Module – 3 (Testing on Live Application)

* **What is RDBMS?**

RDBMS stands for **Relational Database Management System.** RDBMS is a program used to maintain a relational database. RDBMS is the basis for all modern database systems such as MySQL, Microsoft SQL Server, Oracle, and Microsoft Access. RDBMS uses [**SQL queries**](https://www.w3schools.com/sql/default.asp) to access the data in the database.

* **How Many constraints and describes it self**

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

* [NOT NULL](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value
* [UNIQUE](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different
* [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
* [FOREIGN KEY](https://www.w3schools.com/sql/sql_foreignkey.asp) - Prevents actions that would destroy links between tables
* [CHECK](https://www.w3schools.com/sql/sql_check.asp) - Ensures that the values in a column satisfies a specific condition
* [DEFAULT](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column if no value is specified
* [CREATE INDEX](https://www.w3schools.com/sql/sql_create_index.asp) - Used to create and retrieve data from the database very quickly
* **Difference between RDBMS vs DBMS**

| DBMS | RDBMS |
| --- | --- |
| DBMS stores data as file. | RDBMS stores data in tabular form. |
| Data elements need to access individually. | Multiple data elements can be accessed at the same time. |
| No relationship between data. | Data is stored in the form of tables which are related to each other. |
| Normalization is not present. | Normalization is present. |
| DBMS does not support distributed database. | RDBMS supports distributed database. |
| It stores data in either a navigational or hierarchical form. | It uses a tabular structure where the headers are the column names, and the rows contain corresponding values. |
| It deals with small quantity of data. | It deals with large amount of data. |
| Data redundancy is common in this model. | Keys and indexes do not allow Data redundancy. |
| It is used for small organization and deal with small data. | It is used to handle large amount of data. |
| It supports single user. | It supports multiple users. |
| Data fetching is slower for the large amount of data. | Data fetching is fast because of relational approach. |
| The data in a DBMS is subject to low security levels with regards to data manipulation. | There exists multiple levels of data security in a RDBMS. |
| Low software and hardware necessities. | Higher software and hardware necessities. |
| Examples: XML, Window Registry, etc. | Examples: MySQL, PostgreSQL, SQL Server, Oracle, Microsoft Access etc. |

* **What is API Testing?**

**API TESTING** is a software testing type that validates Application Programming Interfaces (APIs). The purpose of API Testing is to check the functionality, reliability, performance, and security of the programming interfaces. In API Testing, instead of using standard user inputs(keyboard) and outputs, you use software to send calls to the API, get output, and note down the system’s response. API tests are very different from GUI Tests and won’t concentrate on the look and feel of an application. It mainly concentrates on the business logic layer of the software architecture.

* **Type of API Testing**

**‍Unit testing**

Unit testing are tests that are written to automatically run with every build of the application. They are written close to the code and should pass when running a build of the application. What type of code coverage is required for the APIs depends on the risk the API carries and what functionalities it holds. Good unit testing is like a good foundation and this aspect should be well thought over as it will carry the rest of the testing effort later down the line.

## Integration Testing

Our APIs are no separated component of a system. Instead they work to integrate all of the moving parts of a system and all of this integration requires proper testing of course. Not only do we need to make sure we are passing on and taking in the correct parameters with their correct constraints but we also need to make sure that we sanitize the incoming traffic in a logical way.

**Performance testing**

Performance testing is one of the non-functional things that is most often overlooked and will most often cause trouble. Be very aware that our testing environments often only contain a fraction of the data that production does so any process that runs might take a lot more time in a real scenario. Performance testing should take everything into account including spikes in traffic and multiple heavy processes running at the same time and interfering with one another. Also be aware that the environment that is used for testing needs to be similar to a production environment to get reliable results.

**Load testing**

Load testing is very similar to performance testing because we are trying to emulate production like traffic but instead of creating spikes in traffic, we want to emulate a constant steady stream of traffic at a normally expected rate. This is to ensure that the API's do not contain memory leaks or other similar defects that might cause issues after running for a prolonged period of time.

Performance testing is one of the non-functional things that is most often overlooked and will most often cause trouble. Be very aware that our testing environments often only contain a fraction of the data that production does so any process that runs might take a lot more time in a real scenario. Performance testing should take everything into account including spikes in traffic and multiple heavy processes running at the same time and interfering with one another. Also be aware that the environment that is used for testing needs to be similar to a production environment to get reliable results.

**Runtime error detection**

During all of these tests, we want to make sure that we have Runtime error detection enabled. This technology allows our APIs to report back any defects that occur while it's in operation.

**Security testing**

This type of testing is very important but often not budgeted well. We need to ensure that proper security testing occurs based on a risk analysis. This type of testing needs to occur by a trained professional as it's very easy to miss things but it does not stop there. Every developer should in essence be a little bit of a security tester when it comes to APIs as they are often the first line of defense into our infrastructure.

Security testing has many aliases such as [pentesting](https://www.wallarm.com/what/penetration-testing-types), PEN testing or penetration testing and needs to focus on several aspects when it comes to api testing. The entry points of our APIs need to be taken into account but also the flow of data and any shadow APIs that are no longer in user but are still in operation.

**Interoperability testing**

Working with 3rd party software providers or even older versions of our software is not something we can take for granted. We need to create a section in our test plan where we describe what testing will be done. Severity and priority of potentials defects need to be discussed based on prior experiences and they need to help define how API testing occurs when the APIs interact with

**Fuzz tests**

As a final test before we validate our application we need to fuzz all the endpoints of our APIs. When [fuzzing](https://www.wallarm.com/what/how-to-hack-api-in-60-minutes-with-open-source#how_to_fuzz) we will send random data to those API endpoints and we need to carefully inspect the results. Our server should not crash from this unexpected traffic and it should not display any odd behavior. Based on a risk analysis, fuzz testing might be performed much more structured or not at all.

**Validation Testing**

During validation testing we need to ensure that the software meets the business requirements. The testers need to evaluate if the test execution results match what is expected and required by the test plan. They will subsequently guide the business users in their User Acceptance Testing (UAT) in which they ensure the stakeholders execute the pre-built testing scenarios and report any deviations on what was expected. If the software is approved.

* **What is Responsive Testing?**

Responsive testing is a process that renders web pages on viewports of multiple devices using CSS media queries based on the user device where the website is accessed. In simple terms, responsive testing ensures how responsive web design is optimized well for all types of screen sizes and resolutions.

* **Which types of tools are available for Responsive Testing**

7 Premium Tools for Performing Responsive Test

1. **Test sigma**
2. **Responsinator**
3. **Screenfly**
4. **Lambda Test**
5. **Am I Responsive?**
6. **Cross Browser Testing**
7. **Browser stack**

* **What re the Full-Form of .IPA, .APK,**

The full Form of .IPA, ,APK is below:

IPA: iOS APP Store Package  
APK: Android Application Package file

* **How to create the step to turn OFF/ON the developer option in Android phone?**

**To OFF/ON the Developer options menu:**

1. Go to "Settings"
2. Tap "About device" or "About phone"
3. Tap “Software information”
4. Tap “Build number” seven times. ...
5. Enter your pattern, PIN or password to enable the Developer options menu.
6. The "Developer options" menu will now appear in your Settings menu.
7. Depending on your device, it may appear under Settings > General > Developer options.