Unit Testing & TDD

Gold - Chapter 8 - Topic 5

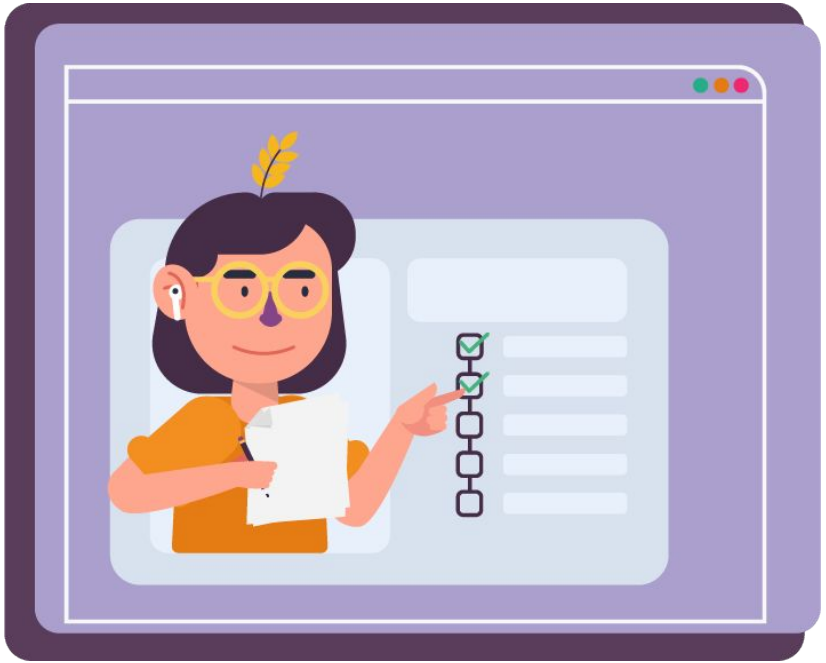
Selamat datang di Chapter 8 Topik 5 online course Fullstack Web dari Binar Academy!

Selamat datang di topic 5🥳

Nah, pada topic ini, kita akan membahas tentang konsep dan fungsi **Testing Driven Development (TTD)** serta setup untuk **testing** di environment Javascript dengan Jest dan Supertest. Markicusss~

Detailnya, kita bakal bahas hal-hal berikut ini:

- Pengenalan konsep dan jenis-jenis testing
- Pengenalan konsep TDD
- Melakukan testing di Javascript dengan Jest dan Supertest
- Integrasi Testing dengan Supertest



BINAR

Ketika kalian membuat sebuah coding kalian pasti akan mencoba atau nge–test apakah berhasil atau tidak? Nah, mirip banget sama TDD yang bantuin kita buat melakukan sebuah test agar resiko error atau bug dapat diminimalisir.

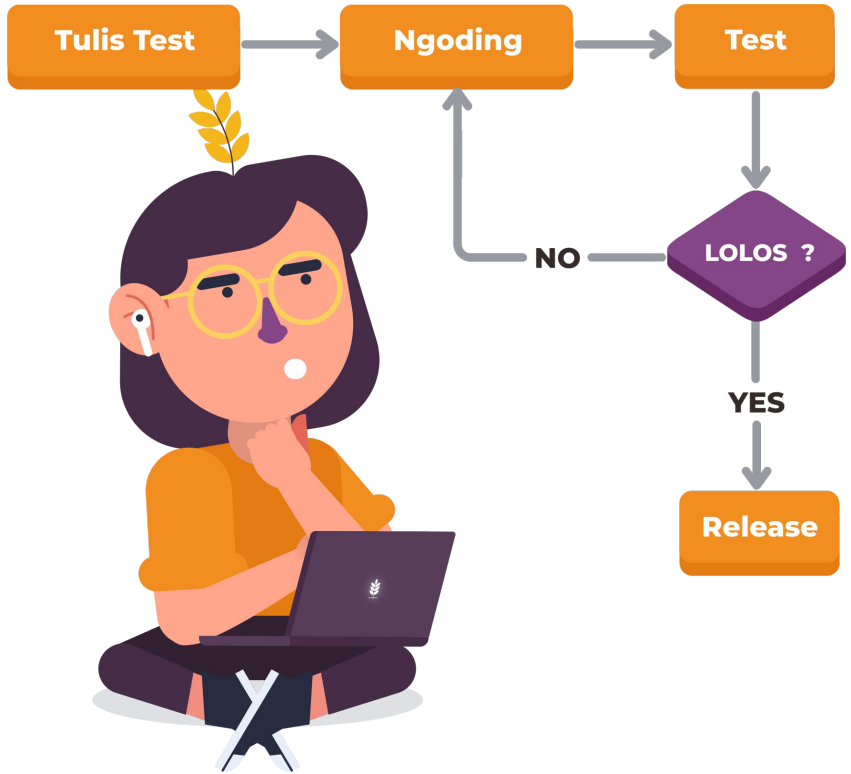
Udah kebayang dikit kan kayak gimana TDD itu? Lanjutttt kuy~

Test Driven Development

Dari pengalaman kalian membuat kode, pasti ga jauh dari yang namanya bug. Sampe-sampe kalian capek dan akhirnya harus healing dulu. Hehe

Nah, karena bug ini sesuatu yang ga terelakkan, terciptalah prinsip development TDD untuk mengatasi masalah bug ini. TDD tuh metode membuat aplikasi yang mana kita nulis test-nya terlebih dahulu baru ngoding. Pasti error dong, kan belum ada kode-nya?

Bener teman-teman, inti dari TDD ini adalah, **kita membuat kode berdasarkan test yang udah kita buat sampe akhirnya kode tersebut lolos test**.



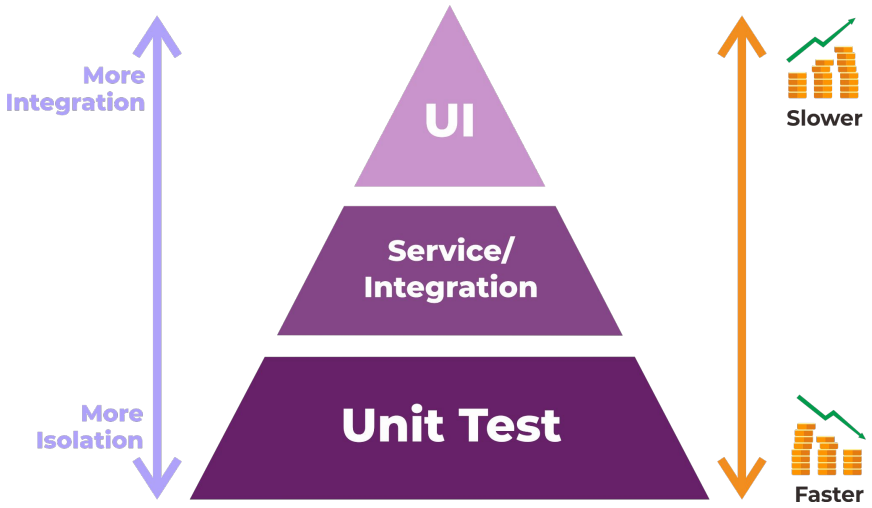
Tingkatan Testing

Mungkin kalian nanya, **kan udah ada QA Engineer, kenapa kita harus nulis test juga?**

Jadi gini teman-teman, testing itu ada tingkatan-tingkatannya, yaitu:

- **End to End Testing**
- **Integration Testing**
- **Unit Testing**

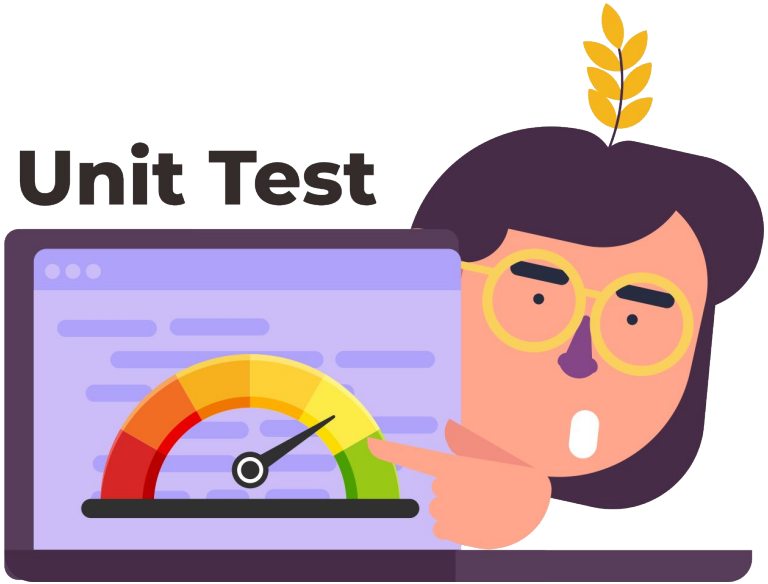
Nah, kita sebagai developer itu biasanya cuma nulis test sampe **unit testing** dan **integration testing** doang, sedangkan end to end testing itu biasanya masuk scope QA Engineer.



Unit Test

Unit test adalah **metode testing yang hanya berfokus pada satu unit aja dalam 1 test case**. Di dalam kode, apa yang kita test di dalam unit testing adalah class, dan function tanpa dependensi ke hal lain.

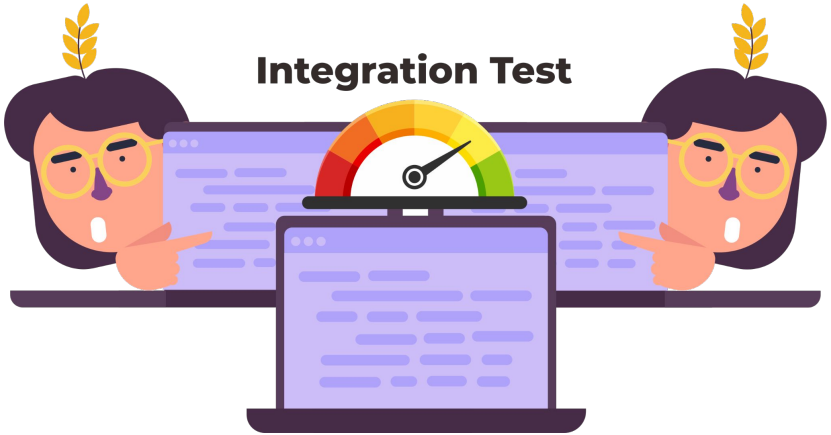
Contohnya kayak gini, kita ingin menulis unit testing untuk UserController, maka sangat wajib hukumnya kalo kita tidak mempedulikan implementasi dari User model, karena User model ini sudah menjadi unit yang berbeda dari UserController. Kita hanya peduli dengan possible value dari method-method yang kita panggil dari User model.



Integration Test

Inti Integration Test adalah **kita melakukan test integrasi antar unit**. Kalo konteksnya back-end, kita bakal test bagaimana back-end nge-handle request pada endpoint X.

Kalo di front-end kita bakal test gimana 1 komponen ini di-render di dalam 1 halaman dan sebagainya.



End-to-End Testing

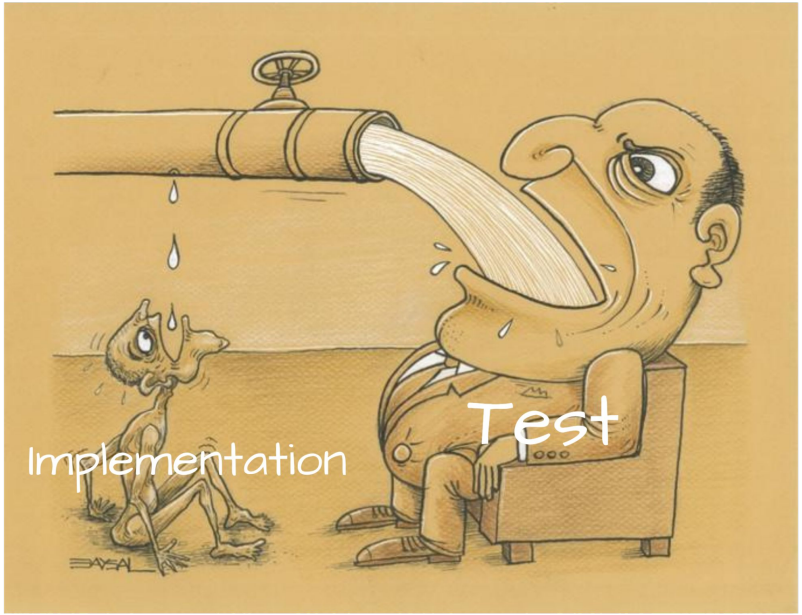
Jenis testing ini tuh intinya **kita ngetest sebagai user di dalam aplikasi kita**. Jadi kita ga boleh ngomongin konteks teknis lagi disini. Yang perlu kita lakukan adalah mendefinisikan secara spesifik parameter testingnya.

Sebagai contoh, **Jika aku membuka halaman Home, maka aku akan melihat teks "Hello World!"**.

Terus Implementasi TDD itu gimana sih?

Nah, biasanya ketika kita nerapin TDD, proses development kita bakalan berat di testing-nya dibanding implementasinya. Maka dari itu biasanya kalo kita implementasi TDD akan menambah beban kerja kita sebagai developer.

Tapi itu semua worth it kok, karena ketika kita mengembangkann aplikasi, kita bisa lebih yakin kalo kode kita ga nyenggol fitur-fitur lama, dan testing-nya pun akan jauh lebih cepat dibanding kita testing manual, apalagi kalo aplikasinya udah gede banget.



Untuk mengimplementasikan TDD, mending kita mulai dari gimana caranya nulis Test. Disini kita bakal belajar cara nulis test di Javascript yak~

Testing di Javascript

Nah untuk melakukan Testing di Javascript, biasanya orang-orang pada pake library yang namanya Jest**.**

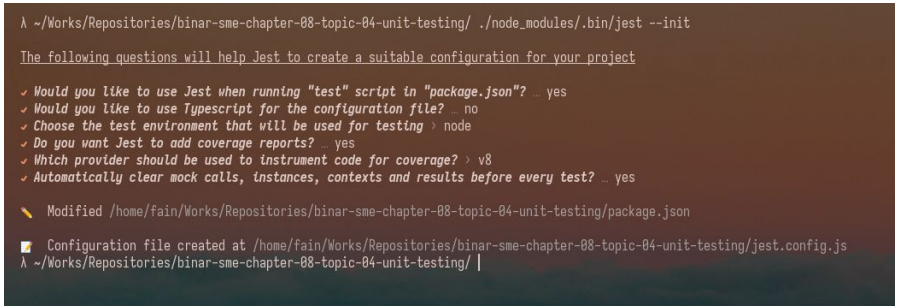
Jest ini berguna untuk **mendefinisikan test case, menjalankan test, dan merekap hasil test**.

Tanpa basa-basi langsung lanjut nge–setup Jest yuk~

Skuy, setup Jest

Untuk melakukan setup jest, kamu bisa jalanin perintah di bawah ini, dan isi jawabannya sesuai dengan gambar di samping.

```
npm install --save-dev jest
./node_modules/.bin/jest --init
```



Menulis Test

Nah, untuk **mengimplementasikan TDD, kamu perlu sebuah requirement terlebih dahulu.** Nah sebagai contoh, kita bisa pake requirement di bawah ini.

- Terdapat sebuah class bernama Dog
- Dog dapat menggongong dan mengeluarkan teks "Woof!" di dalam terminal
- Setiap Dog memiliki name

Dari requirement tersebut, kita bisa membuat test case-nya di dalam Jest. Seperti contoh di samping.

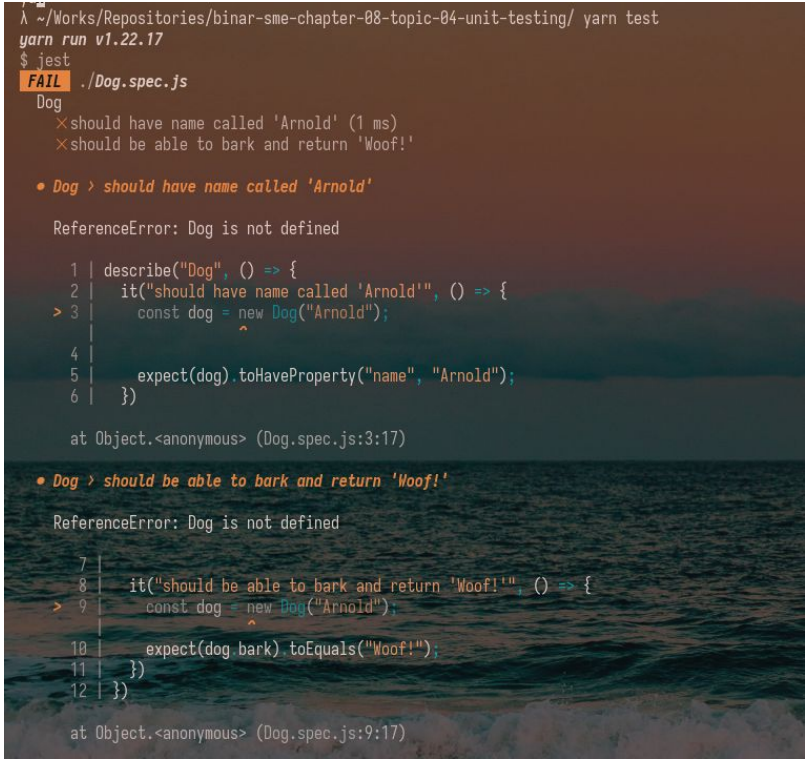
Oh iya, contoh di samping ini ditulis di dalam file **Dog.spec.js**.

```
describe("Dog", () => {
  it("should have name called 'Arnold'", () => {
    const dog = new Dog("Arnold");

    expect(dog).toHaveProperty("name", "Arnold");
  })

  it("should be able to bark and return 'Woof!'", () => {
    const dog = new Dog("Arnold");
    expect(dog.bark()).toEqual("Woof!");
  })
})
```

Menjalankan Test

Untuk menjalankan test, kamu cuma perlu jalanin perintah npm test atau yarn test, **Jest akan otomatis menjalankan file yang memiliki ekstensi .test.js atau .spec.js**.

Output test berdasarkan kode tadi akan seperti gambar di samping. Pasti Gagal, karena kita belum membuat class bernama Dog dan mengimpor-nya di dalam test.



Menulis Implementasi

Nah, sekarang kamu tinggal tulis implementasinya dengan membuat file bernama **Dog.js** dan isi file tersebut dengan implementasi yang sesuai dengan test case.

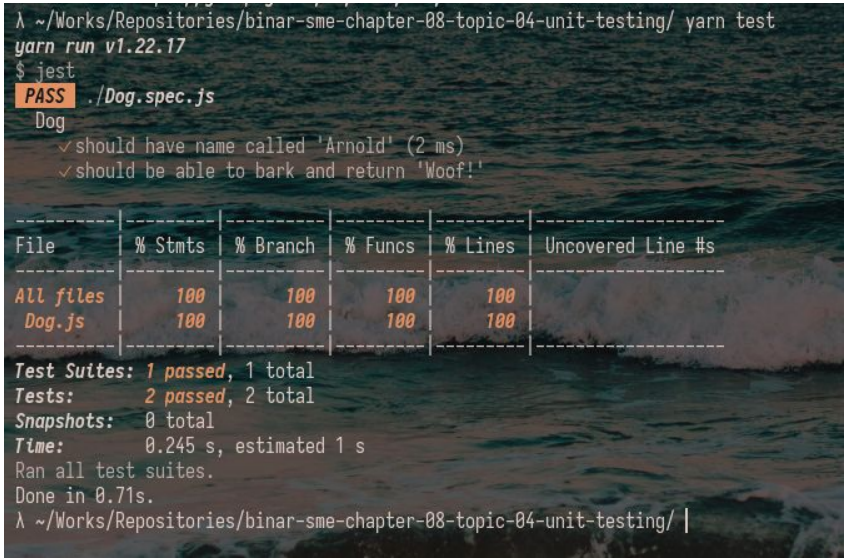
Setelah kamu selesai menulis implementasinya, pastikan class tersebut di-impor di dalam file test.

```
class Dog {
  constructor(name) {
    this.name = name
  }

  bark() {
    return "Woof!";
  }
}

module.exports = Dog;
```

Menjalankan Test Lagi

Setelah kamu selesai dengan implementasi, jangan lupa jalankan test dengan menggunakan **yarn test** atau **npm test** untuk ngetes apakah implementasimu udah bener atau belum.

Jika implementasinya sudah benar, kurang lebih output-nya akan seperti kode disamping. Kode ini bisa kamu liat di repository ini.



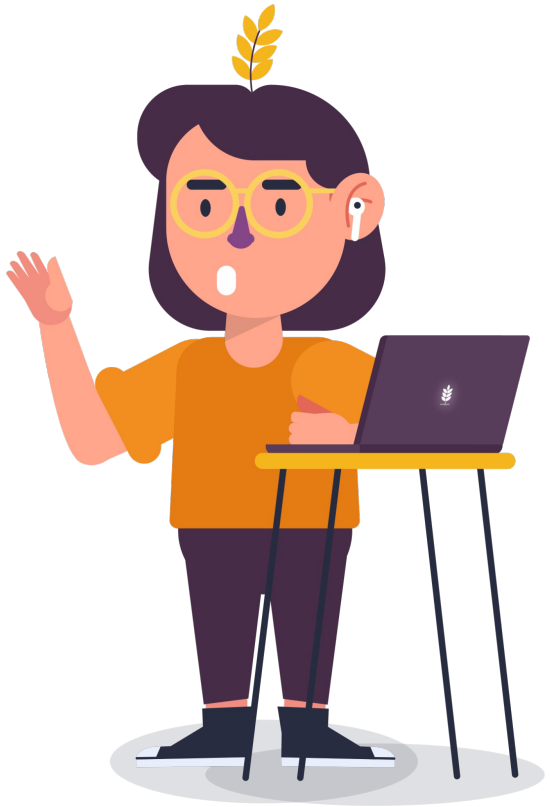
Itu tadi masih testing sederhana, karena kita hanya nge-test 1 class aja.

Di slide berikutnya akan terdapat link ke repository, bagaimana penerapan testing di dalam repository back-end dan front-end. Letsgoooo~

Testing di Back-end

Seperti yang sudah di-mention di atas, dalam **konteks back-end kita akan nge-test endpoint, dan unit testing pada umumnya.**

Contoh implementasi-nya dapat kamu lihat di repository ini.



Hal yang perlu di-test di backend

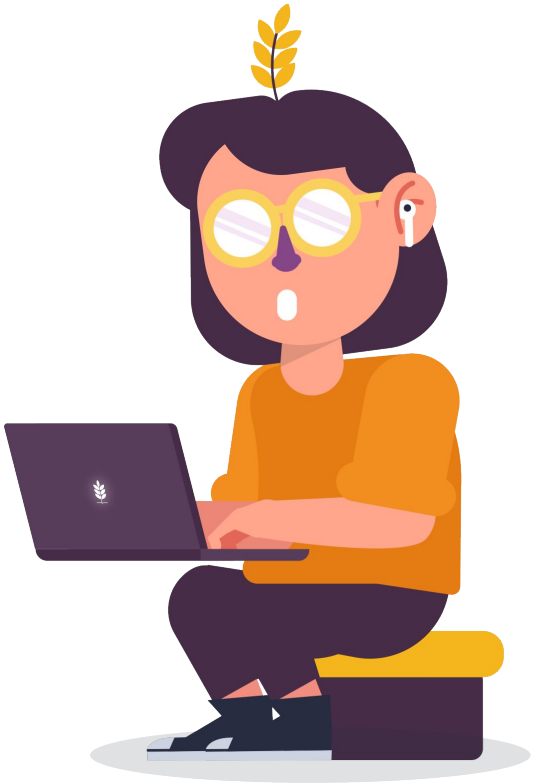
Karena back-end itu soal logika bisnis, yang biasanya dijalankan berdasarkan request dari client pada sebuah endpoint, maka dari itu, di back-end tuh biasanya yang di-test hal-hal ini:

- Endpoint (Integration Test)
- Controller (Unit Test)
- Model (Unit Test)

Testing di Front-end

Seperti yang sudah disebut di-atas, dalam **konteks front-end kita akan ngetest component, dan unit testing pada umumnya.**

Contoh implementasi-nya dapat kamu lihat di repository ini.



Saatnya kita

Quiz!

BINAR

1. Manakah flow yang tepat ketika kita menerapakan TDD?

   A. Ngoding dulu, baru tulis test dan ngejalanin test sampe lolos.

   B. Tulis test dulu baru, jalanin test, baru ngoding sampe lolos test.

   C. Jalanin test dulu, baru ngoding, baru nulis test.

BINAR

1. Manakah flow yang tepat ketika kita menerapakan TDD?

A. Ngoding dulu, baru tulis test dan ngejalanin test sampe lolos.

B. Tulis test dulu baru, jalanin test, baru ngoding sampe lolos test.

C. Jalanin test dulu, baru ngoding, baru nulis test.

Flow yang benar di dalam TDD adalah, kita nulis test-nya dulu, terus jalanin test-nya pasti gagal, lalu baru implementasi sampe lolos test.

BINAR

2. Manakah urutan yang tepat dari test berdasarkan level isolasinya dari terendah sampai tertinggi?

A. End to End Testing, Integration Testing, Unit Testing

B. End to End Testing, Unit Testing, Integration Testing

C. Unit Testing, Integration Testing, End to End Testing

BINAR

2. Manakah urutan yang tepat dari test berdasarkan level isolasinya dari terendah sampai tertinggi?

A. **End to End Testing, Integration Testing, Unit Testing**

B. End to End Testing, Unit Testing, Integration Testing

C. Unit Testing, Integration Testing, End to End Testing

End to End testing hanya mencakup gambaran besar dari suatu fitur, akan sangat sulit untuk melakukan test untuk case yang sangat spesifik.

BINAR

3. Jika kita ingin melakukan Unit Testing terhadap Controller, apakah kita perlu menggunakan Model yang asli untuk kita gunakan sebagai dependensi Controller di dalam test?

A. Tidak, karena kita tidak akan memanggil model sama sekali

B. Tidak, karena kita hanya peduli possible output dari method model yang kita panggil

C. Iya, karena controller sangat dependen dengan model.

BINAR

3. Jika kita ingin melakukan Unit Testing terhadap Controller, apakah kita perlu menggunakan Model yang asli untuk kita gunakan sebagai dependensi Controller di dalam test?

A. Tidak, karena kita tidak akan memanggil model sama sekali

B. Tidak, karena kita hanya peduli possible output dari method model yang kita panggil

C. Iya, karena controller sangat dependen dengan model.

Di dalam test, sangat dianjurkan untuk menggunakan Mock, karena kita hanya mengetes 1 unit saja yang kita isolasi tanpa adanya dependensi di-luar unit tersebut, jadi test kita akan terfokusi disitu.

BINAR

4. Di dalam Back-end, hal apa saja yang menjadi cakupan dari TDD untuk ditest?

A. Endpoint, & Unit

B. View

C. Dokumentasi

BINAR

4. Di dalam Back-end, hal apa saja yang menjadi cakupan dari TDD untuk ditest?

A. Endpoint, & Unit

B. View

C. Dokumentasi

Hal yang paling penting untuk di test di back-end adalah endpoint & masing-masing unit-nya. Karena itu adalah inti dari fitur back-end.

BINAR

5. Di dalam front-end, apa saja yang perlu kita tes sebagai cakupan dari TDD?

A. Third Party Library

B. Back-end API

C. Component, dan Custom Function

BINAR

5. Di dalam front-end, apa saja yang perlu kita tes sebagai cakupan dari TDD?

A. Third Party Library

B. Back-end API

C. Component, dan Custom Function

Kita tidak perlu ngetes Third Party Library dan Back-end API, karena itu masuk ke scope BE dan Third Party Provider.

Referensi

- https://rahmanfadhil.com/test-express-with-supertest/
- https://jestjs.io/docs/expect
- https://www.youtube.com/watch?v=3e1GHCA3GP0

Terima Kasih!

Next Topic

loading...

BINAR