

## **PROJECT**

---

Roberto Booysen (ST10085125)

Keely-Ann Maritz (ST10085428)

Nikita Davids (ST10085223)

## **DIS YEAR 3**

## **XISD6329**

11/29/2023

Courteney Young

*I hereby declare that I did not plagiarise the content of this assignment and that this is my own work.*

Assignment submitted via SafeAssign or Turnitin: ☒ (Tick the Box)

## Table of Contents

1	Azure Boards .....	1
2	Azure Repos .....	3
3	Azure Pipelines .....	4
4	Unit Testing .....	5
5	Load Testing .....	13
6	Auto Deployment.....	14
7	Reference List.....	15

# 1 AZURE BOARDS

Link: [https://dev.azure.com/ST10085428/JLPALC/\\_boards/board/t/JLPALC%20Team/Stories](https://dev.azure.com/ST10085428/JLPALC/_boards/board/t/JLPALC%20Team/Stories)

The screenshot displays the Azure DevOps interface for the JLPALC Team. The left sidebar shows the navigation menu with options like Overview, Work Items, Boards, Backlog, Queries, Delivery Plans, Analytics View, Repos, Pipelines, Test Plans, and /infacts. The main area shows a Kanban board with the following columns: Not Started (1 item), Started (0 items), Testing (0 items), Review (0 items), and Complete (25 items). The 'Complete' column contains 25 stories, each with a title, a status (Done), and a completion date. The stories are numbered 329 through 245, and they all have a status of 'Done' and a completion date of '10/10/2023'.

Story ID	Title	Status	Completion Date
329	As a developer I want to ensure Azure Data Management through PDI Integration in the Project Tool Page	Done	10/10/2023
328	As a developer I want to ensure Error Messages with a User-friendly Error Page	Done	10/10/2023
327	As a developer I want to ensure Project Data Management in the Project Resource Page with PDI Integration	Done	10/10/2023
326	As a developer I want to ensure Project Submission with a User-friendly Image Page	Done	10/10/2023
325	As a developer I want to ensure Azure Data Management with Project PDI Integration	Done	10/10/2023
324	As a developer I want to ensure Error Messages with a User-friendly Error Page	Done	10/10/2023
323	As a developer I want to ensure Azure Data Management with Project PDI Integration	Done	10/10/2023
322	As a developer I want to ensure Project Submission with a User-friendly Image Page	Done	10/10/2023
321	As a developer I want to ensure Azure Data Management with Project PDI Integration	Done	10/10/2023
320	As a developer I want to ensure Project Submission with a User-friendly Image Page	Done	10/10/2023
319	As a developer I want to ensure Azure Data Management with Project PDI Integration	Done	10/10/2023
318	As a developer I want to ensure Project Submission with a User-friendly Image Page	Done	10/10/2023
317	As a developer I want to ensure Azure Data Management with Project PDI Integration	Done	10/10/2023
316	As a developer I want to ensure Project Submission with a User-friendly Image Page	Done	10/10/2023
315	As a developer I want to ensure Azure Data Management with Project PDI Integration	Done	10/10/2023
314	As a developer I want to ensure Project Submission with a User-friendly Image Page	Done	10/10/2023
313	As a developer I want to ensure Azure Data Management with Project PDI Integration	Done	10/10/2023
312	As a developer I want to ensure Project Submission with a User-friendly Image Page	Done	10/10/2023
311	As a developer I want to ensure Azure Data Management with Project PDI Integration	Done	10/10/2023
310	As a developer I want to ensure Project Submission with a User-friendly Image Page	Done	10/10/2023
309	As a developer I want to ensure Azure Data Management with Project PDI Integration	Done	10/10/2023
308	As a developer I want to ensure Project Submission with a User-friendly Image Page	Done	10/10/2023
307	As a developer I want to ensure Azure Data Management with Project PDI Integration	Done	10/10/2023
306	As a developer I want to ensure Project Submission with a User-friendly Image Page	Done	10/10/2023
305	As a developer I want to ensure Azure Data Management with Project PDI Integration	Done	10/10/2023
304	As a developer I want to ensure Project Submission with a User-friendly Image Page	Done	10/10/2023
303	As a developer I want to ensure Azure Data Management with Project PDI Integration	Done	10/10/2023
302	As a developer I want to ensure Project Submission with a User-friendly Image Page	Done	10/10/2023
301	As a developer I want to ensure Azure Data Management with Project PDI Integration	Done	10/10/2023
300	As a developer I want to ensure Project Submission with a User-friendly Image Page	Done	10/10/2023



## 2 AZURE REPOS

Link: [https://dev.azure.com/ST10085428/\\_git/JLPALC](https://dev.azure.com/ST10085428/_git/JLPALC)

The screenshot displays the Azure DevOps interface for the JLPALC repository. The left sidebar shows the project structure with folders like 'data', 'CD', 'images', 'scripts', 'tests', 'utils', and 'workflows'. The main area shows a list of files and folders, including 'New 1', 'data', 'CD', 'images', 'scripts', 'tests', 'utils', and 'workflows'. The bottom section contains a 'Just Love Play And Learn Centre' logo and text.

**Just Love Play And Learn Centre**

The Just Love Play And Learn Centre is a registered charity, established in 2010, dedicated to providing a safe and secure environment for children and young people to play and learn. The Centre is a registered charity, established in 2010, dedicated to providing a safe and secure environment for children and young people to play and learn. The Centre is a registered charity, established in 2010, dedicated to providing a safe and secure environment for children and young people to play and learn.

Robbie Dwyer (07700000000)  
Nicky Dwyer (07700000000)  
Katy Dwyer (07700000000)

### 3 AZURE PIPELINES

Link: <https://dev.azure.com/ST10085428/JLPALC/build>

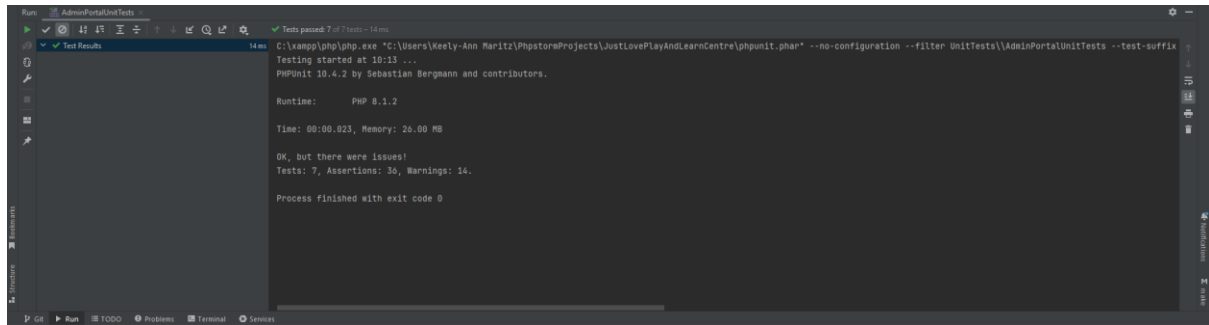
The screenshot shows the Azure DevOps interface for the JLPALC project. The left sidebar contains navigation links: Overview, Boards, Repos, Pipelines (selected), and Environments. The main area is titled 'Pipelines' and shows a table of 'Recently run pipelines'. The table has two columns: 'Pipeline' and 'Last run'. One pipeline is listed: 'JLPALC' with a status of 'Succeeded' (green checkmark) and a last run time of '20m ago'.

Pipeline	Last run
JLPALC	#20231127.2 • Updated README.md Individual CI for main 7c326aa9 20m ago

The screenshot shows the 'Runs' tab for the JLPALC pipeline. The left sidebar is the same as the previous screenshot. The main area is titled 'JLPALC' and shows a table of pipeline runs. The table has three columns: 'Description', 'Stages', and 'Run time'. There are five runs listed, all with a status of 'Succeeded' (green checkmark).

Description	Stages	Run time
#20231127.2 • Updated README.md Individual CI for main f672432f	✓	19m ago 19s
#20231127.1 • Updated README.md Individual CI for main 7c326aa9	✓	20m ago 11s
#20231126.1 • Merged PR 9: updated pages commit Individual CI for main 18531eeb	✓	Yesterday 11s
#20231124.1 • Merged PR 8: Updated php files Individual CI for main 25f36169	✓	Friday 15s
#20231123.1 • Set up CI with Azure Pipelines Individual CI for main 88bc4ee1	✓	Thursday 11s

## 4 UNIT TESTING



Run AdminPortalUnitTests

Test passed: 7 of 7 tests - 14ms

Test Results

C:\xampp\php\php.exe "C:\Users\Keely-Ann Maritz\PhstormProjects\JustLovePlayAndLearnCentre\phpunit.phar" --no-configuration --filter UnitTests\AdminPortalUnitTests --test-suffix

Testing started at 10:13 ...

PHPUnit 10.4.2 by Sebastian Bergmann and contributors.

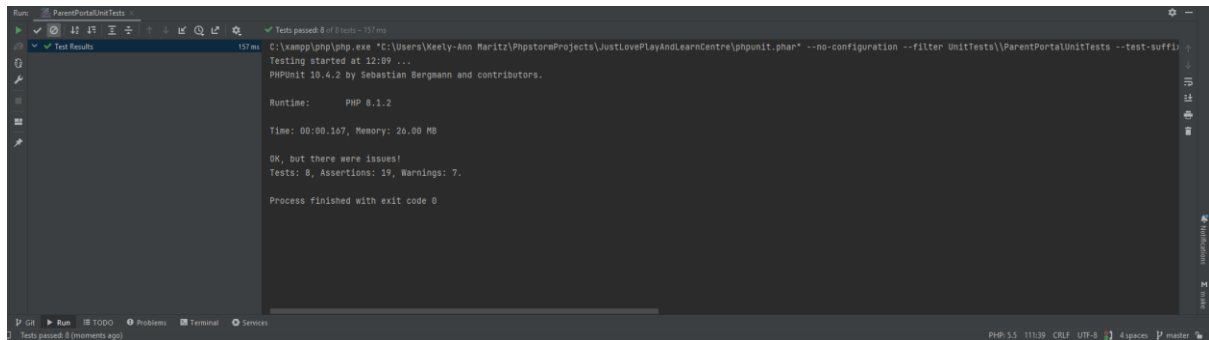
Runtime: PHP 8.1.2

Time: 00:00.023, Memory: 26.00 MB

OK, but there were issues!

Tests: 7, Assertions: 30, Warnings: 14.

Process finished with exit code 0



Run ParentPortalUnitTests

Test passed: 0 of 0 tests - 157ms

Test Results

C:\xampp\php\php.exe "C:\Users\Keely-Ann Maritz\PhstormProjects\JustLovePlayAndLearnCentre\phpunit.phar" --no-configuration --filter UnitTests\ParentPortalUnitTests --test-suffix

Testing started at 12:09 ...

PHPUnit 10.4.2 by Sebastian Bergmann and contributors.

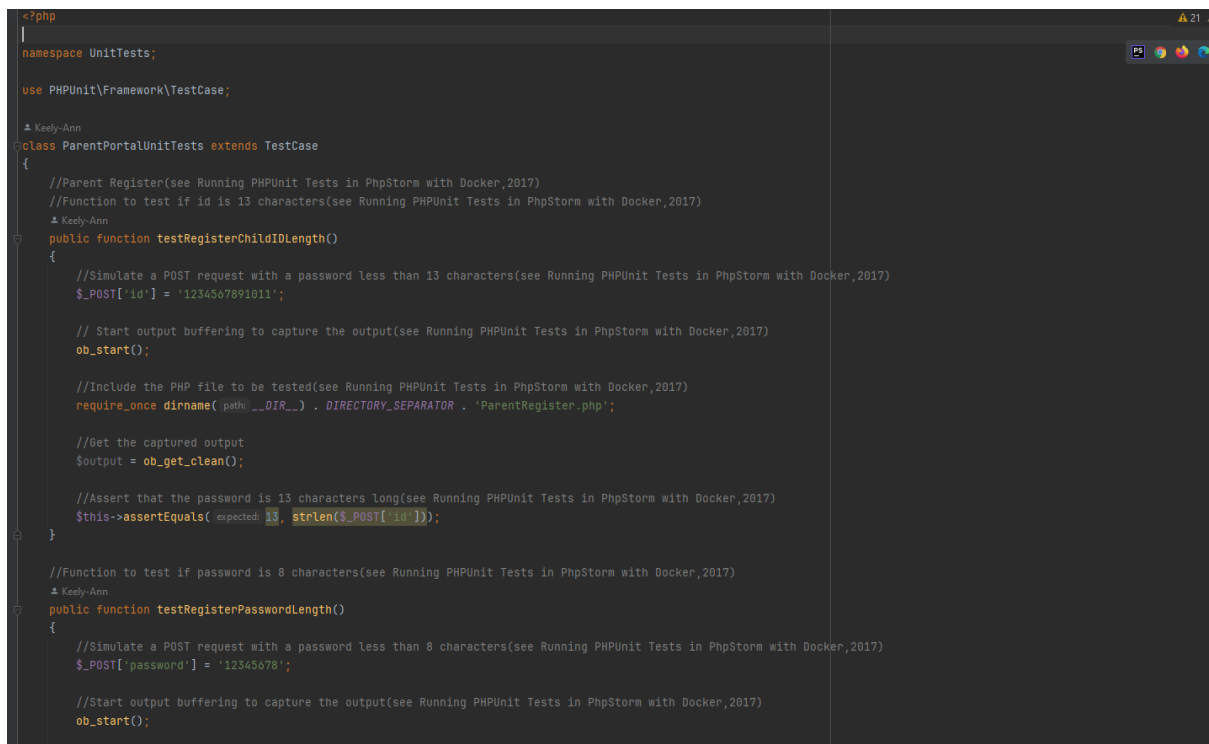
Runtime: PHP 8.1.2

Time: 00:00.167, Memory: 26.00 MB

OK, but there were issues!

Tests: 0, Assertions: 19, Warnings: 7.

Process finished with exit code 0



```
<?php
namespace UnitTests;

use PHPUnit\Framework\TestCase;

class ParentPortalUnitTests extends TestCase
{
    //Parent Register(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    //Function to test if id is 13 characters(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    public function testRegisterChildIDLength()
    {
        //Simulate a POST request with a password less than 13 characters(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $_POST['id'] = '1234567891011';

        // Start output buffering to capture the output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        ob_start();

        //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        require_once dirname(__DIR__) . DIRECTORY_SEPARATOR . 'ParentRegister.php';

        //Get the captured output
        $output = ob_get_clean();

        //Assert that the password is 13 characters long(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $this->assertEquals( expected: 13, strlen($_POST['id']));
    }

    //Function to test if password is 8 characters(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    public function testRegisterPasswordLength()
    {
        //Simulate a POST request with a password less than 8 characters(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $_POST['password'] = '12345678';

        //Start output buffering to capture the output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        ob_start();
    }
}
```

```
//Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
require_once dirname( __DIR__ ) . DIRECTORY_SEPARATOR . 'ParentRegister.php';

//Get the captured output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
$output = ob_get_clean();

//Assert that the password is 8 characters long(see Running PHPUnit Tests in PhpStorm with Docker,2017)
$this->assertEquals( expected: 8, strlen($_POST['password']));
}

//Parent Login(see Running PHPUnit Tests in PhpStorm with Docker,2017)
//Function to test if id is 13 characters(see Running PHPUnit Tests in PhpStorm with Docker,2017)
Keely-Ann
public function testLoginChildIDLength()
{
    //Simulate a POST request with a password less than 13 characters(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $_POST['id'] = '1234567891011';

    // Start output buffering to capture the output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    ob_start();

    //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    require_once dirname( __DIR__ ) . DIRECTORY_SEPARATOR . 'ParentLogin.php';

    //Get the captured output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $output = ob_get_clean();

    //Assert that the password is 13 characters long(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertEquals( expected: 13, strlen($_POST['id']));
}

//Parent Log Ticket(see Running PHPUnit Tests in PhpStorm with Docker,2017)
//Function to test if the data inputted is valid(see Running PHPUnit Tests in PhpStorm with Docker,2017)
Keely-Ann
public function testTicketDataFormat()
{
    //Sample ticket data(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $data = [
        'parent_first_name' => 'Jade',
    ];
```

```
        'parent_last_name' => 'Willard',
        'parent_email' => 'test@example.com',
        'parent_phone' => '0722334511',
        'query' => 'Do not listen',
    ];

    //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    include 'ParentLogTicket.php';

    //Assert that parent_first_name is a non-empty string(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['parent_first_name']);
    $this->assertNotEmpty($data['parent_first_name']);

    //Assert that parent_last_name is a non-empty string(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['parent_last_name']);
    $this->assertNotEmpty($data['parent_last_name']);

    //Assert that parent_email is a non-empty string(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['parent_email']);
    $this->assertNotEmpty($data['parent_email']);

    //Assert that parent_phone is a non-empty string(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['parent_phone']);
    $this->assertNotEmpty($data['parent_phone']);

    //Assert that query is a non-empty string(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['query']);
    $this->assertNotEmpty($data['query']);
}

//Functions to test if users insert a valid phone number(see Running PHPUnit Tests in PhpStorm with Docker,2017)

/**
 * Test case to validate a correct phone number.
 *
 * @dataProvider validPhoneNumbers
 */
Keely-Ann
```

```

public function testValidPhoneNumber($phoneNumber)
{
    //Set the POST data with a valid phone number(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $_POST['parent_phone'] = $phoneNumber;

    //Execute the phone number validation function(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $result = $this->validatePhoneNumber();

    //Assert that the validation result is true for a valid phone number(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertTrue($result);
}

//Helper function to validate the phone number by including the PHP file(see Running PHPUnit Tests in PhpStorm with Docker,2017)
! usage  ▲ Keely-Ann
public function validatePhoneNumber()
{
    //Start output buffering to capture the included file's output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    ob_start();

    //Include the PHP file containing the phone number validation logic(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    include 'ParentLogTicket.php';

    //Clean the output buffer(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    ob_end_clean();

    //Return true if the output is empty (validation passes), false otherwise(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    return ob_get_contents() === '';
}

//Data provider for valid phone numbers(see Running PHPUnit Tests in PhpStorm with Docker,2017)
! usage  ▲ Keely-Ann
public function validPhoneNumbers()
{
    return [
        ['1234567890'],
        ['9876543210'],
    ];
}

```

```

//Tour(see Running PHPUnit Tests in PhpStorm with Docker,2017)
//Function to check for valid time entry(see Running PHPUnit Tests in PhpStorm with Docker,2017)
! usage  ▲ Keely-Ann
public function testIsValid()
{
    //Simulate a POST request with a valid time(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $_POST['time'] = '12:00:00';

    //Start output buffering to capture the output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    ob_start();

    //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    require_once dirname(__DIR__) . DIRECTORY_SEPARATOR . 'Tour.php';

    //Get the captured output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $output = ob_get_clean();

    //Test a valid time(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $result = isValid($_POST['time']);

    //Assert that the time is valid(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertTrue($result);

    //Simulate a POST request with an invalid time(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $_POST['time'] = '07:00:00';

    //Start output buffering to capture the output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    ob_start();

    //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    require_once dirname(__DIR__) . DIRECTORY_SEPARATOR . 'Tour.php';

    //Get the captured output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $output = ob_get_clean();

    //Test an invalid time(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $result = isValid($_POST['time']);

    //Assert that the time is invalid(see Running PHPUnit Tests in PhpStorm with Docker,2017)
}

```

```

        //Assert that the time is invalid(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $this->assertFalse($result);
    }

    //Function to test if email is valid(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    ± Keely-Ann
    public function testIsValidEmail()
    {
        //Simulate a POST request with a valid email(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $_POST['email'] = 'test@example.com';

        //Start output buffering to capture the output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        ob_start();

        //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        require_once dirname(__DIR__) . DIRECTORY_SEPARATOR . 'Tour.php';

        //Get the captured output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $output = ob_get_clean();

        //Test a valid email(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $result = filter_var($_POST['email'], FILTER_VALIDATE_EMAIL);

        //Assert that the email is valid(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $this->assertTrue( condition: $result !== false);

        //Simulate a POST request with an invalid email(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $_POST['email'] = 'invalid_email';

        //Start output buffering to capture the output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        ob_start();

        //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        require_once dirname(__DIR__) . DIRECTORY_SEPARATOR . 'Tour.php';

        //Get the captured output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $output = ob_get_clean();

        //Test an invalid email(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $result = filter_var($_POST['email'], FILTER_VALIDATE_EMAIL);
    }

```

```

public function testIsValidEmail()
{
    //Simulate a POST request with a valid email(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $_POST['email'] = 'test@example.com';

    //Start output buffering to capture the output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    ob_start();

    //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    require_once dirname(__DIR__) . DIRECTORY_SEPARATOR . 'Tour.php';

    //Get the captured output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $output = ob_get_clean();

    //Test a valid email(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $result = filter_var($_POST['email'], FILTER_VALIDATE_EMAIL);

    //Assert that the email is valid(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertTrue( condition: $result !== false);

    //Simulate a POST request with an invalid email(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $_POST['email'] = 'invalid_email';

    //Start output buffering to capture the output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    ob_start();

    //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    require_once dirname(__DIR__) . DIRECTORY_SEPARATOR . 'Tour.php';

    //Get the captured output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $output = ob_get_clean();

    //Test an invalid email(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $result = filter_var($_POST['email'], FILTER_VALIDATE_EMAIL);

    //Assert that the email is invalid(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertFalse( condition: $result !== false);
}

```

```
<?php

namespace UnitTests;

use PHPUnit\Framework\TestCase;

class AdminPortalUnitTests extends TestCase
{
    //Admin Add(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    //Function to test if admin password is 8 characters(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    public function testAdminPasswordLength()
    {
        //Simulate a POST request with an admin password less than 8 characters(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $_POST['password'] = '12345678';

        //Start output buffering to capture the output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        ob_start();

        //Include the PHP file to be tested only if it hasn't been included before(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        include 'AddAdmin.php';

        //Get the captured output(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $output = ob_get_clean();

        //Assert that the admin password is 8 characters long(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $this->assertEquals(8, strlen($_POST['password']));
    }

    //Admin Events(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    //Function to test if the data inputted is valid(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    public function testEventDataFormat()
    {
        //Sample event data(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $data = [
            'event_name' => 'Test Event',
            'event_description' => 'Description of the event',
            'events_file' => 'path/to/image.jpg',
        ];

        //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        include 'AdminEvents.php';

        //Assert that event_name is a non-empty string(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $this->assertIsString($data['event_name']);
        $this->assertNotEmpty($data['event_name']);

        //Assert that event_description is a non-empty string(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $this->assertIsString($data['event_description']);
        $this->assertNotEmpty($data['event_description']);

        //Assert that events_file is a non-empty string and ends with .jpg or .jpeg or .png(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $this->assertIsString($data['events_file']);
        $this->assertNotEmpty($data['events_file']);
        $this->assertMatchesRegularExpression('/\.(jpg|jpeg|png)$/', $data['events_file']);

        //Assert that event_date is in the correct format (YYYY-MM-DD)(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $this->assertIsString($data['event_date']);
        $this->assertNotEmpty($data['event_date']);

        //Use the global DateTime class, not in a namespace(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $this->assertTrue(
            condition: \DateTime::createFromFormat('Y-m-d', $data['event_date']) !== false,
            message: 'Invalid date format. Expected format: YYYY-MM-DD'
        );
    }

    //Function to test if the uploading of images in admin events is in the correct format(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    public function testHandleEventsUpload()
    {
        //Simulate a POST request with a valid file(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $_FILES['events_file'] = [
            'name' => 'test.jpg',
            'type' => 'image/jpeg',
            'tmp_name' => '/tmp/test.tmp',
        ];
    }
}
```

```
        'events_file' => 'path/to/image.jpg',
        'event_date' => '2023-01-01',
    ];

    //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    include 'AdminEvents.php';

    //Assert that event_name is a non-empty string(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['event_name']);
    $this->assertNotEmpty($data['event_name']);

    //Assert that event_description is a non-empty string(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['event_description']);
    $this->assertNotEmpty($data['event_description']);

    //Assert that events_file is a non-empty string and ends with .jpg or .jpeg or .png(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['events_file']);
    $this->assertNotEmpty($data['events_file']);
    $this->assertMatchesRegularExpression('/\.(jpg|jpeg|png)$/', $data['events_file']);

    //Assert that event_date is in the correct format (YYYY-MM-DD)(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['event_date']);
    $this->assertNotEmpty($data['event_date']);

    //Use the global DateTime class, not in a namespace(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertTrue(
        condition: \DateTime::createFromFormat('Y-m-d', $data['event_date']) !== false,
        message: 'Invalid date format. Expected format: YYYY-MM-DD'
    );
}

//Function to test if the uploading of images in admin events is in the correct format(see Running PHPUnit Tests in PhpStorm with Docker,2017)
public function testHandleEventsUpload()
{
    //Simulate a POST request with a valid file(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $_FILES['events_file'] = [
        'name' => 'test.jpg',
        'type' => 'image/jpeg',
        'tmp_name' => '/tmp/test.tmp',
    ];
}
```

```

        'tmp_name' => '/tmp/test.tmp',
        'error' => UPLOAD_ERR_OK,
        'size' => 123,
    ];

    //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    include 'AdminEvents.php';

    //Use an anonymous function to simulate the behavior of handleEventsUpload()(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $handleEventsUpload = function () {
        //Check if the uploaded file is of an allowed image format(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $allowedFormats = ['image/jpg', 'image/jpeg', 'image/png'];

        //Get the file type from the simulated upload(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $fileType = $_FILES['events_file']['type'];

        //Check if the uploaded file type is in the allowed formats(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        if (in_array($fileType, $allowedFormats)) {
            return 'Uploads/test.jpg';
        } else {
            //Handle the case when the file type is not allowed(see Running PHPUnit Tests in PhpStorm with Docker,2017)
            return false;
        }
    };

    //Call the simulated handleEventsUpload function(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $eventPath = $handleEventsUpload();

    //Set the expected path based on the simulated file upload(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $expectedPath = 'Uploads/test.jpg';

    //Assert that the $eventPath matches the expected path(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertEquals($expectedPath, $eventPath);

    //Additional assertion to check if the file type is valid(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertNotEquals( expected: false, $eventPath, message: 'Invalid image format. Allowed formats are: jpg, jpeg, png');
}

//Admin Resources(see Running PHPUnit Tests in PhpStorm with Docker,2017)
//Function to test if the data inputted is valid(see Running PHPUnit Tests in PhpStorm with Docker,2017)

```

```

//Admin Resources(see Running PHPUnit Tests in PhpStorm with Docker,2017)
//Function to test if the data inputted is valid(see Running PHPUnit Tests in PhpStorm with Docker,2017)
Keely-Ann
public function testResourceDataFormat()
{
    //Sample resource data(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $data = [
        'name_of_resource' => 'Test Resource',
        'resource_name' => 'Resource name',
        'resource_date' => '2023-01-01',
        'resource_file' => 'path/to/image.jpg',
    ];

    //Include the PHP file to be tested (if not already included)(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    include 'AdminResources.php';

    //Assert that name_of_resource is a non-empty string(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['name_of_resource']);
    $this->assertNotEmpty($data['name_of_resource']);

    //Assert that resource_name is a non-empty string(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['resource_name']);
    $this->assertNotEmpty($data['resource_name']);

    //Assert that resource_date is in the correct format (YYYY-MM-DD)(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['resource_date']);
    $this->assertNotEmpty($data['resource_date']);

    //Assert that resource_file is a non-empty string and ends with .jpg or .jpeg or .png(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['resource_file']);
    $this->assertNotEmpty($data['resource_file']);
    $this->assertMatchesRegularExpression( pattern: '/\.(jpg|jpeg|png)$/i', $data['resource_file']);

    //Use the global DateTime class, not in a namespace(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertTrue(
        condition: \DateTime::createFromFormat( format: 'Y-m-d', $data['resource_date'] ) != false,
        message: 'Invalid date format. Expected format: YYYY-MM-DD'
    );
}

```

```
//Function to test if the uploading of resources in admin resources is in the correct format(see Running PHPUnit Tests in PhpStorm with Docker,2017)
Keedy-Ann
public function testHandleResourcesUpload()
{
    //Simulate a POST request with a valid file(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $_FILES['resource_file'] = [
        'name' => 'test.jpg',
        'type' => 'image/jpeg',
        'tmp_name' => '/tmp/test.tmp',
        'error' => UPLOAD_ERR_OK,
        'size' => 123,
    ];

    //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    include 'AdminResources.php';

    //Use an anonymous function to simulate the behavior of handleResourcesUpload()(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $handleResourcesUpload = function () {
        //Check if the uploaded file is of an allowed image format(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $allowedFormats = ['image/jpeg', 'image/jpg', 'image/png'];

        //Get the file type from the simulated upload(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $fileType = $_FILES['resource_file']['type'];

        //Check if the uploaded file type is in the allowed formats(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        if (in_array($fileType, $allowedFormats)) {
            return 'Uploads/test.jpg'; //Adjust the return value based on your resource file handling logic(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        } else {
            //Handle the case when the file type is not allowed(see Running PHPUnit Tests in PhpStorm with Docker,2017)
            return false;
        }
    };

    //Call the simulated handleResourcesUpload function(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $resourcePath = $handleResourcesUpload();

    //Set the expected path based on the simulated file upload(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $expectedPath = 'Uploads/test.jpg';

    //Assert that the $resourcePath matches the expected path(see Running PHPUnit Tests in PhpStorm with Docker,2017)

```

```
//Assert that the $resourcePath matches the expected path(see Running PHPUnit Tests in PhpStorm with Docker,2017)
$this->assertEquals($expectedPath, $resourcePath);

//Additional assertion to check if the file type is valid(see Running PHPUnit Tests in PhpStorm with Docker,2017)
$this->assertNotEquals( expected: false, $resourcePath, message: 'Invalid image format. Allowed formats are: jpg, jpeg, png');
}

//Admin Images(see Running PHPUnit Tests in PhpStorm with Docker,2017)
Keedy-Ann
public function testImageDataFormat()
{
    //Sample event data(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $data = [
        'c_name' => 'Test Image',
        'image_description' => 'Description of the event',
        'image_date' => '2023-01-01',
        'image_file' => 'path/to/image.jpg',
    ];

    //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    include 'AdminImages.php';

    //Assert that c_name is a non-empty string(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['c_name']);
    $this->assertNotEmpty($data['c_name']);

    //Assert that image_description is a non-empty string(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['image_description']);
    $this->assertNotEmpty($data['image_description']);

    //Assert that image_date is in the correct format (YYYY-MM-DD)(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['image_date']);
    $this->assertNotEmpty($data['image_date']);

    //Assert that image_file is a non-empty string and ends with .jpg or .jpeg or .png(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['image_file']);
    $this->assertNotEmpty($data['image_file']);
    $this->assertMatchesRegularExpression( pattern: '/\.(jpg|jpeg|png)$/i', $data['image_file']);
}

//Function to test if the uploading of images in admin images is in the correct format(see Running PHPUnit Tests in PhpStorm with Docker,2017)

```

```

//Function to test if the uploading of images in admin images is in the correct format(see Running PHPUnit Tests in PhpStorm with Docker,2017)
Keely-Ann
public function testHandleImagesUpload()
{
    //Simulate a POST request with a valid file(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $_FILES['image_file'] = [
        'name' => 'test.jpg',
        'type' => 'image/jpeg',
        'tmp_name' => '/tmp/test.tmp',
        'error' => UPLOAD_ERR_OK,
        'size' => 123,
    ];

    //Include the PHP file to be tested(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    include 'AdminImages.php';
    //Use an anonymous function to simulate the behavior of handleImageUpload()(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $handleImageUpload = function () {
        //Check if the uploaded file is of an allowed image format(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $allowedFormats = ['image/jpg', 'image/jpeg', 'image/png'];
        //Get the file type from the simulated upload(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        $fileType = $_FILES['image_file']['type'];
        //Check if the uploaded file type is in the allowed formats(see Running PHPUnit Tests in PhpStorm with Docker,2017)
        if (in_array($fileType, $allowedFormats)) {
            return 'Uploads/test.jpg';
        } else {
            //Handle the case when the file type is not allowed(see Running PHPUnit Tests in PhpStorm with Docker,2017)
            return false;
        }
    };

    //Call the simulated handleEventsUpload function(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $imagePath = $handleImageUpload();
    //Set the expected path based on the simulated file upload(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $expectedPath = 'Uploads/test.jpg';
    //Assert that the $eventPath matches the expected path(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertEquals($expectedPath, $imagePath);
    //Additional assertion to check if the file type is valid(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertNotEquals( expected: false, $imagePath, message: 'Invalid image format. Allowed formats are: jpg, jpeg, png');
}
}

```

## 5 LOAD TESTING

The image is a composite of two screenshots. The left screenshot shows a web browser at the URL 'justloveplayandlearncentre.co.za/Tour.php'. The website has a header with the 'JUST LOVE' logo and a navigation bar with links: Home, About Us, Extra Murals, Application, Gallery, Schedule Tour, and Contact Us. The main content area features a 'SCHEDULE TOUR' heading and a welcome message: 'Welcome to Just Love, Learn, and Play's Tour Page! We are delighted to invite you to experience our learning environment firsthand. Scheduling a tour provides you with the opportunity to explore our facilities, meet our dedicated staff, and witness the engaging activities that make our center unique. During the tour, you'll gain valuable insights into our curriculum, facilities, and the warm atmosphere we cultivate for your child's growth and development.' Below this is a form with fields for 'Email', 'Name', 'Date to have a scheduled tour' (with a date picker), and 'Preferred time of scheduled tour' (with a time picker), followed by a 'Schedule Tour' button. The right screenshot shows the Chrome DevTools Performance monitor overlay. It displays a timeline of performance metrics over time. The left sidebar lists metrics: CPU usage (0.1%), JS heap size (2.5 MB), DOM Nodes (1,145), JS event listeners (18), Documents (7), Document Frames (11), Layouts / sec (0), and Style recalcs / sec (0). The main panel shows corresponding charts for each metric, with CPU usage, JS heap size, and DOM Nodes showing significant activity during the tour period.

## 6 AUTO DEPLOYMENT

The screenshot shows the cPanel interface with the 'Git™ Version Control' tool selected. The left sidebar contains the cPanel logo and links to 'Tools', 'WordPress Manager by Softaculous', and 'Softaculous'. The main content area has a search bar at the top right with the text 'Search Tools (/)' and icons for help and close. Below the search bar is the title 'Git™ Version Control'. A yellow warning box states: 'Warning: Your system administrator **must** enable shell access to allow you to view clone URLs.' Below the warning, a paragraph explains that Git is used to maintain a set of files and track changes. A 'Create' button is visible. A search bar with the text 'Search' and a magnifying glass icon is present. To the right of the search bar is a 'Page Size' dropdown set to '10' and navigation buttons '<<', '<', '>', and '>>'. Below these is the text 'Displaying 1 to 1 out of 1 item'. A table with two columns, 'Repository' and 'Repository Path', contains one entry: 'JustLovePlayAndLearnCentre' with the path '/home/justlykc/repositories/JustLove'. To the right of this entry are three buttons: 'Manage', 'History', and 'Remove'. At the bottom left is the cPanel logo and version '110.0.15'. At the bottom right are links for 'Home', 'Trademarks', 'Privacy Policy', 'Documentation', and 'Give Feedback'.

Search Tools (/)

### Git™ Version Control

**Warning:** Your system administrator **must** enable shell access to allow you to view clone URLs.

Create and manage Git™ repositories. You can use Git to maintain any set of files and track the history of changes from multiple editors (version control). For more information, read our [documentation](#).

Create

Search

Page Size: 10 << < > >>

Displaying 1 to 1 out of 1 item

Repository	Repository Path
> JustLovePlayAndLearnCentre	/home/justlykc/repositories/JustLove

Manage History Remove

cPanel 110.0.15

[Home](#) [Trademarks](#) [Privacy Policy](#) [Documentation](#) [Give Feedback](#)

## 7 REFERENCE LIST

Damodaran,S.N.2013.To generate pdf download using tcpdf,14 August 2014.[Online].Available at: <https://stackoverflow.com/questions/18223743/to-generate-pdf-download-using-tcpdf> [Accessed 17 November 2023].

Gosselin, D., Kokoska, D. and Easterbrooks, R. 2011. PHP Programming with MySQL. 2nd edition. Boston, USA. Course Technology. ISBN: 978-0-538-46814-5. (PM1)

Guo,Y. N/A.E Signature using Canvas.[Online].Available at: <https://codepen.io/yguo/pen/OyYGxQ> [Accessed 30 October 2023].

PHP Form Submit To Send Email Contact Form Submit to Email Using PHP.2021. YouTube video, added by Invention Tricks. [Online]. Available at: <https://www.youtube.com/watch?v=9zfZhsV4NF0> [Accessed 17 November 2023].

Running PHPUnit Tests in PhpStorm with Docker. 2017.YouTube video, added by JetBrains.[Online].Available at: <https://www.youtube.com/watch?v=NztrolqNBZA> [Accessed 16 November 2023].

W3Schools.2023. W3Schools. [Online]. Available at: <https://www.w3schools.com/> [Accessed 17 November 2023].