# Testerský slovník časť 1:

(postupne budem zverejňovať pojmy, ktoré som nazbierala)

🔹 **Placeholder**  
Dočasné zástupné miesto vo výstupe (napr. {{meno}}), ktoré sa má nahradiť skutočnou hodnotou z databázy alebo systému.

🔹 **Fallback**  
Záložná hodnota alebo správanie, ktoré systém použije, ak niečo chýba. Napr. ak nie je dostupný preklad, zobrazí sa text v angličtine ako fallback.

🔹 **Rollback**  
Návrat databázy alebo systému do predchádzajúceho stavu, ak niečo zlyhá. Používa sa napríklad po chybe počas aktualizácie údajov.

🔹 **Validácia**  
Overenie, či sú vstupné dáta správne **z hľadiska formy** (napr. e-mail musí obsahovať @, dátum musí byť v správnom formáte).

🔹 **Verifikácia**  
Overenie, či **systém robí to, čo má** – teda či zodpovedá špecifikácii. (Opak validácie: validujeme vstup, verifikujeme správanie systému.)

🔹 **CRON**  
Naplánovaná úloha, ktorá sa spúšťa automaticky v určitý čas alebo interval (napr. každý deň o 2:00 v noci). Používa sa napr. na nočné reporty.

🔹 **API (Application Programming Interface)**  
Rozhranie, cez ktoré medzi sebou komunikujú aplikácie. Tester ho používa na overenie, či systém správne reaguje na požiadavky (napr. vytvorenie používateľa cez POST požiadavku).

Týmto nekončíme, len začíname. Dajte do komentov termíny alebo slovné spojenia, s ktorými ste sa pri testovaní stretli. Tie, ktoré vás zaujali, nevedeli ste, čo znamenajú a pod.

**#tester #testovacislovnik #manualnytester #automatizovanytester #ITtester #QA #ITkariera**

--

# Testerský slovník časť 2:

(postupne budem zverejňovať pojmy, ktoré som nazbierala)

🔹 **Bug**  
Chyba v systéme – niečo nefunguje podľa očakávania alebo špecifikácie. Môže byť funkčná, vizuálna, bezpečnostná, výkonnostná…

🔹 **Test case (testovací prípad)**  
Konkrétny scenár testovania so vstupmi, krokmi, očakávaným výsledkom a realitou. Napr. „Keď zadám neplatný e-mail, zobrazí sa chyba“.

🔹 **Edge case**  
Hraničný alebo zriedkavý scenár, ktorý sa bežne nestáva, ale môže spôsobiť problém. Napr. používateľ s menom „A“ alebo 0 položiek v košíku.

🔹 **Smoke test**  
Rýchla kontrola, či aplikácia vôbec beží a základné funkcie fungujú. Nejde do hĺbky – len overuje, či „nesmrdí“ (odtiaľ názov).

🔹 **Regression test (regresný test)**  
Overenie, či nové zmeny nerozbili to, čo predtým fungovalo. Pomáha zachytiť „nečakané vedľajšie účinky“ pri vývoji.

🔹 **Endpoint**  
Konkrétny bod v API, kam sa posiela požiadavka. Napr. POST /api/login alebo GET /api/users/123. Môžeme ho testovať ručne aj automatizovane.

🔹 **Test data (testovacie dáta)**  
Sada údajov, ktoré používame pri testovaní. Môžu byť reálne, anonymizované, fiktívne alebo náhodne generované.

🔹 **Soft delete**  
Záznam nie je fyzicky zmazaný z databázy – len označený ako „zmazaný“ (napr. deleted\_at nie je NULL). Bežné napríklad pri používateľoch alebo faktúrach.

🔹 **Assert / Assercia**  
Vyjadrenie v teste, ktoré overuje, či výsledok zodpovedá očakávaniu. Napr. assert status\_code == 200 v automatizovanom teste.

--

# Testerský slovník časť 3:

(postupne budem zverejňovať pojmy, ktoré som nazbierala)

🔹 **Mockovanie (Mocking)**  
Simulovanie správania iných častí systému (napr. databázy, API), ktoré ešte nie sú hotové alebo ich nechceme pri teste volať. Používa sa najmä pri automatizovaných testoch.

🔹 **Testovací scenár (Test scenario)**  
Vyššia úroveň ako test case – popisuje, čo sa má testovať, nie krok po kroku. Napr. „Over, že používateľ môže vytvoriť objednávku bez registrácie.“

🔹 **Test plan (testovací plán)**  
Dokument, ktorý popisuje, čo, ako, kedy a kým sa bude testovať. Obsahuje rozsah, prístup, zdroje, časový plán, riziká a kritériá ukončenia.

🔹 **Test coverage (pokrytie testami)**  
Miera toho, **koľko funkčnosti je pokrytej testami**. Môže byť merané podľa funkcií, kódu (napr. % riadkov), scenárov atď.

🔹 **User Story**  
Popis funkcie z pohľadu používateľa. Napr. „Ako registrovaný používateľ chcem vidieť históriu objednávok, aby som si mohol dohľadať, čo som kúpil.“

🔹 **Exploratory testing (prieskumné testovanie)**  
Testovanie bez pevne daných krokov. Tester skúma systém, hľadá nečakané správanie a zapája intuíciu, zvedavosť aj predchádzajúce skúsenosti.

🔹 **Negative test (negatívny test)**  
Test, ktorý **overuje, že systém správne zvládne nesprávne alebo neočakávané vstupy**. Napr. prázdne pole, zlý formát dátumu, nulová hodnota.

🔹 **Acceptance criteria (akceptačné kritériá)**  
Podmienky, ktoré musia byť splnené, aby bola úloha považovaná za hotovú. Pomáhajú testovať, čo má funkcia robiť a čo nie.

🔹 **Regression bug (regresná chyba)**  
Chyba, ktorá sa **znovu objavila po tom, čo bola už raz opravená**. Zvyčajne spôsobená novou zmenou v kóde, ktorá niečo rozbila.

--

# Testerský slovník časť 4:

(postupne budem zverejňovať pojmy, ktoré som nazbierala)

🔹 **Test suite (súbor testov)**  
Zoskupenie viacerých testovacích prípadov, ktoré sa spúšťajú spolu. Môže ísť o testy jednej funkcie, modulu alebo typu (napr. regresné testy).

🔹 **Sanity test**  
Rýchly test malej časti systému, ktorý overí, či má zmysel pokračovať v ďalšom testovaní. Ak sanity test zlyhá, ostatné testy nemajú význam.

🔹 **Flaky test**  
Test, ktorý raz prejde, raz zlyhá, aj keď sa nič nezmenilo. Veľký problém v automatizácii – testovanie je potom nedôveryhodné.

🔹 **Test environment (testovacie prostredie)**  
Konfigurácia systémov, kde sa testuje. Môže mať inú databázu, verziu kódu, práva než produkcia. Dôležité je vedieť, **v akom prostredí testujeme**.

🔹 **Load test (záťažový test)**  
Overuje, ako sa systém správa pri bežnej a zvýšenej záťaži. Napr. 1 000 používateľov naraz, veľké množstvo požiadaviek, dlhé reporty.

🔹 **Performance test (výkonnostný test)**  
Testy, ktoré sledujú rýchlosť reakcie, dobu načítania, spotrebu pamäte, využitie CPU. Cieľ: zistiť, či je systém dostatočne svižný.

🔹 **Test matrix (testovacia matica)**  
Tabuľka, ktorá ukazuje, **čo všetko je pokryté testami** – napr. rôzne kombinácie prehliadačov, zariadení, vstupov alebo rolí.

🔹 **Boundary value (hraničná hodnota)**  
Najnižšia, najvyššia alebo okrajová hodnota vstupu. Hraničné hodnoty bývajú často miestom výskytu chýb. (Napr. ak vek 18 je povolený – čo 17, 18, 19?)

🔹 **Equivalence partitioning (rozdelenie do tried ekvivalencie)**  
Technika, kde vstupy delíme na skupiny, ktoré by sa mali správať rovnako. Stačí otestovať 1 hodnotu z každej skupiny (napr. platné vs. neplatné PSČ).

--

# Testerský slovník časť 5:

(postupne budem zverejňovať pojmy, ktoré som nazbierala)

🔹 **Unit test (jednotkový test)**  
Test malej časti kódu – najčastejšie jednej funkcie alebo metódy. Píšu ho vývojári, overuje, že „tehlička“ systému funguje samostatne správne.

🔹 **Integration test (integračný test)**  
Test viacerých modulov, ktoré spolu komunikujú – napr. databáza + backend, backend + frontend. Cieľ: nájsť chyby v prepojení.

🔹 **System test (systémový test)**  
Kompletné testovanie celej aplikácie ako celku. Overuje, či spolu všetko funguje, ako má – vrátane UI, databázy, API, autentifikácie…

🔹 **UAT (User Acceptance Testing)**  
Akceptačné testovanie – robia ho samotní používatelia alebo zákazník. Overujú, či systém **zodpovedá ich potrebám a požiadavkám**.

🔹 **Static testing (statické testovanie)**  
Testovanie **bez spustenia kódu** – napr. kontrola dokumentácie, revízia špecifikácií, kontrola testovacích prípadov.

🔹 **Dynamic testing (dynamické testovanie)**  
Testovanie **reálneho správania systému počas behu** – teda klikáš, zadávaš vstupy, sleduješ odpovede. Presný opak statického testovania.

🔹 **Smoke vs. Sanity test**

* **Smoke**: rýchla kontrola po novej verzii – spustí sa systém?
* **Sanity**: kontrola konkrétnej opravy alebo funkcie – funguje konkrétna vec, na ktorú sme sa zamerali?

🔹 **Bug lifecycle (životný cyklus chyby)**  
Cesta chyby od objavenia po uzavretie. Fázy: **New → Open → In progress → Resolved → Retested → Closed** (alebo Rejected, Duplicate, Won’t fix…).

🔹 **Test report (testovacia správa)**  
Výstup z testovania – zhrnutie toho, čo bolo testované, čo prešlo, čo zlyhalo, koľko chýb sa našlo a akého typu boli. Dôležité pre rozhodovanie o vydaní verzie.

--

# Testerský slovník časť 6:

(postupne budem zverejňovať pojmy, ktoré som nazbierala)

🔹 **Environment variables (premenné prostredia)**  
Hodnoty špecifické pre prostredie (napr. test, staging, produkcia), ktoré ovplyvňujú beh aplikácie – ako napr. URL databázy, API token, debug mód.

🔹 **Test data management (správa testovacích dát)**  
Organizácia, príprava a čistenie údajov, ktoré sa používajú pri testovaní. Cieľ: mať **relevantné, spoľahlivé a opakovateľné** testovacie scenáre.

🔹 **Defect triage (triedenie chýb)**  
Spoločné stretnutie (testeri, vývojári, PM), kde sa prechádzajú nahlásené chyby, priraďujú sa im priority, zodpovednosti a rozhoduje sa, čo sa bude opravovať.

🔹 **Retesting vs. Regression testing**

* **Retesting**: overujeme, či **konkrétna opravená chyba** už neexistuje.
* **Regression testing**: overujeme, či oprava **nerozbila nič iné**.

🔹 **Precondition / Postcondition**

* **Precondition**: čo **musí byť splnené pred** testom (napr. „používateľ je prihlásený“)
* **Postcondition**: čo **má platiť po** teste (napr. „objednávka je v stave 'spracovaná'“)

🔹 **Severity vs. Priority (závažnosť vs. priorita)**

* **Severity**: ako **vážna je chyba** (napr. systém padne = vysoká závažnosť)
* **Priority**: ako **rýchlo ju treba opraviť** (napr. chyba v texte na hlavnej stránke = nízka závažnosť, ale vysoká priorita)

🔹 **Acceptance test (akceptačný test)**  
Test, ktorý **potvrdzuje, že funkcia robí to, čo má**, podľa zadania. Môže byť formálny (UAT) alebo automatizovaný (napr. cez Cucumber).

🔹 **Automated test (automatizovaný test)**  
Test, ktorý sa spúšťa automaticky (napr. cez skripty alebo CI/CD pipeline). Rýchly, opakovateľný, ale náročný na údržbu.

🔹 **Manual test (manuálny test)**  
Test vykonaný človekom – klikáš, skúšaš, pozoruješ. Pomalší, ale flexibilnejší, ideálny na UI testy, prieskum, alebo keď niečo nie je stabilné.

🔹 **CI/CD (Continuous Integration / Continuous Delivery)**  
Proces, kde sa zmeny kódu automaticky testujú a nasadzujú. Pomáha rýchlemu a bezpečnému vývoju. Tester môže napojiť svoje testy do tohto procesu.

Týmto nekončíme, len začíname. Dajte do komentov termíny alebo slovné spojenia, s ktorými ste sa pri testovaní stretli. Tie, ktoré vás zaujali, nevedeli ste, čo znamenajú a pod.

**#tester #testovacislovnik #manualnytester #automatizovanytester #ITtester #QA #ITkariera**

--

# Testerský slovník časť 7:

(postupne budem zverejňovať pojmy, ktoré som nazbierala)

🔹 **Heuristika**  
Skúsenostné pravidlo, ktoré tester používa na odhadovanie chýb tam, kde nemá jasné špecifikácie. Napr. „čo sa často kazí, bude sa kaziť znova“, alebo „najprv otestuj najčastejšie scenáre“.

🔹 **Bug bash**  
Organizované skupinové testovanie, kde sa do hľadania chýb zapoja testeri, vývojári, PM aj ďalší – cieľom je čo najrýchlejšie nájsť čo najviac chýb.

🔹 **Defect leakage (únik chyby)**  
Chyba, ktorá prešla testovaním, ale našla sa až **v produkcii**. Dôležitý ukazovateľ kvality testovania.

🔹 **Test oracle**  
Zdroje, ktoré ti povedia, **čo je správne správanie systému** – špecifikácie, návrhy, podobný systém, skúsenosti, zákony, zdravý rozum…

🔹 **Test strategy (testovacia stratégia)**  
Vysoká úroveň popisu **ako testovať produkt** – aký bude prístup, čo testovať budeme a čo nie, aké typy testov použijeme a prečo.

🔹 **Test log (testovací záznam)**  
Detailný záznam o tom, **ako test prebiehal** – čo bolo spustené, kedy, s akými dátami a výsledkami. Užitočný pri hľadaní príčiny zlyhania.

🔹 **Session-based testing**  
Forma prieskumného testovania, kde má tester vopred definovaný cieľ, časový rámec a po teste vytvorí záznam o tom, čo skúšal a čo našiel.

🔹 **Quality gate (kvalitný prah)**  
Podmienka, ktorá musí byť splnená, aby systém mohol postúpiť ďalej (napr. z testu do produkcie). Môže byť počet chýb, pokrytie testami, úspešnosť CI testov atď.

🔹 **Shift-left testing**  
Prístup, kde sa testovanie presúva **čo najskôr do vývojového procesu** – začína už pri návrhu, špecifikácii alebo v kóde (napr. unit testy, code review, CI).

🔹 **QA vs QC (Quality Assurance vs Quality Control)**

* **QA**: procesy, ktoré **predchádzajú chybám** (plánovanie, štandardy, prevencia)
* **QC**: konkrétne **hľadanie chýb** v produkte (testovanie, kontrola výsledkov)

Týmto nekončíme, len začíname. Dajte do komentov termíny alebo slovné spojenia, s ktorými ste sa pri testovaní stretli. Tie, ktoré vás zaujali, nevedeli ste, čo znamenajú a pod.

**#tester #testovacislovnik #manualnytester #automatizovanytester #ITtester #QA #ITkariera**

--

# Testerský slovník časť 8:

(postupne budem zverejňovať pojmy, ktoré som nazbierala)

🔹 **Testability (testovateľnosť)**  
Miera, do akej je systém navrhnutý tak, aby sa dal efektívne testovať. Vysoká testovateľnosť znamená jasné logy, jednoduché overenie výstupov, stabilné prostredie a dobrú dokumentáciu.

🔹 **Bug reproducibility (reprodukovateľnosť chyby)**  
Schopnosť opakovane vyvolať chybu rovnakým spôsobom. Ak sa chyba vyskytne len niekedy, je „flaky“ a ťažko sa opravuje.

🔹 **Logging**  
Zaznamenávanie udalostí počas behu aplikácie. Dobrý log pomáha rýchlo pochopiť, čo sa stalo, kde a prečo – často je prvou pomôckou testera aj vývojára pri chybe.

🔹 **Root cause analysis (analýza príčiny)**  
Postup, ktorým zisťujeme, čo presne spôsobilo chybu. Dôležitý krok pri riešení bugov aj pri zlepšovaní procesov (napr. prečo sa chyba dostala do produkcie?).

🔹 **Test steps (testovacie kroky)**  
Konkrétne inštrukcie, ktoré tester vykoná počas testu. Dobré testovacie kroky sú jednoznačné, očíslované a vedú k overiteľnému výsledku.

🔹 **Test evidence (dôkaz testu)**  
Záznam toho, že test prebehol – napr. screenshot, výpis logu, export údajov. Pomáha preukázať výsledok a je dôležitý pri formálnom testovaní.

🔹 **Severity 1 / Priority 1**  
Bežné označenie najvážnejších a najurgentnejších chýb. Severity 1 = systém je nefunkčný. Priority 1 = oprav to hneď. Nie vždy sa tieto dve zhodujú.

🔹 **Happy path (šťastná cesta)**  
Ideálny priebeh scenára, keď všetko prebehne správne bez chýb. Väčšina testov začína práve overením happy path.

🔹 **Testware**  
Všetko, čo vznikne počas testovania: testy, dáta, skripty, výsledky, logy... Testware má svoju hodnotu – treba ho udržiavať a zálohovať.

🔹 **Timebox**  
Časové ohraničenie úlohy alebo testu. Napr. „prieskumné testovanie 60 minút“. Pomáha udržať fokus, najmä pri manuálnych a kreatívnych úlohách.

🔹 **Blocker**  
Chyba alebo okolnosť, ktorá znemožňuje pokračovať v testovaní. Napr. keď nefunguje prihlásenie alebo nemáme prístup do prostredia.

🔹 **Repro steps (kroky na reprodukciu)**  
Popis, ako vyvolať chybu. Jasné a presné repro steps sú základ dobrého bug reportu – bez nich vývojár nevie, čo má opraviť.

🔹 **Golden path**  
Podobné ako „happy path“, ale často širšie – opisuje odporúčaný alebo najčastejší spôsob, ako používateľ systém používa.

🔹 **Test debt (dlh v testovaní)**  
Podobne ako technický dlh – ide o zanedbané alebo chýbajúce testy, ktoré sa „raz vrátia“ a spôsobia problémy. Napr. chýbajúce regresné testy, neautomatizované kritické funkcie.

🔹 **Scripted vs. Unscripted testing**  
Scripted: testujeme presne podľa napísaných krokov.  
Unscripted: testujeme voľne, na základe skúseností, intuície a aktuálneho správania systému.

Týmto nekončíme, len začíname. Dajte do komentov termíny alebo slovné spojenia, s ktorými ste sa pri testovaní stretli. Tie, ktoré vás zaujali, nevedeli ste, čo znamenajú a pod.

**#tester #testovacislovnik #manualnytester #automatizovanytester #ITtester #QA #ITkariera**

--

# Testerský slovník časť 9:

(postupne budem zverejňovať pojmy, ktoré som nazbierala)

🔹 **Test freeze**  
Stav, keď sa už nemenia testy ani prostredie. V tomto bode by už malo prebiehať len spúšťanie existujúcich testov a príprava na vydanie.

**🔹 Zombie bug**

Chyba, ktorá sa opakovane vracia, aj keď bola "opravená". Buď nie je naozaj vyriešená, alebo sa vždy niekto „postará“, aby znovu vstala z mŕtvych.

**🔹 Rubber duck debugging**

Vysvetľovanie problému (často nahlas alebo bábke/kačke) – už samotné rozprávanie pomáha nájsť riešenie. Aj tester to môže využiť na ujasnenie si bug reportu.

**🔹 Test debt (dlh v testovaní)**

Podobne ako technický dlh – keď sa testy odkladajú, skripty sa neudržiavajú alebo chýba pokrytie. Neskôr sa to vypomstí.

**🔹 Happy smoke + grumpy sanity**

Zábavné pomenovanie na zapamätanie rozdielu: smoke test = „všetko ok, môžeme začať“, sanity test = „ok, ale tá konkrétna oprava fakt funguje?“

**🔹 Dogfooding**

Interné používanie vlastného softvéru (napr. vývojári testujú vlastnú appku). Tester by mal byť pri tom – nájde chyby, ktoré vývojári ignorujú.

**🔹 Feature toggle / Flag**

Prepínač, ktorým sa zapína alebo vypína funkcia bez nutnosti meniť kód. Tester musí vedieť, ktoré sú aktívne – inak môže testovať „vzduch“.

**🔹 Dead code**

Časť kódu, ktorá sa nikdy nevykoná – a tým pádom sa často ani netestuje. Ale keď sa v nej niečo zmení, môže rozbiť to, čo „sa nemalo používať“.

**🔹 Red-Green-Refactor**

Prístup pri písaní testov (TDD): najprv test zlyhá (red), potom sa opraví (green), a potom sa kód zlepší (refactor). Tester môže sledovať, či vývojári refaktorujú – alebo len zalepujú.

**🔹 QA translator**

Tester ako tlmočník medzi tým, čo vývojár povedal, čo PM (projektový manažér) myslel, čo klient chcel a čo appka reálne robí.

**🔹 Click monkey**

Posmešné označenie manuálneho testera, ktorý len kliká bez kontextu. V dobrom slúži ako memento: tester má byť zvedavý, nie len klikač.

Týmto nekončíme, len začíname. Dajte do komentov termíny alebo slovné spojenia, s ktorými ste sa pri testovaní stretli. Tie, ktoré vás zaujali, nevedeli ste, čo znamenajú a pod.

**#tester #testovacislovnik #manualnytester #automatizovanytester #ITtester #QA #ITkariera**

--

**🔹 Dirty data**

Testovacie dáta, ktoré sú poškodené, neaktuálne alebo nedokumentované. Skrátka také, čo spôsobia, že test „čudne padne“ a nikto nevie prečo.

➡️ *Poučenie pre testera*: Dobrý test potrebuje dobré dáta. Chaos v dátach vedie k falošným chybám.

**🔹 Boomerang bug**

Chyba, ktorá sa vracia po tom, čo bola uzavretá ako „fixed“. Niekedy kvôli nesprávnemu retestovaniu, niekedy preto, že sa fix aplikoval len *na polovicu systému*.

**🔹 Test pyramid**

Koncept, podľa ktorého by malo byť najviac unit testov (dole), menej integračných (v strede) a najmenej UI testov (hore). Opačná pyramída = testovacie bolenie brucha.

📌 *Tip*: Tester môže upozorniť tím, ak pyramída vyzerá ako zmrzlina postavená na špičke.

**🔹 Time bomb**

Kód, ktorý funguje len do určitého dátumu. Napr. keď je platnosť testovacích dát „do 2022“ a odvtedy všetko padá. Tester = budík, ktorý to má nájsť skôr, než vybuchne.

**🔹 Shadow testing**

Testovanie nového systému na pozadí bez toho, aby ovplyvnil produkciu. Napr. nový výpočet ceny sa spúšťa paralelne s tým starým – a porovnávajú sa výsledky.

**🔹 Mutation testing**

Metóda, pri ktorej sa zámerne vkladajú chyby do kódu, aby sa overilo, či ich testy zachytia. Ak ich nezachytia → testy sú slabé. 🧠 Hardcore testovacia technika.

**🔹 Known issue**

Chyba, o ktorej *vieme*, ale ešte ju nemáme opravenú. Často ju nájde nový tester, s nadšením nahlási... a dostane späť odpoveď „áno, to vieme, máme v Jire od 2019“.

**🔹 Test branching**

Testy sú verzované podobne ako kód. Rôzne branche testov zodpovedajú rôznym verziám aplikácie. Veľká výzva pre koordináciu medzi QA a vývojom.

**🔹 Jumping to conclusions**

Keď tester nahlási bug, ale neotestoval všetky možnosti alebo neoveril, že ide o chybu (a nie napr. nepochopenie feature). 🧐 Dobrá poznámka pre juniorov – pozor na zbrklosť.

**🔹 Fuzzy testing (Fuzz testing)**

Automatizovaný spôsob, ako systém zaplaviť náhodnými, nečakanými alebo „šialenými“ vstupmi. Cieľ: nájsť, kde sa niečo rozsype. Obľúbené pri bezpečnostných testoch.

**🔹 Bug taxonomy**

Systém triedenia chýb podľa typu, dôvodu alebo dôsledku. Tester vie lepšie analyzovať, kde sa opakujú chyby – a QA tím sa z toho môže učiť.

--

**Heisenbug**  
Bug, ktorý zmizne alebo zmení správanie, keď sa ho snažíš debugovať. V bežnom režime padá ako muchotrávka. Zapneš logovanie? Debug mód? Sleduješ ho?

**HTML injection**  
Zraniteľnosť, pri ktorej môže útočník vložiť vlastný HTML kód do stránky – napr. cez formulár alebo URL. Výsledok? Od obyčajného rozhodenia dizajnu až po falošné tlačidlá, iframe, prehliadanie dát iných používateľov…

**XOR**  
(z anglického *exclusive OR*, slovensky **„výlučné ALEBO“**) je logická operácia, ktorá vracia hodnotu **pravda (true)** iba vtedy, keď práve **jedna** z dvoch hodnôt je pravdivá.Ak sú obidve hodnoty rovnaké (t. j. obe **true** alebo obe **false**), výsledkom je **false**.

**XNOR**  
(z anglického *exclusive NOR*, slovensky **„logické rovnocenné“** alebo aj **„negácia výlučného ALEBO“**) je logická operácia, ktorá vracia hodnotu **pravda (true)** iba vtedy, keď **obe** vstupné hodnoty sú rovnaké. Ak sú hodnoty **odlišné** (t. j. jedna **true**, druhá **false**), výsledkom je **false**.

**XAND**  
Pojem XAND nie je štandardnou logickou operáciou ako AND, OR, XOR alebo XNOR. Občas sa vyskytuje ako neformálne alebo chybné označenie pre „negáciu XOR“, no tú správne nazývame XNOR. Pojem XAND nie je štandardnou logickou operáciou ako AND, OR, XOR alebo XNOR. Občas sa vyskytuje ako neformálne alebo chybné označenie pre „negáciu XOR“, no tú správne nazývame XNOR.

**Monkey Testing**  
Pri testovaní mobilných apiek som sa stretla s nástrojom pre náhodné klikanie, ťukanie, scrollovanie, bez plánu. A malo to vcelku prekvapivé výsledky.

**Dropdown**  
Rozbaľovacie menu – používateľské rozhranie, ktoré zobrazuje zoznam možností po kliknutí na pole alebo šípku. Najčastejšie sa používa pri výbere jednej hodnoty zo zoznamu (napr. výber krajiny, roka, kategórie…).

**Checkbox**  
Zaškrtávacie políčko – používateľský prvok, ktorý umožňuje vybrať jednu alebo viacero nezávislých možností. Používa sa tam, kde môže byť vybraných viac hodnôt naraz (napr. „Súhlasím s podmienkami“, „Chcem dostávať newsletter“).

**Label**  
Popisný text, ktorý je priradený k inému prvku používateľského rozhrania – najčastejšie k formulárovému poľu (input, checkbox, radio button…). Pomáha používateľovi pochopiť, čo má do daného poľa zadať alebo čo konkrétny prvok znamená.

**Tooltip**  
Krátky vysvetľujúci text, ktorý sa zobrazí po nabehnutí kurzora myši (alebo zameraní) na prvok – najčastejšie ikonu, tlačidlo alebo pole. Slúži na doplnenie informácií bez zaberania miesta v rozhraní. Môže vysvetliť funkciu, význam alebo upozorniť na pravidlá.

**Sandbox**  
Izolované testovacie prostredie, ktoré simuluje reálne podmienky, ale bez rizika ovplyvnenia produkčných dát alebo systémov. Používa sa na bezpečné experimentovanie, testovanie funkcií, API alebo bezpečnostných scenárov.

**Seed data**  
Vopred pripravené dáta, ktoré sa automaticky nahrajú do databázy pri spustení testovacieho prostredia alebo aplikácie. Slúžia na to, aby mal tester alebo vývojár ihneď k dispozícii niečo, s čím môže pracovať – používateľov, produkty, objednávky, atď.

**Timestamp**  
Časová pečiatka – presný okamih v čase, zaznamenaný vo formáte dátum + čas (napr. 2025-07-07 13:45:00). Používa sa na evidenciu kedy sa niečo stalo: vytvorenie záznamu, zmena, prihlásenie, chyba, transakcia...

**CLI (Command Line Interface)**  
Rozhranie príkazového riadku – spôsob ovládania systému alebo aplikácie pomocou textových príkazov, nie klikania v grafickom rozhraní (GUI). Používa sa najmä vývojármi, administrátormi a testermi na rýchle alebo automatizované operácie.

**Mitigácia**  
Zmiernenie dopadu problému – opatrenie, ktoré nezabráni chybe alebo hrozbe úplne, ale zníži jej vplyv. Používa sa najmä v bezpečnostnom testovaní, rizikovej analýze alebo pri riešení incidentov.

**SDLC (Software Development Life Cycle)**  
Životný cyklus vývoja softvéru – súbor fáz, ktoré vedú od nápadu k hotovej aplikácii. Každá fáza má svoje úlohy, výstupy a zodpovednosti.

Typické fázy SDLC:

* Zber požiadaviek – čo má systém robiť
* Analýza a návrh – ako to bude fungovať a vyzerať
* Implementácia – vývoj kódu
* Testovanie – overenie kvality, funkčnosti, bezpečnosti
* Nasadenie – uvedenie do prevádzky
* Údržba – opravy, aktualizácie, podpora

**Waterfall (vodopádový model)**  
Tradičný spôsob vývoja softvéru, kde sa fázy projektu vykonávajú lineárne – jedna za druhou. Až keď sa dokončí jedna fáza (napr. analýza), môže začať ďalšia (napr. vývoj). Každý krok „padá“ do ďalšieho ako vodopád. Fázy:Zber požiadaviek, Analýza, Návrh, Implementácia, Testovanie, Nasadenie, Údržba

**Agile**  
Prístup k vývoju softvéru, ktorý kladie dôraz na pružnosť, spoluprácu a časté zmeny. Namiesto jedného veľkého plánu sa vyvíja postupne v krátkych cykloch (sprintoch) a tím reaguje na spätnú väzbu počas vývoja.

Tester v Agile:  
– je aktívnou súčasťou tímu od začiatku,  
– testuje priebežne, nielen na konci,  
– spolupracuje s vývojármi, PM aj UX,  
– pomáha písať akceptačné kritériá a user stories,  
– robí exploratívne aj automatizované testy.

**Scrum**  
Najpoužívanejší Agile framework – vývoj prebieha v sprintoch, tím má definované role (Scrum Master, Product Owner, vývojári/testeri) a pravidelné ceremónie.

**Sprint**  
Krátky časový úsek (zvyčajne 1–4 týždne), počas ktorého sa doručí konkrétny výstup – funkcia, oprava, zmena. Sprint má svoj cieľ, plán a retrospektívu.

**Daily (stand-up)**  
Krátke (ideálne 15-minútové) denné stretnutie tímu – každý povie, čo včera robil, čo bude robiť dnes a či niečo blokuje jeho prácu.

**Backlog**  
Zoznam všetkého, čo by produkt mal vedieť – funkcie, opravy, nápady… Prioritizovaný Product Ownerom, tím si z neho ťahá úlohy do sprintu.

**Backlog grooming/refinement**  
Spresňovanie backlogu – tím spolu s PO upresňuje, rozdeľuje, boduje a čistí položky, aby boli pripravené na budúci sprint.

**User story**  
Popis funkčnosti z pohľadu používateľa. Napr. „Ako registrovaný používateľ chcem vidieť históriu objednávok, aby som si mohol niečo dohľadať.“

**Acceptance criteria**  
Podmienky, ktoré musia byť splnené, aby sa story považovala za hotovú. Pomáhajú testerovi určiť, čo overiť.

**Definition of Done (DoD)**  
Dohodnuté kritériá, čo znamená „hotové“. Môže zahŕňať: kód je napísaný, otestovaný, reviewnutý, nasadený a dokumentovaný.

**Sprint review**  
Stretnutie na konci sprintu, kde tím ukazuje stakeholderom, čo vytvoril. Príležitosť na spätnú väzbu.

**Sprint retrospective (retro)**  
Reflexia sprintu – čo išlo dobre, čo nie, čo zlepšiť nabudúce. Cieľ: zlepšovať proces, nie hľadať vinníka.

**Velocity**  
Miera, koľko práce tím zvládne za sprint. Sleduje sa napr. počet story points. Pomáha odhadnúť kapacitu tímu do budúcna.

**AI-under-test**

Testujeme samotnú umelú inteligenciu – teda model, algoritmus alebo systém, ktorý sa správa „inteligentne“. Napr. testujeme, či chatbot odpovedá správne, či ML model klasifikuje e-maily dobre, alebo či odporúčací systém neponúka nezmysly. Tester sa tu zameriava na presnosť, spravodlivosť, stabilitu a chovanie pri rôznych vstupoch.

**AI-assisted testing**

Tester používa AI ako pomocníka pri testovaní – napr. na generovanie testovacích dát, návrh testov, analýzu logov, písanie skriptov alebo odhaľovanie vzorov. AI = testovací kolega (ktorý nepije kávu a píše rýchlejšie než ty). Je to ako mať „copilota“ – ty riadiš, AI navrhuje.

**AI-driven testing**

AI automaticky riadi testovanie – sama analyzuje aplikáciu, rozhoduje, čo testovať, generuje a spúšťa testy, vyhodnocuje výsledky. Cieľ: maximálna automatizácia pomocou AI – najmä pri komplexných systémoch, ktoré sa často menia. Nie vždy ide o úplne samostatné testovanie, ale AI tu hrá hlavnú rolu v rozhodovaní.

Ilustračný obrázok k Testerskému slovníku. Názov Testerský slovník a ikonky kód, checklist, bug, odfajknuté, chemické banky.