CSGrader Programmer's Manual

Zachary Nelson, Brandon Mora, Ryan Hurst

1. Vision Statement

The goal of this software is to expand coverage for the CSGrader by including testing and grading for RLang and Typescript as well as audit logging for CSGrader features. The goal of this document is to include the necessary steps to configure grading for RLang and Typescript documents - and to highlight audit logging features.

2. Introduction

CSGrader is a computer science assignment grading program that tests projects and provides feedback to students, instructors and faculty. When we joined the project, CSGrader had already been in development for years. Our team was tasked with expanding the project by adding Typescript and R language support and also audit logging capabilities for other CSGrader features.

CSGrader works by pulling coding assignments and their tests from version control sites like GitHub to a container. The container is populated with a specific testing platform chosen for compatibility with the CSGrader operations instance and the target language. The testing platform runs the tests on the assignment. After parsing, the assignment is tested for complexity and those results are returned. Both sets of results are sent through the CSGrader interface.

3. Component Overview

1. GraderStruct
   The graderStruct contains all functions for grading operations.
2. Language parser
   Language parser that finds the complexity of a program.
3. Typescript parser
   Fetches/ compiles Typescript assignments and prepares the assignment for grading.
4. RLang parser
   Fetches/ compiles RLang assignments and prepares the assignment for grading.
5. Typescript Grader
   Executes included tests for a typescript assignment and prepares the results for further analysis or returns the results.
6. RLang Grader
   Similar functionality as the Typescript Grader, grades RLang assignment by the number of project tests completed, only returns the results at this point.

4. Tool Overview

1. Docker Containers

Containers that are used as testing platforms. Each container is installed with needed testing frameworks, config files, and project files for testing.

2. AWS Instances

   Interface for running Docker containers as needed.

3. Jest

   Typescript testing framework; provides additional tools for Typescript testing.

4. Testthat

   R testing framework; the standard testing framework for R.

5. Prisma

   Object-relational Mapping with database connectivity; provides access to Audit Logs created during CSGrader operation.

5. Project Repository

https://github.com/Primeeight/P445-F23-CSG-Auto-Grader

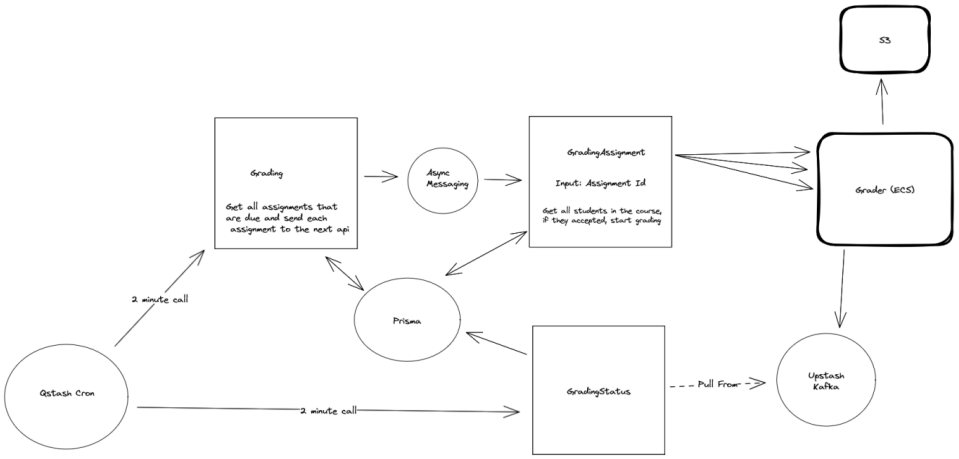
```
TeacherTestResults: Coverage:
        Line: 75%
        Function: N/A
        Branch: 50%
        Instructions: N/A
        Complexity: N/A
Total tests: 2  Passed: 2       Failed: 0
        TestLocation:  adds 1 + 2 to equal 3    Total: 1        Passed
                PASSED  Test:  adds 1 + 2 to equal 3
        TestLocation:  the absolute value of -2 is 2    Total: 1
                PASSED  Test:  the absolute value of -2 is 2

ComplexityResults:
LinesOfCodeCreated: 6257
LinesOfCodeDeleted: 947
GitInformation: [
```

userId	orgId	ObjectId	eventName	description	timeStamp
cluv900770…	uhihep2e785…	a0bx258bkx0puy…	unpublished	User rhurst2001@icloud.com unpublished assi…	2024-04-11T13:28:01.802Z
cluv900770…	uhihep2e785…	a0bx258bkx0puy…	edited	User rhurst2001@icloud.com edited assignmen…	2024-04-11T13:28:08.886Z
cluv900770…	uhihep2e785…	a0bx258bkx0puy…	published	User rhurst2001@icloud.com published assign…	2024-04-11T13:28:19.830Z

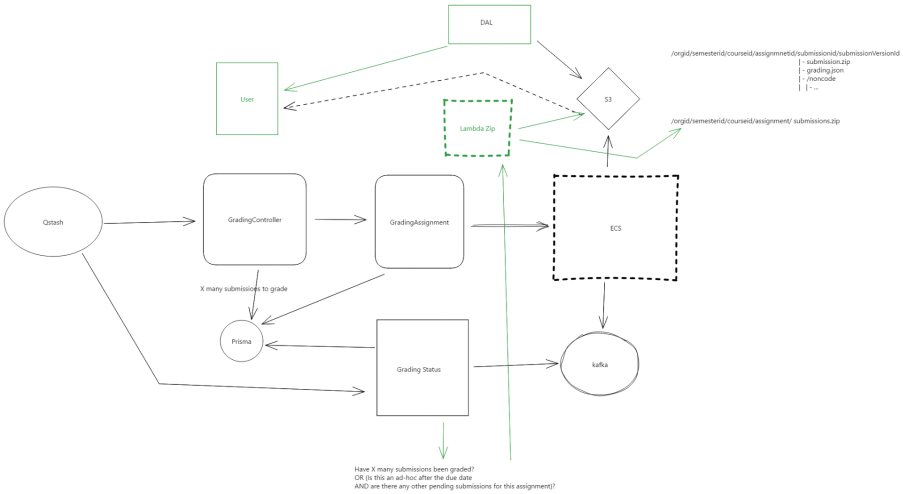
c. Documentation

- Grading Process:

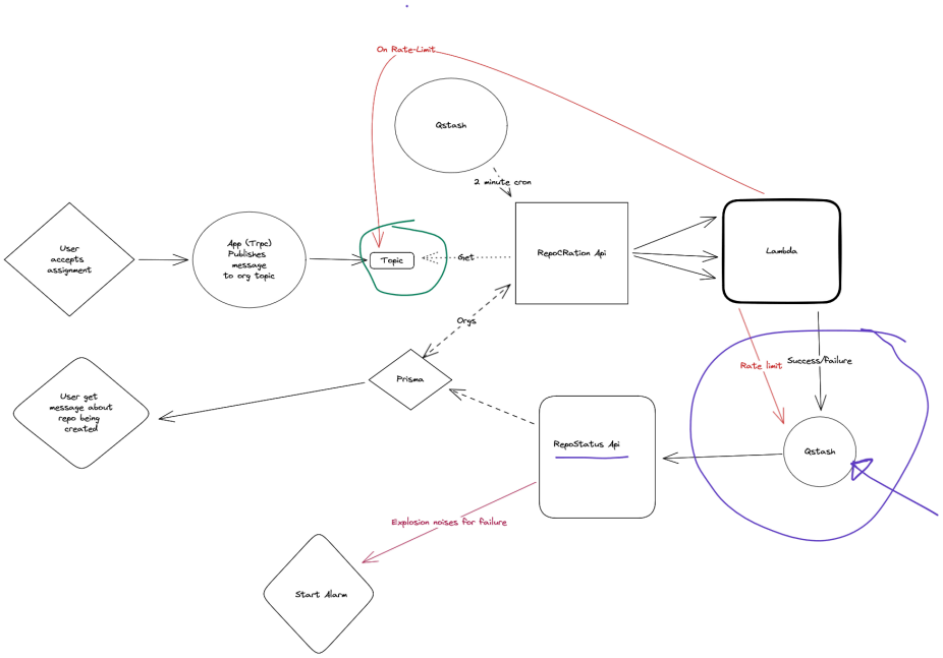
A.



B.



- Repo Creation:



C.

Additional documentation can be found in the repository.

6. Installation

Installation consists of installing the latest version of CSGrader. The latest version of CSGrader has the latest version of Typescript grading, RLang grading, and audit logging. After the initial installation, the testing frameworks Testthat (https://testthat.r-lib.org/) and Jest (https://jestjs.io/docs/getting-started) will need to be installed. After the project and testing frameworks have been configured, the address for repositories to be graded will need to be added to the project configuration.

7. New Platform Installation

Migrating to a new platform is relatively simple. Assuming a preconfigured container (such as through docker) an instance of the CSGrader can be spun up that contains all the necessary prerequisites. Installing Testthat, Jest, or another testing framework would not be necessary. Project configuration, such as selecting a repository to pull assignments from, may still require configuration.

8. Further Development Statement

Zachary Nelson:

If I were to continue to develop the CSGrader project for another year, It'd be my priority to add features for supported languages, new supported languages, and general features as needed. Currently the RLang grader within CSGrader supports reading and parsing assignments created in the R language. I would add more tests to ensure that the RLang grader is able to parse and grade projects.

The goal of the CSGrader has always been to provide support for grading in multiple languages; providing immediate feedback for stakeholders (students, instructors, and faculty). Support for additional languages such as NASM, or F-Sharp has been part of the goal for a long time. Support for these languages would allow students to be better equipped for these classes.

Brandon Mora:

Given a better understanding of the CS Grader environment, I would like to add more language compatibility. Further, I would like to adjust the language parser in order to streamline the process and make it more consistent across languages. Currently, some use an ANTLR generated parser and some don't. Across both the graderStruct and the language parser, there is a need for uniformity and documentation. This would greatly improve the ease with which language compatibility is added by new programmers.

Ryan Hurst

There are several enterprise features that I would like to add in the future. A full settings page that gives the user the ability to change all aspects of their profile such as updating their profile picture or linking a new GitHub account. Additionally, I would like to rework the audit logging functionality to operate without using Prisma middleware, as it does not fit well with the way operations are handled in the grader and will make adding new logs in the future difficult.