1. **Explică pe scurt fiecare nivel de testare, având ca referință testing pyramid.**

### ****Unit Test (Testarea unitară)****

Testarea unitară servește ca fundație a piramidei de testare. Testele unitare au un domeniu de aplicare limitat și asigură că unitățile de cod izolate funcționează conform așteptărilor.

**Integration Test (Testul de integrare)**

Testul de integrare este al doilea pas al piramidei de testare a software-ului. Testele unitare verifică porțiuni mici ale unei baze de cod. Echipa de dezvoltatori ar trebui să efectueze teste de integrare pentru a verifica interacțiunea acestui cod cu alte coduri din aplicație, care cuprind, în general, întregul software. Aceste teste verifică modul în care un cod interacționează cu componentele externe. Aceste componente pot include servicii externe (API-uri), baze de date, servicii web și elemente similare. Testele de integrare asigură că software-ul comunică eficient și preia date exacte pentru a funcționa conform așteptărilor.

Testele de integrare sunt în general mai lente decât testele unitare din cauza necesității interacțiunii cu surse externe. De asemenea, necesită un mediu de pre-producție pentru a rula.

**End-to-end Testing**

Testarea end-to-end verifică cea mai mare parte a codului (întreaga aplicație). Utilizează un mediu de testare și date pentru a simula funcționalitatea din lumea reală. Este cel mai scump de întreținut și are cel mai lent ritm de operare. Deoarece acest test verifică aplicațiile asamblate, este și cel mai dificil de identificat probleme.

Testarea end-to-end este punctul culminant al piramidei de automatizare a testelor, deoarece poate fi faza cea mai consumatoare de timp a procesului de testare. Testarea end-to-end este de obicei fragilă și poate avea dependențe externe nesigure, cum ar fi testele de integrare.

Testarea end-to-end necesită ca dispozitivele reale să semene mai mult cu aplicația de pe smartphone-ul unui utilizator final, deoarece utilizatorii reali raportează probleme pe dispozitivele reale și nu pe cele virtuale.

1. **Să presupunem că avem o aplicație bancară în dezvoltare și business-ul decide să implementeze o bară de căutare (search) în aplicație. Este creat un task de dezvoltarea a acestei funcții, iar ție îți este atribuit task-ul de a o testa. Descrie procesul de la A la Z, acoperind aspecte precum scrierea testelor, testarea efectivă, închiderea task-ului și ce consideri că trebuie sa fie îndeplinit pentru a finaliza task-ul**.

Task-ul este pus în progress,începem prin a citi documentaţia aflată în document requirments, dacă există neclarităţi punem întrebări,continuăm prin a scrie testele manuale, astfel încât să începem rularea testelor scrise anterior, prin urmarea paşilor din test cases. Pentru a avea o testare completă şi pentru a finaliza task-ul toate testele manuale ar trebui sa fie ‘‘passed‘‘, daca acest lucru nu se întâmplă , şi avem teste ‘‘failed‘‘ sau ‘‘retest‘‘, continuăm prin a scrie bug-uri şi le trimitem developer-ului pentru fixare, iar cand bug-urile au fost fixate, finalizăm task-ul , prin rularea aplicaţiei.

**3.Care este avantajul Unit testing-ului față de End-to-end/system testing?**

Testarea unitară se referă la scrierea unor bucăți de cod, denumite cod de testare, care validează codul de producție.

Unit testing in comparaţie cu End-to-end /system testing următoarele avantaje:

Unit testing-ul vizează cele mai mici părți izolate ale software-ului, iar end to end evaluează sistemul ca un întreg integrat.

Unit testing-ul se concentrează pe anumite unități sau componente în timp ce End-to-end testing subliniază călătoriile complete ale utilizatorului, de la început până la sfârșit.

Unit testing-ul este rapid și eficient datorită domeniului de aplicare limitat, End-to-end Testing consumă mai mult timp din cauza naturii sale cuprinzătoare.

Unit testing-ul în general, este simplu, evaluează funcționalități izolate, End-to-end Testing pe de altă parte este complex datorită emulării scenariilor și interacțiunilor din lumea reală.

Unit testing-ul funcționează izolat, liber de influențe externe iar End-to-end testing ia în considerare întregul sistem, inclusiv toate dependențele și interacțiunile.

**4.Care este avantajul End-to-end/system testing-ului față de Unit testing?**

Testarea end-to-end, cunoscută și sub denumirea de testare e2e, este o abordare de testare care verifică ciclul de viață al software-ului de la început până la sfârșit.

End-to-end/system testing are următoarele avantaje comparativ cu unit testing:

Testarea end-to-end este o formă unică de testare a software-ului care examinează întreaga bucată de software de la început până la sfârșit și modul în care se integrează cu interfețele externe. În schimb, testarea unitară implică testarea fiecărei componente software distincte în mod independent.

În timpul testării end-to-end, testerul examinează programul software din punctul de vedere al utilizatorului. În testarea unitară, dezvoltatorul confirmă funcționalitatea bucăților individuale de cod sursă.

Aceste două tipuri de teste sunt foarte diferite unul de celălalt. Testele unitare încearcă să izoleze obiectul sau unitatea testată. Aceasta înseamnă că încearcă să fie cât mai diferiți în ceea ce privește testarea unei bucăți de cod, minimizând în același timp interacțiunile cu alte bucăți de cod.

Testele E2E au o abordare foarte diferită. Testarea E2e are o abordare de dezvoltare bazată pe comportament. În loc să testeze bucăți individuale de cod, ei testează experiența finală reală pe care o vede un utilizator când folosește aplicația.

1. **Ai următoarele funcționalități în aplicație:**

**calcul rată;**

**postare incident în aplicație (erori de plăți, de funcționalitate, de afișare a aplicației);**

**toate câmpurile din pagina de înregistrare sunt obligatorii;**

**plăți IBAN;**

**investiții prin aplicația bancară.**

**Stabilește ce nivele de testare se potrivesc pentru fiecare funcționalitate și argumentează alegerea făcută. O funcționalitate poate avea mai multe niveluri de testare (unit, integration, end-to-end).**

Pentru calcul ratei as alege **unit testing** , deoarece nu cuprinde multe elemente şi se poate testa cu uşurinţă.

Postare incident în aplicație (erori de plăți, de funcționalitate, de afișare a aplicației), aici as alege **integration testing,**  deoarece avem de testat mai multe componenete, şi le putem face simultan.

Toate câmpurile din pagina de înregistrare sunt obligatorii, aş alege **unit testing** ,deoarece nu este foarte complex şi putem testa fiecare componentă în parte.

Plăți IBAN, aş putea face aici **unit testing ,**deoarece datorită importanţei , ar putea crea probleme în rândul utilizatorilor, dacă nu ar funcţiona optim.

Investiții prin aplicația bancară, as aplica aici **end-to-end** , pentru a descoperi eventuale erori ,odată ce tot procesul de testare este gata.