

# TERM AWARE GUARD



# Introduction

Rarely people read the whole data corpus of terms of service when they sign up for companies and we came with a solution:

The primary objective of Term Aware Guard is to simplify the readability of Terms and Conditions, providing a summarized version and ensuring that users are well-informed about data privacy beforehand. The purpose of the app is to summarize and inform users about the terms and conditions of the companies they have signed up for. To improve the model's performance and gather feedback, we evaluate it with the assistance of TruLens.





# + Problem Statement +

Millions of users struggle to understand the complex legal language used in terms and conditions (ToS) documents, leading to potential harm and lack of informed consent.





# Aim



Develop an AI-driven solution to enhance user understanding of terms and conditions (ToS) by simplifying complex legal language, fostering informed consent, and promoting user empowerment in the digital landscape.



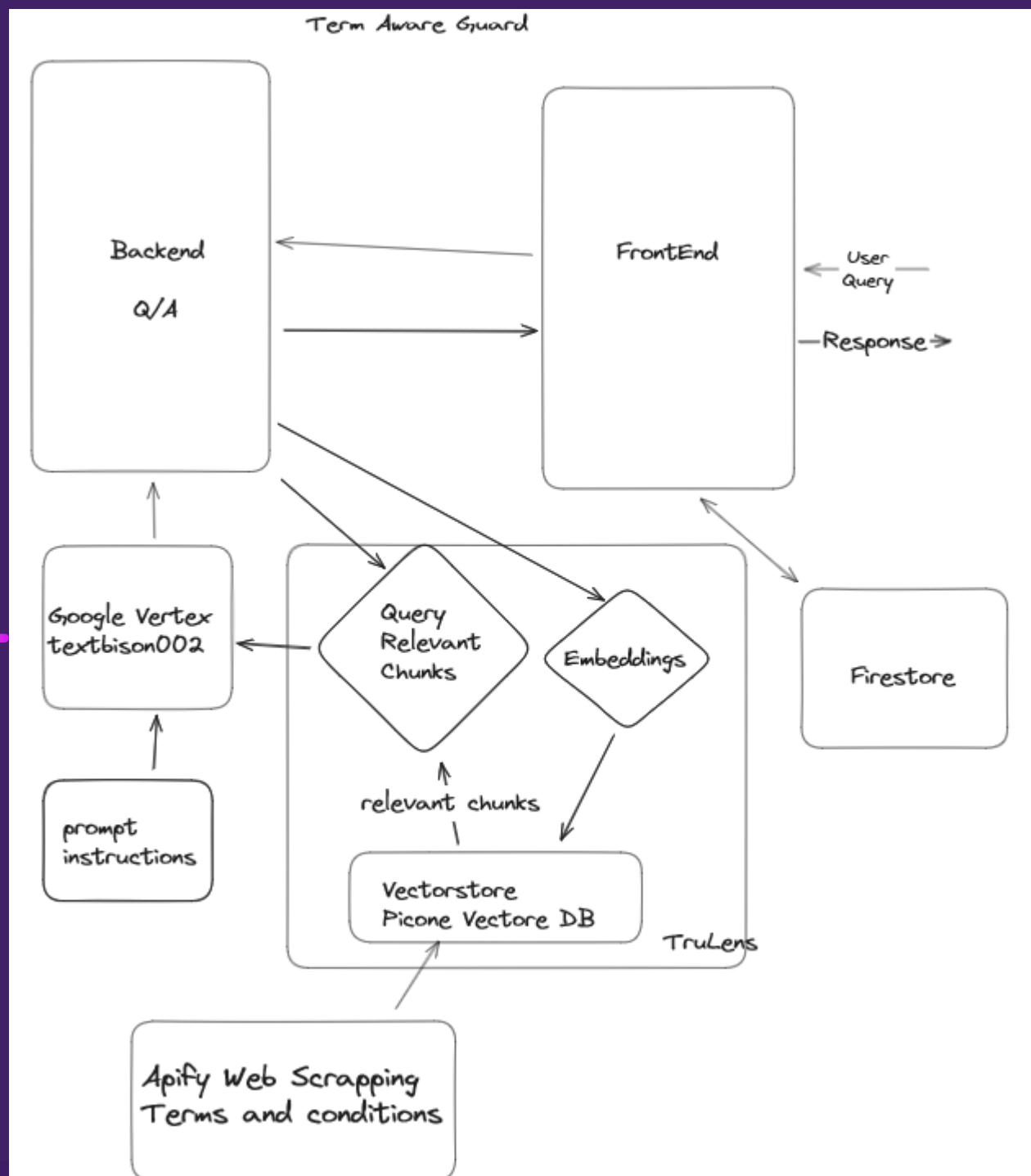


# Business Model

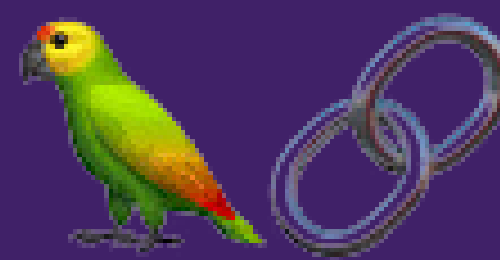
- Free tier, when a users signs up will summarize 4 Terms and Conditions per day
- Paid tier will give unlimited summarizes, Analysis and Visualisation Tab, and a summarisation tab where the user can upload a link of the Terms and conditions and the LLM will summarize it



# Core Technology System Architecture



TS



# Technical



The web app with built with React and Next.js using GCP for hosting, Firebase for storage. GCP is also used for the usage and tuning of the model text-bison002 provided by Google Vertex AI PaLM2

## Features:

- Log in with google
- Chat feature and history
- Dark/light mode

For backend we set up a python programming environment.

We used DigitalOcean as provider for the platform



```
2 from fastapi import FastAPI
3 from fastapi.responses import RedirectResponse
4 from langserve import add_routes
5 from fastapi.middleware.cors import CORSMiddleware
6
7 import dotenv
8
9 dotenv.load_dotenv()
10
11 app = FastAPI()
12
13 app.add_middleware(
14     CORSMiddleware,
15     allow_origins=["*"],
16     allow_credentials=True,
17     allow_methods=["*"],
18     allow_headers=["*"],
19 )
20
21 @app.get("/")
22 async def redirect_root_to_docs():
23     return RedirectResponse("/docs")
24
25
26
27
28 add_routes(app, chain, path="/summarise_tnc")
29
30 if __name__ == "__main__":
31     import uvicorn
32
33     uvicorn.run(app, host="0.0.0.0", port=8000)
34
```

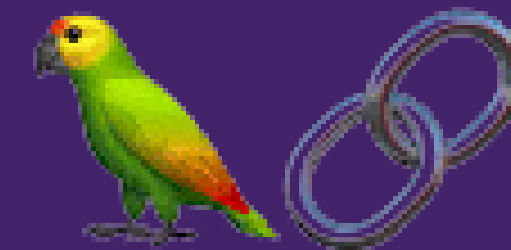
## AI System

- Google Vertex AI
- Apify
- Pinecone Vector DB
- TruLens
- RAG with Langchain





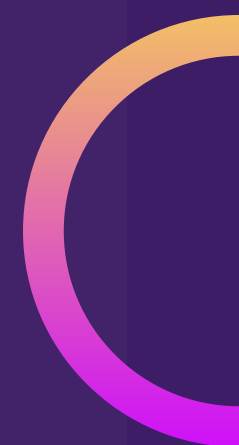
# RAG LangChain



Using RAG as the powerhouse of our app, for our case is much more reliable as Terms and Conditions change very often. Using RAG to enhance the model by context not only turns it into a specialised LLM, but we don't need to fine tune and retrain the model every week, which would be highly inefficient



```
70 _prompt = ChatPromptTemplate.from_messages(  
71     [  
72         ("system", "You are a friendly assistant that summarizes terms and conditions for a given company."),  
73         ("human", ""  
74             Summarize the terms and conditions for signing up for the company ${text} in the format:  
75             Heading of company name  
76             Summary of what is important in Terms and Conditions:  
77             - bullet points  
78             Things to be aware of:  
79             - bullet points  
80  
81             If the company name provided isn't real, dont make up a random response and tell the user to please make sure the company they are asking about is real.  
82  
83     """),  
84     ]  
85 )  
86  
87 chain = _prompt | _model  
88
```

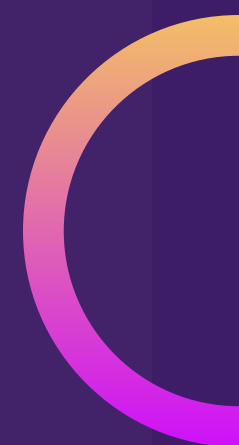


# Apify

## Web Scrapping



We used Apify to web scrapp through 1000 pages of terms and conditions of big companies and we have exported them in our Pinecone Vector Database



# Pinecone VectorStore



We are using Pinecone as our Vector database where we import the context the model needs when a query happens

```
22 pinecone = Pinecone()
23
24 dataset = pinecone_datasets.load_dataset('IthgZdVXyQvgoj61R')
25 dataset.head()
26 pinecone.create_index(
27     name=index_name_v1,
28     metric='cosine',
29     dimension=1536,
30     metadata_config={"indexed":{}}
31 )
32
33 embed = VertexAIEmbeddings(
34     model='text-embedding-bison@002',
35 )
36
37 text_splitter = RecursiveCharacterTextSplitter(
38     chunk_size = 256,
39     chunk_overlap = 20
40 )
41 docs = text_splitter.create_documents(dataset)
42
43 index = pinecone.Index(index_name_v1)
44
45 vectorstore = Pinecone(
46     index, embed.embed_query, docs
47 )
48
```





# Evaluation Using TruLens



TruLens is used to measure the quality and groundedness of our LLM providing feedback and metrics



```
49 qa = RetrievalQA.from_chain_type(
50     llm= model,
51     chain_type="stuff",
52     retriever=vectorstore.as_retriever()
53 )
54
55 tru = Tru()
56
57 vertexai = feedback.vertexai()
58
59 qa_relevance = Feedback(vertexai.relevance).on_input_output()
60
61 qs_relevance = Feedback(vertexai.qs_relevance).on_input().on(Select.Record.app.combine_documents_chain._call.args.inputs.input_documents[:].page_content).aggregate(np.mean)
62
63 truchain = TruChain(qa,app_id='TAG_CHAIN',feedbacks=[qa_relevance, qs_relevance])
64
65 truchain(text)
66
67 tru.run_dashboard()
68
```



# Development Process

Starting from  
the micro hackathon



We initially planned on finetuning the textbison002 model with terms and conditions datasets such as Claudette or CuadV1

We developed it fully in next.js for the micro hackathon and then we transitioned to a hosted VM backend as REST API which frontend calls upon.

We did this to integrate TruLens as an evaluation tool for our LLM system

To enhance its summarization capabilities to an advanced level, instead of fine-tuning on somewhat outdated data, we employ the RAG framework, resulting in much better performance, without hallucinations and with updated information



# Development Process

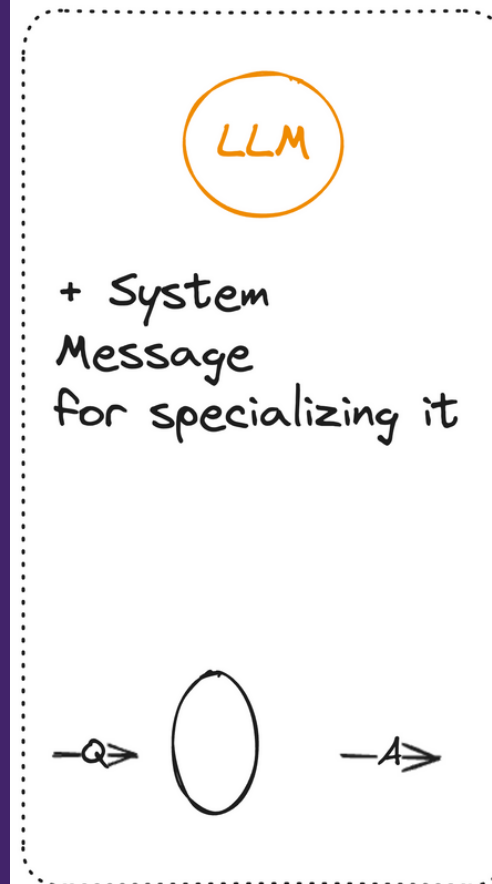


## Brain Storming

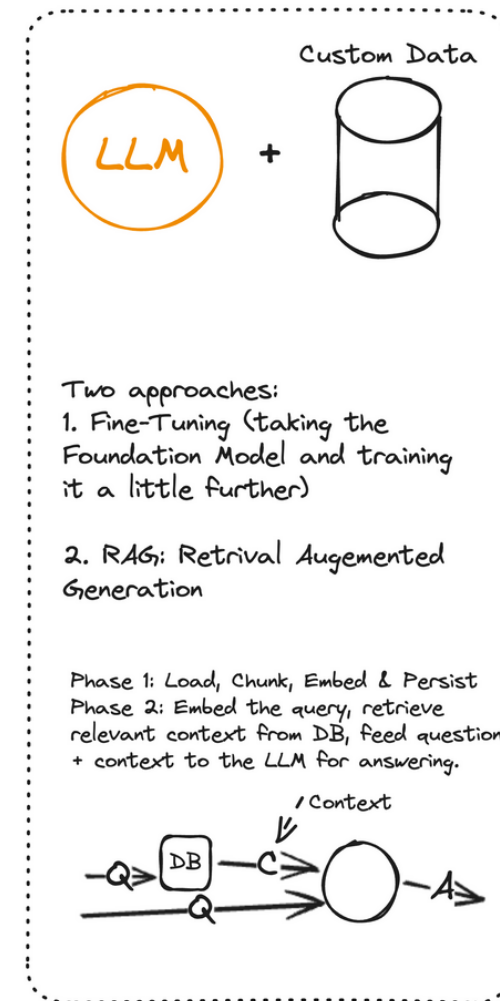


### Different ways to build LLM applications

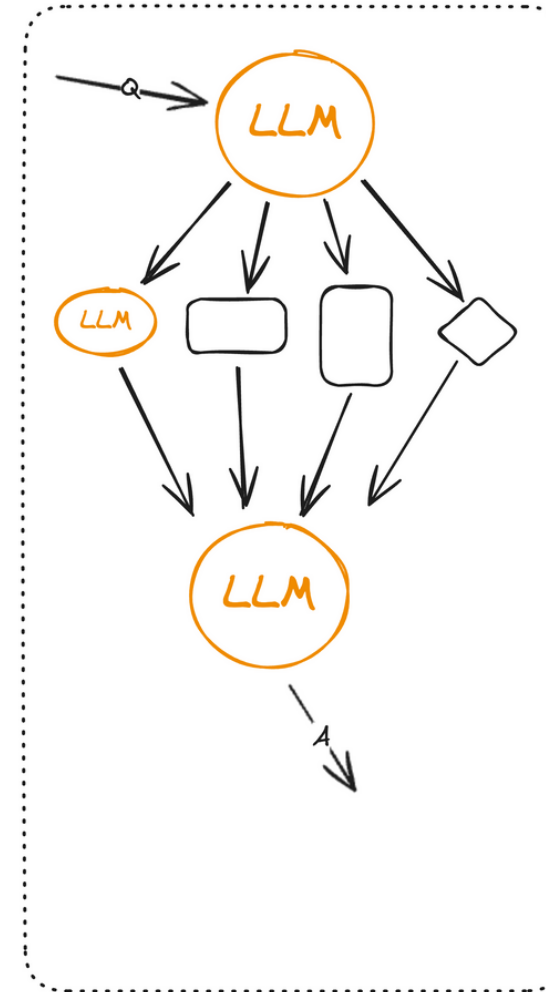
Only LLM



Customization with own data



MRKL: Modular Reasoning and Knowledge Systems a.k.a "Agents"



Evaluation





# Future plans

Finish implementing summarise tab, where users provide a url link a companys' terms and condition and they will get a summarization of it made by the AI. The ToS will go back into the pipeline and be saved in the Vector Database for future uses

Finish implementing Analisis tab, where users can see a real time graph or plot of the data privacy score they have based on the companies they've signed up for





# Demo



☀ Light Mode

💬 New Chat

---

**History**

google

spotify

---

👤 Gracjan

Standard **Summarise**

## Term Aware Guard

🗣 google

**Google**  
Summary of what is important in Terms and Conditions:

- Google may collect and use your personal information, including your name, email address, and IP address, to provide and improve its services.
- Google may use cookies and other tracking technologies to collect information about your activity on its services.
- Google may share your personal information with third parties, such as advertisers and government agencies, in certain circumstances.
- You can control the collection and use of your personal information by Google through your Google Account settings.

**Things to be aware of:**

- Google's Terms and Conditions are subject to change at any time.
- Google may terminate your account or access to its services if you violate its Terms and Conditions.
- Google is not responsible for the content or practices of third-party websites or services that you access through its services.

🗣 amazon

**Amazon Terms and Conditions Summary**  
Summary of what is important in Terms and Conditions:

- You grant Amazon a non-exclusive, worldwide, royalty-free, perpetual, irrevocable right and license to use, reproduce, modify, adapt, publish, translate, create derivative works from, distribute

Enter link of the Terms and conditions ▶



# TEAM



**Arnoan**  
Technical



**Sebastian Momir**  
Technical/non-Technical



**Gracjan Wojciechowski**  
Technical



# THANK YOU

