# Stitching Project

Release date: 3/13/2025

**Due date: 3/21/2025 at 5:00 PM**

The final stitching project will combine the work you did in Assignment 1 parts 1 and 2, Assignment 3, and Assignment 4. You will create an advanced RAG workflow using LLM-based agents where at least one of those agents uses a LoRA fine-tuned model.

## 1. Project Overview and Guidelines

1. You are free to use any agentic AI workflow for your advanced RAG but it must follow these conditions:
   1. Incorporate at least 2 LLM-based agents
   2. One of the agents needs to use a LoRA fine-tuned model
   3. Use a vector db-based retriever (as was done in assignment 1)

## 2. Build your agentic workflow

1. You are free to use any Agentic AI framework to build your advanced RAG, but the lab covered LangGraph and the TA is most familiar with LangGraph
2. One of the agents needs to be a LoRA fine-tuned model. This could be for agents that output free-formatted text (such as generating the final answer), but HuggingFace models may have some support for structured outputs (<https://python.langchain.com/docs/integrations/chat/huggingface/>) if you want to use the fine-tuned model for a different part of the agent workflow such as retrieved document grading or validation
   1. You are allowed and encouraged to load in the model that you fine-tuned in Assignment 3, but you may also fine-tune a new model if you desire

## 3. Build a front-end that takes in a prompt and processes the multi-agent system and delivers an output

1. This requires a little more work than just running the inputs on your Jupyter notebook. You should create a front-end that takes in a prompt and outputs the answer from your agentic RAG.
   1. This could be in the form of a command line interface, where you run the python script and are prompted to enter a question that will be input to the multi-agent system.
   2. A more complex and rewarding approach than a command line interface would be to form a basic web application that functions as a conversational system. There are a variety of python frameworks for building web apps quickly such as Flask.

## 4. Evaluate performance of Advanced RAG system

1. Using a set of 5-10 input questions, find the outputs from the following
   1. Base LLM (no RAG)
   2. Basic RAG (just the retriever and LLM that generates the output based on the retrieved documents)
   3. Advanced agentic RAG with base model (no fine-tuning)
   4. Advanced agentic RAG with fine-tuned model
2. Qualitatively examine the outputs and provide a discussion on how the quality of responses change as you move from the base LLM to the advanced RAG with a fine-tuned agent

## 5. Unit Test and Submit Project

1. Create a separate unit test that can be run in 5-10 minutes similar to assignment 4. In this unit test, you can simply load in the fine-tuned model, you don’t need to include the training code for fine-tuning in the unit test. Provide instructions in your README on how to run this unit test.
2. Submit the final project on Github classroom. Include all the code files, a README on how to run the code and the unit test, the Dockerfile and Docker image name/server location necessary to run the code, and a document showing the outputs as well as a discussion on the outputs.