BICYCLE MANUFACTURING ERP SYSTEM

TEST PLAN

Version *1.0*
02/03/2021

1 INTRODUCTION

1.1 PURPOSE OF TEST PLAN

The Test Plan is a document that will help improve our code through a set of procedures that are set in place when writing the ERP system. Testing serves as a reassurance that the code (or refactored code) works as intended. Testing will also serve as an extension to debugging as well. Some code will not pass the test, and will help the team to find and pinpoint the location of that bug.

Furthermore, tests will make it easier for the team to progress with confidence that there won't be any redos, not only because it is free of bugs, but because the test will ensure that the program works as intended by the specifications.

2 PROCEDURE

2.1 LARAVEL FRAMEWORK

Laravel, the framework that is used to build the ERP system, is a robust and powerful framework for web development. That being said, Laravel comes with an already included testing framework, PHPunit [1], with a testing file, phpunit.xml. By default, the application already has two test directories: "Feature" (to test multiple objects and component interaction) and "Unit" (to test individual components).

Creating test classes is easy: use the Artisan command "make:test" and it will appear in the "test/Feature" directory in the project files, or alternatively, if the goal is to create unit tests, adding "--unit" at the end of the command will create a unit test in the "test/Unit" directory.

Below is an example of a PHPunit test auto generated by Laravel in the test/Unit folder called ExampleTest.php.

ExampleTest.php

```php
<?php

namespace Tests\Unit;

use PHPUnit\Framework\TestCase;

class ExampleTest extends TestCase
{
    /**
     * A basic test example.
     *
     * @return void
     */
    public function testBasicTest()
    {
        $this->assertTrue(true);
    }
}

```

Finally, the “php artisan test” command will run all tests located in the test folder.

```
D:\Projects\SOEN-390-Team5\ERP>php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Unit\ExampleTest
  ⨯ basic test

  PASS  Tests\Feature\ExampleTest
  ⨯ basic test

  Tests:  2 passed
  Time:   0.62s


D:\Projects\SOEN-390-Team5\ERP>
```

2.2 THE MODEL VIEW CONTROLLER MODEL

One of the advantages of using the MVC architecture is that it is much easier to test the individual components. More specifically, the unit test will be one of the main testing techniques that will be used throughout the project due to the nature of the model.

2.3 THE USERS AND THEIR VIEWS

The way the ERP system will work is depending on the user. They will have different responsibilities in the company, which means that they will also have access to different views/components of the ERP system. To test this, the system will check the type of user who is currently logged in and will assert that the view returned is the intended one. For example, suppose that there is a page that only the managers are able to see. The test will create a controller object for manager, suppose a managerController object. By creating this object, the system knows that this user is a manager and will generate specific pages based on the fact that the user is a manager. From there, the remainder of the test is to ensure that the page (the view) is the correct one. Many of the MVC tests will be carried out similarly to this.

2.4 USER LOGIN

There are multiple things that would need to be tested for user login.

First, a user will need to have their username (namely their email) and a password. To test this, there needs to be a test that checks for the three options possible:

1. If the username exists and the password is a match, then it should be able to login. The success criterion is that the user is currently in the ERP system.
2. If the username exists but the password does not match, then the user should still not be able to enter the ERP system. The success criterion is that the user remains on the login page with an error message saying that the user did not enter the correct credentials after submitting a request.
3. If the username does not exist, then the user should not be able to access the ERP system. The success criterion is that the user remains on the login page with an error message saying that the entered username does not exist in the ERP system.

Second, the ERP system will have to check the proper conditions for creating a new user.

1. For security reasons, as well as for company integrity, there should only be one type of user that can create new users. The test would create an instance of the create user page and would check the type of user that comes with the session.

The success criterion would be that the createUser view is only accessed if the currently logged in user is the IT personal.

2. Creating a user should only occur if there is no {username, password} pair in the database in which it is stored. This creates two tests:

    2.1. If the username does not already exist, it <does something>. The success criterion is that the username and password is the last entry in the database in which it is stored, after the request has been submitted.

    2.2. If the username already exists, it will redirect the current user back to the createUserView, and displays an error message saying that the user already exists in the database. The success criterion is that the logged in user is shown the createUserView AND that the database has not registered the entry.

Third, the ERP system should keep track of who is logged into the user. In other words, there cannot be two instances of the same user in the database since this would be a security issue. The test for this would create an instance of a logged in user, and create a second user. For a robust test, there needs to be 2 success criteria:

1. If the user is able to login, the number of instances that is logged in needs to be 1.
2. If the user is already logged in and there is an attempt to login, the user should be redirected to the user login page with an error saying that they are logged in from elsewhere. A successful test will confirm that the current user is at the loginPage view.

3. References

[1] "PHPUnit – the PHP Testing Framework." *Phpunit.de*, 2013, phpunit.de/. Accessed 2 Feb. 2021.