# 1 Interview Transcripts

The transcripts given in this document are transcripts of the interviews described in the paper "From User Stories to End-to-end Web Testing" by Humaid Mollah and Petra van den Bos.

## 1.1 Questions

This subsection states the questions that were asked to each tester. Each question is part of one of the three categories "Select user story", "Transform User Story into Test Cases", or "Implement Test Cases".

**Select User Story**

1. How do you identify the most important functionality of a web application to test?
2. In which cases do you not test a certain feature/functionality of a web application?

**Transform User Story into Test Cases**

3. Does the concept of Basic and alternate flows help identify the most important use case scenarios? Is there a different technique that you use to identify the workflow of an app?
4. How many exceptional/abnormal scenarios do you consider when testing a feature/functionality (referring to alternate flows)? How do you identify them?
5. How many complex use case scenarios (with multiple alternate flows) should you test?
6. How do you choose data inputs for your test cases? Which methods do you use?

**Implement Test Cases**

7. The case study uses tools such as the Selenium Web-Driver and Cypress to implement end-to-end tests. Would you consider these as usable approaches or are there any other tools that you use?

## 1.2 Interview Transcripts

This subsection gives the transcripts of the answers of each tester that was interviewed.

### 1.2.1 Software Tester 1

**Select User Story**

1. Read issue title, look at priority queue in GitLab (not always sorted), discuss with PO's.
2. Small bug fixes, textual changes, basically determine low-risk low impact issues and skip them.

**Transform User Story into Test Cases**

3. Yes, I call the basic flow the happy path.
4. It highly depends, bigger issues have more alternative flows and therewith abnormal ones
5. Again, this depends. Customer expectations, available budgets, used hours on an estimate, and the likelihood of breaking. Also, input validation is easy to automatically test, i.e. it also depends on what "should you test" means (manual or automatic).
6. I choose data inputs using all of these methods (Random, Usage-based, Boundary, Partition)

**Implement Test Cases**

7. I use Cypress but Selenium also works similarly

### 1.2.2 Software Tester 2

**Select User Story**

1. Check the product backlog, choose the one with high priority, it is usually not very difficult to determine
2. Indeed, user stories that are tested in unit tests can be skipped

**Transform User Story into Test Cases**

3. Yes, my definition for alternate/abnormal flows: abnormal flows are flows that will not likely be used by end-users, alternative flows are flows that are likely to be used apart from the regular flow
4. Depends on the functionality you are testing. The 80/20 principle usually does apply though
5. for end-to-end tests, the broader use-case scenarios which involve both the front-end and back-end, so that big features or flows can be tested well
6. Usage-based Testing could be considered the most commonly used option as the choice of data inputs depends on the feature being tested

**Implement Test Cases**

7. Yes, These are the easiest options to implement end-to-end tests.

### 1.2.3 Software Tester 3

**Select User Story**

1. The product owner can help choose the most important one, but on GitLab also you can find the most important user stories for the sprint
2. Bug fixes and maybe small changes in the front end can be avoided

**Transform User Story into Test Cases**

3. Yes, the basic workflow should be recognized and the alternate flows.
4. Depends on the user story you are testing, the bigger functionalities will have more alternate flows
5. This depends on the budget and what is required really
6. All options are usable and it depends on what you are testing, but Boundary Testing is a good technique to test both positive and negative scenarios.

**Implement Test Cases**

7. Yes, these are the ones which are most used, the other ones as you mentioned like Visual locators cannot be always used like for big web apps