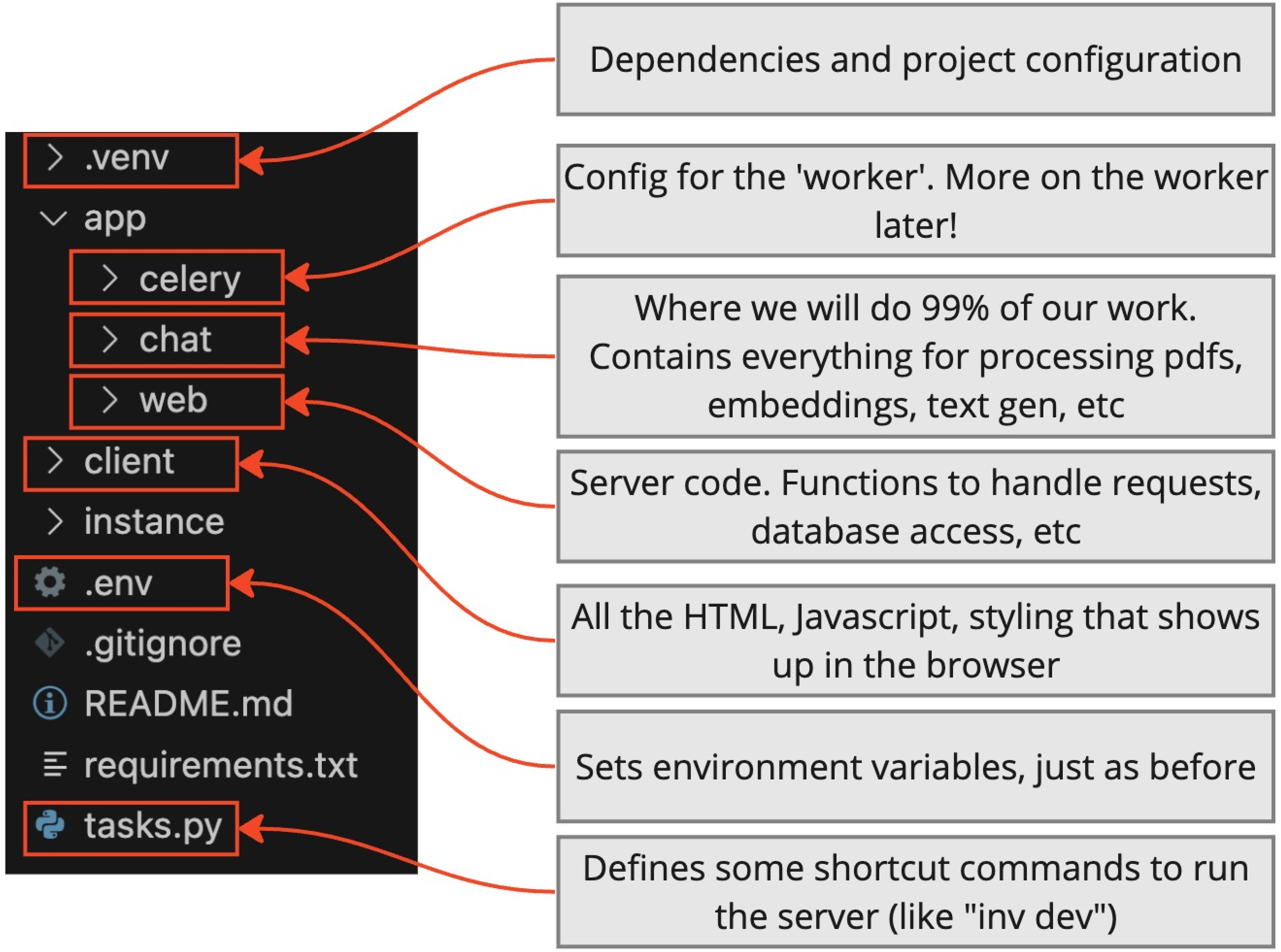
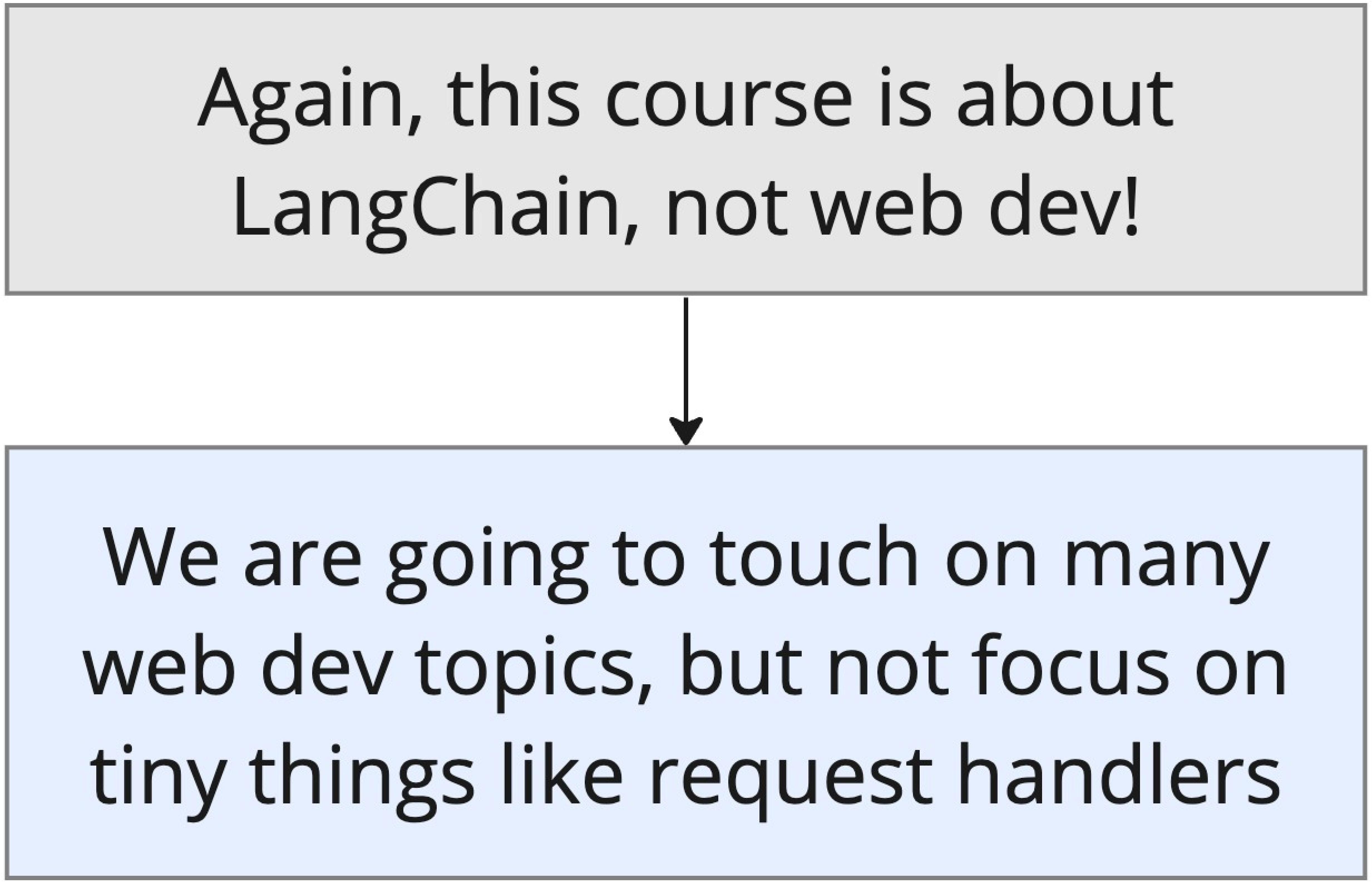

.venv
app
celery
chat
web
client
instance
.env
.gitignore
README.md
requirements.txt
tasks.py
Dependencies and project configuration
Config for the 'worker'. More on the worker later!
Where we will do 99% of our work. Contains everything for processing pdfs, embeddings, text gen, etc
Server code. Functions to handle requests, database access, etc
All the HTML, Javascript, styling that shows up in the browser
Sets environment variables, just as before
Defines some shortcut commands to run the server (like "inv dev")



Again, this course is about LangChain, not web dev!

We are going to touch on many web dev topics, but not focus on tiny things like request handlers

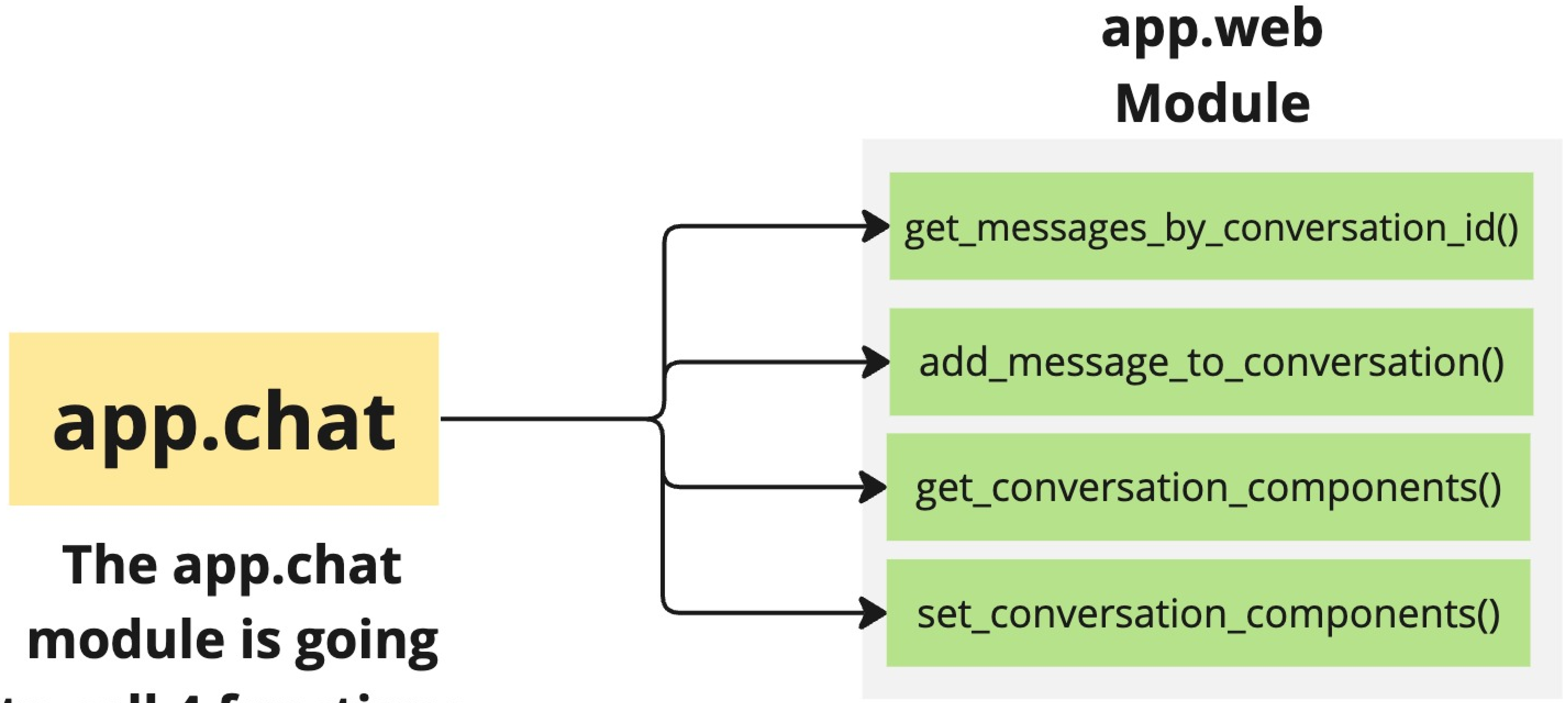
app.chat

We need to build this module

app.web

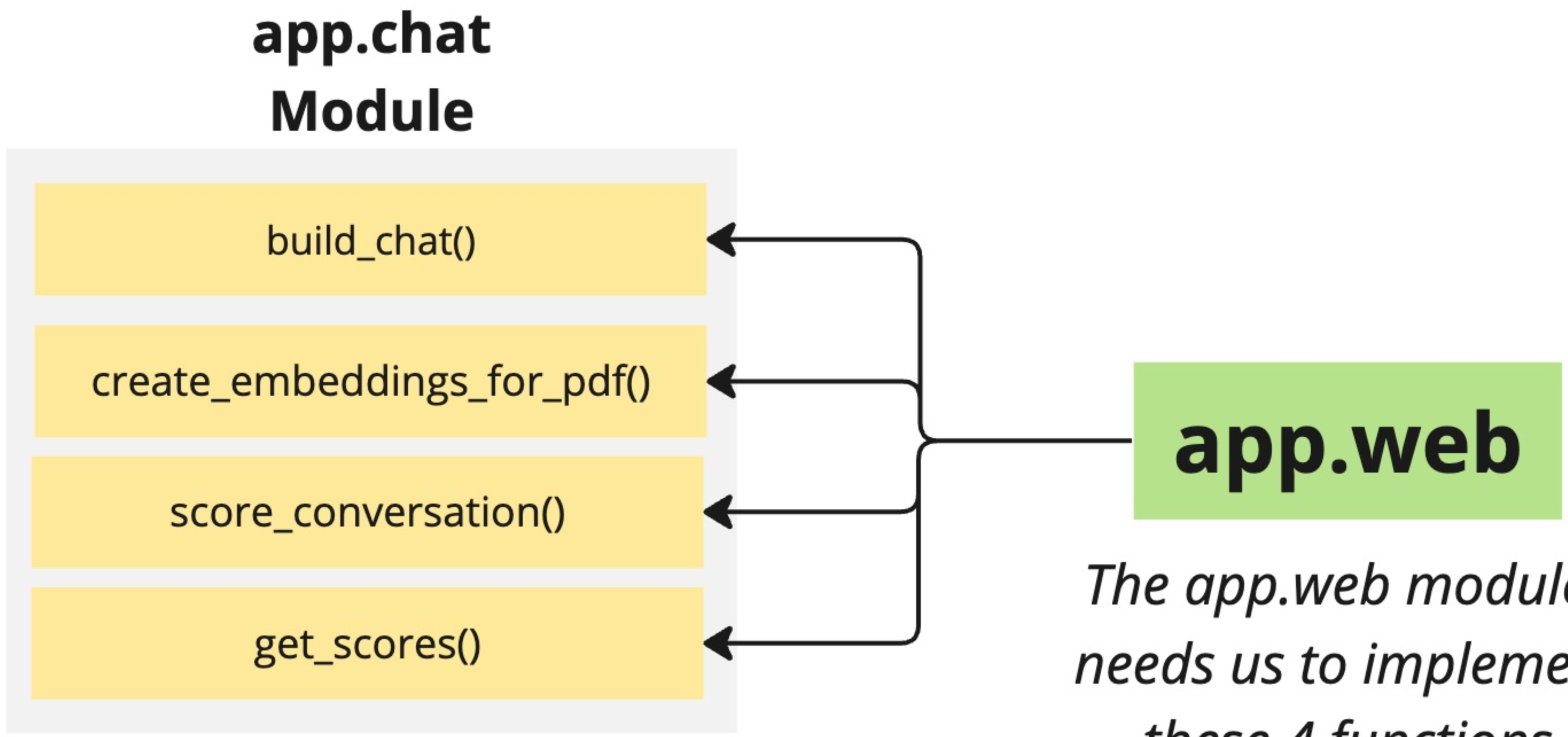
This module is mostly complete. Contains mostly web-dev stuff

I don't want you to have to worry about this module too much!


app.web
Module
app.chat
get_messages_by_conversation_id()
add_message_to_conversation()
get_conversation_components()
set_conversation_components()

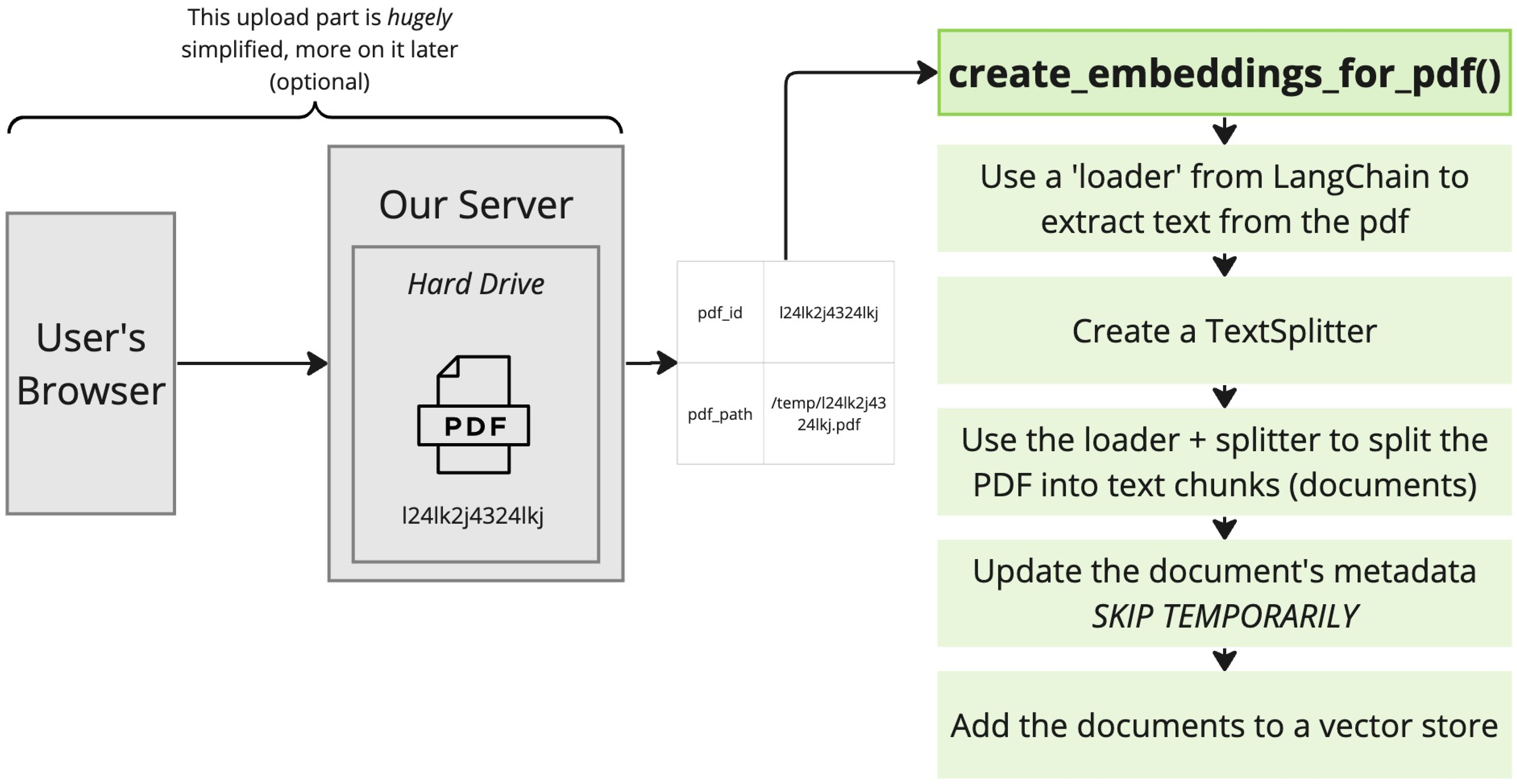
The app.chat module is going to call 4 functions provided by app.web

All of these are already complete! Located in app/web/api.py



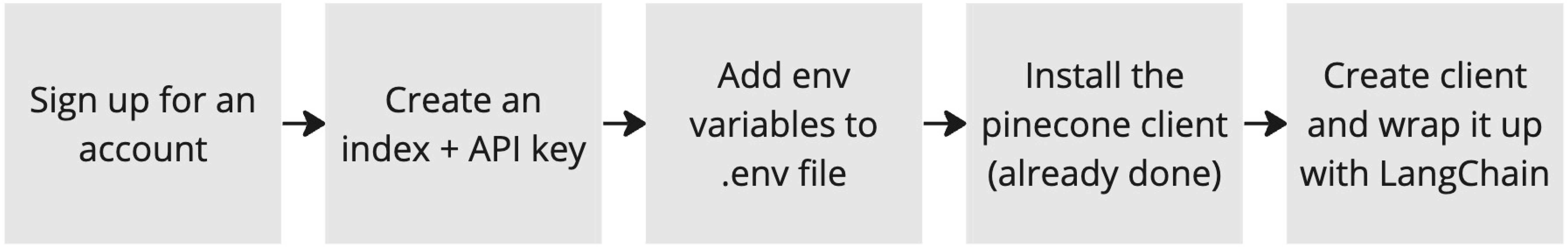
We need to implement these! (plus tons of supporting code)

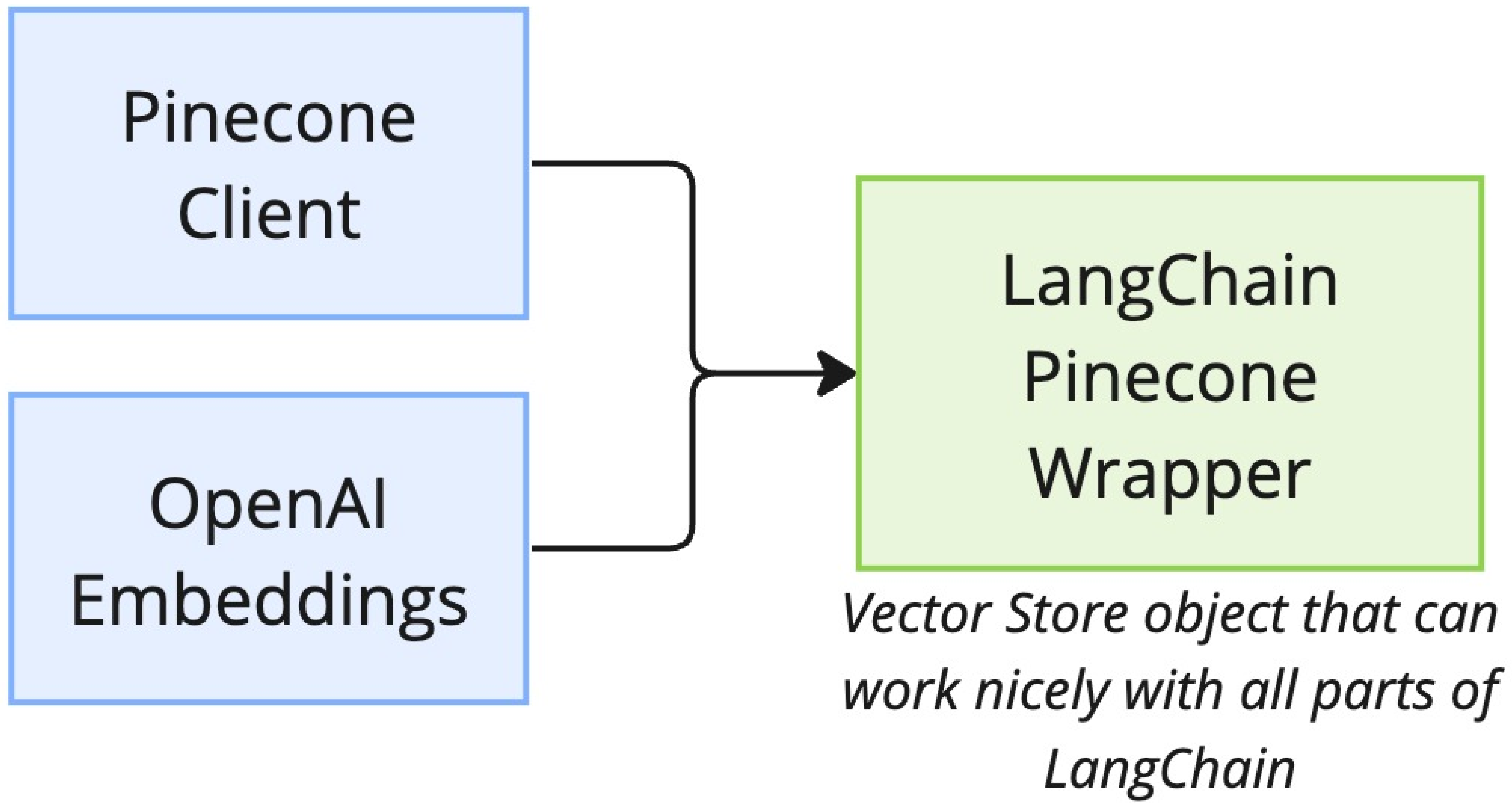
The app.web modules needs us to implement these 4 functions


This upload part is *hugely* simplified, more on it later (optional)
User's Browser
Our Server
Hard Drive
PDF
l24lk2j4324lkj
pdf_id
l24lk2j4324lkj
pdf_path
/temp/l24lk2j4324lkj.pdf
create_embeddings_for_pdf()
Use a 'loader' from LangChain to extract text from the pdf
Create a TextSplitter
Use the loader + splitter to split the PDF into text chunks (documents)
Update the document's metadata *SKIP TEMPORARILY*
Add the documents to a vector store

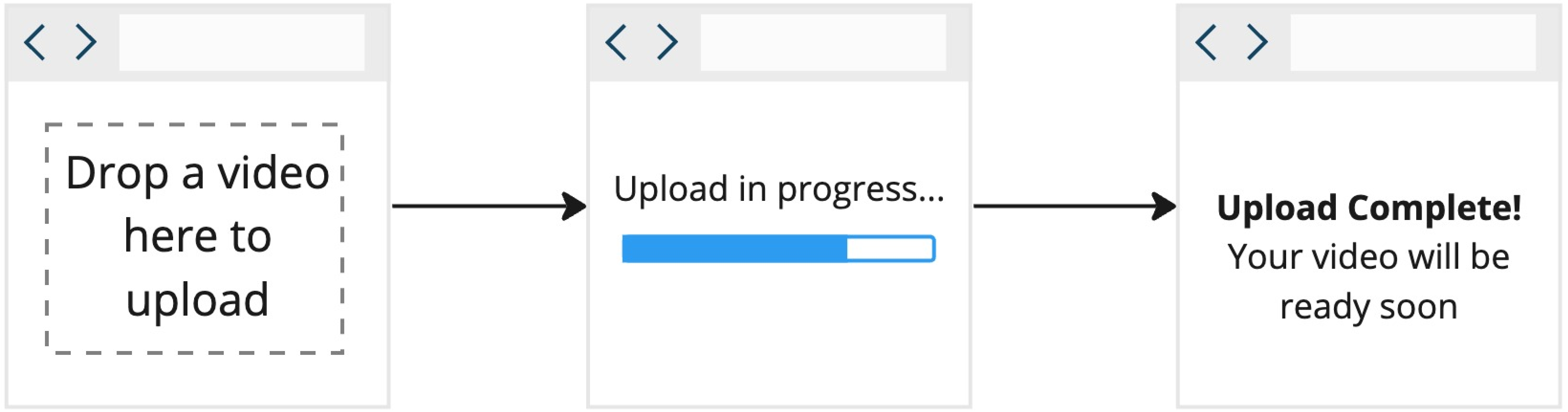
pinecone.io

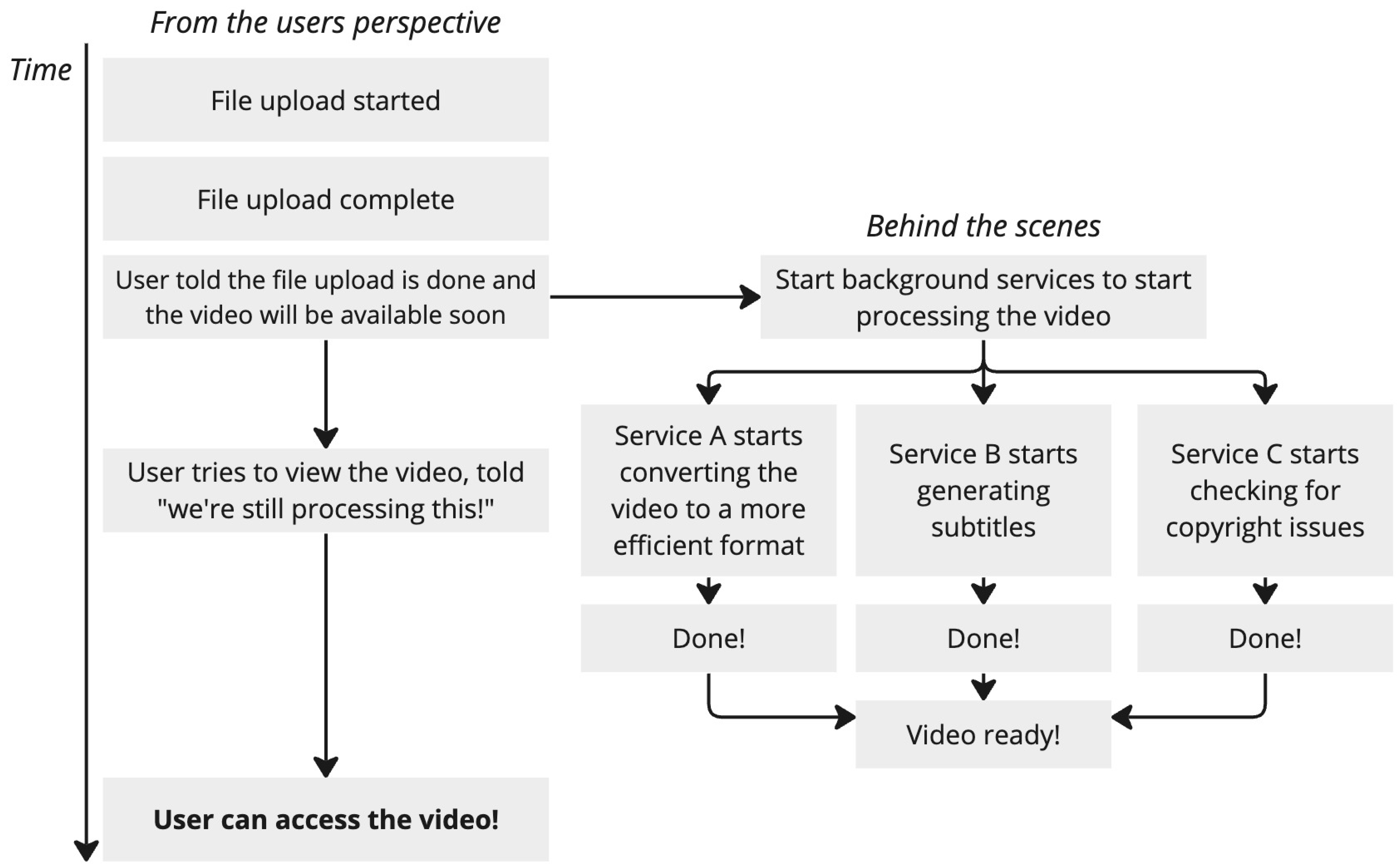
Production-ready hosted vector database


Sign up for an account
Create an index + API key
Add env variables to .env file
Install the pinecone client (already done)
Create client and wrap it up with LangChain

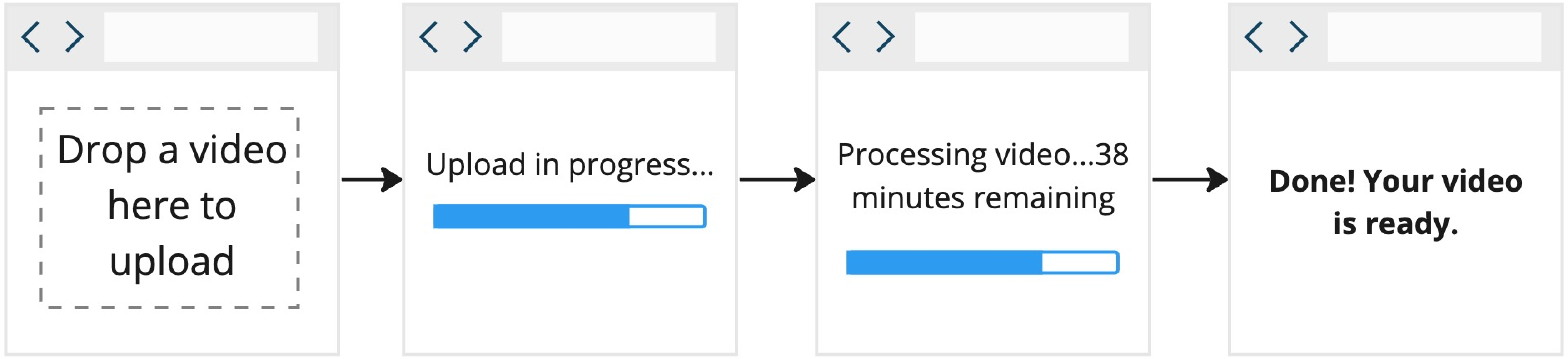

Pinecone Client
OpenAI Embeddings
LangChain Pinecone Wrapper
Vector Store object that can work nicely with all parts of LangChain

Video Upload to Youtube (or similar service)




From the users perspective
Time
File upload started
File upload complete
User told the file upload is done and the video will be available soon
User tries to view the video, told "we're still processing this!"
User can access the video!
Behind the scenes
Start background services to start processing the video
Service A starts converting the video to a more efficient format
Service B starts generating subtitles
Service C starts checking for copyright issues
Done!
Done!
Done!
Video ready!

This would be really bad


Drop a video here to upload
Upload in progress...
Processing video...38 minutes remaining
Done! Your video is ready.

This idea of 'background processing' is used **all the time in every major web app for any task that might take some amount of time**

Sending emails	Processing uploaded files
Report generation	Content moderation
File conversion	Bulk operations
Search indexing	Complex calculations
Recommendation generation	Data import/export

Any user-facing app that uses text gen **will almost definitely use this same pattern**

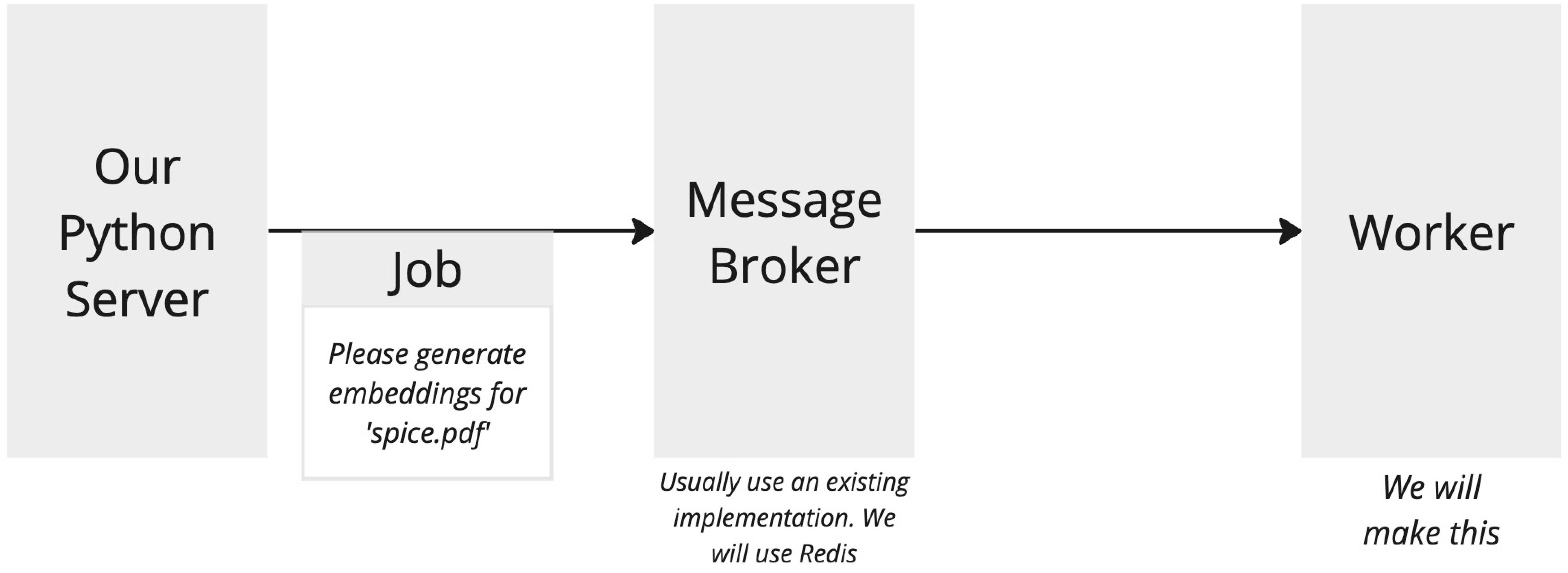
Generating embeddings

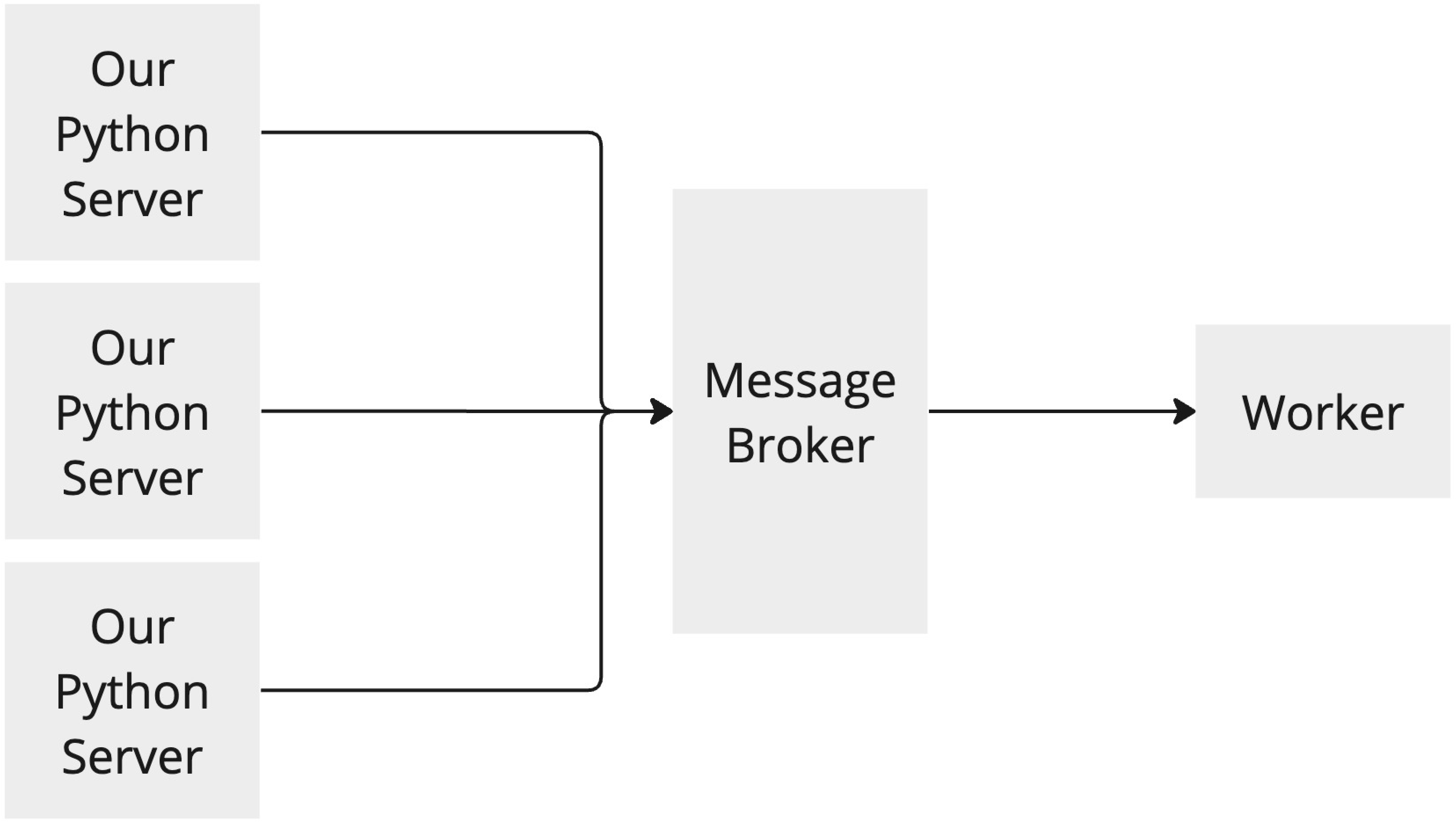
Running an agent

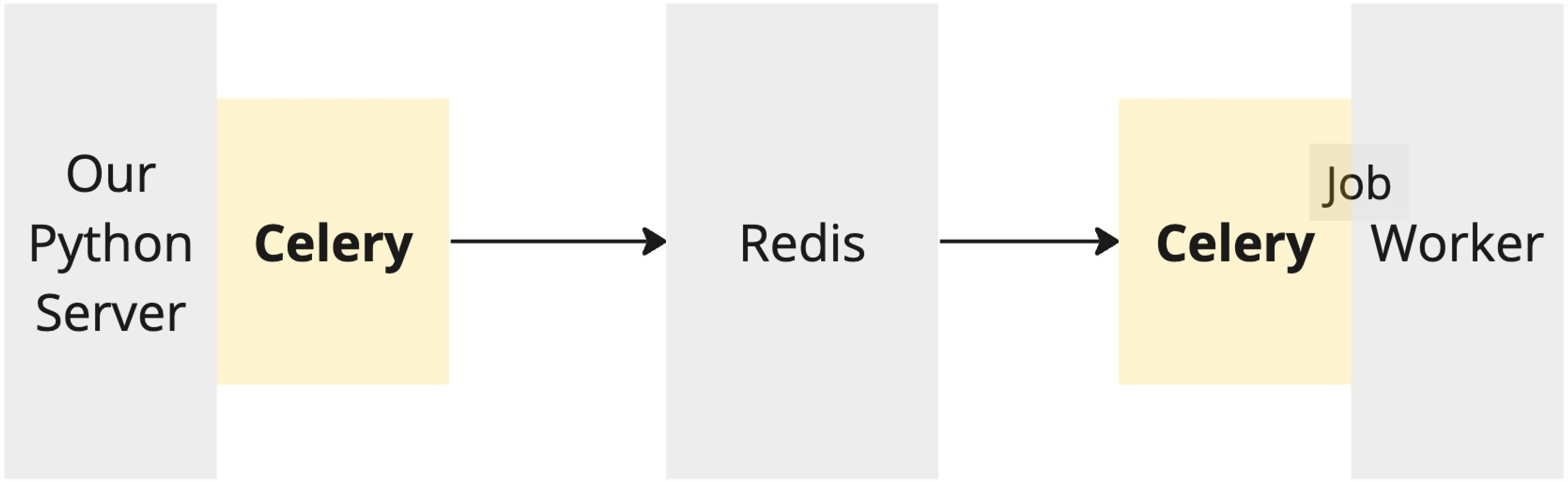
Running tools

Generating a text in bulk

The one exception is when you are generating text to *immediately* show to a user


Our Python Server
Job
Please generate embeddings for 'spice.pdf'
Message Broker
Usually use an existing implementation. We will use Redis
Worker
We will make this


Our Python Server
Our Python Server
Our Python Server
Message Broker
Worker



Celery is going to manage everything about our jobs.

Celery is Python specific

Other languages have similar libraries