**Integration testing**is the process of testing the interface between two software units or modules. It focuses on determining the correctness of the interface. The purpose of integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit-tested, integration testing is performed.

Integration testing is a [software testing technique](https://www.geeksforgeeks.org/software-testing-techniques/)that focuses on verifying the interactions and data exchange between different components or modules of a software application. The goal of integration testing is to identify any problems or bugs that arise when different components are combined and interact with each other. Integration testing is typically performed after unit testing and before system testing. It helps to identify and resolve integration issues early in the development cycle, reducing the risk of more severe and costly problems later on.

**Approaches of Integration Testing**

Incremental integration testing can be further divided into 3 smaller approaches, each also comes with its own advantages and disadvantages that QA teams need to carefully consider for their projects. These approaches are named based on the level of impact of the software components being integrated have on the overall system, including:

* **Bottom-up approach:** perform testing for low-level components first, then gradually move to higher-level components.
* **Top-down approach:** perform testing for high-level components first, then gradually move to lower-level components.
* **Hybrid approach:** combining the two former approaches

To better understand these 3 concepts, we must first define **low-level components** and **high-level components**.

**Applications of Integration Testing**

1. **Identify the components:**Identify the individual components of your application that need to be integrated. This could include the frontend, backend, database, and any third-party services.
2. **Create a test plan:**Develop a test plan that outlines the scenarios and test cases that need to be executed to validate the integration points between the different components. This could include testing data flow, communication protocols, and error handling.
3. **Set up test environment:**Set up a test environment that mirrors the production environment as closely as possible. This will help ensure that the results of your integration tests are accurate and reliable.
4. **Execute the tests:**Execute the tests outlined in your test plan, starting with the most critical and complex scenarios. Be sure to log any defects or issues that you encounter during testing.