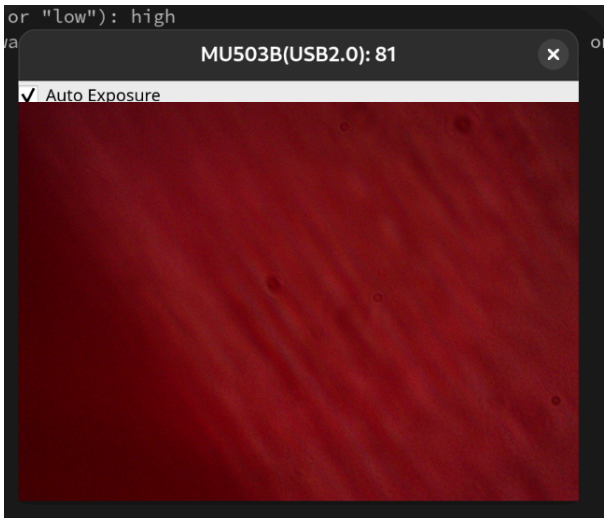
July 7th

qt.py is officially working, I got it to do take an image, first you run app.py, then it calls qt.py, and qt.py sets up the qt window and does all the stuff with the amcam.py api.



(as you can see, an image of it actually working)

however, the 2.9fps is not going to cut it, whatever he did, it sucks

either way, I am moving on, fuck optimizing code, I want it to run lol

July 8th

did the first commit for backend construction, we will see how it progresses over time, I am optimistic, but this will be odd given the structure of qt.py (backend+frontend functions combined). We gotta split it up, so this will be a fun way.

July 15th

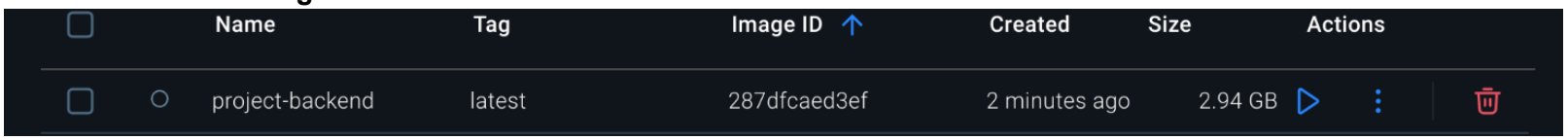
Updates

I have a functional but flawed Fastapi backend for this whole thing. It needs some updates, but all the general features seem to work, except for the resolution changing bug, which I still need to fix (if you change different resolutions during runtime, the code freezes and you don't get new frames)

Two Planned Major Code Changes

- Compose Files found Automatically for Linux/MacOS/Windows scripts
- /Connect removed
- One docker container per predefined device (via serial)
- Make 2 separate Amscope Camera docker containers, each with a 38xxx separate port
- Resolve Naming Issue for Docker Containers (Name:Tag)

Notes - Docker Naming



- At this moment, the container, when built, is named “project-backend”, with tag “latest”. I want this to be changed.
- Ideally, this is “Amscope-Camera” with a tag like “#1” for the actual camera, but I want this setting process to be easy (at least in linux)
- How it works at this moment is that one docker container at a given port can connect to a single camera at a time, but that docker container can see all amscope cameras and connect to individual ones. However, this is not useful if you want to use them in parallel. Ideally, we have one docker container per unique device. I want to make this easy, however, to do this process.

Notes - Broken setup scripts

- Any time we change the shape of this directory structure, the original scripts cannot locate the docker compose files. I want to ensure naming and everything doesn’t complicate stuff.
- Like, for example, the path for docker-compose-backend.yaml and docker-compose-frontend.yaml is currently “/home/framework/personal/github/Amscope-Docker/Code/Project”
- but, if any of those directories change in the path, the Linux/MacOS/Windows scripts will break (they actually currently are since I changed the one directory to ‘Project’, before it was ‘TDC001-project’)
- I would like it to **automatically identify or broadcast the path of the docker compose files** to ensure that this process never breaks and it can properly build each time.
- It will have to be two stages, and I would say the Linux scripts should be updated first before MacOS and Windows

Notes - Container Naming Error

- The line “Container” in the docker compose should be removed. This doesn’t need to be specified and only causes errors. Just remove it.
- For the other frontend container (which I won’t do in this project), also should do this

Notes - Wheelhouse + requirements build problems

```
2025-07-15 16:36:55    File "/usr/local/lib/python3.13/site-packages/uvicorn/importer.py", line 22, in import_from_st
2025-07-15 16:36:55    raise exc from None
2025-07-15 16:36:55    File "/usr/local/lib/python3.13/site-packages/uvicorn/importer.py", line 19, in import_from_st
2025-07-15 16:36:55    module = importlib.import_module(module_str)
2025-07-15 16:36:55    File "/usr/local/lib/python3.13/importlib/__init__.py", line 88, in import_module
2025-07-15 16:36:55    return _bootstrap._gcd_import(name[level:], package, level)
2025-07-15 16:36:55    ~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2025-07-15 16:36:55    File "<frozen importlib._bootstrap>", line 1387, in _gcd_import
2025-07-15 16:36:55    File "<frozen importlib._bootstrap>", line 1360, in _find_and_load
2025-07-15 16:36:55    File "<frozen importlib._bootstrap>", line 1331, in _find_and_load_unlocked
2025-07-15 16:36:55    File "<frozen importlib._bootstrap>", line 935, in _load_unlocked
2025-07-15 16:36:55    File "<frozen importlib._bootstrap_external>", line 1026, in exec_module
2025-07-15 16:36:55    File "<frozen importlib._bootstrap>", line 488, in _call_with_frames_removed
2025-07-15 16:36:55    File "/app/amscope_server.py", line 38, in <module>
2025-07-15 16:36:55    from PIL import Image # Pillow (add to requirements)
2025-07-15 16:36:55    ^^^^^^^^^^^^^^^^^^^^
2025-07-15 16:36:55    ModuleNotFoundError: No module named 'PIL'
```

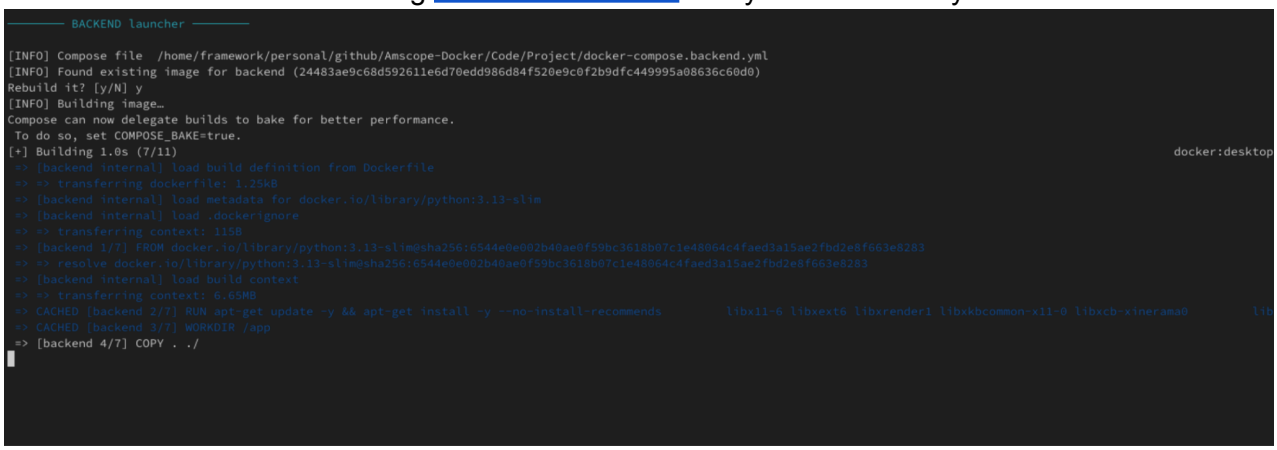
- Nearly always, this is a problem. I haven’t figured out how to have that build out fast. The commands are quite simple, but ya, annoying.
- From the “How To Docker #1” slide 10, there are the two commands you have to run in this directory with the .venv (virtual enviornment) activated:
  “pip freeze --all > requirements.lock”
  “pip download --dest=wheelhouse -r requirements.lock”

```
(.venv) framework@fedora:~/personal/github/Amscope-Docker/Code$ cd Project
(.venv) framework@fedora:~/personal/github/Amscope-Docker/Code/Project$ cd Controller+fastapi
(.venv) framework@fedora:~/personal/github/Amscope-Docker/Code/Project/Controller+fastapi$ pip freeze --all > requirements.lock
(.venv) framework@fedora:~/personal/github/Amscope-Docker/Code/Project/Controller+fastapi$ pip download --dest=wheelhouse -r requirements.lock
ERROR: Could not find a version that satisfies the requirement action_msgs==2.0.2 (from versions: none)
ERROR: No matching distribution found for action_msgs==2.0.2
(.venv) framework@fedora:~/personal/github/Amscope-Docker/Code/Project/Controller+fastapi$
```

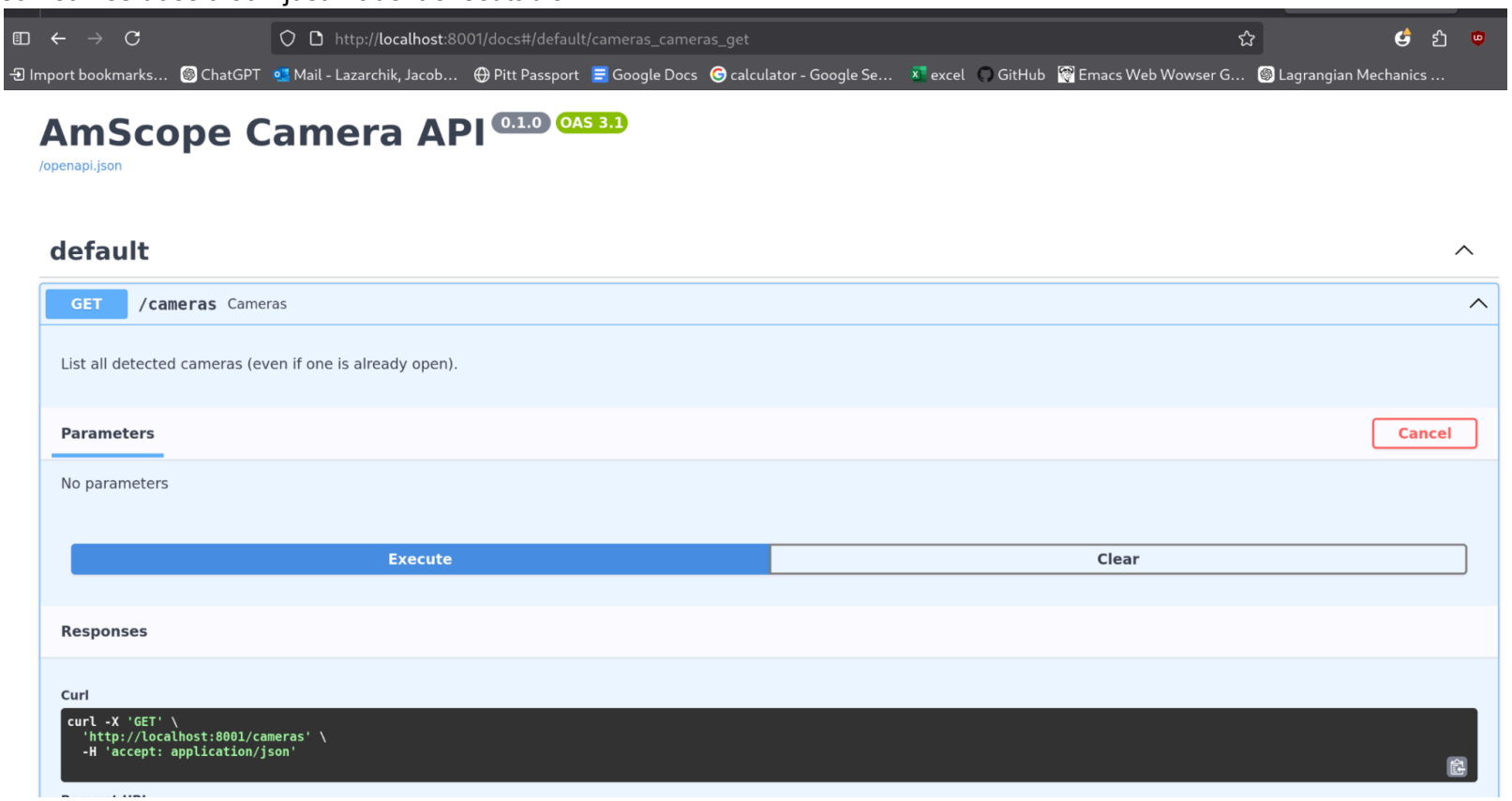
- huh? that’s odd. I will fix this error I guess lol
- seems like it confused my host system’s installed python dependencies with the .venv file, I don’t entirely know why it is pulling from my host OS during this command, but that’s good to know (didn’t know this lol).
- **actually we have to change --all to --local, I should include this in a revised presentation**
- **pip freeze --local > requirements.lock**

Notes - building the backend container again

Let's see if it works. I am running backend-launch.sh in my /Linux directory



btw, run this in the terminal in the directory like "bash backend-launch.sh" if you don't want it to instantly close if it fails. It sometimes does that. I just made it executable.



pulled up my webpage pretty well, but of course, I am on linux and don't have USB passthrough to the docker container set up, either way, it works! good.

Anyway, I will be done for today, I need to make a presentation for tomorrow