CSCI 3308 Software Development Methods and Tools: Milestone 5: Unit Testing

Cameron Maywood, Tyler Aratani, Daniel Holmes, Jaden Grossman, Zhaozhong Peng

University of Colorado Boulder, Boulder, Colorado, 80310

April 18, 2016

Vision

For unknown musicians who need a platform to promote their work and for those looking to discover new music, Apollo is a web-based competitive music platform that allows musicians to promote their work through bracket-style competition. Unlike other music hosting sites, Apollo lets the listeners decide which songs and artists are promoted.

User Acceptance Test Plan

Preliminary unit testing of the Apollo software will be carried out on three primary features of the application:

1. Account creation and login.
2. User profile bio.
3. User profile picture.

The specific user acceptance test cases for each of these features can be described as such:

Account Creation and Login

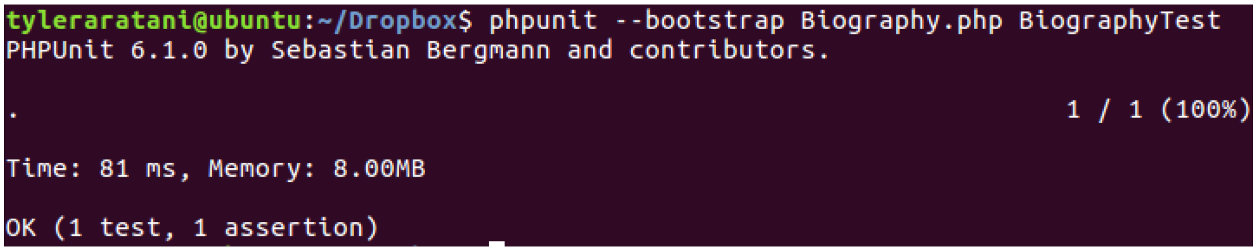
Testing the account creation/login was made relatively easy due to the built-in error handling functionalities within the Parse backend platform that is being utilized. In the case of duplicate usernames or emails being used during an account creation. Parse automatically handles and throws an error, preventing this from occurring.

Profile biography character limit.

Testing the biography upload was pretty simple since most of it is handled before it gets to the PHP such as checking that the input is a string. However, we wanted to implement a character limit in order to control how much memory each biography can take. This was done with a simple if statement to make sure the data did not exceed and certain limit (1000). PHPUnit was then used to make sure that the proper error was received when the criteria was met. A screen shot for the tests can be seen in Figure 1.

Proper functionality of profile picture

To ensure proper functionality of the image upload portion of the profile page we needed to test several elements including; that the file exists, the target directory exists and can be written to, the file does not exceed a set maximum size, and that the file is actually an image. Testing if the file is valid is simple because PHP has developed built-in functions that make it easy to test. The Upload.php checks the validity of the file by simply making sure all the fields for the file array are not empty.

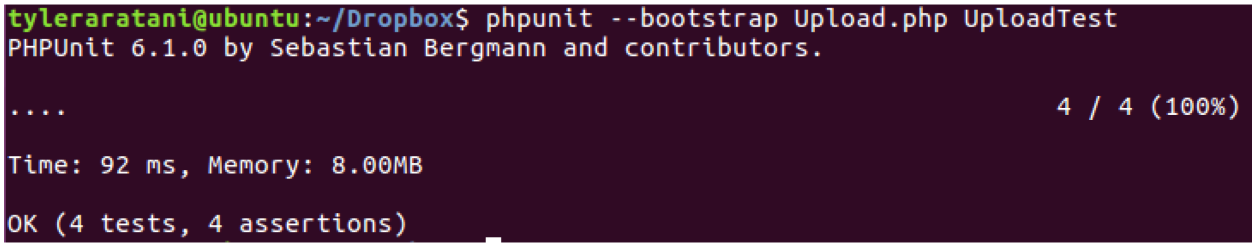


```
tyleraratani@ubuntu:~/Dropbox$ phpunit --bootstrap Biography.php BiographyTest
PHPUnit 6.1.0 by Sebastian Bergmann and contributors.

.                                                                  1 / 1 (100%)

Time: 81 ms, Memory: 8.00MB

OK (1 test, 1 assertion)
```

Figure 1. PHPUnit Biography.php tests.

To test if the file can be uploaded, PHP also offers a built in function named *is_writable*() which takes in a directory and returns true if the file can be written to and and false otherwise. We used PHPUnit to make sure that the directory we set as a default within the Upload.php file can be used to write files. We tested for both cases. Testing the maximum size was also straight forward. The maximum size is set inside of the Upload.php file and to test it we used PHPUnit to make sure we would receive an error if the maximum limit was reached.

The final thing that we tested was whether or not the file was a valid image. More specifically if the file extension was png, jpg, or jpeg. When a file is uploaded, one of the elements that PHP gives is the type and for images the types given are in the form of (image/'type'). To test whether the file was an image, an array was created within the Upload.php file that held the three possible types that would be given by the PHP file object. We then created a function that takes in the file and compares the type given by the PHP object to the values in the array and returns true if the value is found and false if it is not. We also tested this is with PHPUnit by making sure that if the correct types were given, there would not be an error and if a false type was given, the file would not be able to upload. A screen shot of the PHPUnit test running can be seen in Figure 2.



```
tyleraratani@ubuntu:~/Dropbox$ phpunit --bootstrap Upload.php UploadTest
PHPUnit 6.1.0 by Sebastian Bergmann and contributors.

....                                                               4 / 4 (100%)

Time: 92 ms, Memory: 8.00MB

OK (4 tests, 4 assertions)
```

Figure 2. PHPUnit Upload.php tests.