Overview: Our project application is *The Consult* - an AI-powered medical literature search application tailored for clinicians and researchers. We have data collection, a ChromaDB database to support RAG, a fine-tuned LLM, a frontend UI, APIs, and application deployment to support this application. This is the code organization:

```
src/
├── datapipeline/  # Data collection directory
├──finetuning/  # Model fine-tuning directory
├── frontend/  # Frontend UI directory
├── llm-api/  # Application API directory
├── models/  # RAG module directory
└── secrets/  # GCP credentials directory
```

Data Collection: We collect data for RAG as follows:

1) Download and parse Pubmed XML files containing metadata and abstracts
2) Upload the pickled abstracts and metadata to a local directory or a GCS bucket

Technologies Used: Python, Docker, GCS

ChromaDB Database: We insert the data from the data collection step into a ChromaDB collection for RAG as follows:

1) Read data from GCS and create semantic chunks and embeddings from the abstracts
2) Upload chunks, embeddings, and associated metadata to a ChromaDB collection

Technologies Used: Python, Docker, GCS, VertexAI for embeddings, DVC for data versioning

Fine-tuned LLM: We fine-tuned a Gemini 2.5 model to customize responses towards clinicians or researchers. This involves:

1) Generating a question-answer dataset and converting it to the appropriate format
2) Fine-tuning Gemini 2.5 using the generated dataset

Technologies Used: Python, Docker, VertexAI for Gemini and finetuning

Frontend UI: Our frontend displays a search bar to ask medical questions, metadata filtering options, and an LLM response with references. The user can toggle between Clinical Practice and Research mode.
Technologies Used: React, Typescript, Vite, CSS

APIs: We use a FastAPI web service to connect the backend (ChromaDB database and Google Gemini) to our frontend UI. This is how the APIs work:

1) Accept questions and metadata from users in the frontend and send results to API server
2) Server sends user's request to the RAG module and outputs an LLM response
3) Return LLM response and citations back to the frontend

Technologies Used: Python (FastAPI), Docker, VertexAI for Gemini

Deployment: We deploy the application that connects ChromaDB, Google Gemini, the APIs, and the frontend together. The application can be deployed locally or on a virtual machine.
Technologies Used: Uvicorn, Terraform, Google Cloud Run, Google Virtual Machine