Test Cases

# Positieve en negatieve testcases

Het doel van testen is om vast te stellen of de software doet wat het zou moeten doen. Of anders gesteld dat het geen fouten veroorzaakt wanneer dat niet hoort.

In de regel worden voor het testen testgevallen opgesteld op basis van een testontwerptechniek. Het gebruiken van een testontwerptechniek zorgt ervoor dat zowel de goedpaden als de foutpaden worden getest.

Het testen van goedpaden wordt ook wel aangeduid met “Positief testen“ en “Happy path testen“. En het testen van foutpaden wordt ook wel aangeduid met “Negatief testen“ en “Error path testing“.

**Definitie volgens Wikipedia voor goedpaden***In de context van softwareontwikkeling en testen is een goed pad een standaardscenario zonder uitzonderings- of foutcondities. Het goede pad voor een functie die creditcardnummers valideert, is bijvoorbeeld een waar geen van de validatieregels een foutmelding geeft en de uitvoering met succes tot het einde kan doorgaan en er een positief resultaat is.*

**Definitie volgens Wikipedia voor foutpaden***Negatief testen is een methode om aan te tonen dat een applicatie of systeem op een juiste manier omgaat met niet toegestane input en gedrag van de gebruiker.*

Positief testen

Positief testen is het uitvoeren van scenario’s zoals deze normaliter door een eindgebruiker zal worden gebruikt. Bij het uitvoeren van deze scenario’s worden alleen correcte en geldige gegevens gebruikt.

Negatief testen

Negatief testen zijn niet alleen bedoeld om mogelijke tekortkomingen aan het licht te brengen die ernstige gevolgen hebben voor het gebruik van de applicatie, maar kunnen ook een hulpmiddel zijn bij het bepalen van de omstandigheden waaronder de applicatie kan crashen. Ten slotte zorgt het ervoor dat er wordt aangetoond dat er voldoende validaties aanwezig zijn in de software.

Het gaat bij negatief testen er met name om, te testen of bij onjuiste input het systeem of applicatie niet crashd en de verwachte foutmeldingen als resultaat geeft. Het is namelijk een gegeven dat gebruikers soms niet toegestane invoer zullen gebruiken. En dan is het belangrijk dat de applicatie niet crasht en dat er een duidelijke melding wordt gegeven aan de gebruiker.

Voorbeeld positief en negatief testen

Stel we hebben een webapplicatie waar we blogs mee kunnen plaatsen. Voor de applicatie gelden onder andere de volgende vereisten(requirements)

* Het password moet letters en cijfers bevatten.
* Er mogen per gebruiker maximaal 5 blogs per dag worden geplaatst.
* In een blog mogen alleen .jpg foto’s worden gebruikt
* Een blogpost mag niet langer zijn dan 300 worden.

Uitwerking positieve testgevallen

Onderstaand een aantal positieve testgevallen die gebruikt kunnen worden. Het gaat hier dus om gevallen die een valide input hebben en dus niet mogen leiden tot foutsituaties.

* Creëer een password met letters en cijfers
* Creëer 4 blogs op een dag
* Creëer 5 blogs op een dag
* Creëer 1 blog met een .jpg foto in de blog
* Creëer 1 blog met 299 woorden
* Creëer 1 blog met 300 woorden

Uitwerking negatieve testgevallen

Onderstaand een aantal negatieve testgevallen die gebruikt kunnen worden. Het gaat hier dus om gevallen die een niet valide input hebben en dus foutmeldingen moeten geven.

* Creëer een password met alleen letters
* Creëer een password met alleen cijfers
* Creëer 6 blog op een (1) dag
* Creëer een blog met daarin een .png, tif en andere niet .jpg bestand types
* Creëer een blog met daarin 301 woorden
* Creëer een blog met daarin meer dan 301 woorden

Positief en negatief testen kan voor een groot deel gedaan worden door de ontwikkelaars tijdens unittesten. Dat is veel efficiënter dan het testen in een later stadium. Wel is het van belang dat ontwikkelaars zich er van bewust moeten worden dat ze ook negatief moeten testen.

## **Voorbeeld van TestCase**

|  |  |
| --- | --- |
| **Project Name:** Banking System | |
|  | |
| **Test Case** | |
| **Test Case ID:** **BU\_001** | **Test Designed by:** <Name> |
| **Test Priority (Low/Medium/High):** Med | **Test Designed date:** <Date> |
| **Module Name:** Bank login screen | **Test Executed by:** <Name> |
| **Test Title:** Test the Login Functionality in Banking | **Test Execution date:** <Date> |
| **Description:** Verify login with valid username and password |  |

|  |
| --- |
| **Pre-conditions:** User has valid username and password  **Dependencies:** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Navigate to login page |  | User should be able to login | User should is able to login | Pass |  |
| 2 | Provide valid username | User= [example@gmail.com](mailto:example@gmail.com) | Credential can be entered | As Expected | Pass |  |
| 3 | Provide valid password | Password: 1234 | Credential can be entered | As Expected | Pass |  |
| 4 | Click on Login button |  | User logged | User logged successfully | Pass |  |

|  |
| --- |
| **Post-conditions:** User is validated with database and successfully login to the account. The account session details are logged in the database. |

**Bron**: <https://geteasyqa.com/qa/best-test-case-templates-examples/>

## **Sjablonen voor opstellen TestCase**

## Online kan je meerdere sjablonen vinden voor het opstellen van een testcase: bv

## <https://www.softwaretestinghelp.com/test-case-template-examples/>

