Back End Testing

INFO6255

Medi Servattalab

M.Servattalab@northeastern.edu

Fall 2020

Back End Testing

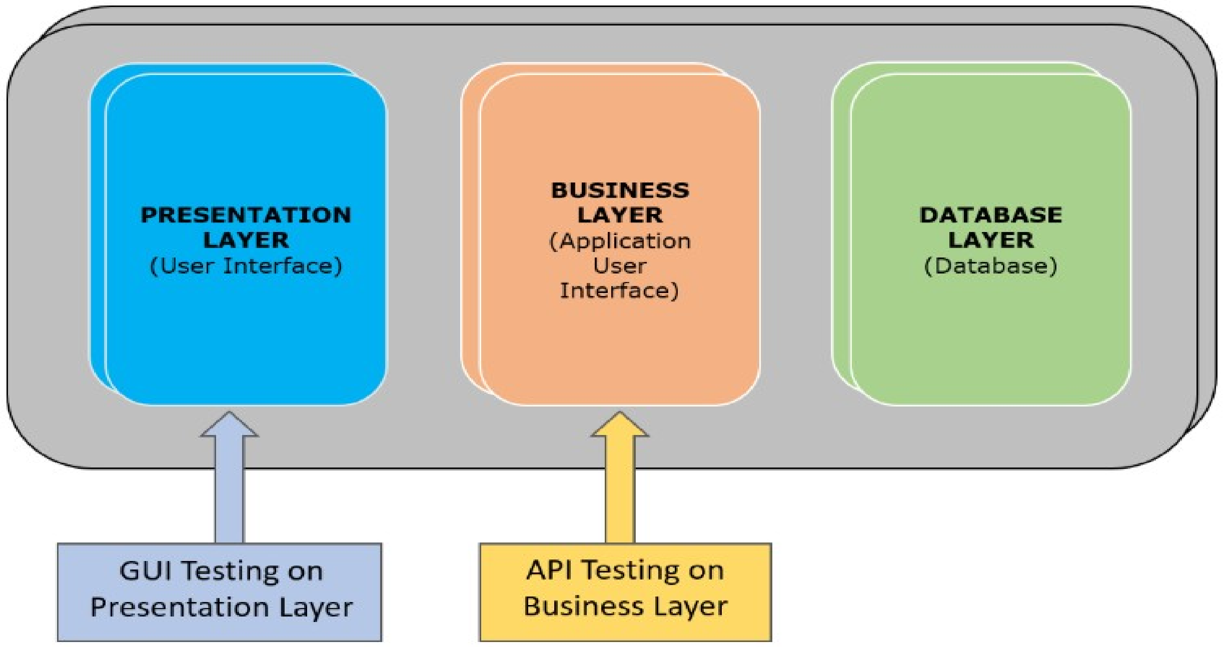
- Back end Testing vs Front End Testing

- **Front-End testing** is the testing performed on the GUI or the Presentation Layer. Also called the **Client Side** testing.

- **Back-End testing** is a type of testing that checks the **Server Side** testing, which includes the **Database** testing.
  - It also includes the **Business Layer - API Testing**

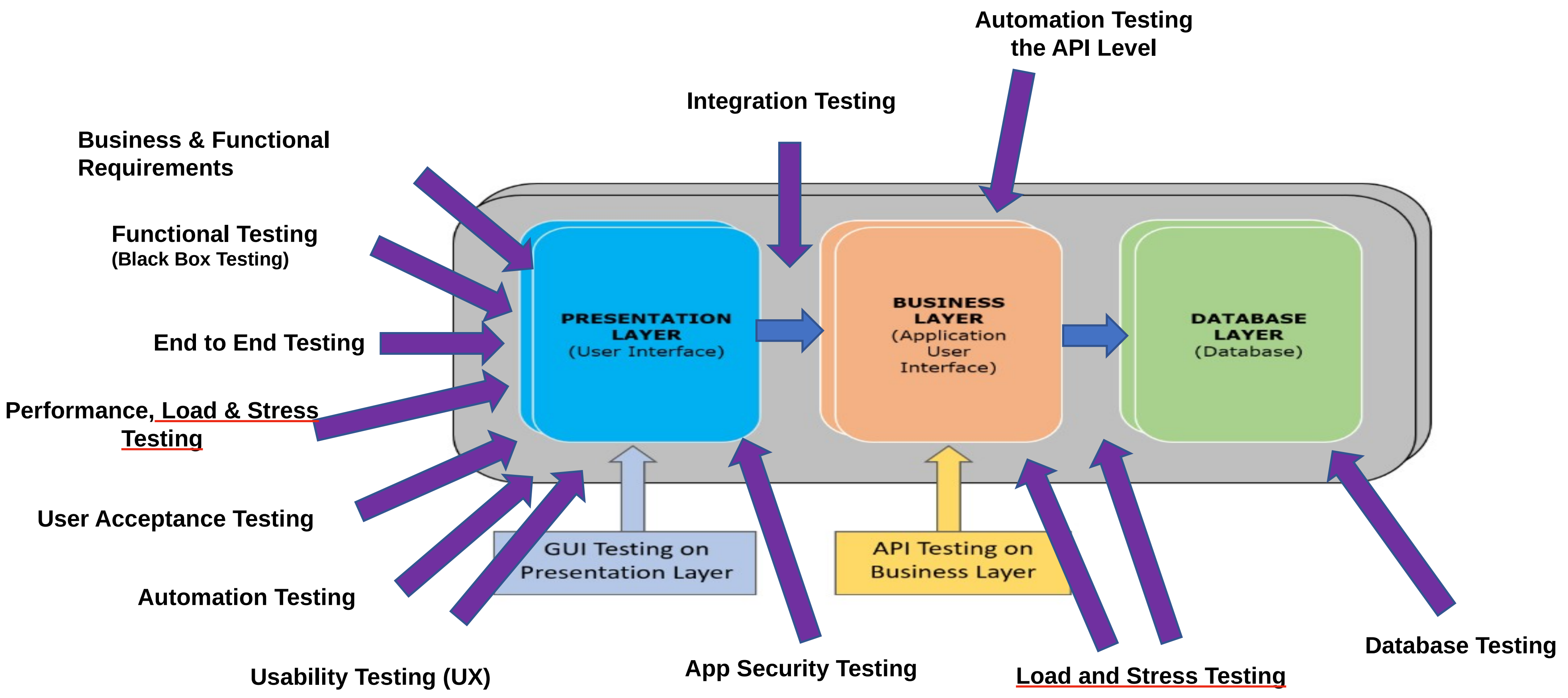
Back End Testing

- **3 Tiers of Architecture:**
  - Front end (Presentation Layer)
  - Application (Business Layer)
  - The Database Layer


PRESENTATION LAYER (User Interface)
BUSINESS LAYER (Application User Interface)
DATABASE LAYER (Database)
GUI Testing on Presentation Layer
API Testing on Business Layer

https://dzone.com/articles/api-testing-and-automation-101-the-essential-guide

Different Types of Testing


Automation Testing the API Level
Integration Testing
Business & Functional Requirements
Functional Testing (Black Box Testing)
End to End Testing
Performance, Load & Stress Testing
User Acceptance Testing
Automation Testing
Usability Testing (UX)
App Security Testing
Load and Stress Testing
Database Testing
PRESENTATION LAYER (User Interface)
BUSINESS LAYER (Application User Interface)
DATABASE LAYER (Database)
GUI Testing on Presentation Layer
API Testing on Business Layer

https://dzone.com/articles/api-testing-and-automation-101-the-essential-guide

What is a Database Management System (DBMS)?

- DBMS allows **creation, definition and manipulation** of database allowing users to **store, process and analyze** data easily. It allows:
  - Creating a Database
  - Storing Data
  - Updating Data
  - Accessing the Data
  - Securing the Data

https://www.studytonight.com/dbms/overview-of-dbms.php

Database Management ystem (DBMS)

- The DBMS manages 3 important items:
  - **The Engine** – allows the data to be accessed.
  - **The Schema** – defines the database logical structure.
  - **The Data** - the actual data elements in the database.

- **A few DB examples:**
  - MySql
  - Oracle
  - SQL Server
  - IBM DB2
  - PostgreSQL
  - Amazon SimpleDB (cloud based) etc.

Why do the Database Testing?

- **Database testing** (also known as Back-end or data **testing)** is **one** of the important types of **testing**.

- The **database** must be tested functionally, **along with load/stress testing** to ensure security and performance.

Advantages and Disadvantages of DBMS

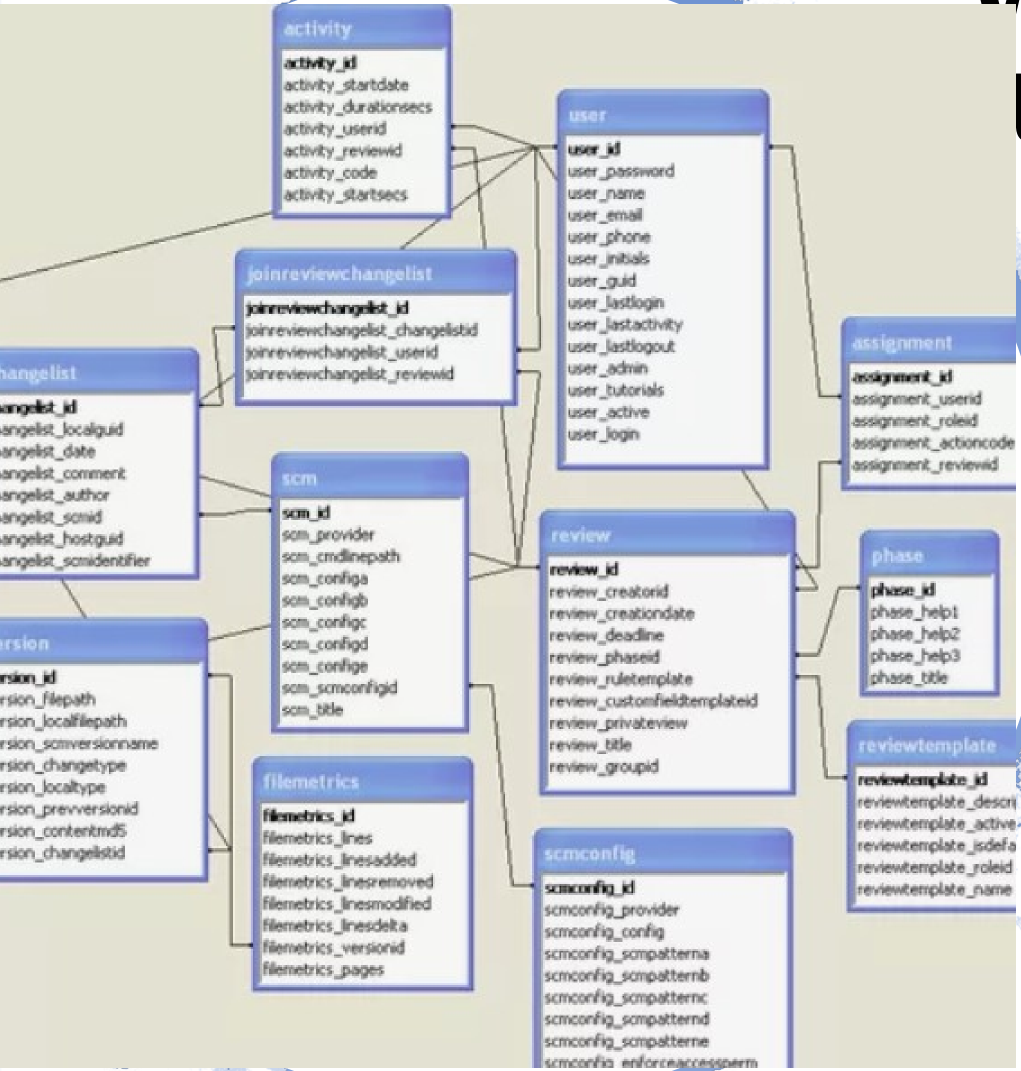
- **Advantages of DBMS**
  - **Segregation** of application program.
  - **Minimal** data duplicity or data redundancy.
  - **Easy** retrieval of data using the Query Language.
  - **Reduced** development time and maintenance need.
  - With **Data Centers in the Cloud**, we now have Database Management Systems capable of storing almost **infinite** data.
  - **Seamless** integration into the **application programming languages** which makes it very easier to add a database to almost any application or website.
- **Disadvantages of DBMS**
  - It's **Complexity.**
  - It’s **Cost**. Except MySQL, which is open source.
  - They are **large** in size.

https://www.studytonight.com/dbms/overview-of-dbms.php

数据库测试的价值

Database Testing Added Value

- Effective and efficient Database testing provides **long-term functional stability** to the overall application.
- Database testing adds **more value** to the overall work by finding out **hidden issues**.
- Any expenditure on database testing is a **long-term investment** which leads to long-term stability and robustness of the application.



What are the Testable tems for the DB Testing?

- Testable items in the Relational Databases involve Validating:
  - The schema
  - Database tables
  - Columns
  - keys and indexes
  - Stored Procedures
  - Triggers
  - Views

https://www.guru99.com/data-testing.html#2

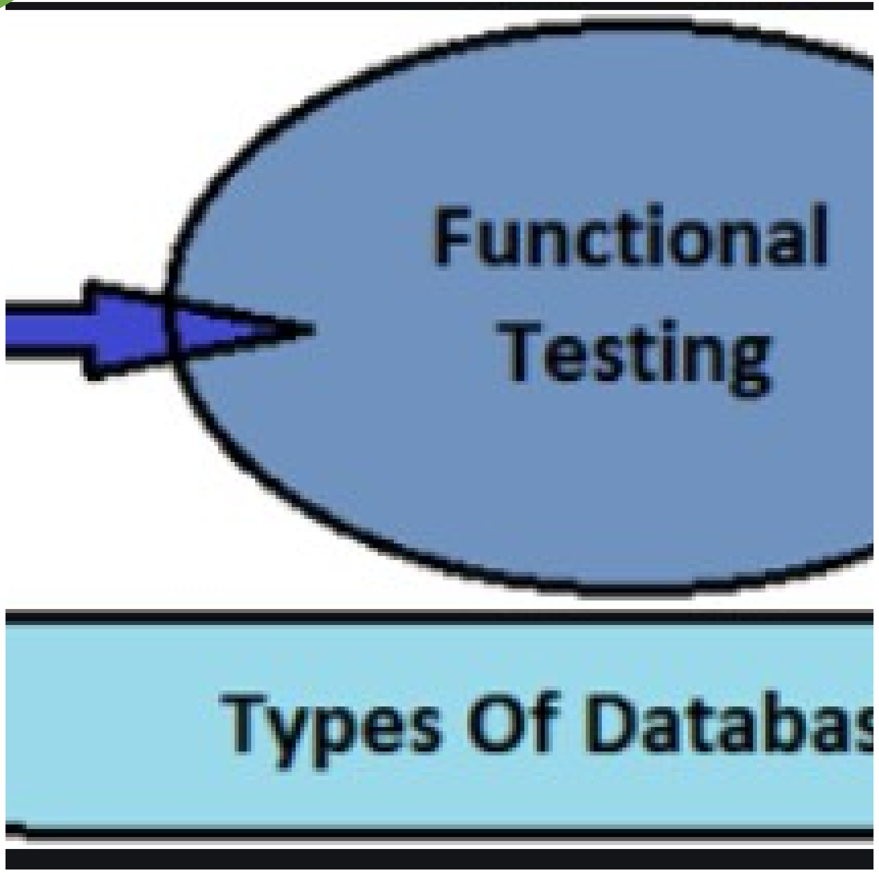
Types of Database Testing

3 Types of Database Testing:

1. **DB Structural Testing** - the validation of all the elements inside the data repository that are used primarily for **storage of data**.

2. **DB Functional Testing -** The requirement specification needs to ensure most of those transactions and operations as performed by **the end users are consistent with the requirement specifications.**

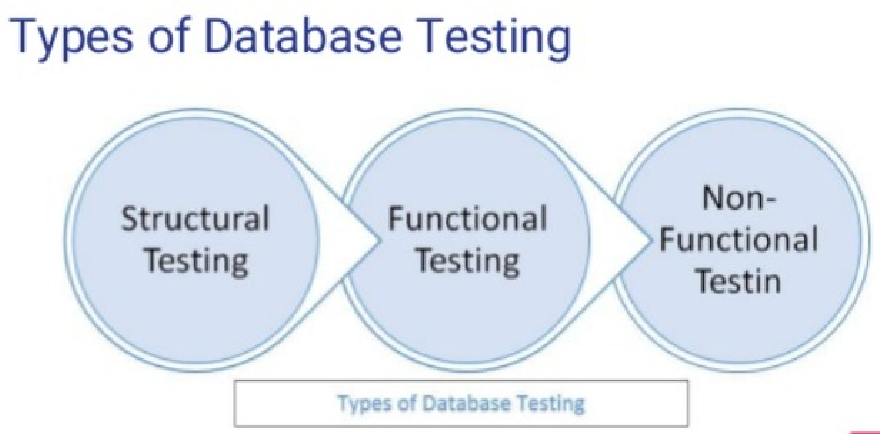
3. **DB Non-functional Testing -** These can be **Load Testing**, **Stress Testing**, **Security Testing**, Usability Testing, and Compatibility Testing



1. DB Structural Testing

- The **Database Structural Testing** includes the following:

A. Schema Validation
B. Database Tables/Column Testing
C. Stored Procedure Testing
D. Trigger testing

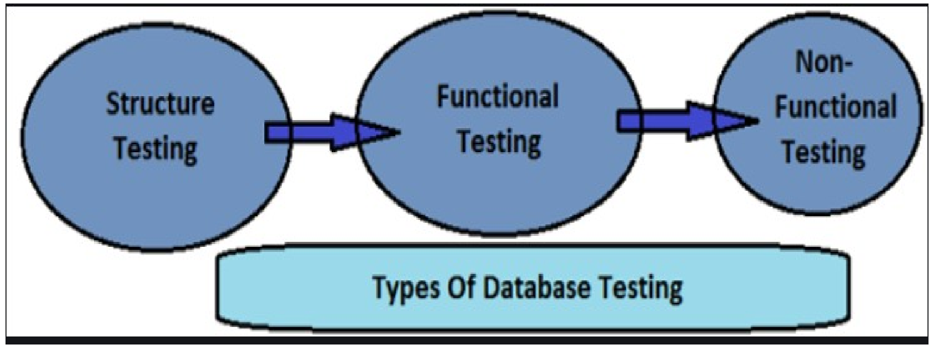


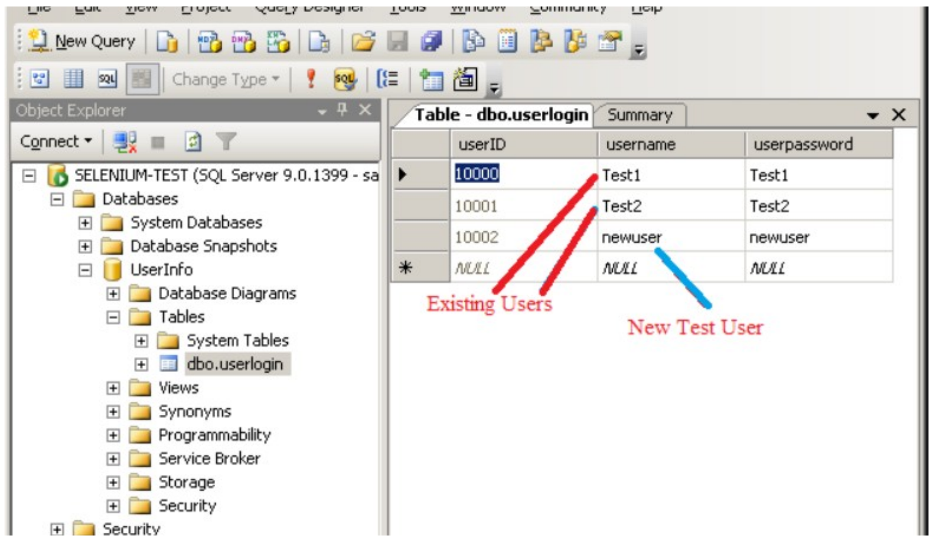
https://www.guru99.com/data-testing.html

1. Database Structural Testing

A. Schema Validation

- The schema mapping between the front end and the back end.
- Any unmapped columns or tables or views. Use the mapping testing tools for validating database schemas:
  - DBUnit Integrated with Ant
  - SQL S





SELENIUM-TEST (SQL Server 9.0.1399 - sa
- Databases
  - System Databases
  - Database Snapshots
  - UserInfo
    - Database Diagrams
    - Tables
      - System Tables
      - dbo.userlogin
    - Views
    - Synonyms
    - Programmability
    - Service Broker
    - Storage
    - Security
- Security

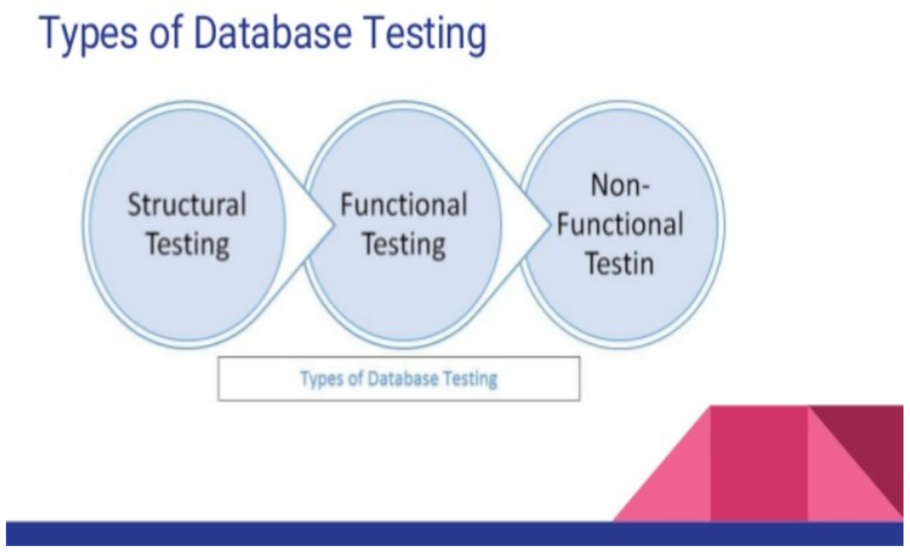
	userID	username	userpassword
▶	10000	Test1	Test1
	10001	Test2	Test2
	10002	newuser	newuser
*	NULL	NULL	NULL

https://www.guru99.com/data-testing.html

1. Database Structural Testing

B. Database Tables/Column Testing

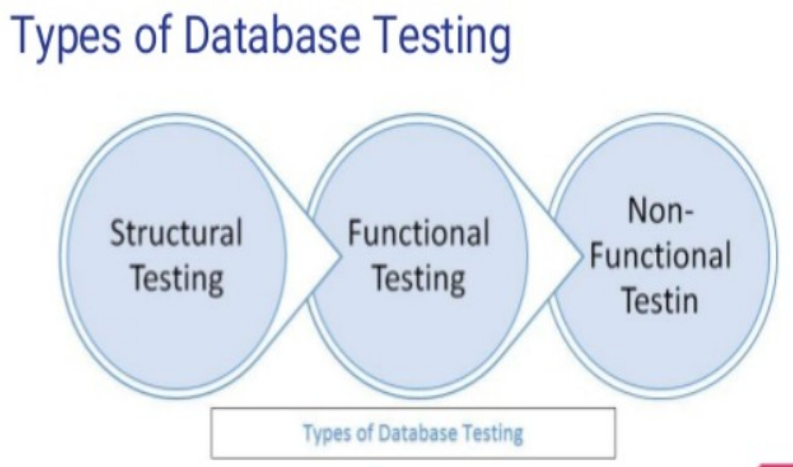
- **Mapping** of the database **fields and columns** in the back end against the mappings in the front end.
- **The length** and **naming convention** of the database fields and columns against the requirements.
- **Validation** of the presence of any unused/unmapped database tables/columns.
- **Validation** of the compatibility of the data type and field lengths.
- **Check** the **Primary and Foreign keys** constraints
  - Check references for **foreign keys** are valid
  - Check for the data types of the **primary key** and the corresponding foreign keys
  - Naming conventions for **all keys and indexes**



1. Database Structural Testing

C. Stored Procedure Testing

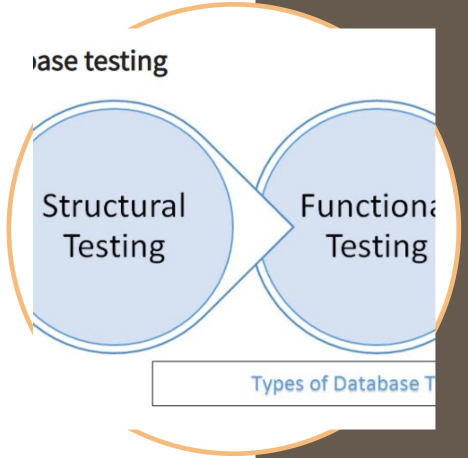
- **Coding** standard conventions.
- **Exception** and error handling.
- **Cover all the conditions**/loops by applying the required input data did properly apply the TRIM operation.
- **Provides** the end user with the required result (Manual).
- **Ensures** the table fields are **being updated** as required by the application under test.
- **Enables** the implicit invoking of the **required triggers.**
- **Presence** of any unused Stored Procedures.
- **Null condition** which can be done at the database level.
- **Integration** of the **stored procedure** modules as per as the requirements of the application under test.



1. Database Structural Testing

D. Trigger testing

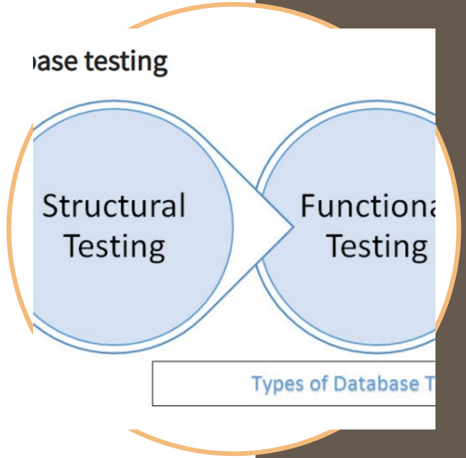
- Coding conventions have been followed during the coding phase of the Triggers.
- The triggers executed for the respective DML transactions have fulfilled the required conditions.
- The **trigger updates** the data correctly once they have been executed.
- **Validation of the required Update /Insert /Delete** triggers functionality in the realm of the application under test.



DML (Data Manipulation Language)

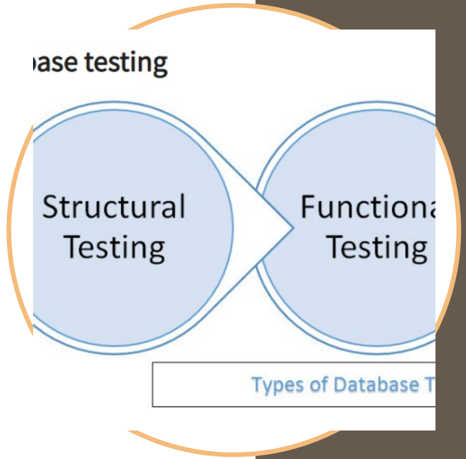
2. Functional Database Testing

- The testers will need to ensure that all of the transactions & operations as performed by the end users are consistent with the **requirement specifications.** What to test for **(19):**
  1. **Field is mandatory** while allowing NULL values in that field.
  2. **The length** of each field is the correct size?
  3. **Similar fields** have same names across tables?
  4. Are there any **computed** fields present in the Database?



2. Functional Database Testing…

5. Is the data **logically** well organized.
6. Is the data **stored in the tables correct** and as per the requirements.
7. Are there any **unnecessary data present** in the application under test.
8. Has the data been **stored as per as the requirement** with respect to data which has been updated from the user interface.
9. Has the **TRIM operations performed** on the data before inserting data into the database under test.

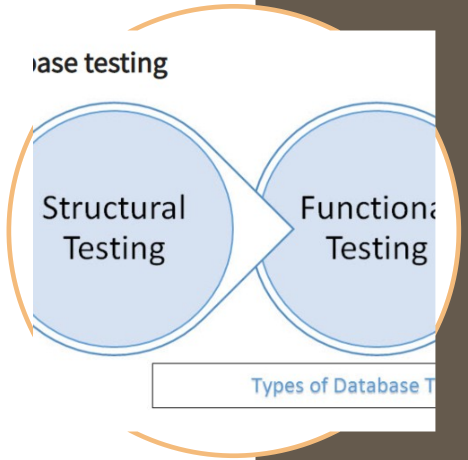


2. Functional Database Testing

10. Has the data been properly **committed** if the transaction has been successfully executed as per the requirements.

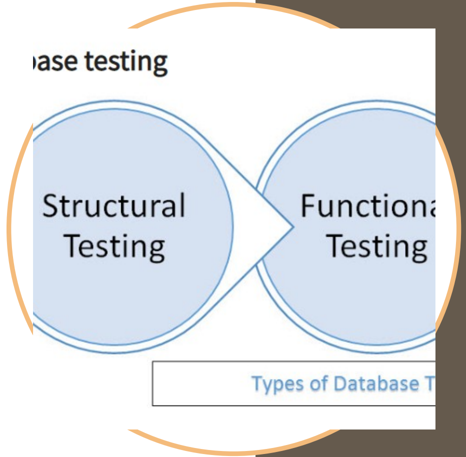
11. Can the data be **rolled back** successfully if the transaction has not executed successfully by the end user.

12. Whether **all the transactions** have been **executed** by using the required design procedures as specified by the system requirement.



2. Functional Database Testing…

13. Login and user security: The application **prevents** the user to proceed further in the application in case of a
    - Invalid username but valid password
    - Valid username but invalid password
    - Invalid username and invalid password
    - Valid username and a valid password (success)
14. The user **is allowed** to perform only those specific operations which are specified by the Functional Requirements.
15. The data is secured from **unauthorized** access.
16. Are different **user roles created with different permissions?**

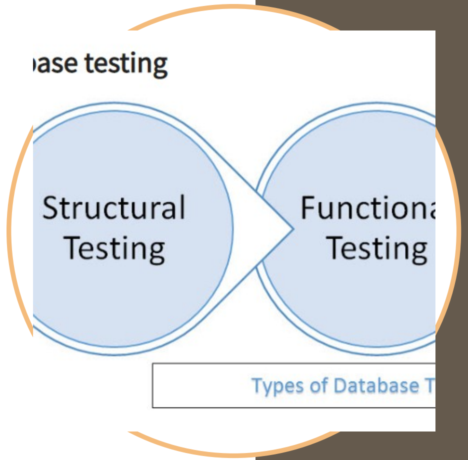


2. Functional Database Testing

17. Do all **users have required levels of access** on the specified Database as required by the business specifications.

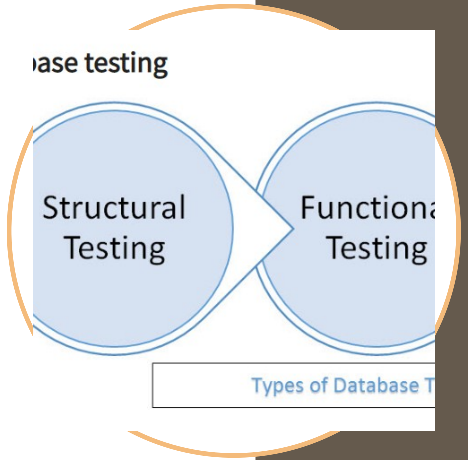
18. Check that sensitive **data like passwords, credit card numbers are encrypted** and not stored as plain text in database.

19. Ensure <u>all accounts </u>should have passwords that are **complex** and not easily guessed.



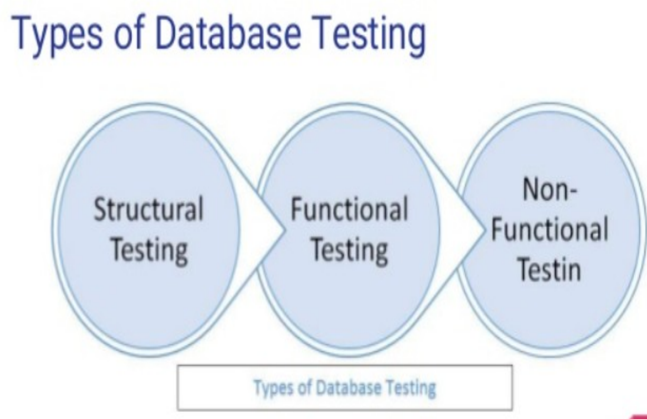
3. Non Functional Database Testing

- The Non Functional Testing can be
  - Load testing
  - Stress Testing
  - Security Testing
  - Compatibility Testing
- The **database** must be tested functionally, **along with load/stress testing** to ensure security and performance.
  - Minimum System Requirements
    - Ensuring that the minimum System requirements have been met per the requirements.



More on Database Load and Stress Testing

- **Overhead is great** but the results are very valuable.
- Should include the most **frequently used user transactions.**
- The observation of the effective **fetch** times.
- Stress test to identify the **breakdown** points.
- Test data **generation** should be considered.
- Should be able to **restore the database** after the test to its original state for retests.
- Tools used: **Load Runner and JMeter**



ACID Properties Test

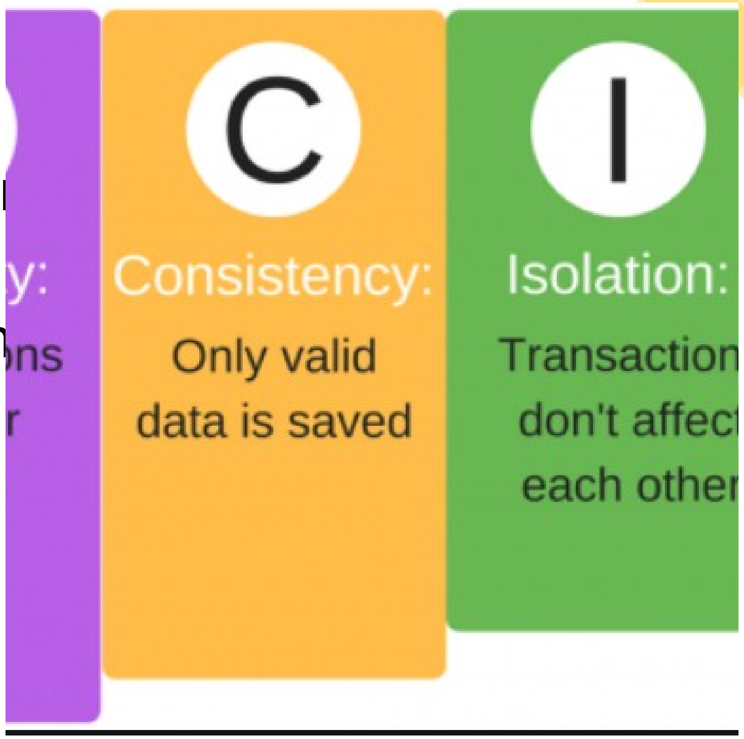
- ACID is an acronym that stands for **Atomicity, Consistency, Isolation and Durability:**
- Atomicity

  You guarantee that either all of the **transaction succeeds o** **none of it does**.

    If one part of the transaction fails, the whole transaction fails. With atomicity, it's either "all or nothing".
- Consistency

  You guarantee that all **data will be consistent**.

    All data will be valid according to all defined rules, including any constraints, cascades, and triggers that have been applied on the database.



ACID Properties Test

- **Isolation**

  Guarantees that all transactions will occur in isolation. No transaction will be affected by any other transaction.

  - So a transaction cannot read data from any other transactio that has not yet completed.

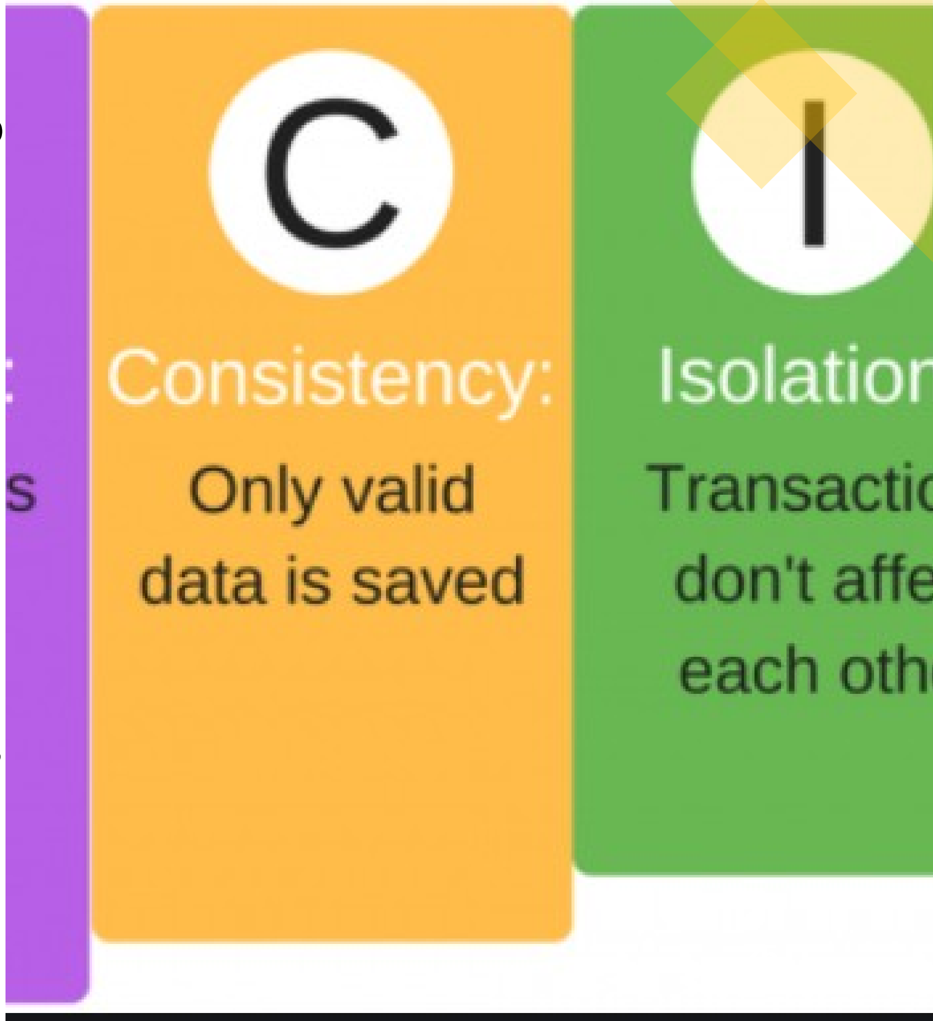
- **Durability**

  Once a transaction is committed, it will remain in the system – even if there's a system crash immediately following the transaction.

  Any changes from the transaction must be stored permanently.

  If the system tells the user that the transaction has succeeded, **the transaction must have, in fact, succeeded**.

https://database.guide/what-is-acid-in-databases/



Open Source Database Testing Tools

1. **Database Benchmark -** designed to **stress test databases** with large data flows.
2. **Database Rider -** aims to bring DBUnit closer to your JUnit tests so database testing will feel like a breeze!
3. **DbFit -** Supports easy **test-driven development** of your database code.
4. **Dbstress -** performance and stress testing tool written in Scala and Akka.
5. **DbUnit -** is a JUnit extension targeted at database-driven projects that, among other things, puts your database into a known state between test runs.
6. **DB Test Driven -** Database test-driven is an open-source unit testing framework for database test-driven development.
7. **HammerDB -** is an open-source database load testing and benchmarking tool.

https://www.softwaretestingmagazine.com/tools/open-source-database-testing-tools/

DB Automation Tools

The best automation tool for **Database testing**:

1. **DataGrip -** a powerful GUI tool for SQL. Smart code completion, on-the-fly analysis, quick-fixes, refactorings that work in SQL files, and more.

2. **Datafactory -** a commercial database testing tool works as data generator and data manager for database testing.

3. **Mockupdata -** Rapidly generates huge amount of data and examines multiple tables at a time for relationship along with foreign key

DB Automation Tools

4. **DTM Data Generator -** is a commercial tool for generating data rows and schema objects for database testing
    - Supports Load Testing, Usability Testing and Performance Testing on database.

5. **SQL test -** uses open-source tSQLt framework, **views, stored procedures and functions**.
    - Allows to run unit tests for SQL Server databases.

6. **tSQLt -** is specifically designed as commercial database unit testing framework dedicated to Microsoft SQL Server

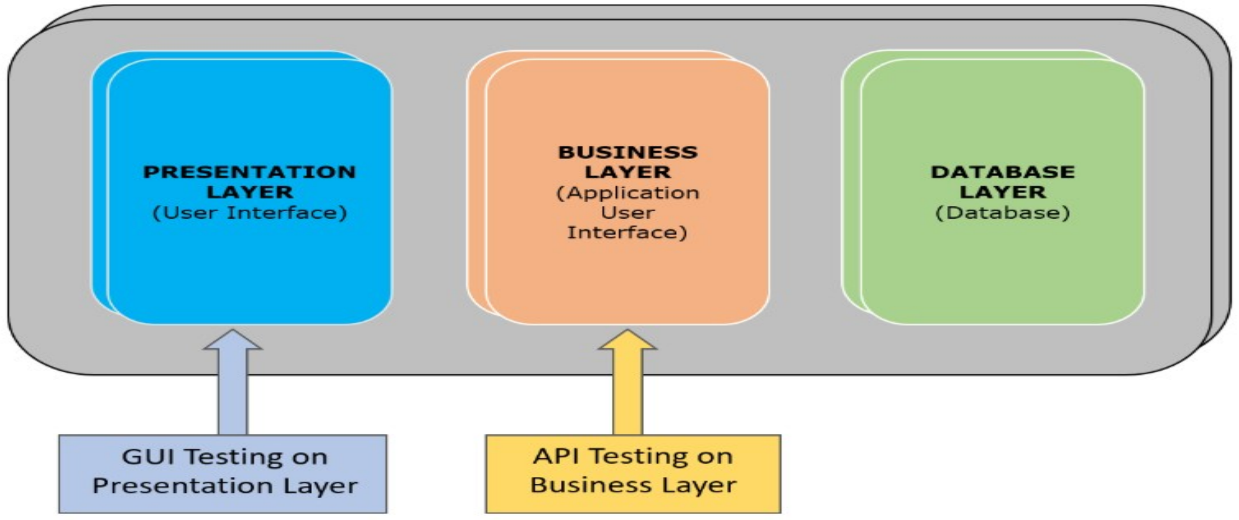
References

- https://www.softwaretestingmagazine.com/tools/open-source-database-testing-tools/
- http://www.agiledata.org/essays/databaseTesting.html
- http://www.methodsandtools.com/tools/dbfit.php
- https://www.guru99.com/data-testing.html
- https://www.studytonight.com/dbms/overview-of-dbms.php
- https://searchsqlserver.techtarget.com/definition/database-management-system
- https://www.quora.com/What-is-the-best-automation-tool-for-database-testing

API (Application Programming Interface) Testing

What is API Testing?

(WebServices vs. API)

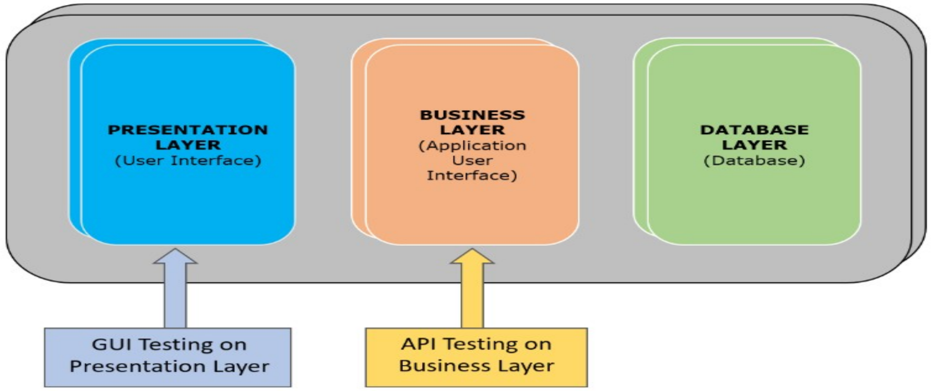

PRESENTATION LAYER
(User Interface)
BUSINESS LAYER
(Application User Interface)
DATABASE LAYER
(Database)
GUI Testing on Presentation Layer
API Testing on Business Layer

API Testing (Application Programming Interface)

定义

- API testing is a software testing type which focuses on determining if the developed APIs meet the following expectations of the Application:
  - **Functionality**
  - **Reliability**
  - **Performance**
  - **Security**
- The API Testing is done at the **Messaging Layer**.
- It is a **crucial component of** a success CI/DevOps practice.

The number of API testers automating more than 50% of their tests is expected to grow by 30% (from 59% to 77%) in the next two years.


PRESENTATION LAYER (User Interface)
BUSINESS LAYER (Application User Interface)
DATABASE LAYER (Database)
GUI Testing on Presentation Layer
API Testing on Business Layer

Definition: Continuous Integration (**CI**) aims at integrating the work produ
repository ...

Web Services

The term **Web Services** describes a standardized way of integrating **Web-based** applications using the XML, SOAP, WSDL and UDDI open standards over an Internet Protocol backbone.

A web service enables communication among various applications installed on different devices by using open standards

Developers can then add the **Web Service** to a GUI (such as a **Web** page or an executable program) to offer specific functionality to users.

API vs Web Service

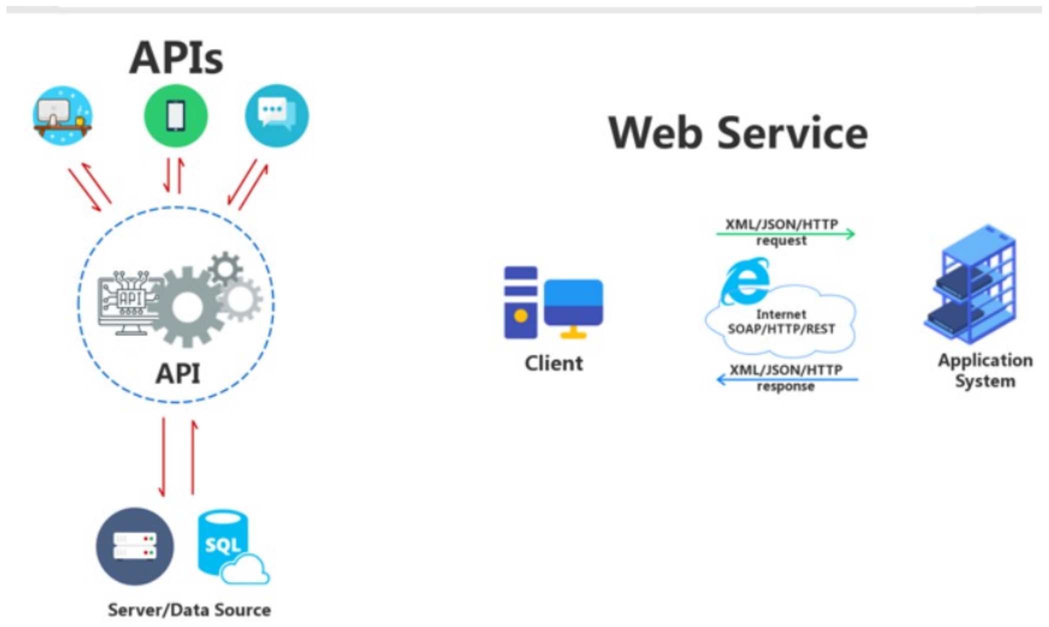
API and Web Service serve as a means of communication. The only difference is that

- **An API acts as an interface** between **two different applications** so that they can communicate with each other.
- An API is a method by which the **third-party vendors** can write programs that interface easily with other programs.

A Web service facilitates interaction between two machines over a network:

- Is designed to have an interface that is depicted in a **machine-processable** format usually specified in **Web Service Description Language (WSDL).**
- Typically, "HTTP" is the most commonly used protocol for communication.
- It also uses **SOAP, REST, and XML-RPC** as a means of communication.
- All Web services are APIs but all APIs are not Web services.

Web Services Components & Operation



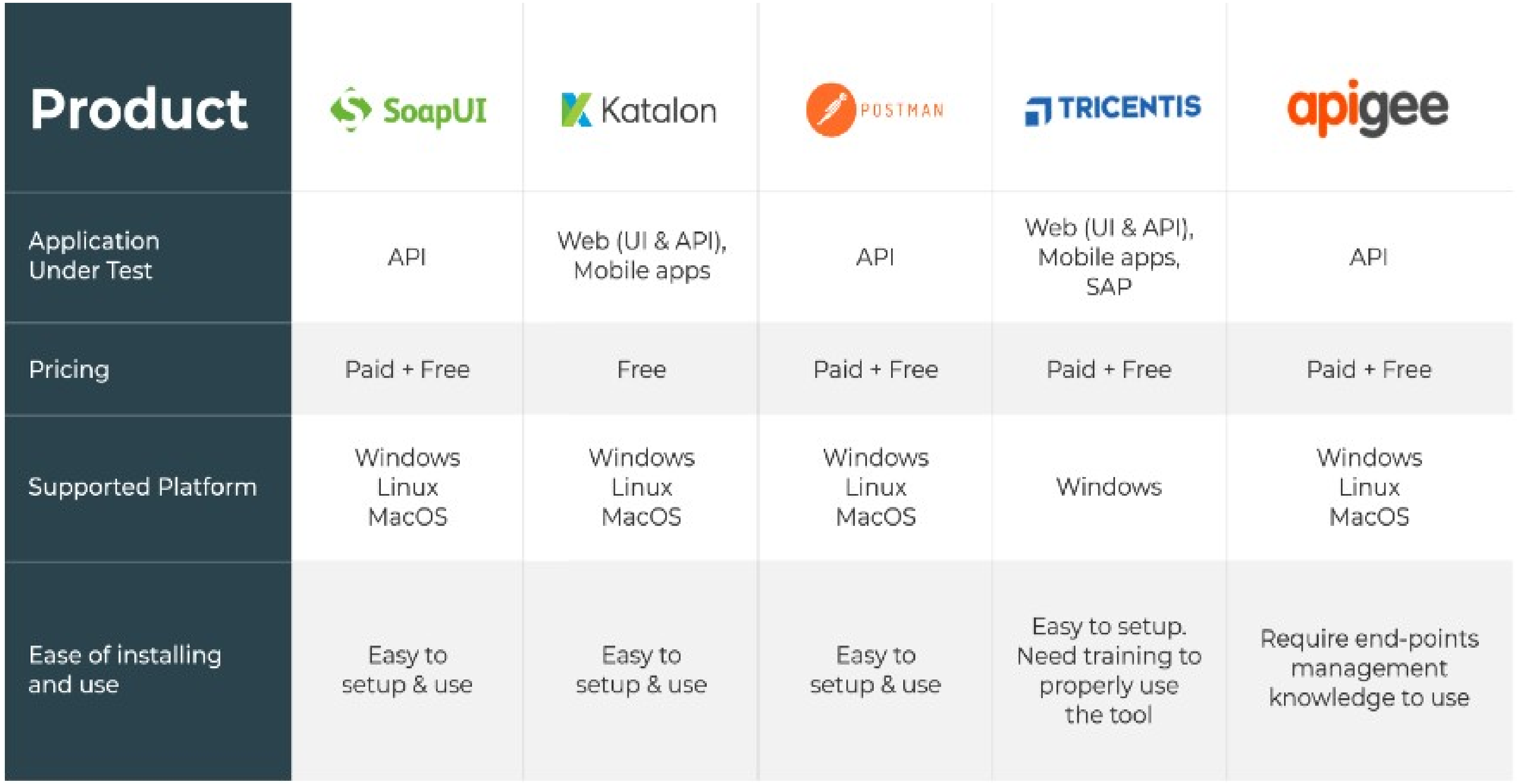
Web Services Components:

- The basic web services platform is XML message format and HTTP request and response. All the standard web services work using the following components
    - SOAP (Simple Object Access Protocol)
    - UDDI (Universal Description, Discovery and Integration)
    - WSDL (Web Services Description Language)

Web Services Operation:

- A web service enables communication among various applications installed on different devices by using open standards:
    - HTML page to send the request and render the received the response via HTTP/HTTPS protocol.
    - XML to tag the request and response data.
    - SOAP to transfer a message over the web.
    - WSDL to describe the availability of web service.

Top 5 API Testing Tools for 2018



Product	SoapUI	Katalon	POSTMAN	TRICENTIS	apigee
Application Under Test	API	Web (UI & API), Mobile apps	API	Web (UI & API), Mobile apps, SAP	API
Pricing	Paid + Free	Free	Paid + Free	Paid + Free	Paid + Free
Supported Platform	Windows Linux MacOS	Windows Linux MacOS	Windows Linux MacOS	Windows	Windows Linux MacOS
Ease of installing and use	Easy to setup & use	Easy to setup & use	Easy to setup & use	Easy to setup. Need training to properly use the tool	Require end-points management knowledge to use

https://medium.com/@alicealdaine/top-10-api-testing-tools-rest-soap-services-5395cb03cfa9

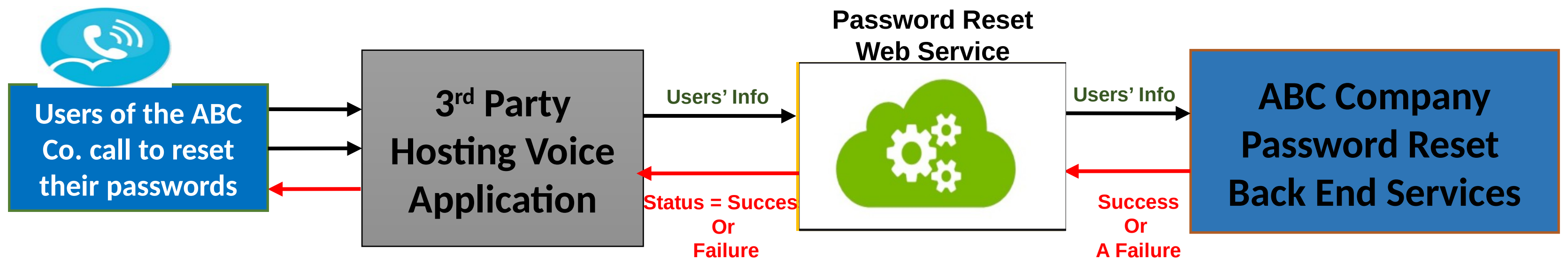
Top 16 Web Service Testing tools

Here is the list of the top online tools for Web Services testing:

1. SoapUI Pro
2. TestMaker
3. WebInject
4. SOAPSonar
5. wizdl
6. Stylus Studio
7. TestingWhiz
8. SOAtest
9. JMeter
10. Storm
11. Postman
12. vRest
13. HttpMaster
14. Runscope
15. Rapise
16. LoadUI NG Pro

An Example of a Web Service Testing

In this example, many users call into this system to reset their passwords using a voice recognition application


Users of the ABC Co. call to reset their passwords
3rd Party Hosting Voice Application
Users' Info
Status = Succes Or Failure
Password Reset Web Service
Users' Info
Success Or A Failure
ABC Company Password Reset Back End Services

All RESTful service responses going to have "status" property, the possible values are "SUCCESS", "FAILURE". In case of failure, the 3rd Party Hosting Voice Application is notified with one of the error code messages listed below:

Error Code	Error Message	Scenario
1010	Password reset service is not available	System unavailable
1011	User account is not active	Not active user or not valid user
1012	This function is not available for this account	Black listed user or physician
1013	Password reset attempt failed	Internal error or sync error
1014	Password complexity failed	Password does not adhere to Fresenius password policy

References

- https://www.softwaretestinghelp.com/web-services-testing-tools/
- https://medium.com/@alicealdaine/top-10-api-testing-tools-rest-soap-services-5395cb03cfa9
- https://www.inflectra.com/Rapise/HelpReadingPane.ashx?filename=Rapise2.0.0.chm&href=tutorial_web_services_rest.htm
- **RESTful Web Services:** *Representational State Transfer (**REST**) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web **service** induce desirable properties, such as performance, scalability, and modifiability, that enable **services** to work best on the Web.*