

# SDLC

## Silver - Chapter 0 - Topic 3

Selamat datang di **Chapter 0 Topic 3**  
online course dari  
**Binar Academy!**



**Halooo, kita Ketemu lagi!** 🎉

Di topic sebelumnya, kamu udah ngerti komponen apa aja yang ada di dalam aplikasi. Nah, di sini kamu bakal belajar **rahasia dapur di balik pembuatan aplikasi**.

Kalau gitu, yuk langsung aja kita kepoin~



### Apa yang bakal kamu pelajari di Topic 3 ini?

Singkatnya, kamu nanti bakal ngerti soal:

- 1) Apa itu Scrum dan Software Development Life Cycle (SDLC)
- 2) Jenis-jenis profesi dan perannya di pembuatan aplikasi
- 3) Langkah pembuatan aplikasi mulai dari roadmap planning sampai jadi product dalam bentuk MVP



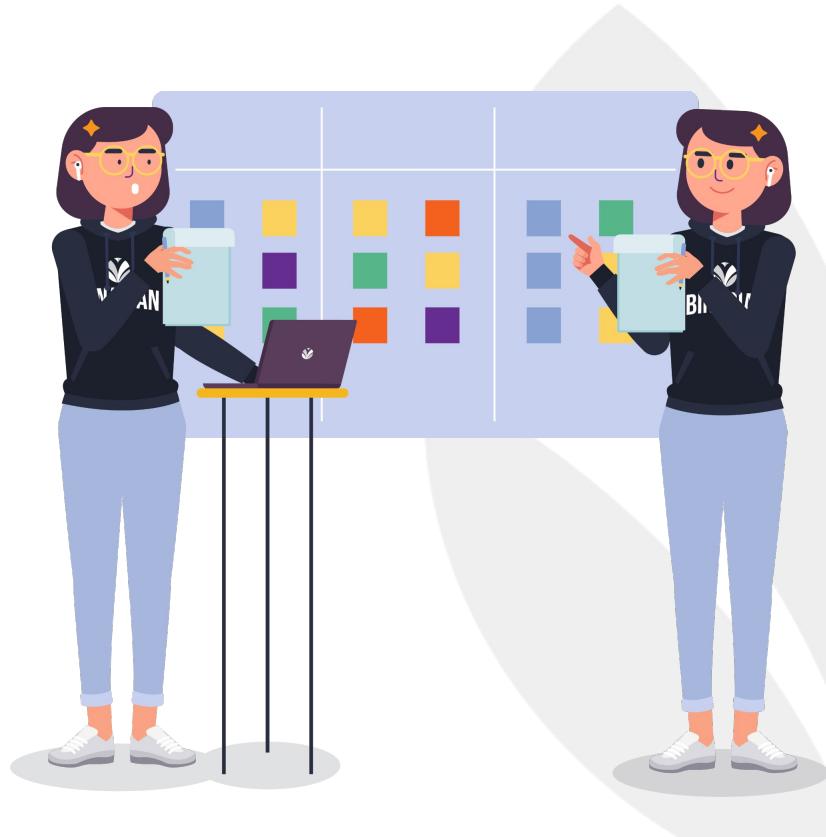
**Perhatian: Topic ini cukup panjang  
dan butuh konsentrasi yang tinggi  
buat mencerna informasinya**

Usahain lagi di lingkungan yang kondusif buat belajar  
ya 😊



Oke, kita mulai! Di topic sebelumnya sempat disinggung soal Agile kan ya. Tapi sebenarnya Agile itu apa sih?

Bikin software itu melibatkan banyak orang dan proses yang panjang. Jadi, kalau gak ada metodologi kerja khusus buat ngatur proses dan pembagian tugasnya, semua bisa jadi berantakan.

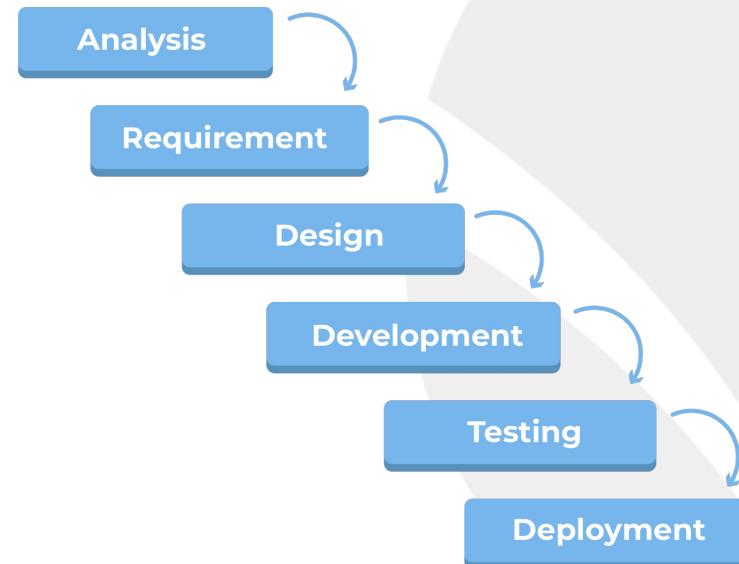


Nah, **Agile termasuk salah satu metodologi kerja pembuatan software, atau disebut software development life cycle (SDLC)**. Tujuan SDLC yaitu buat bikin software dengan kualitas terbaik, tapi dengan biaya terendah, di waktu yang paling cepet. FYI, jenis SDLC gak cuma Agile aja lho. Ada apa lagi ya?



Jadi, sebelum masuk ke zaman aplikasi yang bisa lebih fleksibel diupdate, software konvensional umumnya dikerjain pakai metode SDLC Waterfall

Fase-fase di metodologi kerja Waterfall berbentuk linear dan dikerjakan dalam rentang waktu tertentu buat dapet hasil final.



Tapi, karena smartphone makin ngetrend, perkembangan software juga makin lincah.

Dulu, tiap update versi baru harus rilis CD installer atau download bergiga-giga, sekarang cukup update lewat Play Store atau App Store.

Nah, makanya butuh metode kerja baru buat menyesuaikan dan dari sini lahirlah Agile. **Kelebihan utama Agile adalah produk yang kita bikin bisa lebih cepet menangkap feedback dari market.**



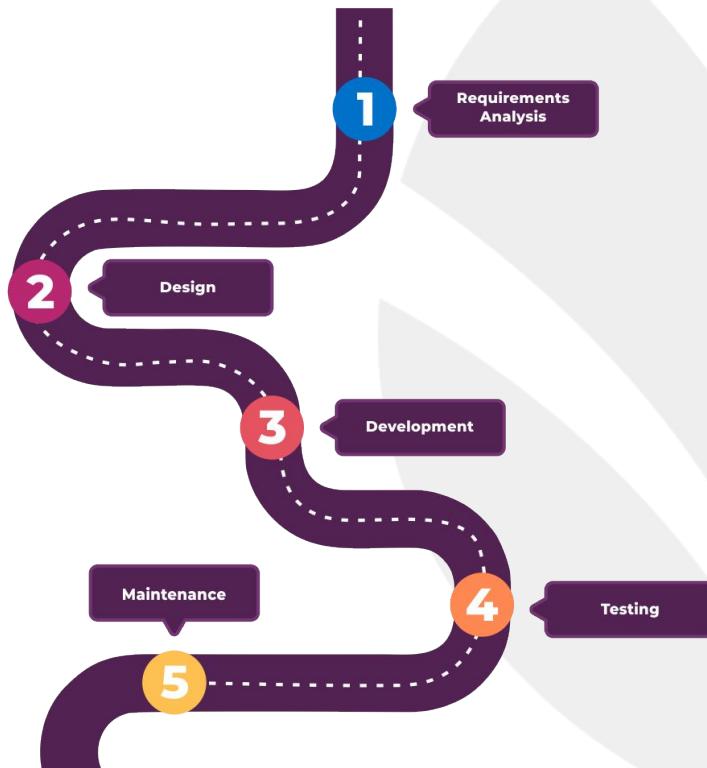
## Intermezzo: Di Agile, secara garis besar ada dua metode turunan yaitu Scrum dan Kanban. Apa bedanya?

- Scrum biasa dipakai buat product development. Contoh: bikin aplikasi.
- Sedangkan Kanban dipakai buat project/product yang melibatkan banyak pihak dari luar tim. Contoh: manufaktur perakitan mobil.

Tapi, kita gak bakal bahas Kanban. Kita bakal fokus ke Scrum karena nantinya kamu dilatih buat jadi seorang software engineer.



Beda sama Waterfall yang linear, Scrum punya fase yang terus-menerus dilakukan. Siklus yang ada di Scrum ini sering disebut juga Software Development Life Cycle (SDLC).

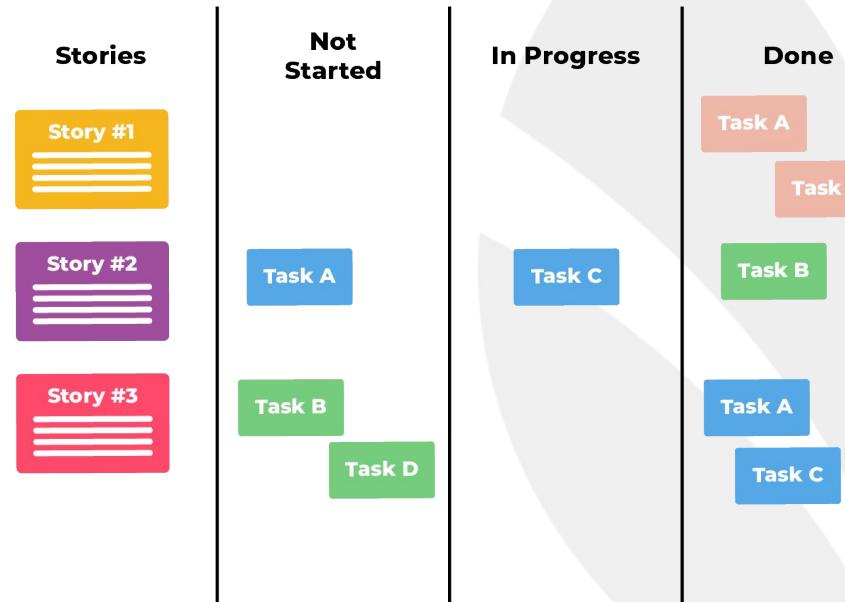


## Karena terus diulang, di Scrum ada satuan waktu konstan dan disebut sprint

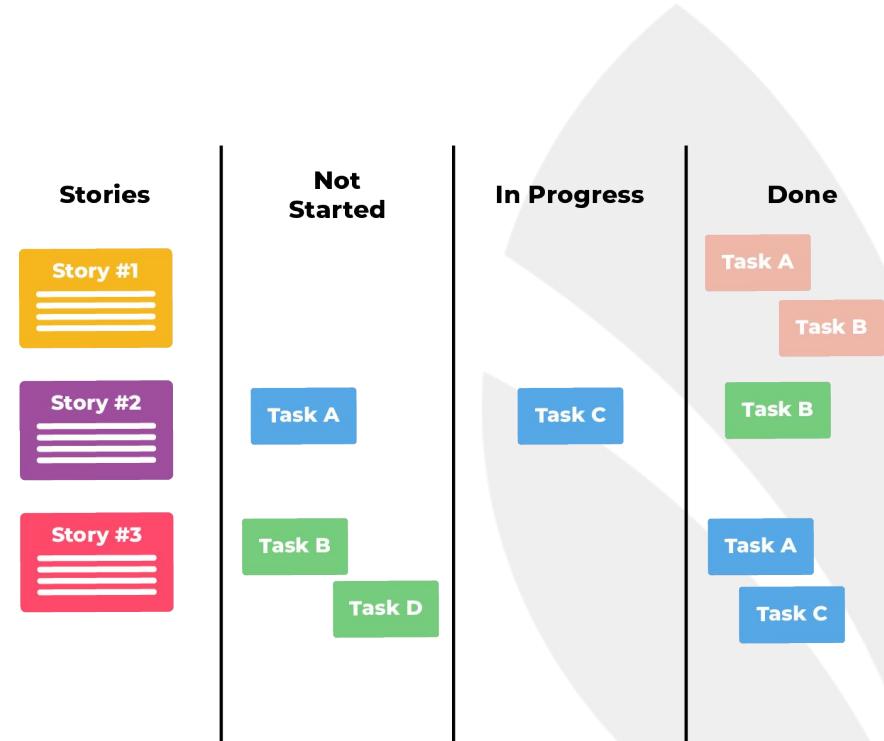
Satu sprint biasanya berdurasi 1 minggu sampai maksimal 1 bulan.

Kita mulai dari dasarnya dulu ya:

- Satu sprint terdiri dari beberapa task yang harus diselesaikan keseluruhan tim
- Seluruh task yang sudah dirumuskan dimasukkan ke product backlog

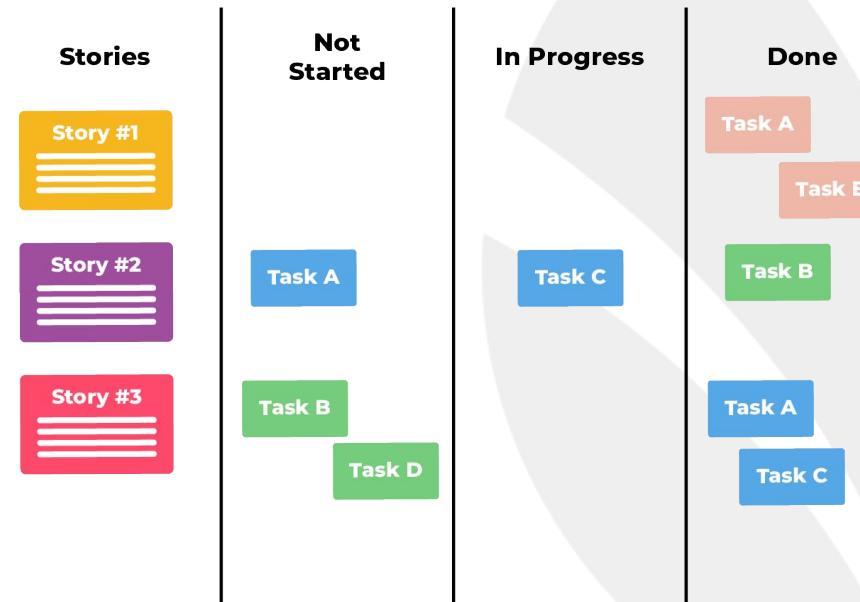


- Sebelum sprint dimulai, ada sesi sprint planning yang mana seluruh anggota tim menentukan jumlah task untuk sprint yang akan datang pada masing-masing anggota
- Task yang sedang dikerjakan dimasukkan ke bagian in progress, sementara task yang sudah selesai akan dimasukkan ke bagian done



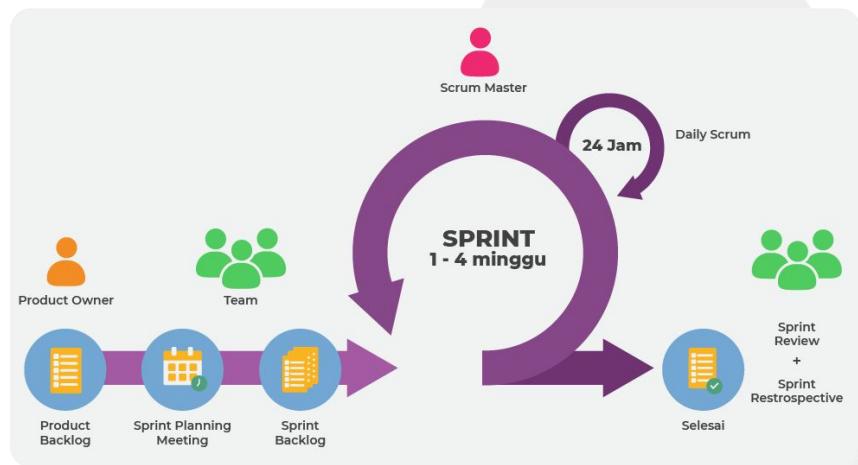
- Apabila ada task yang tidak selesai pada satu sprint, maka task itu akan dimasukkan ke sprint backlog untuk sprint selanjutnya

Cara ngelakuinnya simpel, bisa pakai board yang kayak gambar di samping~



Itu tadi bentuk board dan cara bikinnya, tapi ada juga sesi-sesi khusus yang scrum banget.

- Tiap hari, ada scrum daily atau standup meeting. Meeting harian ini tujuannya buat ngelaporin task apa yang udah dikerjain, obstacle yang ditemui, target task hari ini, dll.
- Sehabis selesai sprint, ada sesi khusus buat ngebahas pencapaian tim yang disebut sprint review.
- Selain ngebahas pencapaian, ada juga pembahasan non-teknis kayak tingkat kebahagiaan, kekurangan, dll yang dirangkum di sesi yang dinamakan **sprint retrospective**.



## Trus antara Waterfall sama Agile-Scrum, yang mana yang lebih bagus?

Gak ada yang lebih bagus atau lebih jelek. **Semua tergantung kebutuhan produknya.**

Tapi, karena kita belajar tentang pembuatan aplikasi yang butuh feedback cepet dari user, yuk kita bahas detail di Agile-Scrum satu-satu!



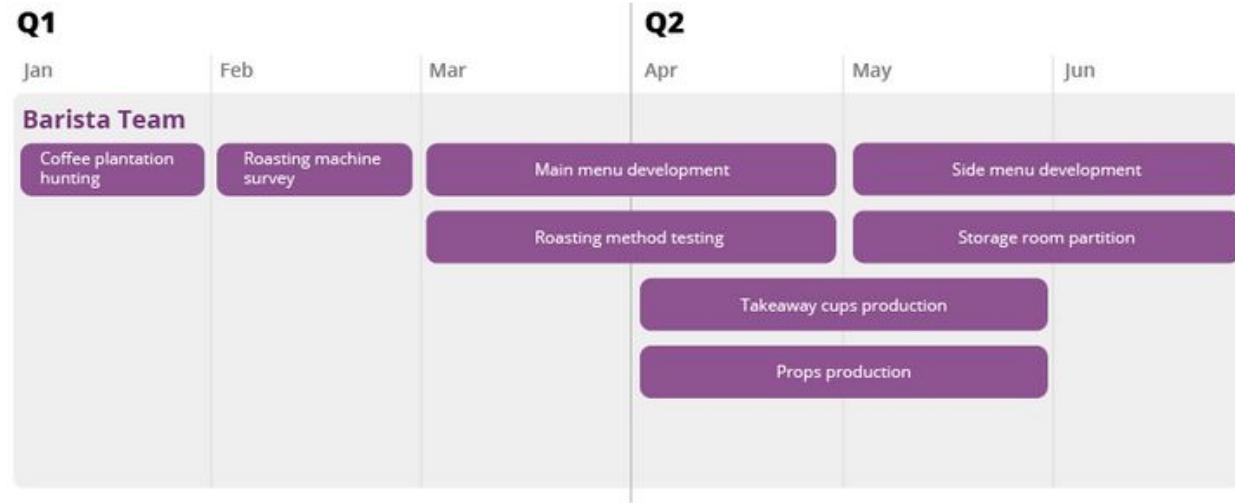
## Jadi, apa dong tugasnya product owner?

Product owner bertugas merumuskan masalah yang ada di dunia nyata, trus cari solusi digital yang bisa bantu selesaikan masalah itu.

Tapi gak segampang itu, product owner harus bisa mimpin riset buat validasi masalah, riset produk dan pasar, sampai bikin strategi produk biar produk yang dibikin sesuai sama kebutuhan pasar, atau biasa disebut product-market fit.

### Product owner

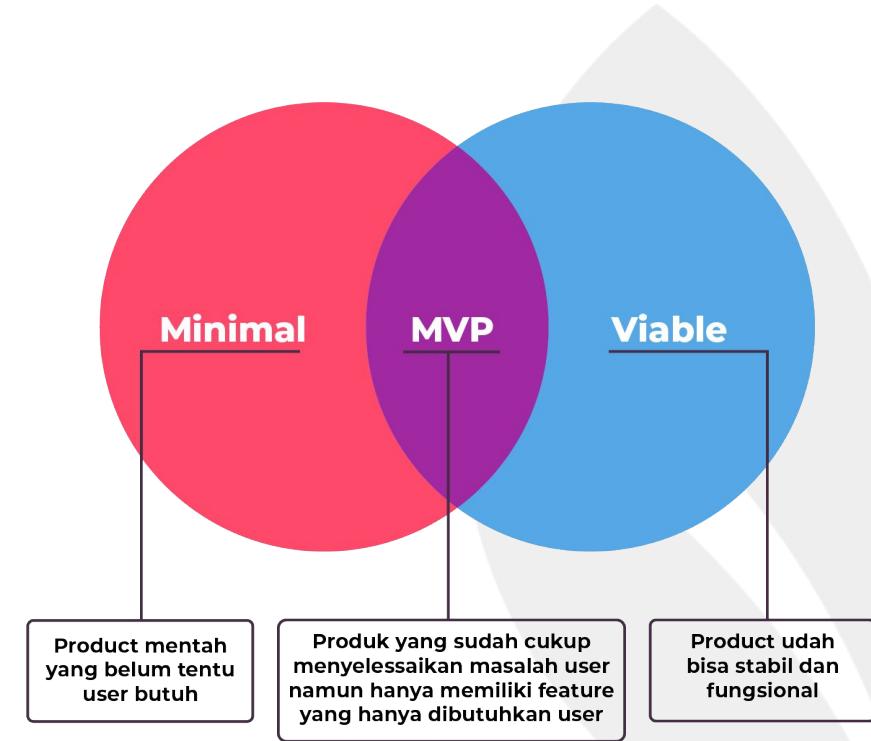




Selain itu, harusnya product owner juga bisa ngatur dalam jangka panjang coffee shop ini ga cuma jualan kopi, tapi bisa jadi tempat sharing komunitas, perpustakaan, atau jual merchandise juga. Nah, **rencana untuk menuju ke objective jangka panjang ini lah yang namanya roadmap plan.**

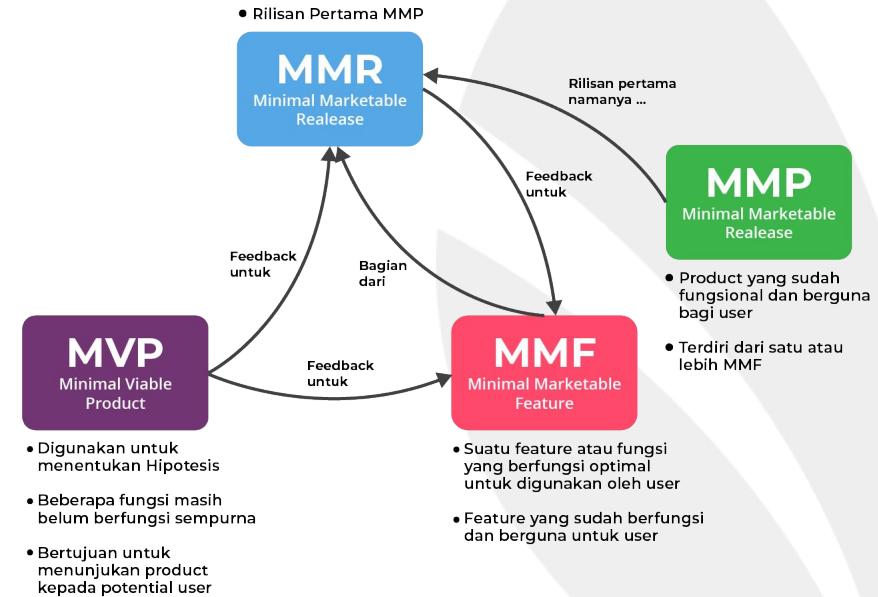
## Tapi, kebanyakan aplikasi nggak dimulai dengan modal besar

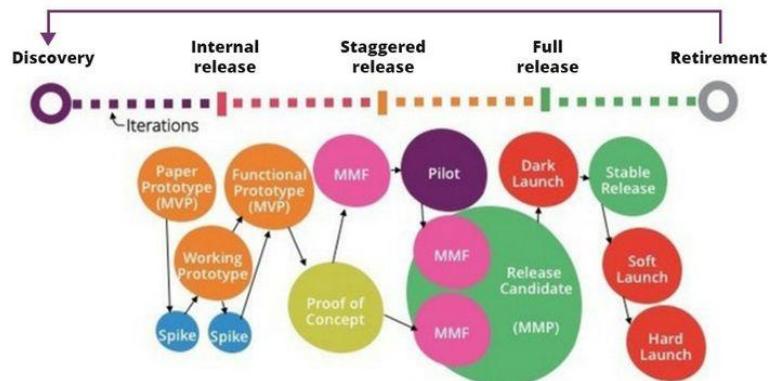
Sebagai contoh, Instagram juga dulunya cuma bisa dipakai di iOS dengan fitur yang cuma "feed" aja. Tapi lambat laun, fiturnya berkembang jadi lengkap sekarang. **Produk dengan fitur minimal ini disebut Minimum Viable Product (MVP).**



Produk yang masih MVP, artinya belum bisa dipasarkan secara luas, atau belum bisa disebut MMP (Minimum Marketable Product).

Ada beberapa tahap yang harus dilalui sebelum sebuah produk bisa dirilis secara luas (Full release).



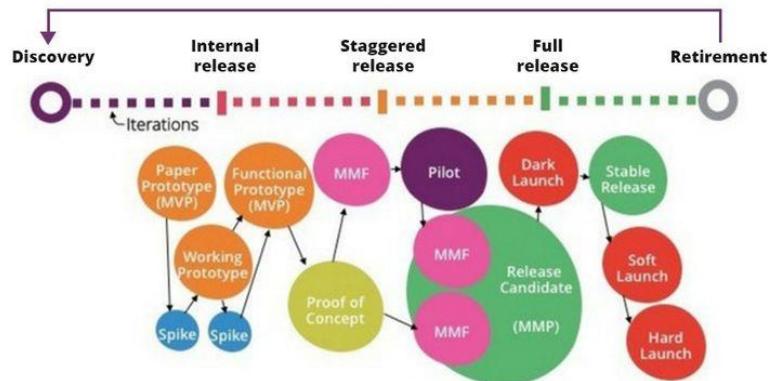


## Proses buat sampai jadi MMP juga panjang

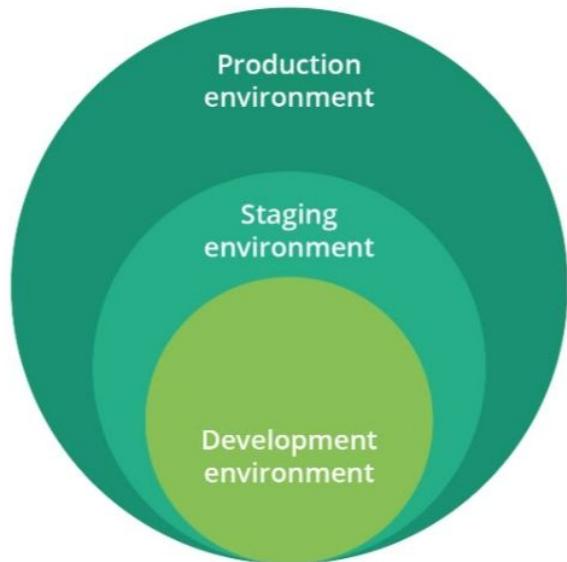
Dibutuhkan siklus riset-development-validasi yang berulang — disebut iterasi — sebelum masuk ke tahap rilis.

Untuk merilis pun juga ada 3 jenis:

- **Internal release:** dirilis ke internal tim produk untuk dilakukan unit test untuk menghimpun technical feedback..



- **Staggered release:** dirilis ke publik secara terbatas untuk menghimpun performance feedback, untuk mengetahui performa dan keandalan produk saat digunakan banyak pengguna
- **Full release:** dirilis ke publik secara luas. Pada versi ini, produk yang dirilis seharusnya sudah dapat diterima pasar, serta fungsional dan stabil secara teknis.

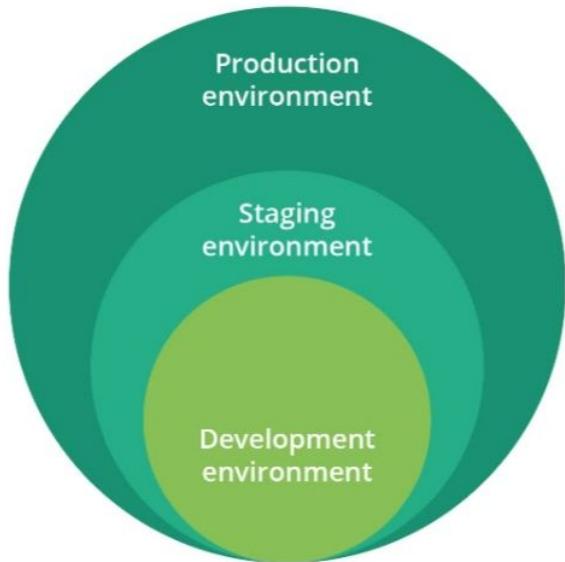


## Selain rilis, ada juga yang namanya environment

Gampangnya, sebuah environment adalah kalangan yang terdiri dari beberapa role. Umumnya environment ada 3 jenis. Apa aja ya?

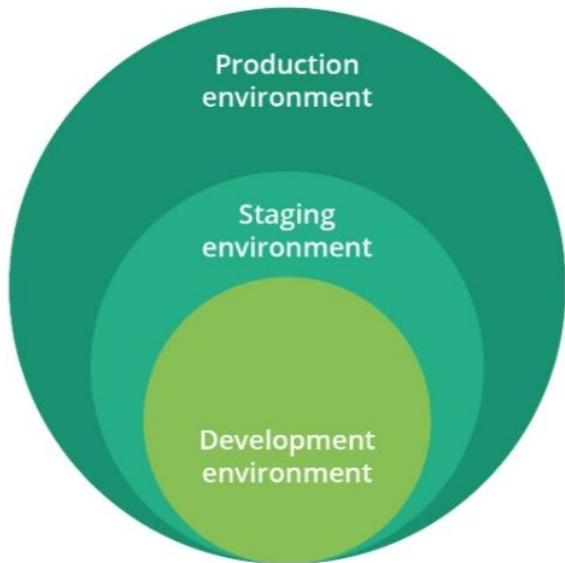
- **Development environment**

Para software engineer menuliskan kode mereka di environment ini. Fase ini bersifat dinamis, sehingga segala perubahan terjadi sangat cepat karena masih dalam masa pengembangan, dan dapat langsung dicoba



- **Staging environment**

Di fase ini berisi versi software yang akan dirilis ke publik. Disinilah biasanya software sudah bisa diuji coba untuk skala internal perusahaan, sambil menunggu persetujuan dari stakeholder sebelum dirilis. Perbaikan dan perubahan kecil masih diperbolehkan terjadi di fase ini



- **Production environment**

Fase ini berisi versi software yang siap dirilis ke publik. Pada fase ini pula tidak dikehendaki adanya perubahan atau pergantian produk, melainkan harus menjadwalkan perubahan tersebut untuk dirilis pada versi berikutnya

## Oke, kita main kuis lagi

Sekelompok mahasiswa tadi akhirnya memutuskan bikin coffee shop dengan menu pertama single origin. Waktu dites ke sesama barista coffee shop itu, rasanya udah pas dan mereka sepakat satu porsi dihargai Rp 20,000.

Sekarang waktunya manggil semua pegawai buat getes kopinya. Ternyata, mereka bilang, harga Rp 20,000 buat rasa kayak gitu dirasa kemahalan dan ada feedback kalau di daerah itu kebanyakan cuma doyan kopi yang dicampur susu. Pertanyaannya, peristiwa ini termasuk dalam product stage dan environment apa?

A	MVP, development environment
B	MVP, staging environment
C	MMP, staging environment

## Internal testing dan Staged testing

### Internal testing

Targetnya adalah end user di dalam tim product development

Dilakukan di development environment

Kecacatan yang ditemui langsung diperbaiki sebelum rilis

Para engineer ikut ngawasin test

### Staged testing

Targetnya adalah end user di luar tim product development, tapi masih dalam satu perusahaan

Dilakukan di staging environment

Semua kecacatan yang ditemukan bakal dilaporin ke tim product development, tapi baru dibenerin di rilis selanjutnya

Para engineer gak ikut ngawasin test

Tepat sekaleee! 😍 Boleh bilang peristiwa ini berlangsung di staging environment dan produknya berada dalam fase MVP.

Pengetesan dilakukan ke semua pegawai (ada di staging environment). Tapi, peracik kopi (setara sama engineer) udah gak bisa ngubah rasa kopi karena udah waktunya rilis.

## Internal testing dan Staged testing

### Internal testing

Targetnya adalah end user di dalam tim product development

Dilakukan di development environment

Kecacatan yang ditemui langsung diperbaiki sebelum rilis

Para engineer ikut ngawasin test

### Staged testing

Targetnya adalah end user di luar tim product development, tapi masih dalam satu perusahaan

Dilakukan di staging environment

Semua kecacatan yang ditemukan bakal dilaporin ke tim product development, tapi baru dibenerin di rilis selanjutnya

Para engineer gak ikut ngawasin test

Jadi, tugas PO pas udah di staging environment adalah mikirin gimana strateginya biar produknya bisa selesaikan masalah user dengan lebih efektif, biar terjadi product-market fit.

Nah, kalau masih ditemukan kekurangan dari sisi kegunaan produk, perbaikan bisa dilakukan di rilis berikutnya.

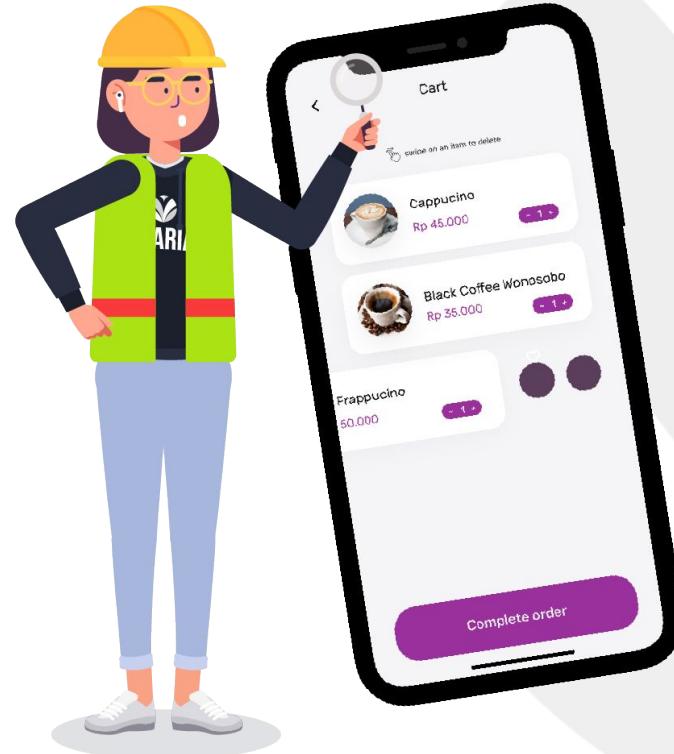
## Di SDLC ada juga yang disebut UI/UX designer

Kalau pakai analogi coffee shop tadi, seorang UI/UX designer bertugas buat ngatur flow gimana orang mulai dateng, parkir kendaraan, pesen, dibuat, sampai bisa nikmatin kopinya.

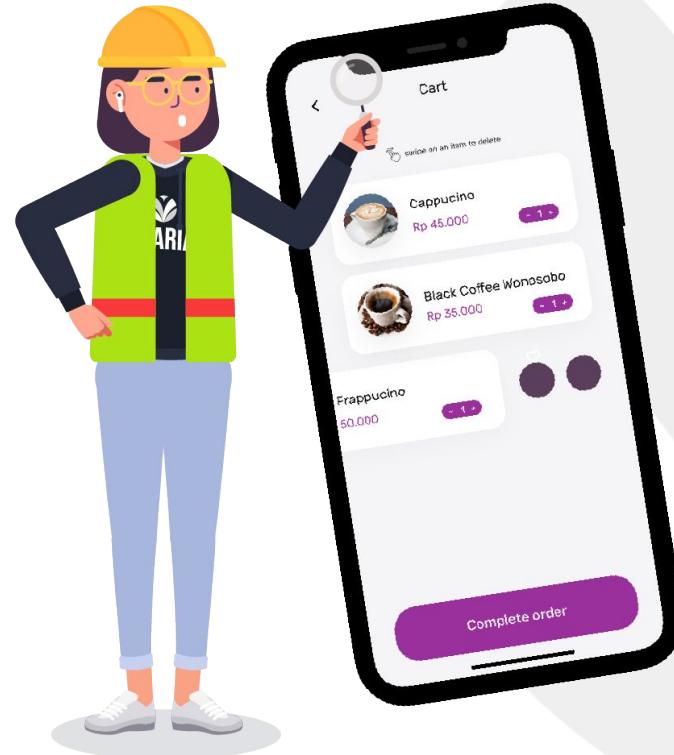


Jadi UI/UX designer ternyata gak sepele dan gak terbatas di desain aja, lho. Lebih dari itu, **tugas utama seorang UI/UX designer bisa dibagi jadi 3 garis besar:**

1. Memetakan ide dan user behavior ke dalam rancangan flow aplikasi. Output dari proses ini biasanya berbentuk flowchart, wireframe, dan lo-fi design.

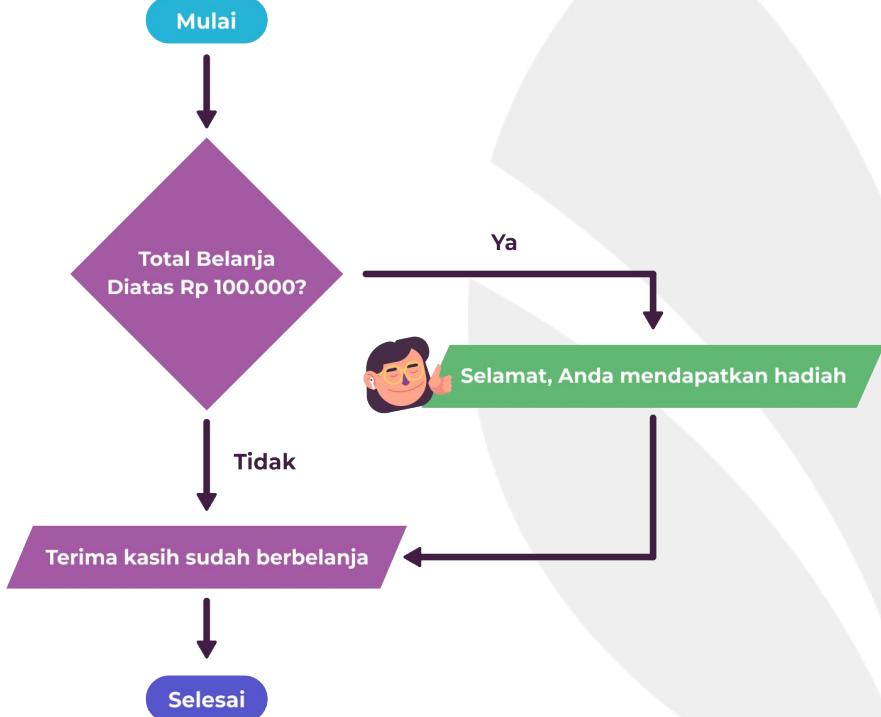


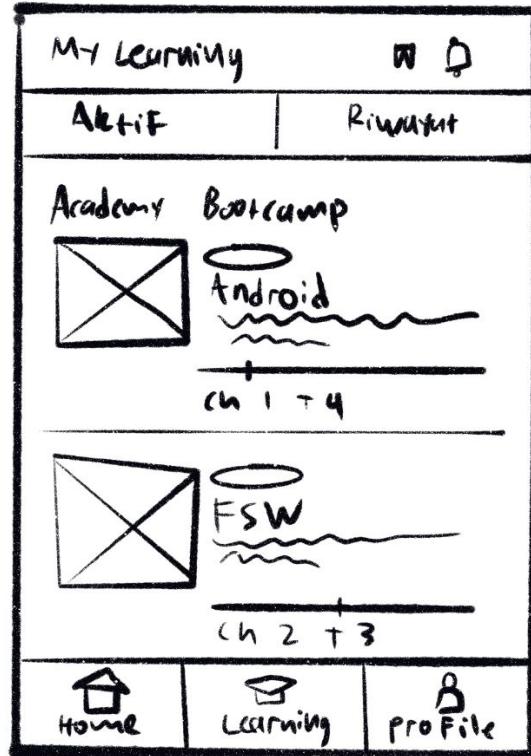
2. Melakukan serangkaian riset/test biar flowchart yang dibikin gak membingungkan dan bisa ngasih experience yang bagus buat user. Output dari proses ini disebut user journey matrix.
  
3. Bikin desain interface yang gak cuma estetis, tapi juga intuitif. Artinya, user bisa paham kegunaan masing-masing fitur tanpa harus ribet menjelaskan. Output dari proses ini disebut hi-fi design.



**Sebelum jadi lo-fi design, sebaiknya seorang UI/UX designer bikin yang namanya flowchart.**

Flowchart ini berisi logika perjalanan user di dalam produk kita yang disampaikan secara sederhana tanpa gambaran interface.



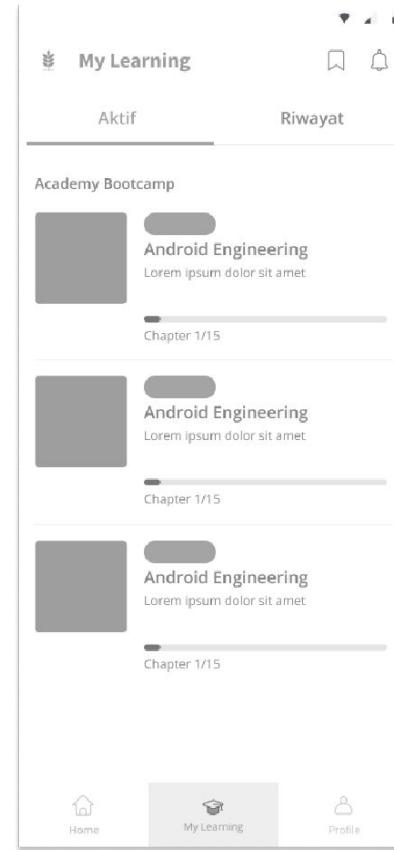


Setelah flowchart jadi, ini waktunya bikin **rancangan kasar per screen. Rancangan ini dinamakan wireframe.**

Wireframe berisi komponen yang ada di tiap screen-nya. Biasanya, rancangan interface kasar ini digambar secara sederhana dengan kotak-kotak beserta ukuran/dimensinya.

Kalau wireframe per screen udah jadi, sekarang waktunya buat sambungin wireframe sesuai logika yang dibikin di flowchart. **Rancangan yang udah setengah jadi ini dinamain low-fidelity design (Lo-Fi).**

Lo-Fi design umumnya berbentuk digital dan dibuat pakai **tools kayak Sketch, Zeplin, atau Figma**. Isinya wireframe yang udah terstruktur sesuai sama perjalanan user dari awal sampai akhir penggunaan produk.



## Lo-Fi design ini dijadikan bahan untuk user testing

Test ini diawasi oleh UI/UX designer dan tujuannya buat menggali potensi user behavior sewaktu nanti app-nya udah jadi.

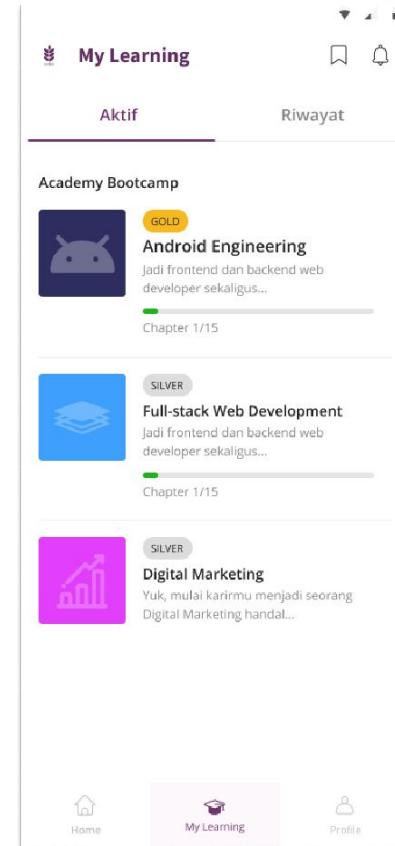
Langkah-langkah di tabel di bawah nantinya didokumentasikan ke dalam **user journey matrix**. Dan dari matrix itu, kelemahan dalam flow bisa kelihatan pertanyaan seperti:

- Bisa nggak sih user mengoperasikan appnya?
- Ngerti gak sih user sama flow yang udah dibikin?



Sehabis semua feedback dikumpulkan, sekarang waktunya buat bikin high-fidelity design (Hi-Fi).

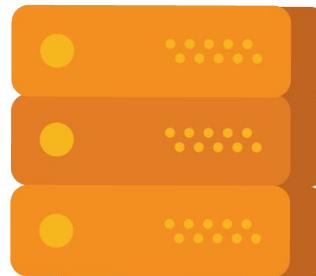
Sederhananya, sebuah Hi-Fi design adalah Lo-Fi yang udah dilengkapi aset visual dan interaksi kayak icon, ilustrasi, typeface, dll.



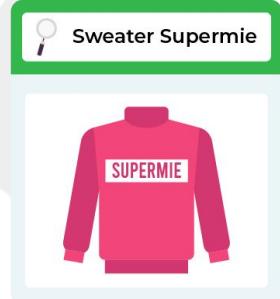
Nah, dari flowchart yang udah dibikin, seorang UI/UX designer bisa mulai mengomunikasikan kebutuhan pemrograman ke software engineer.

Secara garis besar, **software engineer di SDLC bisa dibagi ke dua jenis, yaitu backend engineer dan frontend engineer.**

### Backend Engineer

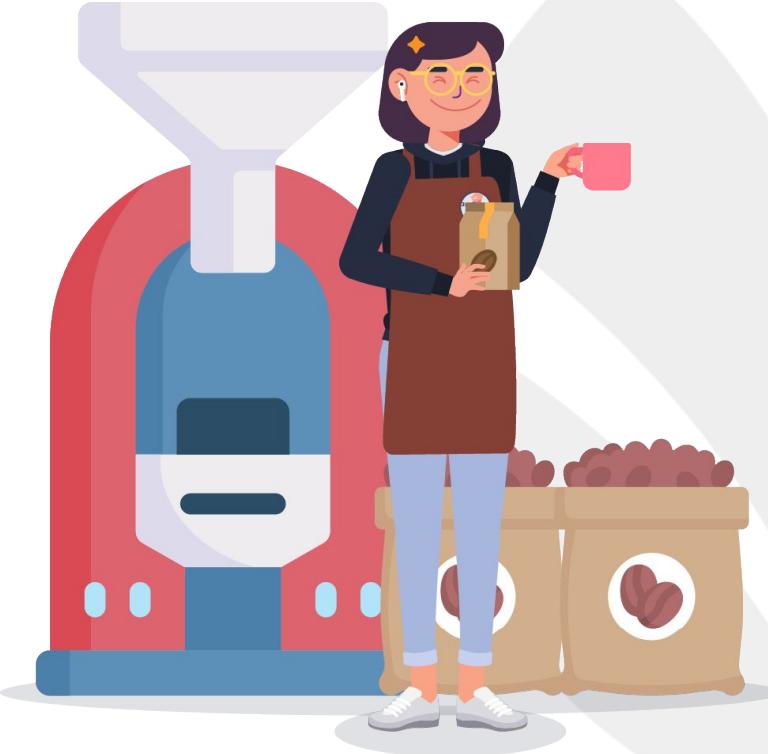


### Frontend Engineer



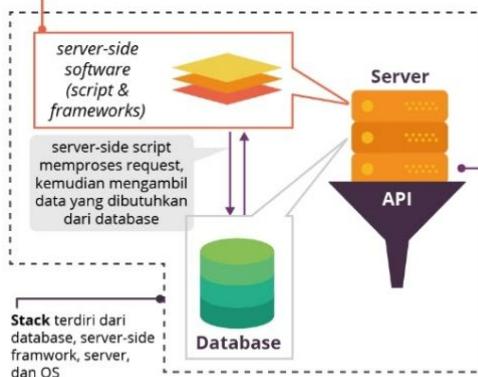
## Sekarang, kita kenalan dulu sama backend engineer~

Kalau kita balik lagi ke pembuatan coffee shop tadi, seorang backend engineer bertugas buat nentuin bahan dasar kopi, cara roasting biji kopi, metode ngeracik, sampai cara menyimpan biji kopi di gudang.



# Backend engineering

**Frameworks** adalah *libraries* dari *server-side* bahasa pemrograman yang menyusun struktur dari sisi backend



Dalam SDLC, seorang backend engineer bertugas **buat mengurus server, bikin skema database, dan bikin API**. Intinya, bikin arsitektur buat eksekusi fitur biar sesuai sama logika flowchart yang disepakati.

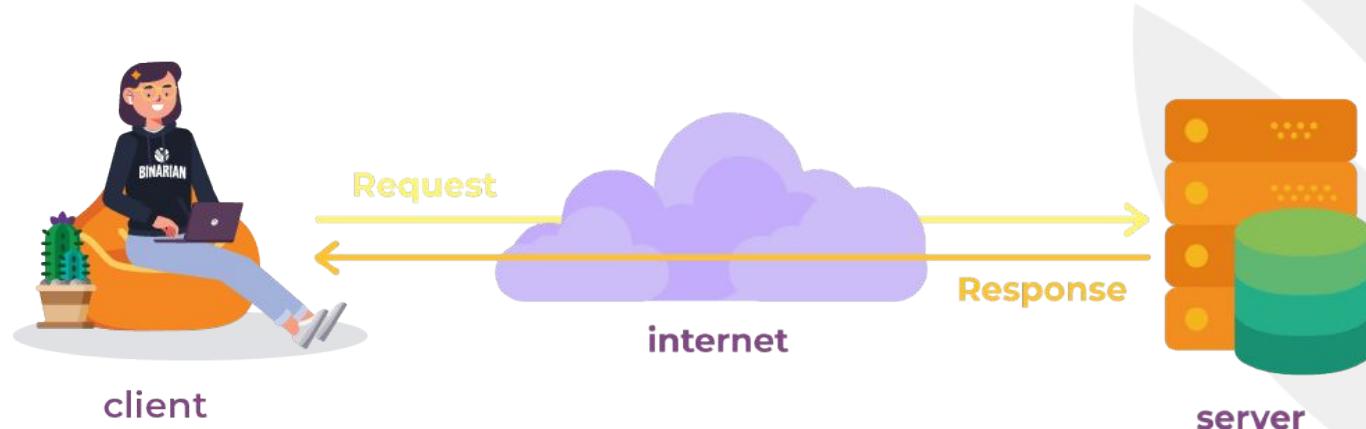
Seorang backend engineer yang baik harus bisa menerjemahkan keinginan UI/UX designer yang ada di flowchart ke dalam bahasa yang lebih teknis. Output dari proses ini adalah application programming interface atau biasa disingkat API.

Inget analogi API yang ada di topic sebelumnya? Nah dalam SDLC, API adalah protokol komunikasi yang fungsinya sebagai pengatur pertukaran database yang berlangsung antara server dan client.



## Kamu bingung? Yuk, kita perdalam dikit soal server dan client

- Yang disebut **server, adalah pihak yang menyediakan service**. (Server → Yang ngasih service, yang nyediain service; hehe). Gampangnya lagi, server selalu ada di backend.
- Kalau **client, kayak klien di dunia nyata, adalah service requester, alias yang minta service**. Nah, kalau client biasanya ada di frontend.



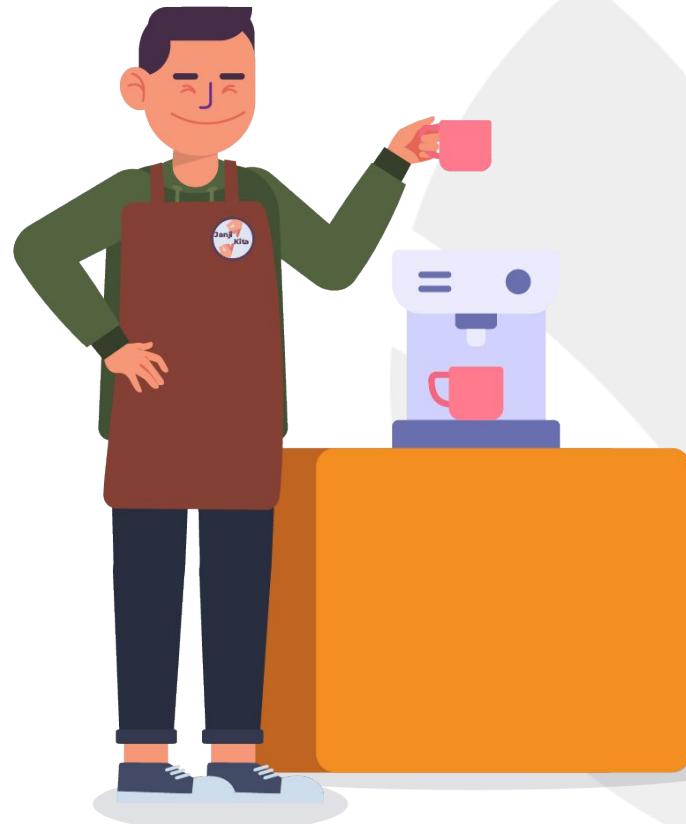
Kayak di dunia nyata, hubungan antara penyedia service dan klien pasti ada kontrak. **Kalau di backend engineering, "kontrak" antara server dan client itulah yang namanya API.**

Sederhananya, isi dari API adalah "perjanjian" yang dibuat antara server dan client tentang cara penyebutan data, data apa aja yang disalurkan, sampai ke cara pengambilan datanya.



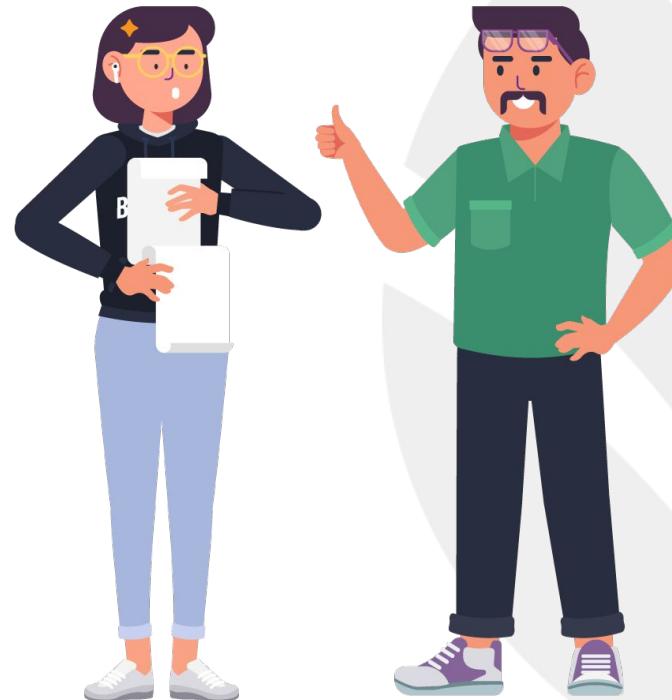
## Yuk kita move on ke frontend engineer!

Kalau backend engineer bisa dibilang kayak "coffee roaster", sekarang frontend engineer adalah "barista garis depan" yang tugasnya nyambut pengunjung, nyeduh kopi, menakar porsi di gelas ukur, nuangin ke gelas, sampai ngehias pakai gambar daun.



Dalam SDLC, seorang frontend engineer bertugas buat bikin apa yang udah didesain UI/UX designer dalam Hi-Fi design bisa jadi kenyataan.

Sebelumnya, di Hi-Fi design belum ada satu kode pun yang bisa diimplementasi. Di sini lah tempat frontend engineer bekerja. Nantinya dia harus bisa **menghubungkan desain yang udah dibikin sama UI/UX designer dan API yang udah dibikin backend engineer.**

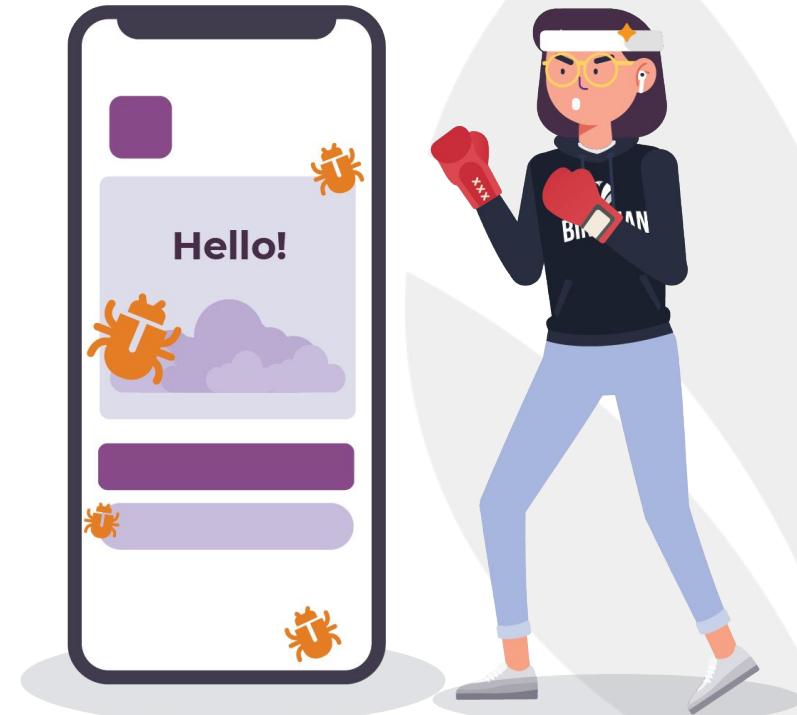


**Output yang dibikin dari hasil kolaborasi UI/UX designer, backend engineer, sama frontend engineer disebut prototype.**

Prototype sederhananya adalah Hi-Fi design bikinan UI/UX designer yang udah diisi coding sama frontend engineer dan udah terhubung ke API yang dibikin sama backend engineer. Alias: desain yang udah bisa dipencet-pencet.



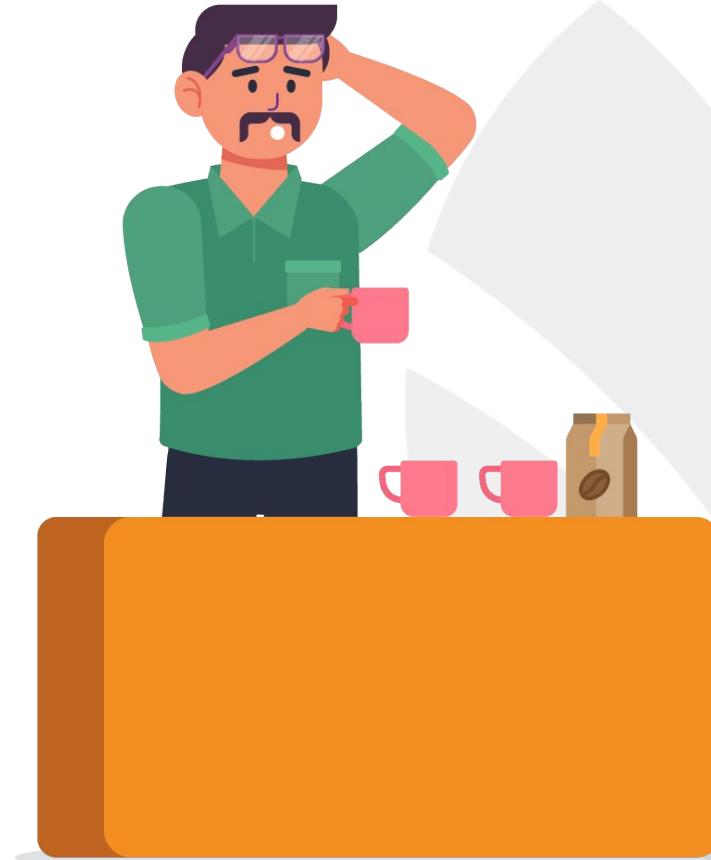
Umumnya, prototype masih dirilis terbatas di development environment dan biasanya masih punya banyak bug. Hal itu terjadi karena unit testing belum dilakukan oleh seorang **QA engineer**.

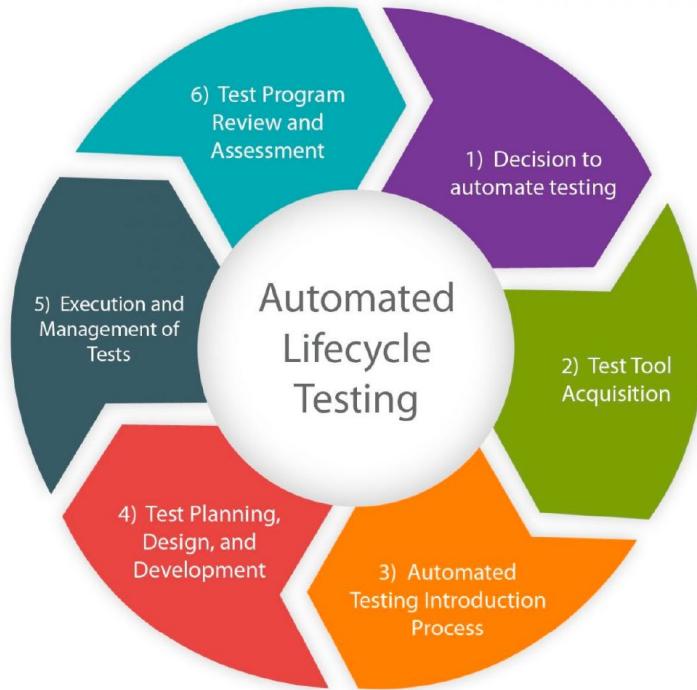


## Ngomong-ngomong soal unit testing, di sini peran QA engineer dibutuhin

QA itu singkatan dari Quality Assurance, alias, Si Penjamin Kualitas. **Tugas seorang QA engineer cukup berat, yaitu memastikan produk yang dirilis minim masalah, seperti bug dan error.**

Buat minimalisir masalah, QA engineer bertanggungjawab buat bikin serangkaian tes yang boleh dilakukan manual atau automated.





## Kenapa boleh dilakukan manual atau automated?

**Setiap tes punya karakteristik sendiri-sendiri.** Gak semua jenis tes bisa diotomatisasi dan beberapa jenis lain bisa melelahkan kalau dilakukan manual.



## Apa aja sih yang biasa dites sama QA engineer?

Buanyaak banget yang bisa dilakuin QA engineer buat mastiin bahwa produknya bener-bener anti cacat. Gak heran, kebanyakan QA engineer bersifat teliti dan perfeksionis.

Test Scenario ID	Login-1	Test Case ID	Login-1B				
Test Case Description	Login – Negative test case	Test Priority	High				
Pre-Requisite	NA	Post-Requisite	NA				
Test Execution Steps:							
S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Launch application	https://www.facebook.com/	Facebook home	Facebook home	IE-11	Pass	[Priya 10/17/201 7:11:44 AM]: Launch successful
2	Enter invalid Email & any Password and hit login button	Email id : invalid@xyz.com Password: *****	The email address or phone number that you've entered doesn't match any account. Sign up for an account.	The email address or phone number that you've entered doesn't match any account. Sign up for an account.	IE-11	Pass	[Priya 10/17/201 7:11:45 AM]: Invalid login attempt stopped

**Setiap tes yang dilakukan oleh QA engineer harus didokumentasikan secara rapi biar bisa dibaca backend dan frontend engineer.**

Jadi, banyak product owner yang dulunya QA engineer karena:

- Tahu luar-dalem permasalahan yang biasa kejadian di SDLC dan tahu cara minimalisirnya.
- Ngerti bahasa pemrograman yang dipakai di pembuatan API oleh backend engineer.
- Perhatian terhadap detail dan tahu cara dokumentasi teknis yang baik.

Gambar di samping adalah contoh dokumentasi teknis oleh QA engineer.

Kalau prototype adalah hasil kolaborasi yang belum dites sama QA engineer, **maka prototype akan masuk ke staging environment buat dites sama QA engineer.**

Sebagai tambahan, staging environment adalah environment yang menjembatani fase development dan production sebelum disetor ke Google Play Store atau Apple App Store.

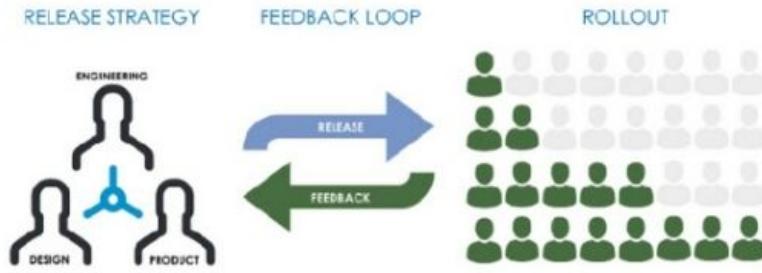


**Staging environment masih termasuk ke internal release, karena yang bisa akses cuma anak dalem kantor**

Cara rilisnya adalah dengan package system (APK atau IPA). Biasanya rilis kayak gini dipakai buat produk yang belum disetor ke Play Store atau App Store.

Ada kasus lain: bagi-bagi APK ini dilakukan karena belum dapet approval dari Play Store atau App Store.





Begitu udah dilakukan test dan siap rilis, ada proses khusus yang harus dilakukan sebelum full release. Namanya staggered release.

**Staggered release ini ciri khasnya yaitu udah bisa diakses publik, tapi masih terbatas.**

Kenapa dinamain staggered? Karena rilis ini dilakukan secara bertahap (rollout). Rilis kayak gini tujuannya buat ngetes seberapa stabil produk kita kalau dipakai sejumlah user.

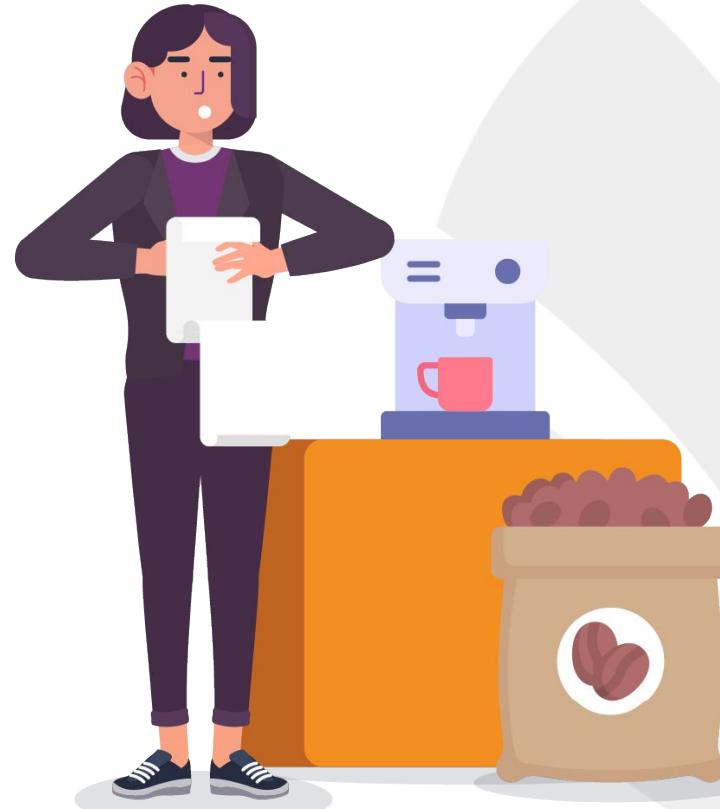
Contohnya, fitur dark mode di Instagram baru bisa dipakai di 5% user di hari pertama, 20% hari berikutnya, dst. sampai semua user bisa pakai.

## Kita balik lagi ya ke kasus coffee shop

Begitu dirilis, ternyata harus ada yang ngatur berjalannya alur pembuatan kopi. Mulai dari yang nyatet keluar-masuk barang, ngabsen team member, sampai ngawasin piket harian.

Akhirnya, salah satu barista inisiatif buat ngelakuin tugas itu. Dia seorang barista yang merangkap jadi operational manager. Atau dalam SDLC, sebutannya scrum master.

**Seorang scrum master bisa disebut leader, tapi secara hierarki, dia masih jadi bagian dalam tim.**



## Apa bedanya product owner sama scrum master? Kan sama-sama leader?

Beda dong. Mereka juga punya tanggung jawab lain.

- **Scrum master:** Gimana caranya keharmonisan dan kerja sama tim terjaga.
- **Product owner:** Gimana caranya produknya bisa product-market fit.

## SCRUM MASTER?



## PRODUCT OWNER?

Contoh lain mungkin kayak pemain bola. Scrum master kayak kapten tim dan product owner kayak pelatihnya.

Si kapten juga termasuk pemain, tapi leadership yang dia punya dipakai buat nyemangatin kalau ada yang drop atau ingetin temennya yang lagi bengong.

Tapi kalau pelatih, tugasnya adalah bikin timnya menang lewat taktik, strategi, pergantian pemain, dll.

## SCRUM MASTER?



## PRODUCT OWNER?

Sekarang balik lagi ke kasus coffee shop.

Karena kopinya laris, pembelinya membludak dan tempat duduknya mulai gak cukup. Mereka juga harus mikirin cara buat layanin pembeli via ojek online.

Akhirnya, mereka kepikiran buat nambah "fitur" drive-thru.

Nah, pembuatan drive-thru ini sama kayak pembuatan mobile app yang dikerjain sama mobile engineer (Android atau iOS).



**Nah, di topic ini kamu udah belajar banyak tentang Software Development Life Cycle. Kita rekap ulang yuk!**

- 1) Metodologi kerja Agile-Scrum dan bedanya sama Waterfall
- 2) Pembuatan roadmap oleh product owner
- 3) Persiapan, riset, dan desain oleh UI/UX designer



- 4) Pembangunan database, API, dan client-server model oleh backend engineer
- 5) Proses aktualisasi desain oleh frontend engineer
- 6) Pengertian tentang app testing oleh QA engineer
- 7) Bedanya product owner sama scrum master
- 8) Kolaborasi pembuatan konsep sampai software staging di dapur app development
- 9) Tahapan rilis produk dan environment yang ada dalam SDLC



Yuhuuu! Kamu udah berhasil  
menamatkan Chapter 0 Topic 3 🎉

Nanti, di topic 4 kita bakal ngebahas  
tentang jenis-jenis kode, bahasa  
pemrograman, dan soal mobile  
engineering. Udah siap?

