

| **Input**  describe('Update User', () => {  it('Successfully update user', () => {  var user = {  "name": "Alfian",  "job": "Quality Assurance"  }  cy.request('PUT', 'https://reqres.in/api/users/2', user).then((response) => {  expect(response.status).equal(200)  expect(response.body.name).to.eq(user.name)  })  });  }); |
| --- |
| **Output** |

**TUGAS**

JSON Schema sendiri adalah sebuah bahasa yang digunakan untuk mendefinisikan struktur dan validasi data dalam format JSON. JSON Schema digunakan untuk memastikan bahwa data yang diterima atau dikirim oleh aplikasi sesuai dengan format dan aturan tertentu, sehingga memungkinkan untuk memvalidasi data JSON dengan cara yang konsisten dan otomatis. Nah ini adalah contoh dari JSON Schema yang sederhana.

| {  "type": "object",  "properties": {  "nama": {  "type": "string"  },  "umur": {  "type": "integer"  },  "alamat": {  "type": "string"  }  },  "required": ["nama"]  } |
| --- |

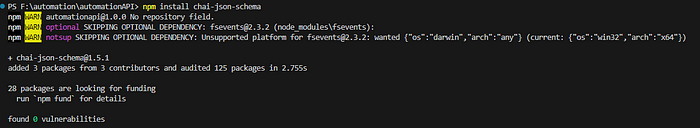
Lalu bagaimana menggunakan JSON Schema ini pada assertion? apa juga keuntungan dari menggunakan JSON Schema sebagai assertion daripada asserion lain? Yang mana sih dokumentasinya? Tenang temen-temen, kita akan bahas *one by one* hehe.

Jadi JSONSchema adalah plugin pada ChaiJS yang memungkinkan untuk melakukan asertasi (assertions) berdasarkan skema JSON menggunakan sintaks yang familiar dengan ChaiJS. Keunggulan menggunakan JsonSchema adalah kita tidak perlu memverifikasi satu per satu dari setiap property, tapi kita dapat langsung memverifikasinya bersamaan. Untuk dokumentasi dari JsonSchema dapat dibaca di link berikut [https://www.chaijs.com/plugins/chai-json-schema.](https://www.chaijs.com/plugins/chai-json-schema/)

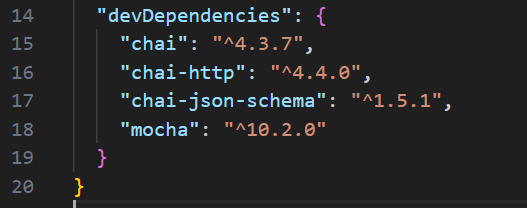
Okey sekarang kita mulai, pertama install chai-json-schema

npm install chai-json-schema

Setelah berhasil menginstall maka akan chai-json-schema sudah berhasil juga ditambahkan di dependencies



berhasil menginstall



dependencies

Disini kita menggunakan JsonSchema untuk memverifikasi respon dari endpoint get detail user. Tambahkan code berikut di file “user.spec.js”

it('get detail user', async () => {

const response = await new user().getUser(token, userId)

expect(response.statusCode).to.be.equal(200);

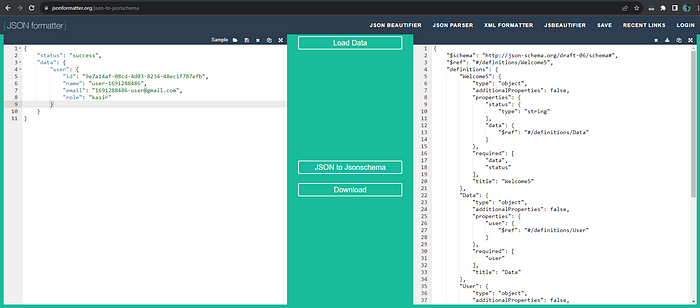
})

Nah code diatas hanya memverifikasi terkait statusCode saja. Untuk menerapkan chai-json-schema kita harus mengimport juga di file ini.

const chaiSchema = require('chai-json-schema');

chai.use(chaiSchema)

Kemudian copy response dari endpoint get detail user, kemudian convert dari json ke JsonSchema <https://jsonformatter.org/json-to-jsonschema> . Dari sini copy JsonSchema yang ada di kanan sisi ini.



json to JsonSchema

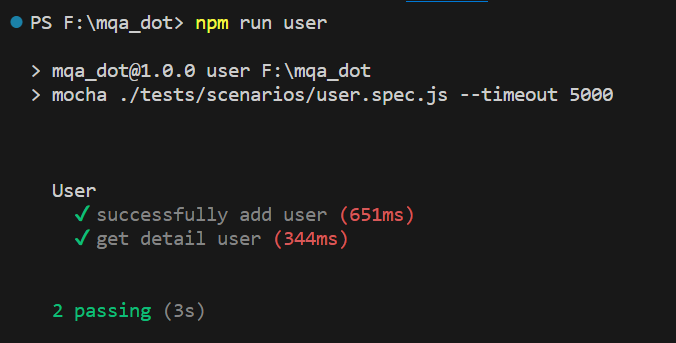
buat sebuah hook expect() baru seperti berikut.

expect(response.body).to.be.jsonSchema()

Selanjutnya paste JsonSchema kedalam method jsonSchema() sehingga it() detail user akan seperti berikut.

| it('get detail user', async () => {  const response = await new user().getUser(token, userId)  expect(response.statusCode).to.be.equal(200);  expect(response.body).to.be.jsonSchema({  "$schema": "http://json-schema.org/draft-06/schema#",  "$ref": "#/definitions/Welcome5",  "definitions": {  "Welcome5": {  "type": "object",  "additionalProperties": false,  "properties": {  "status": {  "type": "string"  },  "data": {  "$ref": "#/definitions/Data"  }  },  "required": [  "data",  "status"  ],  "title": "Welcome5"  },  "Data": {  "type": "object",  "additionalProperties": false,  "properties": {  "user": {  "$ref": "#/definitions/User"  }  },  "required": [  "user"  ],  "title": "Data"  },  "User": {  "type": "object",  "additionalProperties": false,  "properties": {  "id": {  "type": "string",  "format": "uuid"  },  "name": {  "type": "string"  },  "email": {  "type": "string"  },  "role": {  "type": "string"  }  },  "required": [  "email",  "id",  "name",  "role"  ],  "title": "User"  }  }  })  }) |
| --- |

Kemudian jalankan kembali test suite User. Jika telah menampilkan seperti ini berarti kita sudah berhasil melakukan assertion menggunakan JsonSchema.



npm run user

Btw udah jadi satu test suite nih!!

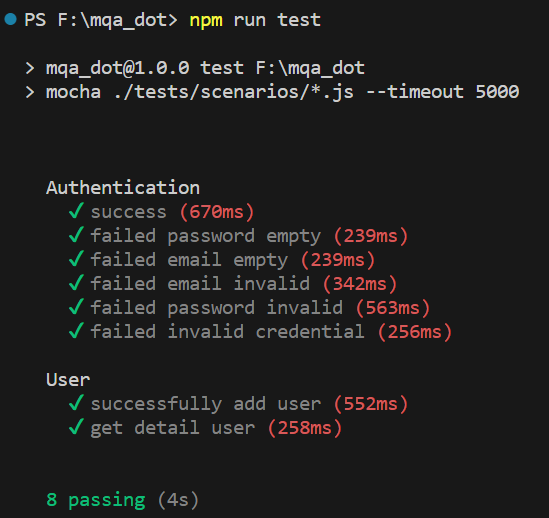
Untuk menjalankan semua test suite, kita hanya perlu menambahkan script di package.json seperti berikut.

"test": "mocha ./tests/scenarios/\*.js --timeout 5000"

Kemudian untuk menjalankannya, ketikkan di command

npm run test

Jika menampilkan seperti ini, maka kita sudah berhasil menjalankan seluruh test suite.



npm run test

Akhirnya kita sudah menyelesaikan tutorial untuk Part 3ini. Untuk repository github dari project diatas dapat diakses di link berikut [github](https://github.com/anisanursekararum/API-Automation-Mocha-Chai). Scenario diatas sangat bisa untuk dikembangkan lagi, apalagi masih banyak modul yang belum dibuat untuk automationnya.