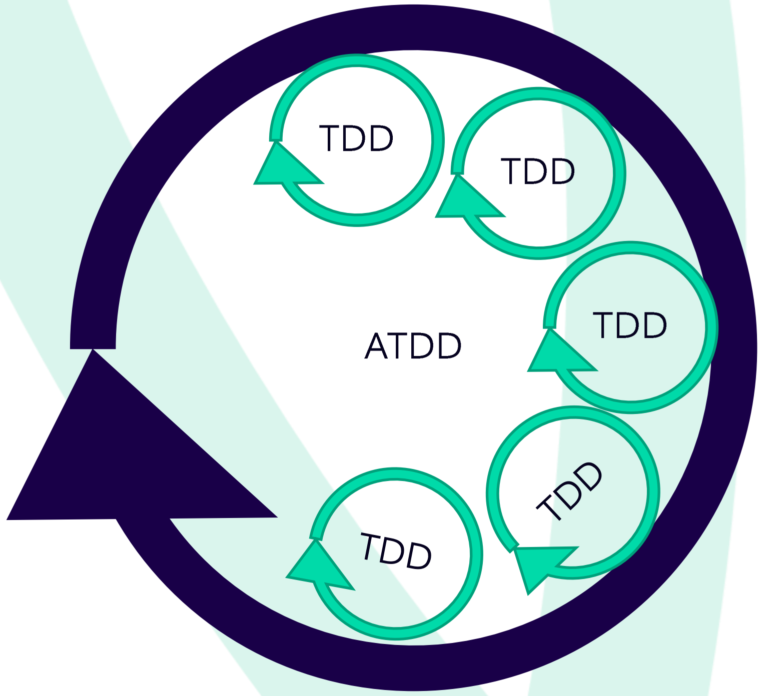
nagarro



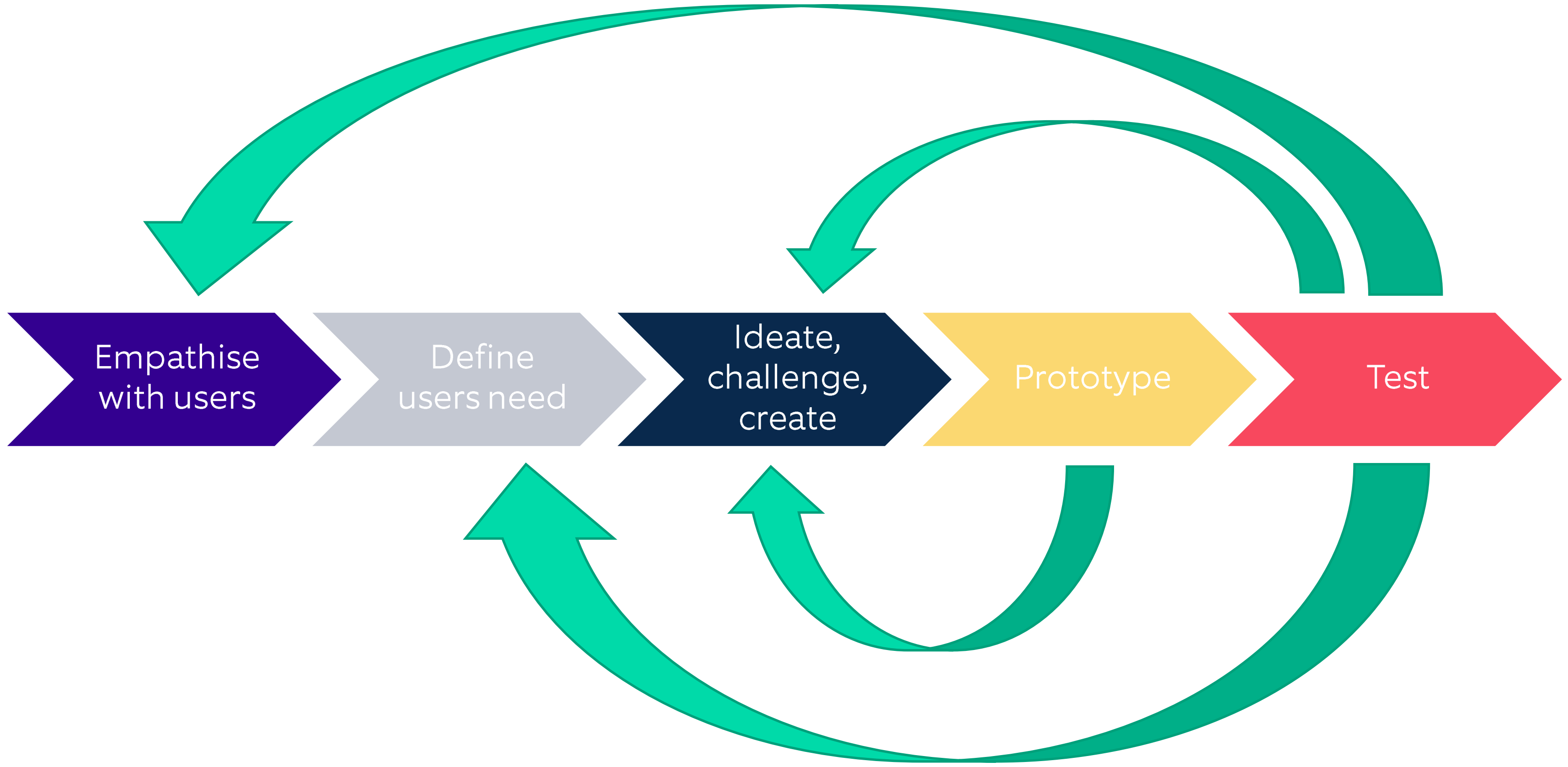
ATDD - Acceptance Test-Driven Development

Excercise – Design Thinking

- Gather in groups (3 – 5 people)
- Walk through the design thinking cycle for an online dinner reservation app
- Think about what features you want to provide to the customer
- Define 3 different acceptance tests that specify the expected behavior and write them down
- Choose a format of your choice for writing the tests
- Present your results to the audience

- 10 minutes for finding tests

Design Thinking



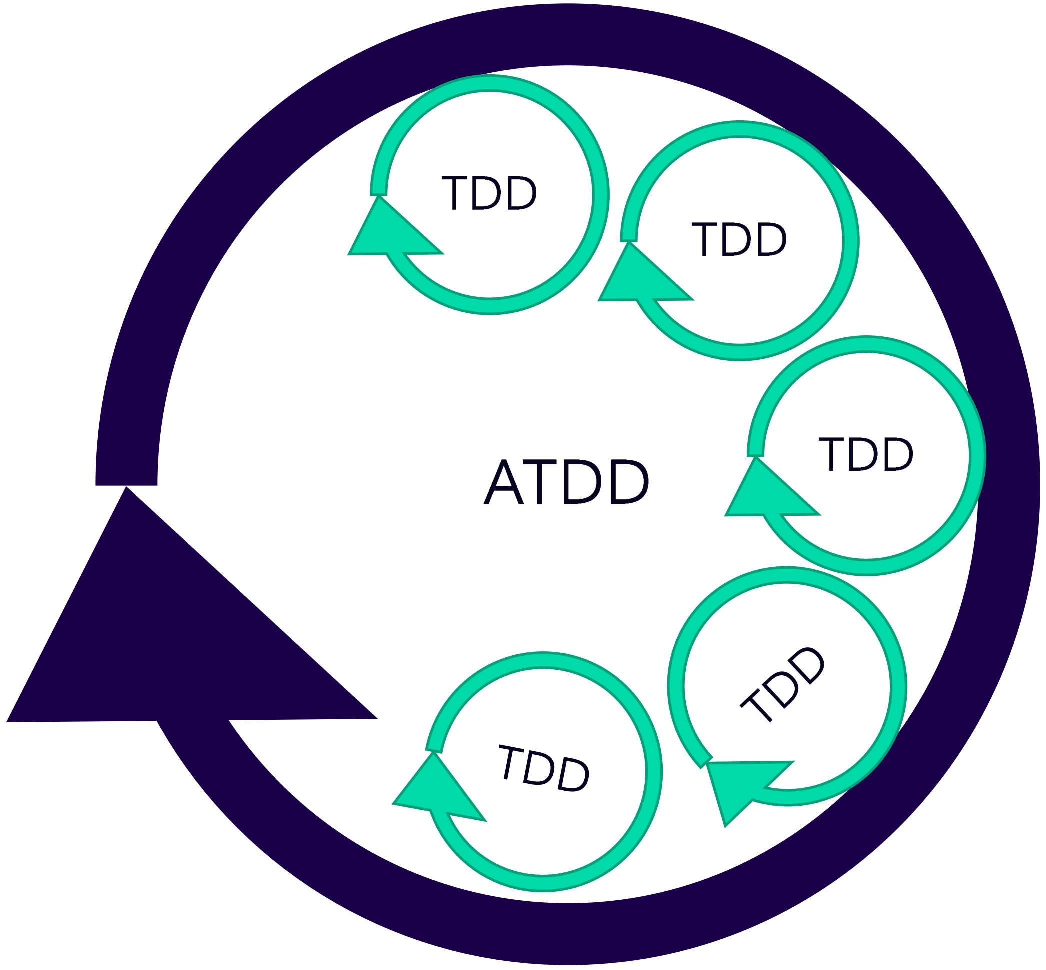
Purpose of Acceptance Tests

- Communication between customers, developers and testers
- Write acceptance tests before starting to code!
- Used as executable specification of the desired features and behaviour
- Serves as living always up-to-date documentation of the system‘s actual behaviour

The ATDD Cycle

The big picture

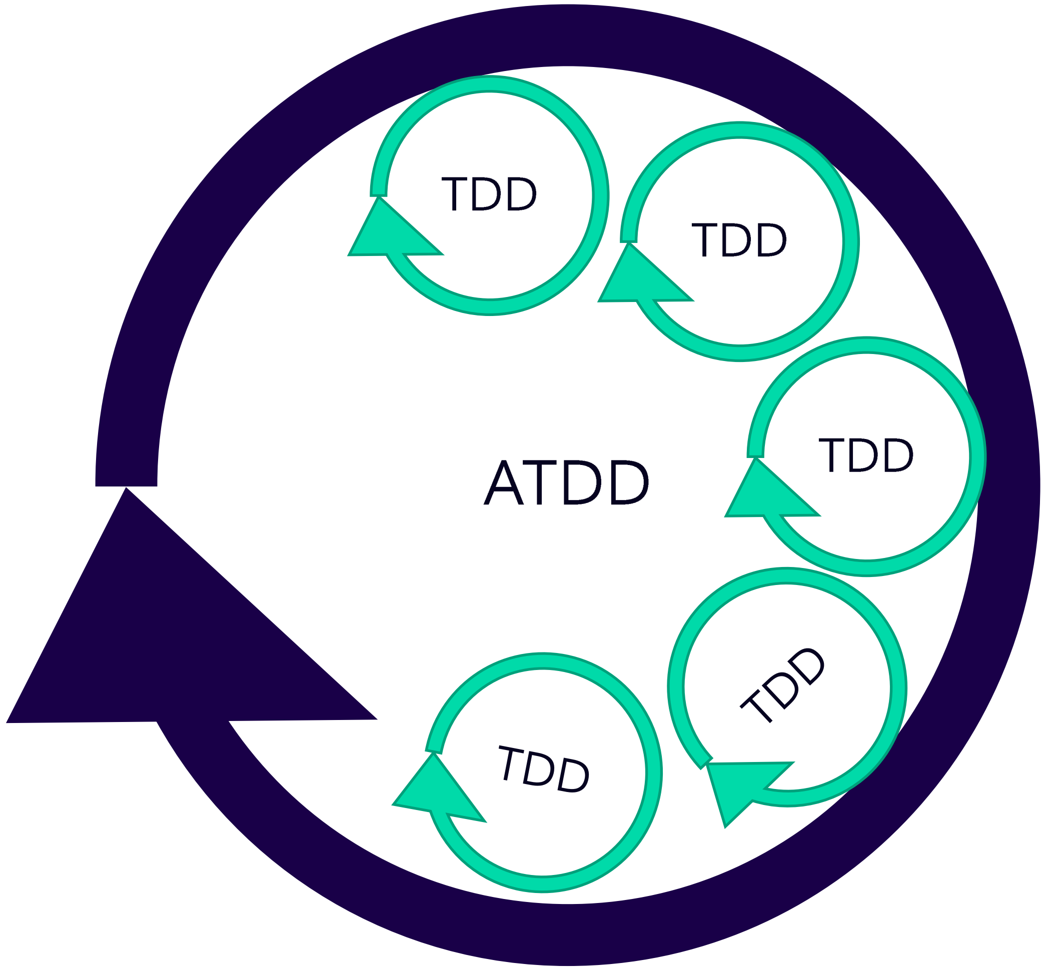
- ATDD: executable acceptance test specifications define the discovered features and expected behavior
- Several TDD-Cycles (plan-red-green-refactor) implement the feature until acceptance tests succeed
- Acceptances tests shall be green at the end of each sprint

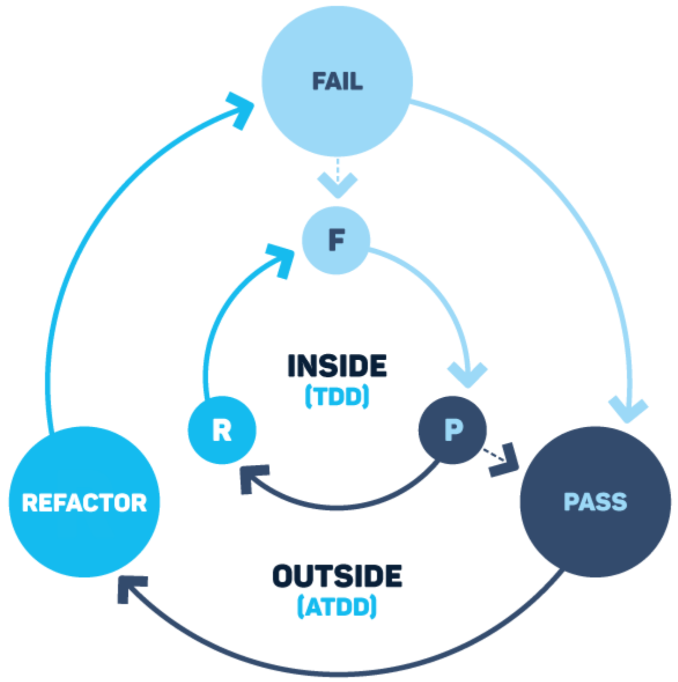


Exercise – Design a BDD Process

Create a BDD process diagram by sticking them to the whiteboard. The placement of the cards should show how your development process would look. Draw arrows between cards to indicate which order to do the activities in and when to iterate or act on feedback.

10 min.





ATDD with Gherkin

An outside in approach

- Conversation will be translated into Gherkin-Script
  - Features
  - Scenarios
- High-level acceptance tests to automate
  - Test must fail
  - Create enough production code to make test pass
- Technique used WITHIN BDD

Tools Supporting Gherkin Language

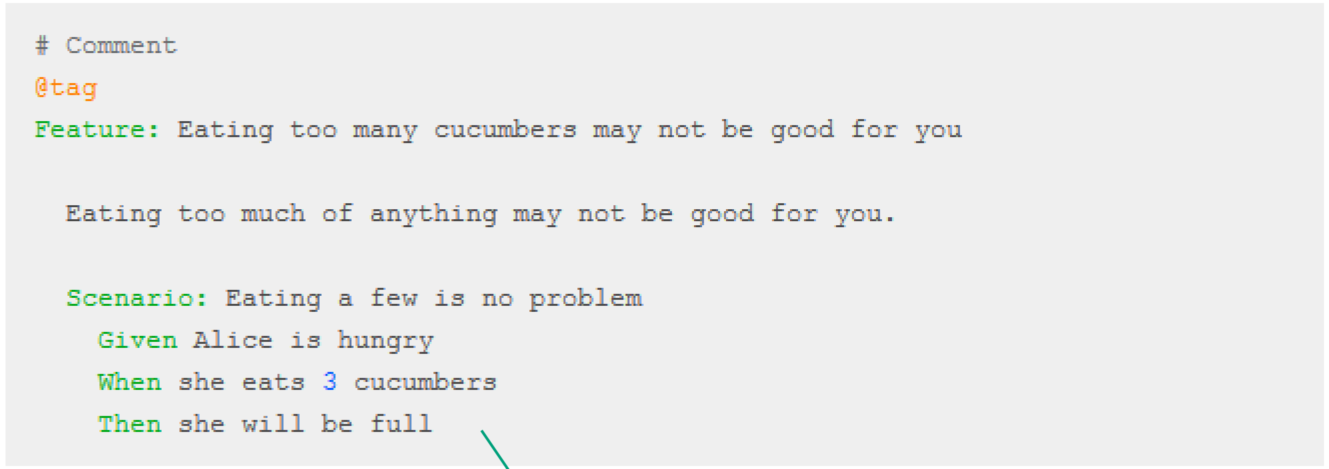
Behaviour driven development

cucumber

Executable Specifications

Free, open source, any platform

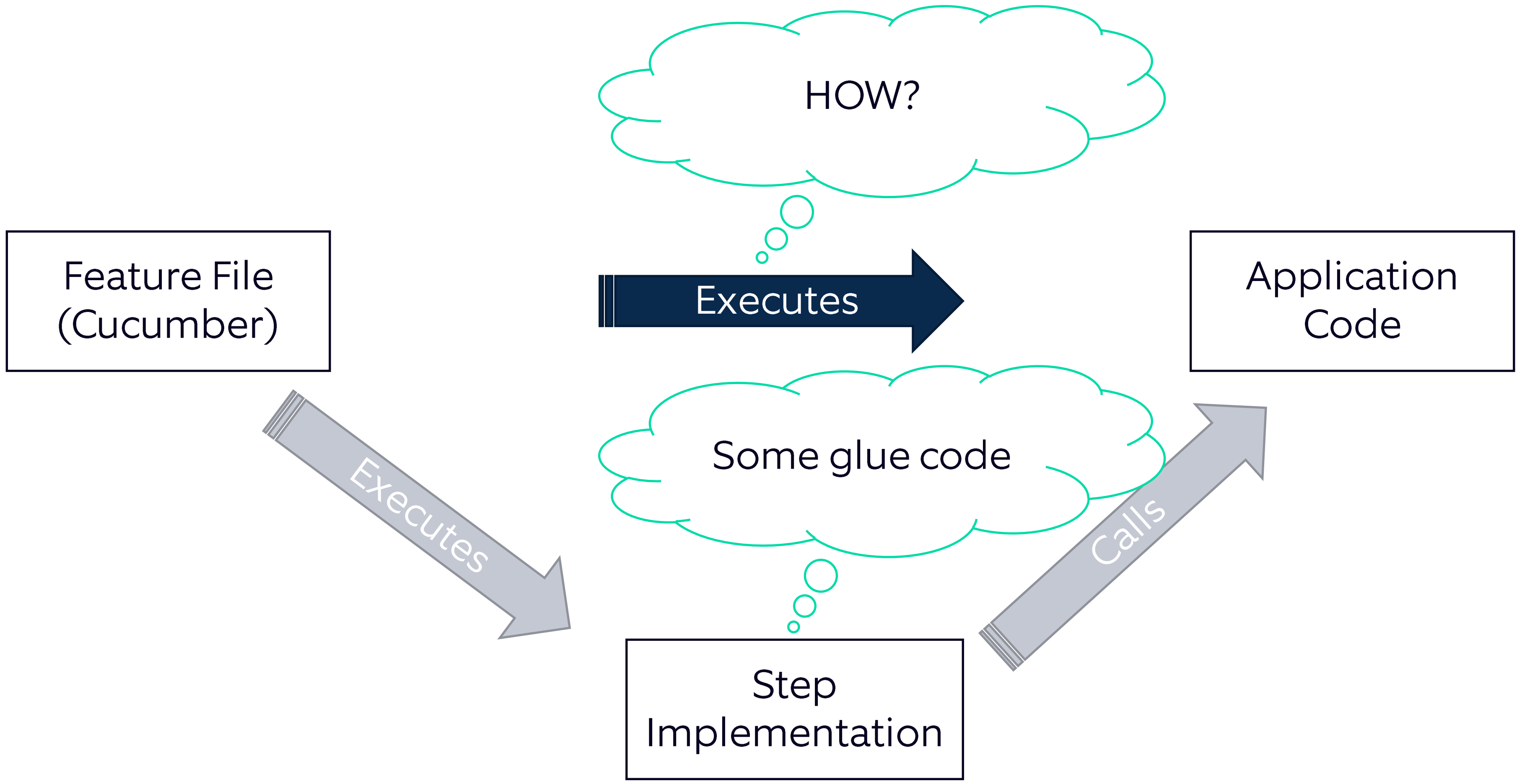
specflow

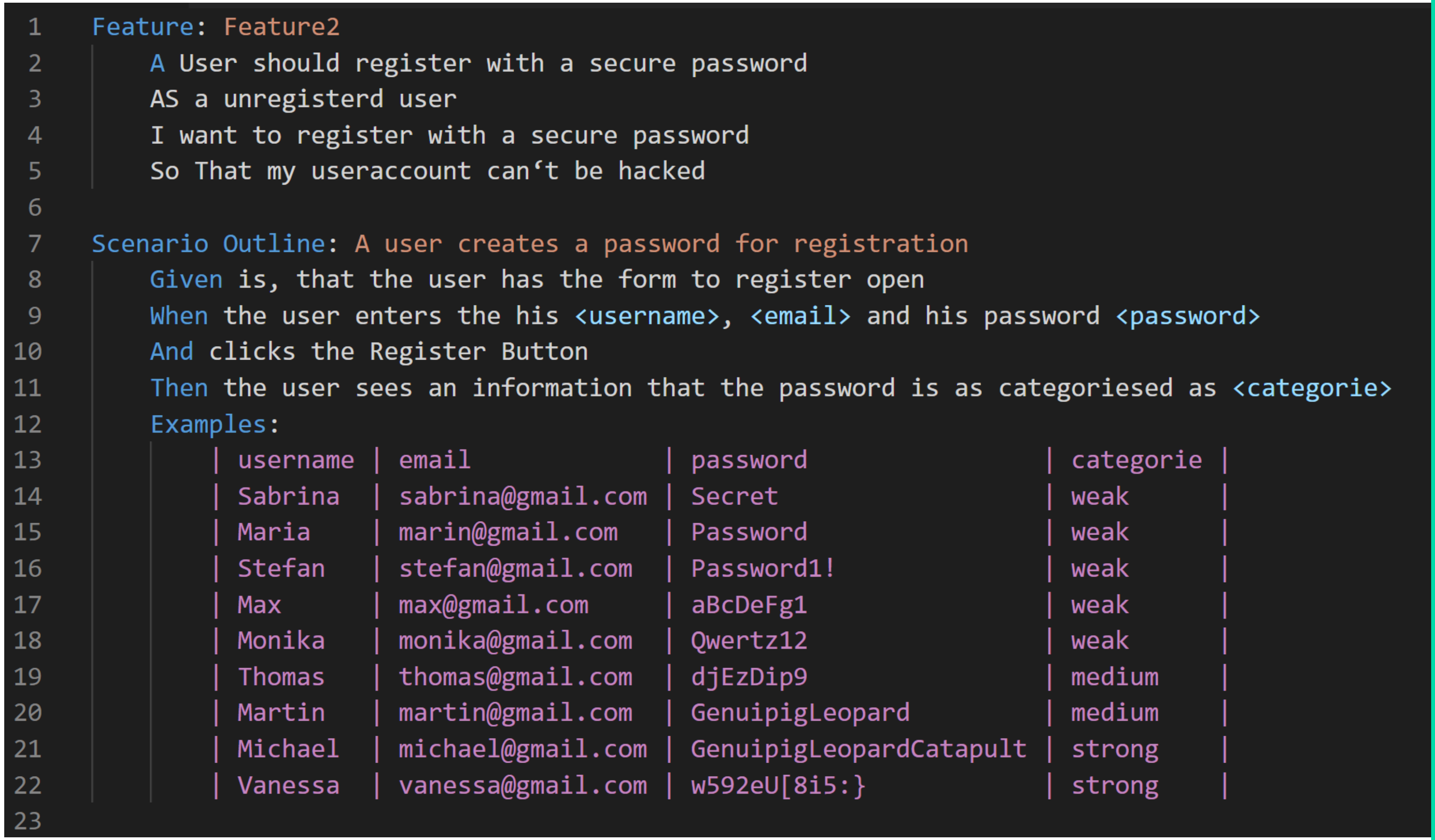


```
# Comment
@tag
Feature: Eating too many cucumbers may not be good for you

  Eating too much of anything may not be good for you.

  Scenario: Eating a few is no problem
    Given Alice is hungry
    When she eats 3 cucumbers
    Then she will be full
```

Write Scenarios in *.feature files

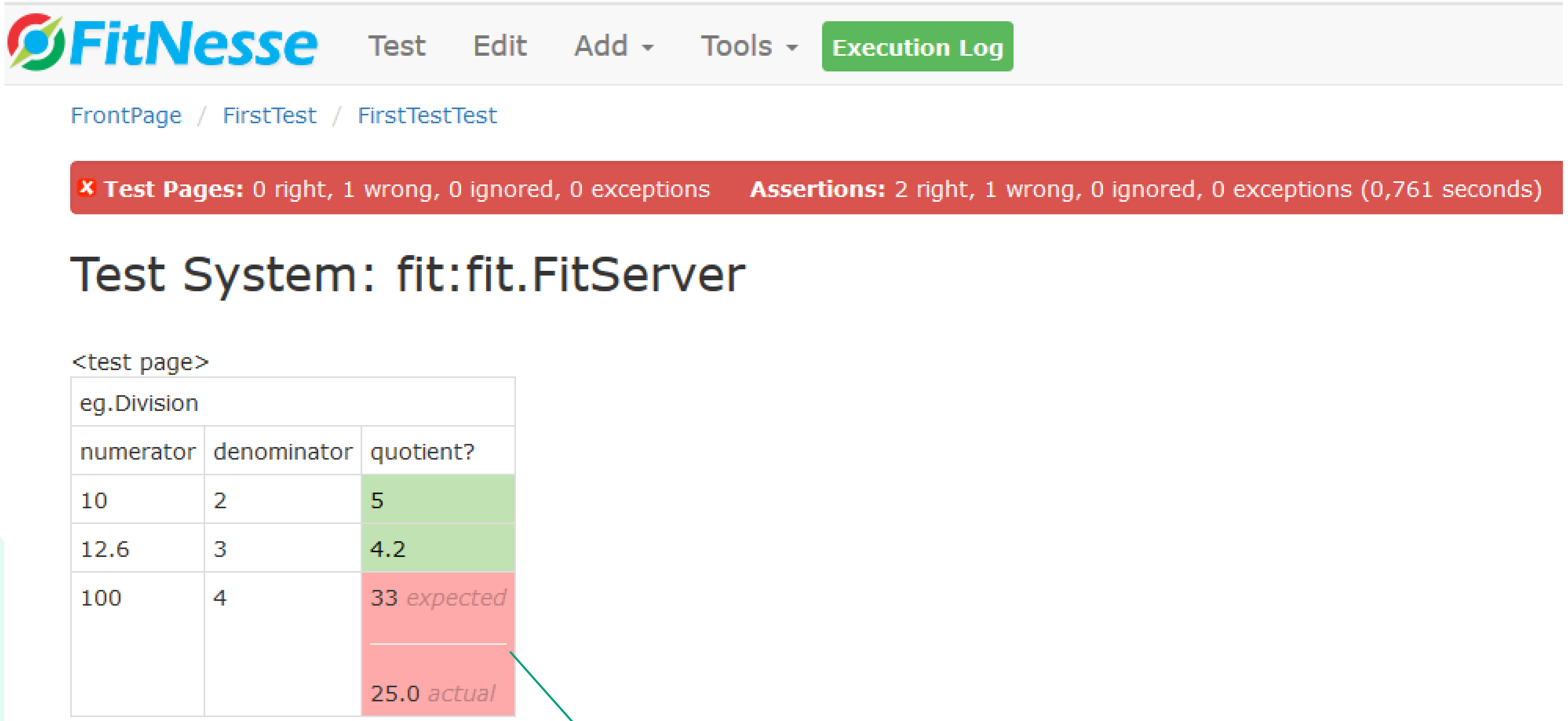
Specflow/Cucumber How is it executed?





```
Feature: Feature2
    A User should register with a secure password
    AS a unregisterd user
    I want to register with a secure password
    So That my useraccount can‘t be hacked

Scenario Outline: A user creates a password for registration
    Given is, that the user has the form to register open
    When the user enters the his <username>, <email> and his password <password>
    And clicks the Register Button
    Then the user sees an information that the password is as categoriesed as <categorie>
    Examples:
        | username | email              | password                 | categorie |
        | Sabrina  | sabrina@gmail.com  | Secret                   | weak      |
        | Maria    | marin@gmail.com    | Password                 | weak      |
        | Stefan   | stefan@gmail.com   | Password1!               | weak      |
        | Max      | max@gmail.com      | aBcDeFg1                 | weak      |
        | Monika   | monika@gmail.com   | Qwertz12                 | weak      |
        | Thomas   | thomas@gmail.com   | djEzDip9                 | medium    |
        | Martin   | martin@gmail.com   | GenuipigLeopard          | medium    |
        | Michael  | michael@gmail.com  | GenuipigLeopardCatapult  | strong    |
        | Vanessa  | vanessa@gmail.com  | w592eU[8i5:}             | strong    |
```

Tools: FitNesse



Test Pages: 0 right, 1 wrong, 0 ignored, 0 exceptions **Assertions:** 2 right, 1 wrong, 0 ignored, 0 exceptions (0,761 seconds)

Test System: fit:fit.FitServer

<test page>

eg.Division		
numerator	denominator	quotient?
10	2	5
12.6	3	4.2
100	4	33 *expected* 25.0 *actual*

Write tests in wiki pages and execute it nagarro

Hands on: ATDD

Hands-On: ATDD – Number Converter

- We will work with some legacy code which is a simple number converter.
- At the beginning you'll find some converters for decimals, binaries and hexadecimals.
- The **ATDD** part is based on **specflow** (https://specflow.org/).
- The tests are divided in
  - feature files
  - C# step files
- You can execute the test by executing the Test Explorer
- The implementation of the tests is missing.
- **Implement the specflow steps** to test the requirements defined in **ConvertingIntoDifferentFormats.feature.**
- Work in repository “13-ATDD”.

See „materials/hands-on/csharp/13a-hands-on_atdd“

Hands-On: ATDD – Number Converter

- We will work with some legacy code which is a simple number converter.
- At the beginning you'll find some converters for decimals, binaries and hexadecimals.
- The **ATDD** part is based on **cucumber** (https://cucumber.io/).
- The tests are divided in
  - feature files
  - a JUnit runner i.e. NumberConverterTest.java
  - step files You can execute the test by executing the Test Explorer
- The implementation of the tests is missing.
- **Implement the specflow steps** to test the requirements defined in **ConvertingIntoDifferentFormats.feature.**
- Work in the repository “13-ATDD”.

See „materials/hands-on/java/13a-hands-on_atdd“