Compile.io

By: Joshua Palamuttam, Michael Riley, Zachary Foote, Zachary Thelen

Table of Contents

Executive Summary

The purpose of this document is to detail the solution implemented to address the issue with the way most student-written code is submitted and graded at Rose-Hulman. Currently, there is no central system for students to submit code and have it quickly graded with feedback. In most cases, Rose-Hulman students will submit code and wait for tests to be ran on the code. It may take a couple days for the students to receive their grade and any feedback. The system being described in this document is a potential solution to this issue: A central system that all CSSE courses at Rose can use to have students submit code that will be compiled and have tests ran on it with near-Instant feedback for the student. This document will also describe the major challenges of the proposed solution, and the overall design of the system.

Introduction

This document is a living document on the Compile.io project developed by Josh Palamuttam, Zach Foote, Zach Thelen, and Mike Riley. Compile.io's purpose is take the problem outlined below and provide an efficient and elegant solution.

Problem Description

Currently, there is no unified grading solutions for CSSE professors at Rose-Hulman. Students submit code through various servers such as the PLC grading server which compiles and runs scheme code for the Programming Language Concepts (PLC) class. Another example is a student might commit their code to a GitHub repository and a professor or a teaching assistant will have to manually run the test cases on the students code and give them a grade accordingly.

Needs of Compile.io

- Professors track and deploy assignments through Git, SVN, and other version control services; there is no centralized grading server.
- Professors give students grades by running tests, then looking at the result of said tests and assigning a grade that corresponds to the test results, often using scripts they themselves create.
- Graders run tests and give feedback to the student
- Professors assign and deploy assignments across multiple sections to many students.
- Students submit assignments via a version control service, or they upload specific files to the course moodle page.
- Student assignments are isolated from other assignments; students are regulated to their own repositories.

Features

- Software will compile multiple languages (Java, Python, Scheme, etc.)
- System will be able to cut off submissions after a deadline
- Software will be able to prevent infinite looping by timing out the code
- Software will be able to prevent code from bleeding server resources
- System will be secure by preventing hacking
- System will authenticate users with RoseFire
- Software will prevent unauthorized students (like students who aren't enrolled in the course) from uploading code
- Software will have different classes that correspond to different professors
- Software will contain a roster for each class
- Student is able to upload code to the server
- Student will be able to view the results of the uploaded code.
- Students will not be able to upload submissions for other students
- Grader should be able to run student code
- Grader should be able to provide feedback to student code
- Professor should be able to specify test cases for an assignment on the system
- Professor can upload an assignment for a specific class
- Professor can upload roster for a class
- Professor can set deadlines for each assignment
- Professors can switch views between different classes
- Professors can edit assignments
- Professor should be able to run student code
- Professor should be able to provide feedback to student code

Use Cases

Actors:

Professor
Student

Use Cases:

1. Student/Professor: Login to system with ID
    a. Description: User enters user ID and password to gain access to the system
    b. Basic flow:
        i. User enters ID
        ii. User enters password
        iii. User submits credentials
        iv. System uses RoseFire to authenticate the credentials the user submitted
        v. RoseFire logs the user in with the appropriate permissions
    c. Alternate flows:
        i. At step 3, user submits invalid credentials
        ii. RoseFire notifies the user that the credentials are invalid and allows the user to try again.
        iii. At any time, the user exits the system
        iv. The system closes
    d. Preconditions: User has the system open
    e. Postconditions: User is logged into the system

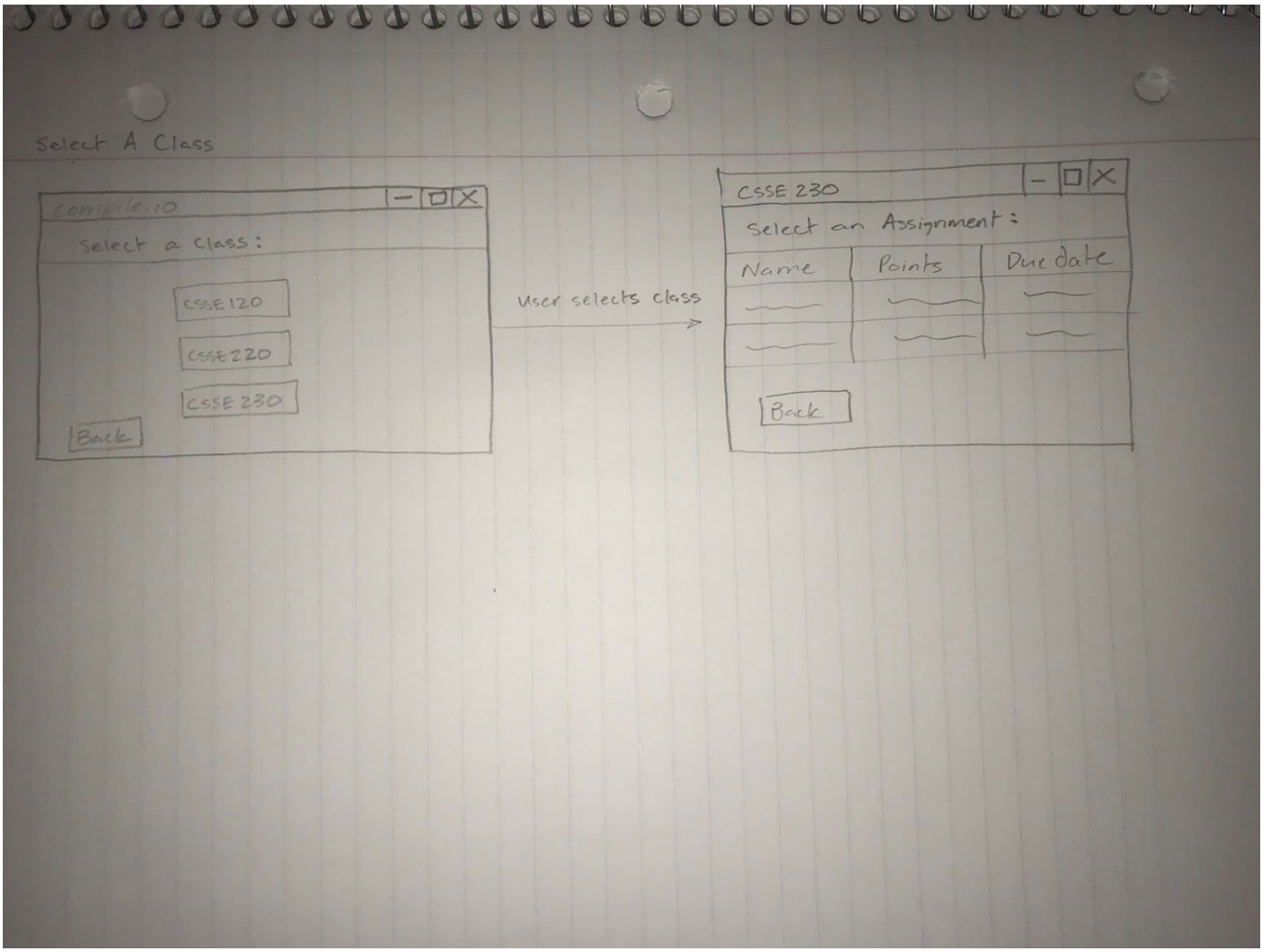
2. Student: Upload a zip file for grading
    a. Description: User uploads a zip file to be graded
    b. Basic flow:
        i. User selects assignment to upload to
        ii. User uploads zip file under the specified assignment
        iii. System runs professor uploaded test cases on the uploaded file
        iv. System adds newly ran test results to Firestore
        v. System adds most recent run to the list of runs on the UI and opens it for the user to view.
    c. Alternate flow:
        i. After step 2, if the user has uploaded code ina language that doesn't match that of the assignment, then the system will return the user to the upload screen and ask them to upload code of the correct language.
    d. Preconditions:
        i. User is logged into the system and has selected the class that they are in and wish to upload files to.
    e. Postconditions:

        i. System has successfully compiled and graded the uploaded code and displayed the result for the user.

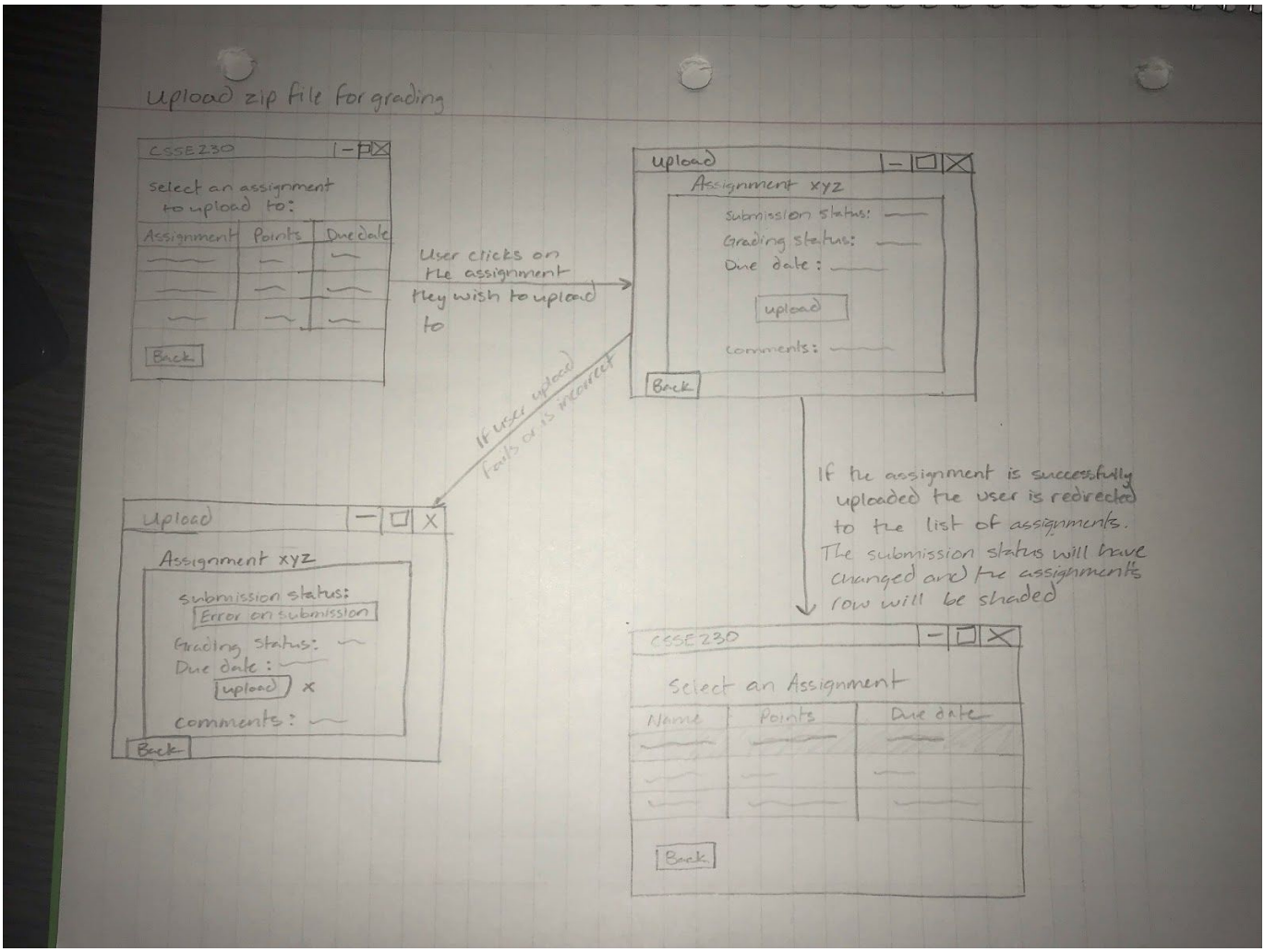
3. Student/Professor: Select a class
    a. Description: User can select which class they are taking in order to pull up the assignment list for that class
    b. Basic flow:
        i. User selects the name of the class that they wish to view
        ii. System brings user to a list of assignments for that class
    c. Alternate flow:
        i. After step 1, if the user is not in the class or has taken it they will be redirected to the list of classes and notified
    d. Preconditions:
        i. User has the system open and is logged in.
        ii. User has access to the assignments for the class they are in, and are able to upload or view results from past assignments
    e. Postconditions:
        i. System has brought the user to a list of assignments for the class they selected.
4. Professor: Upload unit tests
    a. Description: User can upload unit tests to be ran on code that is uploaded to a specified assignment.
    b. Basic flow:
        i. User selects the assignment that they wish to upload unit tests to
        ii. System opens up the assignment
        iii. User selects the option to upload unit tests for that assignment
        iv. System prompts user for a zip file of unit tests
        v. User uploads the unit tests
        vi. System saves the unit tests to the assignment specified to be ran on any code that is uploaded to it.
    c. Alternate flow:
        i. After step 3, if the assignment already has tests uploaded to it then the system will prompt the user as to whether or not they wish to override or edit the previously uploaded tests
    d. Preconditions: User has the system open and is logged in with the class they want to upload unit tests to selected.
    e. Postconditions: The desired assignments have unit tests associated with them that will be ran on any code that is uploaded by students under that assignment.

Low Fidelity Prototypes

Use case: Select a class



Use Case: Upload a file



Solution Description

A more elegant grading server is needed, one that can accept any coding language and run tests for that respective language. That is Compile.io, in a nutshell. Professors will be able to automatically check and grade student assignments against test cases that they have written and posted on the server. This elegant grading server negates professors having to manually run test cases from files downloaded from repository service, like GitHub. Upon submitting submitting their code file(s) to the server, students will receive quick feedback as to which test cases failed, along with the corresponding error messages.

Backend Discussion

Upon reviewing the requirements and researching both methods and tools to be used for the project, the team decided that the back-end will consist of a web server being run by Rose-Hulman on Apache. On this server when students upload code the code will be run on a docker container for the specific assignment that the student is uploading to. This docker container will have the test code on it and will run the test code using the student's uploaded code. The server will then output any error messages that the code outputted, or if no error messages popped up either on the test cases or in compilation the students code is correct and that will display on the front end.

Frontend Discussion

The front end will largely be javascript and css based. The team is looking into possible javascript frameworks like angular in order to facilitate a user friendly application. Usability is the highest concern with regards to the front end, with general look and design not being as important to the clients. As a result the front end will likely be built only enough to work at first, with better designs as backlog features.

Key Challenges

One of the main challenges early on is figuring out what tools we should use to create Compile.io. We settled on using a server created by Rose, and having docker implemented on the server to simulate each of the different coding environments necessary for tests. Since none of us have used Docker, another challenge would be to learn and research Docker to use it in a professional way for this project.

Stakeholder Interview Transcriptions and Notes

Interview with Sid Stamm:

Josh: All right so this interview's for our senior project. Our senior project called compile.IO and it was a project brought into us by a couple of Rose grads from a couple of years, I think three years ago. Yeah okay. Whenever Sam graduated. So what they wanted us to do was create a unified grading place for computer science majors much like RoseBuild. And they weren't aware of RoseBuild and actually we weren't aware of RoseBuild until recently. But we would just like to get your opinion on what we should put on our application, I guess, of compile.IO. So the first question I guess I'd have to ask is what right now, what gripes do you have with the way computer students are submitting their files to you or you know if there's any just gripes you have in general with the current system.

Sid: Yeah big question. Well let me tell you a little bit about how I do grading for 130 to give you an idea of kind of like what I do and then maybe we can talk about what what could be better afterwards. So in 132 we have a bunch of C programming exercises that need to be graded and currently with the students do is they'll check out a Git repository and I will have placed a folder in the Git repository like a template for the C project and they'll have to fill out some functions and maybe write a new file and add some stuff to the file and then commit and push their Git repository. And then when I grade it I check out the students get repositories and I have a really crappy script that does a c-h route into their repository and runs them automated tests on, really crappy automated tests, on their submission. So the grapes I have with this are, there's a number of things, we can start with Git. We've had file system errors on the Raspberry Pis that I don't think is get Gits problem but every once in a while they give repositories get corrupted, and the solution is to delete a couple of internal Git files and pull from the server and it solves the problem. But freshmen are scared of this and rightfully so. I mean first of all Git is not easy to use.So I guess that would be a major gripe is that there's no version control system that's super easy to use without; well with with things like this going on it just makes it even worse. The second thing is that I have to run a script to check out all the repositories and then copy in files and add and commit them to every repository. And so I have a script that does this it goes through checks out all the repos, adds files to each repo and pushes the files and so the students can can get the templates for the assignment. While that's automated, merge conflicts are non-trivial. And so it's kind of a pain to resolve that if they've committed something while I'm doing the population, right. What's even worse is when I cause a merge conflict because they've done a pull. They're doing some work. I push new stuff and then they do a commit before they pull

again. Then they have a merge conflict. And again freshmen, Git, haven't used it before, it freaks them out. Even though it tells them what to do, still freaks them out. So the merge conflicts are kind of like number two. The third thing is like the grading has to be done manually. So I run a script that iterate through the repositories, pulls them, compiles them, runs tests, but the script is pretty fragile and if something doesn't work right I have to go in and manually inspect the files and it's kind of a pain in the butt. And I'm happy to provide you with an example of the script that I use, if you want to see kind of my workflow I can I can demo that for you as well. But it would be really cool if we could have something auto graded, you know: students do a push and it tells them how they did kind of thing, including like formatting and an organization of code like a linter would be nice. We don't have that. Those are kind of my main gripes

Josh: Okay.

Mike: So when you're running code and there's an error in a test case, it just, the script just breaks. Is that what happens?

Sid: Yeah. So the way that, it's a bash script that I have. The tests are in a c file and if one of the tests fail it stops running the c file. It just tells me which test fails. Some of the tests I've written, I'm using a C framework called Unity. It's a very simple C unit test framework. Works really well. You can configure it to move on and run other tests if one of them fails which is nice, and it tells you which test fails. It’s pretty good, but not amazing, but it's pretty good. The bash script that I have runs those tests but it also runs some other tests that are like Io based. And those when one fails just says this one failed and and continues so it works okay. It’s not great.

Josh: All right.

Sid: Also some of the students forget to commit one of their files like they'll forget to do a Git add on a file if they've created a new file or if they modify file they’ll forget to do Git add and I won't get the file. This is especially problematic during exams. OK. And then also some students will Git add everything including binaries and object files and things they don't need to Git add, which makes my scripts break sometimes because I will try and compile it and it won't go all right or when I'll try and push a comments file to them it'll try and push my binaries as well

Josh: That's a pain in the butt.

Sid: Yeah so that's that's Yeah what I got.

Zach: What we were mostly looking at for, in terms of this project, was something along the lines of like a PLC grading server where you would have you as the professor would go in and say OK here's my classes, here's everyone who can access those classes, and here is assignment one, for instance. And you put all those test cases there on the website. And students then go and take their code and upload it simply by just dragging the file in or by finding the file and uploading it. And then it runs the tests, tells the students exactly which test failed. And then based on your preferences could then either immediately give them a grade on that or allow them to go back and fix things until they have a full score or whatever. What do you think about that. Do you think that would be a good way of testing things for for your sake or.

Sid: I think it would be OK. Now there are a lot of tools out there that already do this. I use one, in the early days of 132, for ungraded exercises where a student logs into a website kind of like coding back Web site and they enter code and then hit submit and it runs unit tests and tells them what worked and what didn't. And he shows them the progress bar. So it's kind of similar. They're typing code instead of uploading it but they could drag and drop if they needed to. The problem that I have with this is that you have to write the code in a certain way and have all of the unit test specified perfectly in it. It runs these jobs on the server, which are fine, but input and output for C programs is really hard in this framework and the open source project I've been working with, it's really bad. I can also give you a pointer to that open source project. I had to fork it and make some modifications to get input and output to be testable in this framework. Because python a Java were great in the framework, I had add C. It's really a pain in the ass. I like the idea, it would be cooler if it was automated via like commit hooks. you could easily do that with subversion commit hooks, I imagine you could do the same thing Git commit hooks, where they do a submission and it runs the tests and then emails them a result or they can log into a web interface and see the result because we'd like them to keep things in source control for safety. Drag and drop is great too but that that would be even cooler. Automated tests are freaking great, especially if they get feedback on their formatting and style with a linter so. I don't know, did I answer your question?

Zach: Yeah, no definitely. Just thinking about it. Have you had a chance to look at RoseBuild.

Sid: Mmhm.

Zach: What, are there anything, any things with that that you think you'd like to see different. Because I mean we're kind of in the situation now where we're kind of doing the same thing and we want to add some sort of value. We don't want to just reiterate things that are already done.

Sid: So there are a couple of things that have kept me from using Rosebuild. The first thing is that there is no CI for C built into Rosebuild right now. And so I can't do an C testing on Rosebuild. The second thing that's keeping me from it is the the way assignments are deployed and distributed to students. There are so many repositories created and my current workflow is one repository for student one folder for assignment. And Rose build uses Git repositories one per student per assignment.

Zach: Oh.

Sid: And that's overkill for me and it's complicated for the freshman and I don't like it because I don't, it seems overly complex to me. I'm sure it's very good for some things but it seems overly complex. Something more along the lines of the PLC server would be better. Like if we could add c to the PLC server I'd be pretty happy. Especially if you can do Git commit hooks to add C, and yeah that would be swell.

Zach: So what you'd rather have is, so let me see if this works. Something like a PLC server where you can just upload code to it or what you could do is hook it right up to a repository any repository that you make for a student and then say every like when you do this commit with this sort of a message it sends it to this test and runs it automatically.

Sid: That'd be nice. Bonus points for extra stuff like a nice web interface where I can look at the instructor's repository and click on the template directory and say populate student repos. Where it would do the the Git clone and copy things over and then it can push for me. That would be super great because right now what you have to do is I have to clone all the student repos and I have to keep copy the student repos. And this is OK, it gets worse for things like 232 where some of the student repos have lots and lots of files and stuff and sometimes extra big files. I don't want to do a deep clone. I just want like a subdirectory or something like that. And if we can put that on a server so it clones the stuff, add something, and pushes it or just just does an add and push without doing a clone that would be nice. I know with subversion you can actually take a directory and use url based commands to do adds and pushes.

Josh: I think Git does that too.

Sid: Yeah I don't know how to do it on Git. But that would be super nice. Also with subversion you can just do a sparse checkout where you just check out the route with the list of directories and then it won't populate the directories unless you ask it to. So if there's like a 20 megabyte folder in every student's repo and you don't care about that you don't have check it out. Anyway this is me ranting but yes something like the PLC server with some nice user interface that makes it easy to populate assignments and easy to get like a table of grades would be super cool. C support is kind of like the main thing. None of our automation does C right now, which is lame. It's the oldest language we use here. None of our automation does it.

Josh: Are there any security risks that you can think of right now with what we're trying to do with uploading files. Like have students tried to, you know, hack.

Sid: Yeah absolutely. Yeah. I mean there's there is. You're gonna have to deal with tight access control problems. Right. Where a student who, especially with C, a student who can submit C that's compiled and run may be able to submit clever code that does other things on the server while it's running, like looks through directories and finds other student directories and changes things or something like that. Right. And so you'll have to do careful jailing of the processes. It's totally doable in Linux, like not a problem. Like you can you can set a route that deep in a directory tree and it can't break up into the directory tree out of that route and that that works great. You'll have to do strong verification of whatever is the code as well to make sure that it's not gonna be subject to buffer overflows or something like that cause the C code could do that. Yeah I mean those are kind of the main things. And isolating the students submissions and responses from other students is kind of the main concern you've got. And I think we've taken care of that for the PLC server.

Mike: So basically you can't see other people's work.

Sid: That should be the goal. Or their grades, or you should not be able to submit into other people's repositories, or it's just isolation from student to student, especially if there's automation to do population of the repos where like instructor logs in, clicks the thing, and says populate my student repos. There have to be a good isolation between those two as well because the instructor repos don't just have assignments then they have solutions and things like that.

Josh: The way to combat mode of these problems is we were actually planning on using Docker. And since that's its own instance we would only have the professors test code

in the docker file and we'd run the students code off of the professor's test code in that environment and then spit out.

Sid: So you'd spin up like a doctor instance, put some stuff on it run the code.

Mike: Yeah. Right.

Sid: That seems cool. Is docker overkill for for …

Josh: this kind of project?

Sid: Yeah yeah.

Zach: We were thinking like yeah it'd probably be overkill for like the 132 classes but we're also trying to do things for like 230 and maybe even like senior or uh, software design.

Josh: True, it's for any language.

Sid: Okay yeah. Well it's definitely more extensible. Yeah. Docker would be a good isolation tool for the runtime. Yeah totally. It would be much more flexible than a lot of other things too that are just kind of one offs. Right. Like if there's a new test framework you could tell the instance I like to import a bunch of stuff for them. Cool. Right now what, the tool that I use in 132 does, the automatic check to see if it works tool, it just runs a compile and execute process. And it puts a loader around the students code which kind of jails it a little bit by setting environment variables and doing a bunch of other things. It also it does an include for the student's file so it wraps a bunch of C code around it. That is kind of hacky and fragile. We still need some sort of bootstrapping stuff for the C but it doesn't have to be, you don't have to worry about security as much as long as yeah as it's in a docker instance it should be pretty good.

Josh: Yeah. Those are the questions I had.

Mike: Yeah I mean you pretty much nailed everything.

Josh: All right. Actually no I'm just kidding. I do have more questions. So far on the instructor side of both Rosebuild and moodle, what would you like to see other than like manually putting in student repos and getting a table of grades.

Zach: Is there any is there anything on there that works but you'd like to see it work better.

Josh: Because that's something we haven't seen as students, the instructors side of everything.

Zach: No idea what's on there at all.

Sid: I almost never use moodle or anything. I mean it's a glorified gradebook for me. Everything that I use for distributing assignments obtaining solutions and grading them is a shell script or version control repository that can get rebuilt. I mean most of what I do right now is pretty manual. Yeah. And I have some crappy automation. I'm happy to show you the crappy automation but it's pretty crappy.

Josh: I kind of want to see it if that's okay.

Sid: When you're done with Q&A I'm happy to show you. I think that'll work or at least, do you guys have any more questions?

Zach: I don't have any.

Mike: Nope.

Josh: Okay.

Interview with Nate Chenette:

- As a professor for 230, what do you think are some advantages to using SVN and what things would you change if you could?
- What things would you like to see from the professor side of our system?
- What do you think a system like this needs for a student to be most successful while using it?
    - SVN is good because we have a central, master type of repo that makes it easy for students to commit and checkout work. But SVN doesn't necessarily help the professor with grading other than running the unit tests that are predefined. He has to run a script that pulls all of the students submissions and renames them to one eclipse environment, then he runs the unit tests on each students code to see what they got and then he looks at the code they wrote individually so that he can assign partial credit and grade for efficiency. Also, svn is not the greatest for handling group assignments because all of the repos that are created are done by script, so when students need to have access to multiple repos it can get tricky.
    - A streamline way of running the unit tests, and assigning grades to each students submission based on them. But then also making it easy for professors to view the submitted code and make changes as needed.
    - Making it streamline, and making it less public than GitHub so that there is less of a chance for cheating. 230 is actually moving to start using git so if you had something that could do what we need it to do neatly then it

would be easier to want to use it rather than GitHub or SVN. He isn't sure whether or not we will be able to make it much more stream-line than running scripts to put all the submissions in one eclipse environment. Because ROSEBuild has tried but there isn't much room for improvement (it seems like) in that respect.

- ACM code of ethics changed its stance on intellectual property as they now encourage people to not be so protective over their work. They treat it like the benefit to society is more important. That's tough for an academic standing because you want students to understand that idea but you also still need good engineers and to do that they have to learn the ropes themselves and that is why SVN and GIT are being used over GITHub because GITHub is a little too public for academia in my opinion.
- SVN is script heavy to be able to do the things they want it to do
- Handling group assignments is the biggest issue that could be fixed, make sure that none of the other areas get worse.