Basics of UI Testing

PA1417 Lecture Unit 5

Objectives

Today, you are going to learn about:

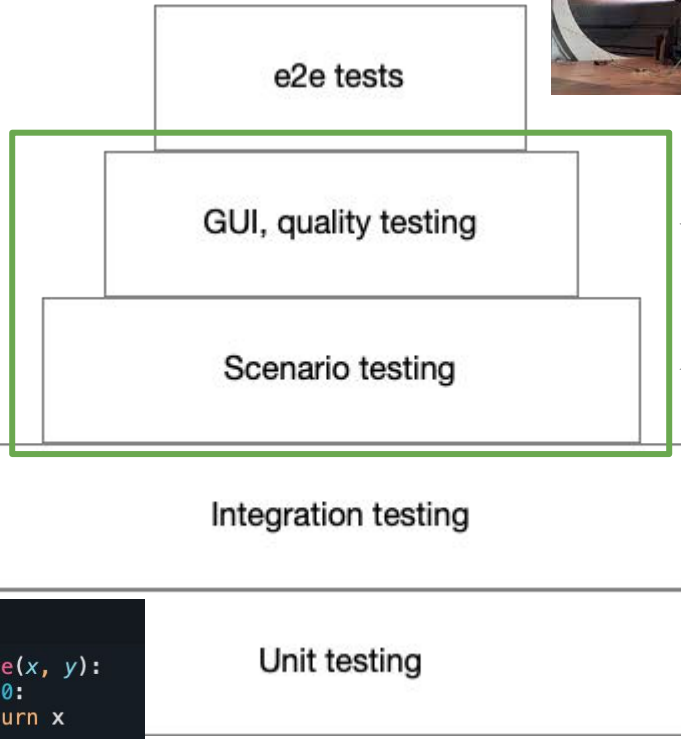
Part I: Theory

- How to apply use cases in test design
- How to design tests for graphical user interfaces (GUI)
- Testing the system through the GUI vs testing the GUI

Part II: Hands-on

- Cypress framework

Test pyramid



Test, if quality of the GUI is up to the standard

Test, if the system can do something useful

Test, if parts can work together

Test, if individual parts work

```
untitled
1  def positive(x, y):
2      if x > 0:
3          return x
4      else:
5          return y
6
```

GUI testing

Graphical User Interface testing focus on two aspects of the system:

Functionality

Does the feature as a whole work?

Unit and integration testing of GUI components

Does individual bits and pieces that make the GUI work?

Quality

How it works?

- Usability

  perform the tasks safely, effectively, and efficiently while enjoying the experience
- Accessibility

  usable by people with disabilities
- User experience

  Overall experience of using the system

How do you test functional aspects of the GUI?

We need use cases!

- A use case is an isolated task that a system can perform, e.g. authenticate a user, upload a file, print a report. User stories make good use cases
- Use cases typically involve a scenario of multiple steps

Example use case: User signup

1. Click “Sign Up” button
2. Enter username
3. Enter password
4. Click submit
5. If all is correct,
   show a welcome message

Extensions:

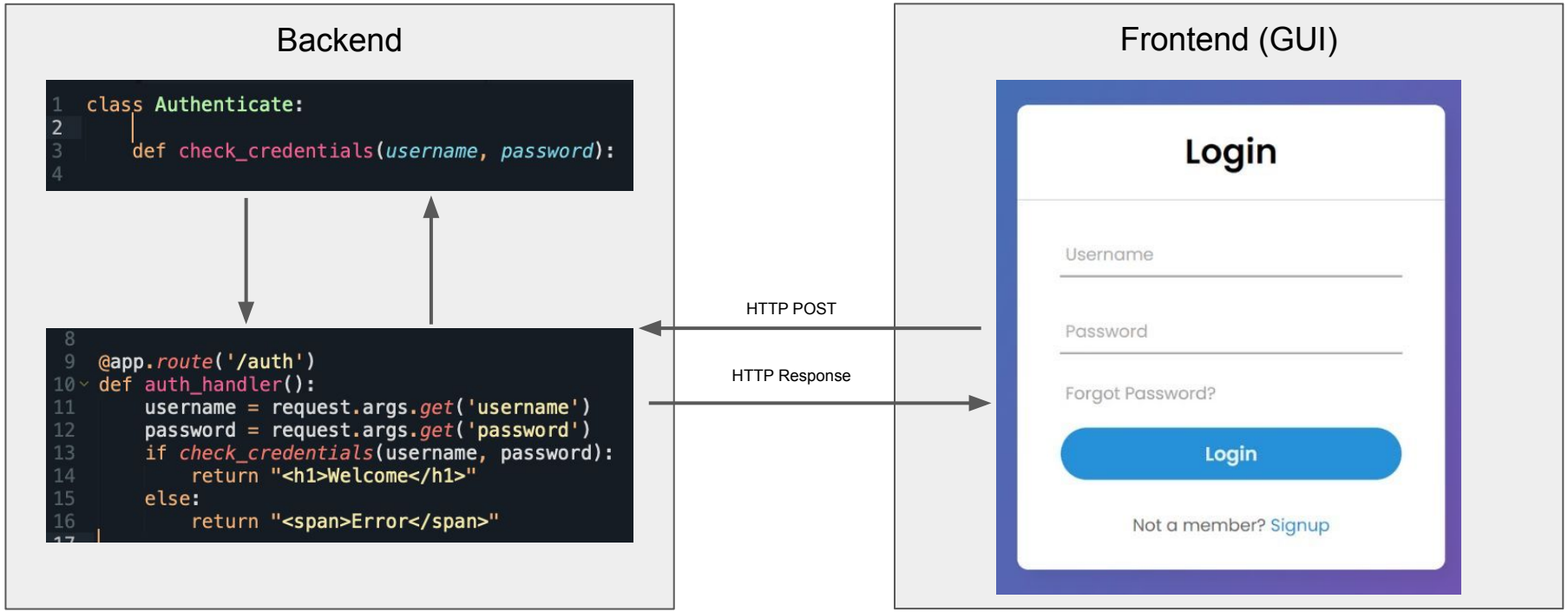
5b. If the username is already in use, show an error.

How many GUI tests do you need?

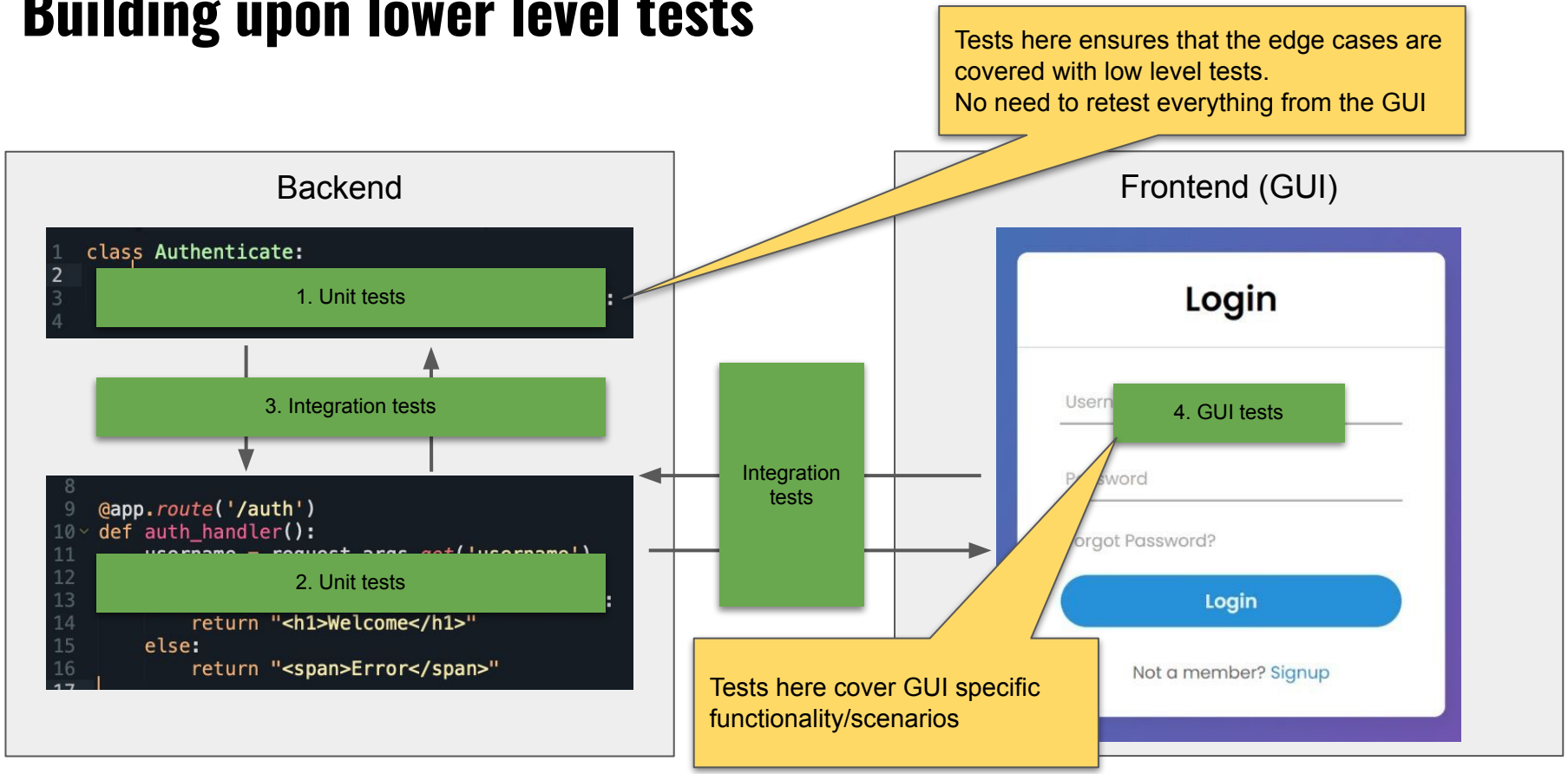
Options include:

1. Test it thoroughly (BVA, EP, .., branch, statement coverage)
    a. Expensive and time consuming due to the large number of parameters and their combinations
    b. Substantial effort to update test cases if the underlying system changes
2. Focus on key scenarios only
    a. A feasible approach
    b. Edge cases remain untested
3. Pure GUI tests build upon lower level tests
    a. The preferred way!
    b. Focus on the test cases relevant for the GUI

Building upon lower level tests



Building upon lower level tests



GUI testing

Graphical User Interface testing focus on two aspects of the system:

Functionality

Does the feature as a whole work?

Unit and integration testing of GUI components

Does individual bits and pieces that make the GUI work?

Quality

How it works?

- Usability

  perform the tasks safely, effectively, and efficiently while enjoying the experience
- Accessibility

  usable by people with disabilities
- User experience

  Overall experience of using the system

How do you test quality of the GUI?

Options include:

1. User tests
    a. Observe users performing tasks on the system (or its prototype), identify pain points, improve, iterate
    b. Expensive to run but very informative
2. Heuristics
    a. Follow a checklist of best practices
3. Measure objective metrics
    a. Task time
    b. Customer retention
    c. Error frequency
    d. Learning curve
    e. Help frequency
    f. etc..

Selecting a test oracle

Quiz: for each of the following test scenarios, what type of oracle is most appropriate?

1. How easy is it to navigate from one page to another on the website?
2. How often does it happen that a user inadvertently creates an item which he/she deletes right afterwards?
3. Does the process of signing up follow the intended workflow?
4. How well can a colorblind person read all text?

The quiz will be available after the lecture at
https://docs.google.com/forms/d/e/1FAIpQLSdTy_7kWzuyFLENkQ3dHDFO9BxwEt2lObB2AN_Dw3wondRGTg/viewform?usp=sf_link

Heuristics approach

There are many good checklists to use:

- Web Content Accessibility Guidelines (WCAG) focus on accessibility
- Platform specific guidelines (Web/iOS/Android)
- GUI framework specific guidelines

Some are supported by tools

- Automated accessibility checking (based on WCAG)
- Browser tools

Interpretation of results is often subjective

Accessibility

Helping disabled users to use your system:

- Public e-services must be accessible to everyone
- Common courtesy to your users

Accessibility for everyone:

- High contrasts helps regular users in bright sunlight
- Easy to read/navigate GUI helps when a user is multitasking
- Web content should support a wide range of devices

Break