

Link GitHub : <https://github.com/Samijaidi/QA-Engineer-sami>

Jawab :

1. QA (Quality Assurance) adalah proses sistematis untuk memastikan bahwa perangkat lunak atau software memenuhi standar kualitas tertentu. seorang QA Engineer memiliki tugas utama yaitu :

- Perencanaan Pengujian: Merencanakan strategi dan metode pengujian yang tepat.
- Pengujian Perangkat Lunak: Melakukan pengujian fungsional dan non-fungsional untuk menemukan dan mendokumentasikan bug.
- Kolaborasi: Bekerja sama dengan tim pengembang dan pemangku kepentingan untuk memastikan kualitas produk.
- Otomatisasi Pengujian: Mengembangkan skrip pengujian otomatis untuk efisiensi.
- Pelaporan dan Analisis: Menganalisis hasil pengujian dan memberikan umpan balik untuk perbaikan.

2. · **Pengujian Fungsional:** Memastikan bahwa aplikasi berfungsi sesuai dengan spesifikasi yang ditetapkan. Contoh: memeriksa fitur login, input form, dll.

· **Pengujian Non-Fungsional:** Menilai aspek lain dari perangkat lunak, seperti kinerja suatu software, keamanan, dan usability. Contoh: mengukur waktu respons aplikasi atau mengevaluasi antarmuka pengguna.

3.· **Analisis Kebutuhan:** Memahami spesifikasi dan fungsi dari sistem atau aplikasi.

· **Perencanaan Pengujian:** Menyusun rencana pengujian, termasuk jenis pengujian yang akan dilakukan.

· **Pembuatan Test Case:** Mengembangkan skenario pengujian berdasarkan spesifikasi.

· **Pelaksanaan Pengujian:** Melakukan pengujian sesuai dengan test case yang telah dibuat sebelumnya.

· **Dokumentasi Hasil:** Mencatat hasil pengujian, termasuk bug yang ditemukan.

· **Pelaporan:** Membuat laporan pengujian untuk tim developer

· **Retesting dan Regression Testing:** melakukan pengujian ulang setelah perbaikan bug dan memastikan tidak ada regresi.

4. Uji regresi adalah jenis pengujian yang dilakukan untuk memastikan bahwa perubahan pada perangkat lunak (seperti perbaikan bug atau penambahan fitur) tidak merusak fungsionalitas yang ada. Ini penting karena:

- Memastikan stabilitas software.
- Mengidentifikasi bug yang mungkin diperkenalkan oleh perubahan baru.
- Mengurangi risiko kegagalan sistem di masa mendatang.

5. **Pengujian Manual:**

- Definisi: Proses pengujian perangkat lunak dilakukan oleh penguji manusia tanpa bantuan automation tools.

- **Kelebihan:**

- * Lebih fleksibel dan dapat menangkap masalah yang mungkin terlewatkan oleh skrip otomatis.
- * Baik untuk eksplorasi dan pengujian ad-hoc.
- **Kekurangan:**
 - * Memakan waktu dan tidak efisien untuk pengujian berulang.
 - * Rentan terhadap kesalahan manusia.

Pengujian Otomatis:

- Definisi: Proses pengujian perangkat lunak dilakukan dengan menggunakan alat otomatis untuk menjalankan tes.

- Kelebihan:

- * Efisien untuk pengujian berulang dan regresi.
- * Dapat menghemat waktu dan biaya dalam jangka panjang.

- Kekurangan:

- * Memerlukan waktu dan usaha awal untuk menyiapkan skrip.
- * Kurang fleksibel untuk perubahan mendadak dalam aplikasi.

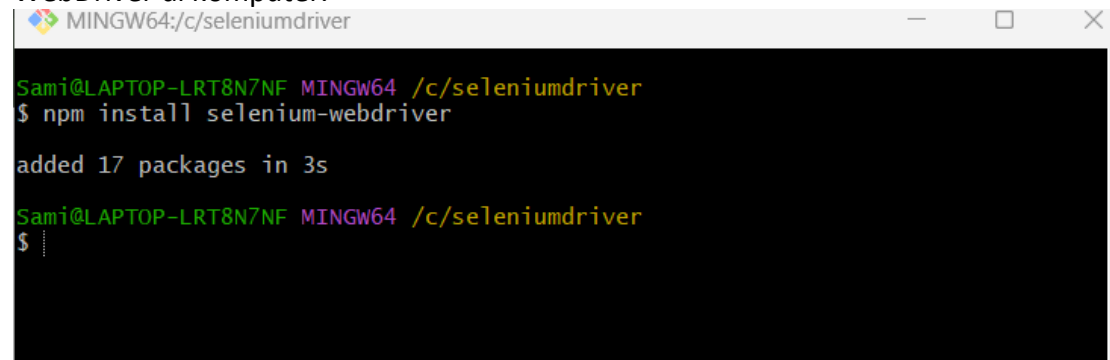
- Gunakan pengujian otomatis ketika:

- * Pengujian perlu dilakukan berulang kali (regresi).
- * Ada kebutuhan untuk pengujian fungsional skala besar.
- * Waktu dan anggaran memungkinkan untuk investasi awal dalam otomatisasi.

6. Proses penggunaan selenium :

1. Install Selenium dan WebDriver

Langkah pertama adalah memastikan bahwa telah menginstal Selenium dan WebDriver di komputer.



```

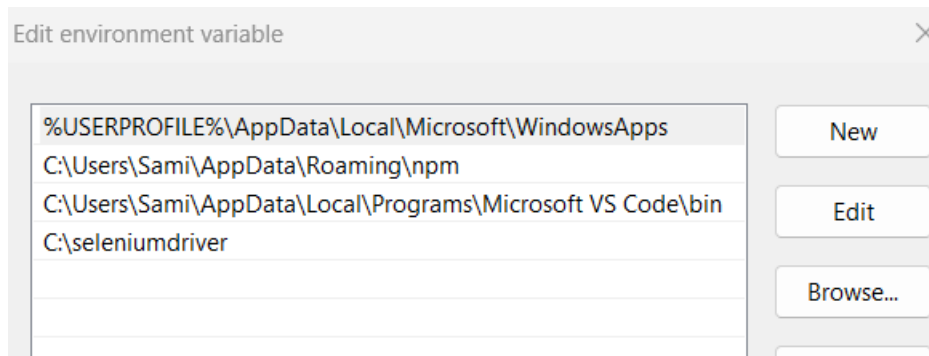
MINGW64:/c/seleniumdriver
Sami@LAPTOP-LRT8N7NF MINGW64 /c/seleniumdriver
$ npm install selenium-webdriver
added 17 packages in 3s
Sami@LAPTOP-LRT8N7NF MINGW64 /c/seleniumdriver
$
  
```

2. Memilih Bahasa Pemrograman

Selenium mendukung banyak bahasa pemrograman. Tentukan bahasa pemrograman yang akan digunakan

3. Setting Environment

Sebelum memulai pengujian, kita perlu menyiapkan lingkungan pengujian. Ini termasuk pengaturan path dan dependencies yang dibutuhkan.



4. Menulis Skrip Pengujian

Setelah semuanya siap, kita bisa mulai menulis skrip untuk mengotomatisasi pengujian. Pastikan skrip mencakup semua skenario yang mungkin terjadi.

```
JS test.js > ...
1  const { Builder, Browser, By, Key, until } = require('selenium-webdriver')
2
3  ;(async function example() {
4      let driver = await new Builder().forBrowser("chrome").build();
5      try {
6          await driver.get('https://www.knitto.co.id/ncr');
7          await driver.findElement(By.name('q')).sendKeys('webdriver', Key.RETURN);
8          await driver.wait(until.titleIs('webdriver - Google Search'), 1000);
9      } finally {
10         await driver.quit();
11     }
12 })()
```

5. Menjalankan Pengujian

Setelah skrip selesai disimpan di code editor, saatnya untuk menjalankan pengujian. Selenium akan mengeksekusi skrip dan memberikan hasil pengujian dalam waktu singkat.

6. Analisis Hasil Pengujian

Setelah pengujian selesai, langkah berikutnya adalah menganalisis hasilnya. Identifikasi bug dan masalah lain yang mungkin ditemukan.

7. Optimasi dan Debugging

Jika ada masalah yang ditemukan, kita bisa melakukan optimasi dan debugging pada skrip untuk memperbaiki dan meningkatkan hasil pengujian.

8. Mengintegrasikan dengan CI/CD Pipeline

Selenium dapat diintegrasikan dengan CI/CD pipeline untuk memastikan pengujian otomatis dilakukan setiap kali ada update pada kode.

9. Melakukan Load Testing

Selain pengujian fungsional, Selenium juga bisa digunakan untuk melakukan load testing untuk melihat seberapa kuat aplikasi web mengatasi traffic tinggi.

10. Dokumentasi Hasil Pengujian

Langkah terakhir adalah mendokumentasikan hasil pengujian. Dokumentasi yang baik akan membantu tim developer dalam memahami dan mengatasi masalah yang ditemukan.

```
$ node test.js
DevTools listening on ws://127.0.0.1:61113/devtools/browser
/b28afd4e-b7b7-457d-b80f-b4c38d1e31a0
C:\nodejs\selenium\node_modules\selenium-webdriver\lib\erro
r.js:521
    let err = new ctor(data.message)
              ^
NoSuchElementException: no such element: Unable to locate eleme
nt: {"method":"css_selector","selector":"*[name='q']"}
(Session info: chrome=129.0.6668.90)
    at Object.throwDecodedError (C:\nodejs\selenium\node_mo
dules\selenium-webdriver\lib\error.js:521:15)
    at parseHttpResponse (C:\nodejs\selenium\node_modules\s
elenium-webdriver\lib\http.js:514:13)
    at Executor.execute (C:\nodejs\selenium\node_modules\se
lenium-webdriver\lib\http.js:446:28)
    at process.processTicksAndRejections (node:internal/pro
cess/task_queues:95:5)
    at async Driver.execute (C:\nodejs\selenium\node_module
s\selenium-webdriver\lib\webdriver.js:744:17)
    at async toWireValue (C:\nodejs\selenium\node_modules\s
elenium-webdriver\lib\webdriver.js:148:15)
    at async C:\nodejs\selenium\node_modules\selenium-webdr
iver\lib\webdriver.js:194:16
    at async forEachKey (C:\nodejs\selenium\node_modules\se
lenium-webdriver\lib\webdriver.js:188:9)
    at async convertKeys (C:\nodejs\selenium\node_modules\s
elenium-webdriver\lib\webdriver.js:193:3)
    at async Driver.execute (C:\nodejs\selenium\node_module
s\selenium-webdriver\lib\webdriver.js:742:22) {
  remoteStackTrace: '\tGetHandleVerifier [0x00437143+25587]
\n' +
    '\t(No symbol) [0x003CA2E4]\n' +
    '\t(No symbol) [0x002C2113]\n' +
    '\t(No symbol) [0x00306F62]\n' +
    '\t(No symbol) [0x003071AB]\n' +
    '\t(No symbol) [0x00347852]\n' +
    '\t(No symbol) [0x0032ABE4]\n' +
    '\t(No symbol) [0x00345370]\n' +
    '\t(No symbol) [0x0032A936]\n' +
    '\t(No symbol) [0x002FBA73]\n' +
    '\t(No symbol) [0x002FC4CD]\n' +
    '\tGetHandleVerifier [0x00714C63+3030803]\n' +
    '\tGetHandleVerifier [0x00766B99+3366473]\n' +
    '\tGetHandleVerifier [0x004C95F2+624802]\n' +
    '\tGetHandleVerifier [0x004D0E6C+655644]\n' +
    '\t(No symbol) [0x003D2C9D]\n' +
    '\t(No symbol) [0x003CFD68]\n' +
    '\t(No symbol) [0x003CFF05]\n' +
    '\t(No symbol) [0x003C2336]\n' +
    '\tBaseThreadInitThunk [0x76177BA9+25]\n' +
    '\tRtlInitializeExceptionChain [0x770DC11B+107]\n' +
    '\tRtlClearBits [0x770DC09F+191]\n'
}
```

****Note :** saya juga telah membuatkan pengujian automation menggunakan selenium dan filenya disimpan di github

7. Kerangka kerja pengujian adalah struktur atau sistem yang menyediakan panduan dan alat untuk mendukung pengembangan, pengelolaan, dan eksekusi pengujian otomatis. Ini mencakup aturan, praktik, dan komponen yang membantu tim pengujian dalam merancang dan menjalankan skrip pengujian secara efektif.

Bagaimana Kerangka Kerja Ini Membantu dalam Pengujian Perangkat Lunak?

- Standarisasi:

Kerangka kerja memastikan bahwa semua skrip uji mengikuti pola dan konvensi yang konsisten, sehingga memudahkan developer dan pengujian dalam memahami dan memelihara kode.

- Reusabilitas:

Komponen yang dibuat dalam kerangka kerja dapat digunakan kembali di berbagai skrip pengujian, mengurangi waktu dan usaha yang diperlukan untuk mengembangkan tes baru.

- Pemeliharaan:

Kerangka kerja memudahkan pemeliharaan skrip pengujian, karena perubahan pada aplikasi hanya perlu diupdate di satu tempat (misalnya, dalam fungsi atau modul tertentu) daripada di setiap skrip.

- Integrasi:

Banyak kerangka kerja mendukung integrasi dengan alat lain seperti alat pelaporan, bug tracking, dan sistem Continuous Integration/Continuous Deployment (CI/CD), yang meningkatkan efisiensi dan kolaborasi.

- Dukungan untuk Pengujian Berbasis Data:

Beberapa kerangka kerja memungkinkan pengujian berbasis data, di mana satu skrip uji dapat dijalankan dengan berbagai set data, meningkatkan cakupan pengujian.

- Pelaporan:

Kerangka kerja sering menyediakan fitur pelaporan yang memudahkan tim untuk melacak hasil pengujian dan masalah yang ditemukan, serta memvisualisasikan progres pengujian.

- Mendukung Berbagai Jenis Pengujian:

Kerangka kerja dapat dirancang untuk mendukung berbagai jenis pengujian, seperti pengujian fungsional, pengujian regresi, dan pengujian performa, membuatnya lebih fleksibel.

****contoh kerangka pengujian seperti Selenium dan Cucumber.**

8. Bug Tracking System (Sistem Pelacakan Bug) adalah alat atau perangkat lunak yang digunakan untuk melacak, mengelola, dan mengontrol bug atau masalah dalam pengembangan perangkat lunak. Sistem ini membantu tim pengembang dan penguji untuk mendokumentasikan masalah yang ditemukan, memantau status perbaikan, dan memastikan bahwa semua masalah ditangani dengan tepat.

Beberapa Fitur Umum dari Bug Tracking System:

- Pelaporan Bug: Pengguna dapat melaporkan bug dengan detail, termasuk langkah untuk mereproduksi masalah, tingkat keparahan, dan status saat ini.

- Manajemen Status: Memungkinkan tim untuk mengubah status bug (misalnya, terbuka, dalam perbaikan, ditutup) dan memantau kemajuan.

- Prioritas dan Kategori: Memungkinkan pengelompokan bug berdasarkan prioritas atau kategori, sehingga tim dapat fokus pada masalah yang lebih kritis terlebih dahulu.

- Kolaborasi: Tim dapat berkolaborasi dan mendiskusikan bug melalui komentar atau forum, meningkatkan komunikasi dan efektivitas.

- Integrasi: Dapat diintegrasikan dengan alat lain seperti alat pengujian, sistem manajemen proyek, dan CI/CD untuk meningkatkan alur kerja.

Beberapa Tools automation yang saya ketahui diantaranya ada :
Selenium, JIRA, Bugzilla, Cucumber, Azure, MantisBT dan Appium.

9. Untuk jawaban no. 9 saya buat test case dalam bentuk .xlsx di GitHub

10. Berikut langkah-langkah yang akan saya lakukan ketika melakukan pengujian Aplikasi Perbankan Online :

1. Uji Penetrasi (Penetration Testing)

Simulasi serangan untuk mengidentifikasi kerentanan dalam aplikasi.

Contoh:

SQL Injection: Menguji apakah input pengguna dapat digunakan untuk mengeksploitasi database.

Cross-Site Scripting (XSS): Menguji apakah penyerang dapat menyisipkan skrip berbahaya ke dalam halaman web yang diakses oleh pengguna lain.

2. Uji Autentikasi

Memastikan bahwa mekanisme autentikasi kuat dan aman.

Contoh:

Kekuatan Kata Sandi: Menguji kebijakan kata sandi untuk memastikan bahwa pengguna tidak dapat menggunakan kata sandi yang lemah.

Autentikasi Dua Faktor (2FA): Memastikan bahwa fitur 2FA berfungsi dengan baik dan mencegah akses tidak sah.

3. Uji Manajemen Sesi

Memastikan sesi pengguna dikelola dengan aman.

Contoh:

Timeout Sesi: Menguji apakah sesi pengguna secara otomatis berakhir setelah periode tidak aktif yang ditentukan.

Re-use Sesi: Menguji apakah sesi yang sudah digunakan tidak dapat digunakan kembali setelah logout.

4. Uji Keamanan Data

Memastikan bahwa data sensitif dilindungi selama penyimpanan dan transmisi.

Contoh:

Enkripsi Data: Menguji apakah informasi sensitif seperti nomor kartu kredit dan data pribadi dienkripsi saat disimpan dan saat ditransfer melalui jaringan.

Transport Layer Security (TLS): Memastikan bahwa komunikasi antara pengguna dan server menggunakan protokol TLS yang kuat.

5. Uji Keamanan API

Memastikan API yang digunakan oleh aplikasi aman.

Contoh:

Otentikasi API: Menguji apakah API memerlukan autentikasi dan memiliki kontrol akses yang tepat.

Validasi Input API: Memastikan bahwa API memvalidasi semua input yang diterima untuk mencegah eksploitasi.

6. Uji Kebijakan Akses

Memastikan pengguna hanya memiliki akses yang sesuai.

Contoh:

Uji Hak Akses Pengguna: Menguji apakah pengguna biasa tidak dapat mengakses fungsionalitas atau data yang hanya untuk admin.

Uji Akses Berdasarkan Peran: Memastikan bahwa sistem membatasi akses berdasarkan peran pengguna dengan benar.

7. Uji Kerentanan

Mengidentifikasi dan menilai kerentanan yang mungkin ada.

Contoh:

Penggunaan Alat Uji Kerentanan: Menggunakan alat seperti OWASP ZAP atau Nessus untuk melakukan pemindaian terhadap aplikasi.

8. Uji Keamanan Fisik dan Infrastruktur

Memastikan bahwa infrastruktur fisik dan jaringan aman.

Contoh:

Keamanan Server: Memastikan bahwa server tempat aplikasi dijalankan aman dan tidak rentan terhadap akses fisik yang tidak sah.

Firewall dan Sistem Deteksi Intrusi: Memastikan firewall dan IDS/IPS berfungsi dengan baik untuk melindungi aplikasi dari serangan.

11. Untuk jawaban no. 11 saya buat test case dalam bentuk .xlsx di GitHub

Link GitHub : <https://github.com/Samijaidi/QA-Engineer-sami>