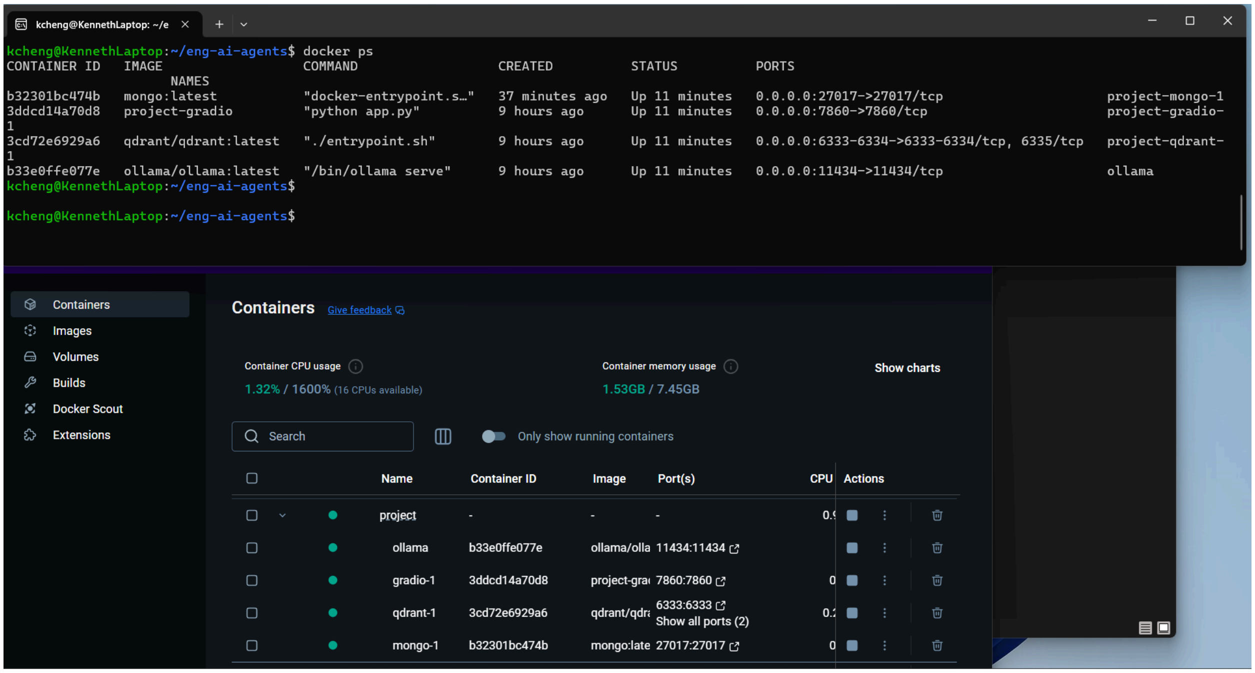
Environment and Tooling Milestone

See the project README.md file for all instructions for running the model and how it works. Consider the README.md a more detailed report. This project report only shows screenshots proving each milestone is complete.

The screenshot below shows the containers started by the project docker compose file(mongo, gradio, qdrant, ollama). ClearML is run from the web instead as shown in the later milestones



```
kcheng@KennethLaptop:~/eng-ai-agents$ docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          PORTS                                      
              NAMES
b32301bc474b   mongo:latest             "docker-entrypoint.s…"   37 minutes ago   Up 11 minutes   0.0.0.0:27017->27017/tcp                   project-mongo-1
3ddcd14a70d8   project-gradio           "python app.py"          9 hours ago      Up 11 minutes   0.0.0.0:7860->7860/tcp                     project-gradio-
1
3cd72e6929a6   qdrant/qdrant:latest     "./entrypoint.sh"        9 hours ago      Up 11 minutes   0.0.0.0:6333-6334->6333-6334/tcp, 6335/tcp   project-qdrant-
1
b33e0ffe077e   ollama/ollama:latest     "/bin/ollama serve"      9 hours ago      Up 11 minutes   0.0.0.0:11434->11434/tcp                   ollama
kcheng@KennethLaptop:~/eng-ai-agents$

kcheng@KennethLaptop:~/eng-ai-agents$
```

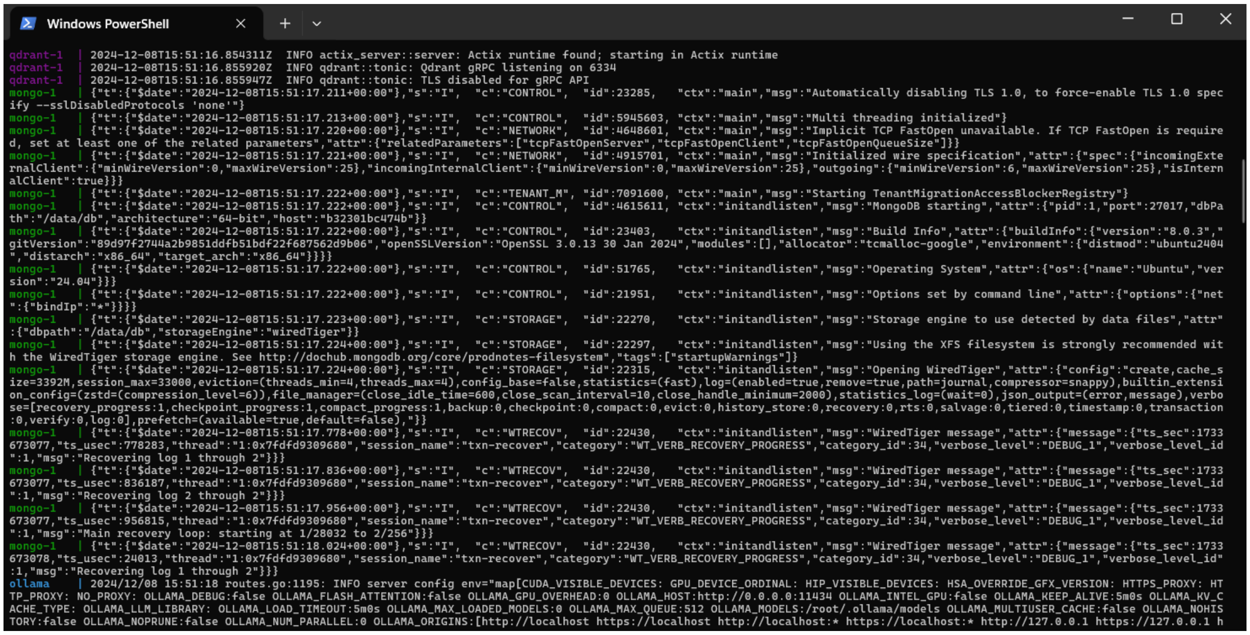
Containers

Container CPU usage
1.32% / 1600% (16 CPUs available)

Container memory usage
1.53GB / 7.45GB

☐		Name	Container ID	Image	Port(s)	CPU	Actions
☐	●	project	-	-	-	0.	
☐	●	ollama	b33e0ffe077e	ollama/olla	11434:11434		
☐	●	gradio-1	3ddcd14a70d8	project-gra	7860:7860	0	
☐	●	qdrant-1	3cd72e6929a6	qdrant/qdr	6333:6333 Show all ports (2)	0.	
☐	●	mongo-1	b32301bc474b	mongo:late	27017:27017	0	

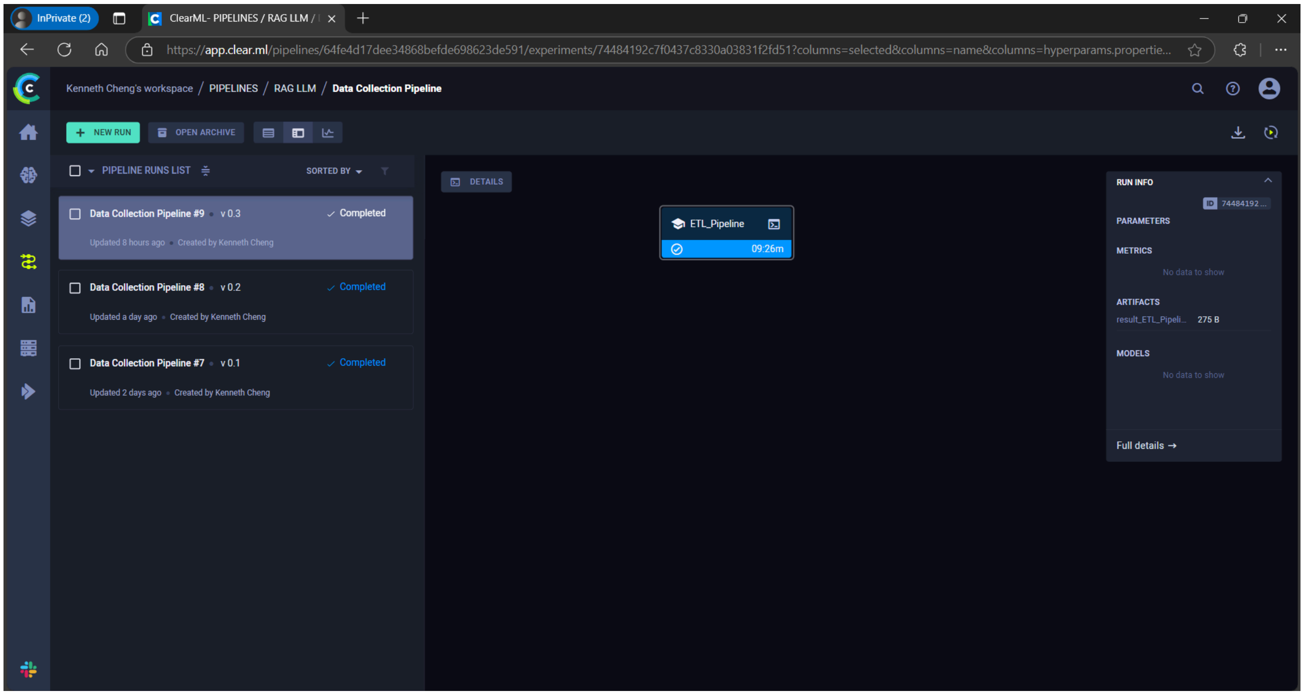
After running docker compose up, you can see the containers respond whenever running files in the project folder if setup correctly. (Ignore the output jargon)



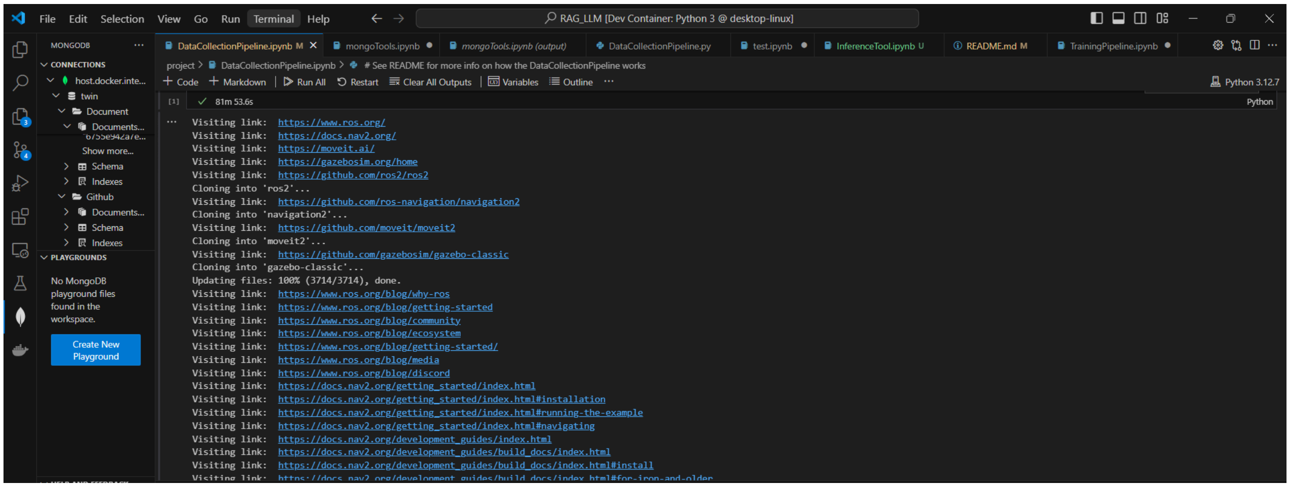
```
qdrant-1  | 2024-12-08T15:51:16.854311Z  INFO actix_server::server: Actix runtime found; starting in Actix runtime
qdrant-1  | 2024-12-08T15:51:16.855920Z  INFO qdrant::tonic: Qdrant gRPC listening on 6334
qdrant-1  | 2024-12-08T15:51:16.855947Z  INFO qdrant::tonic: TLS disabled for gRPC API
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.211+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 spec
ify --sslDisabledProtocols 'none'"}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.213+00:00"},"s":"I",  "c":"CONTROL",  "id":5945603, "ctx":"main","msg":"Multi threading initialized"}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.220+00:00"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"main","msg":"Implicit TCP FastOpen unavailable. If TCP FastOpen is require
d, set at least one of the related parameters","attr":{"relatedParameters":["tcpFastOpenServer","tcpFastOpenClient","tcpFastOpenQueueSize"]}}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.221+00:00"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"main","msg":"Initialized wire specification","attr":{"spec":{"incomingExte
rnalClient":{"minWireVersion":0,"maxWireVersion":25},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":25},"outgoing":{"minWireVersion":6,"maxWireVersion":25},"isIntern
alClient":true}}}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.222+00:00"},"s":"I",  "c":"TENANT_M", "id":7091600, "ctx":"main","msg":"Starting TenantMigrationAccessBlockerRegistry"}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.222+00:00"},"s":"I",  "c":"CONTROL",  "id":4615611, "ctx":"initandlisten","msg":"MongoDB starting","attr":{"pid":1,"port":27017,"dbPa
th":"/data/db","architecture":"64-bit","host":"b32301bc474b"}}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.222+00:00"},"s":"I",  "c":"CONTROL",  "id":23403,   "ctx":"initandlisten","msg":"Build Info","attr":{"buildInfo":{"version":"8.0.3","
gitVersion":"89d97f2744a2b9851ddfb51bdf22f687562d9b06","openSSLVersion":"OpenSSL 3.0.13 30 Jan 2024","modules":[],"allocator":"tcmalloc-google","environment":{"distmod":"ubuntu2404
","distarch":"x86_64","target_arch":"x86_64"}}}}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.222+00:00"},"s":"I",  "c":"CONTROL",  "id":51765,   "ctx":"initandlisten","msg":"Operating System","attr":{"os":{"name":"Ubuntu","ver
sion":"24.04"}}}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.222+00:00"},"s":"I",  "c":"CONTROL",  "id":21951,   "ctx":"initandlisten","msg":"Options set by command line","attr":{"options":{"net
":{"bindIp":"*"}}}}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.223+00:00"},"s":"I",  "c":"STORAGE",  "id":22270,   "ctx":"initandlisten","msg":"Storage engine to use detected by data files","attr"
:{"dbpath":"/data/db","storageEngine":"wiredTiger"}}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.224+00:00"},"s":"I",  "c":"STORAGE",  "id":22297,   "ctx":"initandlisten","msg":"Using the XFS filesystem is strongly recommended wit
h the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem","tags":["startupWarnings"]}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.224+00:00"},"s":"I",  "c":"STORAGE",  "id":22315,   "ctx":"initandlisten","msg":"Opening WiredTiger","attr":{"config":"create,cache_s
ize=3392M,session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,remove=true,path=journal,compressor=snappy),builtin_extensi
on_config=(zstd=(compression_level=6)),file_manager=(close_idle_time=600,close_scan_interval=10,close_handle_minimum=2000),statistics_log=(wait=0),json_output=(error,message),verbo
se=[recovery_progress:1,checkpoint_progress:1,compact_progress:1,backup:0,checkpoint:0,compact:0,evict:0,history_store:0,recovery:0,rts:0,salvage:0,tiered:0,timestamp:0,transaction
:0,verify:0,log:0],prefetch=(available=true,default=false),"}}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.778+00:00"},"s":"I",  "c":"WTRECOV",  "id":22430,   "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1733
673077,"ts_usec":778283,"thread":"1:0x7fdfd9309680","session_name":"txn-recover","category":"WT_VERB_RECOVERY_PROGRESS","category_id":34,"verbose_level":"DEBUG_1","verbose_level_id
":1,"msg":"Recovering log 1 through 2"}}}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.836+00:00"},"s":"I",  "c":"WTRECOV",  "id":22430,   "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1733
673077,"ts_usec":836187,"thread":"1:0x7fdfd9309680","session_name":"txn-recover","category":"WT_VERB_RECOVERY_PROGRESS","category_id":34,"verbose_level":"DEBUG_1","verbose_level_id
":1,"msg":"Recovering log 2 through 2"}}}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:17.956+00:00"},"s":"I",  "c":"WTRECOV",  "id":22430,   "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1733
673077,"ts_usec":956815,"thread":"1:0x7fdfd9309680","session_name":"txn-recover","category":"WT_VERB_RECOVERY_PROGRESS","category_id":34,"verbose_level":"DEBUG_1","verbose_level_id
":1,"msg":"Main recovery loop: starting at 1/28032 to 2/256"}}}
mongo-1   | {"t":{"$date":"2024-12-08T15:51:18.024+00:00"},"s":"I",  "c":"WTRECOV",  "id":22430,   "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":{"ts_sec":1733
673078,"ts_usec":24013,"thread":"1:0x7fdfd9309680","session_name":"txn-recover","category":"WT_VERB_RECOVERY_PROGRESS","category_id":34,"verbose_level":"DEBUG_1","verbose_level_id"
:1,"msg":"Recovering log 1 through 2"}}}
ollama    | 2024/12/08 15:51:18 routes.go:1195: INFO server config env="map[CUDA_VISIBLE_DEVICES: GPU_DEVICE_ORDINAL: HIP_VISIBLE_DEVICES: HSA_OVERRIDE_GFX_VERSION: HTTPS_PROXY: HT
TP_PROXY: NO_PROXY: OLLAMA_DEBUG:false OLLAMA_FLASH_ATTENTION:false OLLAMA_GPU_OVERHEAD:0 OLLAMA_HOST:http://0.0.0.0:11434 OLLAMA_INTEL_GPU:false OLLAMA_KEEP_ALIVE:5m0s OLLAMA_KV_C
ACHE_TYPE: OLLAMA_LLM_LIBRARY: OLLAMA_LOAD_TIMEOUT:5m0s OLLAMA_MAX_LOADED_MODELS:0 OLLAMA_MAX_QUEUE:512 OLLAMA_MODELS:/root/.ollama/models OLLAMA_MULTIUSER_CACHE:false OLLAMA_NOHIS
TORY:false OLLAMA_NOPRUNE:false OLLAMA_NUM_PARALLEL:0 OLLAMA_ORIGINS:[http://localhost https://localhost http://localhost:* https://localhost:* http://127.0.0.1 https://127.0.0.1 h
```

ETL Milestone

The image below shows the ClearML/DataCollectionPipeline.py result in the ClearML website. It takes in links as parameters and outputs a list of texts from documents(documentation) and codes(github files) that come from the links it crawled. ClearML was run only on files related to the "ros" domain to save time. You can run either the ClearML or the ipynb notebook on the full input since they do the same thing, but ClearML requires setup and takes longer while the ipynb prints some output to the terminal, thus ipynb is preferred. Next I will go over the ipynb

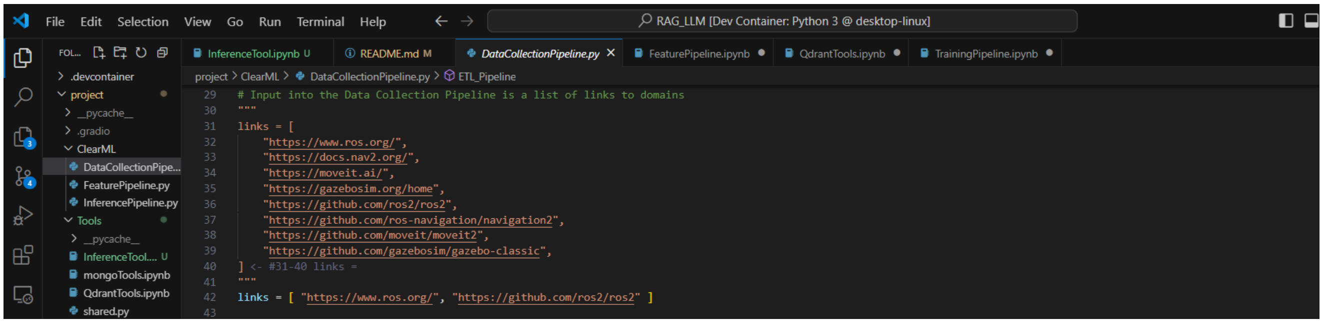

InPrivate (2)
ClearML- PIPELINES / RAG LLM /
https://app.clear.ml/pipelines/64fe4d17dee34868befde698623de591/experiments/74484192c7f0437c8330a03831f2fd51?columns=selected&columns=name&columns=hyperparams.propertie...
Kenneth Cheng's workspace / PIPELINES / RAG LLM / Data Collection Pipeline
NEW RUN
OPEN ARCHIVE
PIPELINE RUNS LIST
SORTED BY
Data Collection Pipeline #9 v 0.3 Completed
Updated 8 hours ago • Created by Kenneth Cheng
Data Collection Pipeline #8 v 0.2 Completed
Updated a day ago • Created by Kenneth Cheng
Data Collection Pipeline #7 v 0.1 Completed
Updated 2 days ago • Created by Kenneth Cheng
DETAILS
ETL_Pipeline
09:26m
RUN INFO
ID 74484192...
PARAMETERS
METRICS
No data to show
ARTIFACTS
result_ETL_Pipeli... 275 B
MODELS
No data to show
Full details →

The image below shows part of the output after running the DataCollectionPipeline.ipynb on the full list of links. It took an entire 81 minutes, so it might make sense to test the model on only the "ros" links(and ask it questions related to only that link). Related code shown next


File Edit Selection View Go Run Terminal Help
RAG_LLM [Dev Container: Python 3 @ desktop-linux]
MONGODB
CONNECTIONS
host.docker.inte...
twin
Document
Documents...
Show more...
Schema
Indexes
Github
Documents...
Schema
Indexes
PLAYGROUNDS
No MongoDB playground files found in the workspace.
Create New Playground
DataCollectionPipeline.ipynb M
mongoTools.ipynb
mongoTools.ipynb (output)
DataCollectionPipeline.py
test.ipynb
InferenceTool.ipynb U
README.md M
TrainingPipeline.ipynb
project > DataCollectionPipeline.ipynb > # See README for more info on how the DataCollectionPipeline works
+ Code + Markdown Run All Restart Clear All Outputs Variables Outline
Python 3.12.7
Python
[1] ✓ 81m 53.6s

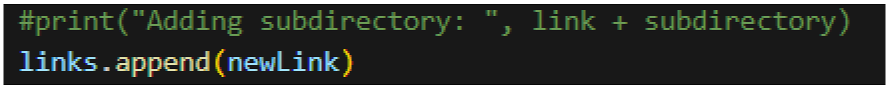
```
Visiting link:  https://www.ros.org/
Visiting link:  https://docs.nav2.org/
Visiting link:  https://moveit.ai/
Visiting link:  https://gazebosim.org/home
Visiting link:  https://github.com/ros2/ros2
Cloning into 'ros2'...
Visiting link:  https://github.com/ros-navigation/navigation2
Cloning into 'navigation2'...
Visiting link:  https://github.com/moveit/moveit2
Cloning into 'moveit2'...
Visiting link:  https://github.com/gazebosim/gazebo-classic
Cloning into 'gazebo-classic'...
Updating files: 100% (3714/3714), done.
Visiting link:  https://www.ros.org/blog/why-ros
Visiting link:  https://www.ros.org/blog/getting-started
Visiting link:  https://www.ros.org/blog/community
Visiting link:  https://www.ros.org/blog/ecosystem
Visiting link:  https://www.ros.org/blog/getting-started/
Visiting link:  https://www.ros.org/blog/media
Visiting link:  https://www.ros.org/blog/discord
Visiting link:  https://docs.nav2.org/getting_started/index.html
Visiting link:  https://docs.nav2.org/getting_started/index.html#installation
Visiting link:  https://docs.nav2.org/getting_started/index.html#running-the-example
Visiting link:  https://docs.nav2.org/getting_started/index.html#navigating
Visiting link:  https://docs.nav2.org/development_guides/index.html
Visiting link:  https://docs.nav2.org/development_guides/build_docs/index.html
Visiting link:  https://docs.nav2.org/development_guides/build_docs/index.html#install
Visiting link:  https://docs.nav2.org/development_guides/build_docs/index.html#for-iron-and-older
```

Consider changing the links in the DataCollection.ipynb file to be similar to those in the ClearML/DataCollection.py file before running it to search only the ROS subdomain.



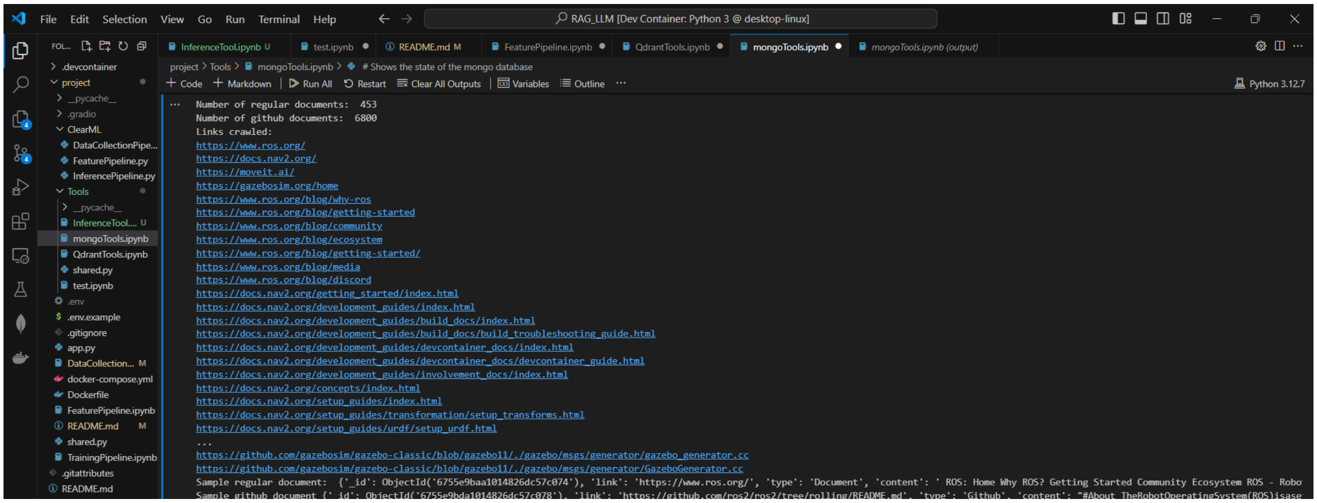
```
# Input into the Data Collection Pipeline is a list of links to domains
"""
links = [
    "https://www.ros.org/",
    "https://docs.nav2.org/",
    "https://moveit.ai/",
    "https://gazebosim.org/home",
    "https://github.com/ros2/ros2",
    "https://github.com/ros-navigation/navigation2",
    "https://github.com/moveit/moveit2",
    "https://github.com/gazebosim/gazebo-classic",
] <- #31-40 links =
"""
links = [ "https://www.ros.org/", "https://github.com/ros2/ros2" ]
```

Alternatively, you can replace the following lines with “pass” to avoid searching all links inside each website(which each would have links of their own and so on, creating a massive tree of links). This large tree of links formed is the main reason the crawling takes over an hour.



```
#print("Adding subdirectory: ", link + subdirectory)
links.append(newLink)
```

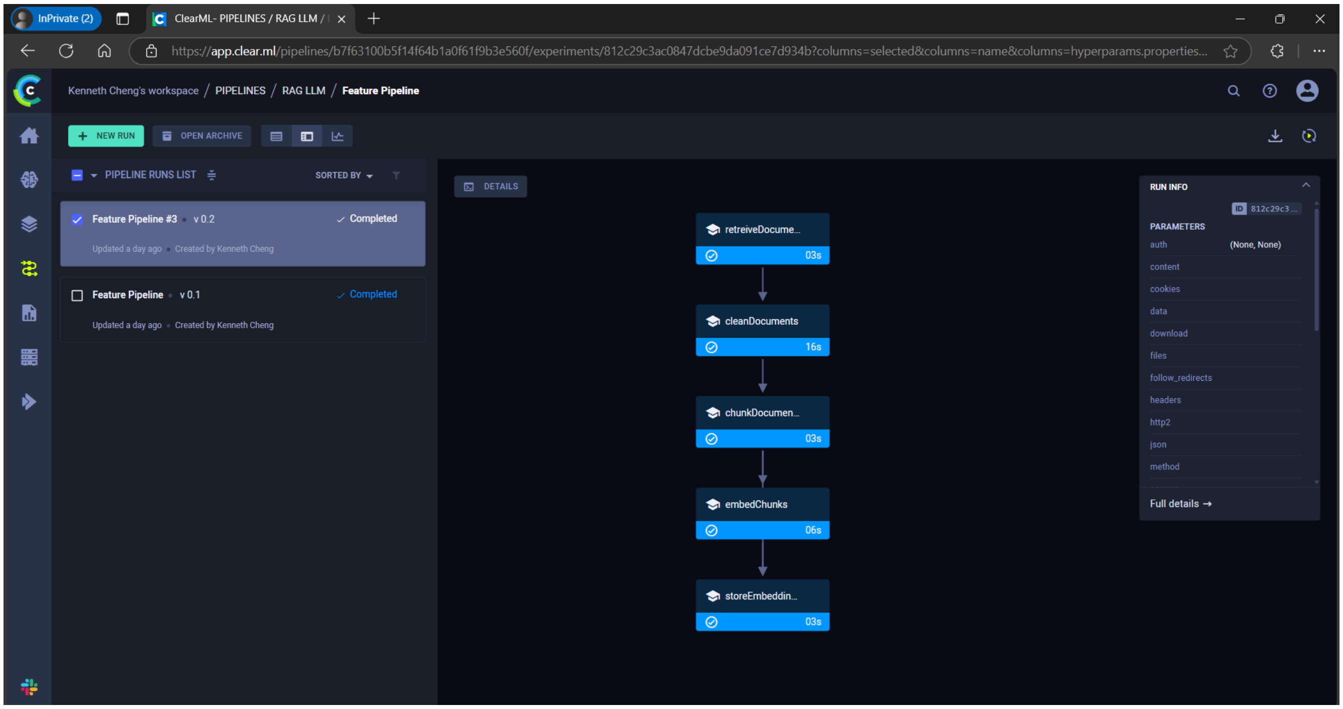
The image below shows Tools/mongoTools.ipynb giving a summary of what was stored in the mongoDB. In total, 453 documents and 6800 code files were processed. And samples of what the documents and code looks like is shown at the bottom. Each piece of data contains an id, link, document type, and text content.



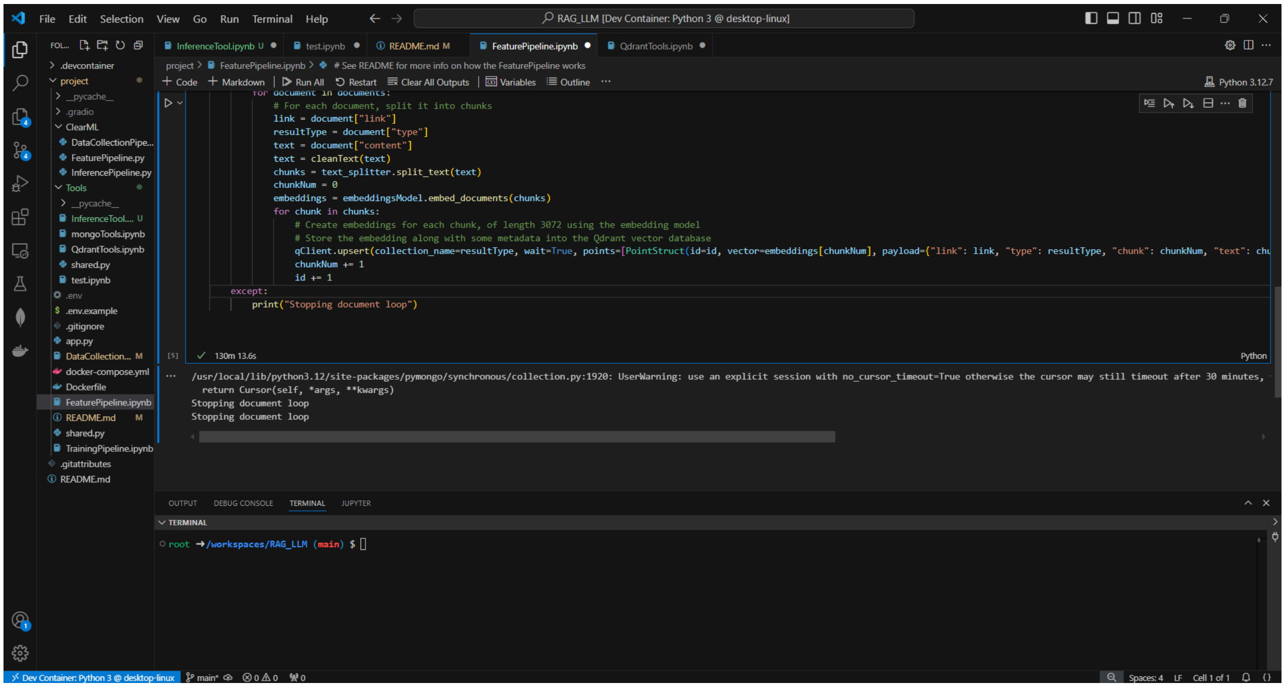
```
Number of regular documents:  453
Number of github documents:  6800
Links crawled:
https://www.ros.org/
https://docs.nav2.org/
https://moveit.ai/
https://gazebosim.org/home
https://www.ros.org/blog/why-ros
https://www.ros.org/blog/getting-started
https://www.ros.org/blog/community
https://www.ros.org/blog/ecosystem
https://www.ros.org/blog/getting-started/
https://www.ros.org/blog/media
https://www.ros.org/blog/discord
https://docs.nav2.org/getting_started/index.html
https://docs.nav2.org/development_guides/index.html
https://docs.nav2.org/development_guides/build_docs/index.html
https://docs.nav2.org/development_guides/build_docs/build_troubleshooting_guide.html
https://docs.nav2.org/development_guides/devcontainer_docs/index.html
https://docs.nav2.org/development_guides/devcontainer_docs/devcontainer_guide.html
https://docs.nav2.org/development_guides/involvement_docs/index.html
https://docs.nav2.org/concepts/index.html
https://docs.nav2.org/setup_guides/index.html
https://docs.nav2.org/setup_guides/transformation/setup_transforms.html
https://docs.nav2.org/setup_guides/urdf/setup_urdf.html
...
https://github.com/gazebosim/gazebo-classic/blob/gazebo11/./gazebo/msgs/generator/gazebo_generator.cc
https://github.com/gazebosim/gazebo-classic/blob/gazebo11/./gazebo/msgs/generator/GazeboGenerator.cc
Sample regular document:  {'_id': ObjectId('6755e9baa1014826dc57c074'), 'link': 'https://www.ros.org/', 'type': 'Document', 'content': ' ROS: Home Why ROS? Getting Started Community Ecosystem ROS - Robo
Sample github document {'_id': ObjectId('6755e9bda1014826dc57c078'), 'link': 'https://github.com/ros2/ros2/tree/rolling/README.md', 'type': 'Github', 'content': "#About TheRobotOperatingSystem(ROS)isase
```

Featurization Pipeline Milestone

The ClearML/FeaturePipeline.py file retrieves documents, cleans them, chunks them, creates embeddings, then stores those embeddings in Qdrant as shown in the ClearML website below


ClearML- PIPELINES / RAG LLM
https://app.clear.ml/pipelines/b7f63100b5f14f64b1a0f61f9b3e560f/experiments/812c29c3ac0847dcbe9da091ce7d934b?columns=selected&columns=name&columns=hyperparams.properties...
Kenneth Cheng's workspace / PIPELINES / RAG LLM / Feature Pipeline
NEW RUN
OPEN ARCHIVE
PIPELINE RUNS LIST
SORTED BY
Feature Pipeline #3 v 0.2 Completed
Updated a day ago • Created by Kenneth Cheng
Feature Pipeline v 0.1 Completed
Updated a day ago • Created by Kenneth Cheng
DETAILS
retreiveDocume...
03s
cleanDocuments
16s
chunkDocumen...
03s
embedChunks
06s
storeEmbeddin...
03s
RUN INFO
ID 812c29c3...
PARAMETERS
auth (None, None)
content
cookies
data
download
files
follow_redirects
headers
http2
json
method
Full details →

The image below shows the FeaturePipeline.ipynb being run. The pipeline took over 2 hours to run(depends on links crawled from DataPipeline). It took so long, the Qdrant cursor/iterator timed out, skipping some links.

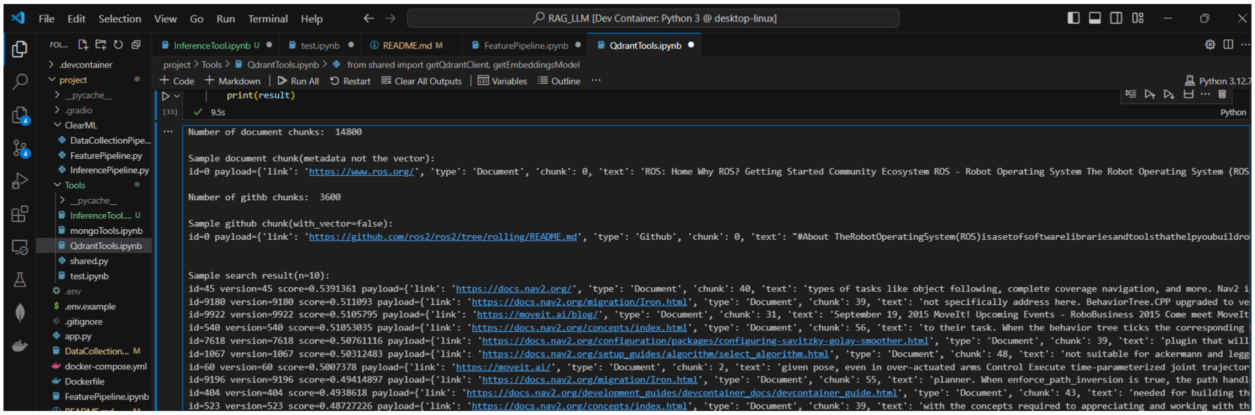


```python
        for document in documents:
            # For each document, split it into chunks
            link = document["link"]
            resultType = document["type"]
            text = document["content"]
            text = cleanText(text)
            chunks = text_splitter.split_text(text)
            chunkNum = 0
            embeddings = embeddingsModel.embed_documents(chunks)
            for chunk in chunks:
                # Create embeddings for each chunk, of length 3072 using the embedding model
                # Store the embedding along with some metadata into the Qdrant vector database
                qClient.upsert(collection_name=resultType, wait=True, points=[PointStruct(id=id, vector=embeddings[chunkNum], payload={"link": link, "type": resultType, "chunk": chunkNum, "text": chu
                chunkNum += 1
                id += 1
    except:
        print("Stopping document loop")
```

```
/usr/local/lib/python3.12/site-packages/pymongo/synchronous/collection.py:1920: UserWarning: use an explicit session with no_cursor_timeout=True otherwise the cursor may still timeout after 30 minutes,
  return Cursor(self, *args, **kwargs)
Stopping document loop
Stopping document loop
```

```
root →/workspaces/RAG_LLM (main) $
```

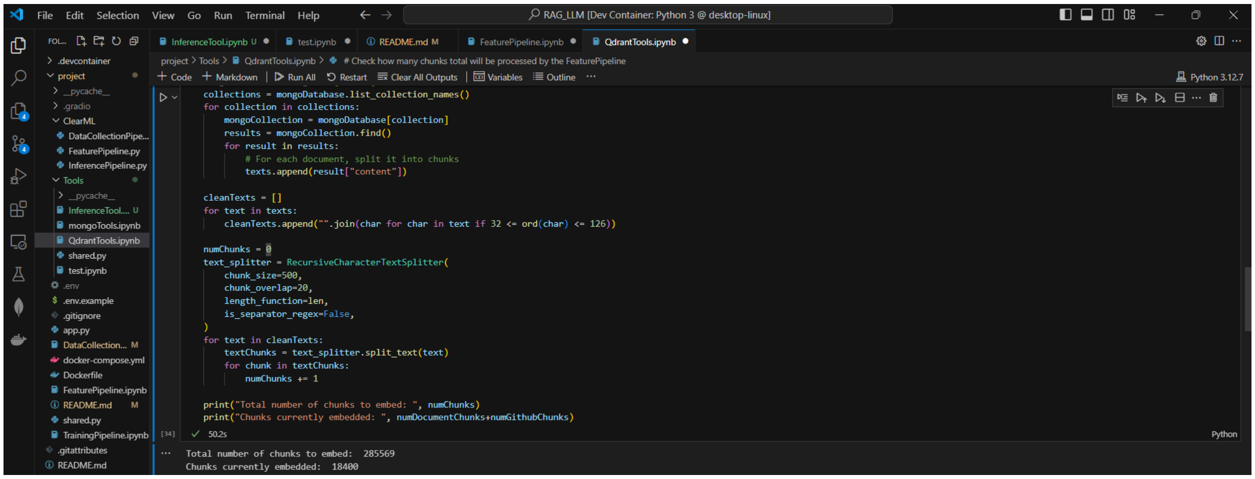
The image below shows the first cell of Tools/qdrantTools.ipynb being used to show summary of the Qdrant vector database after running the FeaturePipeline.ipynb notebook to store embeddings in it.



```
print(result)
```

```
Number of document chunks:  14800

Sample document chunk(metadata not the vector):
id=0 payload={'link': 'https://www.ros.org/', 'type': 'Document', 'chunk': 0, 'text': 'ROS: Home Why ROS? Getting Started Community Ecosystem ROS - Robot Operating System The Robot Operating System (ROS

Number of githb chunks:  3600

Sample github chunk(with_vector=false):
id=0 payload={'link': 'https://github.com/ros2/ros2/tree/rolling/README.md', 'type': 'Github', 'chunk': 0, 'text': "#About TheRobotOperatingSystem(ROS)isasetofsoftwarelibrariesandtoolsthathelpyoubuildro


Sample search result(n=10):
id=45 version=45 score=0.5391361 payload={'link': 'https://docs.nav2.org/', 'type': 'Document', 'chunk': 40, 'text': 'types of tasks like object following, complete coverage navigation, and more. Nav2 i
id=9180 version=9180 score=0.511093 payload={'link': 'https://docs.nav2.org/migration/Iron.html', 'type': 'Document', 'chunk': 39, 'text': 'not specifically address here. BehaviorTree.CPP upgraded to ve
id=9922 version=9922 score=0.5105795 payload={'link': 'https://moveit.ai/blog/', 'type': 'Document', 'chunk': 31, 'text': 'September 19, 2015 MoveIt! Upcoming Events - RoboBusiness 2015 Come meet MoveIt
id=540 version=540 score=0.51053035 payload={'link': 'https://docs.nav2.org/concepts/index.html', 'type': 'Document', 'chunk': 56, 'text': 'to their task. When the behavior tree ticks the corresponding
id=7618 version=7618 score=0.50761116 payload={'link': 'https://docs.nav2.org/configuration/packages/configuring-savitzky-golay-smoother.html', 'type': 'Document', 'chunk': 39, 'text': 'plugin that will
id=1067 version=1067 score=0.50312483 payload={'link': 'https://docs.nav2.org/setup_guides/algorithm/select_algorithm.html', 'type': 'Document', 'chunk': 48, 'text': 'not suitable for ackermann and legg
id=60 version=60 score=0.5007378 payload={'link': 'https://moveit.ai/', 'type': 'Document', 'chunk': 2, 'text': 'given pose, even in over-actuated arms Control Execute time-parameterized joint trajector
id=9196 version=9196 score=0.49414897 payload={'link': 'https://docs.nav2.org/migration/Iron.html', 'type': 'Document', 'chunk': 55, 'text': 'planner. When enforce_path_inversion is true, the path handl
id=404 version=404 score=0.4938618 payload={'link': 'https://docs.nav2.org/development_guides/devcontainer_docs/devcontainer_guide.html', 'type': 'Document', 'chunk': 43, 'text': 'needed for building th
id=523 version=523 score=0.48727226 payload={'link': 'https://docs.nav2.org/concepts/index.html', 'type': 'Document', 'chunk': 39, 'text': 'with the concepts required to appreciating and working with th
```

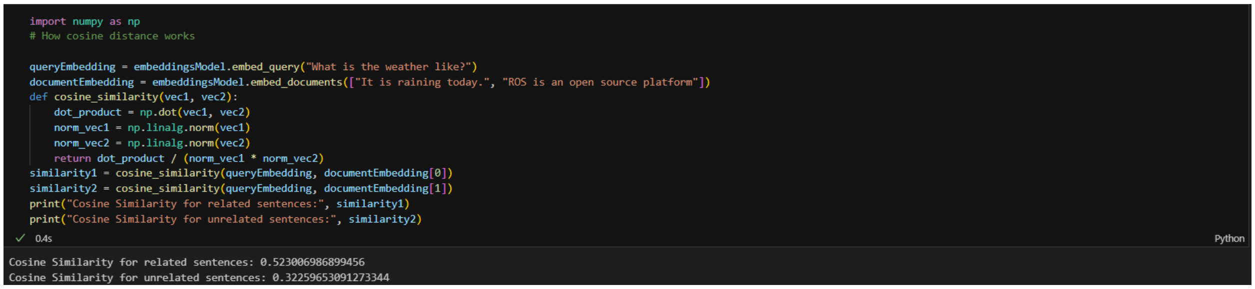
The image below shows the second cell of Tools/qdrantTools.ipynb being used to show how many embeddings were completed.



```
collections = mongoDatabase.list_collection_names()
for collection in collections:
    mongoCollection = mongoDatabase[collection]
    results = mongoCollection.find()
    for result in results:
        # For each document, split it into chunks
        texts.append(result["content"])

cleanTexts = []
for text in texts:
    cleanTexts.append("".join(char for char in text if 32 <= ord(char) <= 126))

numChunks = 0
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=500,
    chunk_overlap=20,
    length_function=len,
    is_separator_regex=False,
)
for text in cleanTexts:
    textChunks = text_splitter.split_text(text)
    for chunk in textChunks:
        numChunks += 1

print("Total number of chunks to embed: ", numChunks)
print("Chunks currently embedded: ", numDocumentChunks+numGithubChunks)
```

```
Total number of chunks to embed:  285569
Chunks currently embedded:  18400
```

The third cell shows a proof of concept for cosine distance(used to retrieve embeddings)

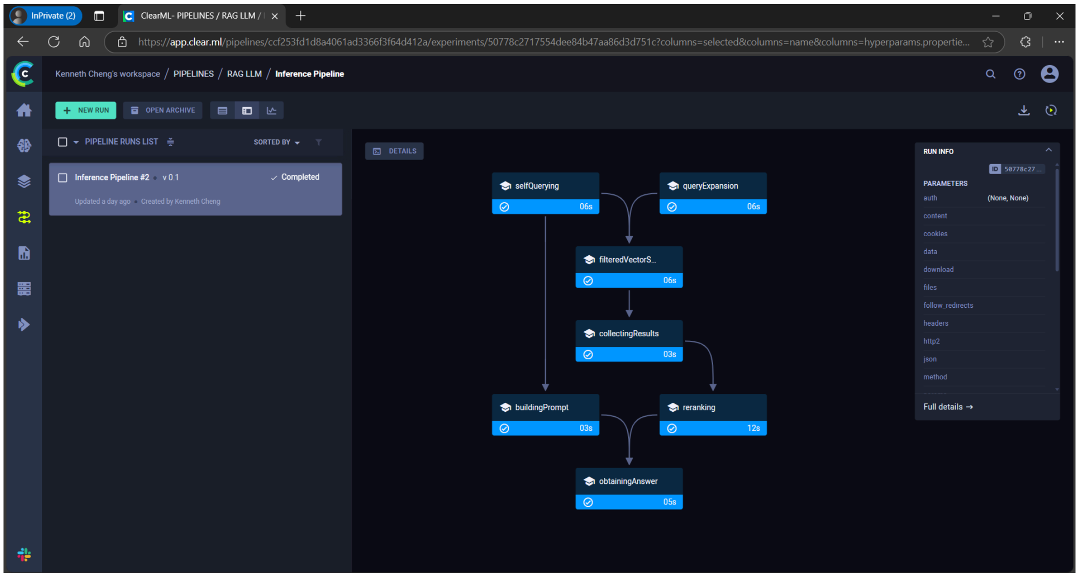


```
import numpy as np
# How cosine distance works

queryEmbedding = embeddingsModel.embed_query("What is the weather like?")
documentEmbedding = embeddingsModel.embed_documents(["It is raining today.", "ROS is an open source platform"])
def cosine_similarity(vec1, vec2):
    dot_product = np.dot(vec1, vec2)
    norm_vec1 = np.linalg.norm(vec1)
    norm_vec2 = np.linalg.norm(vec2)
    return dot_product / (norm_vec1 * norm_vec2)
similarity1 = cosine_similarity(queryEmbedding, documentEmbedding[0])
similarity2 = cosine_similarity(queryEmbedding, documentEmbedding[1])
print("Cosine Similarity for related sentences:", similarity1)
print("Cosine Similarity for unrelated sentences:", similarity2)
```

```
Cosine Similarity for related sentences: 0.523006986899456
Cosine Similarity for unrelated sentences: 0.32259653091273344
```

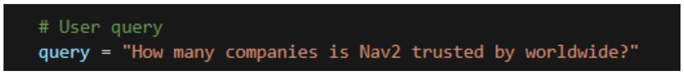
Deploying the App Milestone

To generate an answer: the ClearML/InferencePipeline.py file performs query expansion to generate a related query (rewording query may lead to retrieving more useful embeddings), performs self-querying to get metadata about the query, searches for related vectors/embeddings compared to the query and related query(from query expansion), concatenates the results of the searches, reranks the results to filter out the most useful ones, builds a prompt using the query metadata and results, then feeds the prompt to the model to return the answer. The image below shows the ClearML website giving a overview of what this process looks like

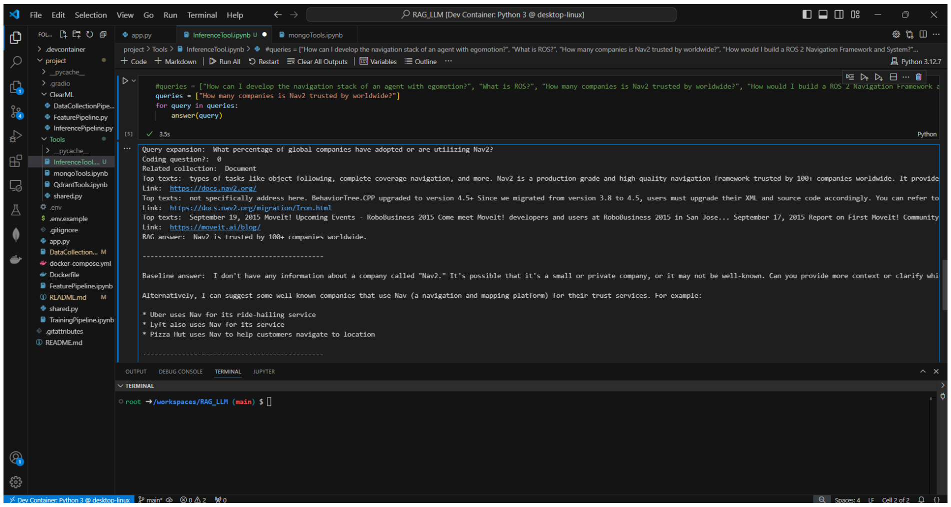


ClearML - PIPELINES / RAG LLM /
https://app.clear.ml/pipelines/ccf253fd1d8a4061ad3366f3f64d412a/experiments/50778c2717554dee84b47aa86d3d751c?columns=selected&columns=name&columns=hyperparams.propertie...
Kenneth Cheng's workspace / PIPELINES / RAG LLM / Inference Pipeline
NEW RUN
OPEN ARCHIVE
PIPELINE RUNS LIST
SORTED BY
Inference Pipeline #2 v 0.1
Completed
Updated a day ago · Created by Kenneth Cheng
DETAILS
selfQuerying 06s
queryExpansion 06s
filteredVectorS... 06s
collectingResults 03s
buildingPrompt 03s
reranking 12s
obtainingAnswer 05s
RUN INFO
50778c27...
PARAMETERS
auth (None, None)
content
cookies
data
download
files
follow_redirects
headers
http2
json
method
Full details →

The image below shows Tools/InferenceTool.ipynb showing the output of each step that was used to generate an answer to a sample prompt along with a comparison to the original model. You can change the query to whatever you want and it will give a similar answer as gradio. The first link confirms that the answer is correct while the baseline model did not know the answer!



```python
# User query
query = "How many companies is Nav2 trusted by worldwide?"
```



File Edit Selection View Go Run Terminal Help
RAG_LLM [Dev Container: Python 3 @ desktop-linux]
app.py
InferenceTool.ipynb U
mongoTools.ipynb
project > Tools > InferenceTool.ipynb > #queries = ["How can I develop the navigation stack of an agent with egomotion?", "What is ROS?", "How many companies is Nav2 trusted by worldwide?", "How would I build a ROS 2 Navigation Framework and System?"...
+ Code + Markdown Run All Restart Clear All Outputs Variables Outline
Python 3.12.7
.devcontainer
project
__pycache__
.gradio
ClearML
DataCollectionPipe...
FeaturePipeline.py
InferencePipeline.py
Tools
__pycache__
InferenceTool... U
mongoTools.ipynb
QdrantTools.ipynb
shared.py
.env
.env.example
.gitignore
app.py
DataCollection... M
docker-compose.yml
Dockerfile
FeaturePipeline.ipynb
README.md M
shared.py
TrainingPipeline.ipynb
.gitattributes
README.md
OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
TERMINAL
Dev Container: Python 3 @ desktop-linux
main*
Spaces: 4 LF Cell 2 of 2

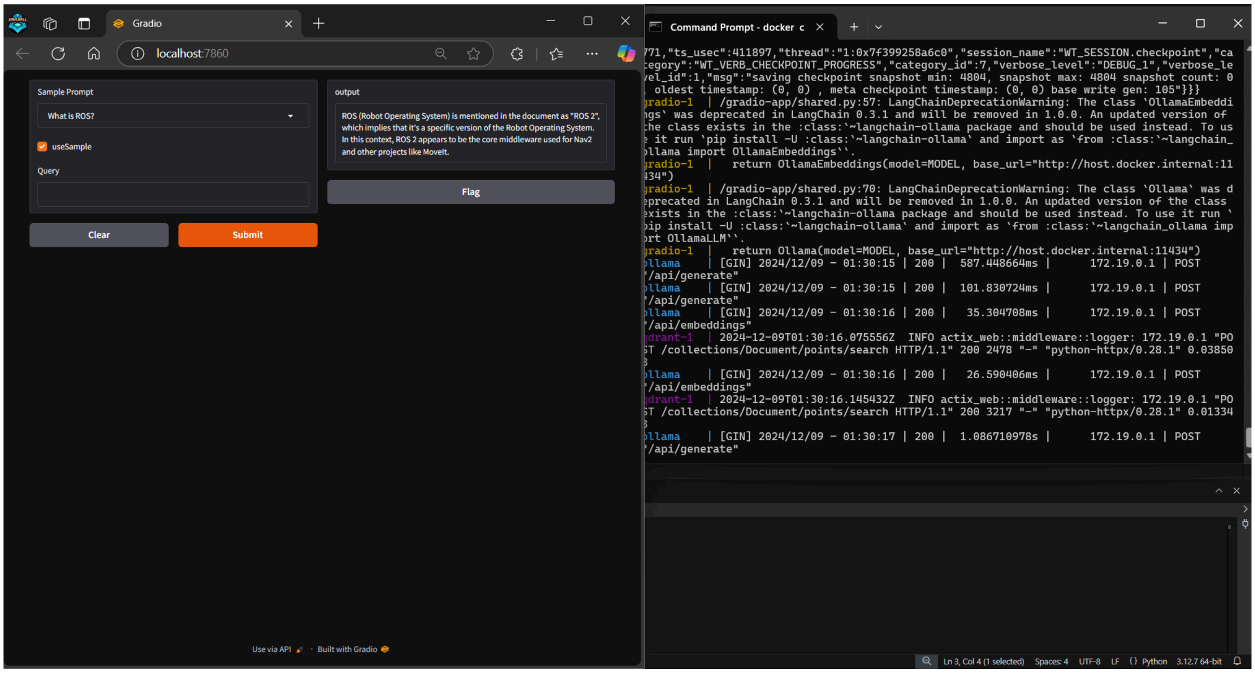
```python
#queries = ["How can I develop the navigation stack of an agent with egomotion?", "What is ROS?", "How many companies is Nav2 trusted by worldwide?", "How would I build a ROS 2 Navigation Framework a
queries = ["How many companies is Nav2 trusted by worldwide?"]
for query in queries:
    answer(query)
```

[5] ✓ 3.5s

```
Query expansion:  What percentage of global companies have adopted or are utilizing Nav2?
Coding question?:  0
Related collection:  Document
Top texts:  types of tasks like object following, complete coverage navigation, and more. Nav2 is a production-grade and high-quality navigation framework trusted by 100+ companies worldwide. It provide
Link:  https://docs.nav2.org/
Top texts:  not specifically address here. BehaviorTree.CPP upgraded to version 4.5+ Since we migrated from version 3.8 to 4.5, users must upgrade their XML and source code accordingly. You can refer to
Link:  https://docs.nav2.org/migration/Iron.html
Top texts:  September 19, 2015 MoveIt! Upcoming Events - RoboBusiness 2015 Come meet MoveIt! developers and users at RoboBusiness 2015 in San Jose... September 17, 2015 Report on First MoveIt! Community
Link:  https://moveit.ai/blog/
RAG answer:  Nav2 is trusted by 100+ companies worldwide.

---------------------------------------------

Baseline answer:  I don't have any information about a company called "Nav2." It's possible that it's a small or private company, or it may not be well-known. Can you provide more context or clarify whi

Alternatively, I can suggest some well-known companies that use Nav (a navigation and mapping platform) for their trust services. For example:

* Uber uses Nav for its ride-hailing service
* Lyft also uses Nav for its service
* Pizza Hut uses Nav to help customers navigate to location

---------------------------------------------
```

```
root ➜ /workspaces/RAG_LLM (main) $ 
```

The image below shows the gradio app being run and its response to a sample query. Additionally, you can see that the docker containers from docker compose up responded when the query was submitted. Alternatively, you can launch app.py, or even run the Tools/InferenceTools.ipynb file with a custom query. There are many ways to get an output as long as ollama is being served either by docker or in your terminal.


Gradio
localhost:7860
Clear
Submit
Flag
Use via API · Built with Gradio
Command Prompt - docker c
Ln 3, Col 4 (1 selected)   Spaces: 4   UTF-8   LF   {} Python   3.12.7 64-bit

Sample Prompt: What is ROS?

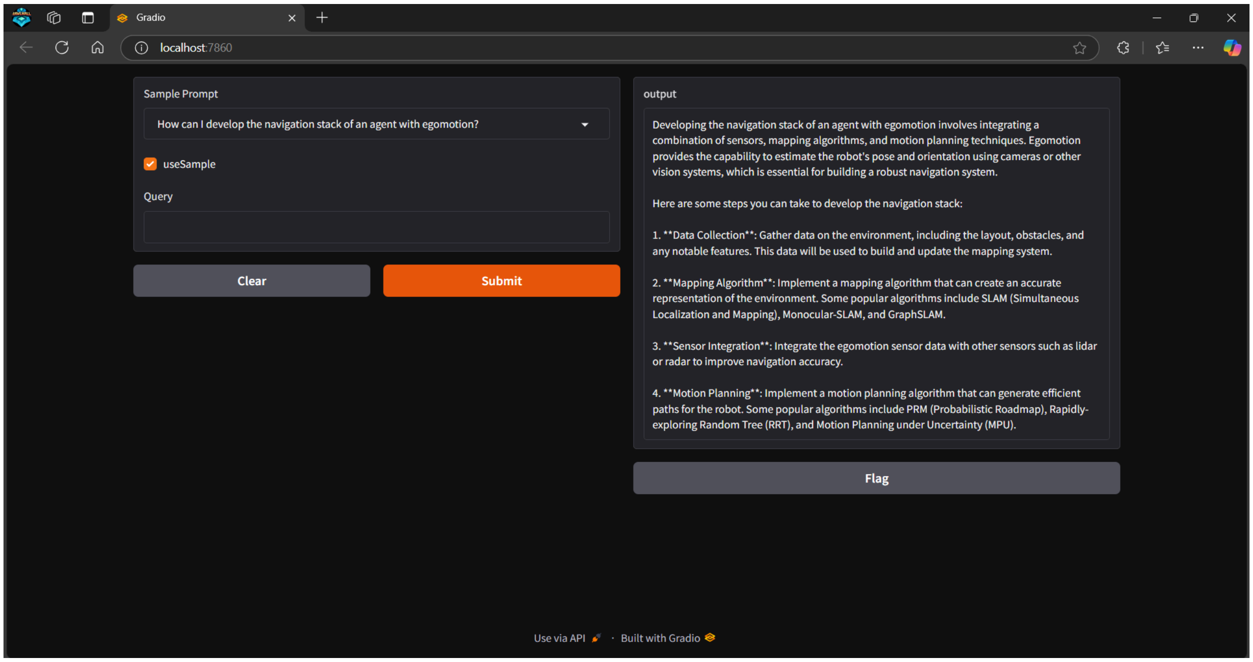
☑ useSample

Query:

output: ROS (Robot Operating System) is mentioned in the document as "ROS 2", which implies that it's a specific version of the Robot Operating System. In this context, ROS 2 appears to be the core middleware used for Nav2 and other projects like MoveIt.

```
771,"ts_usec":411897,"thread":"1:0x7f399258a6c0","session_name":"WT_SESSION.checkpoint","ca
tegory":"WT_VERB_CHECKPOINT_PROGRESS","category_id":7,"verbose_level":"DEBUG_1","verbose_le
vel_id":1,"msg":"saving checkpoint snapshot min: 4804, snapshot max: 4804 snapshot count: 0
, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 105"}}}
gradio-1  | /gradio-app/shared.py:57: LangChainDeprecationWarning: The class `OllamaEmbeddi
ngs` was deprecated in LangChain 0.3.1 and will be removed in 1.0.0. An updated version of
the class exists in the :class:`~langchain-ollama package and should be used instead. To us
e it run `pip install -U :class:`~langchain-ollama` and import as `from :class:`~langchain_
ollama import OllamaEmbeddings``.
gradio-1  |   return OllamaEmbeddings(model=MODEL, base_url="http://host.docker.internal:11
434")
gradio-1  | /gradio-app/shared.py:70: LangChainDeprecationWarning: The class `Ollama` was d
eprecated in LangChain 0.3.1 and will be removed in 1.0.0. An updated version of the class
exists in the :class:`~langchain-ollama package and should be used instead. To use it run `
pip install -U :class:`~langchain-ollama` and import as `from :class:`~langchain_ollama imp
ort OllamaLLM``.
gradio-1  |   return Ollama(model=MODEL, base_url="http://host.docker.internal:11434")
ollama    | [GIN] 2024/12/09 - 01:30:15 | 200 |  587.448664ms |      172.19.0.1 | POST
"/api/generate"
ollama    | [GIN] 2024/12/09 - 01:30:15 | 200 |  101.830724ms |      172.19.0.1 | POST
"/api/generate"
ollama    | [GIN] 2024/12/09 - 01:30:16 | 200 |   35.304708ms |      172.19.0.1 | POST
"/api/embeddings"
qdrant-1  | 2024-12-09T01:30:16.075556Z  INFO actix_web::middleware::logger: 172.19.0.1 "PO
ST /collections/Document/points/search HTTP/1.1" 200 2478 "-" "python-httpx/0.28.1" 0.03850
3
ollama    | [GIN] 2024/12/09 - 01:30:16 | 200 |   26.590406ms |      172.19.0.1 | POST
"/api/embeddings"
qdrant-1  | 2024-12-09T01:30:16.145432Z  INFO actix_web::middleware::logger: 172.19.0.1 "PO
ST /collections/Document/points/search HTTP/1.1" 200 3217 "-" "python-httpx/0.28.1" 0.01334
3
ollama    | [GIN] 2024/12/09 - 01:30:17 | 200 |  1.086710978s |      172.19.0.1 | POST
"/api/generate"
```

The images below show the model's responses to various queries



Sample Prompt

How can I develop the navigation stack of an agent with egomotion?

☑ useSample

Query

output

Developing the navigation stack of an agent with egomotion involves integrating a combination of sensors, mapping algorithms, and motion planning techniques. Egomotion provides the capability to estimate the robot's pose and orientation using cameras or other vision systems, which is essential for building a robust navigation system.

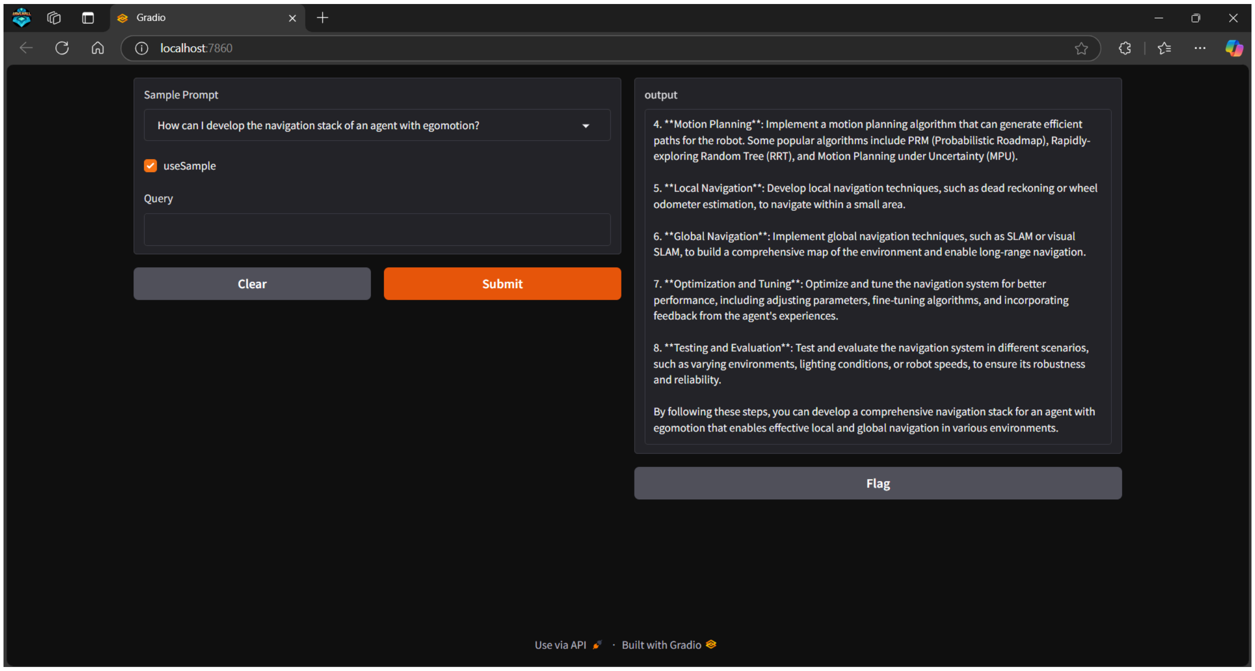
Here are some steps you can take to develop the navigation stack:

1. **Data Collection**: Gather data on the environment, including the layout, obstacles, and any notable features. This data will be used to build and update the mapping system.

2. **Mapping Algorithm**: Implement a mapping algorithm that can create an accurate representation of the environment. Some popular algorithms include SLAM (Simultaneous Localization and Mapping), Monocular-SLAM, and GraphSLAM.

3. **Sensor Integration**: Integrate the egomotion sensor data with other sensors such as lidar or radar to improve navigation accuracy.

4. **Motion Planning**: Implement a motion planning algorithm that can generate efficient paths for the robot. Some popular algorithms include PRM (Probabilistic Roadmap), Rapidly-exploring Random Tree (RRT), and Motion Planning under Uncertainty (MPU).



Sample Prompt

How can I develop the navigation stack of an agent with egomotion?

☑ useSample

Query

output

4. **Motion Planning**: Implement a motion planning algorithm that can generate efficient paths for the robot. Some popular algorithms include PRM (Probabilistic Roadmap), Rapidly-exploring Random Tree (RRT), and Motion Planning under Uncertainty (MPU).

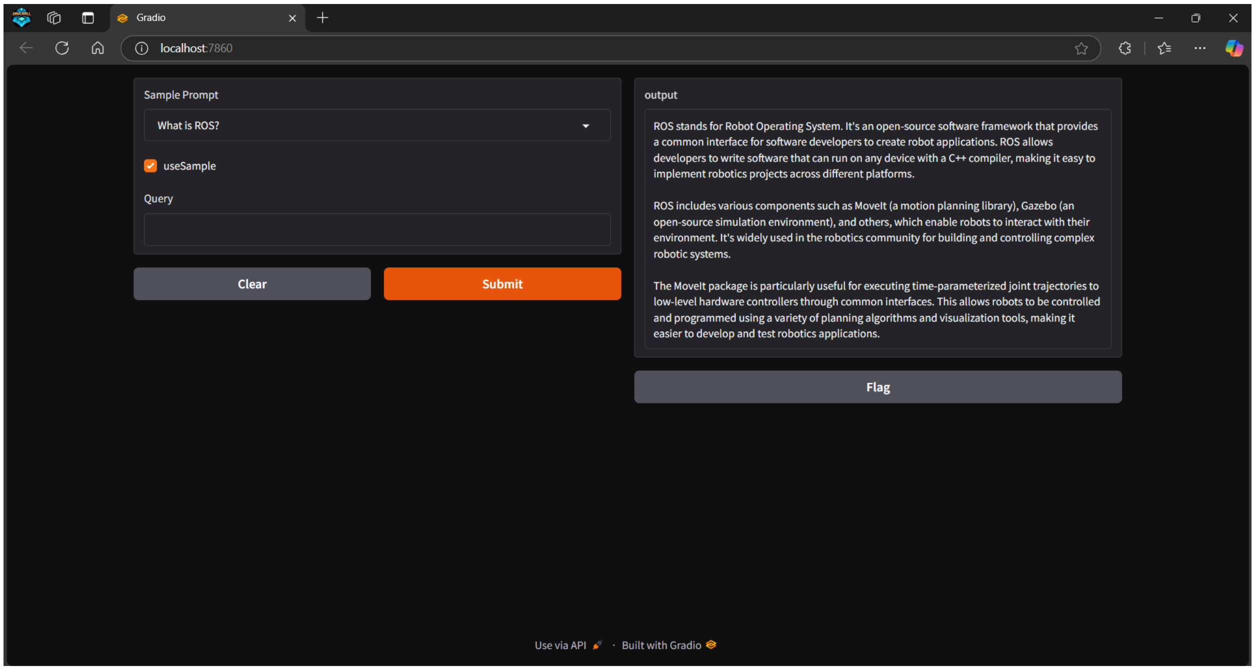
5. **Local Navigation**: Develop local navigation techniques, such as dead reckoning or wheel odometer estimation, to navigate within a small area.

6. **Global Navigation**: Implement global navigation techniques, such as SLAM or visual SLAM, to build a comprehensive map of the environment and enable long-range navigation.

7. **Optimization and Tuning**: Optimize and tune the navigation system for better performance, including adjusting parameters, fine-tuning algorithms, and incorporating feedback from the agent's experiences.

8. **Testing and Evaluation**: Test and evaluate the navigation system in different scenarios, such as varying environments, lighting conditions, or robot speeds, to ensure its robustness and reliability.

By following these steps, you can develop a comprehensive navigation stack for an agent with egomotion that enables effective local and global navigation in various environments.



Sample Prompt

What is ROS?

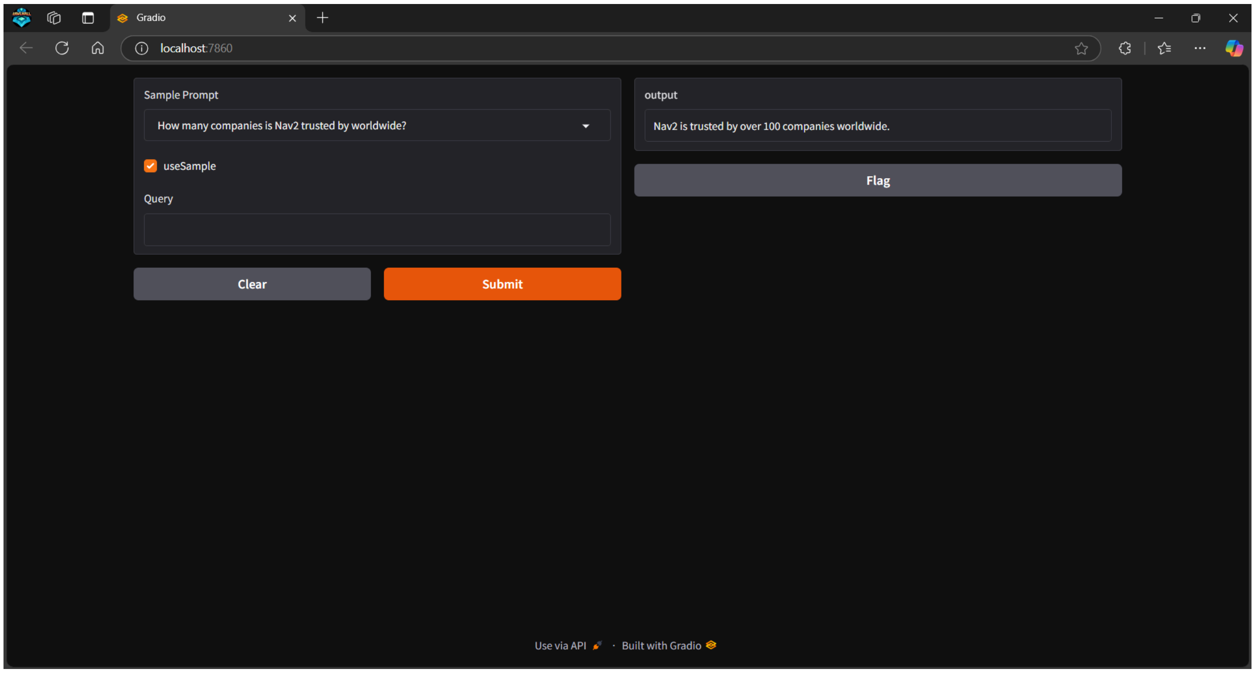
☑ useSample

output

ROS stands for Robot Operating System. It's an open-source software framework that provides a common interface for software developers to create robot applications. ROS allows developers to write software that can run on any device with a C++ compiler, making it easy to implement robotics projects across different platforms.

ROS includes various components such as MoveIt (a motion planning library), Gazebo (an open-source simulation environment), and others, which enable robots to interact with their environment. It's widely used in the robotics community for building and controlling complex robotic systems.

The MoveIt package is particularly useful for executing time-parameterized joint trajectories to low-level hardware controllers through common interfaces. This allows robots to be controlled and programmed using a variety of planning algorithms and visualization tools, making it easier to develop and test robotics applications.



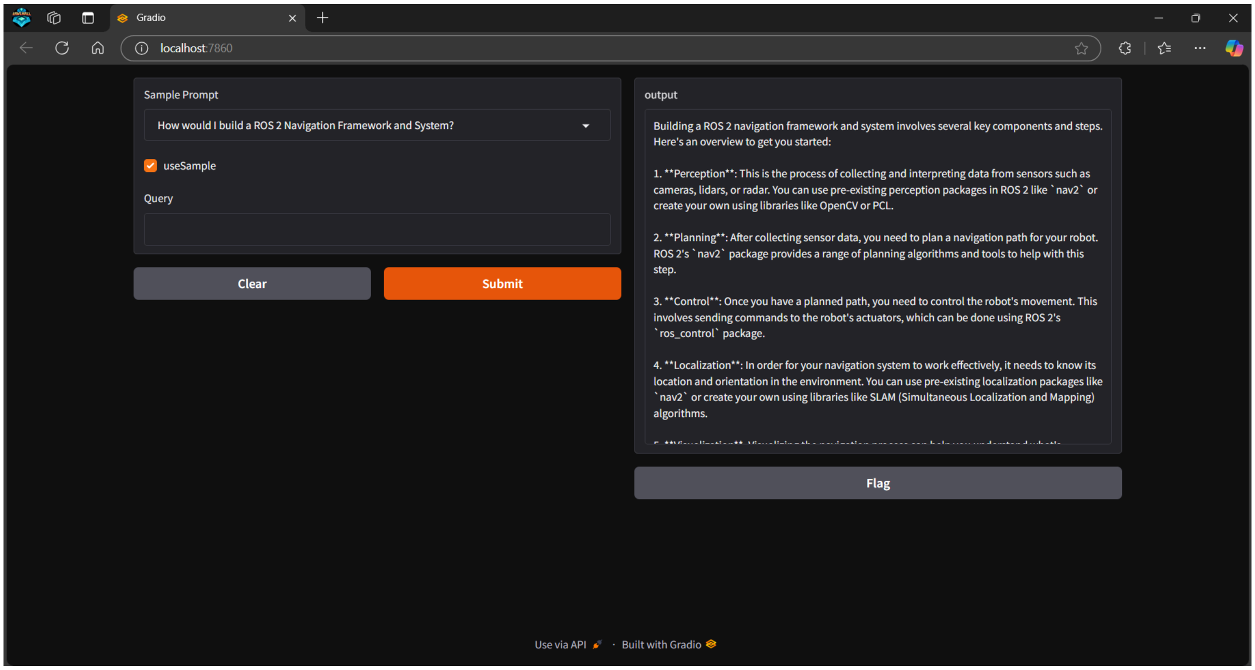
Sample Prompt

How many companies is Nav2 trusted by worldwide?

☑ useSample

output

Nav2 is trusted by over 100 companies worldwide.



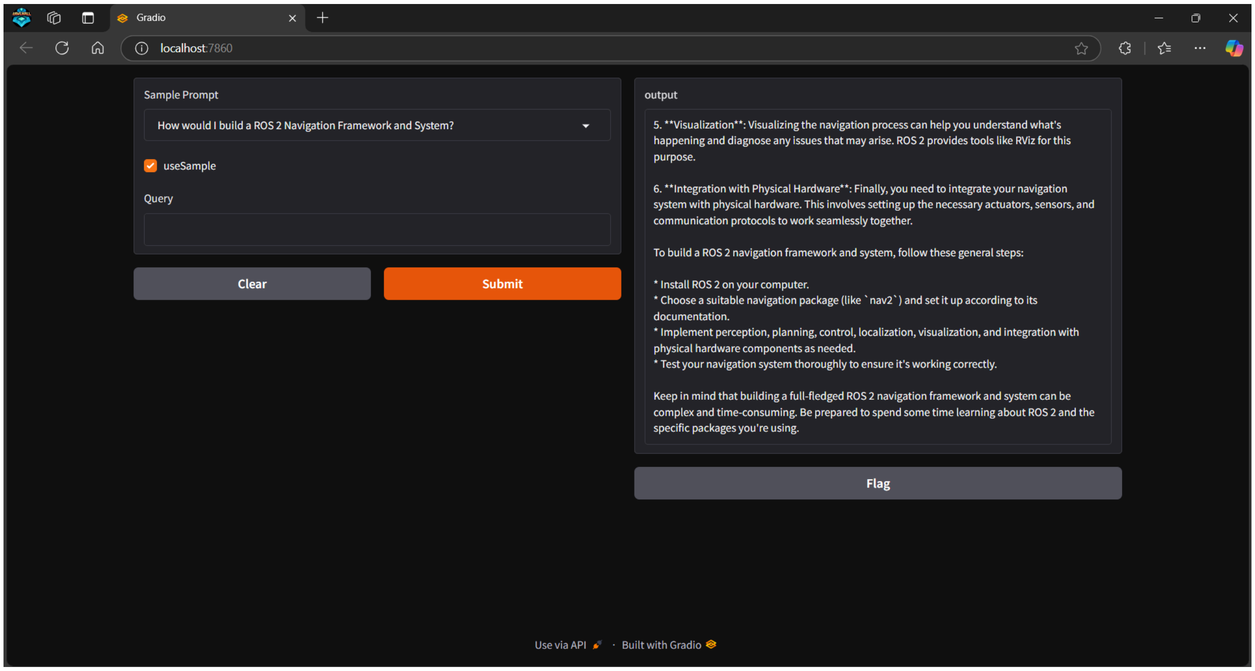
Sample Prompt: How would I build a ROS 2 Navigation Framework and System?

☑ useSample

Query

output

```
Building a ROS 2 navigation framework and system involves several key components and steps.
Here's an overview to get you started:

1. **Perception**: This is the process of collecting and interpreting data from sensors such as
cameras, lidars, or radar. You can use pre-existing perception packages in ROS 2 like `nav2` or
create your own using libraries like OpenCV or PCL.

2. **Planning**: After collecting sensor data, you need to plan a navigation path for your robot.
ROS 2's `nav2` package provides a range of planning algorithms and tools to help with this
step.

3. **Control**: Once you have a planned path, you need to control the robot's movement. This
involves sending commands to the robot's actuators, which can be done using ROS 2's
`ros_control` package.

4. **Localization**: In order for your navigation system to work effectively, it needs to know its
location and orientation in the environment. You can use pre-existing localization packages like
`nav2` or create your own using libraries like SLAM (Simultaneous Localization and Mapping)
algorithms.
```



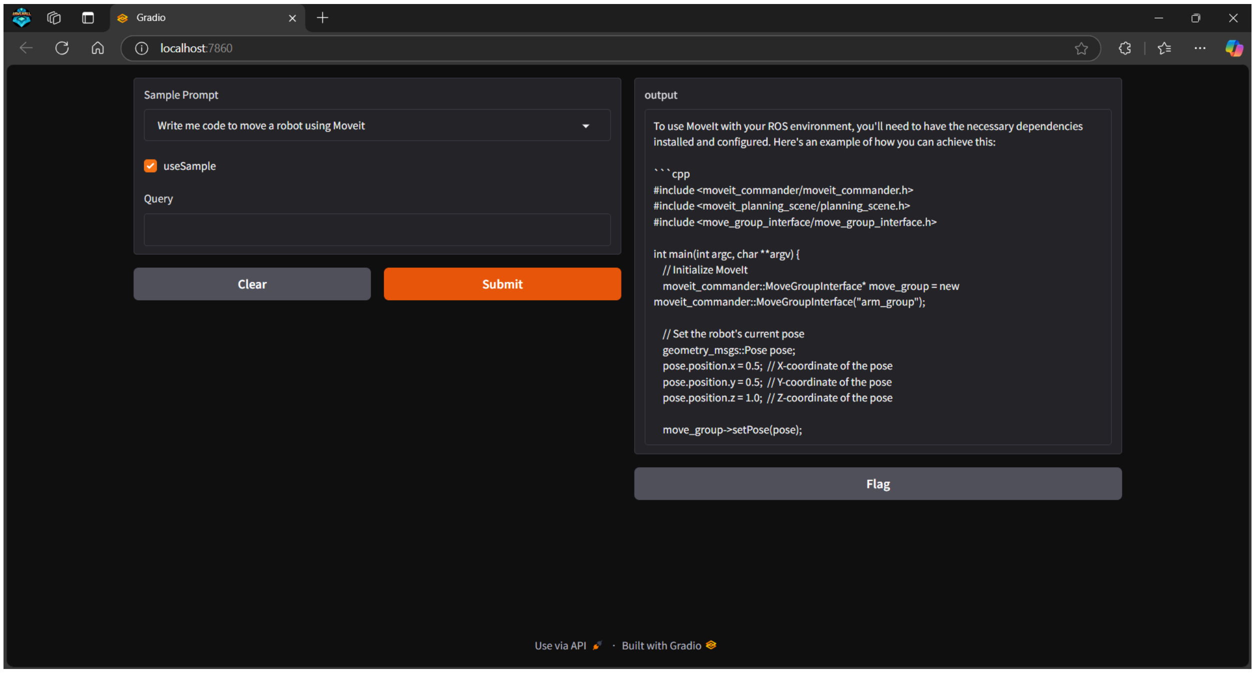
Sample Prompt: How would I build a ROS 2 Navigation Framework and System?

☑ useSample

Query

output

```
5. **Visualization**: Visualizing the navigation process can help you understand what's
happening and diagnose any issues that may arise. ROS 2 provides tools like RViz for this
purpose.

6. **Integration with Physical Hardware**: Finally, you need to integrate your navigation
system with physical hardware. This involves setting up the necessary actuators, sensors, and
communication protocols to work seamlessly together.

To build a ROS 2 navigation framework and system, follow these general steps:

* Install ROS 2 on your computer.
* Choose a suitable navigation package (like `nav2`) and set it up according to its
documentation.
* Implement perception, planning, control, localization, visualization, and integration with
physical hardware components as needed.
* Test your navigation system thoroughly to ensure it's working correctly.

Keep in mind that building a full-fledged ROS 2 navigation framework and system can be
complex and time-consuming. Be prepared to spend some time learning about ROS 2 and the
specific packages you're using.
```



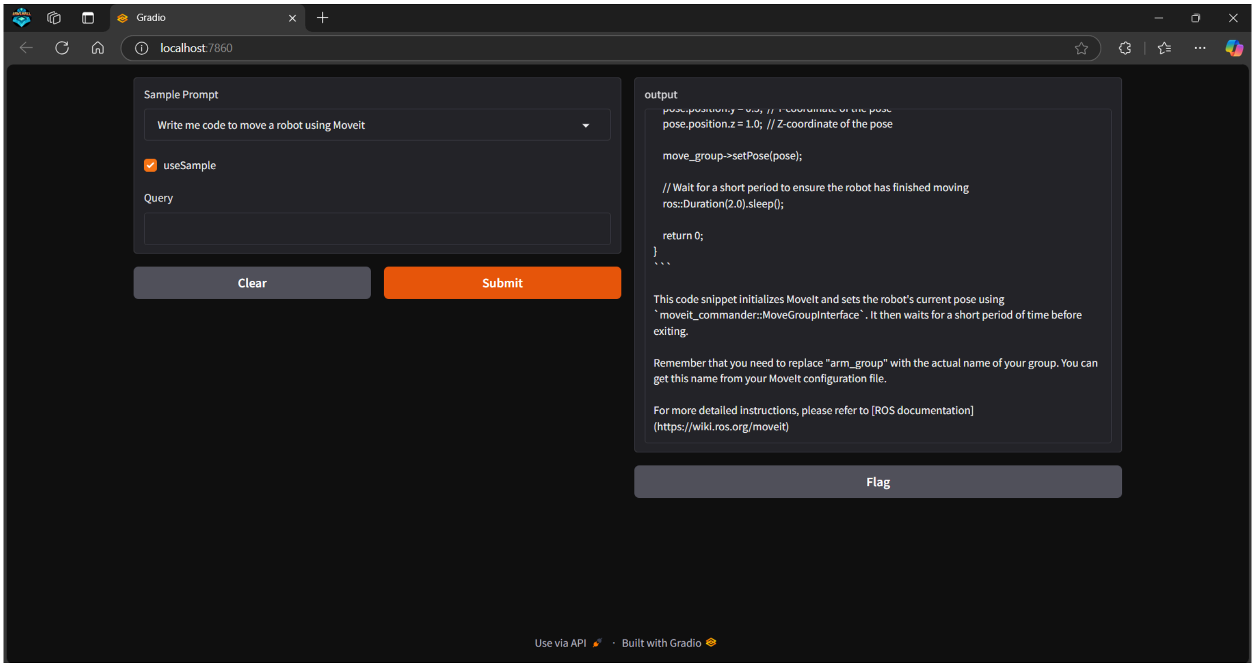
Sample Prompt: Write me code to move a robot using MoveIt

☑ useSample

Query:

output

~~~~
To use MoveIt with your ROS environment, you'll need to have the necessary dependencies
installed and configured. Here's an example of how you can achieve this:

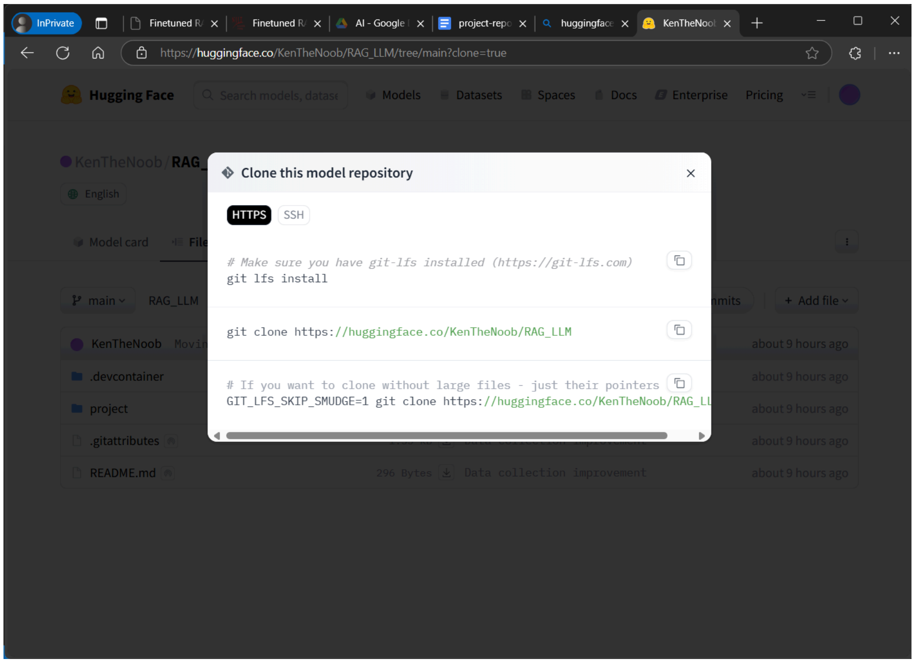
```cpp
#include <moveit_commander/moveit_commander.h>
#include <moveit_planning_scene/planning_scene.h>
#include <move_group_interface/move_group_interface.h>

int main(int argc, char **argv) {
   // Initialize MoveIt
   moveit_commander::MoveGroupInterface* move_group = new
moveit_commander::MoveGroupInterface("arm_group");

   // Set the robot's current pose
   geometry_msgs::Pose pose;
   pose.position.x = 0.5;  // X-coordinate of the pose
   pose.position.y = 0.5;  // Y-coordinate of the pose
   pose.position.z = 1.0;  // Z-coordinate of the pose

   move_group->setPose(pose);
~~~~



Sample Prompt: Write me code to move a robot using MoveIt

☑ useSample

Query:

output

~~~~
   pose.position.z = 1.0;  // Z-coordinate of the pose

   move_group->setPose(pose);

   // Wait for a short period to ensure the robot has finished moving
   ros::Duration(2.0).sleep();

   return 0;
}
```

This code snippet initializes MoveIt and sets the robot's current pose using
`moveit_commander::MoveGroupInterface`. It then waits for a short period of time before
exiting.

Remember that you need to replace "arm_group" with the actual name of your group. You can
get this name from your MoveIt configuration file.

For more detailed instructions, please refer to [ROS documentation]
(https://wiki.ros.org/moveit)
~~~~

The image below shows the model uploaded and pullable on huggingface



Clone this model repository

```
# Make sure you have git-lfs installed (https://git-lfs.com)
git lfs install
```

```
git clone https://huggingface.co/KenTheNoob/RAG_LLM
```

```
# If you want to clone without large files - just their pointers
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/KenTheNoob/RAG_LL
```

Alternatively, you can pull from my github in the README.md, in which case, you start the devcontainer inside the project folder, not the base folder.