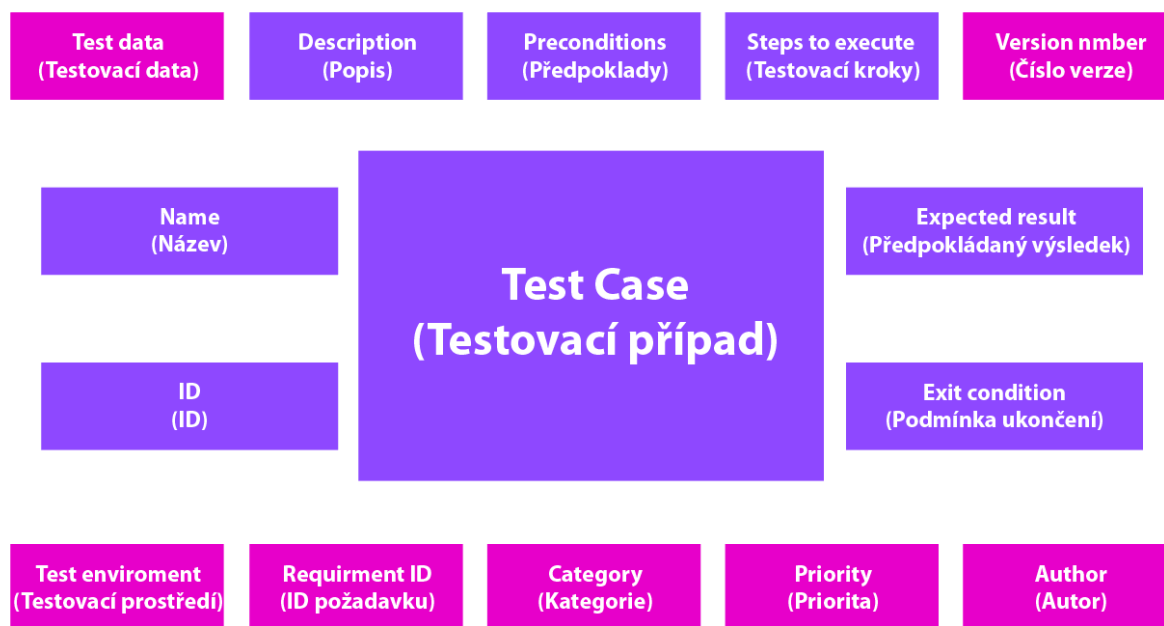


Vytvoření testovacího scénáře definice testovacích dat a analýza výsledku

- **Vytvoření testovacího scénáře**
 - Nástroje pro tvorbu scénářů
 - Krok za krokem proces definice testovacího scénáře
 - Stanovení cílů testu: Co přesně chceme otestovat?
 - Identifikace požadavků a podmínek pro testování
 - Příklady jednoduchých a komplexních testovacích scénářů
- **Co jsou testovací data a jaká je jejich role v testování**
 - Zdroje testovacích dat
 - Pozitivní vs negativní testovací případy (Typy testovacích dat)
 - Rozdělení intervalu
- **Příklady testování**
 - Login page
 - Registration page
 - Kalkulačka
 - E-commerce
- **Jak provést analýzu výsledků testů**
 - Jak interpretovat chyby a odchylky od očekávaných výsledků
 - Dokumentace a reportování výsledků
 - Doporučení pro optimalizaci testovacích scénářů a zlepšení kvality dat
- **Blokový diagram**
- **Testování pomocí rozhodovací tabulky**
- **Use case testování**
- **Testování Přechodových Stavů**
- **Nejčastější chyby a jak se jim vyhnout**
- **Vývoj a testování založené na metodách: TDD, BDD, KDD**
- **Odkazy**

Vytvoření testovacího scénáře je klíčovým procesem v testování softwaru, který pomáhá definovat a strukturovat testy, které bude třeba provést k ověření správnosti a funkčnosti systému. Testovací scénáře jsou nástroje, které usnadňují testování, usměrňují testery a poskytují jasnou strukturu pro provádění testů.



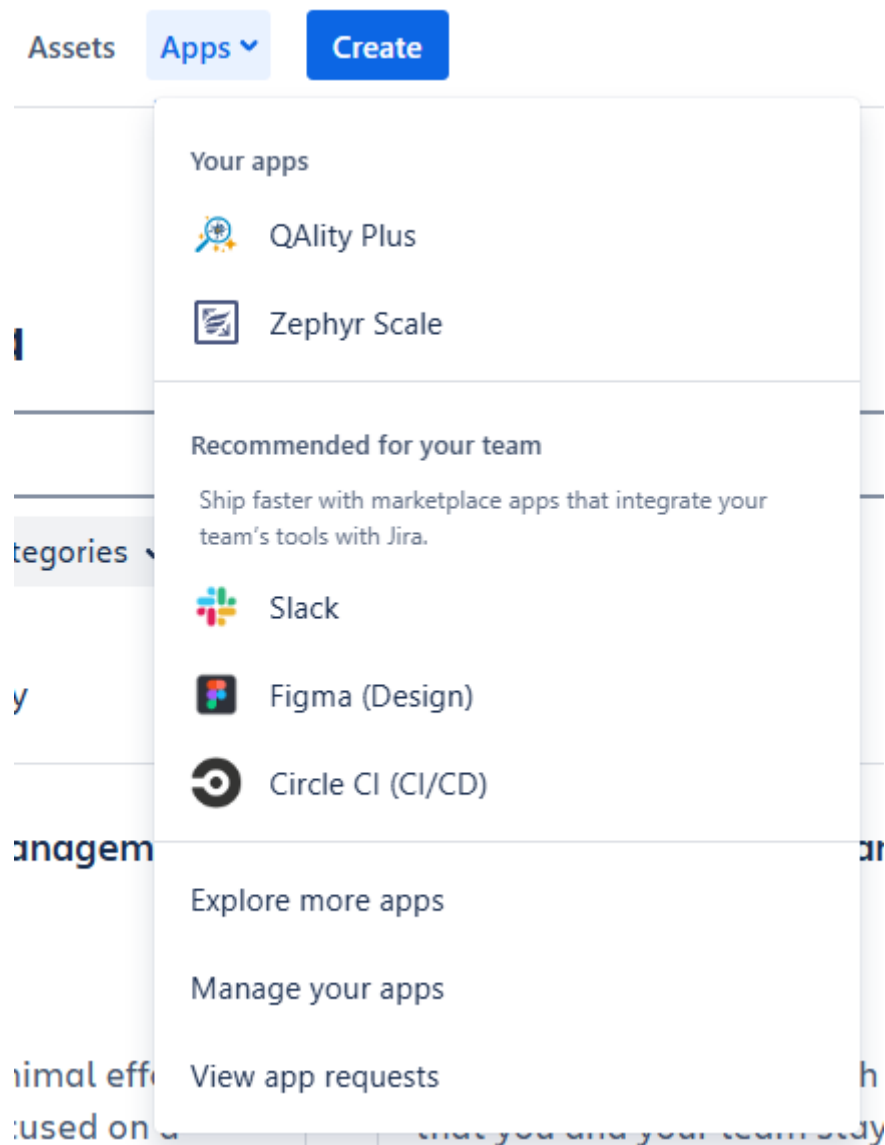
Excel

[illegible]

QAlity – Test management for JIRA

Postup přidání QAlity do JIRA:

- Klikni nahoře na **Apps** a vyber možnost **Manage your apps**:



- Vyhledej **QAlity**, pokud chceš free verzi tak nevybírej QAlity Plus

SEARCH RESULTS FOR JIRA

Q quality

X

Pricing ▾


Trust signals ▾

Categories ▾


Use cases ▾

More filters ▾


Showing 69 matches for your query




QALity Plus - Test Management for Jira


by SolDevelo 

Plan and execute tests with minimal effort so that you and your team stay focused on a common goal


3.7/4  (30)

 CLOUD FORTIFIED


ADDED




QALity - Test Management for Jira

by SolDevelo 


Plan and execute tests with minimal effort so that you and your team stay focused on a common goal

3/4  (22)


 CLOUD SECURITY PARTICIPANT

↓ 3.1k

- Klikni na **Get it now**




QALity - Test Management for Jira

by SolDevelo 

Get it now

Free


OVERALL RATINGS

3/4  (22)

INSTALLS

↓ 3 068

TRUST SIGNALS


 CLOUD SECURITY PARTICIPANT

Overview

Privacy & Security

Support

Add to Jira



QALity - Test Management for Jira

by SolDevelo

3/4 ★★★★★ 22 3 068 installs CLOUD SECURITY PARTICIPANT FREE

QALity - Test Management for Jira will perform the following actions:

- Act on a user's behalf, even when the user is offline
- Administer the host application
- Administer Jira projects
- Delete data from the host application

[Expand all details](#)

By installing this app, you:

- permit Atlassian to share anonymized data with QALity - Test Management for Jira
- agree to Atlassian Marketplace's [terms of use](#)
- agree to SolDevelo's [terms of use](#) and [privacy policy](#)

[App Info](#) [Get it now](#) [Cancel](#)


- Vytvoř nový Task (úkol) a v něm klikni na záložku **Apps**. Vyber možnost QALity – Test Management


Muj první test

[+ Add](#) [Apps](#)

Description

Add a description

 QALity - Test Management

 QALity P... QALity it Management

[+ Add apps](#)

- Při prvním spuštění bude nutno ještě přidat nový typ úkolu. Ten najdeš v JIRA při kliku na **Project settings**

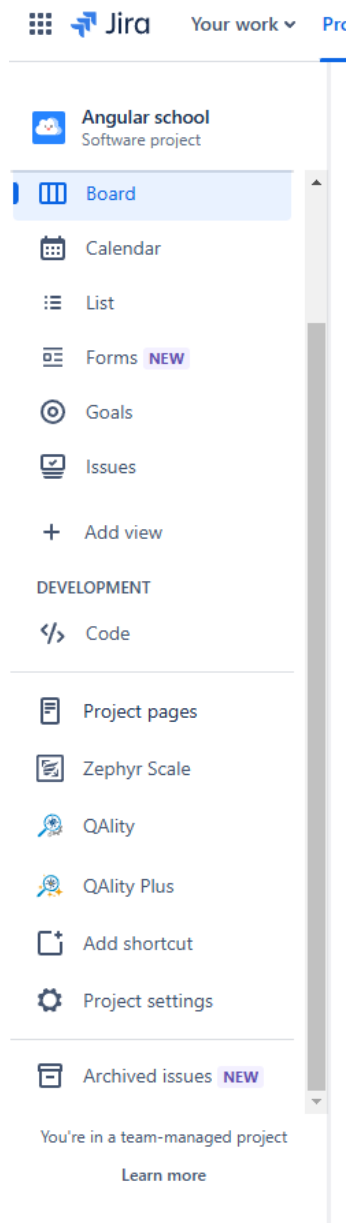
Welcome to QAlity - Test Management for Jira

To start using QAlity - Test Management for Jira, a Project Administrator needs to add a new issue type to the project configuration.

We detected that you don't have issue type needed to use this app.
Add it in issue types settings for this project and then refresh page.

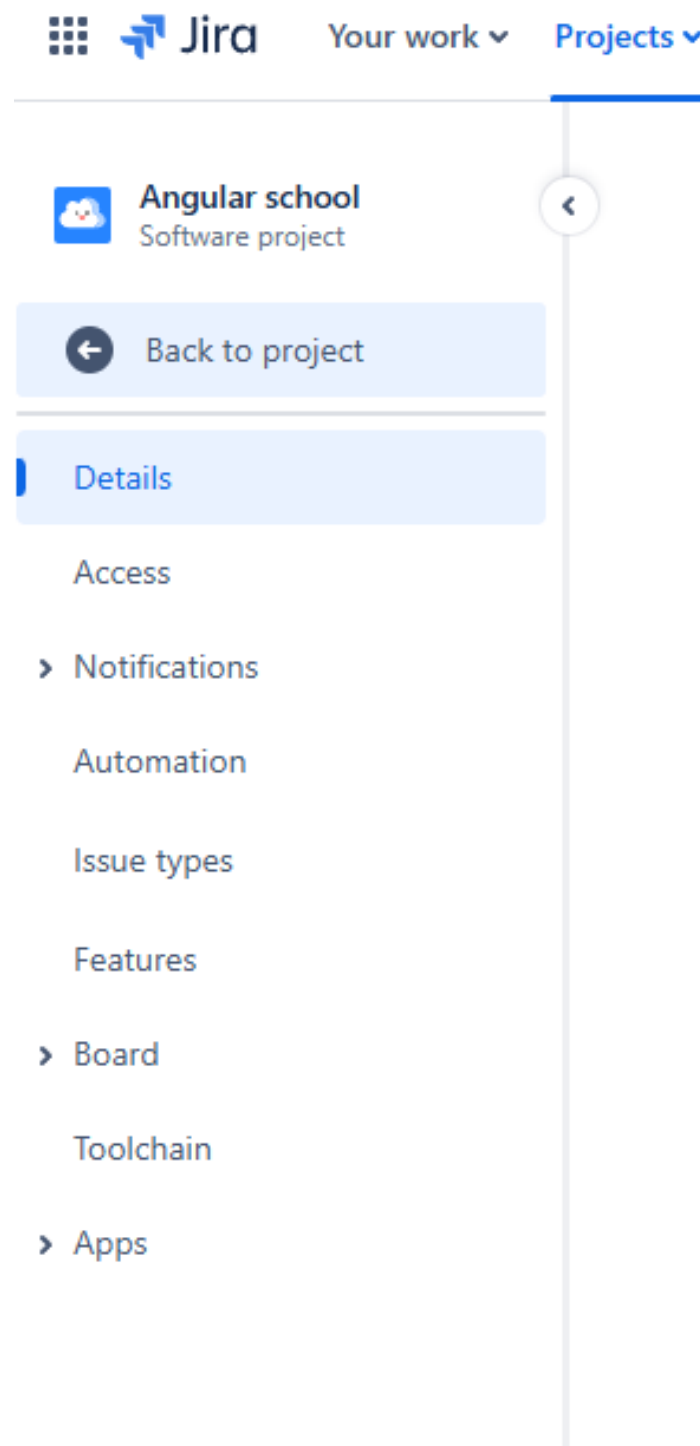
 QAlity Test

[Go to configuration](#)

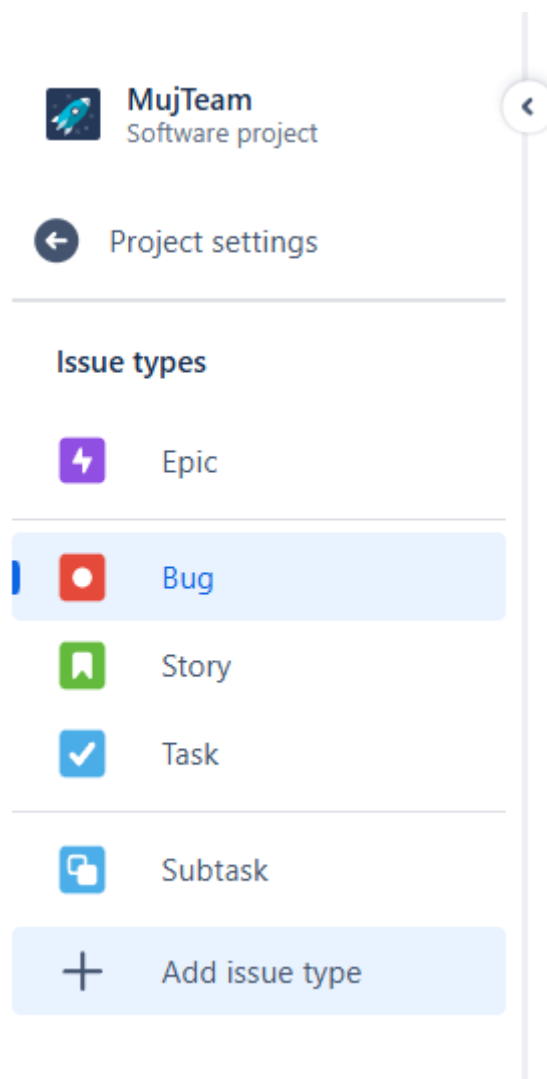


The screenshot shows the Jira project sidebar for 'Angular school' (Software project). The sidebar is divided into several sections. The top section contains navigation links: 'Board' (selected), 'Calendar', 'List', 'Forms' (marked NEW), 'Goals', 'Issues', 'Add view', 'DEVELOPMENT', and 'Code'. The middle section contains links for 'Project pages', 'Zephyr Scale', 'QAlity', 'QAlity Plus', 'Add shortcut', and 'Project settings'. The bottom section contains a link for 'Archived issues' (marked NEW). At the very bottom, it states 'You're in a team-managed project' with a 'Learn more' link.

- Poté klikni na **Issue types**



- Klikni na **Add issue type**



- Do kolonky name zadej **QAility Test**. Název musí být přesně, včetně malých a velkých písmen


Create issue type

Name *

Description

Let people know when to use this issue type

Icon



Change icon

Create
Cancel

Vrať se zpět na task (úkol) a klikni na záložku **Apps**. Vyber možnost QAlity – Test Managment. Test case by se v tuto chvíli měl přidat


Linked issues
 +





is tested by

 MUJ-2
 [Muj prvni test test case](#)
TO DO
x


QAlity - Test Management
 ...

Add Test Case
Detail view
Collapse All Test Cases
Share feedback






 MUJ-2 Valid user name and password
 ...

TEST STEP	TEST DATA	EXPECTED RESULT	TEST ATTACHMENTS
1 Enter valid username	email@gmail.com		
2 Enter valid password	Password001		
3 Click on login button		Successful login	
<input type="text"/>	<input type="text"/>	<input type="text"/>	 or use drag and drop ✓ x

Add new Test Step

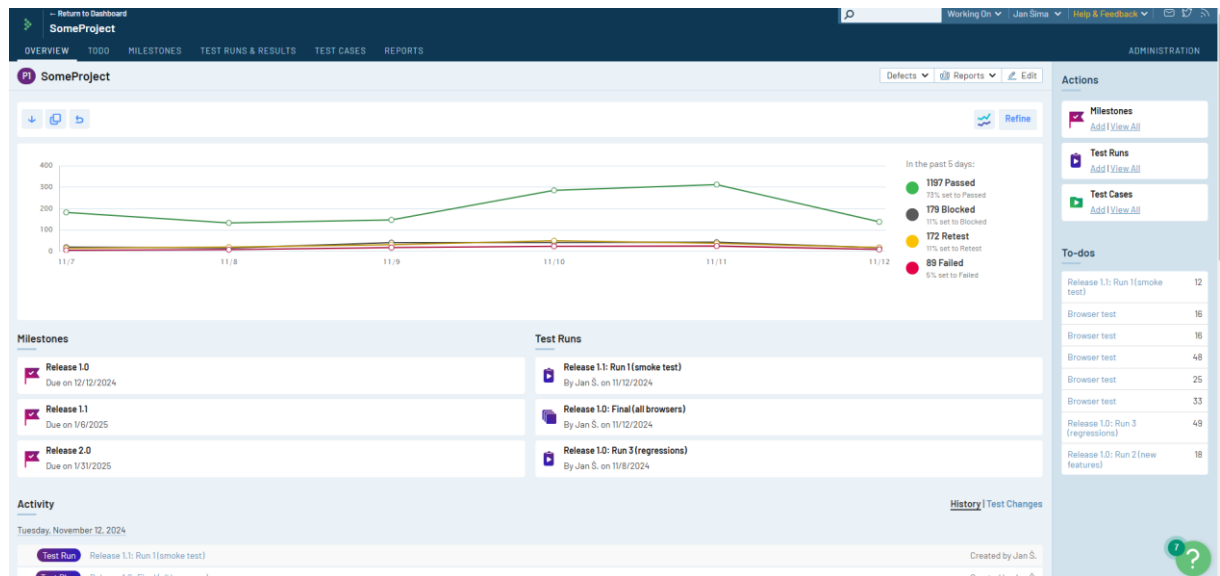
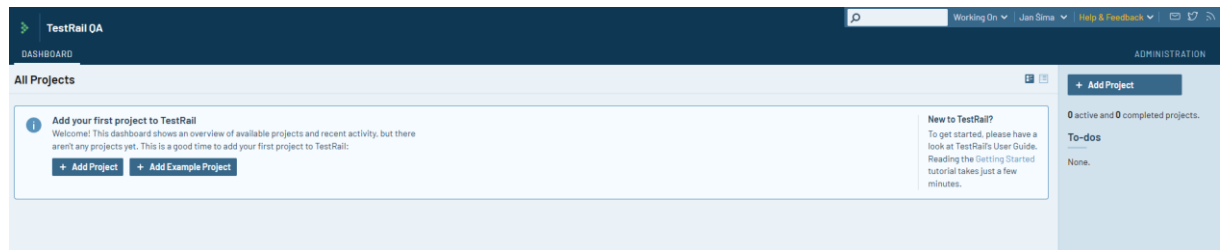


Add a comment...

 Looks good!
  Need help?
  This is blocked...
  Can you clarify...?
  This is on track

Pro tip: press **M** to comment

<https://www.testrail.com>



C2255

SomeCase

Defects

Title *

SomeCase

Section *

Prerequisites

Template *

Test Case (Text)

Type *

Other

Priority *

Medium

Assigned To

None

Estimate

References

Automation Type

None

Preconditions

The preconditions of this test case. Reference other test cases with [C#][e.g. [C17]].

Steps

The required steps to execute the test case.

Expected Result

The expected result after executing the test case.

Krok za krokem proces definice testovacího scénáře

Definice testovacího scénáře je systematický proces, který začíná určením, co bude testováno, a končí definováním konkrétních kroků a očekávaných výsledků. Zde je krok za krokem proces pro vytvoření testovacího scénáře:

1. **Definování testovacího cíle:** Prvním krokem je stanovení, co chceme testovat. Může se jednat o funkčnost konkrétní funkce, ověření kompatibility, výkonu nebo zabezpečení. Cíl testování by měl být jasně definován, například: „Testování přihlášení uživatele do systému.“
2. **Specifikace testovaného systému:** Dalším krokem je specifikování verze aplikace nebo systému, který bude testován. To může zahrnovat specifikace operačního systému, prohlížeče, hardwaru nebo jakýchkoli dalších požadavků na prostředí.
3. **Příprava testovacích dat:** Určete, jaká testovací data budou použita. To může zahrnovat správná, neplatná nebo hraniční data. Příklad: „Používání uživatelského jména a hesla, která jsou známa, nebo testování neplatného uživatelského jména a hesla.“
4. **Kroky pro provedení testu:** Vytvořte krok za krokem seznam akcí, které je tester povinen vykonat. Příklad:
 - Otevřít stránku přihlášení.
 - Zadat správné uživatelské jméno.
 - Zadat správné heslo.
 - Kliknout na tlačítko přihlásit.
5. **Očekávané výsledky:** Pro každý krok by mělo být jasně definováno, co se má stát. Očekávaný výsledek by měl být měřitelný a specifický. Například: „Po úspěšném přihlášení by se měl uživatel dostat na hlavní stránku.“
6. **Požadavky na prostředí:** Uveďte specifikace prostředí, v němž bude test proveden, např. konkrétní verze prohlížeče, operační systém, nebo připojení k databázi.
7. **Očekávané chování:** Testování by mělo ověřit, že aplikace vykonává správně a očekávaným způsobem podle specifikací.
8. **Závěrečné kroky a vyhodnocení výsledků:** Po provedení testu je třeba zhodnotit, zda výsledky odpovídají očekáváním. Pokud test selhal, je nutné identifikovat příčiny problému.

Stanovení cílů testu: Co přesně chceme otestovat?

Stanovení jasných cílů testu je zásadní pro efektivní testování. Cíle testu by měly být specifické, měřitelné, dosažitelné, relevantní a časově ohraničené. Cílem testu je odpovědět na otázku: „Co přesně chceme otestovat a jak to budeme měřit?“

Příklad:

- **Funkčnost:** „Ověřit, že uživatel může úspěšně přihlásit pomocí správných přihlašovacích údajů.“
- **Výkon:** „Otestovat, jak rychle se systém přihlásí při 1000 uživatelských požadavcích.“
- **Kompatibilita:** „Otestovat, zda aplikace běží bez problémů na mobilních zařízeních s Androidem verze 10.“

Pro jasnou definici cílů je důležité:

- **Identifikovat klíčové oblasti testování:** Jaké funkce nebo vlastnosti systému chcete testovat? (např. přihlášení, registrace, rozhraní)
- **Stanovit měřitelné ukazatele:** Co znamená „úspěšné testování“? Například rychlost načítání, bezchybné chování aplikace nebo správné zobrazení dat.
- **Určit výsledek testu:** Co znamená, že test byl úspěšný? Jaké budou metriky nebo chování systému, které nám ukáže, že test je úspěšný?

Identifikace požadavků a podmínek pro testování

Před provedením testu je důležité identifikovat všechny požadavky a podmínky, které jsou nezbytné pro testování. Tyto požadavky mohou být technické, environmentální nebo specifikované uživatelskými případy.

- **Technické požadavky:** Například operační systém, prohlížeč, verze softwaru nebo hardware.
- **Testovací prostředí:** To zahrnuje konfiguraci systémů, testovací nástroje, software nebo připojení k databázi.
- **Testovací data:** Stanovení, jaká data jsou potřeba pro testování – například správná nebo neplatná data.
- **Požadavky na přístup:** Například přístupové práva pro testování aplikace nebo přihlášení administrátora.

Příklady jednoduchých a komplexních testovacích scénářů

- **Jednoduchý testovací scénář:** Tento scénář může otestovat jednu základní funkci systému nebo aplikace, jako je přihlášení.

- **Příklad:**

- Cíl: Ověřit, že uživatel se může přihlásit pomocí správného uživatelského jména a hesla.
- Krok 1: Otevřít stránku přihlášení.
- Krok 2: Zadání platného uživatelského jména a hesla.
- Krok 3: Kliknout na tlačítko „Přihlásit“.
- Očekávaný výsledek: Uživatel je přihlášen a přesměrován na hlavní stránku.

- **Komplexní testovací scénář:** Tento scénář může zahrnovat více funkcí a testovat různé kombinace podmínek nebo vstupních dat. Komplexní testy mohou zahrnovat více kroků, závislostí a různých scénářů chování.

- **Příklad:**

- Cíl: Ověřit, že uživatel se může přihlásit a poté provést změnu hesla.
- Krok 1: Otevřít stránku přihlášení.
- Krok 2: Zadání platného uživatelského jména a hesla.
- Krok 3: Kliknout na tlačítko „Přihlásit“.
- Krok 4: Po úspěšném přihlášení kliknout na „Změnit heslo“.
- Krok 5: Zadání nového hesla.
- Krok 6: Kliknutí na tlačítko „Uložit změny“.
- Očekávaný výsledek: Heslo je úspěšně změněno a uživatel je informován o úspěšné změně hesla.

Co jsou testovací data a jaká je jejich role v testování

Testovací data jsou konkrétní vstupy, které se používají během testování softwaru, aby ověřily správnost, funkčnost a stabilitu aplikace. Testovací data simulují různé scénáře, které může uživatel nebo systém vyvolat během běhu aplikace. Mohou to být například hodnoty zadané do formulářů, údaje o uživatelských účtech nebo různé formáty souborů, které aplikace zpracovává.

Role testovacích dat:

- **Ověření funkcionality:** Testovací data umožňují ověřit, že aplikace reaguje správně na specifické vstupy. Například při testování formulářů lze ověřit, že systém správně validuje zadané údaje (jako je e-mailová adresa nebo telefonní číslo).
- **Zjištění chyb:** Slouží k odhalení problémů a chyb v aplikaci. Například při zadání neplatných dat může aplikace vykázat chybu, která ukáže na problém ve validaci nebo v logice.
- **Pokrytí různých scénářů:** Testovací data umožňují simulovat různé situace, včetně těch nejběžnějších i méně častých, které by mohly být přehlédnuty během vývoje.

Jak vytvořit efektivní a reprezentativní testovací data?

- **Zohlednění realistických scénářů:** Testovací data by měla odrážet skutečné chování uživatelů a situace, které se v praxi mohou vyskytnout. Měly by pokrývat jak běžné, tak i méně časté scénáře.
- **Pokrytí všech možností:** Je důležité zahrnout různé varianty vstupů, aby bylo pokryto co nejvíce testovacích případů. Například při testování zadávání data narození je nutné ověřit různé formáty (DD/MM/YYYY, MM-DD-YYYY) a validní/invalidní data.
- **Zahrnutí okrajových hodnot:** Hraniční hodnoty jsou často nejcitlivější na chyby, proto by měly být vždy zahrnuty ve testovacích datech. Při testování věkového rozsahu 18–60 let by měla být testována hodnota 18 a 60.
- **Zajištění pokrytí všech typů validací:** Měly by být zahrnuty různé kombinace platných a neplatných vstupů pro různé typy validací, jako jsou e-mailová pole, telefonní čísla nebo hesla s různými pravidly (minimální délka, speciální znaky, apod).

Zdroje testovacích dat

Existuje několik způsobů, jak získat testovací data pro konkrétní aplikace:

- **Generátory testovacích dat:** Generátory testovacích dat jsou nástroje nebo skripty, které automaticky vytvářejí testovací data na základě definovaných parametrů. Mohou generovat náhodné nebo specifikované hodnoty pro různé formáty (např. e-mailové adresy, telefonní čísla, uživatelská jména). Příklady nástrojů:
 - [Mockaroo](#): Online generátor dat pro různá pole (jména, adresy, telefonní čísla).
 - [Faker](#): Knihovna pro generování falešných dat v různých programovacích jazycích (Python, Ruby). Pro pokročilé testery, je třeba znát základy programování
- **Skutečná data:** Použití skutečných dat je v některých případech nejlepší cestou, jak simulovat reálné scénáře. Nicméně, při použití skutečných dat je třeba dbát na ochranu osobních údajů (GDPR) a anonymizaci citlivých údajů.
- **Manuální vytvoření testovacích dat:** Pokud jsou specifické požadavky nebo testovací scénáře jedinečné, může být nutné vytvořit testovací data ručně. To může zahrnovat zadání konkrétních hodnot do formulářů a jejich použití v testech.
- **Vytváření dat na základě uživatelských příběhů:** Pokud jsou testovací scénáře součástí širšího testování aplikace na základě uživatelských příběhů (User Stories), testovací data mohou být generována přímo z těchto příběhů. To znamená, že data budou odpovídat konkrétnímu chování, které se očekává v dané situaci.

Pozitivní vs negativní testovací případy (Typy testovacích dat)

Pozitivní testovací případy: Pozitivní testovací případy jsou navrženy tak, aby ověřily, že aplikace funguje podle očekávání, když je zadán platný vstup. Testují základní funkce softwaru, aby zajistily, že při běžných okolnostech je produkován správný výstup.

Příklad pozitivního testovacího případu:

Testovací případ 1: Pozitivní test přihlášení

- Zadejte uživatelské jméno **student** do pole pro uživatelské jméno
- Zadejte heslo **Password123** do pole pro heslo
- Stiskněte tlačítko **Odeslat**
- Ověřte, že došlo k přihlášení uživatele **student**
- Ověřte, že nová stránka obsahuje očekávaný text (např. „Gratulujeme“ nebo „úspěšně přihlášeno“)
- Ověřte, že tlačítko **Odhlásit se** je zobrazeno na nové stránce

Negativní testovací případy: Negativní testovací případy se zaměřují na ověření reakce aplikace na neplatný vstup nebo neočekávané chování uživatele. Pomáhají potvrdit, že software zvládá chyby správně a udržuje stabilitu.

Příklad negativního testovacího případu:

Testovací případ 2: Negativní test uživatelského jména

- Otevřete stránku
- Zadejte uživatelské jméno **incorrectUser** do pole pro uživatelské jméno
- Zadejte heslo **Password123** do pole pro heslo
- Stiskněte tlačítko **Odeslat**
- Ověřte, že je zobrazená chybová zpráva
- Ověřte, že text chybové zprávy je **Vaše uživatelské jméno je neplatné!**

Rozdělení intervalu

Rozdělení intervalu (nebo také Equivalence Partitioning) je technika používaná v testování softwaru k rozdělení vstupního prostoru na různé části (intervaly), které mohou obsahovat podobné hodnoty. Tento přístup pomáhá snížit počet testů tím, že se zaměřuje na reprezentativní hodnoty z každého intervalu, místo testování každé možné hodnoty jednotlivě.

Princip:

Technika rozdělení intervalu předpokládá, že vstupy, které spadají do stejného intervalu, budou mít podobné chování. To znamená, že pokud je program správně otestován na jedné hodnotě z intervalu, bude pravděpodobně fungovat i pro ostatní hodnoty v tomtéž intervalu. Rozdělení intervalu se obvykle dělí na tři typy:

- **Platné hodnoty** (valid partitions) – Vstupy, které by měly být považovány za správné a měly by vést k očekávanému chování.
- **Neplatné hodnoty** (invalid partitions) – Vstupy, které by měly způsobit chybu nebo neplatný výstup.
- **Hraniční hodnoty** (boundary values) – Specifické hodnoty na hranicích intervalu, které bývají zvlášť citlivé na chyby.

Příklad 1:

Pokud máme pole, kde uživatel zadává věk, který by měl být v rozmezí od 18 do 60 let, můžeme rozdělit vstupy do těchto intervalů:

- Platné hodnoty:
 - Věk mezi 18 a 60 (například 25, 35, 50)
- Neplatné hodnoty:
 - Věk menší než 18 (například 10, 15)
 - Věk větší než 60 (například 65, 70)
- Hraniční hodnoty:
 - Věk přesně 18 (dolní hranice)
 - Věk přesně 60 (horní hranice)

Příklad 2:

Systém určuje cenu doručení dle hmotnosti, v dokumentaci jsou následující informace:

- Balík do 5 kg stojí EUR 10 (A)
- Balík do 10 kg stojí EUR 20 (B)
- Balík nad 10 kg stojí EUR 30 (C)

Doručovací společnost nespecifikuje maximální hmotnost balíku. Vstupy můžeme rozdělit do těchto intervalů:

- A, validní hodnoty K1A: {1kg, 2kg, 5kg}, nevalidní hodnoty K2A: {6kg, 7kg, 10kg, 30kg}
- B, validní hodnoty K1B: {6kg, 8kg, 10kg}, nevalidní hodnoty K2B: {1kg, 2kg, 5kg}, K3B: {11kg, 20kg, 30kg}
- C, validní hodnoty K1C: {11kg, 20kg, 1000kg}, nevalidní hodnoty K2C: {4kg, 7kg, 10kg}

Pozitivní případy

- P1, za 5 kg balík, cena doručení bude EUR 10.
- P2, za 8 kg balík, cena doručení bude EUR 20.
- P3, za 20 kg balík, cena doručení bude EUR 30.

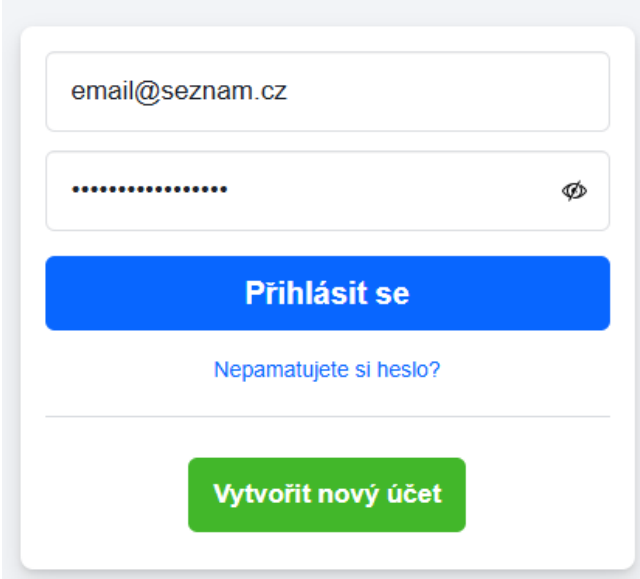
Negativní případy

- P1N, za 6 kg balík, cena doručení nebude EUR 10.
- P2N, za 11 kg balík, cena doručení nebude EUR 20.
- P3N, za 10 kg balík, cena doručení nebude EUR 30.

Login page

Obvykle píšeme testovací případy pro přihlašovací stránku pro každou aplikaci, kterou testujeme. Každá přihlašovací stránka by měla obsahovat následující prvky:

- Textové pole „E-mail/Telefonní číslo/Uživatelské jméno“
- Textové pole „Heslo“
- Tlačítko „Přihlásit se“
- Zaškrťovací políčko „Zapamatovat si mě“
- Zaškrťovací políčko „Zůstat přihlášený“
- Odkaz „Zapomenuté heslo“
- Odkaz „Registrace/Vytvořit účet“
- CAPTCHA



The image shows a login form with the following elements:

- A text input field containing the email address "email@seznam.cz".
- A password input field represented by a series of dots, with an eye icon to its right for toggling visibility.
- A prominent blue button labeled "Přihlásit se" (Login).
- A blue text link below the login button that reads "Nepamatujete si heslo?" (Forgot your password?).
- A green button at the bottom labeled "Vytvořit nový účet" (Create new account).

Příklady funkčních scénářů pro Login page

- Ověřte, že kurzor je po načtení stránky (přihlašovací stránky) zaostřen na textovém poli „Uživatelské jméno“.
- Ověřte, zda funkce tabulátoru funguje správně.
- Ověřte, že klávesy Enter/Tab fungují jako náhrada za tlačítko Přihlásit se.
- **Ověřte, že se uživatel může přihlásit s platnými přihlašovacími údaji.**
- **Ověřte, že se uživatel nemůže přihlásit s neplatným uživatelským jménem a neplatným heslem.**

- **Ověřte, že se uživatel nemůže přihlásit s platným uživatelským jménem a neplatným heslem.**
- **Ověřte, že se uživatel nemůže přihlásit s neplatným uživatelským jménem a platným heslem.**
- **Ověřte, že se uživatel nemůže přihlásit, pokud je uživatelské jméno nebo heslo prázdné.**
- Ověřte, že se uživatel nemůže přihlásit s neaktivními přihlašovacími údaji.
- Ověřte, že tlačítko reset vymaže data ze všech textových polí v přihlašovacím formuláři.
- Ověřte, že přihlašovací údaje, zejména heslo, jsou uloženy v databázi v šifrovaném formátu.
- Ověřte, že kliknutí na tlačítko Zpět v prohlížeči po úspěšném přihlášení nepřepne uživatele do odhlášeného režimu.
- Ověřte, že při opuštění pole Uživatelské jméno nebo Heslo prázdné se zobrazí validační zpráva.
- Ověřte, že při překročení limitu znaků pro pole Uživatelské jméno a Heslo se zobrazí validační zpráva.
- Ověřte, že při zadání speciálního znaku do pole Uživatelské jméno nebo Heslo se zobrazí validační zpráva.
- Ověřte, že je zaškrtačací políčko „Zůstat přihlášen“ ve výchozím stavu nezvolené (závisí na požadavcích, může být zvolené nebo nezvolené).
- Ověřte časový limit přihlašovací relace (Session Timeout).
- Ověřte, že odkaz odhlášení přesměruje na přihlašovací/úvodní stránku.
- Ověřte, že uživatel je po úspěšném přihlášení přesměrován na odpovídající stránku.
- Ověřte, že uživatel je po kliknutí na odkaz Zapomenuté heslo přesměrován na stránku Zapomenuté heslo.
- Ověřte, že uživatel je po kliknutí na odkaz Registrace/Vytvořit účet přesměrován na stránku pro vytvoření účtu.
- Ověřte, že uživatel by se měl být schopen přihlásit s novým heslem po jeho změně.
- Ověřte, že uživatel by se neměl být schopen přihlásit se starým heslem po jeho změně.

- Ověřte, že před žádným znakem hesla nejsou povoleny mezery.
- Ověřte, zda je uživatel stále přihlášen po sérii akcí jako přihlášení, zavření prohlížeče a opětovné otevření aplikace.
- Ověřte, že existují způsoby, jak získat heslo, pokud jej uživatel zapomene.

Příklady nefunkčních scénářů pro Login page

- Ověřte, že kliknutí na tlačítko zpět v prohlížeči po úspěšném odhlášení by nemělo uživatele přenést zpět do přihlášeného stavu.
- Ověřte, že existuje limit na celkový počet neúspěšných pokusů o přihlášení (Počet neplatných pokusů by měl být stanoven podle obchodní logiky. Na základě obchodní logiky bude uživatel vyzván k zadání CAPTCHA nebo bude zablokován).
- Ověřte, že heslo je zadáno ve šifrované podobě (maskovaný formát) v poli pro heslo.
- Ověřte, že heslo lze zkopírovat a vložit. Systém by neměl umožnit uživatelům kopírovat a vkládat heslo.
- Ověřte, že šifrované znaky v poli „Heslo“ by neměly být dešifrovány, pokud jsou zkopírovány.
- Ověřte, že zaškrťovací políčko „Pamatuj si heslo“ není ve výchozím nastavení zaškrtnuto (závisí na obchodní logice, může být zaškrtnuto nebo nezaškrtnuto).
- Ověřte, zda přihlašovací formulář neodhaluje žádné bezpečnostní informace při zobrazení zdroje stránky.

Registration page

Obvykle píšeme testovací případy pro registrační formulář / formulář pro registraci / registrační stránku pro každou aplikaci, kterou testujeme. Zde jsou pole, která se obvykle používají v registračním formuláři:

- Uživatelské jméno
- Jméno
- Příjmení
- Heslo
- Potvrzení hesla
- E-mailová adresa
- Telefonní číslo
- Datum narození
- Pohlaví
- Podmínky užívání
- Tlačítko odeslat
- Přihlásit se (Pokud již máte účet)

Vytvořte si nový účet

Je to rychlé a snadné.

Datum narození ?

11

lis

2024

Pohlaví ?

Žena

Muž

Vlastní

Lidé, kteří používají naše služby, možná nahráli vaše kontaktní údaje na Facebook. [Další informace.](#)

Klepnutím na Zaregistrovat se vyjádříte svůj souhlas s našimi [smluvními podmínkami](#). Přečtěte si v našich [zásadách ochrany osobních údajů](#), jak sbíráme, používáme a sdílíme vaše údaje. V [zásadách používání souborů cookie](#) se zase dozvíte, jak používáme soubory cookie a podobnou technologii. Můžete od nás dostávat SMS upozornění. Jejich odběr můžete kdykoli zrušit.

Zaregistrovat se

[Už máte účet?](#)

Příklady funkčních scénářů pro Registration page

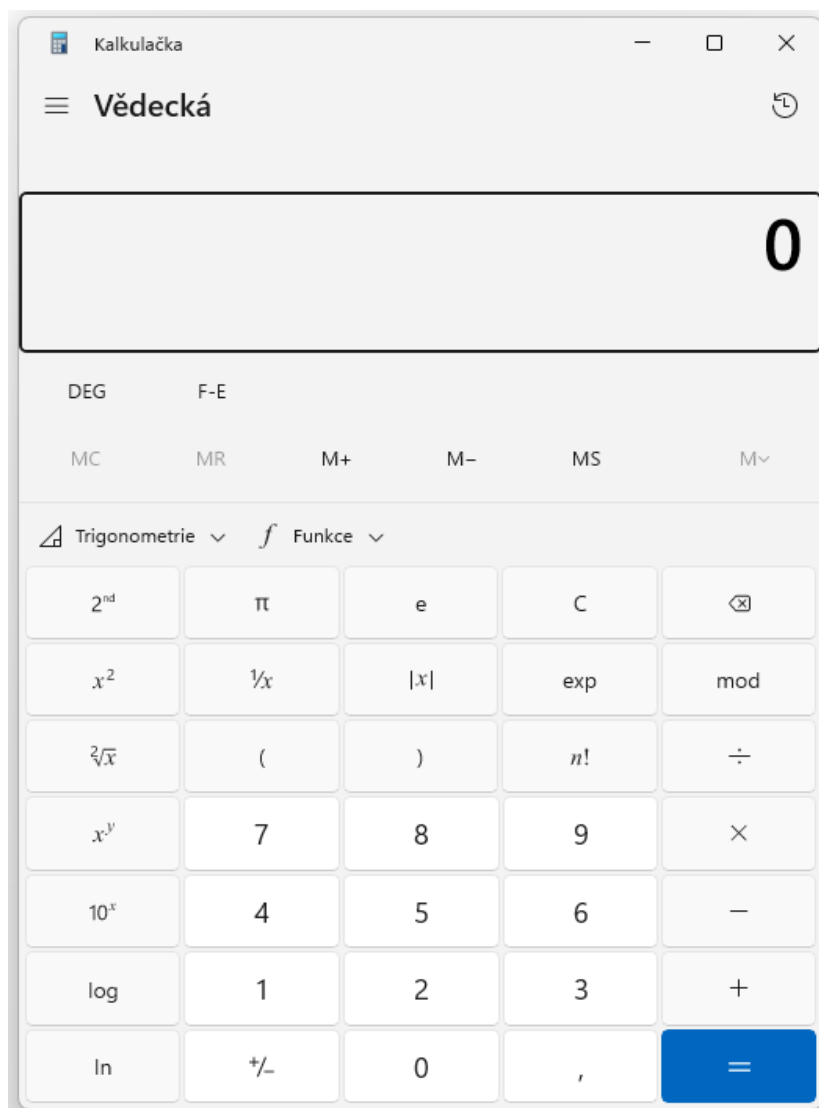
- Ověřte, že registrační formulář obsahuje pole Uživatelské jméno, Jméno, Příjmení, Heslo, Potvrzení hesla, E-mailová adresa, Telefonní číslo, Datum narození, Pohlaví, Místo, Podmínky užívání, Odeslat, Přihlášení (Pokud již máte účet).
- Ověřte, zda funkce tabulátoru funguje správně.
- Ověřte, že klávesy Enter/Tab fungují jako náhrada za tlačítko Odeslat.
- Ověřte, že všechna pole jako Uživatelské jméno, Jméno, Příjmení, Heslo a další pole mají platný zástupný text.
- Ověřte, že se štítky přesunou nahoru, když je textové pole zaostřeno nebo vyplněno (v případě plovoucího štítku).
- Ověřte, že všechna povinná pole jsou označena hvězdičkou (*).
- Ověřte, že po zadání všech povinných polí a kliknutí na tlačítko odeslat se data odešlou na server.

- Ověřte, že systém vygeneruje validační zprávu, když kliknete na tlačítko odeslat, aniž byste vyplnili všechna povinná pole.
- Ověřte, že zadání prázdných mezer v povinných polích způsobí validační chybu.
- Ověřte, že kliknutím na tlačítko odeslat po opuštění volitelných polí se data odešlou na server bez validační chyby.
- Ověřte, že uživatelské jméno nerozlišuje malá a velká písmena (obvykle pole Uživatelské jméno nerozlišuje velká a malá písmena – např. „jan“ a „JAN“ fungují stejně).
- Ověřte, že systém vygeneruje validační zprávu při zadání existujícího uživatelského jména.
- Ověřte limit znaků ve všech polích (zejména v uživatelském jménu a heslu) podle obchodních požadavků.
- Ověřte validaci uživatelského jména podle obchodních požadavků (v některých aplikacích nesmí uživatelské jméno obsahovat číslice ani speciální znaky).
- Ověřte, že validace všech polí odpovídá obchodním požadavkům.
- Ověřte, že pole datum narození neumožňuje zadat datum větší než aktuální datum (některé aplikace mají věkový limit 18 let, v tomto případě je nutné ověřit, zda je věk větší nebo roven 18 let).
- Ověřte validaci e-mailového pole zadáním nesprávné e-mailové adresy.
- Ověřte validaci číselných polí zadáním písmen a speciálních znaků.
- Ověřte, že úvodní a koncové mezery jsou odstraněny po kliknutí na tlačítko odeslat.
- Ověřte, že zaškrtačací políčko „podmínky užívání“ není ve výchozím nastavení zaškrtnuté (závisí na obchodní logice, může být zaškrtnuté nebo nezaškrtnuté).
- Ověřte, že se zobrazuje validační zpráva při kliknutí na tlačítko odeslat bez zaškrtnutí políčka „podmínky užívání“.
- Ověřte, že při zadávání hesla je heslo zobrazeno v šifrované formě.
- Ověřte, zda se heslo a potvrzení hesla shodují.

Kalkulačka

Příklad testování kalkulačky. Může být oblíbenou otázkou při pohovoru. Jako první je dobré pokládat otázky a zjistit podrobnosti o testované aplikaci a pochopit požadavky. Například:

- Je to fyzická kalkulačka?
- Je to digitální kalkulačka?
- Je to běžná kalkulačka, nebo vědecká kalkulačka?
- Je součástí aplikace, nebo je to samostatná aplikace?
- Je to programovatelná kalkulačka?



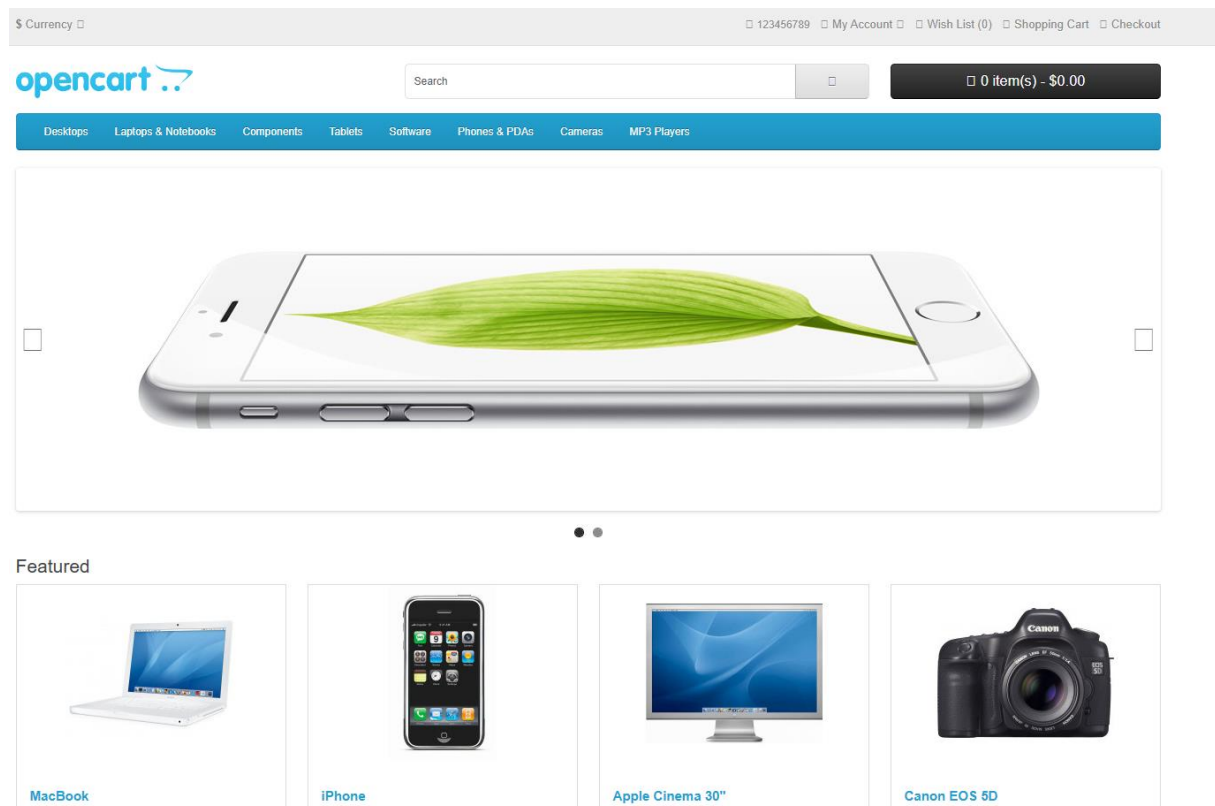
Funkční testovací případy pro kalkulačku:

- Ověřte, zda kalkulačka dokáže provést sčítání dvou celých čísel.

- Ověřte, zda kalkulačka dokáže provést sčítání dvou záporných čísel.
- Ověřte, zda kalkulačka dokáže provést odčítání dvou celých čísel.
- Ověřte, zda kalkulačka dokáže provést odčítání dvou záporných čísel.
- Ověřte, zda kalkulačka dokáže provést odčítání jednoho záporného a jednoho kladného čísla.
- Ověřte, zda kalkulačka dokáže provést násobení dvou celých čísel.
- Ověřte, zda kalkulačka dokáže provést násobení dvou záporných čísel.
- Ověřte, zda kalkulačka dokáže provést násobení jednoho záporného a jednoho kladného čísla.
- Ověřte, zda kalkulačka dokáže provést dělení dvou celých čísel.
- Ověřte, zda kalkulačka dokáže provést dělení dvou záporných čísel.
- Ověřte, zda kalkulačka dokáže provést dělení jednoho kladného a jednoho záporného čísla.
- Ověřte, zda kalkulačka umožňuje dělení nulou.
- Ověřte, zda kalkulačka používá prioritu operací.
- Ověřte, zda při stisknutí dvou operátorů za sebou nejnovější operátor přepíše předchozí operátor.
- Ověřte, zda kalkulačka vrací správný výsledek při operacích s desetinnými čísly.
- Ověřte, zda při vymazání čísel kalkulačka bere v úvahu nejnovější číslo a ne nějaké předchozí číslo s více číslicemi.
- Ověřte, zda tlačítka pro druhou mocninu a druhou odmocninu fungují podle očekávání.

E-commerce

<https://demo.opencart.com/en-gb?route=common/home>



Funkční testovací případy pro E-commerce:

Home

- Tlačítko pro registraci/přihlášení by mělo být viditelné a snadno nalezitelné.
- Odkazy na domovské stránky by měly přesměrovat na zamýšlenou stránku.
- Horní navigace, hamburger menu, kategorie a podkategorie by měly být jasně uvedeny.

Vyhledávání

- Při zadání klíčových slov, jako je název produktu, značka nebo název kategorie (například iPhone, notebook, nejlepší knihy o testování softwaru), by měly být zobrazeny relevantní produkty.
- Při vyhledávání pomocí konkrétního klíčového slova by měly být zobrazeny přesné nebo příbuzné produkty.
- Ve výsledcích vyhledávání by měly být zobrazeny název produktu, obrázek, zákaznické hodnocení a informace o ceně.

- Pokud je použita konkrétní kategorie pro vyhledávání, měly by se zobrazit výsledky z odpovídající kategorie.
- Na vrcholu seznamu by vždy měly být nejrelevantnější produkty.
- I když je produkt uveden v několika kategoriích, měl by se ve výsledcích vyhledávání zobrazit pouze jednou.
- Pokud je v zadání klíčového slova překlep, měly by se zobrazit návrhy.

Popis produktu

- Měly by být zobrazeny podrobnosti o obsahu, jako je název produktu, popis, obrázek, cenové údaje (včetně slev, pokud jsou k dispozici) a sekce Otázky a odpovědi.
- Zákaznické recenze včetně hodnocení hvězdičkami a textu recenze by měly být dostupné.
- Tlačítko „Přidat do košíku“ by mělo být snadno k nalezení.
- Ověřte funkčnost tlačítka „Přidat do košíku“.
- Měla by být dostupná sekce s podobnými produkty nebo produkty, které si zákazníci také koupili.
- Obrázky produktu by měly mít možnost přiblížení (zoom).

Nákupní košík

- Funkce pro dokončení nákupu by měla fungovat podle očekávání.
- Zákazníci by měli být schopni přidávat položky do košíku z jakékoli kategorie obchodu.
- Ověřte přidání položek do nákupního košíku.
- Ověřte odebrání produktů z nákupního košíku.
- Ověřte změnu množství položek. Detaily jako poplatky za dopravu a daně by měly být zobrazeny spolu s cenou produktu.
- Měla by existovat jasná možnost pro odstranění položky z nákupního košíku.
- Cena objednávky by se měla aktualizovat, když zákazník přidá nebo odebere položku z košíku.

Jak provést analýzu výsledků testů

Analýza výsledků testů je klíčovým krokem pro pochopení kvality systému a odhalení problémů, které je třeba vyřešit. Při provádění analýzy je důležité rozlišovat mezi úspěšnými a neúspěšnými testy a určit příčiny těchto výsledků.

- **Úspěšné testy:** Pokud testy probíhají podle očekávání, znamená to, že dané funkce nebo komponenty systému fungují správně. To může potvrdit, že specifikace systému jsou dobře implementovány a že se software chová podle očekávání v daných scénářích.
- **Neúspěšné testy:** Pokud test selže, je nutné zjistit, proč. Je důležité podívat se na chybové hlášky, logy nebo chování systému. Testování může neuspět z různých důvodů, například:
 - **Chyby v kódu:** Problémy v implementaci nebo neodpovídající funkce.
 - **Problémy s prostředím:** Například nesprávně nastavené servery, nesprávné verze softwaru nebo nedostatek systémových prostředků.
 - **Chyby v testech:** Problémy v implementaci testů. Chyby v kódu u automatizovaných testů
 - **Problémy s daty:** Nevalidní nebo špatně formátovaná testovací data.

Pro analýzu výsledků byste měli porovnat skutečné výsledky s očekávanými výsledky a podívat se na specifikace, jak byly definovány na začátku testování.

Jak interpretovat chyby a odchylky od očekávaných výsledků

Při interpretaci chyb je důležité zjistit, co přesně vedlo k odchylkám od očekávaných výsledků. Tento proces zahrnuje několik kroků:

- **Typ chyby:** Zjistěte, zda se jedná o **funkční chybu**, **chybu v uživatelském rozhraní**, **výkonovou chybu** nebo **problém s kompatibilitou**. Například funkční chyby mohou zahrnovat nesprávné výpočty nebo problémy s logikou aplikace, zatímco uživatelské rozhraní může mít chyby v designu nebo nedostatečnou responzivitu.
- **Kroky k reprodukci chyby:** Pokud je to možné, zaznamenejte kroky, které vedou k reprodukci chyby. To pomůže vývojářům přesně pochopit, kde došlo k problému.
- **Chybové zprávy:** Podívejte se na logy, výstupy chybových zpráv nebo jakýkoli debugovací výstup, který by mohl poskytnout užitečné informace pro opravu chyby. Chybové zprávy obvykle obsahují kódy chyb, popisy nebo výstupy, které pomohou určit, co se pokazilo.
- **Důsledky chyb:** Některé chyby mohou mít vážné důsledky (např. ztráta dat nebo bezpečnostní problém), jiné mohou být méně závažné (např. grafické nedostatky). Je důležité priorizovat opravy na základě závažnosti chyb.

Dokumentace a reportování výsledků

Kvalitní dokumentace a reportování testovacích výsledků jsou zásadní pro komunikaci mezi týmy a pro budoucí analýzu. Tento proces zahrnuje následující kroky:

- **Testovací protokol:** Zaznamenejte všechny testy, včetně testovacích kroků, vstupů, očekávaných výsledků a skutečných výsledků. Všechny tyto informace jsou důležité pro opakování testů nebo pro diagnostiku problémů v budoucnu.
- **Záznamy o selháních:** Při každém selhání je třeba podrobně popsat, co se stalo, jaké kroky vedly k chybě, jaké byly zobrazené chybové zprávy a jaké byly další okolnosti.
- **Přehledná zpráva:** Zpráva by měla obsahovat souhrn výsledků testů (co fungovalo a co ne), podrobný popis zjištěných problémů a doporučení pro jejich řešení. Měla by také uvádět počet provedených testů, jejich úspěšnost a časovou náročnost testování.
- **Metodika reportování:** Zprávy by měly být strukturovány v přehledném formátu, který usnadní pochopení výsledků nejen testerům, ale i dalším zúčastněným stranám (vývojářům, manažerům).

Doporučení pro optimalizaci testovacích scénářů a zlepšení kvality dat

Na základě analýzy výsledků testů lze navrhnout optimalizaci testovacích scénářů a zlepšení kvality dat:

- **Optimalizace testovacích scénářů:**

- **Pokrytí:** Pokud testování neodhalilo chyby, může to znamenat, že některé scénáře nebyly dostatečně pokryté. Zvažte přidání nových scénářů, které otestují okrajové případy nebo neobvyklé situace.
- **Automatizace testů:** Pokud jsou některé testy vykonávány opakovaně (například při každé změně verze aplikace), zvažte jejich automatizaci. To ušetří čas a sníží riziko lidské chyby.

- **Zlepšení kvality testovacích dat:**

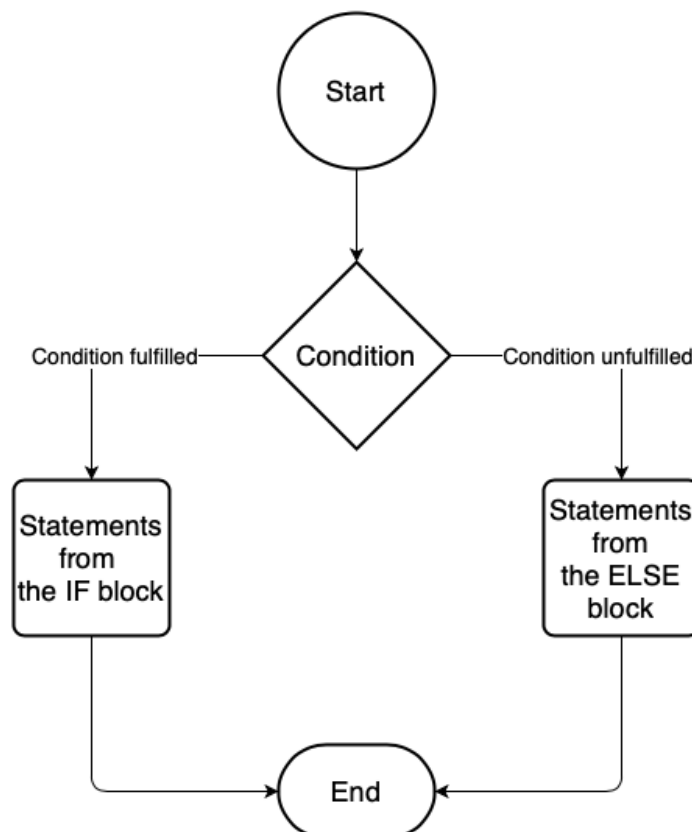
- **Různorodost dat:** Pro efektivní testování je důležité mít dostatečně reprezentativní testovací data. To znamená testování s různými kombinacemi vstupů, které odrážejí skutečné scénáře, včetně validních i nevalidních dat.
- **Generátory testovacích dat:** Využití nástrojů pro generování testovacích dat může zajistit, že testovací data budou pokrývat širokou škálu možných hodnot, včetně hraničních a extrémních případů.
- **Reálná data:** Pokud je to možné, použijte reálná data (například anonymizovaná data z produkčního systému), která zajistí realistické testy.

Blokový diagram

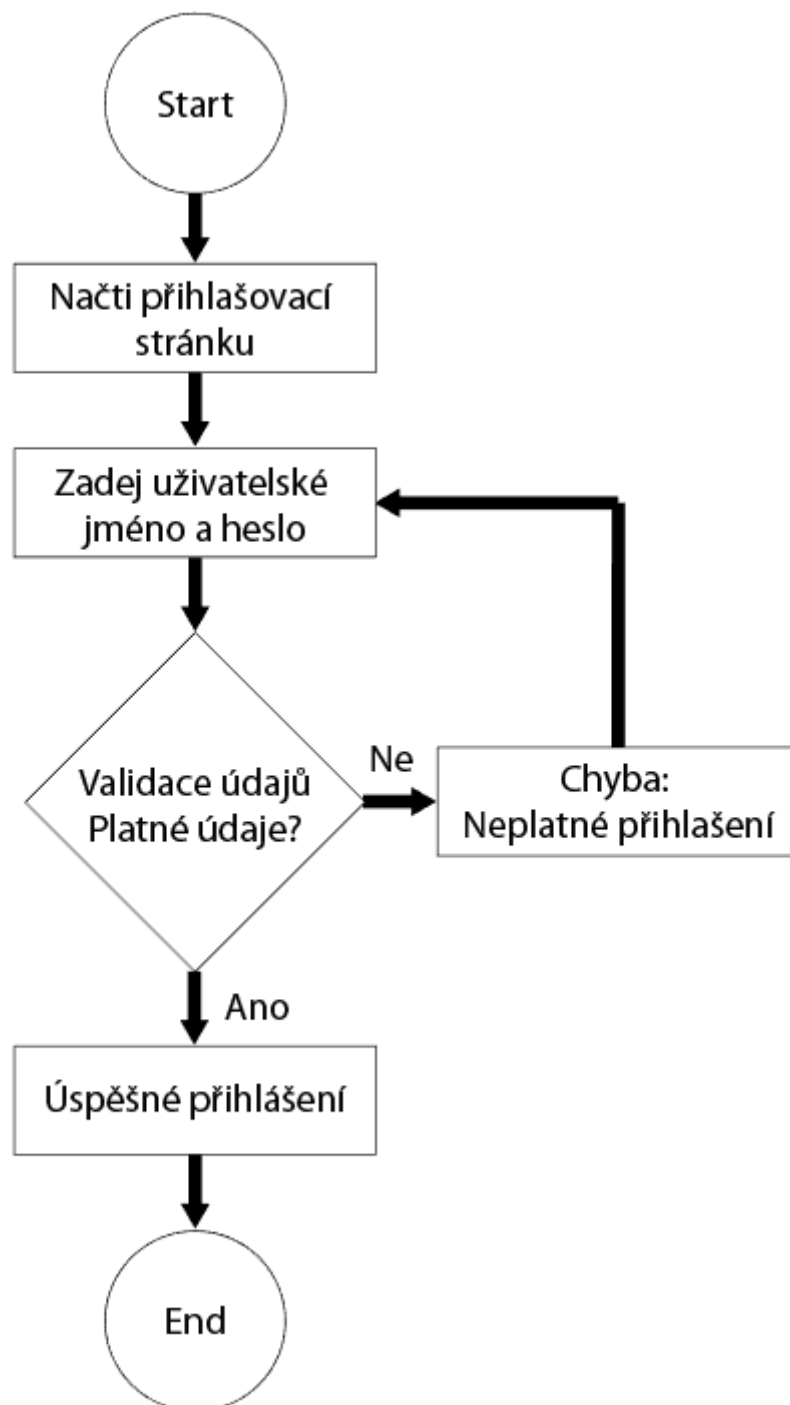
Blokový diagram je grafické znázornění procesu nebo systému, které ukazuje vztahy mezi různými bloky nebo komponentami. Používá se k zjednodušení složitých systémů, procesů nebo algoritmů a zlepšení jejich porozumění. V diagramu jsou jednotlivé kroky nebo komponenty znázorněny jako bloky, které jsou propojeny šipkami, které ukazují pořadí nebo tok procesu.

Vlastnosti blokového diagramu:

- **Bloky:** Každý blok představuje nějaký krok nebo proces. Bloky jsou často obdélníkové a mohou obsahovat text popisující funkci nebo akci, kterou vykonávají.
- **Šipky:** Šipky mezi bloky ukazují pořadí, jakým se proces nebo informace pohybují mezi jednotlivými bloky.
- **Symboly:** Bloky mohou být různého tvaru, přičemž každý tvar může mít jiný význam. Například:
 - **Obdélník** – proces nebo akce
 - **Kosočtverec** – rozhodnutí nebo podmínka (například ano/ne)
 - **Kruh** – spojovací bod nebo start/stop



Příklad 1 Přihlášení



Testování pomocí rozhodovací tabulky

Rozhodovací tabulka (tabulka příčiny a následku) je tabulka popisující kombinaci vstupů a/nebo faktorů (příčin) s jejich odpovídajícími výstupy a akcemi (následky). Rozhodovací tabulka se používá v systémech s komplexní byznys logikou, a tak můžeme pochopit jak systém pracuje a pokrýt všechny byznysové rozhodnutí

Kdy použít rozhodovací tabulku?

- **Složitá logika:** Rozhodovací tabulky jsou obzvláště užitečné při testování aplikací s komplexními rozhodovacími stavy, kde je mnoho podmínek, které se kombinují.
- **Testování více parametrů:** Jsou ideální pro testování funkcionalit, které závisí na několika parametrech, např. při testování různých kombinací oprávnění, přihlašovacích scénářů nebo procesů validace formulářů.
- **Automatizované testování:** Rozhodovací tabulky lze použít jako základ pro automatizované testy, což zajišťuje efektivní a systematické testování.

Příklad 1

Uvedme si příklad testování s rozhodovací tabulkou pro semafor. Představme si, že máme 3 podmínky:

- Svítí zelené světlo (True/False)
- Svítí oranžové světlo (True/False)
- Svítí červené světlo (True/False)

Kombinace světel (K)	Zelené	Červené	Oranžové	Výsledek (reakce řidiče)
K1	True	False	False	Řidič může jet
K2	False	True	False	Stop
K3	False	False	True	Zastav pokud můžeš
K4	False	True	True	Připravit k jízdě

Příklad 2

Uvedme si příklad testování s rozhodovací tabulkou pro přihlašovací proces v aplikaci. Představme si, že máme 3 podmínky:

- Uživatelské jméno je platné (True/False)
- Heslo je platné (True/False)
- Uživatel je aktivní (True/False)

Pro každou kombinaci těchto podmínek určíme, zda by uživatel měl mít přístup k aplikaci nebo ne. Výstupy mohou být například:

- Přístup povolen
- Přístup zamítnut
- Chyba (např. "Nesprávné uživatelské jméno nebo heslo")

Kombinace vstupů (V)	Uživatelské jméno	Heslo	Aktivní uživatel	Výsledek
V1	True	True	True	Přístup povolen
V2	True	True	False	Přístup zamítnut
V3	True	False	True	Chyba: Nesprávné heslo
V4	True	False	False	Chyba: Nesprávné heslo
V5	False	True	True	Chyba: Nesprávné jméno
V6	False	True	False	Chyba: Nesprávné jméno
V7	False	False	True	Chyba: Nesprávné jméno a heslo
V8	False	False	False	Chyba: Nesprávné jméno a heslo

Use case testování

Use Case testování je metoda testování softwaru, která se zaměřuje na ověření, zda systém funguje podle očekávání v konkrétních scénářích, které modelují skutečné použití aplikace. Testování podle **use case** je často používáno pro validaci funkčnosti systému **z pohledu uživatele**, protože use case scénáře reprezentují interakce mezi uživatelem (nebo jinými systémy) a aplikací.

1. Co je to Use Case?

Use case je popis konkrétního scénáře interakce uživatele s aplikací. Tento scénář obvykle obsahuje:

- **Role (uživatele nebo jiný systém):** Kdo interaguje se systémem.
- **Cíle (co chceme dosáhnout):** Co se určitá role snaží dosáhnout prostřednictvím interakce se systémem.
- **Scénář (postup interakce):** Krok za krokem popis toho, jak určitá role interaguje s aplikací, aby dosáhla cíle.
- **Očekávané výsledky:** Výstupy nebo změny ve stavu systému po provedení akce.

Use case testování tedy ověřuje, zda systém splňuje očekávání uživatelů a zda vykonává správné akce pro daný scénář.

2. Výhody Use Case Testování

- **Zaměření na skutečné použití:** Use case testy jsou blízké reálným scénářům použití aplikace, což znamená, že testování bude více odpovídat skutečným potřebám a zkušenostem uživatelů.
- **Usnadňuje komunikaci:** Testování pomocí use case scénářů umožňuje vývojářům, testerům a zainteresovaným osobám lépe porozumět funkčnosti systému.
- **Testování v reálných podmínkách:** Testuje systém v podmínkách, které budou reálné pro uživatele, čímž může odhalit problémy, které by jinak nemusely být detekovány.
- **Identifikace mezery mezi požadavky a implementací:** Tento typ testování často odhalí rozdíly mezi tím, co bylo požadováno v dokumentaci, a tím, co je skutečně implementováno.
- **Zlepšení uživatelské zkušenosti:** Testování s reálnými scénáři používání aplikace pomáhá zajistit, že systém je uživatelsky přívětivý a plně funkční.

Testování Přechodových Stavů

Testování přechodových stavů je technika, která se používá k testování chování systému při přechodu mezi různými stavy. Tento typ testování je důležitý zejména u aplikací a systémů, které mají definované stavy, mezi kterými uživatelé nebo systém přecházejí. Testování přechodových stavů zajišťuje, že všechny přechody mezi stavy jsou správně implementovány a že systém reaguje podle očekávání.

Klíčové aspekty testování přechodových stavů:

- **Stavy systému:** Každý systém nebo aplikace může mít různé stavy, jako jsou přihlášení, odhlášení, čekající na vstup, zpracovávání dat, dokončení úkolu apod. Každý z těchto stavů může mít specifické chování a akce, které musí být provedeny při přechodu mezi nimi.
- **Přechody mezi stavy:** Testování přechodů zahrnuje testování chování aplikace při přechodu z jednoho stavu do druhého. Přechody mohou být vyvolány uživatelskými akcemi (např. kliknutí na tlačítko), časovými událostmi (např. vypršení doby platnosti session), nebo systémovými událostmi (např. potvrzení transakce).
- **Důležitost pokrytí všech přechodů:** Je důležité otestovat všechny možné přechody mezi stavy, aby se zajistilo, že systém správně reaguje na všechny situace. To zahrnuje i nepředvídané nebo chybové přechody, jako jsou např. přechody z neúplného nebo neplatného stavu.

Příklad testování přechodových stavů:

Představme si, že máme systém pro přihlašování uživatele, který má následující stavy:

- Stav 1 (Přihlášení): Uživatel je přihlášen.
- Stav 2 (Odhlášení): Uživatel není přihlášen.
- Stav 3 (Zpracovávání): Uživatel podal formulář a systém ho zpracovává.
- Stav 4 (Úspěch): Systém úspěšně zpracoval přihlášení.
- Stav 5 (Chyba): Došlo k chybě při přihlášení.

Test 1: Přechod z odhlášeného stavu na přihlášený:

- Krok: Uživatel zadá platné přihlašovací údaje a klikne na tlačítko „Přihlásit se“.
- Očekávaný výsledek: Systém přejde ze stavu „Odhlášení“ do stavu „Přihlášení“.

Test 2: Přechod z přihlášeného stavu na zpracování:

- Krok: Uživatel vyplní formulář a klikne na tlačítko „Odeslat“.
- Očekávaný výsledek: Systém přejde ze stavu „Přihlášení“ do stavu „Zpracovávání“.

Test 3: Přejchod z přihlášeného stavu na chybu:

- Krok: Uživatel zadá neplatné údaje při přihlašování (např. špatné heslo).
- Očekávaný výsledek: Systém přejde do stavu „Chyba“.

Nejčastější chyby a jak se jim vyhnout

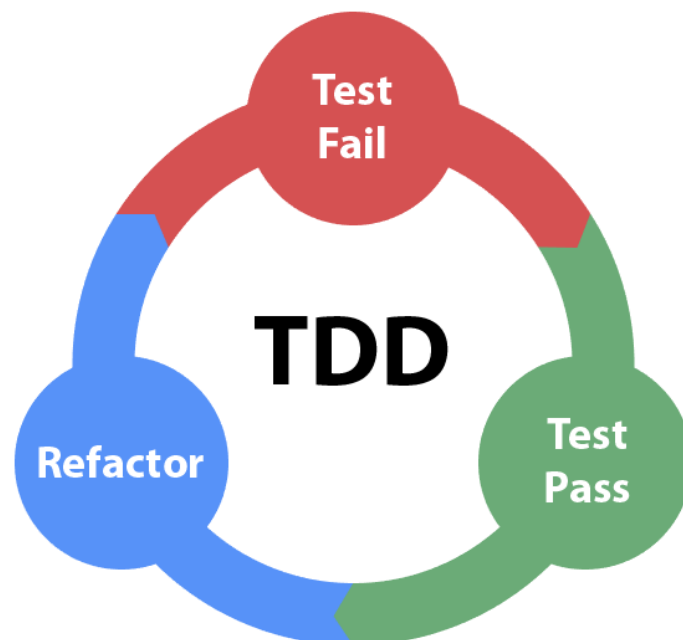
- Nedostatečná specifikace požadavků: Pokud nejsou jasně definovány požadavky, mohou testovací scénáře pokrýt nesprávné oblasti. Důležité je, aby zadání obsahovalo jasná očekávání.
- Chybějící negativní testování: Často se stává, že se soustředíme na pozitivní testovací případy. Důležité je ale zahrnout i negativní testy, které ověří, že aplikace správně reaguje na chybné vstupy nebo neplatné akce.
- Nedostatečná aktualizace testovacích dat: Testovací data by se měla pravidelně revidovat a aktualizovat podle vývoje aplikace. Například při aktualizaci datové struktury nebo změně obchodní logiky.

Vývoj a testování založené na metodách: TDD, BDD, KDD

Test Driven Development

Test Driven Development (TDD) — je jednou z technik agilního vývoje softwaru. Tato technika je založena na několikanásobném opakování několika kroků tak, aby jsme po několika "iteracích" měli fungující aplikaci.

- Vývojář začne práci napsáním testu, např. připojením k databázi a stažením záznamu, spuštěním testu - neprojde.
- Vývojář implementuje funkcionality, test projde.
- Vývojář provádí změny ve zdrojovém kódu, aby odpovídal přijatým standardům.



Behavior Driven Development

Behavior Driven Development (BDD) — je nejen proces vývoje softwaru jako v případě TDD, ale také obecně vývojovým procesem. BDD zahrnuje vytvoření softwaru popisem chování z perspektivy jeho účastníků.

Základ této metody je dozvědět se potřeby, cíle a očekávání zákazníků jak je to možné a vytvořit software, který toto splňuje.

BDD je už použit při stanovení požadavků, které se potom snadno dají převést na akceptační kritéria pro testy.

BDD se vyznačuje použitím přirozeného jazyka, který vyjadřuje chování, očekávání a výsledky, které chápou vývojáři a testéři. Klíčová slova používaná k popisu chování: **FEATURE, SCENARIO, GIVEN, WHEN, THEN.**

```
Feature: Login
```

```
Scenario: Login successful
```

```
Given Open login page  
When User enters correct login name  
And User enters correct password  
And User press Login button  
Then User is logged in
```

```
Scenario: Login failed
```

```
Given Open login page  
When User enters correct login name  
And User enters incorrect password  
And User press Login button  
Then User is not logged in
```

Keyword Driven Development

Keyword-based testing je technika pro manuální i automatické testování. Idea této techniky je připravit test v určeném programovacím jazyce, který je poté prezentován za pomoci jednoho nebo více klíčových slov. Testy používající klíčová slova jsou obvykle uloženy jako "Test Suite" scénáře a může obsahovat mnoho testů.

Klíčová slova jsou nejčastěji použity následovně:

keyword | arguments

- Open page | <http://www.testing.tech>
- Log in | tester tester
- Check if logged in

Výhodou Keyword Driven Development je, že i nezkušení testéři mohou připravovat automatické testy na základě množiny klíčových slov.

Odkazy:

- <https://www.justinmind.com/blog/user-story-examples/>
- <https://www.tutorialspoint.com/selenium/practice/login.php>
- <https://demo.opencart.com/en-gb?route=common/home>
- <https://opensource-demo.orangehrmlive.com/web/index.php/auth/login>
- <https://parabank.parasoft.com/parabank/register.htm>
- <https://www.saucedemo.com/v1/>
- <https://demoqa.com/automation-practice-form>
- <https://practicetestautomation.com/practice-test-login/>
- <https://testrail.en.softonic.com/>
- <https://www.reqview.com/doc/example-requirements-documents/>