Foundations of Software Testing

2.1 Why Testing Matters

Learning objectives

- Why testing is essential in software development
- Common types of testing
- The testing pyramid
- Manual vs automated testing
- Benefits of early and continuous testing

What Is Software Testing?

- Process of verifying that software works as intended
- Detects defects before users find them
- Confirms that new features don’t break old ones
- Ensures reliability and user satisfaction
- A core part of professional development practice

Why Testing Matters

- Software is complex — mistakes are inevitable
- Tests give confidence in code quality
- Reduces bugs, regressions, and production failures
- Saves time and cost in the long run
- Makes refactoring and upgrades safer

Example: The Cost of a Bug

Phase Discovered	Relative Cost to Fix	Example
During development	1x	Unit test fails
During QA testing	10x	Manual tester finds issue
After release	100x+	Customer reports production bug

- Key Point: The later you find a bug, the more expensive it is to fix.

Types of Software Testing

Type	Purpose	Example
Unit Testing	Test small pieces of code (methods)	Test a calculator’s add() method
Integration Testing	Test how components work together	Test service + database
System Testing	Test the complete application	End-to-end workflow
Acceptance Testing	Verify business requirements	Does it meet user needs?

The Testing Pyramid

- (Visual suggested: triangle with 3 levels – Unit, Integration, UI/E2E)
- Foundation = Unit tests (many, fast, low-level)
- Middle = Integration tests (fewer, broader)
- Top = UI / End-to-End tests (few, slow)
- The higher you go, the slower and more expensive the tests become
- Balanced testing gives confidence at all levels

Manual vs Automated Testing

Aspect	Manual	Automated
Execution	Human performs steps	Code performs steps
Speed	Slow	Fast
Repeatability	Inconsistent	Consistent
Cost (long-term)	High	Lower
Best for	Exploratory & UI tests	Repeated checks & regressions

Benefits of Automated Testing

- Run tests automatically after every change
- Quick feedback for developers
- Easier to maintain quality over time
- Enables Continuous Integration (CI)
- Makes codebases more stable and maintainable

Continuous Testing Mindset

- Testing is not a one-time activity
- Should happen throughout development
- Small, frequent tests prevent big surprises
- Testing is part of professional software craftsmanship

Demo: How Bugs Slip Through Without Tests

- Without a test, this bug only appears at runtime.
- With a unit test, it would be caught early.
- Small bugs can break systems.
- Tests act as safety nets.
- You’ll learn to write these tests soon using JUnit.

Lab: Reflection

- Goal: Reflect on testing's importance through real-world thinking.
- Instructions (Individual Reflection):
  - 1. Think of a time where a bug caused a problem.
  - 2. Write a few bullet points describing:
    - What the bug was
    - How it was discovered
    - What impact it had
    - How it could have been prevented with testing

Key Takeaways

- Testing ensures software behaves correctly
- Catches bugs before release
- Saves time, cost, and reputation
- Automated testing = fast, repeatable, reliable
- Forms the foundation for modern software practices
- You'll soon learn how to test APIs automatically with tools like JUnit and Rest Assured