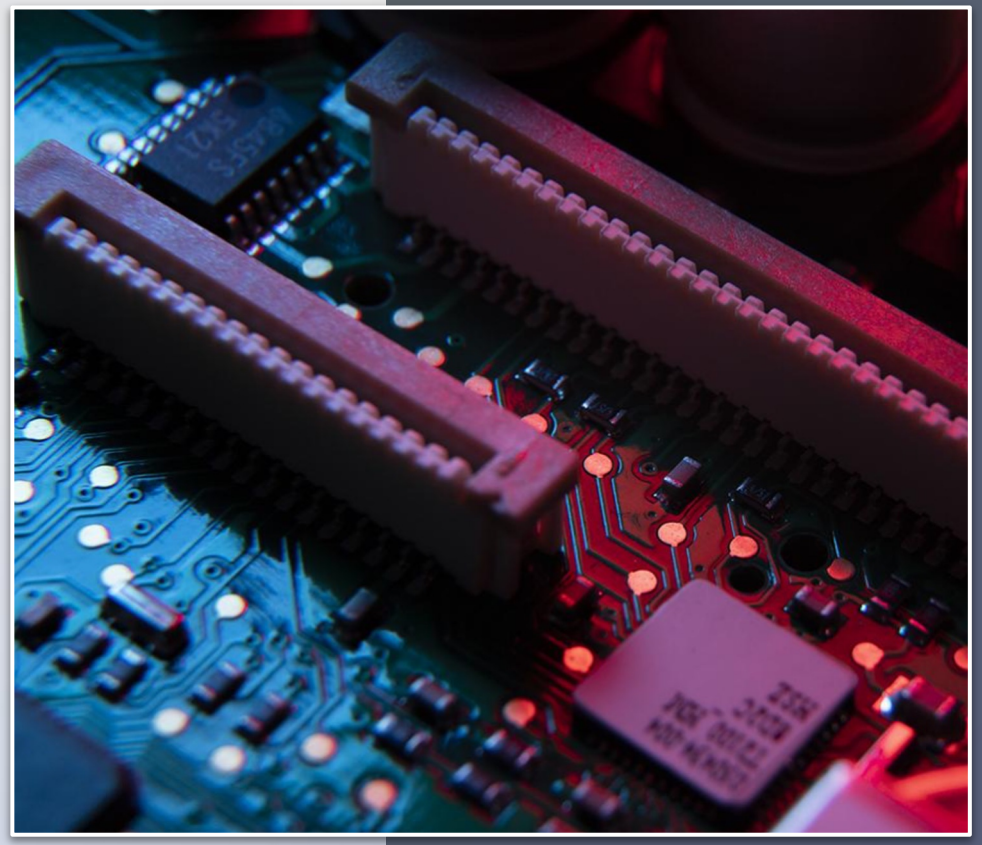
Extended Software Testing



Software Engineering

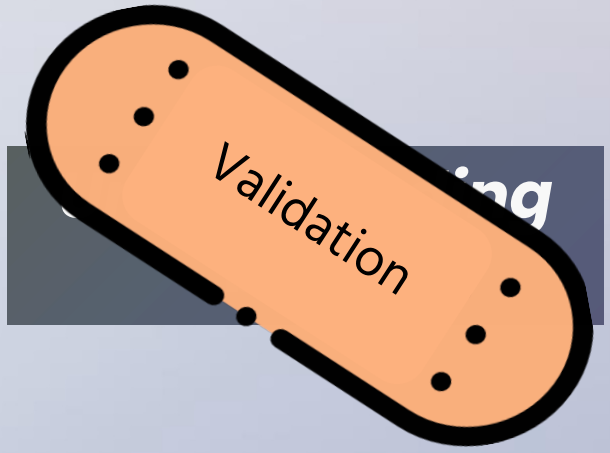
2023 - 2024

Introduction to Software Testing

- Verification & Validation
- Software Testing Types
- Software Testing Processes
- Unit Testing
- Performance Testing
- Record & Playback
- Designing a Testing Process
- Software Engineer & Testing

What is Software Testing?

- Software Testing is a **process** in which you **execute** your software with **simulated** data.
- If the test passes, the program is correct for the given data.
  - If the data is representative for a larger class of data, you can infer that the program is correct for any member of the data set.
  - If the test fails, we have **a bug**
- Two types of bugs:



Verification & Validation

"Testing can only show the presence of errors, not their absence†"

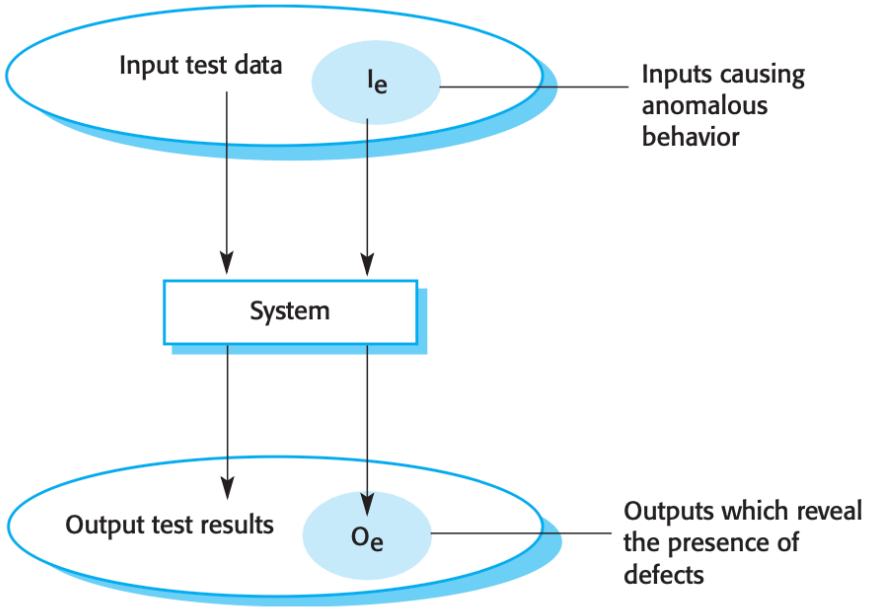
- Verification – Testing that the program runs correctly
  - 2 + 2 = 4 – True
- Validation – Testing that the program does what it is supposed to do
  - You require a list of the top 10 IMDb movies, but you get a queue of the top 10 IMDb movies

†Dijkstra, E. W. 1972. "The Humble Programmer." Comm. ACM 15 (10): 859–66. doi:10.1145/355604.361591

Verification & Validation

"Testing can only show the presence of errors, not their absence†"

- Verification – Testing that the program runs correctly
  - 2 + 2 = 4 – True
- Validation – Testing that the program does what it is supposed to do
  - You require a list of the top 10 IMDb movies, but you get a queue of the top 10 IMDb movies

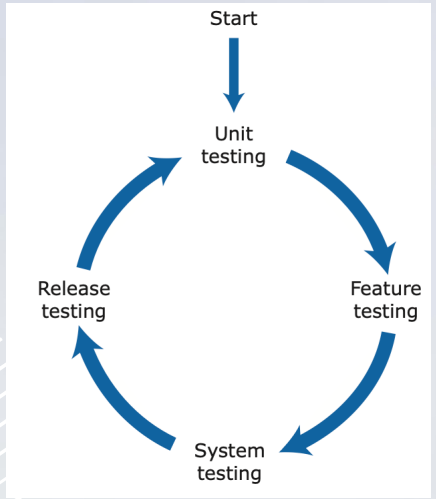


- **Validation:** *Are we building the right product?*
- **Verification:** *Are we building the product right?*

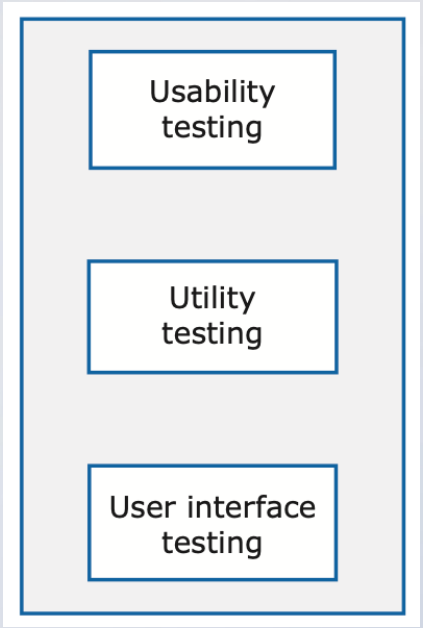
†Dijkstra, E. W. 1972. "The Humble Programmer." Comm. ACM 15 (10): 859–66. doi:10.1145/355604.361591

Testing Types

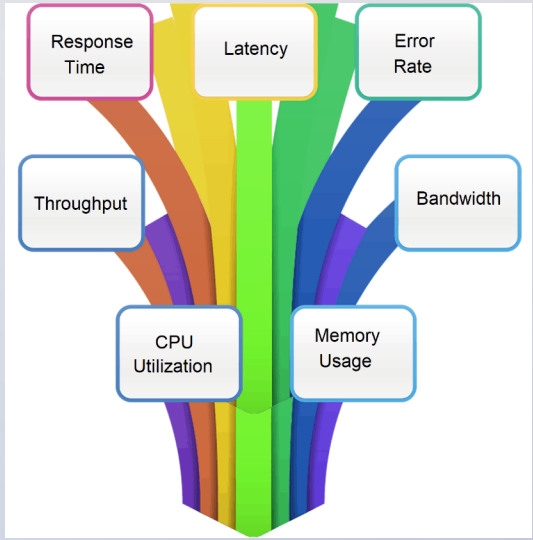
Functional testing



User testing



Performance testing



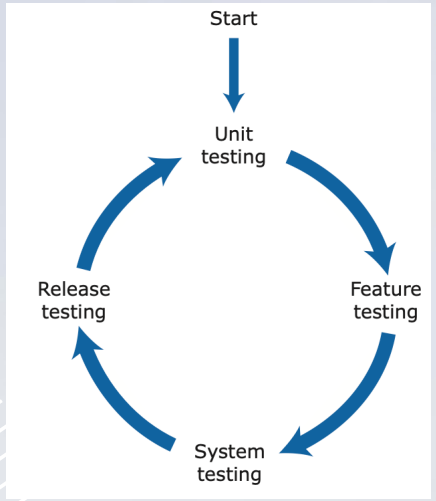
Security testing

Identify attacks

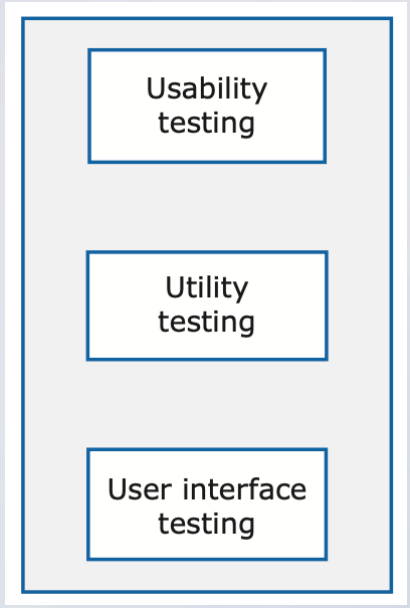
Implement safeguards

Testing Types

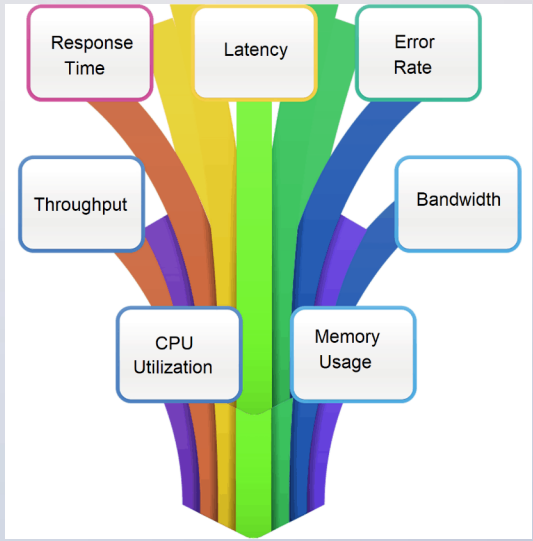
Functional testing



User testing



Performance testing



Security testing

Identify attacks

Implement safeguards

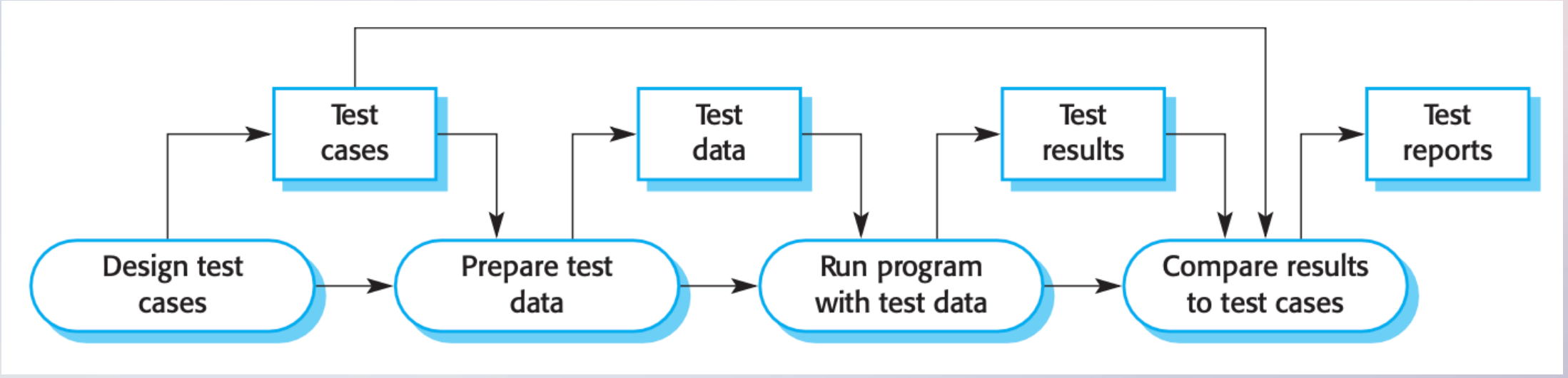
V & V

Validation

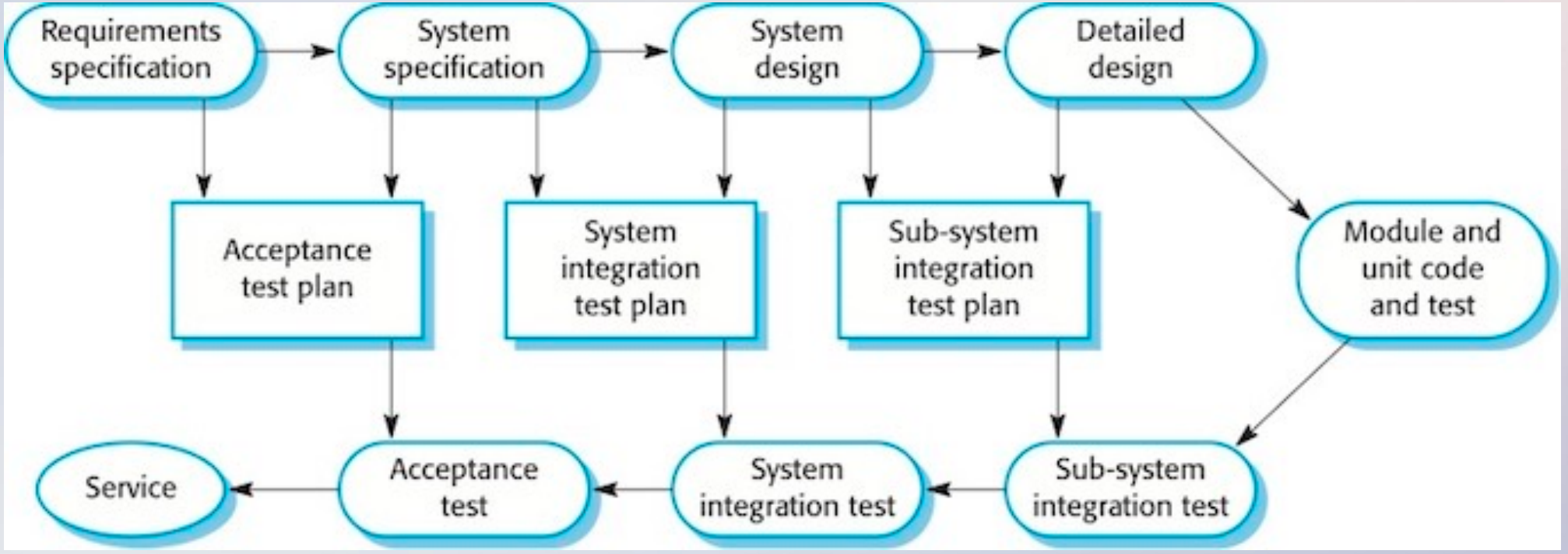
Verification

V & V

Basic Software Testing Process



V-model Testing Process



Test Driven Development

Unit Testing

- Defining and testing the smallest unit of your software.
  - A unit is small enough to be a unit when you can’t reduce the complexity of the test any further.
- Stop over-using it ☺
- When to use it?
  - When you are able to define a unit (equivalent sizes)
  - When the unit si complex or critical.
- When not to use it?
  - When the ”unit” depends on many other services (for example if you have to mock multiple dependencies – DB connection, DTO TranslationService, etc.)
  - When test assertion depends on too many variables (but then again, probably your unit could be broken down) – see next slides

Unit Testing (Bad Example)

Depends on a complex(?) Service function

```
[HttpGet]
Rares Cristea
public async Task<ActionResult<List<Teacher>>> GetAllTeachers()
{
    var result :List<TeacherSendDTO>? = await _teacherService.GetAllTeachers();
    if (result is null)
    {
        return NotFound("Didn't find the list of teachers.");
    }
    return Ok(result);
}
```

Not a lot of actual functionality

Heavily dependent on Database Scenario

Unit Testing (Better Example)

```
public class Department : BaseModel
{
    5 usages
    public long Id { get; set; }

    4 usages
    public String FullName { get; set; }

    5 usages
    public long FacultyId { get; set; }

    1 usage
    public Faculty Faculty { get; set; }

    2 usages
    public ICollection<TeacherDepartmentMembership> TeacherDepartmentMemberships { get; set; }

    1 usage
    public ICollection<Teacher> Teachers { get; set; }
}
```

It's good to make a Unit Test for this function, because:

- Single, well defined purpose
- Can easily define homogenous test cases (departments, and corresponding DTOs)

```
public class DepartmentSendDTO
{
    2 usages
    public long Id { get; set; }

    2 usages
    public String FullName { get; set; }

    2 usages
    public FacultySendDTO Faculty { get; set; }

}
```

```
public DepartmentSendDTO DepartmentToDepartmentSendDto(Department department)
{
    return new DepartmentSendDTO()
    {
        Id = department.Id,
        FullName = department.FullName,
        Faculty = _facultyService.GetSingleFaculty(department.FacultyId).Result
    };
}
```

Still not ideal due to this dependency

Unit test breakdown

```
class Program
    {
        static void Main(string[] args)
        {
            using (SqlConnection connection = new SqlConnection(
                "Server=[SQ_SIXTEEN];Database=[PocketCentral];User ID=[un];Password=[pw];Trusted_Connection=true"))
            {
                connection.Open();
                using (SqlCommand command = new SqlCommand("SELECT * FROM tbl_Terminal", connection))
                {
                    using (SqlDataReader reader = command.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            for (int i = 0; i < reader.FieldCount; i++)
                            {
                                Console.WriteLine(reader.GetValue(i));
                            }
                            Console.WriteLine();
                        }
                    }
                }
            }
        }
    }
```

- Let’s assume the code above.
- It opens a **DB connection**, **reads** some data, and **displays** it in the console.

Unit test breakdown

What can go wrong?

Datbase may not connect

SQL malformed

Parsing a non-iterable object

Trying to write an object without writing overload

```
class Program
    {
        static void Main(string[] args)
        {
            using (SqlConnection connection = new SqlConnection(
                "Server=[SQ_SIXTEEN];Database=[PocketCentral];User ID=[un];Password=[pw];Trusted_Connection=true"))
            {
                connection.Open();
                using (SqlCommand command = new SqlCommand("SELECT * FROM tbl_Terminal", connection))
                {
                    {
                        for (int i = 0; i < reader.FieldCount; i++)
                        {
                            Console.WriteLine(reader.GetValue(i));
                        }
                        Console.WriteLine();
                    }
                }
            }
        }
    }
```

Are these versions of the same problem?

Unit test breakdown

+ Connecting to the database

+ Executing an SQL command

+ Reading, and parsing an object

+ Writing

4 Unit Tests
(each with multiple test values)

```
class Program
    {
        static void Main(string[] args)
        {
            using (SqlConnection connection = new SqlConnection(
                "Server=[SQ_SIXTEEN];Database=[PocketCentral];User ID=[un];Password=[pw];Trusted_Connection=true"))
            {
                connection.Open();
                using (SqlCommand command = new SqlCommand("SELECT * FROM tbl_Terminal", connection))
                {
                    using (SqlDataReader reader = command.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            for (int i = 0; i < reader.FieldCount; i++)
                            {
                                Console.WriteLine(reader.GetValue(i));
                            }
                            Console.WriteLine();
                        }
                    }
                }
            }
        }
    }
```

Unit Testing

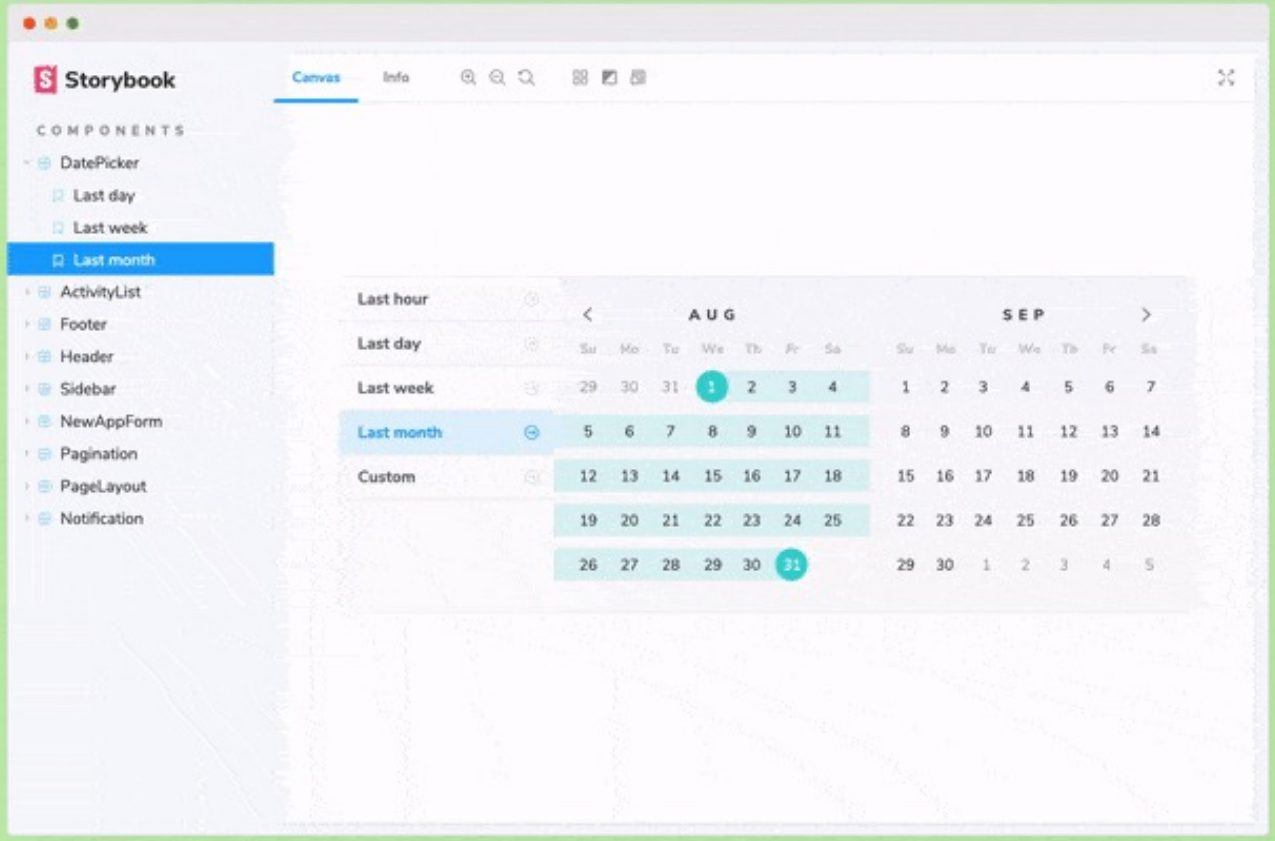
Join at menti.com | use code 2545 1554

Mentimeter

Waiting for players...

Start quiz

Visual "Unit Testing"

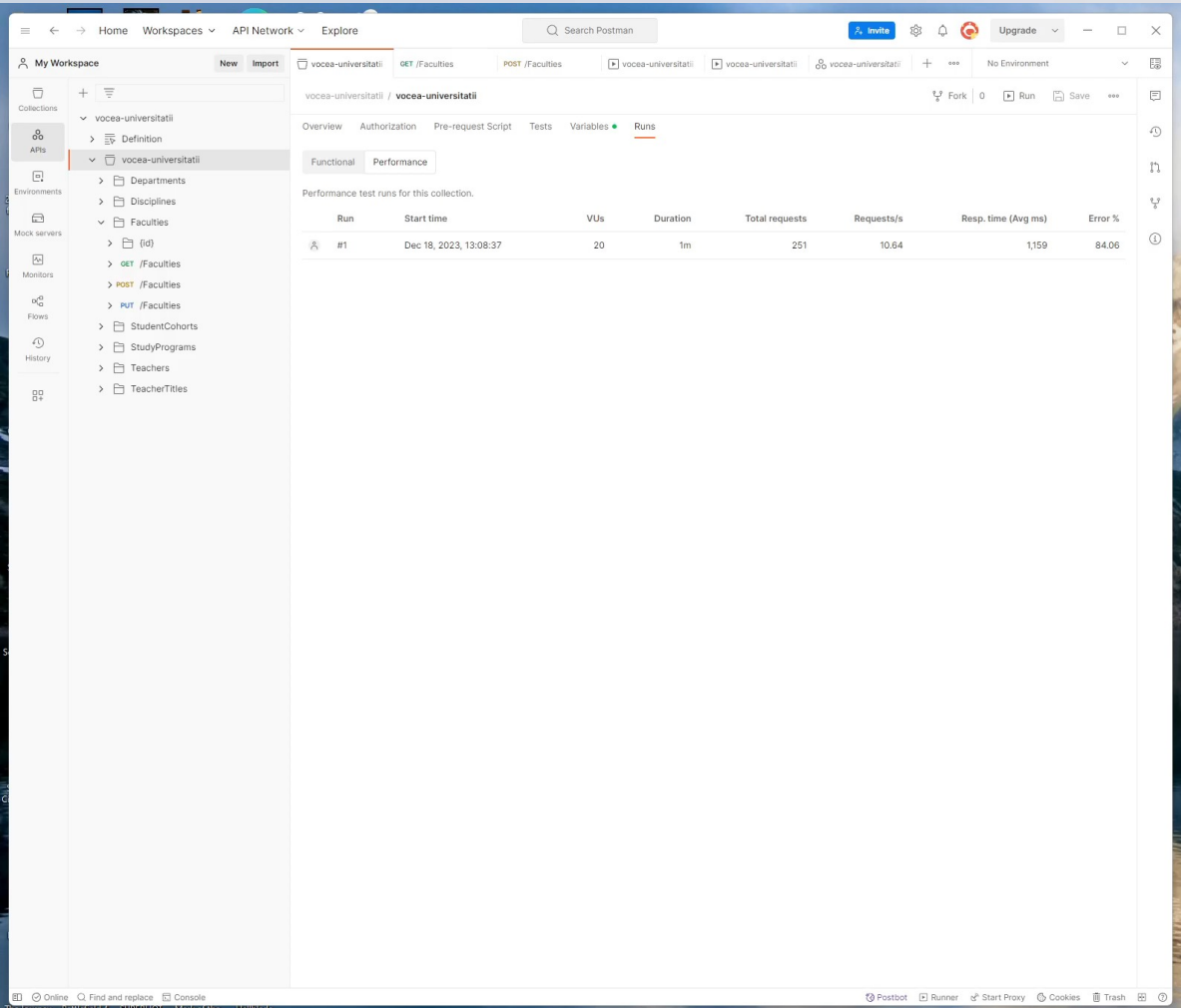


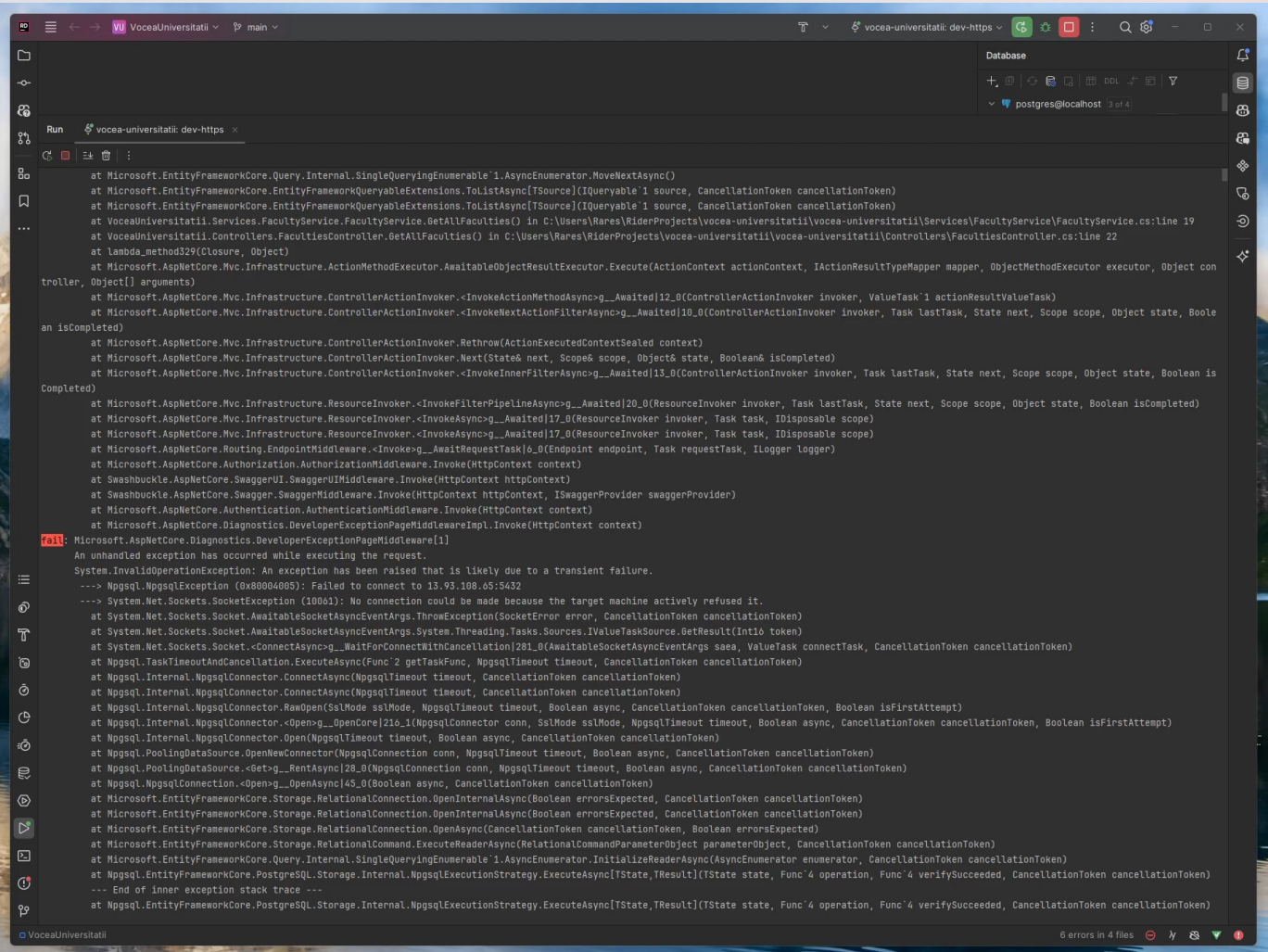
Visual test in a glance

Performance Testing

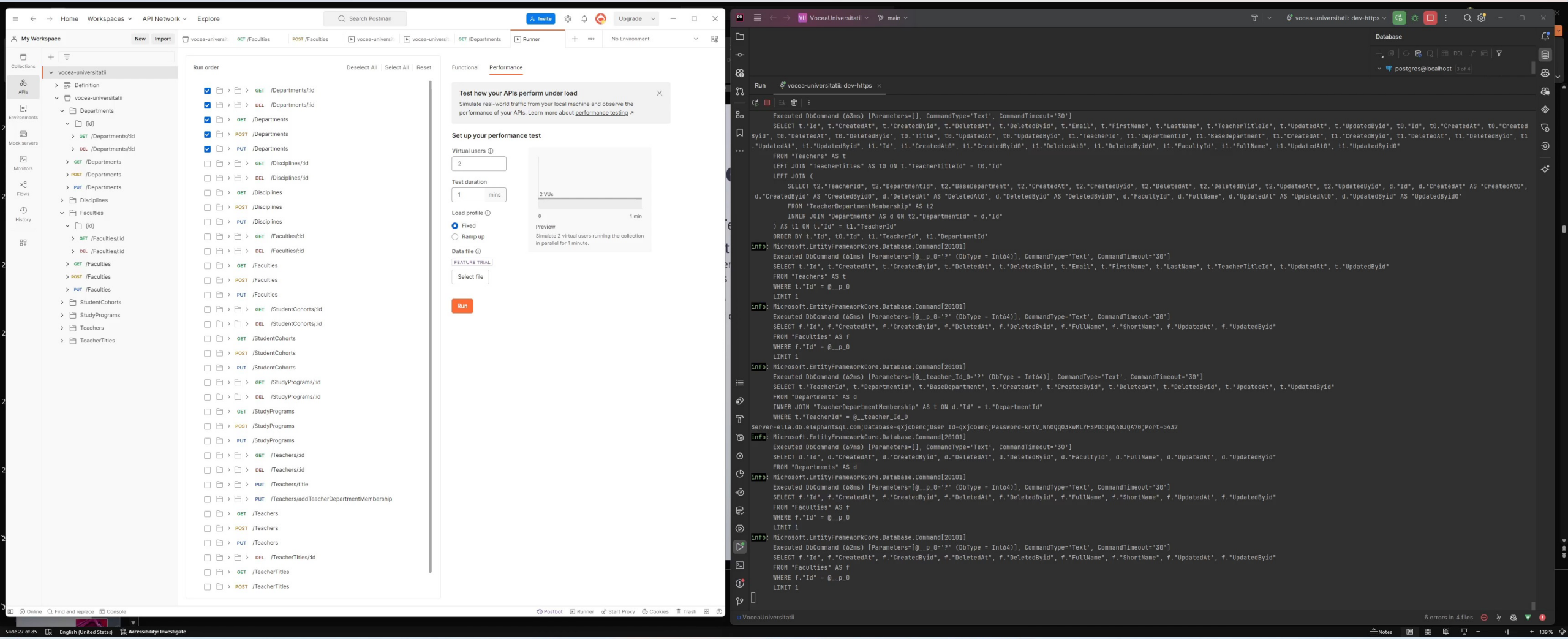
- Load test
- Stress Test
- Scalability Test
- Endurance Testing
- Great explanation: (1) Performance, Load, Stress or Endurance Test? Which do you want? | LinkedIn

Performance Testing





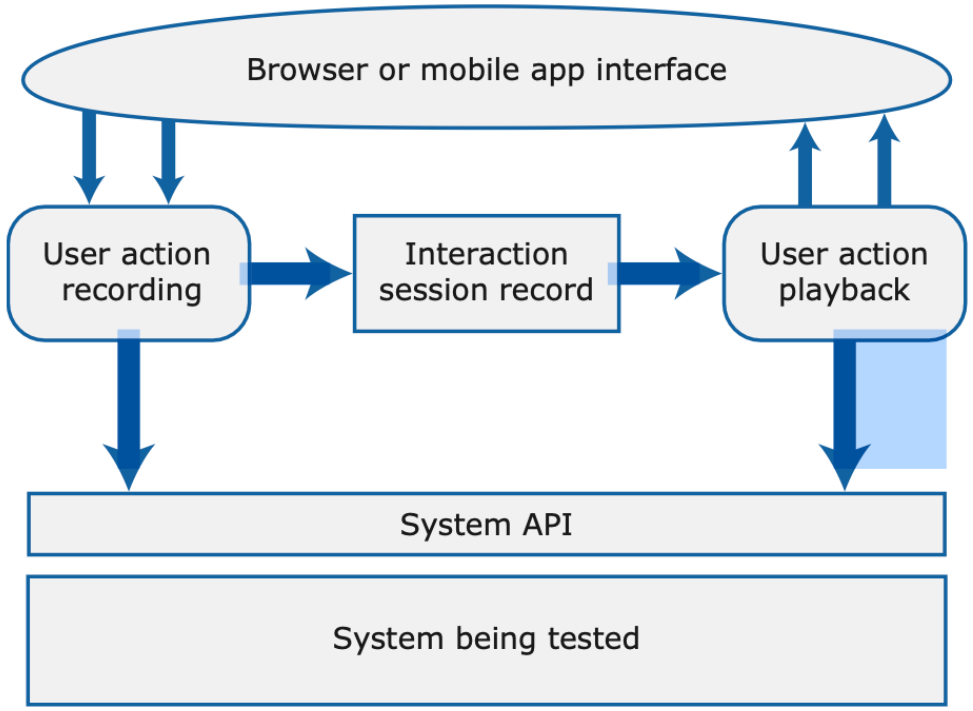
Performance Testing



Testing Techniques – Record and Playback

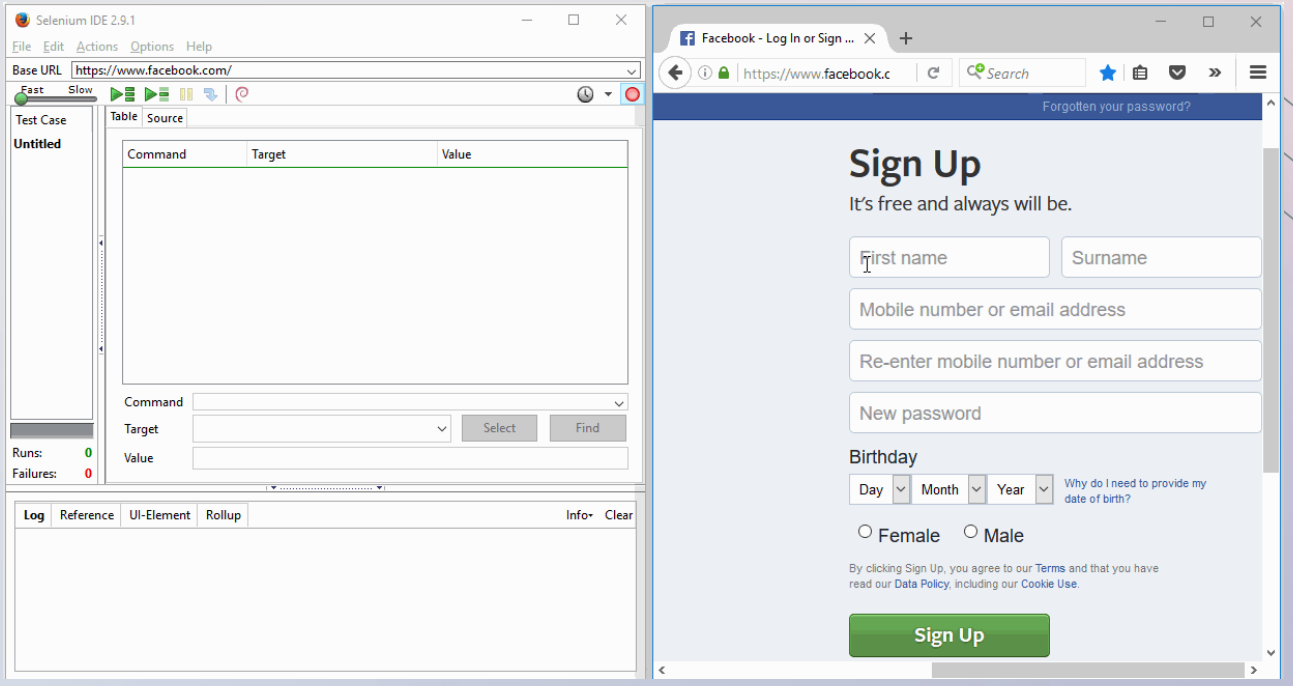
- Suitable for End-To-End Testing
- It tests an entire functionality across the system
- End of the Cycle Technique
- Example: New Account Registration process from front-end.
- Tool: Selenium IDE

Figure 9.7 Interaction recording and playback



Testing Techniques – Record and Playback

- Suitable for End-To-End Testing
- It tests an entire functionality across the system
- End of the Cycle Technique
- Example: New Account Registration process from front-end.
- Tool: Selenium IDE



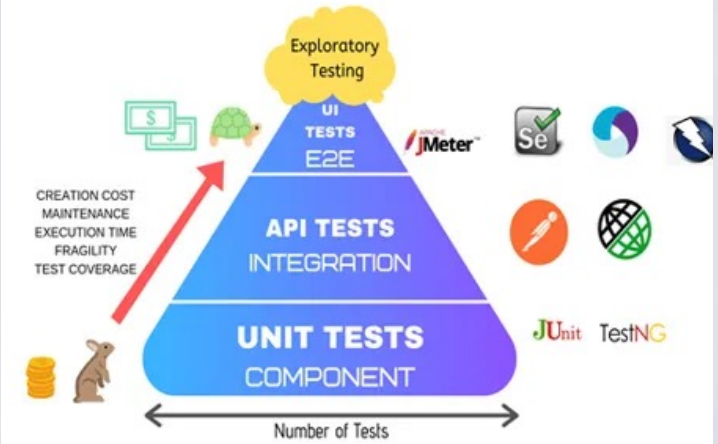
Matching testing types with levels

	Functional Testing	User Testing	Performance Testing	Security Testing
Unit Testing	Partition testing / Guideline based testing	Storybook / Visual Testing (A/B Testing)	Stress testing	Partition testing
Integration Testing	System Testing / Behaviour Driven Testing	Requirements-based testing (Acceptance testing)	Interface Testing	?
End-to-End testing	Release Testing / Behaviour Driven Testing	Manual Scenario Testing	Record and Replay	Third-party White-hat hacking

Designing a Testing Approach for your Software

Test Pyramid Approach

Aim is to Prevent Bugs



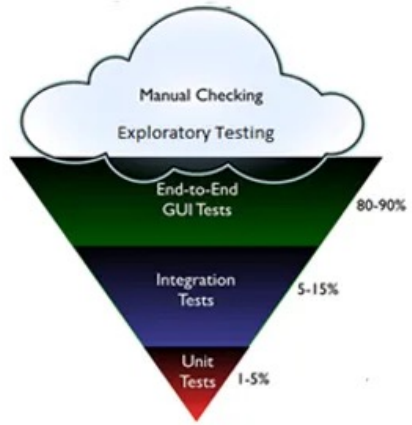
Tool to fix the problem of over-reliance on long-running UI tests.

Detects bugs earlier which reduce project cost and improve overall quality of the Application.

Test Automation (Unit Test, API Test & UI Test) Time is low.

Traditional (Ice-cream Cone) Approach

Aim is to Find Bugs



Number of end-to-end tests required to get a good coverage is much higher and more complex than the number of unit tests plus selected end-to-end tests required.

Invest in automation, but mostly building slow, complex, fragile end-to-end GUI test.

How Google Tests Software. Google was founded in September 1998 by… | by Dilusha Rasangi Kumarage | Medium

Software Engineer's Role in Testing

Software Engineers

Design Documentation → Design Data Structures → Design Architecture→

Write Functional Code →
Review Code →
Unit Test + Integration Test →

Software Engineers in Test

Review the Design→
Check Code Quality & Risks→
Refactor the Code to make it more testable→

Write Unit Testing Frameworks & Automation →
Increase Quality & Test Coverage by adding new features →
Work with Developers →

Test Engineers

Test with their deep Domain knowledge & Product expertise →

Create / Execute Automation Scripts →

Report Bugs →

Act as End User & Analyze Risks→

How Google Tests Software. Google was founded in September 1998 by… | by Dilusha Rasangi Kumarage | Medium

Software Testing Requirements

- Implement a Software Testing Process for your project.
- Use the most appropriate testing process
  - Explain why it was chosen (compare with other methodologies)
  - Describe its components
- Define tests on all three levels (end-to-end, integration, unit)
  - Alternatively, argue why on your project one level is not appropriate.
- The scenarios that you are going to present at the live demo, should be thoroughly tested.
  - Identify the most critical qualities of your software
  - Define test suites that are appropriate for those qualities

Your project must prove that you:

- Can differentiate between multiple testing types
- Know when to apply each
- Implement them, so that they are useful for the application.

Resources Used