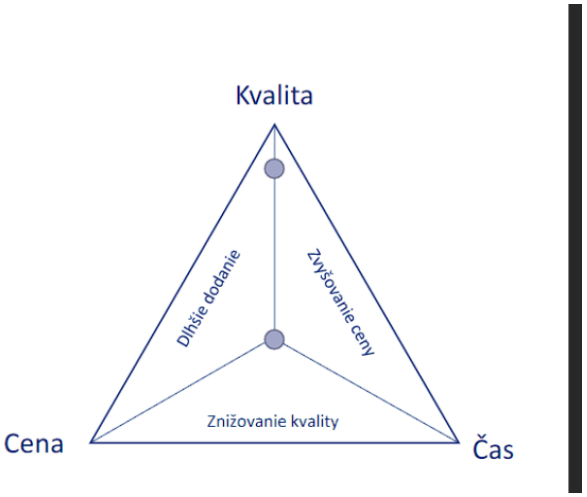
SKÚŠKY ZTS | 2024 + 2025

1.

- **Rozdiel medzi waterfall a agilným vývojom**
  - **Waterfall** – Testovací plán je vytvorený pre uspokojenie požiadaviek, obsahuje zbytočné papierovanie, plán nie je aktuálny ani dodržiavaný, a preto je málo flexibilný.
  - **Agilný prístup** – Progresívne a dynamické plánovanie, veci sa neplánujú v predstihu, plánuje sa na začiatku každej iterácie. Je vysoko flexibilný.
- **Rozdiel medzi verifikáciou a validáciou (napísať aj príklad)**
  - **Verifikácia** – Riešime, či je integrácia alebo kód dobrý. Test je dynamická verifikácia.
  - **Validácia** – Vytvára sa produkt správne? Dôležitá je účasť zákazníka, overenie funkčnosti a poskytnutie služieb. Test je akceptačné testovanie (či softvér robí to, čo má).
- **Čo je to Robot Framework a načo je dobrý**
  - Používa sa pri automatizovanom testovaní. Využíva sa napríklad PyCharm a technológie Python na spúšťanie. Využíva taktiež Selenium knižnicu.
- **Čo je to pokrytie podmienok**
  - Testovacie prípady sa navrhujú za účelom pokrytia každej podmienky. Nemusí to nutne znamenať pokrytie všetkých rozhodnutí.
- **Analýza hraničných hodnôt**
  - Testujeme presne hranicu, číslo pod ňou a číslo nad ňou. (999 , 1000, 1001..)

- **Napísať aspoň 3 techniky testovania podľa špecifikácie**
  - **Analýza hraničných hodnôt** – Definujeme hraničné hodnoty intervalu na oboch stranách.
  - **Rozdelenie do tried ekvivalencie** – Test case má mať špecifický cieľ, na dosiahnutie ktorého má určité vstupy. Testovacie údaje → vektor testov. Vytvoríme intervaly podľa hodnôt a na ich základe vytvoríme triedy ekvivalencie.
  - **Rozhodovacie tabuľky** – Vstupná podmienka, trieda platných vstupov, trieda neplatných vstupov.
  - **Testovanie prechodov medzi stavmi** - Je technika testovania založená na modeli **konečných stavov (state machine)**.

- **Trojuholník kvality**
  - Zachovanie rovnováhy medzi atribútmi – ak sa zmení jedno, ovplyvní to ostatné. Zníženie nákladov sa prejaví na kvalite a dobe dodania.



- **Čo by malo obsahovať nahlásenie bugu/defektu**
  - Čas výskytu, použité údaje, screenshoty. Malo by sa zaoberať iba jedným defektom, byť stručné a presné, obsahovať reprodukčné kroky.

- **3P – Produkt, ľudia, procesy (vysvetliť aj príklad):**
  - **Produkt:** Typ nástroja na testovanie, ktorý je vhodný na konkrétny projekt (napr. výber Selenium na testovanie webových aplikácií).

- **Ľudia:** Tím testerov, ktorí musia akceptovať rozhodnutia, spolupracovať a adaptovať sa na zmeny.
- **Procesy:** Definované metódy a stratégie, ktoré určujú, ako sa testovanie bude vykonávať. Napríklad použitie agilného prístupu v testovacom cykle.

- **Napísať 3 agilné nástroje na testovanie a jeden opísať**
  - **Jira** – Sledovanie bugov, vytváranie reportov.
  - **soapUI** – Podpora architektúry orientovanej na služby REST.
  - **Selenium** – Vysoko rozšírený nástroj, ktorý podporuje rôzne programovacie jazyky.

2.

- **Napíšte aspoň 5 pozícií v testovacom tíme a opíšte ich.**
  1. **Tester** – Vykonáva manuálne alebo automatizované testovanie, overuje správnosť a funkčnosť softvéru podľa požiadaviek.
  2. **Analytik testovania** – Navrhuje testovacie scenáre, identifikuje riziká a pripravuje testovacie dáta na základe požiadaviek.
  3. **Tester automatických testov** – Vyvíja, vykonáva a udržiava automatizované testovacie skripty a reportuje výsledky.
  4. **Vedúci testerov** – Zodpovedá za koordináciu testerov, monitorovanie pokroku testovania a riešenie problémov v rámci testovacieho tímu.
  5. **Manažér testovania** – Definuje testovaciu stratégiu, riadi celý testovací proces, prideľuje zdroje a zabezpečuje komunikáciu medzi tímami.

- **Rozdiel medzi prieskumným testovaním a testovaním na základe skúseností.**
  1. **Prieskumné testovanie** – Systematicky vykonávané ad-hoc testovanie, pri ktorom tester skúma produkt a na základe toho navrhuje testovacie prípady. Ide o interaktívny proces, kde tester dynamicky navrhuje nové testovacie prípady. Tester sa učí používať produkt a potom je tak schopný vytvoriť stratégiu pre vyhodnotenie správania sa produktu – a teda vytvoriť test cases.

2. **Testovanie na základe skúseností** – Tester využíva svoje predchádzajúce skúsenosti, znalosti o podobných systémoch alebo predpoklady o chybách, ktoré sa môžu vyskytovať, aby efektívne vykonal testovanie.

- **Čo obsahuje testovací plán a na čo slúži.**
  - Testovací plán je dokument, ktorého zmyslom je definovať ciele, rámec testovania, spôsob testovania, riziká testovacieho systému, potrebné zdroje, testovacie artefakty, plánované aktivity, harmonogram testovania a metriky. Slúži na zabezpečenie systematického a kontrolovaného testovania.

- **Popíšte automatizované testovanie, techniky a príklady nástrojov, a jeden opíšte detailnejšie.**

  1. **Automatizované testovanie** – Vykonáva rovnaké činnosti ako manuálne testovanie, ale je rýchlejšie, opakovateľné a bezchybné v rámci opakovania. Dokáže rýchlo overovať údaje, vykonávať skripty a efektívne kontrolovať funkčnosť softvéru.

  Typy automatizovaných testov

  1. **regresné testy** - overenie funkčných a nefunkčných požiadaviek
  2. **výkonnostné testovanie** - generovanie záťaže stovky -tisícky používateľov, generovanie záťaže počas 24 hodín
  3. **funkčné testy** - automatické určenie všetkých testovacích prípadov a ich vykonanie, rozsiahly postup testu, ktorý je potrebné bezchybne opakovať
  4. **jednotkové (unit) testy** - sú v princípe vykonávané vždy automatizovane
  5. **bezpečnostné testy** - overovanie odolnosti voči známym a variantným spôsobom napadnutia systému - vykonávanie automatizovaných unit/funkčné testov s cieľom narušenia bezpečnosti
  6. **Príklad nástroja: Robot Framework** – V rámci súborov a syntaxe "robot" si definujeme kroky, ktoré treba vykonať. Podporuje využívanie cyklov, premenných a kontrolu stavov. Umožňuje jednoduché a prehľadné definovanie testovacích prípadov.

- **Vysvetlite základné axiómy testovania.**
  1. **Žiadny netriviálny program nie je možné úplne otestovať.**
  2. **Testovanie slúži na odhaľovanie rizík.**
  3. **Testovanie môže len preukázať prítomnosť defektov, nie ich absenciu.**
  4. **Čím viac defektov nájdeme, tým viac ich pravdepodobne zostáva v produkte.**

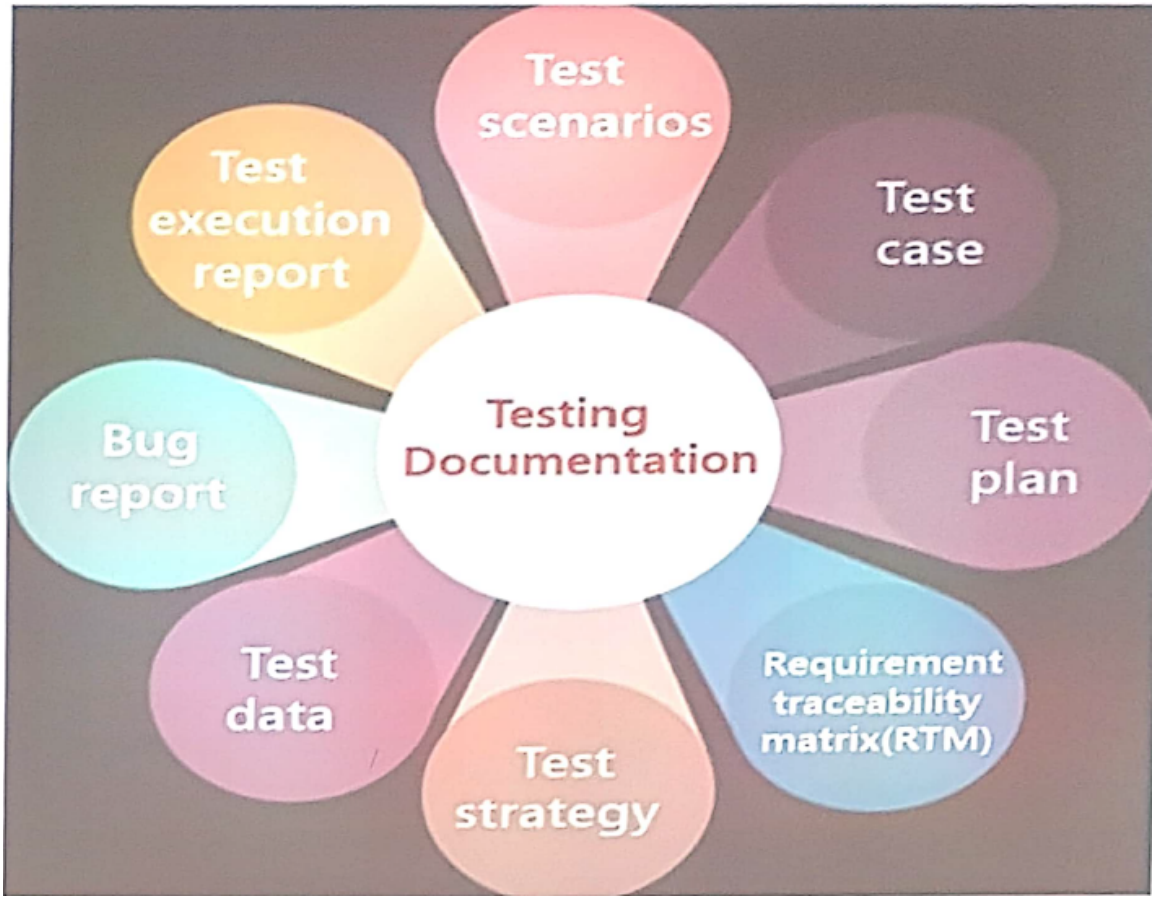
- **Základné techniky manažovania testovania.**
  1. **Zásady testovania** – Definujú filozofiu, hlavné ciele a celkový prístup k testovaniu.
  2. **Stratégia testovania** – Určuje požiadavky a spôsob, akým bude testovanie uskutočnené.
  3. **Hlavný plán testovania** – Zameriava sa na konkrétny projekt a popisuje stratégie a riešenia pre rôzne situácie.
  4. **Plán pre jednotlivé úrovne testovania** – Pri rozsiahlejších projektoch detailne popisuje plány pre každú úroveň testovania.

- **Popíšte testovanie pokrytia príkazov.**
  - Ide o základný typ pokrytia kódu. Testy sa navrhujú tak, aby zabezpečili spustenie všetkých príkazov v kóde aspoň raz, pričom pri 100 % pokrytí dôjde k vykonaniu každého príkazu.

- **Charakteristika dobrého nástroja na testovanie.**
  1. Jednoduché a intuitívne používanie.
  2. Podpora ladenia a hľadania chýb.
  3. Robustná identifikácia objektov.
  4. Podpora testovania v rôznych prostrediach.
  5. Možnosť testovania databáz.
  6. Podpora viacerých platforiem.

- **Pozvaná prednáška GlobalLogic: Testovací dokument obsahuje 8 komponentov, napíšte 4 a opíšte ich.**

- **Test Case (Testovací prípad):**
  - Definuje konkrétny scenár, ktorý má byť otestovaný.
  - Obsahuje: vstupy, očakávané výstupy, kroky na vykonanie testu a predpoklady.
- **Test Scenarios (Testovacie scenáre):**
  - Vyššia úroveň popisu, čo sa má testovať.
  - Príklad: "Overiť funkčnosť prihlasovania."
- **Test Data (Testovacie dáta):**
  - Dáta potrebné na vykonanie testovania, ako napríklad vstupy do formulárov alebo simulované používateľské údaje.
- **Bug Report (Hlásenie chyby):**
  - Dokumentácia každej nájdenej chyby.
  - Obsahuje: názov chyby, popis, kroky na reprodukciu, očakávané a skutočné výsledky, prílohy (napr. screenshoty).



3.

- **Prípady, kedy sa v minulosti netestovanie nevyplatilo**
  1. **Disney: Hra Lion King** – Hra fungovala len na niektorých počítačoch.
  2. **Therac-25 (1985-1987)** – V liečebnom komplexe Therac-25 v pristojí na ožarovanie - po skončení procesu sa zadávali a editovali parametre a po zadaní zostal kurzor na novom riadku. Kontrola prebiehala ale až každých 8 sekúnd. Vzniká nebezpečenstvo pri rýchlej editácii všetkých hodnôt naraz
  3. **Patriot obranný raketový systém (1991)** – V časovači systémových hodín sa akumulovala chyba po 14tich hodinách – >nepresnosť pri mierení. Po viac ako 100 hodinách prevádzky rakety v Saudskej Arábii zlyhali

- **Čo je Postman?**
  - Postman je nástroj, ktorý sa často využíva na testovanie API a rôznych HTTP requestov.
  - **Testovanie API:**
    - Umožňuje testovať REST, SOAP a GraphQL API jednoducho cez užívateľské rozhranie.
    - Môžete posielať rôzne typy HTTP requestov: GET, POST, PUT, DELETE, PATCH, HEAD, OPTIONS atď.
  - **Automatizácia testovania:**
    - Možnosť písať testovacie skripty v jazyku JavaScript priamo v nástroji na kontrolu odpovedí (napr. overenie kódu odpovede, obsahu alebo štruktúry dát).

- **Techniky zisťovania kvality softvéru (SW) + vysvetlenie jednotlivých častí:**
  1. **Statická analýza** – Skúmanie produktu bez jeho spúšťania (napríklad analýza zdrojového kódu).
  2. **Dynamická analýza** – Skúmanie vlastností produktu počas jeho behu. Nejde o testovanie.

3. **Testovanie** – Typická dynamická aktivita, pri ktorej určujeme funkčnosť a celkovú použiteľnosť produktu v nasadení.

- **Čo je test case (všetko k tomu, ako vyzerá):**

Testovací prípad:
Ide o detailný dokument, ktorý popisuje konkrétny scenár na overenie správnej funkčnosti určitej časti softvéru. Testovací prípad stanovuje, aké kroky treba vykonať, aké vstupy použiť a aké výsledky očakávať, aby bolo možné potvrdiť, či softvér funguje podľa špecifikácie.

Ako vyzerá:

1. Má unikátny identifikátor.
2. Obsahuje účel testu alebo jeho zameranie.
3. Špecifikáciu vstupných údajov.
4. Špecifikáciu výstupných hodnôt, vlastností a očakávaného správania.
5. Potreby prostredia na vykonanie testu.
6. Požiadavky na vykonanie testu.
7. Závislosti na iných testovacích prípadoch.

- **Hlásenie defektu (všetko k tomu, ako vyzerá, načrtnúť príklad):**
  - Objavenie konfliktu medzi špecifikáciou a skutočným chovaním
  - za nedostatok sa považuje všetko to, čo znižuje ekonomickú hodnotu produktu
  - pri hlásení defektu tester musí identifikovať či sa jedná o nový defekt, alebo podobný prípadne súvisiaci už s ohlásenými
  - **Proces:**
    - Definujeme, čo nahlasujeme, vrátane času výskytu, použitých údajov, screenshotov a reprodukčných krokov.
    - Hlásenie by sa malo zaoberať iba jedným defektom, malo by byť stručné, presné a zrozumiteľné.
  - **Stavy defektu:**
    - Nový, otvorený, priradený, vyriešený, zatvorený, zamietnutý alebo označený ako duplikát.

- **Ak je defekt duplikát:**
  - Spojí sa s pôvodným hlásením a nahlasuje sa len jedno hlásenie.
- **Definícia defektu:**
  - Defekt je akákoľvek odchýlka od očakávaného správania softvéru.

- **Aké soft skills musí mať tester (GlobalLogic):**
  - **Zvedavý** – Tester musí byť schopný klásť otázky, skúmať detaily a hľadať potenciálne problémy.
  - **Komunikatívny** – Dôležité je efektívne komunikovať v rámci tímu a so zákazníkmi.
  - **Organizovaný** – Schopnosť plánovať svoje aktivity a riadiť čas efektívne.
  - **Kritické myslenie** – Schopnosť analyzovať situácie a nachádzať najefektívnejšie riešenia.
  - **Prispôsobivý** – Rýchlo sa prispôsobiť zmenám a novým technológiám.

- **TTD a DDT – vysvetlenie:**
  - **Test-Driven Development (TDD):**
    - Prístup k vývoju softvéru, pri ktorom sa ešte pred implementáciou píšu testy. Postup:
      1. Napíše sa test na očakávané správanie funkcie či metódy (test zatiaľ zlyhá – RED).
      2. Napíše sa minimálny kód, aby test prešiel (GREEN).
      3. Refaktoruje sa kód, aby bol čitateľný a udržiavateľný.
  - **Data-Driven Testing (DDT):**
  - Výhodné keď treba opakovať testy s rovnakou logikou alebo veľkým objemom údajov
    - Testovacia technika, kde je logika testu oddelená od testovacích dát (napríklad v samostatných súboroch typu CSV, Excel, XML alebo v databáze). Táto technika umožňuje efektívne vykonávať rovnaký testovací scenár s rôznymi vstupmi.

        - Zdroj údajov – databáza, excel, tabuľka...
          • Test načíta vstupné dáta a porovnáva ich s očakávaným výsledkom (riadok po riadku)
          • Niekedy treba vlastný framework

- **Akceptačné testovanie (všetko k tomu):**

  Testovanie zákazníkom / používateľom
  • UAT = user acceptance testing
  • Overuje sa aj akceptovanie produktu ako takého (chýba dokumentácia?)
  • Už sa testuje pri reálnej prevádzke a tester by mal poznať, čo má systém robiť a na čo slúži
  • Treba myslieť aj na normy, štandardy a zákony • -end-to end testovanie – súčasť akceptačných testov

  End to end testovanie:
  • Sledujeme entitu po celú dobú životnosti v systéme (prihlásenie používateľa a jeho práca až po odhlásenie; sledovanie transakcií...)

- **Analýza hraničných hodnôt (zmení to počet test cases?):**
  - Analýza zahŕňa testovanie presne na hranici, číslo pod ňou a číslo nad ňou. Toto prístupy môžu zmeniť počet testovacích prípadov, pretože rozširujú pokrytie hraničných podmienok.

4.

- **Čo to je bug a kde ho môžeme nájsť:**
  - **Bug** je chyba (defekt softvéru), vada alebo nesprávne správanie v softvéri, ktoré spôsobuje, že softvér nefunguje podľa očakávania alebo špecifikácie.
  - **Kde ho môžeme nájsť:**
    - V zdrojovom kóde, počas vývoja.
    - V dokumentácii.
    - Pri integrácii rôznych komponentov alebo modulov.
    - Pri interakcii softvéru s používateľom (napr. chyby v používateľskom rozhraní).

        - V aplikačnej logike, bezpečnosti, výkone, kompatibilite a v iných aspektoch softvéru.
        - **Výskyt bugu:**
            - Systém vykonáva niečo čo by podľa špecifikácie nemal
            - Systém nevykonáva to čo by mal
            - Systém vykonáva niečo čo nie je definované
            - Systém nevykonáva niečo čo nebolo špecifikované ale malo byť
        - **Závažnosť bugu:**
            - Kritická
            - Vysoká
            - Stredná
            - Nízka

- **Čo to je testovací prípad, čo obsahuje, ako sa nahlasuje a aké údaje zahŕňa:**
    - **Testovací prípad** je špecifický scenár navrhnutý na overenie, či softvér spĺňa stanovené požiadavky.
    - **Obsah testovacieho prípadu:**
        1. Unikátny identifikátor.
        2. Účel alebo zameranie testu.
        3. Špecifikácia vstupných údajov.
        4. Očakávané výstupy a správanie.
        5. Potreby testovacieho prostredia.
        6. Kroky na vykonanie testu.
        7. Závislosti na iných testovacích prípadoch.
    - **Ako sa nahlasuje:**
        - Testovací prípad sa nahlasuje prostredníctvom nástrojov na správu testovania (napr. JIRA, TestRail).
        - **Údaje potrebné pri hlásení:**

        - Názov prípadu, ID, opis problému, kroky reprodukcie, výsledok (očakávaný verzus skutočný), závažnosť a priorita.

- **Čo je integračné testovanie, na aké 2 typy sa delí a ich vysvetlenie:**
    - **Integračné testovanie** je typ testovania, ktorý sa vykonáva na overenie správnej spolupráce medzi modulmi alebo komponentmi systému.
    - **Delenie integračného testovania:**
        1. **Veľký tresk** – všetky moduly integrujeme naraz ale problém sa odhalí ťažšie
        2. **Prístup z vrchu** -podľa hier. štruktúry postupne pridávame moduly, ktoré zaistia čiastočné fungovanie pri určitom počte
        3. **Prístup z dola** – z najnižších modulov hore – systém funguje až kým nebude integrovaný ten najvyšší
        4. **Kombinovaný / sendvičový** – kombinuje sa 2+3 → zvrchu sa aplikuje na hlavné moduly, zdola na často využívané a postupne sa pridajú ostatné. Takto sa dajú testovať hlavné a najnižšie moduly súčasne

---

- **Pokrytie hrán – vysvetlenie:**
    - **Definícia:**
        - Pokrytie hrán je technika testovania založená na grafoch reprezentujúcich tok programu. Každá hrana reprezentuje tok medzi dvomi bodmi v programe.
    - **Cieľ:**
        - Vyhodnotiť každú podmienku v programe.
        - Pri každom rozhodnutí sa musia otestovať všetky možné vstupy.
    - **Výhody:**
        - Pomáha zabezpečiť, že všetky možné toky programu sú otestované, čím minimalizuje riziko neočakávaného správania.

- **Vysvetlenie nástroja Selenium:**
    - **Čo to je Selenium:**

        - Selenium je open-source testovací nástroj používaný na automatizované testovanie webových aplikácií.
    - **Časti Selenium:**
        1. **Selenium IDE:**
            - Používa sa ako rozšírenie v prehliadači na jednoduché vytváranie a spúšťanie automatizovaných testov.
        2. **Selenium WebDriver:**
            - Používa sa na písanie a spúšťanie automatizovaných testov pomocou programovacích jazykov (napr. Python, Java).
            - Umožňuje testovať zložité interakcie a scenáre.
        3. **Selenium Grid:**
            - Umožňuje paralelizáciu a škálovanie testovania na viacerých strojoch a prehliadačoch naraz.
    - **Čo s ním dokážeme robiť:**
        - Automatizovať testovanie webových aplikácií.
        - Simulovať interakcie používateľov s webovou aplikáciou (napr. kliknutie na tlačidlo, vyplnenie formulára).
        - Spúšťať testy na rôznych prehliadačoch a platformách.

- **Čo je testovacie orakulum:**
    - Testovacie orakulum je v softvérovom testovaní akýkoľvek zdroj, ktorý jednoznačne určuje očakávaný výsledok testu. Môže ísť o špecifikácie, požiadavky, výpočtové modely alebo iné zdroje.

- **Nástroje výkonnostného testovania:**
    - Príklady nástrojov: **Apache JMeter, Gatling, Locust**.
    - Typy výkonnostných testov:
        1. **Záťažové testy:** Zameriavajú sa na testovanie softvéru pod dlhodobou záťažou, aby sa overila jeho stabilita.
        2. **Stres testy:** Skúmajú správanie softvéru pri extrémnej záťaži, aby sa určili jeho limity.

3. **Testy škálovateľnosti:** Overujú, ako softvér reaguje na budúci nárast záťaže (napríklad zvýšenie počtu používateľov).

- **Manuálne vs. automatizované testovanie:**
  - **Manuálne testovanie:**
    - Tester vykonáva všetky testy manuálne podľa testovacích scenárov.
    - Vhodné pre prípady, ktoré vyžadujú ľudské pozorovanie (napr. UI testy).
    - Nevýhody: časovo náročné, náchylné na chyby pri opakovanej práci.
  - **Automatizované testovanie:**
    - Používajú sa nástroje na vykonávanie testov automaticky.
    - Vhodné pre opakované testy, ako sú regresné testy.
    - Výhody: rýchlosť, presnosť, efektívnosť.
    - Nevýhody: Vyžaduje vyššie počiatočné náklady a technické znalosti.

- **Testovanie komponentov – čo to je:**
  - Testovanie komponentov (unit testing) sa zameriava na overenie správnej funkčnosti jednotlivých častí (komponentov) softvéru.
    1. Testuje ho zvyčajne samotný programátor.
    2. Komponenty (units) sú základné stavebné bloky systému.
    3. Každý komponent sa testuje samostatne.
    4. Vývoj môže byť riadený testami, čo znamená, že sa píšu testy ešte pred samotným kódom (napr. pri TDD).

- **Testovanie hrán vs. testovanie príkazov:**
  - **Statement Coverage (pokrytie príkazov):**
    - Testovanie sa zameriava na to, aby boli spustené všetky jednotlivé príkazy v kóde aspoň raz.
  - **Branch Coverage (pokrytie hrán):**

        - Testovanie sa zameriava na overenie všetkých možných vetiev (cestíčiek) v kóde, aby sa zabezpečilo pokrytie všetkých rozhodnutí.

- **GlobalLogic – Testovanie funkčných a nefunkčných požiadaviek:**
    - **Funkčné testy (testujú funkčnosť softvéru):**
        - **Unit test:** Testovanie jednotlivých komponentov.
        - **Smoke test:** Rýchle overenie základnej funkčnosti softvéru.
        - **Regresný test:** Overenie, že nové zmeny v kóde nespôsobili problémy v už existujúcej funkcionalite.
    - **Nefunkčné (mimofunkčné) testy (testujú vlastnosti softvéru):**
        - **Performance test:** Skúmanie výkonnosti softvéru pri rôznych záťažiach.
        - **Load test:** Overovanie správania systému pod normálnou záťažou.
        - **Stress test:** Testovanie systému pod extrémnou záťažou, aby sa zistili jeho limity.

- **Čo musí obsahovať Bug report:**
    - **Bug report** je dokument, ktorý slúži na nahlásenie chyby v softvéri. Mal by obsahovať:
        1. **Identifikátor chyby (ID):** Unikátny kód chyby.
        2. **Názov chyby:** Stručný popis problému.
        3. **Opis chyby:** Detailné vysvetlenie chyby, vrátane očakávaného a skutočného správania softvéru.
        4. **Kroky na reprodukciu:** Postup, ako chybu znovu vytvoriť.
        5. **Priorita a závažnosť:** Stanovenie, aký vplyv má chyba na systém.
        6. **Prílohy:** Screenshoty, logy alebo iné relevantné súbory.
        7. **Testovacie prostredie:** Informácie o operačnom systéme, verzii softvéru, hardvére atď.
        8. **Stav chyby:** Napr. nový, otvorený, priradený, vyriešený, zamietnutý alebo uzatvorený.

SKÚŠKY 2025

1. **Čo to je akceptačné testovanie a jeho význam - popis, rozdelenie testov, príklady použitia.**
2. **Ako definujeme prevádzkovú spoľahlivosť a uveďte príklad.**
3. **Čo je to cyklomatická zložitosť a načo sa dá použiť.**
4. **Čo je to prieskumné testovanie a kedy je ho možné aplikovať.**
5. **Z čoho sa skladá testovací plán a stručne ho popíšte.**
6. **Navrhnite a popíšte techniku hlásenia defektu - aké údaje sa majú zaznamenať, spôsob zápisu, načrtnite formu reportu, uveďte príklad na jednoduchom probléme.**

1. **Smoke a sanity**
2. **Regresné testovanie, význam príklady**
3. **Životný cyklus defektu a príklad**
4. **Role a popísať**
5. **Trojuholník kvality**
6. **End to End testing**

1. **Popísať a vysvetliť 3P**
2. **Testovací plán ako vyzerá a čo to je**
3. **Integračné testovanie**
4. **Popísať nástroj Selenium**
5. **Výkonnostné testy popísať a príklad**
6. **Rozdiely medzi “manuálnym” a automatizovaným testovaním**
7. **Analýza hraničných hodnôt**
8. **Čo je to bug a graf ceny opravy bugu**

ÚSTNA - individuálna

- Smoke a sanity testy, na čo sú a aký je medzi nimi rozdiel.

- popísať nástroje na testovanie (Selenium IDE, Robot Framework...)
- 4 druhy metód testovania (unit testing,...)
- Ako sa delí integračné testovanie (Veľký tresk,...)
- Manažment testovania
- Akceptačné testovanie (všetko o tom)
- Role v tíme - ako presne fungujú
- Výkonnostné testovanie (popísať nejak metriky)
- Testovací prípad dokumentácia, čo obsahuje
- Nástroje na výkonnostné testovanie
