Rakamin Academy

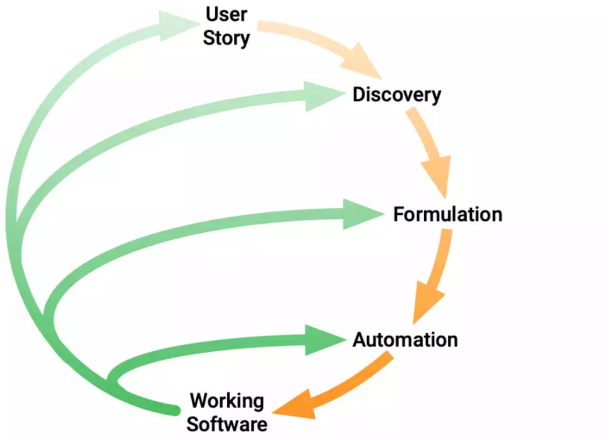
Reading Material

Memahami Testing Principles - Behavior Driven Development (BDD)

READING

Definisi

Behavior Driven Development atau BDD adalah sebuah metodologi pengembangan aplikasi yang menekankan pada kebutuhan bisnis untuk menyelesaikan masalah user. BDD sebetulnya dikembangkan dari metode pendahulunya yaitu TDD atau Test Driven Development dan BDD menggunakan sebuah bahasa khusus atau yang dikenal dengan domain-specific language dengan sintaks yang sudah ditetapkan sebagai panduan untuk menuliskan test.



Jika TDD bertujuan untuk membuat kode yang modular dan rapi yang mampu memenuhi kebutuhan aplikasi tanpa membuat kode aplikasi *bloated.* TDD lebih fokus pada penulisan kode yang harus lolos tes daripada langsung fokus pada inti kebutuhan sebuah aplikasi.
BDD mencoba untuk menambahkan solusi baru yang nantinya akan memudahkan dalam penulisan tes dengan menggunakan domain-specific language. Hal ini membuat kita dapat menuliskan test yang spesifik sesuai user stories terhadap fitur-fitur sebuah aplikasi.

Perbedaan TDD dan BDD

TDD	BDD
Fokus pada perilaku dan kebutuhan aplikasi	Fokus pada implementasi fitur ke aplikasi dari sisi kode melalui unit test
Kolaborasi melibatkan developers, customer, tester	Hanya melibatkan developers

Diawali dengan scenario	Diawali dengan test case
Tim metodologi	Praktek pengembangan aplikasi
Menggunakan bahasa yang sederha dan bahasa inggris	Menggunakan bahasa pemrograman yang sama atau mirip dengan bahasa pemrograman untuk aplikasi
Beberapa tools yang digunakan Cucumber, Dave, JBehave, Spec Flow, Concordian, BeanSpec etc.	Beberapa tools yang digunakan JBehave, JDave, Cucumber, Spec Flow, BeanSpec, FitNesse etc.

Implementasi BDD

Tool yang menggunakan BDD adalah Cucumber dengan domain-specific language bernama Gherkin. Gherkin adalah bahasa berorientasi seperti YAML dan Python yang memang dibuat supaya mudah dibaca dan dipahami oleh manusia. Pada penulisan Gherkin, setiap baris disebut langkah, dirangkai dengan sintaks tertentu yang sudah ditentukan dan setiap fitur aplikasi akan dibalut sebuah file dengan ekstensi feature. Tujuannya adalah untuk mendefinisikan test yang tidak ambigu untuk setiap fitur sehingga dapat diuji secara tepat.

Setelah mengenal Cucumber dan Gherkin maka kita bisa memulai membuat test untuk fitur-fitur aplikasi. Poin penting adalah setiap tes mengikuti format tetap dari Gherkin dan berbahasa Inggris misalnya: ***Given*** *some condition,* ***When*** *something happens,* ***Then*** *you see a result*. Gherkin dimaksudkan agar dapat dibaca manusia, tetapi, seperti format seperti XML, Gherkin tidak selalu mudah dibaca.

Contoh Penulisan BDD Scenario dengan Gherkin

Berikut ini adalah contoh cara menulis fitur BDD untuk tombol login.
Fitur Aplikasi : Login Button
Deskripsi : User dapat login dengan memasukkan kredensial mereka dan mengklik tombol login. Pengguna harus memasukkan username dan password yang valid agar sukses login dan mengarahkan user ke halaman beranda.

Dari fitur dan deskripsi diatas maka jika dituliskan menggunakan bahasa Gherkin untuk Cucumber tool akan menjadi seperti dibawah berikut ini:
Scenario: User able login and direct to homepage
Given the user is on a login page with form fields username, password and login button
And the user has entered username “rakamin”
And the user has entered password “rakamin123”
When the user click the login button
Then the user will be directed to a homepage and a login success show up

Scenario: User failed to login using wrong password
Given the user is on a login page with form fields username, password and login button
And the user has entered username “rakamin”
And the user has entered password “rakaminfailed”
When the user click the login button
Then the form login is displayed again and a login error message shows up

Dari kedua contoh scenario test di atas maka kita dapat melihat ada beberapa sintaks yang digunakan yaitu Scenario, Given, When, And, Then. Tes juga didefinisikan dengan jelas dalam setiap baris berupa step sehingga mudah dimengerti.

Manfaat BDD

BDD fokus pada kebutuhan user dan perilaku dari aplikasi. Hal terpenting dari penggunaan BDD adalah konsisten dalam menentukan test yang sesuai dengan fitur-fitur aplikasi yang sudah direncanakan. Jika aplikasi sudah lolos test maka aplikasi tersebut mengimplementasikan fitur yang benar sesuai dengan kebutuhan user dan diharapkan mampu memberikan problem solving.
Jika sebuah tim sudah menggunakan Test-Driven Development (TDD) atau Acceptance Test-Driven Development (ATDD), berikut manfaat yang dapat mereka manfaatkan dari Behavior-Driven Development:

1. BDD mampu memberikan panduan yang lebih tepat dan terorganisir antara developer, tester dan stakeholders lainnya sehingga menghasilkan kolaborasi yang lebih efektif.
2. Syntax yang digunakan dalam BDD dirancang untuk mudah dipahami manusia sehingga siapapun dapat membaca dan memahami scenario step meskipun kemampuan antar team member berbeda-beda.
3. Beberapa BDD tool juga sudah menyediakan fitur untuk auto generate end-user documentation dari BDD features, sehingga akan mempersingkat waktu dan meningkatkan kualitas dari pengujian.

BDD atau Behavior Driven Development juga disebut sebagai NLP exercise, NLP adalah Neuro Linguistic Programming yang berarti bahwa BDD dimaksudkan untuk mengganti istilah aplikasi lolos pengujian menjadi pengujian untuk mengetahui perilaku yang benar dari sebuah aplikasi.

Tantangan BDD

1. BDD tidak terlalu cocok untuk agile development dan cenderung cocok dengan waterfall model. Hal ini karena waterfall model akan mengetahui requirement dari awal dan itu bisa dipastikan untuk kemudian dibuat test scenario dengan contohnya Gherkin pada Cucumber. Namun pada agile development bisa jadi sebuah fitur akan secara konstan berubah dan hal ini mengakibatkan perubahan test scenario secara terus menerus.
2. Meneruskan poin 1, namun BDD dapat dimanfaatkan sebagai maintenance test pada agile development seperti sanity atau regression testing.
3. Harus belajar menuliskan test scenario dengan formula bahasa Gherkin.
4. Tantangan paling berat mungkin pada penulisan test case yang spesifik. BDD akan menghasilkan test yang bagus apabila jika kita menulis scenario dengan tepat sasaran dan efektif. Jangan sampai kita melewatkan bagian critical dari sebuah fitur hanya karena kita kurang menuliskan step atau berbeda cara pandang dengan developer. Kuncinya adalah tester, developer dan stakeholders harus tetap berkomunikasi untuk menjaga BDD tetap sesuai harapan.