PHPUnit loads configuration from the second file if the first one does not exist in the working directory. Also, you can create the bootstrap. php file by initializing autoloader and your framework’s environments:

<?php

defined('YII\_DEBUG') or define('YII\_DEBUG', true); defined('YII\_ENV') or define('YII\_ENV', 'test');

require( DIR . '/../vendor/autoload.php');

require( DIR . '/../vendor/yiisoft/yii2/Yii.php');

Finally, you can install PHPUnit via Composer (locally or globally) and use the phpunit console command in the directory with the XML configuration file.

PHPUnit scans the testing directory and finds files with the \*Test. php suffix. All your test classes must extend the PHPUnit\_Framework\_Testcase class and contain public methods with the test\* prefix like this:

class MyTest extends TestCase {

public function testSomeFunction()

{

$this->assertTrue(true);

}

}

In the body of your tests, you can use any of the existing assert\* methods:

$this->assertEqual('Alex', $model->name);

$this->assertTrue($model->validate());

$this->assertFalse($model->save());

$this->assertCount(3, $items);

$this->assertArrayHasKey('username', $model->getErrors()); $this->assertNotNull($model->author);

$this->assertInstanceOf('app\models\User', $model->author);

Also, you can override the setUp() or tearDown() methods for adding expressions that will be run before and after each test method.

For example, you can define own base TestCase class by reinitializing the Yii application:

<?php

namespace book\cart\tests;

use yii\di\Container; use yii\web\Application;

abstract class TestCase extends \PHPUnit\_Framework\_TestCase {

protected function setUp()

{

parent::setUp();

$this->mockApplication();

}

protected function tearDown()

{

$this->destroyApplication(); parent::tearDown();

} protected function mockApplication()