BICYCLE MANUFACTURING ERP SYSTEM

TEST PLAN

Version *4.0*
07/04/2021

1

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

SPRINT 1

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.

SPRINT 2

2.5 USER LOGOUT

A very important feature that works alongside with the login functionality is the ability to logout of an account. This of course serves to protect one's account and their private information, but to also allow for the changing of accounts. In order to test the feature, a simple test must be conducted.

1. Upon successfully logging into an existing user's account, a button must appear on every page which allows for the user to logout. The button must be clickable on every single page to assure proper functionality.

2. If the user clicks logout, the session must be terminated, and the page must be redirected to the login screen.

3. If the page is redirected, changing the URL to an account specific one denies access such as /jobs and will once again redirect the user to the login page.

2.6 JOBS

The “jobs” view is used to see which orders currently need to be completed in a table format. Therefore, the test cases are as follows:

1. When an order is created, it should show up in the corresponding database table and the number of rows should be increased by 1.

2. The order should also be visible in the view after inserting a row in the table. Hence, after an order has been placed, the page should refresh and the order should be visible in the jobs table.

3. If an order is cancelled, the ERP user should be able to delete the order and it will also be reflected in the corresponding database, meaning that it should be deleted in the database once that request has been processed.

4. Once an entry is deleted, another test is that it will fail to retrieve the deleted entry.

2.7 ASSEMBLY

For assembly, for sprint 2, there is not a lot of functionality related to it. It serves as a way for the one using it to know what is needed to either build a bike in terms of parts, and what is needed to build parts in terms of materials. Currently, the only test that can be done is the MVC test mentioned in 2.1 of this document.

2.8 INVENTORY

There are various aspects to be tested in the case of tracking the stock of the company. In each aspect, the focus is related to bicycles, parts, and materials.

Firstly, it is necessary to allow for the addition of new information into the stock system in the case that new bikes/parts/materials are made available. In order to test this, there needs to be a test that checks for the following criteria.

1. If the user clicks the add button a form must be displayed for the user in order to input the desired specifications.

2. If the user clicks the add button on the pop-up form, it must be validated such that invalid characters aren't present and no fields can be empty. If invalid, the bikes/parts/materials must not be added to the inventory and a notification should appear.

3. If a bike/part/material of that exact same specification already exists, it must be added \to quantity within the table rather than creating an already existing row.

4. If a bike/part/material is unique and doesn't already exist, it must be added into its own separate row within the table.

Secondly, it is important to allow for the deletion of new bikes/parts/materials within the stock system. This is important if a bike got sold, a part or material was used for a bike, an error was made in the development process, or an order got cancelled. In order to test this, there needs to be a test that checks for the following criteria.

1. If the user clicks the delete button, a form must be displayed for the user to confirm their request.

2. If confirmed, desired bikes/parts/materials are removed from the table and database.

3. If canceled, desired bikes/parts/materials are not removed from the table and database.

Thirdly, it is important to allow for the editing of new bikes/parts/materials within the stock system. This is crucial if mistakes were made during the inputting process, or in case changes were made last minute to the order. In order to test this, there needs to be a test that checks for the following criteria.

1. If the user clicks the edit button, a form must be displayed for the user in order to input the desired specifications.

2. If the user clicks the edit button on the pop-up form, it must be validated such that invalid characters aren't present and no fields can be empty. If invalid, the bikes/parts/materials must not be edited in the inventory and a notification should appear.

3. If a bike/part/material of that exact same specification already exists, it must be added to quantity within the table while deleting the previous existing row.

4. If a bike/part/material is unique and doesn't already exist, it must be added into its own separate row within the table.

SPRINT 3

2.9 USER INSERTION

A very important feature that works alongside with the admin functionality is the ability to insert an account in the system database.

1. Upon successfully logging into the admin account, a manage users button must appear.

2. A create user button will then appear on the screen. Upon clicking that button, an additional UI displays allowing the admin to enter the new user's information.

3. Once validated and confirmed, the new user is created with the inputted information into the database.

4. Upon failed validation, no user is created and a message is displayed.

2.10 USER EDITING

Another important feature which is part of the admin functionality is the ability to edit an account in the system database.

1. Upon successfully logging into the admin account, a manage users button must appear.

2. A list of current users will then be listed on the view, which each have a button for editing.

3. Upon clicking that button, an additional UI displays allowing the admin to enter new information.

4. Once validated and confirmed, the new information is then inputted into the database for that specific user.

5. Upon failed validation, no information is changed and a message is displayed.

2.11 JOBS

There are various objectives to be tested in the case of tracking the jobs of the company.

Firstly, it is necessary to allow for the addition of new jobs into the system in the case that new orders are received. In order to test this, there needs to be a test that checks for the following criteria.

1. If the user clicks the add new job button, a form must be displayed for the user in order to input the job details.

2. If the user clicks the confirm button on the pop-up form, it must be validated such that the field is not empty. If invalid, the job must not be added to the jobs database and a notification should appear.

3. If valid information entered, it must be added into its own separate row within the jobs table.

Secondly, it is necessary to allow for the deletion of jobs into the system in the case that a job is complete. In order to test this, there needs to be a test that checks for the following criteria.

1. If the user clicks the delete job button, the database must also remove this entry.

2. The table should then be updated with the correct jobs.

Thirdly, it is important to allow for the toggling of job status in the system. This is crucial to queue/complete orders. In order to test this, there needs to be a test that checks for the following criteria.

1. If the user clicks the Toggle Status button, the database must change this entry's status.

2. The table should then be updated with the correct status.

2.12 LOGS

There are two main things to be testing when looking at the logs.

Firstly, for organization purposes, it is important to be able to download the csv files for the logs. In order to test this, there needs to be a test that checks for the following criteria.

1. If the user clicks the Export button, the software must return the logs in a csv file.

Secondly, for additional organization purposes, it is necessary to be able to cycle through, access logs, action logs, and all.

1. If the user clicks the All tab, the software must return the logs for all types.

2. If the user clicks the Access tab, the software must return the logs for access types.

3. If the user clicks the Action tab, the software must return the logs for action types.

2.13 SALES LOGS

There are two main things to be testing when looking at the sales logs.

Firstly, for bookkeeping and accounting reasons, it is important to be able to look at old sales by year or month. In order to test this, there needs to be a test that checks for the following criteria.

1. If the user clicks the Choose a Time button, the software must return the sales of that time frame.

2. The returned sales correlate to the selected time frame as well as display the correct information.

Secondly, for additional bookkeeping and accounting purposes, it is necessary to be able to cycle through, monthly sales logs, yearly sales logs, and all sales logs.

1. If the user clicks the All tab, the software must return the sale logs for all types.

2. If the user clicks the Monthly tab, the software must return the sale logs for monthly report.

3. If the user clicks the Yearly tab, the software must return the sale logs for yearly report.

3. References

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