# Dokumentasi Pengujian HTTP API dengan Mocha dan Chai

API Testing merupakan tipe pengujian dimana pengujian dilakukan secara langsung untuk memastikan apa yang diuji sudah sesuai dengan ekspektasi atau tidak. API testing yang dilakukan dengan *tools* Mocha dan Chai.

Sample API yang akan diuji adalah "REST apis with Nodejs without using any framework or external library" (*source:* https://github.com/aditya-sridhar/simple-rest-apis-nodejs-without-frameworks.git).

Langkah pertama adalah dengan melakukan *clone* ke local pada sample API.

```
Microsoft Windows [Version 10.0.17134.885]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\stu>git clone https://github.com/aditya-sridhar/simple-rest-apis-nodejs-without-frameworks.git
Cloning into 'simple-rest-apis-nodejs-without-frameworks'...
remote: Enumerating objects: 26, done.
remote: Total 26 (delta 0), reused 0 (delta 0), pack-reused 26
Unpacking objects: 100% (26/26), done.

C:\Users\stu>
```

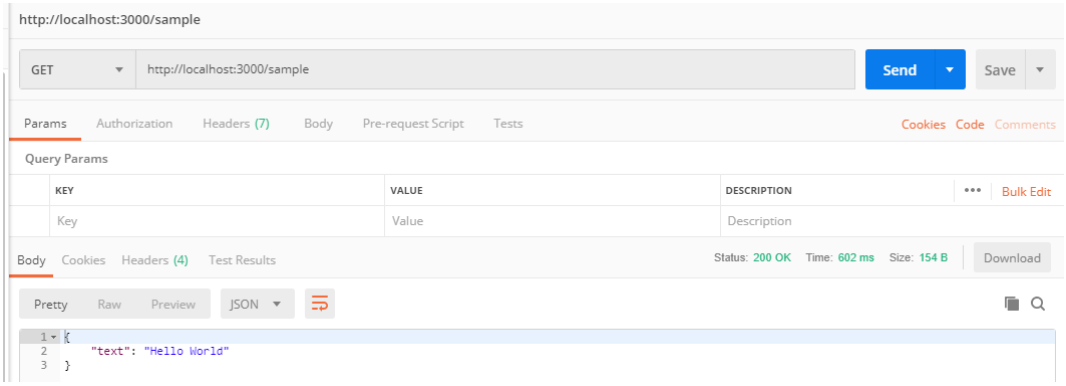
Setelah berhasil melakukan *clone* pada sample API, kemudian menjalankan aplikasi.

```
C:\Users\stu\simple-rest-apis-nodejs-without-frameworks>node server.js
Server running at http://127.0.0.1:3000/
```

Sebelum melakukan test, untuk memastikan apakah aplikasi sudah berjalan dengan baik maka akan dicoba pemanggilan endpoint pada API. *Tool* yang digunakan untuk melakukan hal tersebut adalah Postman. Postman dalam konteks ini, berperan sebagai *tool* untuk memanggil API dan mengecek apakah terdapat response atau tidak API dari API yang akan diuji. Endpoint yang akan diuji adalah:

- "/sample" dengan menggunakan method GET
- "/test" dengan menggunakan method POST
- "/test123" dengan menggunakan method POST
- "/hello" dengan menggunakan method GET
- "/hello_name?name= dengan menggunakan method GET

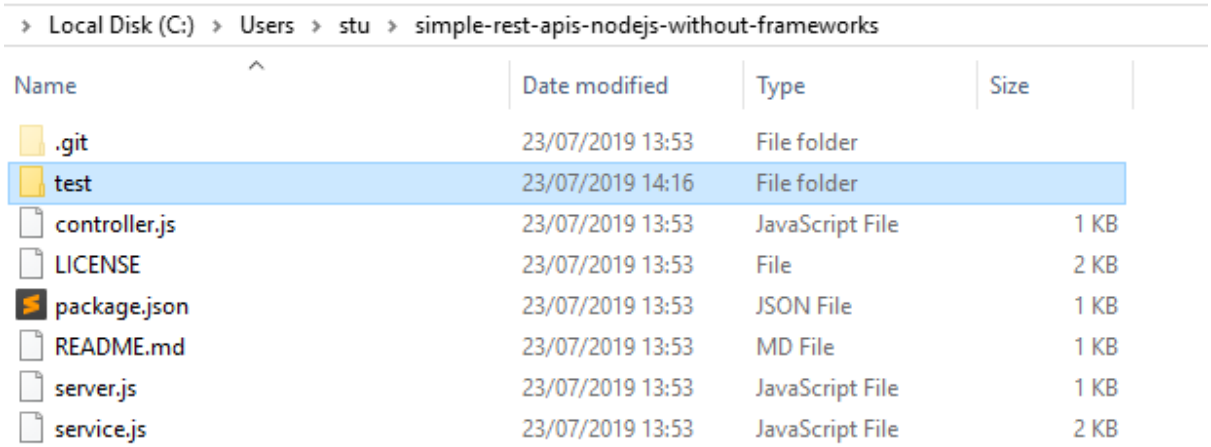
## 1. Pengujian Endpoint “/sample” Dengan Menggunakan Method GET




Dari gambar diatas, dapat dilihat bahwa pada saat API endpoint /sample dipanggil, API tersebut memberikan response text:"Hello World" dan succesful response yaitu 200 yang menandakan bahwa request HTTP berhasil.

Apabila telah memenuhi response seperti diatas, maka akan dilakukan API testing menggunakan Mocha dan Chai.

Pada direktori sample API yang berada di lokal, di tambahkan satu folder yang bernama test. Folder tersebut dibuat untuk menampung file test case yang akan di buat.

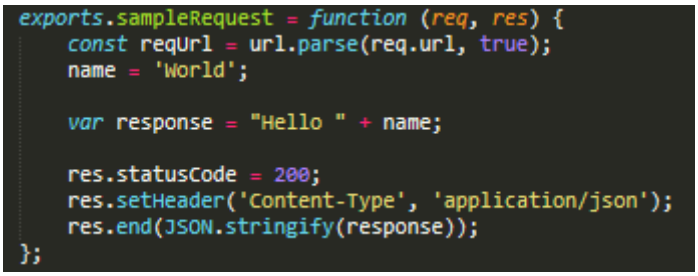



Kemudian, dalam folder test, akan dibuat satu file yaitu sample_test.js. Dalam file tersebut akan dituliskan script untuk melakukan API Testing pada endpoint yang akan diuji.

Berdasarkan, potongan code pada file controller.js dimana salah satu endpoint yang akan diuji adalah ‘/sample’.

```
    //GET Endpoint
    if (requrl.pathname == '/sample' && req.method === 'GET') {
        console.log('Request Type:' +
            req.method + ' Endpoint: ' +
            requrl.pathname);

        service.sampleRequest(req, res);
```

Dimana pada saat endpoint ‘/sample’ dipanggil, maka akan dihasilkan response text:"Hello World" yang ditunjukkan pada potongan code dibawah ini.



```
exports.sampleRequest = function (req, res) {
    const requrl = url.parse(req.url, true);
    name = 'World';

    var response = "Hello " + name;

    res.statusCode = 200;
    res.setHeader('Content-Type', 'application/json');
    res.end(JSON.stringify(response));
};
```

Didalam file sample_test.js, hal pertama yang dilakukan adalah melakukan import modul-modul yang diperlukan. Chai merupakan assertion library yang akan menggunakan style expect untuk melakukan assertion pada respon API. chai-http merupakan plugin dalam melakukan HTTP response.

```
//author: anggifrecelia

let chai = require('chai');
let expect = require('chai').expect;
let chaiHttp = require('chai-http');

chai.use(chaiHttp);

describe('API Testing', () => {

  it('it should success to retrive all data', (done) => {
    chai.request('localhost:3000') // chai meminta base URL dan port
        .get('/sample') // 'get'  adalah REST method dan '/sample' adalah endpoint API yang di test
        .end((err, res) => {
          // kita melakukan response assertion
          expect(res).to.have.status(200);

        });
     done();
  });
});
```

Penggunaan MochaJS dalam hal ini terdiri dari describe() dan it(). describe() untuk mendeklarasikan butir test case sedangkan it() adalah butir test case.

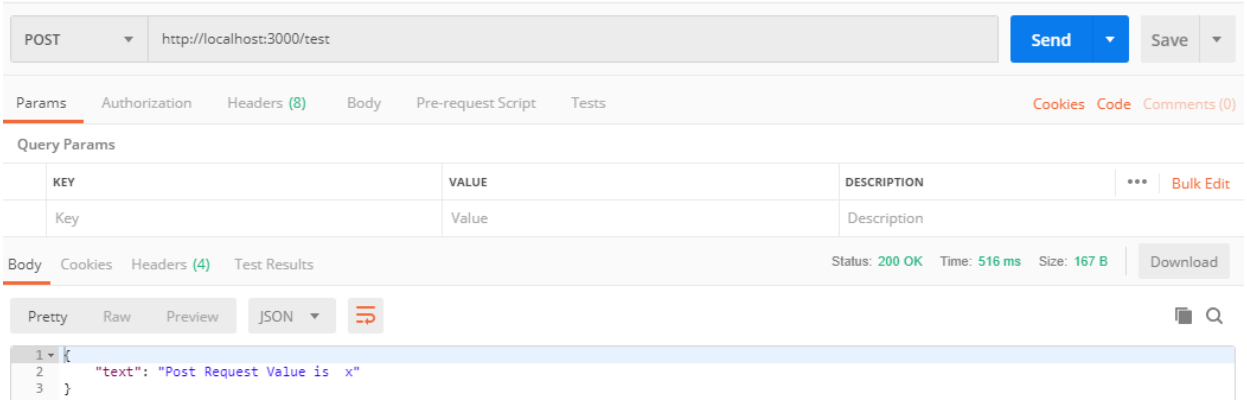
Untuk melakukan API testing maka dijalankan perintah mocha test pada command prompt.

```
C:\Users\stu\simple-rest-apis-nodejs-without-frameworks>mocha test


  API Testing
    √ it should success to retrive all data (72ms)


  1 passing (128ms)


C:\Users\stu\simple-rest-apis-nodejs-without-frameworks>
```

Berdasarkan hasil yang diperoleh melalui perintah: mocha test, maka didapatkan pengujian endpoint yang dilakukan bertanda hijau (passed) yang menunjukkan bahwa respon dari pengujian tersebut sesuai dengan ekspektasi.

**2. Pengujian Endpoint "/test" Dengan Menggunakan Method POST**

Sebelum melakukan pengecekan pada endpoint /test dengan menggunakan mocha dan chai, terlebih dahulu dilakukan pengecekan terhadap endpoint /test dengan menggunakan Postman. Hal ini dilakukan untuk mengetahui apakah endpoint ini menghasilkan respon atau tidak. Pengecekan endpoint dengan menggunakan postman dapat dilihat pada gambar dibawah.




Berdasarkan gambar diatas, dapat dilihat bahwa pada saat endpoint /test dipanggil dengan menggunakan method POST, maka akan memberikan respon "text": "Post Request Value is x" dan succesful response yaitu 200 yang menandakan bahwa request HTTP berhasil.

Berdasarkan, potongan code pada file controller.js dimana salah satu endpoint yang akan diuji adalah '/test'.

```
    // POST Endpoint
} else if (reqUrl.pathname == '/test' && req.method === 'POST') {
    console.log('Request Type:' +
        req.method + ' Endpoint: ' +
        reqUrl.pathname);

    service.testRequest(req, res);
```

Dimana pada saat endpoint ‘/test’ dipanggil, maka akan dihasilkan response "text": "Post Request Value is x" yang ditunjukkan pada potongan code pada file service.js.

```
exports.testRequest = function (req, res) {
    body = '';

    req.on('data', function (chunk) {
        body += chunk;
    });

    req.on('end', function () {

        var postBody = 'x';

        var response = {
            "text": "Post Request Value is  " + postBody
        };

        res.statusCode = 200;
        res.setHeader('Content-Type', 'application/json');
        res.end(JSON.stringify(response));
    });
};
```

Kemudian melakukan pengujian endpoint /test dengan method POST menggunakan mocha dan chai, dengan menuliskan script terlebih dahulu pada file sample_test.js. Seperti yang dapat dilihat pada gambar dibawah, bahwa pada saat penulisan script untuk method POST ditambahkan fungsi .send() yang menandakan bahwa terjadi pengiriman payload ‘post request value is x’ bersamaan dengan request POST. Assertion yang dilakukan adalah bahwa respon diharapkan menghasilkan status 200 yang berarti request berhasil.

```
describe('/POST Testing', () => {

  it('it should success to retrive all data', (done) => {
  chai.request('localhost:3000')
    .post('/test')
    .send('Post Request Value is  x')
    .end((err, res) => {

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

    });
   done();

  });
```

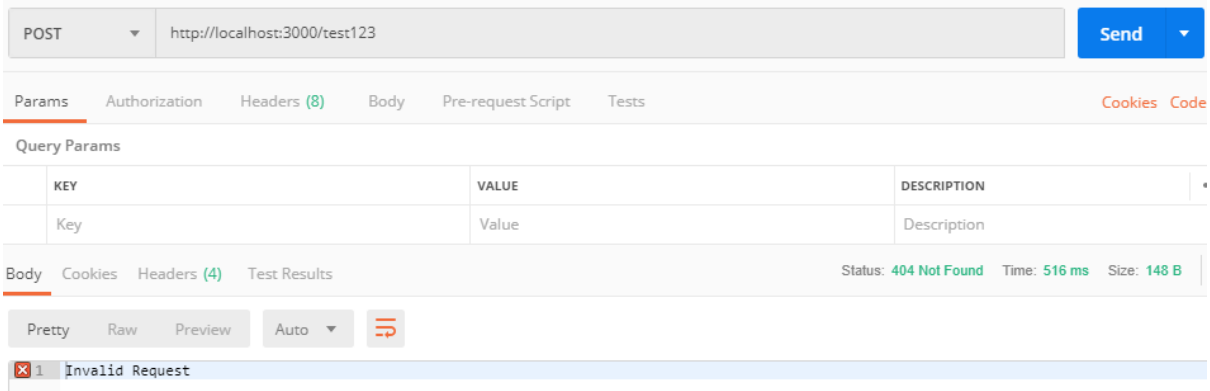
Untuk melakukan API testing maka dijalankan perintah mocha test pada command prompt.

```
C:\Users\stu\simple-rest-apis-nodejs-without-frameworks>mocha test


  /GET Testing
    √ it should success to retrive all data (90ms)

  /POST Testing
    √ it should success to retrive all data


  2 passing (271ms)
```

Berdasarkan hasil yang diperoleh melalui perintah: mocha test, maka didapatkan pengujian endpoint /test dengan menggunakan method POST yang dilakukan bertanda hijau (passed) yang menunjukkan bahwa respon dari pengujian tersebut sesuai dengan ekspektasi.

### 3. Pengujian Endpoint "/test123" Dengan Menggunakan Method POST

Sebelum melakukan pengecekan pada endpoint /test123 dengan menggunakan mocha dan chai, terlebih dahulu dilakukan pengecekan terhadap endpoint /test123 dengan menggunakan Postman. Hal ini dilakukan untuk mengetahui apakah endpoint ini menghasilkan respon atau tidak. Pengecekan endpoint dengan menggunakan postman dapat dilihat pada gambar dibawah.




Berdasarkan gambar diatas, dapat dilihat bahwa pada saat endpoint /test123 dipanggil dengan menggunakan method POST, maka akan memberikan respon Invalid Request dan status 404 Not Found yang menandakan bahwa server tidak dapat menemukan url yang diminta atau url tidak dikenali.

Berdasarkan, potongan code pada file controller.js dimana salah satu endpoint yang akan diuji adalah '/test123'.

```
    // POST Endpoint
} else if (reqUrl.pathname == '/test' && req.method === 'POST') {
    console.log('Request Type:' +
        req.method + ' Endpoint: ' +
        reqUrl.pathname);

    service.testRequest(req, res);

} else {
    console.log('Request Type:' +
        req.method + ' Invalid Endpoint: ' +
        reqUrl.pathname);

    service.invalidRequest(req, res);

}
```

Dimana pada saat endpoint '/test123' dipanggil, maka akan dihasilkan response Invalid Request yang ditunjukkan pada potongan code pada file service.js.

```
exports.invalidRequest = function (req, res) {
    res.statusCode = 404;
    res.setHeader('Content-Type', 'text/plain');
    res.end('Invalid Request');
};
```

Kemudian melakukan pengujian endpoint /test123 dengan method POST menggunakan mocha dan chai, dengan menuliskan script terlebih dahulu pada file sample_test.js. Seperti yang dapat dilihat pada gambar dibawah, bahwa pada saat penulisan script untuk method POST ditambahkan fungsi .send() yang menandakan bahwa terjadi pengiriman payload 'Invalid Request' bersamaan dengan request POST. Assertion yang dilakukan adalah bahwa respon diharapkan menghasilkan status 404 yang berarti request gagal.

```
  it('it should be return invalid request', (done) => {
  chai.request('localhost:3000')
  .post('/test123')
  .send('Invalid Request')
  .end((err, res) => {

      expect(res).to.have.status(404);

  });
  done();
  });
});
```

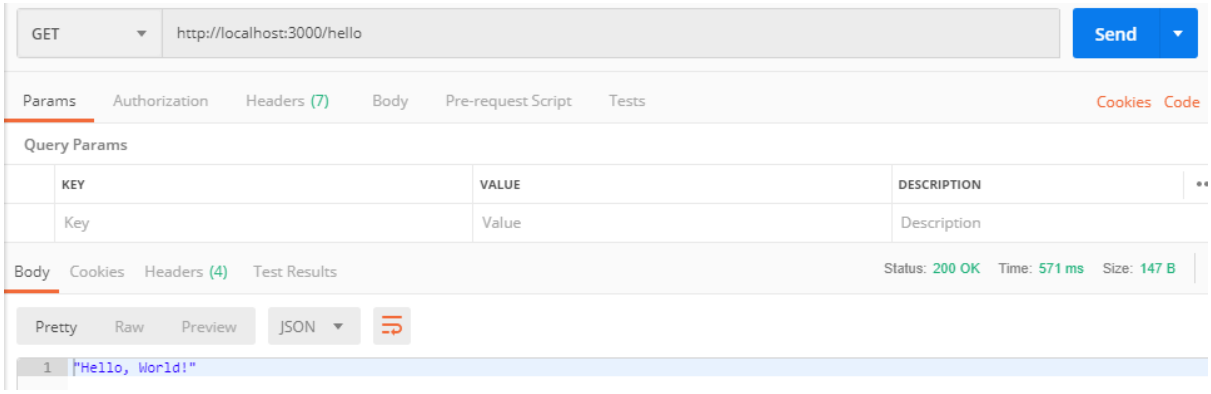
Untuk melakukan API testing maka dijalankan perintah mocha test pada command prompt.

```
C:\Users\stu\simple-rest-apis-nodejs-without-frameworks>mocha test


  /GET Testing
    √ it should success to retrive all data (69ms)

  /POST Testing
    √ it should success to retrive all data
    √ it should be return invalid request


  3 passing (223ms)
```

Berdasarkan hasil yang diperoleh melalui perintah: mocha test, maka didapatkan pengujian endpoint /test123 dengan menggunakan method POST yang dilakukan bertanda hijau (passed) yang menunjukkan bahwa respon dari pengujian tersebut sesuai dengan ekspektasi.

### 4. Pengujian Endpoint "/hello" Dengan Menggunakan Method GET

Sebelum melakukan pengecekan pada endpoint /hello dengan menggunakan mocha dan chai, terlebih dahulu dilakukan pengecekan terhadap endpoint /hello dengan menggunakan Postman. Hal ini dilakukan untuk mengetahui apakah endpoint ini menghasilkan respon atau tidak. Pengecekan endpoint dengan menggunakan postman dapat dilihat pada gambar dibawah.




Berdasarkan gambar diatas, dapat dilihat bahwa pada saat endpoint /hello dipanggil dengan menggunakan method GET, maka akan memberikan respon "Hello, World!" dan succesful response yaitu 200 yang menandakan bahwa request HTTP berhasil.

Berdasarkan, potongan code pada file controller.js dimana salah satu endpoint yang akan diuji adalah '/hello'.

```
// GET Endpoint to return string "Hello, world!"
}else if (reqUrl.pathname == '/hello' && req.method === 'GET'){
    console.log('Request Type:' +
        req.method + ' Endpoint: ' +
        reqUrl.pathname);

    service.helloRequest(req, res);
```

Dimana pada saat endpoint '/hello' dipanggil, maka akan dihasilkan response "Hello, World!" yang ditunjukkan pada potongan code pada file service.js.

```
exports.helloRequest = function (req, res) {
    const reqUrl = url.parse(req.url, true);
    world = 'World';

    var response = "Hello" + ", " + world + "!";

    res.statusCode = 200;
    res.setHeader('Content-Type', 'application/json');
    res.end(JSON.stringify(response));

}
```

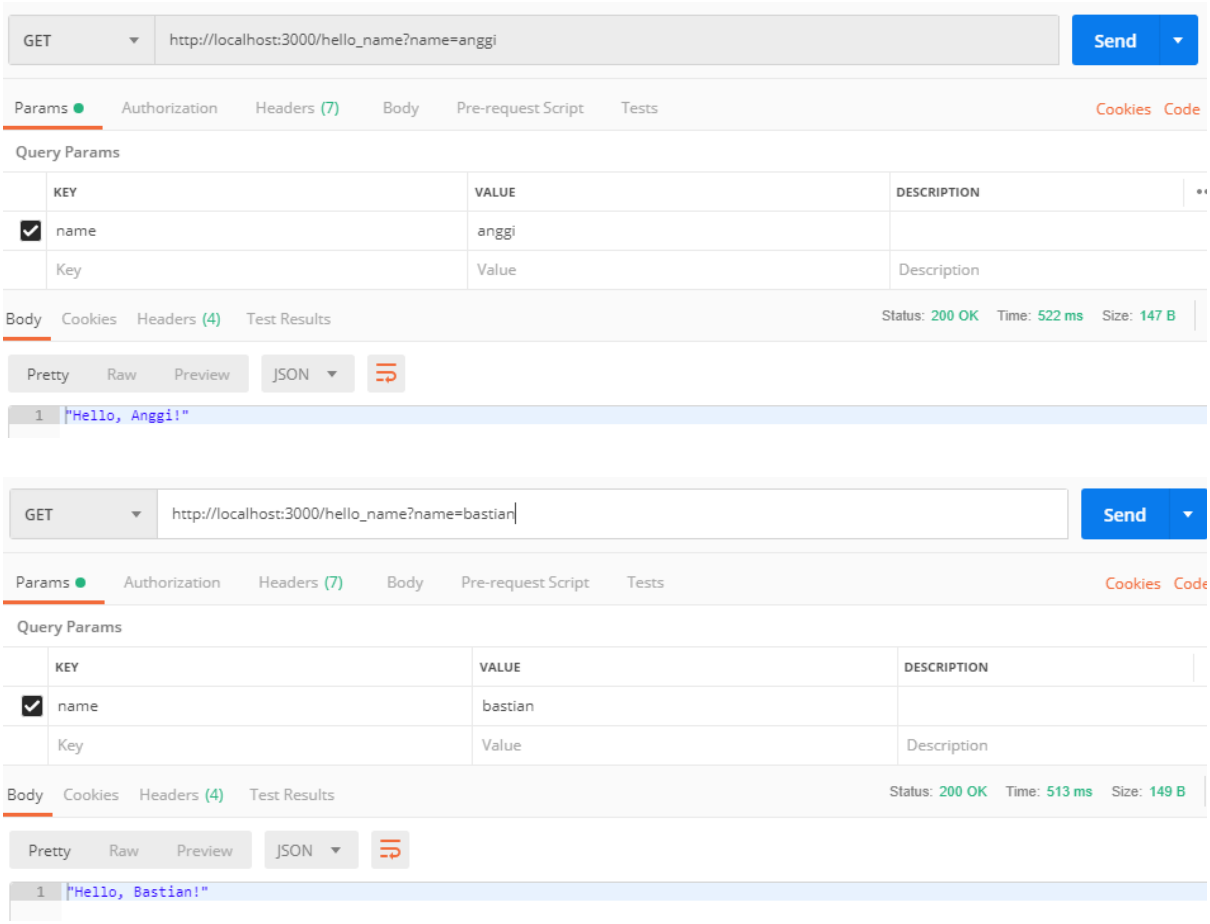
Kemudian melakukan pengujian endpoint /hello dengan method GET menggunakan mocha dan chai, dengan menuliskan script terlebih dahulu pada file sample_test.js. Seperti yang dapat dilihat pada gambar dibawah. Assertion yang dilakukan adalah bahwa respon diharapkan menghasilkan status 202 yang berarti request berhasil.

```
it('it should success to retrive Hello, World!', (done) => {
  chai.request('localhost:3000') // chai meminta base URL dan port
      .get('/hello') // 'get'  adalah REST method dan '/hello' adalah endpoint API yang di test
      .end((err, res) => {
        // kita melakukan response assertion
        expect(res).to.have.status(200);

      });
   done();
});
```

Kemudian melakukan API testing maka dengan menjalankan perintah mocha test pada command prompt.

**5. Pengujian Endpoint "/hello_name" Dengan Menggunakan Method GET**

Sebelum melakukan pengecekan pada endpoint /hello_name dengan menggunakan mocha dan chai, terlebih dahulu dilakukan pengecekan terhadap endpoint /hello_name dengan menggunakan Postman. Hal ini dilakukan untuk mengetahui apakah endpoint ini menghasilkan respon atau tidak. Pengecekan endpoint dengan menggunakan postman dapat dilihat pada gambar dibawah.




Dari pengujian yang dapat dilihat pada gambar diatas, bahwa pengujian endpoint /hello_name dilakukan untuk mengecek apakah pada saat endpoint /hello_nane dipanggil, endpoint tersebut dapat memberikan respon string dinamis sesuai dengan parameter yang dikirim. Misalnya apabila endpoint /hello_name?=anggi dipanggil maka akan memberikan respon "Hello, Anggi!" dan succesful response yaitu 200 yang menandakan bahwa request HTTP berhasil.

Berdasarkan, potongan code pada file controller.js dimana salah satu endpoint yang akan diuji adalah '/hello_name'.

```
//GET Endpoint to return dynamic string
} else if (reqUrl.pathname == '/hello_name' && req.method === 'GET') {
    console.log('Request Type:' +
        req.method + ' Endpoint: ' +
        reqUrl.pathname);

    service.hello_nameRequest(req, res);
```

Dimana pada saat endpoint '/hello_name' dipanggil, maka akan dihasilkan response respon string dinamis sesuai dengan parameter yang dikirim yang ditunjukkan pada potongan code pada file service.js.

```
exports.hello_nameRequest = function (req, res) {
    const reqUrl = url.parse(req.url, true);
    name = '';
    if (reqUrl.query.name) {
        name = reqUrl.query.name
    }

    var response = "Hello" + ", " + name.charAt(0).toUpperCase() + name.slice(1) + "!";

    res.statusCode = 200;
    res.setHeader('Content-Type', 'application/json');
    res.end(JSON.stringify(response));

}
```

Kemudian melakukan pengujian endpoint /hello_name dengan method GET menggunakan mocha dan chai, dengan menuliskan script terlebih dahulu pada file sample_test.js. Seperti yang dapat dilihat pada gambar dibawah. Assertion yang dilakukan adalah bahwa respon diharapkan menghasilkan status 202 yang berarti request berhasil.

```
  it('it should be returns a dynamic string according to the parameters sent', (done) => {
    chai.request('localhost:3000')
        .get('/hello_name')
        .end((err, res) => {

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

        });
    done();
  });

});
```

Kemudian melakukan API testing maka dengan menjalankan perintah mocha test pada command prompt.

```
C:\Users\stu\simple-rest-apis-nodejs-without-frameworks>mocha test


  /GET Testing
    √ it should success to retrive all data (79ms)
    √ it should success to retrive Hello, World!
    √ it should be returns a dynamic string according to the parameters sent

  /POST Testing
    √ it should success to retrive all data
    √ it should be return invalid request


  5 passing (256ms)
```

Berdasarkan hasil yang diperoleh melalui perintah: mocha test, maka didapatkan pengujian HTTP API dengan menggunakan method GET dan POST yang dilakukan bertanda hijau (passed) yang menunjukkan bahwa respon dari pengujian sudah sesuai dengan ekspektasi.