01/24/2024:

* Went through a rough architecture/system design for the software
* Discussed what a package should carry, what will be on the front end, where to put the ML model, and how to get logging information from the backend and display it on the front end in real time

01/29/2024:

* Learned and used WebSocket to get logging information from the ML script and display it on the front end in real time
* Refactored ML code so that every function is in a designated class
* Refactored manager.py and mnist\_model.py to enable manager to update progress, allowing mnist\_model to focus on running ML tasks
* Setup a websocket in server’s app.py, pass it to manager, and let manager.py emit updates to the client app.js
* Currently debugging why socket.io is emitting the same information multiple times (id 0 emits 1 time, id 1 emits 2 times, id 2 emits 3 times, …) 🡪 Used socketio.once() to solve the problem
* Learned how to close/disconnect a WebSocket to avoid memory leaks/security issues.
* Used ChatGPT to refactor code and make it cleaner.
* Q: Don’t know how to check if the websocket is properly closed.

02/07/2024

* Carrying out issue #1: minimal front to back
* Just a simple front-to-back integration that lets users pick the params and train the model immediately. No Producer yet.
* Currently debugging this error: Access to fetch at 'http://localhost:9000/start\_experiment' from origin 'http://localhost:3000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled. 🡪 Here’s what I found: The error stems from a security mechanism that browsers implement called the same-origin policy. The same-origin policy fights one of the most common cyber attacks out there: cross-site request forgery. In this maneuver, a malicious website attempts to take advantage of the browser’s cookie storage system. 🡪 Identified that it is mostly a **backend** problem, simply created a key ‘Access-Control-Allow-Origin’ – value ‘http://localhost:3000’ for the backend response object (which is a dict/json)
* Also modified the code a bit to remove proxy in package.json, set the server to use port 9000 instead of 5000.