Learning Objectives

Learners will be able to…

- **Describe the purpose of user acceptance testing**
- **Identify the steps in the user acceptance testing process**
- **Identify common tools for user acceptance testing**

info

Make Sure You Know

There is no prerequisite knowledge for this assignment.

Limitations

This is a theoretical discussion about user acceptance testing. We will see a practical application of this kind of testing in the next assignment.

User Acceptance Testing

What is User Acceptance Testing?

User acceptance testing is a method of software testing that is used to determine whether or not a system satisfies the business requirements that guided its design and development. The purpose of user acceptance testing is to ensure that the software meets the user requirements and performs as expected in a real-world environment. Acceptance tests can be performed by the customer, either directly or through representatives, to determine whether or not to accept the system. The tests are designed to confirm that the system meets the requirements specified in the system specification. Acceptance tests are often based on scenarios or use cases that describe the user interactions with the system.

Scenarios are short descriptions of how a user interacts with a system. They describe a user's actions, the system's expected responses, and the results of the interaction. Scenarios can be used to illustrate a particular use case or a set of use cases.

Use cases are more detailed, formal descriptions of how a user interacts with a system. They are usually written as a narrative in which a user interacts with a system, typically to achieve a particular goal. Each use case describes a particular user interaction with the system and includes details such as the user's objectives, steps taken to achieve those objectives, and results of the interaction. Use cases are used to identify and document the functional requirements of a system.

Scenarios and Use Cases Creation

Creating Scenarios and Use Cases

Creating scenarios and use cases for a system requires a thorough understanding of the system and its requirements.

The first step is to identify the user themselves, it is important to determine the different types of users who will interact with the system, as well as any special requirements or scenarios that might arise.

Then consider the user's goals and objectives, and then create a narrative that describes how the user will interact with the system to achieve those goals. Once the narrative is complete, it can be broken down into individual steps and scenarios, which can then be used to create the use cases. Use cases should clearly describe the steps taken, the expected results, and any special conditions that must be met in order for the use case to be considered successful.

Personas are made-up individuals who represent actual use cases of a product. They are not strictly necessary for creating scenarios and use cases, but they can be helpful when it comes to understanding the user's goals, motivations, and behaviors. Personas provide a way to put a face to the user and to think about how the user might interact with the system. Personas can also be used to create more detailed scenarios and use cases that are tailored to the user's specific needs and behaviors. And first of all they allow the whole team to meet their end user.

Creating personas for a system requires a thorough understanding of the users and their needs. The first step is to identify the types of users who will interact with the system. Once the user types have been identified, it is important to research and gather data about the users in order to create detailed profiles. This data should include demographic information, behavioral patterns, goals and objectives, and any other relevant information. Once the data has been gathered, it can be used to create detailed personas that represent the different types of users. Personas should be detailed enough to provide insight into the users' motivations, behaviors, and preferences. To add a visual effect, you can create your persona in a Facebook profile manner(for example). Add their favorite colors, favorite style, browser, used devices, and you will hesitate less when developing design guidelines or testing plans.

Your next steps will depend on the type of use case you are trying to create. Generally, you will need to define the goal of your use case, identify the actors involved, determine the primary and secondary flows of the use case, and describe how the use case will be implemented. You may also need to consider how to test the use case to ensure that it works as desired.

Scenario and Use Case Example

User Acceptance Testing Example

Below is an example of an acceptance testing scenario and use case. Here, the user is searching for information using Google.

> **Goal:** To allow users to search for information on the web using Google.
>
> **Actors:** User[here can be your persona]
>
> **Primary Flow:**
> 1. User navigates to the Google homepage
> 2. User enters a search query into the search bar
> 3. Google displays a list of results related to the query
>
> **Secondary Flow:**
> 1. User clicks on a result from the list of results
> 2. Google displays the webpage associated with the result
>
> **Implementation:**
> The use case will be implemented using HTML, CSS, and JavaScript.
>
> **Testing:**
> The use case will be tested to ensure that the search results are accurate, the results are displayed correctly, and that the user is able to navigate to the correct webpage associated with the result.

Goal

The goal of acceptance testing is to determine if the user requirements are met. It involves checking if the product meets the specific user needs, is easy to use and understand, and is fit for purpose. The goal of the test should be clearly defined and documented to ensure that all stakeholders understand and agree to the objectives.

Actors

The actors involved in acceptance testing are the end user, the developers, the testers, and the stakeholders. The end user should be given access to the product to evaluate it and provide feedback. The developers and testers should ensure that the product meets the requirements specified by the user before it is released. The stakeholders should review the product and provide input on any changes that should be made.

Sometimes personas are used instead of humans. Personas are key actors in acceptance testing. They are fictional representations of real users and are used to ensure that the product is designed and tested with the needs of the user in mind. Personas should be created to represent different types of users, such as beginner, intermediate, and advanced users, and should include details such as demographic information, goals, and preferences. Personas should be used throughout the acceptance testing process to ensure that all user needs are considered.

Primary Flow

The primary flow of acceptance testing should cover the major features and functionality of the product. It should include testing for usability, performance, security, and other areas. The primary flow should also include testing for any issues that may arise during the use of the product.

Secondary Flow

The secondary flow of acceptance testing should cover any additional features and functionality that may be added to the product. This could include testing for compatibility with other products, bug fixes, or performance enhancements. It should also include testing for any edge cases that may arise and any special requirements that need to be met.

Implementation

The implementation of acceptance testing should be done in a systematic and repeatable manner. The test cases should be written and documented, and the tests should be executed and monitored. The results should be recorded and analyzed to ensure that the product is functioning correctly.

Testing

The testing should be conducted in an environment that is as close to the actual user environment as possible. This will help to ensure that the product is tested in a realistic setting and that bugs and issues can be identified and addressed quickly. Once the tests are complete, the results should be reported and reviewed, and any necessary changes should be made to the product.

User Acceptance Testing Steps

Testing Steps

Step 1

Identify Users: Identify the users who will be involved in the user acceptance testing process. This should include both business and technical users, such as end users, developers, QAs, and stakeholders. It should also include any other users who may have input on the product, such as product managers, support staff, and other personnel. The user identification process should be managed by the product owner or product manager. They should be responsible for ensuring that all the necessary users are included in the process.

Step 2

Create Use Cases: Create user acceptance test cases that accurately reflect the user requirements. The use cases should be comprehensive, covering all the functions and features of the product. They should also be detailed enough to identify any potential issues or problems with the product. The use cases should be created by the developers and QAs, in collaboration with the stakeholders and end users. The stakeholders should provide input on the user requirements and the developers should create the test cases to reflect those requirements.

Step 3

Outline Criteria: Outline the criteria for user acceptance. The criteria should be based on the user requirements, and should include criteria for usability, performance, security, and other areas. The criteria should be established by the stakeholders and the developers, and should be reviewed and agreed upon by all parties involved.

Step 4

Provide Access: Provide the users with access to the application and the necessary resources for testing. This should include access to the product itself, as well as any test data and tools needed to complete the tests. The product owner or manager should be responsible for ensuring that the users have the necessary access and resources.

Step 5

Execute Tests: Execute the tests and document the results. The tests should be performed in an environment that is as close to the actual user environment as possible. The results should be documented in detail and any issues should be noted. The QAs should be responsible for executing the tests and documenting the results.

Step 6

Analyze Data: Analyze data and feedback from users. This should include any issues that were identified during the tests, as well as any suggestions or recommendations made by the users. The QAs and stakeholders should be responsible for analyzing the data and feedback from the users.

Step 7

Resolve Issues: Resolve any issues that arise. This should include fixing any bugs, making any necessary changes to the product, and retesting to ensure that the issues are resolved. The developers should be responsible for resolving any issues that arise.

Step 8

Retest: Retest the application after any fixes or changes. This should ensure that the product is functioning correctly and that any issues that were identified have been resolved. The QAs should be responsible for retesting the application after any changes or fixes are made.

Step 9

Sign Off: Obtain sign-off from users that the application meets their requirements. This should include a review of the test results and any feedback from the users. Once the product meets the user requirements, the users should be able to sign off on the product. The product owner or manager should be responsible for obtaining the user sign-off.

User Acceptance Testing Tools

User-Focused Testing Tools

Even though user acceptance testing revolves around personas, use cases, and other narrative-driven artifacts, software tools are still used with the testing itself. One common type of tool focuses on individuals interacting with the system and providing feedback. These tools are much more sophisticated than a separate survey after the fact.

Tools like Qualaroo and Marker.io offer ways for users to provide feedback (including filing bug reports) in the moment and from the same browser window as the website being tested. This kind of software allows for variance of individual users while making the act of feedback a more streamlined process.

Some tools like Userbrain provide not only testing feedback tools, but also have a large pool of quality assurance testers that can be used for the tests. You can even filter down the pool of testers to make sure your system is being tested by a wide variety of individuals.

Automated Testing Tools

While having individuals test your system is beneficial, it is not without its drawbacks. Working with individuals can be time consuming and it does not scale very well. That is why user acceptance testing is done with automated tools as well.

Products like TestComplete and Ranorex work with desktop, web, and mobile applications. They provide graphical tools to quickly and easily create tests and produce reports based on the output.

Some of these tools are robust suites of tools that replace some of the previous testing tools we have already discussed. Others are modular in nature and can easily fit into your testing workflow without causing any disruptions.

Cucumber.js

Cucumber is a testing tool based on a behavior-driven development (BDD) framework which is used to write acceptance tests for web applications. It allows automation of functional validation in an easily readable and understandable format. We are giving special attention to this tool as we will be using it in the next assignment.