Table of Contents

**Backend Testing**2

**Types of Backend Testing** 2

**Functional Testing** 4

**Non-Functional Testing** 4

**Backend Testing Validation**5

**Basic Architecture and Integration of Backend testing**……….……………………………………………….6

**Tools For Backend Testing**………………………………………………………………………………………………..7

**Pytest**……………………………………………………………………………………………………………………………..8

**Pytest Framework** 8

# **Backend Testing**

Testing is the process of verifying and assessing the entire software or application to ensure that the developed software or product is working properly as per the specification. There are many ways of testing, various levels of testing, and testing processes. These are all carried out in the testing phase to check the entire developed product. The term backend generally refers to server-side deployment**.** The process is exclusively running in the backend which is not visible to the user but only the expected outcome is shown to the user. In every web application, there will be a backend language to accomplish the task.

For Example, while uploading the details of the students in the database, the database will store all the details. When there is a need to display the details of the students, it will simply fetch all the details and display them. Here, it will show only the result, not the process and how it fetches the details.

Backend Testing is a way of validation that checks the database or server-side of the web application. The main role of backend testing is to check the application layer and the database layer. It uncovers an error or bug in the database or server-side.

For implementing backend testing, it is required to have knowledge for server-side or database language. It is also known as **Database Testing.** Backend testing is a very crucial as if something goes wrong or error occurs at the server-side, it will not halt that task or the output will get differed or sometimes it will also cause problems such as data loss, deadlock, etc.,

**Types of Backend Testing**

### The following are the different types of backend testing:

### **Structural Testing-** Structural testing is the process of validating all the elements that are present inside the data repository and are primarily used for data storage. It involves checking the objects of front-end developments with the database mapping objects.

**Types of Structural Testing:** The following are the different types of structural testing:

* **Table and Column Testing:** In Table and Column Testing, the properties of the Tables and Columns are mapped correctly on both the server side and the front side. It Validates right input is provided by the user as per the requirements. Column datatype should be correct and Values in the columns should match correctly with the database values. Tables and columns which are not in use should be identified.

For example, if we send the wrong datatype for the column on the server-side which is different from the front-end then it will raise an error.

* **Schema Testing:** In this Schema Testing, the tester will check for the correctly mapped objects. This is also known as **mapping testing.** It ensures whether the objects of the front-end and the objects of the back-end are correctly matched or mapped. It will mainly focus on schema objects such as a table, view, indexes, clusters, etc., In this testing, the tester will find the issues of mapped objects like table, view, etc.,
* **Key and Indexes Testing:** In Key and Indexes Testing, we validates the key and indexes of the columns and validates the key constraints are provided for the mentioned columns. It verifies the length and size of the indexes, naming conventions of the keys. Based on the requirement, creation of clustered and non-clustered indexes for the tables are created.

For example, Correct reference with parent table is ensured for the primary key.

* **Trigger Testing:** It ensures that the executed triggers are fulfilling the required conditions of the DML transactions. It validates whether the triggers make the data updates correctly when we have executed them. It checks the coding conventions are followed correctly during the coding phase of the triggers. It ensures that the trigger functionalities of update, delete, and insert.
* **Stored Procedures Testing:** Stored procedure results are validated. It checks whether the stored procedure contains the valid conditions for looping and conditional statements as per the requirement. It Validates the unused stored operations, cursor operations, TRIM operations. It ensures that the required triggers are implicitly invoked by executing the stored procedures.
* **Database Server Validation Testing:** It validates the database configuration details as per the requirements. It validates that the transactions of the data are made as per the requirements. It validates the user’s authentication and authorization. For Example, If wrong user authentication is given, it throws an error.

### **Functional Testing**

### **Functional Testing -** Functional Testing is the process of evaluating the functionality of the software application, its features against the requirements. It focuses on validating the expected output for the input provided.

### There are different types of functional testing:

### **Black Box Testing:** Black Box Testing is the process of checking the functionalities of the integration of the database. This testing is carried out at the early stage of development and hence It is very helpful to reduce errors. It consists of various techniques such as boundary analysis, equivalent partitioning, and cause-effect graphing.

### **White Box Testing: -**White Box Testing is the process of validating the internal structure of the database. The database triggers, functions, views, queries, and cursors will be checked in this testing.

* It validates the database schema, database table, etc, the coding errors in the triggers , in queries are handles and eliminated

**Non-Functional Testing**

**Non-Functional Testing -**Non-functional testing from backend testing perspective is the process of performing load testing, stress testing, and checking minimum system requirements are required to meet the requirements. It will also detect risks, and errors and optimize the performance of the database.

* **Load Testing:** Performance and scalability of the database is tested in the Backend Load testing.

It validates the behavior of the software when many users perform or logged simultaneously.

For example, when thousands of the user access the application simultaneously then software or the application should not create any traffic issues then load testing is successful.

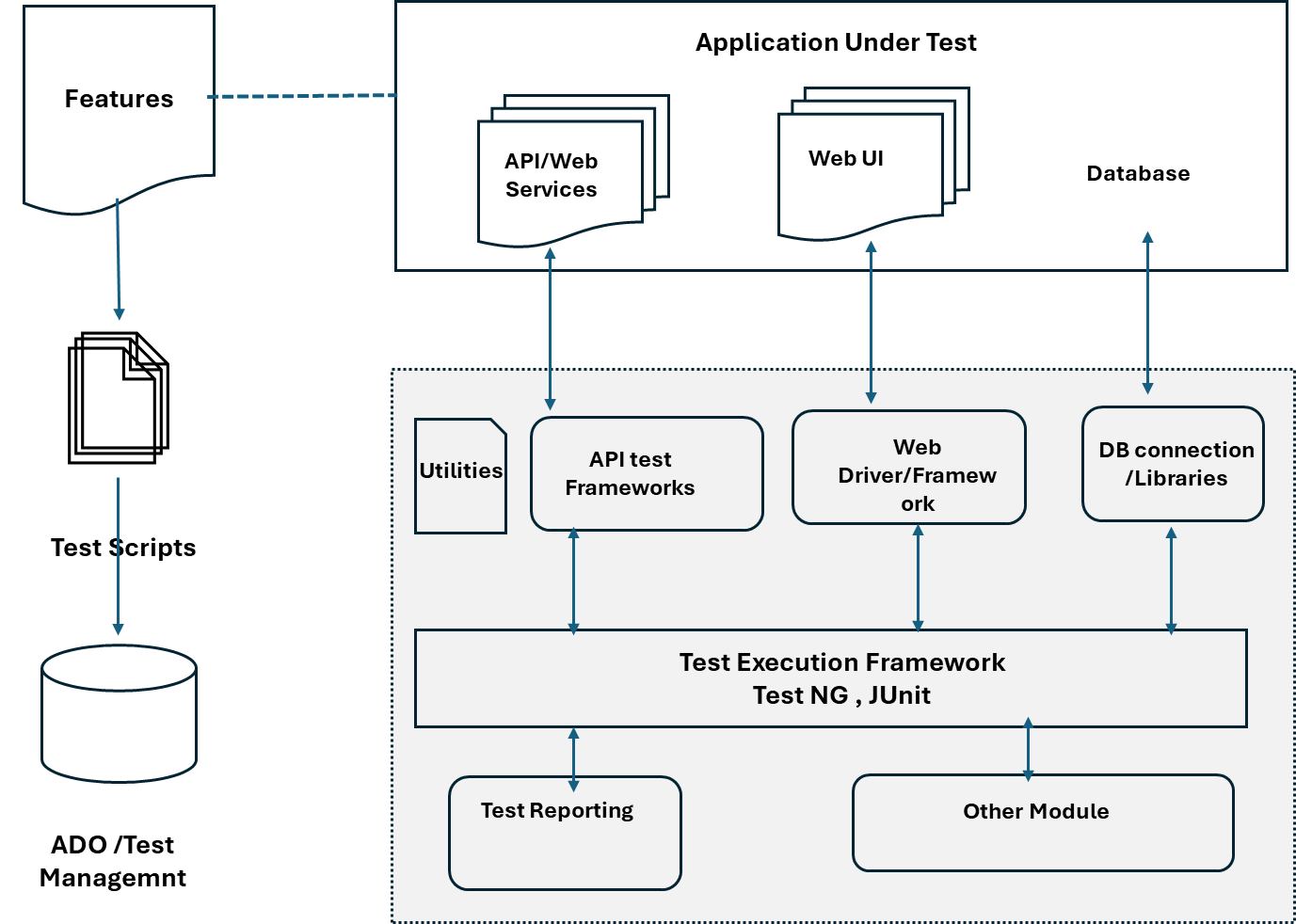
* **Stress Testing:** Stress testing is a testing process that is performed to identify the breakpoint of the system. to furnish such tests an application is loaded till the stage the system fails, This point is known as a breakpoint of the database system. It assesses and analyzes after the breakage of system failure.

### **Backend Testing Validation**

The following are some of the factors for backend testing validation:

* **Performance Check:** It validates the performance of each individual test and the system behavior.
* **Sequence Testing:** Backend testing validates that the tests are distributed according to the priority.
* **Database Server Validations:** In this, ensures that the data fed through for the tests is correct or not.
* **Functions Testing:** In this, the test validates the consistency in transactions of the database.
* **Key and Indexes:** In this, the test ensures that the accurate constraint and the rules of constraints and indexes are followed properly.
* **Data Integrity Testing:** It is a technique in which data is verified in the database whether it is accurate and functions as per requirements.
* **Database Tables:** It ensures that the created table and the queries for the output are providing the expected result.
* **Database Triggers:** Backend Testing validates the correctness of the functionality of triggers.
* **Stored Procedures:** Backend testing validates the functions, return statements, calling the other events, etc., are correctly mentioned as per the requirements,
* **Schema:** Backend testing validates that the data is organized in a correct way as per the business requirement and confirms the outcome

**Basic Architecture and Integration of Backend testing**



### **Backend testing Usage :** Backend testing is to perform to validate the issue which are internal to system and configuration, it help to detect :

* **Performance check**: This validates the performance of each test and the system behavior. This includes checking for performance bottlenecks and ensuring the application can handle increasing traffic and demand.
* **Sequence testing**: This validates the tests are run in the sequential order according to the priority provided for the proper functioning of the system and the outputs are accurate.
* **Database server validations:** This ensures the data fed through for the tests is correct. This includes checking for invalid data types, missing data, and out-of-range values.
* **Functions testing:** This validates the consistency in transactions of the database. This includes checking for errors in the application’s calculations and ensuring that the application behaves as expected in different scenarios.
* **Key and indexes:** This ensures that the accurate constraints and the rules of constraints and indexes are correctly followed. This includes checking for errors in the application’s data structures and ensuring the application can retrieve data efficiently.
* **Data integrity testing:** This is a technique in which data is verified in the database to determine whether it is accurate, and functions as required. This includes checking for data corruption and ensuring that the data is consistent with the application’s business logic.
* **Database tables:** This ensures that the created table and the queries for the output are providing the expected result. This includes checking for errors in the application’s database schema and ensuring that the application can insert, update, and delete data correctly.

### **Tools For Backend Testing**

There are some commonly used tools:

**Pytest** : The pytest framework makes it easy to write small, readable tests, and can scale to support complex functional testing for applications and libraries.

**Postman**: Postman is a popular tool for API testing, development and debugging of the API. It allows you to send and receive HTTP requests and responses, and it has many features for testing APIs, such as mocking, stubbing, and logging.

**Data Factory**: You can use the Data factory to create an automated workflow to perform data-driven test cases using your database as a data source. It provides a programmatic UI for ease of testing.

**Data Generator**: Using Data Generator, you can generate any data row and schema objects per your backend testing requirement.

**Testsigma**: The cloud-based test automation tool for front-end and back-end testing, with API testing capabilities and soon-to-be-released database testing features.

**TurboData**: TurboData helps to generate test data with foreign keys and select, update, and delete SQL commands.

### **Advantages of Backend Testing**

The following are some of the benefits of backend testing:

* Errors are easily detectable at the earlier stage.
* It avoids deadlock creation on the server-side.
* Web load management is easily achieved.
* The functionality of the database is maintained properly.
* It reduces data loss.
* Enhances the functioning of the system.
* It ensures the security and protection of the system.
* While doing the backend testing, the errors in the UI parts can also be detected and replaced.
* Coverage of all possible test cases.

### **Disadvantages of Backend Testing**

The following are some of the disadvantages of backend testing:

* Good domain knowledge is required.
* Providing test cases for testing requires special attention.
* Investment in Organizational costs is higher.
* It takes more time to test.
* If more testing becomes fails then It will lead to a crash on the server-side in some cases.
* Errors or Unexpected results from one test case scenario will affect the other system results also.

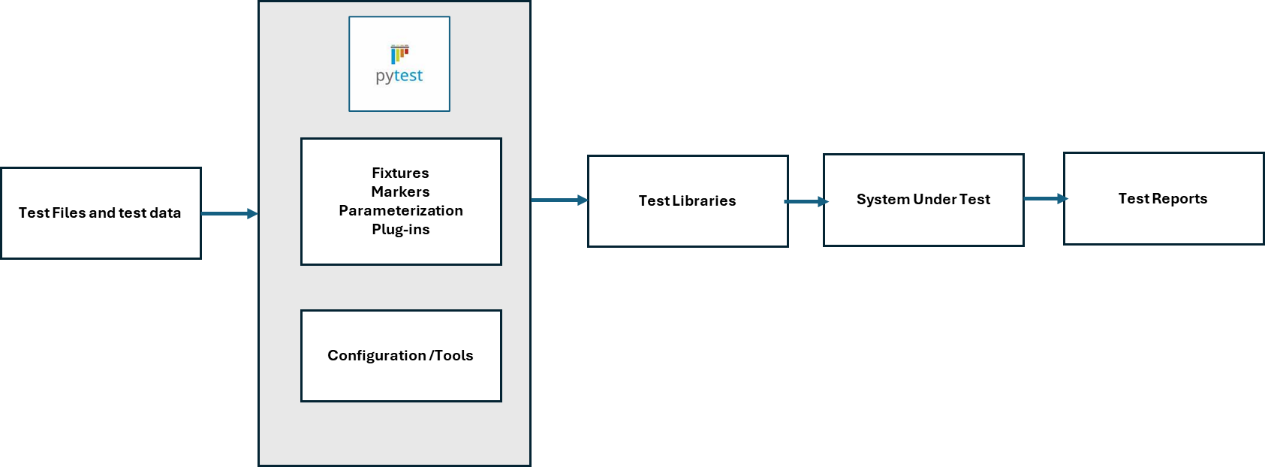
**Pytest**

The pytest framework makes it easy to write small, readable tests, and can scale to support complex functional testing for applications and libraries. Pytest is a python based testing framework, which is used to write and execute test codes. In the present days of REST services, pytest is mainly used for API testing even though we can use pytest to write simple to complex tests, i.e., we can write codes to test API, database, UI, etc.

Advantages of Pytest- The advantages of Pytest are as follows −

* It can run multiple tests in parallel, which reduces the execution time of the test suite.
* It has its own way to detect the test file and test functions automatically, if not mentioned explicitly.
* It allows us to skip a subset of the tests during execution.
* It allows us to run a subset of the entire test suite.
* It is free and open source and has very simple syntax which make its usage easy
* Its enables Effortless Concurrency
* It allows Maximum Code Reuse
* It is easy for integration and give good Plugins

**Pytest Framework**



**Pytest Environment Set up and integration**

To start with Pytest we need to have Pyhton installation is in place.

To start the installation with specific version execute the following command −

**pip install pytest == 2.9.1**

else To install the latest version of pytest, execute the following command −

**pip install pytest**

once installation is done we can Create a new directory and start creating the test file avigate into the directory through command line.

Basic example with test square functionality :

**#import math**

**def test\_sqrt():**

**num = 25**

**assert math.sqrt(num) == 5**

**def testsquare():**

**num = 7**

**assert 7\*7 == 40**

**def tesequality():**

**assert 10 == 11**

Once test is created it can be run as below through CLI

**Pytest**

**Test Methods**

We run all tests with pytest -v. When writing a test, if we initialize the function’s name with the phrase test\_ python understands that it is a test module.

If we want to predetermine which classes and methods are tests, we should create a file named Pytest.ini and add the following lines.

Pytest.ini contains search patterns for test files, classes and functions.

[pytest]  
python\_files = test\_\*  
python\_classses = \*Tests  
python\_functions = test\_\*

After determining that the tests start with check\_ or test\_ in pytest.ini, python will run them all when you say pytest -v regardless of what directory the test file is under.

**Test Structure**: Below is the following package tree and focus on the directories that are related to testing.The recommendation is to keep tests in different package within the project. This way, the tests will have nothing to do with the main structure of the project. Tests need to be isolated and not violate any other part of the project.As for the components that the **tests/**\* might need it in order to work properly, it’s recommended to keep the requirements of the tests inside another separate package called **testing/** next to the **tests/** directory

A screenshot of a computer program

Description automatically generated

**Pytest Features:**

**Pytest provide the feature which make the framework robust**

**Marker Usage**- To use the marker, need to include the mark module in pytest into the project as follows.it enables the grouping and tagging of test cases which can be used to sequence the test cases.

from pytest # import mark

Then tag the tests by mark(@ using the decorator) and test can be run in a group. Mark decorator identifies a class or function as one that can be searched by keyword.

Examples :

pytest -m “not ui” → doesn’t run tests with mark ui

pytest -m “backend not ui” → run tests with mark backend but does not run with mark ui

**Test Fixtures -** Fixtures are functions that run before each test function. Generally, fixtures are great to use to set up data to run tests. Common usages are to set up a database connection, URLs, and input data. Instead of running the same code, we can attach fixtures. Fixtures brings reusability code and provide maintainability.

**Parameterization -** Pytest supports parametrization. To enable parameterization, we need to import mark and annotate tests with a parameter marker. Parameterization tests look cleaner with large sets of data.

**Pytest-Html – with this feature it generate a nice readable HTML layout of HTML report for the test cass which are executed to enable this we need to install Pytest -html in CLI**

**pip install pytest-html**

**Plug-ins - The Pytest provides wide set of plug-ins and integration which enhance the module's functionality.**

**Xdist -once integrate this plug-in it enables Pytest to tests in parallel.**

#installs plug-in

pip install pytest-xdist

**Reporting-** Pytest supports generating reports in JUnit format. By creating the XML file CI/CD systems like Jenkins can read the log files.

pytest --junitxml=path

**Unittest Compatibility – it supports r**unning unittest based test To run the existing unit test style test suite using Pytest.

**pytest tests**

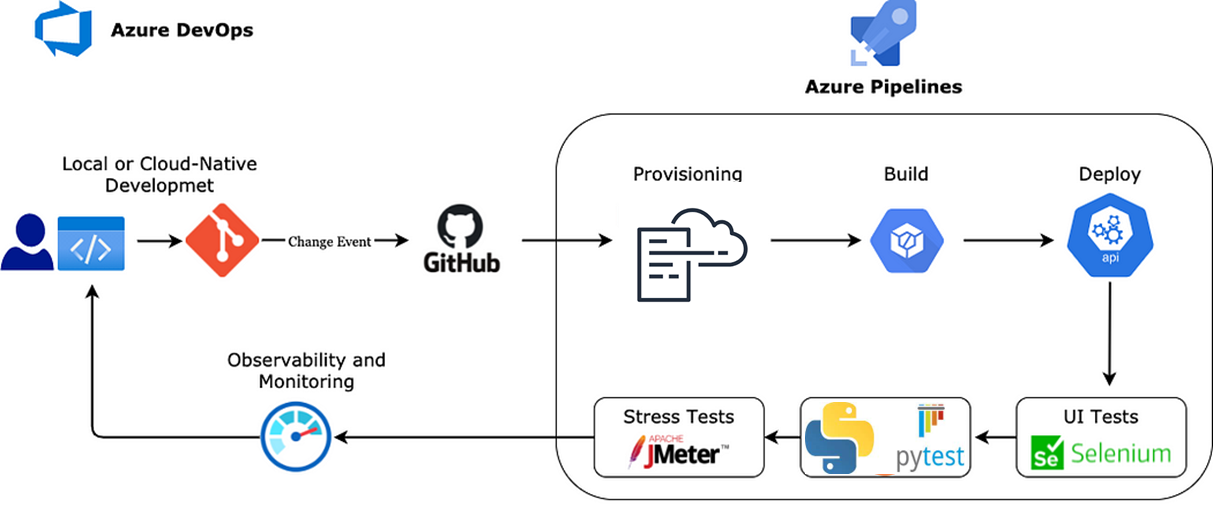
1

**Integration with Azure Pipeline**:

Pytest is easier to use with Microsoft Azure Pipelines. Just run pytest with this plugin and see your test results in the Azure Pipelines UI. This capability gives :

* Formats the PyTest output to show test docstrings and module names instead of just test case names in the Azure Pipelines UI.
* Uploads test results automatically, no need for a separate test results upload command
* Displays the number of failed tests if there were failures as an error message in the UI
* Automatically formats code coverage and uploads coverage data if pytest-cov is installed

To install “pytest-azurepipelines” need to run : **$ pip install pytest-azurepipelines**

****

**Implementation steps**

1. Before running pytest on azure devops , pip install pytest-azure-devops should be installed
2. to have the result of the tests published directly in build use Pytest reporting plugin option to create a result file with JUNIT xml file format ;
3. once you have test result as a JUNIT xml files you can use standard PublishTestResults task to publish results in the build.

**Pytest Usage :**

These are some tests where Pytest framework is very useful

**Unit testing** - The unittest module is a built-in testing framework in Python, available as part of the standard library. The primary goal of the unittest module is to facilitate the creation of test cases and test suites to verify the behavior and correctness of individual units of code

**Parameterized Testing**: pytest supports parameterized testing, allowto run the same test logic with multiple sets of input data. This is valuable for testing different scenarios and edge cases.

**Funtional Testing :** Pyetst plugin and componenet support very well the functional testing brings modularity and resuabilty the framework supports easy to write small, readable tests, and can scale to support complex functional testing for applications and libraries.

**Behavior driven test** - with pytest-bdd which is a pytest plugin it allows to script behavior-driven development (BDD) tests using the Gherkin syntax. H

**Concurrent testing**- it support API automation testing and concurrent execution, It speed up test execution, especially on computers with multiple CPU cores.

**Distributed testing -** allows tests to be executed on multiple nodes and test runs can be completed faster with distributed testing. It runs a set of tests per node, for distributed testing.

Apart from above it support Regression ,Smoke , Integration test