For making improvements to the original prototype and scaling the model according to the user needs, the following steps can be implemented:

1.) Using a bigger language model along with the VectorDB: the original implementation had RAG and the language model separate but a better way is to send the output of the RAG as a context to the language model which would augment the answer according to the user query.
By bigger model in terms of param size, a 3B or 1B parameter model is more than enough for this particular use case. (Llama 3B model, Falcon 7B model)
Link-Enhancing LLMs with Vector Database with real-world examples | Qwak

2.) In case instruction tuning or DPO (Direct Preference Optimization) is being implemented with the bigger model, involving all the parameters in the training process would be computationally inefficient making the requirements of effective parameter tuning
Apply PEFT (Parameter efficient Fine-tuning) by freezing most parameters.
Link- Parameter-Efficient Fine-Tuning using 🤗 PEFT (huggingface.co)
Link-Introduction to Low Rank Approximation (LoRA) | Medium

3.) For implementing better RAG systems in terms of chunking or storage, the following links can be referred to for the same
Link-rag-from-scratch/rag_from_scratch_1_to_4.ipynb at main · langchain-ai/rag-from-scratch (github.com)

Microsoft has recently open-sourced Graph RAG for building RAG-based applications. You can also check it out for scaling up your applications depending on user capacity.