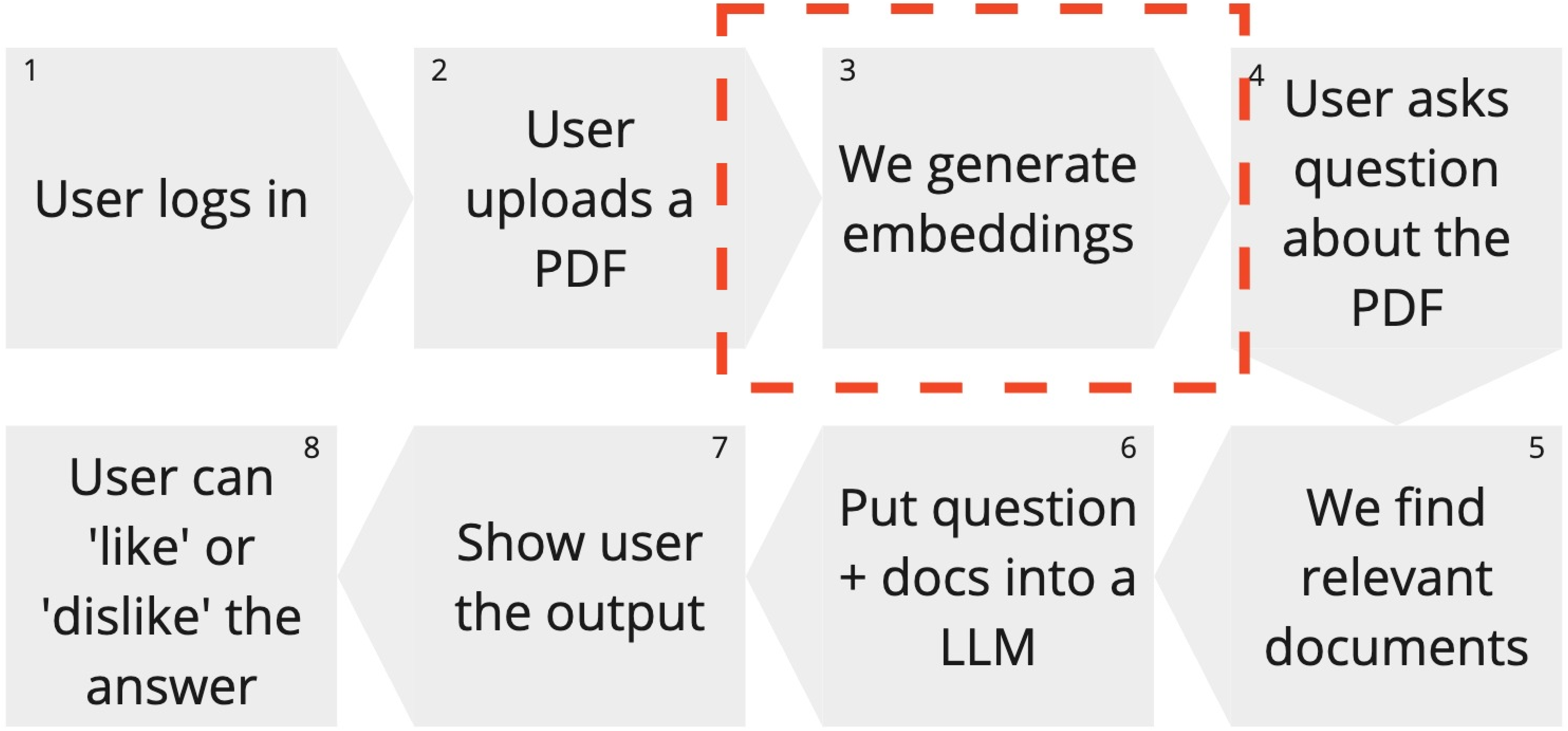
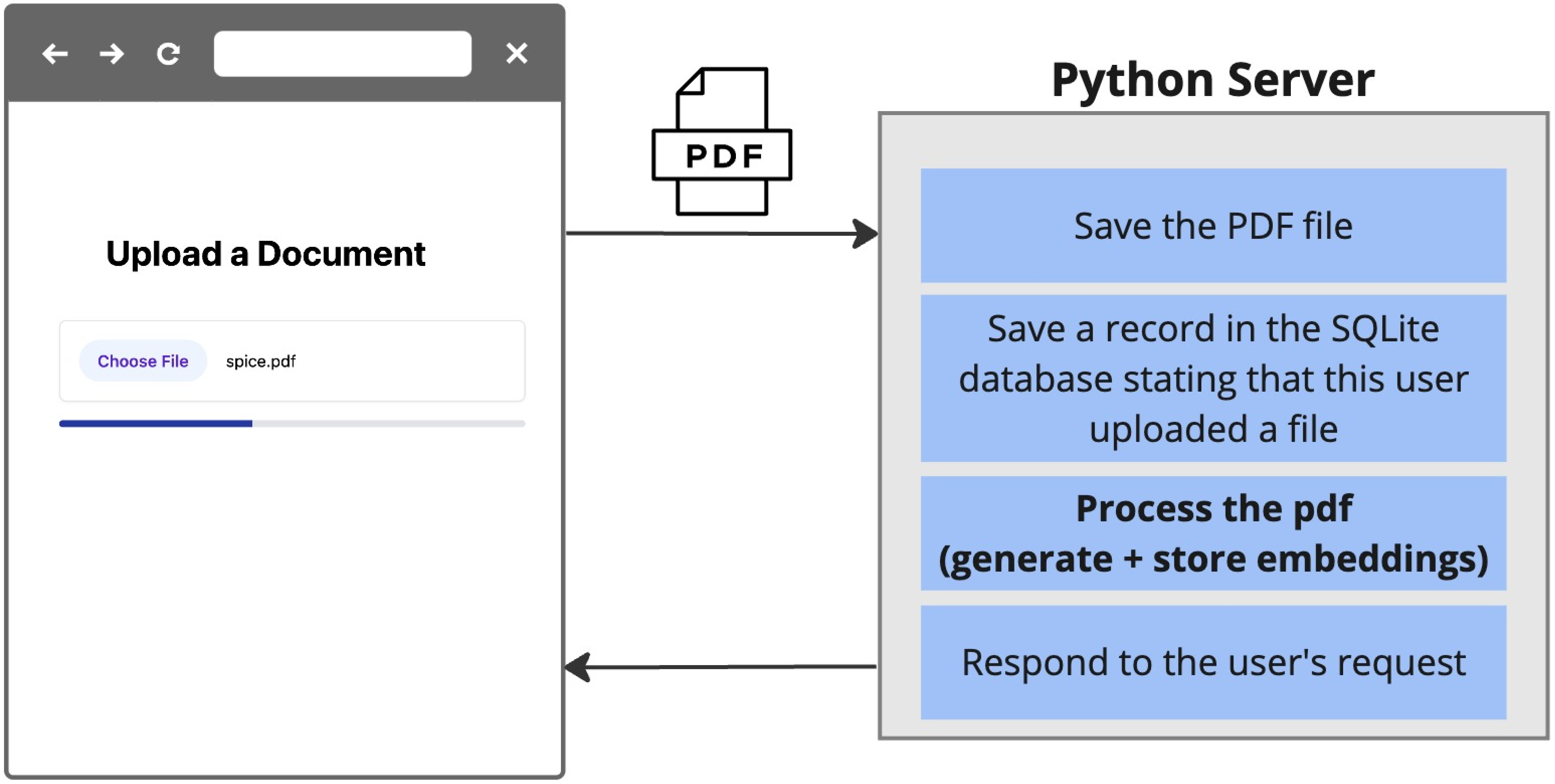
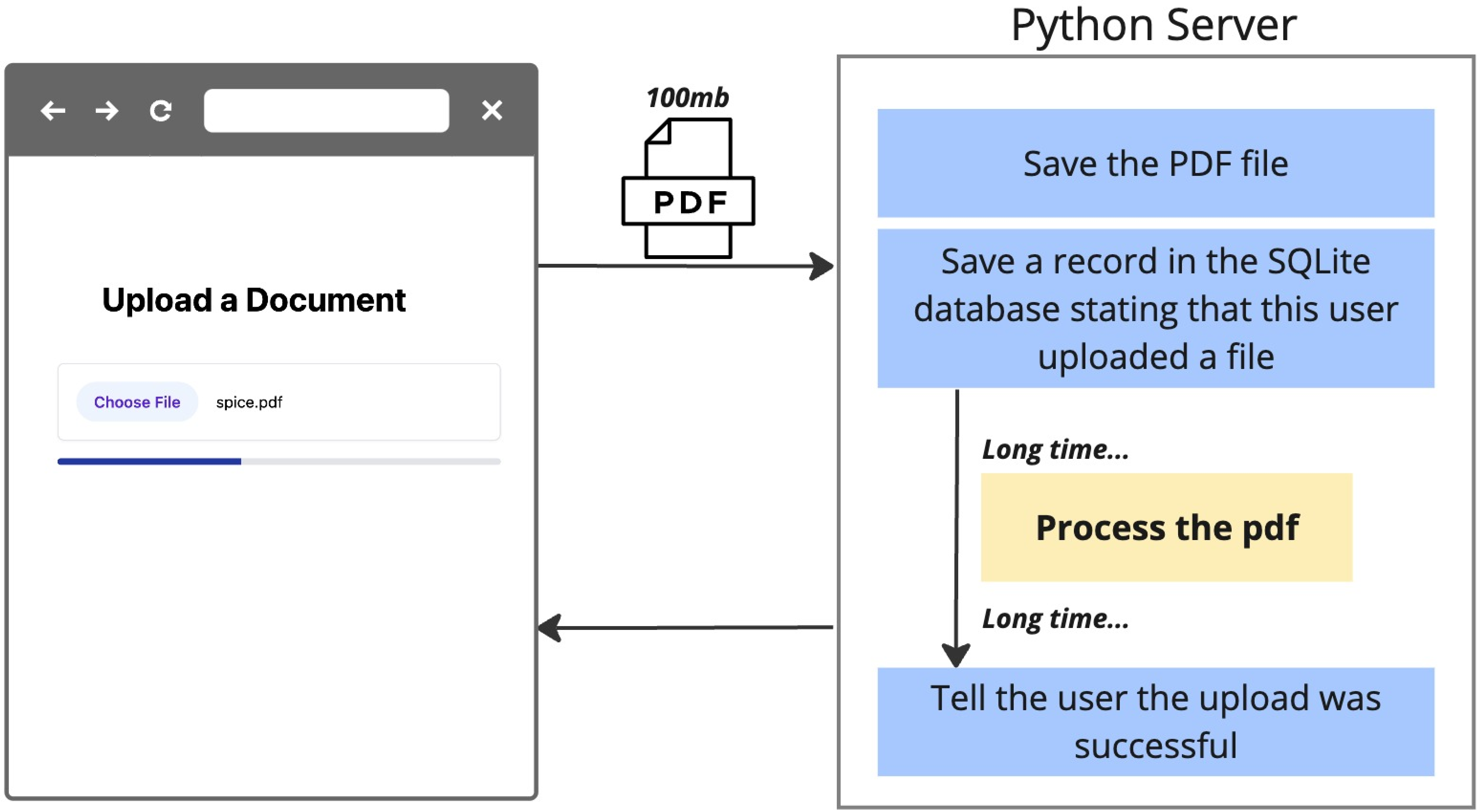

1
User logs in
2
User uploads a PDF
3
We generate embeddings
4
User asks question about the PDF
5
We find relevant documents
6
Put question + docs into a LLM
7
Show user the output
8
User can 'like' or 'dislike' the answer

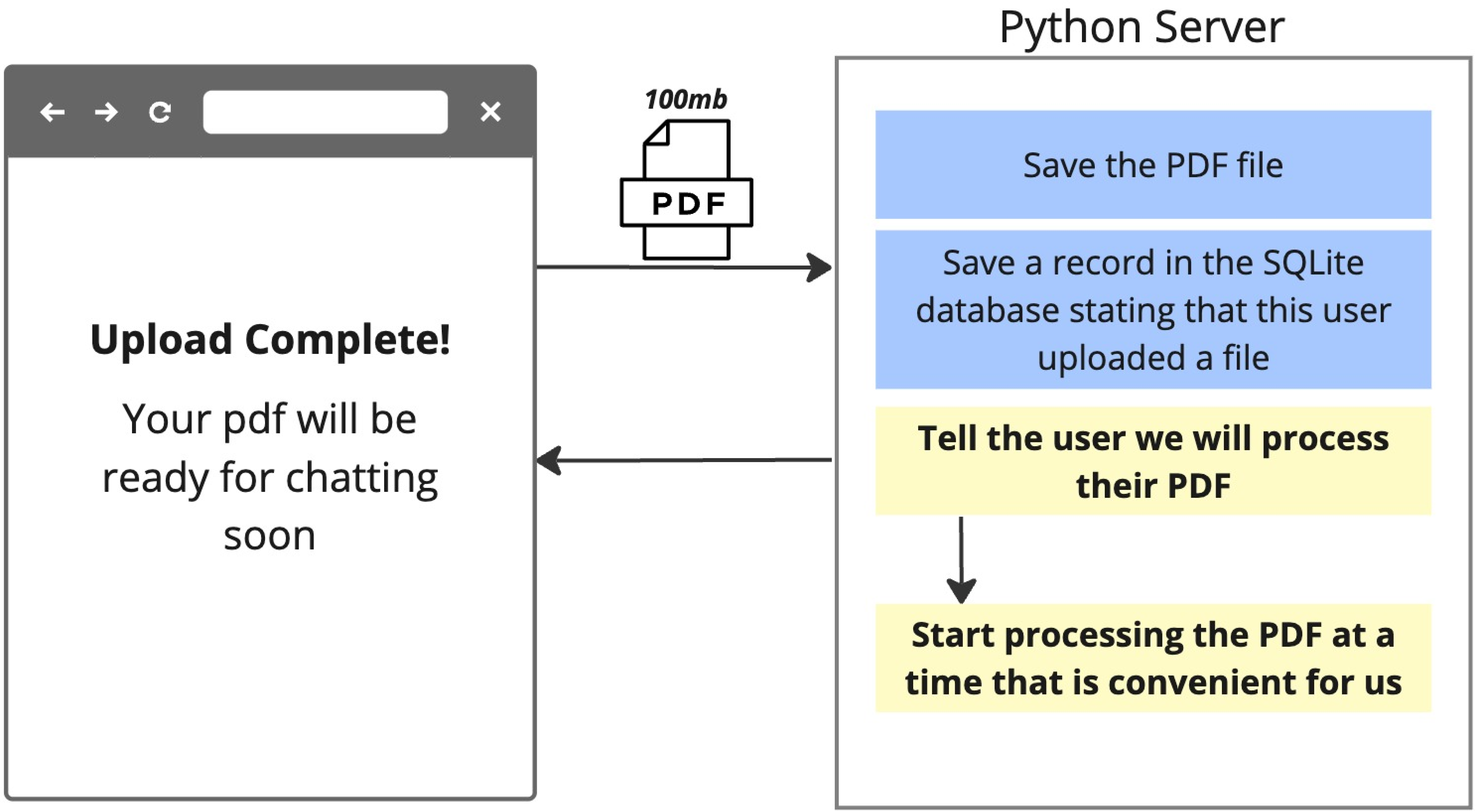

Upload a Document
Choose File
spice.pdf
PDF
Python Server
Save the PDF file
Save a record in the SQLite database stating that this user uploaded a file
Process the pdf (generate + store embeddings)
Respond to the user's request

Waiting to process the PDF before responding is bad UX



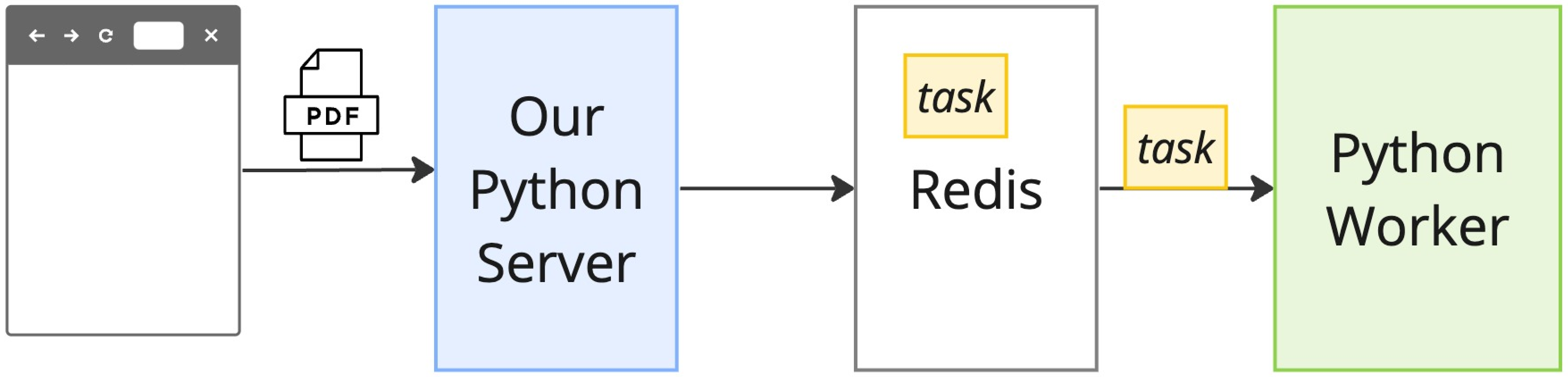
Python Server
Upload a Document
Choose File
spice.pdf
100mb
PDF
Save the PDF file
Save a record in the SQLite database stating that this user uploaded a file
Long time...
Process the pdf
Long time...
Tell the user the upload was successful

Extremely common pattern: defer the long-duration work to be done later


Python Server
100mb
PDF
Upload Complete!
Your pdf will be ready for chatting soon
Save the PDF file
Save a record in the SQLite database stating that this user uploaded a file
Tell the user we will process their PDF
Start processing the PDF at a time that is convenient for us

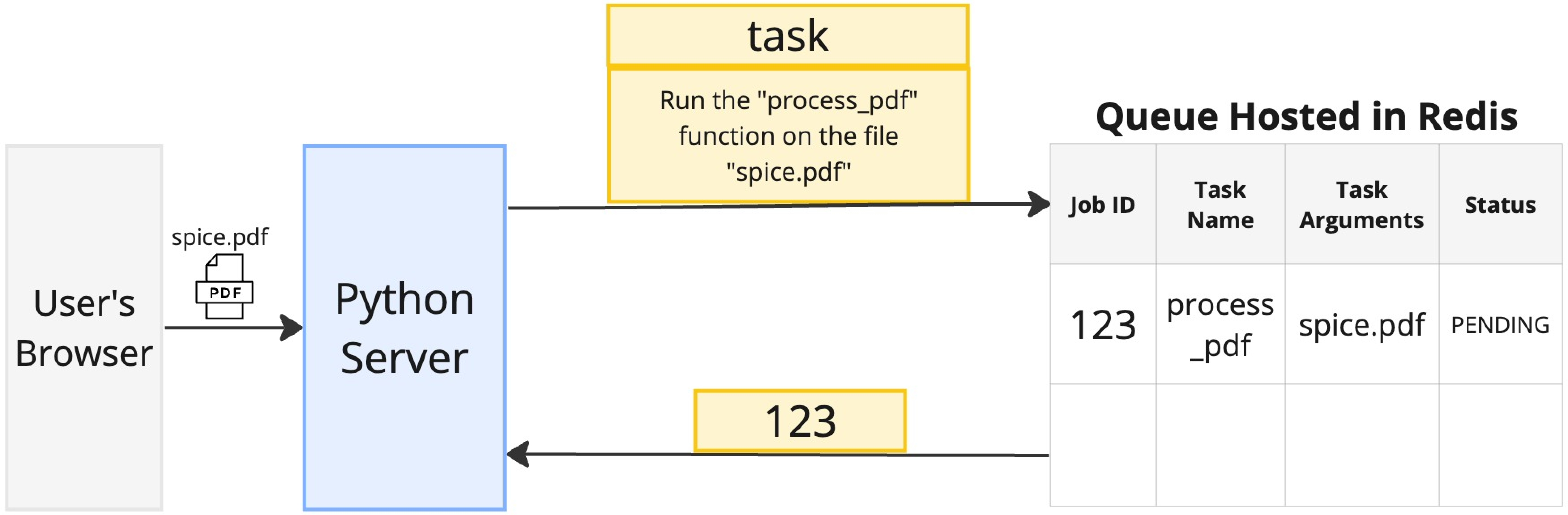
"Worker Queue" or "Background Job" Pattern

Extremely common in web dev.
***Almost** guaranteed you will use this with text gen.*

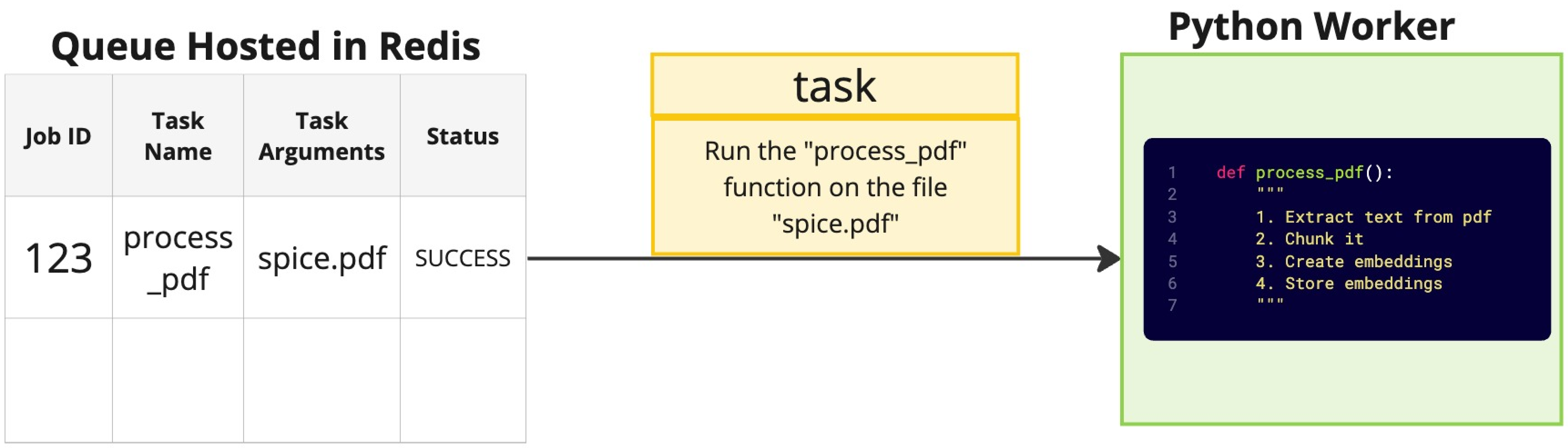


Redis is a free, open source, in-memory database. Extremely commonly used

Program that processes long-running "jobs"

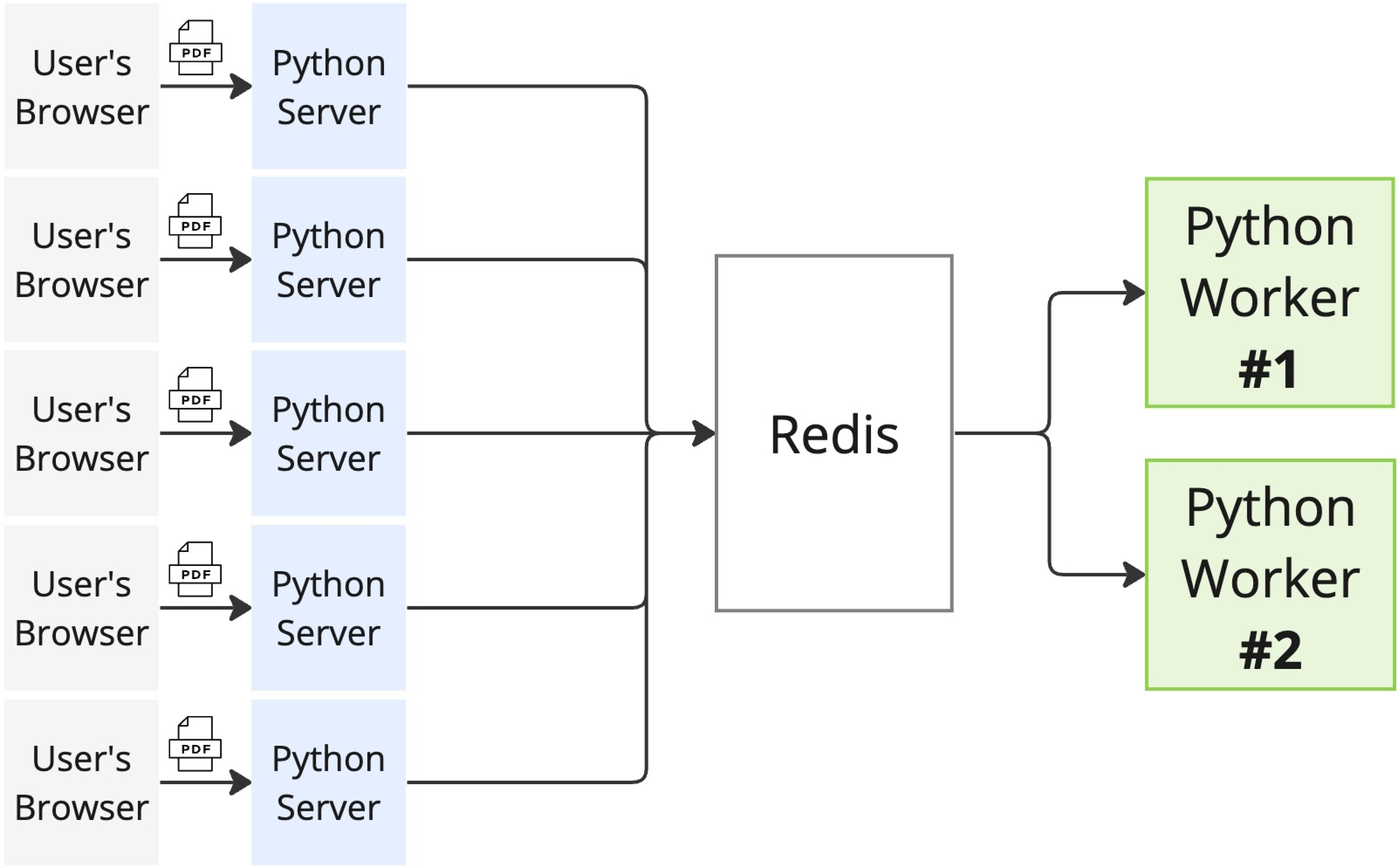


Job ID	Task Name	Task Arguments	Status
123	process_pdf	spice.pdf	PENDING

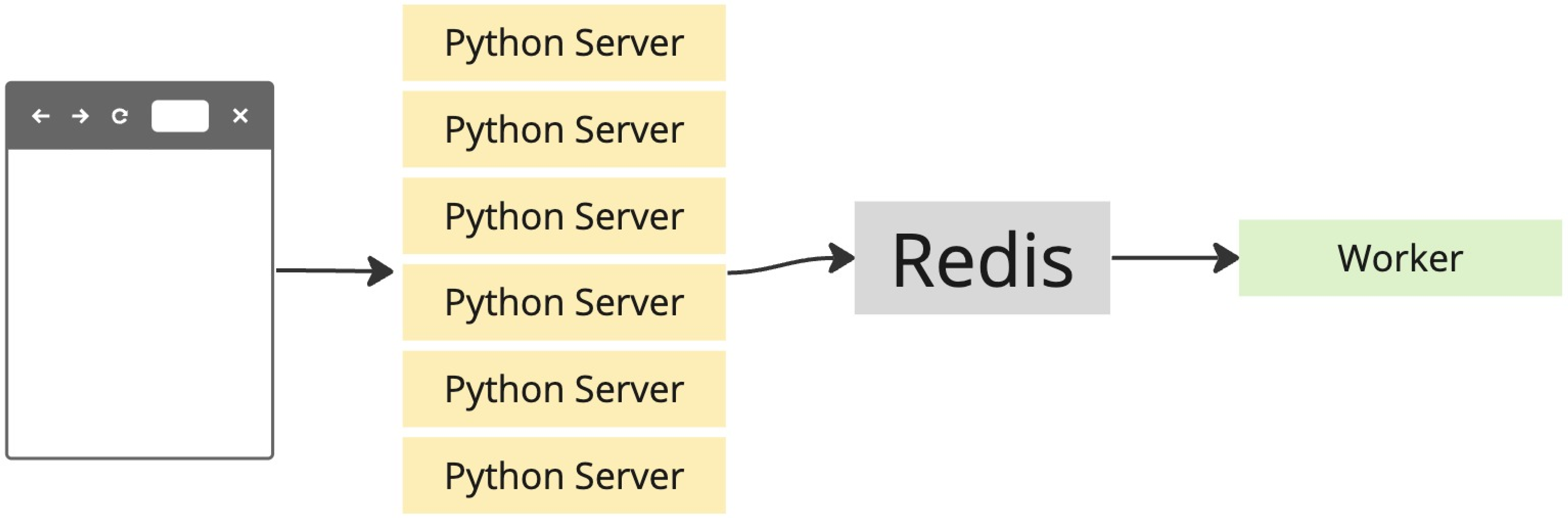


Job ID	Task Name	Task Arguments	Status
123	process _pdf	spice.pdf	SUCCESS

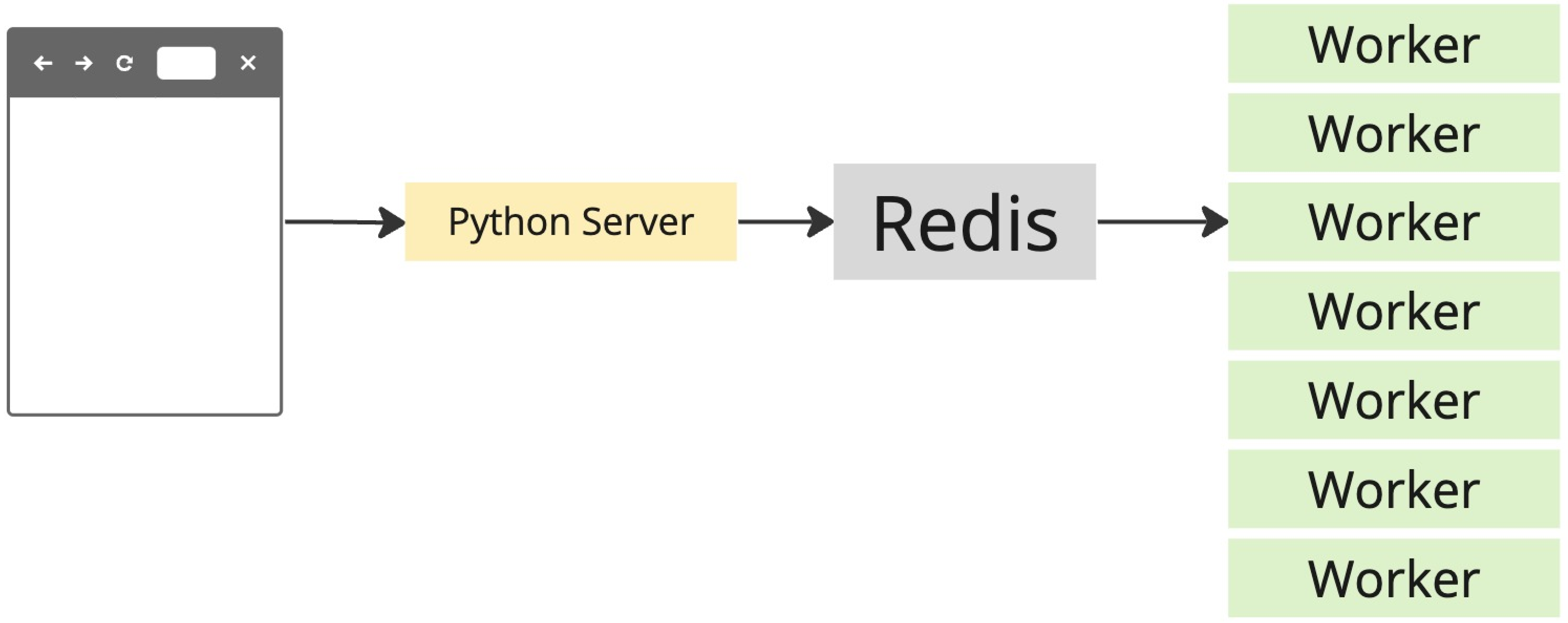
```python
def process_pdf():
    """
    1. Extract text from pdf
    2. Chunk it
    3. Create embeddings
    4. Store embeddings
    """
```


User's Browser
PDF
Python Server
User's Browser
PDF
Python Server
User's Browser
PDF
Python Server
User's Browser
PDF
Python Server
User's Browser
PDF
Python Server
Redis
Python Worker #1
Python Worker #2

Expecting a lot of traffic but few uploads?

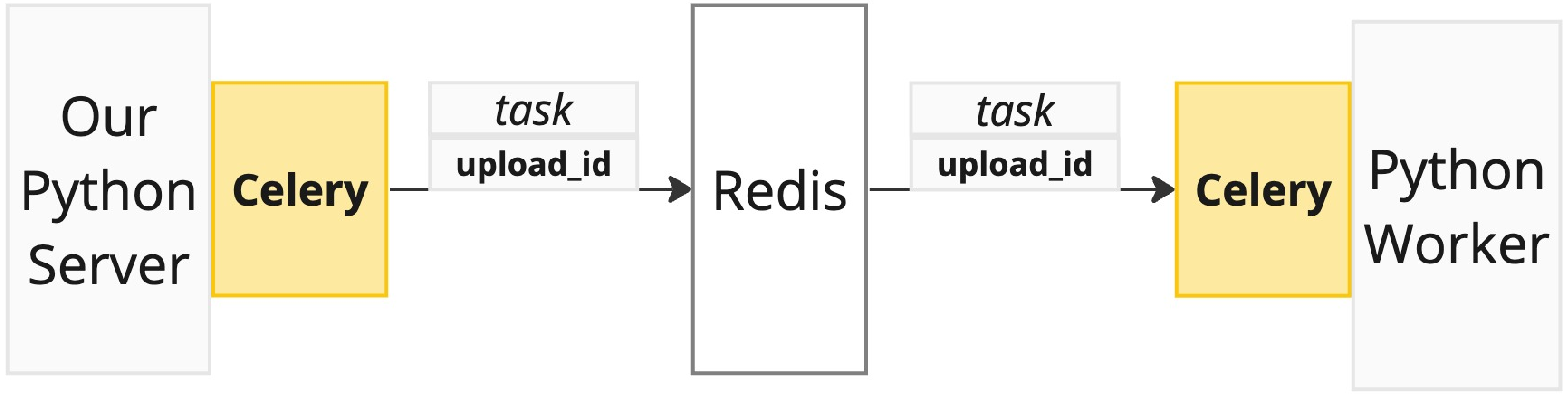


Expecting little traffic but a ton of uploads?



This worker pattern is *extremely* common any time you have a long-running task

- Generating embeddings
- Generating text that won't immediately be shown to a user
- Running an agent with many steps
- Any work that might take awhile to complete


Our Python Server
Celery
task
upload_id
Redis
task
upload_id
Celery
Python Worker

Background Jobs Setup

1	Install celery (already done)
2	Set up a Redis server instance
3	Add config to app to tell Celery how to connect to Redis
5	Start the worker process. "inv devworker"
4	Define a 'job'. Write a function and apply a decorator to it
6	Call the job function to queue it up!

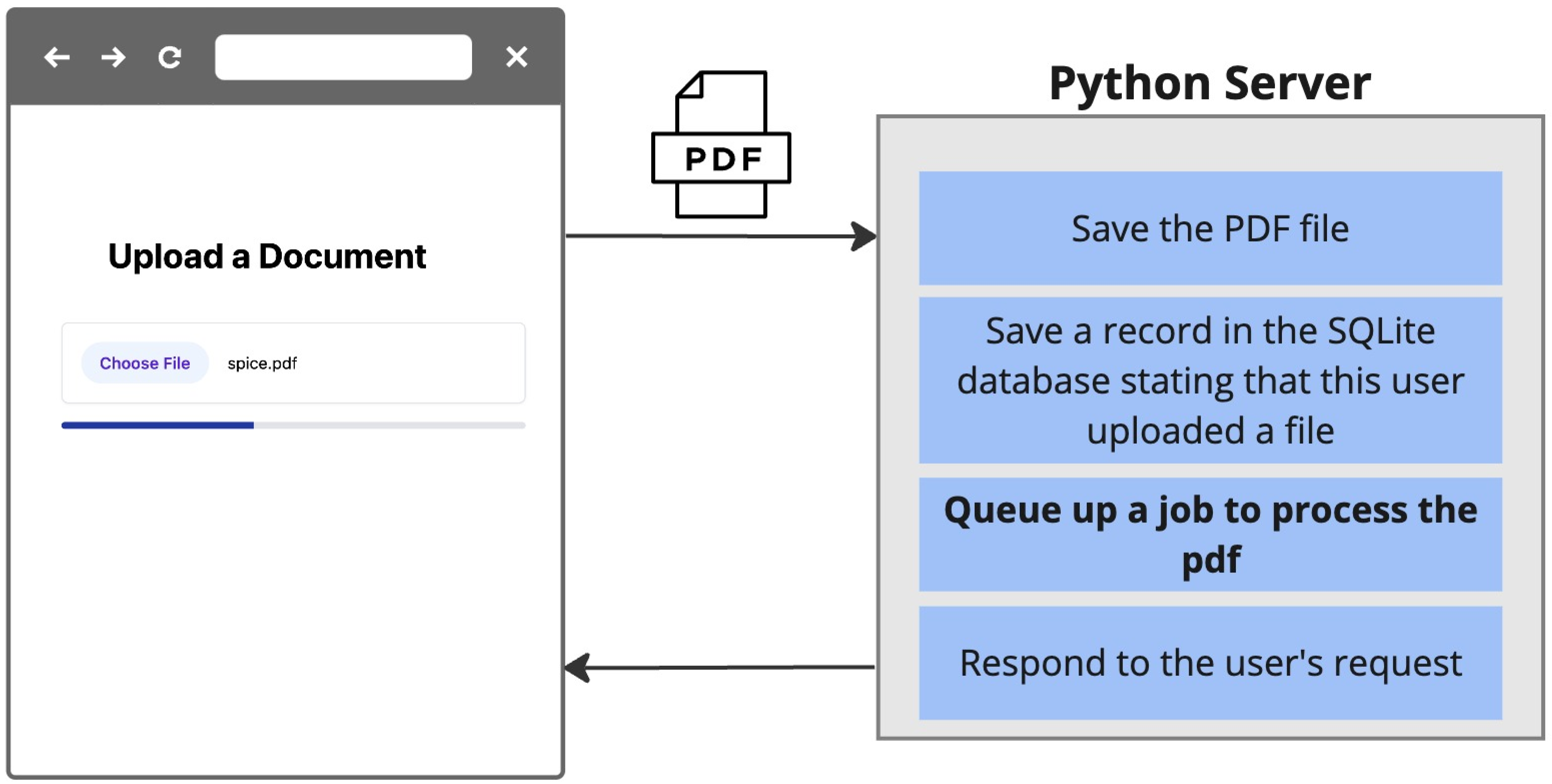
SQLite List of Uploads

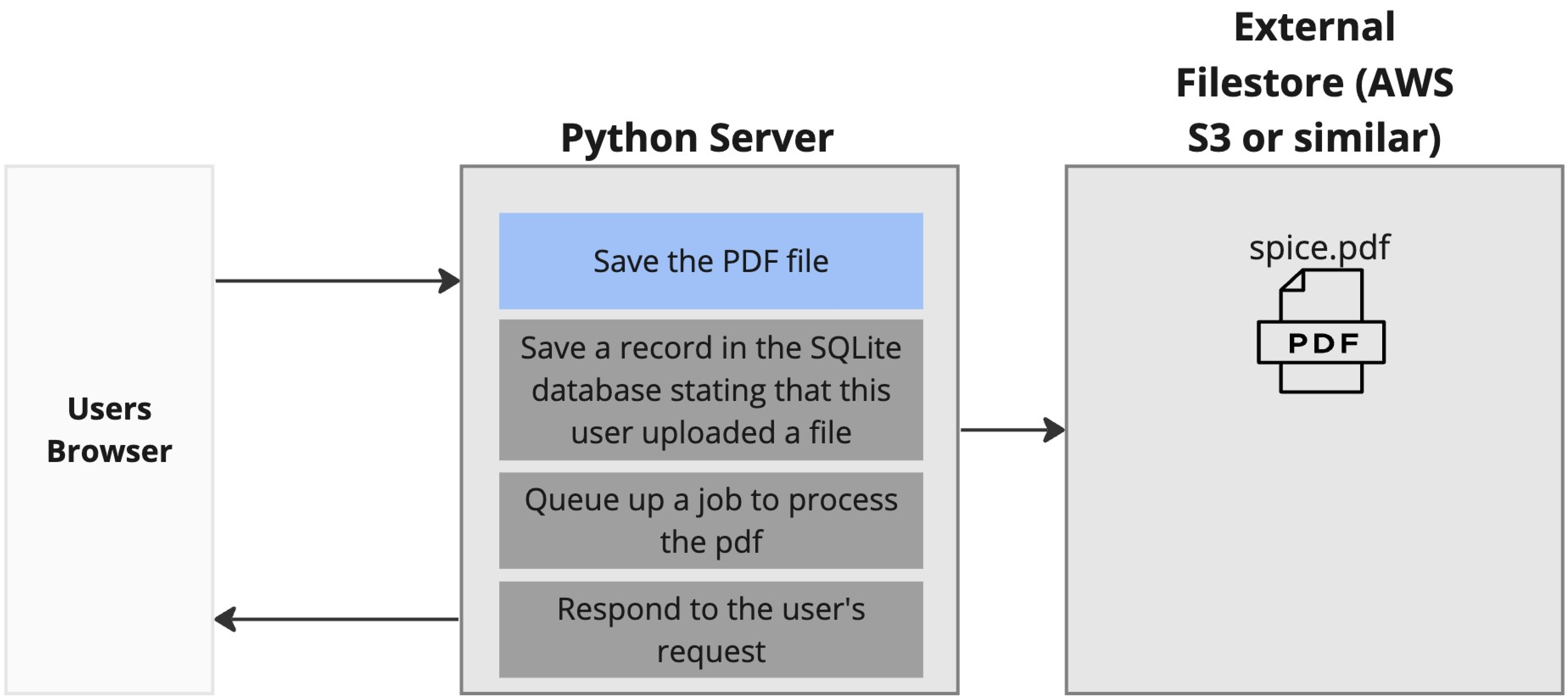
id	name	file_id	user_id
123	spice.pdf	15abc19	5ed9

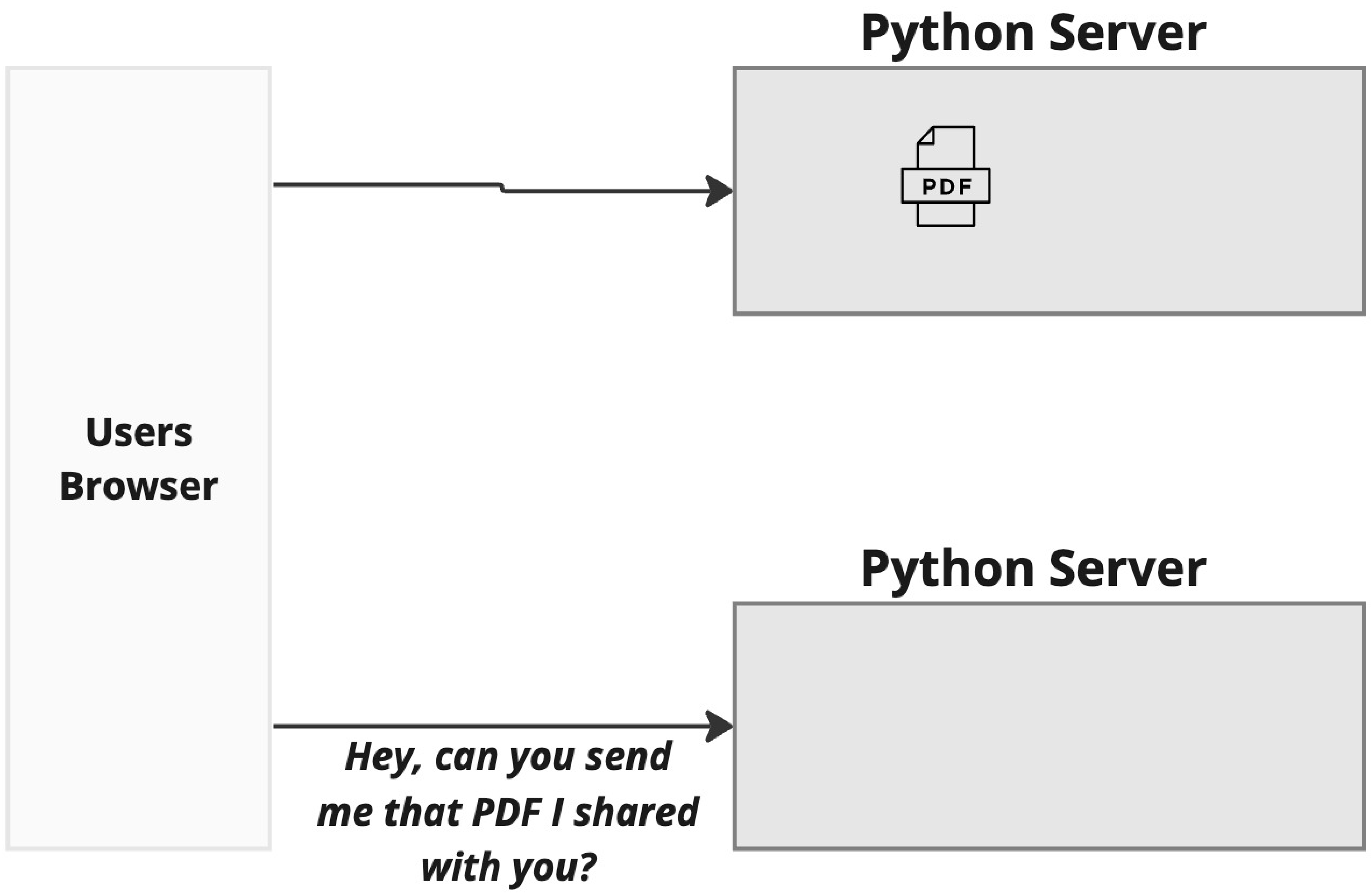
Outside Filestore

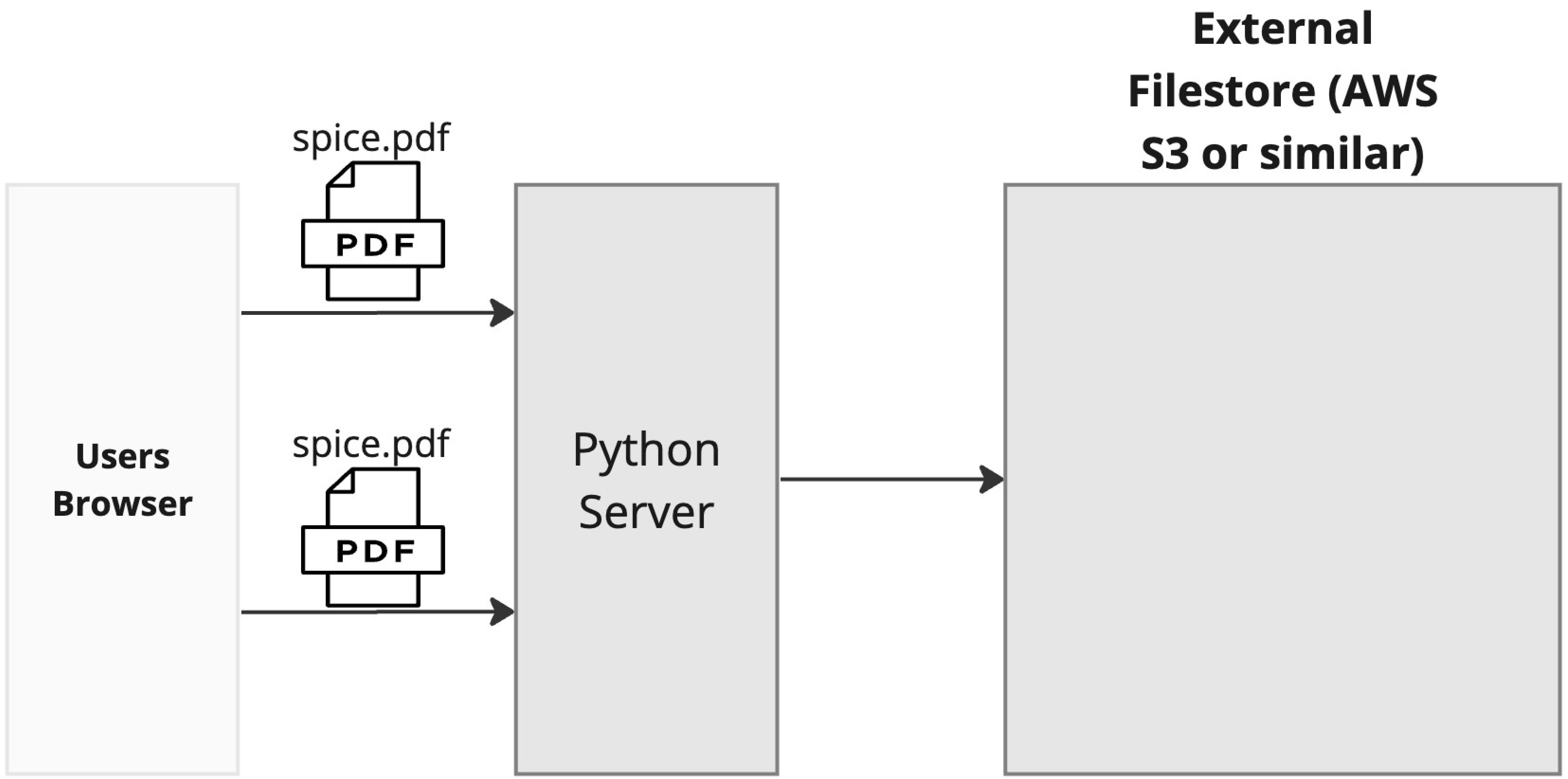
PDF

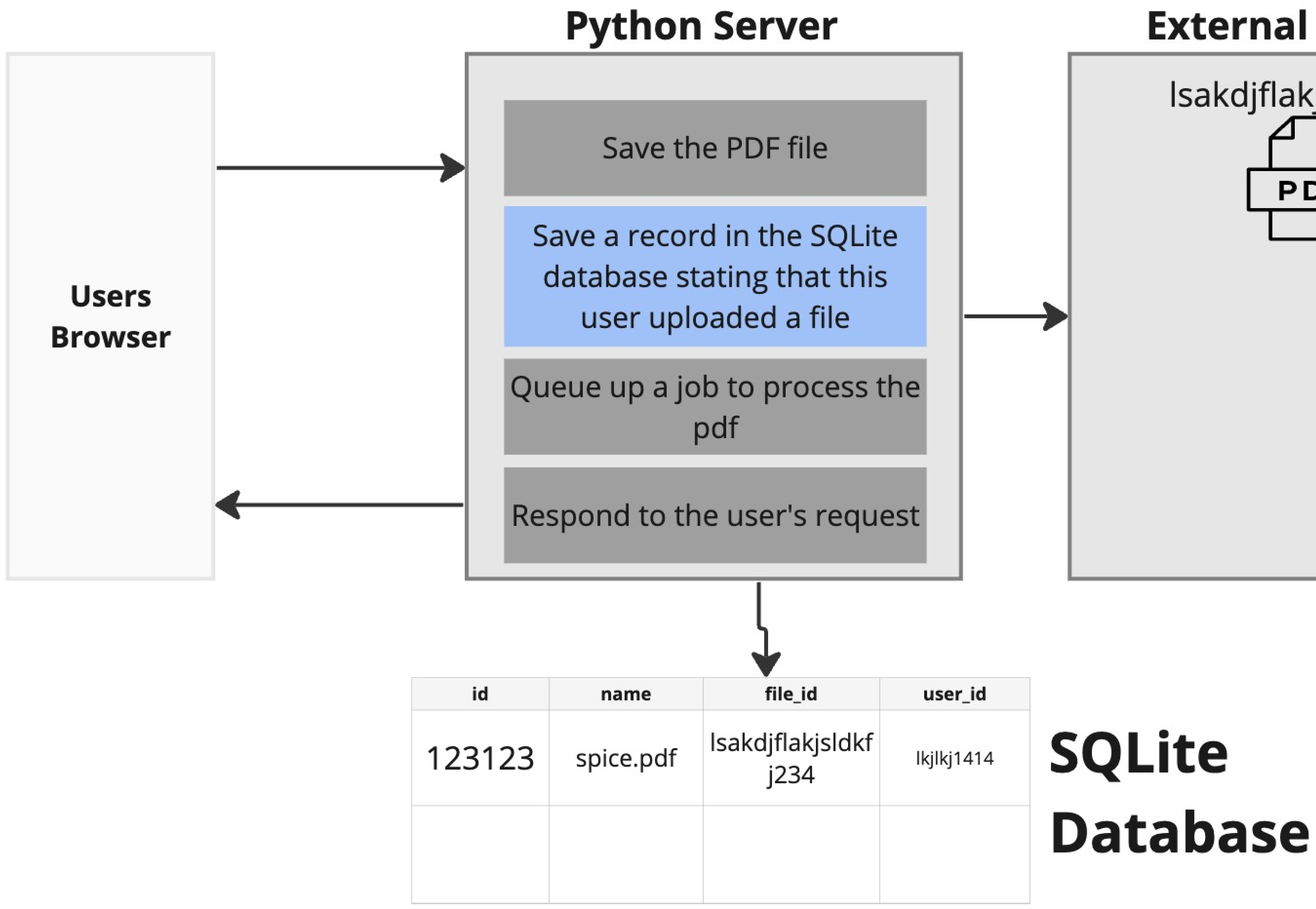
15abc19


Upload a Document
Choose File
spice.pdf
PDF
Python Server
Save the PDF file
Save a record in the SQLite database stating that this user uploaded a file
Queue up a job to process the pdf
Respond to the user's request

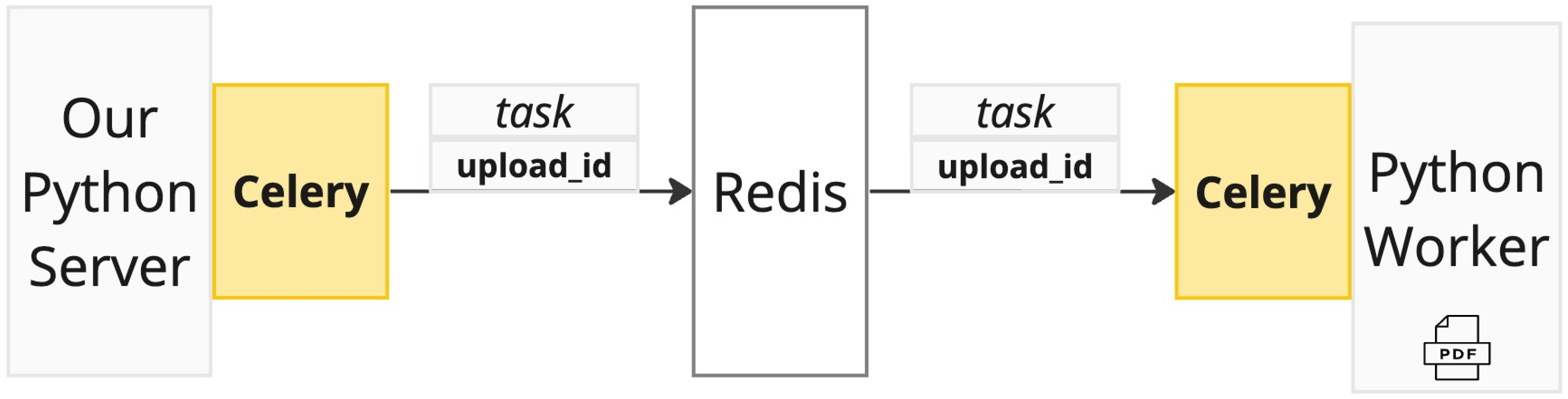

Users Browser
Python Server
Save the PDF file
Save a record in the SQLite database stating that this user uploaded a file
Queue up a job to process the pdf
Respond to the user's request
External Filestore (AWS S3 or similar)
spice.pdf
PDF


Python Server
PDF
Users Browser
Python Server
Hey, can you send me that PDF I shared with you?


External Filestore (AWS S3 or similar)
spice.pdf
PDF
Users Browser
spice.pdf
PDF
Python Server

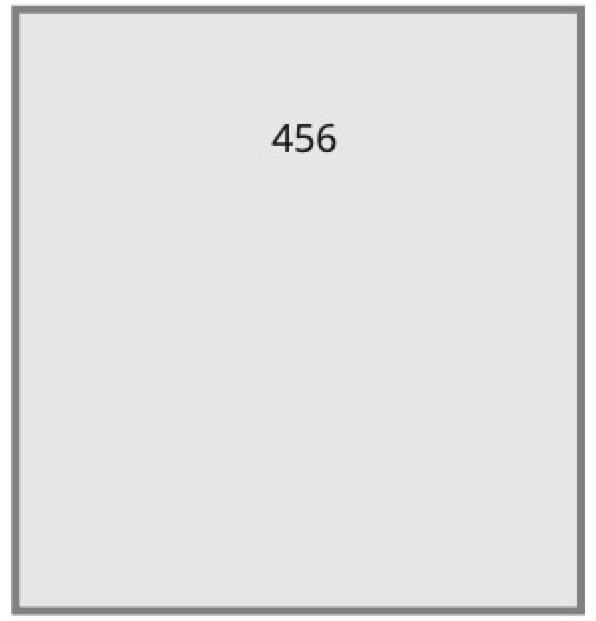

Python Server
External Filestore
lsakdjflakjsldkfj234
PDF
Users Browser
Save the PDF file
Save a record in the SQLite database stating that this user uploaded a file
Queue up a job to process the pdf
Respond to the user's request
SQLite Database

id	name	file_id	user_id
123123	spice.pdf	lsakdjflakjsldkfj234	lkjlkj1414



List of Uploads store in SQLite

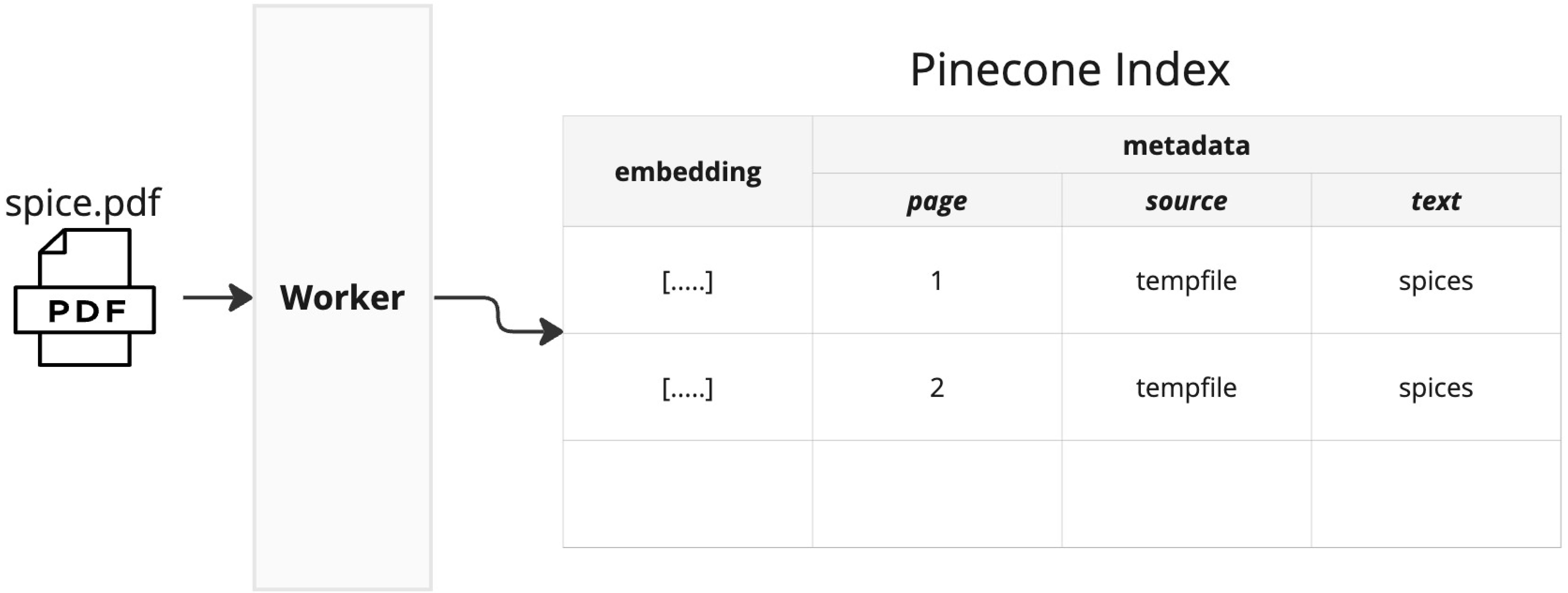
id	name	file_id	user_id
123	spice.pdf	456	789



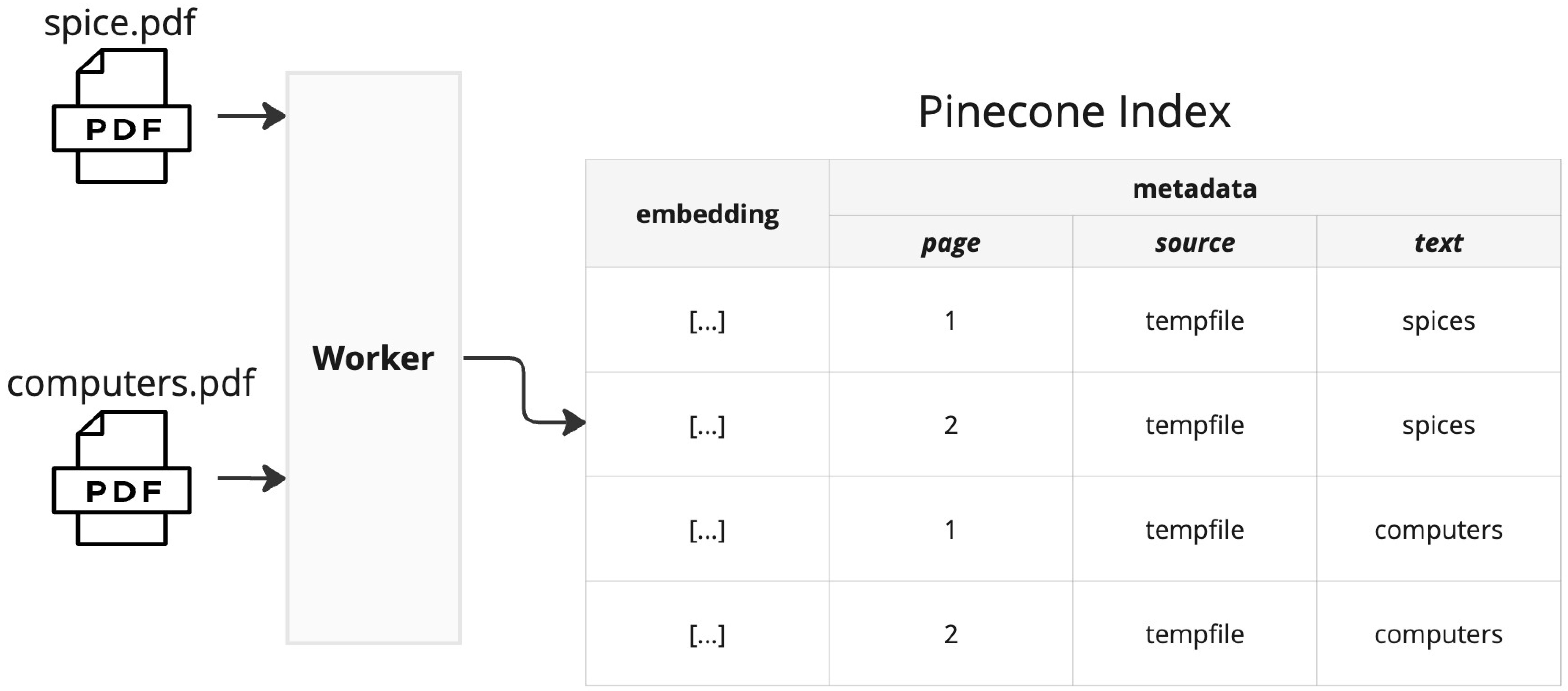
External Filestore

pinecone.io

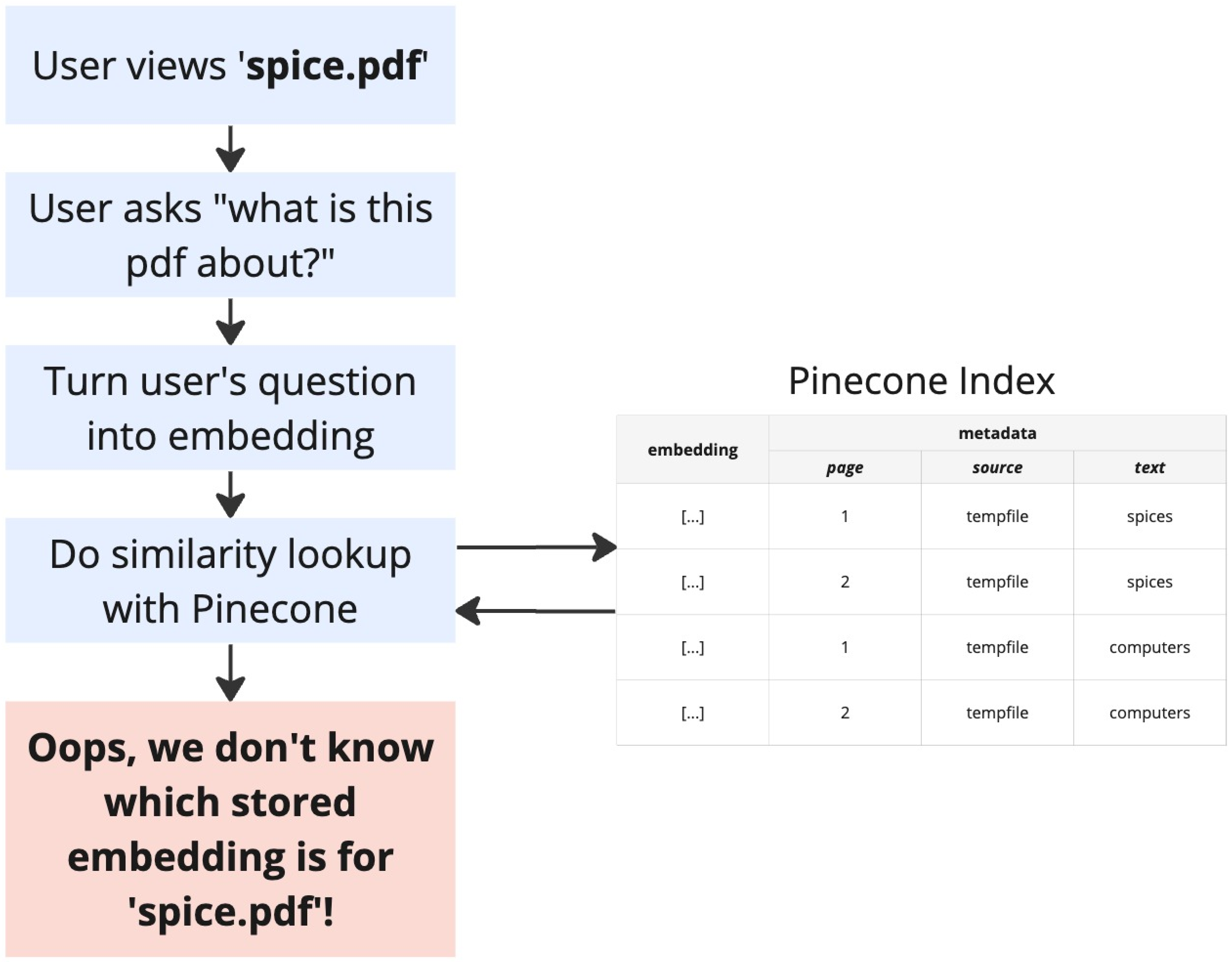
Production-Ready Hosted
Vector Database



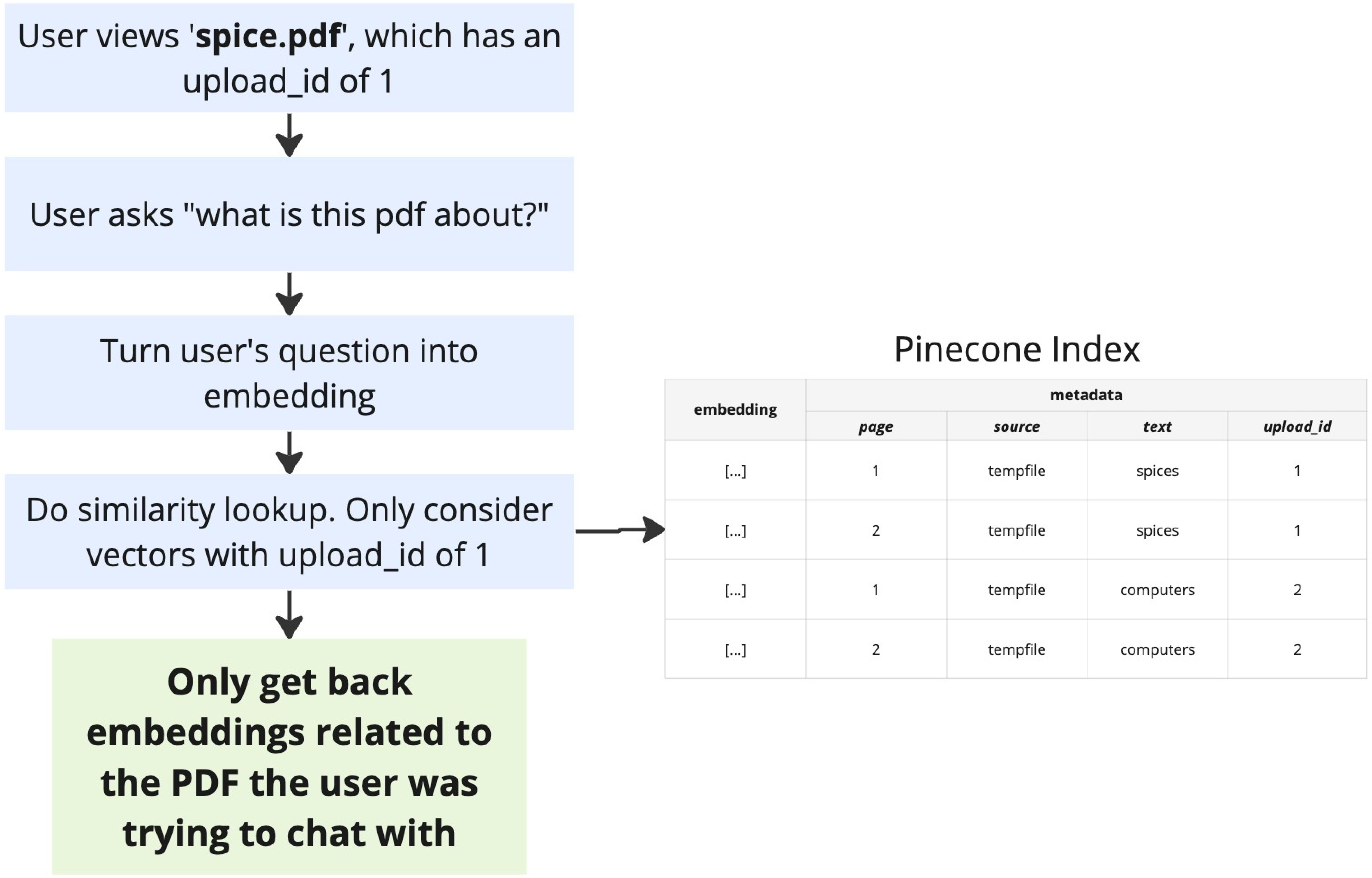
embedding	metadata		
	page	***source***	***text***
[.....]	1	tempfile	spices
[.....]	2	tempfile	spices



embedding	metadata		
	page	***source***	***text***
[...]	1	tempfile	spices
[...]	2	tempfile	spices
[...]	1	tempfile	computers
[...]	2	tempfile	computers


User views '**spice.pdf**'
User asks "what is this pdf about?"
Turn user's question into embedding
Do similarity lookup with Pinecone
Oops, we don't know which stored embedding is for 'spice.pdf'!
Pinecone Index

embedding	**metadata**		
	page	***source***	***text***
[...]	1	tempfile	spices
[...]	2	tempfile	spices
[...]	1	tempfile	computers
[...]	2	tempfile	computers


User views '**spice.pdf**', which has an upload_id of 1
User asks "what is this pdf about?"
Turn user's question into embedding
Do similarity lookup. Only consider vectors with upload_id of 1
Only get back embeddings related to the PDF the user was trying to chat with
Pinecone Index

embedding	**metadata**			
	page	***source***	***text***	***upload_id***
[...]	1	tempfile	spices	1
[...]	2	tempfile	spices	1
[...]	1	tempfile	computers	2
[...]	2	tempfile	computers	2