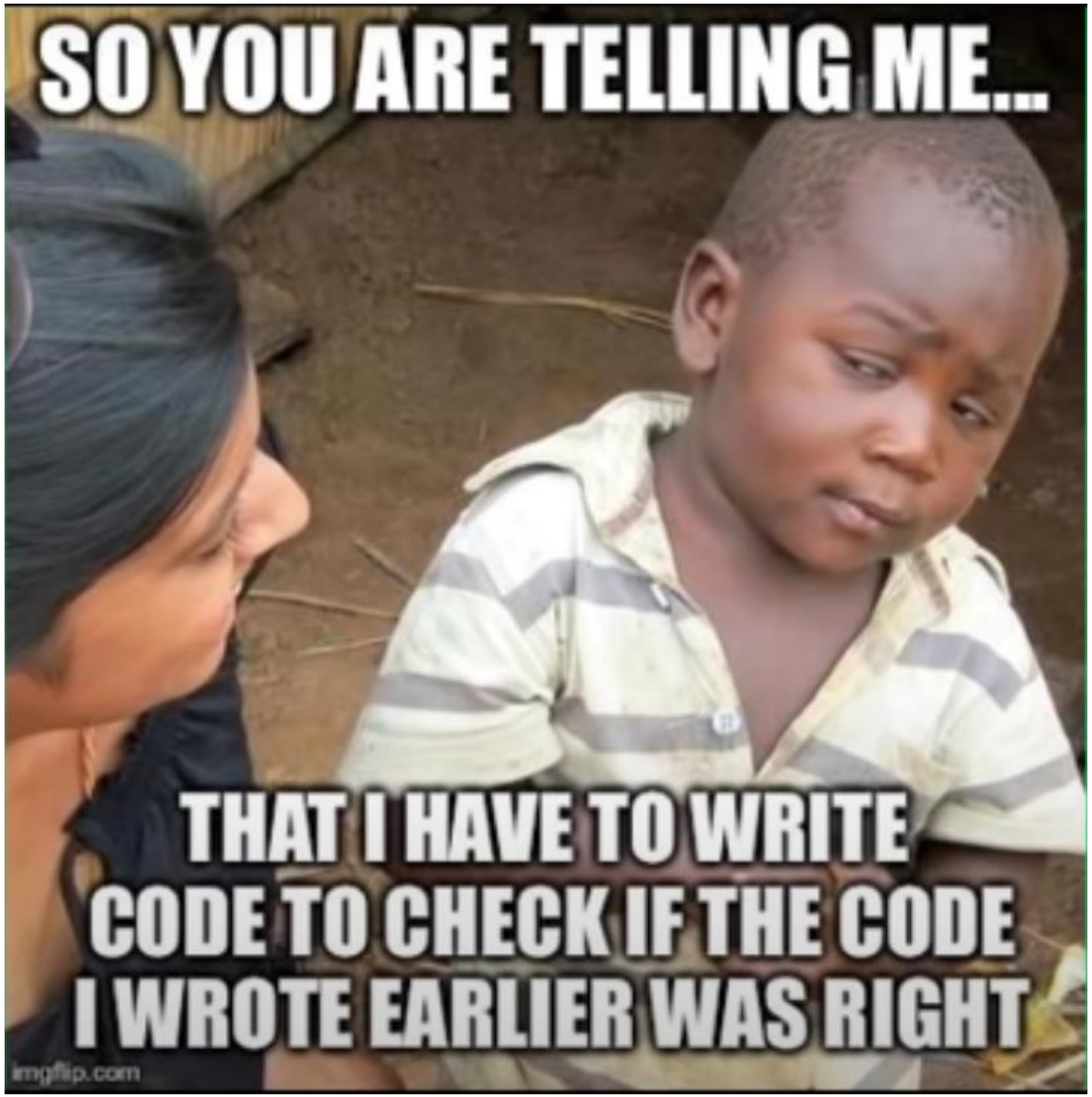
# SOFTWARE TESTING

# What is Software Testing

- Testing in software development is a process in which all parts of an application, are tested for proper operation.
- Tests are ran throughout the development process to ensure quality of service.
- The more frequently tests are ran, the less likely you will get compatibility issues between new features/functionalities and existing ones.
- Testing saves money, adds security and ensures product quality and customer satisfaction.


SO YOU ARE TELLING ME...
THAT I HAVE TO WRITE CODE TO CHECK IF THE CODE I WROTE EARLIER WAS RIGHT
imgflip.com

# DevOps Toolchain

Plan

Planning
Requirement Gathering
Project Management

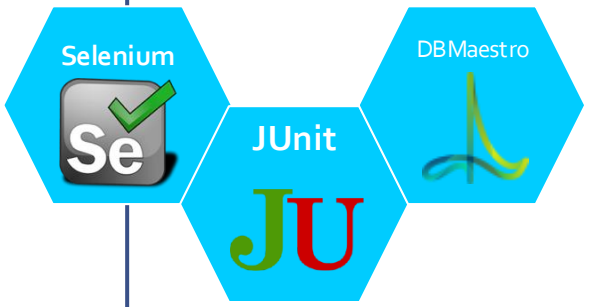
Develop

Issue Tracking
Source Control
Dev Env
CD

Test

Continuous
Testing

Release




Deployment
Config Management
Containers
Scripting

Operate

Monitoring
Issue Discovery

# Important Note

- You should know manual testing to run automated testing
- Automated testing doesn't necessarily require a fancy tool

# Why Creating Tests is Skipped Sometimes?

- It takes a lot of work to create unit tests for all your scenarios
- Common thought: I'll just push my code now and fix the problems along the way

# Why Creating Tests Should not be Skipped?

- As the application grows, errors become more common and harder to fix
- With a team working on a project, it could become difficult to trace down the person responsible for fixing an error once the application is in production
- Running manual tests might be faster for one iteration, but it will become much more time consuming in the long run

# What Tests Should We Run?

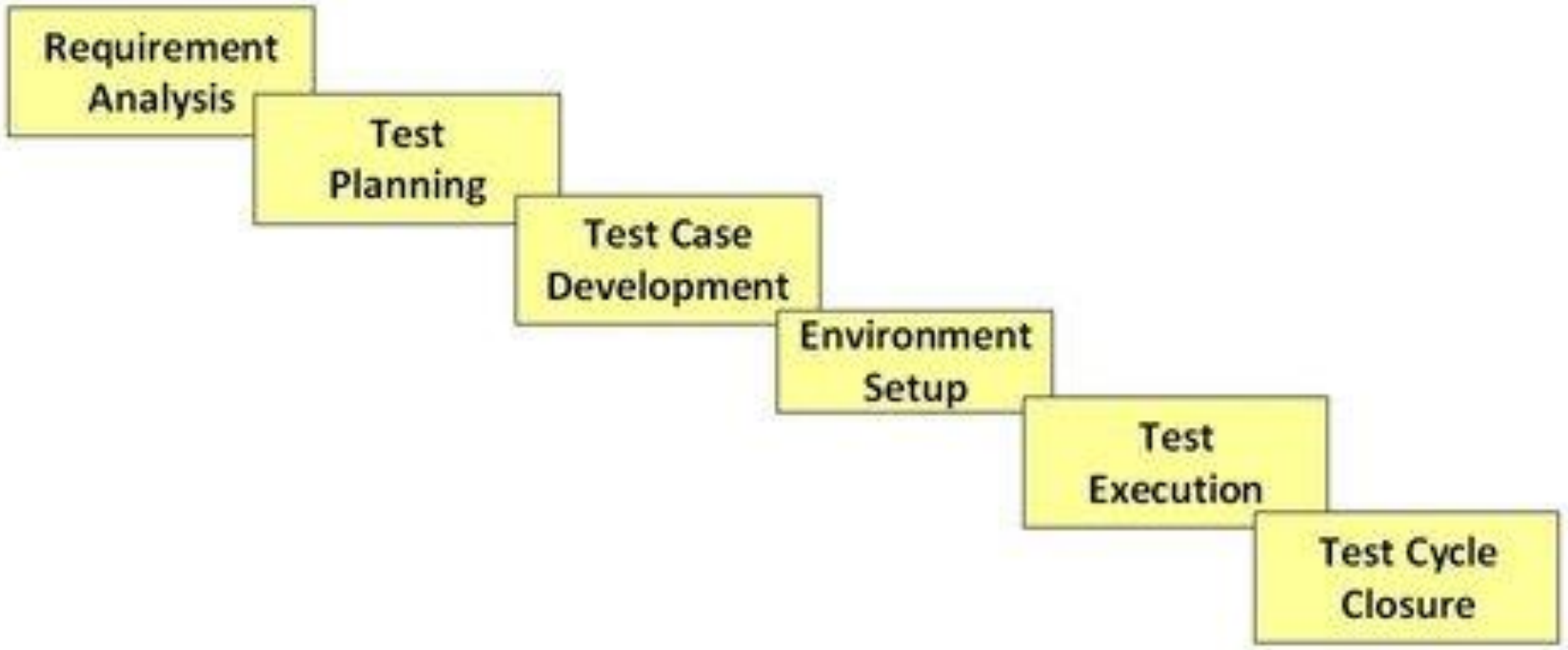
- End-to-End testing gets some mixed feelings, as only 1 error in the testing chain can break the entire test
- Testing too much can be counter-productive
- A preferred approach might be to distribute your testing where you put more emphasis on unit testing
- Example: Mike Wacker, a google developer, suggests in an article he wrote a distribution of 70% unit testing, 20% integration testing, and only 10% end-to-end testing.

# Software Testing Life Cycle (STLC)

- STLC is a series of activities to conduct, which identify the tests that need to be executed and when to execute them, in aims to achieve specific software quality goals.
- It is an integral part of the software development life cycle.
- It starts as soon as the development phase is done.

# Software Testing Life Cycle (STLC)

- STLC is composed of the following phases:



# Software Testing Life Cycle (STLC)

- Requirements Analysis:
  - The test team studies the project requirements to identify testable ones.
  - Types of tests to be performed are identified, and test environment details are prepared.
  - Automation feasibility for the testing project is also done in this stage

# Software Testing Life Cycle (STLC)

- Test Planning:
  - Prepare test strategy documents for various types of tests
  - Select and setup test tools
  - Plan resources and determine roles and responsibilities

# Software Testing Life Cycle (STLC)

- Test case development:
  - Involves the creation and verification of test cases with automation scripts (if applicable) after the testing plan is ready.
  - Create test data (if needed)

# Software Testing Life Cycle (STLC)

- Test environment setup:
  - Decide under what software and hardware conditions to perform testing.
  - Perform smoke tests on the build

# Software Testing Life Cycle (STLC)

- Test execution:
  - Execute tests as planned
  - Document test results
  - Produce defect (bug) reports if present.
  - Track defects until they are resolved

# Software Testing Life Cycle (STLC)

- Test cycle closure:
  - Is the completion of test execution.
  - Evaluate cycle completion time, cost, quality, and other criteria
  - Produce test completion reports.
  - Collect test results.

# Types of Software Testing

- Non-Functional Testing
- Functional Testing

# Types of Software Testing

- Non-Functional Testing:
  - Type of software testing to check non-functional aspects (performance, delay, reliability…)
  - Designed to test the usability of the software
  - Collects measurements for future research and development

# Types of Software Testing

- Some Non-Functional Testing Parameters:
  - Security
  - Reliability
  - Availability
  - Scalability
  - Efficiency
  - Portability
  - …

# Types of Software Testing

- Non-Functional Testing Types:
  - Performance Testing
  - Load Testing
  - Compatibility Testing
  - Usability Testing
  - Stress Testing
  - Scalability Testing
  - Security Testing
  - Disaster Recovery Testing
  - …

# Types of Software Testing

- Functional Testing:
  - Type of software testing that validates the software system against functional requirements
  - The aim is to test each function of the software application.
  - Functional testing mainly involves **black box testing**

# Types of Software Testing

- Functional Testing Steps:
  - Understand functional requirements
  - Identify test input based on the requirements with expected outputs
  - Execute test cases and compare the expected and actual outcomes

# Types of Software Testing

- Functional Testing Types:
  - Unit Testing
  - Integration testing
  - System Testing
  - Acceptance Testing

# Types of Software Testing

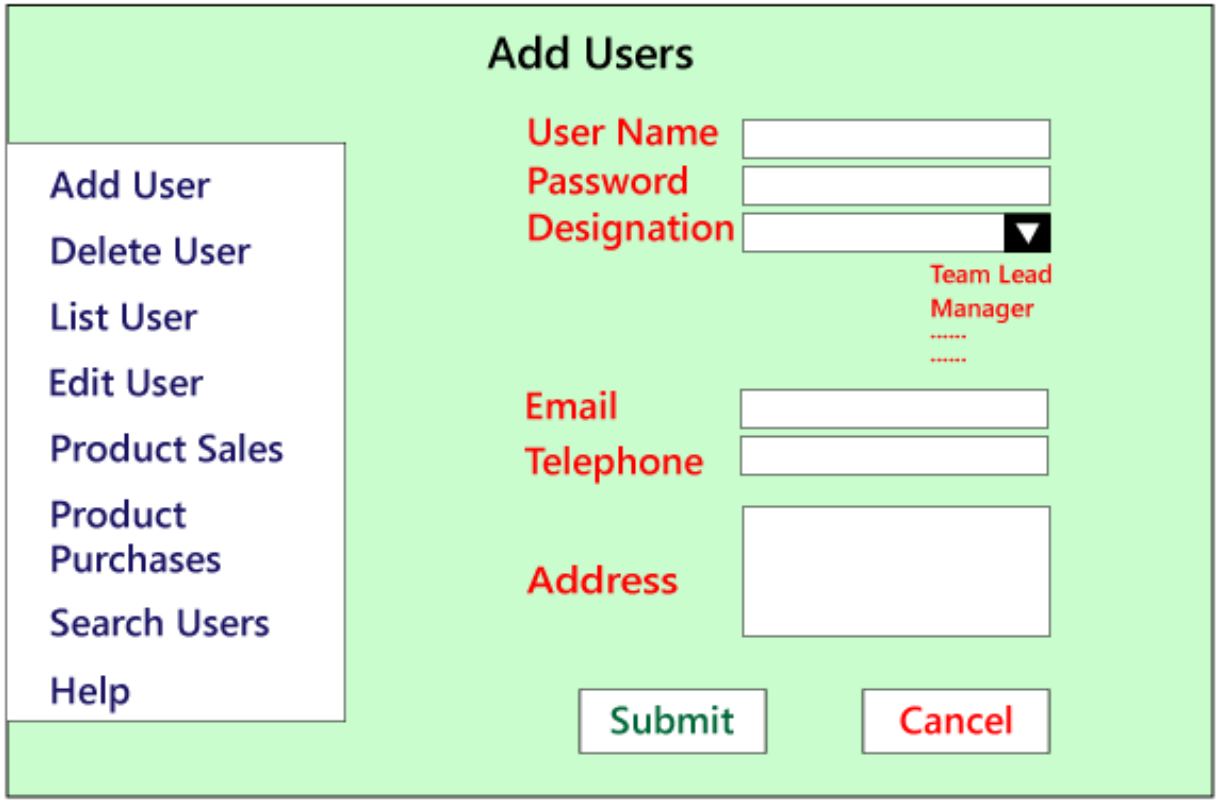
- Unit Testing:
  - Individual units or components of a software are tested independently.
  - The aim is to validate that each unit of the software code performs as intended.
  - Done through the development phase by the developers.
  - It is a **white box testing**

# Types of Software Testing

- Integration Testing:
  - Is a type of testing where software components are integrated and tested as a group.
  - The aim is to detect any compatibility or interaction issues that might exist between these components.

# Types of Software Testing

- Integration Testing Example:



# Types of Software Testing

- System Testing (End-to-End):
  - Is a level of testing that validates the operability of the fully integrated system.
  - Connection to all external entities (databases, APIs…) is also included.
  - Test User Experience with the application.

# Types of Software Testing

- Different Types of System Testing (End-to-End):
  - Usability Testing
  - Load Testing
  - Regression Testing
  - Recovery testing
  - Migration Testing
  - Hardware/Software testing
  - …

# Types of Software Testing

- Acceptance Testing:
  - The system is put in a deployment environment to be tested by the end client and users for acceptance.
  - The aim is to check and evaluate the system's compliance with the business requirements.
  - normally uses the Black Box Testing method and is executed manually.

# Automatic Testing

- Is done through an automation tool
- less time is needed in exploratory tests and more time is needed in maintaining test scripts while increasing overall test coverage
- Well suited for large projects

# Why Automatic Testing?

- Software tests must be repeated often during the development cycle
- For each release of the software, it may be tested on all supported operating systems and/or browsers.
- Manually repeating these tests on every cycle could be very time consuming
- Once created, automatic tests can be repeated over and over without any additional cost.
- The operation of automated tests is also much quicker than manual tests.

# Automatic Testing

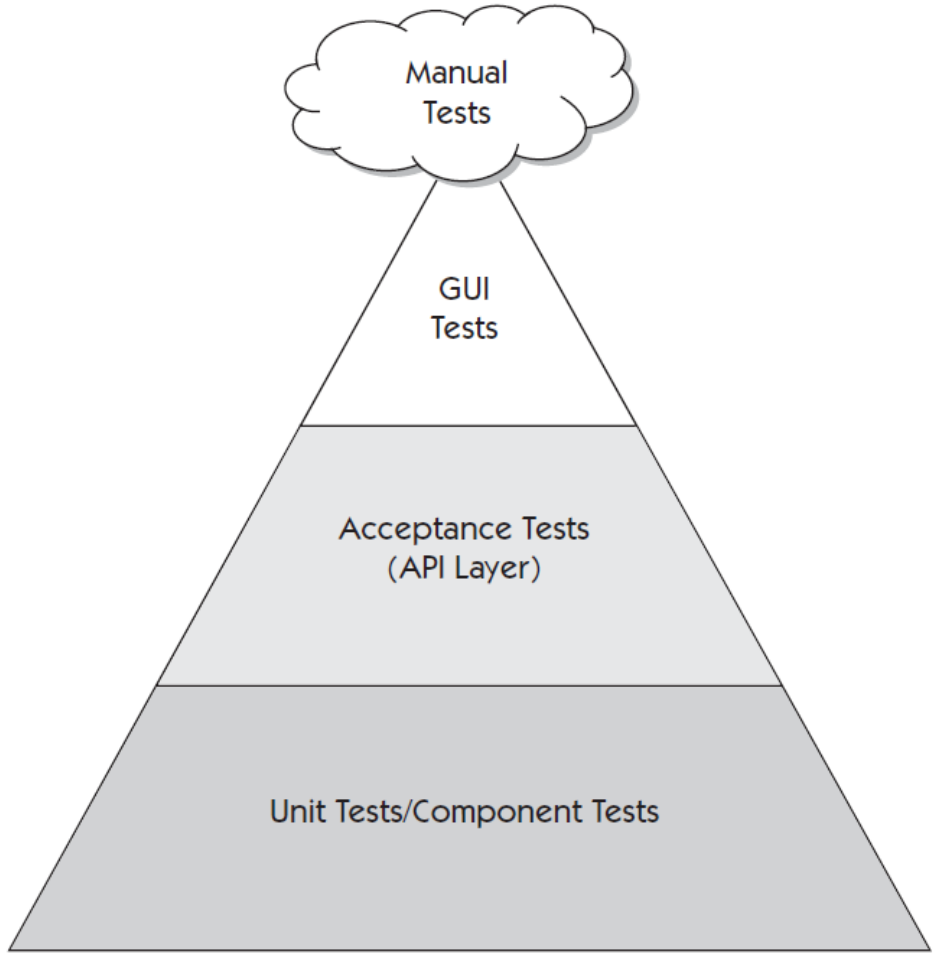
- Agile projects depend on automation.
- Good-enough automation frees the team to deliver high-quality code frequently.
- Automation is a vast topic:
  - writing simple shell scripts
  - setting up session properties
  - creating robust automated tests

# Why Automate

- Manual testing takes too long.
- Manual processes are error prone.
- Automation frees people to do their best work.
- Automated tests give feedback early and often.
- Tests provide documentation.
- Automation can be a good return on investment.

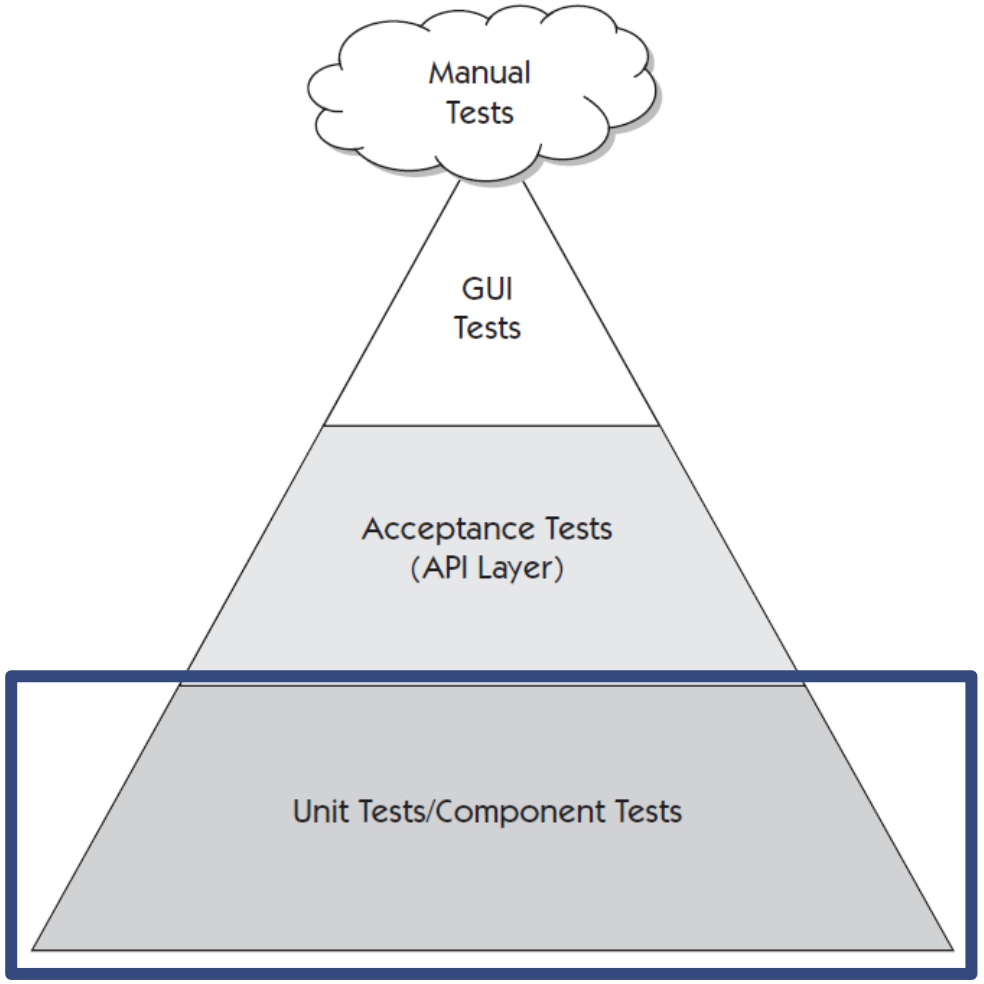
# Test Automation Pyramid

- Shows three different layers of automated tests



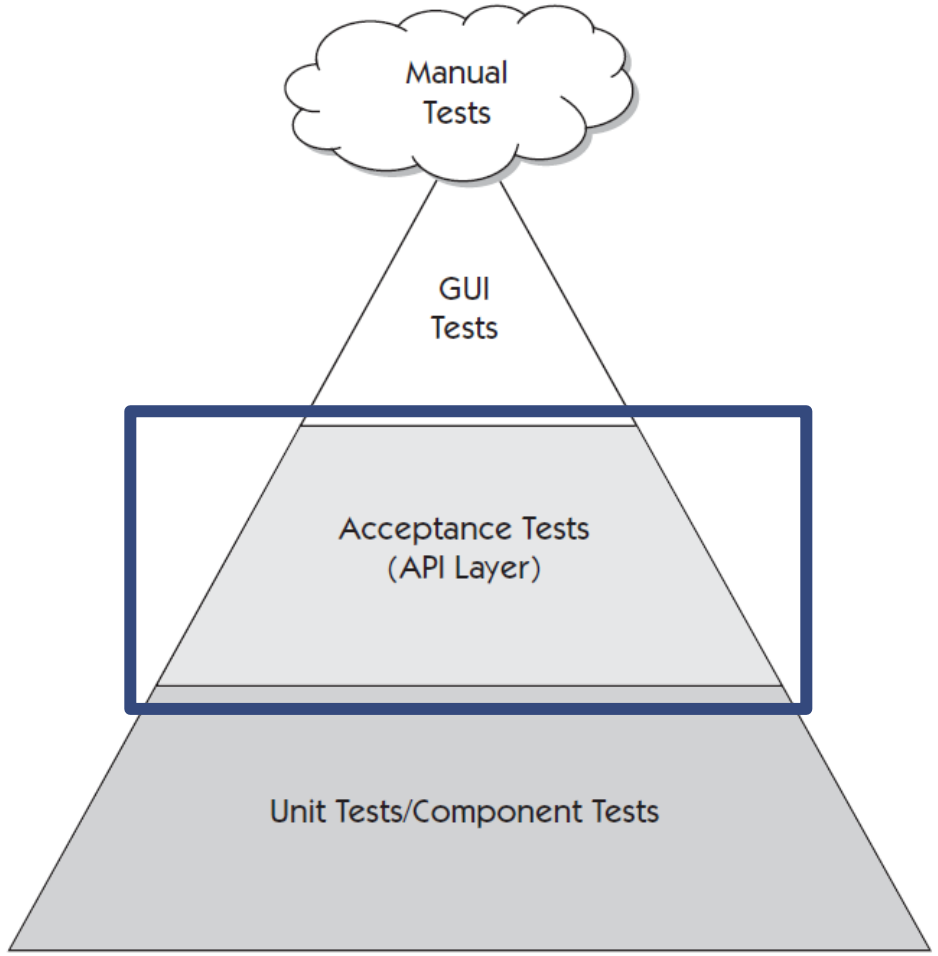
# Test Automation Pyramid

- The lowest tier is the foundation of the tests, mainly made up of robust unit tests and component tests.
- These tests are generally written in the same language as the system under test



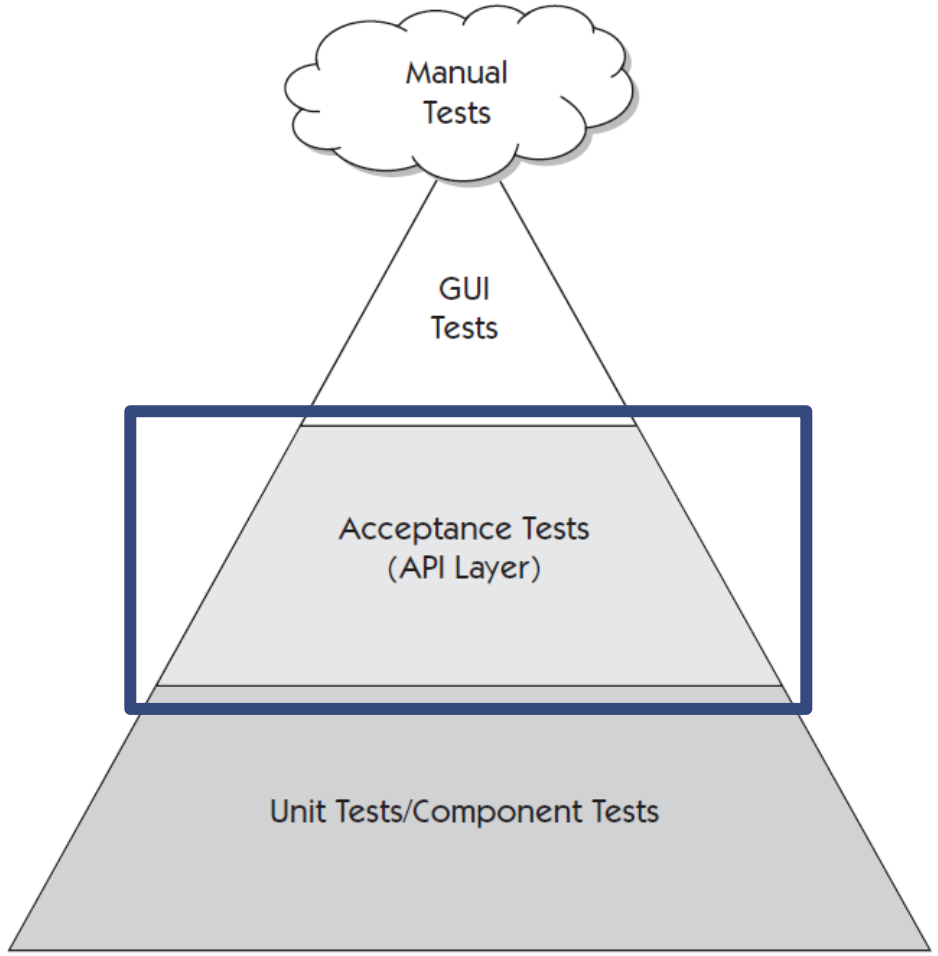
# Test Automation Pyramid

- The middle tier includes most of the automated business-facing tests written to support the team.
- These are the functional tests that verify that we are "building the right thing".
- The tests in this layer may include "story" tests and "acceptance" tests.



# Test Automation Pyramid

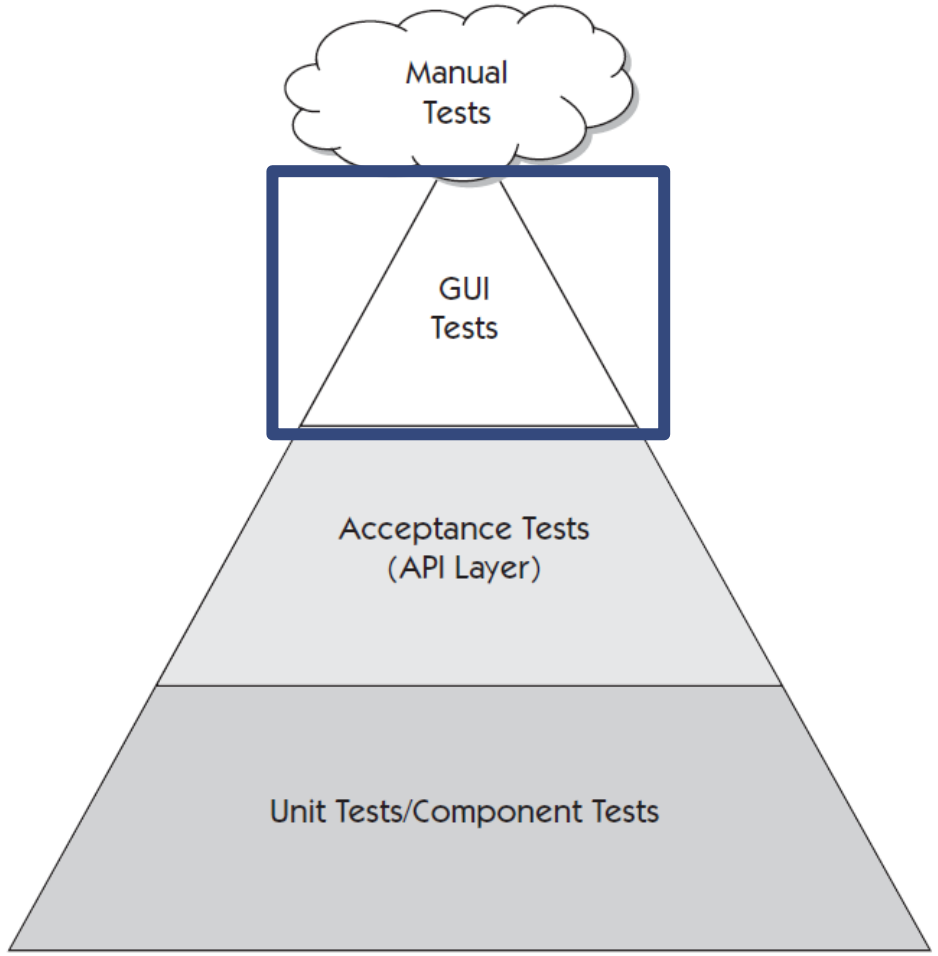
- These tests operate at the API level or "behind the GUI", testing the functionality directly without going through the GUI.



# Test Automation Pyramid

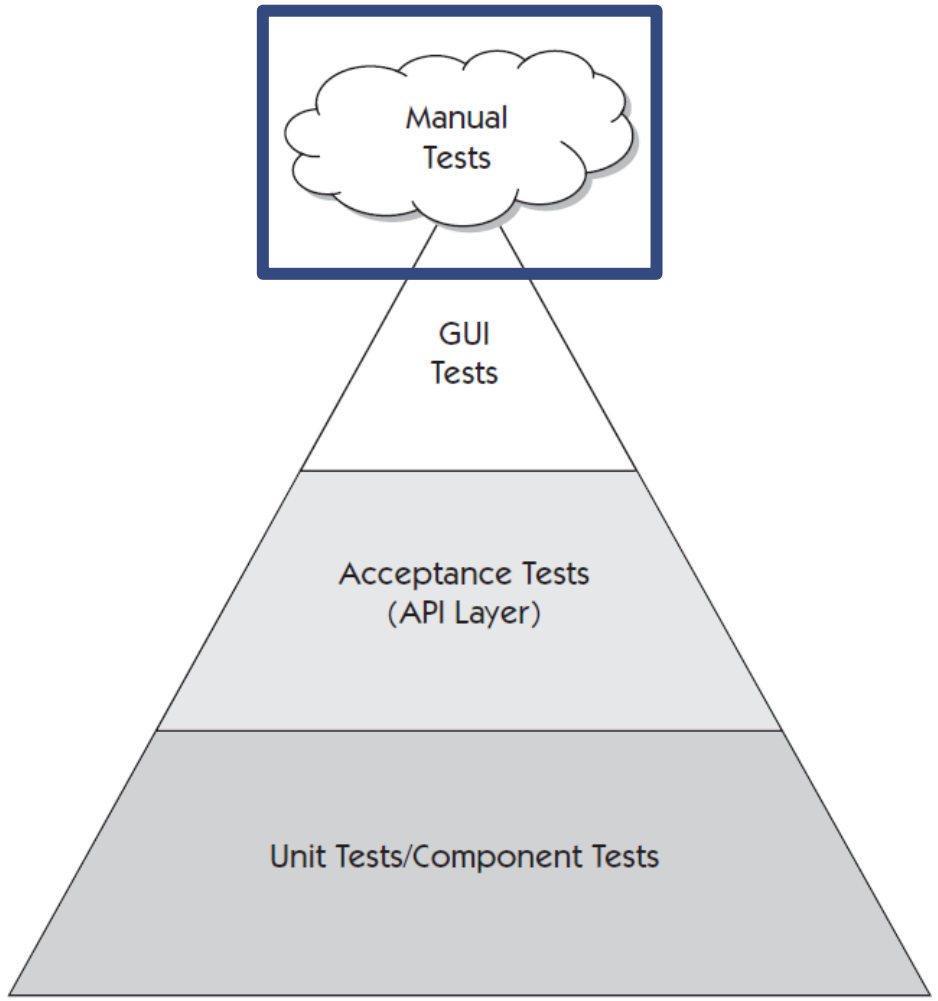
- The top tier represents what should be the smallest automation effort.
- These tests are the ones done through the GUI, the ones that actually operate and manipulate the presentation layer.

Because components of the user interface tend to be changed often, these tests are much more brittle. Ex: just renaming HTML elements could cause a test script to fail



# Test Automation Pyramid

- No matter how many automated tests they have, most systems also need manual testing activities, such as exploratory testing and user acceptance testing.
- The bulk of the testing must be automated, or our manual testing won't give a good return on investment.



# What Can We Automate?

- Any tedious or repetitive task involved in developing software is a candidate for automation.
- Unit tests and component tests.
- API or Web Services tests.
- Testing behind the GUI.
- Testing the GUI (Tool selection is key for successful GUI automation). An important note here is a stable GUI objects naming convention.
- Load Tests: Like generate a high-volume attack to verify whether a website can be hacked or can handle a large load without some tool framework.

# What Shouldn't We Automate?

- Usability Testing: Observing users in action, debriefing them on their experiences, and judging the results, as well as logging user actions.
- Tests that Will Never Fail: As an example, making sure that the second address field in an HTML form is not required; Once the verification is done manually once the chance of someone changing it to be required are very slim.
- One-Off Tests




Appium: an Open Source tool for automating mobile applications



Specflow: is a testing framework supporting BDD practices in .NET framework.




Selenium: an open-source automated testing framework for web applications



Cucumber: A tool for testing Behavior-Driven Development (BDD)



Postman: a tool for testing calls to APIs