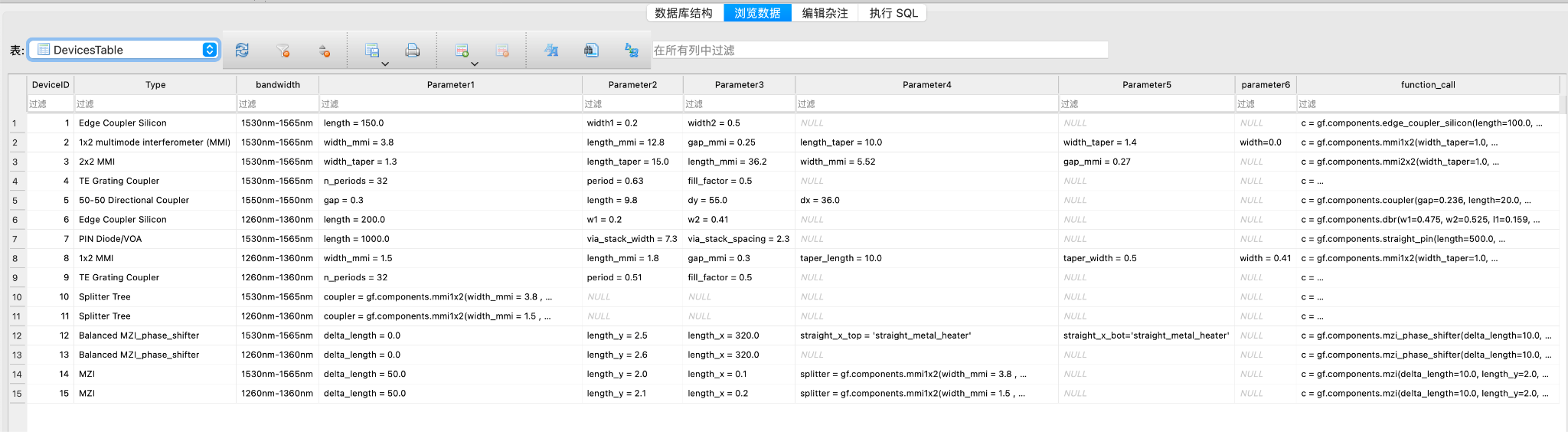
1. **weekly-meeting-presentation**
2. **semantic-search-sqlite**: semantic search with a sqlite database, I also uploaded a sample database, a txt file on how to run semantic search on localhost, and a review document as well

* The txt file includes the terminal commands(for Mac users) to run a semantic search on localhost, an openai api key is required to calculate embeddings and run queries. Current embeddings were calculated based on “Type” and “bandwidth” from the SQLite database for people to find devices based on names and operating wavelength



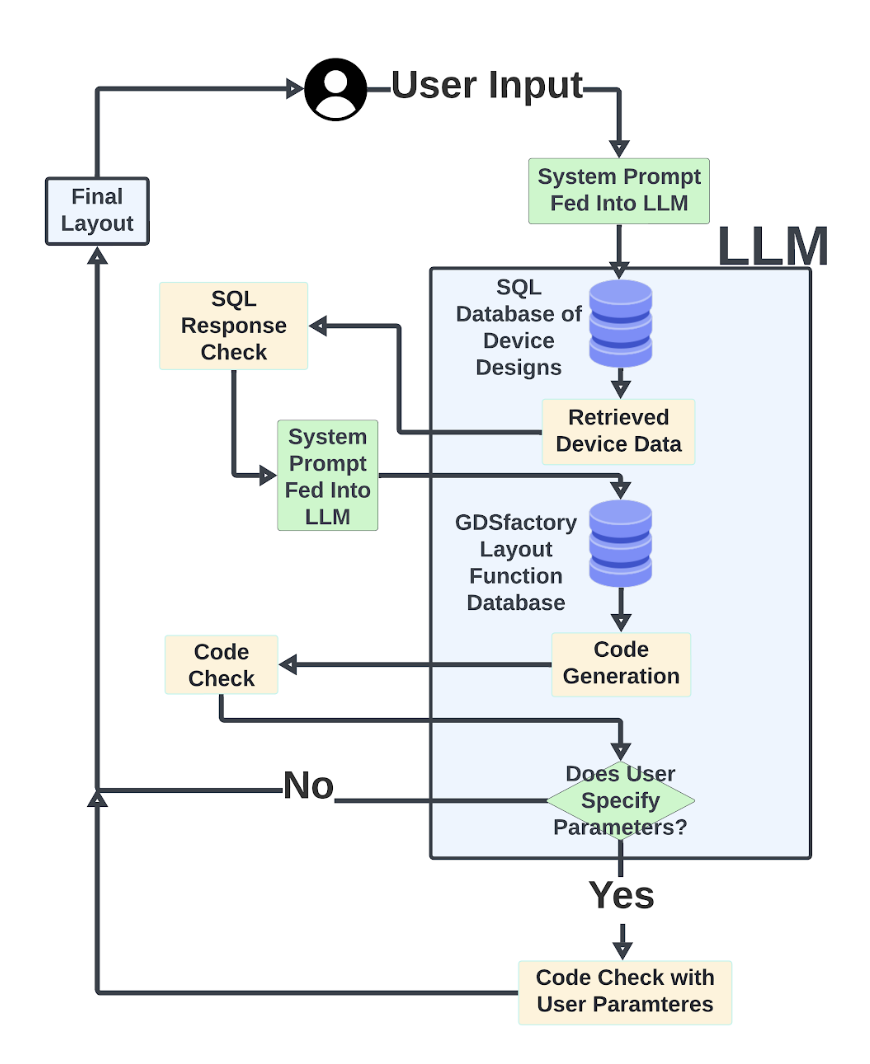
* The metadata.json in the folder determines the method of similarity search being implemented, I used “faiss\_search\_with\_scores” but there are other methods such as cosine similarity etc. The metadata file also includes how many entries are returned and points to the embeddings of the database. Details can be found in the tutorial link here: <https://medium.com/@joseph.r.martinez/implement-semantic-search-with-datasette-b2fc20a6f1d9>
* This folder also includes a review document

1. **semantic-search-assisted-llm**:

* The approach is that given some context data, we first perform a semantic search (using a sentence transformer model) to find the most relevant context and then pass it to the llm to generate a more valid response
* limitations: this approach is best suited to be a Q&A agent in a certain area rather than what we wanted the llm to do. Also, I haven’t tested with photonics knowledge specifically and the size of context data are relatively small right now, since we decided to switch to another approach (Fibers: a more generic solution that uses beam search + LLM), I didn’t spend a lot of time investigating this approach

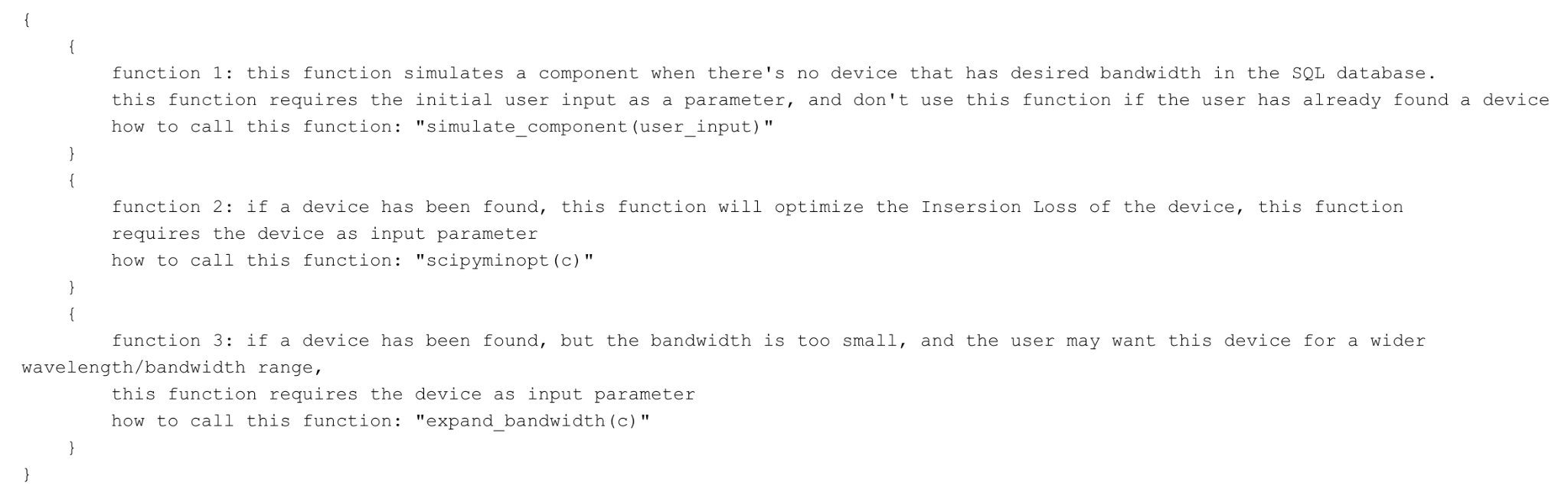
1. **old-llm-single-device-layout**: this is the approach described in the conference paper, the folder contains the code, the workflow diagram, a document explaining the workflow, and a sample database, you can directly run the code if you place the code in the same folder as the database and /gdsfactory together

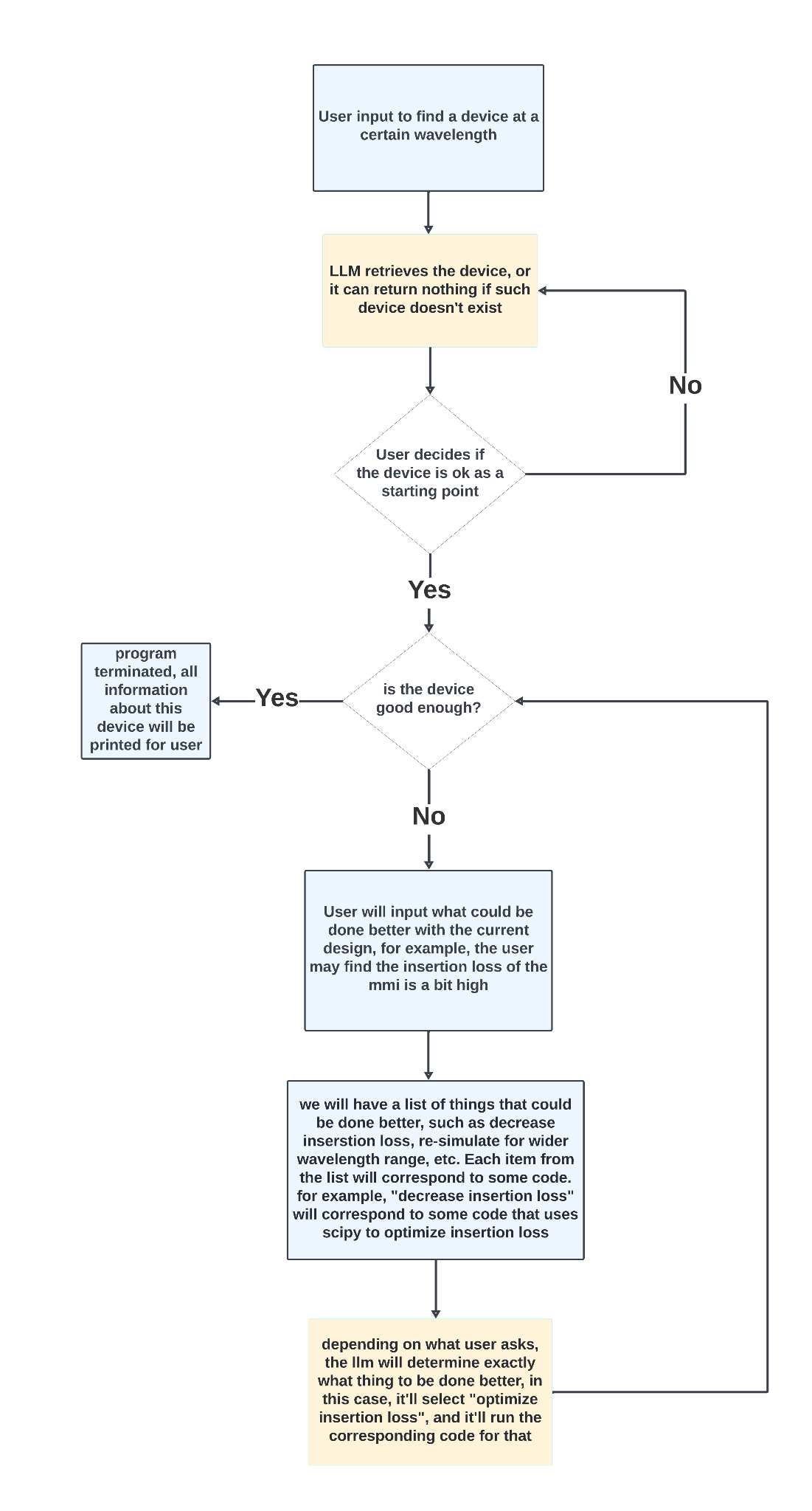
* single-device-layout.docx is the document explaining the overall approach and how to run the code



1. **llm-single-device-simulation**: I modified the above approach after submitting the conference paper because we wanted to incorporate the Python component classes into the workflow instead of only drawing the gdsfactory layout. there was a workflow diagram as well, but I didn’t investigate too much with this approach because we eventually found out Fibers was a better approach since it was more generic

* Llama\_index\_sql\_vector\_database.ipynb is where the code is stored, a replicate API is required to run the llm
* Inside the folder /gdsfactory is a txt file that includes function names such as device simulation, optimization, and re-simulation for an existing device, the purpose of this file is that the user may have issues with the current design of the device and they can ask the llm to find the best-suited function to call



* device-simulation.db is the sqlite database of previous designs

1. **LLM-review**: this folder only has the Starcoder2 review so far
2. **FIbers**: the zip file is the current version of the GitHub repo to run Fibers, I also uploaded documents of Fibers explained (ZIjian wrote this document, he is a uoft PhD student who created Fibers), a test result, and how to use Fibers. Fibers use litellm which supports a variety of platforms such as openai, replicate, mistral, etc. and each platform may require a different API key to run their model

* This is the GitHub link: <https://github.com/EvoEvolver/InstRunner/tree/main> I’m the admin of this repo so if you want to see the code on GitHub, just message me and I’ll add you to the repo

1. **AI photonic designer-Jason Liu**: this includes all the rough notes I’ve taken for this entire term