Why moodle?

Moodle attracted the interest of our group both because of its level of documentation, and because we felt as a group that we could work effectively with this project. The prospect of working with php, a language nobody on our team has used other than on occasion, also excited all three of us. Sure enough, we fairly quickly worked through the entire build process and had Moodle up and running in about an hour.

moodle

Test Plan

Testing Process

In order to test the desired method of the class we are interested in, we want to call a program at the command line that runs each test. The program will automatically run all of the tests one by one. The results of the tests will be compared to the expected results of the test cases, and then a test report with the results and other info will be displayed in a browser window.

Requirements Traceability

TC01 | Creating a valid timestamp

Timestamps are very important for every system, particularly for our endeavors in the Unix systems. In such systems, they are based on a running total of seconds that begins at the Unix epoch, which is advantageous as they can represent all timezones with a single measurement. As such, this facilitates users' ability to see a website like Moodle) in real time as opposed to the timezone the server is located in.

Tested Items

"Convert_to_timestamp" in the "type_base" class of the Moodle project/moodle/calendar/classes/type_base.php

Testing Schedule

For the next week, we need to establish core functionality between script, driver, and php testing classes. Once established, we will spend this first week testing the Unix Timestamp method of the type_base class with its many test cases. The following weeks will be dedicated to applying the same techniques to other test cases that we will discover. Methods that use any arithmetic to calculate grades, or calendar dates would be ideal. Successful connections to the database/browser could be another test. We are currently looking into this and several other possibilities.

Test Recording Procedures

All of the actual test results will be compared to the expected test results contained in various appropriately labeled json files.

The results of the comparisons described above will be displayed in a browser. The information will be properly and clearly labeled in a uniform manner with each test case getting its own portion of the report.

Hardware and Software Requirements

Linux OS

A computer

Constraints

We have a limited schedule to figure out the relation of different pieces to our framework.

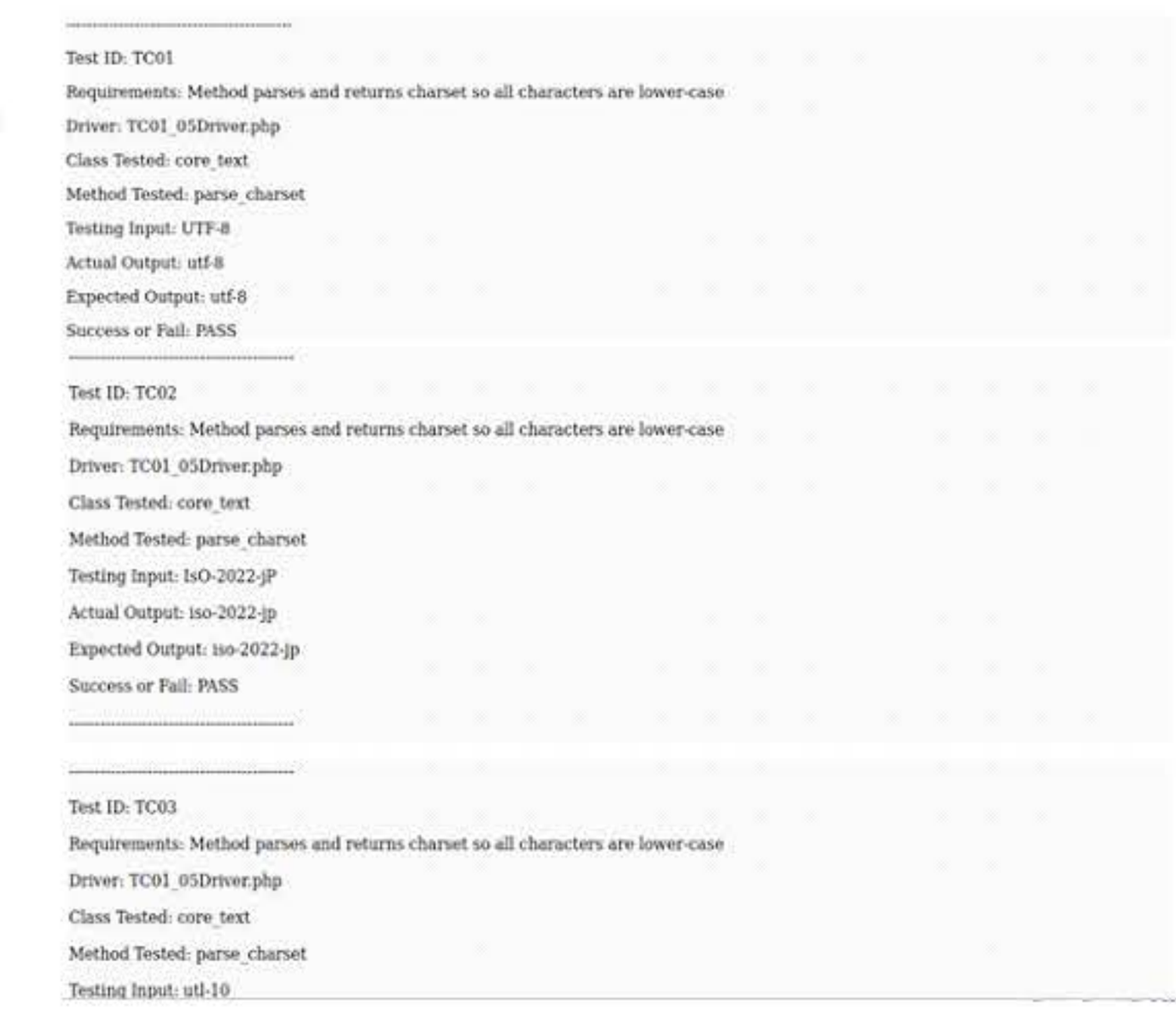
Building our first Driver & Script

Initial Driver

Initial Script

Output

AOBTD TESTING FRAMEWORK



Hurdles Faced

As with every part of the project so far, we faced a number of hurdles we had to overcome. Aside from the typical struggles with technology/computers and simple errors like syntax errors, we faced two real challenges during this phase. The first involved a config file that we didn't realize had to be tailored to each individual user. The other problem was actually related to this issue, though is a different issue altogether involving the same file.

Our initial issue was relatively easy to solve. Having configured the script in a way that it should at least run, two of us were receiving a rather obscure error message about a file named config.php while the third group member's code was running perfectly. After some investigating, we realized that all three of us had our config.php files set with the Moodle database details for the group member whose code was working. After correcting these discrepancies by filling the config.php file with our own Moodle database details, our code ran without any issue, yet this wasn't the end of the hurdles we faced.

The other significant issue came with the resolution of our previous dilemma. Having personalized our c onfig.php files, we realized that pushing our updates to our github repo and pulling them would write over each other's config.php files, thus necessitating us to perform the same fix from the previous issue e ach time we pull from our repo. To solve this, it was decided that we should include our config.php file in a .gitIgnore folder so that github will ignore the config.php file when we push to and pull from our repo. These were the biggest issues we faced over the course of this deliverable, and thankfully we seem to have a handle on them.

Full Testing Suite

Refactored parse_charset driver

break_up_long_words driver

The Refactored Script

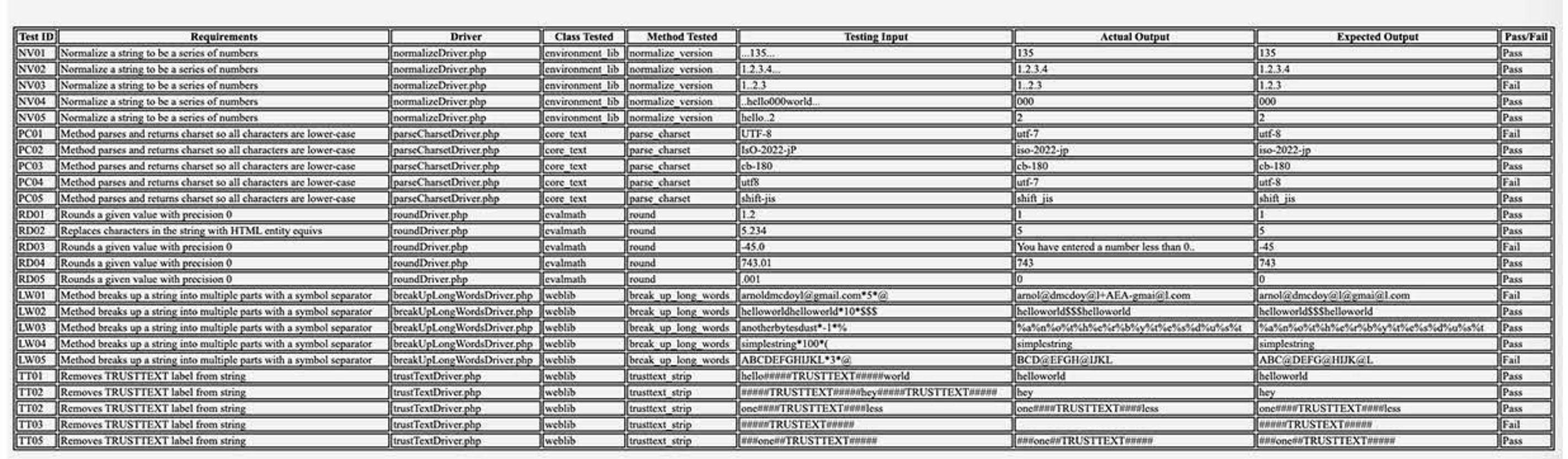
parse_charset method signature in Moodle

```
public static function parse_charset($charset)
```

break_up_long_words method signature in Moodle

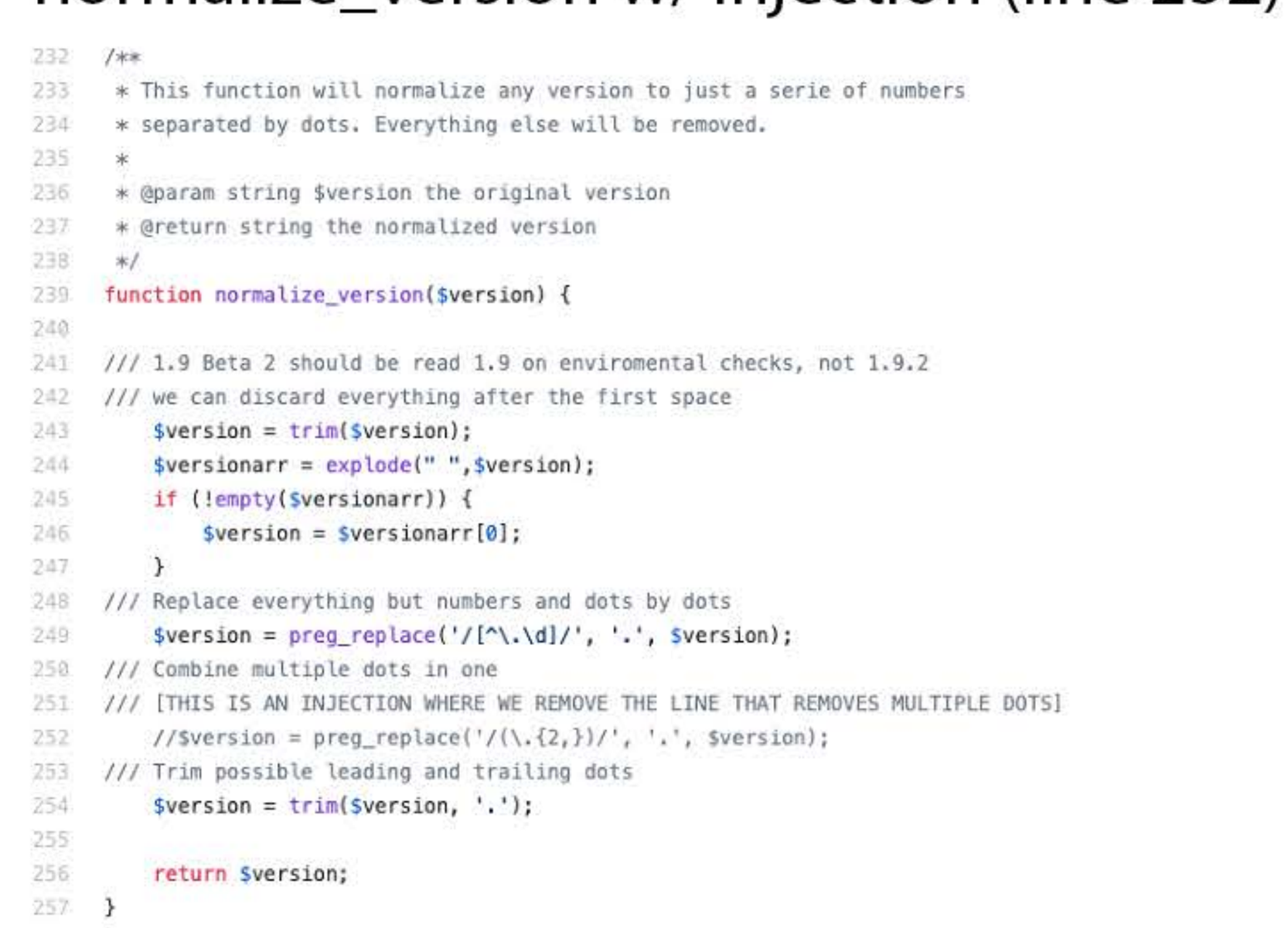
```
function break_up_long_words($string, $maxsize=20, $cutchar=' ')
```

AOBTD TESTING FRAMEWORK

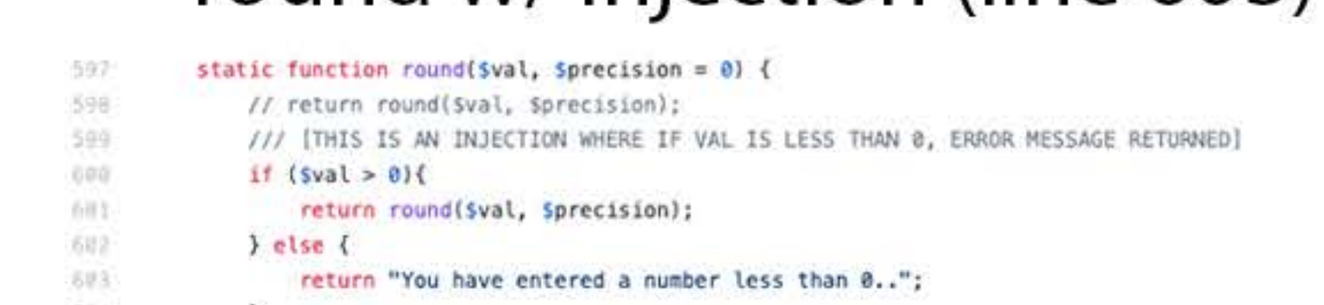


Error Insertion and Analysis

normalize_version w/ injection (line 252)



round w/ injection (line 603)



trusttext_strip w/ injection (line 1634)

break_up_long_words w/ injection (lines 1010-1012)

parsecharset w/ injection (lines 136-138)

Overall Experience

As a whole, this project was not what any of us expected. We all thought that working on an open source project and developing a program such as the one we wrote would be different than from how it actually was. Of course while we each had our own unique expectations, we were all still surprised by how even the simple things such as working in a team turned out to be different. Nonetheless it was still a very challenging and constructive endeavor, and it's safe to say that we all learned something.

I think one of the biggest misconceptions we had about working with an open source project is that we figured it would be, well, better. Moodle is obviously a large project with years of development behind it, something that in and of itself is impressive. And for the most part, Moodle not only functioned well, but appeared to be fairly polished in terms of documentation and project architecture. However, we were still surprised at how some small things within Moodle seemed wrong or misleading, such as how our round test method actually just truncated the input. Perhaps a more glaring mistake is that some methods lacked any significant documentation. Considering the size of the project, we also found this to be surprising.

One of the other surprising factors for us was the need for efficiency. We covered the refactor process fairly in depth previously in this report, but one small piece of that refactor process involved reworking some code that worked terribly inefficiently. While we actually never ran into this issue, we were aware that another project group with a similar strategy ran into an issue that caused their program to crash. Clearly this scared us enough to go and quickly solve the issue so that our program won't crash as a result of our being inefficient. In hindsight, this makes perfect sense, but it was a tad surprising to us that such a seemingly innocuous design choice could wreak havoc like that down the road.

One of the final surprises for us was learning how to work as a team. We pretty quickly divvied up the roles and duties we would each have throughout the course of this project, but we weren't always good at sticking to them. Ultimately we managed to push out a product that we can all be proud of, but there were moments when the roles shifted because things needed to get done. None of this is to say that we clashed as a team or anything like that, in fact we remained largely disagreement free and never degenerated into any sort of argument. Still, the changing team dynamics and responsibilities on the fly and as we encountered problems was a surprise to each of us.