Code Academy

Lecturer

Rokas Slaboševičius

Integration Testing in C#

Data

Today you will learn

01 Integration Testing

02 Types of integration testing

Integration Testing

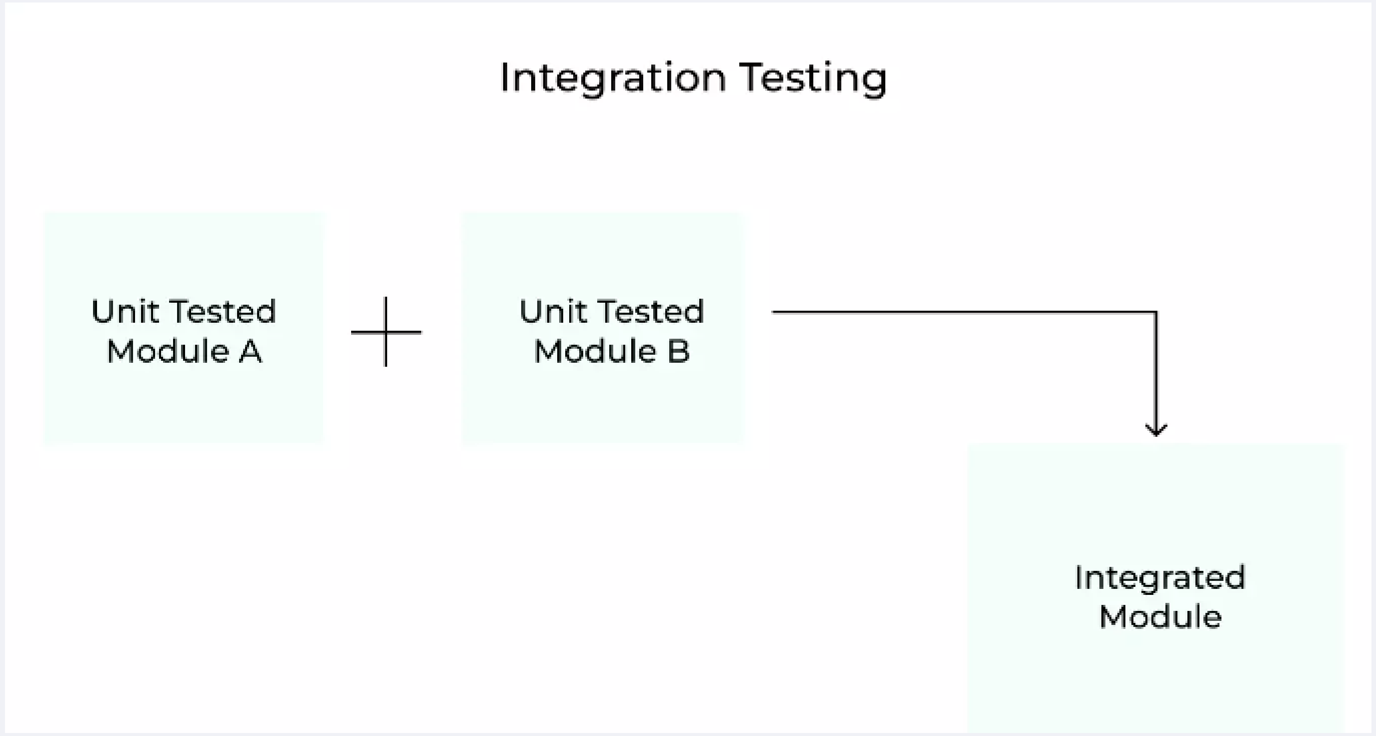
- Integration testing focuses on verifying the functionality and performance of various software modules when combined. Unlike unit testing, which validates individual components in isolation, integration testing seeks to uncover defects in the interactions between integrated units.

- It's ensuring that different musicians in an orchestra play in harmony, not just individually well.

Integration Testing

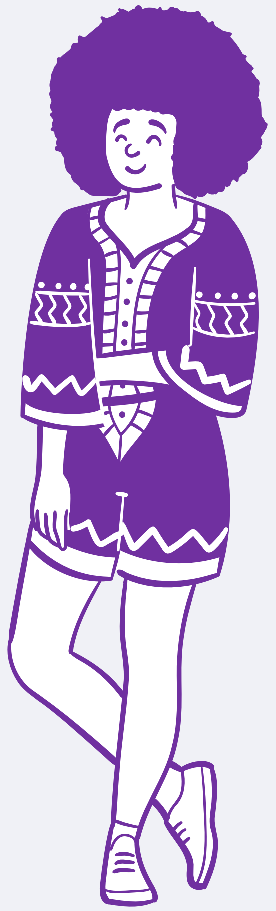
- Consider a web application where the front-end and back-end components work perfectly alone but fail when data is passed between them. Integration testing aims to identify and solve these mismatches early on.

- It's about confirming that all parts of the software communicate correctly, preventing costly bugs in later stages of development.

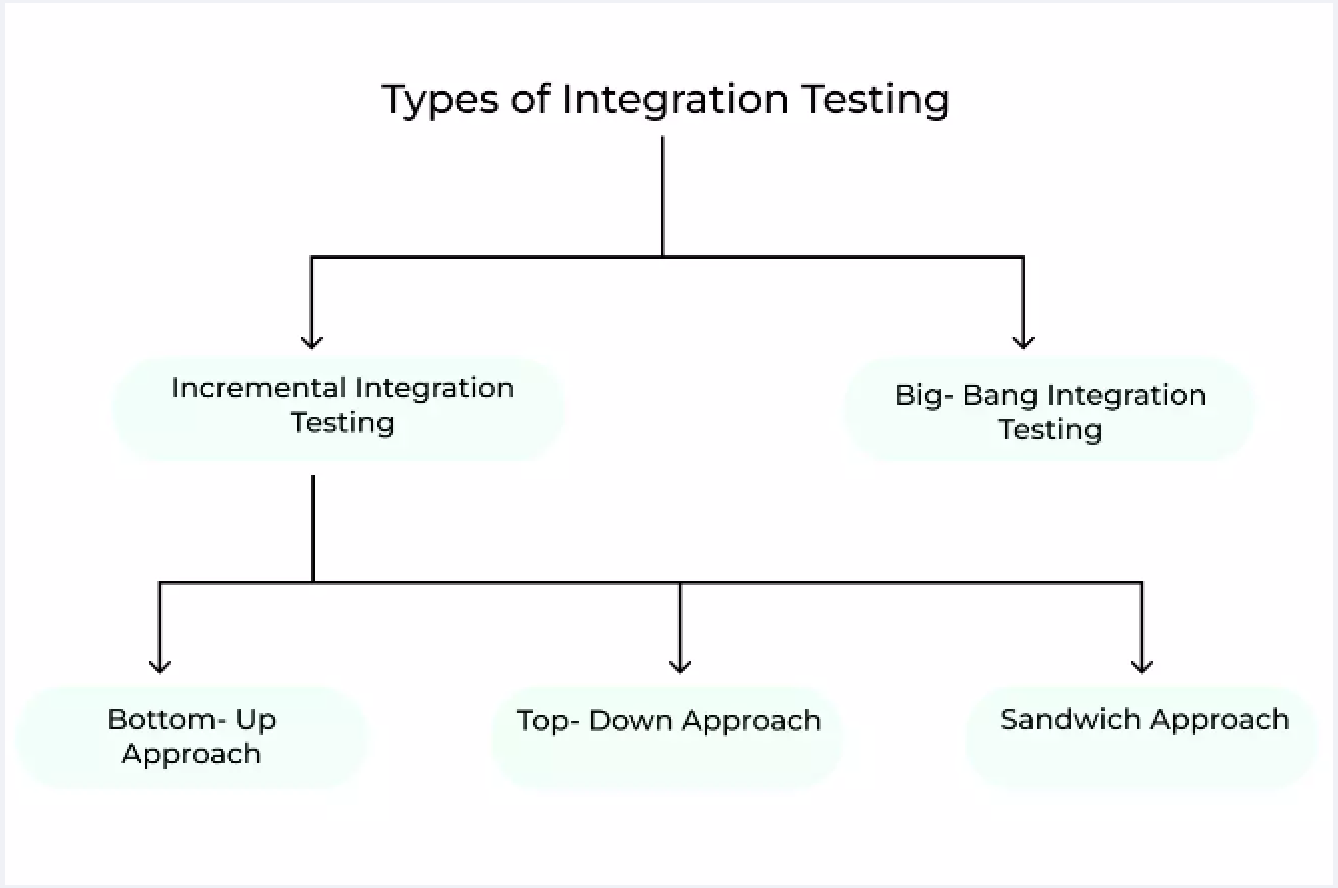


Stubs and Drivers

- **Stubs Example:** In a flight booking system, a stub could simulate the payment processing module to test the booking interface without needing the actual payment service.
- **Drivers Example:** A driver could call the database access layer directly to test it in isolation from the rest of the system.



Types of Integration Testing


Types of Integration Testing
Incremental Integration Testing
Big- Bang Integration Testing
Bottom- Up Approach
Top- Down Approach
Sandwich Approach

Big Bang Integration Testing

- Imagine integrating a complete e-commerce system all at once and then beginning tests. The complexity could lead to numerous errors popping up, making it hard to pinpoint their origins.
- While it's straightforward, the risk of discovering major issues late in the process makes it less favored for complex projects.

Top Down Integration Testing

- Testing starts with the user interface and moves down to backend services, using stubs to simulate the lower modules until they're ready for testing.
- This method helps in identifying issues in the upper levels of the software stack early, making it suitable for projects where top-level functionality is a priority.

Bottom Up Integration Testing

- Begin testing with database operations, moving up to API services, and finally the user interface, using drivers to simulate higher-level modules.
- This approach often leads to a more stable foundation, as the core services are tested thoroughly first.

Incremental Integration Testing

- Integrating and testing the login module, followed by the user profile management, and then the transaction processing module, gradually building up the application.
- Combines the advantages of both top-down and bottom-up approaches, facilitating easier fault isolation and providing a more manageable testing process.

Best Practices for Integration Testing

- Start integration testing early
- Test in small batches
- Include negative testing
- Automate testing
- Include performance testing

Task 1

- Write integration tests for Student Information System: