|  |  |
| --- | --- |
| Test ID: | databaseconstructInvalidParTest1 |
| Description: | Testing the creation of a database object with invalid parameter |
| Test Steps | Step 1: Create database object $m with 'EUR' as a parameter |
| Step 2: assertInstanceOf(database::class, $m) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databaseconstrucValidParTest2 |
| Description: |  |
| Test Steps | Step 1: Create database object $m with no parameters |
| Step 2: assertInstanceOf(database::class, $m) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databasesearchForStudentsFoundTest3 |
| Description: | Test search for students funciton of database with valid query |
| Test Steps | Step 1: Create database object $db |
| Step 2: assertEquals(count($db->searchForStudents("jcarm012@fiu.edu")), 10) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databasesearchForStudentsNotFoundTest4 |
| Description: | Test search for students funciton of database with invalid query |
| Test Steps | Step 1: Create database object $db |
| Step 2: assertEquals(count($db->searchForStudents("beaver")), 0) where "beaver is not in the database |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databasegetStudentFoundTest5 |
| Description: | Test database for getStudent function with valid query to return array of student. |
| Test Steps | Step 1: Create database object $db |
| Step 2: assertEquals(count($db->getStudent("2654698")), 10) where "2654698" is found in the database. |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databasegetStudentNotFoundTest6 |
| Description: | Test database for getStudent function with invalid query to return array of student. |
| Test Steps | Step 1: Create database object $db |
| Step 2: assertEquals(count($db->getStudent("stickle")), 0) where "stickly" is not found in the database |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databaseaddStudentTest7 |
| Description: | Test database for AddStudent function with valid query to insert student record into the database. |
| Test Steps | Step 1: Create database object $db |
| Step 2: assertEquals($db->addStudent(1651248, "ricky", "roberts", "Ricke@roberts.com", 0, 1253481, "where am I", "everywhere", -5), "pass") |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databasegenPassTest8 |
| Description: | Test database function genPass with valid input |
| Test Steps | Step 1: create input string $pwd = "somePass1234" |
| Step 2: create input String $email = "some@email.com"; |
| Step 3: create database object $db |
| Step 4: create database object $db2 |
| Step 5: assertEquals($db->genPass($pwd, $email),$db2->genPass($pwd, $email)) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databaseupdateStudentPasswordTest9 |
| Description: | Test database function updateStudentPassword with valid input |
| Test Steps | Step 1: Create input string $email = "jcarm012@fiu.edu" |
| Step 2: Create database object $db |
| Step 3: Create input $pwd = genPass($email, "Jeff\_1230") |
| Step 4: ($db->updateStudentPassword($email, $pwd), "pass") |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databaseupdateStudentPasswordInValidTest10 |
| Description: | Test database function updateStudentPassword with invalid input |
| Test Steps | Step 1: Create input string $email = "jcarm012@fiu.edu" |
| Step 2: Create database object $db |
| Step 3: assertEquals($db->updateStudentPassword($email, NULL), "fail") |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databaseaddGameValidTest11 |
| Description: | Test database function genPass with valid input |
| Test Steps | Step 1: Create input $db = new database(); |
| Step 2: Create input $semester = "TestSemester"; |
| Step 3: Create input $courseID = "1234567Test"; |
| Step 4: Create input $schedule = "3:45-5 Test"; |
| Step 5: Create input $section = "abcdSectionTest"; |
| Step 6: Create input $isActive = -5; |
| Step 7: Create input $courseNumber = "TestCourseNumber"; |
| Step 8: assertEquals($db->addGame($semester,$courseID,$section,$schedule,$isActive,$courseNumber), "pass") |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databaseaddGameinValidTest12 |
| Description: | Test database function genPass with invalid input |
| Test Steps | Step 1: Create input $db = new database(); |
| Step 2: Create input $semester = "TestSemester"; |
| Step 3: Create input $courseID = "1234567Test"; |
| Step 4: Create input $schedule = "3:45-5 Test"; |
| Step 5: Create input $section = "abcdSectionTest"; |
| Step 6: Create input $isActive = -5; |
| Step 7: Create input $courseNumber = "TestCourseNumber"; |
| Step 8: assertEquals($db->addGame($semester,$courseID,$section,$schedule,$NULL,$courseNumber), "fail") |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databaseupdateAdminPasswordValidTest13 |
| Description: | Test database function updateAdminPassword with a valid input |
| Test Steps | Step 1: Create input string $email = "test@test.com" |
| Step 2: Create input string $password = "NewTestPassword"; |
| Step 3: Create database object $db |
| Step 4: assertEquals ($db->updateAdminPassword($email,$password), 'pass') |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databaseupdateAdminPasswordinValidTest14 |
| Description: | Test database function updateAdminPassword with a invalid input |
| Test Steps | Step 1: Create input string $email = "test@test.com" |
| Step 2: Create input string $password = "NewTestPassword"; |
| Step 3: Create database object $db |
| Step 4: assertEquals($db->updateAdminPassword($email,NULL), 'fail'); |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databasesetStudentActiveValidTest15 |
| Description: | Test database function studentActive with a valid input |
| Test Steps | Step 1: Create input string $email = "Ricke@roberts.com" |
| Step 2: Create input $isActive = rand(2,10) |
| Step 3: Create database object $db |
| Step 4: assertEquals($db->setStudentActive($email,$isActive), 'pass') |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databasesetStudentActiveinValidTest16 |
| Description: | Test database function studentActive with a invalid input |
| Test Steps | Step 1: Create input string $email = "Ricke@roberts.com" |
| Step 2: Create input $isActive = "not an int" |
| Step 3: Create database object $db |
| Step 4: assertEquals($db->setStudentActive($email,$isActive), 'fail') |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databaseAdminActiveValidTest17 |
| Description: | Test database function AdminAcvite with a valid input |
| Test Steps | Step 1: Create input string $email = "test@test.com" |
| Step 2: Create input $isActive = rand(1,10) |
| Step 3: Create database object $db |
| Step 4: assertEquals($db->setStudentActive($email,$isActive), 'pass') |
| Tester: | Jeffrey Carman |

|  |  |
| --- | --- |
| Test ID: | databaseAdminActiveinValidTest18 |
| Description: | Test database function AdminAcvite with a invalid input |
| Test Steps | Step 1: Create input string $email = "test@test.com" |
| Step 2: Create input $isActive = "not an int" |
| Step 3: Create database object $db |
| Step 4: assertEquals($db->setStudentActive($email,$isActive), 'fail') |
| Tester: | Jeffrey Carman |

|  |  |
| --- | --- |
| Test ID: | databaseupdateStudentHotelValidTest19 |
| Description: | Test database function updateStudentHotel with valid input |
| Test Steps | Step 1: Create input string $email = "jcarm012@fiu.edu" |
| Step 2: Create input string $hotel = 52 |
| Step 3: Create database object $db |
| Step 4: assertEquals($db->updateStudentHotel($student, $hotel false) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databaseupdateStudentHotelInValidTest20 |
| Description: | Test database function updateStudentHotel with invalid input |
| Test Steps | Step 1: Create input string $email = "jcarm012@fiu.edu" |
| Step 2: Create input string $hotel = 52 |
| Step 3: Create database object $db |
| Step 4: assertEquals($db->updateStudentHotel($student, NULL), false) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databasegetGameValidTest21 |
| Description: | Test database function getGame with a valid input |
| Test Steps | Step 1: Create input $game = 1 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getGame($game)), 7) |
| Tester: | Jeffrey Carman |

|  |  |
| --- | --- |
| Test ID: | databasegetGameinValidTest22 |
| Description: | Test database function getGame with a invalid input |
| Test Steps | Step 1: Create input $game = -1 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getGame($game)), 0) |
| Tester: | Jeffrey Carman |

|  |  |
| --- | --- |
| Test ID: | databasegetGroupValidTest23 |
| Description: | Test database function getGroup with valid input |
| Test Steps | Step 1: Create input $group = 46 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getGroup($group)), 10) |
| Tester: | Jeffrey Carman |

|  |  |
| --- | --- |
| Test ID: | databasegetGroupinValidTest24 |
| Description: | Test database function getGroup with invalid input |
| Test Steps | Step 1: Create input $group = -1 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getGroup($group)), 0) |
| Tester: | Jeffrey Carman |

|  |  |
| --- | --- |
| Test ID: | databasegetGameByCourseValidTest25 |
| Description: | Test database function getGamebyCourse with valid input |
| Test Steps | Step 1: Create input $course = 75648 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getGameByCourseNumber($course)), 7) |
| Tester: | Jeffrey Carman |

|  |  |
| --- | --- |
| Test ID: | databasegetGameByCourseinValidTest26 |
| Description: | Test database function getGamebyCourse with invalid input |
| Test Steps | Step 1: Create input $course = 75699 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getGameByCourseNumber($course)), 0) |
| Tester: | Jeffrey Carman |

|  |  |
| --- | --- |
| Test ID: | databasegetLocationValidTest27 |
| Description: | Test database function getLocation with valid input |
| Test Steps | Step 1: Create input $loc = 12345 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getLocation($loc)), 2) |
| Tester: | Jeffrey Carman |

|  |  |
| --- | --- |
| Test ID: | databasegetLocationinValidTest28 |
| Description: | Test database function getLocation with invalid input |
| Test Steps | Step 1: Create input $loc = 123456 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getLocation($loc)), 0) |
| Tester: | Jeffrey Carman |

|  |  |
| --- | --- |
| Test ID: | databasegetAllGamesTest29 |
| Description: | Testing database function getallgames with valid input |
| Test Steps | Step 1: create database object $db |
| Step 2: assertGreaterThan(0,count($db->getadvertising()), 0) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databasegetAdvertisingValidTest30 |
| Description: | Testing database function newHotel with valid input |
| Test Steps | Step 1: create database object $db |
| Step 2: assertGreaterThan(0,count($db->newHotel('abc Test', 12345, "economy", 3, 50000, 35000, 50.00, -1))) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databasenewHotelValidTest31 |
| Description: | Test database function searchForhotels with valid input |
| Test Steps | Step 1: Create input $name = "ABC Marketing" |
| Step 2: Create database object $db |
| Step 3: assertGreaterThan(0, count($db->searchHotel($name))) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databaseSearchHotelsValidTest32 |
| Description: | Test database function searchForhotels with invalid input |
| Test Steps | Step 1: Create input $name = "zzz" |
| Step 2: Create database object $db |
| Step 3: assertLessrThan(0, count($db->searchHotel($name))) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databaseSearchHotelsInValidTest33 |
| Description: | Test database function getGroupsforGame with valid input |
| Test Steps | Step 1: Create input $id = 3 |
| Step 2: Create database object $db |
| Step 3: assertGreaterThan(0, count($db->getGroupsforGame($id))) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databasegetGroupsForGameValidTest34 |
| Description: | Test database function getGroupsforGame with invalid input |
| Test Steps | Step 1: Create input $id = -2 |
| Step 2: Create database object $db |
| Step 3: assertLessThan(0, count($db->getGroupsforGame($id))) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | databasegetGroupsForGameinValidTest35 |
| Description: | Test database function getGroupsforGame with invalid input |
| Test Steps | Step 1: Create input $id = -2 |
| Step 2: Create database object $db |
| Step 3: assertLessThan(0, count($db->getGroupsforGame($id))) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | AdminControllerconstructInvalidParTest36 |
| Description: | Test AdminController object with invalid parameter |
| Test Steps | Step 1: Create AdminController Object with 'aaa' as a parameter |
| Step 2: (AdminController::class, $m) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | AdminControllerconstrucValidParTest37 |
| Description: | Test AdminController object with valid parameter |
| Test Steps | Step 1: Create AdminController Object with no parameter |
| Step 2: (AdminController::class, $m) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | AdminControllerCreatePageTest38 |
| Description: | Test AdminController function createPage |
| Test Steps | Step 1: Create AdminController object $a |
| Step 2: Create input array $value = $a->viewUsersForGame(3); |
| Step 3: Create string $return = "<form action='' method='post'>  <div class='bs-example'>". "<div class='panel panel-default'>".  "<div class='panel-body' align='center'>". "<h2>User Accounts Management</h2>".  "<a href='http://marketsim-dev.cis.fiu.edu/admin/ManagePage.php?addAdminUser=Add+Admistrative+User' class='btn btn-primary'>Add Admistrative User</a>".  "<a href='http://marketsim-dev.cis.fiu.edu/admin/ManagePage.php?addBotUser=Add+Bot+User' class='btn btn-primary'>Add Bot User</a>".  //"<a href='http://marketsim-dev.cis.fiu.edu/admin/ManagePage.php?addBotUser=Add+Bot+User&textbox\_bot\_id=&textbox\_bot\_fname=&textbox\_bot\_lname=&textbox\_bot\_email=' class='btn btn-primary'>Add Bot User</a>".  "<a href='http://marketsim-dev.cis.fiu.edu/admin/ManagePage.php?viewAllGames=View+All+Games' class='btn btn-primary'>View All Games</a>".  "<a href='http://marketsim-dev.cis.fiu.edu/admin/ManagePage.php?viewAllUsers=View+All+Users' class='btn btn-primary'>View All Users</a>".  "</div>". "<div class='' align='center'>".  "<div class='panel-footer clearfix' style='width: 1060px' >".  $value. "</div>". "</div>". "</div>". "</div>". "</form>" |
| Step 4: assertEquals($db->updateStudentHotel($student, NULL), false) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | AdminControlleraddAdminPageTest39 |
| Description: | Test database function addAdminPage with valid input |
| Test Steps | Step 1: Create AdminController object $a |
| Step 2: Create string $result = "<div class='' align='left'>".  "<h3>First Name</h3>". "<input type = 'Text' name = 'textbox\_fname' pattern='.{1,}' placeholder='Javier' title='Name of the administrator account'>". "<h3>Last Name</h3>". "<input type = 'Text' name = 'textbox\_lname' placeholder = 'Andrial' title='Last name of Admin account. not required' >". "<h3>Email</h3>". "<input type = 'email' name = 'textbox\_email' pattern='[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$' placeholder='admin@marketsim.edu'>".  "<h3>Password</h3>". "<input type = 'password' name = 'textbox\_password1' pattern='.{8,}' placeholder='password' title='8 character passowrd minimum'>".  "<h3>Re-enter Password</h3>". "<input type = 'password' name = 'textbox\_password2' pattern='.{8,}' placeholder='password' title='those same 8 characters you just forgot go here'>".  "<h3>Secret Question</h3>". "<input type = 'Text' name = 'textbox\_secretQuestion' pattern='.{1,}' placeholder='Whats my favorite website?' title='Used to recover your account incase you forget those 8 characters again'>".  "<h3>Secret Answer</h3>". "<input type = 'Text' name = 'textbox\_secretAnswer' pattern='.{1,}' placeholder='marketsim-dev.cis.fiu.edu'>". "<br />". "<br />"."<br />".  //"<a href='http://marketsim-dev.cis.fiu.edu/admin%20shit/adminUsers.php?textbox\_fname=&textbox\_lname=&textbox\_email=&textbox\_password1=&textbox\_password2=&textbox\_secretQuestion=&textbox\_secretAnswer=&button\_addAdmin=Create+Admistrative+Account' class='btn btn-primary'>View All Users</a>".   "<input name='button\_addAdmin' type = 'submit' value = 'Create Admistrative Account' class='btn btn-primary'/>".  "</div>" |
| Step 3: assertEquals($a->addAdminPage(), $result) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | AdminControlleraddBotPageTest40 |
| Description: | Test database function addBotPage with valid input |
| Test Steps | Step 1: Create AdminController object $a |
| Step 2:"<div class='' align='left'>".  "<h3>ID</h3>".  "<input type = 'Text' name = 'textbox\_bot\_id' pattern='^[+-]?\d' placeholder='-1' title='a unique number to identify this user'>".  "<h3>First Name</h3>".  "<input type = 'Text' name = 'textbox\_bot\_fname' placeholder='Javier' title='first name of the bot'>".  "<h3>Last Name</h3>".  "<input type = 'Text' name = 'textbox\_bot\_lname' placeholder='Andrial' title='lastname, hope you're not running out of creativity! email is coming next'>".  "<h3>Email</h3>".  "<input type = 'email' name = 'textbox\_bot\_email' pattern='[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$' placeholder='bot@marketsim.edu' title='doesnt have to be real'>".  "<br />".  "<br />"."<br />".  "<input type = 'submit' value = 'Create Bot Account' class='btn btn-primary' title='only works if pressed with both eyes closed'/>".  "</div>" |
| Step 3: assertEquals($a->addAdminPage(), $result) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | AdminControlleraddBotHotelPageTest41 |
| Description: | Test database function addBotHotelPage with valid input |
| Test Steps | Step 1: Create AdminController object $a |
|  | Step 2: create database obj $db |
|  | Step 3 Create new hotel, $db->newHotel("bot Hotel Test", 12345, "economy", 1, 50000.00, 50000, 50, -1) |
|  | Step 4: assertGreaterThan(0,count($db->newHotel("bot Hotel Test", 12345, "economy", 1, 50000.00, 50000, 50, -1))) |
|  | Step 5: Create string $result = "<div class='' align='left'>".  "<h2>Create the Bot's Hotel</h2>". "<h3>Which Game to Assign bot to?</h3>".  "<input type = 'Text' name = 'textbox\_Hotel\_game\_id' require>". "<h3>Name of Hotel</h3>". "<input type = 'Text' name = 'textbox\_Hotel\_name' pattern='^[+-]\d' placeholder='FIU inn'>". "<h3>Location</h3>". "<input type = 'Text' name = 'textbox\_Hotel\_fname' placeholder='Florida'>".  "<h3>Hotel Type</h3>".   "<label class='radio-inline' >".  "<input type='radio' name='optradio' value ='economic'>Economic".  "</label>".  "<label class='radio-inline' for 'mid'>".  "<input type='radio' name='optradio' value ='midrange'>Midrange".  "</label>".  "<label class='radio-inline' for 'lux'>".  "<input type='radio' name='optradio' value ='luxury'>Luxury".  "</label>".   "<br />".  "<br />"."<br />".  "<input type = 'submit' value = 'Create Bot Account' class='btn btn-primary'/>".  "</div>" |
|  | Step 3: assertEquals($a->addBotHotelPage(-4,"bot4", "bot4","bot4@marketsim.com"), $result) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | AdminControllercreateGamePageTest42 |
| Description: | Test database functioncreateGamePage with valid input |
| Test Steps | Step 1: Create AdminController object $a |
| Step 2:"<div class='' align='left'>".  "<h2>Game Creation Page</h2>".    "<h3>Course Number</h3>".  "<input type = 'text' name = 'textbox\_courseNumber' pattern='[0-9]{5}' placeholder = '88529' title='5 digit number'>".    "<h3>Course ID</h3>".  "<input type = 'text' name = 'textbox\_courseID' placeholder = 'MAR2015'>".    "<h3>Section</h3>".  "<input type = 'text' name = 'textbox\_section' placeholder='U02'>".    "<h3>Semester of Game</h3>".  "<input type = 'text' name = 'textbox\_semester' placeholder='FALL 2016'>".    "<h3>Course Meeting time</h3>".  "<input type = 'text' name = 'textbox\_schedule' placeholder='mwf 12pm-1:30pm'>".   "<br />".  "<br />"."<br />".   "<input name='button\_addGame' type = 'submit' value = 'Create Game' class='btn btn-primary'/>".  "</div>" |
| Step 3: assertEquals($a->createGamePage(), $result) |
| Tester: | Jeffrey Carman |
| Result: | Pass |