**Task 1:**

**Problem:** Learning a new programming language is hard.

**Solution:** Build a fine-tuned, agentic RAG application called TidyChat that can answer conceptual questions about utilizing the tidyverse packages, and assist in building an intuitive understanding of good coding principles. Additionally, users can explore updates to packages and additional ways to utilize the tidyverse in their workflows.

Ultimately, this application could serve as an assistant for students and educators when focusing on the R programming language. Additional documents including core textbooks “R for Data Science”, “Tidy Models”, “ggplot2” and others could add benefits and result in code generation help in addition to the current assistance.

**Task 2:**

**LLM:** Both gpt-4o-mini and gpt-4.1-mini

The use of closed-source models is appropriate for this application, given that the documents used for RAG are publicly available. The gpt-4o-mini model is used as the ‘improvement’ from the gpt-4.1-mini.

**Embedding Model**: snowflake-arctic-embed-m

The use of an open-source embedding model is appropriate given that we will be fine tuning for improved performance. This allows for future modifications when new documents and information become available, or if the scope and direction of the application change.

**Orchestration**: LangGraph

LangGraph is appropriate given that it’s widely used amongst AI engineering teams and professionals, due to it’s scalability, flexibility and documentation. As the application progresses and incorporates additional agents, this product will continue to be the best option.

**Vector Database**: Qdrant

Qdrant is appropriate for this application due to it’s low cost (free) and the speed (low latency) even with larger volumes of data. Additionally, as the application improves and users are allowed to upload their own documents, Qdrant provides the necessary capabilities to scale with the application.

**Monitoring**: LangSmith

In conjunction with LangGraph, LangSmith provides a comprehensive set of tools to monitor the activity of the application. This allows users and teams to investigate inputs, outputs and intermediate steps, and integrates with tools above and beyond LangChain.

**Evaluation**: RAGAS

RAGAS is appropriate for this application due it’s ease of implementation and it’s ability to capture improvements in LLM responses based on changes to inputs (RAG). This allows us to continually assess changes and measure improvements as the application expands.

**User Interface**:

The app utilizes Chainlit, an interactive and easily accessible user interface. Chainlit allows for rapid deployment with low overhead and no cost. This is an industry standard for application development and demos that will be expanded in the future.

**Use of agents**: Tavily

The tavily tool will be used to provide updates and announcements from tidyverse key contributors and the tidyverse website.

**Task 3:**

The following posts were compiled and utilized for Retrieval Augmented Generation:

1. Tidy Help: https://www.tidyverse.org/help/
2. Learn Tidyverse with AI: https://www.tidyverse.org/blog/2025/04/learn-tidyverse-ai/
3. Tidy Models updates Q1 2025: https://www.tidyverse.org/blog/2025/02/tidymodels-2025-q1/
4. Experiments with LLM: https://www.tidyverse.org/blog/2025/01/experiments-llm/
5. Tidyverse Packages: https://www.tidyverse.org/packages/

The documents were chunked using the baseline recursive character text splitter, a current best-practice within the industry. Additional updates for advanced retrieval, utilizing metadata with chunking, and allowing users to upload their own documents for RAG are ongoing.

Additionally, users of the app will have access to the Tavily search tool for additional information gathering.

Currently we anticipate questions similar to the following:

* Why should I use tidy principles?
* What are ways to utilize LLM’s to help me code better?
* What are the latest updates to the packages?
* What packages are available?
* Who maintains these packages?
* Who are key contributors and leading experts on these packages?

**Task 4:**

Here is a link to the current version of the TidyChat tool on Hugging Face:

[TidyChat - a Hugging Face Space by bsmith3715](https://huggingface.co/spaces/bsmith3715/CERT_CHALLENGE)

Task 5 and Task 7:

Baseline performance compared to fine tuned performance:

|  |  |  |
| --- | --- | --- |
| **Metric** | **Baseline** | **Fine-Tuned** |
| **Context Recall** | **0.5258** | **0.5889** |
| **Faithfulness** | **0.8521** | **0.7677** |
| **Factual Correctness** | **0.5667** | **0.5775** |
| **Answer Relevancy** | **0.8159** | **0.9735** |
| **Context Entity Recall** | **0.3222** | **0.4178** |
| **Noise Sensitivity** | **0.2472** | **0.2349** |

Note that the fine-tuned model performed better on 4 of the 6 evaluation metrics including Context Recall, Factual Correctness, Answer Relevancy and Context Entity Recall. This suggests that fine-tuning the embedding model, utilize the cohere ranker for retrieval, and utilizing the gpt-4o-mini for response generation improved the performance of our RAG portion.

Task 6:

The fine-tuned embedding model can be found at the following link:

[bsmith3715/legal-ft-cert-challenge\_final · Hugging Face](https://huggingface.co/bsmith3715/legal-ft-cert-challenge_final)

In conclusion, this application serves as a proof of concept that can be expanded and improved in several ways:

1. User uploads for RAG
2. Additional documentation for fine-tuning embedding models including core textbooks.
3. Additional Agents: arxiv agent or other publication research agent