or not. If the right conditions are achieved the test takes the customer to the main service. The user can choose to skip the test and go directly to chat with the expert from the landing page.

The interaction with the expert happens via chat which is shown in Figure 14. Inspiration for the design of the view has been taken from OpenAI. Also, many of the features which OpenAI provides are utilized as well. For example the possibility

of copying experts' answers, saving the conversation, and starting a new chat with the expert. OpenAI's ChatGPT has used a similar design and because it has a wide amount of usage similar adaptations have been utilized in other LLM applications. It therefore provides a sense of familiarity for those who interact with LLM models.

7 Evaluation

The following chapter introduces the evaluation activity from the DSRM framework and continues to explain how the applications main functionality was evaluated. Later in the chapter the differentiation between parametric memory only and retrieval-augmented generation is evaluated. After that, other LLMs with the help of RAG are compared against each other. Finally the discussion theorizes about the evaluated factors.

7.1 Evaluate Artefact

The final activity utilized in this paper is Evaluate Artefact [40]. This activity estimates how good the artefact is in solving the addressed problem and how well it attains the requirements. In other words, the Evaluate Artefact phase tries to answer the following question:

“How well does the artefact solve the explicated problem and fulfil the defined requirements?”

It is also important to set goals for the evaluation process. The goals for the artefact can be summarized into:

- testing the artefact against the problem;
- estimating how well the artefact satisfies the functional and non-functional requirements;
- reflecting on its fundamental theories and implementation principle;
- exploring the possibility of unintentional repercussions; and
- discovering areas for enhancement through additional design efforts.

The conducted evaluation practised ex ante evaluation where the artefact is estimated while the design and development are still ongoing.

7.2 Evaluation of Agent by Human-Feedback

The evaluation process started with the critical feature which was the agent as a service. The team had access to experts from the field of insurance justice. One of the team members volunteered to create imaginary scenarios based on real-life situations that the expert had witnessed over the years when working in the field. The evaluator also knew the ideal answers to the questions. These questions were given as input for the agent to answer, after which the evaluator would give a grade between zero and ten, where zero stands for worst outcome and ten stands for optimal answer. The description would be given if the answer's grade would be under ten. The table 5 presents the evaluated results of the query.

7.2.1 Agent with Parametric and Non-Parametric Memory (GPT-4 turbo)

The presented results were conducted by using RAG with GPT-4 Turbo which used the selected data from 5.5. The results in general were better than expected as can be observed from table 5. Out of the ten scenarios, nine out of ten were given full points and one scenario (table ID 1) was given a score of seven out of ten. The lower grading was caused due to the fact that the agent should have mentioned that the insurance company should compensate the pension from the accident, adding tools and aid setup for the home. Furthermore, the agent hallucinated that “Tuukka” would be treated from bed even though there was no implication about it. Another hallucination was that the agent made up a concept where Tuukka would have rights to discounts from traffic insurance payments, however, there isn’t any discount of this sort. Finally, the most critical right that was not listed was that Tuukka should have the right to personal assistance.

It is worth mentioning that the first tests conducted with the expert were not documented; they were mainly conducted to gain more insight into the functionality, however, they received similar scores.

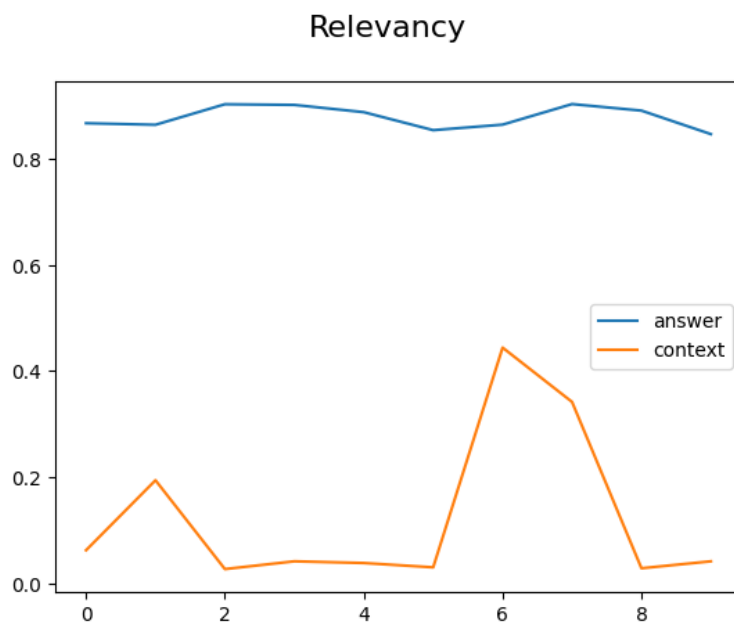


Figure 15: Context relevancy and answer relevancy. X-axis displays the query ID and Y-axis show the score.

Other metrics that were used when estimating were the answer and context relevance presented in Figure 15. The presented scores were evaluated via embedchain’s²¹ evaluation method. The answer relevance compares the relevance between the question and the answer. Where the context relevance determines the relevance of the context and question. The table shows that the score for context relevance ranges from 2% to 44% which is not optimal. This means that the retrieved context is not well aligned

²¹<https://docs.embedchain.ai/components/evaluation>

with the question which might suggest that the non-parametric approach has little impact on the end result. However, the answer relevance performed much better, extending from 84% to 90%. This indicates that the grading from experts correlates with the answer's relevancy.

Next, the paper will evaluate the effect of using just parametric memory against a few identical test cases.

7.2.2 Agent with Parametric Memory (GPT-4 turbo)

It is important to evaluate the impact of the RAG in general and see how much of the knowledge is actually from the parametric memory since many of the tests were able to score full points from the expert.

The results that were provided by just GPT4, show that the answers differ but the formation stays the same. After evaluating four questions from the table 6. The expert mentioned that the model generated concepts that don't exist and generally the answers presented weaker results than the ones that utilized RAG. The initial evaluation indicates that without RAG the model tends to hallucinate more.

The overall conclusion is that using RAG mitigates the risk of hallucination also in Finnish law, thus it is the correct feature to use in the future as well.

7.3 General Evaluation of Open-Source Models

Even though the self-hosted model is not part of the application it can still be evaluated manually. These estimations can give more insights into what model to pick for further development. The method for evaluating these models is similar as in the 7.2.2 chapter. Several random queries are selected and the responses are compared to the answers which were able to gain full grades.

A quantized model was constructed for running it on a local computer. The Viking model was not able to answer the questions directly, however, this was mentioned on their hugging face page. The model is a base model and would need further fine-tuning for most of the tasks. Prompt engineering techniques, like in-context learning, were tested, but no good answer was produced. Instead, one answer was often repeated over and over again. The model would need fine-tuning to be more like an instruction-based model. The quantification could also have affected the response slightly.

The Llama 3 8b model was tested next. The model was available for testing in several remote sites. So for evaluation of the model, there was no need to host it locally or in a cloud environment. Similarly, the model was not able to produce anything valuable and would require more effort by fine-tuning the model and better prompt engineering. It performed better than Viking 7 B and was able to make more coherent answers. Nevertheless, when compared to GPT-4 Turbo it was only able to produce a few similar answers and at some point, it started to hallucinate.

7.4 Reliability of Human-Feedback

Based on the results from an expert one could argue that the results showed exceptionally good performance. However, the first four tests from the table were evaluated in person when they were generated. This gave little time for the expert to further analyze, which could impact the grade. Compared to the other six tests which were analyzed remotely without any time limit or pressure. Another complication is the amount of tests and expert evaluators. It would strengthen the evaluation if there would be peer reviews also in the evaluation process where discussion about the answers would take place between the experts and would reach a common grade through dialog. Also, if there were more evaluators there could be more test cases. Ten test cases does not cover the whole entirety, nor the most common scenarios there could potentially be. Thus, the agent should be thought of as a guiding assistant and not as an actual expert from the field. Further, evaluation should be conducted either by similar manners or by conducting an “ex post evaluation” where the artefact is evaluated when it is implemented to use [40]. This could be carried out by publishing the early prototype for limited usage for the clients to get acquainted with the prototype, its limitations and to be able to estimate the given answers whilst using the prototype in their organizations.

8 Discussion

The following chapter discusses the limitations and future direction of the project. Furthermore, the project's initial concept and general discussion about the multi-agent system as a service is also covered within this chapter.

8.1 Generalization

The study developed an agent that used RAG technology combined with the GPT-4 Turbo model which is accessible via web application to help users gain accessibility to traffic insurance law. The human feedback evaluation by an expert revealed that the system could generate satisfactory answers and was able to reduce hallucinations. However, it was found that the context relevance metric was low, indicating that there is a need for improvements in document retrieval.

The findings suggest that utilizing RAG techniques in traffic insurance law offers more accurate results compared to just employing an LLM. Future refinement should focus on enhancing the retrieval process and extending the human feedback evaluation to improve the reliability of the answers. Moreover, implementing a multi-agent system could scope the user's necessities, thus providing more relevant responses.

8.2 Resource limitations

The limitations of the project can be culminated in three aspects. The first limitation was the lack of time since the general objective for the company was to produce a prototype. It took the majority of the time to create the groundwork so therefore a lot of time couldn't be focused on researching and prototyping the agent. This would have provided us with the capability to further research the company's own model.

The second obstacle was the lack of data. One of the initial challenges when trying to develop a simulated set of expertise was the unavailability of data. Many of the scenarios would need a doctor's analysis of the injuries caused in the traffic accidents and how they relate to the accident, this data would be used when debating the compensations.

The third difficulty was the lack of expertise. The limitations of the experts challenged the testing and generation of multiple agents. Part of the challenge with MAS as a service was the lack of experts that could have widened the knowledge sources for different agents that could have potentially played different roles in insurance cases. Also, the process of generating one agent could have needed more experts to cross-validate the responses and increase the amount of test cases.

8.3 Related Works

In this sub-section, the project is compared against other related research and shown how they differ and what ideas could be implemented in this project.

Even though RAG has inspired the development of several tools like Embedchain and LangChain for easy implementation of RAG, the concept of utilizing it in the

legal domain specifically in the Finnish legal field seems to be relatively new. Only a small amount of papers were found about the topic. Additionally, they differ in terms of technologies used in these projects and from a legal subject point of view.

A study about LLM-based assistance in GDPR implementation used a similar setup with a slightly different library [41]. The library in question was LangChain which was also considered for this project. The demonstrated system managed to handle beginner and intermediate legal questions well when evaluated in a qualitative manner. However, these evaluated questions were asked in English which might have a more positive impact on the generative process. Furthermore, the presented system employed multiple agents for various tasks depending on the question's complexity. Only one agent was used for this thesis project due to its relatively simple questions. Still, inspiration for employing multiple agents for different tasks looks promising when observing the results from agent-based systems.

As mentioned in Chapter 2.3.2 one of the researches conducted for housing disputes did not show promising evaluation from an expert [36]. However, in that research, an older model was used for the generative task which might explain why this paper got conflicting results when compared.

Another research with a similar goal to provide legal support for tax issues found that three out of eight questions were useful based on expert evaluation [42]. The paper suggests utilizing metadata filtering within the RAG framework for producing better quality and more precise responses.

Based on the human-feedback evaluation, our system was able to generate adequate responses when employed with embedchain, GPT-4 Turbo, and a hand-picked selection of legal context stored in ChromaDB vector database. However, similar difficulties with retrieving relevant documents correlate with the mentioned studies in this subsection.

8.4 Future Development

The following sub-section discusses the future research and developments of the application. It deliberates around previously set requirements and lessons learned during the design phase. Potential improvements are presented as well. By reflecting on the outcomes of this research, new possible directions for the project are explored.

8.4.1 Scalability of the Agent

The created logic for the agent supports different types of sources and configurations that enable the scalability of different agents with other expertise. Ideally, other experts in different fields of insurance law can be included in the architecture and they would be running in their own containers and API endpoints. Several agents were created during the project but many of them served one objective, to help in traffic insurance cases which would have been used mostly in multi-agent systems. Nevertheless, to create other agents to serve in law one would need experts of that knowledge base to pinpoint the material they use on a daily basis and to validate the agent's capabilities.

8.4.2 Multi-Agent System as a Service

The MAS still relies on multiple experts willing to create test scenarios and validate the credibility's of the results. It would also consume more processing power than a simple query. For a startup company, this would consume a lot of the already limited resources. During the brainstorming session, several ideas were explored like consulting customized MAS solutions for private or public organizations to help them with law processes. This would offer a real-life scenario where the logic could be integrated and could potentially give access to experts in the organization to help with the testing.

Another concept was a graphical interface for creating visual MAS. This framework would allow users to easily create agents with their own purpose and configuration which would then be communicating with each other, thus creating a pipeline for the input to be processed. The pipeline is abstract and is defined by the user by connecting agent nodes with edges. The possibility of using RAG for each agent was also planned. The setup would then be converted to Python code and hosted as a service. One could say that the framework would be low-code but for MAS processes. The solution could potentially have offered an agile way to create customized legal processes and an accessible way for experts to create their own setups without needing to understand coding. Consequently, some initial sketches and prototypes were created with both mentioned ideas but were ultimately postponed due to the fact that they shifted the focus too much from the original scope.

The most realistic way to implement MAS architecture would be to introduce assistants to the expert agent. This assistant agent could be hosted in a smaller model like Llama 8B or 22B for small talk in order to get more details about the client's incident and then summarize the conversation and pass it along to the already available expert agent. This could potentially reduce the cost and offer a more private approach as the summarized case would be anonymized.

8.4.3 Improving the RAG

The RAG showed weak results when it was evaluated by the retrieval context. Thus this factor needs improving so that the LLM model has more reliable references. One way to achieve this is to implement a technique called metadata filtering. The idea behind metadata filtering is that the stored data sources would get descriptive keywords e.g. time, categories, or something more abstract. In theory, this should improve the context relevance.

Embedchain offers a parameter for storing these metadata for the function which is responsible for fetching the data from a remote or local source. This might however need real-life experts to assist when the keywords are generated or a more experimental approach with an LLM to generate keywords that would suit the content.

Because the designing process concluded that one agent would be enough to assist the client. Much of the data sources were allocated to one agent responsible for storing, retrieving, and forming responses. So instead of storing all in one vector database, the data could be divided into multiple different databases with their own categories. For

example, one could hold traffic insurance law and another the general processes. This way the search space could be narrowed.

8.4.4 Self-Hosted over Third-Party LLM

An important goal for the startup is to host its own LLM model(s) to increase privacy and potentially decrease the cost. Ideally, the model would be small so that it wouldn't consume too much processing power, but efficient enough to give similar results as the current design with an external model. Still, the roadmap for achieving it needs further research and development.

As mentioned previously one way to achieve this could be by improving the RAG so that the model would have more accurate context to rely on. Another way could be to fine-tune the parametric memory of the model to better remember the law. Moreover, the model could be fine-tuned to handle instructions in higher quality by feeding synthetic response data from bigger models like GPT-4 Turbo.

These were several concepts that could be further researched.

9 Conclusion

The rise of LLMs has gained popularity in recent years and shaped the way people interact with information. It is no exception within the legal domain. The thesis dives into traffic insurance law with the goal of providing intelligent legal consulting by answering users' questions by retrieving relevant law and generating top of the retrieved content a suitable answer. The research follows design science practices for achieving the artefact. Where the study explores different potential ideas for initial problems until the final solution has been recognized and developed.

The state-of-the-art prototype was accomplished by introducing an agent that specialized in retrieval-augmented generation. The agent utilizes embedchain for RAG operations, custom prompts, and selected legal content. Where RAG process consists of communication between ChromaDB and OpenAI's GPT-4 Turbo model. A web application was developed around the agent for user inference.

The artefact was able to generate satisfactory level responses based on the evaluation conducted by an expert from the legal field. Unfortunately, the context relevance performed poorly when evaluated with embedchain. However, this study found that using the RAG technique instead of a pure LLM model, reduced the hallucination and thus improved the answers quality.

This research emphasizes how metadata filtering could refine context relevancy, thus increasing the answer quality. Furthermore, it speculates how employing another agent to assist the expert agent could further improve the quality as the agent would summarize conversations with the client. The assisting agent could use a self-hosted model to increase the privacy of the application by anonymizing personal information via prompt engineering.

Thus future research will focus on improving retrieval capabilities and implementing a multi-agent system which was researched in this thesis but never implemented in the state-of-the-art prototype.

Overall, the study offers important insight into how to enhance the availability of traffic insurance law with the help of an expert from that field and by utilizing the RAG technique.

References

- [1] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, “A comprehensive overview of large language models,” *arXiv preprint arXiv:2307.06435*, 2023.
- [2] J. Roberts, “How powerful are decoder-only transformer neural models?” *arXiv preprint arXiv:2305.17026*, 2023.
- [3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [4] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, “The llama 3 herd of models,” *arXiv preprint arXiv:2407.21783*, 2024.
- [5] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, “Measuring massive multitask language understanding,” *arXiv preprint arXiv:2009.03300*, 2020.
- [6] R. Luukkonen, J. Burdge, E. Zosa, A. Talman, V. Komulainen, V. Hatanpää, P. Sarlin, and S. Pyysalo, “Poro 34b and the blessing of multilinguality,” *arXiv preprint arXiv:2404.01856*, 2024.
- [7] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.
- [8] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, “A survey of quantization methods for efficient neural network inference,” in *Low-Power Computer Vision*. Chapman and Hall/CRC, 2022, pp. 291–326.
- [9] I. Frîncu, “In search of the perfect prompt,” *Aalto University*, 2023.
- [10] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, “Large language models are zero-shot reasoners,” *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.
- [11] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.
- [12] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

- [13] I. Ma, A. K. Martins, and C. V. Lopes, “Integrating ai tutors in a programming course,” *arXiv preprint arXiv:2407.15718*, 2024.
- [14] T. Boccato, M. Ferrante, and N. Toschi, “Two-phase rag-based chatbot for italian funding application assistance,” 2024.
- [15] P. Finardi, L. Avila, R. Castaldoni, P. Gengo, C. Larcher, M. Piau, P. Costa, and V. Caridá, “The chronicles of rag: The retriever, the chunk and the generator,” *arXiv preprint arXiv:2401.07883*, 2024.
- [16] T. Mikolov, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.
- [17] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” *arXiv preprint arXiv:1908.10084*, 2019.
- [18] O. Ram, Y. Levine, I. Dalmedigos, D. Muhlgay, A. Shashua, K. Leyton-Brown, and Y. Shoham, “In-context retrieval-augmented language models,” *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 1316–1331, 2023.
- [19] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, “Ragas: Automated evaluation of retrieval augmented generation,” *arXiv preprint arXiv:2309.15217*, 2023.
- [20] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, “Reflexion: Language agents with verbal reinforcement learning,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [21] J. S. Park, J. O’Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, “Generative agents: Interactive simulacra of human behavior,” in *Proceedings of the 36th annual acm symposium on user interface software and technology*, 2023, pp. 1–22.
- [22] T. Liang, Z. He, W. Jiao, X. Wang, Y. Wang, R. Wang, Y. Yang, Z. Tu, and S. Shi, “Encouraging divergent thinking in large language models through multi-agent debate,” *arXiv preprint arXiv:2305.19118*, 2023.
- [23] C. Qian, X. Cong, C. Yang, W. Chen, Y. Su, J. Xu, Z. Liu, and M. Sun, “Communicative agents for software development,” *arXiv preprint arXiv:2307.07924*, 2023.
- [24] E. Hyvönen, M. Tamper, A. Oksanen, E. Ikkala, S. Sarsa, J. Tuominen, and A. Hietanen, “Lawsampo: A semantic portal on a linked open data service for finnish legislation and case law,” *The Semantic Web: ESWC*, pp. 110–114, 2020.
- [25] M. Frosterus, J. Tuominen, and E. Hyvönen, “Facilitating re-use of legal data in applications—finnish law as a linked open data service,” in *Legal Knowledge and Information Systems*. IOS Press, 2014, pp. 115–124.

- [26] A. Oksanen, M. Tamper, J. Tuominen, E. Mäkelä, A. Hietanen, and E. Hyvönen, “Semantic finlex: Transforming, publishing, and using finnish legislation and case law as linked open data on the web,” in *Knowledge of the Law in the Big Data Age*. IOS Press, 2019, pp. 212–228.
- [27] J. Ramos *et al.*, “Using tf-idf to determine word relevance in document queries,” in *Proceedings of the first instructional conference on machine learning*, vol. 242, no. 1. Citeseer, 2003, pp. 29–48.
- [28] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.
- [29] X. Rong, “word2vec parameter learning explained,” *arXiv preprint arXiv:1411.2738*, 2014.
- [30] Q. Le and T. Mikolov, “Distributed representations of sentences and documents,” in *International conference on machine learning*. PMLR, 2014, pp. 1188–1196.
- [31] S. Sarsa *et al.*, “Information retrieval with finnish case law embeddings,” *University of Helsinki, Department of Computer Science*, 2019.
- [32] F. Rahutomo, T. Kitasuka, M. Aritsugi *et al.*, “Semantic cosine similarity,” in *The 7th international student conference on advanced science and technology ICAST*, vol. 4, no. 1. University of Seoul South Korea, 2012, p. 1.
- [33] J. Savelka and K. D. Ashley, “The unreasonable effectiveness of large language models in zero-shot semantic annotation of legal texts,” *Frontiers in Artificial Intelligence*, vol. 6, p. 1279794, 2023.
- [34] L. Derczynski, “Complementarity, f-score, and nlp evaluation,” in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, 2016, pp. 261–266.
- [35] L. Martin, N. Whitehouse, S. Yiu, L. Catterson, and R. Perera, “Better call gpt, comparing large language models against lawyers,” *arXiv preprint arXiv:2401.16212*, 2024.
- [36] M. I. Rafat, “Ai-powered legal virtual assistant: Utilizing rag-optimized llm for housing dispute resolution in finland.” *Haaga-Helia*, 2024.
- [37] A. Koubaa, “Gpt-4 vs. gpt-3.5: A concise showdown,” 2023.
- [38] S. Masoudnia and R. Ebrahimpour, “Mixture of experts: a literature survey,” *Artificial Intelligence Review*, vol. 42, pp. 275–293, 2014.
- [39] J. Cui, Z. Li, Y. Yan, B. Chen, and L. Yuan, “Chatlaw: Open-source legal large language model with integrated external knowledge bases,” *arXiv preprint arXiv:2306.16092*, 2023.

- [40] P. Johannesson and E. Perjons, *An introduction to design science*. Springer, 2014, vol. 10.
- [41] M. E. Mamalis, E. Kalampokis, F. Fitsilis, G. Theodorakopoulos, and K. Tarabanis, “A large language model based legal assistant for governance applications,” *OSF*, 2024.
- [42] T. Roberts, “Utilizing generative artificial intelligence for tax and legal consultancy: Design science approach,” *Helsingin yliopisto*, 2024.

A Appendix

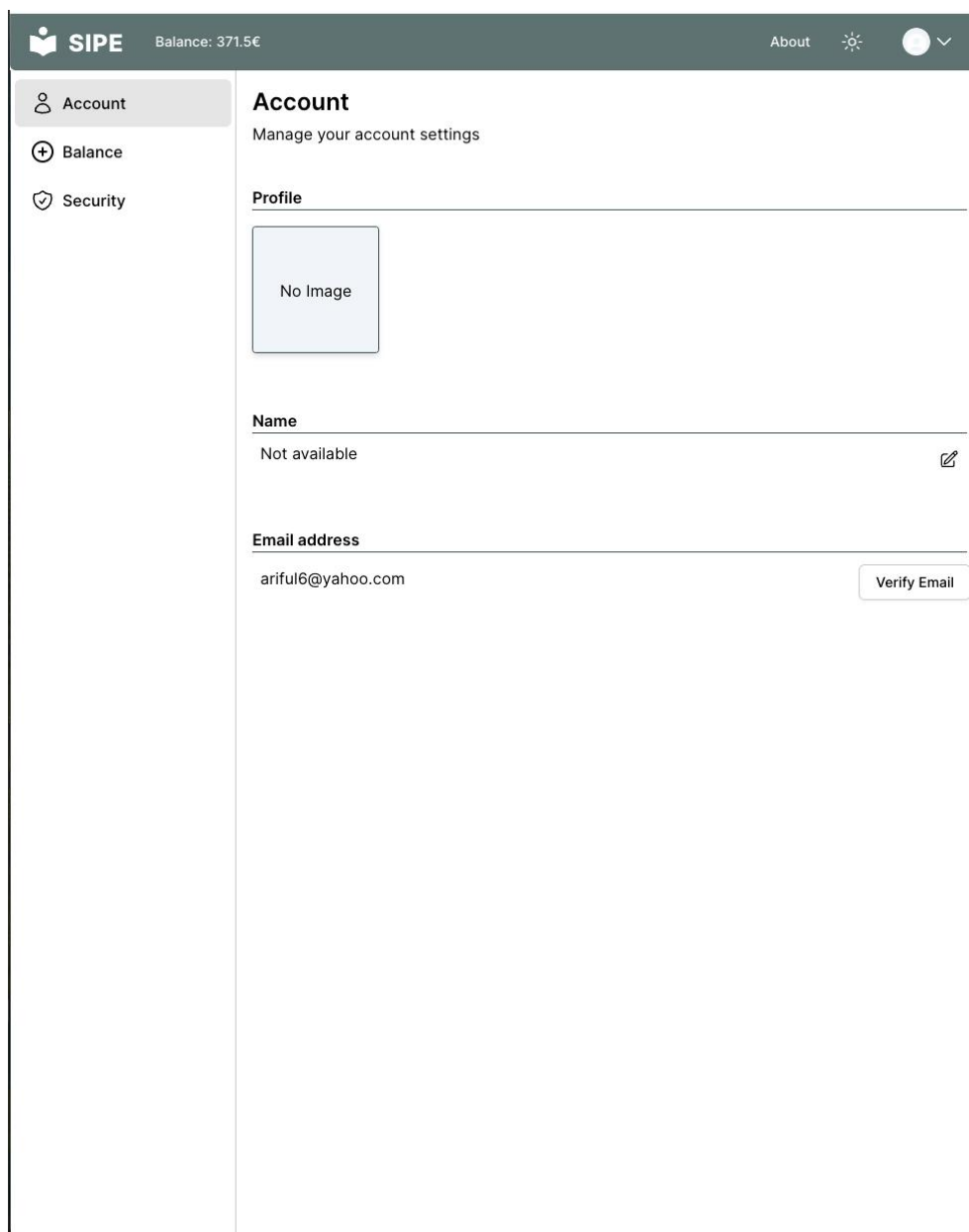


Figure 16: User interface of the account management functionality in the web application.

ID	Question	Des	Answer	Grade
----	----------	-----	--------	-------

1	<p>15-vuotias koululainen, Tuukka ajoi mopollaan ojaan ja sai onnettomuudessa vaikean aivovamman. Tuukka käytti kypärää mutta se meni onnettomuudessa rikki. Tuukka oli sairaalassa 6 kuukautta, jonka jälkeen kotiutui. Hän tarvitsi apua kaikissa päivittäisissä toiminnoissa. Hän ei voinut palata kouluun. Tuukka todettiin täysin työkyvyttömäksi. Mitä liikennevaluutuslain ja vammaispalvelulain mukaisia oikeuksia Tuukalla on?</p>	<p>Myöhemmin hänellä tulee vielä oikeus tapaturmaeläkkeeseen joka olisi vastannut hänen mahdollisen ammattinsa palkkaa, mutta sitä ei tässä vaiheessa tarvitse vielä välttämättä informoida. Perheen kodin muutostyöt tulee vakuutusyhtiön myös kustantaa ja kaikki apuvälineet. Pisteet 8/10. (AI keksi, että Tuukka on vuoteeseen hoidettava, vaikka sitä ei sanottu. AI oletti näin.) Sillä ei ole varsinaisesti merkitystä, koska oikeudet ovat samat. Huomasin, että AI vastasi, että perheellä on oikeus liikennevakuutusmaksun alennukseen. Sellaista alennusta ei ole olemassa. On auton verovapaus uutta autoa ostaessa, ja myös ajoneuvon käyttöveron vapautus kun on invaparki oikeus. AI en infonnut oikeudesta henkilökohtaiseen apuun mikä on iso juttu. Pisteet alenee 7/10. Kielellisesti vastaus oli aivan täydellinen. Tuukan perheen tulee ottaa yhteyttä omaan vakuutusyhtiönsä. Ei tarvitse ottaa yhteyttä liikennevakuutuskeskukseen, kun vakuutusyhtiö on tiedossa.</p>	<p>Tuukalla on useita oikeuksia liikennevakuutuslain ja vammaispalvelulain mukaisesti hänen tilanteessaan: 1. Hoitotuki: Tuukka on oikeutettu I luokan hoitotukeen, koska hän tarvitsee apua kaikissa päivittäisissä toiminnoissa ja on vuoteeseen hoidettava. 2. Vaatelisä: Tuukka voi saada vaatelisän, koska hän saattaa tarvita erityisiä apuvälineitä, jotka kuluttavat vaatteita tavanomaista enemmän. 3. Muutostyöt autoon: Jos Tuukan perheellä on tarve hankkia auto Tuukan kuljettamiseen, he voivat saada korvauksia välttämättömiin muutostöihin autossa, kuten ajohallintalaitteet ja pyörätuolin nostolaitteet. 4. Invalidialennus liikennevakuutuksesta: Jos Tuukan perhe hankkii auton, he voivat olla oikeutettuja alennukseen liikennevakuutusmaksuista. 5. Vammaisen pysäköintilupa: Tuukka voi saada vammaisen pysäköintiluvan, mikä helpottaa hänen saattamistaan ja kuljettamistaan. 6. Kuljetuspalvelut: Tuukka on oikeutettu kuljetuspalveluihin kunnan sosiaalitoimesta, koska hän ei voi käyttää julkisia liikennevälineitä. Näitä palveluita voidaan käyttää esimerkiksi asiointiin, harrastuksiin ja virkistystoimintaan. 7. Koulukuljetus: Koska Tuukka ei kykene käyttämään julkista liikennettä ja on todettu täysin työkyvyttömäksi, hän on oikeutettu koulutoimen järjestämiin koulukuljetuksiin tai vastaavaan avustukseen, mikäli hän pystyy palaamaan kouluun tai vastaavaan oppimisympäristöön tulevaisuudessa. On tärkeää, että Tuukan perhe ottaa yhteyttä paikalliseen sosiaalitoimeen, koulutoimeen ja Liikennevakuutuskeskukseen saadakseen tarkempaa tietoa ja ohjeistusta näiden tukien hakemisessa.</p>	7
---	---	---	--	---

2	<p>35- vuotias Matti oli liikenneonnettomuudessa ja menehtyi onnettomuuspaikalla. Hän oli naimisissa ja hän oli kahden alaikäisen lapsen isä. Mitä korvauksia liikennevaluutus yhtiön tulee korvata?</p>	X	<p>Liikennevakuutuksesta korvataan lähiomaisille seuraavat korvaukset:</p> <ol style="list-style-type: none"> 1. **Hautauskulut:** Kulut korvataan kohtuulliseen määrään asti. 2. **Henkilövahingon perusturva:** Kuolemantapauksessa maksetaan perussumma lähiomaisille, yleensä leskelle ja alaikäisille lapsille. 3. **Elatuskorvaus:** Jos vainaja on elättänyt perhettään, voivat lähiomaiset saada korvausta menetetyistä elatuksesta. Elatuskorvauksen määrä ja kesto ovat tapauskohtaisia ja niitä laskettaessa otetaan huomioon vainajan tulot, perheen muut tulot ja muut vastaavat tekijät. On tärkeää ottaa yhteyttä liikennevakuutusyhtiöön mahdollisimman pian onnettomuuden jälkeen, jotta korvausprosessi voidaan aloittaa. Vakuutusyhtiö auttaa dokumenttien keräämisessä ja tarvittavien lomakkeiden täyttämässä. 	10
---	--	---	---	----

3	<p>Työtön, 60-vuotias Pirkko jäi suojatiellä auton töytäisemäksi. Hänen jalkansa murtui ja hän oli sairaalahoitossa 6 päivää. Pirkon housut ja takki menivät rikki onnettomuudessa. Pirkko asui yksin ja vammansa takia hän tarvitsi kodinhoidollista apua 3 kertaa viikossa. Pirkko sai sairauslomaa 4 kuukautta. Hänen aikaisempi tulonsa oli Kelan työmarkkinatuki. Mitä korvauksia Pirkko voi liikennevakuutuksesta saada?</p>	X	<p>Pirkko voi liikennevakuutuksesta saada seuraavia korvauksia:</p> <ol style="list-style-type: none"> 1. Sairauskulut: Kaikki onnettomuudesta aiheutuneet lääkärin- ja sairaalakulut korvataan täysimääräisesti. 2. Vaatteiden korvaus: Onnettomuudessa vahingoittuneet vaatteet korvataan niiden uusimisarvon mukaan. 3. Kotitalousapu: Mikäli Pirkko tarvitsee kotitalousapua vammansa takia, hänellä on oikeus saada korvaus kodin siivouksesta ja muista välttämättömistä kotitöistä. 4. Tulonmenetykskorvaus: Työmarkkinatuen saajana Pirkko voi saada ansionmenetykskorvausta, jos onnettomuus vaikuttaa hänen kykyynsä palata työmarkkinoille. 5. Päiväraha: Pirkko voi saada päiväkorvausta sairausloma-ajalta, mikäli se ylittää oman omavastuuajan. <p>Näiden korvausten saamiseksi Pirkon tulee ottaa yhteyttä liikennevakuutuksen tarjoajaan ja toimittaa tarvittavat todistukset ja lääkärinlausunnot.</p>	10
---	--	---	--	----

4	<p>70-vuotias eläkeläinen, Paavo jäi suojatiellä auton töytäisemäksi. Hänellä rikkoutui silmälasit. Paavo vietiin ambulanssilla sairaalaan, jossa lääkäri totesi, että Paavolle oli onnettomuudessa tullut ruhjeita, jotka paranisivat viikossa. Mitä korvauksia Paavo voi auton liikenevakuutuksesta saada?</p>	<p>kysymyksessä tilapäinen haitta oli liian lyhyt korvaukseen, mutta vastauksessa lukikin, että voi saada korvausta. Mahdollistahan on, että haitta olisikin vähän pitempi..</p>	<p>Paavo voi hakea korvauksia seuraavista: 1. Sairaanhoidokulut: Ambulanssikuljetus ja sairaalassa annettu hoito. 2. Rikkoutuneet silmälasit: Korvausta voidaan hakea silmälasien korjaamisesta tai uusimisesta aiheutuneisiin kustannuksiin. 3. Haittakorvaus: Vaikka Paavon ruhjeet paranevatkin viikossa, jos onnettomuudesta on aiheutunut hänelle tilapäistä haittaa, hän voi olla oikeutettu haittakorvaukseen. Paavon tulee ottaa yhteyttä onnettomuuden aiheuttaneen ajoneuvon liikennevakuutuksen myöntäneeseen vakuutusyhtiöön korvausten hakemiseksi. On tärkeää toimittaa kaikki tarvittavat todisteet, kuten lääkärinlausunto ja kuitit mahdollisista kuluista.</p>	10
---	--	--	--	----

5	<p>75-vuotias, eläkeläinen, Kyllikki jäi suojatiellä auton alle. Hänet vietiin ambulanssilla ensiapuun, jossa lääkäri totesi, että Kyllikki oli saanut ruhjeita, joiden paraneminen kestäisi 4 viikkoa. Kyllikki tarvitsi kotonaan viikottaista siivous ja kauppa-apua. Mitä liikennevakuutuslain mukaisia oikeuksia Kyllikki voi saada?</p>	X	<p>Kyllikki voi saada liikennevakuutuslain mukaisesti korvauksia tapahtuneen onnettomuuden johdosta. Nämä korvaukset voivat sisältää: 1. **Sairaanhoitokulut**: Kaikki Kyllikin onnettomuuden johdosta tarvitsemat lääketieteelliset hoitokulut, kuten lääkärissä käynnit, lääkkeet ja mahdolliset jatkohoidot, korvataan liikennevakuutuksesta. 2. **Tilapäinen haitta**: Kyllikin toipumisaikana, joka lääkärin mukaan kestää noin 4 viikkoa, hän voi saada korvausta tilapäisestä haitasta, mikä kattaa yleensä fyysisen ja henkisen kärsimyksen. 3. **Kotitalousapu**: Koska Kyllikki tarvitsee kotonaan apua siivouksessa ja kaupassa käynnissä, hän voi saada korvausta kotitalousavun järjestämiseen. Tämä tarkoittaa, että liikennevakuutuksesta voidaan korvata Kyllikille tarpeellisen avun kustannukset, jotta hän pystyy selviytymään päivittäisistä toiminnoistaan onnettomuuden aiheuttaman toimintakyvyn heikkenemisen vuoksi. Kyllikin tulee olla yhteydessä onnettomuuden aiheuttaneen ajoneuvon liikennevakuutuksen tarjoajaan korvausten hakemiseksi. On suositeltavaa, että hän kerää kaikki onnettomuuteen liittyvät asiakirjat, kuten lääkärinlausunnot ja kuitit mahdollisista kuluista, jotta korvausprosessi sujuisi mahdollisimman mutkattomasti.</p>	10
---	--	---	--	----

6	<p>14 -vuotias yläastetta käyvä koululainen, Liisa jäi suojatiellä auton alle ja sai vaikean aivovamman. Hän käyttää pyörätuolia ja hän tarvitsee toisen henkilön apua kaikissa arjen toiminnoissaan. Hän ei voi koskaan jatkaa opintojaan ja on täysin työkyvytön. Mitä liikennevakuutus voi korvata hänelle?</p>	X	<p>Liikennevakuutus voi korvata Liisalle seuraavia asioita liittyen hänen vaikeaan aivovammaansa ja sen seurauksiin: 1. **Sairaanhoitokulut:** Kaikki Liisan vamman hoitoon liittyvät kustannukset, kuten sairaalassaolo, lääkkeet, terapiat ja muut hoitoon liittyvät menot. 2. **Henkilökohtainen apu:** Jos Liisa tarvitsee henkilökohtaista apua arjen toiminnoissaan, liikennevakuutus voi korvata tarvittavien avustajien kustannukset. 3. **Kuntoutus:** Liikennevakuutus kattaa tarvittavan kuntoutuksen, joka voi sisältää esimerkiksi fysioterapiaa, toimintaterapiaa ja muuta vamman vaatimaa kuntoutusta. 4. **Muutostyöt:** Liisan kotiin ja mahdollisesti autoon tarvittavat muutostyöt, jotka ovat välttämättömiä hänen liikkumisensa ja toimintakyönsä tukemiseksi. 5. **Pysyvän haitan korvaus:** Koska Liisa on täysin työkyvytön ja hänen elämänlaatunsa on merkittävästi heikentynyt, hän voi saada korvausta pysyvästä haitasta. 6. **Eläke:** Jos Liisa on työkyvytön tapahtuman seurauksena, hän voi olla oikeutettu liikennevakuutuksesta maksettavaan eläkkeeseen, joka kattaa osan menetetyistä ansiotuloista. 7. **Muut taloudelliset menetykset:** Liikennevakuutus voi myös korvata muita Liisan tai hänen perheensä taloudellisia menetyksiä, jotka liittyvät vammaan ja sen aiheuttamiin muutoksiin elämässä. On tärkeää olla yhteydessä vakuutusyhtiöön mahdollisimman pian onnettomuuden jälkeen, jotta kaikki korvaukseen liittyvät asiat saadaan käsiteltyä asianmukaisesti.</p>	10
---	--	---	--	----

7	Mitä vammais- palvelulain mukaisia oikeuksia hänellä on?	X	<p>Liisalla on useita oikeuksia vammais- palvelulain mukaisesti, jotka tukevat hänen arkeaan ja elämänlaatuaan. Näitä ovat: 1. **Henkilökohtainen apu:** Tämä kattaa avun arjen toiminnoissa, kuten henkilöko- htaisessa hygieniassa, aterioinnissa ja li- ikkumisessa. 2. **Kuljetuspalvelut:** Li- isa on oikeutettu kuljetuspalveluihin, jotka mahdollistavat hänen osallistumisensa har- rastuksiin, asiointiin ja muuhun virkistys- toimintaan. Nämä palvelut eivät kata terveyden- ja sairaanhoidon matkoja, mutta hänelle voidaan tarjota muita järjeste- lyjä näihin tarpeisiin. 3. **Vammaisen pysäköintilupa:** Tämä helpottaa Liisan ja hänen avustajiensa liikkumista, mah- dollistaen paremmat parkkipaikat lähellä määränpäitä. 4. **Muutostyöt asun- toon:** Jos Liisan asunto vaatii muu- tostöitä hänen vammansa vuoksi, kuten es- teettömän kulkureitin tai erityisvarustelun kylpyhuoneeseen, hän voi saada tukea näihin. 5. **Apuvälineet:** Liisa on oikeutettu tarvitsemiinsa apuvälineisiin, kuten pyörätuoliin ja mahdollisesti muihin laitteisiin, jotka helpottavat hänen arkeaan. 6. **Koulutus ja opetus:** Vaikka Liisa ei voi jatkaa opintojaan tavallisessa muo- dossa, hänellä on oikeus saada tukea ja mahdollisuuksia koulutukseen ja opetuk- seen, jotka soveltuvat hänen tilanteeseensa. 7. **Koulukuljetus:** Vaikka Liisa ei voi jatkaa opintojaan tavallisen opetussu- unnitelman mukaan, jos hän pystyy os- allistumaan johonkin muotoon koulutuk- sesta, hänellä on oikeus koulukuljetukseen. Jokainen palvelu ja tuki arvioidaan yksilöl- lisesti Liisan tarpeiden mukaan, ja niiden tavoitteena on mahdollistaa mahdollisim- man itsenäinen ja laadukas elämä hänelle.</p>	10
---	---	---	---	----

8	<p>35- vuotias Matti oli liikenneonnettomuudessa ja menehtyi onnettomuuspaikalla. Hän oli naimisissa ja hän oli kahden alaikäisen lapsen isä.</p> <p>Mitä korvauksia liikennevaluutus yhtiön tulee korvata?</p>	X	<p>Liikennevakuutusyhtiön tulee korvata seuraavat: 1. Kulukorvaukset: Mattiin liittyvät hautajaiskustannukset ja muut mahdolliset kulut, jotka ovat aiheutuneet hänen kuolemansa johdosta. 2. Elatuksen menetyksen korvaukset: Koska Matti oli perheen elättäjä, hänen puolisonsa ja alaikäiset lapsensa ovat oikeutettuja elatuksen menetyksen korvauksiin. Tämä tarkoittaa taloudellista tukea, joka korvaa Mattin tulon menetyksen perheelle. 3. Henkilövahinkojen korvaukset: Vaikka Matti menehtyi, on mahdollista, että liikennevakuutusyhtiö korvaa myös henkilövahinkoihin liittyviä korvauksia, kuten kärsimyskorvauksen. Kaikki korvaukset riippuvat vakuutusehdoista ja tapauksen yksityiskohdista. Lisäksi, jos Mattiin sovelletaan myötävaikutusalennusta (esim. jos hän olisi ollut osasyllinen onnettomuuteen), korvauksia voidaan alentaa. Korvausten tarkka määrä ja laatu määritellään tapauskohtaisesti.</p>	10
---	---	---	--	----

9	<p>40- vuotias yksin asuva työssä käyvä hammaslääkäri, Pirkko, oli liikenne onnettomuudessa, minkä seurauksena hänen oikean kätensä peukalo amputoitiin. Sairasulomaa oli 3 kuukautta, minkä jälkeen hän ei voinut enää palata omaan ammattiinsa. Mitä korvauksia liikennevakuutus korvaa?</p>	X	<p>Pirkolle liikennevakuutuksesta korvataan seuraavia: 1. **Sairaanhoitokustannukset**: Kaikki onnettomuudesta aiheutuneet lääkärikäynnit, leikkaukset ja muut hoitotoimenpiteet. 2. **Ansiomenetyt**: Koska Pirkko ei pystynyt työskentelemään sairausloman aikana, hänelle korvataan menetetty ansiotulo. 3. **Kipu ja särky sekä tilapäinen haitta**: Korvauksia maksetaan myös fyysisestä ja mahdollisesti henkisestä kärsimyksestä, jonka onnettomuus on aiheuttanut. 4. **Pysyvä haitta**: Pirkon peukalon menetys ja ammatinharjoittamisen estyminen oikeuttavat hänet pysyvän haitan korvaukseen. 5. **Kuntoutus**: Hänen kuntoutuksensa kustannukset korvataan, jonka tavoitteena on parantaa hänen työ- ja toimintakykyään ja auttaa häntä mahdollisesti löytämään uusi ammatti. Lisäksi, jos Pirkolla on muita vakuutuksia, hänellä voi olla oikeus lisäkorvauksiin näistä vakuutuksista.</p>	10
10	<p>37- vuotias työssä käyvä sähkömies, Mikko oli liikenne onnettomuudessa, jossa hänen jalkansa murtui. Mikko oli autoa ajaessaan 1,8 promillen humalassa. Sairausloma oli 3 kuukautta. Jalan toimintakyky ei palautunut ennalleen, vaan hänelle jäi lievä pysyvä haitta. Mitä korvauksia Mikko voi saada?</p>	X	<p>Mikko voi saada seuraavia korvauksia: 1. **Ansionmenetykskorvaus**: Kuntoutuksen ajalta eli sairausloman aikana, perustuen onnettomuushetken työansioihin. 2. **Haittaraha**: Pysyvästä lievästä haitasta, jonka suuruus määräytyy sosiaali- ja terveysministeriön asetuksen mukaisen haittaluokituksen perusteella. 3. **Mahdolliset muut korvaukset**: Riippuen vahingon laadusta ja vakuutuksista, kuten esimerkiksi hoitokulut tai muut erityiskulut, jotka liittyvät onnettomuuden seurauksena syntyneisiin tarpeisiin. On kuitenkin huomioitava, että Mikon humalataila onnettomuushetkellä voi vaikuttaa korvausten määrään tai korvauskelpoisuuteen, riippuen vakuutusehdoista ja sovellettavasta lainsäädännöstä.</p>	10

Table 5: Evaluation results for RAG generated (includes GPT-4 Turbo) responses from expert. It contains identifier, experts created question, feedback from expert, generated answer and grade given by the expert (0 - 10)

ID	Question	Answer	Experts comment
1	<p>35- vuotias Matti oli liikenneonnettomuudessa ja menehtyi onnettomuuspaikalla. Hän oli naimisissa ja hän oli kahden alaikäisen lapsen isä. Mitä korvauksia liikennevaluutus yhtiön tulee korvata?</p>	<p>Liikennevakuutuksesta korvataan lähiomaisille seuraavat korvaukset:</p> <ol style="list-style-type: none"> 1. Hautauskulut: Kulut korvataan kohtuulliseen määrään asti. 2. Henkilövahingon perusturva: Kuolemantapauksessa maksetaan perussumma lähiomaisille, yleensä leskelle ja alaikäisille lapsille. 3. Elatuskorvaus: Jos vainaja on elättänyt perhettään, voivat lähiomaiset saada korvausta menetetyistä elatuksesta. Elatuskorvauksen määrä ja kesto ovat tapauskohtaisia ja niitä laskettaessa otetaan huomioon vainajan tulot, perheen muut tulot ja muut vastaavat tekijät. <p>On tärkeää ottaa yhteyttä liikennevakuutusyhtiöön mahdollisimman pian onnettomuuden jälkeen, jotta korvausprosessi voidaan aloittaa. Vakuutusyhtiö auttaa dokumenttien keräämisessä ja tarvittavien lomakkeiden täyttämässä.</p>	<p>1, kysymyksessä on henkilövahingon perusturva sana mitä ei löydy mistään, mutta omaisilla voi olla oikeus korvauksiin, joten sisältö on oikea. Elatus korvaus on elatuksen menetyksestä maksettava korvaus. AI on keksinyt sanoja, kun lakitekstissä niitä ei ole!</p>

2	<p>Työtön, 60-vuotias Pirkko jäi suojatiellä auton töytäisemäksi. Hänen jalkansa murtui ja hän oli sairaalahoitossa 6 päivää. Pirkon housut ja takki menivät rikki onnettomuudessa. Pirkko asui yksin ja vammansa takia hän tarvitsi kodinhoidollista apua 3 kertaa viikossa. Pirkko sai sairauslomaa 4 kuukautta. Hänen aikaisempi tulonsa oli Kelan työmarkkinatuki. Mitä korvauksia Pirkko voi liikennevakuutuksesta saada?</p>	<p>Pirkko voi liikennevakuutuksesta saada seuraavia korvauksia:</p> <ol style="list-style-type: none"> 1. **Sairauskulut**: Kaikki onnettomuudesta aiheutuneet lääkärin- ja sairaalakulut korvataan täysimääräisesti. 2. **Vaatteiden korvaus**: Onnettomuudessa vahingoittuneet vaatteet korvataan niiden uusimisarvon mukaan. 3. **Kotitalousapu**: Mikäli Pirkko tarvitsee kotitalousapua vammansa takia, hänellä on oikeus saada korvaus kodin siivouksesta ja muista välttämättömistä kotitöistä. 4. **Tulonmenetykskorvaus**: Työmarkkinatuen saajana Pirkko voi saada ansionmenetykskorvausta, jos onnettomuus vaikuttaa hänen kykyynsä palata työmarkkinoille. 5. **Päiväraha**: Pirkko voi saada päiväkorvausta sairausloma-ajalta, mikäli se ylittää oman omavastuuajan. <p>Näiden korvausten saamiseksi Pirkon tulee ottaa yhteyttä liikennevakuutuksen tarjoajaan ja toimittaa tarvittavat todistukset ja lääkärinlausunnot.</p>	<p>Ensimmäiset versiot olivat parempia ja oikeampia, siis 2, ja 3 vastaukset. Näissä uusissa oli käytetty eri sanoja. Ansionmenetykskorvaus on oikea sana 2 vastaukseen. Ei tulonmenetykskorvaus ja päiväraha! (niin kuin aiemmassa vastauksessa. 3:ssa oli myös otettu huomioon tilapäinen haitta, niin kuin pitääkin.</p>
---	--	--	---

3	<p>75-vuotias, eläkeläinen, Kyllikki jäi suojatiellä auton alle. Hänet vietiin ambulanssilla ensiapuun, jossa lääkäri totesi, että Kyllikki oli saanut ruhjeita, joiden paraneminen kestäisi 4 viikkoa. Kyllikki tarvitsi kotonaan viikottaista siivous ja kauppa-apua. Mitä liikennevakuutuslain mukaisia oikeuksia Kyllikki voi saada?</p>	<p>Liikennevakuutuslain mukaan henkilö, joka loukkaantuu liikenneonnettomuudessa Suomessa, on oikeutettu korvauksiin liikennevakuutuksesta. Kyllikki voi saada korvausta seuraavasti:</p> <ol style="list-style-type: none"> 1. Sairaanhoitokulut: Kaikki Kyllikin hoitoon liittyvät kulut, kuten lääkärikäynnit, lääkkeet ja muut hoitotoimenpiteet, korvataan liikennevakuutuksesta. 2. Tulonmenetys: Jos Kyllikki saisi työtuloa, hän voisi saada korvausta myös menetetyistä ansiotuloista. Koska hän on eläkeläinen, tämä kohta ei välttämättä koske häntä, mutta on mainitsemisen arvoisen. 3. Kotitalousapu: Koska Kyllikki tarvitsee onnettomuuden johdosta apua kotonaan, kuten siivous- ja kauppa-apua, hän voi saada korvauksen näistä kuluista. Liikennevakuutus kattaa yleensä tämän tyyppiset avun tarpeet, jos ne ovat lääketieteellisesti perusteltuja ja niitä tarvitaan onnettomuuden seurauksena. <p>Kyllikin tulee ottaa yhteyttä onnettomuuden aiheuttaneen ajoneuvon liikennevakuutusalan tarjoajaan korvausprosessin aloittamiseksi. On suositeltavaa, että hän kerää kaikki lääketieteelliset dokumentit ja kuitit kulujen todistamiseksi. Lisäksi Kyllikki voi tarvittaessa pyytää apua tai neuvoa esimerkiksi oman kuntansa sosiaalipalveluista tai oikeusaputoimistosta.</p>	<p>Ensimmäiset versiot olivat parempia ja oikeampia, siis 2, ja 3 vastaukset. Näissä uusissa oli käytetty eri sanoja. Ansionmenetykskorvaus on oikea sana 2 vastaukseen. Ei tulonmenetykskorvaus ja päiväraha! (niin kuin aiemmassa vastauksessa. 3:ssa oli myös otettu huomioon tilapäinen haitta, niin kuin pitääkin.</p>
---	--	---	---

4	<p>14 -vuotias yläastetta käyvä koululainen, Liisa jäi suojatiellä auton alle ja sai vaikean aivovamman. Hän käyttää pyörätuolia ja hän tarvitsee toisen henkilön apua kaikissa arjen toiminnoissaan. Hän ei voi koskaan jatkaa opintojaan ja on täysin työkyvytön. Mitä liikennevakuutus voi korvata hänelle?</p>	<p>Liikennevakuutus korvaa Liisan tapauksessa henkilövahinkoja, jotka ovat seurausta liikenneonnettomuudesta. Tässä tilanteessa Liisa voi saada korvauksia seuraavasti:</p> <ol style="list-style-type: none"> 1. Sairaanhoitokulut: Kaikki Liisan tarvitsemat lääketieteelliset toimenpiteet, kuten sairaalakulut, lääkkeitä, kuntoutus ja muut hoitotoimet. 2. Kuntoutuskorvaus: Erityisen kuntoutuksen kustannukset, jotka auttavat häntä sopeutumaan vammaansa. 3. Henkilökohtainen apu: Mikäli Liisa tarvitsee jatkuvaa apua päivittäisissä toiminnoissa, korvataan myös tämän avun kustannukset. 4. Korvaus ansionmenetyksestä: Koska Liisa ei kykene työskentelemään, hänelle voidaan myöntää korvausta tulevaisuuden ansionmenetyksestä. 5. Kärsimyskorvaus: Korvaus henkisestä kärsimyksestä ja kivusta, jota onnettomuus ja sen seuraukset ovat aiheuttaneet. 6. Muut kulut: Mahdolliset muutosten tekemiset kotiin pyörätuolin käyttöön sopivaksi, erikoisvälineet ja muut tarvittavat varusteet. <p>On tärkeää, että Liisan huoltajat tai lailliset edustajat ottavat yhteyttä vakuutusyhtiöön mahdollisimman pian onnettomuuden jälkeen prosessin aloittamiseksi. Lisäksi on hyvä konsultoida taloudellista neuvonantajaa tai lakimiestä varmistamaan, että kaikki tarpeelliset korvaukset käsitellään asianmukaisesti.</p>	X
---	--	--	---

Table 6: Evaluation results for pure model (GPT-4 Turbo) generated responses from expert. Table shows identifier, questions from Table 2, generated answer and feedback from expert. The question's 1, 2, 3, 4 equals to 8, 3, 5, 6 from table 5.

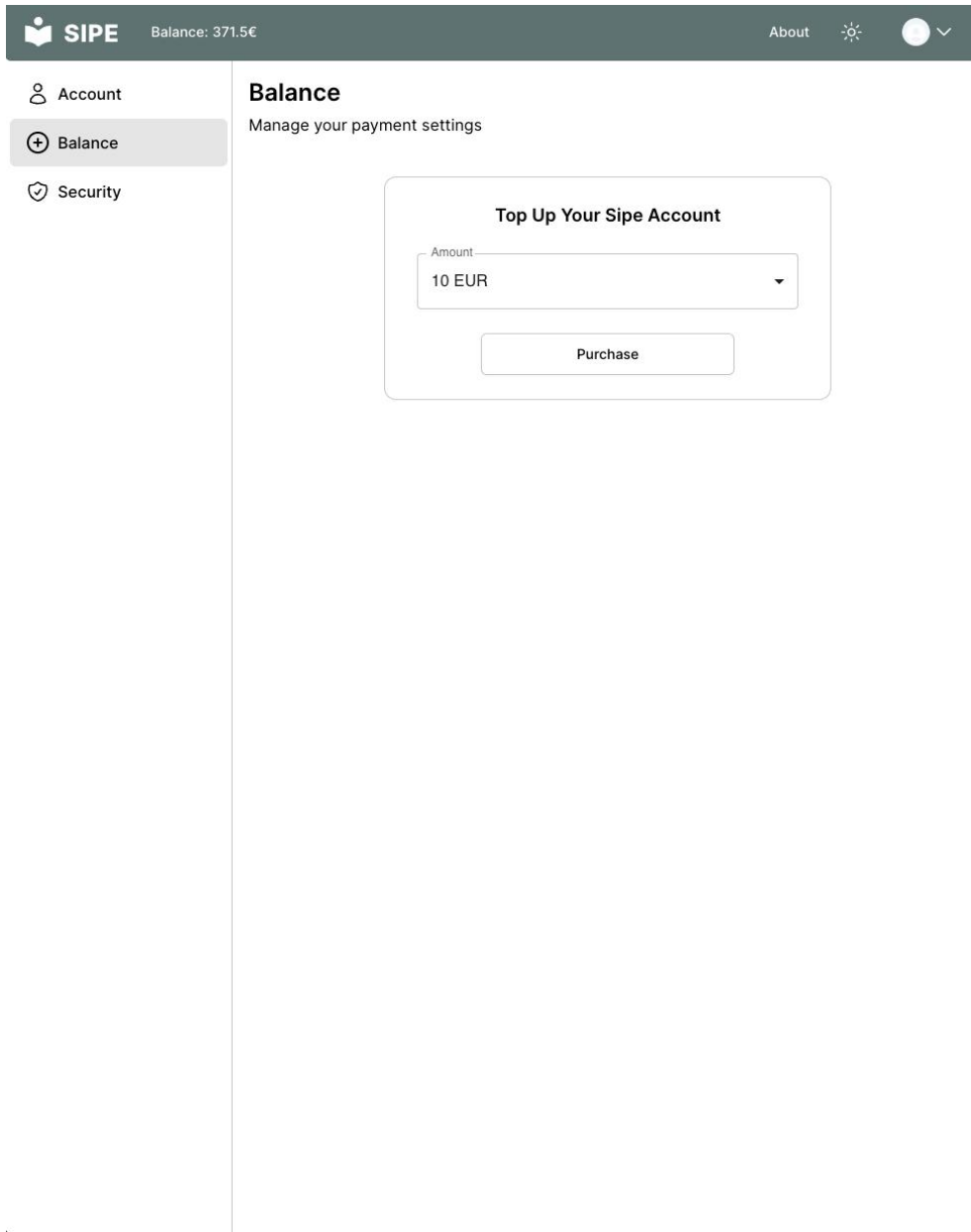


Figure 17: User interface of the credit purchasing functionality in the web application.

Quick query

Chat

Figure 18: User interface of the landing page in the web application.

Welcome Back

Email Address *

Password *

Forgot password?

Log in

Or continue with

Google

Facebook

Don't have an account?

Figure 19: User interface of the login/register functionality in the web application.

Traffic accident happened? when did it happen?

At free time

During school / work

Work / school travel

Figure 20: User interface of the quick test functionality in the web application.