ŽIVOTNI CIKLUS TESTIRANJA I OSNOVNI POJMOVI

Ideja: testiranje nije samo pokretanje programa. Ono se planira, priprema, izvršava i završava analizom rezultata.

1. Scrum i Kanban - kratko

Scrum organizuje rad u kratkim ciklusima - Sprintovima. Kanban tabla prikazuje gde se posao trenutno nalazi.

PRODUCT BACKLOG → SPRINT BACKLOG → IN PROGRESS → TESTING → DONE

2. Životni ciklus testiranja - STLC

PLANIRANJE → ANALIZA I DIZAJN → PRIPREMA → IZVRŠAVANJE → REZULTATI I IZVEŠTAJ → ZAVRŠETAK

Izvršavanje testova je samo jedan deo procesa testiranja.

3. Ko učestvuje?

- Test Manager / Test Lead - organizuje i prati testiranje.
- Tester / QA - priprema i izvršava testove, beleži rezultate i prijavljuje defekte.
- Automation Tester - automatizuje testove.
- Developer - ispravlja defekte i sarađuje u proveri.

Kvalitet je odgovornost celog tima.

4. Greška, defekt i otkaz

Greška (engl. error) jeste rezultat ljudske aktivnosti, bilo za vreme specifikacije zahteva ili tokom pisanja programa. Na primer, neka funkcionalnost može biti pogrešno specificirana, što će kasnije dovesti do pogrešne implementacije, ili se može napraviti neka greška u kodiranju, koja će dovesti do neispravnog rada programa. Posledica greške se naziva mana, defekt ili bug – programu nešto nedostaje, ili ima funkciju koja se ne ponaša ispravno. Otkaz (engl. failure) nastaje kada sistem nije u mogućnosti da obavi funkciju koju korisnik od njega zahteva, jer se aktivira i izvršava defektni kod.

5. Osnovni pojmovi testiranja

Test scenario

Test scenario opisuje šta želimo da proverimo u aplikaciji. On predstavlja širu situaciju ili funkcionalnost koju treba testirati i ne sadrži detaljne korake.

Primer: Proveriti rezervaciju obroka u aplikaciji Školska kantina.
To je širok zadatak. Još uvek nismo rekli kako ćemo proveriti rezervaciju niti koje ćemo podatke koristiti.

Test slučaj (Test Case)

Test slučaj predstavlja konkretnu proveru. Njime preciziramo početne uslove, korake koje tester izvršava, podatke koje koristi i rezultat koji očekuje.

TC01 – Uspešna rezervacija obroka
Preduslov: učenik je prijavljen, dostupno je 5 porcija pizze.
Korak: učenik izabere „Pizza“ i klikne Rezerviši.
Očekivani rezultat: rezervacija je uspešna i broj porcija se smanjuje sa 5 na 4.

Jedan test scenario može imati više test slučajeva.

Test podaci (Test Data)

Test podaci su konkretne vrednosti koje koristimo prilikom izvršavanja testa.

Primer:
Učenik: Marko Marković
Obrok: Pizza
Broj dostupnih porcija: 5

Za drugi test možemo promeniti samo podatak na 0 porcija i proveriti da li će aplikacija pravilno zabraniti rezervaciju.

Skup testova (Test Suite)

Test Suite predstavlja grupu testova koji pripadaju istoj funkcionalnosti ili koje želimo zajedno da izvršimo.

TS01 – Testiranje rezervacije obroka
TC01 – rezervacija kada ima 5 porcija
TC02 – rezervacija poslednje porcije
TC03 – pokušaj rezervacije kada ima 0 porcija
TC04 – učenik već ima rezervaciju
TC05 – otkazivanje rezervacije

Pokretač testova (Test Runner)

Kada imamo veliki broj automatizovanih testova, ne moramo svaki test pojedinačno da pokrećemo. Test Runner je komponenta ili alat koji pokreće testove i prikuplja rezultate njihovog izvršavanja.

Primer: Imamo 50 automatizovanih testova za aplikaciju Školska kantina. Test Runner ih pokrene i na kraju dobijemo informaciju koji testovi su prošli, a koji nisu.

Rezultat testa (Test Result)

Kada izvršimo test, dobijeni stvarni rezultat upoređujemo sa očekivanim rezultatom.

Primer 1: Očekivano: 4 porcije. Stvarno: 4 porcije. → PASSED – test je prošao.

Primer 2: Očekivano: 4 porcije. Stvarno: 5 porcija. → FAILED – test nije prošao.

Test koji nije prošao ukazuje da postoji problem koji treba analizirati; sam FAILED još ne govori automatski šta je tačan uzrok.

6. AI kao test asistent

Prvo sami predložimo testove. Zatim pitamo AI da pronađe još slučajeva. Na kraju proveravamo: Da li AI predlog zaista proizlazi iz zahteva?

Pravilo: AI predlog nije automatski tačan - QA tim ga proverava.

Za proveru - praktičan zadatak

Aplikacija „Školska kantina - Rezervacija obroka“

Pokrenite dobijenu aplikaciju i pažljivo pročitajte pravila sistema prikazana na vrhu stranice. Ne pretpostavljajte unapred da aplikacija radi ispravno, ali nemojte ni polaziti od toga da je svaka funkcija neispravna.

Vaš zadatak je da:

- Napišete jedan test scenario - navedite šta želite da proverite.
- Za taj scenario osmislite najmanje 3 test slučaja.
- Za svaki test slučaj navedite test podatke i očekivani rezultat.
- Tek nakon toga izvršite testove u aplikaciji i zapišite stvarni rezultat.
- Za svaki test odredite status: PASSED ili FAILED.
- Ako je neki test FAILED, napišite kratko koje pravilo sistema nije ispunjeno.

Važno: Test slučaj osmislite pre izvršavanja. Cilj nije da nasumično tražite greške, već da proverite da li se sistem ponaša u skladu sa zadatim pravilima.

Literatura

- Testiranje softvera, Univerzitet Singidunum, 2018.
- ISO/IEC/IEEE 29119-1:2022 - Software testing: General concepts.
- ISO/IEC/IEEE 29119-2:2021 - Software testing: Test processes.
- Schwaber, K. & Sutherland, J. (2020), The Scrum Guide.