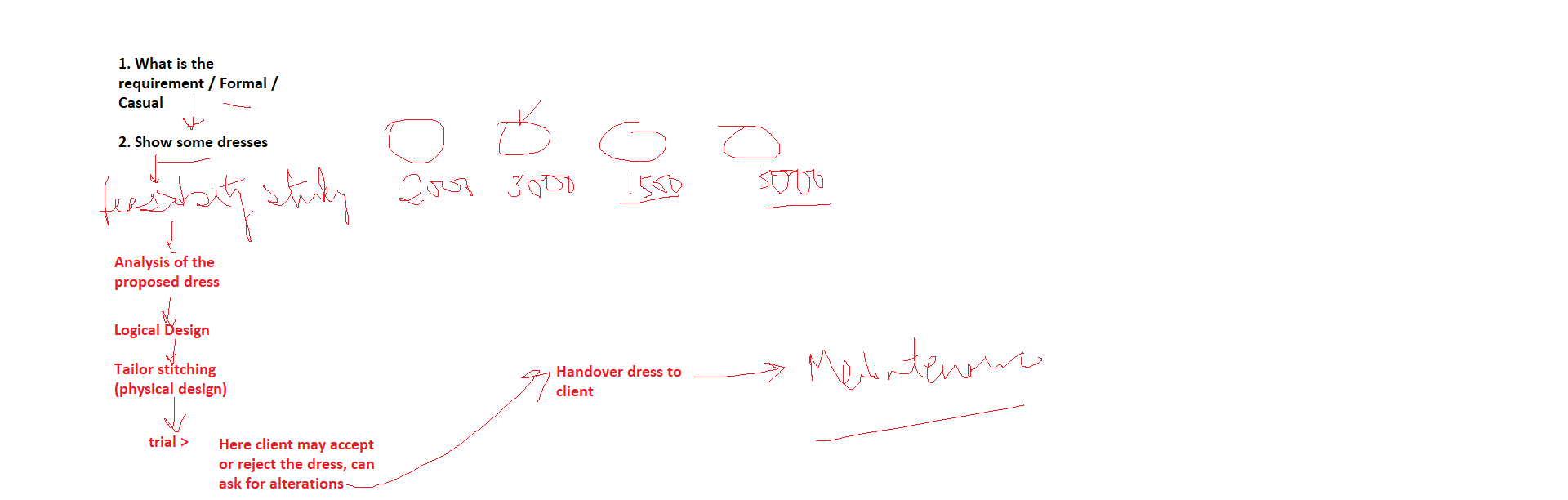
SDLC > System Development Life Cycle > Different stages through which a system goes during its life time

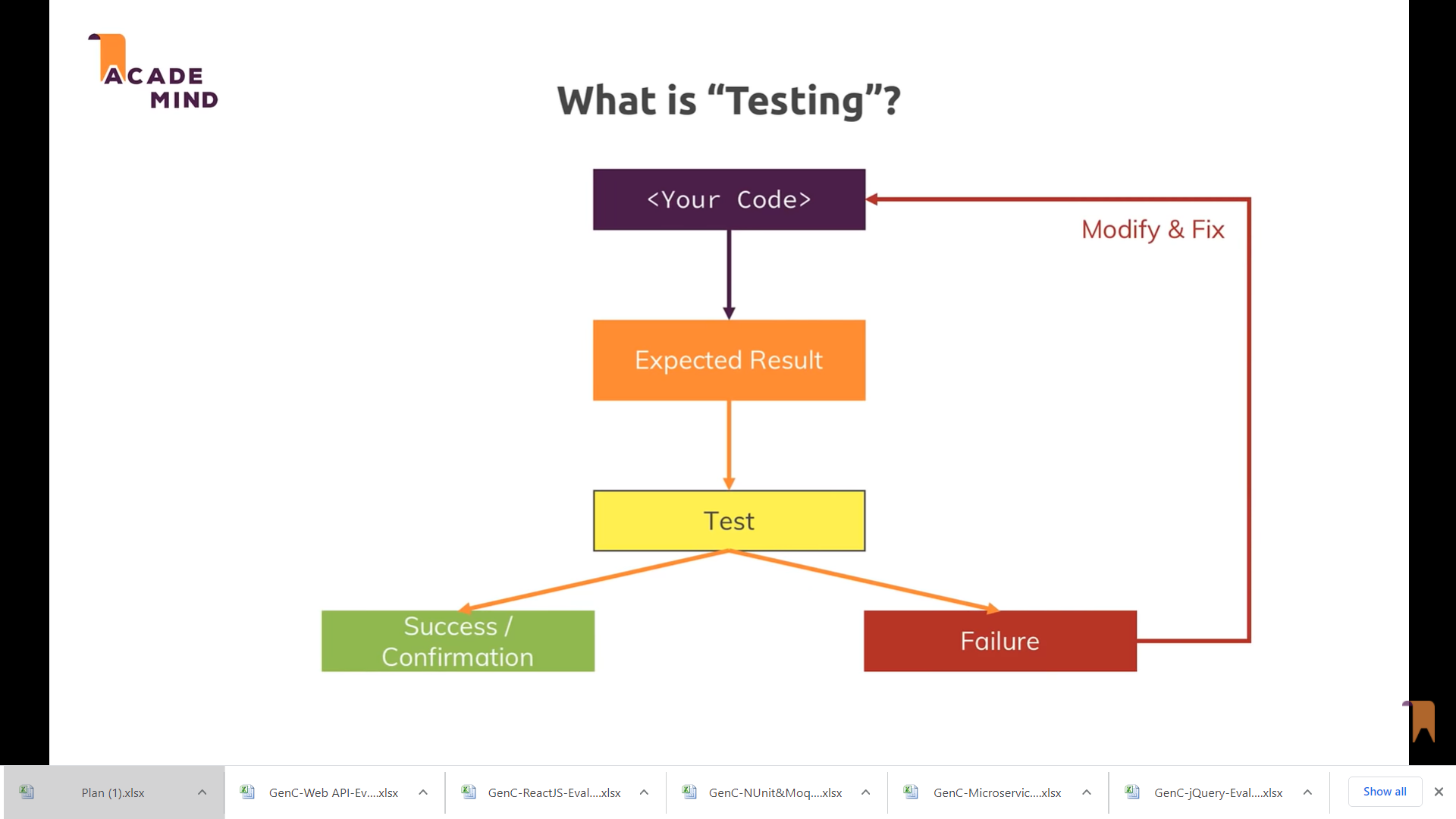
1. **Problem Identification** > What is the requirement, Analyse the current situation (The most critical phase) as we do not know the exact requirements (output)
2. Feasiblity Study > Selecting the best solution that is feasible for the client & end users (output)
3. System Analysis > Study & analyse the selected solution, Anaylsis of proposed system (output) > SRS (Software Requirement Specfication Document) , it’s a contract between dev team & the client
4. System Design > Design the system
   1. Logical Design(DFD , ER Diagram , FLowcharts) (output)
   2. Physical Design (coding , start making databases)
5. Testing(output)
6. Implementation (Give system to client, install on client side)
7. Post Implentataion Review
8. Maintenance (Correct Bugs, add new features) (The most longest phase)

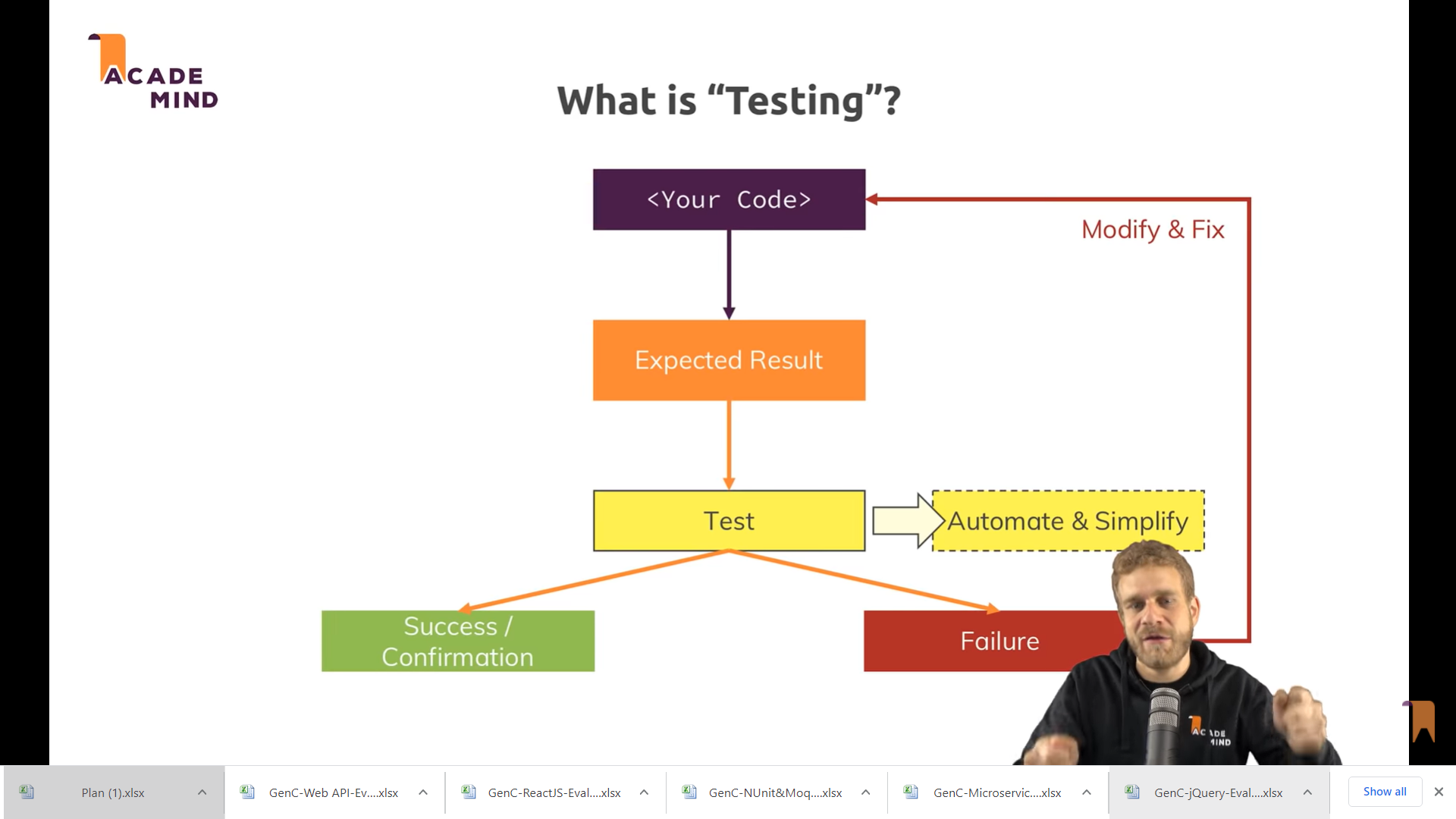


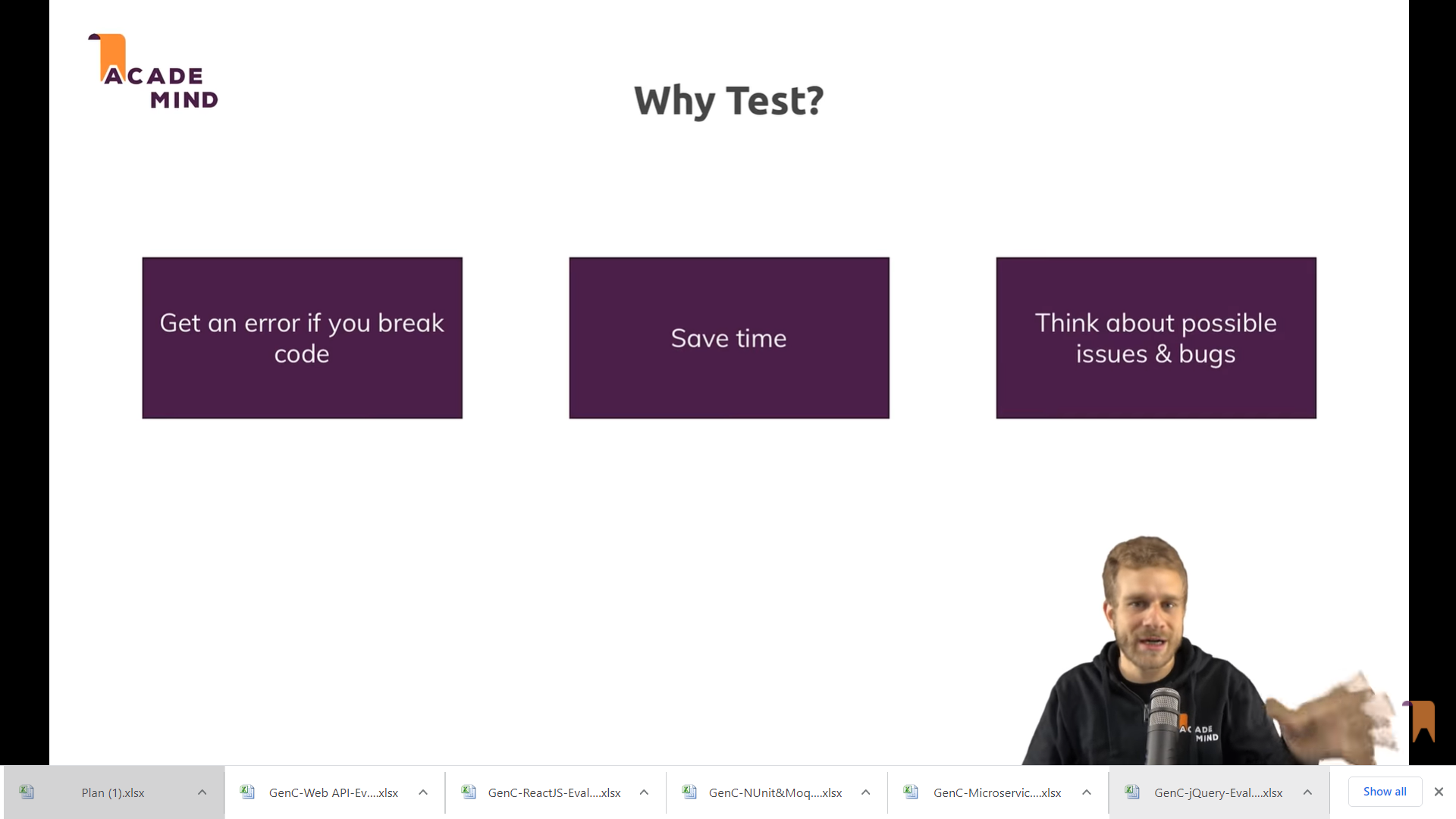
Testing means that what was mentioned are you getting that or not

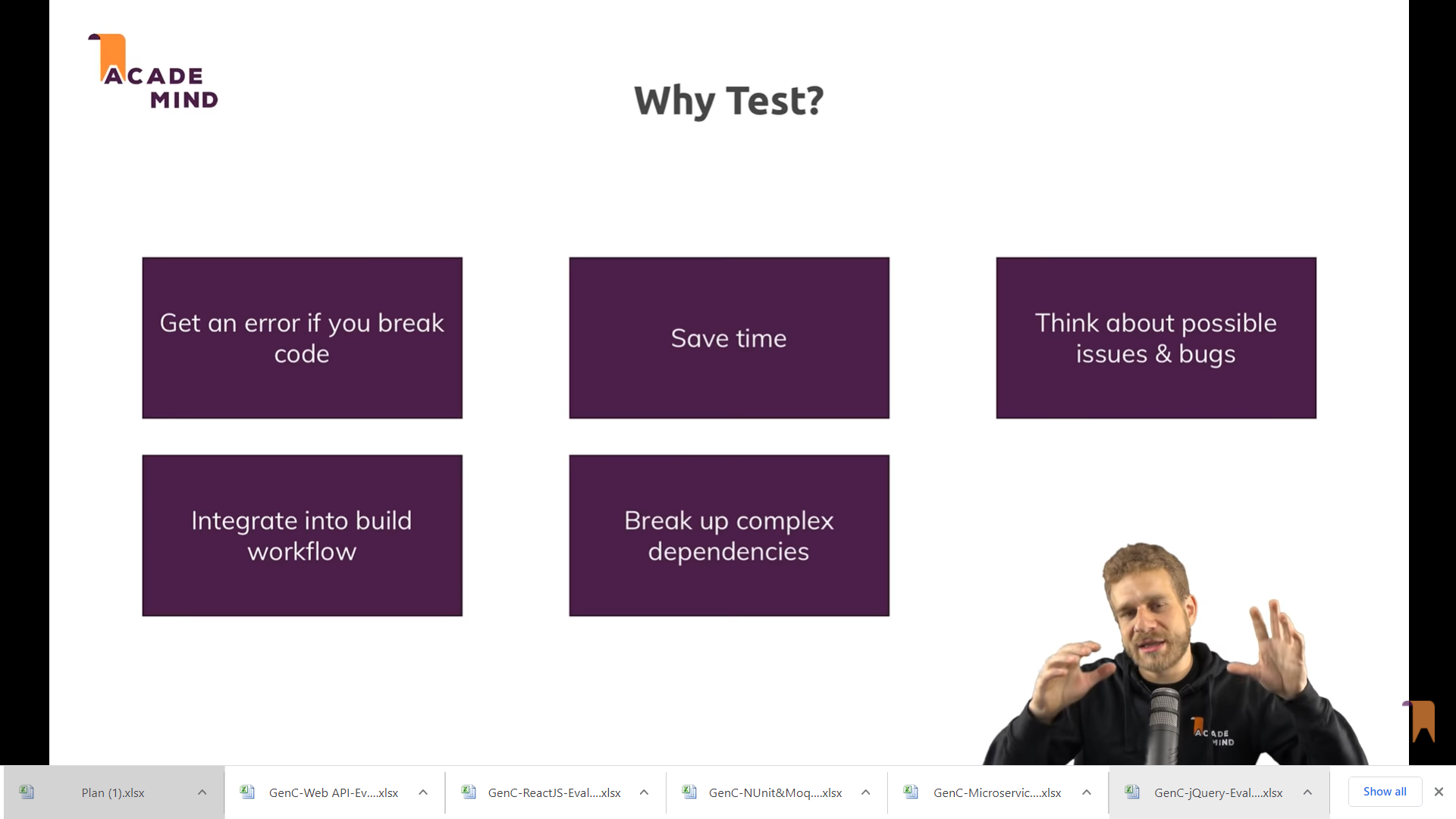
Testing can be done at dfferent levels

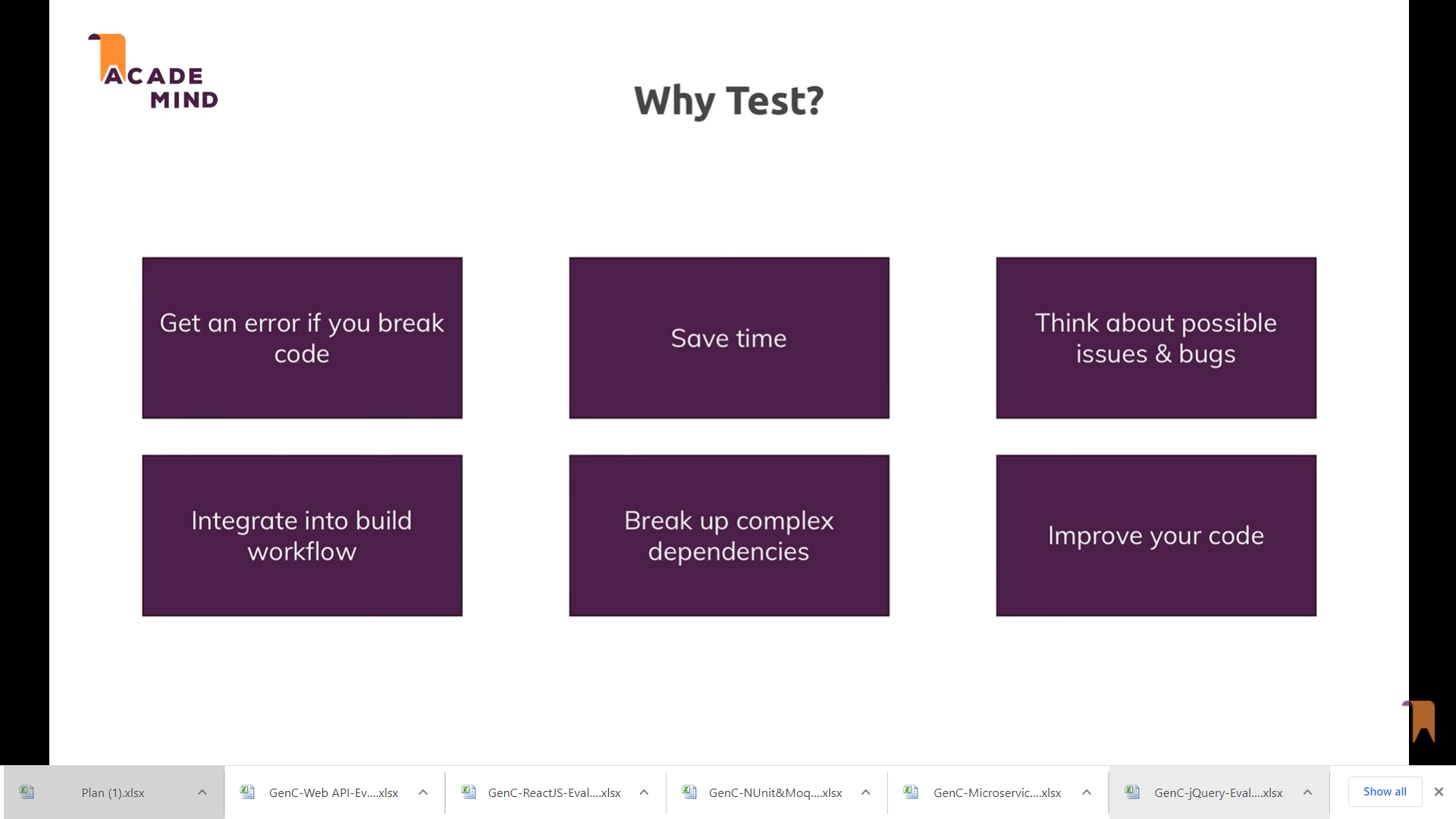
1. Unit Test (Testing a single function , developer)
2. Integration Test (combine some methods , when function1 calls function2 , are they able to work together) , developer
3. System Test > Here you run the complete system as a whole , develpoment team
4. UA Test > User acceptance testing , client
5. Black Box Test > Tester do not go into details of code , they focus on functionality
6. White Box Test > Tester knows about the coding, go into details of code , they focus on coding part
7. Load Test
8. Stress Test
9. Regression Test

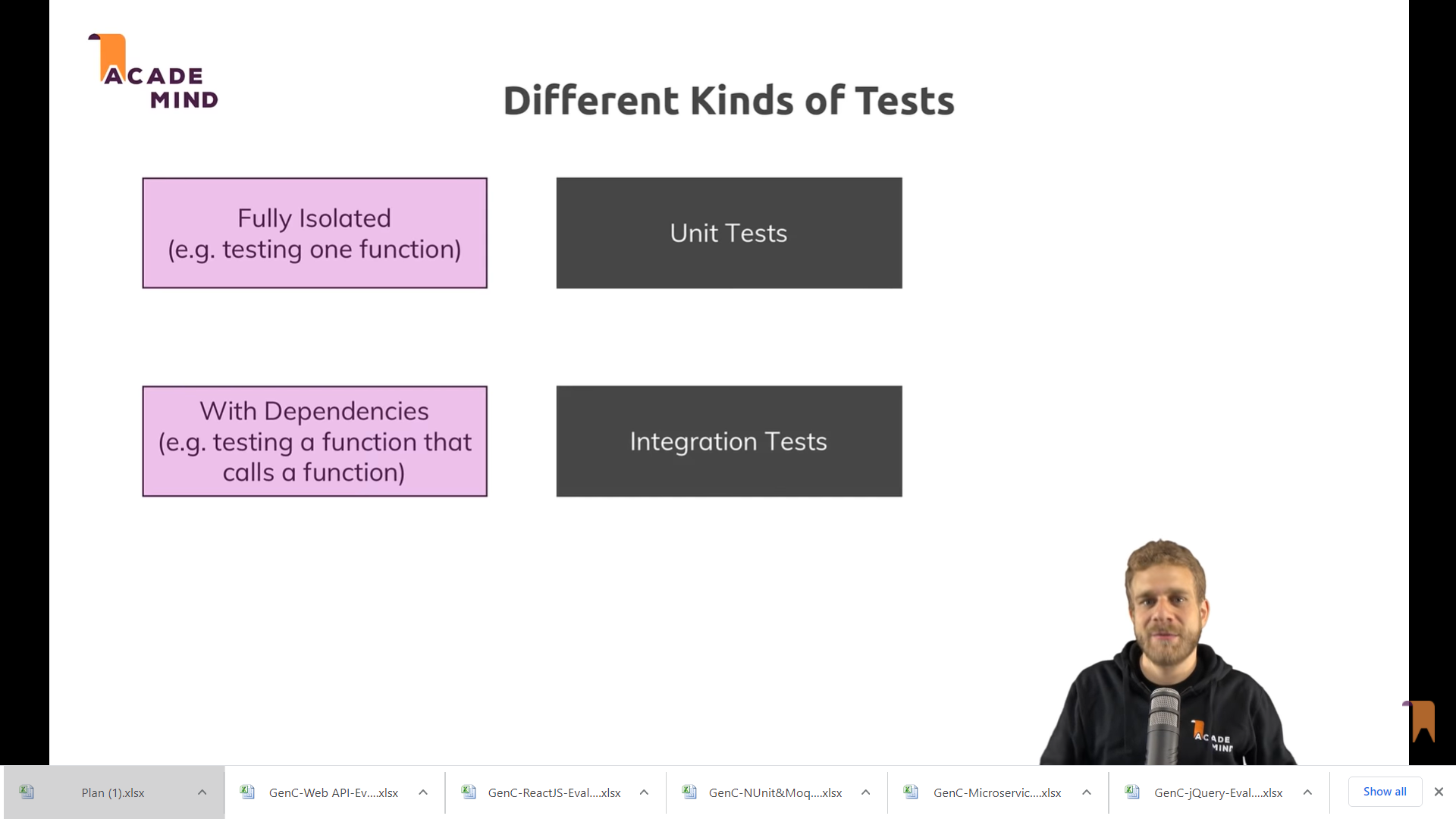


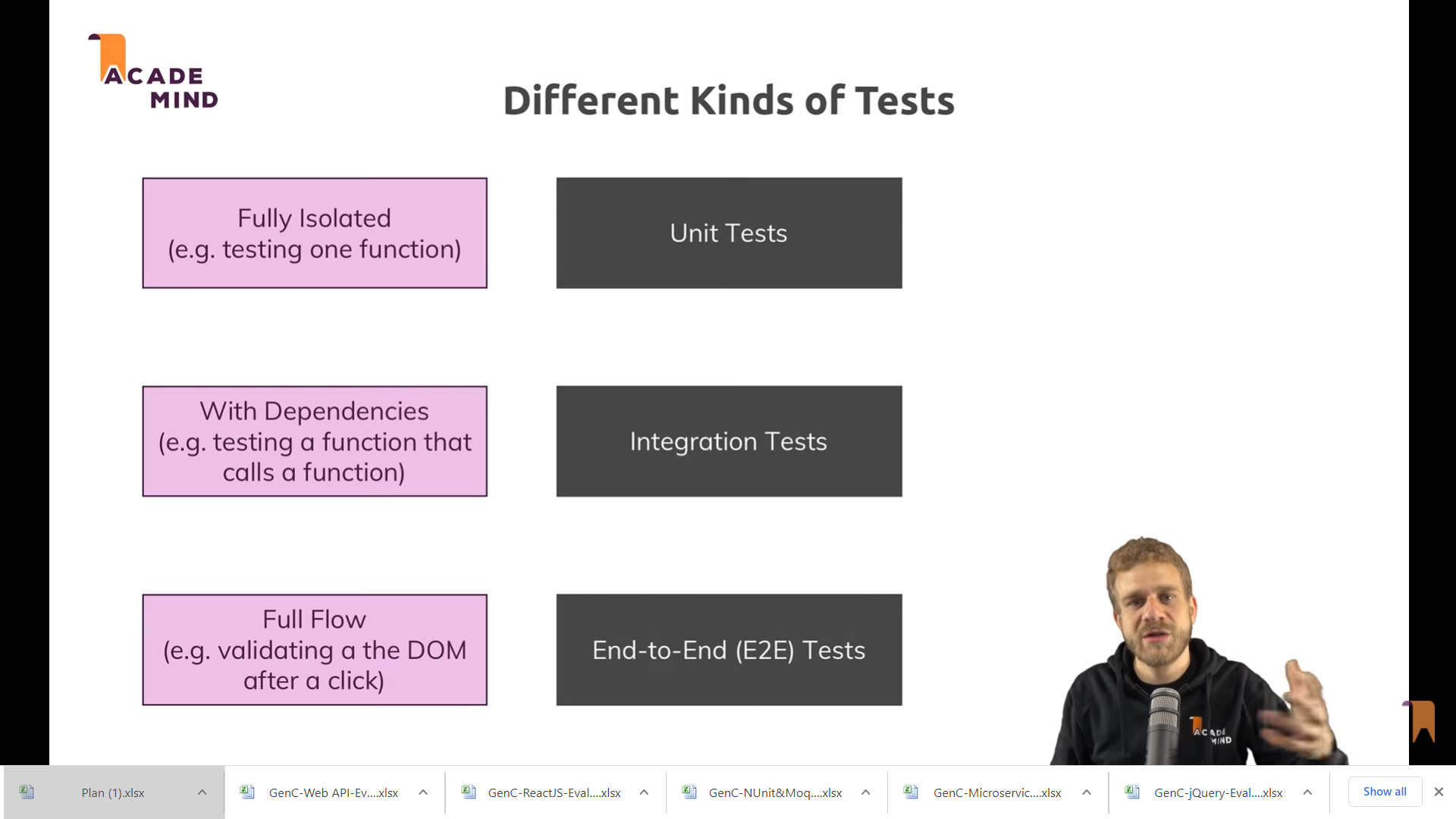


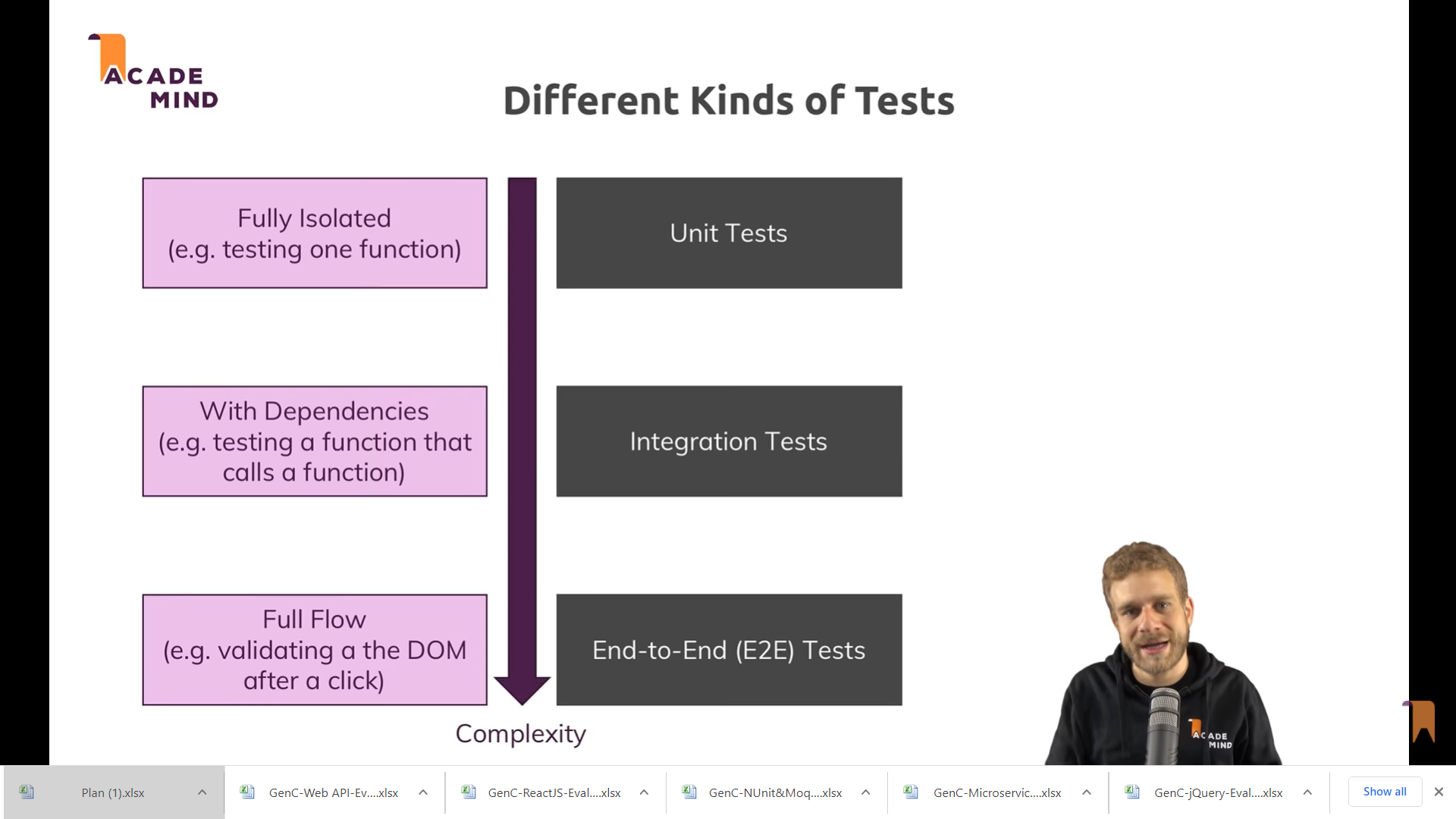


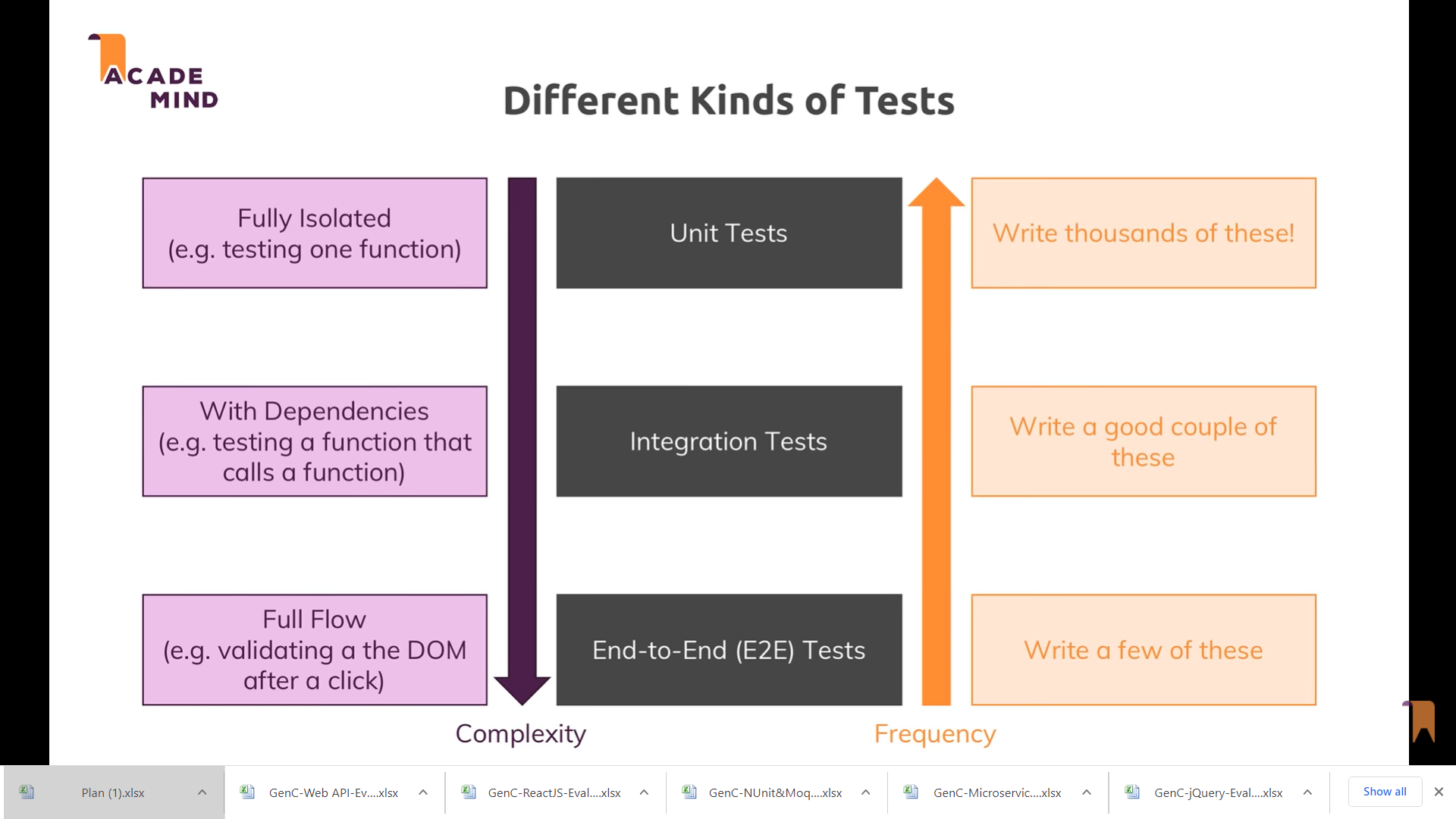


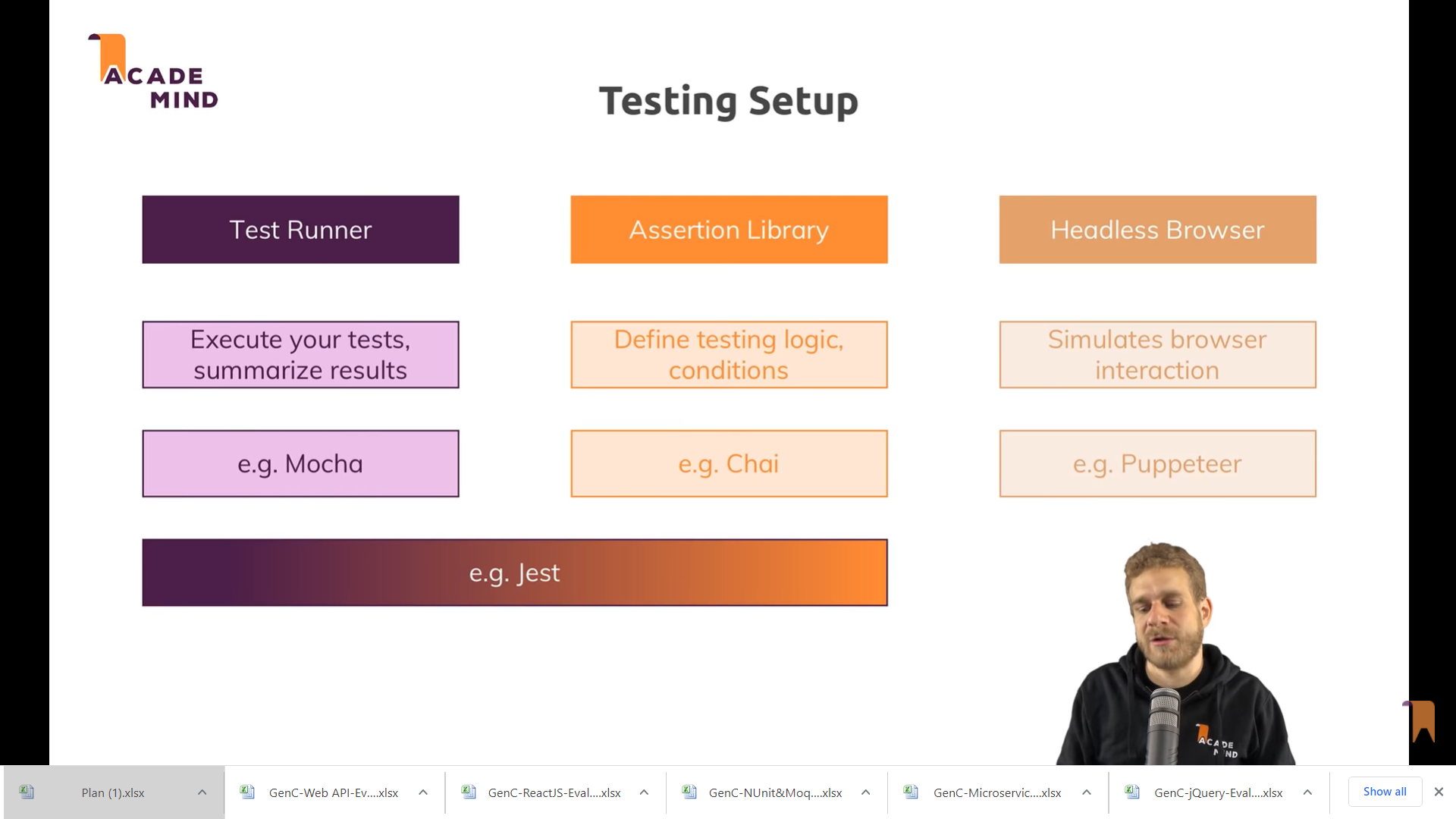


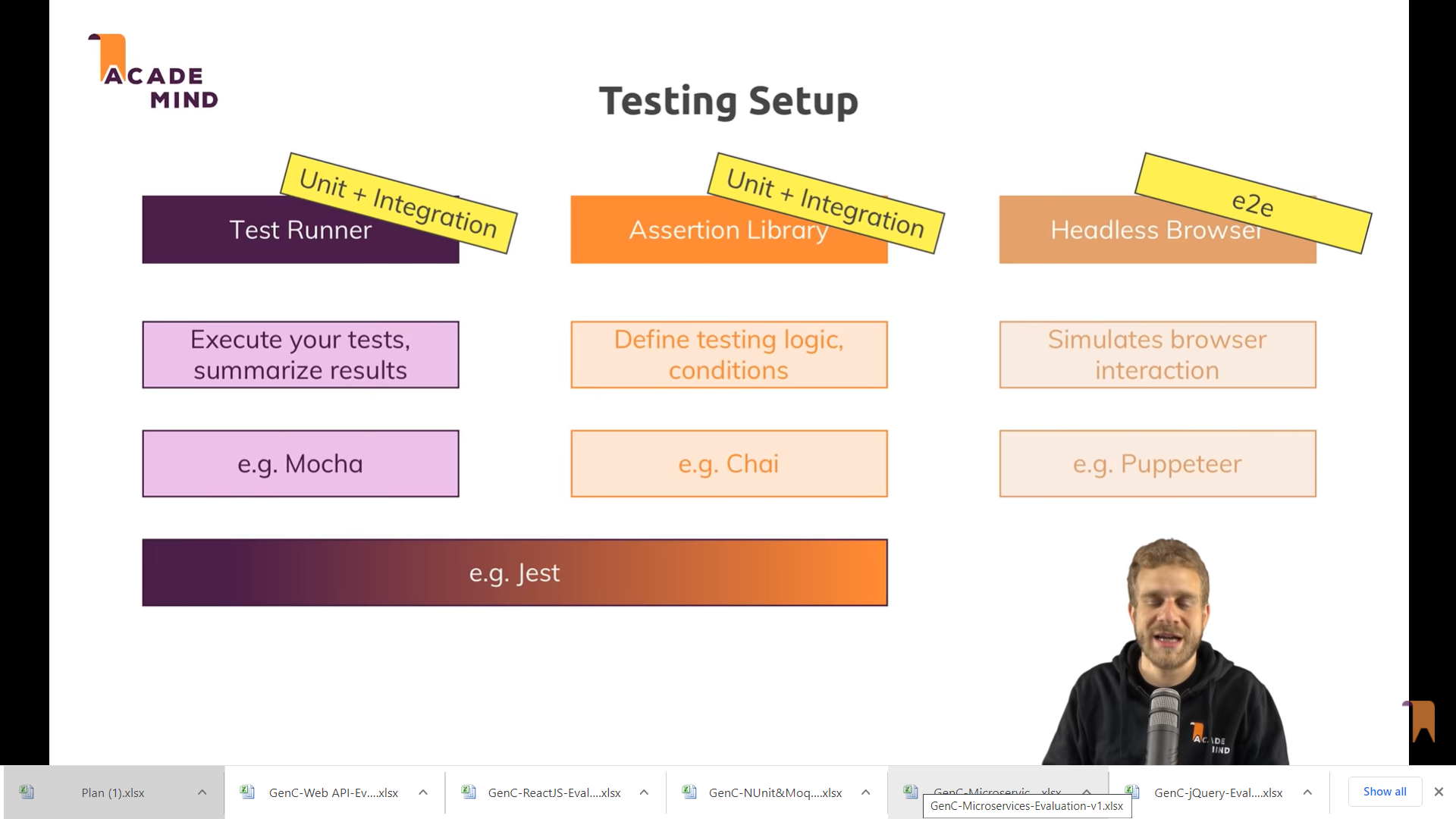












[**https://www.sitepoint.com/unit-test-javascript-mocha-chai/**](https://www.sitepoint.com/unit-test-javascript-mocha-chai/)

**Mocha** and **Chai** are two JavaScript frameworks commonly used together for unit **testing**. **Mocha** is a **testing** framework that provides functions that are executed according in a specific order, and that logs their results to the terminal window.

**Mocha** allows asynchronous testing, test coverage reports, and use of any assertion library. **Chai** is a BDD / TDD assertion library for NodeJS and the browser that can be delightfully paired with any javascript testing framework. Basically, **mocha** is a framework and **chai** is a library.

**Mocha** is a feature-rich JavaScript test framework running on Node. js and in the browser, making asynchronous testing simple and fun. **Mocha** tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases.

Install node.js

Install moca & chai

npm install mocha chai --save-dev

This installs the packages mocha and chai. [Mocha](https://mochajs.org/) is the library that allows us to run tests, and [Chai](http://chaijs.com/) contains some helpful functions that we’ll use to verify our test results.

* To include Chai, add var chai = require('chai');

at the top of the test file.

* Run the tests using the mocha command, instead of opening a browser.

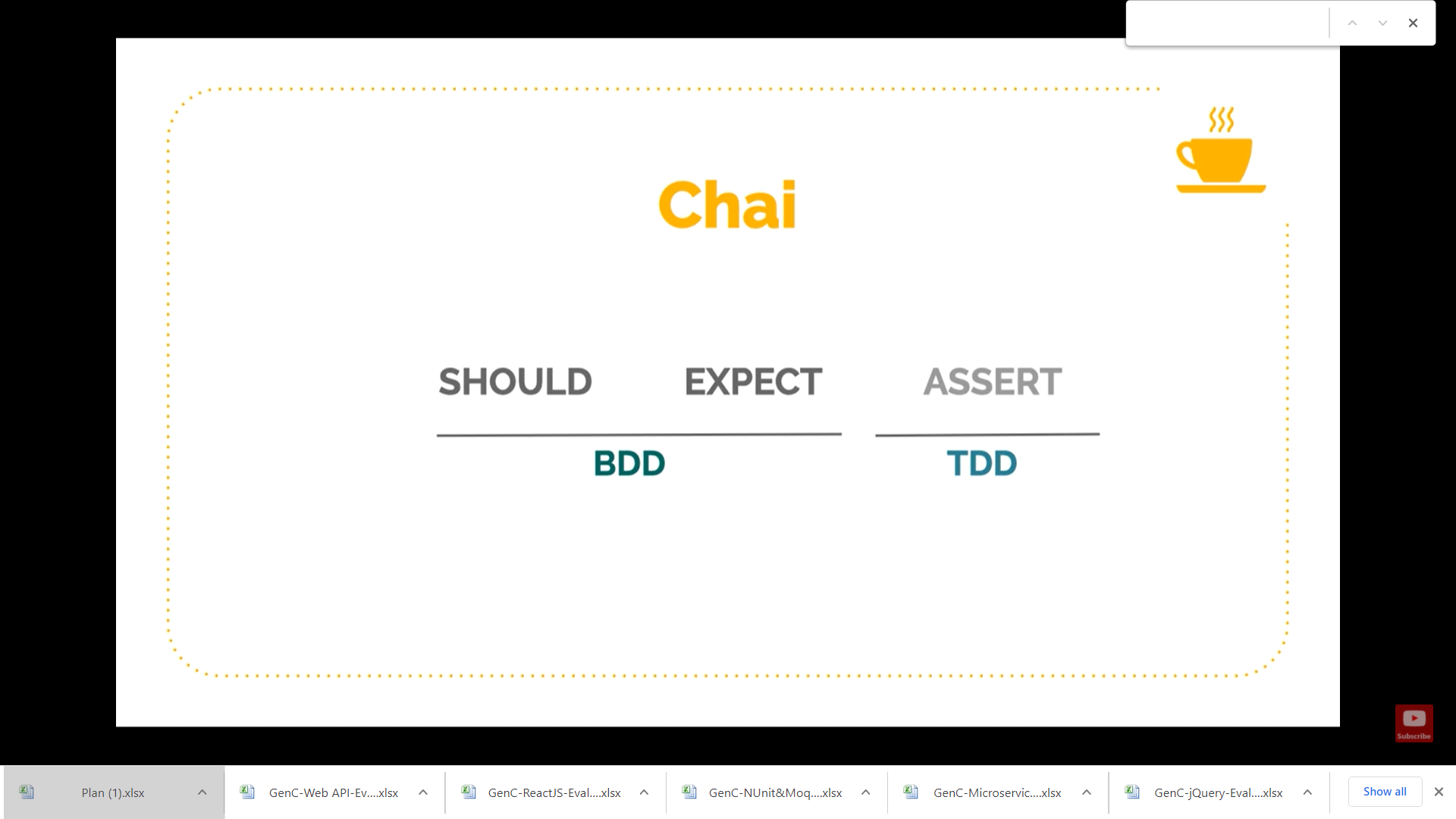
Setting up a Directory Structure

You should put your tests in a separate directory from your main code files. This makes it easier to structure them, for example if you want to add other types of tests in the future

The most popular practice with JavaScript code is to have a directory called test/ in your project’s root directory. Then, each test file is placed under test/someModuleTest.js. Optionally, you can also use directories inside test/

create test.js





**Md MochaChaiDemo**

**Cd MochaChaiDemo**

**Npm init**

**// initsilase applixatpon**

**Npm install –save mocha chai**

{

  "name": "mochachaidemo",

  "version": "1.0.0",

  "description": "",

  "main": "index.js",

  "scripts": {

**"test": "mocha"**

  },

  "author": "",

  "license": "ISC",

  "dependencies": {

**"chai": "^4.3.4",**

**"mocha": "^8.3.2"**

  }

}

**Test**

**AAA**

**Arrange Act Assert**

**Create a folder test , In that file first.js**

module.exports = function()

{

    return "hello";

}

**In firstTest.js**

var assert = require('chai').assert;

var first = require('../first');

describe('First', function()

{

it('first should return hello', function(){

assert.equal(first(), 'hello');

});

});

**Npm run test OR npm test**

**2nd example**

**.Js file**

module.exports=function(value)

{

    return value + 5;

}

**addTest.js**

var assert = require('chai').assert;

var add = require('../add');

describe('Add', function()

{

    var result = add(5);

it('it should add 5 to value and return 10', function(){

assert.equal(result,10);

});

it('it should return number', function(){

    assert.typeOf(result,'number');

});

});

3rd Example

Prog.js

module.exports = function add(n1,n2)

{

    return n1+n2;

}

**Subtract.js**

module.exports = function subtract(n1,n2)

{

    return n1-n2;

}

**Progtest.js**

var assert = require('chai').assert;

var add = require("../prog");

var subtract = require("../subtract");

describe('add' , function()

{

    var result = add(10,20);

    it('it should add 10 and 20 and return 30', function()

    {

        assert.equal(result,30);

    });

    it('it should add 10 and 20 and return 30', function()

    {

        assert.equal(result,30);

    });

});

describe('subtract' , function()

{

    var result = subtract(10,20);

    it('it should add 10 and 20 and return -10', function()

    {

        assert.equal(result,-10);

    });

});

[**https://www.c-sharpcorner.com/article/javascript-testing-using-mocha-and-chai/**](https://www.c-sharpcorner.com/article/javascript-testing-using-mocha-and-chai/)

[**https://www.sitepoint.com/unit-test-javascript-mocha-chai/**](https://www.sitepoint.com/unit-test-javascript-mocha-chai/)