

Peer-graded Assignment: Final Project: Build an AI RAG Assistant Using LangChain
 Deadline May 6, 11:59 PM EDT

Ready for the assignment?

You will find instructions below to submit.

Instructions My submission Discussions

Before submitting your work and reviewing your peers, please ensure you have completed all the hands-on labs in the Project: Generative AI Applications with RAG and LangChain course. This project has six concepts, including document loader, text splitter, embedding, vector database, retriever, and Gradio interface, across eight labs for creating a RAG application based on LangChain. You will be graded on a selection of tasks you completed in the labs, plus screenshots displaying the different exercises of your deployed application. You are free to choose any platform to run your project. Additionally, you can use the lab from "Construct a QA Bot that Leverages LangChain and LLM to Answer Questions from Loaded Documents" to run this final project and capture the necessary screenshots.

One or more of your peers will grade your final assignment. Similarly, you will grade one of your peers who completed this capstone project. Your peer evaluator will be provided with the tasks, sample screenshots for tasks that require them, and objective grading criteria so they can determine how many points to assess for each task.

> Scenario

Imagine you work as a consultant for Quest Analytics, a small but fast-growing research organization.

In today's fast-paced research environment, the sheer volume of scientific papers can be overwhelming, making it nearly impossible to stay up-to-date with the latest developments.

The researchers at Quest Analytics have been struggling to find the time to examine countless documents, let alone extract the most relevant and insightful information.

You have been hired to build an AI RAG assistant that can read, understand, and summarize vast amounts of data, all in real time. Follow the below tasks to construct the AI-powered RAG assistant to optimize the research endeavors at Quest Analytics.

> Project tasks and deliverables

The LLM used for the following tasks can be 'mistralai/mixtral-8x7b-instruct-v01' sourced from watsonx.ai API.

Task 1: Load document using LangChain for different sources (10 points)

(This task corresponds with Exercise 1 in the lab "Load Documents Using LangChain for Different Sources" from Module 1)

Capture a screenshot (saved as pdf_loader) that displays both the code used and the first 1000 characters of the content after loading the paper [link](#).

Task 2: Apply text splitting techniques (10 points)

(This task corresponds with Exercise 2 in the lab, "Apply text splitting techniques to enhance model responsiveness.")

Submit a screenshot (saved as 'code_splitter.png') that displays the code used to split the following LATEX code and its corresponding results.

```
latex_text = ""
```

```
\documentclass{article}
```

```
\begin{document}
```

```
\maketitle
```

```
\section{Introduction}
```

Large language models (LLMs) are a type of machine learning model that can be trained on vast amounts of text data to generate human-like language. In recent years, LLMs have made significant advances in various natural language processing tasks, including language translation, text generation, and sentiment analysis.

\subsection{History of LLMs}

The earliest LLMs were developed in the 1980s and 1990s, but they were limited by the amount of data that could be processed and the computational power available at the time. In the past decade, however, advances in hardware and software have made it possible to train LLMs on massive datasets, leading to significant improvements in performance.

\subsection{Applications of LLMs}

LLMs have many applications in the industry, including chatbots, content creation, and virtual assistants. They can also be used in academia for research in linguistics, psychology, and computational linguistics.

\end{document}

""

Task 3: Embed documents (10 points)

(This task corresponds with Exercise 1 in the lab “Embed documents using *watsonx*’s embedding model.”)

Submit a screenshot (saved as ‘embedding.png’) that displays the code used to embed the following sentence and its corresponding results, which display the first five embedding numbers.

query = "How are you?"

Task 4: Create and configure vector databases to store embeddings (10 points)

(This task corresponds with Exercise 1 in the lab “Create and Configure a Vector Database to Store Document Embeddings”)

Submit a screenshot (saved as ‘vectordb.png’) that displays the code used to create a Chroma vector database that stores the embeddings of the document ‘[new-Policies.txt](#)’ and then conduct a similarity search for the following query with the top 5 results used.

query = "Smoking policy"

Task 5: Develop a retriever to fetch document segments based on queries (10 points)

(This task corresponds with Exercise 1 in the lab “Develop a Retriever to Fetch Document Segments based on Queries.”)

Submit a screenshot (saved as ‘retriever.png’) that displays the code used to use ChromaDB as a retriever and conduct a similarity search with the top 2 return results.

The document you can use is ‘[new-Policies.txt](#)’.


The query you can use is:

query = "Email policy"

Task 6: Construct a QA Bot that leverages the LangChain and LLM to answer questions (10 points)

(This task corresponds with the lab “Construct a QA Bot That Leverages the *LangChain* and LLM to Answer Questions from Loaded Documents.”)

Submit a screenshot (saved as 'QA_bot.png') that displays the QA bot interface you created based on the lab "Construct a QA Bot That Leverages the LangChain and LLM to Answer Questions from Loaded Documents." Also, the picture should display that you uploaded a PDF and are asking a query to the bot.

The PDF you can use is available [here](#) .

The query you can use is:

query = "What this paper is talking about?"

➤ Grading Criteria Overview

You are required to submit the following items to get peer-reviewed:

Screenshots of outputs for each task. The number of screenshots/images and points associated with each task is mentioned below:

Task 1: 1 file (10 points)

Task 2: 1 file (10 points)

Task 3: 1 file (10 points)

Task 4: 1 file (10 points)

Task 5: 1 file (10 points)

Task 6: 1 file (10 points)

➤ Submission guidelines

Use the LLM models sourced from watsonx.ai API that we have shown you in previous sections for the above tasks and then take a screenshot of the output.

You will be graded as per the following rubric (60 points).