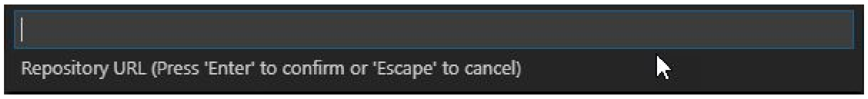
Assignment 4 - **PHP**Unit - Gamebook

We are going to create a website called gamebook, where users can share video game recommendations with friends.

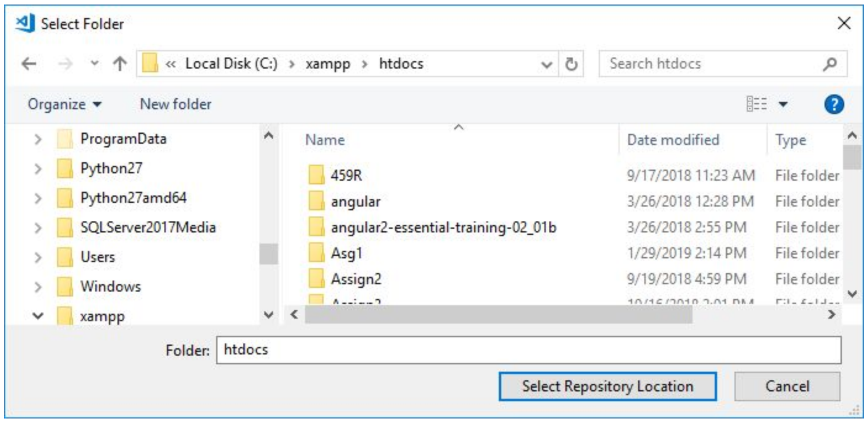
The following steps will help you create the application:

1. Create a new repository on **`GitHub`** named gamebook.
2. If you have turned in assignment 1, delete all the Magento-related files from your `htdocs` directory (if you haven't already done so).
3. Open **`VS Code`** and within vscode open the folder `htdocs`.
4. **`VS Code`** ships with a **`Git`** **S**ource **C**ontrol **M**anager (SCM) extension.
5. You can clone a **`Git`** repository with the `Git: Clone` command in the Command Palette (`Ctrl+Shift+P`). Open the Command Palette and type `Git: Clone`.
6. A little dialog will replace the Command Palette like the one shown below:

   Repository URL (Press 'Enter' to confirm or 'Escape' to cancel)

   Paste the URL of the remote repository (copied from GitHub) or type the full URL and press enter.
7. A file dialog then appears like the one shown below:

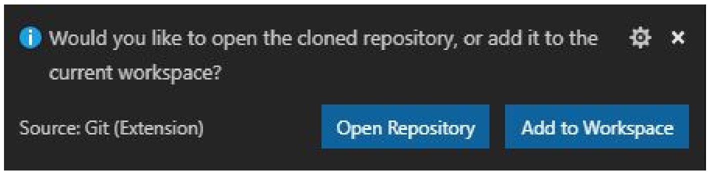
   

Name	Date modified	Type
459R	9/17/2018 11:23 AM	File folder
angular	3/26/2018 12:28 PM	File folder
angular2-essential-training-02_01b	3/26/2018 2:55 PM	File folder
Asg1	1/29/2019 2:14 PM	File folder
Assign2	9/19/2018 4:59 PM	File folder

   Folder: htdocs

   You should select the parent directory under which the cloned repository will be built. In our case, the folder is `htdocs`. Cloning the repository creates the gamebook directory.

8. Once you click the "Select Repository Location" button, then you get another dialog like the one below:


Would you like to open the cloned repository, or add it to the current workspace?

Source: Git (Extension)

Select the "Open Repository" button so the project is now open in **VS Code**.

9. Now you can press the source control icon on the far left of the **VS Code** window . Note we have cloned the master branch locally, but there is still a master branch called "`origin master`" out on GitHub.

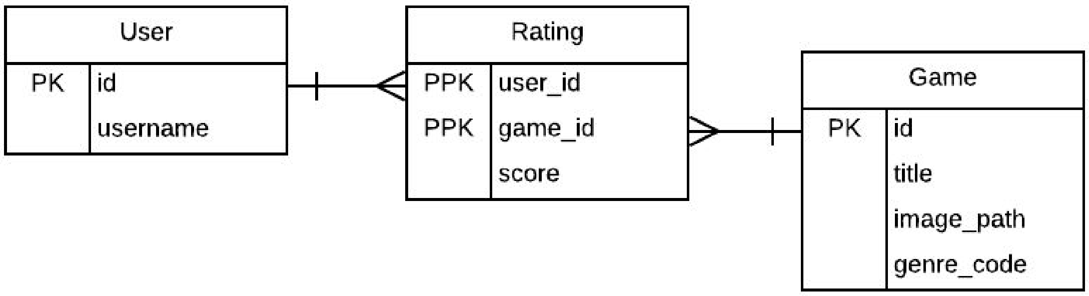
10. In **VS Code**, select View > Terminal from the File menu. In the terminal window run the following commands:

```
composer require --dev phpunit/phpunit ^8
```

Now, **PHP**Unit is installed.

11. Now, we want to exclude the `vendor` folder from being uploaded to **GitHub**. Create a file called `.gitignore` in the `gamebook` directory. Inside the file, type vendor/ and save the file.

12. You will be creating a database along with three entity classes in **PHP** that represents the following **ERD**:



13. Open up **MySQL Workbench** using the user `root` with blank password and run the **SQL** script below.

```
--drop database gamebook_test;
create database gamebook_test;
use gamebook_test;
create table gamebook_test.game (
    id int(10) unsigned auto_increment,
    title varchar(50),
    image_path varchar(255),
    genre_code varchar(255),
    primary key (id)
);

create table gamebook_test.user (
    id int(10) unsigned auto_increment,
    username varchar(50),
    primary key (id)
);

create table gamebook_test.rating (
    user_id int(10) unsigned,
    game_id int(10) unsigned,
    score tinyint(1),
    primary key (user_id, game_id)
);

insert into gamebook_test.user values(1, 'Dan');
insert into gamebook_test.game (title, image_path, genre_code) values('Game 1 - Duke
Nukem', 'images/game1.png', 'shooter');
insert into gamebook_test.game (title, image_path, genre_code) values('Game 2 - Mortal
Kombat', 'images/game2.png', 'shooter');
insert into gamebook_test.game (title, image_path, genre_code) values('Game 3 -
Assassin\'s Creed', 'images/game3.png', 'shooter');
insert into gamebook_test.game (title, image_path, genre_code) values('Game 4 - Legend
of Zelda', 'images/game4.png', 'adventure');
insert into gamebook_test.game (title, image_path, genre_code) values('Game 5 -
Fallout', 'images/game5.png', 'zombies');
insert into gamebook_test.game (title, image_path, genre_code) values('Game 6 - Mario
Brothers', 'images/game6.png', 'jumping');
insert into gamebook_test.rating values(1,1,5);

select * from game
```

14. Create three folders underneath the gamebook directory: `src` , `web` and `tests`.

15. Inside the `src` folder, create two additional folders: `Entity` and `Repository`. Inside the `web` folder, create a file named `index.php`.

16. Download the `images.zip` file from Canvas and unzip the images into the `web` folder so that the images appear inside the `images` folder.

17. In the file named `web/index.php` paste the code below into the file:

```
<?php
require __DIR__ . "/../src/Repository/GameRepository.php";

$repo = new GameRepository();
$games = $repo->findByUserId(1);

?>
<html>
<body>
<h1>Gamebook Ratings</h1>
<ul>
<?php foreach ($games as $game): ?>
   <li>
       <span class="title"><?php echo $game->getTitle() ?></span><br>
       <a href="add-rating.php?game=<?php echo $game->getId() ?>">Rate</a>
       <?php echo $game->getAverageScore() ?><br>
       <img src="<?php echo $game->getImagePath() ?>">
   </li>
<?php endforeach ?>
</ul>
</body>
</html>
```

18. Now inside the `src/Entity` folder, create a file named `Game.php` and paste the code below into that file:

```
<?php
class Game {
   protected $title;
   protected $imagePath;
   protected $ratings;
   protected $id;
   protected $genreCode;

   public function __construct($id = null)  {
       $this->id = $id;
   }
   public function getId() {
       return $this->id;
   }
   public function getGenreCode() {
       return $this->genreCode;
   }
   public function setGenreCode($value) {
       $this->genreCode = $value;
   }
   public function getAverageScore() {
       $ratings = $this->getRatings();
       $numRatings = count($ratings);

       if ($numRatings == 0) {
           return null;
       }
```

```
        $total = 0;
        foreach ($ratings as $rating) {
            $score = $rating->getScore();
            if ($score === null) {
                $numRatings--;
                continue;
            }
            $total += $score;
        }
        return $total / $numRatings;
    }
    public function toArray() {
        $array = [
            'title' => $this->getTitle(),
            'imagePath' => $this->getImagePath(),
            'ratings' => [],
        ];
        foreach ($this->getRatings() as $rating) {
            $array['ratings'][] = $rating->toArray();
        }
        return $array;
    }
    public function isRecommended($user) {
        $compatibility = $user->getGenreCompatibility($this->getGenreCode());
        return $this->getAverageScore() / 10 * $compatibility >= 3;
    }
    public function getTitle() {
        return $this->title;
    }
    public function setTitle($value) {
        $this->title = $value;
    }
    public function getImagePath() {
        if ($this->imagePath == null) {
            return 'images/placeholder.png';
        }
        return $this->imagePath;
    }
    public function setImagePath($value) {
        $this->imagePath = $value;
    }
    public function getRatings() {
        return $this->ratings;
    }
    public function setRatings($value) {
        $this->ratings = $value;
    }
}
```

19. Now, still inside the `src/Entity` folder, create a file named `User.php` and paste the code below into that file:

```
<?php
class User
{
    protected $ratings;

    public function getRatings() {
        return $this->ratings;
    }
    public function setRatings($value) {
        $this->ratings = $value;
```

```
   }

   public function findRatingsByGenre($genreCode) {
       $filteredRatings = [];
       foreach ($this->getRatings() as $rating) {
           if ($rating->getGame()->getGenreCode() == $genreCode) {
               $filteredRatings[] = $rating;
           }
       }

       return $filteredRatings;
   }
   public function getGenreCompatibility($genreCode) {
       $ratings = $this->findRatingsByGenre($genreCode);
       $numRatings = count($ratings);

       if ($numRatings == 0) {
           return null;
       }

       $total = 0;
       foreach ($ratings as $rating) {
           $score = $rating->getScore();
           if ($score === null) {
               $numRatings--;
               continue;
           }
           $total += $score;
       }
       return $total / $numRatings;
   }
}
```

20. Now, still inside the `src/Entity` folder, create a file named `Rating.php` and paste the code below into that file:

```
<?php
class Rating {

   protected $game;
   protected $user;
   protected $score;

   public function toArray() {
       return [
           'score' => $this->getScore(),
       ];
   }
   public function getGame() {
       return $this->game;
   }
   public function setGame($value) {
       $this->game = $value;
   }
   public function getUser() {
       return $this->user;
   }
```

```
    public function setUser($value) {
        $this->user = $value;
    }
    public function getScore() {
        return $this->score;
    }
    public function setScore($value)  {
        $this->score = $value;
    }
}
```

21. Now, inside the `src/Respository` folder, create a file named `GameRepository.php` and paste the code below into that file:

```
<?php

require __DIR__ . "/../Entity/Game.php";
require __DIR__ . "/../Entity/Rating.php";

class GameRepository {
    protected $pdo;
    public function __construct() {
        $this->pdo = new PDO(
            'mysql:host=localhost;dbname=gamebook_test',
            'root',
            null);
    }
    public function findById($id) {
        $statement = $this->pdo->prepare('SELECT * FROM game WHERE id = ?');
        $statement->execute([$id]);
        $game = $statement->fetchObject('Game');
        return $game;
    }
    public function saveGameRating($gameId, $userId, $score) {
        $statement = $this->pdo->prepare(
           'REPLACE INTO rating (game_id, user_id, score)
           VALUES(?, ?, ?)');
        return $statement->execute([$gameId, $userId, $score]);
    }
    public function findByUserId($id) {
        $games = [];
        $statement = $this->pdo->prepare('SELECT * FROM game');
        $statement->execute([$id]);
        $games_array = $statement->fetchAll();
        //var_dump($games_array);
        foreach($games_array as $game_array){
            $game = new Game($game_array['id']);
            $game->setTitle($game_array['title']);
            $game->setImagePath($game_array['image_path']);
            $rating = new Rating();
            $rating->setScore(4.5);
            $game->setRatings([$rating]);
            $games[] = $game;
        }
        return $games;
    }
}
```

22. Now, back to the `web` folder, create a file called `add-rating.php` and paste the following code below into the file:

```php
<?php
require __DIR__ . "/../src/Repository/GameRepository.php";

$clean = [];
$clean['game'] = filter_var($_GET['game'], FILTER_SANITIZE_NUMBER_INT);

$repo = new GameRepository();
$game = $repo->findById($clean['game']);

if ($_SERVER['REQUEST_METHOD'] == 'POST') {

   $clean['score'] = filter_var($_POST['score'], FILTER_SANITIZE_NUMBER_INT);
   $clean['user'] = 1;
   $clean['screenshot'] = filter_var($_FILES['screenshot']['tmp_name'],
       FILTER_SANITIZE_STRING);
   $repo->saveGameRating($clean['game'], 1, $clean['score']);

   $extension = pathinfo($clean['screenshot'], PATHINFO_EXTENSION);
   move_uploaded_file($clean['screenshot'],
       __DIR__.'/screenshots/'.$clean['game'].'-'.$clean['user'].'.jpg');
   header("Location: /");
   exit;
}
?>
<h1><?php echo $game->getTitle(); ?></h1>
<form method="POST" enctype="multipart/form-data">
   <input type="number" name="score" min="1" max="5">
   <input type="file" name="screenshot">
   <input type="submit" value="Save">
</form>
```

23. Still in the `web` folder, create a file called `api-games.php` and paste the following code into the file:

```php
<?php

require __DIR__ . "/../src/Repository/GameRepository.php";

$request_body = json_decode(file_get_contents('php://input'), true);

$clean = [];
$clean['user'] = filter_var($request_body['user'],
FILTER_SANITIZE_NUMBER_INT);

$repo = new GameRepository();
$games = $repo->findByUserId($clean['user']);
$data = [];
foreach ($games as $game) {
   $data[] = $game->toArray();
}
header('Content-Type: application/json');
echo json_encode([
   'data' => $data,
]);
```

24. Inside the `tests` folder, create a file called `GameTest.php` and paste in the code that follows. You will be implementing the methods in red.

```
<?php
use PHPUnit\Framework\TestCase;

require __DIR__ . "/../src/Entity/Game.php";
require __DIR__ . "/../src/Entity/Rating.php";
require __DIR__ . "/../src/Entity/User.php";

class GameTest extends TestCase {

   public function testImage_WithNull_ReturnsPlaceholder()  {
       $game = new Game();
       $game->setImagePath(null);
       $this->assertEquals('images/placeholder.png', $game->getImagePath());
   }

   public function testImage_WithPath_ReturnsPath() {

   }

   public function testAverageScore_WithoutRatings_ReturnsNull() {

   }

   public function testAverageScore_With6And8_Returns7() {

   }

   public function testAverageScore_WithNullAnd5_Returns5()
   {
       $rating1 = $this->createMock(Rating::class);
       $rating1->method('getScore')->willReturn(null);

       $rating2 = $this->createMock(Rating::class);
       $rating2->method('getScore')->willReturn(5);

       $game = $this->getMockBuilder(Game::class)
           ->setMethods(array('getRatings'))
           ->getMock();
       $game->method('getRatings')->willReturn([$rating1, $rating2]);

       $this->assertEquals(5, $game->getAverageScore());
   }

   public function testIsRecommended_WithCompatibility2AndScore10_ReturnsFalse()
   {
   }

   public function testIsRecommended_WithCompatibility10AndScore10_ReturnsTrue()
   {
   }
}
```

25. Still inside the `tests` folder, create a file called `UserTest.php` and paste the following code shown below (you will be writing the method in red):

```
<?php

use PHPUnit\Framework\TestCase;

require __DIR__ . "/../src/Entity/Game.php";
require __DIR__ . "/../src/Entity/Rating.php";
require __DIR__ . "/../src/Entity/User.php";

class UserTest extends TestCase
{
   public function testGenreCompatibility_With8And6_Returns7()
   {
       $rating1 = $this->createMock(Rating::class);
       $rating1->method('getScore')->willReturn(6);

       $rating2 = $this->createMock(Rating::class);
       $rating2->method('getScore')->willReturn(8);

       $user = $this->getMockBuilder(User::class)
           ->setMethods(array('findRatingsByGenre'))
           ->getMock();
       $user->method('findRatingsByGenre')->willReturn([$rating1, $rating2]);

       $this->assertEquals(7, $user->getGenreCompatibility('zombies'));
   }

   public function testRatingsByGenre_With1ZombieAnd1Shooter_Returns1Zombie()
   {
   }
}
```

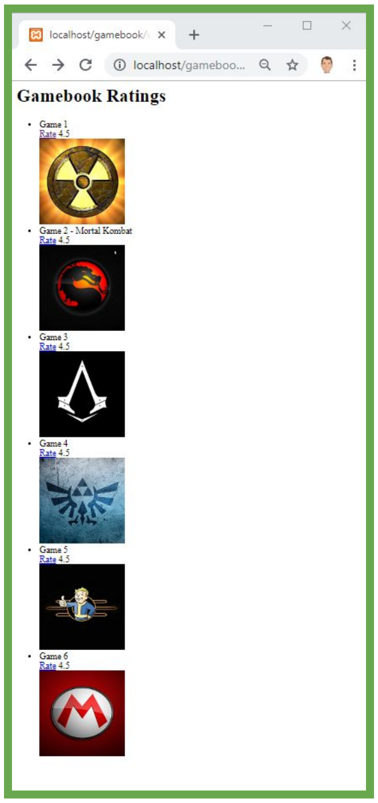
26. Run the following command to run the Game unit tests:

```
vendor/bin/phpunit --bootstrap ./vendor/autoload.php --testdox tests/GameTest.php
```

27. Run the following command to run the User unit Tests:

```
vendor/bin/phpunit --bootstrap ./vendor/autoload.php --testdox tests/UserTest.php
```

28. **FOUR** Screenshots required for submission

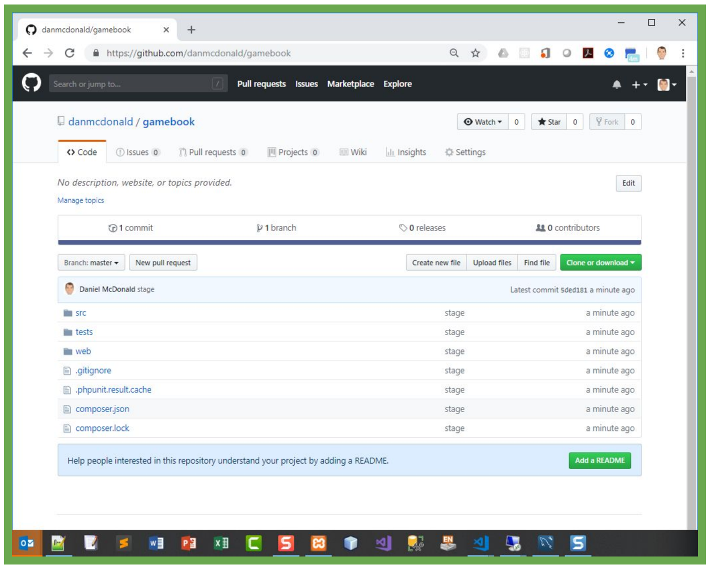


localhost/gamebook/

localhost/gameboo...

Gamebook Ratings

- Game 1
  Rate 4.5
- Game 2 - Mortal Kombat
  Rate 4.5
- Game 3
  Rate 4.5
- Game 4
  Rate 4.5
- Game 5
  Rate 4.5
- Game 6
  Rate 4.5



danmcdonald/gamebook

https://github.com/danmcdonald/gamebook

danmcdonald / gamebook

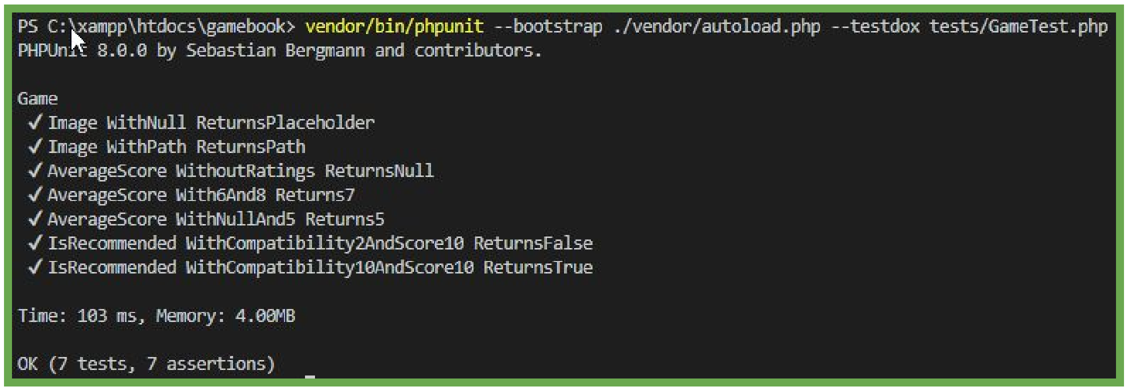
No description, website, or topics provided.

1 commit | 1 branch | 0 releases | 0 contributors

Daniel McDonald stage — Latest commit 5ded181 a minute ago

src	stage	a minute ago
tests	stage	a minute ago
web	stage	a minute ago
.gitignore	stage	a minute ago
.phpunit.result.cache	stage	a minute ago
composer.json	stage	a minute ago
composer.lock	stage	a minute ago

Help people interested in this repository understand your project by adding a README.

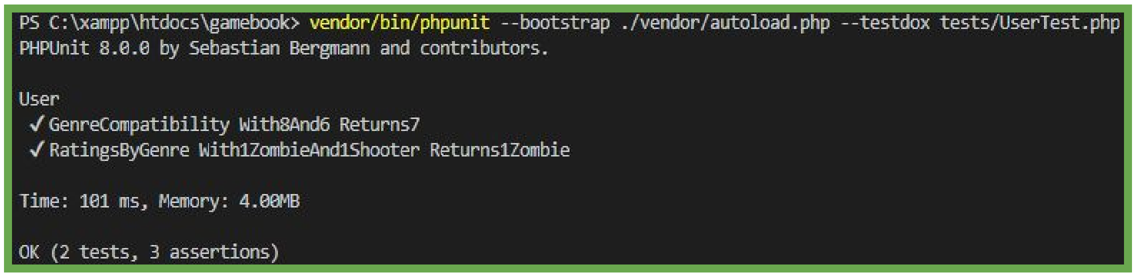


```
PS C:\xampp\htdocs\gamebook> vendor/bin/phpunit --bootstrap ./vendor/autoload.php --testdox tests/GameTest.php
PHPUnit 8.0.0 by Sebastian Bergmann and contributors.

Game
 ✓ Image WithNull ReturnsPlaceholder
 ✓ Image WithPath ReturnsPath
 ✓ AverageScore WithoutRatings ReturnsNull
 ✓ AverageScore With6And8 Returns7
 ✓ AverageScore WithNullAnd5 Returns5
 ✓ IsRecommended WithCompatibility2AndScore10 ReturnsFalse
 ✓ IsRecommended WithCompatibility10AndScore10 ReturnsTrue

Time: 103 ms, Memory: 4.00MB

OK (7 tests, 7 assertions)
```



```
PS C:\xampp\htdocs\gamebook> vendor/bin/phpunit --bootstrap ./vendor/autoload.php --testdox tests/UserTest.php
PHPUnit 8.0.0 by Sebastian Bergmann and contributors.

User
 ✓ GenreCompatibility With8And6 Returns7
 ✓ RatingsByGenre With1ZombieAnd1Shooter Returns1Zombie

Time: 101 ms, Memory: 4.00MB

OK (2 tests, 3 assertions)
```