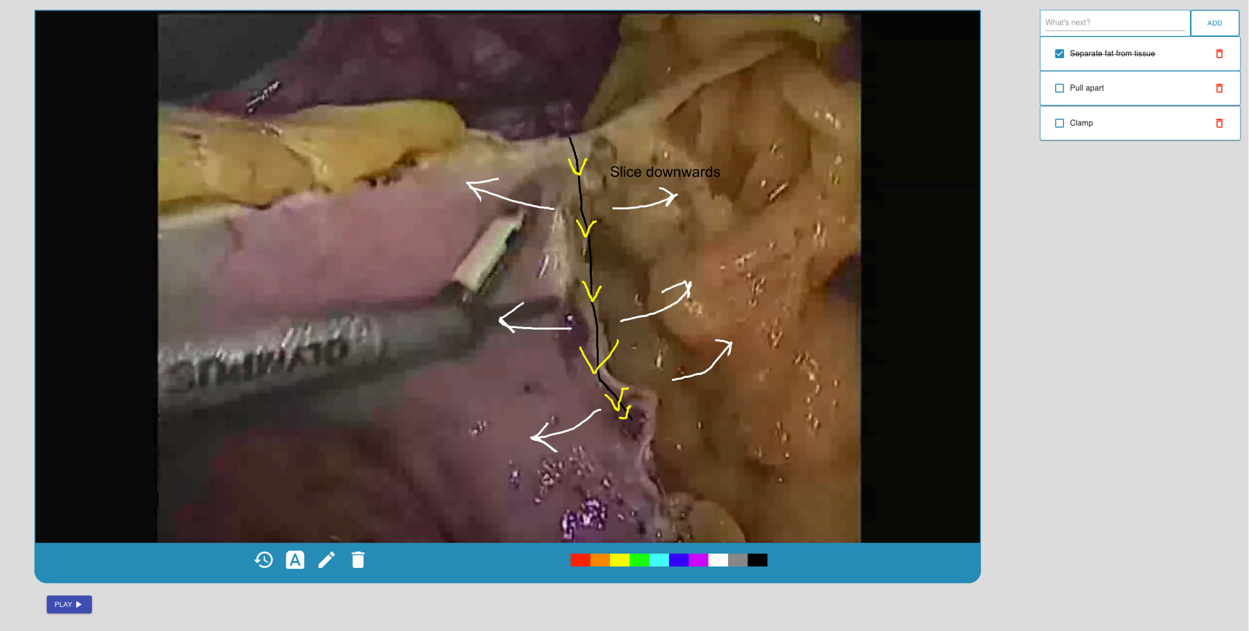
Arcos™

Educational Surgery

This application makes it easy for a professor to observe a surgical student's plans before executing them, while in a live environment.



- ☑ ~~Separate fat from tissue~~
- □ Pull apart
- □ Clamp

The tool works with any USB connected camera, and lets the user freeze the image, allowing them to

- Mark up the image capture
- Add textual annotations
- Get reviewed by a supervisor.

Additionally, there is a Task List in which items can be marked completed or deleted. With these features, a professor can be sure their student works safely.

Minimum System Requirements for the Host

- 2.8Ghz CPU
- 8GB of RAM
- USB 3.0
- Linux or macOS
- An install of FFMPEG on the PATH

The USB requirement is the most important; without the transfer speed provided, the application will really falter. Finally, the client computer should have a modern web browser; Firefox on macOS is known to not work.

Technical Challenges

The team had a lot of challenges with our front end test framework; this was our recurring devil. Ultimately, a long conference with Venkat helped us understand the necessary techniques; unfortunately, a lot of the app's code had already been written without a test-first mentality. We've worked to overcome this problem, helped along by the fact that our backend code was much more straightforward to test; thus, we were able to work in a TDD method for that portion.

Our front end coverage is increasing rapidly, and by the time this document is read the coverage numbers will likely be out of date; the final number will be presented in our product demo. At this time, we're sitting at 88% coverage on the back end and 46.5% total coverage for the front end.

Key Things Learned

React! We all learned React for the first time — some of use had never even used JavaScript. This has been a blessing and a curse, since that lack of familiarity was part of what led to our primary technical challenge: test frameworks. But, at the same time, we all learned a valuable framework in the production of a complete, working application.

We also had several team members working in Go for the first time, and we also dabbled in the new framework WebRTC, which was integral to our project. Basically, we learned how to take a product from concept drawing to something that's only a few tweaks from being marketable (this is primarily a function of us not having access to something like an AWS or App Engine instance to work on). This was a skill in itself, and taught all of us how to pick up technology as needed.