Zaynab Alsalih

Här följer två kontexter som ska appliceras på inlämningen:

● Kontext 1 - Ett backendsystem som består av en rad olika API:er, som enda gränssnittet. Där det inte finns traditionella slutanvändare. Du arbetar som en del av teamet med testning och med CI-delar.

● Kontext 2 - En lösning som finns på webben med ett antal underliggande backendsystem. Webbgränssnittet är det huvudsakliga fokuset för användare. Du jobbar som en acceptanstestare utanför teamet som gör lösning, du har inte någon direkt kontakt med teamets medlemmar eller interna system.

För Betyg G ska inlämningen innehålla:

● Förklara syfte och begränsningar för de testverktyg vi gått igenom på kursen (d.v.s SoapUI, Selenium, Cucumber och Jenkins), samt resonera kring varför eller varför de inte skulle vara lämpliga att använda för respektive kontext. (1-2 sidor)

SoapUI är ett verktyg för att testa webbtjänster; dessa kan vara SOAP Web Services såväl som RESTful Web Services eller HTTP-baserade tjänster. SoapUI är ett Open Source och helt gratis verktyg. man kan göra funktionstestning, prestandatestning, interoperabilitetstestning, regressionstestning och mycket mer. Dess funktionalitet omfattar webbtjänstinspektion, anrop, utveckling, simulering. Kräver ingen programmeringskunskap för att komma igång. Begränsning är att man behöver hitta ApI dokument för websidor som man kan inte ha i all webplatser. Jag tycker att man kan änvända den verktyg på båda två kontext om man kan hitta API dokumentation för projekt.

Selenium:det är bra för kontext 2 för att man kan prova och testa webbgränssnittet utan att bli en del av teams eller man behöver inte att hitta dokumentation av projekt. Selenium är ett automatiseringstestverktyg som används för att utföra funktionstestet på webbapplikationer. Selenium är ett JavaScript-verktyg som används för att interagera med webbsidor för att testa funktionaliteten och ta bort den repetitiva manuella testansträngningen för programmeraren eller testaren. testfallen kan skrivas på valfritt språk som Java, C #, Python, etc. på vad språkprogrammeraren är bekväm med och det kan också köras på vilken plattform som helst.Begränsninger tester som interagerar med applikationen via det grafiska användargränssnittet (GUI) är väldigt känsliga för förändringar. Därför är det ingen bra ide att automatisera tester mot en applikation som ännu inte är stabil. Att den inte är stabil betyder att det kommer att förändras, modifieras eller rättas, vilket leder till att testerna kommer sluta fungera.

**Cucumber** Testare kan med hjälp av ramverket läsa och implementera tester utan några krav på programmeringskunskap. Cucumber är Open-source och är ett av de mest populära ramverken för att skriva tester.Med rätt hantering av sina scenarior i Cucumber får man:Design – Vägledning med acceptanskriterier / Dokumentation – Beskrivning över “vad hanterar systemet”/ Regressionstester – Med ett Continuous Integration-verktyg kan dessa scenarior köras varje natt eller t.om. varje gång en utvecklare gör en förändring.

Jenkins: det är bra för kontext 1 för att man kan dela sitt jobb med teams i CI pipeline . Jenkins underlättar kontinuerlig integration och kontinuerlig leverans. Den stöder versionskontrollverktyg som Git. Jenkines hjälper till att automatisera de delar av mjukvaruutveckling som är relaterade till konstruktion, testning och driftsättning.

● Beskriv processen för Continuous Integration, samt resonera om dess styrkor och svagheter, behövs inte för varje kontext. (½ - 1 sida) ○ OBS! Denna fråga ersätts av fördjupningen i VG, om man väljer att göra den)

Utvecklare jobbar ofta mot flera brancher, där olika projekt eller delar av ett projekt har sin egen branch. Till sist läggs flera brancher ihop till en produkt eller tjänst. Inom Continuous Integration (CI), eller kontinuerlig integration, använder utvecklingsteamet versionshantering och integrerar sitt arbete frekvent till ett och samma ställe. Genom att dela upp koden i flera brancher, eller delar, kan man genomföra snabbare.

Varje kodändring byggs och verifieras sedan med hjälp av automatiserade tester för att upptäcka fel så tidigt som möjlig. På så vis kan man snabb och tidig få en signal på om något i koden är fel eller behöver göras om.

Eftersom kontinuerlig integration handlar om att själva byggandet och testningen som sker före leverans, är det också en förutsättning för den kontinuerliga leveransen. Där den kontinuerliga integrationen slutar tar alltså den kontinuerliga leveransen vid. Bägge spelar en viktig roll för att ta sig framåt i utvecklings- och leverans processen.

● Beskriv DevOps samt hur testning ser ut i denna process (½ - 1 sida), behövs inte för varje kontext

DevOps är metoden innebär att utvecklare och drifttekniker samarbetar och tillämpar gemensamma principer för att underlätta och effektivisera hela produktion- och leveransflödet. Så det är ett verktyg som man kan på ett snabbare och smidigare sätt bygga, testa och kontinuerligt lansera i ett automatiserat flöde och att man kan gå tillbaka till en tidigare version om något fel upptäcks efter en release. Man kunna säga att det är en full livscykelhantering av en applikation - ett sätt att mer strukturerat och gemensamt kunna utveckla en produkt/tjänsts mjukvara eller system.

Hur testning ser ut i denna process?

* **Plan** – planera och definiera krav och tekniska specifikationer
* **Create** – utveckla och bygga, sätta upp miljöer och förbereda automatisering
* **Verify** – verifiera och testa
* **Package** – versionshantering
* **Release** – processer för godkännande, ändringshantering och lanseringsautomatisering
* **Konfiguration** – konfigurering och hantering av infrastruktur, verktyg för infrastruktur som kod
* **Monitor** – övervaka prestanda, resursutnyttjande, skalning, användarupplevelse etc.

● Beskriv hur man jobbar med versionshantering och vilka svårigheter som finns samt hur skulle versionshantering kunna se ut för respektive kontext? (½ - 1 sida)

Versionshanteringssystem är ett verktyg som hjälper team att hantera ändringar av ex. källkoden över tid. Ett exempel på program som låter och hantera versioner är git.

När man arbetar med ett versionshanteringssystem arbetar varje programmerare med en lokal version av systemet. Efter ändringar laddar programmeraren upp ändringarna i den nya versionen (på engelska: "commit"). Svårigheter Den nya versionen blir då tillgänglig för de andra som arbetar i projektet. Om två personer arbetar med samma kodbas vill man se till att de kan arbeta parallelt, att deras ändringar inte påverkar varandra och att det går enkelt att slå ihop ändringarna när de är klara.