BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Proses pelaksanaan kerja magang di PT GOTO Tbk dimulai dengan tahap onboarding di perusahaan GOTO. Fokus onboarding adalah untuk mendapatkan pemahaman yang lebih mendalam tentang visi dan misi perusahaan, budaya kerja, serta beberapa tugas yang perlu dilaksanakan selama bekerja di PT GOTO Tbk. Penempatan pada magang ini terjadi di anak perusahaan, khususnya di divisi *data warehouse* pada tim *data fabric experience*. Dalam konteks ini, berada di bawah bimbingan Rahmat Hidayat, yang menjabat sebagai mentor dan *Senior Software Engineer* di tim tersebut, menjadi suatu pengalaman yang membawa wawasan dan arahan yang sangat berharga. Proses *onboarding* ini memberikan landasan yang kokoh untuk memahami lingkungan kerja dan tanggung jawab yang akan diemban selama menjalani magang di perusahaan ini.

3.2 Tugas yang Dilakukan

Proses kerja magang di dalam tim *data fabric experience* melibatkan pengembangan sistem *frontend* dan *backend* dashboard data GOJEK. Di dalam tim, tidak terdapat perbedaan pekerjaan antara Software Engineer maupun Software Engineer Intern, dapat berupa *story* atau bisa dikatakan tugas, *review code* dari kolega, dan melakukan analisa.

Team data fabric experience fokus mengerjakan 3 servis utama yang merupakan *guardian*(sistem *backend* untuk pemberian akses, *dex* (sistem *backend serverless* untuk menjadi perantara autentikasi dengan *data resource)*, dan *datlantis*(dashboard frontend) dengan menggunakan teknologi-teknologi berupa :

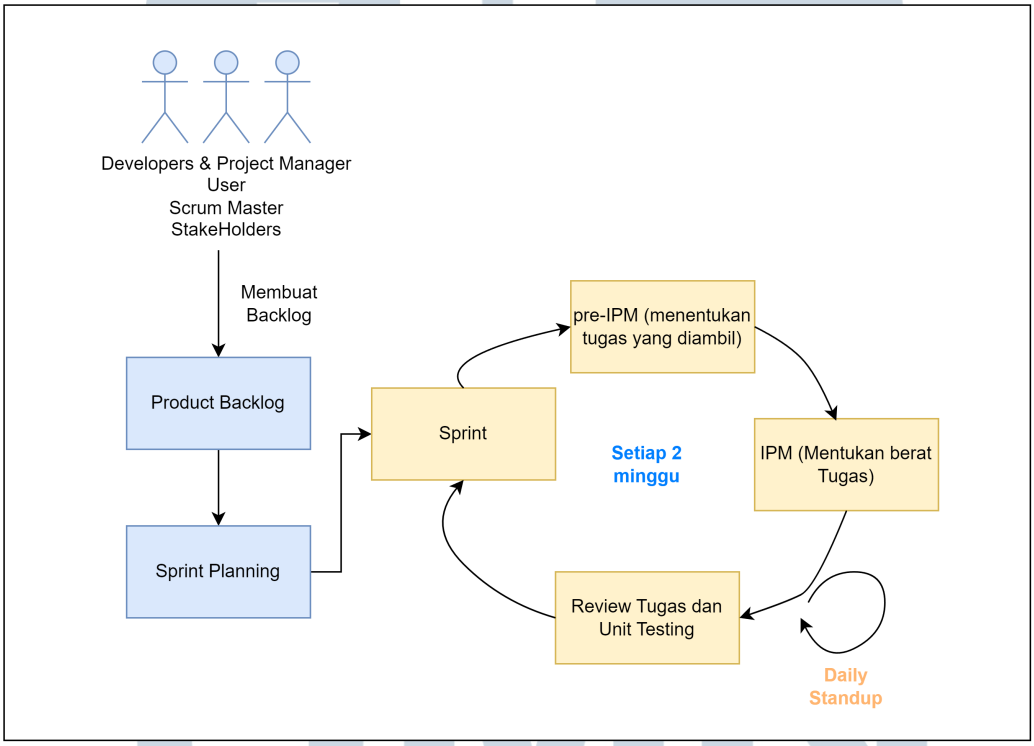
- Bahasa Go (*backend*)
- React Typescript (*frontend*)
- REST API

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	- *Onboarding* GOTO secara general - *Onboarding* Team dan menyiapkan laptop dan konfigurasinya
2	- *Onboarding* ke aplikasi Guardian yang akan dikerjakan - Mengerjakan tugas untuk membuat *pagination* untuk *backend* Guardian
3	- Mengintegrasikan tampilan *pagination* pada *frontend* yang telah dibuat pada *backend* - Membuat api untuk *filter* dan *search*
4	- Mengintegrasikan tampilan *search & filter* pada *frontend* melalui api - Membuat api baru untuk *filter*
5	- *QA test* buat story yang telah dibuat - Memperbaiki *bug* pada *filter*
6	- Mendapat *task* baru yaitu mengubah *flow* dari *form* untuk *appeal request*
7	- Memperbaiki dan *QA test* dari *task* pengubahan form
8	- Mendapatkan *task* baru membuat *component* baru - Mendapatkan *task* baru membuat api untuk DEX
9	- Kerja bersama mentor untuk membuat api pembuatan *resource* untuk DEX - *Meeting* dengan tim *Data Streaming* untuk mendapatkan *requirements* untuk api
10	- Membuat *testing* dan *test cases* di GO buat api DEX
11	- Menyiapkan dan memperbaiki api untuk membuat api *resource*
12	- Mendapatkan task baru untuk mengintegrasikan dan membuat form untuk api di *frontend*
13	- Melakukan testing dan menyiapkan task pembuatan integrasi api di *frontend*
14	- Mendapatkan task baru untuk menambahkan fitur untuk *update* data dengan *key* baru
15	- Melakukan testing dan *meeting* bersama divisi *data batching*
16	- Mendapatkan task baru untuk memindahkan sebuah *plugin* ke *framework* baru

3.2.1 Uraian Tugas

Pada bagian ini akan dijelaskan tugas-tugas yang telah dilakukan selama proses kerja magang. Pada proses kerja pada tim *fabric experience* menggunakkan *Agile Software Development Cycle* dengan bantuan alat *JIRA Software* untuk membuat *storyboard*.



Gambar 3.1. Software Development Life Cycle

Proses kerja atau *Software Development Life Cycle* dilakukan secara *agile methodology* dengan bantuan Kanban yang menggunakkan *software* JIRA Board. Proses diawali dengan perencanaan, Developers, Project Manager, User, Scrum Master, StakeHolders dapat membuat sebuah *story* atau tugas. Kemudian *story* tersebut akan dianalisa *requirements-requirements* yang dibutuhkan. Dari requirements tersebut, *senior, lead, dan manager engineer* akan mendesign solusi apa saja yang paling optimal untuk *requirements* di atas. Setelah cukup banyak *story* sudah terkumpul maka akan masuk ke tahap berikutnya

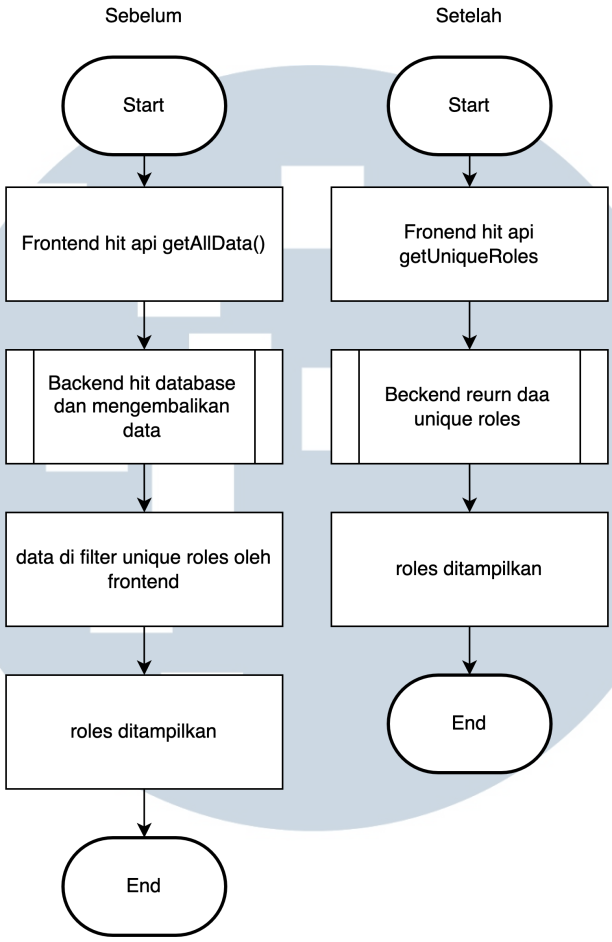
Tugas-tugas/*stories* akan di rangkum dalam sebuah *sprint* yang berlangsung

selama 2 minggu. Setiap 2 minggu akan dilakukan pre-IPM dan IPM yang merupakan meeting untuk menentukan *story* apa saja yang menjadi prioritas dan menentukan*story point* yang berupa angka atau keberatan tugas tersebut. Setelah di tentukan artinya sprint sudah dimulai. *Stories*/tugas tersebut akan diambil oleh skala prioritas. Untuk setiap tugas yang diberikan, akan ada persyaratan khusus yang harus dipenuhi, dan durasi waktu penyelesaian tugas akan ditentukan bersama-sama. Pemantauan dan implementasi dibantu dengann penggunaan git. Saat sebuah tugas dimulai, akan dibuat sebuah *branch* baru pada *repository* tersebut. Setelah sebuah tugas telah diselesaikan, akan masuk ke tahap *review* dan *QA testing*.

Tahap pengujian pada tim *fabric-experience* ada 3 bagian, lokal, staging, dan produksi. lokal merupakan tahap dimana pengujian dilakukan dengan data pribadi dan pada koneksi pribadi. Setelah pada lokal proses *review* dan *testing* sudah berhasil maka akan berlanjut pada tahap berikutnya yaitu staging. Staging merupakan tahap dimana perubahan dari branch kita sudah dilakukan *pull request/merge request*. Pada tahap ini dilakukan integrasi perubahan ke dalam kode yang akan di *push* ke tahap produksi. Setelah semua *testing* dan pengujian telah dilakukan maka kita kan membuat *release* baru dan *deploy* ke produksi. Setelah *deploy* akan dilakukan proses pemantauan jika terjadi hal-hal yang tidak diinginkan atau tidak sesuai dengan tujuan awal. Jika sudah terkendali, maka dapat dirangkum bahwa tugas yang dilakukan telah berhasil di integrasikan dan di *deploy*. Setelah setipa sesi 2 minggu dalam pre-IPM, akan dilakukan review sprint, berapa *story* yang berhasil dikerjakan, sedang dijalankan, dan dalam proses pengujian. Hal tersebut yang akan menjadi tolak ukur untuk pembuatan *sprint* berikutnya.

A. Pembuatan dan integrasi api List Roles

Pembuatan fitur api ini memiliki tujuan untuk melakukan *filtering* data dengan *roles* pada *dashboard frontend*. Fitur *filter* ini awalnya sudah ada namun dilakukan di *frontend* yang kurang optimal. Sebelum perubahan, filtering roles dilakukan di frontend. Pembuatan dan integrasi API List Roles bertujuan meningkatkan efisiensi. Maka dibuat sebuah api yang khusus untuk mendapatkan semua *roles* dari tabel *database*. Pembuatan api menggunakan bahasa GO sebagai backend dengan bantuan gRPC untuk proses pembuatan api. Pada bagian *Frontend* digunakan React untuk pembuatan komponen dan api.[3]



Gambar 3.2. Flowchart pengubahan proses pengambilan *roles*

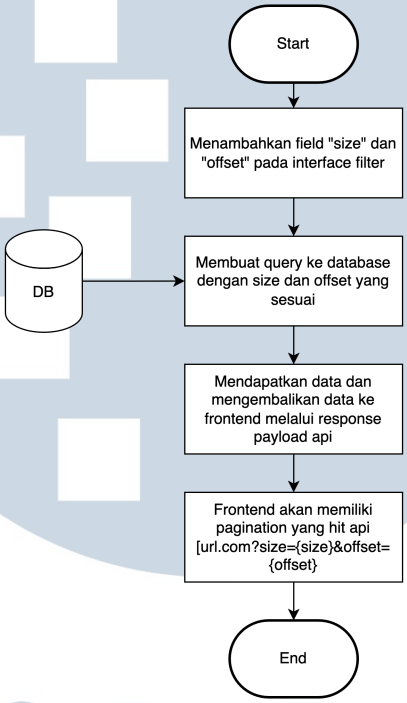
Pada Gambar 3.2, dengan hasil pembuatan api tersebut di atas, proses dapat dilakukan lebih cepat dan mengeluarkan proses yang tidak seharusnya terjadi di *frontend*.

B. Membuat pagination

Pada fitur yang dikerjakan sekarang, implementasi pagination menjadi sebuah langkah krusial dalam meningkatkan efisiensi pengambilan data dari server ke frontend. Dengan menggunakan teknik pagination, kita dapat memecah dataset besar menjadi halaman-halaman kecil yang dapat diambil secara terpisah. Dua parameter utama yang digunakan dalam pagination ini adalah *size* dan *offset*.

Size merujuk pada jumlah item atau data yang diambil pada setiap halaman. Sebagai contoh, jika ukuran halaman diatur menjadi 10, maka setiap permintaan

data akan mengembalikan 10 item. Di sisi lain, *offset* menentukan titik awal dari mana data pada halaman tertentu harus diambil. Jika kita memiliki halaman 2 dengan ukuran 10 dan offset 10, kita akan melompati 10 item pertama dan mulai mengambil data dari item ke-11. Pada gambar 3.3 berikut akan menjelaskan proses kerja pembuatan pagination.



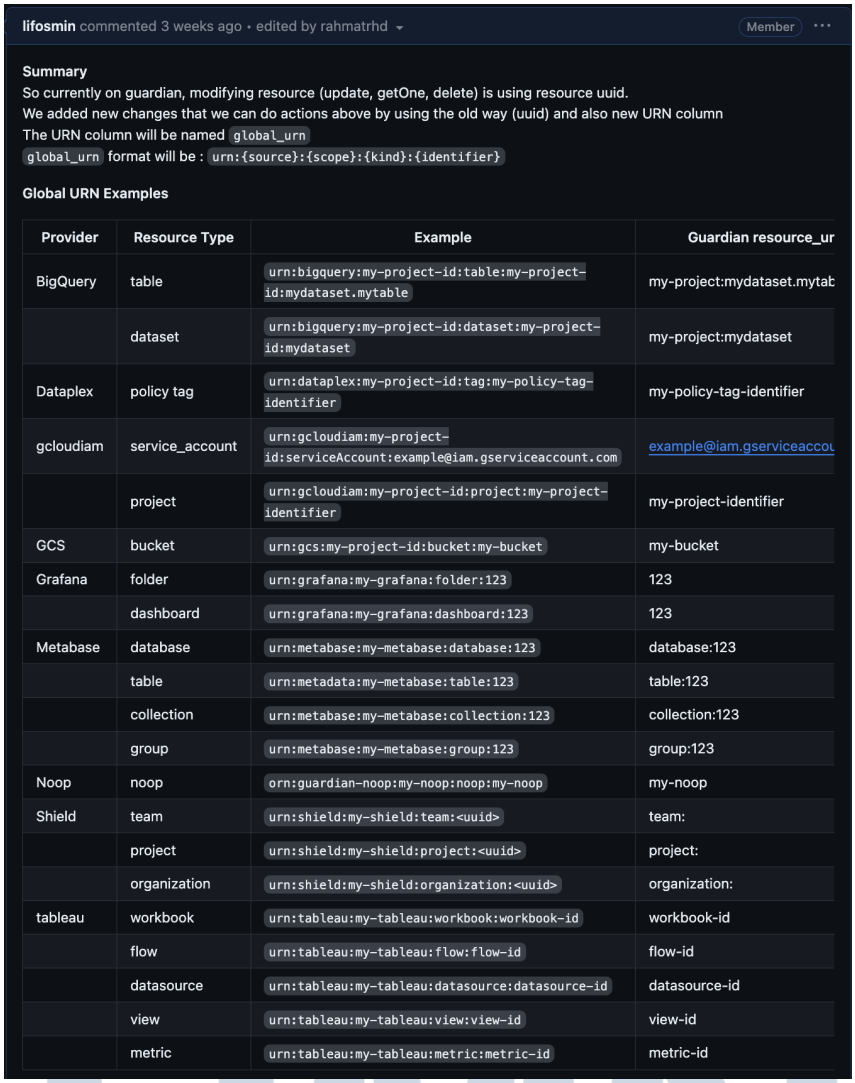
Gambar 3.3. Flowchart pagination

Dengan menerapkan *size* dan *offset*, kita dapat mengoptimalkan kinerja aplikasi. Misalnya, dalam kasus dataset berjumlah 100 item, halaman-halaman yang dibuat akan mengambil sejumlah item yang ditentukan dengan ukuran tertentu dan memulai dari posisi yang ditentukan dengan *offset*. Hal ini tidak hanya meningkatkan responsivitas aplikasi dengan memungkinkan pengguna untuk melihat sebagian dari data dengan cepat, tetapi juga mengurangi beban server karena hanya sejumlah kecil data yang dikirimkan pada satu waktu. Dengan demikian, pengguna akan mendapatkan pengalaman yang lebih baik tanpa perlu menunggu seluruh dataset diunduh, sementara server tetap efisien dalam penggunaan sumber daya.

C. Membuat variable baru untuk *identifier* melakukan query data spesifik

Pada bagian ini, akan ditambahkan pada Guardian. Guardian adalah alat untuk akses data yang dapat diperluas dan universal dengan alur kerja akses otomatis dan kontrol keamanan di seluruh penyimpanan data, sistem analitis, dan produk cloud. Guardian menyediakan akses data yang dinamakan *resource* dan disediakan oleh *provider*. Pengambilan atau *query resource* sekarang hanya dilakukan dengan menggunakan kata kunci id *resource*. Pada tugas ini akan ditambahkan sebuah *identifier* baru bernama global$_u$*rn*.[4]

Proses pembuatan variabel ini dilakukan secara analisa dan diskusi antar tim dan rekan kerja. Pembuatan variabel ini ditentukan sebagai *unique index* dan memiliki format. Format ada di gambar 3.4 sebagai berikut



lifosmin commented 3 weeks ago • edited by rahmatrhd — Member

Summary

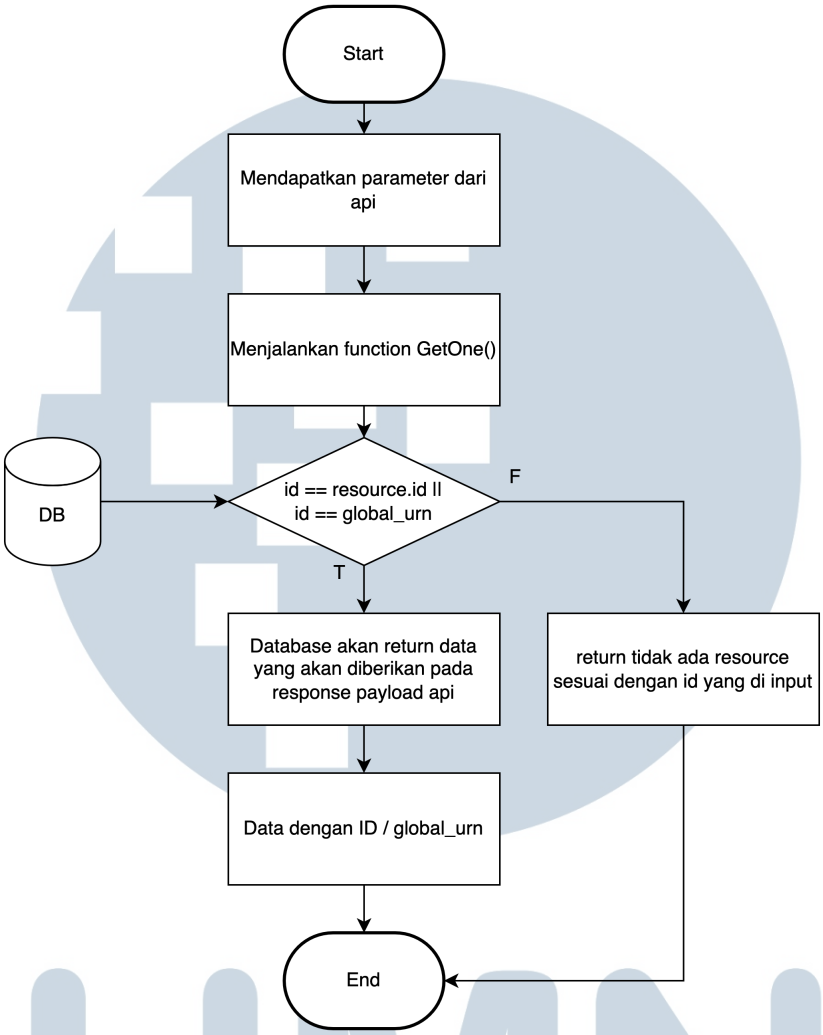
So currently on guardian, modifying resource (update, getOne, delete) is using resource uuid.
We added new changes that we can do actions above by using the old way (uuid) and also new URN column
The URN column will be named `global_urn`
`global_urn` format will be : `urn:{source}:{scope}:{kind}:{identifier}`

Global URN Examples

Provider	Resource Type	Example	Guardian resource_ur
BigQuery	table	`urn:bigquery:my-project-id:table:my-project-id:mydataset.mytable`	my-project:mydataset.mytab
	dataset	`urn:bigquery:my-project-id:dataset:my-project-id:mydataset`	my-project:mydataset
Dataplex	policy tag	`urn:dataplex:my-project-id:tag:my-policy-tag-identifier`	my-policy-tag-identifier
gcloudiam	service_account	`urn:gcloudiam:my-project-id:serviceAccount:example@iam.gserviceaccount.com`	example@iam.gserviceaccou
	project	`urn:gcloudiam:my-project-id:project:my-project-identifier`	my-project-identifier
GCS	bucket	`urn:gcs:my-project-id:bucket:my-bucket`	my-bucket
Grafana	folder	`urn:grafana:my-grafana:folder:123`	123
	dashboard	`urn:grafana:my-grafana:dashboard:123`	123
Metabase	database	`urn:metabase:my-metabase:database:123`	database:123
	table	`urn:metadata:my-metabase:table:123`	table:123
	collection	`urn:metabase:my-metabase:collection:123`	collection:123
	group	`urn:metabase:my-metabase:group:123`	group:123
Noop	noop	`orn:guardian-noop:my-noop:noop:my-noop`	my-noop
Shield	team	`urn:shield:my-shield:team:<uuid>`	team:
	project	`urn:shield:my-shield:project:<uuid>`	project:
	organization	`urn:shield:my-shield:organization:<uuid>`	organization:
tableau	workbook	`urn:tableau:my-tableau:workbook:workbook-id`	workbook-id
	flow	`urn:tableau:my-tableau:flow:flow-id`	flow-id
	datasource	`urn:tableau:my-tableau:datasource:datasource-id`	datasource-id
	view	`urn:tableau:my-tableau:view:view-id`	view-id
	metric	`urn:tableau:my-tableau:metric:metric-id`	metric-id

Gambar 3.4. Format penamaan varibel baru

Setelah menentukan format varibel, kemudian dilakukan implementasi pada guardian dengan flowchart berikut pada gambar 3.5



Gambar 3.5. Flowchart proses pengambilan data dengan id

Proses ini dapat membantu meningkatkan skalabilitas sistem, langkah yang diambil adalah membuat variabel baru dari hasil query terhadap *resource* tertentu denga *provider* yang ditentukan. Dengan cara ini, kita dapat mengelola dan memanfaatkan informasi yang diperoleh dari penyedia layanan tersebut secara lebih efisien, menjadikan sistem lebih mudah diperluas, dan memberikan fleksibilitas dalam mengelola sumber daya. Dengan variabel baru ini, dapat dilakukan berbagai operasi dan manipulasi data yang diperlukan untuk mendukung kebutuhan sistem secara menyeluruh.

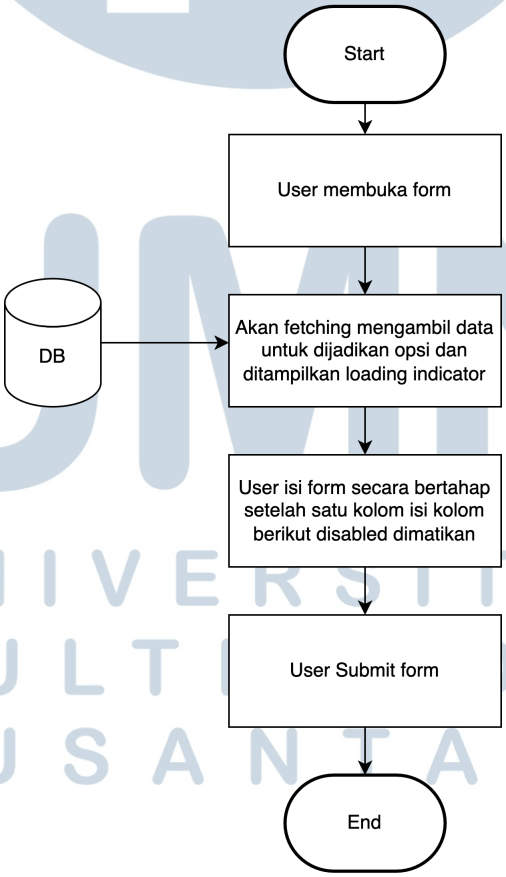
D. Memperbaiki Antarmuka Dashboard Form

Tugas ini melibatkan penyempurnaan Desain Antarmuka Pengguna (UI) dan Pengalaman Pengguna (UX) saat ini untuk meningkatkan keseluruhan kegunaan dan keterlibatan aplikasi. Ini mencakup penyempurnaan elemen visual, optimalisasi navigasi, dan memastikan perjalanan pengguna yang mulus dan intuitif. Pengubahan *Form* ini menggunakan bahasa React.

Pengubahan ini dimulai dengan *width* untuk kolom *input* yang kecil dan UX untuk saat pengisian form yang masih membuat pengguna bingung. Tugas ini telah ditambahkan beberapa fitur baru sebagai berikut

- *width input* yang awalnya kecil di ubah menjadi lebih luas
- *flow* pengisian form dilakukan secara bertahap sehingga kolom yang belum seharusnya diisi akan di *disabled*

Pada gambar 3.6 akan menunjukan *flow* user mengisi form



Gambar 3.6. Flowchart pengisian form

3.3 Kendala dan Solusi yang Ditemukan

A. Kendala yang Ditemukan

Berikut merupakan kendala yang ditemukan saat proses pelaksanaan magang

- Ketidakfahaman terhadap layanan-layanan, struktur kode dan istilah yang digunakan pada tim Gojek
- Kendala komunikasi karena sebagian tim yang berada pada Bangalore, India

B. Solusi

Berikut merupakan solusi untuk kendala kendala yang di temukan

- Bertanya kepada mentor dan melakukan banyak literatur tentang layanan-layanan utama supaya menjadi lebih familiar
- Memanfaatkan Google Meet sebagai platform untuk pertemuan atau diskusi, dan selalu bersikap aktif dalam berkomunikasi berlebihan untuk mengurangi risiko terjadinya miskomunikasi.