COMP1531

3.2 - User Acceptance

How do we know we've met the requirements?

INVEST

- I = Independent: user story could be developed independently and delivered separately
- N = Negotiable: avoid too much detail.
- V = Valuable: must hold some value to the client
- E = Estimable: *we'll get to this in a later lecture*
- S = Small: user story should be small
- T = **Testable**

User Acceptance Criteria

- Break down a user story into criteria that must be met for the user, or customer, to accept
- Written in natural language
- Can be refined before implementation

Example

As a user, I want to use a search field to type a city, name, or street, so that I could find matching hotel options.

- The search field is placed on the top bar
- Search starts once the user clicks “Search”
- The field contains a placeholder with a grey-colored text: “Where are you going?”
- The placeholder disappears once the user starts typing
- Search is performed if a user types in a city, hotel name, street, or all combined
- The user can’t type more than 200 symbols

Best practices

- Acceptance criteria should not be too broad
- ... but nor should they be too narow
- Minimise technical detail
  - They can be more technical than the story itself, but client still needs to understand them
- While they can be updated during development, they should first be written *before* it starts

From Criteria to Testing

- *Acceptance Tests* are tests that are performed to ensure acceptance criteria have been met
- Not all acceptance criteria can easily be mapped to *automated* acceptance tests
- Acceptance tests are *black-box* tests

Aside: white-box vs black-box testing

- Black box testing - choosing test data only from the specification
- White box testing - choosing test data with knowledge of the implementation

Example 2:

As a user, I can log in through a social media account.

- Can log in through Facebook
- Can log in through LinkedIn
- Can log in through Twitter

Scenario Oriented AC

- The Acceptance criteria from before are often referred to a rule-based AC
- Sometimes it is preferable to have AC that describe a scenario
- This can be done in the Given/When/Then format:
  - *Given* some precondition
  - *When* I do some action
  - *Then* I expect some result

Example 3:

As a user, I want to be able to recover the password to my account, so that I will be able to access my account in case I forgot the password.

Scenario: Forgot password

Given: The user has navigated to the login page

When: The user selected forgot password option

And: Entered a valid email to receive a link for password recovery

Then: The system sent the link to the entered email

Given: The user received the link via the email

When: The user navigated through the link received in the email

Then: The system enables the user to set a new password

Which one to use?

- Rule-based acceptance criteria are simpler and generally work for all sorts of stories
- Scenario-based AC work for stories that imply specific user actions, but don't work for higher-level system properties (e.g. design)
- Scenario-based AC are more likely to be implementable as tests
- **While scenario-based AC are worth knowing about, for your project we recommend simple rule-based AC**

Further reading

- https://www.mountaingoatsoftware.com/blog/the-two-ways-to-add-detail-to-user-stories
- https://www.altexsoft.com/blog/business/acceptance-criteria-purposes-formats-and-best-practices/
- https://dzone.com/articles/acceptance-criteria-in-software-explanation-exampl