

Working As A Back Engineer

Silver - Chapter 1 - Topic 5

**Selamat datang di Chapter 1 Topic 5
online course Back End Java dari
Binar Academy!**



Hello everything, kita ketemu lagi!



Setelah mempelajari Operator, Conditional & Looping, pada topic ini, kita bakal mengelaborasi tentang **Working as Back End Engineer.**

Kita belajar mulai dari konsep Agile dan Waterfall, perbedaan Agile dan Waterfall, sampai cara bikin fitur berdasarkan User Story pada project.

Yaudin langsung aja kita intip slide materi~



Detailnya, kita bakal bahas hal-hal berikut ini:

- Perbedaan Agile dengan Waterfall
- Identifikasi requirement berdasarkan Functional Specification Document dan User Stories
- Cara membuat sebuah feature berdasarkan FSD
- Bagaimana membuat fitur berdasarkan User Story pada project yang menggunakan Agile



Pada pertandingan olahraga, ada 2 tim populer yang dipertemukan untuk tahu mana yang juara.

Di Back End, kita juga bakal cari tahu mana yang terbaik dari 2 metode yang paling sering dipakai, yaitu:

Agile vs Waterfall.



Sebagai awal, kita harus tahu dulu pekerjaan Back End Engineer itu kayak gimana~

Pekerjaan seorang backend engineer tentunya nggak cuma ngoding aja, nih.

Seorang backend engineer juga harus **mampu menganalisa user requirement** karena code yang dikembangkan nggak mungkin dibikin dengan sesuka hati.

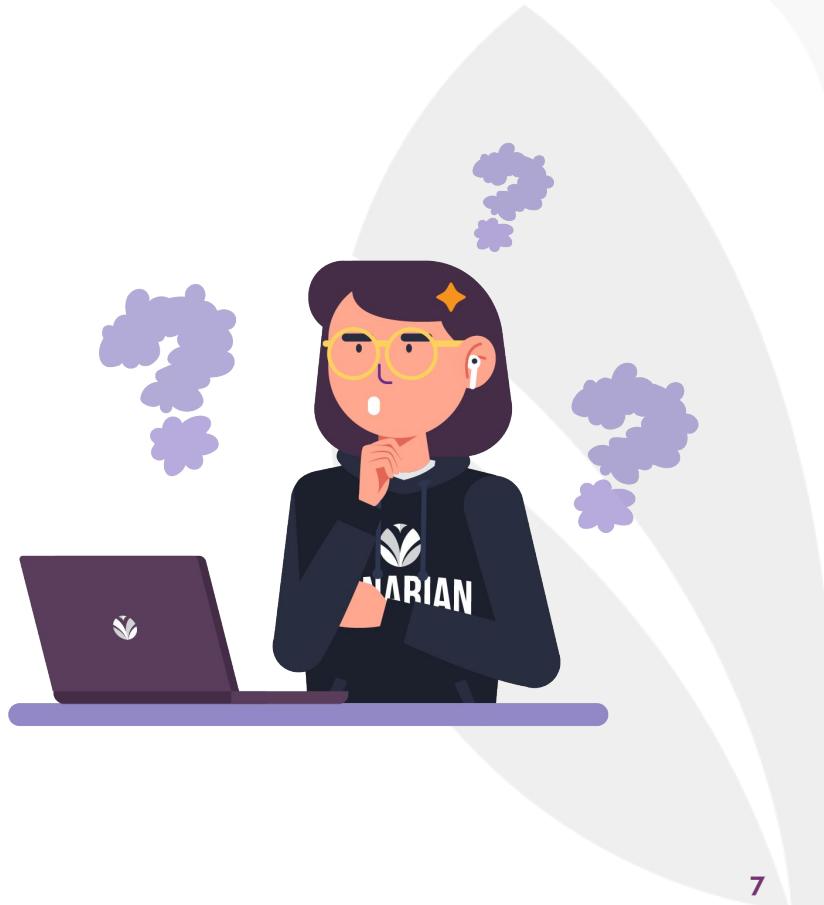
Tentunya business logic yang dibikin ini harus mendukung requirement yang udah diberikan.



Pada topic ini kita bakal bahas gimana developer bekerja sesuai dengan requirement yang diberikan pada methodology Agile dan Waterfall.

“Agile? Waterfall? Itu apa, sih?”

Tenang, Bun. Biar ada gambaran, kita bahas satu-satu ya!



Jadi, apa itu metode Agile dan Waterfall?

Metode project management Agile dan Waterfall merupakan metode yang paling sering dipakai dalam me-manage project SDLC.

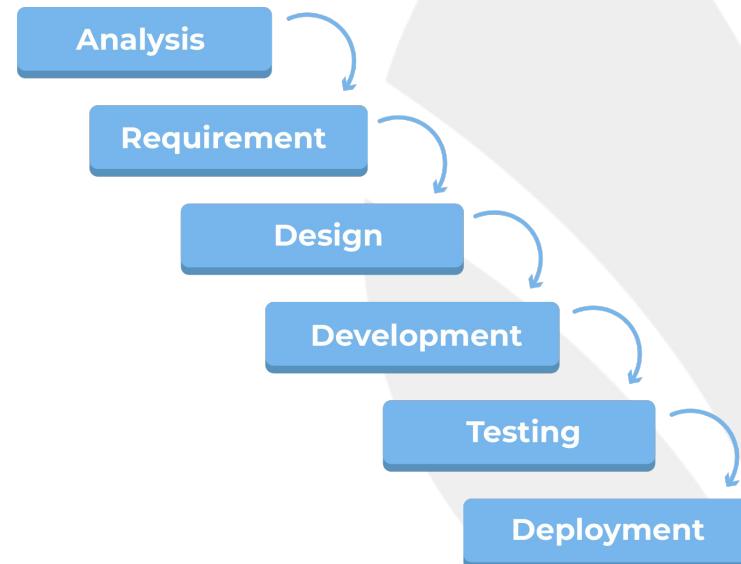
Detailnya kayak gimana, langsung aja next slide~



Metode Waterfall alias metode air terjun~

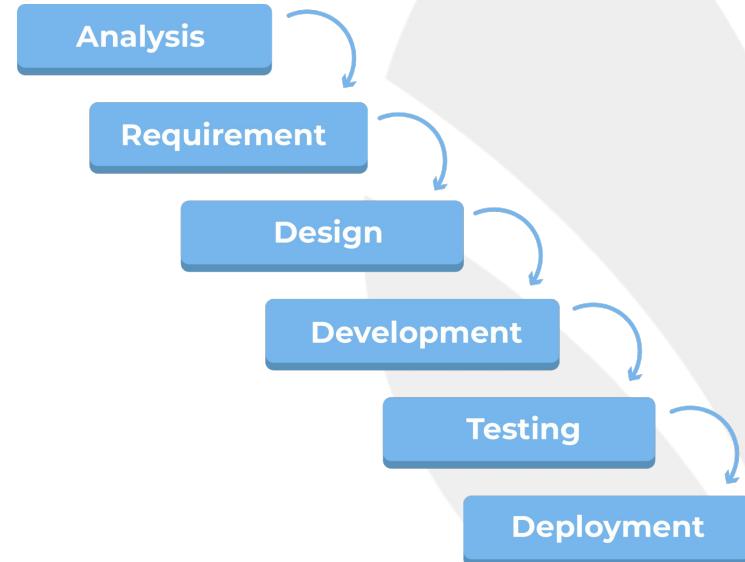
Merupakan pendekatan dalam project SDLC yang menggunakan siklus lama.

Metode ini menekankan pada **fase yang berurutan dan sistematis**.



Kayak yang kamu lihat di samping, metode waterfall ini prosesnya mirip kayak air terjun yang mengalir.

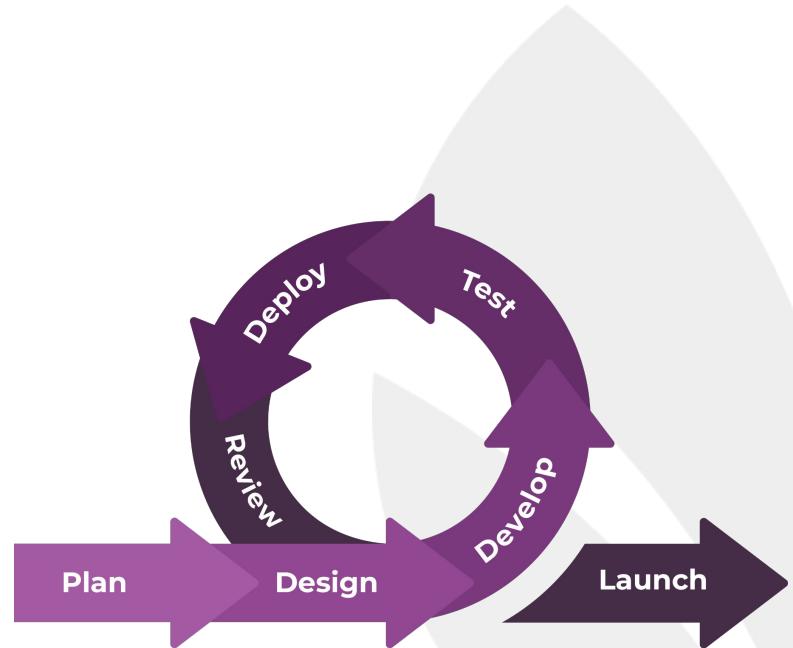
Jadi, setiap tahap dikerjakan secara berurutan mulai dari atas hingga ke bawah.



Metode Agile yang mirip kayak roda berputar~

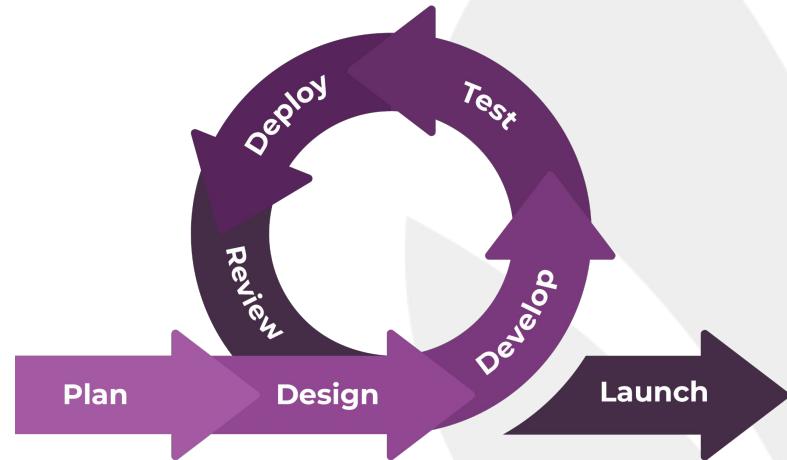
Agile merupakan pendekatan project SDLC yang didasarkan pada **proses penggeraan yang dilakukan secara berulang**.

Pada metode ini, aturan dan solusi yang disepakati dilakukan dengan kolaborasi antar tiap tim secara terorganisir dan terstruktur.



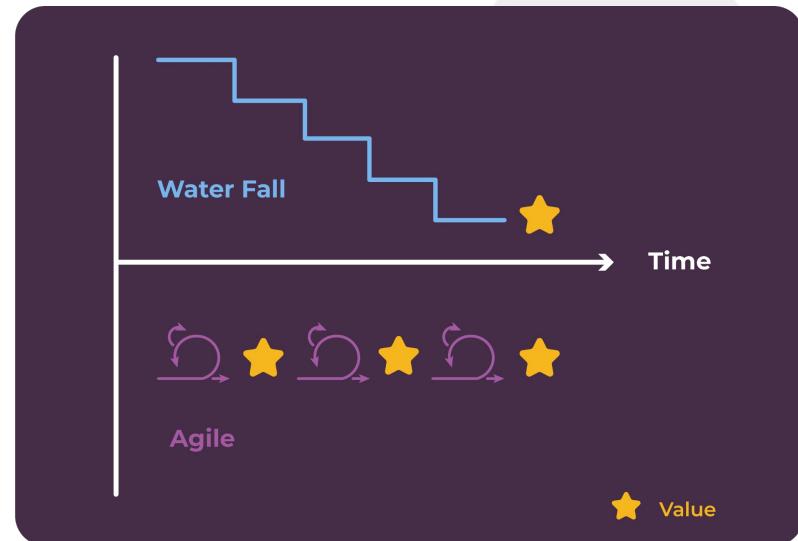
Selain itu, Agile butuh adaptasi yang cepat dalam mengatasi setiap perubahan.

Nilai terpenting dari Agile development ini adalah memungkinkan sebuah tim dalam mengambil keputusan dengan cepat, kualitas dan prediksi yang baik, serta punya potensi yang baik dalam menangani setiap perubahan.



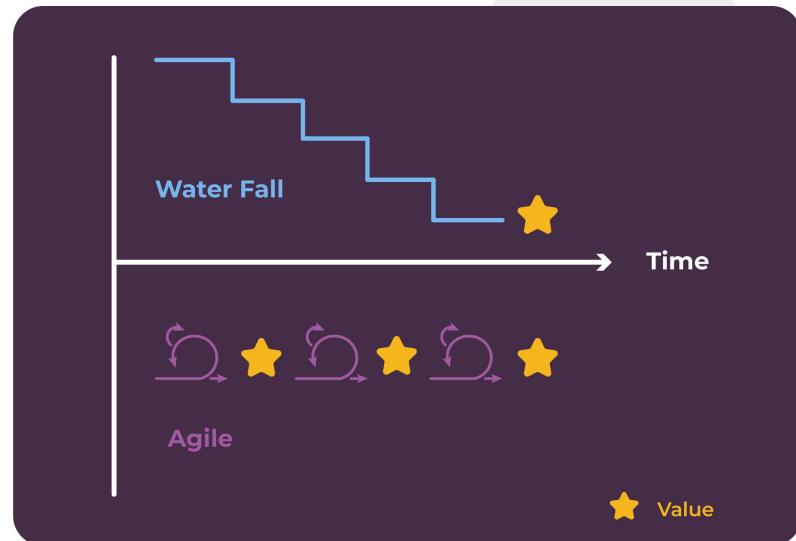
Perbedaan keduanya ada di iterasi siklusnya, bestie~

Di Waterfall, iterasi siklus tersebut berlangsung dalam satu waktu yang lama, sedangkan siklus di Agile berlangsung dengan iterasi yang lebih kecil.



Misalnya nih, dalam periode waktu 3 bulan di metode Waterfall cuma terjadi satu siklus, kalau pakai Agile, maka dalam 3 bulan bisa jadi 3 siklus.

Hal ini tentunya bakal mempengaruhi gimana seorang engineer bekerja, Sob.



“Kasih tahu detail pengaruhnya, boleh?”

Boleh, boleh, boleh.

Berikut pengaruh agile dan waterfall dalam penerapannya:

- Waterfall bakal bekerja sangat baik buat lingkup kerja yang udah jelas.

Sedangkan Agile cocok dipakai buat project yang punya ketidakpastian yang tinggi.

	Water Fall	Agile
Project Scope	Fixed	Variable
Organizations	Departments	Teams
Client Involvement	At the beginning	Frequent collaboration
Documentation	Comprehensive	Only what is build



- Metode Waterfall biasanya melibatkan tim besar. Oleh karena itu, koordinasinya jadi cenderung lambat.
Tapi di sisi lain, Agile bisa bekerja dengan tim kecil. Itulah kenapa koordinasinya lebih cepat.👍

	Water Fall	Agile
Project Scope	Fixed	Variable
Organizations	Departments	Teams
Client Involvement	At the beginning	Frequent collaboration
Documentation	Comprehensive	Only what is build

- Metode Waterfall cuma melibatkan pelanggan di tahap awal proyek.

Pada metode Agile, keterlibatan pelanggan hampir bisa dijumpai di semua tahapan proyek.

	Water Fall	Agile
Project Scope	Fixed	Variable
Organizations	Departments	Teams
Client Involvement	At the beginning	Frequent collaboration
Documentation	Comprehensive	Only what is build



- Karena Waterfall punya waktu yang lebih panjang, biasanya dokumentasi dari setiap prosesnya lebih baik dibandingkan Agile.

	Water Fall	Agile
Project Scope	Fixed	Variable
Organizations	Departments	Teams
Client Involvement	At the beginning	Frequent collaboration
Documentation	Comprehensive	Only what is build

Sipp deh! Udah mulai kebayang tentang metode Agile dan Waterfall.

Selanjutnya kita bakal intip-intip materi **Functional Specification Document**.

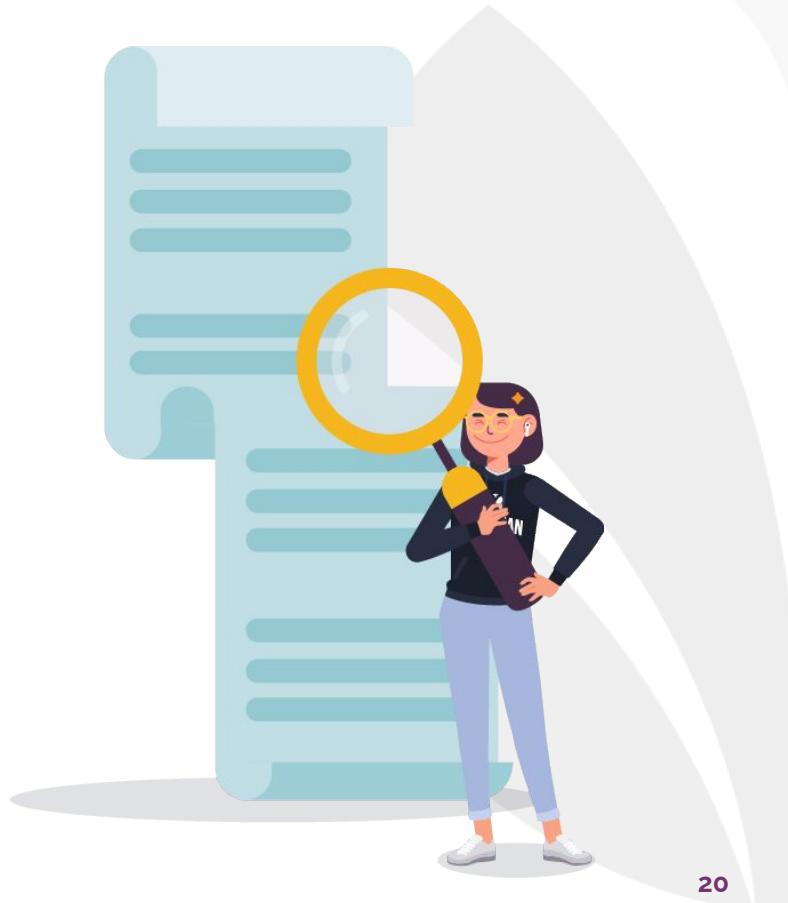


Dokumen dalam metode Waterfall~

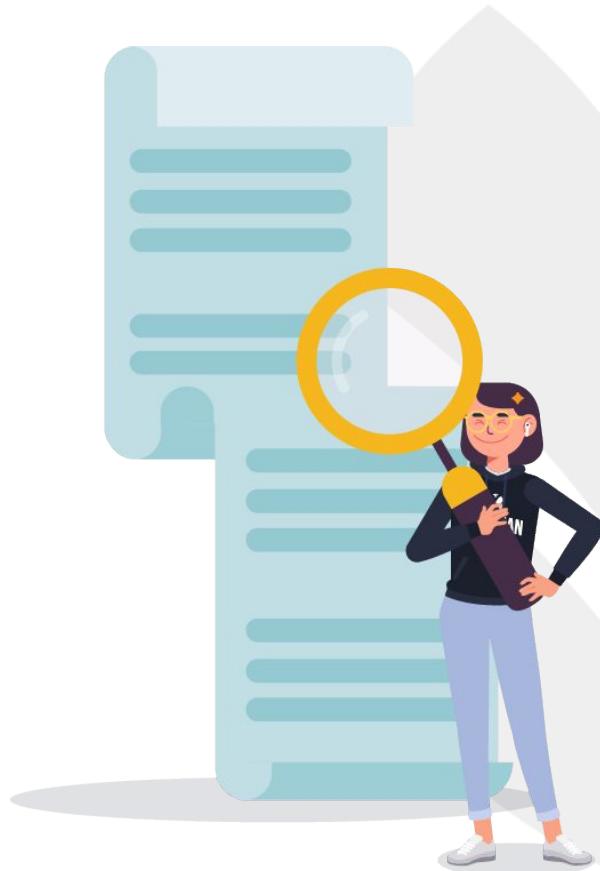
Pada metode Waterfall, banyak banget dokumen yang dilibatkan selama project. Dokumen tersebut antara lain:

- **Business Requirement Document (BRD)**, biasanya berisi tentang business rule dan goals dari business tersebut.

Document ini dibuat oleh Tim Product yang nantinya bakal dianalisis oleh Business Analyst dan diolah jadi sebuah FSD.

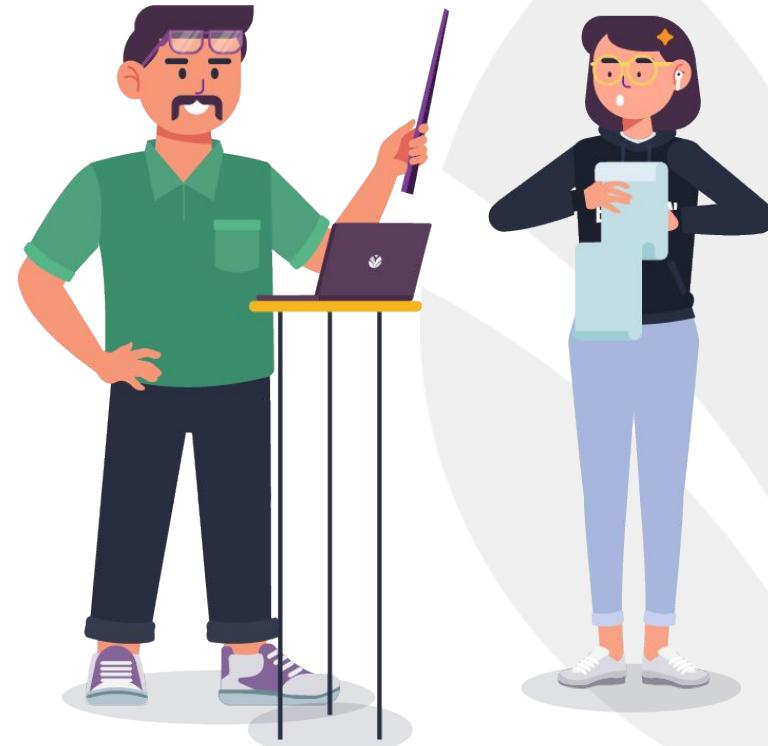


- **Functional Specification Document (FSD)**, yang nantinya bakal diolah jadi sebuah TSD sama Engineer
- **Technical Specification Document (TSD)**



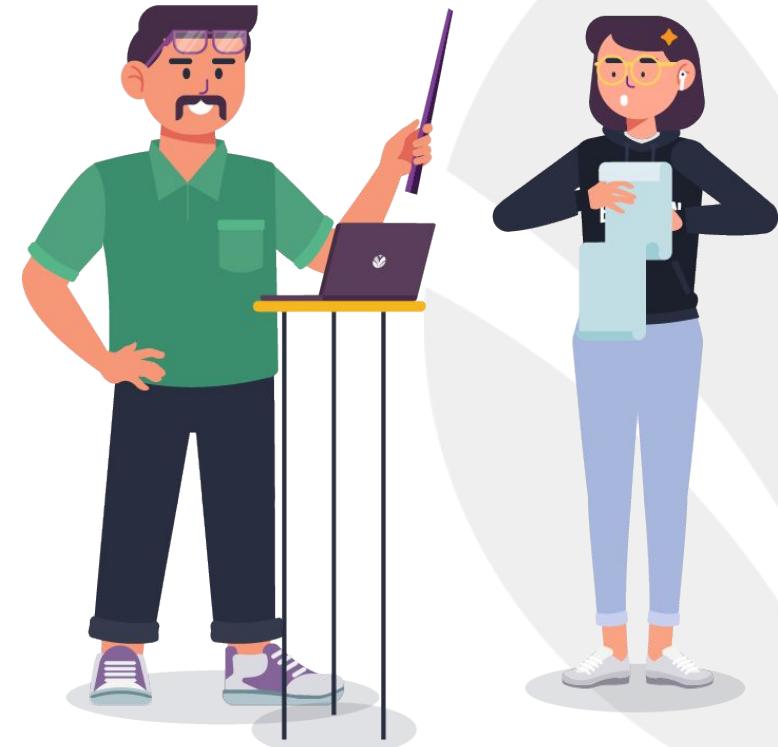
“Sebenarnya FSD itu apa?”

Functional specification document (FSD) adalah dokumen yang berisi **spesifikasi fungsional dari fitur atau modul sebuah software** yang disusun oleh seorang Business Analyst.



FSD biasanya berisi definisi modul dan aktivitas yang dilakukan.

Nggak cuma itu, bisa juga berisi Use Case Diagram, Role, Business Rules, Activity Diagram dan penjelasan dari Activity Diagram, Prototype UI dan masih banyak lagi~



Jadi, FSD tuh ibarat istilah yang bilang “sekali dayung, dua tiga pulau terlampaui~”

Soalnya gini, beberapa dokumen yang biasanya ada di Technical Specification Document, ada juga di FSD.

Dokumen yang ada di Technical Specification itu tuh kayak ERD, API contract, spesifikasi resource, dan performance specification.

Iya, kamu bisa ketemu dokumen itu semua di FSD.

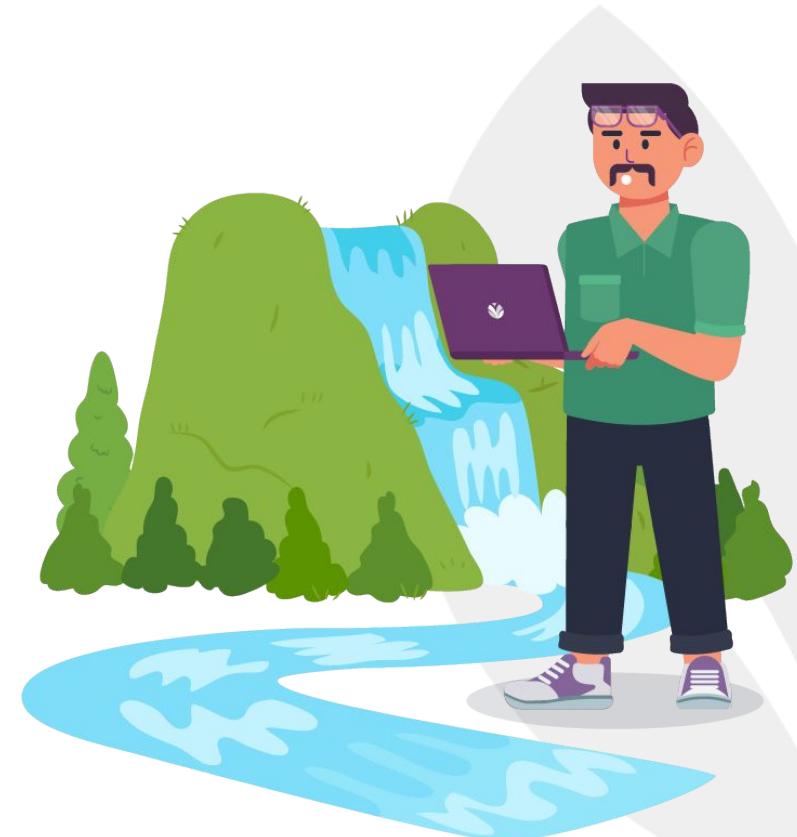


Dengan FSD, seorang engineer di metode Waterfall diarahkan untuk menganalisis requirement dari software.

Kalau udah dianalisis terus engineer tersebut bakal mulai merancang system secara teknikal dan melakukan coding.

Kamu bisa lihat contoh sebuah template dari sebuah FSD melalui referensi berikut, ya.

Functional Specification Document



Tadi kita udah menyinggung sedikit tentang Use Case. Kamu tahu arti dari Use Case?

So, Use Case merupakan **diagram yang paling sering digunakan pada FSD**.



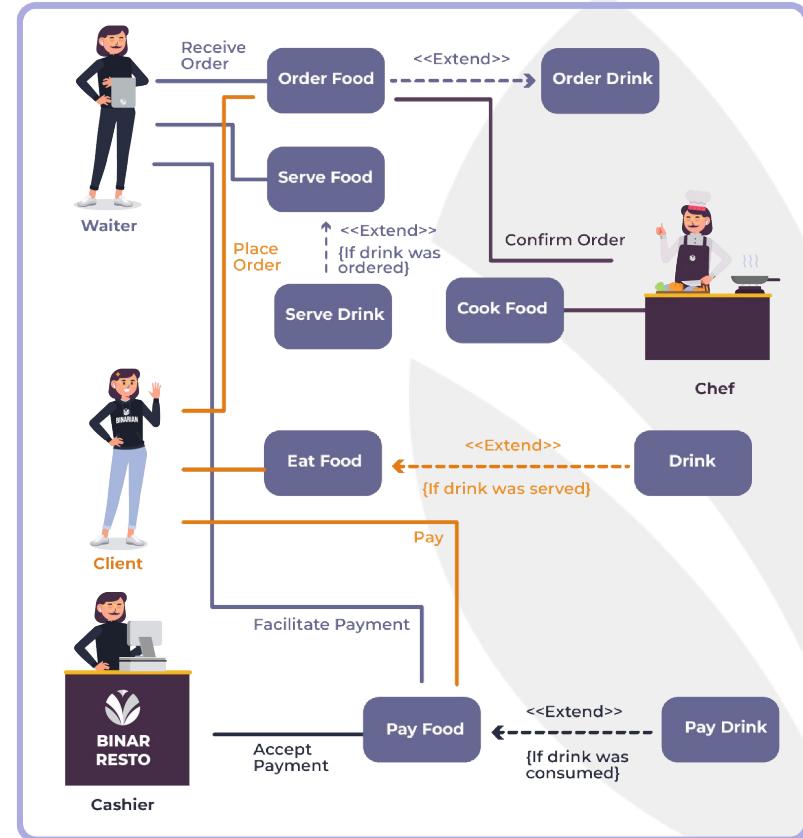
Use case diagram adalah satu dari berbagai jenis diagram **UML (Unified Modelling Language)** yang menggambarkan hubungan interaksi antara sistem dan aktor, serta bisa mendeskripsikan tipe interaksi antara si pengguna sistem dengan sistem yang digunakannya.



Berikut contoh use case diagram!

Kalau kita perhatikan nih, di contoh tersebut ada empat role yang masing-masing dihubungkan dengan garis.

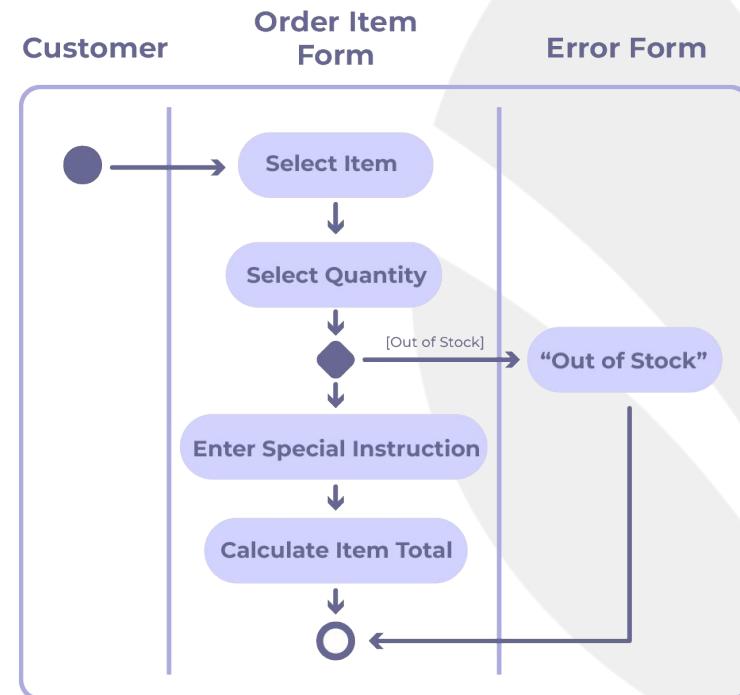
Garis tersebut merupakan gambaran sebuah interaksi.



Selain Use Case Diagram, ada juga Activity Diagram~

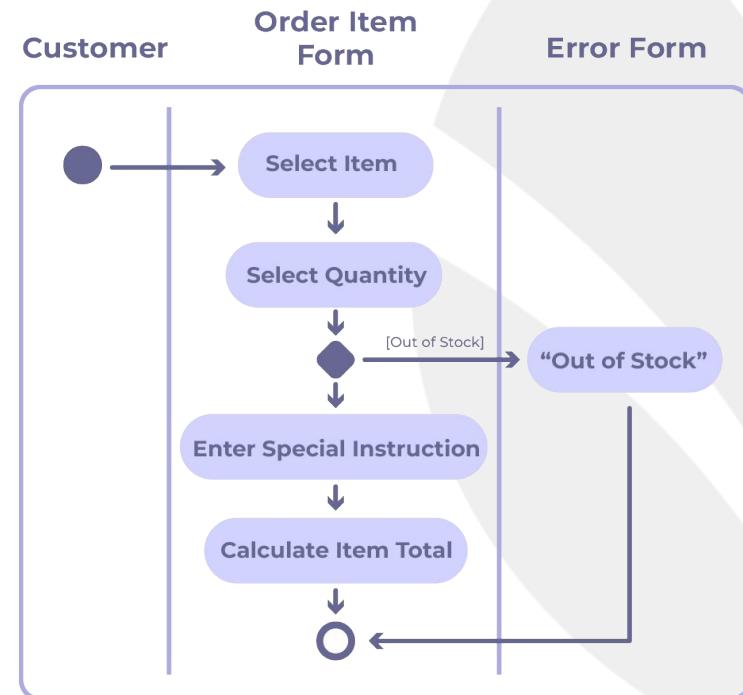
Activity diagram merupakan diagram kedua yang paling umum dipakai pada sebuah FSD.

Ssstt.. si Activity Diagram ini merupakan pengembangan dari Use Case yang punya alur aktivitas-aktivitas juga.



Lihat deh contoh tampilan Activity Diagram disamping!

Activity diagram adalah diagram yang bisa memodelkan proses-proses yang terjadi pada sebuah sistem.



Nggak cuma informasi yang ada di dokumen FSD aja, nih.

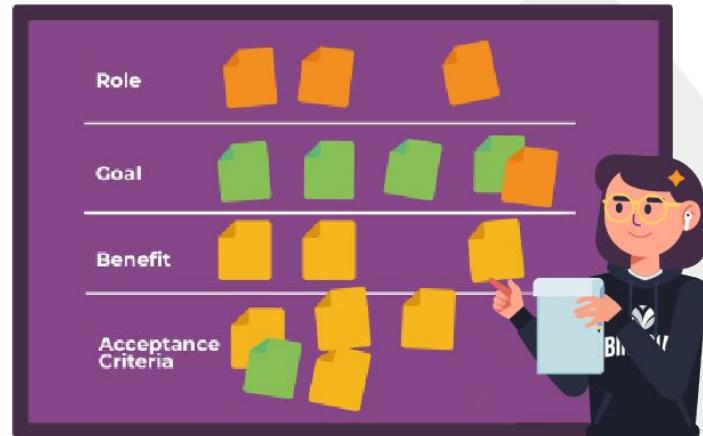
Ternyata Back End Engineer bisa dapetin informasi dari requirement yang namanya **User Stories**.

Kira-kira isinya kayak gimana, ya?



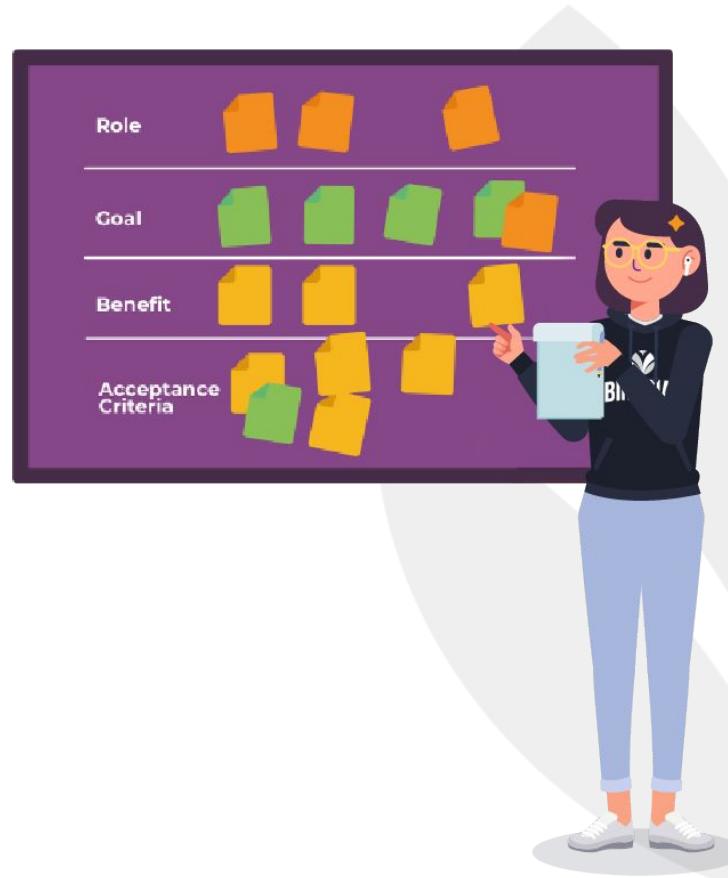
“Terus requirement yang diterima sama Back End Engineer itu kayak gimana?”

Beda sama Waterfall, biasanya requirement yang diterima oleh seorang engineer di metode Agile berbentuk **user story**.



User story adalah **sebuah penjelasan yang ditulis dalam bahasa sederhana dari sudut pandang pengguna produk.**

Jadi seluruh fitur dijelaskan secara sederhana dengan sudut pandang user.



Biasanya user story punya format kayak gini nih~

as a (description of user), I want (functionality, so that (benefit).

Dari formatnya, bisa kita deskripsikan sebagai berikut:

- **As a** sebagai seorang user yang membagikan cerita di dalam user stories.
- **I want** sebagai fitur atau fungsi yang bakal dikembangkan ke depannya.
- **So that** merupakan hasil yang didapatkan setelah fungsi dikembangkan dengan baik.

Title :
Priority :
Estimate :

User Story

*As a [description of user],
I want [functionality],
so that [benefit].*

Acceptance Criteria

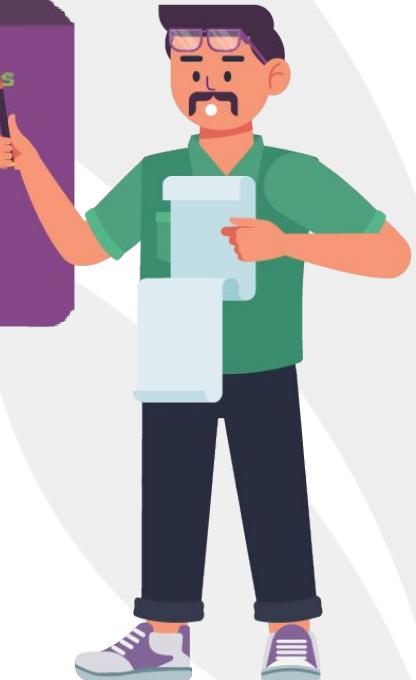
*Given [how things begin]
When [action taken]
Then [outcome of taking action]*

Contohnya kayak gini nih~

Mas Gun adalah seorang Manager. Dia punya ekspektasi buat anggota timnya. Kira-kira bunyinya begini:

“Sebagai seorang Manager (**as a**), aku mau bisa paham progress dari anggota timku (**i want**). Jadi aku bisa report keberhasilan dan kesalahan timku dengan lebih baik (**so that**)”.

Sebagai seorang Manager,
aku mau bisa paham progress
dari anggota timku.
Jadi aku bisa report
keberhasilan dan kesalahan
timku dengan lebih baik.



Pada Agile development, biasanya nggak ada Business Analyst. Walaupun begitu, Tim Product bakal berkomunikasi langsung sama Tim Engineer atau Developer.

Jadi, **user stories ini dikelola langsung oleh tim Product.**



**Beda sama FSD, user stories nggak
memuat informasi sebanyak FSD~**

Yap, jadi engineer harus melakukan analisis secara mandiri.

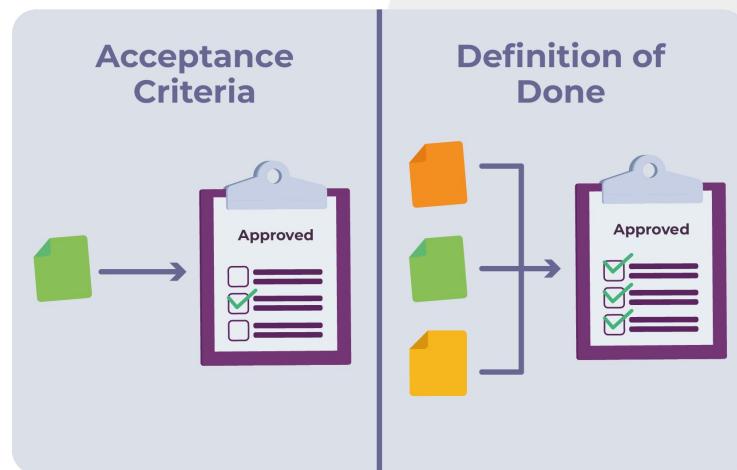
Tapi, biasanya di masing-masing user story punya **acceptance criteria** dan **definition of done**.



- acceptance criteria
- definition of done

Konsepnya kayak gini nih~

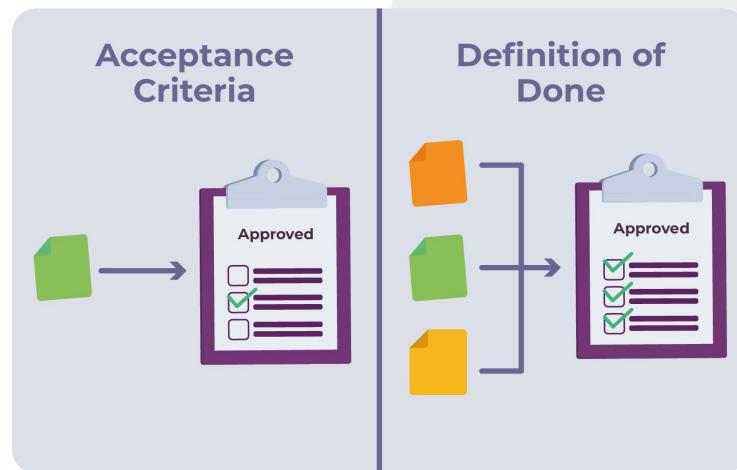
- **Acceptance criteria** adalah kriteria-kriteria dimana sebuah story dianggap telah memenuhi syarat, jadi bisa dianggap sudah berhasil dibuat.
- **Definition of done** adalah kumpulan aktivitas yang perlu dilakukan buat mendefinisikan sebuah user story selesai didevelop.



Berikut adalah masing-masing contoh dari Acceptance Criteria dan Definition of Done~

- **Contoh acceptance criteria**

I can access the progress detail of team members.

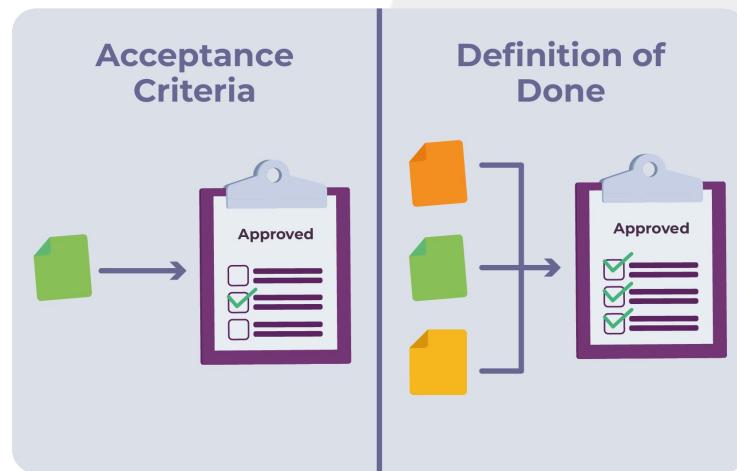


- **Contoh definition of done**

Unit testing coverage 70%.

Code review passed.

Performance test passed.



Biar pemahaman kamu makin ajib tentang User Story, coba kamu jawab soal berikut

Menurut kamu, apakah contoh user story di samping sudah baik dan sesuai kriteria? Jangan lupa berikan alasan ya baik itu jawabannya iya atau tidak.

Silakan renungkan jawabannya selama 5 menit. Kalau udah ketemu, kamu simpan untuk kita cek di halaman berikutnya~

**As a user, I want to cancel a
reservation**

Ternyata oh ternyata, jawabannya adalah user story tersebut kurang baik.

Lho kenapa? Karena **kurang spesifik dan sulit terukur**.



Iya kayak gini, nih. Banyak pertanyaan yang akan muncul, seperti:

- Apakah user mendapatkan refund secara penuh atau ada pemotongan?
- Apakah proses ini berlaku untuk semua hotel?
- Seberapa lama user bisa membatalkan reservasinya?
- Seberapa jauh proses reservasi yang bisa dibatalkan?



Karena tadi sudah dipaparkan contoh yang kurang baik, berikut adalah contoh user story yang ideal sesuai dengan case sebelumnya:

- As an user, I am emailed a confirmation when I cancel a reservation
- As a premium member, I can cancel a reservation up to the last minute
- As a non premium member, I can cancel a reservation up to 24 hours in advance



Materi selanjutnya akan lebih detail ngomongin Waterfall.

Materi ini ada kaitannya sama UML, ERD, dan Sequence Diagram.

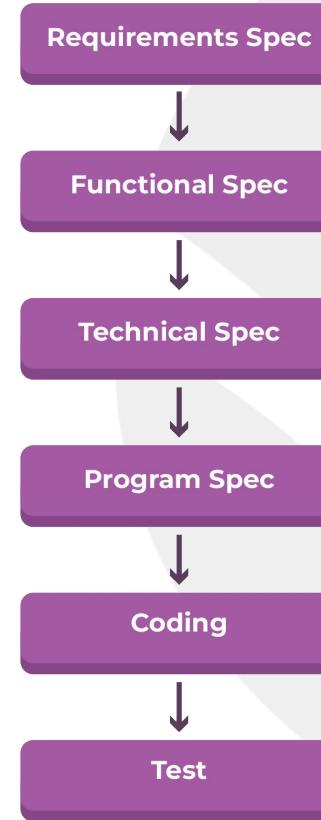
So, ladies and gentleman, please welcome
✨Waterfall: Membuat sebuah feature
berdasarkan FSD ✨



Dari FSD, seorang engineer dapat menentukan domain yang terlibat, memanfaatkan Use Case diagram.

Selain itu, melalui Activity Diagram kita bisa menentukan lingkup service buat masing-masing activity.

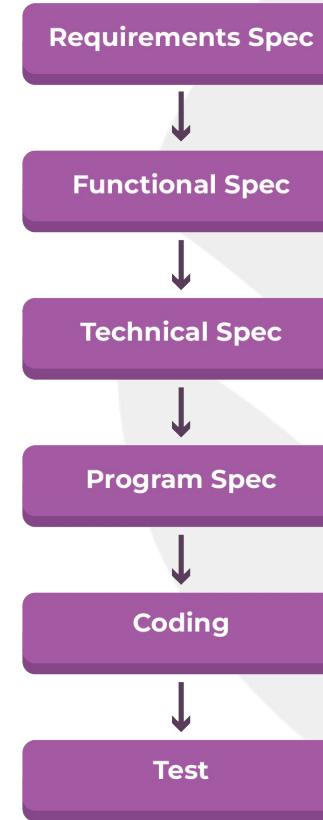
Hal ini tentunya bakal berpengaruh pada table yang mau dibuat, dan interaksi masing-masing table.



Kalau pakai sequence diagram, kita bisa tahu urutan proses yang terjadi di sebuah program~

Dengan begitu, seorang engineer nggak langsung melakukan implementasi pada code.

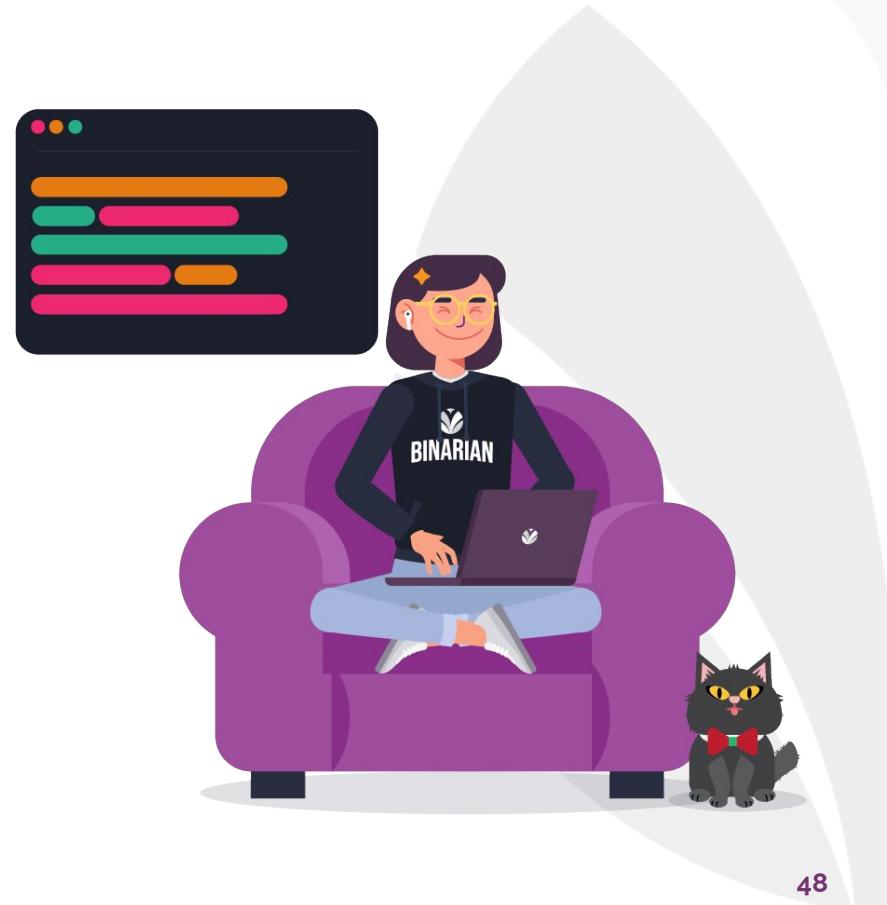
Melainkan perlu tetap menyediakan dokumentasi berupa Technical Specification Document yang berisi spesifikasi yang lebih teknikal.



Setelah TSD disetujui, maka engineer mulai melakukan coding~

Tentunya tahap pembuatan dokumentasi ini jadi hal yang prioritas banget karena bakal jadi satu tahapan development di dalam iterasinya.

Seluruh deadline penggerjaan biasanya udah ditentukan pada awal iterasi.



Kalau tadi udah dijelaskan cara bikin feature di Waterfall, sekarang kita bakal coba bikin di Agile.

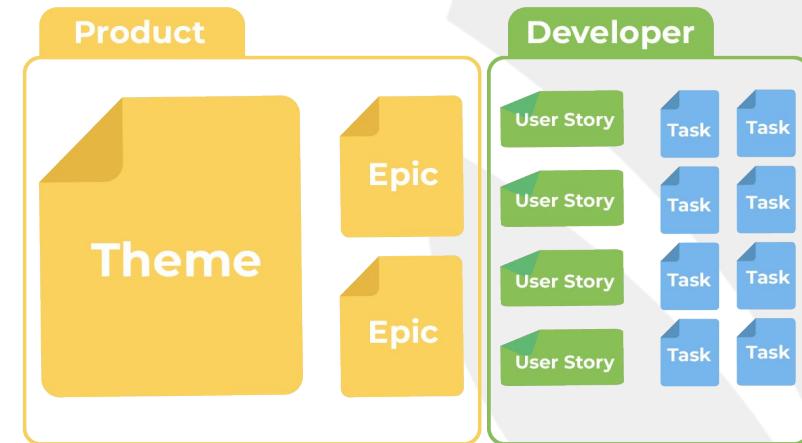
Kita bakal belajar membuat sebuah Feature berdasarkan User Story.



Cara bikinnya tuh kayak gini~

Oh iya, sekarang perhatikan gambar di samping, ya!

- Tim Product bakal bikin satu theme yang besar, kemudian memecahnya ke dalam bentuk epic.
- Tim Developer bakal menerima requirement pas udah berbentuk user story.
- Tim Developer bakal menentukan user story yang mau diambil pada suatu iterasi di fase planning.



Terus...

Developer harus mampu mengira-ngira seberapa kompleks, waktu yang dibutuhkan, dan prioritas buat mengerjakan suatu user story.

Effort yang dibutuhkan buat mengerjakan satu user story biasanya diukur sama **story point**.

Menentukan story point itu mirip kayak menilai usaha mengangkat batu sesuai beratnya



Dari sebuah user story, biasanya tim developer bakal melakukan breakdown pada story tersebut supaya nantinya jadi sebuah task yang lebih kecil.

Menentukan story point itu mirip kayak menilai usaha mengangkat batu sesuai beratnya



Bebannya ringan, nih!
Kita kasih nilai 1 pt, ya!



Hmm, kalo yang ini
poinnya 4 karena
sedikit lebih berat.



Waks, ini mah poinnya
10! Pasti butuh banyak
effort ngangkatnya!!

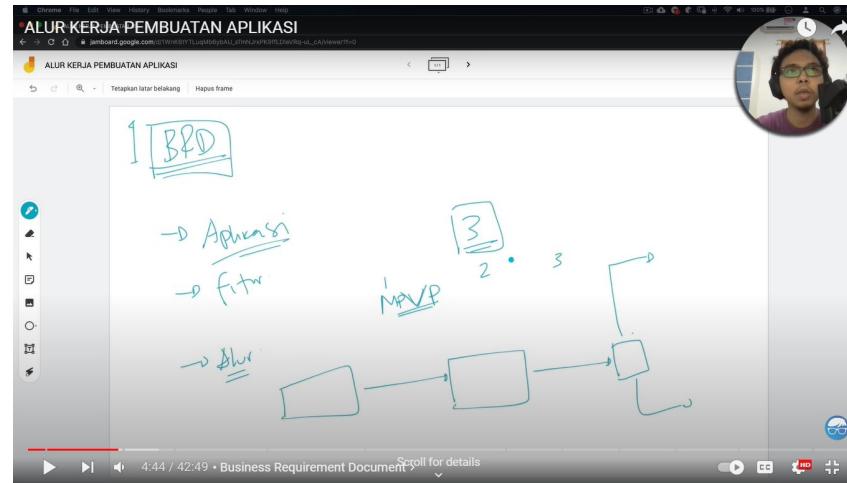


Task tersebut berisi hal yang lebih spesifik dan technical yang bakal dilakukan oleh developer.

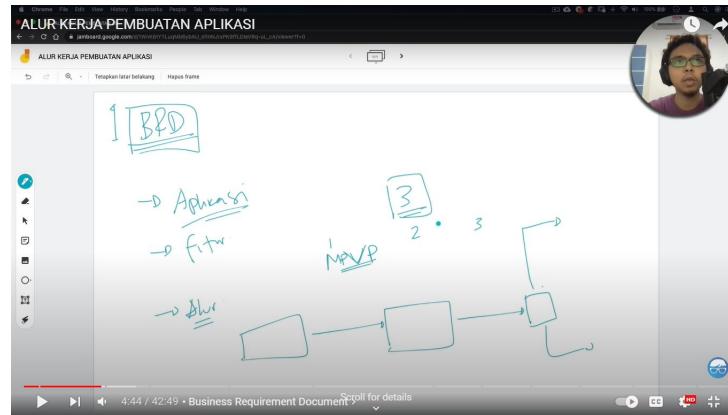
Selain itu, task breakdown ini biasanya dilakukan pas udah beres menentukan user story mana yang mau dikerjakan.

Sehingga, ketika developer melakukan development, biasanya dokumentasi nggak jadi prioritas. Selesai deh. Yeay!





Biar kamu makin kebayang, udah disiapin nih satu referensi video yang bisa kamu tonton.
Isi penjelasannya seputar proses pembuatan aplikasi.



Oh iya, walaupun proses yang dijelasin di video bisa diterapin ke suatu perusahaan, tetapi we always have to remember, kalau tiap perusahaan punya standar alur kerja masing-masing.

Sehingga sangat mungkin tiap perusahaan akan berbeda juga prosesnya

Alur Kerja Pembuatan Aplikasi

Pipp pipp! Pengumuman~pengumuman~

Ada tugas yang perlu kamu selesaikan, nih. Silakan untuk membuat contoh user story yang baik dari contoh cerita berikut!



Seorang user ingin membuat suatu aplikasi e-commerce jual beli. Buyer diharapkan bisa membayar via transfer bank, e-wallet, virtual account dan paylater. Dari situ, User tersebut memiliki sebuah request, yaitu:

- Sebagai buyer, saya ingin bisa check out lebih mudah

Dari cerita di atas, buatlah minimal 3 contoh user story yang benar beserta dengan acceptance criteria-nya.

Nanti diskusikan hasilnya bersama facilitator dan teman sekelas pada pertemuan berikutnya. Selamat mengerjakan 😊



Setelah membahas topic Working as A Back End Engineer, kamu jadi paham kalo seorang Back End Engineer perlu mempertimbangkan aspek business logic ketika ngoding.

Coba deh kamu bayangin, misalnya kamu berada di posisi sebagai developer yang diminta untuk membuat sebuah website. Namun user story yang diberikan kurang spesifik dan terukur. Nah, apa yang kira-kira akan kamu lakukan?



Nah, selesai sudah pembahasan kita di Chapter 1 topic 5 ini.

Selanjutnya, kita bakal bahas tentang Clean Code.

Penasaran kayak gimana? Yuk langsung ke topik selanjutnya~

