QA Technical Test - Gita Pratti Nadapdap

1. “Software Quality Assurance” berkaitan dengan tugas dalam proses penjaminan mutu sebuah aplikasi atau perangkat lunak. Seorang SQA bertanggung jawab untuk menghasilkan produk yang berkualitas tinggi dengan bug/error seminimal mungkin. Karena sebaik-baiknya proses pengujian yang sudah dilakukan, tidak menutup kemungkinan ada kesalahan/bug pada produk tersebut. Jadi, menurut saya SQA itu perlu melakukan pengujian semaksimal mungkin dan merasa percaya diri akan produk (setiap release feature) yang telah diuji.

   Menurut apa yang masih saya pahami, metode testing baik black box testing white box testing perlu dilakukan. Hanya saja yang paling utama itu adalah penerapan black box testing (Usability Testing, Performance Testing, Smoke testing, Stress testing, Sanity testing, Regression testing, dll.)

2. Black box testing pengujian yang dilakukan berdasarkan detail aplikasi seperti tampilan aplikasi (UI), fungsi-fungsi pada aplikasi, dan bisnis flow sesuai yang diinginkan oleh customer. Pengujian ini tidak melihat dan menguji source code program.

   White Box Testing adalah pengujian yang dilakukan untuk menguji suatu aplikasi atau software dengan melihat modul untuk memeriksa dan menganalisis benar salahnya kode program aplikasi tersebut. Jika modul telah diproduksi dalam output yang tidak memenuhi requirement, kode akan dijalankan ulang dan diperiksa lagi sampai sesuai dengan kebutuhan. singkatnya White Box Testing ini menguji dengan cara melihat Pure Code dari suatu aplikasi atau software yang diuji tanpa memperdulikan tampilan dari aplikasi tersebut.

   Untuk yang menggunakan black-box atau white-box, baik QA dan programmer akan menggunakan kedua metode tersebut. Umumnya, QA akan dominan menjalankan metode black box dan programmer menjalankan metode white box testing seperti unit test (untuk memudahkan programmer memeriksa kesalahan yang mungkin ada pada code dan dapat segera memperbaiki kesalahan tersebut).

3. SDLC merupakan kerangka kerja atau manajemen proyek terstruktur dalam membangun aplikasi atau perangkat lunak dari awal hingga hasil akhir yang diharapkan. Tujuan dari Software Development Life Cycle adalah untuk menciptakan proses produksi yang efektif dan berkualitas tinggi agar dapat memenuhi atau melampaui harapan client sesuai dengan anggaran dan jadwal yang telah ditentukan.

   Untuk metode SDLC terdiri dari beberapa jenis seperti Waterfall Model, Iterative Model, V-Model dan yang paling banyak diterapkan dalam bidang IT yaitu metode Agile (karena lebih fleksibel dan beradaptasi sesuai kebutuhan proyek yang ingin dibangun).

Secara garis besar tahapan dalam SDLC terdiri dari beberapa tahapan, yaitu:
Planning → Analysis → Design → Development → Testing → Implementation dan Release → Maintenance

Menurut saya, seorang QA/QC berperan dalam seluruh tahapan SDLC. Namun akan paling berperan penting pada tahap:

- Testing (menguji keseluruhan fitur pada perangkat lunak), kemudian diikuti tahap
- Implementation & release (sebelum fitur benar-benar rilis ke user, perlu dilakukan User Acceptance Test agar kesalahan yang masih mungkin terjadi tidak sampai pada user), lalu
- Maintenance (biasanya pada tahap ini dilakukan perbaikan bug, upgrade sistem, dan peningkatan fitur. Sehingga QA perlu melakukan pengujian ulang seperti sanity test dan smoke test).

4. Penyebab terjadinya bug pada suatu aplikasi bisa karena berbagai faktor. Namun yang paling sering saya temui, seperti:
   - Syntax error (ex: yg harusnya di render itu channel_id, tapi di code programnya dibuat ke class_id)
   - Logical error/kesalahan logic di programnya (ex: dalam sebuah aplikasi ecommerce akan mempengaruhi hasil perhitungan, mis: diskon = 15%, harga = 100.000 harga akhir yang diharapkan adalah 85.000, tetapi karena kesalahan itu, harga akhir yang sebenarnya akan berubah, bisa lebih rendah atau lebih tinggi dari harga yang diharapkan)
   - Runtime error (ex: aplikasi atau program berhenti berfungsi/crash)

5. Dapat diakses pada url berikut:
   Test case Register

6. List Scenario:

Scenario	Expected result
Tombol A dan tombol panah kanan ditekan	Peluru tunggal tidak dapat ditembakan
Tombol A dan tombol panah kiri ditekan	Peluru tunggal tidak dapat ditembakan
Tombol A dan panah bawah ditekan	Peluru tunggal tidak dapat ditembakan
Tombol A dan panah atas ditekan	Peluru tunggal tidak dapat ditembakan
Tombol B dan tombol panah kanan ditekan	Tidak dapat melontarkan roket
Tombol B dan tombol panah kiri ditekan	Tidak dapat melontarkan roket

Tombol B dan tombol panah bawah ditekan	Tidak dapat melontarkan roket
Tombol B dan tombol panah atas ditekan	Tidak dapat melontarkan roket
Tekan tombol A dan tahan panah up/atas	Peluru dilontarkan secara horizontal
Tombol atas/up ditekan atau ditahan	Peluru tidak dapat dilontarkan
Tombol A dan B ditekan bersamaan dan tidak ditahan	Peluru maupun roket dilontarkan 1x
Tombol A dan B ditekan bersamaan dan ditahan	Menembakkan banyak peluru

7. Automation Script
   fileName: get-post-endpointWebsiteFMS.js

```
const chai = require('chai');
const chaiHttp = require('chai-http');
chai.use(chaiHttp);
const expect = chai.expect;

const getDashboard = '/';
const postEndpointLogin = '/login';
const postEndpointLoginFailed = '/login-failed'; //endpoint ini saya asumsikan
dari: Mengembalikan respon 4XX jika kombinasi username & password BUKAN
merupakan data pengguna yang valid
const postEndpointLogout = '/logout';
const getEndpointData = '/data';

const dashboardUrl = 'https://qa-interview.srcli.xyz/';
const loginUrl = 'https://qa-interview.srcli.xyz/login'; //url login saya
asumsikan sendiri karena tidak saya temukan pada soal/url yang diberikan
const logoutUrl = 'https://qa-interview.srcli.xyz/logout'; //url logout saya
asumsikan sendiri karena tidak saya temukan pada soal/url yang diberikan
const body = {
 username: 'root',
 password: 'root123',
};

//Endpoint Get
```

```
describe('Get Halaman Utama Website Financial Management System', function ()
{

 it('Berhasil menampilkan halaman utama berisi pesan selamat datang dan
penjelasan singkat website', async function () {

   const startTime = help.startTime();
   const res = await chai
     .request(dashboardUrl)
     .get(getDashboard)
     .set(

       req.createNewCoreHeaders({
         'X-Signature': headerSignature,
         'X-TimeStamp': timeStamp
       })
     )
     .send();
   const responseTime = help.responseTime(startTime);

   report.setPayload(this, res, responseTime);

   expect(res).to.have.status(200);
 });
});

describe('User Get Data', function () {

 it('Menampilkan dengan 2 tabel yang berisi daftar 10 pemasukan dan
pengeluaran terakhir should succeed', async function () {

   const startTime = help.startTime();
   const res = await chai
     .request(getEndpointData)
     .post(loginUrl)
     .set(req.createNewCoreHeaders())
     .send(body);
   const responseTime = help.responseTime(startTime);

   report.setPayload(this, res, responseTime);
```

```
    expect(res).to.have.status(302);
  });
});

//Endpoint Post
describe('User Login Validation', function () {

  it('User login should succeed', async function () {

    const startTime = help.startTime();
    const res = await chai
      .request(postEndpointLogin)
      .post(loginUrl)
      .set(req.createNewCoreHeaders())
      .send(body);
    const responseTime = help.responseTime(startTime);

    report.setPayload(this, res, responseTime);

    expect(res).to.have.status(302);
  });

  it('Using invalid username & password should fail', async function () {
    const body = {
      username: 'roots',
      password: 'salahGan07',
    };

    const startTime = help.startTime();
    const res = await chai
      .request(postEndpointLoginFailed)
      .post(loginUrl)
      .set(req.createNewCoreHeaders())
      .send(body);
    const responseTime = help.responseTime(startTime);

    report.setPayload(this, res, responseTime);

    expect(res).to.have.status(400);
  });
});
```

```
describe('User Logout Validation', function () {

 it('Logout should succeed', async function () {

   const startTime = help.startTime();
   const res = await chai
     .request(postEndpointLogout)
     .post(logoutUrl)
     .set(req.createNewCoreHeaders())
     .send();
   const responseTime = help.responseTime(startTime);

   report.setPayload(this, res, responseTime);

   expect(res).to.have.status(302);
 });
});
```