**Tema pentru acasă 1[​](https://textbook.edu.goit.global/lms-qa-homework/ro/docs/ver1.0/hw-02/" \l "tema-pentru-acas%C4%83-2" \o "Direct link to heading)**

1. Explică etapele Software development life cycle (SDLC). Trebuie să explici în linii mari ceea ce se întâmplă la nivel de echipă în fiecare etapă. Etapele care trebuie explicate sunt:
   * Requirement analysis (Planning)
   * Design
   * Development
   * Testing
   * Deployment

**Requirement analysis (Planning)**

* Obținerea de la client a informațiilor necesare (cerințe);
* Sistematizarea cerințelor;
* Construirea unei relații;
* Documentare;
* Definirea problemelor, obiectivelor și resurselor (cum ar fi personalul și costurile);
* Explorarea posibilităților de soluții alternative;
* Analizarea cum să faci un produs mai bun decât al concurenților.

**Design**

* Elaborarea arhitecturii;
* Definirea elementelor, componentelor, modulelor, tipurilor de date;
* Dezvoltarea design-ului produsului.

**Development**

* Scriere cod (front și back).

**Testing**

* Verificare;
* Validare;
* Alte lucruri bune ale vieții.

**Deployment**

* Configurarea mediului;
* Instalarea mediului;
* Suport.

1. Pentru fiecare etapă din SDLC explică ce activități întreprinde QA-ul.
   * Requirement analysis (Planing)

- colaborarea cu business analyst pentru efectuarea unui studiu de fezabilitate a cerințelor, analizarea riscurilor posibile, crearea planurilor de testare și construirea unei strategii pentru abordarea testării QA (fiecare proiect necesită o abordare individuală datorită unicității sale), ce tipuri de testare se vor utiliza, etc.

* + Design

- revizuirea design-ului, verificarea posibilității de testare a acestuia, verificare dacă arhitectura design-ului întrunește toate cerințele funcționale și nefuncționale;

- specialiștii QA fac Data Flow Diagram (DFD) împreună cu design-erii UI/UX și o documentează;

- la final, inginerii QA testează design-ul după finalizarea acestuia pentru a imita comportamentul utilizatorului final.

* + Development

- se poate efectua asigurarea calității la dezvoltarea produsului odată ce software-ul este creat sau e posibil să se utilizeze abordarea TDD — dezvoltarea bazată pe teste. TDD înseamnă că software-ul este supus unui proces de testare în timpul dezvoltării după fiecare repetare.

- creează ghiduri și manuale de utilizator pentru produs pentru a le livra utilizatorului final;

- elaborează documentația de testare pentru a se asigura că toate defectele au fost detectate și că toate s-au rezolvat.

* + Deployment

- release către client, dar se oferă în continuare mentenanță, suport, se fac conturi pentru acesta, se fac îmbunătățiri solicitate, etc.

1. Avem urmatoarele environments: dev, test, staging, production. Descrie activitățile și tipurile de testare folosite pe aceste medii. Gândește-te unde se execută unit tests, testarea manuală funcțională, regression, smoke etc. Trebuie să explici ce fel de testare, unde și de ce se execută pe acel environment.

|  |  |  |
| --- | --- | --- |
| **Mediu de testare** | **Tip de testare** | **De ce aici** |
| Development | Teste preliminare: unit testing, integration testing etc | Acest mediu constă dintr-un server partajat de obicei de mai mulți developeri care muncesc împreună la același proiect. De aceea, când sunt implicați mai mulți developeri, un developer va avea o copie a codului sursă pe propria mașină și apoi modificările sunt trimise către repozitoriu, iar mai apoi integrate în trunchi sau ramură (branch). |
| Se verifică funcționalitatea codului nou sau modificat prin tehnici automate și neautomate (unit tests sunt scrise de programator).  Se poate fie efectua un test după un altul fie se pot crea mai multe medii de testare care permit efectuarea testării în paralel, în funcție de gradul de complexitate al mediului nostru de testare. Se verifică compatibillitatea dintre codul vechi și nou, performanța sistemului și așa mai departe. |
| Testing | Functional manual testing | Testarea funcțională manuală implică ca testerii să execute cazuri de testare fără a utiliza instrumente de automatizare a testelor sau scripturi de testare. În schimb, ei interacționează manual cu aplicația software pentru a identifica și evalua funcționalitatea acesteia. |
| Staging | Regression testing  (Testele de regresie reprezintă o suită de teste prin care se testează toată aplicația. Se iau test case-urile pe care le-am făcut și se parcurg. Se fac înainte de deployment.) | Pentru a verifica cum funcționează întreaga aplicație într-un mediu similar cu producția. |
|  | Smoke testing  Ad hoc testing | Când programatorul implementează o funcționalitate nouă, ne spune că e gata și o putem testa. Trebuie să vedem că se poate deschide aplicația, că merg butoanele, că ne putem loga, etc. |
|  | Sanity testing | Un tip de testare îngustă care este suficientă pentru a demonstra că o anumită funcție acționează în conformitate cu cerințele prevăzute în specificația cerințelor. Este folosită pentru a determina starea de sănătate a unei anumite părți a aplicației în urma configurațiilor făcute în aceasta sau în mediu. |
|  | Performance and load testing | Test ce simulează comportamentul utilizatorului și sarcina pentru a se asigura că funcționează corespunzător. |
|  | System testing | Pentru a se asigura că fiecare component sau serviciu dintr-un sistem funcționează corespunzător. |
|  | Alpha testing (usability, acceptance) | Alpha testing evaluează funcționalitatea performanței unui produs într-un mediu de prelansare controlat. Implică verificarea caracteristicilor individuale și a diagramelor utilizatorului pentru a scoate la iveală potentiale defecte care trebuie remediate. |
|  | Beta testing | Beta testing se află între faza de testare alfa și lansarea produsului și oferă posibilitatea de a primi un feedback de la utilizator. Spre deosebire de natura clară a testului alfa, testul beta este asemenea unei cutii închise: participanții implicați nu cunosc structura sau codul sistemului, la fel ca și utilizatorii finali ai produsului. |
|  | Penetration tests | Testarea aplicației în ceea ce privește vulnerabilitatea |
| Production | Feature flagging  Disaster recovery tests | Mediul cel mai puțin predispus la schimbări, cu teste minime pentru a asigura o funcționare stabilă și sigură.  Pentru a se asigura că aplicația funcționează acceptabil chiar și în cazul apariției unei erori la acest nivel. |