

A RESTFUL FRAMEWORK FOR WRITING, RUNNING, AND EVALUATING
CODE IN MULTIPLE ACADEMIC SETTINGS

by

Christopher Ban

A PROJECT PROPOSAL

Presented to the Faculty of
The School of Computing at the Southern Adventist University
In Partial Fulfilment of Requirements
For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Rick Halterman

Collegedale, Tennessee

September, 2015

A RESTFUL FRAMEWORK FOR WRITING, RUNNING, AND EVALUATING
CODE IN MULTIPLE ACADEMIC SETTINGS

Christopher Ban, M.S.

Southern Adventist University, 2015

Adviser: Rick Halterman

In academia, students and professors want a well-structured and implemented framework for writing and running code in both testing and learning environments. The current limitations of the paper and pencil medium have led to the use and/or creation of many different online grading systems. However, no known system provides all of the essential features our client is interested in. Our proposed system, developed in conjunction with Doctor Halterman, would offer the ability to build modules from flat files, allow code to be compiled and run in the browser, provide users with immediate feedback, support multiple languages, and offer a module designed specifically for an examination environment.

Contents

Contents	v
List of Figures	vii
1 Introduction	1
2 Background	5
2.0.1 Learning Management Systems	6
2.0.1.1 Analysis: Moodle	7
2.0.1.2 Analysis: Edmodo	9
2.0.1.3 Analysis: The Virtual Programming Lab	10
2.0.1.4 Analysis: Javabrat	11
2.0.2 Automated Programming Evaluation Systems	12
2.0.2.1 Analysis: Web-based Center for Automated Testing	14
2.0.2.2 Analysis: Turing’s Craft CodeLabs	15
2.0.2.3 Analysis: CourseMarker	16
3 Proposal	19
3.1 Quiz and testing module	20
3.2 Learning module	25

3.3	Server-side code execution	25
3.4	Testing security and integrity	26
3.5	Task Delineation	28
4	Evaluation Plan	29
4.1	Objective	29
4.2	Subjective	30
5	Conclusion	33
	Bibliography	35

List of Figures

2.1	A simple Moodle quiz module.	8
2.2	A simple VPL module assignment [1].	11
2.3	A basic Javabrat assignment example [2].	13
2.4	A simple Web-CAT module assignment [3].	14
3.1	A test module example generated from a well-formed JSON object. . .	23

Chapter 1

Introduction

In academia, students and professors want a well-structured and implemented framework for writing and running code in both testing and learning environments. In a test or quiz setting, students prefer to type and have the ability to execute code. In an instructional medium such as a handout or online textbook, students would prefer to try out code snippets right in the browser without the need to install or switch to another program. Some on the other hand would like the ability to build exams and quizzes via structured text files without multiple, repetitive steps that are time consuming. Finally, automated and near-immediate grading and feedback remains a huge advantage to both parties. Our client, Doctor Halterman of Southern Adventist University, has recognized this call for a framework that satisfy these needs.

Programming tests and quizzes today often are given through the paper and pencil medium. This medium is not desirable as it does not allow for immediate feedback, which can frustrate both teachers and students. Students must wait for results and teachers must invest large amounts of time in grading students' answers. Another problem that arises due to this medium is the inability to

type and format code. For students in a high-pressure environment such as a quiz or test there is not always time to neatly write code. Erasing or revising written code can further reduce readability. As a result, teachers and students are forced to read and work with messy answers. Finally, this medium does not allow for the possibility of compiling and running code. The ability to execute code allows students to fix small mistakes such as syntax errors and other problems that do not necessarily have any implication on the testing content.

Online textbooks and other similar learning environments suffer from similar problems (i.e., no immediate feedback, and readability issues), with the addition of the need to switch to a different application completely. Using these media require students to leave the browser, have a compiler or IDE installed, and copy or write code snippets in order to try concepts out on their own.

These issues have led to the use and/or creation of two separate products: Learning Management Systems (LMS), and automated evaluation systems. An LMS (e.g., Moodle, and Edmodo) can provide a web interface for building instructional articles, exams, and quizzes and provide students with a common interface. An LMS can also serve as a secure environment for testing and quizzes, and offer features such as multiple attempts, instant feedback, and question randomization. LMSs also have the capability to integrate add-ons which can provide the ability to compile and run code written by students, such as Moodle's plug-in feature [4].

Modifying or building quizzes and exams via an LMS requires multiple, repetitive steps that are more time consuming than editing a simple flat file. In addition, current LMS do not give statistics on test and quiz properties, such as time spent per question. Existing add-ons for an LMS such as Moodle do not address this issue, and the compiler features do not contain some

of the requested properties mentioned above (e.g., immediate evaluation and feedback).

As for automatic evaluation systems, students can type and execute code, run tests, and receive immediate feedback. However, widely used solutions simply support assignments and code challenges, excluding examination environments and general practice environments.

Therefore, based on our client's interests stated above, we believe no application exists which allows users to quickly build learning modules, exams, or quizzes via structured text files for students to use, that also give them the ability to write and run code and receive immediate feedback on submitted work.

This project addresses these issues by creating a framework which provides a medium with the following goals:

- Allow professors to quickly build activities via simple flat files
- Allow students to be able to type code in an environment that provides syntax highlighting
- Allow students to compile and run code in a controlled environment
- Allow for the ability to build activities specifically for an examination environment
- Allow students and professors to get immediate statistics and feedback
- Allow for the support of multiple programming languages

Given this framework, professors can quickly and easily create quizzes, tests, and other learning environments, and students can edit and compile code and receive immediate feedback. We believe this will help professors limit time

spent on building and grading tests and quizzes, and help students focus more on actual test content, rather than on formatting and cleaning up answers.

For the remainder of our paper, we first cover the background in Chapter 2. Chapter 3 and Chapter 4 will cover our proposal and the evaluation respectively. Finally, Chapter 5 presents our conclusion on the research found and our future work on the Master's Project.

Chapter 2

Background

Currently there are virtually no systems in widespread use that provide automated feedback tailored for programming. However there is instead many different niche systems in use. The most popular systems in use in academics today are Learning Management Systems (LMS). These products provide teachers and students with a familiar medium in which to access assignments, grades, and even tests and quizzes. These systems also allow for modules to be added in order to provide new features developed by third parties. In addition to these systems there exists more focused projects that attempt to provide similar services, specifically tailored toward programming in education. In this section we explore the abilities of popular LMSs, and open source programming evaluation systems which attempt to provide similar functionality.

Our background analysis is broken into two different sections, Learning Management Systems in section 2.0.1 and automated programming evaluation systems in section 2.0.2, taking into consideration their strengths and weaknesses as follows:

- Test and quiz building

LMS	Users
Moodle	71,054,217
Edmodo	49,000,000
SuccessFactors	28,000,000
Blackboard	20,000,000
TOPYX	20,000,000
SkillSoft	19,000,000
Instructure	18,000,000
Desire2Learn	15,000,000
Cornerstone	12,400,000
Schoology	8,500,000
Meridian Knowledge Solutions	8,500,000
eLogic Learning	5,000,000
Latitude Learning	3,500,000
eFront	3,500,000
Docebo	2,100,000
Litmos	2,000,000
TalentLMS	600,000
DigitalChalk	425,000
Collaborize Classroom	350,000
Educadium	75,000

Table 2.1: The top 20 most popular LMS packages, ranked in order of total users as of 6/3/15 [5].

- Code formatting and compiling ability
- Answer evaluation and feedback

These three topics will provide a perspective on how the system functions and how well it addresses our goals.

2.0.1 Learning Management Systems

The following subsection will evaluate the LMS based systems selected for analysis. Table 2.1 shows the most popular LMS software packages [5], from which we select the two most popular LMSs (Moodle [6] and Edmodo [7]) for anal-

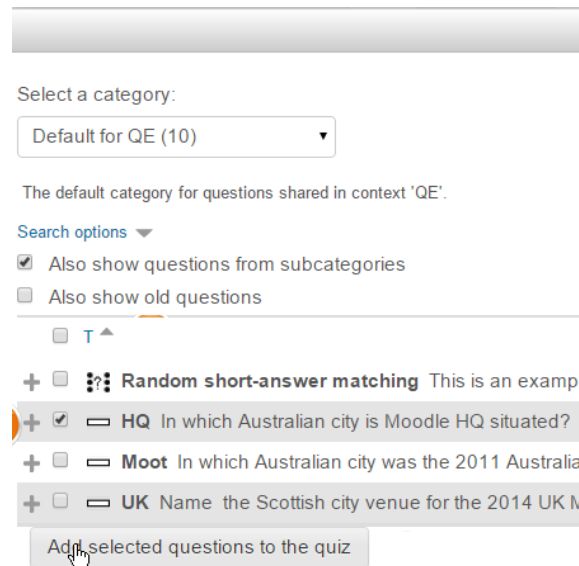
ysis along with two LMS modules for Moodle called the Virtual Programming Lab [1], and Javabrat [2].

2.0.1.1 Analysis: Moodle

Moodle [6] is a free, open-source LMS for producing modular internet-based courses that support a modern social constructionist pedagogy. Because it is open-source, Moodle has a large community contributing many modules that extend its feature set.

Test and quiz building Moodle has a quiz activity module which allows the teacher to design and build quizzes consisting of a large variety of question types, including multiple choice, true-false, and short answer questions. To make a new quiz/test assignment in Moodle, the teacher may either import a quiz from a range of different flat file formats [8], or manually add a quiz module to the course. When creating a quiz manually, the teacher must first create the quiz activity. To add a new question to the quiz's question bank, the teacher must click 'Add' and then '+ a new question'. From the next screen, the teacher must choose the specific question type. Once the question form is filled in with the question and any necessary options are added the question can be saved and added to a question bank shown in figure 2.1. The teacher must then add questions to the quiz from the question bank.

With all of the steps in manually building quizzes, teachers must invest time in adding items to question banks on top of formulating the actual question content. While importing questions from flat files is useful in avoiding this overhead, the type of questions available to be asked are still limited by Moodle's standard questions types [9].



Select a category:

Default for QE (10) ▼

The default category for questions shared in context 'QE'.

Search options ▼

☒ Also show questions from subcategories

☐ Also show old questions

☐ T ▲

+ ☐ **Random short-answer matching** This is an examp

+ ☒ **HQ** In which Australian city is Moodle HQ situated?

+ ☐ **Moot** In which Australian city was the 2011 Australia

+ ☐ **UK** Name the Scottish city venue for the 2014 UK M

Add selected questions to the quiz

Figure 2.1: A simple Moodle quiz module.

Code formatting and compiling Moodle quiz activity modules can accept multiple choice and text input. These methods allow students to type code, which is an improvement on the pencil and paper medium. However, these input methods do not allow students to properly format code, or show syntax highlighting. Similarly, students do not have the ability to execute code and see results produced by the compiler which limits the types of questions that can be presented.

Answer evaluation and feedback Moodle quiz activity modules provide the ability to evaluate and grade answers immediately and give students feedback, which is another improvement on the pencil and paper medium. Unfortunately, due to question limitations mentioned above, there is no ability to run test cases on student defined code and grade submissions automatically.

2.0.1.2 Analysis: Edmodo

Edmodo [7] is a free social learning platform for schools. It provides teachers and students with a secure way to share classroom materials, and access homework, grades.

Test and quiz building Edmodo has a quiz activity allows teachers to create online quizzes for students and receive instant feedback on grade results. Unlike Moodle, all Quizzes and Quiz questions must be created within Edmodo and cannot be imported. The steps to do so are comparable to Moodle's, and incurs similar overhead. To create a quiz in Edmodo, a teacher must create a quiz activity, and manually select a question type. Once the question form is filled in with all necessary information the question can be saved and added to a question bank.

Similarly with Moodle, Edmodo is limited by the building overhead and the types of questions available to be asked (Multiple choice, True False, Short Answer, and Fill in the blank).

Code formatting and compiling Again, very similar to Moodle, Edmodo quiz activities can accept multiple choice and text input. These methods allow students to type code, however these input methods do not allow students to properly format code, or show syntax highlighting. To add to this, students do not have the ability to execute code and see results produced by the compiler which limits the types of questions that can be presented.

Answer evaluation and feedback Edmodo quiz activities provide the ability to get results feedback immediately, which is another improvement on the pen-

cil and paper medium. Due to question limitations mentioned above however, there is no ability to run test cases on student's code and grade submissions automatically in that manner.

2.0.1.3 Analysis: The Virtual Programming Lab

The Virtual Programming Lab (VPL) [1] is an activity module for Moodle that manages programming assignments and whose main features are:

- Students can edit program source code in the browser
- Students can run programs interactively in the browser
- Students and teachers can run tests to review the programs

Test and quiz building This activity module for Moodle is unfortunately tailored for programming assignments. Because of this, it is not possible to create an assignment in a test or quiz format.

Code formatting and compiling VPL allows files to be edited from the browser using the code editor component, as shown in figure 2.2. It also has the ability to supply initial skeleton code. This added functionality allows students to run and evaluate programs without requiring them to install compilers or IDEs. The code editor component provides formatting as well as syntax highlighting, which is useful for students both reading and writing code.

Answer evaluation and feedback VPL allows for scripts to be set to evaluate every submission, which enables students to receive feedback on their code based on test cases. However, this takes more time to set up than simply defin-

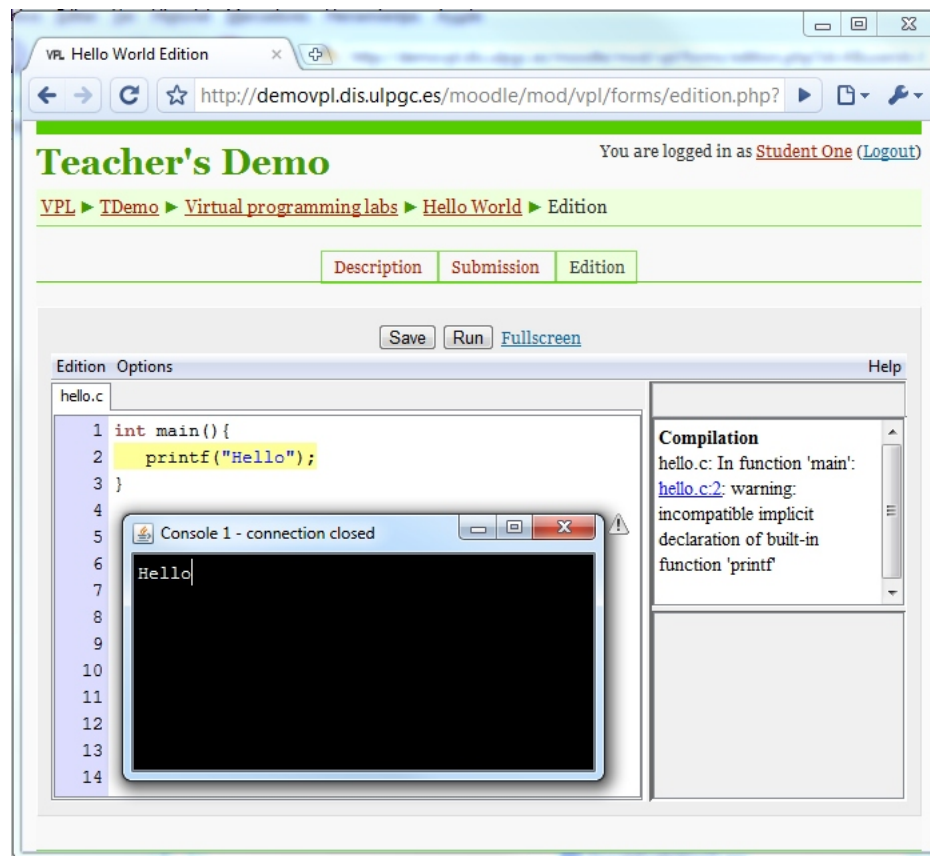


Figure 2.2: A simple VPL module assignment [1].

ing test cases and is geared more towards assignments instead of quizzes and tests.

2.0.1.4 Analysis: Javabrat

Javabrat [2] is a web-based grader useful for students learning Java and Scala languages. It is primarily in the form of a Moodle plugin that facilitates instructors to grade Java assignments.

- Students can edit program source code in the browser
- Students and teachers can add problems to be solved

- Students and teachers can run tests to review the programs and automatically grade them

Test and quiz building Similar to VPL, Javabrat is tailored towards programming assignments. Because of this, it is not possible to create an assignment in a test or quiz format.

Code formatting and compiling Javabrat allows for code to be typed directly into the browser, but does not support syntax highlighting, as shown in figure 2.3. Similar to VPL, Javabrat has the ability to provide an initial skeleton code for students to build on. This setup allows students to type out code and evaluate results without having to leave the browser.

Answer evaluation and feedback With Javabrat, program results are evaluated immediately. When a student completes a solution, it can be evaluated in the browser by running pre-defined test cases. Once the evaluation is done, Javabrat sends back an HTML report file generated by the back-end as shown in the lower part of figure 2.3.

2.0.2 Automated Programming Evaluation Systems

The following subsection will evaluate the automated programming evaluation systems selected for analysis. From our research we select three popular automated programming evaluation systems: Web-CAT [10], Turing's Craft [11], and CourseMarker [12].

1

Marks: 10

PerimeterTester

Write a PerimeterTester program that constructs a Rectangle object and then computes and prints its perimeter. Use the getWidth and getHeight methods. Also print the expected answer.

PerimeterTester.java

```

import java.awt.Rectangle;

/**
 * Constructs a Rectangle object and then computes and prints its perimeter.
 */
public class PerimeterTester
{
    public static void main(String[] args)
    {
        Rectangle r1 = new Rectangle(5, 10, 20, 30);
        double perimeter = 2 * r1.getWidth() + 2 * r1.getHeight();
    }
}

```

Check

Click Check to test your program and generate a report for the grader. You can generate reports as often as you like. Simply fix your code and click Check again.

Evaluation Summary

compile	pass
tester	pass
Total:	100%

Compiling Main Program

pass

Compiling 1 source file to /opt/glassfish-data/webLabRat/problem_repository/s

Testing Main Program

pass

Perimeter: 100.0
Expected: 100

Student Files

IMPORTANT:

When you are done with all problems, click "Submit all and finish" before the deadline. Otherwise, your homework will not be graded.

If you want to come back later to finish, click "Save without submitting"

Save without submitting

Submit all and finish

Figure 2.3: A basic Javabrat assignment example [2].

The screenshot shows the 'New Submission' page for a student named Stephen Edwards. The page is titled 'New Submission' and 'Upload Your File(s)'. It displays a list of steps: Step 1 (Pick the course), Step 2 (Pick the assignment), Step 3 (Upload your file(s)), Step 4 (Confirm your submission), and Step 5 (View your results). The current step is Step 3. The assignment is 'CS 1(2) Project J4: Java calculator (full static checks)'. A table shows the student's previous submissions:

No.	Time	Score
3	08/16/06 10:06PM	44.0
2	08/16/06 05:32PM	69.0
1	08/16/06 05:26PM	28.1

Below the table, instructions state: 'Upload a single zip, jar, tar, or tgz file containing your Java source files, including your JUnit test cases. Test cases must have class names ending in "Test" or "Tests" (case-insensitive). So far, you have uploaded the file **Java-calculator.zip** (1.4kb) for submission. You can replace this with a different file, or choose to continue.' A 'Choose the file to upload:' section includes a text input field, a 'Browse...' button, and navigation buttons: '< Back', 'Next >', and 'Cancel'.

Figure 2.4: A simple Web-CAT module assignment [3].

2.0.2.1 Analysis: Web-based Center for Automated Testing

The Web-based Center for Automated Testing (Web-CAT) [10] is a flexible automated grading system designed to process computer programming assignments. Its main features are:

- Students can edit program source code in the browser
- Provide immediate feedback based on test-cases and code coverage
- Enable flexibility and extensibility by way of a plug-in based architecture
- Provide a secure and portable product

Test and quiz building Similar to VPL, Web-CAT is tailored for programming assignments. Because of this, it is not possible to create an assignment in a test or quiz format.

Code formatting and compiling Web-CAT does not allow files to be edited from the browser. It also has the ability to supply initial skeleton code. Without this functionality, students must use their own compiler or IDE. In Web-CAT, students are required to upload their source code to the correct assignment as shown in figure 2.4.

Answer evaluation and feedback Web-CAT uses a composite scoring system, which enables students to receive feedback on their code based on their code correctness, test completeness, and test validity. These three measures, taken as percentages, are then used to form a composite score [13]. This formula ensures that no aspect of the approach can be ignored, or the student's score will suffer. Code correctness is based solely on student written tests, unlike the other two scores. For test completeness, instructors must choose what level of coverage should be used for testing completeness, and for test validity, instructors must set specific reference tests to ensure the student's tests are accurate.

2.0.2.2 Analysis: Turing's Craft CodeLabs

Turing's Craft CodeLab [11] is the commercial equivalent to WebToTeach [14], developed to provide fully interactive, hands-on practice environments for learning computer programming. The supported topics start with the imperative programming core of Python, C, C++, and Java and then go on to address procedural and object-oriented programming. The exercises are targeted at a typical CS1 syllabus.

Its main features are:

- Students can edit and run source code in the browser

- Provide immediate feedback based on test-cases in the browser
- Provide teachers with a selection of exercises ranging from short and focused to more complicated problems intended to be used as teaching aids

Test and quiz building Because Turing's Craft is targetted towards programming assignments and challenges, it does not support quiz or exam formats.

Code formatting and compiling The Turing's Craft engine is hosted and supported by Turing's Craft and the CodeLab runs in the browser, so students are able to write and run code directly in the browser. Unfortunately, it does not support syntax highlighting.

Answer evaluation and feedback Turing's Craft CodeLab provides student with immediate feedback on submitted solutions. With each submission, the instructor's roster is automatically updated regarding solution correctness. As it is intended to be used as a teaching aid, Turing's Craft research found that CodeLab works best when required as 5-10

2.0.2.3 Analysis: CourseMarker

CourseMarker [12] was developed for the assessment of java programming skills of CS students. CourseMarker has been tailored to the students need by teachers and students suggestions, building off of an older automated assessment tool called "Ceilidh" [15]. Its main feature is to allow students and teachers to run tests and receive in-depth metrics and assessments immediately. CourseMarker must be installed instead of accessing it through a browser.

	Exam environment	Code formatting	Immediate feedback	Multi-lang support
Moodle	X		X	
Edmodo	X		X	
VPL		X	X	X
Javabrat			X	
Web-CAT			X	X
Turing's Craft			X	X
CourseMarker			X	X

Table 2.2: A visualization of how many of this project's goals are met by each solution evaluated above.

Test and quiz building Similar to VPL and Web-CAT, CourseMarker is tailored for programming assignments. Because of this, it is not possible to create an assignment in a test or quiz format.

Code formatting and compiling CourseMarker does not allow files to be edited from the client, but it does have the ability to supply initial skeleton project files. Without this functionality, students must use their own compiler or IDE.

Answer evaluation and feedback CourseMarker provides grading results immediately once code solutions are submitted. Students receive feedback on typography, scalability, and test cases. With typography, CourseMarker checks the program layout, indentation, and usage of comments. Regarding scalability, CourseMarker attempts to use different sized datasets when possible. Lastly, CourseMarker evaluates solutions by checking for exercise dependent features defined by the teacher.

In summary, as seen in table 2.2, each of the popular solutions reviewed met different parts of our client's overall goals but none met all of them. While a few such as Web-CAT came close, many of the widely used solutions do not offer everything this project is attempting to accomplish.

Chapter 3

Proposal

This proposed solution creates a RESTful web service that provides the following services:

- A quiz/test module generated from a quick and easy flat file which defines questions, options, scores, etc. in a medium that allows for code formatting and execution
- A learning module which provides an insertable interface to compile and execute code generated from a flat file which defines the problem description, skeleton code, and test-cases

Access to a server and its filesystem are the only requirements to become functional, along with the Javascript framework Node.js which provides the tools needed to host this RESTful service and handle requests. This service will allow an experienced user to quickly build exams and quizzes via flat text files, and quickly modify an existing document (exam or quiz) via a simple text editor. By using Javascript to make a simple AJAX call to our RESTful service, an instructor can populate anything from an empty webpage to an existing site

such as an online text-book with either a full quiz/test or learning module to compile and execute code. The service guarantees the integrity and security of quizzes and tests. As with all testing, a proctor can provide an additional layer of security. The service's security is covered in more detail in section 3.4.

3.1 Quiz and testing module

A quiz and test module allows students to take a programming test or quiz on the computer with the ability to type and compile code and get instant grading feedback. It is generated by a simple flat file created by a teacher, and is expected to follow the Javascript Object Notation (JSON) format. If a teacher would like to add line breaks or specific formatting to specific fields, they can use '«' and '»' to denote text that needs to be parsed (e.g., «formatted text»).

The flat data file is uploaded to the server, and if malformed by the custom formatting mentioned above, parsed to form a JSON object which is then stored in the specified class folder. This JSON object has the following structure:

```
0 {  
1  "0": {  
2      "questionType": "",  
3      "language": "",  
4      "problem": "",  
5      "skeleton": "",  
6      "input": [""],  
7      "output": [""],  
8      "points": [""],
```

```
9         "difficulty": ""
10     },
11     "prop": {
12         "test_length": ""
13     }
```

Data-structure 3.1: JSON datafile object structure

The test is divided into two main sections: Multiple choice first, then coding questions. Students are allowed to view the whole test, but once they begin compiling, their answers are recorded and frozen (more on this later in section 3.4). In regards to programming problems, multiple choice questions can many times be solved simply by having access to a compiler. An example of this would be a question asking what the output would be after running a code snippet. Because of this, code compilation and execution must be prohibited until after the student completes the multiple choice section.

Within this structure, "questionType" defines whether the question is a coding question (where students are expected to write code), or a multiple choice question (where students are expected to answer questions; e.g., what result is produced by a given code snippet). Each field serves the same function for either type of question, excluding the "input" and "output" fields. The "language" field defines the programming language used, "problem" defines the question, "skeleton" defines the initial skeleton code, "points" provides the point value, and "difficulty" defines a difficulty level used to produce suggested completion times.

For coding questions, the "input" field is an array that holds different test-cases. This input value can represent any input a keyboard can provide in a

command-line setting. Each element in the "input" array is a different test case that will be run against student's submissions. The "output" field represents the expected output produced after running each test case from the "input" field.

Multiple choice questions on the other hand, use the "input" field a little differently. With this question type, the "problem" field describes the overall instructions and the "skeleton" field describes the corresponding code snippet, identical to coding questions (except it is non-editable). However, "input" is again an array, but now it is a multi-dimensional array, with each internal array holding a sub-question and an array of options from which students can choose. The "output" field dictates the indices of the correct options.

The structure also contains a "prop" field which can hold information about the activity. For example, by default it is used to define the property "test_length", which specifies the time length of the test.

On request, this data file is parsed and used to generate content. Once the server registers an HTTP POST to the "/getModule" URL, it evaluates the requested datafile described above and generates the necessary content and scripts needed to serve the specified test or quiz by walking through each property in the JSON datafile and inserting those values into templated code.

There are two variables that get packaged into the response object sent after a successful "/getModule" call. The first is a response_html variable which contains the html to be inserted. The server holds different element templates in variables, such as the problem statement template and the I/O template which are chunks of HTML code with placeholders. As the server walks through the JSON data object, these templates are appended to the aforementioned response_html variable with their placeholders replaced with indexing and datafile specified information in order to formulate the full module which

is then inserted into the Document Object Model (DOM) element specified by the AJAX call (generally the HTML body element for tests and quizzes). The second variable is a script called `response_script` and is evaluated on the client once the initial AJAX call returns successfully. This script holds all the module logic used for everything from object initialization to navigation and exam submission.

Practice Exam

Enter or modify the code below and press 'Compile' to execute and view results.

43:43
Submit Exam

Complete the following function that counts the even numbers in an integer vector. For example, if vector `vec` contains the elements 3, 5, 2, -1, and 2, the call

`count_evens(vec)`

would evaluate to 2, since `vec` contains two even integers. The function returns zero if the vector is empty. The function does not affect the contents of the vector. A simple test main function is provided for your convenience.

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int count_evens(const vector<int>& v){
6
7     return -1; //Replace with working code
8
9 }
10
11 int main() { // Make vector
12     cout << "Enter vector size: ";
13     cin >> size;
14     vector<int> v(size);
15     // Populate the vector
16     for (int i = 0; i < size; i++) {
17         cout << "Enter element " << i << ": ";
18         cin >> v[i];
19     }
20
21     // Test the function
22     cout << count_evens(v);
23 }
24

```

Inputs will be read in order for every line read performed.

New Input: +

Current Inputs: ▼ ▲

Compile

Suggested time: 20:40
Commit

<< Prev

1 2 3 4 5

Next >>

Figure 3.1: A test module example generated from a well-formed JSON object.

```

1  "0": {
2      "questionType": "code",
3      "language": "c++",
4      "problem": "Description omitted for length",
5      "skeleton": "#include <iostream>\n#include <vector>\nusing
        namespace std;\n\nint count_evens(const vector<int>& v
        ){\n\n    return -1; //Replace with working code\n\n}\n\n
        nint main() {//    Make vector \nint size;\n    cout << \"
        Enter vector size: \";\n    cin >> size;\n    vector<int> v(
        size);\n    //    Populate the vector\n        for (int i = 0; i
        < size; i++) {\n    cout << \"Enter element \" << i << \"
        \";\n    cin >> v[i];\n    }\n\n    //    Test the function\n
        cout << count_evens(v);\n}\n",
6      "input": ["1"],
7      "output": ["0"],
8      "points": ["15"],
9      "difficulty": "1"
10 },
11 "prop":{
12     "test_length": "45"
13 }
14 }

```

Data-structure 3.2: JSON datafile object structure used to generate figure 3.1.

Figure 3.1 shows an example test generated from a JSON object datafile shown in figure 3.2 following an initial AJAX call from the client browser.

Once a student is finished, the "Submit Exam" button is pressed, which packages up all answers and sends them to the server to be evaluated. Once evaluated, the student receives their score and the server creates a complete record of the submission. This record includes statistics such as the question information, submitted answer, expected answer, time elapsed (for each question and exam completion), question score and total test score.

With this module students can type in an editor that provides syntax highlighting, helps with formatting (e.g., indentation, bracket matching, etc.), and allows for the compilation and execution of code. The module also provides immediate grading feedback.

3.2 Learning module

A learning module is a module that allows students to try out code snippets and examples for themselves without having to leave the browser. This is essentially tailored for media such as online text-books and practice problems. The same JSON object datafile is used, but in this case only the "language", "skeleton", "input", and "output" fields are relevant.

With this module, students can run code and attempt to pass predefined test cases along with their own in order to easily practice and learn new concepts without having to leave the instructional medium.

3.3 Server-side code execution

In order to evaluate programming problems, the code written by students must be compiled and executed. An advantage to using the Node.js framework is

the ability to spawn child processes that can execute shell scripts. This allows the Node.js server to receive the code snippet, spawn a child shell process, and receive any output data in the callback.

When a student clicks a 'Compile' button, a blank text-area will be populated by results. Once the server receives an HTML POST to '/compile', it will write the code to a temporary file and launch a process to compile and execute it with any specified input. Once the program completes execution, any output is sent back to the client in the POST response. The student will then see the execution results in the result window. Even though there is no default timeout for POST interactions, we plan on defining a maximum compile time window. If any compilation or execution goes beyond that limit, it will be terminated.

There are several ways of doing this, one of which is already built into Node.js within the `Child_process` library called `spawn`. This function accepts a command, inputs, and other option fields. When the function callback executes, the returned `ChildProcess` object will hold any output data created by the command up to a 200 kilobyte maximum.

Initially we plan to support C++14 and Python 3.5.0 code execution, but support for more languages can easily be added.

3.4 Testing security and integrity

Security is important for any sort of testing module. Since the AJAX call's response calls the script containing the necessary logic for the client, it exists only in that namespace. Because of this namespace, interaction through the console window cannot reach the code or any variables during normal execution. In order to manipulate anything, a malicious user would have to use browser

developer tools such as breakpoints, and even then it would not accomplish anything beyond wasting their allotted time and disturb the presentation of questions. This is because no exam solutions ever touch the client side and all submissions are evaluated server-side.

As an added layer of protection, any variables created are local variables and locked using the `Object.freeze()` Javascript method. This method prevents malicious users from adding new properties to the frozen object; prevents existing properties from being removed; and prevents existing properties, or their enumerability, configurability, or writability, from changing. In essence, the object is effectively immutable. This method is mainly used to lock the answers given from the multiple-choice part of the exam once students begin the programming section in order to keep students from checking their answers in the multiple-choice section.

Regarding server-side code execution, students could attempt to take advantage of this systems proposed architecture and submit malicious code. We plan on using both Unix filesystem privileges as well as some code parsing features to help protect the system. An algorithm benchmarking system created by Chen et al. [16], which attempts to accomplish its goals in a similar server-side setting, used sandboxing and regular expressions to try and filter out malicious code. Approaches similar to theirs is something we will attempt to learn from and use.

Lastly, we suggest that a proctor be used in addition to these features as we have no control over student collaboration and communication. Similarly, products such as Respondus' LockDown Browser [17] can also provide additional security by way of a custom browser that locks down the testing environment. Together, we hope these layers of security will limit the ways in which students

Task	Estimated Work Hours
Server Implementation	70
Examination module	85
Server-side Examination evaluation	90
Learning module	60
Administration module	80
Server-side code execution and evaluation	120
Final changes, bug fixes, testing, and documentation	60
Project Completion	565

Table 3.1: A visualization of how many of this project’s goals are met by each solution evaluated above.

could share or cheat in these tests.

3.5 Task Delineation

Development will be broken up into seven major categories for testing and development. Table 3.1 shows an overview of major tasks, including estimated work hours. The front-end development for the Learning and testing modules, as well as server implementation, and exam evaluation are already near completion. By looking at current development progress and time spent, together with what is left to complete, gave us an a good estimation on the total work hours for each category.

Chapter 4

Evaluation Plan

This chapter discusses how we will evaluate our solution and assess its success.

The evaluation method will consist of two different methods:

- Objective: Criteria in this category will be proven by analytical methods during the assessment
- Subjective: Criteria in this category will be demonstrated empirically and will measure value added

4.1 Objective

Our objective evaluation will rely on pass fail processes. When possible we create our code tests as unit tests. Unit tests are well understood in the field of computer science. However, unit tests may not cover all features. For those features we will use a simple check list. The following list of features will be tested:

_____ Modules are served correctly and consistently across platforms

- _____ Questions, sub-questions, and answers are all generated and displayed correctly
- _____ Server receives, compiles, and runs submitted code and reports results correctly
- _____ Submissions are received and results are reported successfully
- _____ All question specific test-cases run successfully
- _____ All solutions are evaluated against all question specific test-cases correctly
- _____ Exam results are recorded and stored successfully

Component Completion: Section 3 provided a list of features which our solution must cover. In order for our project to be considered complete and successful, it must provide all of these components.

4.2 Subjective

Subjective data is defined as data that may or may not represent the integrity of the application. This data tends to be biased, or opinionated, and could possibly misrepresent the results of the project as a whole. However, this data is still valuable as it can lead to improvements in the system.

Survey This data will be gathered by either an in-class survey or via an in-person interview, a sample survey is shown in table 4.1. By having Southern Adventist University's CPTR-124 students use prototypes of the testing module, we hope to receive useful feedback. The questions presented cover both design

related questions as well as feature specific questions. These areas allow us to discover what the user felt or thought when using our solution, and how usable the system was to the user. The responses to these questions will allow for a larger understanding of how the user reacts to our solution, and give insight to future improvements.

	1 - Very negative	2	3 - Neutral	4	5 - Very positive
How clear was the navigation					
How clear were the elements purposes					
Overall, how would you rate usability					

Did you find any errors or bugs that disrupted your usage?

What, if any, changes would you like to see made?

Table 4.1: A sample survey of questions to be asked by students who attempt to use the system.

Chapter 5

Conclusion

Currently, students and professors want a well-structured and implemented framework for writing and running code in both testing and learning environments. The current widely used solutions cover most of the goals of this project in some form, shown in table 2.2, however we did not find one that met all of the goals according to our client's specifications stated in section 1.

In this paper, we propose a separate framework to remedy the above issues. Based on our research, our proposed solution will help both students and teachers in many aspects of programming examinations as described in section 1. The opportunity to type and format code, along with the ability to compile and execute code is helpful to students and allows them to focus on actual test content and not trivial syntax or formatting issues. This solution also can help to dramatically cut down on the time it takes to build and grade exams due to the simple flat file format and automated grading. Therefore this solution will be of great benefit to the student as well as the professor, saving them both time and effort.

For future work, we will add support for other learning environment mod-

ules such as online textbooks. We will also research the possibility of implementing collaboration and monitoring features in order to view progress and work in teams, along with the feasibility of including graphics related questions.

Bibliography

- [1] D. Thiébaut, "Automatic evaluation of computer programs using moodle's virtual programming lab (vpl) plug-in," *J. Comput. Sci. Coll.*, vol. 30, no. 6, pp. 145–151, Jun. 2015. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2753024.2753053>
- [2] A. Patil, "Automatic grading of programming assignments," 2010.
- [3] S. Edwards. (2006, April) The web-cat community: Resources for automated grading and testing. [Online]. Available: <http://web-cat.org/group/web-cat>
- [4] Moodle. (2015) Moodle: Installing add-ons. [Online]. Available: https://docs.moodle.org/25/en/Installing_add-ons
- [5] J. Barrish. (2015, June) Top lms software. Capterra Inc. [Online]. Available: <http://www.capterra.com/learning-management-system-software/#infographic>
- [6] Moodle. (2015) The moodle project. [Online]. Available: <https://moodle.org/>
- [7] Edmodo. (2015) Edmodo: Connect with students and parents online. [Online]. Available: <https://www.edmodo.com/>

- [8] Moodle. (2015) Moodle: Question import formats. [Online]. Available: https://docs.moodle.org/25/en/Installing_add-ons
- [9] Moodle. (2015) Moodle: Question types. [Online]. Available: https://docs.moodle.org/24/en/Question_types
- [10] S. H. Edwards, "Work-in-progress: Program grading and feedback generation with web-cat," in *Proceedings of the First ACM Conference on Learning @ Scale Conference*, ser. L@S '14. New York, NY, USA: ACM, 2014, pp. 215–216. [Online]. Available: <http://doi.acm.org/10.1145/2556325.2567888>
- [11] D. Arnow and O. Barshay. Turing's craft. [Online]. Available: <http://www.turingscraft.com/codelabs.php>
- [12] C. A. Higgins, G. Gray, P. Symeonidis, and A. Tsintsifas, "Automated assessment and experiences of teaching programming," *J. Educ. Resour. Comput.*, vol. 5, no. 3, 2005. [Online]. Available: <http://doi.acm.org/10.1145/1163405.1163410>
- [13] S. Edwards. (2006, April) What is web-cat. [Online]. Available: <http://wiki.web-cat.org/WCWiki/WhatIsWebCat>
- [14] D. Arnow and O. Barshay, "Webtoteach: An interactive focused programming exercise system," in *Proc. IEEE*, 1999, pp. 12A9–39.
- [15] E. Foxley, C. Higgins, and A. Tsintsifas, "The ceilidh system: a general overview," in *Second Annual Computer Assisted Assessment Conf*, 1996.
- [16] M.-Y. Chen, J.-D. Wei, J.-H. Huang, and D. T. Lee, "Design and applications of an algorithm benchmark system in a computational problem solving

environment,” in *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ser. ITICSE '06. New York, NY, USA: ACM, 2006, pp. 123–127. [Online]. Available: <http://doi.acm.org/10.1145/1140124.1140159>

- [17] Respondus. (2015) Respondus lockdown browser. [Online]. Available: <https://www.respondus.com/products/lockdown-browser/>
- [18] Edmodo. (2015) Edmodo: Create and send a new quiz. [Online]. Available: <https://support.edmodo.com/hc/en-us/articles/205004764--Create-and-Send-a-New-Quiz-Teacher->