Pengenalan
Software Quality Assurance

Adelbertus Larry Dennis Lumban Toruan
Rangga Perwiratama
Djoko Budiyanto Setyohadi

Buku Seri : Pengembangan Sistem

Buku Seri Pengembangan Sistem

Pengenalan
Software Quality Assurance

Adelbertus Larry Dennis Lumban Toruan
Rangga Perwiratama
Djoko Budiyanto Setyohadi

Universitas Atma Jaya Yogyakarta

Pengenalan
Software Quality Assurance

Buku Seri Pengembangan Sistem

Oleh : Adelbertus Larry Dennis Lumban Toruan
Rangga Perwiratama
Djoko Budiyanto Setyohadi

Cetakan ke- 05 04 03 02 01

Tahun 28 27 26 25 24

Diterbitkan oleh
UNIVERSITAS ATMA JAYA YOGYAKARTA
Jl. Babarsari No. 5-6 Yogyakarta 55281
Telp. +62 274 487711
E-mail: lib.publisher@uajy.ac.id

ISBN: 978-623-10-2029-1 (PDF)

Pengenalan
Software Quality Assurance

Buku Seri Pengembangan Sistem

Kata Pengantar

Perangkat lunak memainkan peran yang semakin penting dalam kehidupan kita, dari aplikasi ponsel pintar hingga perangkat lunak rumah tangga dan sistem informasi perusahaan. Namun, dengan meningkatnya kompleksitas perangkat lunak, tantangan yang dihadapi pengembang juga semakin besar. Perlu kami diinformasikan bahwa salah satu tantangan utama dalam pengembanagn sistem adalah memastikan bahwa perangkat lunak yang dihasilkan memenuhi standar kualitas yang tinggi. Inilah mengapa Software Quality Assurance (SQA) sangat penting. SQA adalah suatu pendekatan sistematis untuk memastikan bahwa perangkat lunak yang dikembangkan memenuhi persyaratan yang telah ditetapkan, serta memastikan bahwa proses pengembangan perangkat lunak berjalan sesuai dengan standar yang ditetapkan.

Buku ini bertujuan untuk memberikan pemahaman yang komprehensif tentang konsep-konsep dasar SQA, praktik terbaik, dan metode yang dapat digunakan untuk meningkatkan kualitas perangkat lunak. Kami membahas berbagai aspek SQA, mulai dari pengenalan dasar tentang apa itu SQA, hingga praktik terbaik dalam implementasinya. Kami juga membahas berbagai metode dan teknik yang dapat digunakan dalam SQA, seperti pengujian perangkat lunak, analisis kode, dan manajemen konfigurasi. Selain itu, kami juga membahas pentingnya pengukuran kualitas perangkat lunak dan bagaimana melakukan pengukuran tersebut dengan efektif.

Salah satu aspek penting dari SQA adalah manajemen test case. Test case adalah langkah-langkah yang didefinisikan dengan baik untuk menguji fungsi atau fitur tertentu dari perangkat lunak. Dalam buku ini,

kami akan membahas tentang bagaimana manajemen test case dapat dilakukan dengan efektif menggunakan alat-alat seperti QA Touch dan Postman. QA Touch adalah salah satu perangkat lunak manajemen uji yang paling populer. Ini membantu tim pengembangan perangkat lunak dalam mengelola siklus hidup pengujian, termasuk perencanaan pengujian, pembuatan dan eksekusi test case, serta pelaporan hasil pengujian. Kami akan membahas bagaimana QA Touch dapat membantu dalam mengelola test case dengan efektif, sehingga memastikan bahwa setiap aspek dari perangkat lunak diuji dengan baik sebelum dirilis. Selain itu, kami juga akan membahas tentang penggunaan Postman dalam manajemen test case. Postman adalah alat pengujian API yang powerful yang dapat digunakan untuk membuat, mengelola, dan mengotomatisasi pengujian API. Kami akan menjelaskan bagaimana Postman dapat digunakan untuk membuat test case yang efektif untuk menguji berbagai fungsi API, sehingga memastikan bahwa API berfungsi dengan baik dan memberikan hasil yang diharapkan.

Buku ini ditujukan untuk para praktisi di bidang pengembangan perangkat lunak, mahasiswa yang belajar tentang SQA, dan siapa pun yang tertarik untuk memahami lebih lanjut tentang bagaimana memastikan kualitas perangkat lunak yang tinggi. Kami berharap buku ini dapat menjadi sumber pengetahuan yang berguna dan membantu dalam memahami SQA dengan lebih baik. Terakhir kami mengucapkan terima kasih atas semua dukungan terhadap buku ini. Kami berharap semoga buku ini dapat memberikan manfaat yang besar bagi Anda dalam pengembangan perangkat lunak yang berkualitas. Selamat membaca.

Daftar Isi

Bab 1.

Pengenalan

1.1. Siapakah Software Quality Assurance Engineer?

Sebelum mengetahui apakah itu seorang s*oftware quality assurance engineer*, ada baiknya untuk mengetahui apakah itu *software quality assurance* (dalam bahasa Indonesia, penjaminan mutu perangkat lunak). Berdasarkan definisi IEEE (Institute of Electrical and Electronics Engineers), penjaminan mutu perangkat lunak merupakan sebuah pola sistematik dan terencana dari seluruh tindakan penting untuk memberikan keyakinan yang cukup bahwa sebuah barang atau produk sesuai dengan persyaratan teknikal yang sudah ditetapkan.

Definisi lanjutan dari penjaminan mutu perangkat lunak oleh IEEE yaitu sekumpulan aktivitas-aktivitas yang didesain untuk mengevaluasi proses yang dimana produk-produk dikembangkan atau diproduksi. Adapun defisini dari kualitas *software* berdasarkan IEEE yaitu tingkat dimana sebuah sistem, komponen ataupun proses sesuai dengan persyaratan yang sudah ditentukan dan tingkat dimana sebuah sistem, komponen ataupun proses sesuai dengan harapan pelanggan atau kebutuhan pengguna. Sehingga, tujuan utama dari penjaminan mutu ini yaitu menerapkan cara-cara untuk mengurangi biaya supaya kualitas yang diberikan oleh sebuah produk terjamin pada saat proses pengembangan dan produksi [1]. Berawal dari tujuan ini, muncul sebuah pertanyaan

bagaimana caranya untuk dapat mengurangi biaya serta kemungkinan waktu untuk dapat menjamin mutu perangkat lunak.

Kemudian, siapakah seorang rekayasawan penjaminan mutu perangkat lunak? Mengambil kalimat dari tujuan penjaminan mutu perangkat lunak serta pertanyaan tadi, dapat disimpulkan pekerjaan apa yang akan dilakukan.Seseorang yang merancang sebuah rencana (dengan menerapkan kaidah-kaidah ilmu dalam pelaksanaannya [2]) untuk mengurangi biaya dalam menjamin mutu perangkat lunak saat proses pengembangan dan produksi.

1.2. Pekerjaan seorang SQA *Engineer*

Setelah mengetahui arti dari penjaminan mutu perangkat lunak, kualitas software dan jawaban dari siapakah seorang SQA *engineer*, pertanyaan nyata selanjutnya adalah apakah pekerjaan seorang SQA *engineer*?

Mengambil dari situs [3], terdapat lima kunci utama pekerjaan seorang rekayasawan mutu perangkat lunak. Lima kunci ini adalah:

1. Membentuk rencana-rencana uji coba sehingga sebuah perangkat lunak dapat bekerja dengan baik setelah pengembangan yang panjang
2. Menjaga pengembangan perangkat lunak tetap dalam lingkup dan persyaratan yang sesuai saat terjadi ada atau tidaknya perubahan atas permintaan pelanggan.
3. Menjadi jembatan komunikasi antara pelanggan dan tim pengembang setelah dan sebelum produk diberikan kepada pelanggan.
4. Menjadi perangkat tinjauan sebelum produk sampai kepada pelanggan sehingga dapat memberikan informasi produk secara lengkap akan kemampuan-kemampuannya, kekurangan yang ada ataupun peningkatan yang dapat ditambahkan di masa depan.

5. Menjaga seluruh tim pengembang tetap patuh dengan kebijakan perusahaan atau organisasi pengembang perangkat lunak.

Berdasarkan lima pernyataan diatas, pekerjaan seorang rekayasawan mutu perangkat lunak di seluruh tim pengembangan dapat dipastikan memiliki tanggung jawab yang sama dengan beberapa tambahan yang disesuaikan dengan kebutuhan perusahaan atau organisasi pengembang. Pernyataan diatas tersebut juga menjadi gambaran selanjutnya akan praktik nyata dilapangan. Sebelum dan sesudah sebuah perangkat lunak diluncurkan, seorang SQA *engineer* akan terus memantau perkembangan yang terjadi.

Dalam tahap pengembangan, seluruh persyaratan dan lingkup kemampuan perangkat lunak akan terus mendapat tinjauan sehingga tidak ada yang tertinggal dan dapat bekerja semestinya. Kemudian pada tahap produksi, produk akan terus diawasi jikalau ternyata terdapat kekurangan saat pemakaian dan jika pelanggan menginginkan kemampuan lebih banyak karena merasa kemampuan produk saat itu dapat ditingkatkan lebih lanjut, maka permintaan tersebut akan diteruskan oleh tim penjaminan mutu dan didiskusikan dengan tim pengembang. Pemantauan produk menjadi inti keseluruhan pekerjaan pada praktik dilapangan, sehingga sesuai dengan arti penjaminan mutu perangkat lunak tersebut, dari awal hingga akhir pengembangan dan sampai digunakan oleh pelanggan.

Bab 2.

Alur Pemrograman Secara Singkat

2.1. Variabel

Demi pembelajaran alur pemrograman, bahasa pemrograman Javascript digunakan supaya pembelajaran lebih nyaman untuk mengaplikasikan langsung kode-kode uji coba kedepannya. Hal ini tidak menutup kemungkinan bagi para pembaca yang sudah paham dengan alur pemrograman untuk membaca bab ini sehingga dapat mengerti sintak-sintak dari bahasa Javascript. Jika para pembaca merasa sudah akrab dengan bahasa Javascript, pembaca diperbolehkan melewati bab ini dan masuk pada bab selanjutnya yaitu Bab 3. Uji Coba, Kemampuan Utama

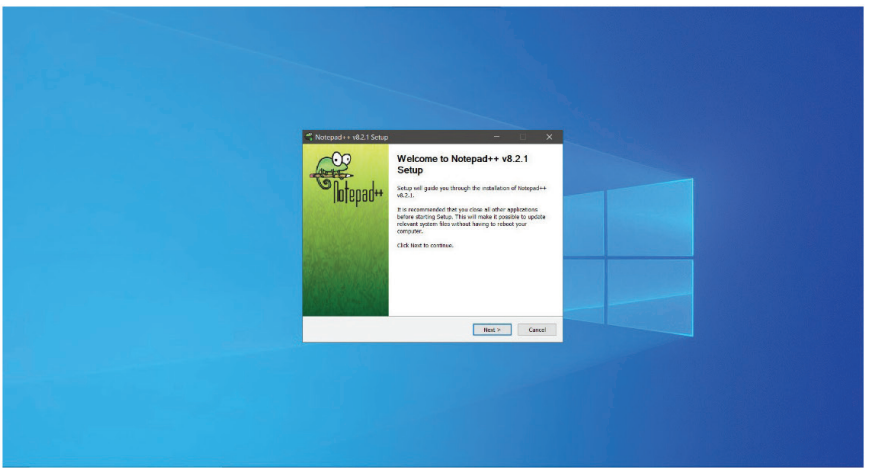
Javascript merupakan bahasa pemrograman untuk pengembangan web, biasanya dipakai untuk melakukan perubahan secara dinamis pada komponen web sehingga situs web dapat terlihat responsif saat pemakaian. Adapun selain melakukan perubahan secara dinamis pada komponen web, sebagai bahasa pemrograman secara umum Javascript dapat juga digunakan untuk melakukan komputasi, manipulasi dan validasi data [4]. Javascript sendiri diciptakan oleh Brendan Eich pada tahun 1995 sebagai alat untuk mengembangkan browser untuk perusahaan Netscape dengan produk browsernya bernama Netscape 2 [5].

Sebelum masuk lebih lanjut dalam mengenal Javascript, ada sebuah pertanyaan yang pernah penulis dapatkan seputar menjadi seorang SQA *engineer*. Apakah mengetahui bahasa pemrograman diperlukan jika

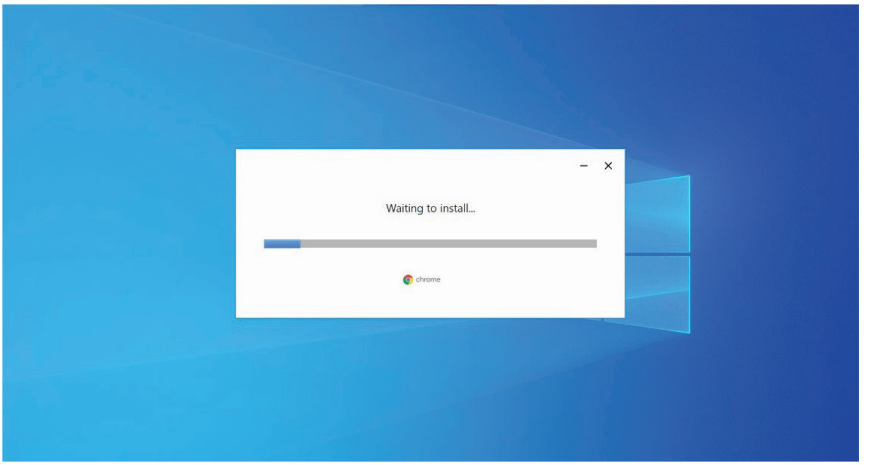
menjadi seorang SQA *engineer*? Jawabannya adalah seharusnya iya. Mengapa? Karena seorang rekayasawan penjaminan mutu perangkat lunak diharapkan dapat mengerti alur kerja dan logika sebuah perangkat lunak meskipun tidak diharuskan untuk mengetahui cara kerja seluruh sintak.

Dengan mengerti alur kerja perangkat lunak yang akan diuji cobakan, seorang SQA *engineer* juga diharapkan mampu mengotomatisasi pekerjaan uji coba. Terkait dengan tugas ini membuat SQA *engineer* membutuhkan kapabilitas dalam hal yang terkait dengan pembuatan skrip pemrograman sehingga otomatisasi bisa dilakukan. Proses otomatisasi akan membantu banyak bagi seorang SQA *engineer*, karena kuantitas tugas-tugas uji coba dilapangan yang tidak sedikit. Perlu juga diketahui bahwa seorang SQA *engineer* diharapkan mampu melakukan uji coba multi platform, yang terkadang melakukan hingga lima platform sekaligus.

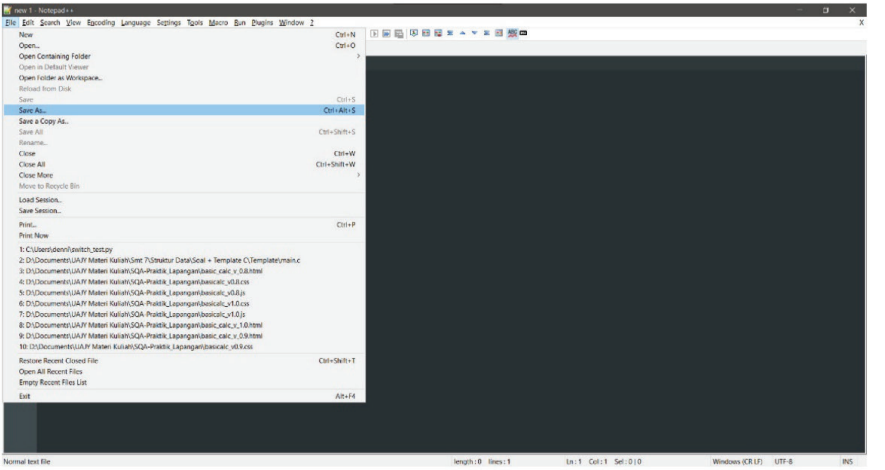
Berikut ini akan kami paparkan penggunaan aplikasi text editor, dengan contoh yang tervisualisasikan diharapkan pembaca dapat mengikuti pembelajaran lebih mudah. Lebih lanjut aplikasi editor seperti Notepad ++ dan sebuah browser (Edge, Firefox ataupun Chrome) dapat digunakan untuk membuat kode Javascript. Berikut cara untuk menyediakan tempat pembelajaran pengkodean Javascript.



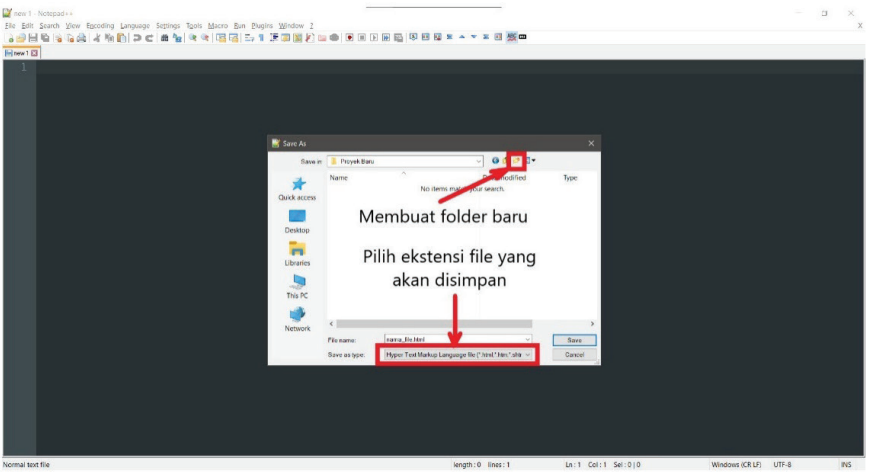
Gambar 2.1. Pemasangan Notepad ++ - Windows



Gambar 2.2. Pemasangan Browser Chrome - Windows



Gambar 2.3. Menyimpan File Pada Aplikasi Notepad ++



Gambar 2.4. Membuat Folder Dan Mengganti Ekstensi File

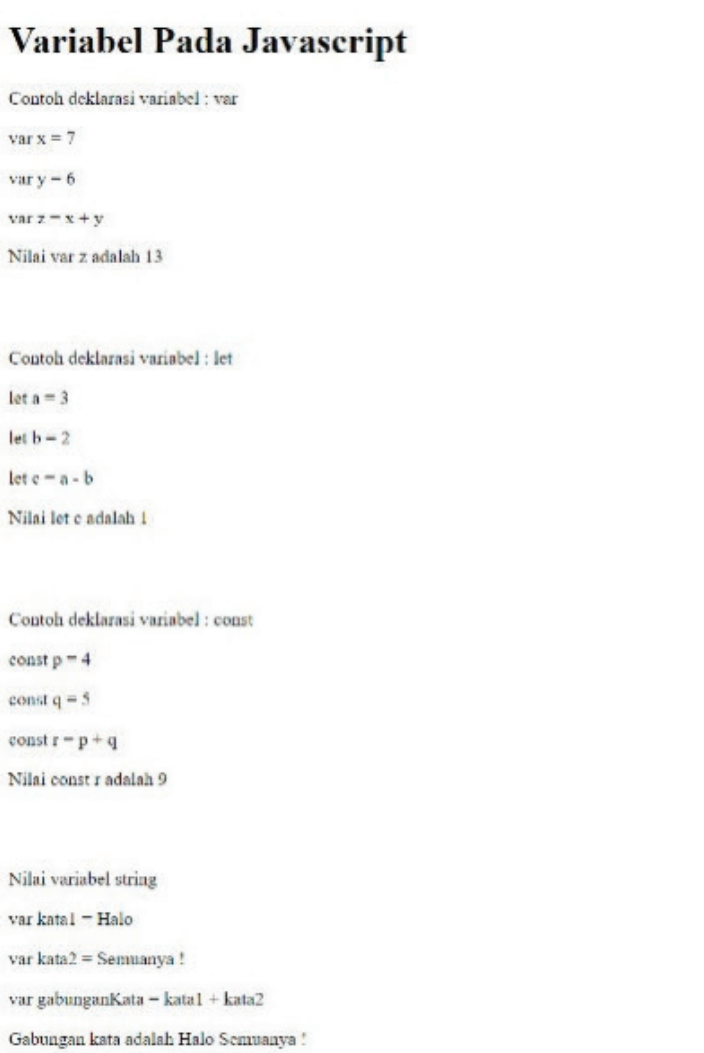
Variabel merupakan tempat bagi bahasa pemrograman untuk menyimpan data. Data yang disimpan merupakan sebuah nilai yang akan diberikan baik langsung pada saat pengkodean atau saat program berjalan. Deklarasi variabel pada Javascript sebagai berikut:

```
var x= 5;
var y = 8;
var z = x + y;

let x = 5;
let y = 8;
let z = x + y;

const x = 6;
const y = 9;
const z = x + y;
```

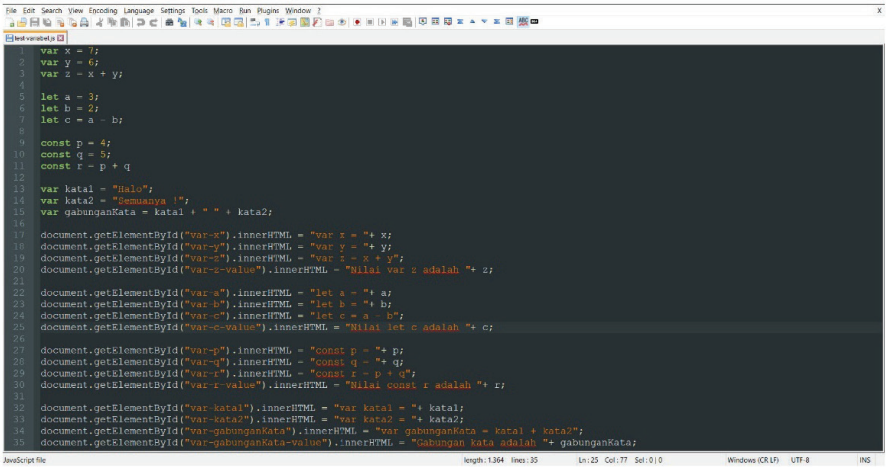
Selain dengan menggunakan kata `var`, `let` dan `const` untuk mendeklarasikan sebuah variabel, variabel dapat langsung ditulis hanya menggunakan nama variabel saja. Hal ini menandakan variabel tersebut belum dideklarasikan sama sekali atau sudah dideklarasikan sebelumnya, sebagai contoh :

```
x = 3;
y = 4;
z = x + y;
```

Apa perbedaan mendasar dari deklarasi variabel-variabel di atas dan kapan memakai deklarasi tersebut? Jawabannya, deklarasi variabel secara umum sama, tetapi perbedaan antara deklarasi `let` dan `const` adalah deklarasi dengan `let` digunakan jika nilai yang akan digunakan berubah dari waktu ke waktu dan `const` jika nilai tersebut tidak akan berubah (tetap). Deklarasi variabel dengan `const` menjadi pemakaian pada umumnya, tetapi deklarasi dengan var dapat juga dipakai [5].

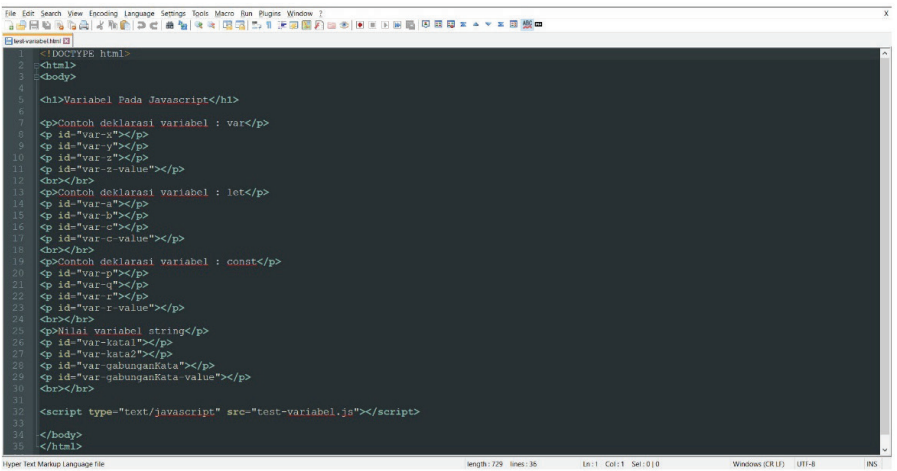


Gambar 2.5. Variabel Tertampil



```
var x = 7;
var y = 6;
var z = x + y;

let a = 3;
let b = 2;
let c = a - b;

const p = 4;
const q = 5;
const r = p + q

var kata1 = "Halo";
var kata2 = "Semuanya !";
var gabunganKata = kata1 + " " + kata2;

document.getElementById("var-x").innerHTML = "var x = "+ x;
document.getElementById("var-y").innerHTML = "var y = "+ y;
document.getElementById("var-z").innerHTML = "var z = x + y";
document.getElementById("var-z-value").innerHTML = "Nilai var z adalah "+ z;

document.getElementById("var-a").innerHTML = "let a = "+ a;
document.getElementById("var-b").innerHTML = "let b = "+ b;
document.getElementById("var-c").innerHTML = "let c = a - b";
document.getElementById("var-c-value").innerHTML = "Nilai let c adalah "+ c;

document.getElementById("var-p").innerHTML = "const p = "+ p;
document.getElementById("var-q").innerHTML = "const q = "+ q;
document.getElementById("var-r").innerHTML = "const r = p + q";
document.getElementById("var-r-value").innerHTML = "Nilai const r adalah "+ r;

document.getElementById("var-kata1").innerHTML = "var kata1 = "+ kata1;
document.getElementById("var-kata2").innerHTML = "var kata2 = "+ kata2;
document.getElementById("var-gabunganKata").innerHTML = "var gabunganKata = kata1 + kata2";
document.getElementById("var-gabunganKata-value").innerHTML = "Gabungan kata adalah "+ gabunganKata;
```

Gambar 2.6. Deklarasi Variabel Pada Javascript



```
<!DOCTYPE html>
<html>
<body>

<h1>Variabel Pada Javascript</h1>

<p>Contoh deklarasi variabel : var</p>
<p id="var-x"></p>
<p id="var-y"></p>
<p id="var-z"></p>
<p id="var-z-value"></p>
<br></br>
<p>Contoh deklarasi variabel : let</p>
<p id="var-a"></p>
<p id="var-b"></p>
<p id="var-c"></p>
<p id="var-c-value"></p>
<br></br>
<p>Contoh deklarasi variabel : const</p>
<p id="var-p"></p>
<p id="var-q"></p>
<p id="var-r"></p>
<p id="var-r-value"></p>
<br></br>
<p>Nilai variabel string</p>
<p id="var-kata1"></p>
<p id="var-kata2"></p>
<p id="var-gabunganKata"></p>
<p id="var-gabunganKata-value"></p>
<br></br>

<script type="text/javascript" src="test-variabel.js"></script>

</body>
</html>
```

Gambar 2.7. Tampilan Variabel Pada HTML

Terdapat variabel khusus dalam bahasa Javascript, variabel tersebut adalah variabel *array*. Variabel ini dapat menyimpan lebih dari satu nilai dalam deklarasinya. Deklarasi variabel ini sebagai berikut:

```
const newArray = ["Coffee", "Tea", "Water"];
const menuArray = [];
menuArray[0] = "Ice-cream";
menuArray[1] = "Cookies";
menurray[2] = "Crisp";
```

Supaya dapat mengakses nilai yang ada dalam sebuah array, maka cukup menyatakan variabel beserta dengan nomor indeks dari array tersebut.

Variabel Array Pada Javascript

Nilai-nilai Pada Indeks Array

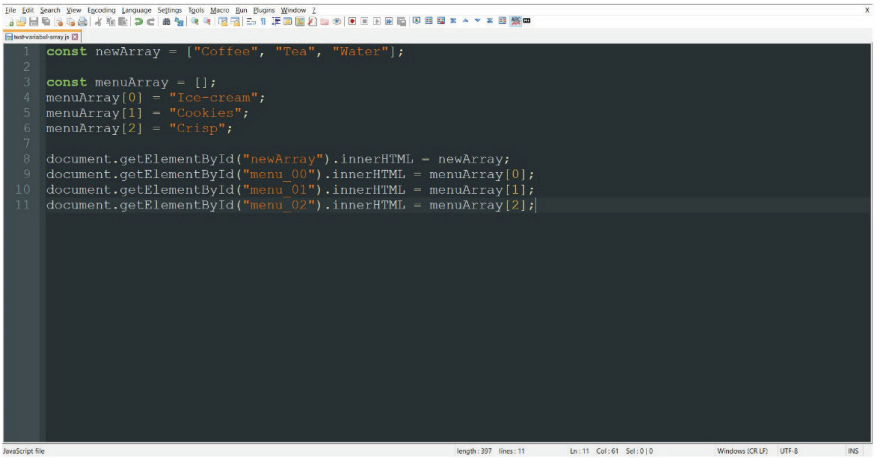
Coffee,Tea,Water

Ice-cream

Cookies

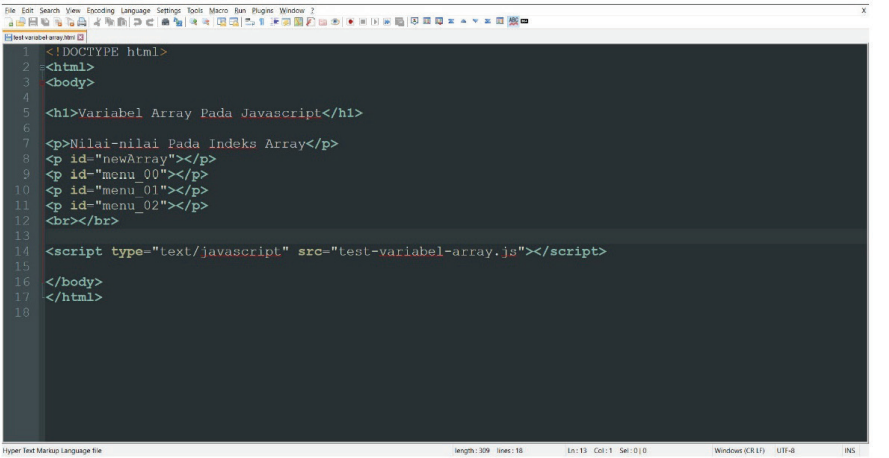
Crisp

Gambar 2.8. Tampilan Variabel Array Tertampil



```
const newArray = ["Coffee", "Tea", "Water"];

const menuArray = [];
menuArray[0] = "Ice-cream";
menuArray[1] = "Cookies";
menuArray[2] = "Crisp";

document.getElementById("newArray").innerHTML = newArray;
document.getElementById("menu_00").innerHTML = menuArray[0];
document.getElementById("menu_01").innerHTML = menuArray[1];
document.getElementById("menu_02").innerHTML = menuArray[2];
```

Gambar 2.9. Tampilan Variabel Array Pada Javascript



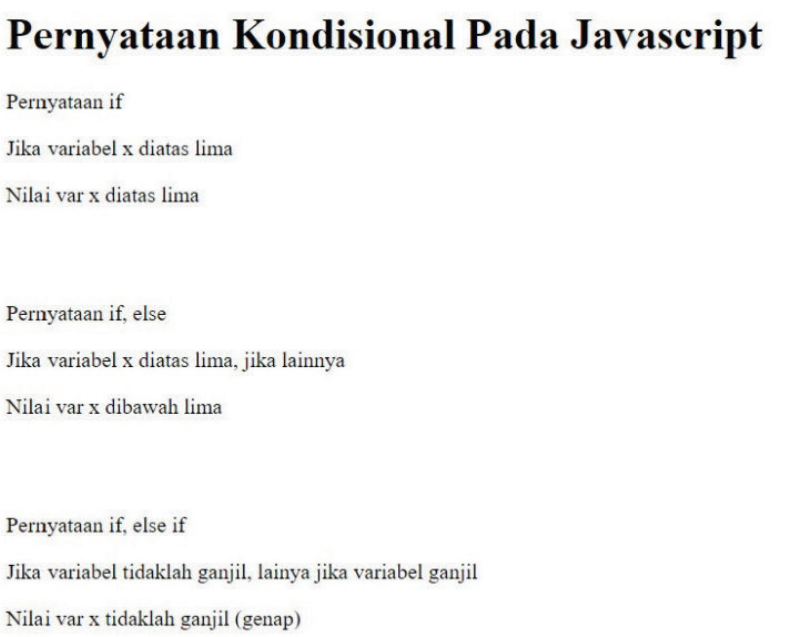
```
<!DOCTYPE html>
<html>
<body>

<h1>Variabel Array Pada Javascript</h1>

<p>Nilai-nilai Pada Indeks Array</p>
<p id="newArray"></p>
<p id="menu_00"></p>
<p id="menu_01"></p>
<p id="menu_02"></p>
<br></br>

<script type="text/javascript" src="test-variabel-array.js"></script>

</body>
</html>
```

Gambar 2.10. Tampilan Variabel Pada HTML

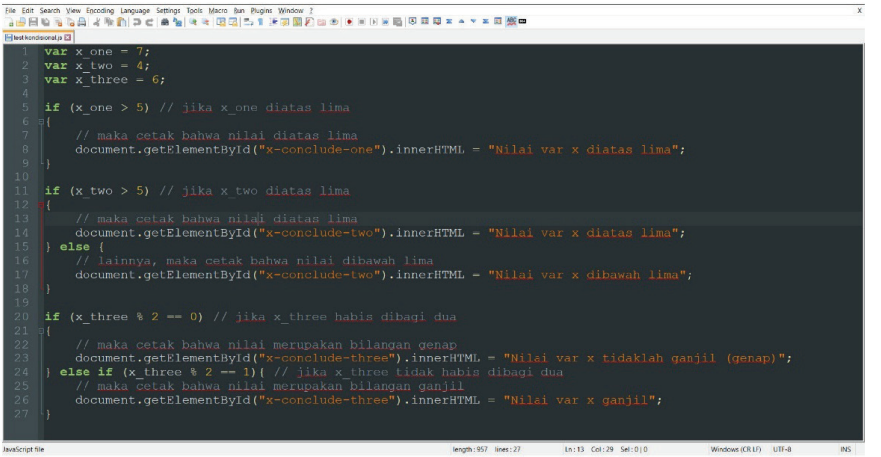
2.2. Pernyataan *if, else*

Pernyataan kondisional merupakan pernyataan yang dibutuhkan ketika suatu aksi membutuhkan keputusan yang berbeda ketika suatu syarat terpenuhi atau tidak terpenuhi. Pernyataan kondisional pada Javascript sebagai berikut:

```
if (kondisi_pertama_terpenuhi) {
// maka aksi yang dilakukan
}
if (kondisi_pertama_terpenuhi) {
    // maka aksi yang dilakukan
} else { // jika lainnya
    // maka aksi lain yang dilakukan
}
if (kondisi_pertama_terpenuhi) {
    // maka aksi pertama yang dilakukan
}
else if (kondisi_kedua_terpenuhi) {
    // maka aksi kedua yang dilakukan
}
```

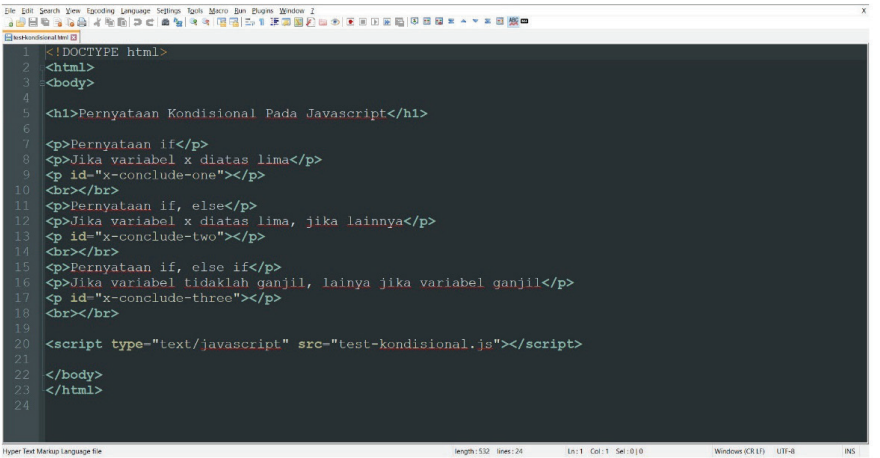


Gambar 2.8. Pernyataan Kondisional Tertampil



```
var x_one = 7;
var x_two = 4;
var x_three = 6;

if (x_one > 5) // jika x_one diatas lima
{
    // maka cetak bahwa nilai diatas lima
    document.getElementById("x-conclude-one").innerHTML = "Nilai var x diatas lima";
}

if (x_two > 5) // jika x_two diatas lima
{
    // maka cetak bahwa nilai diatas lima
    document.getElementById("x-conclude-two").innerHTML = "Nilai var x diatas lima";
} else {
    // lainnya, maka cetak bahwa nilai dibawah lima
    document.getElementById("x-conclude-two").innerHTML = "Nilai var x dibawah lima";
}

if (x_three % 2 == 0) // jika x_three habis dibagi dua
{
    // maka cetak bahwa nilai merupakan bilangan genap
    document.getElementById("x-conclude-three").innerHTML = "Nilai var x tidaklah ganjil (genap)";
} else if (x_three % 2 == 1){ // jika x_three tidak habis dibagi dua
    // maka cetak bahwa nilai merupakan bilangan ganjil
    document.getElementById("x-conclude-three").innerHTML = "Nilai var x ganjil";
}
```

Gambar 2.9. Tampilan Pernyataan Kondisional Pada Javascript



```
<!DOCTYPE html>
<html>
<body>

<h1>Pernyataan Kondisional Pada Javascript</h1>

<p>Pernyataan if</p>
<p>Jika variabel x diatas lima</p>
<p id="x-conclude-one"></p>
<br></br>
<p>Pernyataan if, else</p>
<p>Jika variabel x diatas lima, jika lainnya</p>
<p id="x-conclude-two"></p>
<br></br>
<p>Pernyataan if, else if</p>
<p>Jika variabel tidaklah ganjil, lainya jika variabel ganjil</p>
<p id="x-conclude-three"></p>
<br></br>

<script type="text/javascript" src="test-kondisional.js"></script>

</body>
</html>
```

Gambar 2.10. Tampilan Pernyataan Kondisional Pada HTML

2.3. Perulangan *for* dan *while*

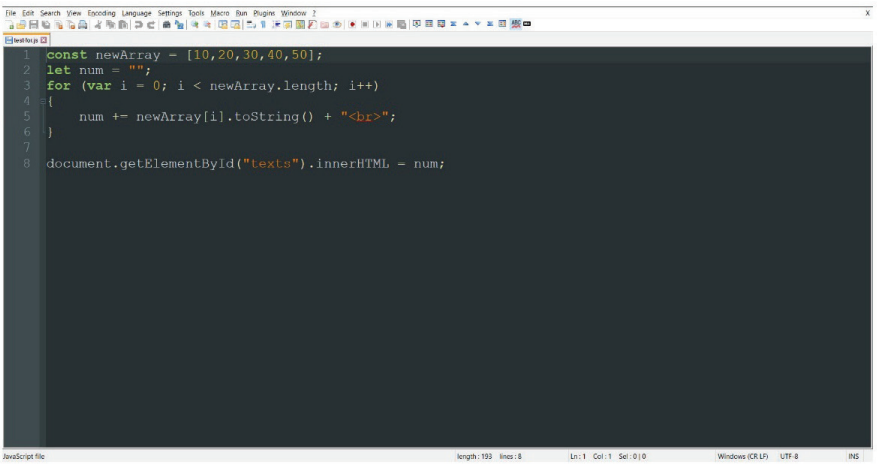
Perulangan dapat mempermudah sebuah aksi yang melakukan tugas yang sama dengan penulisan kode yang lebih sedikit. Penulisan pengulangan *for* pada Javascript sebagai berikut:

```
for (var i = 0; i < var_max; i++) {
// maka lakukan aksi ini
}
```

Perulangan for Pada Javascript

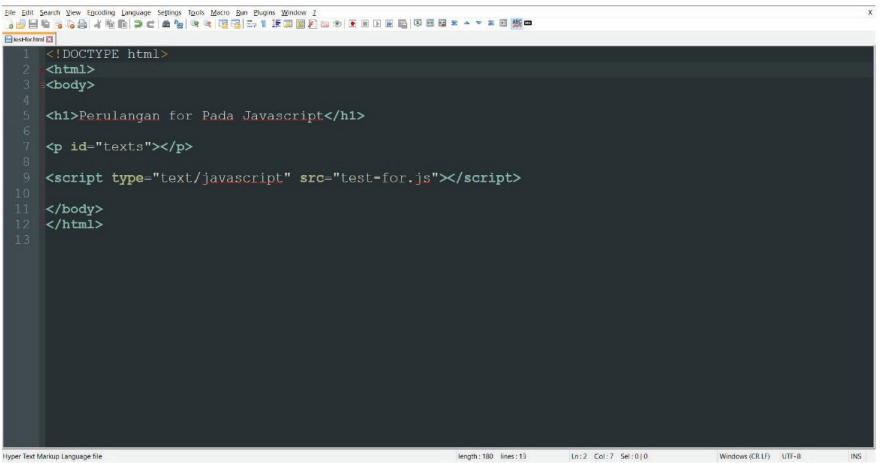
10
20
30
40
50

Gambar 2.11. Perulangan *For* Tertampil



```
const newArray = [10,20,30,40,50];
let num = "";
for (var i = 0; i < newArray.length; i++)
{
    num += newArray[i].toString() + "<br>";
}

document.getElementById("texts").innerHTML = num;
```

Gambar 2.12. Tampilan Perulangan *For* Pada Javascript



```
<!DOCTYPE html>
<html>
<body>

<h1>Perulangan for Pada Javascript</h1>

<p id="texts"></p>

<script type="text/javascript" src="test-for.js"></script>

</body>
</html>
```

Gambar 2.13. Tampilan Perulangan *For* Pada HTML

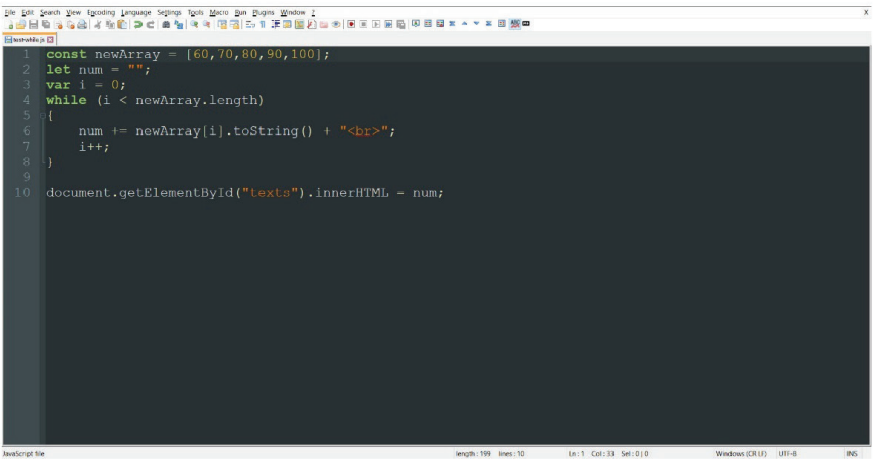
Perulangan *while* pada Javascript sebagai berikut:

```
while (i < var_max) {
    // maka lakukan aksi ini
i++;
}
```

Perulangan while Pada Javascript

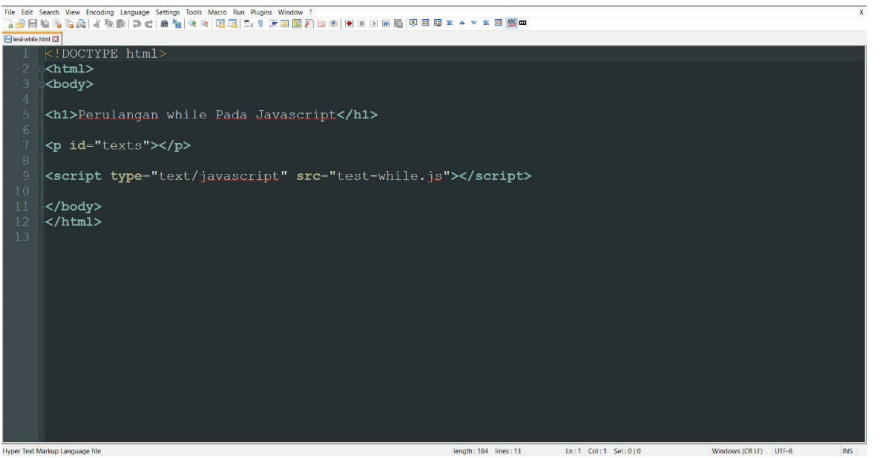
60
70
80
90
100

Gambar 2.14. Perulangan *While* Tertampil



```
const newArray = [60,70,80,90,100];
let num = "";
var i = 0;
while (i < newArray.length)
{
    num += newArray[i].toString() + "<br>";
    i++;
}

document.getElementById("texts").innerHTML = num;
```

Gambar 2.15. Tampilan Perulangan *While* Pada Javascript



```
<!DOCTYPE html>
<html>
<body>

<h1>Perulangan while Pada Javascript</h1>

<p id="texts"></p>

<script type="text/javascript" src="test-while.js"></script>

</body>
</html>
```

Gambar 2.16. Tampilan Perulangan *While* Pada HTML

2.4. Fungsi

Fungsi adalah satu blok kode yang menyelesaikan sebuah tugas tertentu. Fungsi dapat berisi parameter bawaan, ataupun tidak memiliki parameter sama sekali. Nilai hasil dari sebuah fungsi dapat berupa nilai dengan tipe data tertentu, obyek ataupun tidak memberikan sebuah nilai sama sekali. Pada Javascript penulisan sebuah fungsi terlihat sebagai berikut:

```
function named_function(param_one, param_two) {
// fungsi akan memberikan (mengembalikan)
sebuah nilai di akhir perjalanan kode
return param_one * param_two;
}
```

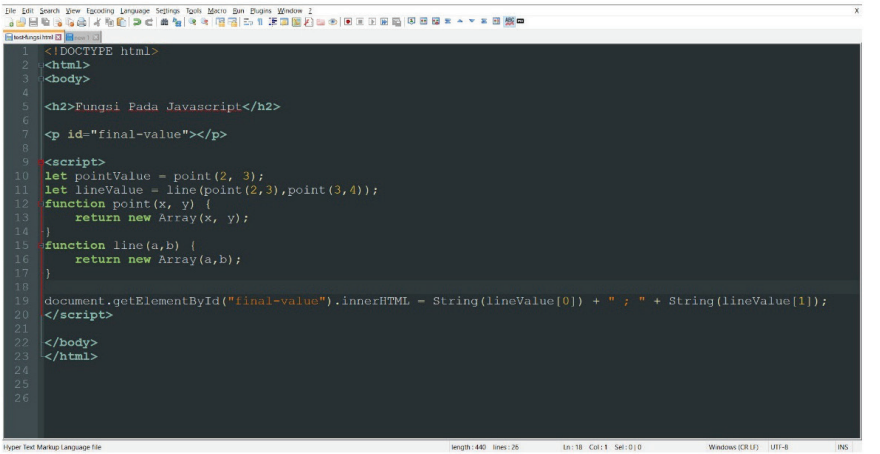
Jika ingin memanggil fungsi yang sudah dibuat, maka penulisannya seperti berikut :

```
function point(x, y) {
return new Array(x,y);
}
let pointValue = point(2, 3);
```

Karena fungsi dapat mengembalikan sebuah nilai, maka sebuah fungsi juga dapat dipasangkan kedalam sebuah variabel. Selain itu, fungsi juga dapat langsung dipasangkan ke fungsi dikarenakan ketika fungsi dipanggil, nilai hasil yang diberikan akan langsung dipasangkan kedalam parameter fungsi. Berikut contohnya:

```
function point(x, y) {
    return new Array(x, y);
}
function line(a,b) {
    return new Array(a,b);
}
let pointValue = point(2, 3);
let lineValue = line(point(2,3),point(3,4));
```

Gambar 2.17. Fungsi Tertampil



```
<!DOCTYPE html>
<html>
<body>

<h2>Fungsi Pada Javascript</h2>

<p id="final-value"></p>

<script>
let pointValue = point(2, 3);
let lineValue = line(point(2,3),point(3,4));
function point(x, y) {
    return new Array(x, y);
}
function line(a,b) {
    return new Array(a,b);
}

document.getElementById("final-value").innerHTML = String(lineValue[0]) + " ; " + String(lineValue[1]);
</script>

</body>
</html>
```

Gambar 2.18. Tampilan Fungsi Pada HTML

2.5. Obyek

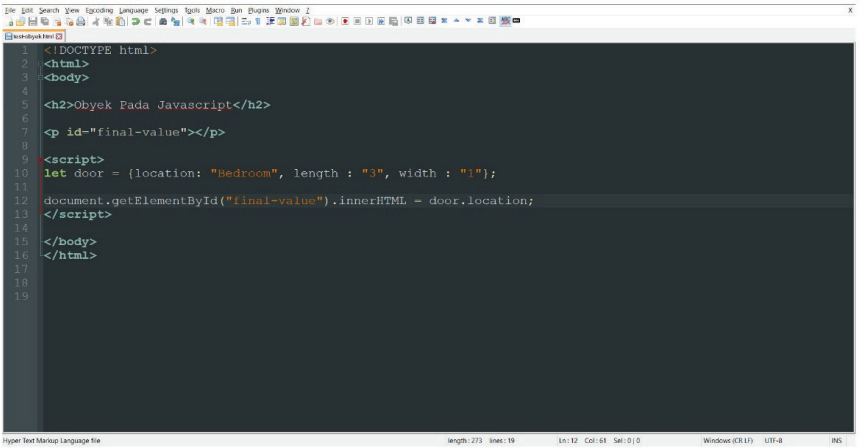
Obyek dalam kehidupan nyata merupakan kata untuk menunjukkan sebuah benda. Seperti contoh sebuah pintu. Pintu dapat dilihat, dibuka dan ditutup. Layaknya seperti pada kehidupan nyata, obyek dalam pemrograman memiliki makna yang sama sehingga membuat sebuah obyek praktis memiliki properti / atribut yang terpasang dan metode / aksi yang dapat dilakukan. Metode-metode sebuah obyek biasanya berupa kumpulan fungsi ataupun prosedur. Pembuatan obyek pada Javascript sebagai berikut :

```
let door = {location: "Bedroom", length :"3",
width : "1"};
```

Pemanggilan properti sebuah obyek dapat dilihat sebagai berikut:

```
door.location;
door["location"];
```

Gambar 2.19. Properti Obyek Tertampil

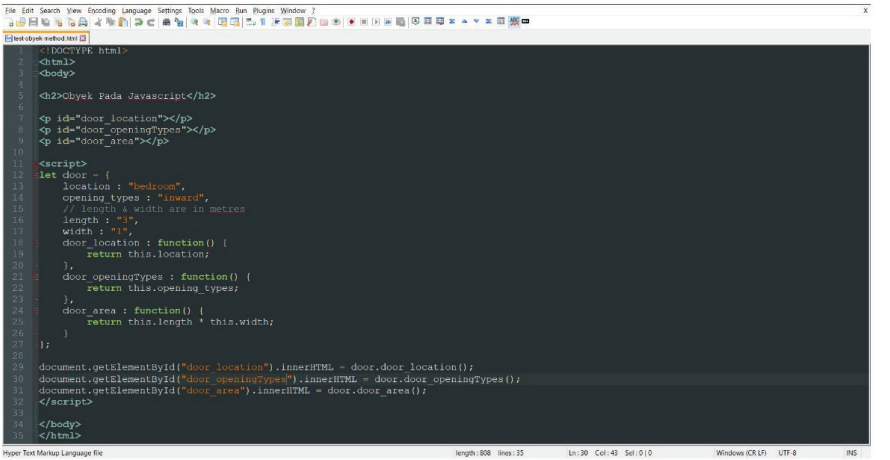


Gambar 2.20. Tampilan Properti Obyek Pada HTML

Jika sebuah obyek memiliki beberapa metode, maka pemanggilan metode tersebut sama seperti memanggil sebuah properti. Berikut contoh obyek dengan metode dan pemanggilannya:

```
let door = {
location : "bedroom",
    opening_types : "inward",
    // length & width are in metres
length : "3",
width : "1",
    door_location :function() {
return this.location;
    },
    door_openingTypes : function() {
        return this.opening_types;
    },
    door_area : function() {
        return this.length * this.width;
    }
};
door.door_location();
door.door_area();
door.door_openingTypes();
```

Sintak `this` pada obyek merupakan sintak untuk menunjukkan obyek tersebut yang sedang dipakai. Dalam arti lain menunjuk pada obyek "ini".



```
<!DOCTYPE html>
<html>
<body>

<h2>Obyek Pada Javascript</h2>

<p id="door_location"></p>
<p id="door_openingTypes"></p>
<p id="door_area"></p>

<script>
let door = {
    location : "bedroom",
    opening_types : "inward",
    // length & width are in metres
    length : "3",
    width : "1",
    door_location : function() {
        return this.location;
    },
    door_openingTypes : function() {
        return this.opening_types;
    },
    door_area : function() {
        return this.length * this.width;
    }
};

document.getElementById("door_location").innerHTML = door.door_location();
document.getElementById("door_openingTypes").innerHTML = door.door_openingTypes();
document.getElementById("door_area").innerHTML = door.door_area();
</script>

</body>
</html>
```

Gambar 2.21. Tampilan Metode Obyek Pada HTML

Obyek Pada Javascript

bedroom

inward

3

Gambar 2.22. Metode Obyek Tertampil

2.6. Latihan Pemrograman

Latihan pemrograman dengan Javascript ini akan menguji tingkat kepahaman pembaca dalam sintak-sintak dasar pada Javascript. Jika pembaca menemukan kesulitan dalam membuat programnya, pembaca dapat melihat solusi diakhir subbab ini atau dapat melihat referensi dan dokumentasi pemrograman Javascript.

Buatlah dua buah obyek bernama pembeli, dan penjual mie pangsit. Obyek pembeli memiliki properti berupa nama depan (tipe data string),nama belakang (tipe data string), nomor antrian (tipe data integer, nomor bebas berurut) dan jumlah pemesanan (tipe data integer). Metode-metode didalam obyek adalah fungsi mengembalikan nomor antri pembeli (nama fungsi bebas),fungsi mengembalikan nama pembeli (nama fungsi bebas) dan fungsi mengembalikan jumlah pesanan (nama fungsi bebas).

Obyek penjual memiliki properti berupa harga pangsit (tipe data integer) dan nama pegawai (tipe data string). Metode-metode didalam obyek adalah fungsi mengembalikan harga pangsit (nama fungsi bebas) dan fungsi mencetak nama pegawai (nama fungsi bebas). Adapun fungsi global yaitu fungsi untuk mengembalikan total harga (nama fungsi bebas) dan mencetak struk pemesanan (nama fungsi bebas).

Program akan memiliki interaksi antar obyek sebagai berikut. Pembeli akan membuat nama, nomor antrian dan jumlah pesanan. Penjual akan membuat nama dan harga. Kemudian data-data obyek tersebut akan diolah didalam fungsi global. Fungsi yang mengembalikan totalharga akan mengolah data yang diterima dari pembeli berupa jumlah pemesanan dan dari penjual berupa total harga.Jumlah obyek pembeli maksimal lima sebagai bentuk latihan dengan seluruh obyek pembeli dimasukkan kedalam sebuah `array`. Fungsi global akan mengiterasi array tersebut dan mencetak informasi struk secara berurut. Informasi struk dibuat kedalam sebuah `array` untuk menghindari kesulitan dalam mencetak seluruh informasi struk.

Kesimpulan dari latihan pemrograman adalah pembaca diharapkan mengetahui pemrograman dasar dalam bahasa Javascript sehingga dapat mengerti alur kerja sebuah program saat dilakukan pengujicobaan serta pembaca dapat dengan nyaman untuk membuat alur pemrograman kecil dengan mandiri ketika melakukan pengujicobaan secara otomatis.

Solusi dari latihan pemrograman Javascript:

Latihan Pemrograman Javascript

Gambar 2.23. Hasil Cetak Struk

```html
<!DOCTYPE html>
<html>
    <!-- Warung Mie Pangsit Ada/-->
    <head>
        <title>Warung Mie Pangsit Ada</title>
    </head>
    <body>

    <h2 style="text-align:center">Latihan Pemrograman Javascript</h2>

    <div style="text-align:center" id="cetak"></div>

    <script type="text/javascript" src="lat_js.js"></script>

</body>
</html>
```

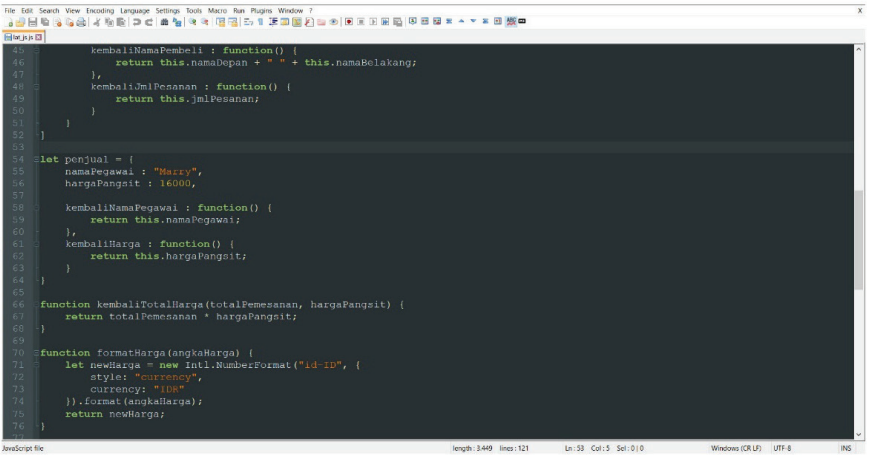
Gambar 2.24. HTML Cetak Struk

```
inisialisasiPembeliArr = [
    pembeli_1 = {
        namaDepan : "John",
        namaBelakang : "Doe",
        noAntrian : 1,
        jmlPesanan : 2,

        kembaliNoAntrian : function() {
            return this.noAntrian;
        },
        kembaliNamaPembeli : function() {
            return this.namaDepan + " " + this.namaBelakang;
        },
        kembaliJmlPesanan : function() {
            return this.jmlPesanan;
        }
    },

    pembeli_2 = {
        namaDepan : "Samy",
        namaBelakang : "MoD",
        noAntrian : 2,
        jmlPesanan : 3,

        kembaliNoAntrian : function() {
            return this.noAntrian;
        },
        kembaliNamaPembeli : function() {
            return this.namaDepan + " " + this.namaBelakang;
        },
        kembaliJmlPesanan : function() {
            return this.jmlPesanan;
```

Gambar 2.25. Inisialisasi Obyek 1

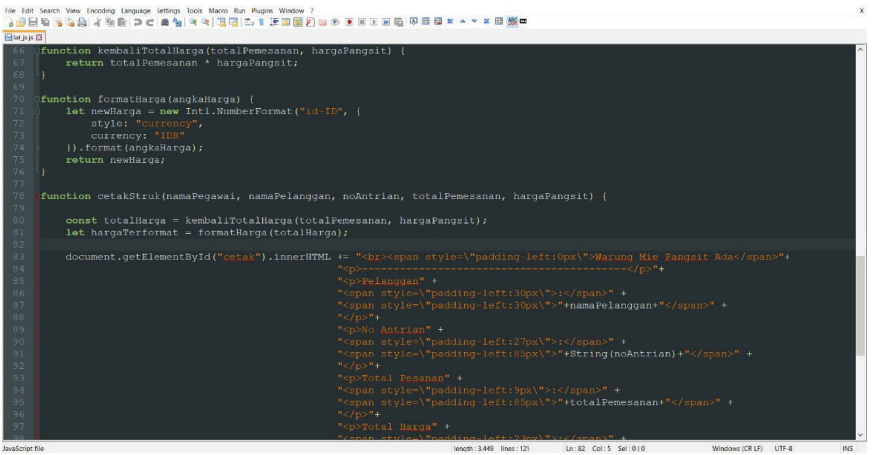
```
        kembaliNoAntrian : function() {
            return this.noAntrian;
        },
        kembaliNamaPembeli : function() {
            return this.namaDepan + " " + this.namaBelakang;
        },
        kembaliJmlPesanan : function() {
            return this.jmlPesanan;
        }
    },

    pembeli_3 = {
        namaDepan : "Irv",
        namaBelakang : "Whit",
        noAntrian : 3,
        jmlPesanan : 1,

        kembaliNoAntrian : function() {
            return this.noAntrian;
        },
        kembaliNamaPembeli : function() {
            return this.namaDepan + " " + this.namaBelakang;
        },
        kembaliJmlPesanan : function() {
            return this.jmlPesanan;
        }
    }
]

let penjual = {
    namaPegawai : "Marry",
    hargaPangsit : 16000,
```

Gambar 2.26. Inisialisasi Obyek 2



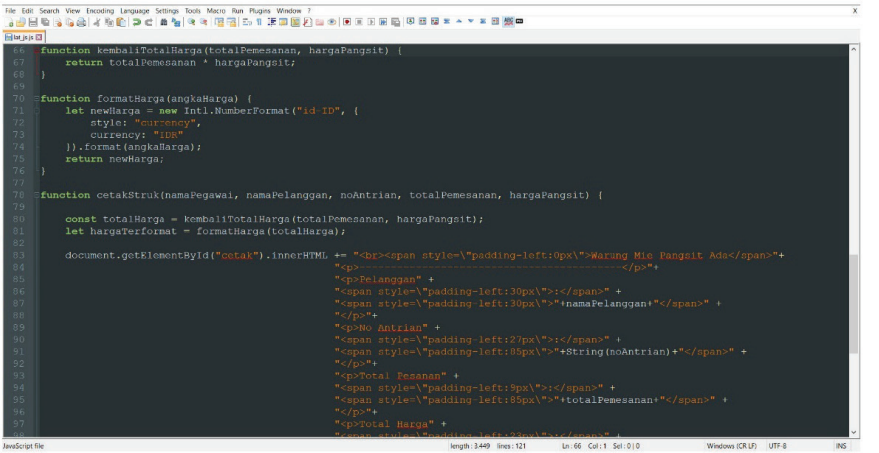
```
        kembaliNamaPembeli : function() {
            return this.namaDepan + " " + this.namaBelakang;
        },
        kembaliJmlPesanan : function() {
            return this.jmlPesanan;
        }
    }
]

let penjual = {
    namaPegawai : "Marry",
    hargaPangsit : 16000,

    kembaliNamaPegawai : function() {
        return this.namaPegawai;
    },
    kembaliHarga : function() {
        return this.hargaPangsit;
    }
}

function kembaliTotalHarga(totalPemesanan, hargaPangsit) {
    return totalPemesanan * hargaPangsit;
}

function formatHarga(angkaHarga) {
    let newHarga = new Intl.NumberFormat("id-ID", {
        style: "currency",
        currency: "IDR"
    }).format(angkaHarga);
    return newHarga;
}
```

Gambar 2.27. Inisialisasi Variabel 3



```
function kembaliTotalHarga(totalPemesanan, hargaPangsit) {
    return totalPemesanan * hargaPangsit;
}

function formatHarga(angkaHarga) {
    let newHarga = new Intl.NumberFormat("id-ID", {
        style: "currency",
        currency: "IDR"
    }).format(angkaHarga);
    return newHarga;
}

function cetakStruk(namaPegawai, namaPelanggan, noAntrian, totalPemesanan, hargaPangsit) {

    const totalHarga = kembaliTotalHarga(totalPemesanan, hargaPangsit);
    let hargaTerformat = formatHarga(totalHarga);

    document.getElementById("cetak").innerHTML += "<br><span style=\"padding-left:0px\">Warung Mie Pangsit Ada</span>"+
                                                  "<p>----------------------------------------</p>"+
                                                  "<p>Pelanggan" +
                                                  "<span style=\"padding-left:30px\">:</span>" +
                                                  "<span style=\"padding-left:30px\">"+namaPelanggan+"</span>" +
                                                  "</p>"+
                                                  "<p>No Antrian" +
                                                  "<span style=\"padding-left:27px\">:</span>" +
                                                  "<span style=\"padding-left:85px\">"+String(noAntrian)+"</span>" +
                                                  "</p>"+
                                                  "<p>Total Pesanan" +
                                                  "<span style=\"padding-left:9px\">:</span>" +
                                                  "<span style=\"padding-left:85px\">"+totalPemesanan+"</span>" +
                                                  "</p>"+
                                                  "<p>Total Harga" +
```

Gambar 2.28. Fungsi-Fungsi 1



```
function kembaliTotalHarga(totalPemesanan, hargaPangsit) {
    return totalPemesanan * hargaPangsit;
}

function formatHarga(angkaHarga) {
    let newHarga = new Intl.NumberFormat("id-ID", {
        style: "currency",
        currency: "IDR"
    }).format(angkaHarga);
    return newHarga;
}

function cetakStruk(namaPegawai, namaPelanggan, noAntrian, totalPemesanan, hargaPangsit) {

    const totalHarga = kembaliTotalHarga(totalPemesanan, hargaPangsit);
    let hargaTerformat = formatHarga(totalHarga);

    document.getElementById("cetak").innerHTML += "<br><span style=\"padding-left:0px\">Warung Mie Pangsit Ada</span>"+
                                                  "<p>----------------------------------------</p>"+
                                                  "<p>Pelanggan" +
                                                  "<span style=\"padding-left:30px\">:</span>" +
                                                  "<span style=\"padding-left:30px\">"+namaPelanggan+"</span>" +
                                                  "</p>"+
                                                  "<p>No Antrian" +
                                                  "<span style=\"padding-left:27px\">:</span>" +
                                                  "<span style=\"padding-left:85px\">"+String(noAntrian)+"</span>" +
                                                  "</p>"+
                                                  "<p>Total Pesanan" +
                                                  "<span style=\"padding-left:9px\">:</span>" +
                                                  "<span style=\"padding-left:85px\">"+totalPemesanan+"</span>" +
                                                  "</p>"+
                                                  "<p>Total Harga" +
```

Gambar 2.29. Fungsi-Fungsi 2

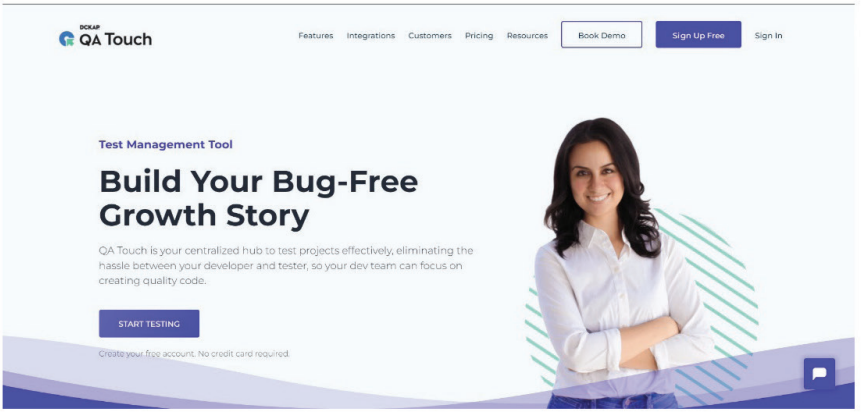
Bab 3.

Uji Coba, Kemampuan Utama

3.1. Alat Manajemen *Test Case*

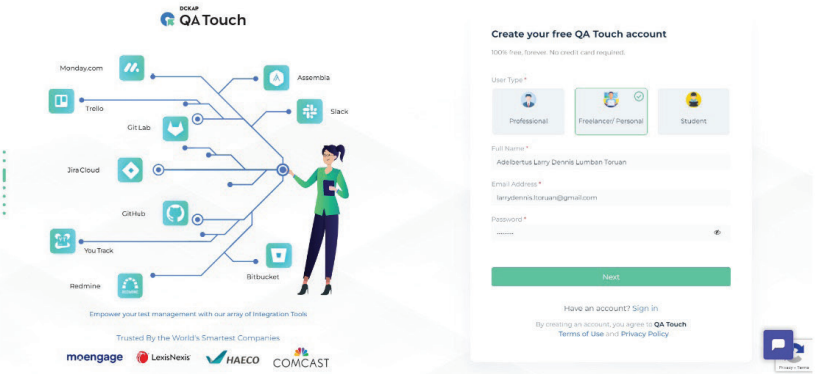
Alat manajemen *test case* pada dasarnya memiliki fungsi untuk mengatur uji coba-uji coba yang akan dilakukan baik secara manual ataupun secara otomatis. Uji coba yang diatur memiliki informasi pada umumnya seperti nama uji coba dan langkah-langkah uji coba. Informasi tambahan lain mengenai uji coba umumnya dapat ditambahkan dalam alat manajemen uji coba, bahkan dimungkinkan untuk menambahkan skor uji coba tersebut jika menginginkan urutan prioritas uji coba. Alat manajemen uji coba yang akan digunakan didalam buku ini adalah perangkat lunak QA Touch (https://www.qatouch.com).

QA Touch dapat digunakan secara gratis untuk penggunaan pribadi dan memiliki tampilan yang ramah terhadap pengguna. Langkah pendaftaran dapat dilakukan dengan cara berikut:



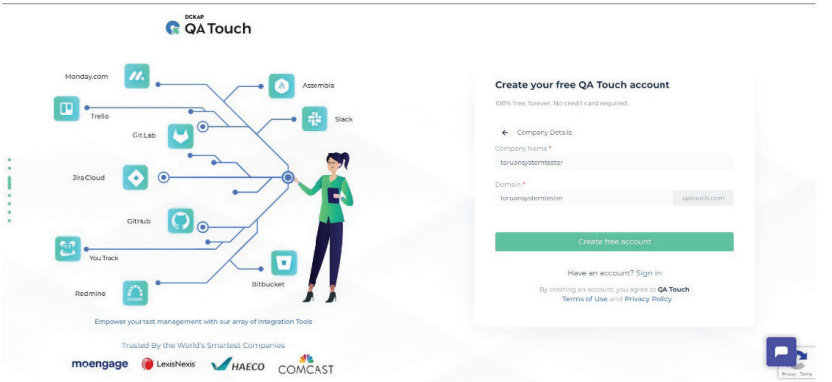
Gambar 3.1. Halaman Utama QA Touch

1. Pada halaman utama QA Touch, tekantombol 'Sign Up Free' atau klik tautan :https://register.qatouch.com/



Gambar 3.1. Halaman Registrasi

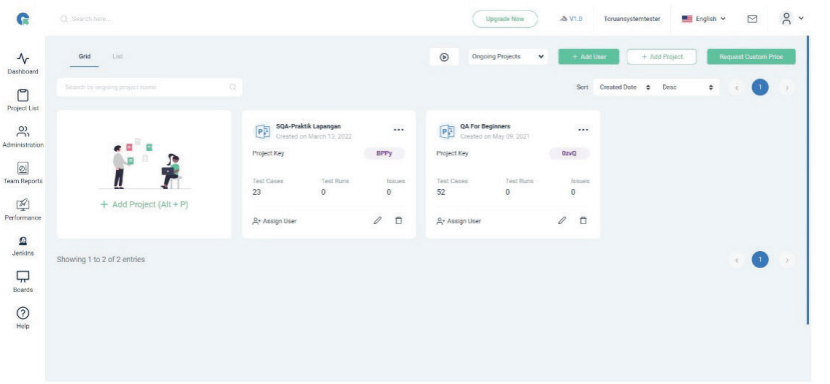
2. Setelah sampai pada halaman registrasi, pada kanan bawah halaman web, isikan nama lengkap, alamat email dan kata sandi
3. Klik tombol 'Next'



Gambar 3.2. Halaman Registrasi, Informasi Lanjutan

4. Selanjutnya, akan sampai pada halaman untuk mengisikan nama perusahaan dan domain. Kedua informasi dapat diisi sesuai keinginan pembaca.
5. Klik tombol *'Create free account'*
6. Setelah pembuatan akun, verifikasi pembuatan akun melalui email yang sudah dikirimkan

7. Jika semua tahap diatas sudah diselesaikan, maka dapat login dengan akun yang sudah dibuat.Tampilan ruang kerja utama akan muncul setelah login.

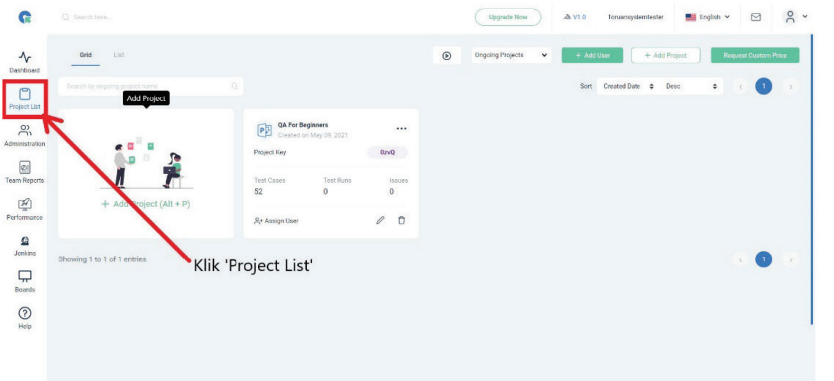


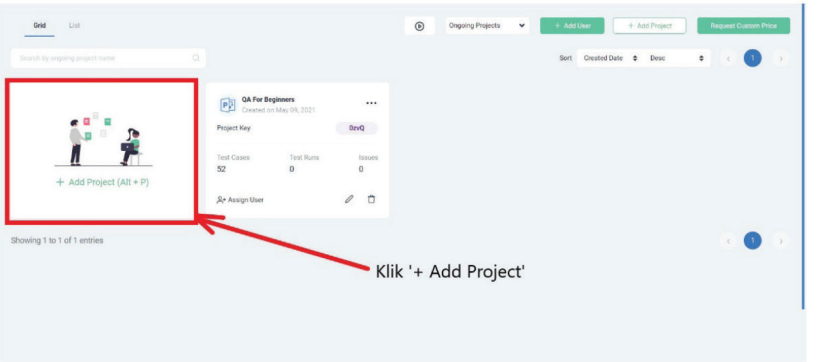
Gambar 3.3. Tampilan Ruang Kerja Utama QA Touch

QA Touch memliki beberapa fitur yang dapat dilihat melalui *navgiation bar* disisi kiri, seperti daftar-daftar proyek yang dikerjakan sebagai fitur utama, laporan tim, performa tim dan integrasi Jenkins sebagai fitur tambahan terbaru. Fitur utama hanya menjadi lingkup penggunaan dalam latihan pembuatan *test case*.

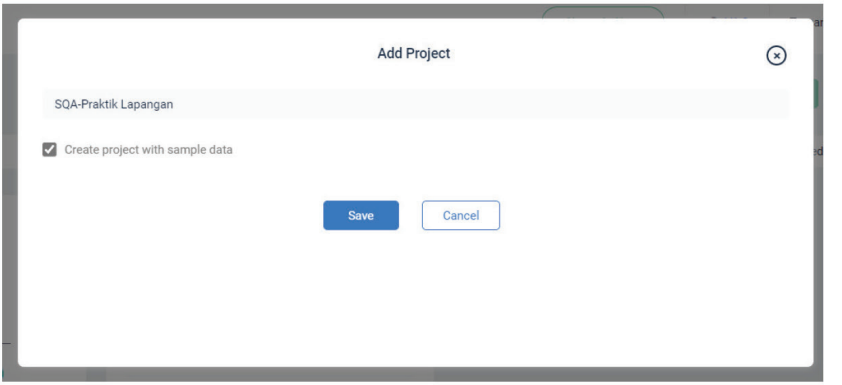
Pembuatan sebuah test case harus terlebih dahulu dilakukan pembuatan folder proyek. Langkahnya sebagai berikut:

1. Klik tab 'Project List' yang berada pada *navigation bar*.

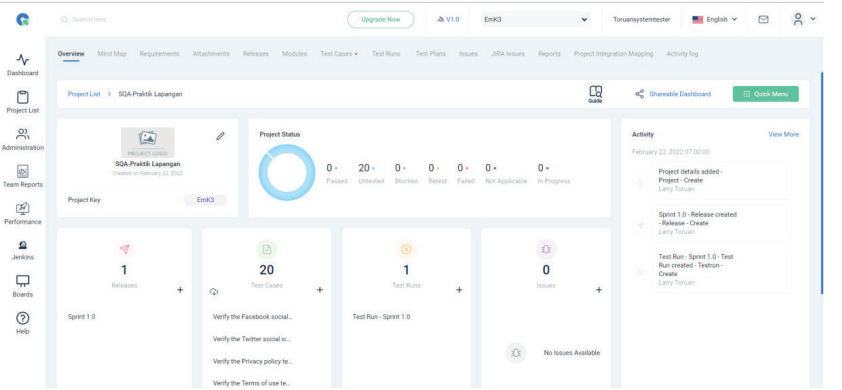




2. Pada tampilan utama ‘Project List’, klik obyek ‘+ Add Project’.

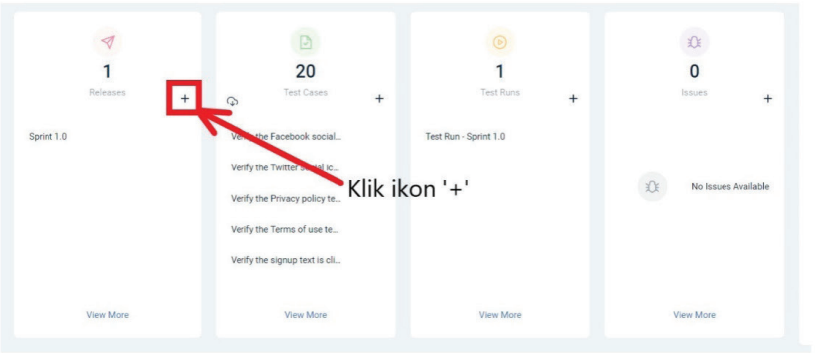


3. Muncul sebuah kotak untuk pengisian informasi sebuah proyek. Isi nama proyek tersebut kemudian klik tombol ‘Save’

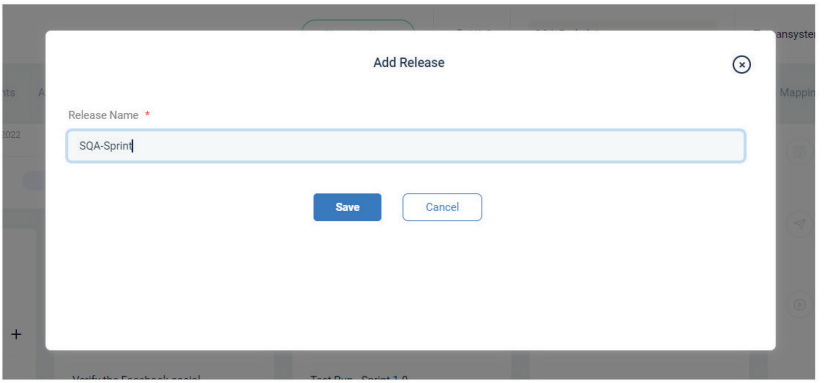


4. Setelah membuat direktori proyek, maka akan diarahkan kedalam tampilan folder proyek.

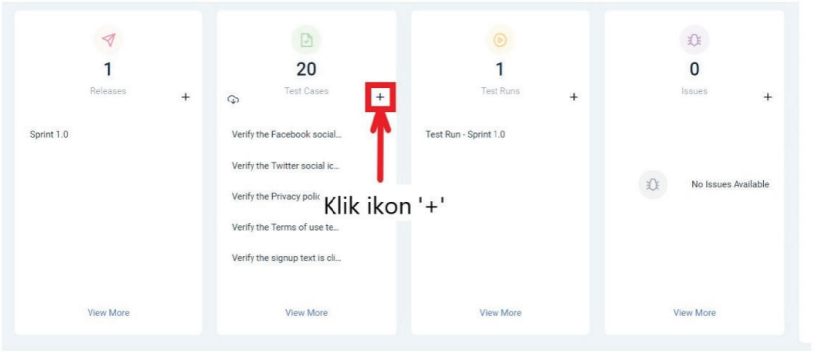
5. Didalam direktori proyek terdapat subdirektori seperti ‘Releases’, ‘Test Cases’, ‘Test Runs’ dan ‘Issues’.

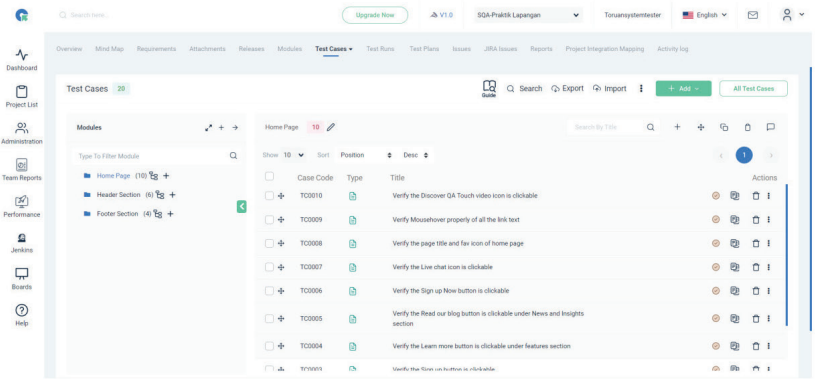


6. Klik ikon ‘+’ pada obyek ‘Releases’. Ikon ini akan menampilkan kotak pengisian informasi rilisan. Isikan nama rilisan dan klik tombol simpan.

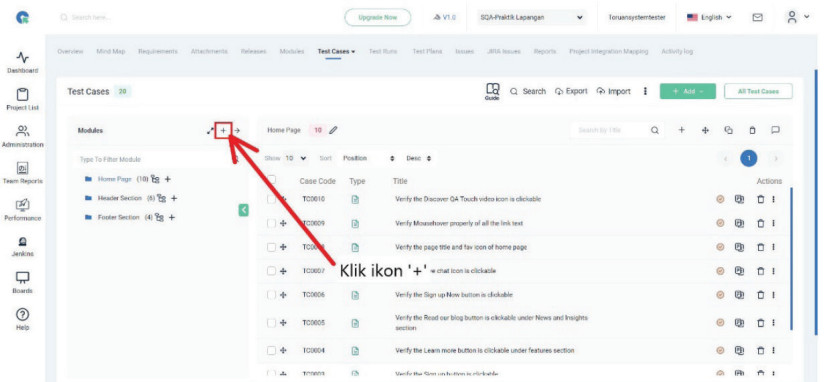


7. Klik ikon ‘+’ pada obyek ‘Test Cases’. Ikon tersebut akan mengarahkan ke tampilan modul-modul dan daftar *test case* modul yang dipilih.

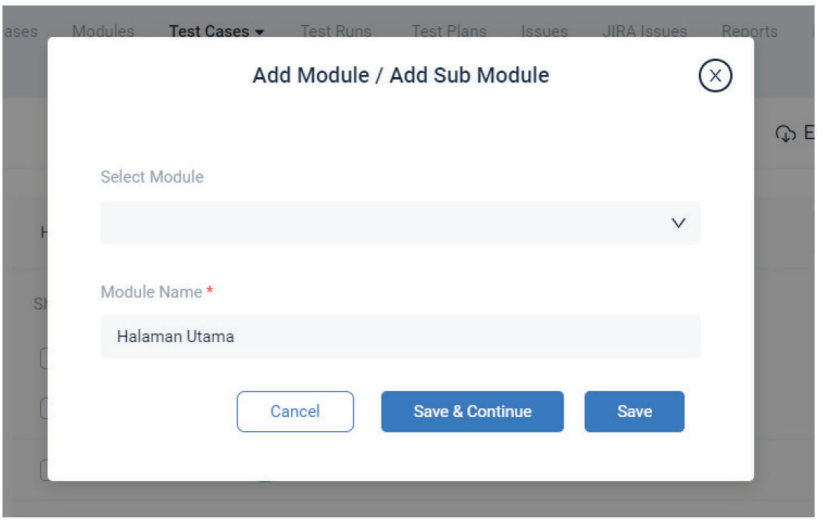




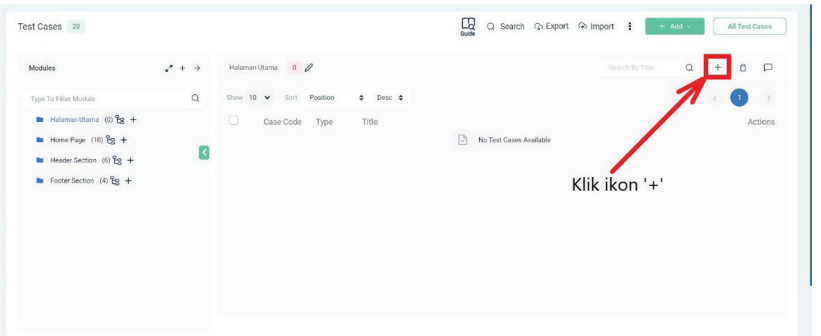
8. Pada tampilan utama pembuatan *test case*, klik ikon '+' pada bagian 'Modules' untuk membuat folder modul.



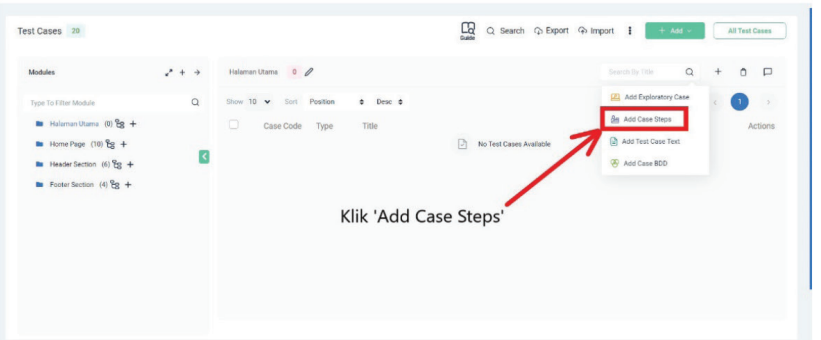
9. Muncul kotak pengisian informasi sebuah modul. Isi nama modul dan klik tombol simpan.



10. Kembali pada tampilan utama 'Test Cases' Klik ikon '+' pada bagian modul disisi kanan (dalam contoh ini nama modul adalah Halaman Utama)

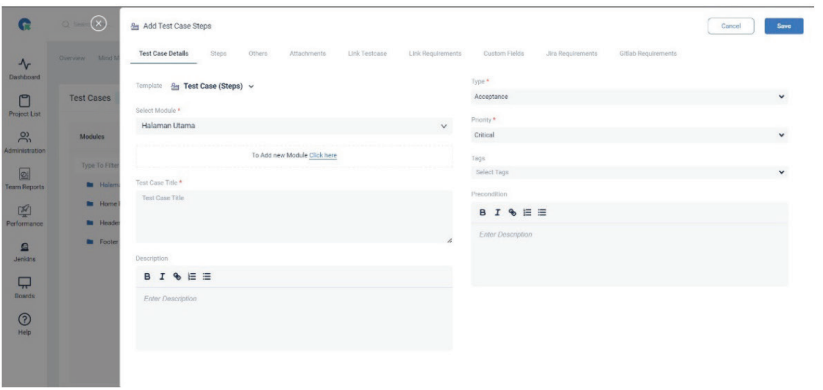


11. Muncul sebuah *dropdown* untuk memilih jenis *test case*. Klik 'Add Test Case (Steps)'

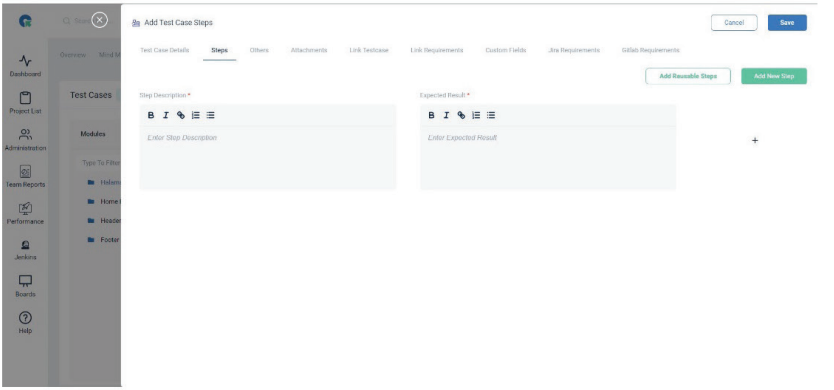


12. Tampilan untuk mengisi seluruh informasi pembuatan *test case* akan muncul.

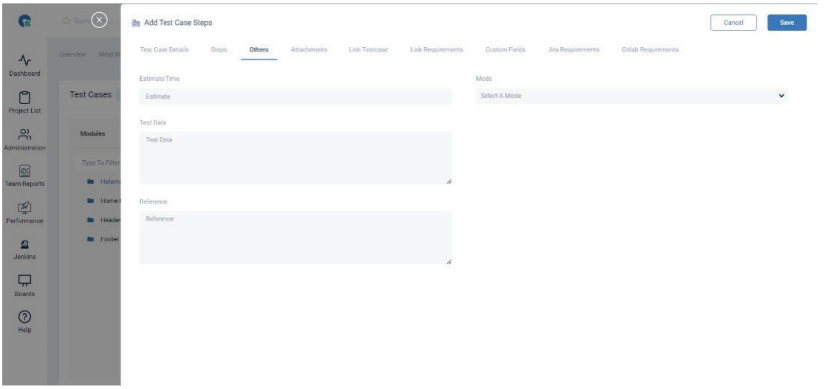
13. Tab 'Test Case Details' berisikotak inputan data direktori modul (subjudul "Select Module"), judul*test case* (subjudul "Test Case Title"), tipe (subjudul "Type"), prioritas (subjudul "Priority"), tag (subjudul "Tag") dan Prekondisi (subjudul "Precondition").



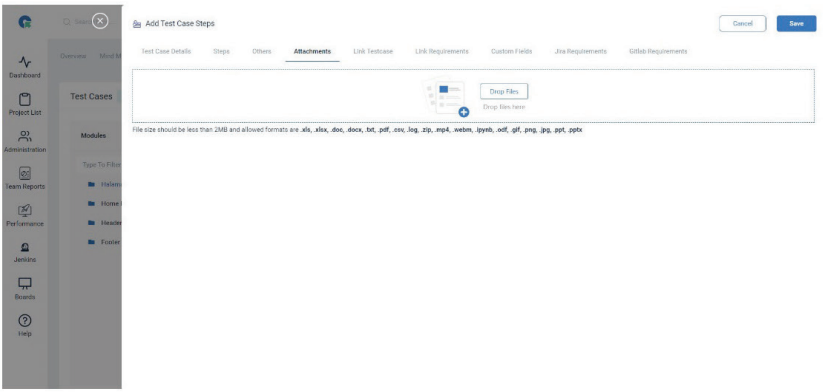
14. Tab 'Steps' berisi kotak inputan deksripsi tahapan (subjudul 'Step Description') dan hasil yang diekspetasikan (subjudul 'Expected Result')



15. Tab 'Others' akan berisi kotak inputan estimasi waktu (subjudul 'Estimate Time'), data uji coba (subjudul 'Test Data'), referensi (subjudul 'Reference') dan mode (subjudul 'Mode').



16. Tab 'Attachment' berisi kotak unggahan file-file.



Pembuatan *test case* secara mendetail pada aplikasi QA Touch dapat melihat pada bagian **3.2 Menulis Test Case**.

3.2. Menulis Test Case

Penulisan *test case* pada umumnya mengikuti format sebagai berikut:

1. Terdapat judul test case
2. Terdapat langkah uji coba
3. Terdapat kesimpulan dari hasil uji coba

Tergantung kebijakan dan peraturan dari perusahaan/organisasi pengembang, format penulisan akan mendapat tambahan lain tetapi tiga poin tersebut merupakan inti sebuah *test case* yang dapat digunakan pada uji coba secara umum. Contoh sebuah *test case* yang memungkinkan untuk ditemukan:

ID - TEST CASE	JUDUL TEST CASE	JENIS TEST CASE	SUB JUDUL TEST CASE	LANGKAH UJI COBA	KESIMPULAN	STATUS	INFORMASI TAMBAHAN
WEB-01	Input Box Username	POSITIF	Uji coba pada input box username. Uji coba untuk memeriksa jika inputan dapat menerima sepanjang 16 karakter	1. Buka halaman web login (www.example.com) 2. Pada input box username, masukkan karakter bertipe data string sepanjang 16 3. Klik, tombol 'Submit'	Input box dapat menerima seluruh karakter string, tetapi masih mengandung whitespace. Input box sebaiknya memotong whitespace yang mengekor untuk menghemat memori	WARNING	- Diperiksa pada tanggal : 18-02-2022. - Diperiksa pada versi 1.0 - Diperiksa pada web browser Edge
WEB-02	Input Password Field	POSITIF	Uji coba pada input password field. Uji coba untuk memeriksa jika password field dapat menutupi inputan	1. Buka halaman web login (www.example.com) 2. Pada input password field, masukkan sembarang karakter	Karakter pada field inputan tertampil secara tertutup.	PASS	- Diperiksa pada tanggal : 18-02-2022. - Diperiksa pada versi 1.0 - Diperiksa pada web browser Edge
WEB-03	Input Box Username	NEGATIF	Uji coba pada input box username. Uji coba untuk memeriksa jika inputan tidak dapat menerima lebih dari 16 karakter	1. Buka halaman web login (www.example.com) 2. Pada input box username, masukkan karakter bertipe data string sepanjang 17 3. Klik, tombol 'Submit'	Input box dapat menerima karakter string dengan panjang lebih dari 16. Input box sebaiknya membatasi penerimaan maksimum 16 karakter	FAIL	- Diperiksa pada tanggal : 18-02-2022. - Diperiksa pada versi 1.0 - Diperiksa pada web browser Edge
WEB-04	Input Password Field	NEGATIF	Uji coba pada input password field. Uji coba untuk memeriksa jika password field tidak dapat menerima karakter khusus (+,-,=,%,$,?)	1. Buka halaman web login (www.example.com) 2. Pada input password field, masukkan karakter dengan beberapa karakter khusus. Contoh : 'password123+' 3. Klik tombol 'Submit'	Input field dapat menerima karakter string dengan karakter khusus. Input field seharusnya tidak dapat menerima karakter khusus yang sudah ditetapkan.	FAIL	- Diperiksa pada tanggal : 18-02-2022. - Diperiksa pada versi 1.0 - Diperiksa pada web browser Edge

Gambar 3.1 Contoh Pembuatan *Test Case* – Praktik Lapangan

Dipecahkan satu persatu maka :

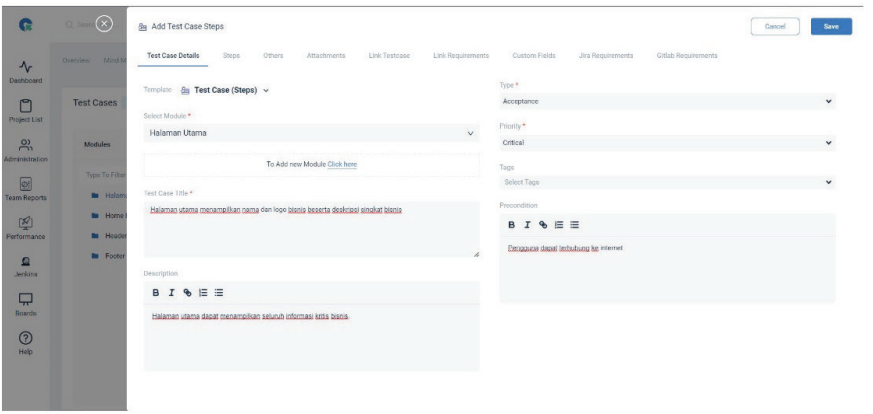
1. ID – *Test Case* : Menunjukkan nomor *test case*. Hal ini penting mengingat judul *test case* bisa diduplikasi.
2. Judul *Test Case* : Menunjukkan judul uji coba komponen.
3. Jenis *Test Case* : Menunjukkan jenis sebuah uji coba. Dapat berupa uji coba secara positif dan negatif.
4. Subjudul *Test Case* : Menunjukkan deskripsi dari judul uji coba.
5. Langkah Uji Coba : Menunjukkan langkah-langkah uji coba.
6. Kesimpulan : Menunjukkan hasil dari uji coba dengan penjelasan singkat.
7. Status : Menunjukkan status uji coba jika komponen lolos uji coba, tidak lolos uji coba atau lolos dengan masalah tambahan yang mengikuti.

8. Informasi Tambahan: Menunjukkan informasi tambahan uji coba seperti pada nomor versi aplikasi yang diuji coba, alat uji coba dsb.

Jika pembuatan *test case* berjenis negatif (hasil uji coba disengaja menghasilkan kegagalan), maka sebaiknya pesan kesalahan diberikan sebagai bukti uji coba tersebut berhasil. Alasannya adalah jika terdapat alur yang tidak diinginkan dan tidak sesuai dengan jalan yang sudah didesain, maka aplikasi masih dapat mengendalikan jalannya program [6].

Selain pembuatan *test case* melalui cara manual seperti diatas, pembuatan *test case* dapat dilakukan dengan lebih teratur melalui aplikasi manajemen *test case*. Aplikasi QA Touch akan menjadi alat yang dapat digunakan sebagai contoh pembuatan test case (Lihat bagian **3.1. Aplikasi Manajemen Test Case** untuk melihat pengenalan QA Touch). Berikut pembuatan *test case* dengan aplikasi QA Touch:

- **Detail *Test Case***



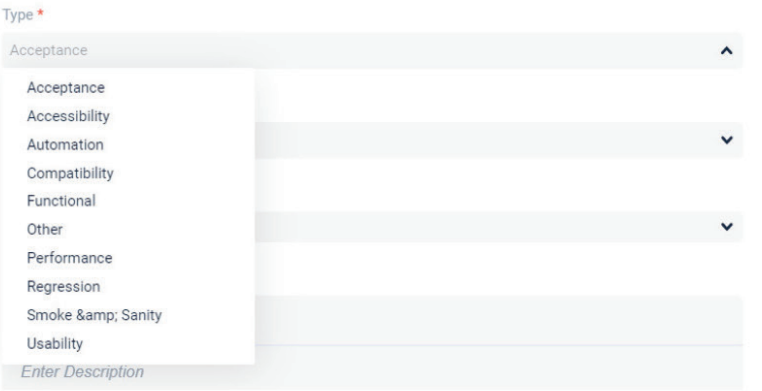
Gambar 3.2. Detail Test Case Pada QA Touch

Kotak-kotak inputan dan *dropdown* pada 'Detail *Test Case*' dapat dijelaskan sebagai berikut:

1. ***Select Module***. Berisi informasi modul direktori yang akan menjadi tempat kumpulan *test case*yang sedang dibuat. Jika menginginkan direktori baru, dapat langsung membuatnya

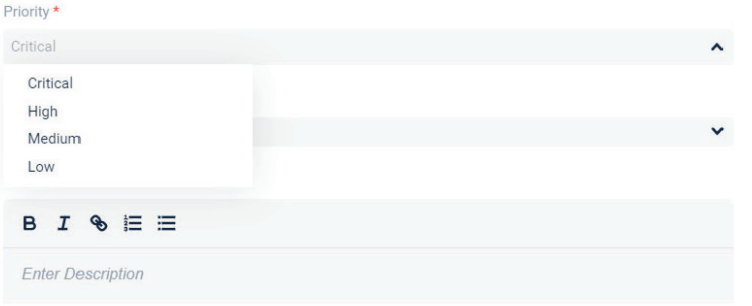
pada saat pembuatan test case tersebut dengan mengklik tautan 'Click Here' pada bagian *To Add new Module*.

2. ***Test Case Title***. Berisi judul *test case*.
3. **Description**. Berisi deskripsi *test case* yang akan dibuat. Deksripsi *test case* dapat berupa informasi apa saja yang dibutuhkan dan bersifat opsional.
4. ***Type***. Berisi tipe-tipe *test case* yang akan dibuat. Tipe *Acceptance* berarti menguji-cobakan penerimaan persyaratan. Tipe *Accessibility* berarti menguji-cobakan kemampuan untuk mendapatkan akses. Tipe *Automation* berarti uji coba secara automasi. Tipe *Compatibility* berarti tahap uji coba akan melihat unit atau fitur dapat bekerja di versi atau perangkat lainnya. Tipe *Functional* berarti tahap uji coba akan melihat kemampuan fungsional unit atau fitur. Tipe *Other* berarti uji coba akan melihat hal-hal lain diluar dari tipe-tipe umum uji coba. Tipe *Performance* berarti tahap uji coba akan melihat performa aplikasi atau perangkat dalam menjalankan permintaan. Tipe *Regression* berarti uji coba pada tahap sebelumnya akan dilakukan kembali untuk memastikan tidak terdapat kesalahan yang muncul. Tipe ini biasa disebut uji coba ulang [13]. Tipe *Smoke* berarti uji coba akan melihat jika fitur atau fungsi penting dapat bekerja sehingga dapat dilakukan uji coba lainnya [14]. Tipe *Sanity* berarti uji coba hanya akan melihat memeriksa fitur atau fungsi baru [15]. Tipe *Usability* berarti uji coba akan dilakukan dengan tahapan-tahapan yang disesuaikan untuk para pengguna. Uji coba ini akan didampingi oleh pihak pengembang dengan beberapa pengguna sebagai perwakilan [16].



Gambar 3.3. Tipe-Tipe Test Case

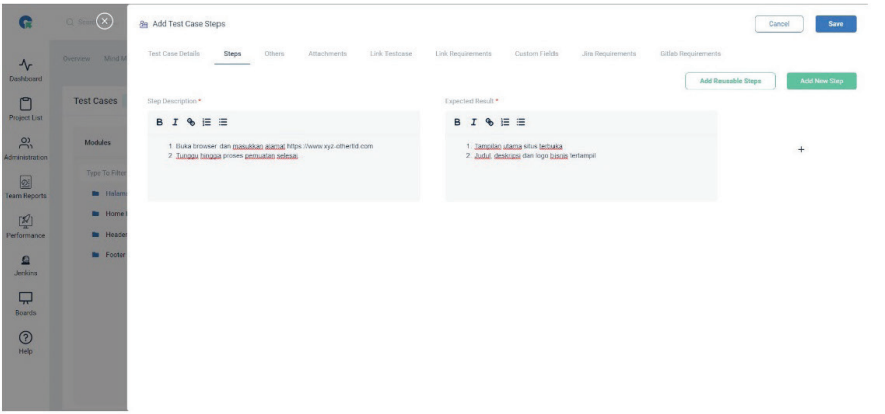
5. ***Priority***. Berisi tingkat prioritas-prioritas uji coba. Prioritas *critical* merupakan prioritas tertinggi dan *low* merupakan prioritas terendah.



Gambar 3.4. Tingkat Prioritas Test Case

6. ***Tags***. Berisi tag-tag yang sudah dibuat untuk disematkan ke sebuah *test case*.
7. ***Precondition***. Berisi deskripsi *test case* yang mengharuskan persyaratan tersebut terpenuhi terlebih dahulu untuk dapat melakukan uji coba.

- **Tahapan-Tahapan**

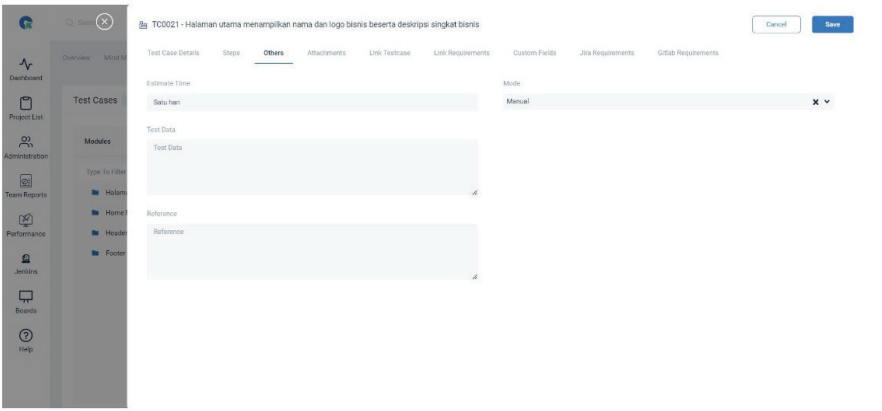


Gambar 3.5. Tahapan-Tahapan Test Case

Kotak-kotak inputan pada 'Tahapan-Tahapan' dapat dijelaskan sebagai berikut:

1. ***Step Description***. Berisi deskripsi tahapan-tahapan uji coba. Tahapan-tahapan ini dijelaskan secara rinci untuk mendapatkan hasil yang diinginkan.
2. ***Expected Result***. Berisi hasil yang diinginkan dari tahapan uji coba. Hasil yang diinginkan dapat berupa fitur atau fungsi bekerja dengan semestinya atau menghasilkan pesan kesalahan jika diinginkan tahap uji coba mengalami kegagalan.

- **Lainnya**

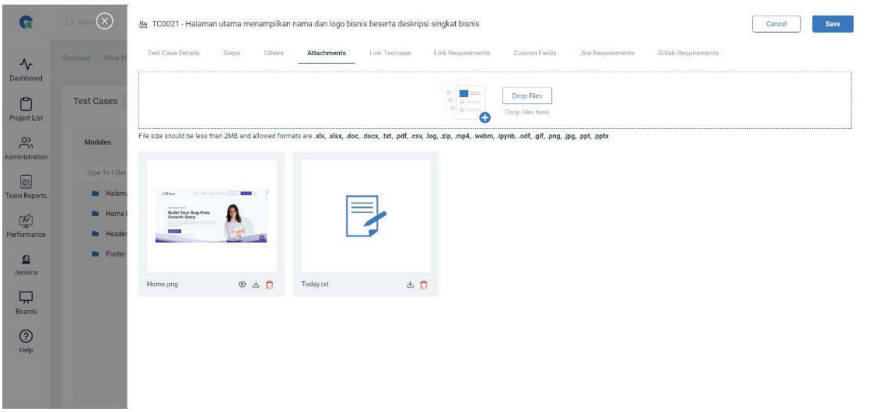


Gambar 3.6. Informasi Lain Test Case

Kotak-kotak inputan pada 'Lainnya' dapat dijelaskan sebagai berikut:

1. ***Estimate Time***. Berisi estimasi waktu / lama dilakukannya uji coba.
2. ***Test Data***. Berisi kumpulan-kumpulan data yang diperlukan untuk melakukan uji coba. Contoh kumpulan nama pengguna beserta kata sandinya, kumpulan nomor-nomor pemesanan barang.
3. ***Reference***. Berisi informasi berupa referensi-referensi lanjutan yang dibutuhkan saat uji coba
4. ***Mode***. Berisi informasi berupa mode uji coba. Mode dapat berupa manual dan automasi.

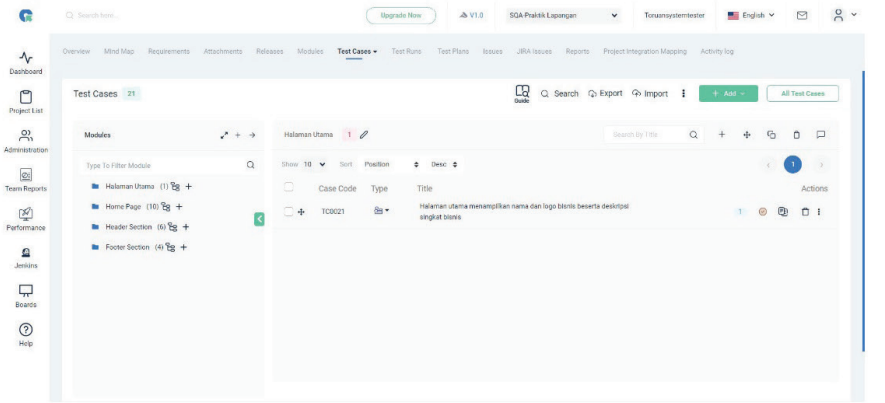
- **Attachments**



Gambar 3.6. Sematan Test Case

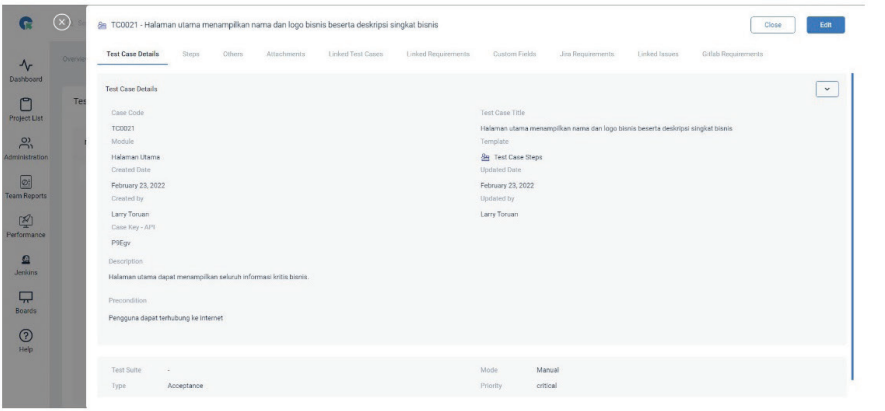
Tab 'Lampiran' berisi kotak unggahan file-file yang dapat ditambahkan. File-file yang diunggah dapat berupa sebelum atau sesudah uji coba atau file-file penting lain untuk mendukung uji coba.

Jika pembuatan *test case* sudah selesai maka akan terlihat pada daftar *test case* terdapat *test case* yang sudah dibentuk beserta nomor kodenya.



Gambar 3.7. Test Case Terbuat

Mengklik obyek test case tersebut akan menampilkan informasi mendetail obyek *test case*. Mengklik tab-tab lainnya akan menampilkan semua informasi yang sudah ditambahkan.



Gambar 3.8. Detail Test Case Terbuat

3.3. Latihan Pertama, Uji Coba Antarmuka

Bagian-bagian latihan ini disarankan untuk membuatnya menggunakan aplikasi QA Touch. Meskipun penulisan manual dapat dilakukan, dengan menggunakan aplikasi manajemen *test case* penulisan uji coba dapat dilakukan lebih mudah dan terstrukur. Terdapat tiga bagian latihan untuk uji coba antarmuka dengan dua latihan pertama memiliki solusi diakhir bagian. Akan tetapi pembaca diharapkan untuk

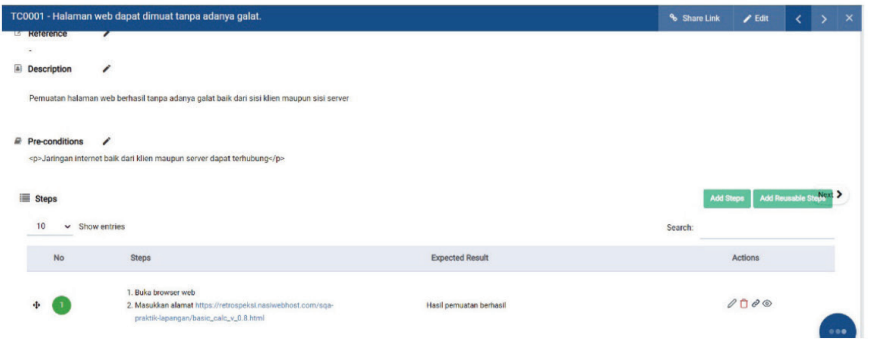
menyelesaikan latihan tersebut dengan mandiri kemudian membaca solusi diakhir dengan alasan pembaca dapat semakin mahir dalam penulisan uji coba.

Latihan pertama dari uji coba antarmuka ini akan memeriksa sebuah halaman web sederhana yang memiliki fungsi layaknya sebuah kalkulator sederhana. Pembaca dapat mengunjungi situs https://testsheepnz.github.io/BasicCalculator.html untuk melihat dan menguji aplikasi web tersebut.

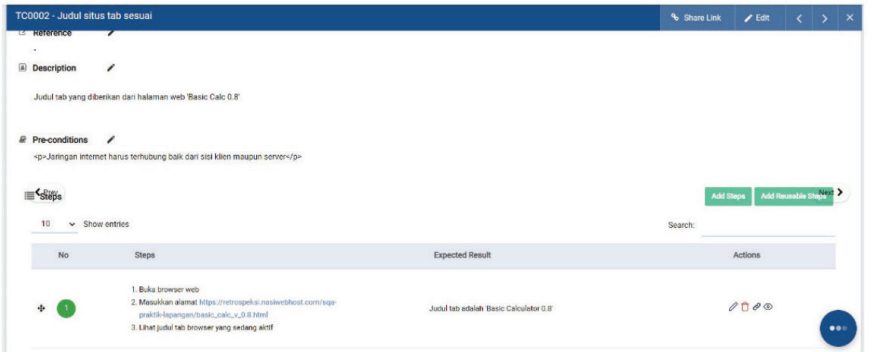
Berikut kriteria penerimaan yang akan diperiksa:

1. Halaman web dapat dimuat tanpa adanya galat
2. Judul situs tab adalah "Basic Calculator"
3. Judul halaman web adalah "Basic Calculator"
4. Terdapat deskripsi singkat dibawah judul
5. Aritmatika dari kedua operasi seharusnya sesuai dengan subjudul
6. Pilihan dropdown adalah "Add", "Substract", "Multiply", "Divide"dan ,"Concatenate"
7. Tulisan hasil selalu dalam bentuk *bold*
8. Hanya dapat memasukkan angka
9. Tidak dapat memasukkan nilai kosong (null)
10. Desain sesuai dengan *mockup*

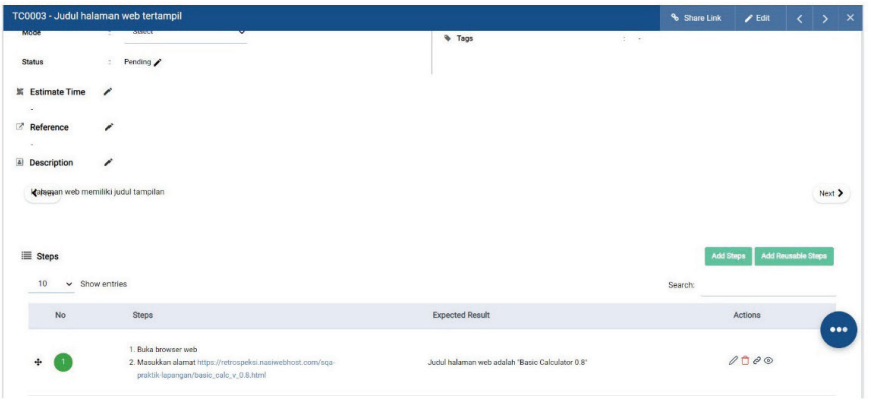
Solusi dari latihan pertama dapat melihat gambar pembuatan *test case* berikut:



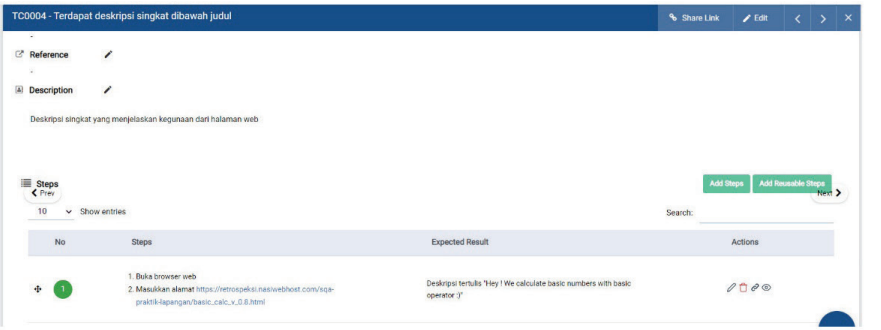
Gambar 3.9. Solusi Ke-1



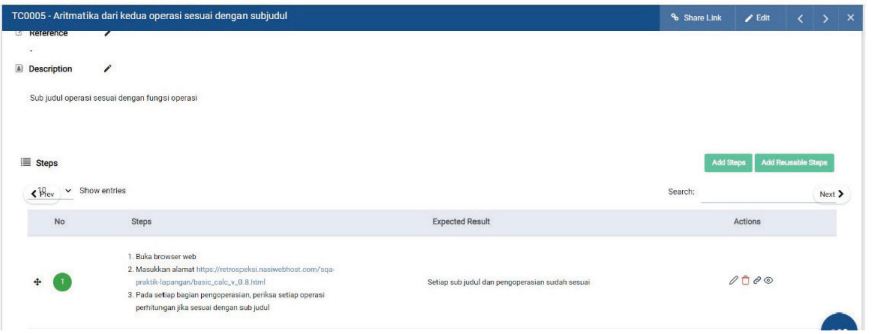
Gambar 3.10. Solusi Ke-2



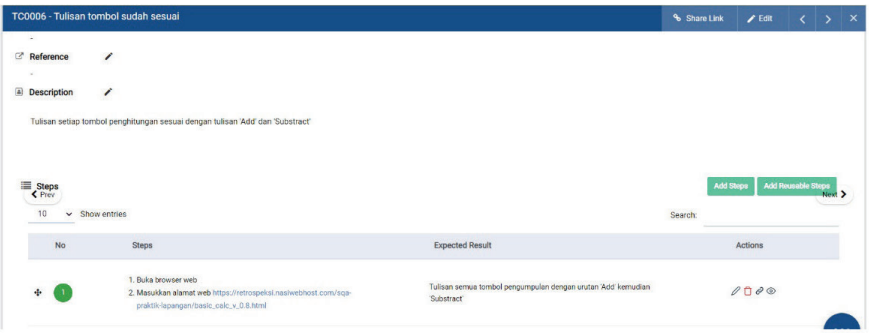
Gambar 3.11. Solusi Ke-3



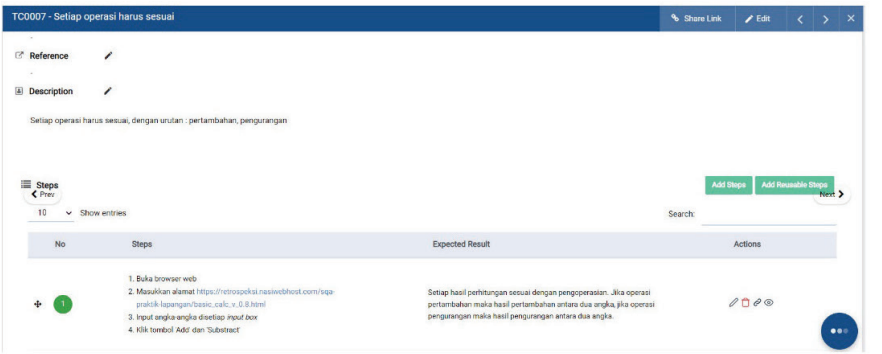
Gambar 3.12. Solusi Ke-4



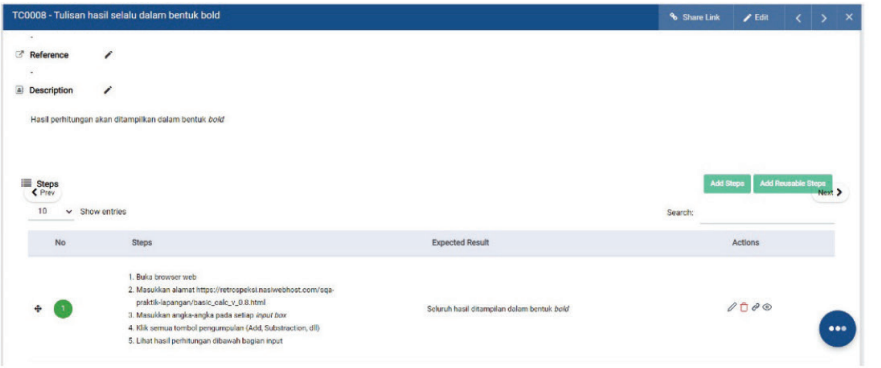
Gambar 3.13, Solusi Ke-5



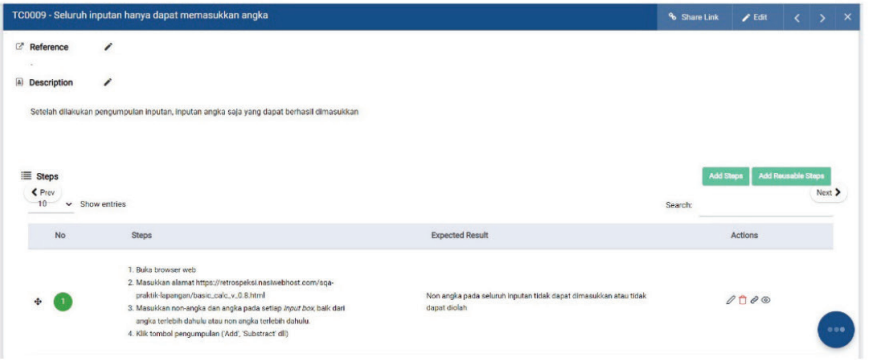
Gambar 3.14. Solusi Ke-6



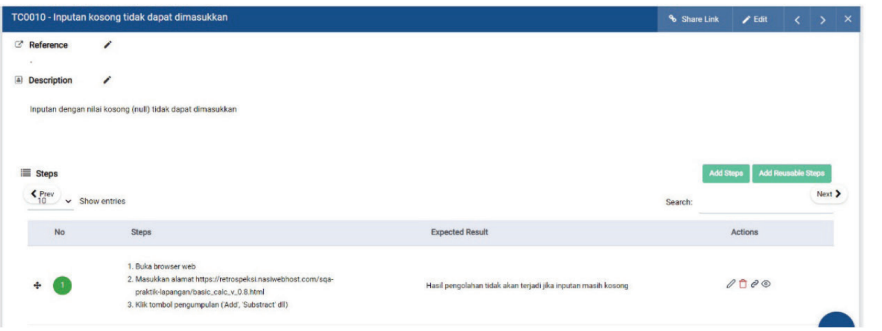
Gambar 3.15. Solusi Ke-7



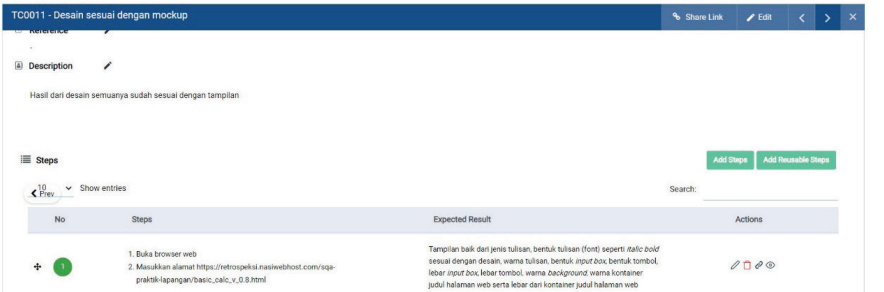
Gambar 3.16. Solusi Ke-8



Gambar 3.17. Solusi Ke-9



Gambar 3.18. Solusi Ke-10



Gambar 3.19. Solusi Ke-11

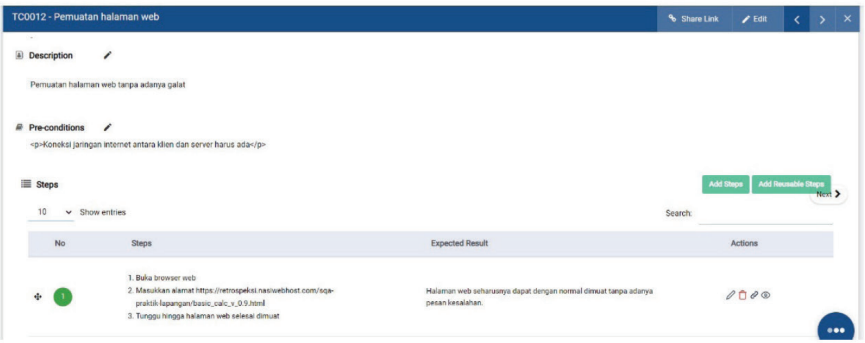
3.4. Latihan Kedua, Uji Coba Antarmuka

Latihan kedua dari uji coba antarmuka ini akan memeriksa kembali halaman web sebelumnya dengan beberapa perbaikan yang sudah diberikan serta terdapat fungsi baru. Pembaca dapat mengunjungi situs https://retrospeksi.nasiwebhost.com/sqa-praktik-lapangan/basic_calc_v_0.9.html untuk melihat dan menguji aplikasi web tersebut. Pembaca juga dapat melihat gambar desain melalui tautan https://retrospeksi.nasiwebhost.com/sqa-praktik-lapangan/mockups/basic_calc_v0.9.png.

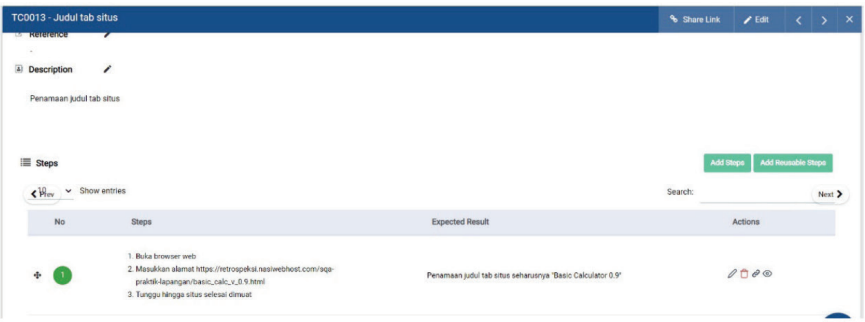
Kriteria penerimaan dari latihan kedua adalah sebagai berikut:

1. Halaman web dapat dimuat tanpa adanya galat
2. Judul situs tab adalah "Basic Calculator 0.9"
3. Judul halaman web adalah "Basic Calculator"
4. Terdapat deskripsi singkat dibawah judul "Hey! We calculate basic numbers with basic operator :)"
5. Aritmatika dari ketiga operasi seharusnya sesuai dengan subjudul
6. Tulisan tombol adalah "Add", "Substract" & "Multiplication"
7. Setiap operasi harus sesuai, dengan urutan: pertambahan, pengurangan, perkalian
8. Tulisan hasil selalu dalam bentuk *bold*
9. Hanya dapat memasukkan angka
10. Tidak dapat memasukkan nilai kosong (null)
11. Desain sesuai dengan *mockup*

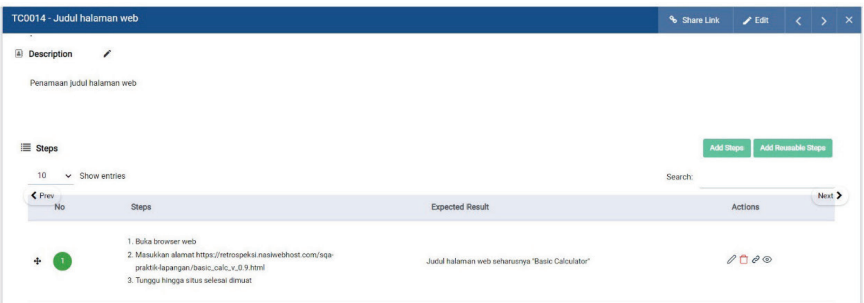
Jika pembaca sudah menyelesaikan latihan, solusi pembuatan *test case* dapat melihat melalui gambar berikut:



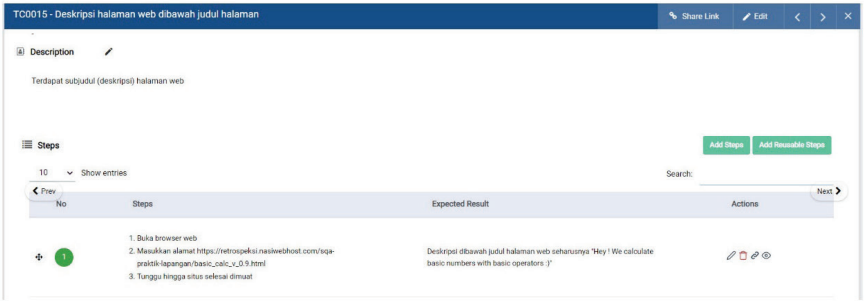
Gambar 3.20. Solusi Ke-1



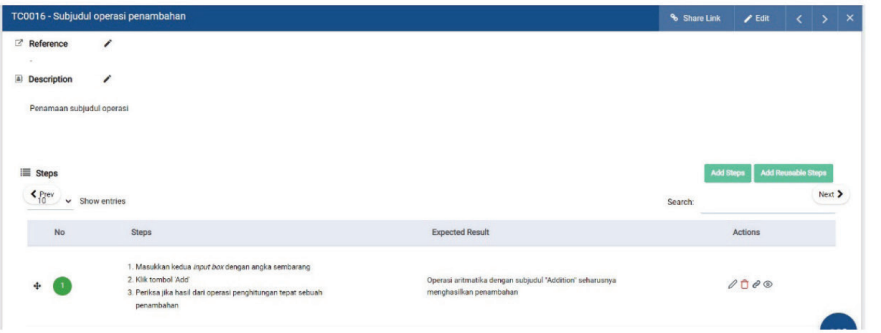
Gambar 3.21. Solusi Ke-2



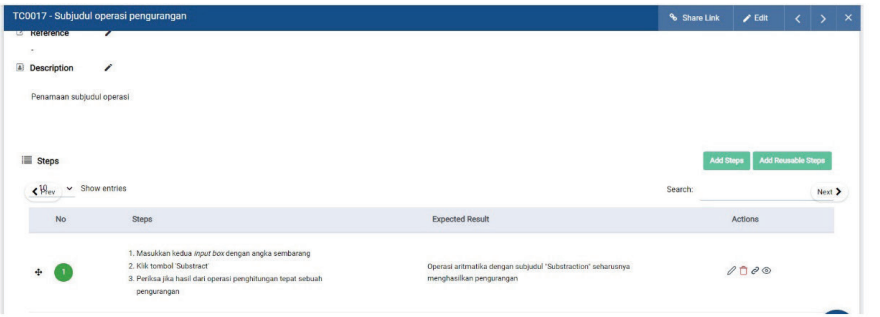
Gambar 3.22. Solusi Ke-3



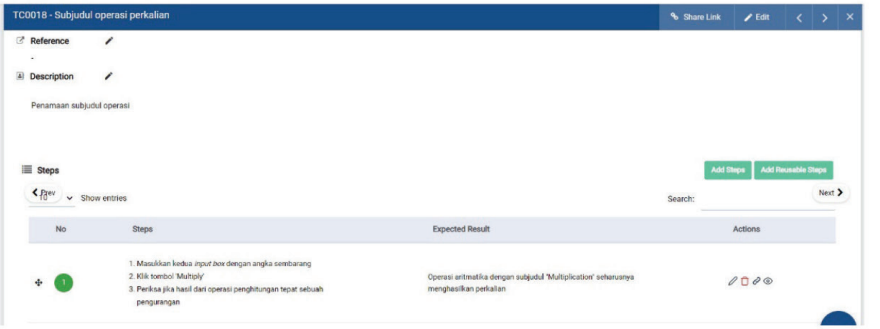
Gambar 3.23. Solusi Ke-4



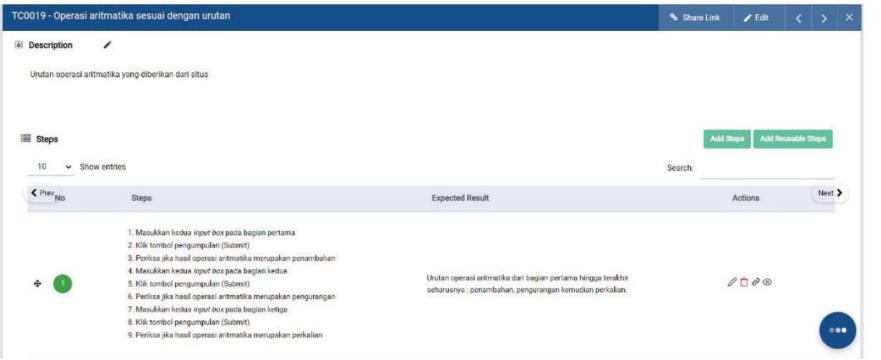
Gambar 3.24. Solusi Ke-5



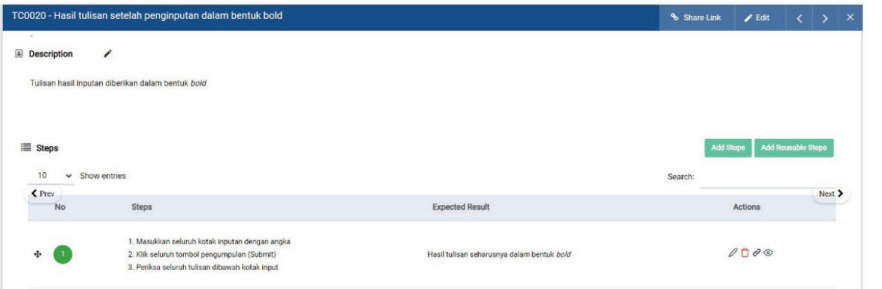
Gambar 3.25. Solusi Ke-6



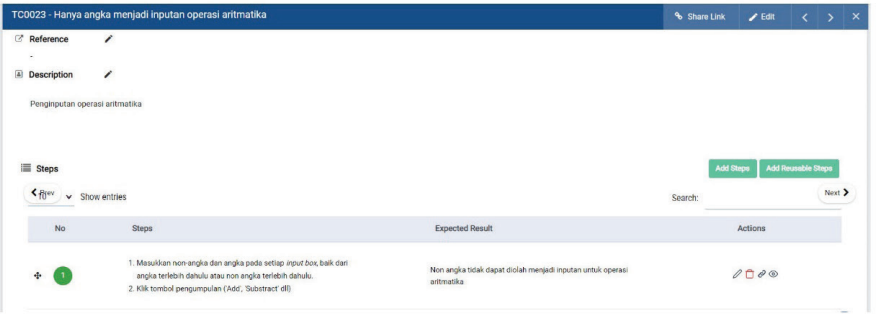
Gambar 3.26. Solusi Ke-7



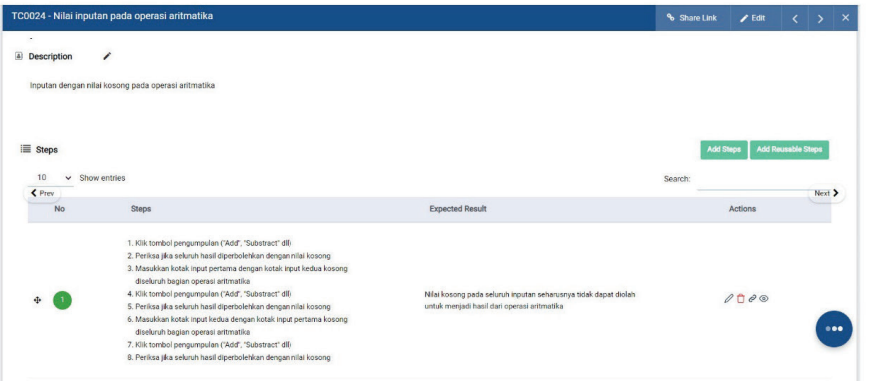
Gambar 3.27. Solusi Ke-8



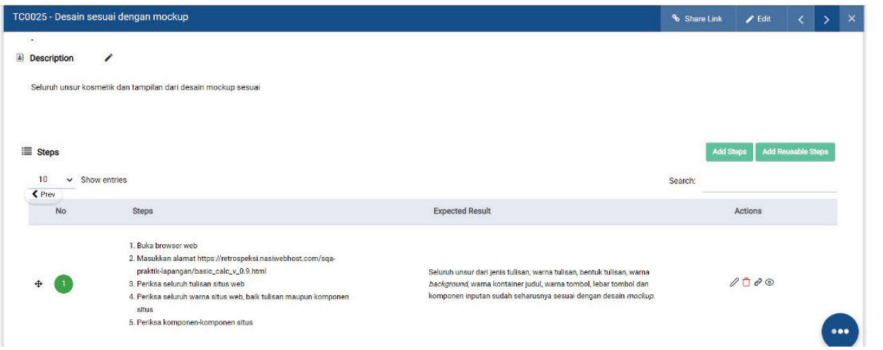
Gambar 3.28. Solusi Ke-9



Gambar 3.29. Solusi Ke-10



Gambar 3.30. Solusi Ke-11



Gambar 3.31. Solusi Ke-12

3.5. Latihan Ketiga, Uji Coba Antarmuka

Latihan ketiga dari uji coba antarmuka ini akan memeriksa kembali halaman web sebelumnya dengan juga menggunakan kriteria penerimaan sebelumnya. Pembaca dapat mengunjungi situs https://retrospeksi.nasiwebhost.com/sqa-praktik-lapangan/basic_calc_v_1.0.html untuk melihat dan menguji aplikasi web tersebut. Pembaca juga dapat melihat gambar desain melalui tautan https://retrospeksi.nasiwebhost.com/sqa-praktik-lapangan/mockups/basic_calc_v1.0.png. Pembaca dapat menambahkan *test case* lainnya selain dari kriteria penerimaan sebagai latihan pemeriksaan penjelajahan (*exploratory test)*. Situs web merupakan aplikasi final sehingga tidak ada tambahan perbaikan ataupun perubahan tampilan. Solusi pembuatan test case pada latihan ketiga tidak disediakan dikarenakan terdapat pemeriksaan penjelajahan. Pembaca

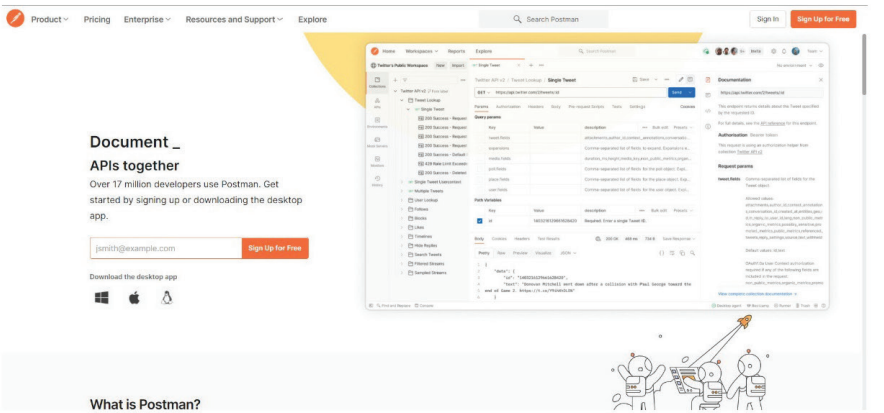
hanya dapat melihat solusi pembuatan *test case* sebelumnya sebagai referensi pembuatan *test case*.

Bab 4.

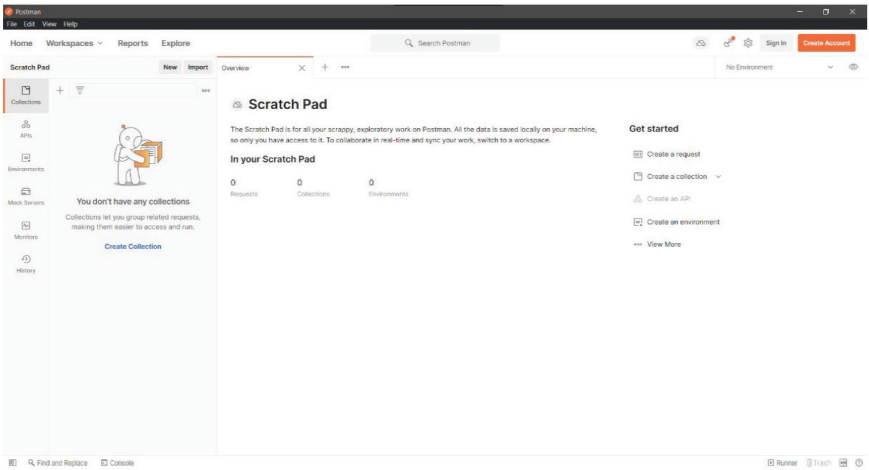
Uji Coba, Kemampuan Lanjutan

4.1. Pengenalan Aplikasi Postman

Postman merupakan platform aplikasi untuk pengembangan API (Application Programming Interface) [17]. Aplikasi ini dapat menyimpan koleksi-koleksi permintaan API dan dapat melakukan uji coba kepada koleksi permintaan API tersebut. API sendiri merupakan sebuah daftar permintaan yang disediakan oleh sebuah program dan permintaan tersebut dapat berupa informasi yang akan ditampilkan hingga permintaan untuk pengolahan data [18]. Permintaan API dapat diberikan dalam bentuk yang disesuaikan untuk kebutuhan penggunaan sehingga pekerjaan menjadi lebih efisien dan efektif.

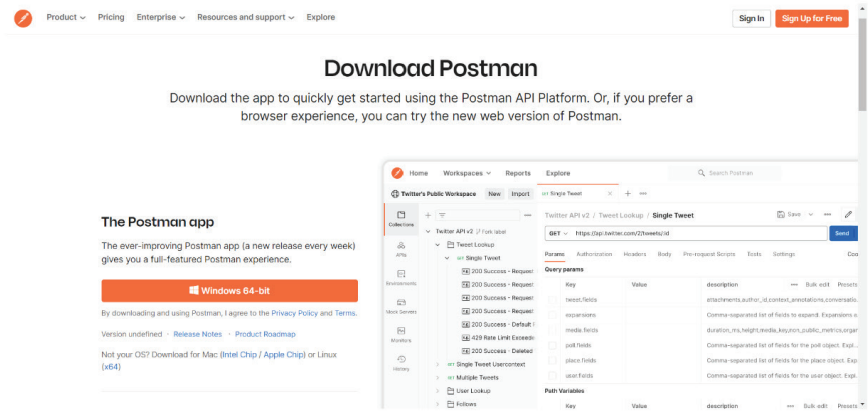


Gambar 4.1. Tampilan Web Aplikasi Postman



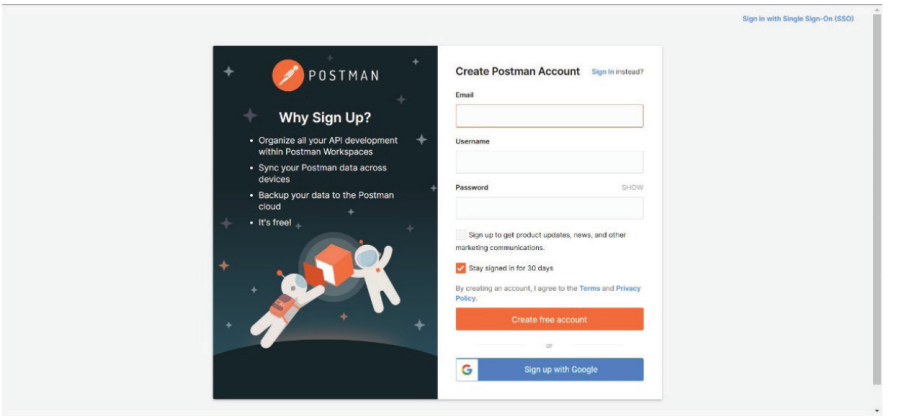
Gambar 4.2. Tampilan Aplikasi Postman Pada Dekstop

Aplikasi Postman dapat diunduh untuk berbagai macam sistem operasi. Silahkan mengunduh aplikasinya melalui tautan berikut:https://www.postman.com/downloads



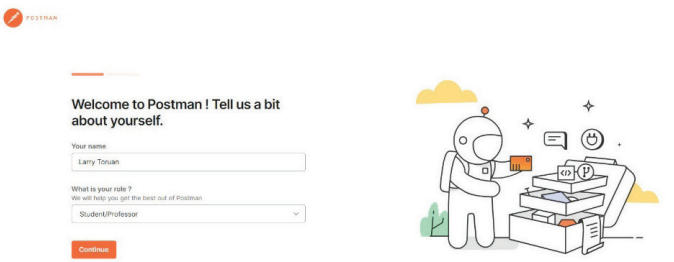
Gambar 4.3. Tampilan Halaman Unduh Aplikasi Postman

Supaya dapat menggunakan aplikasi Postman, lakukan registrasi dengan cara membuat akun terlebih dahulu. Registrasi dapat dilakukan melalui tautan berikut:https://identity.getpostman.com/signup



Gambar 4.4. Tampilan Registrasi Aplikasi Postman

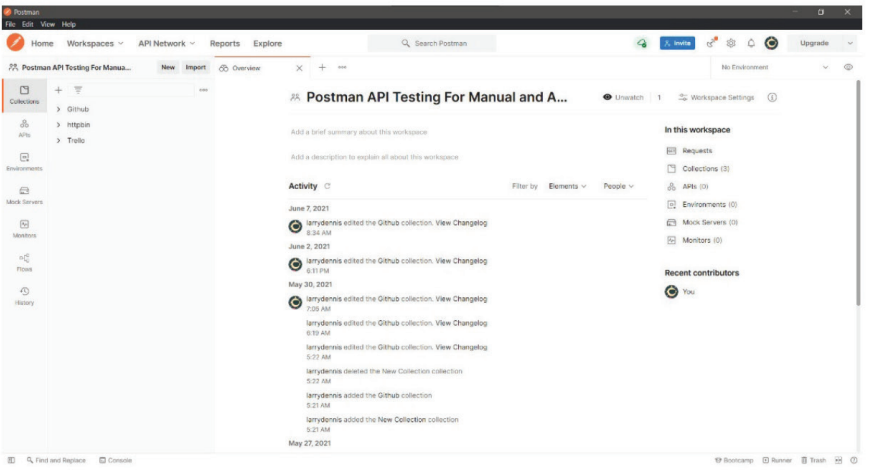
Langkah-langkah registrasi Postman mirip seperti pembuatan akun untuk aplikasi QA Touch hanya saja tidak ada pembuatan nama domain. Informasi seperti alamat email, nama pengguna, kata sandi dan tipe peranan. Setelah memasukkan informasi tersebut, pilih untuk menjalankan tanpa tim.



Gambar 4.5. Pengisian Informasi Nama Dan Role

Gambar 4.6. Lanjut Tanpa Tim

Setelah melakukan registrasi dengan membuat akun Postman dan mengkonfirmasi alamat email yang digunakan, selanjutnya dari tampilan aplikasi Postman otomatis akan mendaftarkan akun yang sudah dibuat. Tampilan utama terlihat ruang kerja sebagai berikut. Jika belum masuk ketampilan ruang kerja, klik tombol *dropdown* 'Workspaces' dan kemudian klik 'My Workspace'.



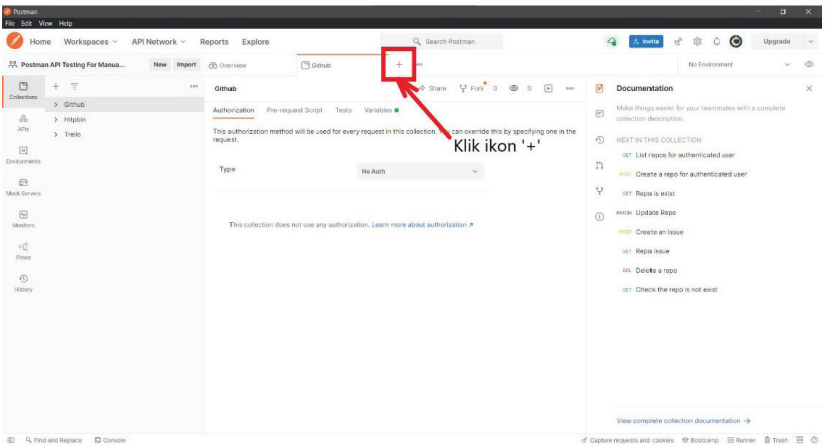
Gambar 4.7. Ruang Kerja

4.2. Membuat Permintaan API

Permintaan API dalam skenario dunia nyata dibuat oleh para pengembang aplikasi sehingga seorang SQA yang melakukan uji coba tidak perlu membuat tautan-tautan yang menyediakan permintaan API. Alasannya, pembuatan permintaan API sering memiliki kompleksitas sehingga pengembangan dan uji coba menjadi tugas terpisah. Adapun jika ingin melakukan uji coba permintaan API harus dipastikan aplikasi yang dikembangan menyediakan permintaan API, jika tidak tersedia maka tidak dapat diuji coba.

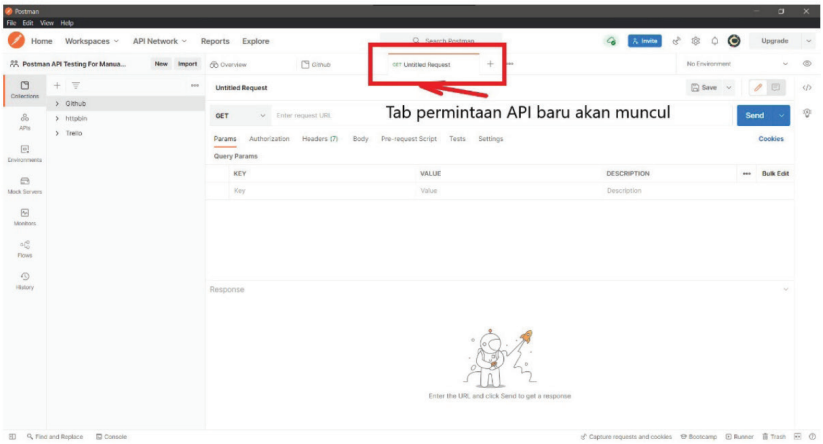
Pembuatan permintaan API dalam aplikasi Postman sebagai berikut :

1. Pada tempat ruang kerja yang sedang aktif, klik ikon '+' untuk menambahkan sebuah permintaan API.



Gambar 4.8. Ikon '+' Pada Permintaan API

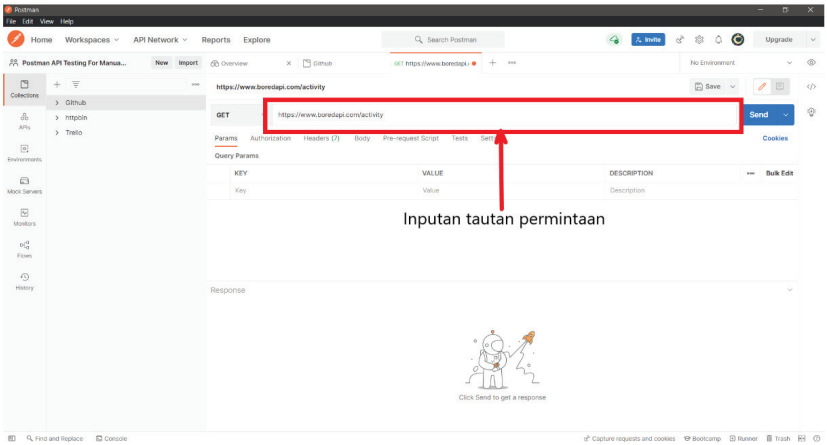
2. Tab permintaan API baru akan muncul dan memiliki judul 'GET Untitled Request'.



Gambar 4.9. Tab Permintaan API

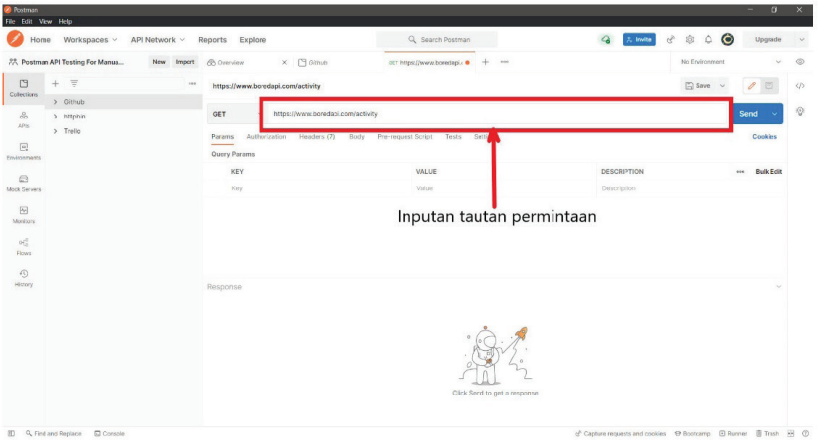
3. Pada tab permintaan API baru yang sedang dipilih / aktif, terdapat bagian-bagian fungsional permintaan yang mirip layaknya ketika membuat permintaan halaman web dalam sebuah browser web. Pada bagian 'Params' terdapat tabel kolom 'Key', 'Value', dan 'Description'

4. Permintaan API kali ini hanya akan menggunakan sebuah tautan permintaan.Situs https://www.boredapi.com/ dapat digunakan untuk uji coba API. Dokumentasi API situs juga dapat dilihat melalui https://www.boredapi.com/documentation. Pada aplikasi Postman, salin dan tempel atau ketik tautan http://www.boredapi.com/api/activity/ pada kotak inputan tautan permintaan.



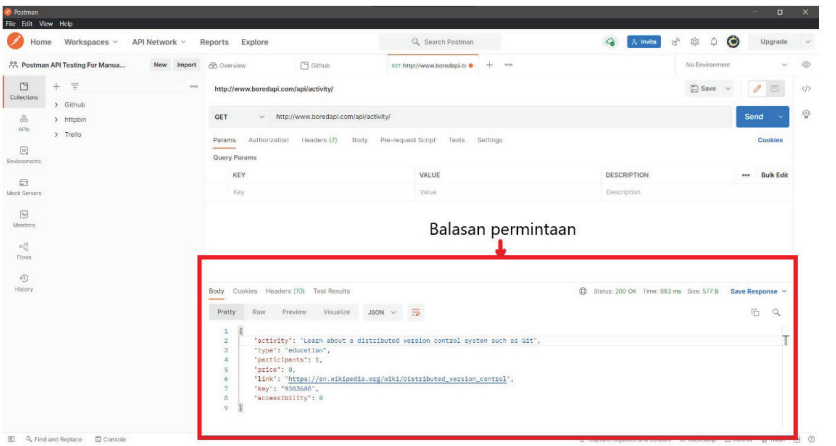
Gambar 4.10. Inputan Permintaan API

5. Setelah kotak inputan tautan permintaan terisi, selanjutnya pastikan tombol *dropdown* jenis permintaan (terletak disebelah kiri kotak inputan permintaan) berjenis 'GET'. Selanjutnya, klik tombol berwarna biru dengan tulisan 'Send' untuk mengirim permintaan.



Gambar 4.11. Jenis Permintaan Dan Tombol Kirim

6. Jika permintaan sukses maka balasan permintaan akan tertampil dibawah ruang fokus permintaan dengan kode status berupa 200 berwarna hijau. Balasan ini merupakan *Bodyrequest* (isi balasan)

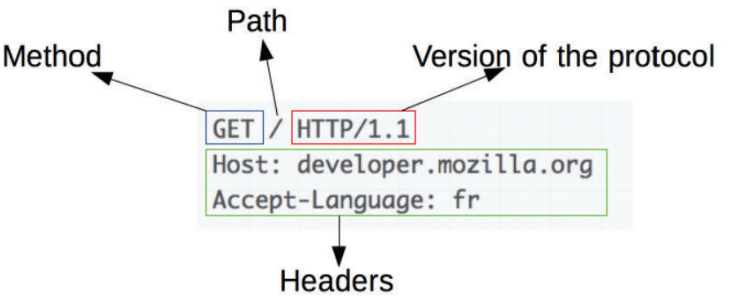


Gambar 4.12. Balasan Permintaan API

Layaknya seperti sebuah browser web, aplikasi Postman mengirim permintaan dan menerima balasan dengan menggunakan protokol HTTP. Protokol HTTP memiliki bagan pesan permintaan sebagai berikut [19]:

Requests

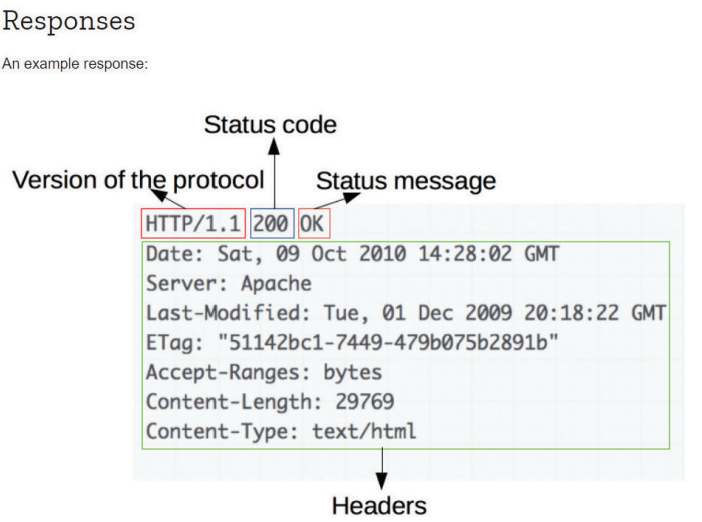
An example HTTP request:



Gambar 4.13. Bagan Pesan Permintaan

- Address (Alamat situs)
- Request Method (Metode permintaan, umumnya GET, POST dan HEAD). Metode permintaan menyatakan jenis metode yang dipakai untuk mengirimkan permintaan API. Metode GET akan meminta informasi baru, metode POST akan memberikan informasi baru dengan menyertakan data yang diberikan dan metode HEADmrip seperti metode GET, metode akan meminta informasi baru pada bagian *header*.
- Headers (Informasi tambahan, opsional)
- Body (Isi permintaan, opsional. Jika menggunakan metode POST maka diperlukan)

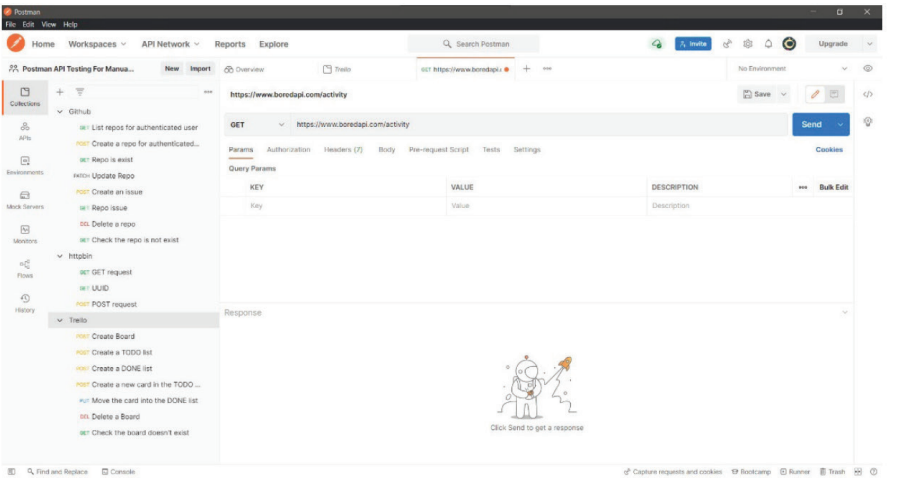
Kemudian bagan pesan balasan sebagai berikut [19]:



Gambar 4.14. Bagan Pesan Balasan

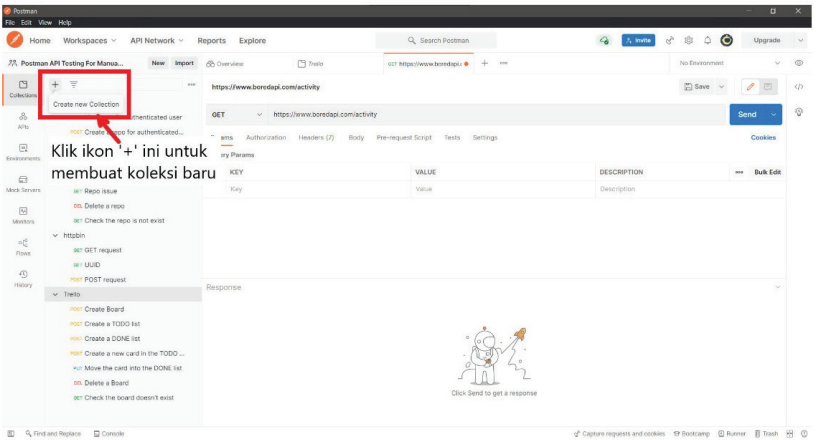
- Status code (Kode status. Kode dapat berupa 2nn, 4nn, 3nn dan 5nn). Kode status menunjukkan jika permintaan berhasil atau tidak serta alasannya.
- Status message (Pesan status). Pesan status mendeskripsikan maksud dari kode status secara pendek.
- Headers (Informasi tambahan balasan)
- Body (Isi balasan)

Aplikasi Postman sendiri memiliki fitur koleksi sebagai tempat untuk mengatur kumpulan-kumpulan permintaan API. Koleksi pada sebuah ruang kerja dapat memiliki beberapa direktori didalamnya. Koleksi dapat kita buat dengan langkah sebagai berikut :



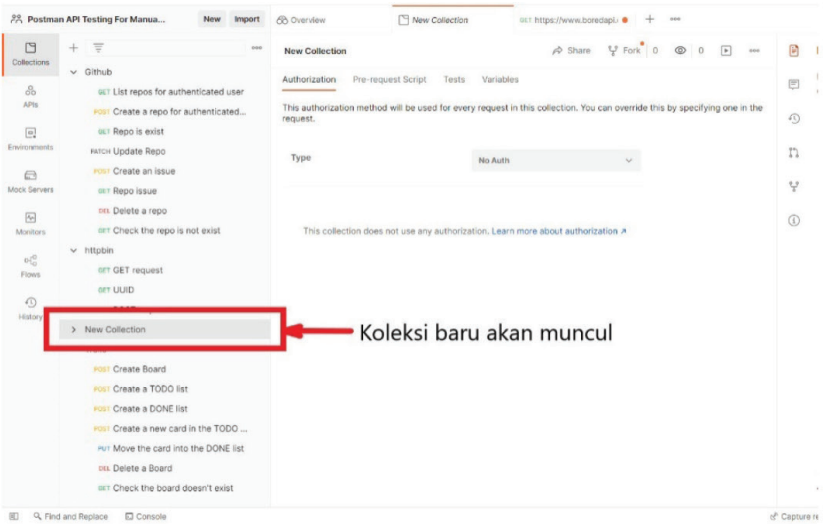
Gambar 4.15. Kumpulan Koleksi

1. Klik ikon '+'pada bagian kumpulan koleksi disamping ruang fokus permintaan API.



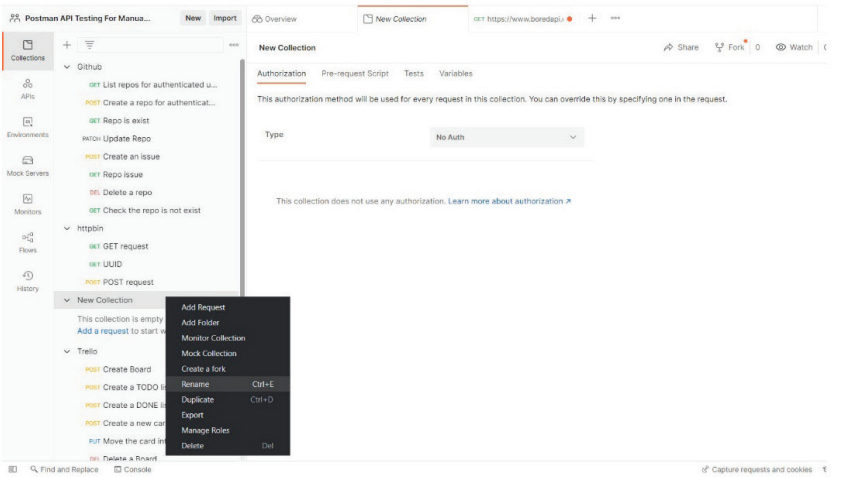
Gambar 4.16. Ikon '+' Pada Kumpulan Koleksi

2. Terdapat koleksi baru dengan tulisan 'New Collection'



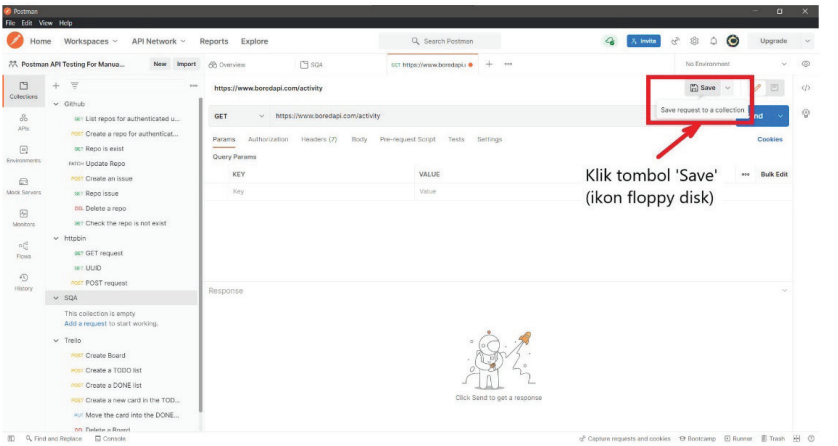
Gambar 4.17. Koleksi Baru

3. Jika ingin mengganti nama koleksi, dapat mengklik kanan koleksi tersebut, kemudian pilih 'Rename'. Nama koleksi lama akan disorot, menandakan siap untuk ditulis ulang.



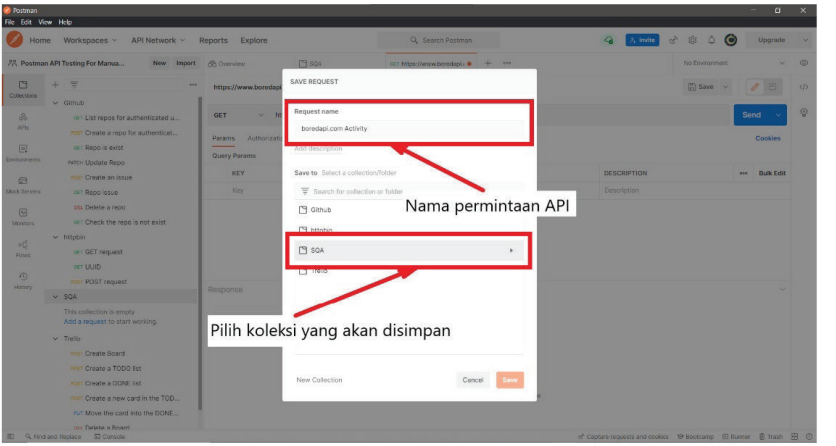
Gambar 4.18. Penamaan Ulang Koleksi

4. Karena isi koleksi masih kosong, maka koleksi dapat disi dengan menyimpan permintaan API yang sudah dibuat sebelumnya. Untuk menyimpan permintaan API klik tombol 'Save'.



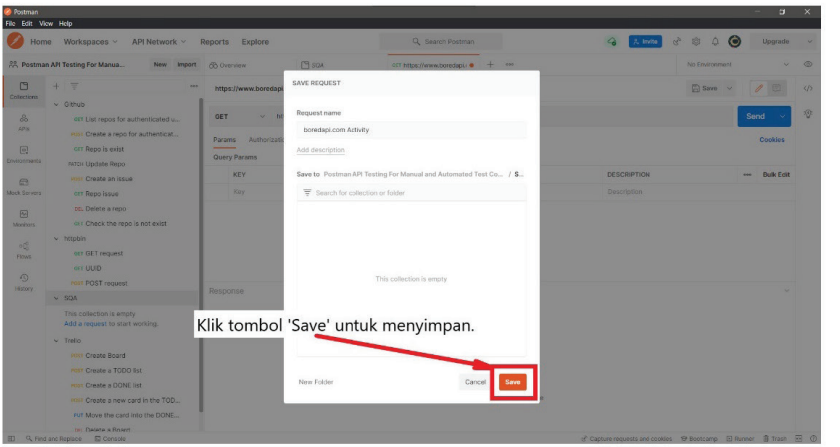
Gambar 4.19. Ikon '+' Untuk Menyimpan Permintaan API

5. Muncul tampilan pop-up penyimpanan permintaan. Nama permintaan dapat ditulis ulang dengan memasukkan nama baru dikotak inputan 'Request name'. Permintaan API kemudian dapat disimpan kedalam koleksi baru dengan memilihnya di bagian 'Save to'.



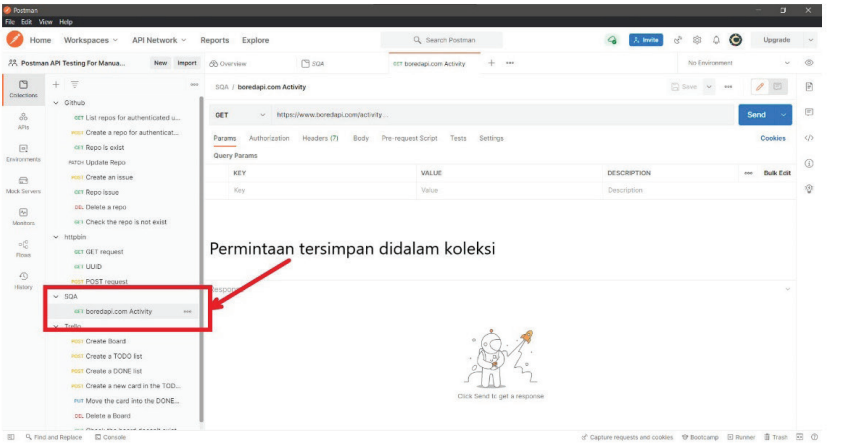
Gambar 4.20. Pengisian Informasi Permintaan API

6. Klik tombol 'Save' untuk menyimpan permintaan API kedalam koleksi yang sudah dipilih.



Gambar 4.21. Menyimpan Seluruh Informasi Permintaan API

7. Terlihat permintaan API yang sudah disimpan terdapat pada koleksi yang baru saja dibuat. Permintaan tersebut juga aktif dan terpilih.

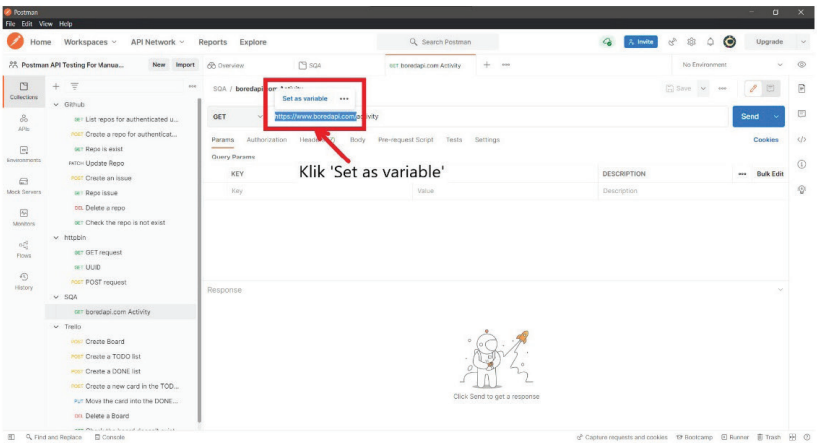


Gambar 4.22. Permintaan API Tersimpan Pada Koleksi Baru

Penyimpanan permintaan API pada contoh diatas hanya digunakan jika uji coba dilakukan pada satu buah *endpoint* (*endpoint* merupakan tautan situs yang memiliki alamat yang sama, hanya berbeda direktori). Jika terdapat banyak *endpoint* yang akan diuji coba, maka langkah terbaik adalah menyimpan alamat API kedalam sebuah variabel.

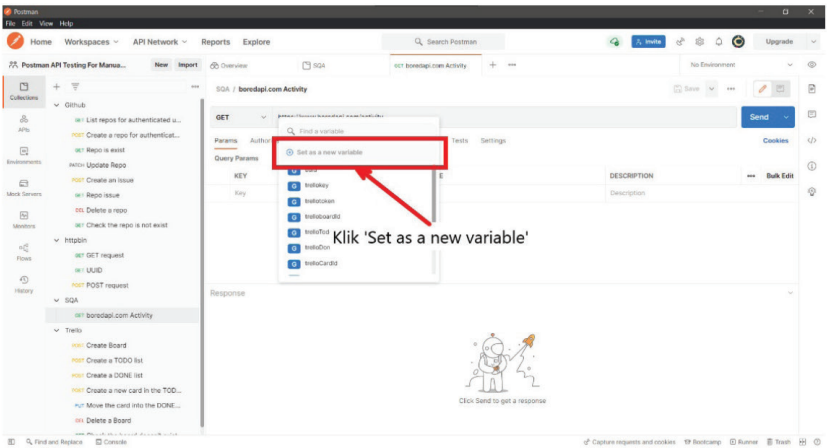
Alasannya adalah jika terdapat perubahan direktori *endpoint* atau terdapat penambahan *endpoint* baru maka tidak diperlukan untuk merubah keseluruhan alamat API. Berikut langkah untuk menyimpan alamat API kedalam sebuah variabel:

1. Sorot alamat API, maka akan muncul langsung pop-up dengan tulisan 'Set as variable'. Klik pop-up tersebut.



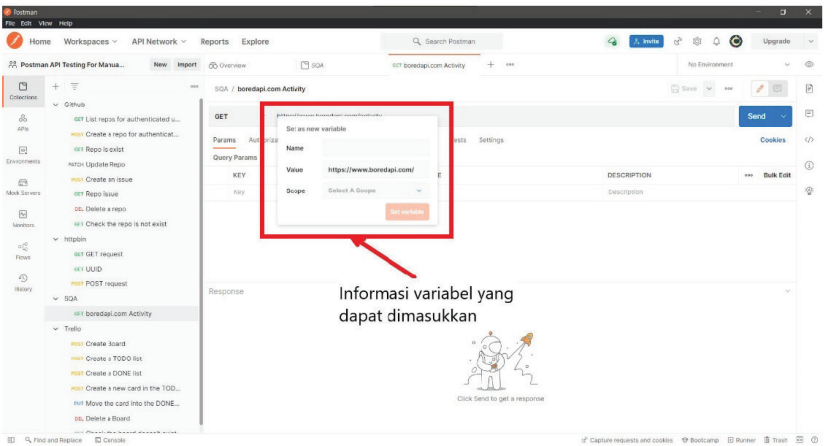
Gambar 4.23. Sorotan Alamat API

2. Akan muncul pop-up daftar variabel (jika ada). Untuk menyimpan kedalam variabel baru, maka klik 'Set as a new variable'.

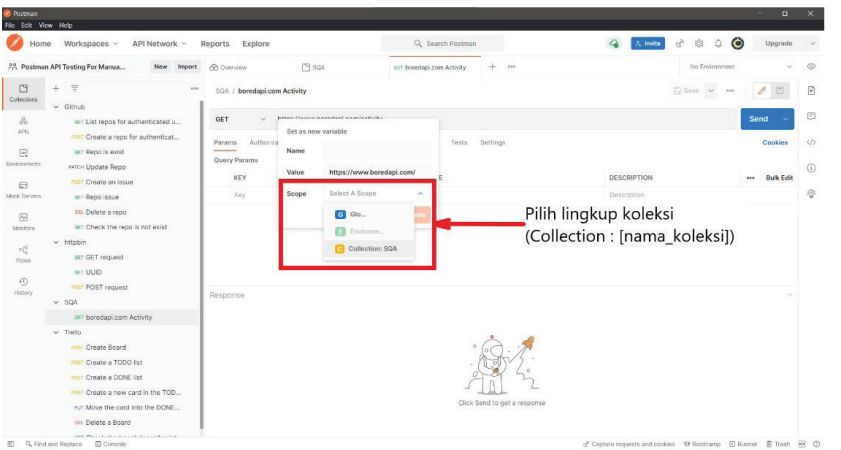


Gambar 4.24. Menyimpan Sebagai Variabel

3. Muncul tampilan baru dengan beberapa kotak inputan untuk nama (Name), nilai (Value) dan lingkup (Scope). Inputan nama akan memberikan nama variabel, inputan nilai akan memberikan alamat API dan lingkup akan memberikan lingkup variabel yang akan disimpan. Lingkup variabel yang dapat disimpan adalah global, lingkungan dan koleksi. Lingkup koleksi menjadi target lingkup penyimpanan variabel ini.

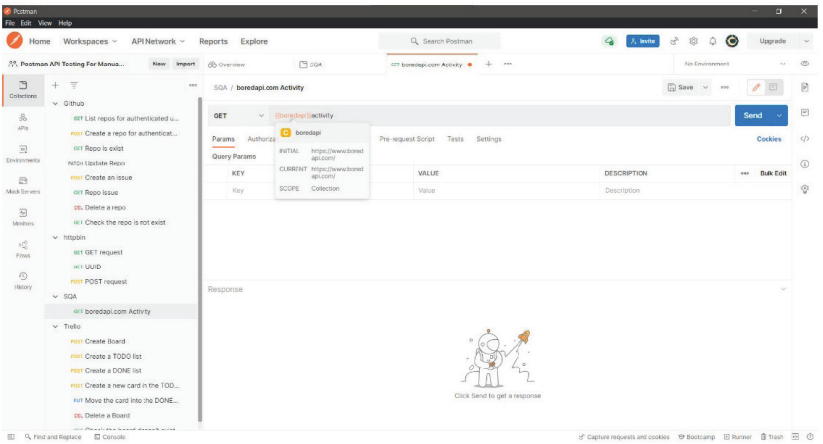


Gambar 4.25. Informasi variabel



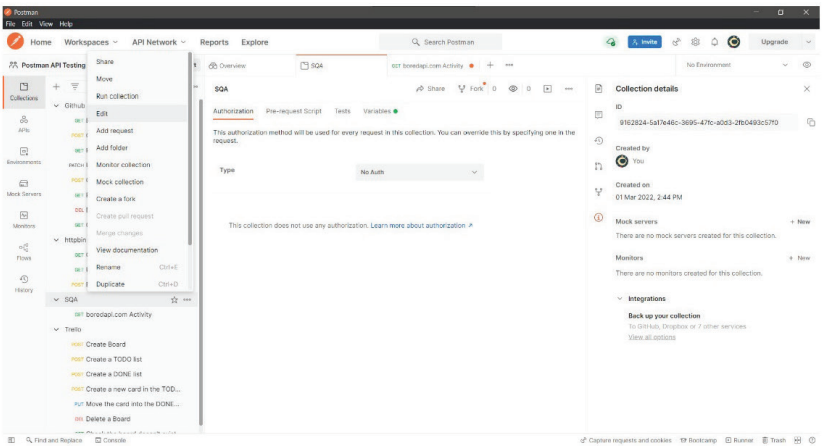
Gambar 4.26. Pilihan Lingkup Variabel

4. Simpan variabel dengan mengklik tombol ‘Set variabel’. Jika variabel sudah tersimpan maka dalam kotak inputan alamat permintaan API akan menggantikan nilai alamat dengan variabel yang sudah dibuat.



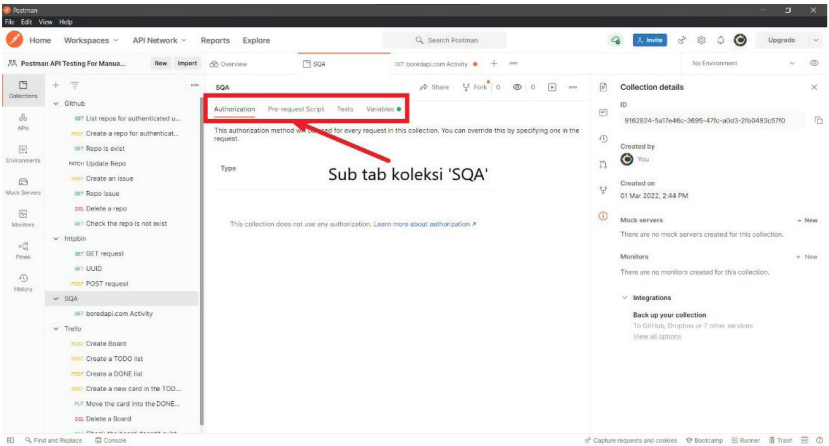
Gambar 4.27. Variabel Alamat API Terbuat

5. Variabel didalam koleksi yang sudah dibuat juga dapat dilihat melalui direktori koleksi yang dipilih.Pada koleksi yang terpilih, gerakkan pointer ke direktori koleksi tersebut. Kemudian terdapat tiga titik pada ujung kanan direktori, klik kemudian pilih ‘Edit’.



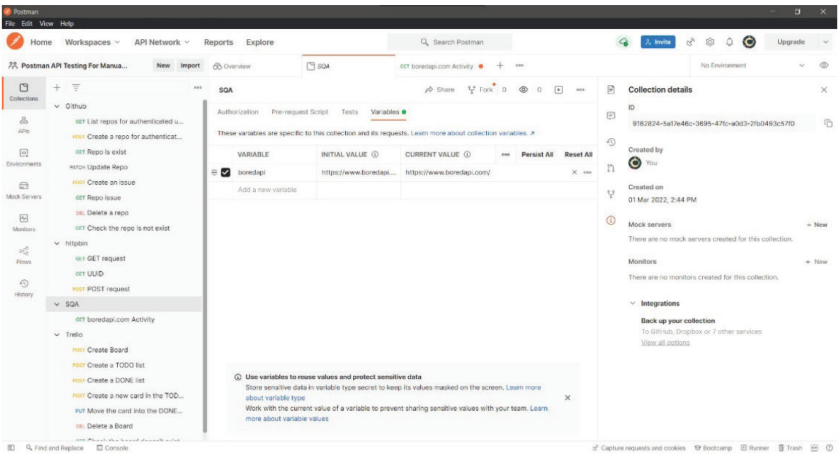
Gambar 4.28. Direktori Koleksi

6. Direktori koleksi akan muncul sebagai tab baru dan terlihat sub tab 'Variables' memiliki tanda baru.



Gambar 4.29. Subtab Koleksi

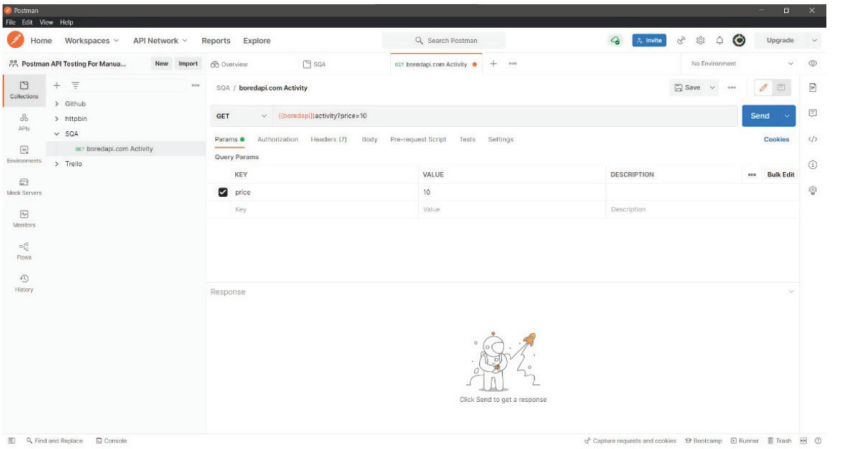
7. Klik sub tab 'Variables', dari sub tab tersebut dapat melihat daftar variabel yang sudah dibuat untuk koleksi tersebut. Nilai variabel tersebut dapat juga diganti jika diinginkan serta dapat juga menambahkan variabel baru. Kolom 'Initial Value' merupakan nilai saat pertama kali variabel ditetapkan dan kolom 'Current Value' merupakan nilai variabel saat ini. Nilai variabel dapat berubah sewaktu-waktu dengan ditunjukkan melalui kolom 'Current Value'



Gambar 4.30. Subtab Variabel Koleksi

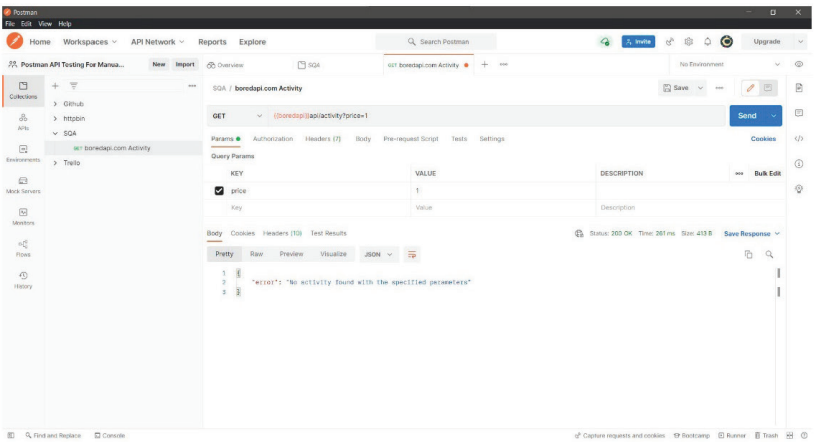
Permintaan API yang digunakan sebelumnya dapat diberikan sebuah parameter *query*. Parameter yang akan diberikan tersebut layaknya jika permintaan API yang digunakan dapat memberikan informasi yang sesuai dengan permintaan pengguna atau hal-hal spesifik dari informasi yang akan diterima. Berikut cara menambahkan parameter *query* untuk memberikan informasi spesifik dari permintaan API. Permintaan API yang digunakan yaituhttps://www.boredapi.com/api/activity :

1. Pada tampilan permintaan API yang difokuskan, sub bagian 'Query Params' terdapat kolom 'Key'. Isi kolom tersebut dengan nama *price*. Kolom nilai untuk *price* yaitu 1.0.



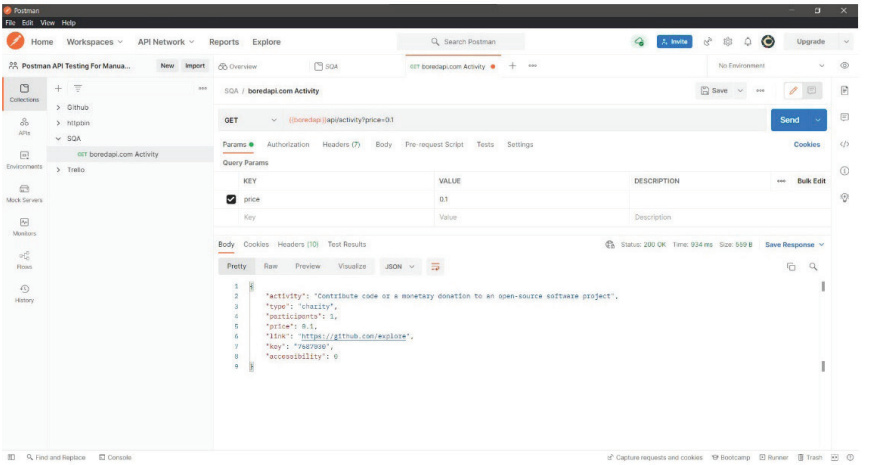
Gambar 4.31. Parameter Query

2. Kirim permintaan API tersebut dengan mengklik tombol 'Send'
3. Terlihat balasan permintaan dibagian 'Response'



Gambar 4.32. Balasan Harga Kosong (Tidak Ditemukan)

Balasan yang diterima mendapat kode '200 OK' akan tetapi isi balasan menyatakan bahwa tidak ada aktivitas dengan nilai harga yang dimaksud. Permintaan API tersebut dinyatakan sukses dan tidak memiliki kesalahan sehingga bukan terdapat galat pada permintaan API. Perlu diketahui, kode status menjadi acuan jika permintaan API tersebut gagal atau berhasil sehingga dalam kasus ini kode 200 menunjukkan permintaan berhasil. Adapun isi balasan dari permintaan API ini memiliki deskripsi yang jelas. Kadangkala isi balasan permintaan API mungkin tidak memiliki balasan yang jelas seperti dalam kasus ini, dimungkinkan untuk mendapat balasan hanya dengan obyek kosong ({}) atau sebuah array kosong ([]). Jika terdapat balasan kosong tersebut dan kode menunjukkan nomor 200, dapat dipastikan bahwa permintaan API berhasil, tetapi aplikasi tidak memiliki data dengan nilai tersebut. Supaya yakin bahwa permintaan API tersebut memang dapat bekerja dengan baik, gantilah nilai *price* tersebut dengan 0.1. Lihatlah balasan di bagian 'Response'.

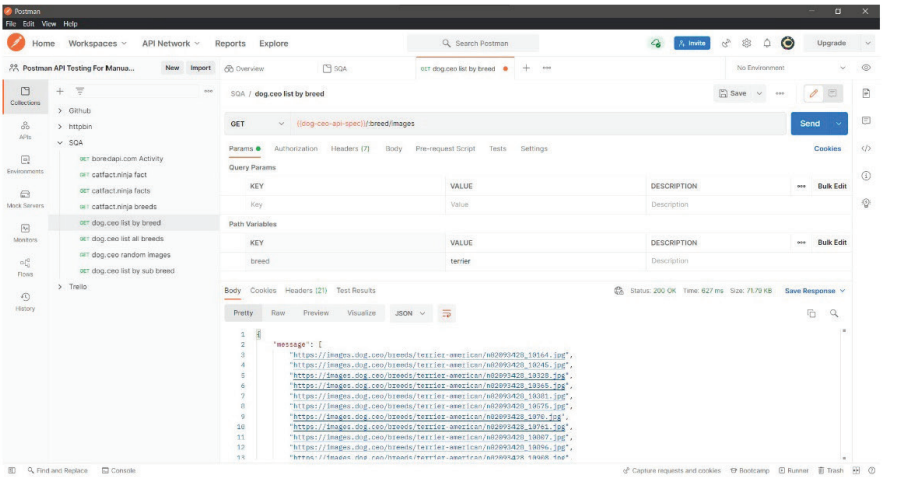


Gambar 4.33. Balasan Harga Tidak Kosong (Ditemukan)

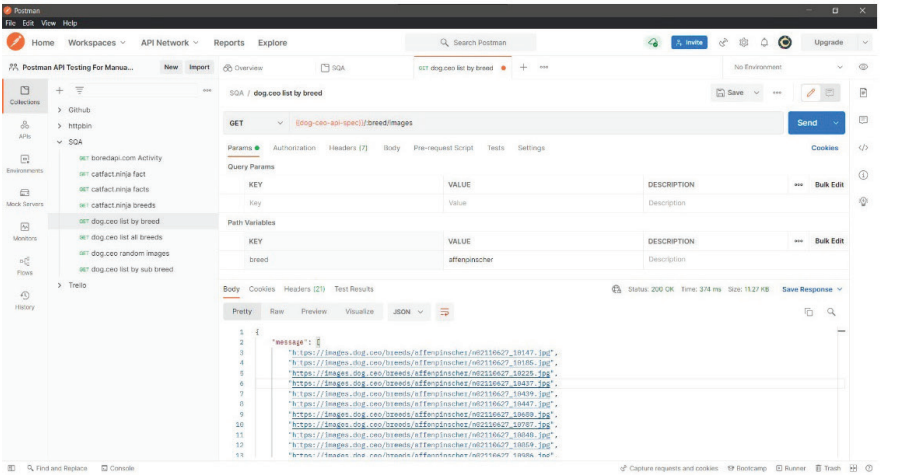
Pembaca diharapkan untuk melatih parameter *query* ini dengan menggabungkan parameter lainnya untuk melihat bagaimana balasan yang diterima sesuai dengan ekspektasi yang diharapkan atau tidak. Nilai parameter query juga sebaiknya diganti dengan nilai-nilai yang sesuai dan tidak sesuai untuk melihat apakah balasan yang diberikan sudah seharusnya atau justru terdapat galat. Jika ingin melihat lebih banyak

permintaan API, kunjungi situs https://catfact.ninja/atauhttps://dog.ceo/dog-api/.

Permintaan API juga dapat diberikan sebuah variabel direktori (Path Variable). Variabel direktori bekerja seperti parameter *query* akan tetapi nilai yang dapat berubah merupakan direktori alamat. Contoh penggunaan variabel direktori dapat melihat dua gambar berikut:



Gambar 4.34. Variabel Direktori 'terrier'



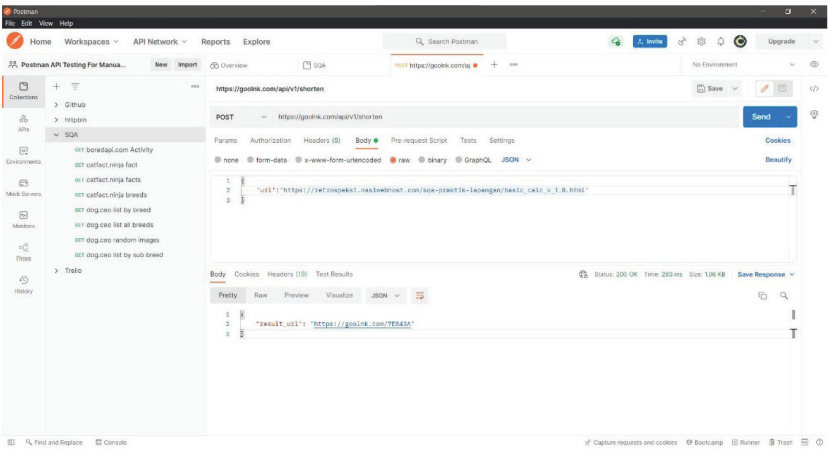
Gambar 4.35. Variabel Direktori 'affenpinscher'

Perubahan nilai variabel akan memberikan balasan dengan informasi yang berbeda, seperti pada dua gambar diatas. Pada gambar

pertama direktori 'terrier' akan memberikan kumpulan gambar ras dari terrier sedangkan direktori 'affenpinscher' akan memberikan kumpulan gambar ras dari affenpinscher.

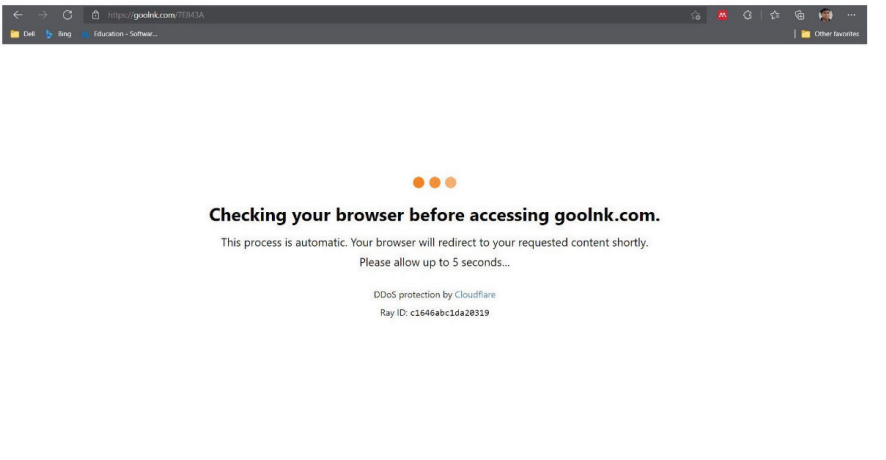
Metode-metode permintaan API yang sudah diuji coba sebelumnya merupakan metode jenis GET. Metode GET merupakan jenis metode yang meminta data dari sumber alamat yang diminta. Metode permintaan API selanjutnya yang dapat diuji coba yaitu metode POST. Jenis metode tersebut merupakan metode yang memberikan data baru kepada sumber alamat yang dituju. Metode tersebut berbeda dari metode GET dikarenakan metode GET dimaksudkan untuk meminta data, sehingga metode tersebut tidak dapat digunakan untuk merubah atau menambah data dari sumber [20]. Berikut cara menambah permintaan API dengan metode POST :

1. Klik tombol dropdown pada alamat permintaan API pada sisi kanan. Pilih POST
2. Masukkan Alamat tautan berikut:https://goolnk.com/api/v1/shorten
3. Masukkan parameter *query* bernama url. Kemudian masukkan nilai parameter berupa https://www.bing.com/
4. Klik tombol 'Send'

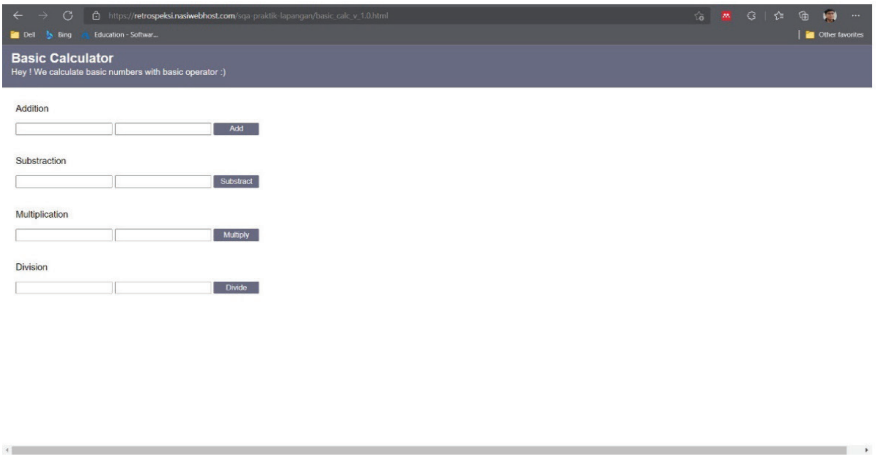


Gambar 4.36. Permintaan API Metode POST

Permintaan API tersebut akan memberikan balasan berupa hasil tautan yang sudah dipendekkan. Bukti dari tautan bisa dikunjungi dapat melihat gambar dibawah



Gambar 4.37. Pemuatan Cloudflare

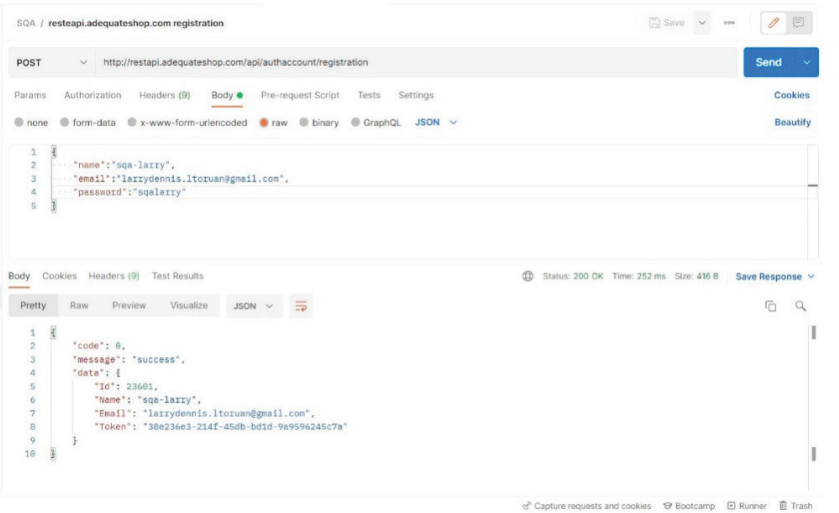


Gambar 4.38. Situs Terkunjungi

Metode permintaan API melalui POST pada praktik dilapangan biasanya membutuhkan sebuah otorisasi pesan dengan sebuah token (Authorization Header). Permintaan API dengan otorisasi ini dibutuhkan untuk dapat melakukan permintaan API dengan aman. Menggunakan alamat api http://restapi.adequateshop.com(dokumentasi: https://www.appsloveworld.com/sample-rest-api-url-for-testing-with-

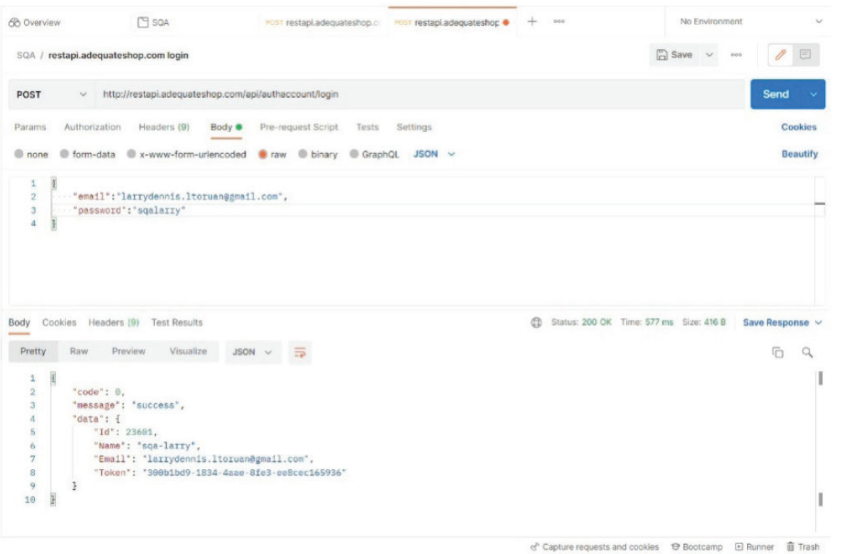
authentication/),berikut permintaan API dengan menggunakan otorisasi (Authorization Header) pada aplikasi Postman:

1. Masukkan alamat api dan permintaan untuk registrasi akun.



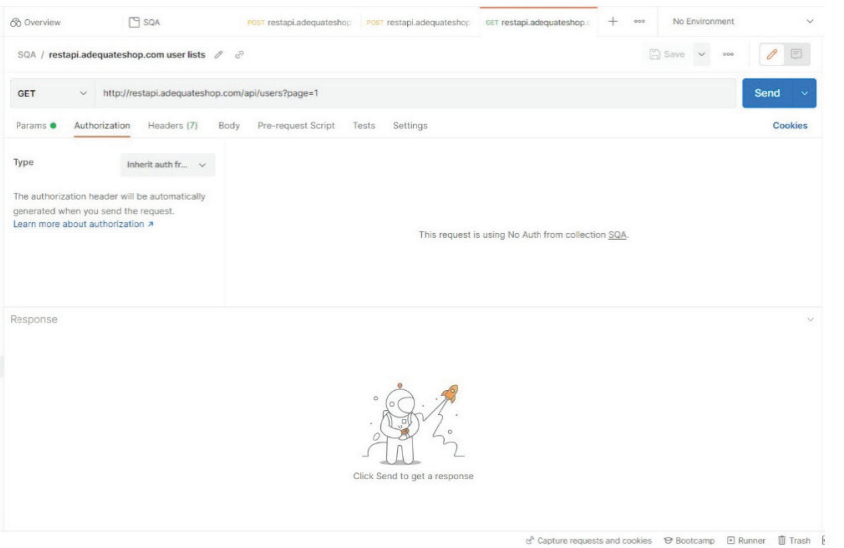
Gambar 4.39. Registrasi Ke Alamat API

2. Klik tombol 'Send'. Kemudian login dengan menggunakan informasi akun yang sudah dibuat.

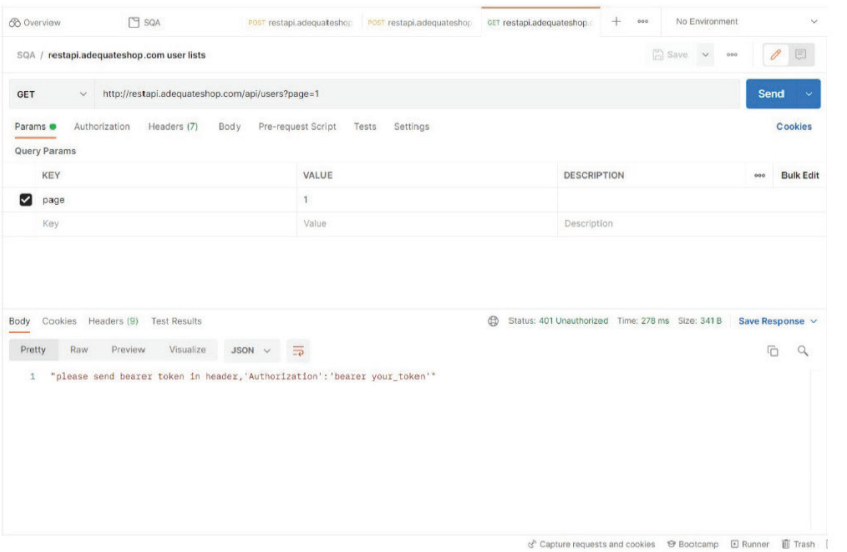


Gambar 4.40. Login Ke Alamat API

3. Setelah login, kita coba mengambil data kumpulan pengguna dari alamat tersebut tanpa menggunakan otorisasi. Balasan status yang diterima adalah kode '401 Unauthorized', menandakan belum terotorisasi untuk melakukan permintaan.

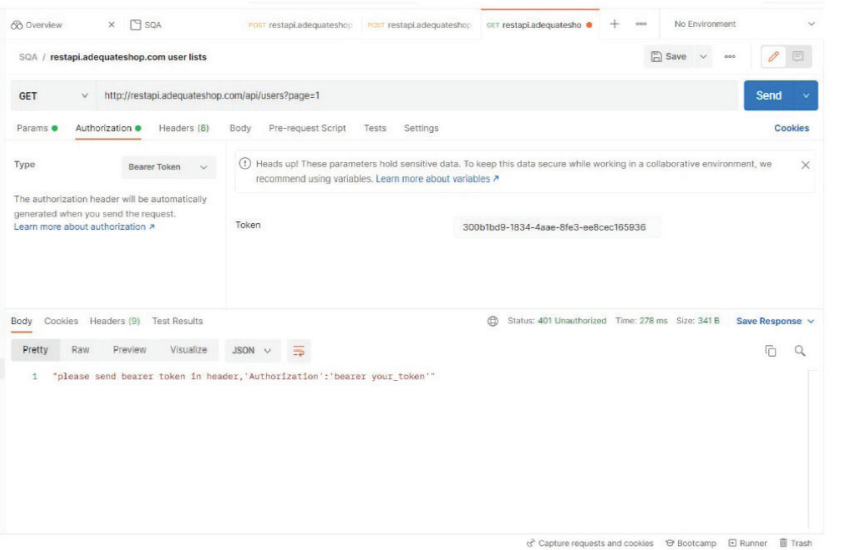


Gambar 4.41. Token Otorisasi Kosong



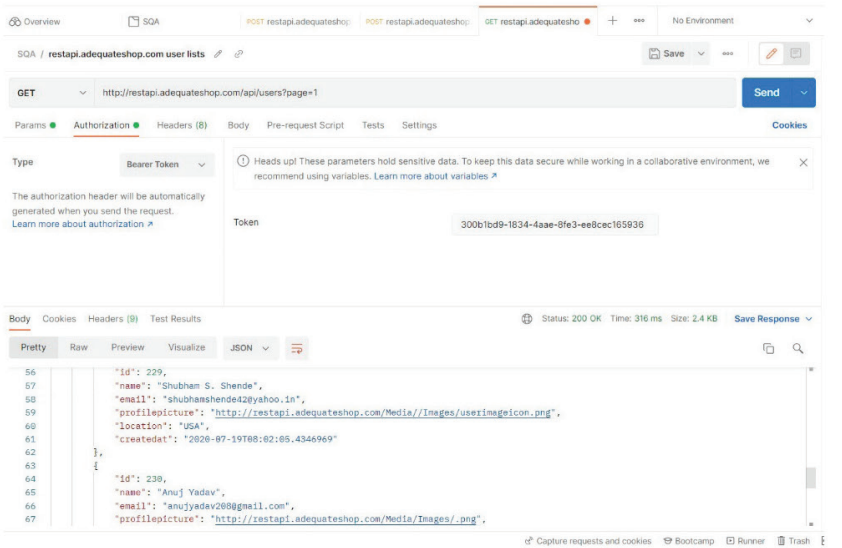
Gambar 4.42.Permintaan API Tidak Terotorisasi (Ditolak)

4. Kemudian kita coba kembali mengambil data kumpulan pengguna dari alamat yang sama dengan menggunakan token otorisasi. Token yang diambil berasal dari data login. Salin token tersebut, kemudian pilih sub tab 'Authorization' dibawah kotak inputan permintaan API. Klik tombol dropdown 'Type' dan pilih 'Bearer Token'. Tempel token kedalam kotak inputan 'Token'.



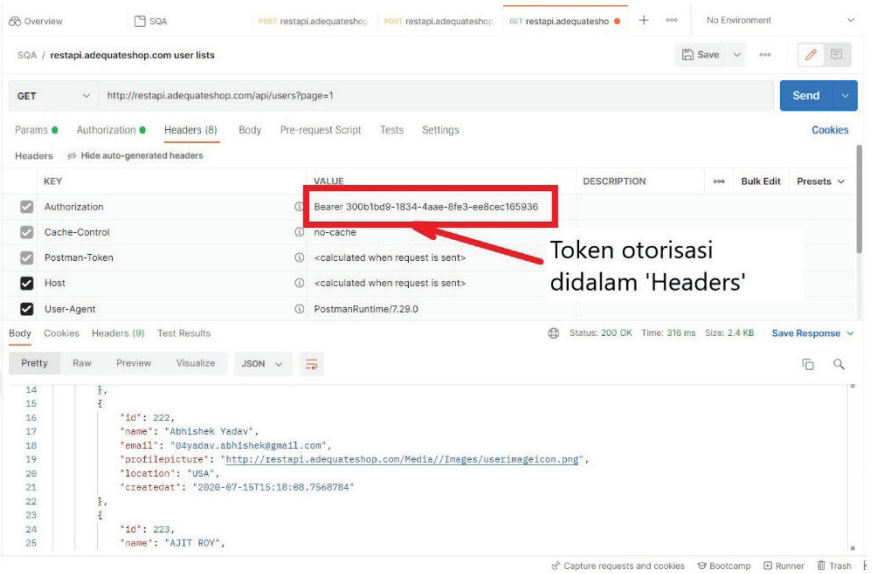
Gambar 4.43. Token Terisi

5. Klik tombol 'Send'. Balasan diterima dengan kode status '200 OK' dan didalam *bodyresponse* terdapat kumpulan data yang diminta.



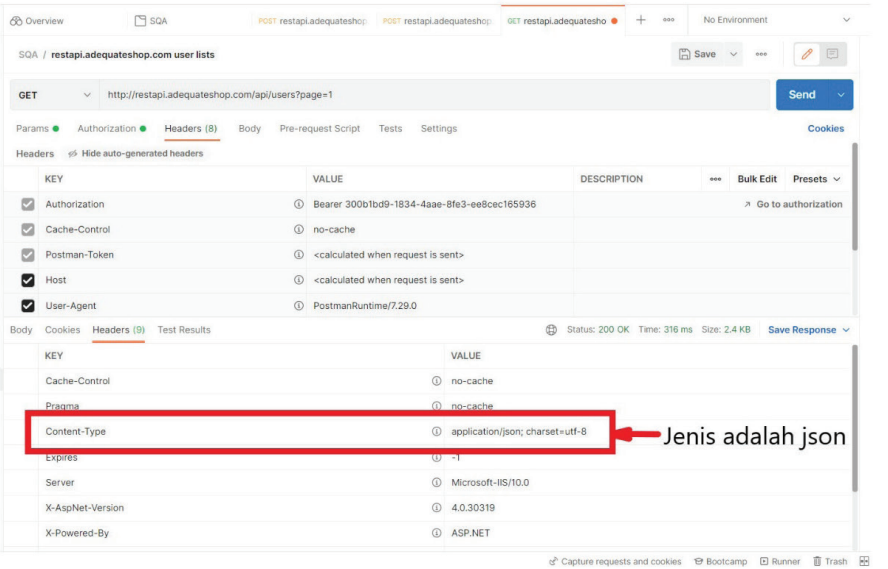
Gambar 4.44. Permintaan Terotorisasi (Diterima)

Authorization Header sendiri sebenarnya terletak pada didalam *header* dari sebuah permintaan API. Header dari sebuah permintaan API dapat dilihat dengan mengklik sub tab 'Headers'.



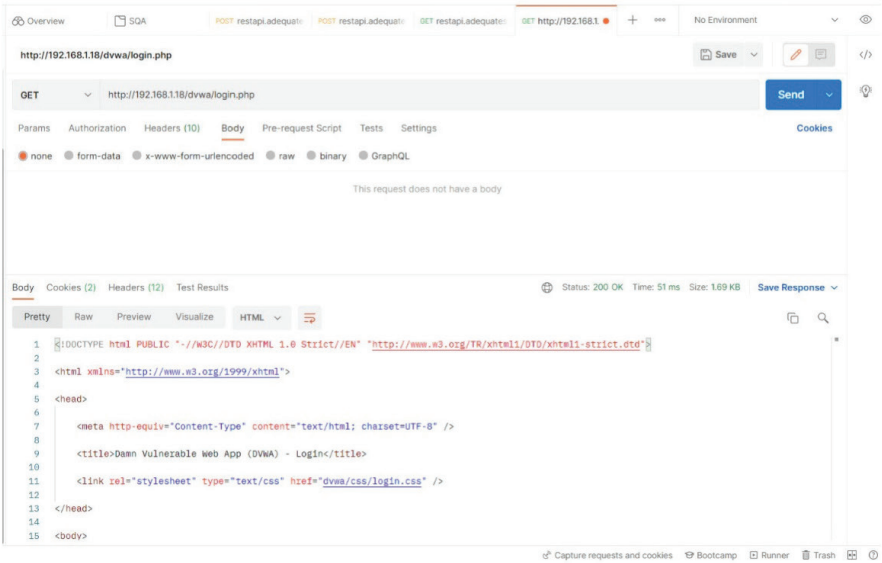
Gambar 4.45. Header Dari Permintaan

Terdapat token otorisasi yang sudah dipakai didalam *header* permintaan API tersebut. Tidak hanya itu, informasi lainnya seperti aplikasi yang dipakai untuk mengirim permintaan API juga diberikan. *Header* tidak hanya terdapat saat pengiriman permintaan API dilakukan, penerimaan balasan juga akan menerima *header*. Tab 'Headers' pada bagian 'Response' dapat dilihat untuk mengetahui informasi tersebut. Contoh penting dari balasan seperti isi konten yang diterima (Content-Type) berjenis json dengan enkoding berjenis 'utf-8'.



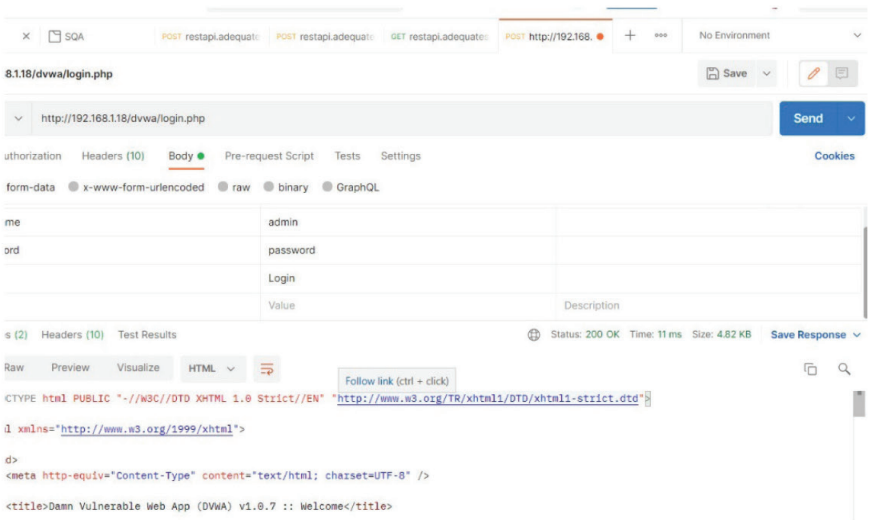
Gambar 4.46. Header Dari Pesan Balasan

Ketika membuat permintaan API dengan metode POST, terdapat *body* permintaan yang dikirimkan dalam berbagai bentuk. *Body* permintaan yang sudah dipraktikan merupakan format JSON. Format JSON merupakan sebuah obyek Javascript yang dapat diterima oleh mesin untuk berkomunikasi, biasanya dipakai untuk aplikasi web dan server [21]. Tergantung bentuk permintaan API tersebut, format lain dapat dikirimkan untuk menyesuaikan jenis permintaan yang dapat diterima oleh mesin tersebut. Contoh lain permintaan API dengan menggunakan *body* form-data:



Gambar 4.47. Body Balasan

Setelah mengisi data form dan mengirim permintaan API ke alamat tersebut, balasan diterima dengan kode status '200 OK' yang menandakan berhasil login dan sampai pada halaman utama.



Gambar 4.48. Body Permintaan API Dengan Form-Data

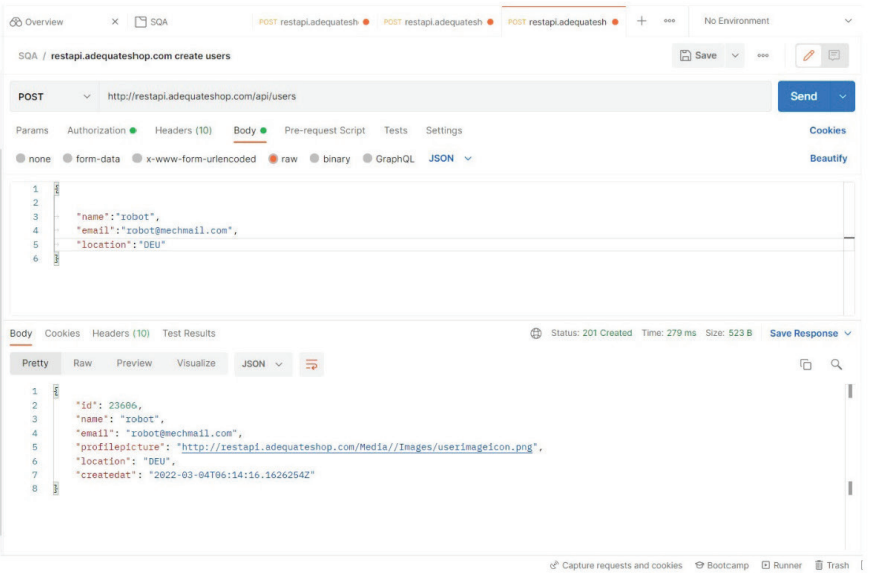
4.3. Latihan Pertama, Uji Coba API

Latihan pertama ini akan mencoba membuat sebuah permintaan API dengan metode POST. Alamat permintaan API adalah:

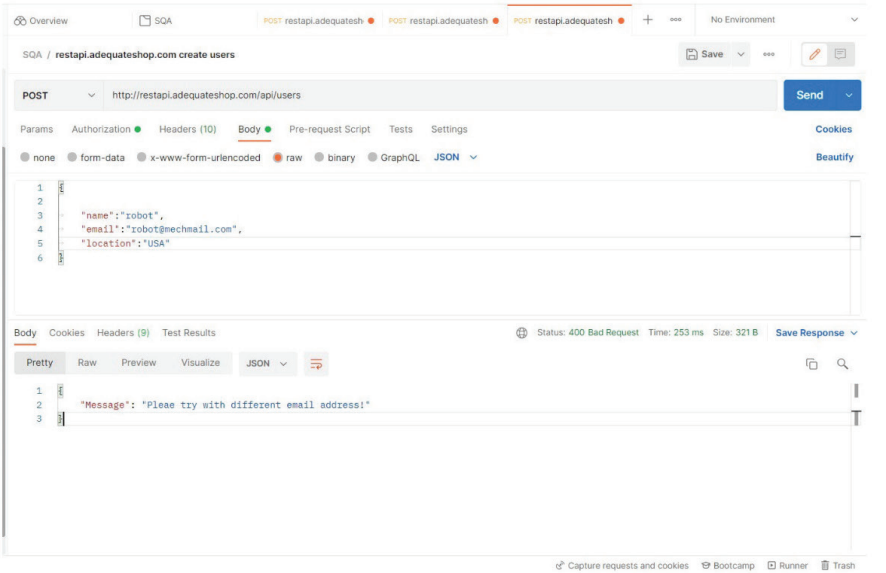
http://restapi.adequateshop.com/api/users

Berdasarkan cara membuat permintaan API sebelumnya, permintaan kali ini akan mendapat sedikit modifikasi pada *body* permintaan. Modifikasi terdapat pada nilai 'name', 'email' dan 'location'. Kemudian dengan membuat permintaan dengan alamat yang sama, modifikasi hanya pada nilai 'location' selain dari nilai 'location' yang sudah diberikan sebelumnya. Setelah mendapat seluruh balasan, periksalah jika sudah sesuai dengan dokumentasi permintaan API yang ada.

Solusi dari latihan pertama dapat melihat gambar dibawah ini:



Gambar 4.49. Solusi Latihan Pertama, Pembuatan User Berhasil



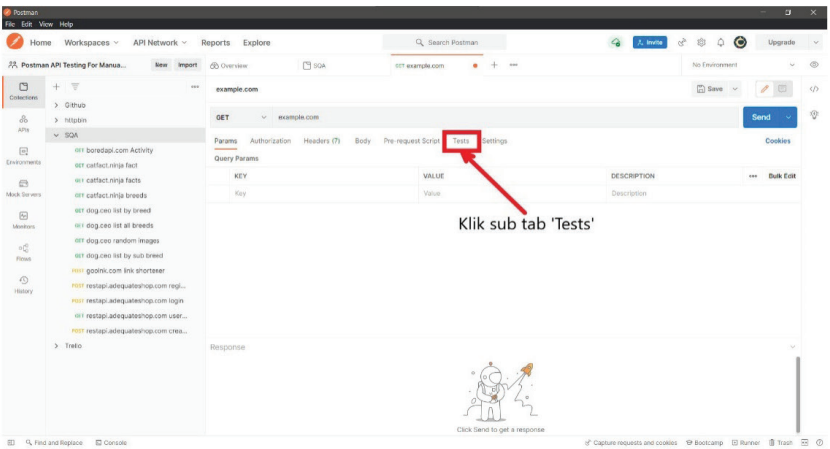
Gambar 4.50. Solusi Latihan Pertama, Pembuatan UserGagal

4.4. Latihan Kedua, Uji Coba API

Sebelum masuk ke latihan kedua dari uji coba API, terdapat beberapa penjelasan yang perlu disampaikan sehingga pembaca dapat lebih memahami uji coba API dengan skenario yang lebih rumit. Penjelasan ini termasuk bagaimana membuat sebuah uji coba dari pembuatan API dan bagaimana cara untuk membuat data dari sebuah permintaan API dapat diberikan kepermintaan API lainnya.

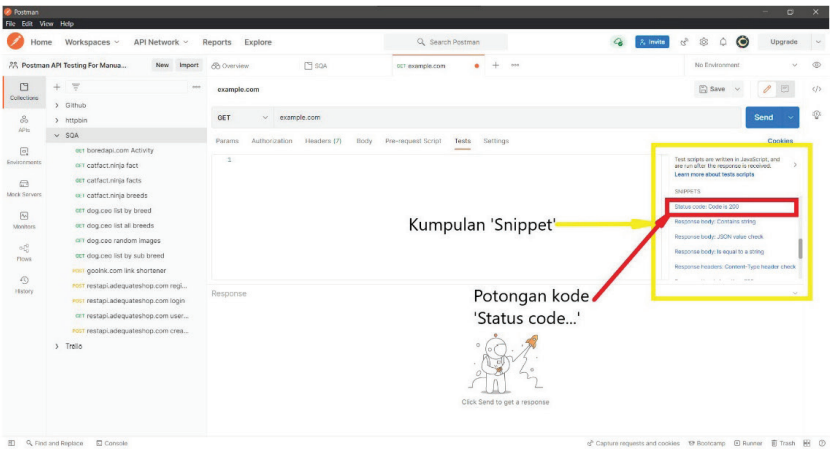
Uji coba dari pembuatan API menggunakan aplikasi Postman sebenarnya hanya menyatakan jika permintaan API yang dikirimkan sudah sesuai dengan balasan yang diberikan. Balasan yang sesuai seperti kode status yang diinginkan menjadi nilai yang diuji cobakan untuk menyatakan pengiriman dan penerimaan API lolos uji coba. Berikut cara pembuatan sebuah uji coba dari permintaan API:

1. Buatlah permintaan API yang baru, isi kotak inputan permintaan API dengan https://www.example.com. Klik sub tab 'Tests'.



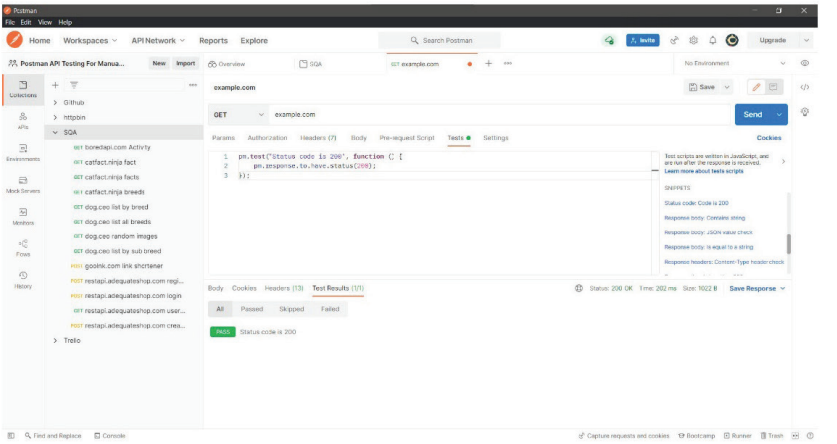
Gambar 4.51. Subtab Tests Pada Permintaan API

2. Pada bagian 'Snippets' gulir kebawah dan klik potongan kode bernama 'Status code: Code is 200'.



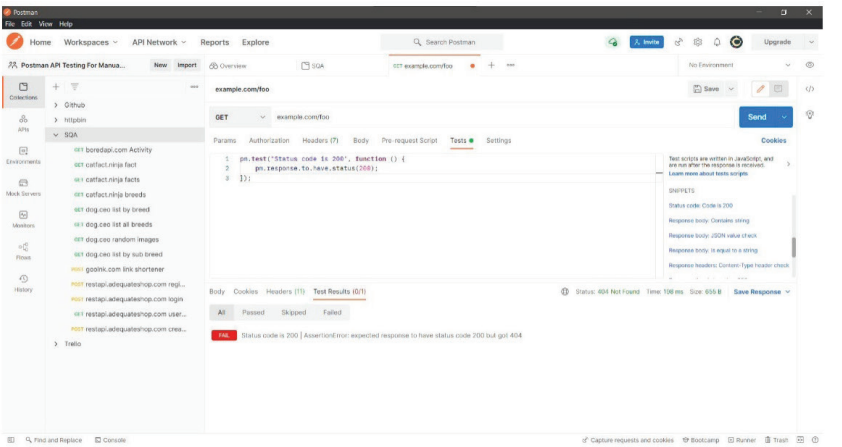
Gambar 4.52. Potongan Kode

3. Potongan kode akan muncul di badan 'Tests'. Potongan kode tersebut akan memeriksa jika balasan dari permintaan API yang diberikan mendapat kode status 200 yang menandakan permintaan sukses diterima.



Gambar 4.53. Asersi Berhasil

4. Jika balasan sebelumnya berhasil, maka kita akan coba untuk membuat permintaan tersebut memiliki balasan yang gagal. Rubah alamat permintaan API sehingga hasil uji coba akan memberikan balasan gagal.



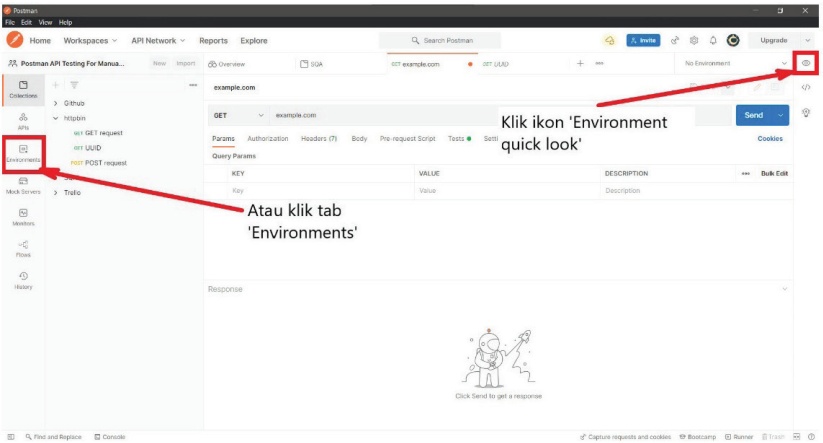
Gambar 4.54. Asersi Gagal

Terdapat alasan mengapa pembuatan uji coba dicoba untuk gagal setelah pembuatan uji coba berhasil sesuai dengan ekspektasi. Alasan tersebut tidak lain adalah untuk memastikan uji coba yang dibuat dapat memberikan informasi yang tepat ketika uji coba dinyatakan lulus atau dinyatakan gagal jika seharusnya terdapat kesalahan pada komponen yang diuji coba.

Potongan kode tersebut jika dipecah memiliki sebuah fungsi bernama `pm.test`. Fungsi tersebut digunakan untuk membuat bentuk uji coba. Fungsi ini dapat digunakan dengan beberapa asersi yang berhubungan (kode status 200 dengan 201, 205 dsb) serta tidak akan menghentikan seluruh uji coba jika terjadi sebuah galat. Fungsi tersebut juga menerima dua buah parameter, dengan parameter pertama merupakan pesan yang dapat diisi secara bebas bertipe data string dan sebuah pemanggilan fungsi balikan untuk mengeksekusi perintah-perintah yang ada didalam fungsi balikan tersebut. Potongan kode tersebut memiliki fungsi didalam fungsi balikan bernama `pm.response`. Fungsi ini digunakan untuk memeriksa balasan baik pada kode status, *body* balasan ataupun *header* balasan.

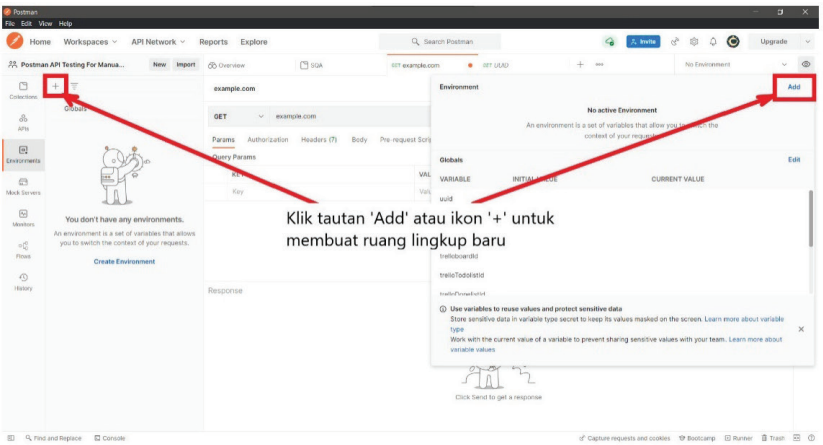
Variabel-variabel yang ada pada aplikasi Postman memiliki lingkup variabel yang berbeda-beda, perbedaan ini dimaksudkan untuk memisahkan kegunaan variabel-variabel tersebut saat permintaan API dijalankan. Lingkup variabel yang tersedia pada aplikasi Postman antara lain lingkup global (Global), lingkup lingkungan (Environment), lingkup koleksi (Collection) dan lingkup lokal (Local) [22]. Lingkup variabel tersebut memiliki tingkatan pemakaian. Lingkup lokal akan menjadi variabel dengan tingkat tertinggi yang berarti variabel lokal menjadi pilihan terdahulu untuk pemakaian variabel dan lingkup global menjadi variabel pilihan terakhir untuk pemakaian variabel jika tidak ada variabel dari lingkup lain terpilih. Membuat variabel dalam lingkup lingkungan dapat dilakukan dengan cara berikut:

1. Klik tab ‘Environments’ atau ikon ‘Environment quick look’.



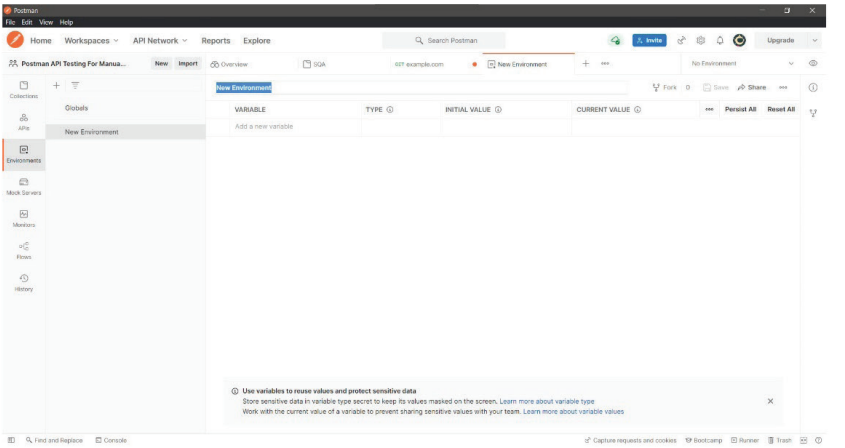
Gambar 4.55. Ruang Lingkup

2. Pada tab ‘Environments’ klik ikon ‘+’ untuk menambah ruang lingkup dan pada ‘Environment quick look’ click ‘Add’.

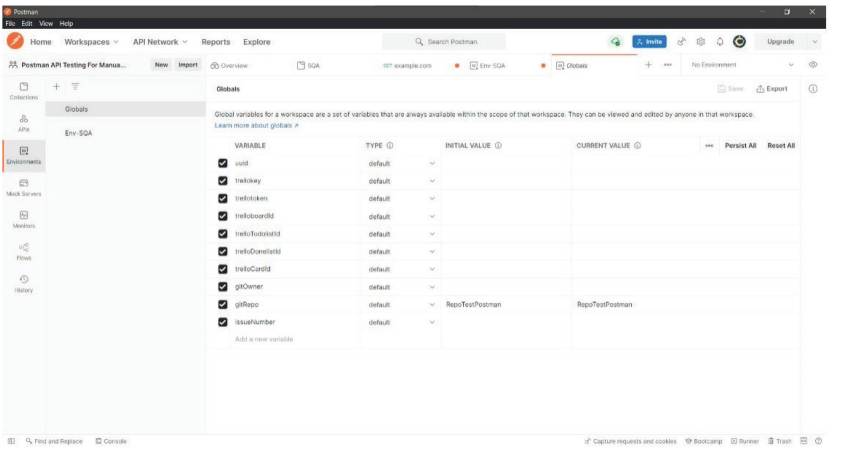


Gambar 4.56. Membuat Ruang Lingkup

3. Ruang lingkup baru akan dibuat dengan nama ‘New Environment’. Nama ruang lingkup kemudian dapat diganti sesuai dengan kebutuhan. Ruang lingkup global secara otomatis sudah tersedia, sehingga hanya menambahkan variabel baru.

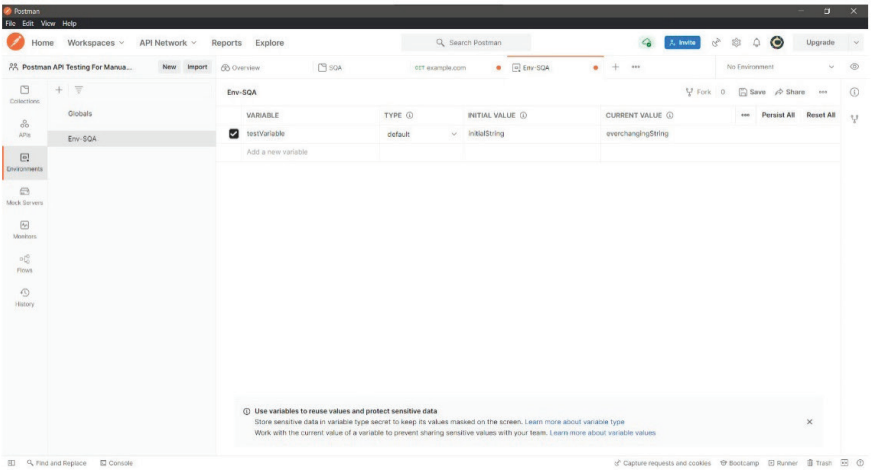


Gambar 4.57. Ruang Lingkup Baru



Gambar 4.58. Ruang Lingkup Global

4. Pada kolom ‘Variable’ diisi dengan nama variabel dan nilai awalan diisi pada kolom ‘Initial Value’. Kolom ‘Current Value’ dapat diisi jika nilai variabel yang diberikan tidak tetap. Perbedaan mendasar dari ‘Current Value’ dan ‘Initial Value’ adalah nilai ‘Initial Value’ seperti layaknya mendeklarasikan sebuah variabel dengan nilai bawaan, sedangkan ‘Current Value’ mendeklarasikan sebuah variabel dengan nilai tidak konstan atau berubah-ubah sewaktu-waktu. Nilai pada kolom ‘Current Value’ menjadi acuan penggunaan nilai variabel.



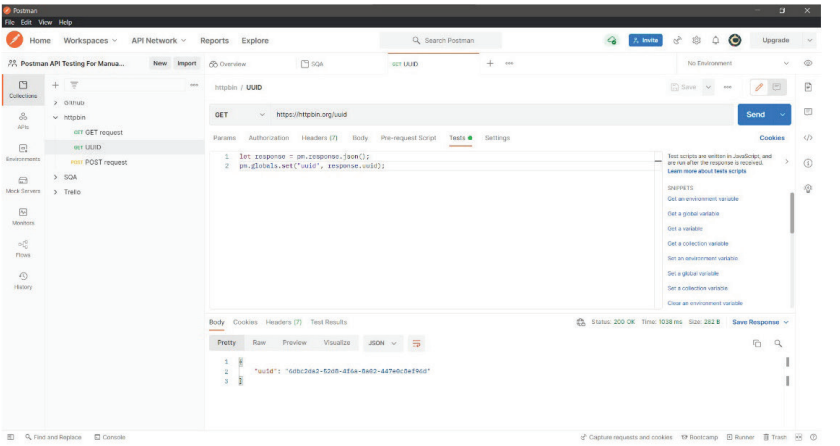
Gambar 4.59. Ruang Lingkup Bernama 'Env-SQA'

Penggunaan ruang lingkup variabel menjadi penting ketika uji coba permintaan API menjadi sangat banyak dan terdapat lebih banyak koleksi yang akan di uji coba. Ruang lingkup variabel juga berguna untuk membuat permintaan API berantai sehingga permintaan-permintaan API dapat dijalankan secara simultan. Contoh penggunaannya sebagai berikut:

1. Buat permintaan API dengan alamat https://httpbin.org/uuid, metode permintaan yang digunakan adalah GET. Klik sub tab 'Tests', isi badan 'Tests' dengan kode sebagai berikut:

```
let response = pm.response.json();
pm.globals.set("uuid", response.uuid);
```

2. Buatlah variabel global dengan nama 'uuid'. Kemudian, kirim permintaan API.



Gambar 4.60. Pemberian Nilai Variabel Global 'uuid'

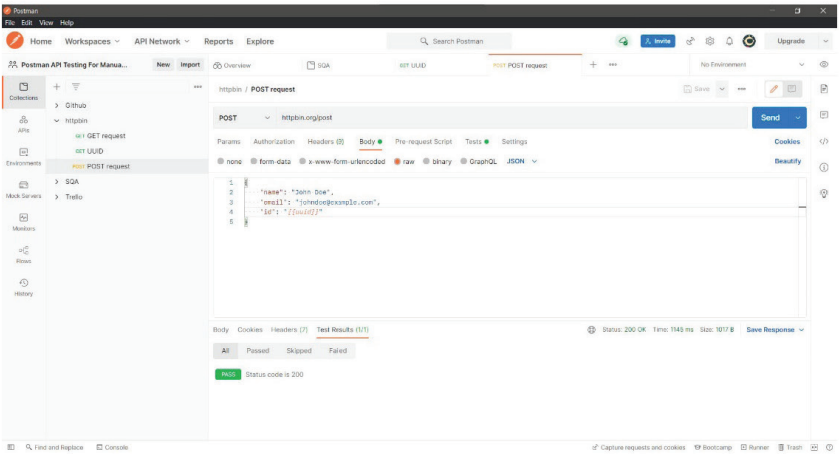
3. Terdapat balasan dengan data 'uuid'. Periksalah jika data 'uuid' tersebut sama dengan nilai variabel global pada aplikasi Postman.

4. Buat permintaan API baru dengan alamat https://httpbin.org/post, metode permintaan yang digunakan adalah POST. Klik sub tab 'Body' dan isi badan 'Body' tersebut dengan data dalam format JSON. Data tersebut sebagai berikut:

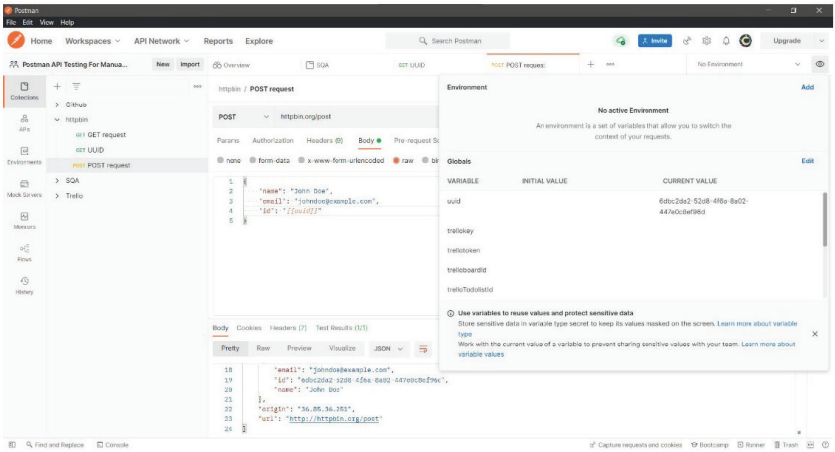
```
{
"name": "John Doe",
"email": "johndoe@example.com",
"id": "{{uuid}}"
}
```

5. Klik sub tab 'Tests', isi badan 'Tests' tersebut dengan potongan kode untuk memeriksa jika kode status yang diberikan bernomor 200. Tujuan dari mengisi badan 'Tests' yaitu untuk menguji coba jika permintaan API mendapat balasan yang sesuai.

6. Kirim permintaan API tersebut. Terlihat balasan yang didapatkan berisi data nomor 'uuid' yang sudah didapatkan dari permintaan API sebelumnya.



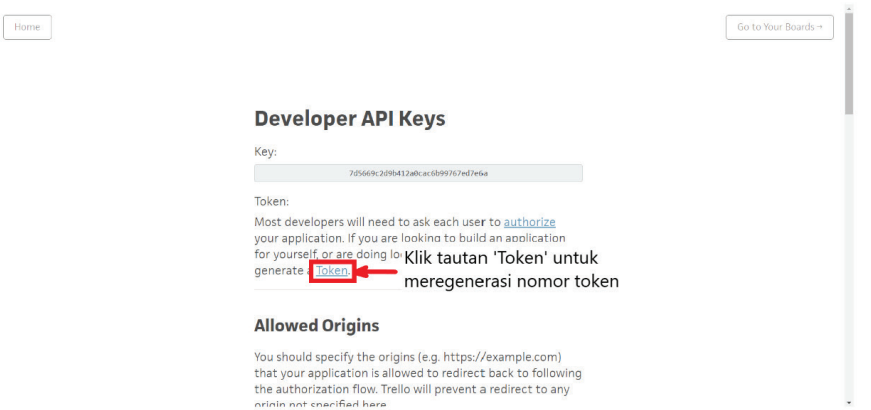
Gambar 4.61. Variabel Global Dipakai Pada Permintaan API



Gambar 4.62. Balasan Dengan Menggunakan Variabel Global

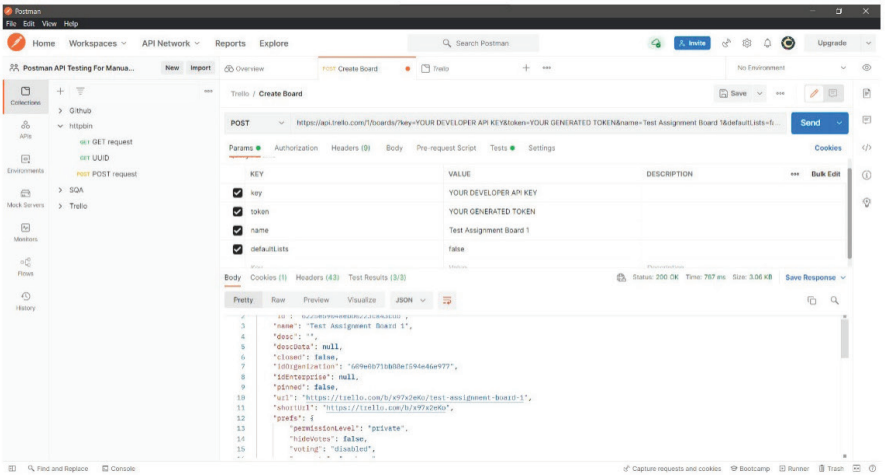
Potongan kode pada langkah pertama dapat dijelaskan sebagai berikut. Variabel 'response' pertama kali dideklarasikan sebagai variabel lokal dan akan mendapat obyek dari balasan permintaan API pada aplikasi Postman dalam format JSON. Aplikasi Postman akan menerima seluruh balasan permintaan tersebut dan disimpan secara sementara. Fungsi 'pm. response' merupakan fungsi yang digunakan untuk memberikan akses ke data yang sudah diterima tersebut dengan '.json()' merupakan fungsi untuk merubah data tersebut kedalam obyek format yang diinginkan, dalam kasus ini format JSON yang diinginkan [23].

Latihan kedua dari uji coba API ini akan menggunakan dokumentasi dari aplikasi Trello (https://developer.atlassian.com/cloud/trello/rest/api-group-boards/).Trello merupakan aplikasi manajemen proyek yang dapat digunakan secara gratis. Pembaca harus mendaftar terlebih dahulu untuk dapat menggunakan aplikasi Trello (https://trello.com/signup). Setelah mendaftarkan sebuah akun, dibutuhkan kunci API dari aplikasi Trello supaya dapat menggunakan permintan-permintaan API yang sudah disediakan. Tatutan berikut :https://trello.com/app-keydapat digunakan untuk langsung mengambil kunci API. Kunci API ini akan diperlukan disetiap parameter permintaan API dengan nama parameternya adalah 'key'. Nomor token juga diperlukan supaya akses ke permintaan API diperbolehkan. Nomor token yang sudah diregenerasi juga akan diisi kedalam parameter permintaan API dengan nama parameternya adalah 'token'.

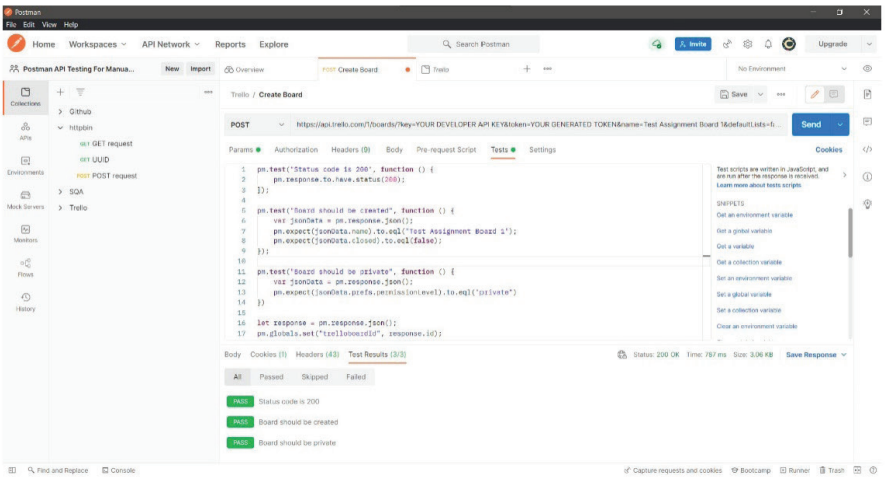


Gambar 4.63. Generasi Token

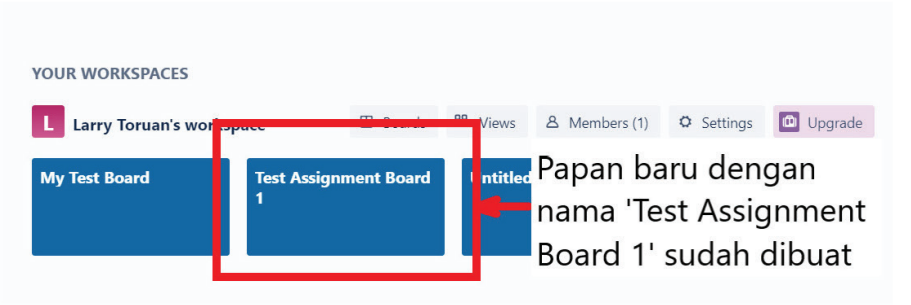
Berikut contoh pembuatan papan pada aplikasi Trello dengan menggunakan permintaan API:



Gambar 4.64. Pembuatan Papan Melalui Permintaan API



Gambar 4.65. Badan Uji Coba Pembuatan Papan, Seluruh Uji Coba Lulus



Gambar 4.66. Bukti Pembuatan Papan Pada Aplikasi Trello

Latihan uji coba dengan aplikasi Trello yaitu membuat sebuah koleksi dengan permintaan-permintaan API yang sudah ditentukan untuk membuat sebuah kartu 'list' hingga memeriksa jika papan sudah dihapus. Solusi dari pembuatan permintaan-permintaan API dapat dilihat di akhir sub bab. Permintaan-permintaan API-nya sebagai berikut:

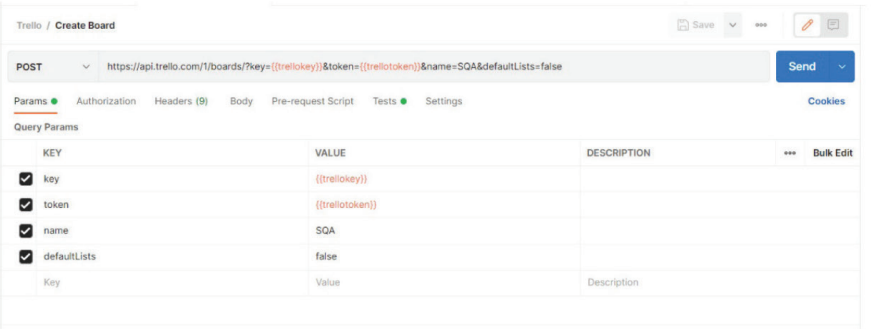
1. Buatlah terlebih dahulu sebuah ruang lingkup (environment) untuk menyimpan seluruh variabel yang akan diperlukan.
2. Buatlah sebuah papan dengan parameter-parameter yang wajib dibutuhkan beserta informasi uji coba yang menyatakan jika permintaan berhasil diterima, papan telah dinyatakan ada dan sudah dibuat serta status dari papan adalah privat. Nama papan ini adalah 'SQA Trello Board'
3. Buatlah sebuah susun 'list' bernama 'TODO' dengan parameter-parameter yang wajib dibutuhkan beserta informasi uji coba yang menyatakan jika permintaan berhasil diterima, nama dari susunan sudah sesuai serta terdapat variabel global untuk ditetapkan nilai nomor id susun 'list'. Variabel ini untuk nomor id 'TODO'.
4. Buatlah sebuah susun 'list' bernama 'DONE' dengan parameter-parameter yang wajib dibutuhkan beserta informasi uji coba yang menyatakan jika permintaan berhasil

diterima, nama dari susunan sudah sesuai serta terdapat variabel global untuk ditetapkan nilai nomor id susun 'list'. Variabel ini untuk nomor id 'DONE'.

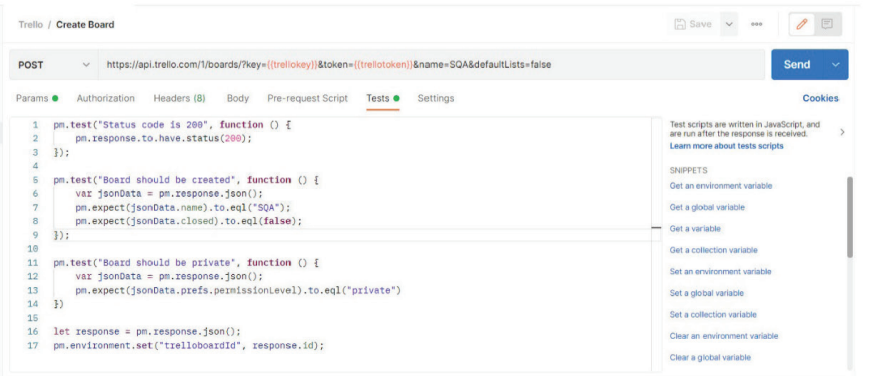
5. Buatlah sebuah kartu didalam susun 'list' bernama 'TODO' dengan parameter-parameter yang wajib dibutuhkan beserta informasi uji coba yang menyatakan jika permintaan berhasil diterima, nama dari kartu sesuai dengan nama 'Belajar Aplikasi Postman', nama kartu dinyatakan ada didalam susun 'list' tersebut, serta nama kartu dinyatakan berada pada papan 'SQA Trello Board'.
6. Pindahkan kartu yang sudah dibuat didalam susun 'list' 'TODO' kedalam susun 'list' 'DONE'. Isilah parameter-parameter permintaan API yang wajib dibutuhkan berserta informasi uji coba yang menyatakan jika permintaan berhasil diterima, nama kartu sudah sesuai dengan nama yang sudah ditetapkan sebelumnya serta kartu dinyatakan ada didalam susun 'list' tersebut.
7. Hapuslah papan tersebut dengan parameter-parameter yang wajib dibutuhkan beserta informasi uji coba yang menyatakan jika permintaan berhasil diterima.
8. Periksa jika papan tersebut sudah tidak ada lagi. Isilah dengan parameter-parameter yang wajib dibutuhkan beserta informasi uji coba yang menyatakan jika permintaan tidak dapat ditemukan.

Perlu diperhatikan untuk setiap pengisian parameter-parameter yang memiliki unsur nomor id, nomor token dan kunci API harus diambil dari variabel global yang sudah ditetapkan. Pengisian nilai parameter-parameter ini tidak diperbolehkan dilakukan secara manual (seperti salin dan tempel).

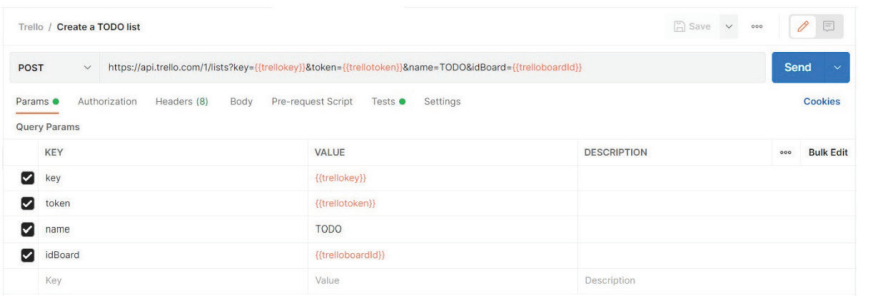
Solusi dari pembuatan permintaan-permintaan API dapat melihat gambar-gambar berikut dibawah ini :



Gambar 4.67. Pembuatan Papan, Parameter Wajib

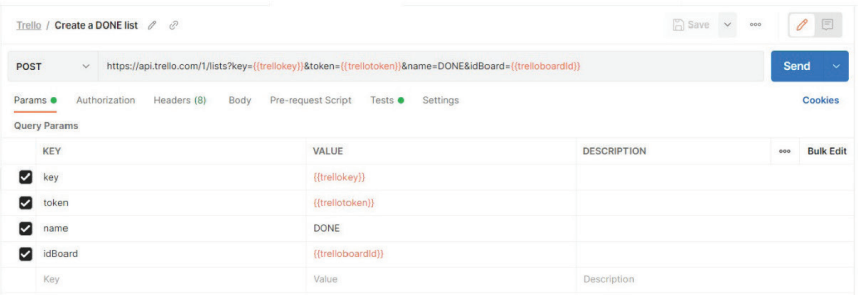


Gambar 4.68. Pembuatan Papan, Badan Uji Coba



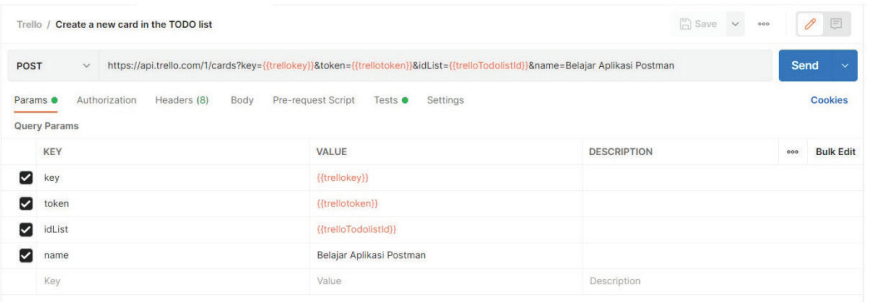
Gambar 4.69. Pembuatan List TODO, Parameter Wajib

Gambar 4.70. Pembuatan List TODO, Badan Uji Coba

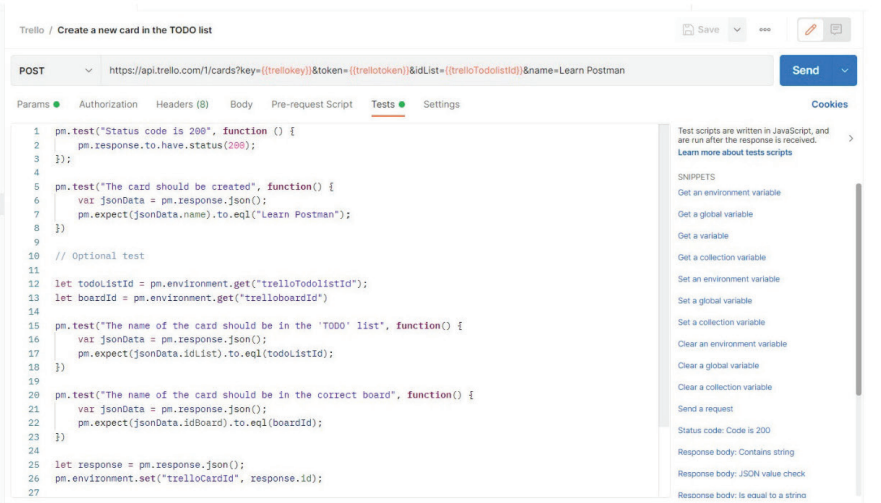


Gambar 4.71. Pembuatan List DONE, Parameter Wajib

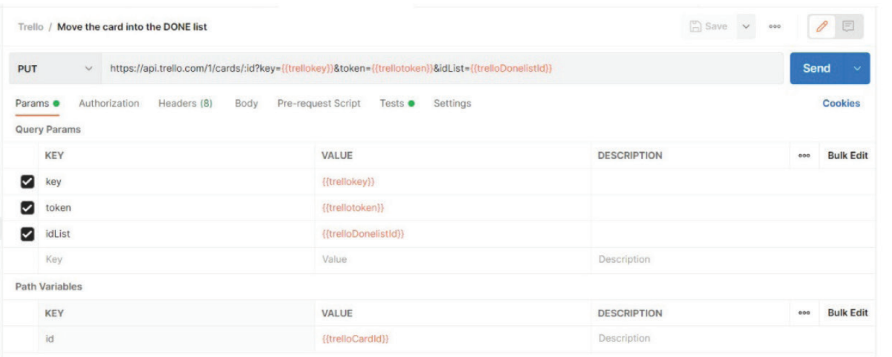
Gambar 4.72. Pembuatan List DONE, Badan Uji Coba



Gambar 4.73. Pembuatan Kartu, Parameter Wajib

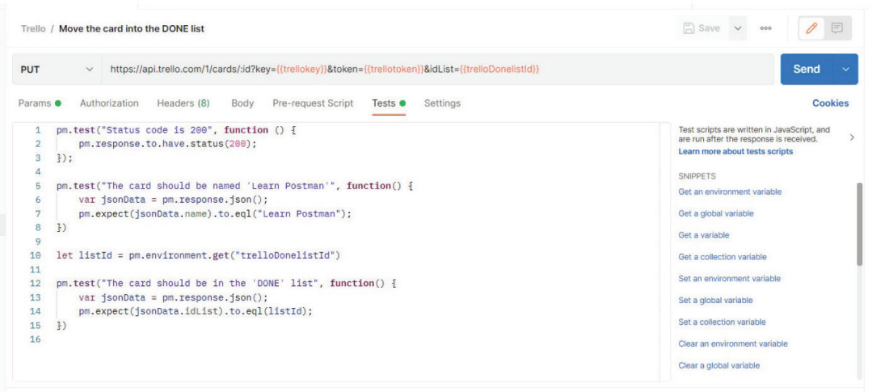


Gambar 4.74. Pembuatan Kartu, Badan Uji Coba

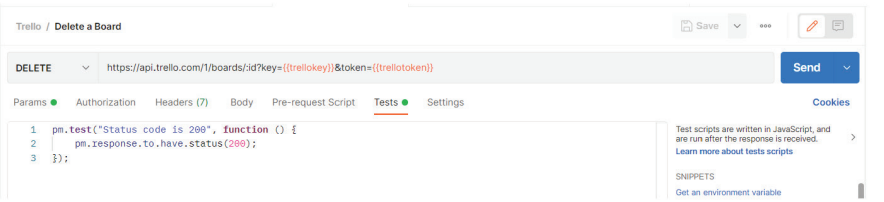


Gambar 4.75. Pemindahan Kartu, Parameter Wajib

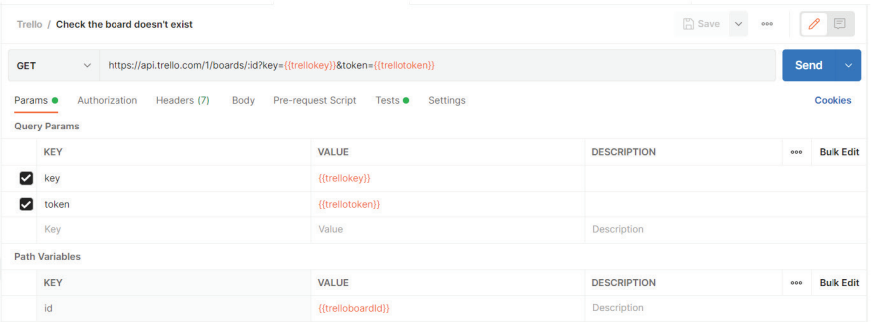
Gambar 4.76. Pemindahan Kartu, Badan Uji Coba



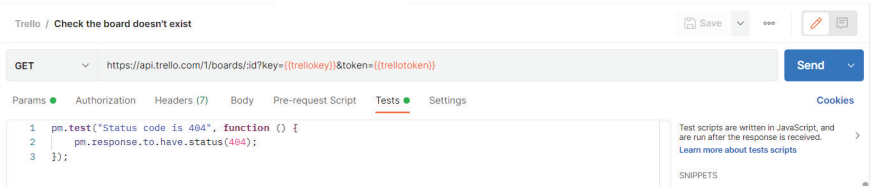
Gambar 4.77. Penghapusan Papan, Parameter Wajib



Gambar 4.78. Penghapusan Papan, Badan Uji Coba



Gambar 4.79. Pemeriksaan Papan, Parameter Wajib



Gambar 4.80. Pemeriksaan Papan, Badan Uji Coba

4.5. Otomatisasi Dengan Aplikasi Postman

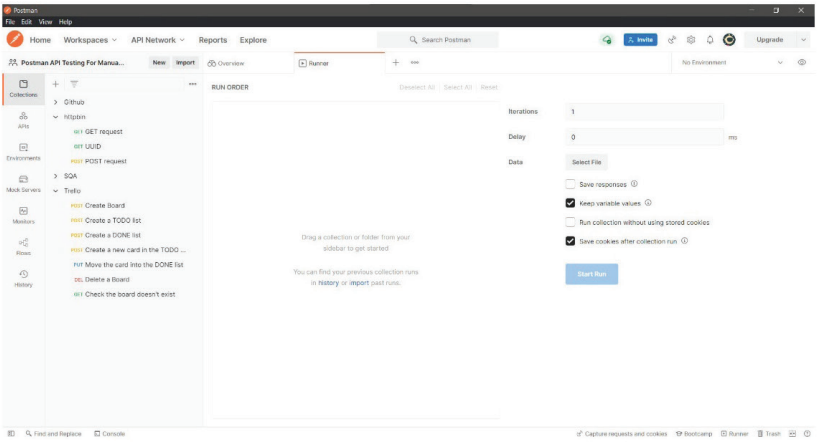
Uji coba permintaan API pada kenyataan di lapangan memiliki banyak sekali alamat permintaan API, terdapat ratusan bahkan ribuan alamat permintaan API dalam satu aplikasi. Jumlahnya yang begitu banyak membuat uji coba secara manual sangatlah tidak efektif dan dapat rentan terhadap kesalahan-kesalahan manusia. Otomasi uji coba menjadi solusi supaya uji coba dapat dilakukan dalam waktu yang lebih singkat dan mengurangi kesalahan-kesalahan yang dapat dibuat oleh manusia. Otomasi juga digunakan untuk menguji coba kembali permintaan-permintaan API yang sudah dibuat supaya dapat memastikan kesalahan lama tidak muncul kembali ketika fungsi atau fitur baru aplikasi diluncurkan.

Aplikasi Postman memiliki dua opsi untuk melakukan otomasi permintaan API. Pertama menggunakan 'Collection Runner' dan yang kedua adalah 'Postman Monitor'. 'Collection Runner' dapat digunakan langsung melalui aplikasi Postman dan memiliki langkah yang sangat sederhana. 'Postman Monitor' layaknya seperti 'Collection Runner' juga dapat digunakan langsung melalui aplikasi Postman. 'Postman Monitor' memiliki kegunaan yang berbeda dari 'Collection runner. Opsi uji coba

ini menggunakan interval waktu sehingga dijalankan pada waktu-waktu tertentu serta memiliki fitur berbayar dikarenakan opsi ini dapat menguji coba banyak koleksi dari permintaan API dan 'Collection Runner' hanya dapat menguji coba satu koleksi pada satu waktu. 'Postman Monitor' dapat digunakan untuk versi gratis, tetapi dibatasi dengan 1000 permintaan API setiap bulannya.

Berikut penggunaan 'Collection Runner' pada aplikasi Postman:

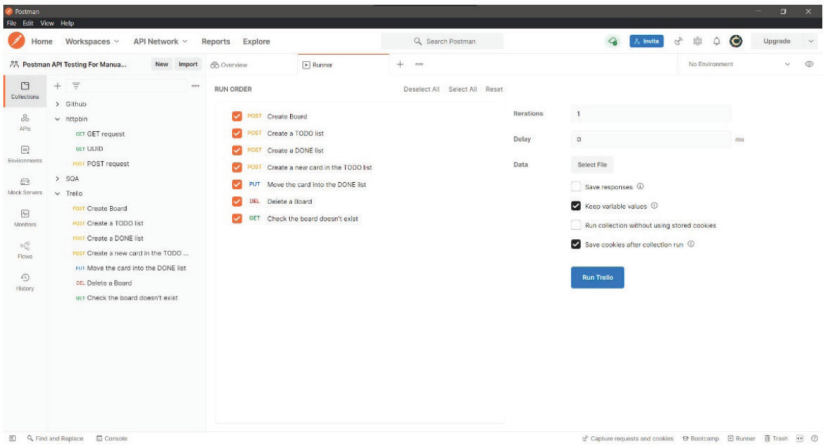
1. Pada bagian Menu aplikasi Postman, klik 'File' kemudian klik 'New Runner Tab'. Jika ingin menggunakan jalan pintas, tekan 'Control + Shift + R'.



Gambar 4.81. Tab Collection Runner

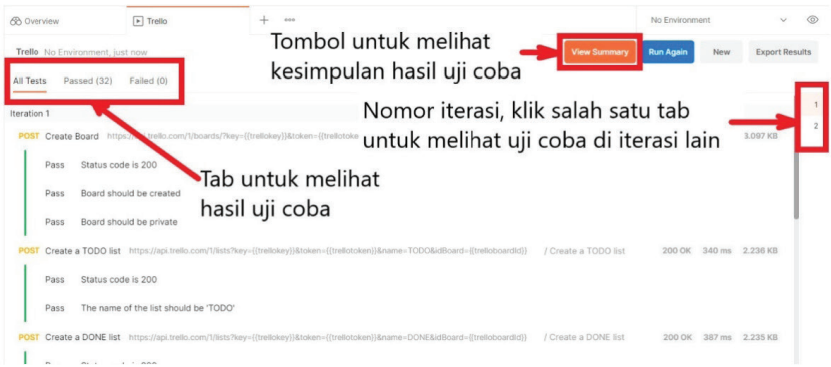
2. Tampilan tab otomasi dapat terlihat memiliki dua bagian, sebelah kiri menjadi tempat untuk mengumpulkan seluruh permintaan API yang akan diotomasi sedangkan sebelah kanan merupakan pemilihan opsi seberapa banyak iterasi dilakukan (Iteration), lama penundaan (Delay), sample data yang akan dipilih (Data, tombol Select File), opsi untuk menyimpan balasan (Save responses), opsi untuk menyimpan nilai-nilai variabel (Keep variable values), opsi untuk menjalankan otomasi tanpa kuki yang disimpan (Run collection without using stored cookies) dan opsi untuk menyimpan kuki setelah uji coba koleksi dijalankan (Save cookies after collection run).

3. Untuk memasukkan seluruh permintaan API dari latihan kedua sebelumnya, klik dan gerakkan koleksi tersebut ke bagian kiri tab 'Collection Runner'.



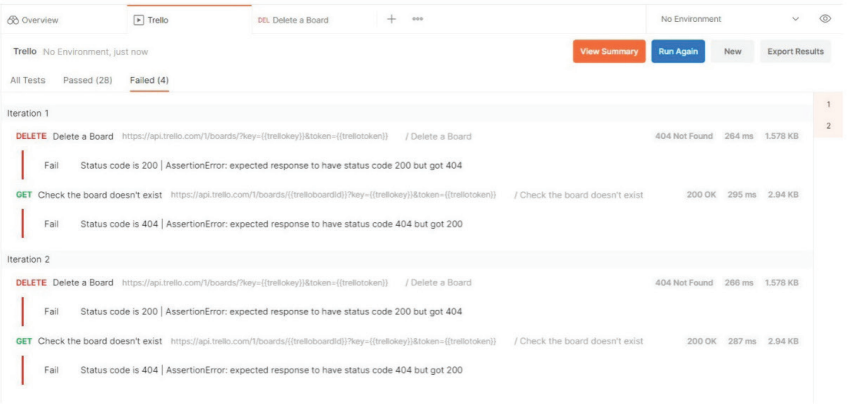
Gambar 4.82. Tab Collection Runner, Koleksi Terpilih

4. Untuk opsi pada bagian kanan tab, pilihlah untuk menyimpan balasan. Opsional, opsi seperti jumlah iterasi dan waktu tunda dapat dirubah jika diinginkan. Klik tombol 'Run <nama_koleksi>' untuk menjalankan uji coba otomasi.



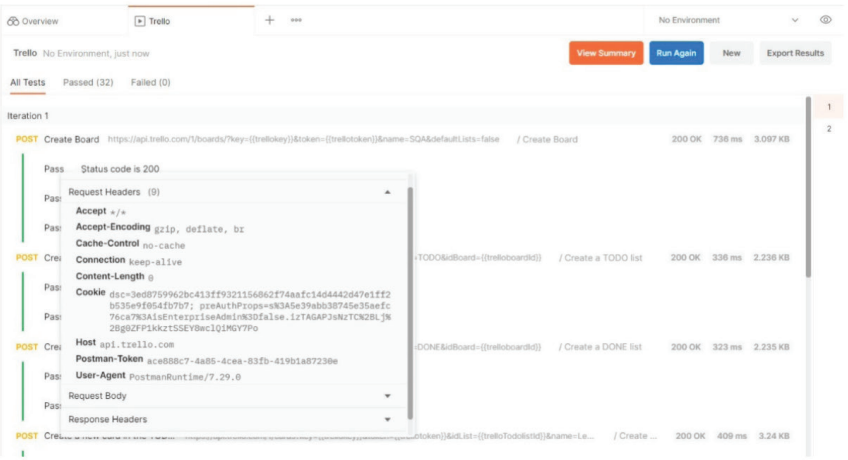
Gambar 4.82. Hasil Otomasi Uji Coba

5. Jika terdapat uji coba yang gagal, maka uji coba tersebut akan tertangkap dan dilaporkan didalam tab 'Failed'.



Gambar 4.83. Hasil Otomasi Uji Coba Yang Gagal

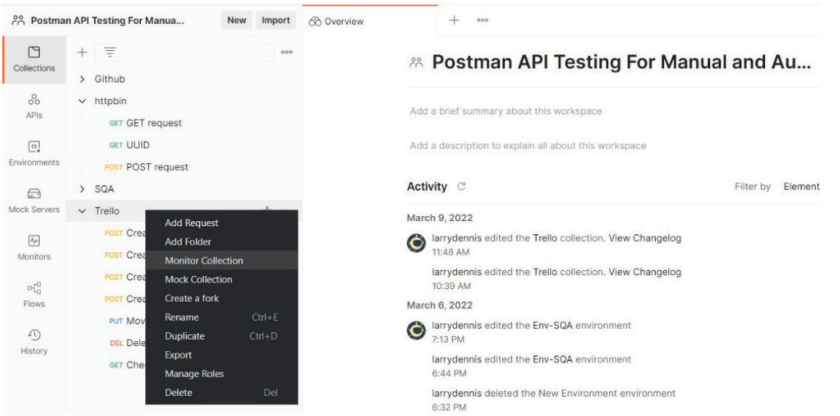
6. *Header* permintaan API juga dapat dilihat dengan cara mengklik nama permintaan API yang sudah di uji coba



Gambar 4.84. Header Iterasi

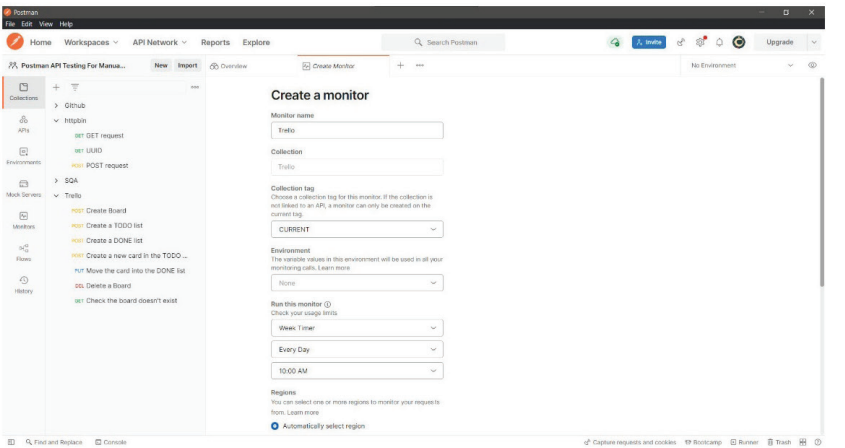
'Postman Monitor' pada aplikasi Postman memiliki tampilan yang berbeda dari 'Collection Runner'dikarenakan maksud dari penggunaan 'Postman Monitor' adalah uji coba pada waktu yang sudah ditentukan. Berikut cara penggunaan 'Postman Monitor':

1. Pada kumpulan koleksi, klik kanan pada nama koleksi tersebut, kemudian pilih 'Monitor Collection'.



Gambar 4.85. Opsi Koleksi Monitor

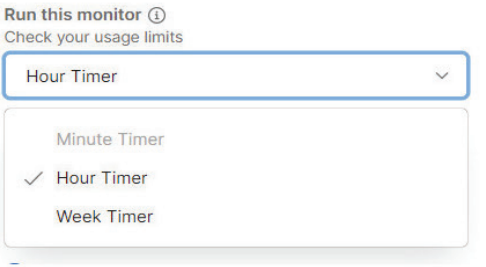
2. Tab baru bernama 'Create monitor' akan muncul beserta dengan nama monitor dari koleksi.



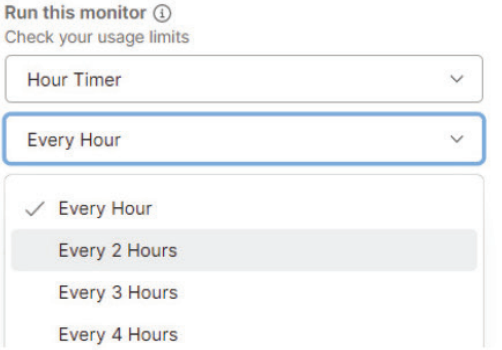
Gambar 4.86. Pembuatan Koleksi Monitor Tertampil

3. Nama monitor dapat dirubah sesuai dengan keinginan. Opsi tag koleksi hanya dapat dirubah jika koleksi tersebut terhubung dengan API lainnya. Opsi 'environment' dapat diganti dengan lingkup variabel lainnya, kita akan memilih lingkup yang dipakai saat latihan kedua (Postman Monitor tidak dapat mengimpor variabel global). Bagian 'Run this monitor' memiliki opsi-opsi yang berbeda. Opsi

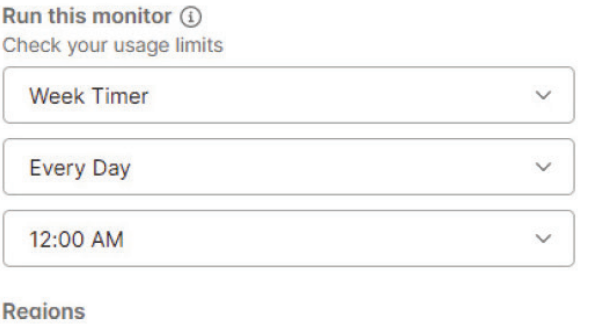
pertama menjadi kunci untuk membuat interval uji coba. Pilihan uji coba disediakan dalam bentuk setiap beberapa menit, setiap beberapa jam atau dalam satu minggu. Contohnya jika kita ingin menggunakan interval waktu dalam satu minggu, maka dalam opsi kedua dapat dipilih dalam satu minggu uji coba dijalankan pada hari-hari kerja atau hanya pada hari tertentu dengan jam tertentu. Opsi setiap beberapa jam akan memberikan opsi kedua berupa dalam setiap berapa jam uji coba dilakukan.



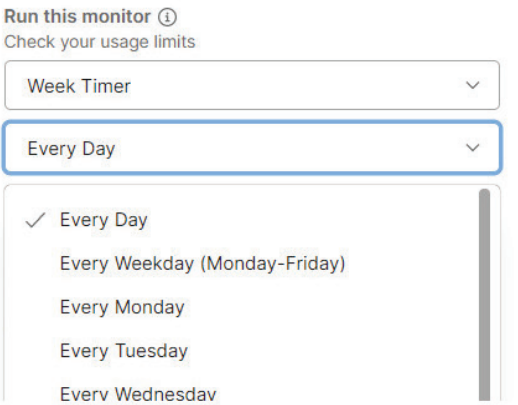
Gambar 4.87. Interval Uji Coba



Gambar 4.88. Interval Uji Coba, Waktu Perjam



Gambar 4.89. Interval Uji Coba, Waktu Perminggu

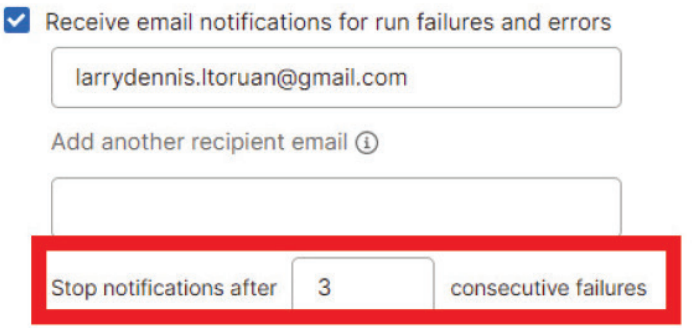


Gambar 4.90. Interval Uji Coba, Waktu Perminggu, Pengaturan Harian

4. Untuk latihan uji coba ini, kita akan memilih interval waktu dilakukan setiap jam dalam sehari.Kemudian gulir kebawah untuk melihat opsi lainnya.

5. Bagian 'Regions' opsi otomatis menjadi pilihan utama dikarenakan penggunaan layanan gratis.

6. Notifikasi email dapat tidak dipakai atau ditambahkan untuk mengirimkan laporan hasil uji coba. Didalam opsi tersebut terdapat pembatas pengiriman notifikasi jika terdapat sejumlah kegagalan uji coba berturut-turut.



Gambar 4.91. Notifikasi Email, Kegagalan Berturut-turut

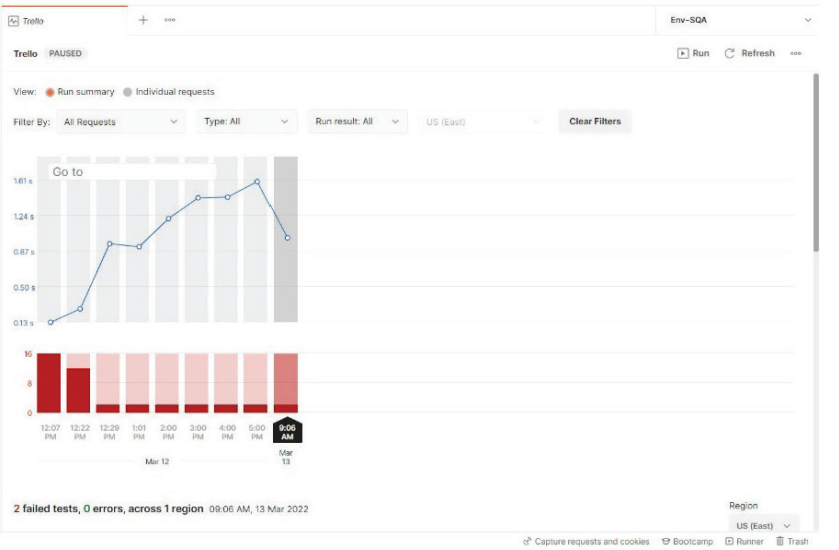
7. Opsi 'Set request timeout' akan membuat monitor uji coba tidak dapat dilakukan lebih dari lama waktu yang sudah ditentukan, misalnya selama lima menit (300,000 mili detik) maka monitor uji coba akan

berhenti berjalan jika permintaan yang dijalankan lebih dari lima menit.

8. Opsi 'Set delay between requests' akan membuat setiap uji coba diundur dalam selang waktu tertentu, misalnya satu setengah detik (1,500 mili detik) maka setiap uji coba baru akan berjalan setelah satu setengah detik berlalu.

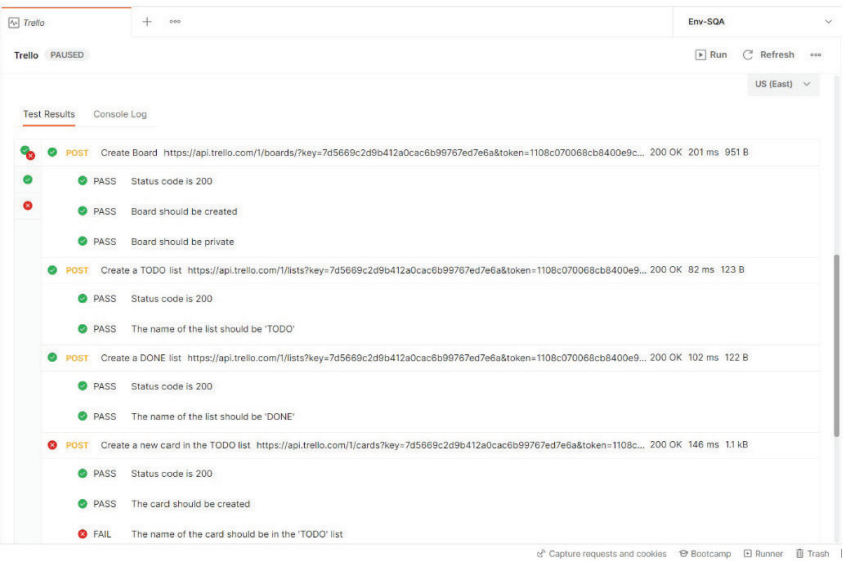
9. Klik tombol 'Create Monitor' untuk membuat monitor uji coba

10. Dikarenakan interval waktu yang dipilih adalah setiap jam, maka kita baru dapat hasilnya setelah satu jam berlalu. Monitor ini juga dapat diedit dengan cara mengklik tiga titik yang berada didalam tab monitor tersebut.Klik tombol 'Run' untuk langsung menjalankan monitor uji coba

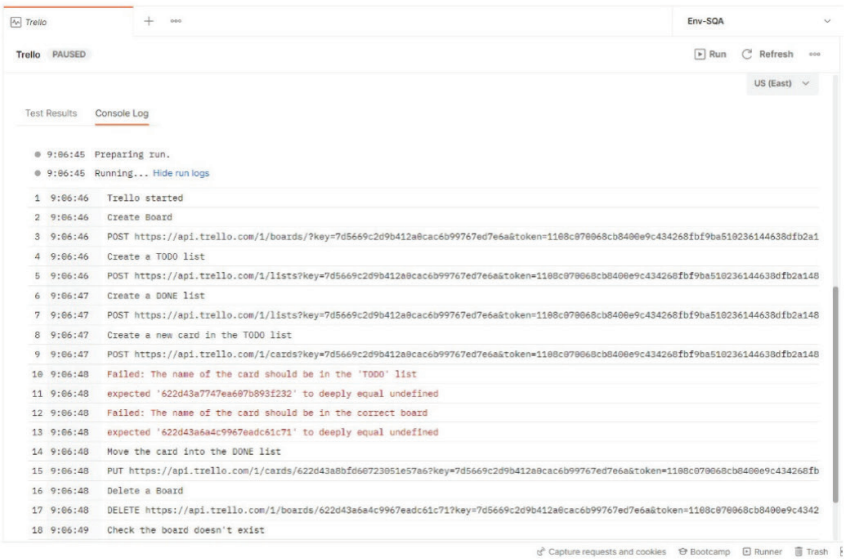


Gambar 4.92. Hasil Uji Coba Menggunakan Postman Monitor

11. Hasil dari keseluruhan uji coba dapat dilihat dengan cara menggulir kebawah. Terdapat sub tab 'Test Results' untuk dapat melihat hasil uji coba yang sudah dibuat dan sub tab 'Console Log' untuk dapat melihat lebih detail uji coba yang sudah dilakukan.

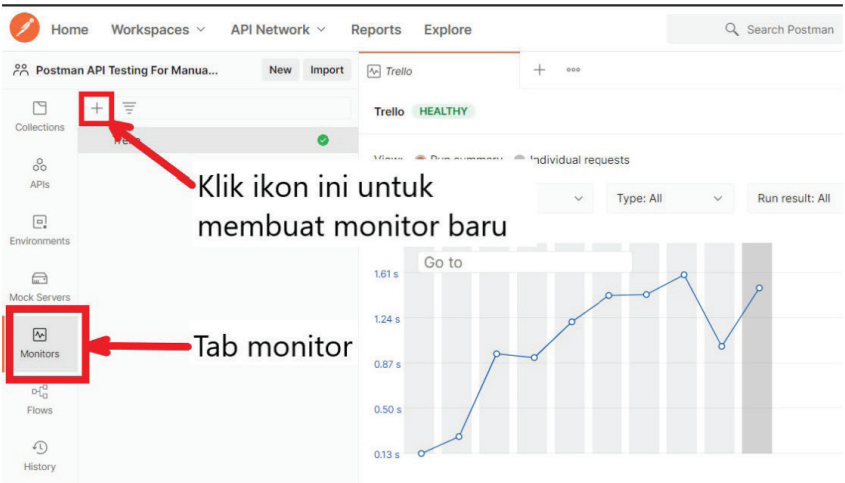


Gambar 4.93. Hasil Badan Uji Coba Monitor



Gambar 4.94. Log Konsol Hasil Uji Coba Monitor

12. Seluruh pembuatan monitor dapat dilihat dengan cara mengklik tab 'Monitors' disamping kiri aplikasi. Pembuatan monitor baru juga dapat dilakukan melalui tab tersebut.

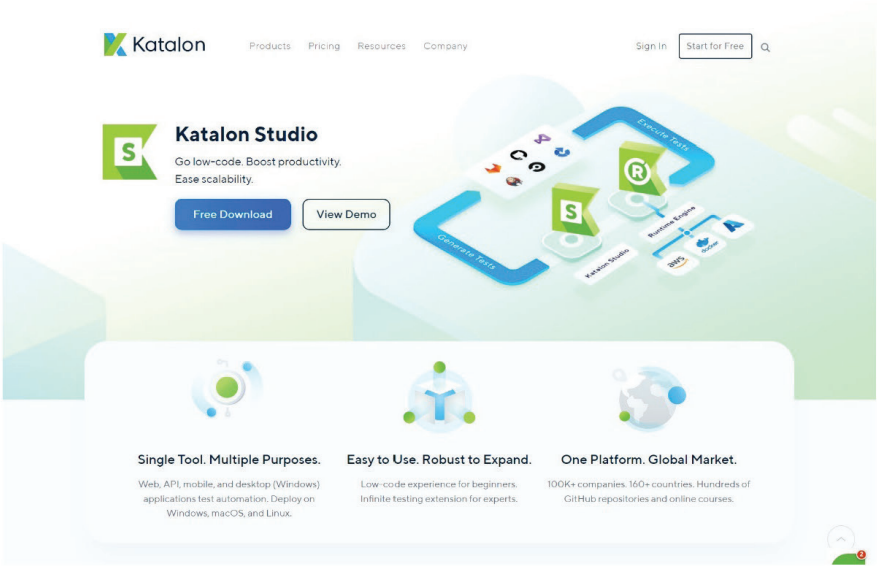


Gambar 4.95. Tab Monitor

Otomasi yang dijalankan dengan 'Postman Monitor' dilakukan melalui server yang sudah disediakan oleh perusahaan Postman sehingga terdapat perbedaan lokasi server uji coba dengan server tempat dimana aplikasi web yang akan diuji coba kan terjadi. Perlu diingat, terdapat peraturan-peraturan ataupun ijin yang mungkin dibutuhkan jika menginginkan server diluar jangkauan server produksi dapat diuji cobakan melalui server lain.

4.6. Pengenalan Aplikasi Katalon

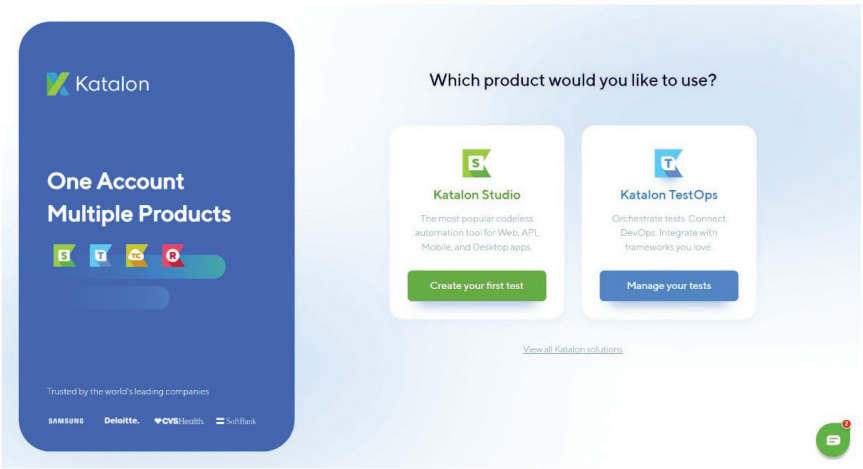
Katalon Studio adalah sebuah IDE (Integrated Development Environment) khusus untuk penggunaan uji coba secara otomasi. Aplikasi Katalon Studio dapat digunakan tanpa harus membuat sebuah kode program untuk menjalankan uji coba otomasi dikarenakan penggunaannya yang penuh dengan GUI (Graphical User Interface) sehingga pemula dapat menggunakannya dengan sangat mudah. Uji coba dengan aplikasi Katalon Studio dapat dilakukan diberbagai platform seperti web, *mobile*, desktop dan API.



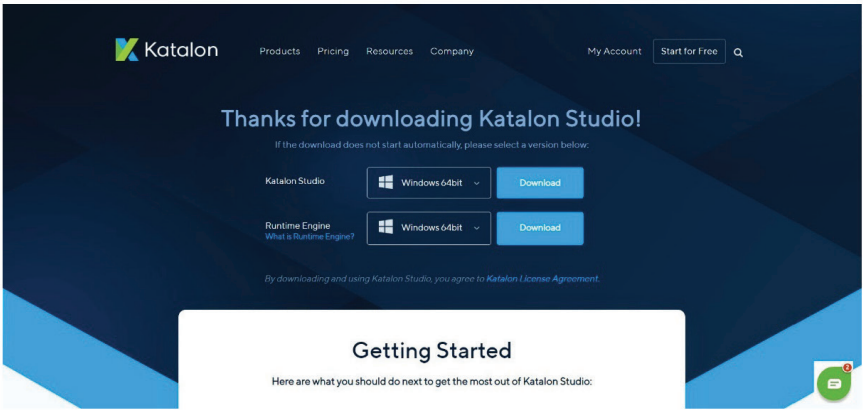
Gambar 4.96. Web Utama Katalon Studio (https://katalon.com/katalon-studio)

Aplikasi Katalon Studio sendiri memiliki fitur-fitur berbayar untuk penggunaan uji coba skala besar. Akan tetapi, versi gratis masih disediakan jika penggunaannya dalam skala kecil. Supaya dapat menggunakan aplikasi Katalon Studio, registrasi dengan membuat akun terlebih dahulu melalui website https://katalon.com/sign-up.

Jika registrasi sudah selesai, terdapat dua aplikasi yang disediakan oleh Katalon. Aplikasi Katalon Studio yang dipilih untuk penggunaan otomasi uji coba.Tergantung sistem operasi yang akan dipakai, Katalon Studio tersedia untuk sistem operasi Windows (64-bit dan 32-bit), macOS dan sistem operasi berbasis kernel Linux.

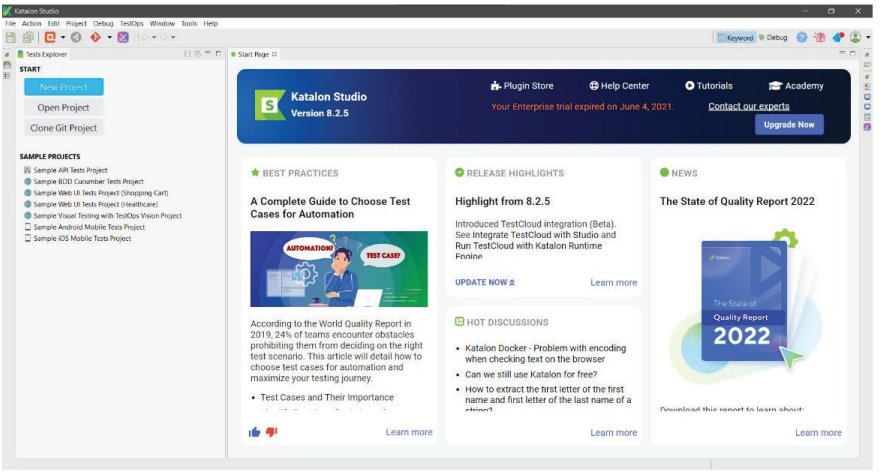


Gambar 4.97. Produk-Produk Katalon



Gambar 4.98. Unduh Katalon Studio

Aplikasi Katalon Studio kemudian dapat langsung digunakan tanpa instalasi. Pastikan aplikasi dijalankan sebagai pihak administrator (pengguna super).

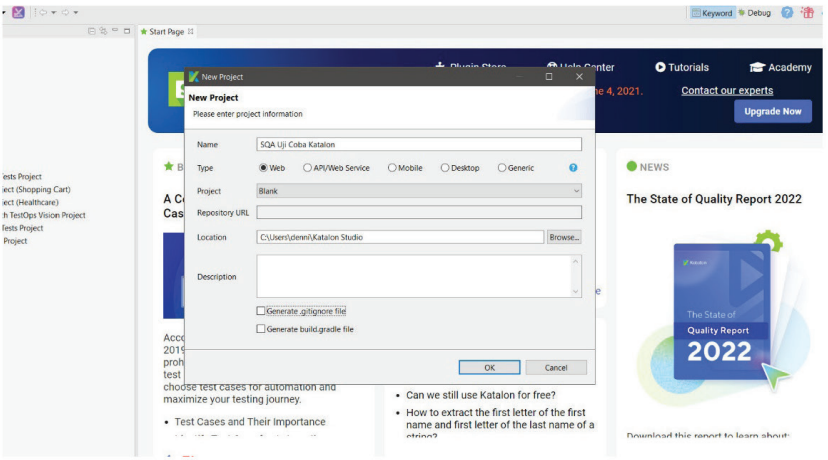


Gambar 4.98. Tampilan Utama Aplikasi Katalon Studio

4.7. Otomatisasi Dengan Aplikasi Katalon

Otomasi dengan aplikasi Katalon Studio secara sederhana dapat dilakukan dengan mudah. Secara garis besar, aplikasi Katalon Studio akan merekam setiap aksi manual yang akan dilakukan saat uji coba melalui antarmuka aplikasi (contoh registrasi atau login akun melalui sebuah situs) sehingga aplikasi Katalon Studio dapat mereplika langkah-langkah uji coba. Kemampuan mereplika ini tidak semua dapat dilaksanakan karena tergantung kompleksitas komponen aplikasi, kemungkinan perekaman bisa gagal.Latihan sederhana dari otomatisasi dengan aplikasi Katalon akan menggunakan website demo yang sudah disediakan oleh Katalon. Latihan otomasi ini juga dapat dilihat langsung melalui blog yang sudah diposting oleh Katalon [25]. Berikut langkah-langkah uji coba sederhana dengan aplikasi Katalon Studio :

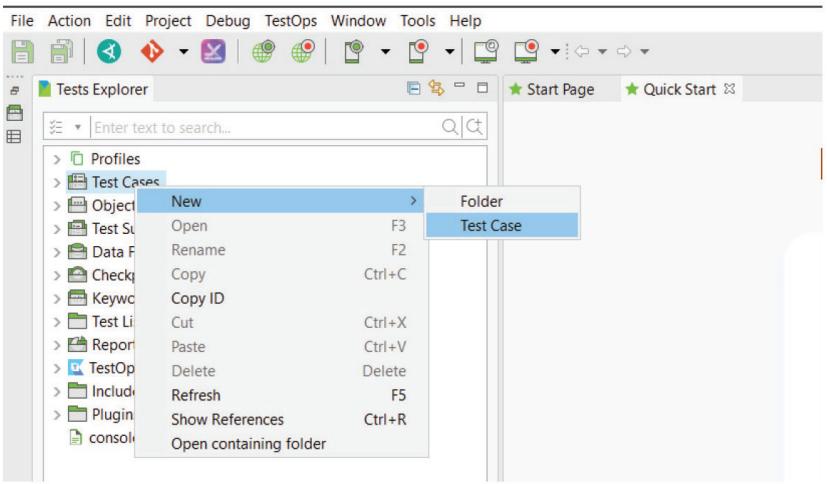
1. Pada halaman utama aplikasi Katalon Studio, klik tombol 'New Project'



Gambar 4.99. Pembuatan Proyek Baru

2. Tampilan jendela untuk membuat proyek baru akan muncul. Isi nama proyek dengan 'SQA Uji Coba Katalon'. Tipe uji coba adalah 'Web'. Lokasi direktori proyek dapat dipilih bebas dengan mengklik 'Browse...' dan deskripsi dapat dibiarkan kosong. Klik tombol 'OK' untuk membuat proyek

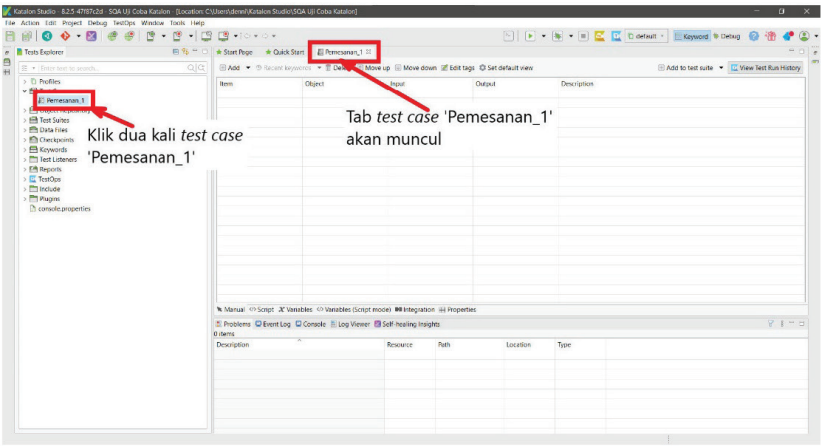
3. Pada bagian kiri tab 'Tests Explorer', klik kanan pada subtab 'Test Cases'. Kemudian pilih 'New' dan pilih 'Test Case'



Gambar 4.100. Membuat Test Case

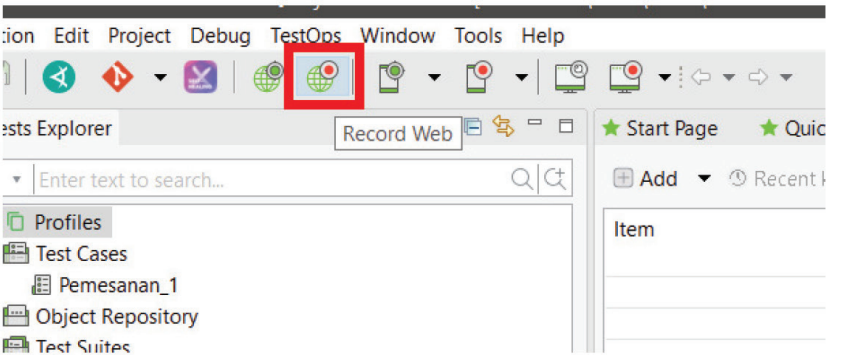
4. Isilah nama *test case* dengan 'Pemesanan_1'. Klik 'OK' untuk membuat *test case*

5. Sebuah test case kosong akan dibuat. Klik dua kali pada test case tersebut yang muncul didalam subtab 'Test Cases', akan muncul isi detail test case di ruang kerja utama.



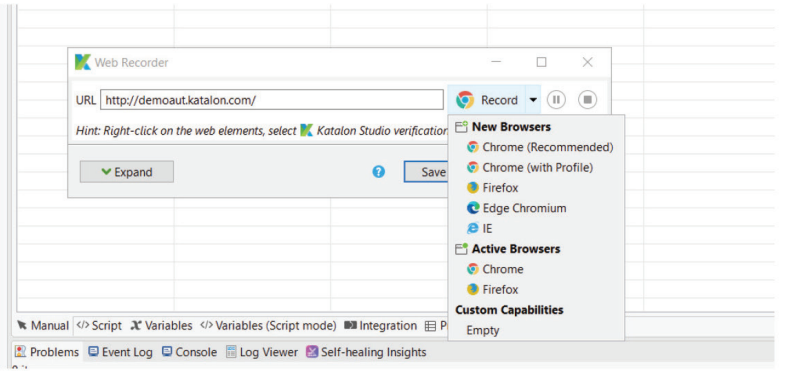
Gambar 4.101. Test Case Baru Tertampil

6. Untuk menangkap seluruh uji coba manual yang akan dilakukan, klik tombol 'Record Web'.



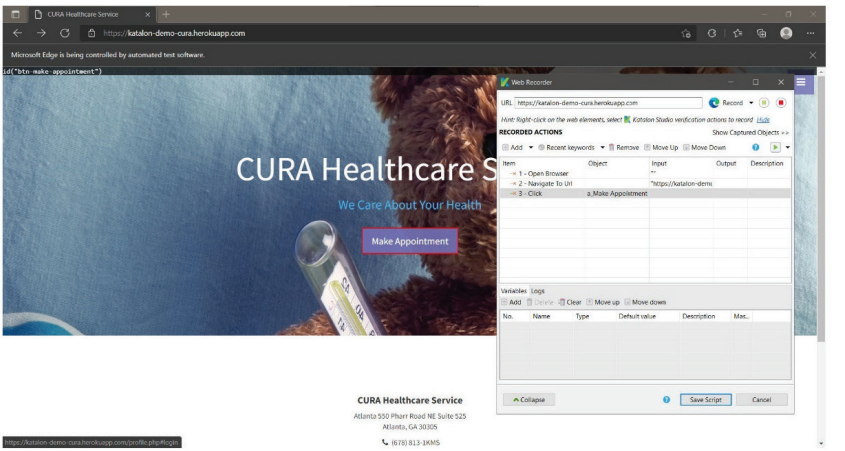
Gambar 4.102. Tombol Record Web

7. Jendela 'Web Recorder' akan muncul.Jenis browser yang akan digunakan dapat dipilih sesuai dengan browser web yang tersedia. Isi alamat url dengan https://katalon-demo-cura.herokuapp.com/. Klik 'Record' untuk memulai penangkapan. Jendela baru dari browser yang dipilih akan muncul dengan alamat url yang sudah dimasukkan.



Gambar 4.103. Tampilan Penangkapan Baru

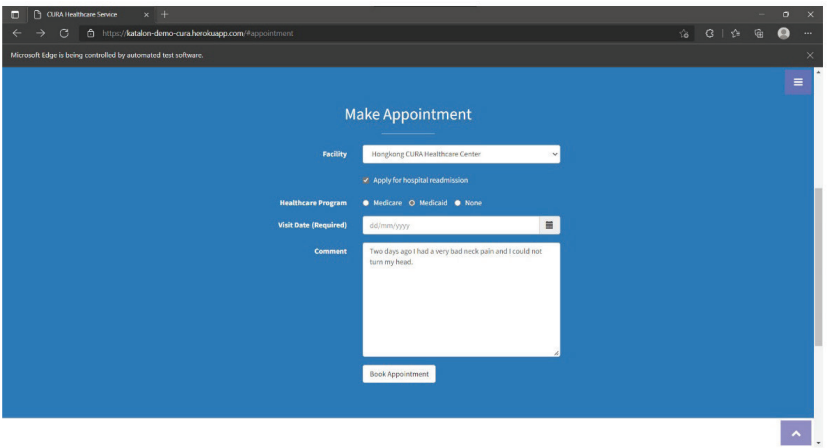
8. Pada jendela 'Web Recorder', klik tombol 'Expand' untuk dapat melihat rekaman-rekaman elemen yang sudah didapatkan. Pada halaman situs yang akan diuji cobakan, klik tombol 'Make Appointment'. Aksi tersebut kemudian akan terekam dan tertampil di kolom 'Item'



Gambar 4.104. Web Recorder, Aksi yang Terekam

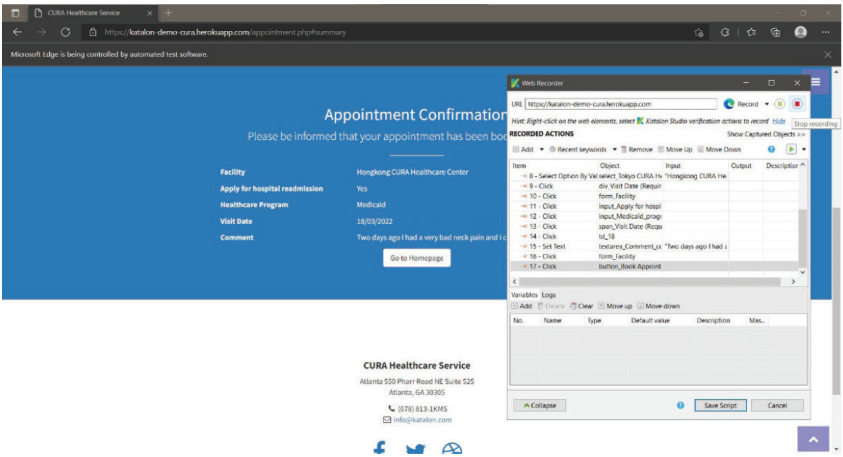
9. Tombol tersebut akan membawa ke halaman login. Isi kotak input username dengan nama akun demo 'John Doe' dan kotak input password 'ThisIsNotAPassword'. Perhatikan pada bagian 'Recorded Actions', sebuah item baru akan muncul dengan nama 'Set Text' yang menandakan aksi penginputan data juga akan direkam. Kemudian klik tombol 'Login'

10. Jika login berhasil, maka akan dibawa ke halaman untuk membuat janji temu. Pilih 'Facility' bernama 'Hongkong CURA Healthcare Center', kemudian pilih 'Apply for hospital readmission' serta 'Medicaid' sebagai 'Healthcare Program'. Isi tanggal 'Visit Date' secara manual dengan tanggal tiga hari setelah hari ini dan isi 'Comment' dengan pesan "Two days ago I had a very bad neck pain and I could not turn my head.". Kemudian klik tombol 'Book Appointment'.

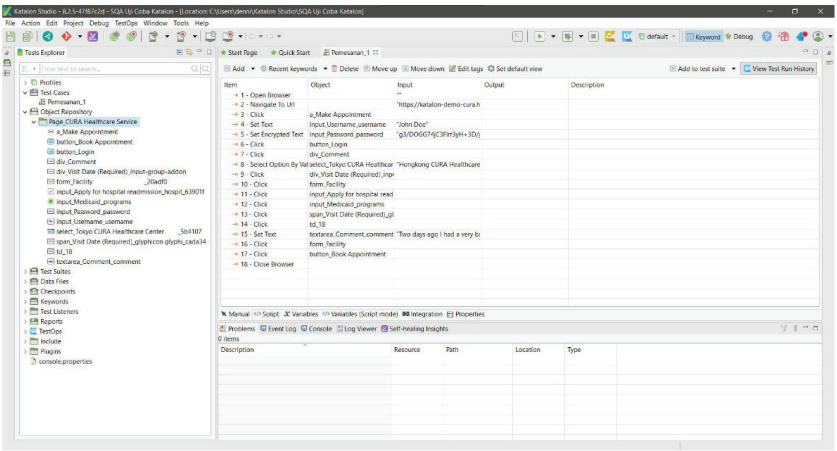


Gambar 4.105. Pembuatan Janji Temu

11. Konfirmasi janji temu kemudian ditampilkan sebagai bentuk bahwa janji temu sudah dibuat. Pada jendela 'Web Recorder', klik 'Stop Recording' untuk menghentikan penangkapan. Kemudian klik tombol 'Save Script' kemudian klik 'OK' untuk menyimpan seluruh aksi pembuatan *test case*. Seluruh obyek penangkapan kemudian akan tersimpan dalam direktori 'Page CURA Healthcare Service' didalam subtab 'Object Repository'.

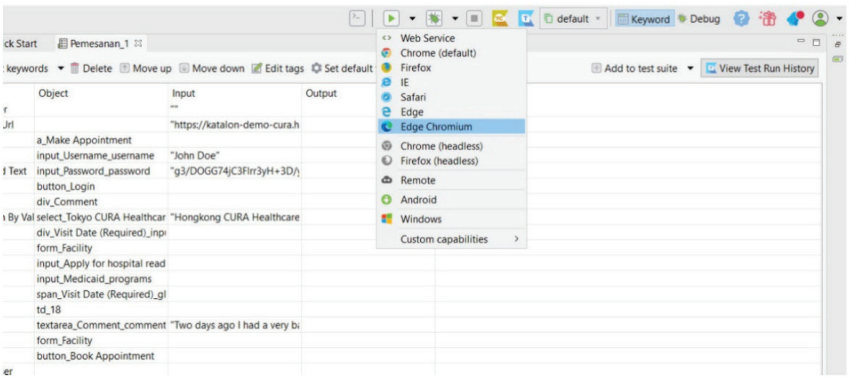


Gambar 4.106. Pembuatan Janji Temu Berhasil

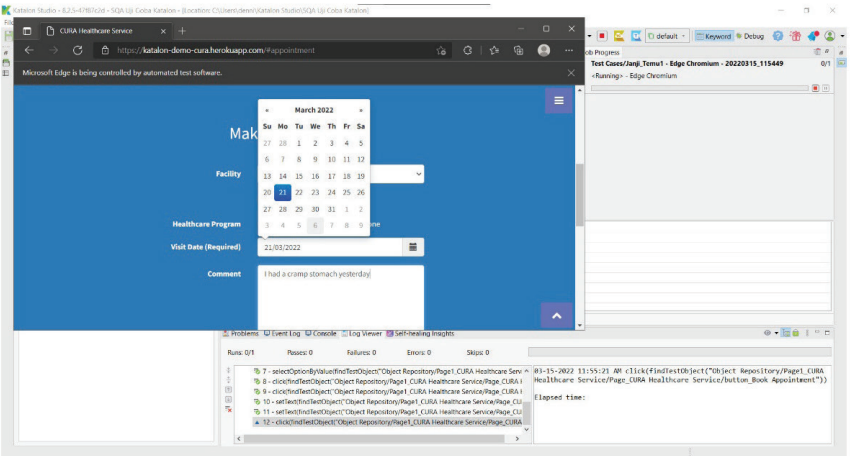


Gambar 4.107. Obyek Tersimpan, Test Case Otomatis Terbuat

12. Klik tombol 'Run' untuk menjalankan seluruh *test case* yang sudah dibuat secara langsung, browser web juga dapat dipilih langsung saat menjalankan uji coba.



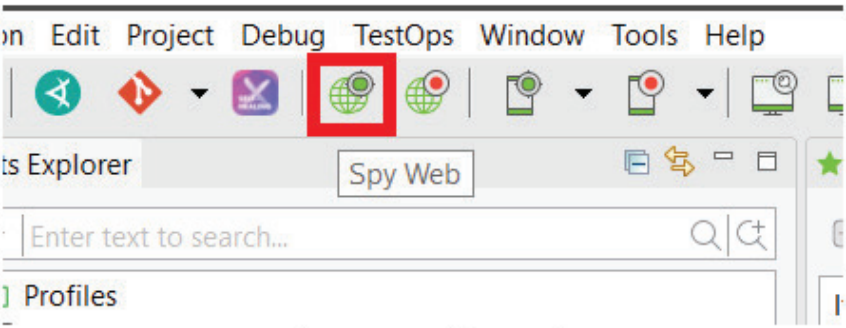
Gambar 4.107. Menjalankan Otomasi Uji Coba



Gambar 4.108. Tampilan Otomasi Uji Coba Menyeluruh

Aplikasi Katalon Studio juga menyediakan cara penangkapan yang lebih baik dari menangkap secara manual seluruh aksi uji coba yang sudah dilakukan. Tombol 'Spy Web' dapat menangkap satu obyek komponen didalam web sehingga seluruh pilihan-pilihan yang diberikan dapat dirubah nanti tanpa harus melakukan uji coba yang sama secara keseluruhan. Berikut cara menangkap dengan 'Spy Web' :

1. Klik tombol 'Spy Web', jendela 'Spy Web' akan muncul. Pilih browser web yang diinginkan untuk memulai penangkapan.



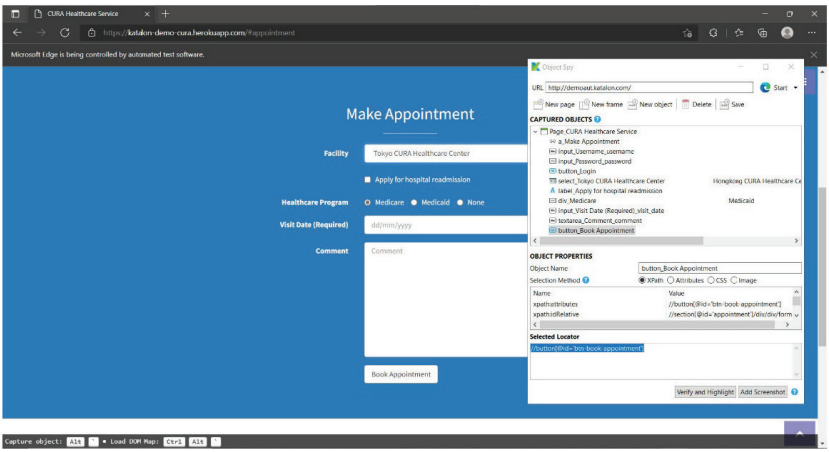
Gambar 4.109. Tombol Spy Web

2. Jendela browser web baru akan muncul dengan tampilan opsi yang sedikit berbeda dari 'Record Web'. Obyek-obyek dari situs dapat ditangkap dengan cara menggerakkan pointer ke arah obyek tersebut, kemudian tekan Alt+*back qoute*. Garis bewarna hijau yang mengelilingi obyek tersebut akan muncul yang menandakan obyek tersebut berhasil ditangkap dan pada bagian 'Captured Objects' terdapat nama obyek tersebut.

Gambar 4.110. Spy Web, Obyek Tertangkap

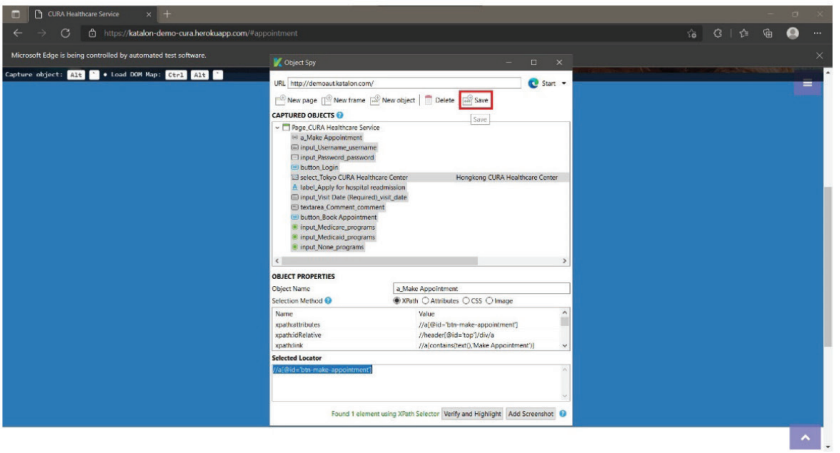
3. Masuk ke halaman login, kota input username dan password serta tombol login menjadi target tangkapan.

4. Pada halaman janji temu, obyek tombol dropdown, kotak centang, tombol radio, kotak input komentar dan tombol pesan janji temu menjadi target tangkapan selanjutnya.

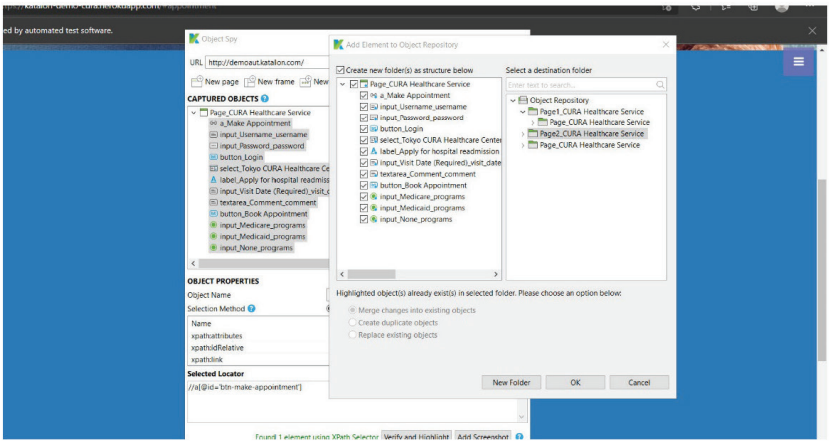


Gambar 4.111. Spy Web, Janji Temu, Obyek Tertangkap

5. Tekan dan sorot obyek-obyek yang sudah tertangkap, kemudian simpan dengan mengklik tombol 'Save'.



Gambar 4.112. Spy Web, Soroti Dan Simpan

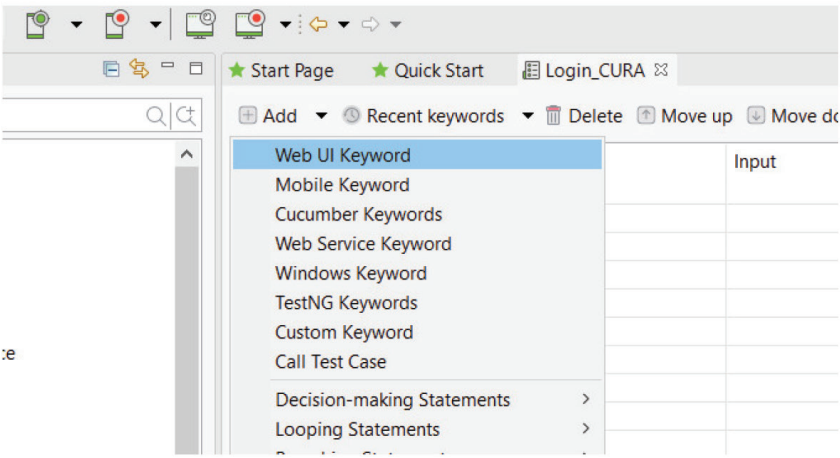


Gambar 4.112. Repositori Obyek

6. Seluruh obyek akan disimpan dalam subtab 'Object Repository'

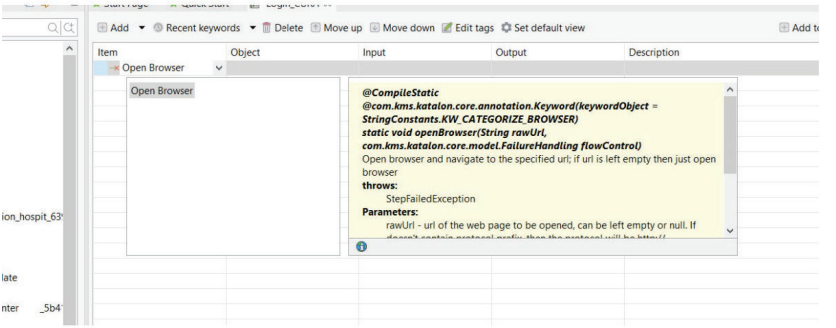
7. Buat *test case* baru dengan nama bebas dengan cara klik kanan pada sub tab 'Test Cases' kemudian klik 'New' dan klik 'Test Case'

8. Pada tab *test case* baru tersebut, klik tombol 'Add', kemudian pilih 'Web UI Keyword'.



Gambar 4.113. Pembuatan Test Case, Plihan Web UI Keyword

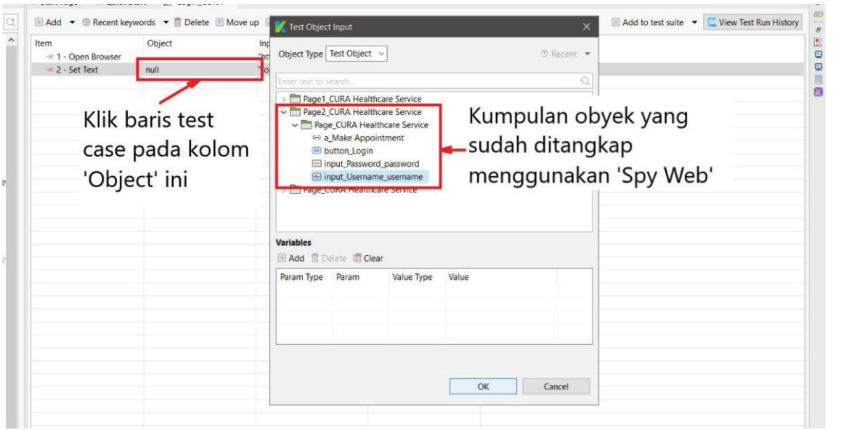
9. Masukkan tipe *test case* bernama 'Open Browser'



Gambar 4.114. Tipe Test Case, Open Browser

10. Pada kolom 'Input' klik baris dari *test case* 'Open Browser', akan muncul jendela 'Input'. Nilai dapat dimasukkan dengan alamat url https://katalon-demo-cura.herokuapp.com/, kemudian klik 'OK'.

11. Lakukan pembuatan *test case* manual tersebut untuk dipasangkan obyek-obyek lainnya. Beberapa *test case* memiliki kolom 'Object' yang dapat dipasangkan dengan obyek yang sudah ditangkap.



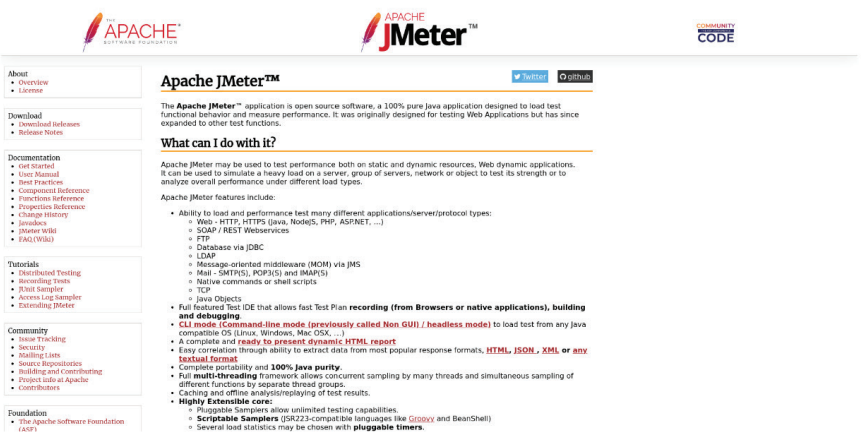
Gambar 4.115. Pemasangan Obyek Tangkapan Manual

4.8. Pengenalan Aplikasi Jmeter

Apache JMeter adalah alat open-source berbasis Java yang digunakan untuk menguji kinerja dan beban aplikasi perangkat lunak. JMeter awalnya dirancang untuk menguji aplikasi web tetapi sejak itu telah diperluas untuk mencakup berbagai jenis layanan dan protokol. Beberapa fitur utama JMeter meliputi:

1. Testing Load and Performance: JMeter dapat digunakan untuk melakukan tes beban dan kinerja pada aplikasi web untuk melihat bagaimana mereka berperforma di bawah berbagai kondisi beban.
2. Multiple Protocols: JMeter mendukung berbagai protokol seperti HTTP, HTTPS, FTP, LDAP, JDBC, SOAP, dan banyak lagi.
3. GUI Mode: JMeter memiliki antarmuka pengguna grafis (GUI) yang memungkinkan pengguna untuk membuat, mengkonfigurasi, dan menjalankan tes.
4. Non-GUI Mode: JMeter juga dapat dijalankan dalam mode non-GUI yang cocok untuk pengujian otomatisasi dan berkelanjutan.
5. Extensible: JMeter dapat diperluas dengan menambahkan plug-in untuk mendukung fungsionalitas tambahan.
6. Reporting: JMeter menyediakan laporan dan grafik yang komprehensif untuk menganalisis hasil pengujian.

Dengan menggunakan JMeter, pengembang dan penguji dapat mensimulasikan beban pengguna dan mengidentifikasi potensi bottleneck dan masalah kinerja dalam aplikasi mereka sebelum dirilis ke lingkungan produksi. Aplikasi JMeter dapat diunduh dari situs resmi Apache JMeter di https://jmeter.apache.org/.

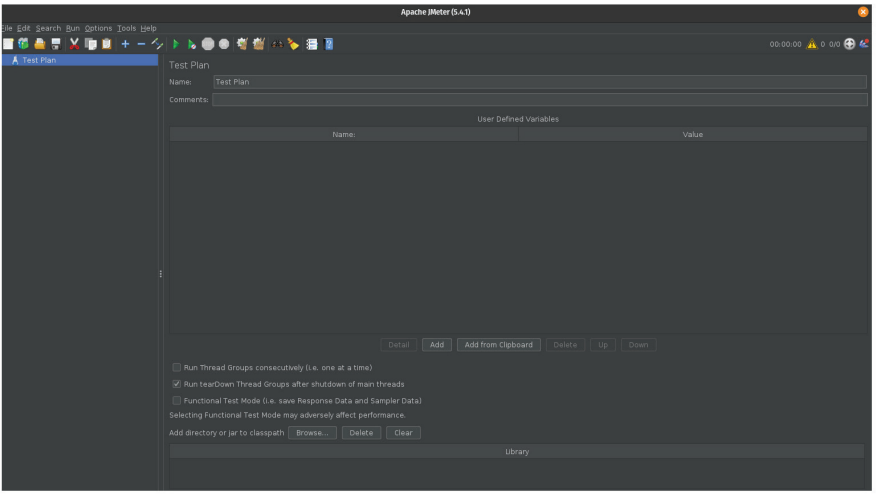


Gambar 4.116. Situs Resmi Jmeter

Berikut adalah langkah-langkah untuk mengunduh JMeter:

1. Buka browser web dan kunjungi situs resmi Apache JMeter di https://jmeter.apache.org/.
2. Di halaman utama, klik tautan "Download Releases" di menu navigasi.
3. Anda akan diarahkan ke halaman unduhan, di mana Anda dapat memilih versi JMeter yang ingin diunduh. Biasanya, Anda ingin mengunduh versi stabil terbaru yang direkomendasikan.
4. Klik tautan unduhan yang sesuai dengan sistem operasi Anda (Windows, Mac, atau Linux).
5. File yang diunduh biasanya berupa arsip zip atau tar.gz. Setelah diunduh, ekstrak file tersebut ke direktori pilihan Anda.
6. Setelah diekstraksi, Anda dapat menjalankan JMeter dengan membuka direktori yang telah diekstrak dan menjalankan file jmeter.bat (untuk Windows) atau jmeter (untuk Linux/Mac).

Pastikan Anda telah menginstal Java di sistem Anda, karena JMeter memerlukan Java untuk berjalan. Anda dapat mengunduh Java dari situs resmi Oracle atau menggunakan distribusi open-source seperti OpenJDK.



Gambar 4.117. Halaman Kerja Utama Aplikasi Jmeter

4.9 Pengenalan Test Plan pada Jmeter

Bab ini membahas berbagai komponen JMeter dan membantu mengembangkan skrip pengujian JMeter pertama. Pada bagian ini akan dijelaskan cara untuk menjalankan dan menghentikan/menutup pengujian JMeter. Sebelum kita mulai mengembangkan skrip pengujian JMeter, mari tinjau komponen rencana pengujian (Test Plan).

Skrip pengujian JMeter terdiri dari bagian-bagian yang terurut dan hierarkis yang disusun dalam struktur pohon.

Test Plan adalah komponen terpenting dan dasar dari pohon. Test Plan adalah tempat mengonfigurasi nama, deskripsi, dan variabel pengguna untuk rencana pengujian.

Ada satu atau lebih grup thread di setiap pengujian. Anak dari test plan adalah Thread Group. Kasus penggunaan diwakili oleh setiap thread group. Kumpulan thread dapat ditentukan dengan jumlah thread, waktu ramp-up, dan atribut berguna lainnya yang memungkinkan pengelolaan perilakunya.

Ada satu atau lebih komponen Controller di setiap kumpulan thread. Alur eksekusi ditentukan oleh controller logika.Controller memilih Sampler mana yang akan dijalankan selanjutnya. Banyak driver logika

yang sudah diinstal sebelumnya di JMeter menawarkan aliran kontrol yang presisi. Misalnya, percabangan disediakan oleh controller If dan Switch, dan aliran iterasi disediakan oleh pengontrol ForEach, While, dan For. Setiap konstruksi pemrograman memiliki controller. Jika perlu, controller baru dapat dibuat menggunakan pendekatan JMeter Plugin API.

Sampler adalah sub-elemen dari controller atau thread group. Ini menghubungi server dengan permintaan. Setiap protokol memerlukan sampler yang berbeda. JMeter menyertakan beberapa sampler siap pakai. Misalnya, sampel permintaan HTTP digunakan untuk mengirim permintaan HTTP. Metode plugin JMeter dapat digunakan untuk membuat sampel khusus.

Saat Test Plan berjalan mengkompilasi dan mengumpulkan data kinerja, Listener mengasimilasi respons server. Listener digunakan untuk menampilkan grafik. Untuk menafsirkan dan memahami hasil tes kinerja, setiap skrip tes memerlukan minimal satu listener.

4.10. Membuat Test Plan

Setelah JMeter terbuka, pada menu utama, klik File > New untuk membuat Test Plan baru.

Tambahkan Thread Group:

1. Klik kanan pada Test Plan > Add > Threads (Users) > Thread Group.
2. Thread Group mengatur jumlah pengguna (threads) dan pengaturan beban lainnya.
3. Atur Number of Threads (jumlah pengguna virtual), Ramp-Up Period (waktu untuk memulai semua pengguna), dan Loop Count (jumlah pengulangan).

Tambahkan Sampler HTTP Request:

1. Klik kanan pada Thread Group > Add > Sampler > HTTP Request.

2. Di sini konfigurasi permintaan HTTP yang ingin diuji.
3. Isi Server Name or IP dengan URL target (misalnya, example.com).
4. Isi Path dengan jalur spesifik dari permintaan (misalnya, /index.html).

Tambahkan Listener:

1. Listener digunakan untuk melihat dan menganalisis hasil tes.
2. Klik kanan pada Thread Group > Add > Listener > View Results Tree atau View Results in Table.
3. Menjalankan Tes
4. Simpan Test Plan:

Simpan Test Plan Anda dengan klik File > Save.

Jalankan Tes:

1. Klik tombol hijau Start di toolbar (atau tekan Ctrl+R).
2. Tes akan mulai berjalan, dan Anda bisa melihat hasilnya di Listener yang telah ditambahkan.
3. Analisis Hasil

View Results Tree:

1. Listener ini menampilkan hasil setiap permintaan secara terperinci.
2. Anda dapat melihat respons lengkap, header, dan waktu respons.

View Results in Table:

1. Listener ini menampilkan hasil dalam bentuk tabel yang lebih ringkas.
2. Anda dapat melihat waktu respons, ukuran respons, dan status dari setiap permintaan.

Bab 5.

Persiapan Wawancara & Praktik Lapangan

5.1. Lingkungan Uji Coba

Terdapat beberapa lingkungan uji coba yang umum digunakan dalam pengembangan perangkat lunak. Beberapa lingkungan tersebut adalah :

- *Localhost* (host lokal). Pengembangan perangkat lunak dilakukan pada ruang lingkup komputer pengembangan yang berarti hanya pengembang tersebut yang dapat mengakses dan melakukan hal-hal lainnya. Pemakaian dalam host lokal biasanya hanya untuk pengembangan.
- *Development server*. Pengembangan perangkat lunak yang sudah dikembangkan dapat diletakkan pada server untuk para pengembang. Pada server ini, pengujian masih belum dilakukan oleh para QA dan hanya para pengembang yang melakukan perubahan ataupun pengujian sementara untuk memeriksa *bug-bug* yang ada. Perangkat lunak yang terpasang pada server ini sangatlah tidak stabil sehingga hanya pengujian- pengujian terbatas saja yang mungkin dapat diberikan.
- *Staging server*.Pengembangan perangkat lunak yang sudah dikembangkan dan siap untuk diuji coba. Pada server ini,

pengujian dapat dilakukan oleh para QA untuk memeriksa *bug-bug* yang masih terlewat, ataupun dibutuhkan perbaikan yang masih belum sesuai dengan kriteria penerimaan (acceptance criteria).

— *Production server*. Pada server ini, hanya pengujian seperti *smoke test*yang dapat dilakukan dikarenakan perangkat lunak yang sudah terpasang pada server ini sudah melalui uji coba yang menyeluruh dan sudah siap untuk digunakan oleh para klien. Pengujian lainnya yang mungkin hanya sebatas untuk memastikan fitur-fitur atau kemampuan utama dapat berjalan. Pengujian pada server ini juga diharapkan untuk sangat berhati-hati mengingat perubahan data pada server ini akan mengakibatkan perubahan integritas data.

5.2. Tipe-tipe Uji Coba

Tipe-tipe uji coba pada sebuah perangkat lunak memiliki metode yang berbeda-beda. Akan tetapi terdapat dua buah dasar uji coba [7]. Tipe-tipe uji coba yang sering digunakan yaitu :

— *Big Bang Testing*. Uji coba ini akan menguji keseluruhan aplikasi yang sudah dikembangkan. Uji coba dilakukan ketika semua fitur dan fungsi sudah dikembangkan atau produk lengkap.

— *Incremental testing*. Uji coba ini akan menguji bagian-bagian dari perangkat lunak, sehingga satu buah fungsi dan fitur terlebih dahulu akan diuji coba, kemudian fungsi dan fitur lain yang akan ditambahkan diuji coba kembalihingga seluruh fungsi dan fitur menjadi sebuah produk lengkap. Langkah-langkah dalam uji coba *incremental* ini adalah uji coba modul perangkat lunak (module tests), uji coba kumpulan modul yang saling terhubung (integration tests) dan seluruh sistem atau uji coba penerimaan (system tests).

Selain dua dasar dari uji coba pada perangkat lunak, terdapat klasifikasi uji coba perangkat lunak [8]. Klasifikasi ini terdapat dua buah yaitu :

- *Black box testing* (Uji coba fungsionalitas). Uji coba inimelihat bagaimana hasil keluaran sebuah perangkat lunak sudah sesuai atau terdapat kesalahan didalamnya. Uji coba ini tidak akan melihat bagaimana perangkat lunak mengolah masukkan dan cara memproses masukkan tersebut.
- *White box testing* (Uji coba struktural). Uji coba ini akan melihat bagaimana sebuah perangkat lunak mengolah masukkan dan cara memproses masukkan tersebut untuk mengidentifikasi sebuah kesalahan.

Terdapat istilah uji coba-uji coba lainnya yang dapat ditemukan umum dalam praktik dilapangan. Uji coba tersebut sebagai berikut [9]:

- *End-to-endtests* (Uji coba antar pengguna). Uji coba ini termasuk memakan banyak waktu. Uji coba akan meniru bagaimana alur sebuah pengguna dapat memakai kemampuan-kemampuan yang diberikan oleh perangkat lunak. Uji coba tersebut dapat dimulai dari paling sederhana seperti pengguna melakukan login hingga contoh skenario untuk membayar barang belanja online.
- *Acceptance tests* (Uji coba penerimaan). Uji coba ini menguji jika seluruh persyaratan bisnis dapat terpenuhi. Uji coba diujikan pada produk lengkap sehinggaakan fokus pada perilaku pengguna.
- *Performance tests* (Uji coba performa). Uji coba ini menguji kemampuan sebuah perangkat lunak dalam menerima banyaknya permintaan dari pengguna. Uji coba ini tidak termasuk fungsional melainkan bagaimana sebuah perangkat dapat diandalkan, stabil dan dapat digunakan pada waktunya.
- *Smoke tests* (Uji cobakabut) :Uji coba ini menguji fungsionalitas sebuah perangkat lunak bagian dasar. Uji

coba dilakukan pada kemampuan utama perangkat lunak dan memeriksa jika perangkat dapat berjalan semestinya saat dipasang diperangkat klien.

5.3. Rencana Uji Coba

Perencanaan uji coba dilaksanakan ketika metodologi uji coba sudah ditentukan. Metodologi uji coba ditentukan terlebih dahulu karena perangkat lunak yang akan diuji coba memiliki strategi, tingkat standar dan pendekatan yang berbeda [10]. Berikut, dokumentasi sebuah rencana uji coba secara umum [11] :

1. Lingkup Uji Coba
   1.1. Komponen-komponen perangkat lunak yang akan diuji coba, termasuk nama, versi dan versi perubahan.
   1.2. Dokumen pelengkap sebagai dasar rencana uji coba, termasuk nama dan versi setiap dokumen.
2. Lingkungan Uji Coba
   2.1. Tempat Uji Coba
   2.2. Konfigurasi perangkat keras dan perangkat tegar yang dibutuhkan
   2.3. Organisasi yang mengikuti
   2.4. Sumber daya manusia yang dibutuhkan
   2.5. Persiapan dan pelatihan yang dibutuhkan untuk tim uji coba
3. Detail Uji Coba
   3.1. Identifikasi uji coba
   3.2. Tujuan obyektif dari uji coba
   3.3. Referensi silang antara desain dokumen dengan dokumen persyaratan
   3.4. Uji coba kelas
   3.5. Tingkat uji coba (uji coba unit, integrasi dan sistem)

3.6. Persyaratan perkara uji coba

3.7. Persyaratan khusus seperti perhitungan lama tanggapan dan syarat keamanan.

3.8. Data yang akan disimpan

4. Jadwal Uji Coba (Jadwal diberikan untuk setiap uji coba atau grup uji coba)

4.1. Lama waktu persiapan

4.2. Lama waktu uji coba

4.3. Lama waktu perbaikan galat

4.4. Lama waktu uji coba regresi

5.4. Software Development Lifecycle (SDLC)

Software development lifecycle merupakan sebuah proses dimana seluruh proses pengembangan sebuah perangkat lunak dibahas secara menyeluruh. Terdapat model-model yang dipakai dalam SDLC, model yang umum dipakai antara lain *waterfall*, *prototyping*, *spiral* dan *agile*. Model *agile* pada zaman ini menjadi model SDLC yang paling banyak digunakan dalam pengembangan perangkat lunak dikarenakan proses pengembangan mengarah pada unit-unit kecil terlebih dahulu, kemudian seluruh unit akan digabungkan untuk menghasilkan sebuah produk. Melalui model ini, pengembangan perangkat lunak dapat mengurangi waktu dan juga biaya, terlebih proses uji coba yang dilakukan fokus pada kerja fungsi perangkat tersebut sehingga identifikasi kesalahan menjadi lebih mudah. Bentuk model *agile* sebagai berikut [12] :

1. Perencanaan
2. Analisa persyaratan
3. Desain
4. Pengembangan
5. Uji coba unit
6. Peluncuran

Metodologi *agile* sendiri terdapat berbagai macam. Beberapa yang paling sering digunakan yaitu *Scrum*, *Lean* dan *Extre programming* dengan *Scrum* menjadi penggunaan terpopuler.

5.5. Browser Web

Browser web adalah aplikasi yang digunakan untuk mengunjungi sebuah situs yang disimpan dalam sebuah server. Situs tersebut memiliki setidaknya satu buah halaman web yang disediakan. Tergantung kegunaan situs tersebut, sebuah aplikasi web dapat dibuat untuk melakukan berbagai macam aktivitas dengan kegunaan utama yaitu menyediakan informasi, mengolah data dsb. Browser web yang paling banyak digunakan (Januari 2022) adalah Chrome dengan browser web Edge dan Firefox mengikuti setelahnya.

2022	**Chrome**	**Edge**	**Firefox**	**Safari**	**Opera**
January	80.1 %	7.3 %	5.5 %	3.9 %	2.3 %
2021	**Chrome**	**Edge**	**Firefox**	**Safari**	**Opera**
December	81.0 %	6.6 %	5.5 %	3.7 %	2.3 %

Sumber :https://www.w3schools.com/browsers/default.asp

Perlu diketahui bahwa browser web yang akan digunakan berpengaruh terhadap tampilan aplikasi web sehingga proses uji coba akan lebih baik jika mengambil seluruh kemungkinan penggunaan aplikasi di browser web yang berbeda.

5.6. Alat-Alat Umum Uji Coba

Tidak dapat dipungkiri bahwa uji coba secara automasi menjadi kebutuhan proses uji coba perangkat lunak. Alasannya tidak lain dikarenakan aplikasi yang dikembangkan akan mengalami penambahan kegunaan dan fungsi seiring dengan bertambahnya kebutuhan dari pengguna. Beberapa alat-alat umum uji coba secara automasi yang sering digunakan adalah :

1. Katalon Studio

Sumber :https://www.katalon.com/katalon-studio/

2. Selenium

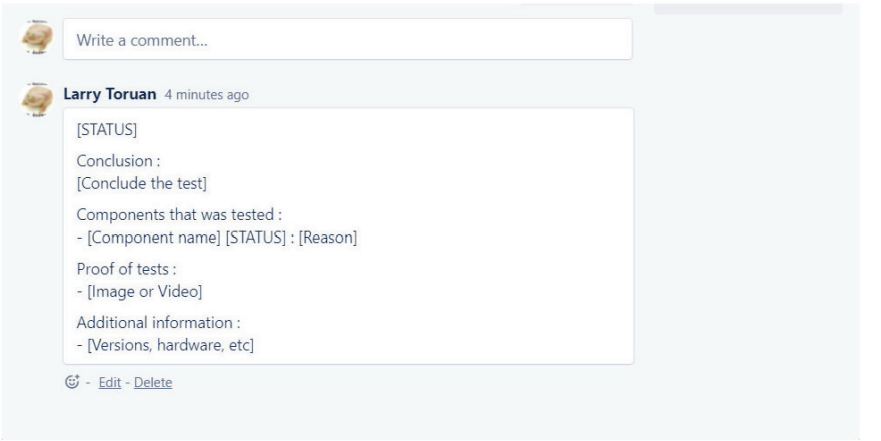
Sumber :https://www.selenium.dev/

3. Appium

Sumber :https://appium.io/

5.7. Penulisan Hasil Uji Coba Pada Aplikasi Manajemen Proyek

Menggunakan model *agile* dalam pengembangan sebuah perangkat lunak, aplikasi manajemen proyek digunakan sebagai alat untuk mengawasi jalannya pengembangan perangkat lunak. Terdapat berbagai macam aplikasi manajemen proyek, beberapa yang populer seperti Trello (https://trello.com/) dan Jira (https://www.atlassian.com/software/jira). Hasil uji coba pada komponen-komponen akan dituliskan didalam aplikasi manajemen proyek sebagai bentuk laporan kecil dalam pengembangan perangkat lunak. Berikut sebuah contoh penulisan hasil uji coba didalam aplikasi manajemen proyek :



Dipecahkan satu persatu maka :

1. **Status**. Berisi status sebuah fungsional / fitur / unit yang sudah diuji cobakan. Status yang diberikan antara lain *Pass*, *Warning* dan *Fail*. Status*Pass* menyatakan fungsional / fitur / unit tersebut lolos uji coba dan layak untuk digunakan. Status *Warning* menyatakan fungsional / fitur / unit tersebut lolos uji coba tetapi memiliki kecacatan lain yang dapat mempengaruhi fungsional / fitur / unit lain sehingga menyebabkan sebuah galat atau kesalahan. Status *Fail* menyatakan fungsional / fitur / unit tersebut tidak lolos uji coba dan harus dikembangkan ulang.

2. ***Conclusion***.Berisi kesimpulan suatu uji coba. Kesimpulan ini merupakan rangkuman dari seluruh uji coba yang dilakukan pada suatu fungsional / fitur / unit.
3. ***Components that was tested***. Berisi detail komponen-komponen yang sudah diuji cobakan. Detail ini akan berisi nama komponen beserta status dari setiap komponen dan alasan dari status yang didapatkan. Jenis-jenis status yang diberikan sama seperti poin nomor 1.
4. ***Proof of tests***. Berisi bukti-bukti hasil uji coba. Bukti uji coba yang dapat berupa gambar tahapan ataupun video.
5. ***Additional Information***. Informasi tambahan biasanya berisi nomor versi perangkat lunak yang sedang diuji cobakan atau versi unit / paket yang diuji cobakan. Informasi lainnya seperti penggunaan jenis hardware atau browser web dapat ditambahkan jika diperlukan.

Bab 6.

Pengalaman Pragmatis

Penulis ingin menceritakan beberapa pengalaman yang ditemukan ketika melaksanakan magang sebagai seorang *quality assurance*. Pengalaman ini juga ditambah dengan beberapa fakta nyata dari rekan kerja di beberapa tempat lainnya. Cerita ini akan menjelaskan kesulitan-kesulitan serta hal-hal penting dalam komunikasi dengan rekan-rekan kerja yang akan berhubungan langsung pekerjaan *quality assurance*. Akan tetapi, tidak semua dapat dijadikan acuan secara langsung mengingat setiap tempat memiliki lingkungan kerja dan tugas-tugas yang berbeda sehingga baiknya mengambil kesimpulan secara umum dari seluruh cerita-cerita tersebut.

Banyaknya uji coba sudah dipastikan akan menjadi tugas keseharian seorang *quality assurance*. Hal ini tidak lain disebabkan karena memang pengembangan sebuah perangkat lunak memiliki modul-modul atupun komponen yang akan diperiksa supaya bekerja sesuai dengan kriteria yang sudah diberikan. Uji coba juga dapat semakin kompleks jika seandainya modul atau komponen yang akan di uji coba membutuhkan modul atau komponen lainnya. Jawaban otomasi sudah menjadi solusi utama dalam memecahkan jumlah uji coba yang begitu banyak sehingga diharapkan untuk dapat menggunakan otomasi sebanyak mungkin pada banyak uji coba. Waktu menjadi sumber daya paling berharga dalam pengembangan perangkat lunak, baik dari sisi pengembang maupun *quality assurance*.

Jika banyak uji coba masih dilakukan secara manual, penulis menyarankan untuk melakukan pemeriksaan segera mungkin karena uji coba dapat tiba-tiba menjadi kompleks (seperti modul-modul yang membutuhkan satu sama lain dan lagi waktu merupakan sumber daya yang sangat berharga) karena tujuan uji coba sudah dipastikan untuk mencari seluruh kemungkinan kesalahan dan mengecilkan peluang munculnya galat. Sebagai kesimpulan, tidak boleh adanya ruang untuk terjadinya kesalahan perangkat lunak.

Beberapa hal perlu diperhatikan dalam uji coba antarmuka, baik dari sisi web ataupun mobile yaitu pastikan untuk dapat melihat balasan API yang diberikan. Melihat balasan API dapat dipastikan bahwa hal apapun yang terjadi dibelakang layar dapat diketahui dan dibandingkan dengan tampilan di antarmuka. Perhatikan juga jika komponen-komponen tersebut akan bereaksi dan mempengaruhi komponen lainnya ataupun data yang disajikan. Sedikit cerita, penulis mendapat penampakan langsung dari sebuah komponen centang pemilihan masalmasih tercentang ketika salah satu baris dalam tabel tidak terpilih.

Terakhir, komunikasi yang baik antar rekan kerja sangatlah penting. Seluruh tugas tidak dapat diselesaikan jika komunikasi yang baik tidak berjalan. Perlu diperhatikan, komunikasi yang tepat sasaran, jelas dan tidak berbelit-belit menjadi kunci utama komunikasi dapat menyelesaikan masalah-masalah yang dihadapi. Konsep kerja sama dalam sebuah tim menjadi gambaran yang penting untuk diingat sehingga komunikasi yang baik sudah dipastikan menjadi hasil dari penyelesaian masalah-masalah yang akan dihadapi.

Daftar Pustaka

[1] Software quality assurance from theory to implementation page-29

[2] KBBI - Rekayasa

[3] https://www.codecademy.com/resources/blog/what-is-a-qa-engineer/

[4] https://www.w3schools.com/whatis/whatis_js.asp

[5] https://www.w3schools.com/js/js_history.asp

[6] https://www.w3schools.com/js/js_variables.asp

[7] Software quality assurance from theory to implementation page-233 [259]

[8] Software quality assurance from theory to implementation page-182 [209]

[9] Software quality assurance from theory to implementation page-187 [214]

[10] https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing

[11] Software quality assurance from theory to implementation page-218 [245]

[12] Software quality assurance from theory to implementation page-228 [255]

[13] https://www.w3schools.in/sdlc-tutorial/agile-model/

[14] Software quality assurance from theory to implementation page-229 [256]

[15] https://softwaretestingfundamentals.com/smoke-testing/

[16] https://www.linkedin.com/pulse/sanity-smoke-regression-testing-zepri-togatorop/?originalSubdomain=id

[17] https://www.usability.gov/how-to-and-tools/methods/usability-testing.html#:~:text=Usability%20testing%20refers%20to%20evaluating,testing%20it%20with%20representative%20users.&text=The%20goal%20is%20to%20identify,participant's%20satisfaction%20with%20the%20product.

[18] https://www.postman.com/product/what-is-postman/

[19] https://youtu.be/tgbRY96q-KM

[20] https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

[21] https://www.w3schools.com/tags/ref_httpmethods.asp

[22] https://fileinfo.com/extension/json

[23] https://learning.postman.com/docs/sending-requests/variables/

[24] https://learning.postman.com/docs/writing-scripts/script-references/postman-sandbox-api-reference/#scripting-with-response-data

[25] https://medium.com/katalon-studio/a-sample-web-automation-test-project-9c532237c2bd

Penulis

Adelbertus Larry Dennis Lumban Toruan adalah lulusan dari Universitas Atma Jaya Yogyakarta dengan gelar Sarjana Informatika, yang memiliki minat besar dalam administrasi sistem, dukungan TI, jaringan, server Linux, dan teknologi informasi. Dia pernah menjabat sebagai Asisten Dosen untuk mata kuliah Jaringan Komputer ketika kuliah, meningkatkan pemahaman mahasiswa tentang konsep OSI Layer dan protokol TCP/IP serta membantu dalam penilaian. Selama berkarier sebagai QA Engineer di PT. Codemi Global, dia menguji berbagai fitur, secara proaktif mencari bug, dan bekerja sama dengan pengembang. Sebagai tutor di Linux Study Club, dia mengajarkan teknik keamanan siber dan memberikan bantuan dalam pemecahan masalah sistem operasi Linux, yang berpuncak pada tesis tentang pengembangan Sistem Pembelajaran Linux menggunakan metode Capture The Flag.

Rangga Perwiratama adalah seorang profesional berdedikasi yang tinggal di Yogyakarta, Indonesia. Sejak Juli 2023, ia telah mengajar sebagai dosen di Universitas Atma Jaya Yogyakarta, tempat di mana ia juga meraih gelar Sarjana Teknik dan Magister di bidang Informatika. Dengan pengalaman yang luas dalam teknik DevOps, Rangga pernah menjabat sebagai Senior DevOps Engineer di Mamikos.com. Sebelumnya, ia menghabiskan waktu yang cukup lama di Cakap.com, di mana ia tidak hanya memimpin tim DevOps tetapi juga berkontribusi sebagai

Full Stack Engineer. Keahlian Rangga mencakup berbagai aspek dari Red Hat System Administration hingga Amazon Web Services (AWS) DevOps Engineering.

Djoko Budiyanto Setyohadi menerima gelar B.E. gelar di bidang Teknik Elektro dari Universitas Gadjah Mada, Yogyakarta, Indonesia pada tahun 1990, M.Eng. gelar di bidang Ilmu Komputer, Manajemen Informasi dari Asian Institute of Technology, Bangkok, Thailand, pada tahun 1998 dan Ph.D. gelar di bidang Ilmu Komputer dari Universitas Kebangsaan Malaysia. Beliau merupakan Guru Besar Informatika pada Fakultas Teknik Industri Universitas Atma Jaya Yogyakarta. Minat penelitiannya saat ini meliputi Sosio Informatika, Sistem Informasi, Human Computer Interaction dan Rekayasa Data.

Pengenalan
Software Quality Assurance

Buku Seri : Pengembangan Sistem

Perangkat lunak memainkan peran yang semakin penting dalam kehidupan kita, dari aplikasi ponsel pintar hingga perangkat lunak rumah tangga dan sistem informasi perusahaan. Namun, dengan meningkatnya kompleksitas perangkat lunak, tantangan yang dihadapi pengembang juga semakin besar. Salah satu tantangan utama adalah memastikan bahwa perangkat lunak yang dihasilkan memenuhi standar kualitas yang tinggi.

Pertama, buku ini akan membahas pentingnya Software Quality Assurance (SQA) dalam dunia perangkat lunak. SQA adalah pendekatan sistematis untuk memastikan bahwa perangkat lunak memenuhi standar kualitas dan proses pengembangannya berjalan sesuai standar.

Kedua, buku ini bertujuan untuk memberikan pemahaman yang komprehensif tentang konsep-konsep dasar SQA, praktik terbaik, dan metode untuk meningkatkan kualitas perangkat lunak. Buku ini membahas berbagai aspek SQA, mulai dari pengenalan dasar hingga praktik terbaik dalam implementasinya, juga membahas metode dan teknik seperti pengujian perangkat lunak, analisis kode, dan manajemen konfigurasi, serta pentingnya pengukuran kualitas perangkat lunak.

Ketiga, buku ini menyoroti manajemen test case dan langkah-langkah yang didefinisikan dengan baik untuk menguji perangkat lunak. Buku ini juga membahas penggunaan alat-alat seperti QA Touch dan Postman untuk manajemen test case yang efektif. Buku ini ditujukan untuk praktisi di bidang pengembangan perangkat lunak, mahasiswa yang belajar tentang SQA, dan siapa pun yang ingin memahami lebih lanjut tentang memastikan kualitas perangkat lunak. Penulis berharap buku ini menjadi sumber pengetahuan yang bermanfaat dan membantu Anda memahami SQA lebih baik.

ISBN: 978-623-10-2029-1(PDF)

Universitas Atma Jaya Yogyakarta
Jl. Babarsari No. 5-6 Yogyakarta 55281
Telp. +62 274 487711
E-mail: llb.publisher@uajy.ac.id