FINAL PRESENTATION SCRIPT

Person 1:{

//todo

}

Person 2:{

//todo

}

Person 3:{

//todo

}

Travis:

“Thank you Person 3(I think it’s Michael right?). Welcome everyone, I’m Travis and I’ll be walking you through a short demo momentarily.

As you can see here, I’m first presenting the three cmdprompts powering Git Going. A flask python REST api, a nodejs/express REST api, and a standard react APP. These power the website, as you can see here. This is our login page, and we’ll run some tests on it. First, the standard user input on every website, the empty case. It throws an error, as it should. Next we try without a password, and again, another error. Now we fill in both fields and get an incorrect username. In the code, the comparison of user input is against a literal string, but the logic is in place to make an authorization API call. The server I use here isn’t robust enough to handle it on its own, yet. Now with both fields correct, we pull up the localstorage to show how the site tracks login state. While this value is true, we’re loggedin, but if it’s deleted or false, it redirects us to the login page. As you can see, we can’t access certain pages without this state. Of course, users don’t want to delete something in local storage when they want to log out, so we give them a log out button! This sets their state to logged out, returning them again to the login screen. Now we’ll look at user registration and how it pushes entered data into our AWS hosted MYSQL database. Here you can see our latest entry, as well as many of my trial runs recording the demo, lol. Now we’ll also check for standard error messages for incorrect fields. It all checks out so we’ll click the redirect to home. This obviously should happen automatically, but again, the API I designed isn’t robust enough to handle the redirect itself yet. Now to show you guys the primary functionality of the site, a code review that displays the diff between two files! First we select our pullrequest.json file to structure the data as a dictionary for the API request. \*pause\* And if that sentence confused you, ask Michael about it after the fact, because just hearing it from him the first time made my brain whimper a little. \*unpause\* With all that taken care of, the code review is created on a file, previously defined, with some changes I’ve already made! As you can see, there are – symbols to indicate what’s been removed, and + symbols to indicate what’s been added. Now we’re make some changes to the same file ourselves!. As you can see, the new diff is generated on our site! Now let’s take a look at the flask API that’s been running the entire time, listening for requests. As you can see, I’ve made a few diffs already. It’s pretty neat, huh? Now to wrap up, let me demo for you, just a little, how the express api can handle DB requests through the browser! First I pull everyone in users. Quite a list, we saw it before. Then I check for admin, which maybe wasn’t the best idea because there’s no one there. And finally, I do a check for a name I’d been using in previous test cases. And it all checks out! That concludes the demo, and

now I’ll take a moment to talk about the road ahead. Obviously, we have a lot of work ahead of us still, but I am proud to be here at the end of our first Capstone quarter with my group. We’ve all grown so much as students, as developers, and as friends. And as we take a moment to recharge, and reflect, I know we’ve already got our eyes on what’s to com. As demonstrated, certain key features still require implementation; Integrating Christina’s amazing notification system into our review system. Generating inline commenting and messaging on reviews. Having a login system that actually, y’know, authenticates people. Our database API needs to be much more robust, capable of securely retrieving information. This and more lie before us, but I gotta say, look at how far we’ve come.

Thank you, very much everyone. I’ll now open the floor for questions.