**Project Title: Dental Clinic Management System**

**Phase IV: Software Testing**

**Team Leader Name: Enisa Mishka**

**Submission Deadline: May 7, 2024, 23:59**

**1.Introduction to Testing**

Software testing is the process of looking through software to find bugs or mistakes and make sure it complies with requirements. It seeks to confirm that the program operates accurately and exhibits expected behavior in various scenarios. Testers create test cases, carry them out, and then examine the outcomes to find any problems. Software testing serves to raise the quality, security, and performance of software products.

Objectives:

There are various objectives when it comes to software testing.

* Validating that the program runs as expected and meets all specified requirements
* Identifying defects or issues within the application
* Ensuring that user needs and quality standards are satisfied by a system
* Checking security features’ effectiveness, performance levels, and overall reliability among others.

Types of testing:

* Functional Testing: This examination ensures whether the software operates as intended on specific requirements.
* Non-functional Testing: This type of examination emphasizes performance, usability, reliability and security.
* Manual Testing: Conducted by human testers alone without employing any automation tools for this activity.
* Automated Testing: Using automation tools to run test scripts and compare actual results with expected outcomes.
* Regression: Ensuring that new amendments have not negatively affected existing operations within programs.
* Integration Testing: It checks how various parts or components interact with one another in a system under test environment (software module)
* User Acceptance Testing (UAT): This is done by end-users who validate if they meet their needs or not.

**2.Purpose of Testing:**

The purpose of software testing is to uncover mistakes made in the software development phase so that it can run smoothly. It ensures that the software meets user needs & performs well. Testing also helps prevent potential problems that may arise in the future. In general, testing improves software quality & makes it more reliable.

**3.Focus on Testing a Single Component:**

*Choose a component (class, module, function, etc.) from your codebase for testing:*

The purpose of this PHP code is to validate that the getPatientById method in a Patient class works as intended. It is a PHPUnit test case. In the setUp method, it first establishes a database connection and makes an instance of the Patient class. Next, the testGetPatientById function calls the getPatientById method with a specified patient ID, inserts test data into the database, and verifies that the data returned matches the anticipated patient data. This test verifies the accuracy and dependability of the getPatientById method by ensuring that patient data is appropriately retrieved from the database. This test case contributes to maintaining the functionality and integrity of the Patient class within the application by utilizing PHPUnit's setup/teardown functionality and assertion techniques.

*Explain why testing this component is important, considering its role, complexity, and*

*impact on the system:*

Role: Retrieving patient information from the database using a unique ID is mostly dependent on the getPatientById function. Since patient data is essential to the operation of a healthcare system, it is critical to guarantee the precision and dependability of this approach in order to preserve data integrity and give medical professionals accurate patient information.   
Complexity: A number of aspects, such as database interfaces, data validation, and error handling, may contribute to the method's complexity. Complexities may result in errors or faults when obtaining patient data, which emphasizes the necessity of extensive testing to find and fix any problems.   
Effect on the System: The getPatientById method has a big effect on the system since it has a direct bearing on how dependable and easy to use the patient-related features are in the healthcare system. Any errors or inaccuracies in retrieving patient data could lead to incorrect diagnoses, treatment delays, or compromised patient care, highlighting the importance of rigorous testing to ensure the method performs as intended.

**4.Preparing Test Cases:**

Normal Inputs:

-Run tests using the typical inputs that the part is meant to receive.  
-Verify that the component operates as intended under normal circumstances.   
  
Edge Cases:

-Conduct tests using inputs that are extreme or borderline, like the maximum or lowest values. ---Examine the way the component responds to inputs that are beyond its capacity.

Invalid Inputs:

-Conduct tests using incorrect or non-expected inputs.   
-Ensure that the component accepts these inputs politely and, if necessary, generates unambiguous error messages.

Combining Scenarios:

-Experiment with various input combinations to account for every scenario that might arise.   
-Examine the component's behavior with a combination of edge, normal, and invalid inputs.

Error Handling:

-Evaluate how the part reacts to unforeseen problems or errors.   
-Make sure it responds to problems effectively and offers useful

**5.Choosing Testing Frameworks:**

A framework-independent library for PHP unit testing is called PHPUnit. Unit testing is a technique used to evaluate short code segments against expected results.

* Installation:

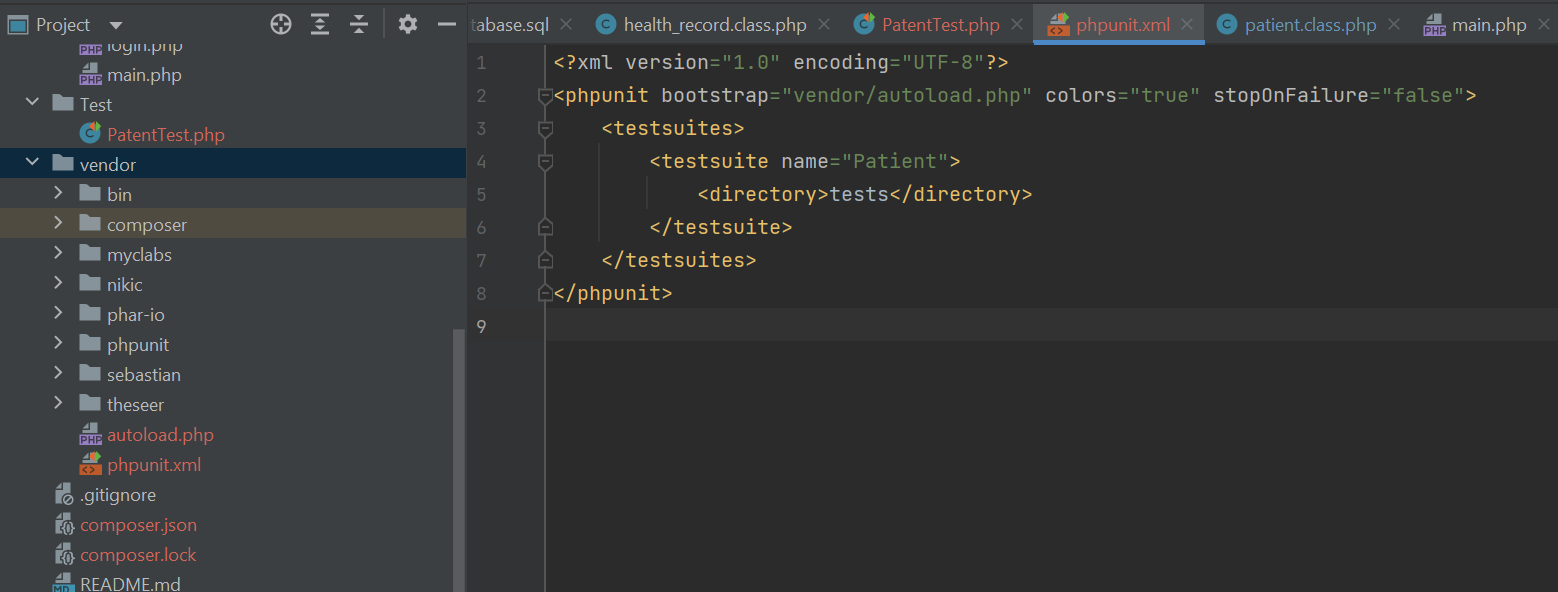
Is needed to install PHPUnit using Composer, which is the recommended way

composer require --dev phpunit/phpunit

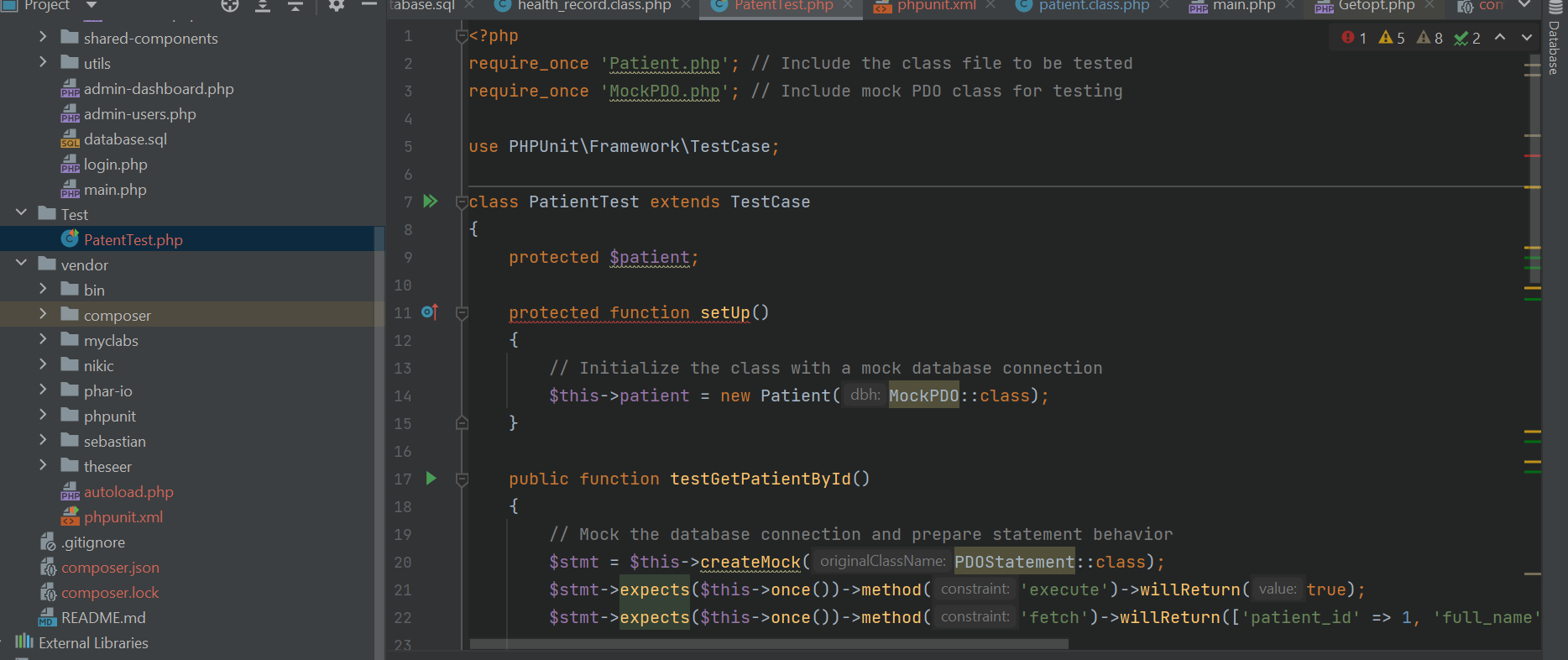
* Create a tests directory in your project root if it doesn't exist already.
* I created a directory named test and insider the tests directory, and I created PHP files named PatientTest.php based on my project.
* Each test case class should extend the PHPUnit\Framework\TestCase class provided by PHPUnit.
* To run PHPunit use command ./vendor/bin/phpunit

**6.Writing Test Code:**

PHPUnit configuration file using XML configuration that allows to customize various aspect of testing .

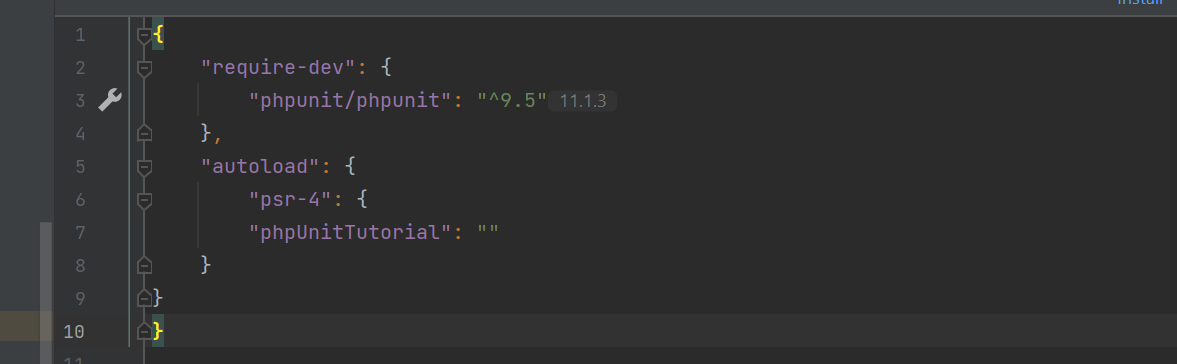
****

I have created a folder named test in my project directory and inside the test folder I have created a new php file named PatientTest which will contain my PHPUnit test class for testing Patient Class.





Composer json



**7.Running Tests:**

*Explain how to execute tests and interpret the results, covering passing, failing, and*

*error scenarios.*

Create a tests directory in your project root if it doesn't exist already. Inside the tests directory, create PHP files for your test cases. These files should have names that end with Test.php. Each test case class should extend the PHPUnit\Framework\TestCase class provided by PHPUnit.

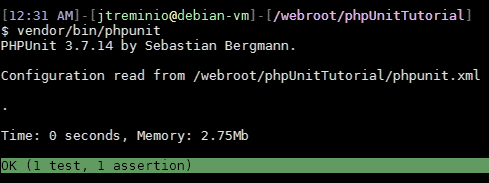
./vendor/bin/phpunit tests

./vendor/bin/phpunit tests/MyTest.php to run the test

Passing Tests: If all tests pass without any issues, PHPUnit will display a summary indicating that all tests were successful.

Failing Tests: If a test fails, PHPUnit will provide detailed information about the failure, including the test method that failed, the expected result, and the actual result.

Error Scenarios: Errors occur when there is an unexpected condition or exception during the execution of a test. PHPUnit distinguishes between test failures and errors. Failures occur when an assertion fails, while errors occur due to unexpected exceptions or conditions.



**8.Test Coverage:**

Test coverage is a valuable metric that helps ensure the quality and reliability of your code. By generating and analyzing coverage reports, you can identify areas for improvement and make informed decisions about where to focus your testing efforts. Ultimately, higher test coverage leads to more robust and maintainable code.

Interpret Coverage Reports:

* Look at the coverage percentage to see how much of your code is covered by tests.
* Identify areas of your code that are not covered by tests.
* These are potential places where bugs could hide. Focus on improving coverage in critical or complex parts of your code.

Improve Test Coverage:

* Write additional tests to cover untested code paths, edge cases, and error conditions.
* Refactor your code to make it more testable and modular.
* Regularly review and update your coverage metrics as your codebase evolves.