QA Automation nədir və niyə vacibdir?

Keyfiyyət təminatı və avtomatlaşdırılmış testlər

Müasir proqram təminatı inkişafında **Keyfiyyət Təminatı (QA)** mühəndislərinin rolu getdikcə daha da vacib olur. İstifadəçilərin keyfiyyətli və stabil proqram təminatı ilə qarşılaşması üçün test mühəndisləri sistemləri dərin analiz edir, səhvləri tapır və düzəldir.

QA Automation isə test prosesini avtomatlaşdıraraq daha sürətli və səmərəli hala gətirən bir yanaşmadır. Əgər əl ilə testlər hər dəfə təkrar icra edilməli olursa, avtomatlaşdırılmış testlər kod vasitəsilə bu prosesi sürətləndirir və insan müdaxiləsini minimuma endirir.

QA Automation-in üstünlükləri

- **Vaxt qənaəti:** Test prosesləri əl ilə aparıldıqda saatlarla vaxt ala bilər. Avtomatlaşdırılmış testlər saniyələr içində nəticə verir.
- **Dəqiqlik və etibarlılıq:** Manual testlər insan səhvlərinə açıqdır, amma avtomatlaşdırılmış testlər eyni ssenarini hər dəfə dəqiqliklə yerinə yetirir.
- **Maliyyə qənaəti:** Uzun müddətli perspektivdə test avtomatlaşdırması şirkətlərə işçi gücündən və resurslardan qənaət etməyə imkan verir.
- **Davamlı inteqrasiya:** QA mühəndisləri avtomatlaşdırılmış testləri **CI/CD** (Continuous Integration / Continuous Deployment) prosesinə daxil edərək kod dəyişikliklərini anında test edə bilərlər.
- **Geniş miqyasda test imkanı:** Avtomatlaşdırılmış testlər minlərlə ssenarini paralel şəkildə yoxlaya bilər, əl ilə bunu həyata keçirmək isə çox çətindir.

QA mühəndisi üçün avtomatlaşdırma bacarıqları

Bir QA mühəndisi test avtomatlaşdırması sahəsində müvəffəq olmaq üçün aşağıdakı bacarıqlara sahib olmalıdır:

- **Proqramlaşdırma bilikləri** (Java, Python, JavaScript)
- **Test framework-lər** (Selenium, Cypress, Playwright)
- **API testləri və Postman**
- **CI/CD inteqrasiyası və DevOps prinsipləri**
- **Test skriptlərinin yazılması və Page Object Model (POM)**

---

2. Test Avtomatlaşdırmasının Əsasları

Test avtomatlaşdırması nədir?

Test avtomatlaşdırması, proqram təminatının işləməsini yoxlamaq üçün yazılmış skriptlər və alətlər vasitəsilə testlərin avtomatik icra edilməsidir. Bu proses insan müdaxiləsini azaldır, testləri sürətləndirir və daha geniş miqyasda yoxlama aparmağa imkan yaradır.

Test səviyyələri

Test avtomatlaşdırması fərqli səviyyələrdə tətbiq edilir:

- **Unit Testlər** – Proqramın kiçik hissələrinin düzgün işlədiyini yoxlayır.
- **Integration Testlər** – Bir neçə modulun birlikdə necə işlədiyini test edir.
- **API Testlər** – Backend servislərinin cavablarını yoxlayır.
- **UI Testlər** – İstifadəçi interfeysinin funksional olub-olmadığını test edir.
- **Performance Testlər** – Sistemə yük düşdükdə onun necə davrandığını analiz edir.

Test avtomatlaşdırmasının yanaşmaları

Bəzi mühüm test strategiyaları aşağıdakılardır:

1. **Data-Driven Testing** – Fərqli test məlumatları ilə test skriptləri icra edilir.
2. **Keyword-Driven Testing** – Əvvəlcədən müəyyən edilmiş açar sözlərə əsaslanan testlər.
3. **Behavior-Driven Development (BDD)** – Testlərin Gherkin dili ilə yazıldığı və biznes tələbləri ilə uyğunlaşdırıldığı yanaşma.
4. **Regression Testing** – Yeni kod dəyişikliklərindən sonra sistemin əvvəlki funksiyalarının pozulmadığını yoxlamaq.

Ən populyar test framework-ləri

Avtomatlaşdırılmış testlər üçün ən çox istifadə olunan framework-lər:

- **Selenium** – Web testləri üçün populyar framework.
- **Cypress** – Müasir UI test avtomatlaşdırması aləti.
- **Playwright** – Microsoft tərəfindən hazırlanmış web test framework.
- **JUnit/TestNG** – Java ilə test skriptləri yazmaq üçün framework-lər.
- **RestAssured** – API testləri üçün istifadə olunan Java kitabxanası.

- **JMeter** – Performance testləri aparmaq üçün geniş istifadə olunan alət.

QA mühəndislərinin avtomatlaşdırmaya yanaşması

Uğurlu QA mühəndisləri aşağıdakı prinsiplərə əsaslanır:

- **Testlərin yenilənməsini təmin etmək**
- **Modular kod yazmaq və təkrar istifadə edilə bilən skriptlər yaratmaq**
- **CI/CD inteqrasiyası ilə testləri avtomatik icra etmək**
- **Page Object Model (POM) ilə kodun strukturunu yaxşılaşdırmaq**

---

3. QA Automation Alətləri və Texnologiyalar

Selenium – Web Test Avtomatlaşdırması

Selenium, veb tətbiqlər üçün ən geniş istifadə olunan **test avtomatlaşdırma framework**-lərindən biridir. Test mühəndislərinə müxtəlif brauzerlərdə testlər icra etməyə imkan yaradır və bir neçə proqramlaşdırma dili ilə işləyir (Java, Python, C# və s.).

Əsas xüsusiyyətləri:

- **WebDriver API** ilə brauzerləri avtomatlaşdırır.
- **Headless browser** testləri üçün dəstək verir.
- Testləri paralel icra etməyə imkan yaradır.

Kod nümunəsi (Selenium WebDriver, Java)

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;


public class SeleniumTest {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.google.com");
        System.out.println("Səhifə başlığı: " + driver.getTitle());
```

```
        driver.quit();
    }
}
```

Cypress – Müasir UI Test Aləti

Cypress, **frontend testlər** üçün ideal framework-dür. O, **JavaScript ilə yazıldığı üçün müasir veb tətbiqlərinə** uyğun gəlir.

Əsas üstünlükləri:

- **Real vaxt rejimində** test icra edir.
- Testlərin yazılması və başa düşülməsi asandır.
- **Asinxron hadisələri** idarə edə bilir.

Kod nümunəsi (Cypress UI testi)

```
describe('Google Testi', () => {
    it('Sayta giriş və başlıq yoxlaması', () => {
        cy.visit('https://www.google.com');
        cy.title().should('include', 'Google');
    });
});
```

API Testləri – Postman və RestAssured

Backend servislərinin işləməsini yoxlamaq üçün **API testləri** mühüm rol oynayır. Bunun üçün **Postman** və **RestAssured** geniş istifadə olunur.

Kod nümunəsi (RestAssured API testi, Java)

```
import io.restassured.RestAssured;
import static io.restassured.RestAssured.*;

public class ApiTest {
```

```
    public static void main(String[] args) {
        RestAssured.baseURI = "https://jsonplaceholder.typicode.com";
        given()
            .when().get("/posts/1")
            .then().statusCode(200)
            .log().all();
    }
}
```

Performance Testləri – JMeter və LoadRunner

Proqramların **yük altında** necə işlədiyini test etmək üçün **JMeter** və **LoadRunner** kimi alətlərdən istifadə edilir.

Əsas funksiyalar:

- **Stress test** – Sistemə maksimum yük düşdükdə onun necə işlədiyini yoxlayır.
- **Spike test** – Ani yüklənmələrin təsirini test edir.
- **Load test** – Sistem normal iş rejimində necə işlədiyini ölçür.

---

4. Test Framework-ləri və Kodlaşdırma

Test Framework nədir?

Test framework-ləri test avtomatlaşdırmasının əsasını təşkil edir. Onlar **test skriptlərinin strukturunu müəyyənləşdirir, test icrasını idarə edir və nəticələri təqdim edir**. Framework-lər olmadan testlər nizamsız və idarəolunmaz ola bilər.

Ən populyar test framework-ləri

QA mühəndisləri tərəfindən geniş istifadə olunan test framework-lərindən bəziləri:

✅ **JUnit və TestNG** – Java ilə avtomatlaşdırılmış testlər yazmaq üçün framework-lər.
✅ **Selenium** – Web testləri üçün ən populyar framework.
✅ **Cypress** – JavaScript əsaslı UI test framework-ü.
✅ **Playwright** – Microsoft tərəfindən hazırlanmış müasir web test framework.

✅ **RestAssured** – API testləri üçün istifadə olunan Java kitabxanası.
✅ **Appium** – Mobil tətbiqləri test etmək üçün framework.

JUnit və TestNG ilə test skriptlərinin yazılması

JUnit və TestNG, Java ilə **unit testləri və avtomatlaşdırılmış testlər** yazmaq üçün geniş istifadə olunur. **JUnit daha sadə**, TestNG isə **daha güclü test idarəetmə imkanları** təqdim edir.

Kod nümunəsi (JUnit ilə sadə test skripti)

```
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;


public class SampleTest {

    @Test

    void testAddition() {

        int sum = 2 + 3;

        assertEquals(5, sum);

    }

}
```

Bu test **2 + 3 hesablamasının nəticəsini yoxlayır** və əgər cavab gözlənilən kimi **5** olarsa, test uğurla keçmiş sayılır.

Kod nümunəsi (TestNG ilə test icrası)

```
import org.testng.annotations.Test;

import static org.testng.Assert.*;


public class SampleTestNG {

    @Test

    public void testMultiplication() {

        int result = 2 * 4;
```

```
        assertEquals(result, 8);
    }
}
```

TestNG, JUnit-dən fərqli olaraq **test metodlarını daha geniş idarəetmə imkanları ilə təmin edir**.

Page Object Model (POM) – Kod Strukturlaşdırma

QA mühəndisləri test kodlarının **çox istifadəyə yararlı və strukturlaşdırılmış** olması üçün **Page Object Model (POM)** yanaşmasını istifadə edirlər.

📌 **POM nədir?**
Page Object Model, test kodlarını **müxtəlif komponentlərə bölərək** onların **yenidən istifadə edilə bilən** olmasını təmin edir.

Kod nümunəsi (Selenium və POM yanaşması)

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

public class LoginPage {
    WebDriver driver;

    @FindBy(id = "username")
    WebElement usernameField;

    @FindBy(id = "password")
    WebElement passwordField;

    @FindBy(id = "loginButton")
```

```
    WebElement loginButton;

    public LoginPage(WebDriver driver) {
        this.driver = driver;
        PageFactory.initElements(driver, this);
    }

    public void login(String username, String password) {
        usernameField.sendKeys(username);
        passwordField.sendKeys(password);
        loginButton.click();
    }
}
```

✅ **POM modeli test kodlarını daha səliqəli və optimallaşdırılmış** edir, eyni elementlərdən **fərqli testlərdə istifadə etmək imkanı yaradır**.

CI/CD inteqrasiyası və avtomatlaşdırılmış testlər

✅ **Continuous Integration (CI)** – Kod dəyişiklikləri edildikdə testlərin avtomatik icra olunmasını təmin edir.
✅ **Continuous Deployment (CD)** – Kod **testlərdən keçdikdən sonra** avtomatik olaraq sistemə inteqrasiya edilir.

CI/CD alətləri:

- **Jenkins** – Testlərin və kod inteqrasiyasının avtomatlaşdırılması.
- **GitHub Actions** – Kod dəyişikliklərini idarə edən CI/CD aləti.
- **CircleCI** – CI/CD prosesini sürətləndirən alət.

---

5. Praktiki Layihələr və Real Case Study-lər

Layihə 1: Web Tətbiqi üçün Selenium Testləri

Bu layihədə **Selenium WebDriver** istifadə edərək bir veb səhifənin əsas funksiyalarını test edəcəyik.

✅ **Məqsəd:** Google axtarış səhifəsində bir test ssenarisi yazmaq.
✅ **İcra olunacaq ssenari:**

- Səhifəyə daxil olmaq
- Axtarış çubuğuna dəyər daxil etmək
- Axtarış nəticələrinin düzgün gəldiyini yoxlamaq

Kod nümunəsi (Selenium ilə Google axtarış testi, Java)

```
import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;


public class GoogleSearchTest {

    public static void main(String[] args) {

        System.setProperty("webdriver.chrome.driver", "chromedriver.exe");

        WebDriver driver = new ChromeDriver();


        driver.get("https://www.google.com");

        WebElement searchBox = driver.findElement(By.name("q"));

        searchBox.sendKeys("QA Automation");

        searchBox.submit();


        System.out.println("Səhifə başlığı: " + driver.getTitle());

        driver.quit();
```

```
    }
}
```

Layihə 2: API Testləri – RestAssured ilə Real Tətbiq

Burada **RestAssured** ilə bir REST API endpoint-in doğru cavab verdiyini test edəcəyik.

✅ **Məqsəd:** API-nin işlək olduğunu yoxlamaq.
✅ **İcra olunacaq ssenari:**

- Endpoint-ə GET sorğusu göndərmək
- HTTP status kodunu yoxlamaq
- Cavabda lazım olan məlumatların olub-olmadığını test etmək

Kod nümunəsi (RestAssured ilə API testi, Java)

```
import io.restassured.RestAssured;
import static io.restassured.RestAssured.*;
import static org.hamcrest.Matchers.*;

public class ApiTest {
    public static void main(String[] args) {
        RestAssured.baseURI = "https://jsonplaceholder.typicode.com";

        given()
            .when().get("/posts/1")
            .then().statusCode(200)
            .body("userId", equalTo(1))
            .log().all();
    }
}
```

Layihə 3: CI/CD inteqrasiyası – Jenkins və GitHub Actions

Burada kod dəyişiklikləri edildikdə testlərin **CI/CD pipeline** vasitəsilə avtomatik icra edilməsini təmin edəcəyik.

✅ **Məqsəd:** Kodun hər dəyişiklikdən sonra test edildiyini yoxlamaq.
✅ **İcra olunacaq ssenari:**

- Test skriptini GitHub repository-də saxlamaq
- Jenkins və GitHub Actions vasitəsilə avtomatik icra etmək

GitHub Actions konfiqurasiya nümunəsi

```
name: Selenium Tests

on:
  push:
    branches:
      - main

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - name: Repository-yə giriş
        uses: actions/checkout@v3

      - name: Java quraşdırılması
        uses: actions/setup-java@v3
        with:
```

```
    java-version: '17'


- name: Testlərin icrası
  run: mvn test
```

Layihə 4: Performance Test – JMeter ilə Stress Test

Burada JMeter ilə bir sistemin **yük altında necə işlədiyini** test edəcəyik.

✅ **Məqsəd:** Serverin müəyyən sayda sorğu altında düzgün işlədiyini yoxlamaq.
✅ **İcra olunacaq ssenari:**

- JMeter test ssenarisi yaratmaq
- 1000 paralel istifadəçi ilə sistemə test etmək
- Cavab vaxtlarını ölçmək

---

6. Best Practices və Tövsiyələr

1. Effektiv Test Skriptləri Yazmaq

Testlərin uğurlu olması üçün onların **səmərəli, təmiz və geniş miqyasda istifadə edilə bilən** olması vacibdir. Bunun üçün aşağıdakı metodlardan istifadə etmək tövsiyə olunur:
✅ **Kodun təmiz yazılması** – Lazımsız kodlardan qaçmaq və sadə strukturlar istifadə etmək.
✅ **Reusable komponentlər yaratmaq** – Test kodunun müxtəlif bölmələrində eyni kodu istifadə etməyə imkan yaradan Page Object Model (POM) kimi yanaşmalardan istifadə etmək.
✅ **Dinamik və parametrik testlər yazmaq** – Hardcoded dəyərlərdən qaçmaq və test giriş məlumatlarını dinamik olaraq ötürmək.

Kod nümunəsi (Page Object Model ilə kodun təmiz saxlanması)

```
public class LoginPage {
   WebDriver driver;


   @FindBy(id = "username") WebElement usernameField;
   @FindBy(id = "password") WebElement passwordField;
```

```
    @FindBy(id = "loginButton") WebElement loginButton;

    public LoginPage(WebDriver driver) {
        this.driver = driver;
        PageFactory.initElements(driver, this);
    }

    public void login(String username, String password) {
        usernameField.sendKeys(username);
        passwordField.sendKeys(password);
        loginButton.click();
    }
}
```

✅ **Bu yanaşma kodun səliqəli və modul şəkildə yazılmasını təmin edir.**

---

2. Kodun Təmizliyi və Optimallaşdırma Metodları

Kodun **oxunaqlı və sadə** olması uzunmüddətli baxımdan mühüm əhəmiyyət daşıyır. Bunun üçün:

✅ **DRY (Don't Repeat Yourself) prinsipinə əməl etmək** – Təkrar kod yazmaqdan qaçmaq.

✅ **Modular kod yazmaq** – Hər funksiyanı ayrı bloklara bölərək testlərin idarə edilməsini asanlaşdırmaq.

✅ **İstifadəçi dostu hesabatlar yaratmaq** – Test nəticələrini aydın şəkildə təqdim edən test hesabatları yaratmaq.

---

3. QA Mühəndisləri üçün Karyera Tövsiyələri

Test mühəndisləri daim öz biliklərini inkişaf etdirməli və **sənaye trendlərini izləməlidirlər**. Bunun üçün aşağıdakı tövsiyələr faydalıdır:

✅ **Proqramlaşdırma biliklərini genişləndirmək** (Java, Python, JavaScript).

✅ **Yeni test framework-ləri və alətləri öyrənmək** (Cypress, Playwright, Postman).
✅ **DevOps və CI/CD inteqrasiyası biliklərini inkişaf etdirmək**.
✅ **Peşəkar sertifikatlar əldə etmək** – ISTQB, Selenium WebDriver Certified Professional və s.

---

7. Nəticə və Gələcək Perspektivlər

QA Avtomatlaşdırmasının Əhəmiyyəti

Test avtomatlaşdırması müasir proqram təminatı inkişafında **əhəmiyyətli rol oynayır**. Əl ilə testlər mühüm olsa da, avtomatlaşdırılmış testlər **sürət, dəqiqlik və miqyas baxımından daha üstün** olur.
Şirkətlər test mühəndisləri üçün **CI/CD proseslərinə inteqrasiya olunan və optimallaşdırılmış test infrastrukturu** yaratmağa çalışırlar.

✅ **Test avtomatlaşdırması inkişaf etdikcə:**

- **Testlərin icra sürəti artır** və nəticələr daha tez əldə edilir.
- **İnsan səhvləri azalır** və testlər daha etibarlı olur.
- **Avtomatlaşdırılmış testlər DevOps və CI/CD ilə tam uyğunlaşır**.

Avtomatlaşdırmanın Gələcək İnkişafı

Müasir texnologiyaların **süni intellekt və maşın öyrənməsi ilə birləşməsi**, test mühəndislərinə **daha ağıllı və adaptiv testlər yaratmaq** imkanı verir.

📌 **Gələcəkdə test avtomatlaşdırmasında əsas istiqamətlər:**
✅ **Süni intellekt əsaslı test alətləri** – AI köməkçisi avtomatik test ssenariləri yarada və kod səhvlərini təhlil edə bilər.
✅ **Self-healing testlər** – Testlər sistemdə dəyişikliklər olduqda **avtomatik uyğunlaşaraq** işləyə bilər.
✅ **Codeless test avtomatlaşdırması** – Texniki olmayan şəxslər belə testlər yarada biləcək.
✅ **Blockchain testləri** – Smart contract-lar və digər texnologiyalar üçün test avtomatlaşdırması inkişaf edir.

Süni İntellektin QA Sahəsində Rolu

Süni intellekt test mühəndislərinə **effektiv, ağıllı və optimallaşdırılmış test yanaşmaları** yaratmağa kömək edir. AI əsaslı sistemlər:

- 🔹 **Test ssenarilərini avtomatik optimallaşdırır**
- 🔹 **Kodun keyfiyyətini analiz edir və səhvləri müəyyən edir**
- 🔹 **Regression testləri üçün adaptiv metodlar təqdim edir**

Məsələn, **Test.ai** və **Applitools** kimi AI əsaslı alətlər, **vizual testləri və UI səhvlərini avtomatik analiz edir** və insan müdaxiləsini minimuma endirir.

Yekun Söz

QA mühəndisləri üçün avtomatlaşdırma **artıq lüks deyil, zərurətdir**. Test mühəndisləri **davamlı öyrənməli və sənaye trendlərini izləməlidirlər**.

✅ **İnkişaf etmək istəyən hər bir test mühəndisi:**

- **Yeni framework-ləri öyrənməli**
- **Kod keyfiyyətini artırmalı**
- **CI/CD inteqrasiyası və DevOps biliklərini genişləndirməli**
- **Süni intellekt əsaslı avtomatlaşdırma metodlarını mənimsəməlidir**

QA sahəsində uğur qazanan mütəxəssislər **təkmilləşən texnologiyalara uyğunlaşaraq** keyfiyyət təminatına töhfə verir.
