

PROJECT

Roberto Boysen (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

The screenshot shows the Azure DevOps Boards interface for the JLPALC Team. The left sidebar includes options like Overview, Boards, Work items, Boards, Backlog, Sprints, Queries, Delivery Plan, Analytics views, Repos, Pipelines, Test Plans, and Artifacts. The main area displays a 'Stories' board with the following columns: Not Started, Started, Testing, Review, and Complete. There are 14 stories listed, each with a summary, assignee (Sally Anne Marks or Robert Bosman), and status (Closed). The stories are:

- 139 As a developer I want to ensure Accurate User Data Management through PHP Integration in the Admin User Page (Sally Anne Marks, Closed, 448)
- 140 As a developer I want to remove User and Item from a Single Entity View Page (Robert Bosman, Closed, 448)
- 158 As a developer I want to implement Image Uploading with a Custom Frame in Image Page (Sally Anne Marks, Closed, 448)
- 164 As a developer I want to ensure Accurate Data Management with Database API integration (Robert Bosman, Closed, 448)
- 172 As a developer I want to ensure Accurate Tenant Data Management with Shared API integration (Sally Anne Marks, Closed, 448)
- 221 As a developer I want to enable Accurate Access and Authorization with a Shared Approach (Sally Anne Marks, Closed, 448)
- 222 As a developer I want to Improve Tenant User Information and Functional Tenant Home Page (Robert Bosman, Closed, 448)
- 223 As a developer I want to enhance Insert, Update and Delete Operations with a Shared Approach and Value Binding (Sally Anne Marks, Closed, 448)
- 224 As a developer I want to Improve Tenant Access and Authorization with Shared API (Sally Anne Marks, Closed, 747)
- 225 As a developer I want to ensure Secure Registration with Secure Data Handling and Error Reporting (Sally Anne Marks, Closed, 448)
- 226 As a developer I want to modify Public Functionality with a Shared Approach (Sally Anne Marks, Closed, 448)
- 46 As a developer I want to Develop an Interactive Session Page with Iterative Design (Sally Anne Marks, Closed, 448)
- 126 As a developer I want to ensure Accurate Cloud Management with History Database Interaction and Error Handling (Robert Bosman, Closed, 345)
- 243 As a developer I want to enhance Image Storage and Reduce Redundancy and/or Redundancy (Sally Anne Marks, Closed, 448)
- 252 As a developer I want to enable Accurate User Management with Database Interaction and Error Handling (Robert Bosman, Closed, 448)

Azure DevOps ST1008542B / JLPALC / Boards / Boards

JLPALC Team

Overview Boards Work items Boards Backlogs Sprints Queries Delivery Plans Analytics views Repos Pipelines Test Plans Artifacts

Project settings

Board Analytics View as Backlog

New Active 0/5 Resolved 0/5 Closed 0/5

+ New item

239 Event Participation Facilitation
Kelly-Ann Maritz State: Closed 2/2

278 Secure User Logout Button
Kelly-Ann Maritz State: Closed 1/1

251 User-friendly Image Management
Nikita Orlhette Davids State: Closed 2/2

264 Streamlined User Communication through Efficient Ticket Submission
Roberto Booyens State: Closed 2/2

282 Intuitive Parent Registration Page
Kelly-Ann Maritz State: Closed 1/1

283 Secure Data Handling and Error Management for Parent Registration
Nikita Orlhette Davids State: Closed 1/1

297 Intuitive Parent Login Page
Kelly-Ann Maritz State: Closed 1/1

306 Secure Parent Login with Database Interaction and Error Management
Roberto Booyens State: Closed 1/1

312 Coherent Navigation and Visual Identity for Parent Portal
Nikita Orlhette Davids State: Closed 1/1

320 Informative and Functional Parent Home Page
Roberto Booyens State: Closed 1/1

329 Seamless Access to Parent Resources
Kelly-Ann Maritz State: Closed 2/2

342 Effortless Event Registration with Well-Designed Events Page
Roberto Booyens State: Closed 2/2

355 User-friendly Images Page for Image Submissions
Nikita Orlhette Davids State: Closed 4/4

360 Logout Button for User Control and Navigation
Kelly-Ann Maritz State: Closed 1/1

320 Interactive Resources Page
Nikita Orlhette Davids State: Closed 2/2

204 Informative and Secure Admin Home Page
Nikita Orlhette Davids State: Closed 1/1

202 Enhanced User Navigation and Branding in Admin Portal
Roberto Booyens State: Closed 1/1

188 User-Friendly Admin Access and Authentication
Roberto Booyens State: Closed 2/2

2 AZURE REPOS

Link: https://dev.azure.com/ST10085428/_git/JLPALC

3 AZURE PIPELINES

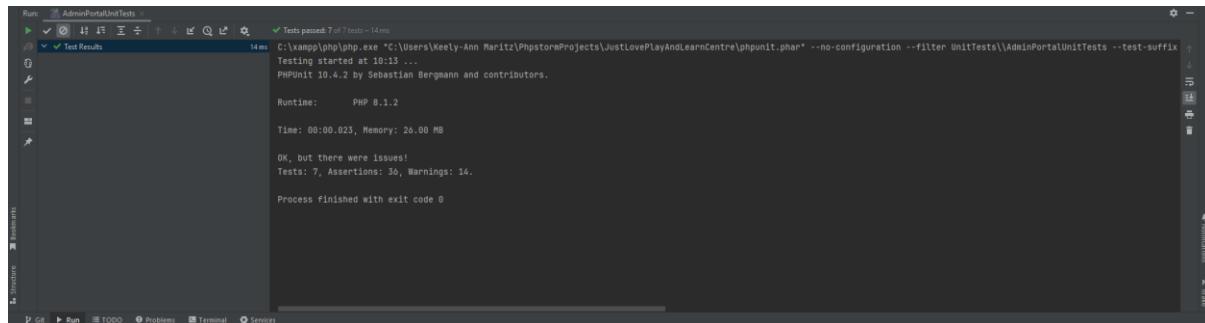
Link: https://dev.azure.com/ST10085428/JLPALC/_build

The screenshot shows the Azure DevOps Pipelines dashboard for the project JLPALC. On the left, there's a sidebar with navigation links: Overview, Boards, Repos, Pipelines (which is selected), Pipelines, Environments, Test Plans, and Artifacts. The main area is titled 'Pipelines' and shows a list of 'Recently run pipelines'. One pipeline, 'JLPALC', is listed with its last run details: '#20231127.2 - Updated README.md' (Individual CI for main branch, commit f072432f, run 7c326ea9) was completed 19s ago. A 'New pipeline' button is located at the top right.

This screenshot shows the detailed view of the JLPALC pipeline. The left sidebar includes Project settings. The main area displays a table of recent pipeline runs under the 'Runs' tab. Each row shows a green checkmark icon, the run ID (e.g., #20231127.2), the commit message (e.g., 'Updated README.md'), the branch (main), the commit hash (e.g., f072432f), and the stage status (green circle). To the right of the table, there are columns for 'Last run' (with time ago and duration), 'Edit' (button), and 'Run pipeline' (button).

Description	Stages	Last run
#20231127.2 - Updated README.md Individual CI for main branch Commit f072432f Run 7c326ea9	Green circle	19m ago 19s
#20231127.1 - Updated README.md Individual CI for main branch Commit 7c326ea9	Green circle	20m ago 11s
#20231126.1 - Merged PR 9: updated pages commit Individual CI for main branch Commit 10531eeb	Green circle	Yesterday 11s
#20231124.1 - Merged PR 8: Updated php files Individual CI for main branch Commit 25f36169	Green circle	Friday 15s
#20231123.1 - Set up CI with Azure Pipelines Individual CI for main branch Commit 88bc4ae1	Green circle	Thursday 11s

4 UNIT TESTING



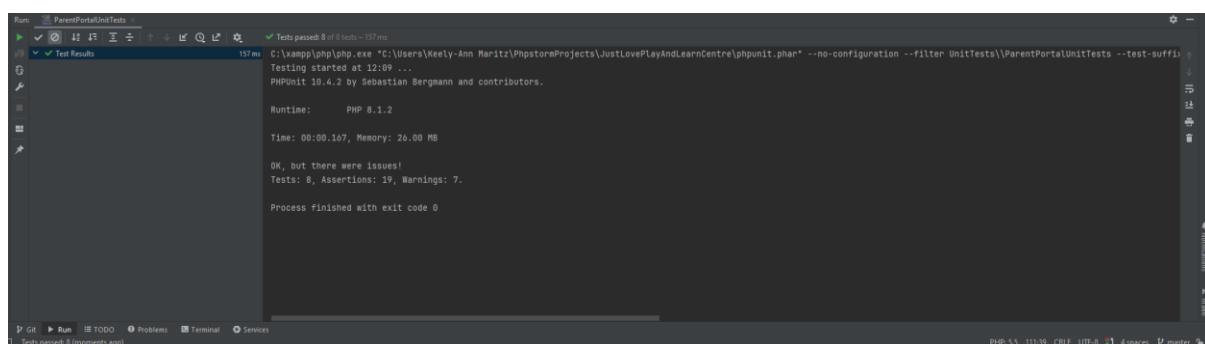
```
Run AdminPortalUnitTests
  ➜  Test Results 14ms C:\xampp\php\php.exe "C:\Users\Keely-Ann Maritz\PhpstormProjects\JustLovePlayAndLearnCentre\phpunit.phar" --no-configuration --filter UnitTests\AdminPortalUnitTests --test-suffix
  Testing started at 10:13 ...
  PHPUnit 10.4.2 by Sebastian Bergmann and contributors.

  Runtime:    PHP 8.1.2

  Time: 00:00.023, Memory: 26.00 MB

  OK, but there were issues!
  Tests: 7, Assertions: 36, Warnings: 14.

  Process finished with exit code 0
```



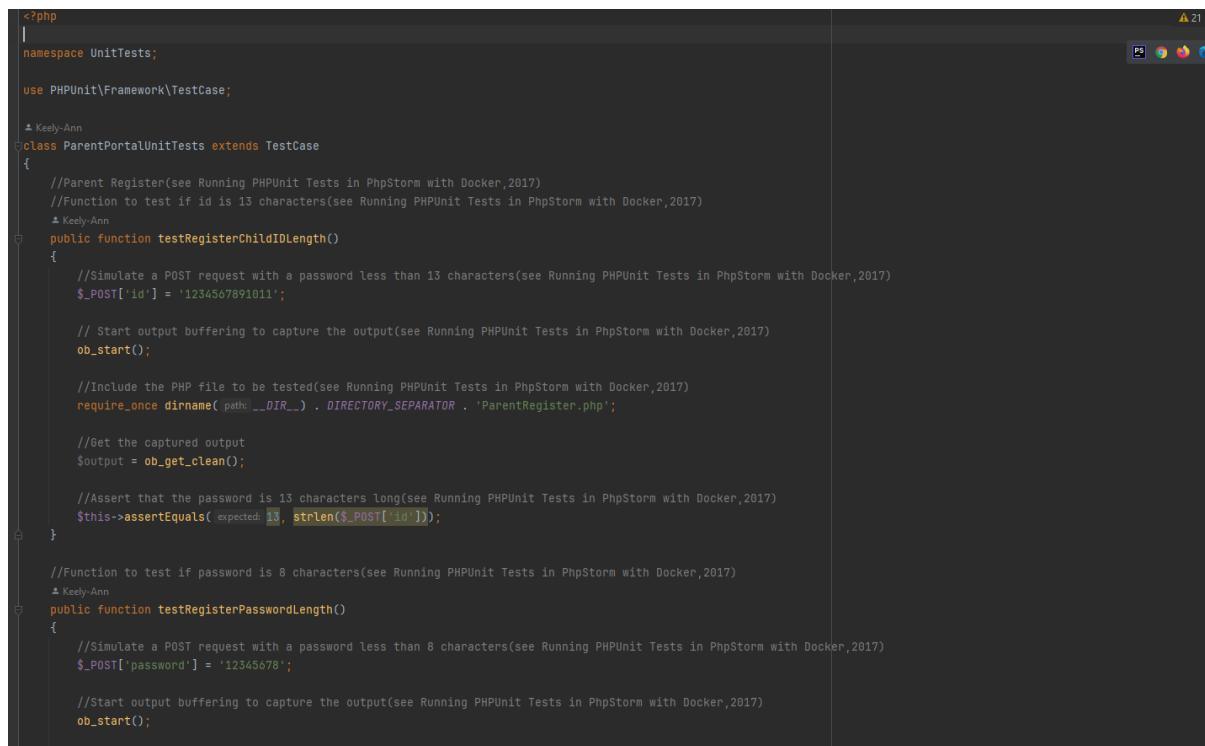
```
Run ParentPortalUnitTests
  ➜  Test Results 157ms C:\xampp\php\php.exe "C:\Users\Keely-Ann Maritz\PhpstormProjects\JustLovePlayAndLearnCentre\phpunit.phar" --no-configuration --filter UnitTests\ParentPortalUnitTests --test-suffix
  Testing started at 12:09 ...
  PHPUnit 10.4.2 by Sebastian Bergmann and contributors.

  Runtime:    PHP 8.1.2

  Time: 00:00.167, Memory: 26.00 MB

  OK, but there were issues!
  Tests: 8, 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)
▲ Keely-Ann
public function testIsTimeValid()
{
    //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( path: __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)
    $result = isTimeValid($_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( path: __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)
    $result = isTimeValid($_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)
+ Keddy-Ann
public function testIsEmailValid()
{
    //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( path: __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: 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( path: __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: FILTER_VALIDATE_EMAIL]);
}
```

```
public function testIsEmailValid()
{
    //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( path: __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: 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( path: __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: FILTER_VALIDATE_EMAIL]);
}


```

```
<?php

namespace UnitTests;

use PHPUnit\Framework\TestCase;

// Keely-Ann
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)
    // Keely-Ann
    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)
        $this->assertEquals( expected: 8, strlen($_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)
    // Keely-Ann
    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',
        ];
    }
}
```

```
'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']);
$this->assertMatchesRegularExpression( pattern: '/\.(jpg|jpeg|png)$/', $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(
    conditions: \DateTime::createFromFormat( format: '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)
// Keely-Ann
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-Anne
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']);
    $this->assertMatchesRegularExpression(pattern: '/\.(jpg|jpeg|png)$/', $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)
# Keely-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/jpg', 'image/jpeg', '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)
# Keely-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']);

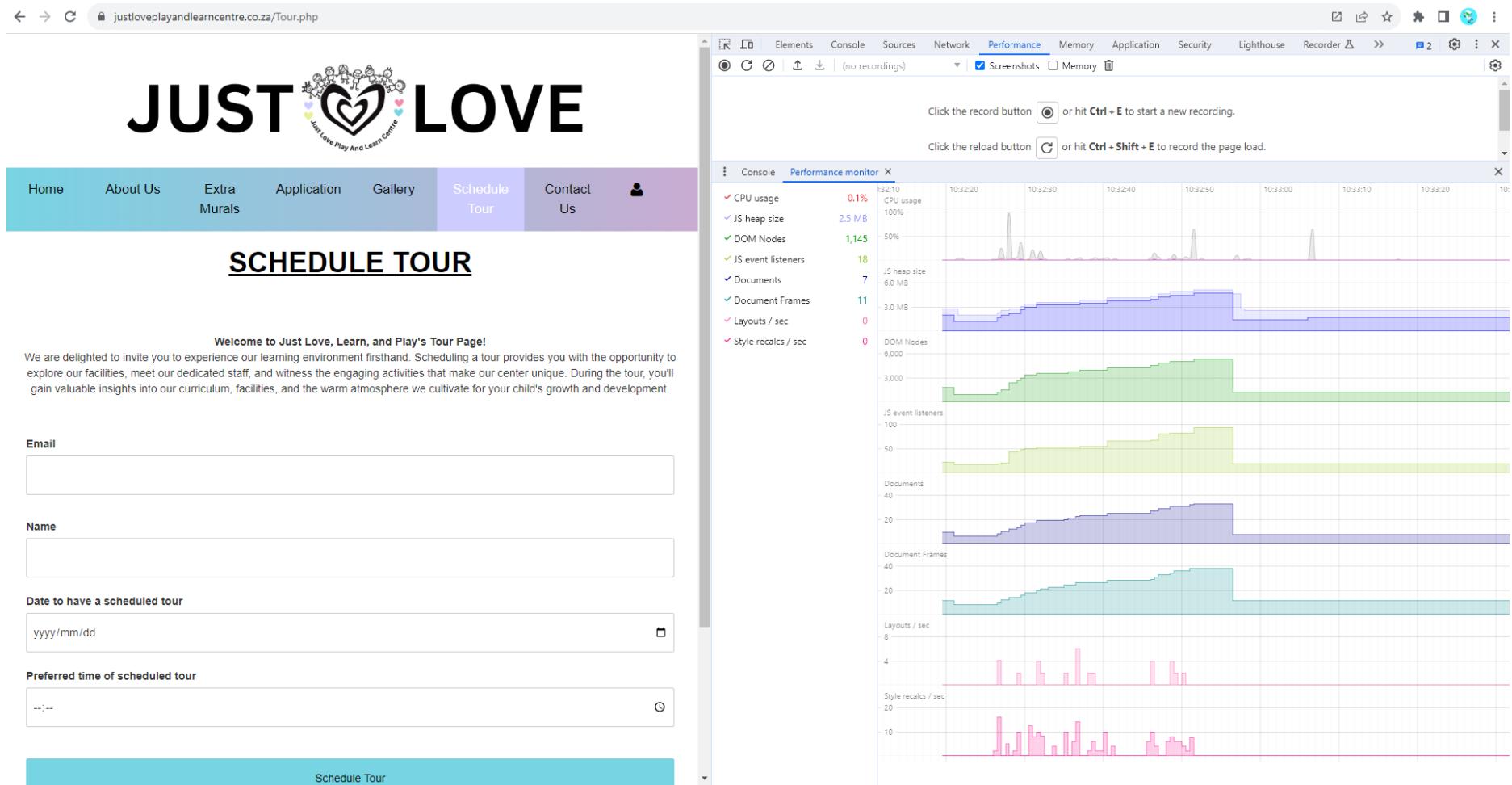
    //Assert that image_file is a non-empty string and ends with .jpg or .jpeg or .png(see Running PHPUnit Tests in PhpStorm with Docker,2017)
    $this->assertIsString($data['image_file']);
    $this->assertNotEmpty($data['image_file']);
    $this->assertMatchesRegularExpression( pattern: '/\.(jpg|jpeg|png)$/',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



6 AUTO DEPLOYMENT

The screenshot shows the cPanel Git™ Version Control interface. On the left, there's a sidebar with the cPanel logo and links for Tools, WordPress Manager by Softaculous, and a warning about shell access. The main area has a header "Git™ Version Control" with a search bar and navigation icons. A yellow warning box states: "Warning: Your system administrator **must** enable shell access to allow you to view clone URLs." Below it, a message says: "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](#)." A "Create" button is at the top right. The main content area shows a table with one item:

Repository	Repository Path	Actions
> justLovePlayAndLearnCentre	/home/justlykc/repositories/justLove	Manage History Remove

Below the table, it says "Displaying 1 to 1 out of 1 item". At the bottom, there are links for Home, Trademarks, Privacy Policy, Documentation, and 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].