NewLeaf Final Presentation

Moodle Automated Testing Framework

CSCI 362 - Fall 2020
Luke McGuire, Chris Taylor, Kasper Dugaw

Picking a Project

Canvas

What is it?

Why we considered it?

Why didn't we choose it?

WheelMap

What is it?

Why we considered it?

Why didn’t we choose it?

Moodle

What is it?

Why we chose it?

Building Moodle

Good documentation

Final build works

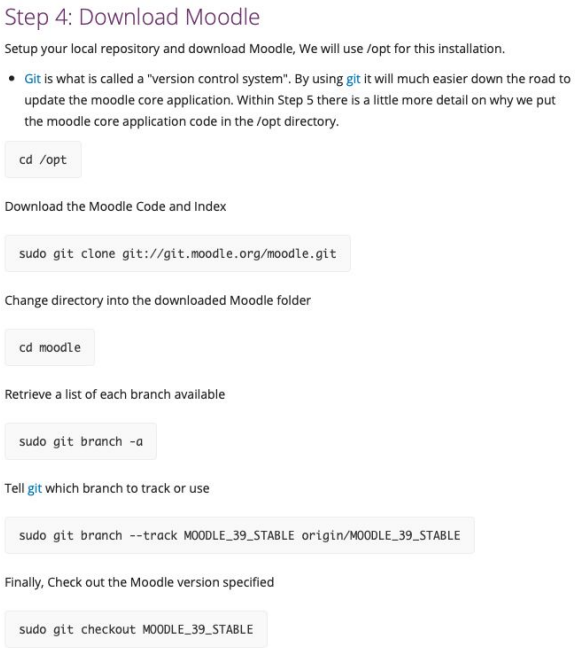
Running their tests

External libraries

Good Documentation

Step by step instructions

Detailed reasoning



Step 4: Download Moodle

Setup your local repository and download Moodle, We will use /opt for this installation.

- Git is what is called a "version control system". By using git it will much easier down the road to update the moodle core application. Within Step 5 there is a little more detail on why we put the moodle core application code in the /opt directory.

```
cd /opt
```

Download the Moodle Code and Index

```
sudo git clone git://git.moodle.org/moodle.git
```

Change directory into the downloaded Moodle folder

```
cd moodle
```

Retrieve a list of each branch available

```
sudo git branch -a
```

Tell git which branch to track or use

```
sudo git branch --track MOODLE_39_STABLE origin/MOODLE_39_STABLE
```

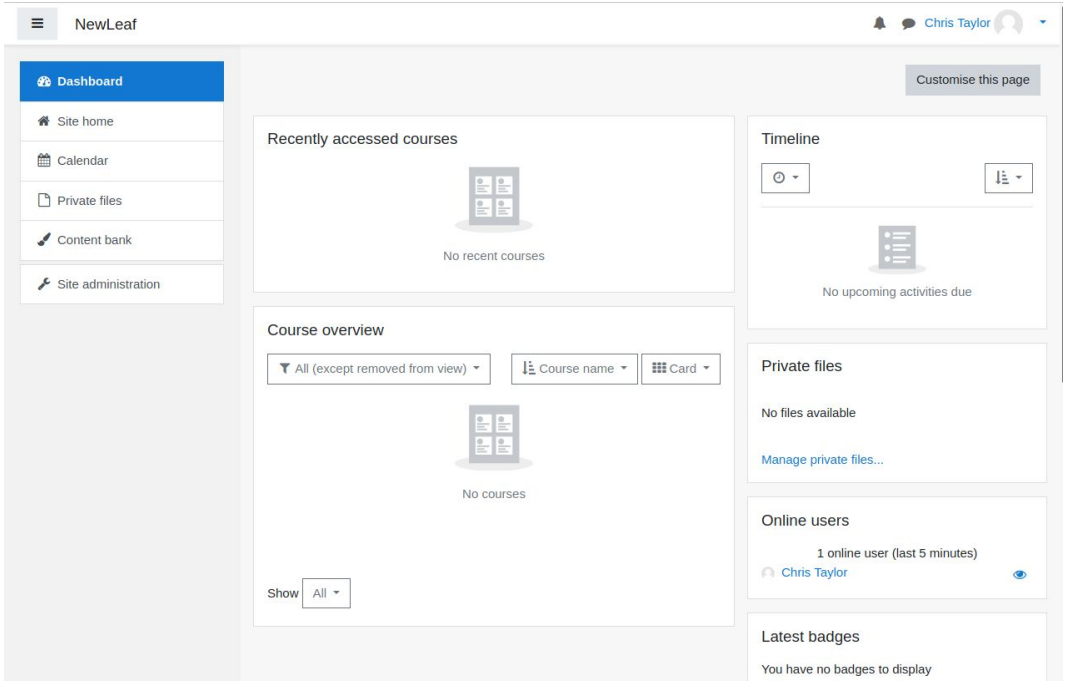
Finally, Check out the Moodle version specified

```
sudo git checkout MOODLE_39_STABLE
```

Final Build Works

Works with MySQL server

Can edit services

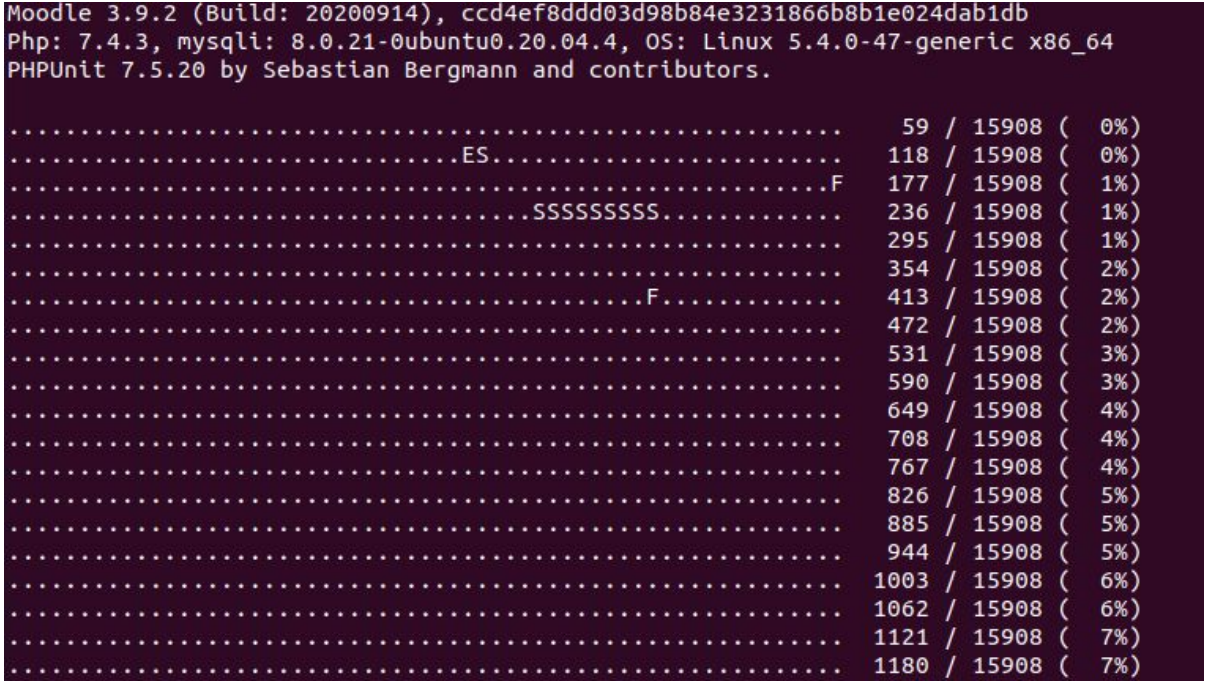


Running Their Tests

PHPUnit

Overall success or failures

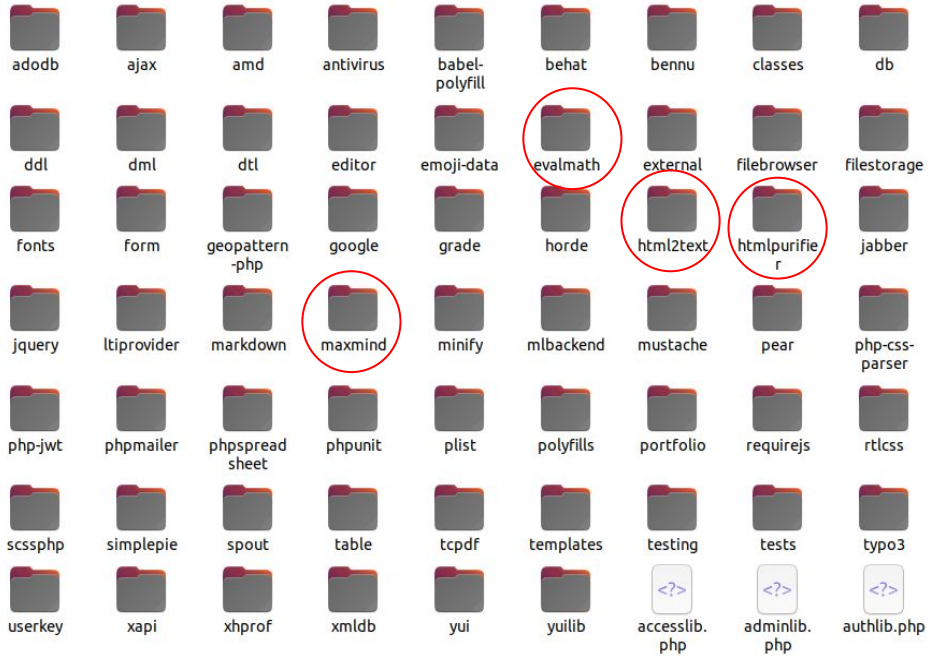
Detailed output



External Libraries

Majority not being tested with PHPUnit

Circles indicate libraries tested with our framework



The Test Plan

Developing Test Cases

Test Case Template

Test Cases

Developing Test Cases

Find method to test
Figure out example input
Solve expected output NOT using the method
Create JSON file with required fields

Test Case Template

{

“id”: Positive integer,

“driver”: “Driver”

“requirement”: “Requirement”,

“component”: “Component Name”,

“method”: “Method Name”,

“input”: “Input”,

“output”: “Expected Output”

}

evaluate() Method


```
{
        "id": 1,
        "driver": "evalmath_evaluate.driver.php",
        "requirement": "basic addition functionality",
        "component": "evalmath",
        "method": "evaluate",
        "input": "4+5",
        "expected": 9
}
```

getText() Method


```
{
    "id": 6,
    "driver": "html2text_getText.driver.php",
    "requirement": "extract text from html with multiple nested tags",
    "component": "html2text",
    "method": "getText",
    "Input": "PCFET0NUWVBFIGh0bWw+PGh0bWw+PGJvZHk+PGgxPlRoZSBIZWFkaW5nPC9oMT48cD5BIHBhcmFncmFwaC48L3A+PC9ib2R5PjwvaHRtbD4=",
    "expected": "VEhFIEhFQURJTkcKCkEgcGFyYWdyYXBoLgo="
}
```


THE HEADING

A paragraph.

<!DOCTYPE html> <html><body>
<h1>The Heading</h1>
A paragraph.
</body></html>

cidr() Method


```
{
        "id": 11,
        "driver": "maxmind_GeoIp2_Util_cidr.driver.php",
        "requirement": "convert ip/prefix to cidr notation",
        "component": "maxmind_GeoIp2_Util",
        "method": "cidr",
        "input": "1.2.3.4/16",
        "expected": "1.2.0.0/16"
}
```

getSigFigs() Method


```
{
	"id": 16,
	"driver": "htmlpurifier_UnitConverter_getSigFigs.driver.php",
	"requirement": "calculation of significant figures in decimal strings",
	"component": "htmlpurifier_UnitConverter",
	"method": "getSigFigs",
	"input": "0001.0123",
	"expected": "5"
}
```

ifthenelse() Method


```
{
        "id": 26,
        "driver": "evalmath_ifthenelse.driver.php",
        "requirement": "evaluating if and returning then or else",
        "component": "evalmath",
        "method": "ifthenelse",
        "input": "true,3,4",
        "expected": 3
}
```

Designing the Framework

- Our framework consists of three main components
  - The Controller (runAllTests.py)
  - Drivers
  - Test Cases

The Controller: runAllTests.py

- Loading Test Cases
  - Test cases are loaded, executed, and saved individually to reduce memory load
- Error Handling
  - The controller handles a variety of errors including malformed test cases and drivers
- Output Generation
  - Output is displayed in the default web browser in a sortable table

Drivers

Accepts input at as an argument

Separates values by delimiter if needed

Interfaces with the Moodle source code

Passes input to method

Returns the result to stdout

```php
<?php

//
// maxmind_GeoIp2_Util_cidr.driver.php
//
// lukem1
// Newleaf
// 16 November 2020
//


// Requirements\
require "./project/moodle/lib/maxmind/GeoIp2/Util.php";


// Input must be passed as an argument
if (!(sizeof($argv) == 2)) {
    echo "Error: Driver expected 1 argument and recieved " . (sizeof($argv)-1) . ".\n";
    return -1;
}


// Method to perform a test
// Echos: { "output": output }
function test($input) {
    //echo "Input: " . $input . "\n";

    $ary = explode("/", $input);

    $util = new GeoIp2\Util;
    $output = $util->cidr($ary[0], (int) $ary[1]);

    echo "{ \"output\": \"" . $output . "\" }";
}


// Run the test with input from argv

$in = $argv[1];

test($in);
```

Injecting Faults and Testing the Framework

- To test our framework we implemented 5 fault cases to inject into the Moodle source code
- We developed 1 fault case for each of the initial 5 drivers we developed
- These fault cases are designed to mimic errors we though a developer might make

Fault 1

```
138     protected $callbackSearch = array(
139         '/<(h)[123456]( [^>]*)?>(.*?)<\/h[123456]>/i',          // h1 - h6
140         '/[ ]*<(p)( [^>]*)?>(.*?)<\/p>[ ]*/si',                 // <p> with surrounding whitespace.
141
142
143                                                                 // FAULT: typo br -> brk
144         '/<(brk)[^>]*>[ ]*/i',                                  // <br> with leading whitespace after the newline.
145
146
147
148         '/<(b)( [^>]*)?>(.*?)<\/b>/i',                          // <b>
149         '/<(strong)( [^>]*)?>(.*?)<\/strong>/i',                // <strong>
150         '/<(th)( [^>]*)?>(.*?)<\/th>/i',                        // <th> and </th>
151         '/<(a) [^>]*href=("|\')([^"\']+)\2([^>]*)>(.*?)<\/a>/i' // <a href="">
152     );
```

In the definition of a regex filter used for finding html break elements we replaced
 with <brk>.

Fault 2

```
17    public static function cidr($ipAddress, $prefixLen)
18    {
19        $ipBytes = inet_pton($ipAddress);
20        $networkBytes = str_repeat("\0", \strlen($ipBytes));
21
22        $curPrefix = $prefixLen;
23        for ($i = 0; $i < \strlen($ipBytes)-1 && $curPrefix > 0; $i++) { // FAULT: Added -1
24            $b = $ipBytes[$i];
25            if ($curPrefix < 8) {
26                $shiftN = 8 - $curPrefix;
27                $b = \chr(0xFF & (\ord($b) >> $shiftN) << $shiftN);
28            }
29            $networkBytes[$i] = $b;
30            $curPrefix -= 8;
31        }
```

We modified the execution range of the for loop here.

Fault 3

```
192     public function getSigFigs($n)
193     {
194         $n = ltrim($n, '0+'); // FAULT: Changed '0+-' to '0+'
195         $dp = strpos($n, '.'); // decimal position
196         if ($dp === false) {
197             $sigfigs = strlen(rtrim($n, '0'));
198         } else {
199             $sigfigs = strlen(ltrim($n, '0.')); // eliminate extra decimal character
200             if ($dp !== 0) {
201                 $sigfigs--;
202             }
203         }
204         return $sigfigs;
205     }
```

Here we removed a minus sign so that this function no longer trims negative numbers properly.

Faults 4-5

```
// if the token is a binary operator, pop two values off the stack, do the operation, and push the result back on
} elseif (in_array($token, array('+', '+', '*', '/', '^', '>', '<', '==', '<=', '>='), true)) { // FAULT: Changed '-' to '+'
    if (is_null($op2 = $stack->pop())) return $this->trigger(get_string('internalerror', 'mathslib'));
    if (is_null($op1 = $stack->pop())) return $this->trigger(get_string('internalerror', 'mathslib'));
    switch ($token) {
        case '+':
            $stack->push($op1+$op2); break;
        case '-':
            $stack->push($op1-$op2); break;
        case '*':
```

```
585     static function mod($op1, $op2) {
586         return $op2 % $op1; // FAULT: Swapped the locations of $op2, $op1
587     }
```

In these two cases we placed a + operator where a - should be and reversed the order of the variables in a modulo operation.

Automating Fault Injection With MoodleMod

- To automate the fault injection process we developed another script

  ./scripts/moodleMod.sh <action>

- Inject action
  - Copies fault definition files into the Moodle source code
- Reset action
  - Resets the Moodle repository to its original state

This script also automates the process of cloning the Moodle repo with the clone action.

Five Step Setup

1. **Install PHP:** sudo apt install php7.4-cli ...
2. **Clone the repository:** git clone https://github.com/csci-362-02-2020/New-Leaf
3. **Clone Moodle:** ./scripts/moodleMod.sh clone
4. **(optional) Inject faults:** ./scripts/moodleMod.sh inject
5. **Run the tests:** ./scripts/runAllTests.py

id	component	method	input	expected	output	result
1	evalmath	evaluate	4+5	9	9	Pass
2	evalmath	evaluate	5-6	-1	-1	Pass
3	evalmath	evaluate	5*5	25	25	Pass
4	evalmath	evaluate	25/5	5	5	Pass
5	evalmath	evaluate	100>10	1	1	Pass
6	html2text	getText	PCFET0NUW ...	VEhFIEhFQU ...	VEhFIEhFQU ...	Pass
7	html2text	getText	PHA+cGFyYW...	cGFyYWdyYX ...	cGFyYWdyYX ...	Pass
8	html2text	getText	PHA+cGFyYW...	cGFyYWdyYX ...	cGFyYWdyYX ...	Pass
9	html2text	getText	PGgxPmhlYW ...	SEVBRElORz ...	SEVBRElORz ...	Pass
10	html2text	getText	PGJvZHk+PH ...	cGFyYWdyYX ...	cGFyYWdyYX ...	Pass
11	maxmind_Geo ...	cidr	1.2.3.4/16	1.2.0.0/16	1.2.0.0/16	Pass
12	maxmind_Geo ...	cidr	1.2.3.4/31	1.2.3.4/31	1.2.3.4/31	Pass
13	maxmind_Geo ...	cidr	1.2.3.4/0	0.0.0.0/0	0.0.0.0/0	Pass
14	maxmind_Geo ...	cidr	ffff:ffff:ffff:ffff:f...	ffff::/16	ffff::/16	Pass
15	maxmind_Geo ...	cidr	2acd:ffff::aabb...	2acd:ffff::aa00...	2acd:ffff::aa00...	Pass
15	maxmind_Geo ...	cidr	2acd:ffff::aabb...	2acd:ffff::aa00...	2acd:ffff::aa00...	Pass
16	htmlpurifier_ ...	getSigFigs	0001.0123	5	5	Pass
17	htmlpurifier_ ...	getSigFigs	1.1	2	2	Pass
18	htmlpurifier_ ...	getSigFigs	12.304	5	5	Pass
19	htmlpurifier_ ...	getSigFigs	-10.01	4	4	Pass
20	htmlpurifier_ ...	getSigFigs	000.000	0	0	Pass
21	evalmath	mod	7,7	0	0	Pass
22	evalmath	mod	7,6	1	1	Pass
23	evalmath	mod	100,50	0	0	Pass
24	evalmath	mod	50,60	50	50	Pass
25	evalmath	mod	1,1	0	0	Pass
26	evalmath	ifthenelse	true,3,4	3	3	Pass
27	evalmath	ifthenelse	1,3,4	3	3	Pass
28	evalmath	ifthenelse	0,3,4	4	4	Pass
29	evalmath	ifthenelse	1+1==2,3,4	3	3	Pass
30	evalmath	ifthenelse	0==0,7,4	7	7	Pass

Summary: 30 / 30 tests passed.

Results

Left: Results prior to fault injection.

Right: Results post fault injection.

id	component	method	input	expected	output	result
1	evalmath	evaluate	4+5	9	9	Pass
2	evalmath	evaluate	5-6	-1	PHP Fatal error: Uncauaht Error:	Fail
3	evalmath	evaluate	5*5	25	25	Pass
4	evalmath	evaluate	25/5	5	5	Pass
5	evalmath	evaluate	100>10	1	1	Pass
6	html2text	getText	PCFET0NUW ...	VEhFIEhFQU ...	VEhFIEhFQU ...	Pass
7	html2text	getText	PHA+cGFyYW...	cGFyYWdyYX ...	cGFyYWdyYX ...	Pass
8	html2text	getText	PHA+cGFyYW...	cGFyYWdyYX ...	cGFyYWdyYX ...	Fail
9	html2text	getText	PGgxPmhlYW ...	SEVBRElORz ...	SEVBRElORz ...	Pass
10	html2text	getText	PGJvZHk+PH ...	cGFyYWdyYX ...	cGFyYWdyYX ...	Pass
11	maxmind_Geo ...	cidr	1.2.3.4/16	1.2.0.0/16	1.2.0.0/16	Pass
12	maxmind_Geo ...	cidr	1.2.3.4/31	1.2.3.4/31	1.2.3.0/31	Fail
13	maxmind_Geo ...	cidr	1.2.3.4/0	0.0.0.0/0	0.0.0.0/0	Pass
14	maxmind_Geo ...	cidr	ffff:ffff:ffff:ffff:f...	ffff::/16	ffff::/16	Pass
15	maxmind_Geo ...	cidr	2acd:ffff::aabb...	2acd:ffff::aa00...	2acd:ffff::aa00...	Pass
16	htmlpurifier_ ...	getSigFigs	0001.0123	5	5	Pass
17	htmlpurifier_ ...	getSigFigs	1.1	2	2	Pass
18	htmlpurifier_ ...	getSigFigs	12.304	5	5	Pass
19	htmlpurifier_ ...	getSigFigs	-10.01	4	5	Fail
20	htmlpurifier_ ...	getSigFigs	000.000	0	0	Pass
21	evalmath	mod	7,7	0	0	Pass
22	evalmath	mod	7,6	1	6	Fail
23	evalmath	mod	100,50	0	50	Fail
24	evalmath	mod	50,60	50	10	Fail
25	evalmath	mod	1,1	0	0	Pass
26	evalmath	ifthenelse	true,3,4	3	3	Pass
27	evalmath	ifthenelse	1,3,4	3	3	Pass
28	evalmath	ifthenelse	0,3,4	4	4	Pass
29	evalmath	ifthenelse	1+1==2,3,4	3	3	Pass
30	evalmath	ifthenelse	0==0,7,4	7	7	Pass

Summary: 23 / 30 tests passed.

Live Demo

Conclusions

- How our Framework Turned Out
  - Efficiency and error handling
  - Report generation and features
  - Automation and ease of use
- Overall Thoughts
  - Project Management Skills
  - Designing and Implementing Tests
  - Working on Open Source projects

Q&A

NewLeaf Automated Testing Framework

Luke McGuire, Chris Taylor, Kasper Dugaw

College of Charleston Department of Computer Science

Introduction

In this project our team designed, implemented, and tested an automated PHP testing framework and several test cases for Moodle, an open source learning management system which we found lacked tests for external libraries included in their source code.

Framework Design

Our framework consists of three main components, the controller, test cases, and drivers.

The controller is responsible for parsing and executing test cases as well as generating the results into a report.

The test cases are a set of JSON files which specify individual tests and provide the controller with the data necessary to execute them including the driver that should be used to run the test, the input that should be given to the functionality being tested, and the expected output of the provided input.

Finally, the drivers are responsible for interfacing with the Moodle code and providing the controller with the ability to test a particular method or class and returning the results of a test back to the controller.

Implementing Tests

After implementing our framework we developed 30 tests for the Moodle project for 6 functions found in the project's libraries. The modular design of our framework provides a simple framework for creating new drivers and tests, and examples of these can be seen in Figures 3 and 4 respectively.

Figure 1: Test results prior to fault injection.

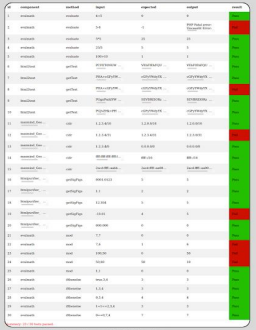


Figure 2: Test results post fault injection.

Testing and Results

To validate our testing framework we designed a system to inject faults into the Moodle source code and executed our framework before and after injecting the faults. The results of this procedure can be seen in Figures 1 and 2.

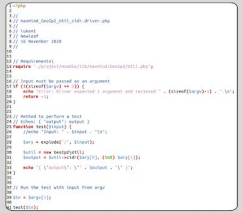


Figure 3: One of the drivers written for our framework.

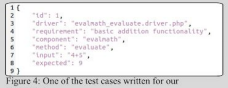


Figure 4: One of the test cases written for our framework.

Conclusions

Over the course of this project our team learned quite a bit about the software engineering process, including project planning and management, designing and producing good tests, and how to effectively work as a team on a software project.

We were also pleased with how our testing framework itself ended up turning out. Although we had some setbacks, we were not only able to design and implement our vision, but expand upon it. After completing the basic functionality of our framework earlier than expected we were able to spend time improving or project by adding features including improved output generation, improved error handling, and scripts to automate the process of fault injection. Additionally, we were also glad we were able to produce something we felt was valuable by writing our tests for Moodle libraries that did not have pre-existing tests.

To conclude, if it's not already clear, we were very satisfied with the results of our project and our development as a team and as software engineers.

Acknowledgments

Thanks to the Moodle project and its contributors:
https://moodle.org/

Thanks to our instructor and faculty advisor:
Dr. Jim Bowring

Additional Information

Our GitHub repository, which contains the entirety of our project including source code and our final report can be found at the link below.

https://github.com/csci-362-02-2020/New-Leaf