*Lab 06 Building Question-Answering with watsonx.ai and Streamlit with Retrieval Augmented Generation*
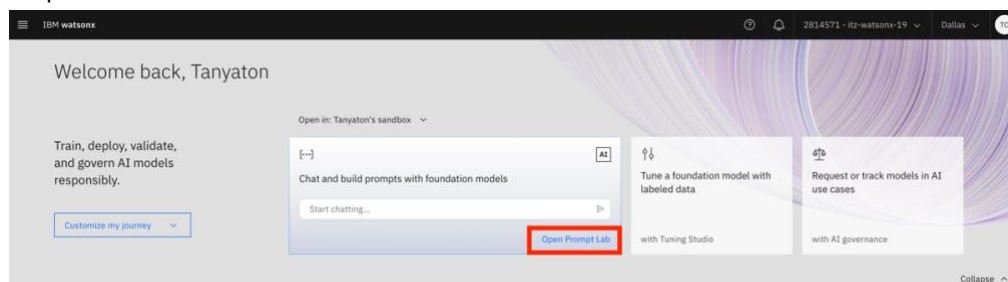
# Level 1: Using watsonx.ai prompt lab to build RAG application

## 1.    RAG application with Watsonx.ai Prompt Lab

In this step, you will experience watsonx.ai Prompt lab document grounding feature, which can perform RAG out of the box.
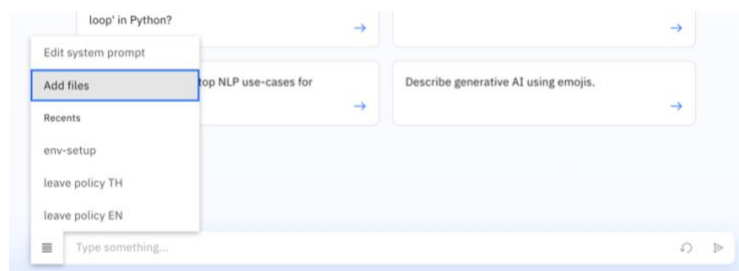
1.1 Go to you watsonx.ai and login with IBM id you created
1.2 Go to prompt lab



1.3  Add file **'leave.pdf'** to watsonx.ai chat

Go to Chat Menu > Add file > Browse file > Enter Name > Create

Choose **embedding model** and **chunk size** to embed your document in **Advance setting. Please take note of the embedding model you are using for the next steps.** In this example we will be using *ibm/slate-125m-english-rtrvr*, you can also choose other embedding models.

1.4 After the document is uploaded in your watsonx.ai asset, try asking a question about the document you added in prompt lab.

Here is a list of questions related to the document **leave.pdf**. Please also review the document and try additional questions

| |
|---|
| How many types of leave are there? Plese make a list |
| What do I do if I want to take some days off because I am sick? |
| How many days are allowed for casual leave? |
| Can I leave for a week and still get paid? |
| can i transfer my remaining casual and earned leave to next year? |
| What are the conditions for maternity leave? |
| Would I still get paid for maternity leave? |
| When is the earliest I can take maternity leave |

1.5 Repeat the steps 3 and 4 with different documents, try asking question related to the document and observe the performance

## 2.    Prompt Notebook with Chat - Prompt Lab Notebook v1.1.0

In this second step, we will look deeper into the code behind the application, enabling you to create your own version later in this lab.

### 2.1 Download Prompt Notebook

2.2.1 Go to 'save' symbol on top right corner >save as

**2.2.2** Choose save your work as **Notebook,** and choose **View project after saving.**



**2.2.3** The notebook will be shown in your browser, you can also access your saved notebooks from your **sandbox project** on your Homepage
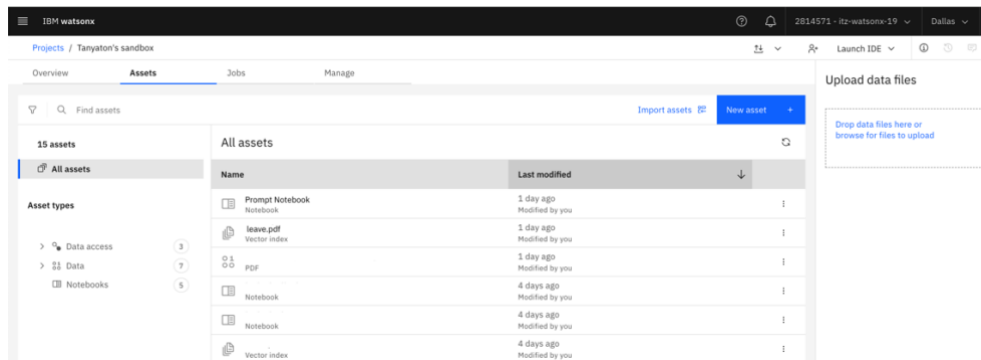


**2.2.4** Go to ✎ , try running and read through the notebook, use your *watsonx_api_key* while running through the lab. In this notebook, ***chroma db*** is used for vector database

**2.2.5** Under **Defining Vector Index** section, <span style="color:red">**please take a note of your** *vector_index_id* **to use in later steps**</span>

## Defining the vector index

Initialize the vector index to query when chatting with the model.

```
In [ ]:  from ibm_watsonx_ai.client import APIClient

         from ibm_watsonx_ai.foundation_models import Embeddings
         from ibm_watsonx_ai.foundation_models.utils.enums import EmbeddingTypes

         emb = Embeddings(
             model_id=vector_index_properties["settings"]["embedding_model_id"],
             credentials=wml_credentials,
             project_id=project_id,
             params={
                 "truncate_input_tokens": 512
             }
         )

         wml_credentials = get_credentials()
         client = APIClient(credentials=wml_credentials, project_id=project_id, space_id=space_id)

         vector_index_id = "6d4ac7c8-a0bb-41be-85b3-575253119fa1"
         vector_index_details = client.data_assets.get_details(vector_index_id)
         vector_index_properties = vector_index_details["entity"]["vector_index"]
```

# 3.    Create your own Question-Answering app using Watsonx.ai dataset

After we have experience question answering page created by [watsonx.ai](watsonx.ai), it's time we create our own version! We will be using the same database from the document we already uploaded to our [watsonx.ai](watsonx.ai) assets, but with different webpage we created using **streamlit**

## 3.1 Store your data in Vector Database

3.1.1    In `ingestion.py`, fill in the *vector_index_id* you obtained from step 2.2.5, and *model_id_emb* for an embedding model you used

```
ingestion.py ×
 - Building Question-Answering with watsonx.ai and Streamlit with Retrieval Augmented Generation (Transient) > Level_1_English > INGESTION >
  1    from function import initiate_username, connect_watsonx_embedding, hydrade_milvus, connect_to_milvus
  2
  3    #---------- settings ----------- #
  4
  5    model_id_emb = "" # Example "ibm/slate-125m-english-rtrvr"
  6    vector_index_id = "" # Example "d56b0957-942c-4059-baac-436d15a3b288"
  7    #------------------------------------------------------------------
  8
  9    collection_name = initiate_username()
 10    client, model_emb = connect_watsonx_embedding(model_id_emb)
 11    connect_to_milvus()
 12    collection = hydrade_milvus(collection_name, client, vector_index_id, model_id_emb)
 13
 14    print("The Collection Name is: ", collection_name)
 15    with open("collection_name.txt", "w") as file:
 16        file.write(collection_name)
```

3.1.2    Open a new terminal and run `**podman exec -it incubation /bin/bash**` to execute the container

```
(genai3) → gen_ai_incubation_watsonx_th git:(lab6) × podman exec -it incubation  /bin/bash
root@3e4cf521210a:/usr/src/app#
```

3.1.3    cd to your lab's base deirectory, then run `**python ingestion.py**` on your terminal to start ingesting documents from watsonx to our platform



3.1.4    Here, `**collection_name.txt**` will be generated. The file contains the name of your milvus collection. The name also shows on your terminal. Please keep note of this collection name

## 3.2 Starting the app

3.2.1    cd into this lab's base directory then locate to MAINAPP folder

3.2.2    Put `cert.pem` file you receive from you email inside the MAINAPP folder

3.2.3    Edit the *settings* section in `app.py`. Change the *model_id_emb*, *vector_index_id* (from step 2.2.5), *collection_name* (from step 3.2.3)

```
app.py    ×
TH > lab 06 - Building Question-Answering with watsonx.ai and Streamlit with Retrieval Augmented Generation (Transient) > Level_1_English
 1  import logging
 2  import os
 3  import streamlit as st
 4  from pymilvus import Collection
 5  from function import (generate_prompt_en, connect_watsonx_embedding, display_streaming_response,
 6                        connect_watsonx_llm, connect_to_milvus, find_answer,
 7                        display_response, display_hits_dataframe)
 8
 9  #---------- settings ---------- #
10
11  model_id_llm = 'mistralai/mixtral-8x7b-instruct-v01'
12  model_id_emb = "" # Example "ibm/slate-125m-english-rtrvr"
13  vector_index_id = "" # Example "d56b0957-942c-4059-baac-436d15a3b288"
14  collection_name = "" # Choose the collection name generated from ingestion.py, Example jEEODDYY1aSc
15
16  #==============================================================================
```

3.2.4    Go to `function.py` file, and adjust the dimension to match the embedding model you have chosen in step 1.3 according to the table below. The default value is 384.

### IBM embedding models

The following table lists the supported embedding models that IBM provides.

*Table 1. IBM embedding model in watsonx.ai*

| Model name | API model_id | Billing class | Maximum input tokens | Number of dimensions | More information |
|---|---|---|---|---|---|
| slate-125m-english-rtrvr | ibm/slate-125m-english-rtrvr | Class C1 | 512 | 768 | Model card |
| slate-30m-english-rtrvr | ibm/slate-30m-english-rtrvr | Class C1 | 512 | 384 | Model card |

### Third-party embedding models 🔗

The following table lists the supported third-party embedding models.

*Table 2. Supported third-party embedding model in watsonx.ai*

| Model name | API model_id | Provider | Billing class | Maximum input tokens | Number of dimensions | More information |
|---|---|---|---|---|---|---|
| all-minilm-l12-v2 | sentence-transformers/all-minilm-l12-v2 | Open source natural language processing (NLP) and computer vision (CV) community | Class C1 | 256 | 384 | • Model card |
| multilingual-e5-large | intfloat/multilingual-e5-large | Microsoft | Class C1 | 512 | 1024 | • Model card <br> • Research paper |

- For a list of which models are provided in each regional data center, see Regional availability of foundation models.
- For information about billing classes, see Watson Machine Learning plans.

```
function.py M ×
TH > lab 06 - Building Question-Answering with watsonx.ai and Streamlit with Retrieval Augmented Generation (Transient) > Level_1_Eng
                                                                                              > creat
 1  import gzip
 2  import json
 3  import operator
 4  import string
 5  import random
 6  from pymilvus import connections, utility, Collection, CollectionSchema, FieldSchema,DataType
 7  import os
 8  from dotenv import load_dotenv
 9  from ibm_watsonx_ai.client import APIClient
10  from ibm_watsonx_ai.foundation_models import Embeddings
11
12  embedding_dimension = 384 #adjust the value according to your choice of embedding
13
```

Check out this link for more details

3.2.5    Run the app by running the command `**streamlit run app.py**` on your terminal.

3.2.6    Go to your browswer and go to http://localhost:8501/ to access local host

After following these steps, you will be able to see your own question-answering web running on your local host. Please try prompting questions related to the document, or use list of sample questions from steps 1.4

For further experiments, you can repeat all the steps with different documents and embedding models to see the contrast.

# Level 2: Using watsonx.ai prompt lab to build RAG application

In Level 1, we build a web application using document uploaded through watsonx.ai. In this level we take it further and building an end-to-end RAG web application that allow user to upload file right in the webpage that support Thai language. In this app, we use *kornwtp/SCT-model-phayathaibert* from sentence transformer as our embedding model, since it supports Thai language.

## Starting the app

1.    cd into this lab's base directory
2.    Put `**cert.pem**` file you receive from you email inside the folder
3.    Run the app by running the command `**streamlit run app.py**` on your terminal.
4.    Go to your browser and go to http://localhost:8501/

After following these steps, you will be able to see your own question-answering web running on your local host. Please try prompting questions related to the document, or use list of sample questions below

| How many types of leave are there? Plese make a list | มีลาประเภทใดบ้าง |
|---|---|
| What do I do if I want to take some days off because I am sick? | ฉันควรทำอย่างไรถ้าฉันต้องลาเนื่องจากเจ็บป่วย |
| How many days are allowed for casual leave? | ฉันสามารถลาพักผ่อนได้กี่วัน |
| Can I leave for a week and still get paid? | ฉันสามารถลาไปหนึ่งสัปดาห์แล้วยังได้รับเงินเดือนได้ไหม |
| can i transfer my remaining casual and earned leave to next year? | ฉันสามารถโอนวันลาพักผ่อนที่เหลือจากปีนี้ออกไปไปปีหน้าได้ไหม |
| What are the conditions for maternity leave? | เงื่อนไขสำหรับการลาคลอดคืออะไร |
| Would I still get paid for maternity leave? | ฉันจะได้รับค่าจ้างในช่วงลาคลอดไหม |
| When is the earliest I can take maternity leave | ฉันสามารถลาคลอดได้เร็วที่สุดเมื่อไร |