# Compile.io

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

# Table of Contents:

[Compile.io](#_op001ivgb8nn)

[Table of Contents:](#_kj2glsme2lve)

[Introduction](#_e1e7lpcynm8z)

[Problem Description](#_8056tf0dp9m)

[Solution Description](#_vzesho7i0p4n)

[Back-end Discussion](#_k89a2us112fp)

[Front-end Discussion](#_whp6ull1go05)

[Key Challenges](#_5gpbww8ovflj)

[Design Analysis](#_68qgzwvozenq)

# 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.

# IntroductionThis 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

1. Currently there is no centralized system at Rose-Hulman Institute of Technology for students to upload code from a class that can be tested using a professor’s test cases.

# Features

## System

* 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 student hacking
* Simple front end to view the results of the uploaded code.
* Login using RoseFire
* Software will prevent unauthorized students from uploading code
* Software will have different classes that correspond to different professors
* Software will contain a roster for each class

## Student

* Student is able to upload code (zip file)
* Show code results from test cases
* Only students enrolled in class can upload code to that respective class
* Software will contain a page that shows the student the results of past test runs
* Software will allow a student to login through rosefire

## Professor

* 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 between different classes
* Professors can edit assignments
* Software will allow professors to view various classes they teach
* Software will allow professors to change and remove tests for their classes

# 

# 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

*Sid Stamm, Professor of Computer Science at Rose-Hulman Institute of Technology*

*Steve Chenoweth, Professor of Computer Science at Rose-Hulman Institute of Technology*