BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Praktik kerja magang dilakukan pada departemen *software engineering*, khususnya pada Squad C yang bergerak pada bidang Android *mobile app development* sebagai Android *developer intern*. Squad ini dikepalai oleh Dennis selaku kepala departemen *software engineering* dan *interim developer tech lead* sebelum digantikan oleh Made Wahyu Kharisma Candra Kirana selaku *developer tech lead*. Squad ini bertugas untuk melakukan *maintainance* dan *development* dari salah satu produk digital Harian Kompas, yaitu Kompas.id berbasis Android.

Pada saat praktik magang berjalan, Squad C terdiri dari 4 *developer*, 1 *developer tech lead*, 2 QA, dan beberapa anggota dari divisi *product*. Jumlah anggota dari divisi *product* pada Squad C tidak pasti karena tim *product* bekerja berdasarkan *funnel*. Hal ini berarti tim *product* ditugaskan berdasarkan inisiatif yang sedang berjalan, bukan *squad*. Dalam satu *sprint* yang berdurasi dua minggu, terdapat beberapa *sprint ceremonies meeting* seperti pada tabel 3.1.

Tabel 3.1. Jadwal *Sprint ceremonies* yang dilakukan dalam satu *sprint*

Hari	*Sprint Ceremony*
Minggu Ke - 1	
Senin	*Sprint Review* + *Retrospective* (*Tentative*)
Selasa	*Sprint Planning*
Rabu	-
Kamis	*Daily Stand Up*
Jumat	*Daily Stand Up* + *Report Severity Bug*
Minggu Ke - 2	
Senin	*Daily Stand Up*
Selasa	*Daily Stand Up* + *Sprint Grooming*
Rabu	*Daily Stand Up*
Kamis	*Daily Stand Up*
Jumat	*Daily Stand Up*

Dalam *sprint review*, dilakukan peninjauan pada terhadap semua *task* atau yang biasa disebut *cards* yang dikerjakan pada akhir *sprint* berjalan. Dalam

meeting, QA menunjukkan hasil dari *cards* yang telah dikerjakan, baik yang merupakan fitur baru, peningkatan fitur yang sudah ada, ataupun *bug fix*. Lalu *tech lead* dan anggota tim *product* yang berkaitan dengan *cards* yang ada akan mengkonfirmasi apakah *card* tersebut dapat di tutup atau akan dibawa ke *sprint* selanjutnya apabila belum sesuai kriteria ataupun selesai. Tidak seluruh anggota tim perlu hadir dalam pertemuan ini dengan pertimbangan bahwa para anggota tim dapat menggunakan waktu ini untuk mengerjakan pekerjaan lain yang dimilikinya. Secara *tentative*, *sprint retrospective* juga dilakukan oleh para anggota tim untuk berdiskusi mengenai kinerja tim serta menyampaikan berbagai masukkan atau kritik yang dimilikinya, baik mengenai alur kerja tim, efisiensi pekerjaan, masalah pribadi, dan sebagainya.

Lalu dalam *sprint planning*, dilakukan perencanaan mengenai *cards* yang akan diambil pada *sprint* selanjutnya. Hal ini meliputi penentuan *story point* yang menunjukkan estimasi tingkat kesulitan suatu *card* yang harus dikerjakan oleh para anggota tim. *Story point* dapat dijadikan sebagai suatu patokan untuk menentukan durasi waktu yang diperlukan untuk menyelesaikan suatu *card* seperti pada tabel 3.2. Lalu, para anggota tim berdiskusi untuk menentukan siapa yang akan mengambil *cards* yang ada. Setelah itu, akan dilakukan estimasi waktu untuk setiap kartu oleh pemiliknya. Terakhir, apabila estimasi waktu melebihi jumlah *working hours* yang tersedia, maka akan terdapat *card/s* yang dipindahkan ke *sprint* selanjutnya.

Tabel 3.2. Estimasi waktu pengerjaan berdasarkan *Story point*

Story Point	*Max Work Hours*
1	4 Jam
2	8 Jam
3	16 Jam
5	24 Jam
8	40 Jam
13	48 Jam

Pada *daily stand up*, setiap anggota tim melaporkan apa yang dilakukannya pada hari sebelumnya dan apa yang mau dikerjakannya pada hari tersebut. Anggota tim juga dapat melaporkan kendala-kendala yang ditemui saat mengerjakan suatu *card*. Notulen akan mencatat hal-hal tersebut dan juga menulis angka Crashlytics dari Firebase dalam bentuk persentase untuk menunjukkan stabilitas

dari versi aplikasi terbaru serta *all-version*. Terdapat juga pertemuan *severity bugs*, dimana QA akan memaparkan *bugs-bugs* yang telah ditemukan dari *regression testing* yang telah dilakukan dan anggota-anggota tim akan berdiskusi untuk mengklasifikasikannya menjadi beberapa kategori, seperti *improvement*, *not implemented*, *out of scope*, *on progress*, *bug fix*, ataupun *hot fix*. Terdapat juga tingkat severity dari *critical* hingga *minor*.

Terakhir, terdapat *sprint grooming* dimana anggota tim akan membahas mengenai *cards* yang akan dijalankan pada *sprint* selanjutnya. QA dan *tech lead* akan memaparkan informasi mengenai *cards* yang akan ada dan berdiskusi dengan anggota tim. Hal ini dapat meliputi strategi pendekatan terhadap *cards* tersebut serta apakah memungkinkan untuk semua *cards* untuk diselesaikan pada *sprint* selanjutnya.

3.2 Tugas yang Dilakukan

Dalam pelaksanaan kerja magang, dilakukan beberapa jenis tugas sebagai berikut.

1. Memahami *codebase* dan *flow* dari aplikasi Kompas.id pada sistem aplikasi *mobile* Android dengan bimbingan rekan-rekan kerja.
2. *User story*, yaitu pembuatan fitur baru berdasarkan permintaan tim *product* sebagai berikut.
    - Fitur *log status*.
    - Fitur *membership error*.
    - Fitur Kompas.id versi *landscape*.
3. *Improvement*, yaitu melakukan pengembangan terhadap fitur-fitur yang sudah ada sebelumnya sebagai berikut.
    - *Shimmer loading* pada halaman akun saya.
    - *Refresh* detail akun pada halaman akun saya.
4. *Bug fix*, yaitu perbaikan terhadap kekurangan-kekurangan ataupun *bugs* yang ditemukan oleh pihak QA sebagai berikut.
    - *Placeholder coachmark* tersangkut.

- Tampilan REPOLA muncul berulang.
- Video pengguna REGON terblokir *overlay paywall* saat masih mempunyai kuota.
- *Overlay paywall* video dengan harga *null* bagi pengguna ANON yang tidak memiliki kuota.
- Bagian topik terkait masih muncul walaupun kosong pada detail artikel.
- Toast *bookmark* menghalangi *metered paywall* pada detail artikel.
- TETOPI bar muncul sesaat ketika *login* akun REGON dari *paywall*.
- TETOPI bar tidak muncul pada halaman *tab* terbaru.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang pada Harian Kompas sebagai Android Developer intern dapat diuraikan seperti pada tabel 3.3.

Tabel 3.3. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke-	Pekerjaan yang dilakukan
1-2	Mengerjakan *learning module* dan mempelajari *codebase*.
3-4	Membuat fitur *log status*.
5-6	Membuat fitur *log status*, membuat fitur *membership error* dan *bug fixing*.
7-8	Mengerjakan *Bug fixing* dan *improvement*.
9-10	*Bug fixing* dan membuat fitur *membership error*.
11-12	*Bug fixing* dan *improvement* pada pada menu status langganan pada halaman dasbor.
13-14	*Bug fixing* dan membuat fitur iframe *banner* Kompas.id mode *landscape*.
15-16	*Bug fixing* fitur REPOLA muncul berulang dan TETOPI *bar* tidak muncul
17-18	Membuat halaman *E-Paper* Saya versi *landscape*

Pada minggu pertama dan kedua pelaksanaan magang, rekan-rekan developer memberikan sebuah learning module berupa pembuatan sebuah aplikasi sederhana. Aplikasi ini memuat fitur berupa menampilkan daftar *game* dari sebuah

API, *unlimited scroll* dengan *paging*, *search*, menampilkan detail dari sebuah *game*, serta menyimpan *game* favorit pada basis data lokal. Terdapat juga beberapa persyaratan seperti menggunakan jetpack compose, MVVM, *dependency injection*, dan sebagainya. Pembelajaran *codebase* juga dilakukan.

Pada minggu ketiga dan keempat, dilakukan pembuatan fitur *log status*. Fitur ini menambahkan suatu tampilan mengenai data pengguna serta perangkat yang digunakannya. Hal ini meliputi email, GUID, ID transaksi apabila pengguna telah berlangganan, tipe perangkat, versi OS, versi aplikasi yang terpasang pada perangkat pengguna, serta versi aplikasi terbaru. Terdapat juga tombol untuk menyalin seluruh informasi tersebut. Fitur ini dikembangkan untuk memudahkan pengguna dalam memberikan informasi-informasi yang diperlukan kepada *hotline* ketika menemukan suatu masalah pada aplikasi Kompas.id. Namun, dikarenakan beberapa faktor seperti masih kurangnya pengertian pada *codebase* serta *architecture pattern* yang digunakan, tugas ini belum selesai dan disalin pada *sprint* selanjutnya.

Pada minggu kelima dan keenam, dilanjutkan pembuatan fitur *log status* sampai selesai, dilakukan pembuatan fitur mengenai *membership error*, serta melakukan berbagai *bug fixing*. Fitur *membership error* yang dikerjakan adalah *banner* pada paywall untuk halaman *text-to-speech* atau disingkat TETOPI, *e-paper*, serta *e-book*. *Banner* ini akan muncul apabila pengguna telah membeli paket berlangganan, namun data dari *backend* masih menunjukkan bahwa akun pengguna belum memilikinya.

Pada minggu ketujuh dan kedelapan, dilakukan sebuah *improvement* dan berbagai *bug fixing*. Tugas *improvement* meliputi penambahan loading state pada menu informasi saya di dasbor yang sebelumnya menunjukkan data default menjadi sebuah *shimmer*. Dilakukan juga perbaikan pada *banner membership error* yang masih belum sesuai dengan desain, yaitu warna pada *dark mode* serta jarak *banner* dari atas layar. Lalu dilakukan juga perbaikan atas *bugs* pada halaman detail artikel video yaitu pengguna *register-only* atau REGON tidak dapat memutar video saat kembali dari halaman berlangganan ketika masih punya kuota.

Pada minggu kesembilan dan kesepuluh, dilakukan pembuatan *banner membership error* pada menu Akun Saya dan Status Langganan di Dasbor serta detail artikel, serta beberapa *bug fixing*. Pada menu Dasbor, ditambahkan banner pada halaman Akun Saya apabila akun pengguna mengalami *membership error* serta placeholder pada halaman Status Langganan. *Bug fixing* yang dilakukan adalah ketika pengguna *anonymous* atau ANON tidak punya kuota, maka

overlay paywall akan muncul dengan harga *null* ketika seharusnya terdapat harga berlangganan yang tertera. Selain itu, juga dilakukan perbaikan pada halaman detail artikel dimana *section* topik terkait tetap muncul ketika kosong.

Pada minggu kesebelas dan keduabelas, dilakukan berbagai *bug fixing* dan *improvement* terhadap *behaviour refresh* saat pengguna mendapatkan *membership*. *Improvement* ini menambahkan fitur *scroll-to-refresh* pada menu akun saya pada dasbor sehingga ketika user mendapatkan *membership*, pengguna dapat melakukan *refresh* untuk memperbaharui informasi mengenai status langganan pada menu informasi saya. Dilakukan juga *bug fixing* yang berupa *placeholder coachmark* yang tersangkut ketika di *click-to-skip* dengan cepat, TETOPI bar yang muncul sesaat saat seharusnya tidak.

Pada minggu ketigasbelas dan keempatbelas, dilakukan pengembangan *widget* pada *mainscreen* Kompas.id versi *landscape*. Widget disesuaikan untuk penampilan Kompas.id versi *landscape* dan diletakkan pada posisi layar paling atas yang sebelumnya berada dibawah artikel pertama.

Pada minggu kelimabelas dan keenambelas, dilakukan beberapa *bug fixing*. Bug fixing yang dilakukan adalah TETOPI bar tidak muncul pada halaman *tab* terbaru dan REPOLA menjadi *double* ketika pengguna *idle* selama sekitar 15 menit. Terakhir, pada minggu ketujuhbelas dan kedelapanbelas, dilakukan *improvement* terhadap tampilan *cardthumb* serta topik hangat pada Kompas.id versi *landscape*. Dilakukan juga pembuatan halaman *E-Paper* Saya mode *landscape*.

3.3.1 Fitur / *User Story*

A. *Log status*

A.1 *User Requirement*

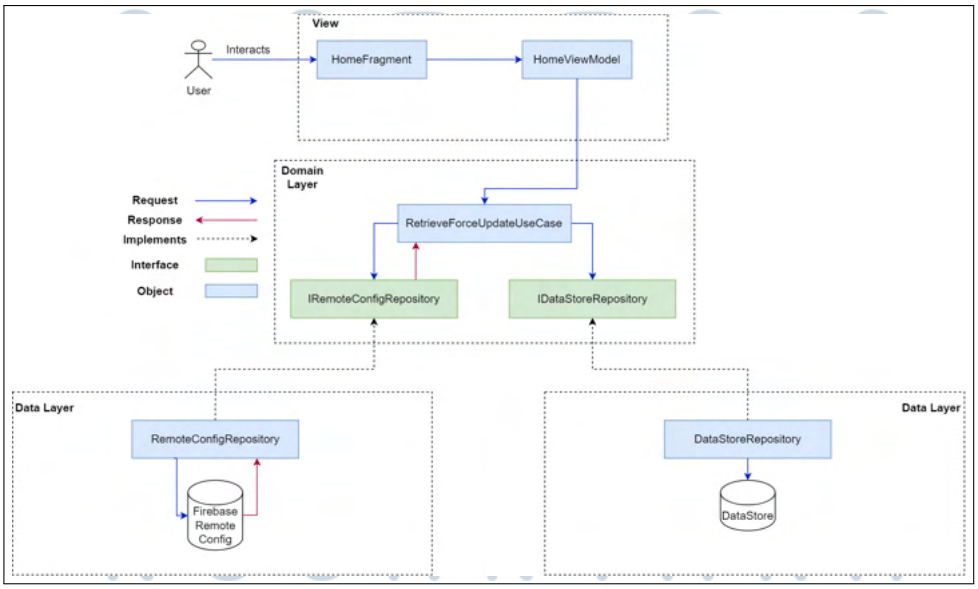
Sebelum fitur ini dikembangkan, *hotline* Harian Kompas mengalami kendala dimana para pengguna yang ingin mengajukan suatu komplain mengalami kesulitan untuk mencari informasi-informasi yang diperlukan oleh *hotline* untuk melakukan *issue tracing*.

Untuk mengatasi kendala tersebut, tim *product* telah mendesain sebuah halaman yang menampilkan informasi-informasi penting yang diperlukannya dan dapat disalin oleh pengguna dengan mudah. Pengguna kemudian harus dapat menyalin seluruh informasi ini dengan menekan sebuah tombol. Informasi-informasi yang akan ditampilkan adalah sebagai berikut.

1. Email: Email yang digunakan oleh pengguna dan hanya tersedia sebagai REGON (hanya memiliki akun) ataupun SUBER (pengguna yang telah berlangganan).
2. GUID (*Guest User ID*): ID yang bersifat unik untuk setiap pengguna REGON ataupun SUBER.
3. ID Transaksi: ID transaksi yang hanya ada untuk SUBER.
4. Tipe Perangkat: Tipe perangkat yang digunakan oleh pengguna.
5. Versi OS: Versi OS dari perangkat pengguna.
6. Versi Saat Ini: Versi aplikasi Kompas.id yang terpasang pada perangkat pengguna saat ini.
7. Versi Terbaru: Versi aplikasi Kompas.id terbaru yang tersedia pada Playstore.

A.2 Perancangan

A.2.1 Pengambilan Data Max Version dan Version Code dari Firebase Remote Config



Gambar 3.1. *Flow* Pengambilan Data dari Firebase Remote Config

Gambar 3.1 merupakan *flow* pengambilan data *max version* serta *version code* yang berasal dari Firebase Remote Config yang akan digunakan pada fitur

log status untuk disimpan pada DataStore secara lokal. Saat pengguna membuka aplikasi dan sampai pada halaman utama, maka proses ini akan dimulai.

Saat pengguna membuka *homepage* dari aplikasi, maka sistem akan mengakses HomeViewModel untuk melakukan *invoke* terhadap *use case* RetrieveForceUpdateUseCase untuk melakukan pengambilan data *max version* dan *version code* aplikasi dari Firebase Remote Config. Di dalam use case tersebut, dilakukan akses terhadap RemoteConfigRepository melalui *interface* yang dimilikinya untuk mendapatkan data tersebut dalam bentuk Json pada kode 3.1.

```
{
  "versionCode": "171",
  "maxVersion": "2.17.0"
}
```

Kode 3.1: Bentuk Data Json *version code* dan *max version* dari FirebaseRemoteConfig

Dalam RemoteConfigRepository, diambil sebuah *instance* dari Firebase Remote Config dan *instance* tersebut ditempatkan pada suatu variabel untuk mengaksesnya. Kemudian, sebuah adapter Moshi akan dibangun untuk mengakses data dalam bentuk Json. Adapter akan digunakan untuk melakukan parsing Json ke dalam bentuk model ForceUpdateDTO pada tabel 3.4.

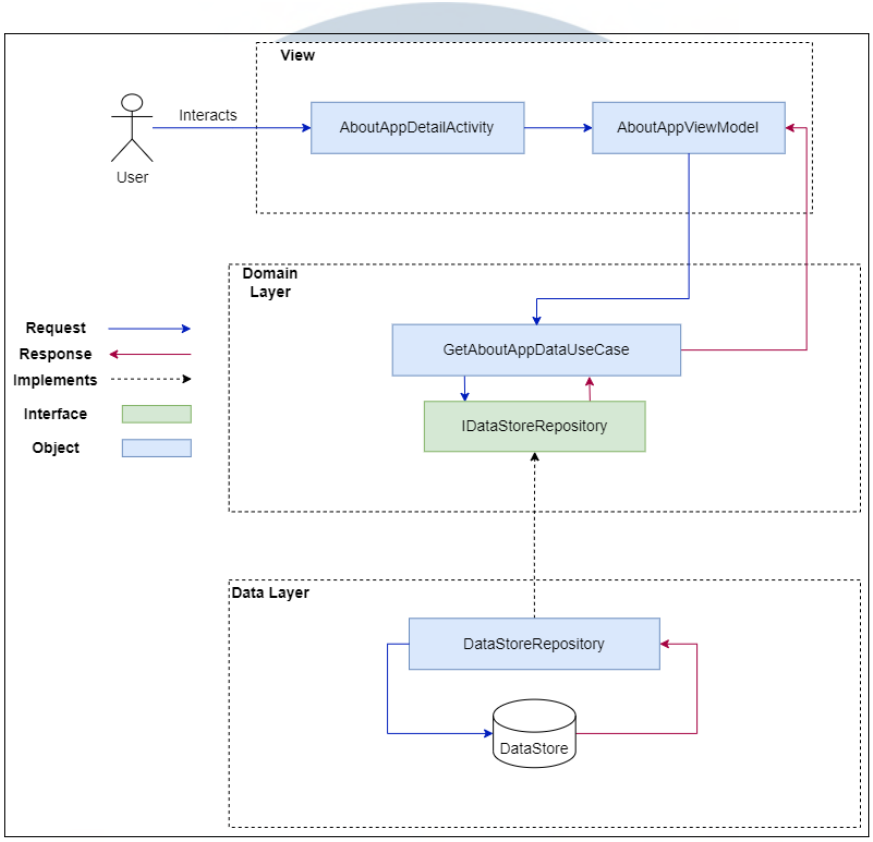
Tabel 3.4. *Model* ForceUpdateDTO

Variable Name	*Type*
versionCode	*String: Nullable*
appVersion	*String: Nullable*

Kemudian, data *max version* dan *version code* aplikasi akan diambil dari Firebase Remote Config menggunakan sebuah *key* dalam bentuk string. Remote Config juga akan diatur untuk memiliki *minimal fetch interval*, yaitu waktu minimal antar *fetch* data pada Firebase Remote Config sebesar 50 menit untuk menghindari melakukan *calling* secara berlebihan pada Firebase Remote Config.

Setelah itu, data yang telah dilakukan *parsing* oleh adapter dari bentuk JSON ke dalam model ForceUpdateDTO akan dikembalikan kepada *use case* yang telah memanggilnya, yaitu RetrieveForceUpdateUseCase.Terakhir, data *max version* dan *version code* akan dikirim kepada DataStoreRepository melalui *interface* yang dimilikinya untuk disimpan menggunakan *key*, *value pair* pada DataStore lokal.

A.2.2 Penampilan Data pada Halaman Tentang Aplikasi



Gambar 3.2. *Flow* Penampilan Data pada Halaman Tentang Aplikasi

Gambar 3.2 adalah *flow* penampilan halaman Tentang Aplikasi, dimana fitur *log status* berada. Saat pengguna membuka halaman Tentang Aplikasi, AboutAppViewModel akan dipanggil serta akan dilakukan observasi terhadap variabel *LiveData* yang akan menampung data yang diperlukan nanti. AboutAppViewModel akan kemudian melakukan *invoke* terhadap *use case* GetAboutAppDataUseCase untuk mengambil data yang perlu ditampilkan.

Pada *use case* ini, akan dilakukan pengambilan *max version*, *version code*, *email*, *GUID*, dan ID transaksi dengan mengakses DataStore lokal menggunakan DataStoreRepository melalui *interface* yang dimilikinya. Kemudian, data-data yang terkait dengan sistem perangkat genggam seperti *manufacturer*, model, versi OS perangkat, dan versi aplikasi akan diambil langsung dari sistem.

Data-data yang telah dikumpulkan sebelumnya kemudian dipetakan kepada model AboutAppEntities pada tabel 3.5 dan dikembalikan kepada AboutAppViewModel yang memanggil *use case*. Data yang telah diterima AboutAppViewModel lalu di *post* pada variabel *livedata*. AboutAppDetailActivity

kemudian melakukan *binding* terhadap data baru yang telah di observasi dan menampilkan informasi-informasi tersebut.

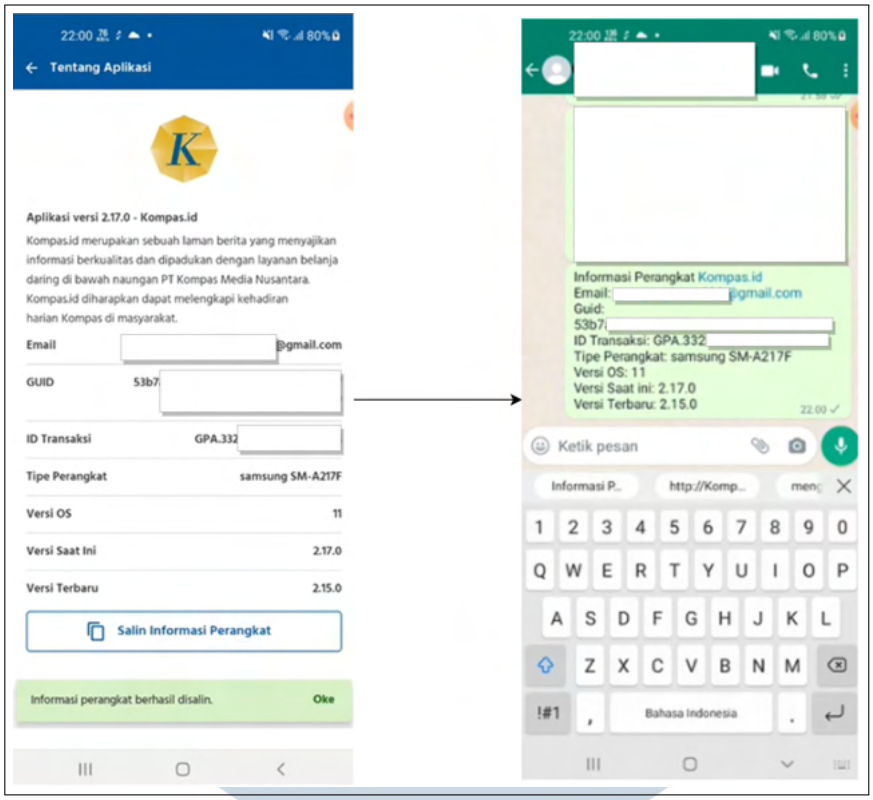
Tabel 3.5. Model AboutAppEntities

Variable Name	*Type*
versionCode	*String: Nullable*
maxVersion	*String: Nullable*
email	*String: Nullable*
guid	*String: Nullable*
manufacturer	*String: Nullable*
model	*String: Nullable*
osVersion	*String: Nullable*
currentVersion	*String: Nullable*
transactionId	*String: Nullable*

A.3 Implementasi

Gambar 3.3. Tampilan Fitur *Log Status*

Gambar 3.3 adalah implementasi dari fitur *log status* yang menampilkan informasi-informasi yang diperlukan oleh *hotline* saat pengguna melaporkan sebuah masalah. Tersedia juga sebuah tombol untuk menyalin informasi-informasi tersebut dengan menekannya.



Gambar 3.4. Tampilan Fitur *Log Status* Saat Menyalin Informasi Perangkat

Gambar 3.4 adalah *flow* dari *copy and paste* data-data yang tertera pada fitur *log status* kepada suatu aplikasi pengiriman pesan. Dapat dilihat juga bahwa terdapat notifikasi *toast* pada aplikasi yang menunjukkan bahwa data telah berhasil disalin.

B. *Membership Error*

B.1 *User Requirement*

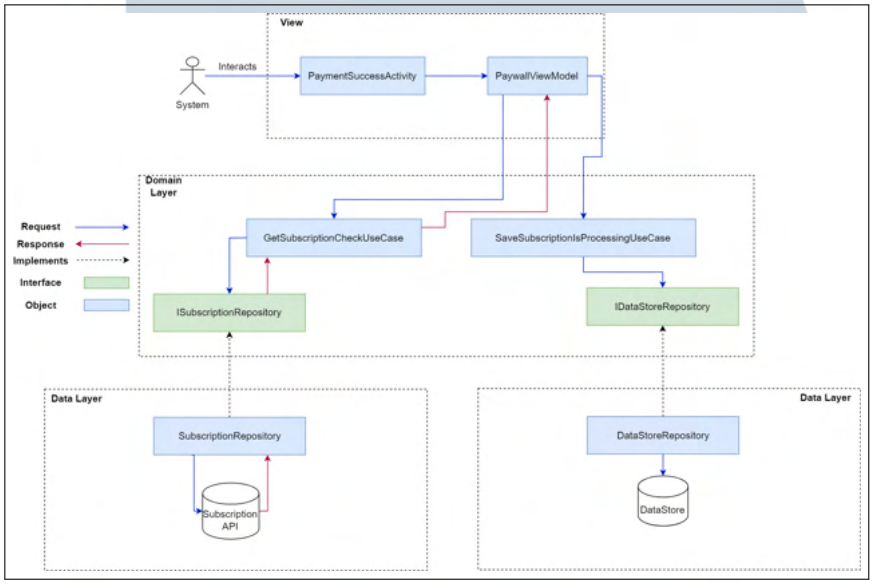
Saat pengguna Kompas.id telah berhasil berlangganan, terkadang terjadi situasi dimana *benefit* berlangganan tersebut tidak masuk secara langsung ke akun pengguna. *Benefit* ini meliputi berbagai fitur-fitur premium seperti akses tak terhingga terhadap artikel berbayar, membaca *e-paper* dan *e-book* dan *text-to-speech*. Pada akhirnya, beberapa pengguna memberikan rating buruk terhadap aplikasi Android Kompas.id sebagai akibat dari permasalahan tersebut.

Untuk mengatasi permasalahan tersebut, dikembangkanlah fitur *membership error* untuk memberikan informasi akurat terhadap status langganan dari akun pengguna. Fitur ini meliputi pemberitahuan dalam beberapa bentuk apabila terjadi

kendala dalam status langganan pengguna serta pemberitahuan apabila kendala tersebut telah diselesaikan dan pengguna dapat mengakses fitur-fitur.

B.2 Perancangan

B.2.1 Pengecekan Status *Membership* Saat Pembayaran Berhasil



Gambar 3.5. *Flow* Pengecekan Status *Membership* pada Sistem *Backend* Saat Pembayaran Berhasil

Gambar 3.5 adalah *flow* pengecekan ketika seorang pengguna berhasil melakukan pembayaran, maka sistem akan memasuki PaymentSuccessActivity. Pada *activity* tersebut, pertama akan dilakukan pemanggilan terhadap pada PaywallViewModel serta observasi pada variabel LiveData subscription. PaywallViewModel akan kemudian melakukan *invoke* pada use case GetSubscriptionCheckUseCase.

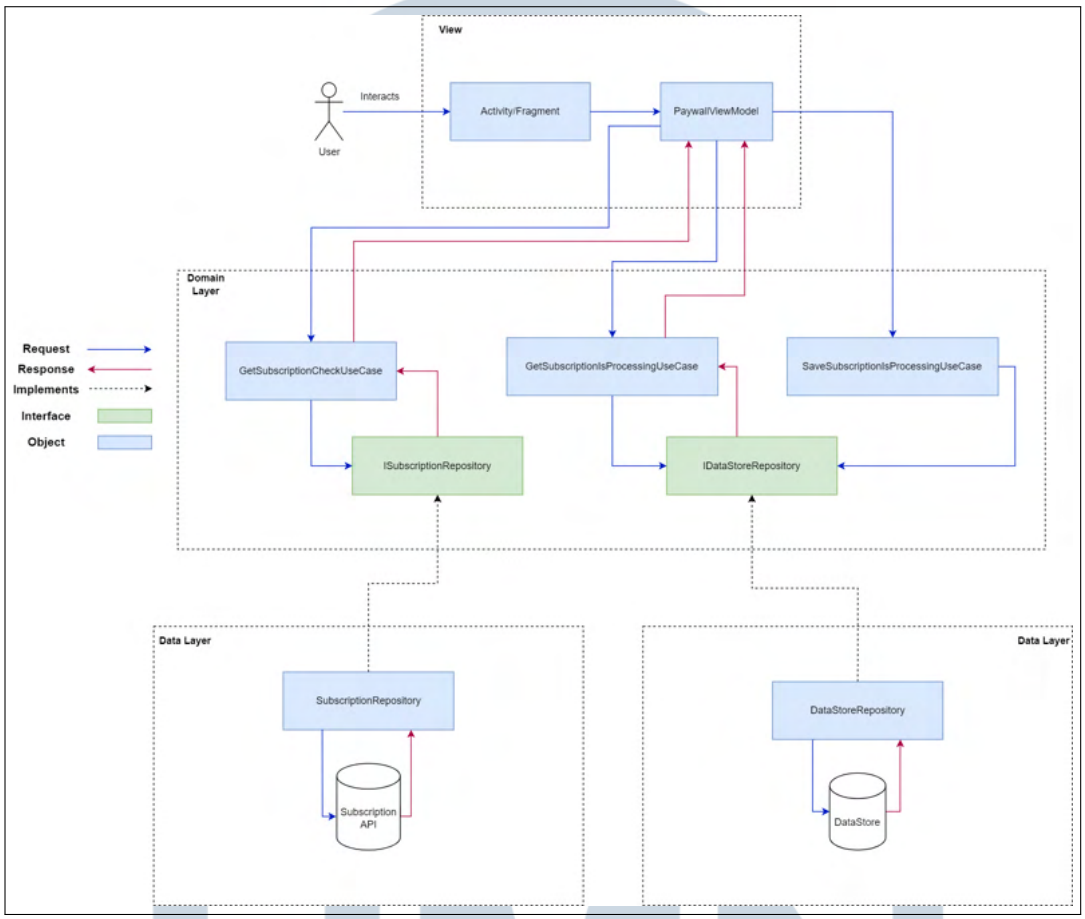
Use case akan melakukan request kepada API subscription Kompas.id menggunakan subscription repository melalui *interface* yang dimilikinya. Lalu, data yang diperoleh kemudian dikirim kembali ke PaywallViewModel dalam bentuk model SubscriptionCheckEntities pada tabel 3.6. Lalu PaywallViewModel akan melakukan post pada variabel subscription dan diterima oleh PaymentSuccessActivity yang telah melakukan observe.

Tabel 3.6. Model SubscriptionCheckEntities

Variable Name	*Type*
firstName	*String: Nullable*
lastName	*String: Nullable*
email	*String: Nullable*
isLogin	*String: Nullable*
isAccess	*String: Nullable*
startDate	*String: Nullable*
endDate	*String: Nullable*
membershipSlug	*String: Nullable*
membershipTitle	*String: Nullable*
guid	*String: Nullable*
subscriptionStatus	*Int*

Apabila subscription status dari pengguna sudah berupa 2 pada data API subscription Kompas.id, maka akan ditampilkan halaman berhasil berlangganan. Namun apabila belum, maka akan ditampilkan halaman langganan sedang diproses dan suatu *flag* berupa TRUE akan disimpan pada DataStore secara lokal melalui DataStoreRepository menggunakan *key*, *value pair* untuk menandakan bahwa langganan sedang diproses. Banner-banner pada berbagai halaman pun akan ditampilkan berdasarkan *flag* tersebut.

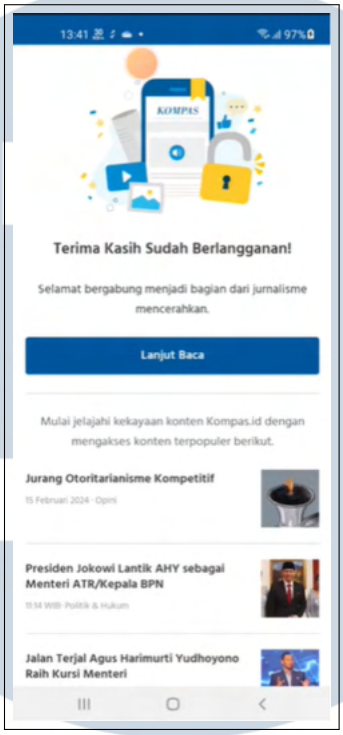
B.2.2 Pengecekan Status *Membership* pada Berbagai Activity dan Fragment



Gambar 3.6. *Flow* Pengecekan Status *Membership* pada Berbagai Activity dan Fragment

Ketika pengguna membuka beberapa activity atau fragment tertentu, maka akan dilakukan pengecekan kembali terhadap status berlangganan pada API *subscription* Kompas.id seperti pada gambar 3.6. Apabila status berlangganan pengguna masih belum 2 atau SUBER pada sistem *backend* dan *flag* berlangganan sedang diproses adalah TRUE pada *DataStore* lokal, maka *banner-banner* status berlangganan sedang diproses akan ditampilkan. Namun bila status berlangganan pengguna sudah menjadi 2 atau SUBER pada data dari API *subscription* Kompas.id, maka akan dilakukan perubahan *flag* tersebut menjadi FALSE pada DataStore secara lokal melalui interface yang dimiliki DataStoreRepository. Banner-banner yang sebelumnya menandakan bahwa status langganan pengguna sedang diproses pun akan dihilangkan.

B.3 Implementasi

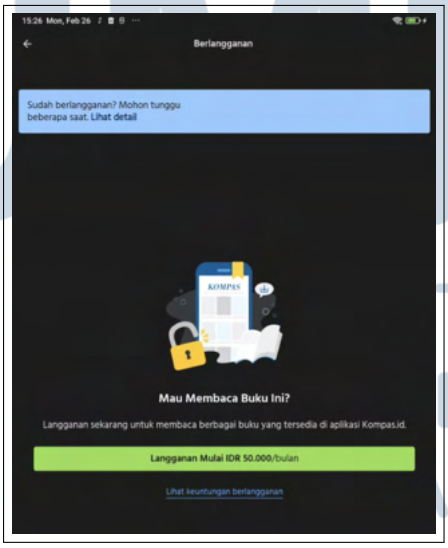


Gambar 3.7. *Payment Success Page*

Gambar 3.7 adalah *halaman ketika data subscription* pada sistem *backend* telah menandakan bahwa pengguna sudah berstatus SUBER saat pengguna berhasil melakukan pembayaran. Pengguna sudah dapat langsung mengakses fitur-fitur *premium* yang tersedia.

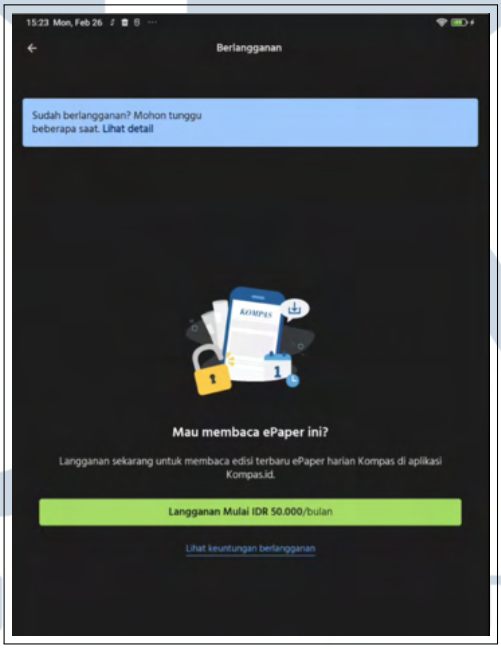
Gambar 3.8. *Payment Processing Page*

Gambar 3.8 adalah halaman ketika data *subscription* pada sistem *backend* belum menandakan bahwa pengguna sudah berstatus SUBER namun pengguna telah berhasil melakukan pembayaran. Pengguna belum dapat langsung mengakses fitur-fitur *premium* yang tersedia.



Gambar 3.9. *Paywall* Buku

Gambar 3.10. *Paywall* Tetopi

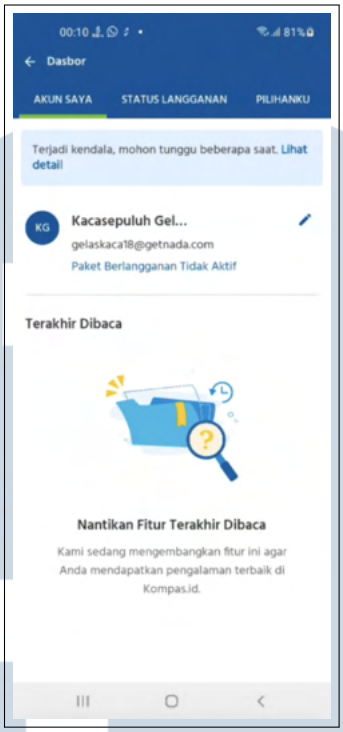


Gambar 3.11. *Paywall E-paper*

Gambar 3.9, 3.10, dan 3.11 adalah implementasi fitur *membership processing* pada *paywall*. Pengguna akan diarahkan ke halaman-halaman tersebut ketika mengakses fitur-fitur *premium* ketika belum memiliki status akun sebagai SUBER. *Banner* di atas halaman akan ditampilkan ketika status berlangganan akun pengguna sedang diproses.

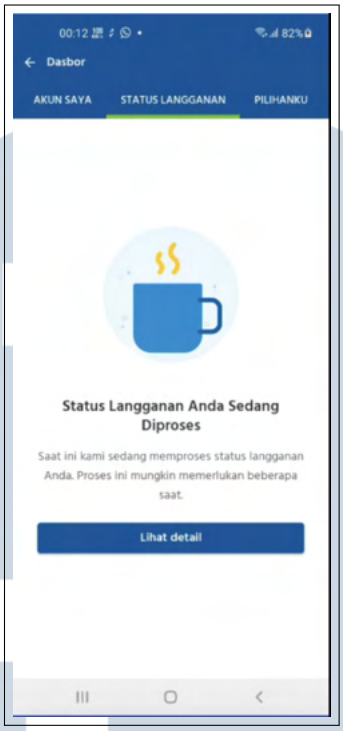
Gambar 3.12. Halaman Detail Artikel

Gambar 3.12 adalah halaman detail artikel yang terhalang oleh *metered paywall* atau MP ketika status langganan akun pengguna belum berupa SUBER. *Banner* di atas MP akan ditampilkan ketika status berlangganan akun sedang diproses.



Gambar 3.13. Halaman Akun Saya pada Dasbor

Gambar 3.13 merupakan halaman Akun Saya pada menu Dasbor di halaman Akun. *Banner* juga akan ditampilkan di atas halaman ketika status berlangganan akun pengguna sedang diproses. Apabila teks ”lihat detail” pada *banner-banner* di halaman lainnya pada gambar 3.9, 3.10, 3.11, dan 3.12 ditekan, maka akan diarahkan ke halaman ini.



Gambar 3.14. Halaman Status Langganan pada Dasbor

Gambar 3.15. *Bottom Sheet* Fitur *Membership Error*

Gambar 3.14 merupakan tampilan halaman Status Langganan ketika status langganan akun pengguna sedang diproses. Tampilan bottomsheet pada gambar

3.15 akan ditampilkan ketika pengguna menekan tulisan ”lihat detail” pada *banner* di gambar 3.13 ataupun tombol ”lihat detail” pada gambar 3.14.

C. Kompas.id Versi *Landscape*

C.1 Halaman E-Paper Saya

C.1.1 *User Requirement*

Halaman *E-Paper* Saya adalah halaman yang berisi *e-paper* yang telah diunduh oleh pengguna. Pengguna dapat mengakses halaman tersebut secara *offline* untuk membaca *e-paper* yang sudah diunduh. Tim produk meminta untuk pembuatan halaman ini dalam versi *landscape* untuk keperluan kolaborasi dengan pihak manufaktur RSE.

C.1.2 Perancangan

Untuk *flow* yang digunakan secara keseluruhan serupa dengan versi *portrait*, dengan perbedaan pengecekan jumlah *grid* menggunakan *grid layout manager*. Hal ini dilakukan karena perbedaan jumlah *e-paper* yang ditampilkan per baris, yaitu dua pada mode *portrait* dan empat pada mode *landscape*.

Selain itu dilakukan beberapa penyesuaian untuk ukuran tampilan mode *landscape*, seperti berbagai komponen yang ada serta ukuran *dialog box* untuk penghapusan *e-paper* yang telah diunduh untuk menjadi kotak selebar 355dp, berbeda dengan versi *portrait* yang mengambil lebar keseluruhan lebar layar.

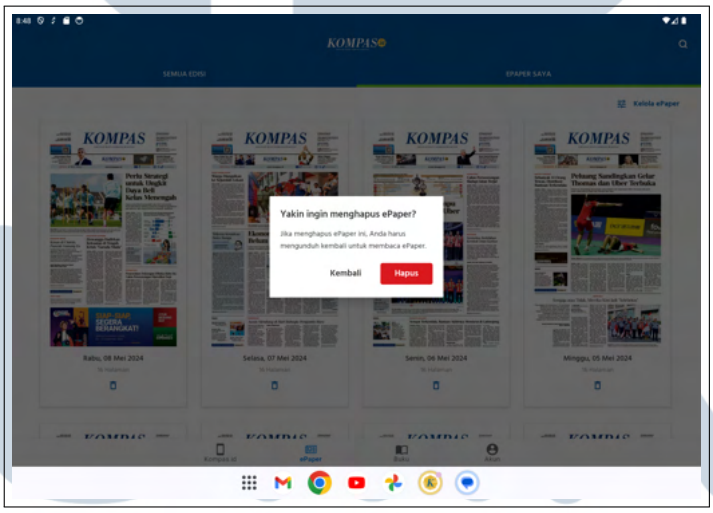
C.1.3 Implementasi

Gambar 3.16. Halaman E-paper Saya Versi *Portrait*

Gambar 3.17. Halaman E-paper Saya Versi *Landscape*

Gambar 3.17 adalah implementasi dari fitur Kompas.id versi *landscape*. Dapat dilihat perbedaan jumlah *grid* yang berupa dua pada *portrait* seperti pada gambar 3.16 dan empat pada versi *landscape* seperti pada gambar 3.17.

Gambar 3.18. *Dialog Box* Hapus E-Paper Versi *Portrait*



Gambar 3.19. *Dialog Box* Hapus E-Paper Versi *Landscape*

Gambar 3.19 adalah implementasi perubahan *dialog box* pada halaman E-Paper saya versi *landscape*. Lebar *dialog box* ini jauh lebih kecil dibandingkan dengan versi *portrait* pada gambar 3.18 yang mengambil lebar keseluruhan layar.

C.2 Widget pada *Mainscreen*

C.2.1 *User Requirement*

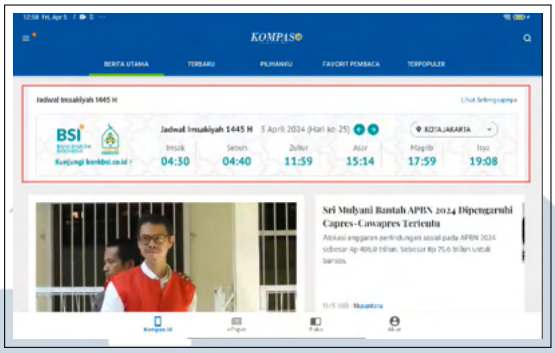
Pada halaman utama atau *mainscreen*, terdapat suatu *widget* Iframe yang menampung konten dari *webview*. Widget ini berisi konten dari sebuah sponsor yang ingin berpromosi pada platform Kompas.id ataupun inisiatif dari Kompas sendiri seperti *banner quick count* pada saat PILPRES 2024 lalu. Tim produk meminta untuk pembuatan *widget* ini dalam versi *landscape* untuk keperluan kolaborasi dengan pihak manufaktur RSE.

C.2.2 Perancangan

Perbedaan antara *widget* pada mode *portrait* dan *landscape* hanyalah penempatannya. Pada mode *portrait*, *widget* diletakkan setelah artikel pertama, sedangkan pada mode *landscape* diletakkan pada bagian paling atas.

C.2.3 Implementasi

Gambar 3.20. *Widget* iFrame pada Mode *Portrait*



Gambar 3.21. *Widget* iFrame pada Mode *Landscape*

Gambar 3.21 adalah implementasi *widget* iframe pada halaman Berita Utama. Dapat diperhatikan perbedaannya dengan versi *portrait* terletak pada posisinya yang berada pada paling atas halaman, sedangkan pada mode *portrait* seperti pada gambar 3.20 terletak di bawah artikel pertama.

3.3.2 *Improvement*

A. *Shimmer Loading* pada Halaman Akun Saya

A.1 Permasalahan

Gambar 3.22. Tampilan Data Muncul *Default Text* Ketika *Data* Sedang *Loading*

Sebelum *improvement* ini diterapkan, pada halaman Akun Saya saat data pengguna sedang dimuat, muncul *default text* seperti pada gambar 3.22 sebelum data pengguna ditampilkan. Hal ini memiliki potensi untuk membingungkan pengguna karena ketidaksesuaian data dengan yang seharusnya ditampilkan, terutama penulisan ”Paket Berlangganan Aktif”.

A.2 Implementasi

Untuk mengatasi masalah ini, dirancanglah sebuah desain *shimmer* seperti pada gambar 3.23. *Shimmer* ini akan ditampilkan ketika data pengguna sedang dalam *state loading*.

Gambar 3.23. *Shimmer* Pada Halaman Akun Saya Ketika *Data* Sedang *Loading*

B. *Refresh* Detail Akun Pada Halaman Akun Saya

B.1 Permasalahan

Sebelum *improvement* ini diterapkan, data pada halaman Akun Saya hanya dapat di *refresh* apabila pengguna keluar dari menu Dasbor dan masuk kembali. Hal ini dapat menyebabkan pengguna untuk tidak dapat mengetahui kemungkinan adanya pembaharuan terhadap data akun pengguna, seperti status langganan, e-mail, dan nama pengguna.

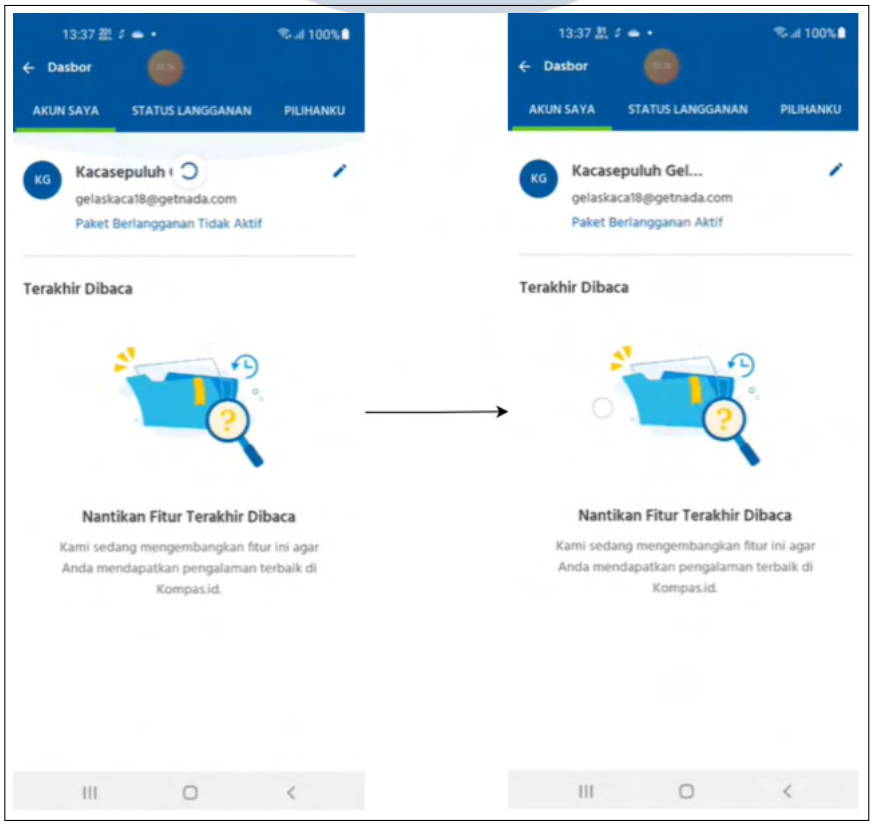
Untuk mengatasi permasalahan tersebut, dibuatlah sebuah sistem *swipe-to-refresh* pada halaman akun saya agar pengguna dapat melakukan refresh dan mendapatkan data terbaru terkait dengan informasi-informasi di atas.

B.2 Perancangan

Gambar 3.24. *Flowchart Swipe-To-Refresh* pada Halaman Akun Saya

Seperti pada gambar 3.24, ketika pengguna melakukan *swipe-to-refresh* maka data pada adapter recyclerview akan dihapus dan akan ditukar oleh data baru yang diperoleh dari API *subscription*. Tampilan akhir akan menunjukkan data pengguna terbaru setelah proses selesai.

B.3 Implementasi



Gambar 3.25. Implementasi *Improvement Swipe-To-Refresh* pada Halaman Akun Saya

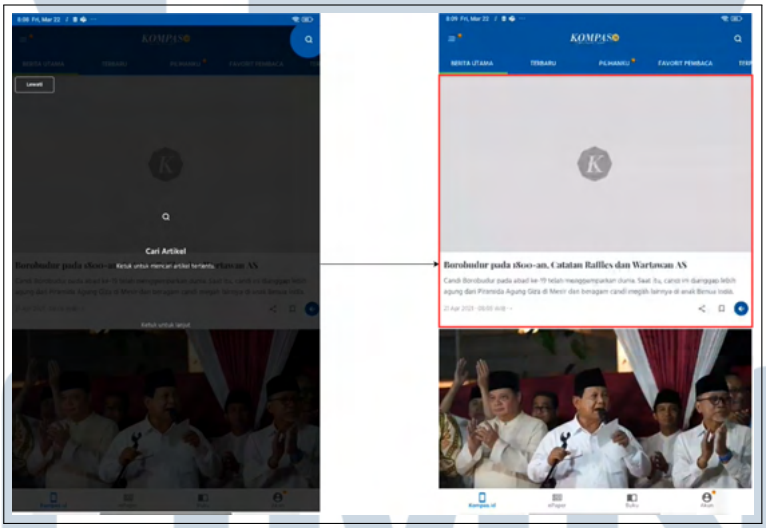
Gambar 3.25 menunjukkan situasi dimana seorang pengguna telah mendapatkan status langganan aktif atau SUBER, tetapi data pada halaman akun saya belum terbaharui.

Dengan melakukan *swipe-to-refresh*, maka pengguna dapat melihat status langganan yang telah diperbaharui pada halaman tersebut tanpa harus meninggalkannya.

3.3.3 *Bug Fixing*

A. *Placeholder Coachmark* Tersangkut pada Halaman Utama

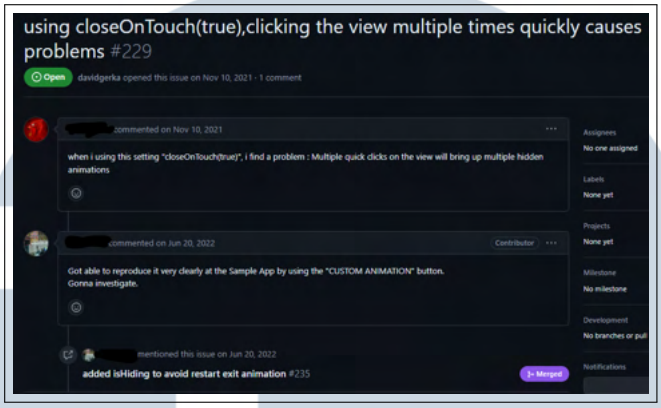
A.1 *User Requirement*



Gambar 3.26. *Dummy Article Coachmark* Tersangkut pada Halaman Utama

Pada saat pengguna pertama kali masuk pada aplikasi Kompas.id pada perangkatnya (*fresh-install*), maka akan ditampilkan *coachmark* atau *overlay* untuk menjelaskan beberapa fitur yang tersedia pada halaman utama. Namun, apabila pengguna menekan *coachmark* tersebut untuk melewati prosesnya dengan sangat cepat, maka akan muncul kemungkinan untuk *dummy article* yang digunakan untuk tersangkut pada tampilan halaman utama seperti pada gambar 3.26.

A.2 Solusi



Gambar 3.27. Pengguna Lain *Library* yang Kemungkinan Mengalami Kasus Serupa

Akar permasalahan ini sulit untuk ditemukan karena fitur *coachmark* diimplementasikan menggunakan *library* eksternal. Pada halaman Github, terdapat pengguna lain pada gambar 3.27 yang sepertinya memiliki permasalahan yang sama dan telah menemukan solusi. Namun, solusi tersebut sepertinya belum dirilis pada platform Jitpack.io, yaitu *platform* untuk merilis *library* dari suatu *repository* Github.

Maka, solusi yang dilakukan adalah menyalin seluruh kode dari *branch master repository* Github *coachmark* tersebut kepada *codebase* proyek Android Kompas.id. Setelah hal tersebut dilakukan, kasus tersebut belum pernah terjadi kembali.

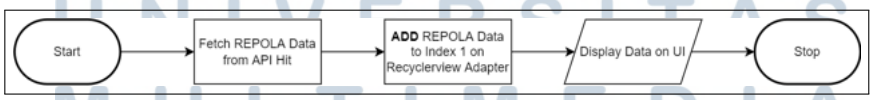
B. Tampilan REPOLA Muncul Berulang

B.1 *User Requirement*

Gambar 3.28. Tampilan REPOLA Berulang

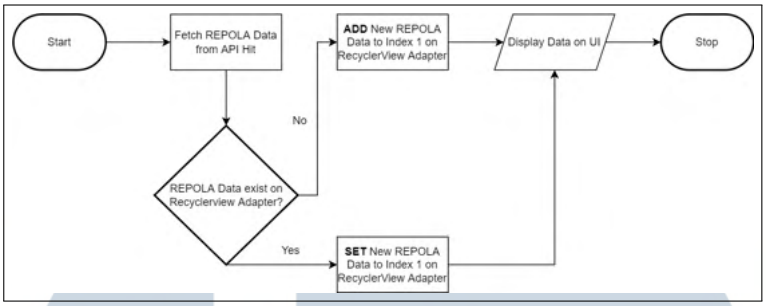
Terdapat suatu kasus *bug* pada fitur REPOLA, tepatnya pada tampilan REPOLA di *tab* Berita Utama pada halaman utama saat user melakukan tindakan mengunci perangkat genggam dan menyalakannya kembali. Apabila hal ini dilakukan, maka jumlah tampilan REPOLA pada halaman Berita Utama akan bertambah secara terus menerus seperti pada gambar 3.28 dimana seharusnya hanya ditampilkan satu buah.

B.2 Solusi



Gambar 3.29. *Flowchart* Fitur REPOLA Sebelum Perbaikan

Permasalahan ini terjadi karena pada saat *bug* tersebut ditemukan, belum ada pengecekan terhadap apakah sudah ada REPOLA pada halaman Berita Utama pada saat ditambahkan setiap kali API REPOLA dipanggil pada *lifecycle* onResume seperti pada gambar 3.29. Hal ini menyebabkan REPOLA untuk terus menerus ditambahkan pada adapter recyclerview sehingga menjadi lebih dari satu.



Gambar 3.30. *Flowchart* Fitur REPOLA Setelah Perbaikan

Untuk mengatasi hal tersebut, ditambahkanlah suatu pengecekan apakah sudah ada REPOLA sebelumnya pada recyclerview seperti pada gambar 3.30. Apabila belum, maka akan ditambahkan pada *index* pertama recyclerview halaman Berita Utama. Apabila sudah, maka akan dilakukan *set* atau indeks pertama yang sudah mengandung REPOLA yang lama akan ditukar dengan *viewholder* REPOLA yang baru.

Dengan melakukan perbaikan ini, maka hanya akan ada satu buah REPOLA yang ditampilkan seperti pada gambar 3.31. Di bawah REPOLA akan terdapat elemen-elemen lain, seperti artikel.

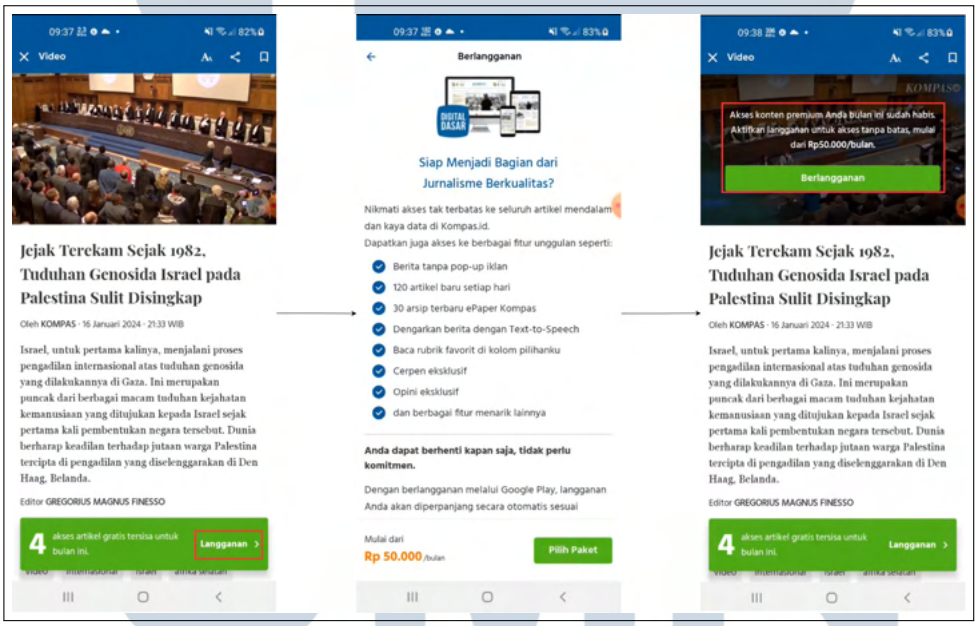
Gambar 3.31. Tampilan Fitur REPOLA Setelah Diperbaiki

C. Halaman Artikel Video

C.1 Video Pengguna REGON Terblokir *Overlay Paywall* Saat Masih Mempunyai Kuota

C.1.1 *User Requirement*

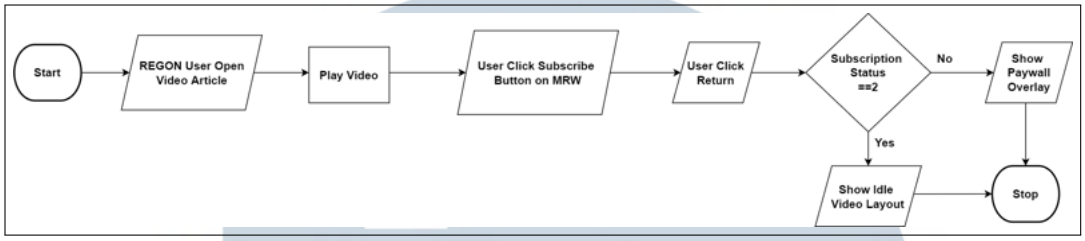
Seperti pada gambar 3.32, pada saat pengguna akun REGON yang masih punya kuota untuk membuka detail artikel video dan memainkan video lalu menekan tombol berlangganan pada MRW atau *Metered Register Wall*, maka pengguna akan diarahkan ke halaman berlangganan. Namun ketika menekan tombol kembali dari halaman berlangganan tersebut, ditampilkan *overlay paywall* pada video.



Gambar 3.32. Tampilan *Flow* Permasalahan Fitur Video Bagi Pengguna REGON Sebelum Diperbaiki

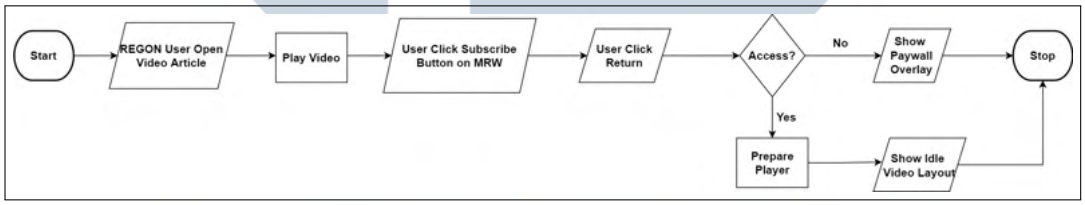
Pengguna hanya dapat kembali menonton video dari awal ketika keluar dari halaman detail artikel video tersebut dan kembali. Seharusnya, pengguna tetap dapat memainkan video setelah kembali dari halaman berlangganan karena pengguna telah menggunakan kuota yang dimilikinya untuk membuka detail artikel video tersebut.

C.1.2 Solusi

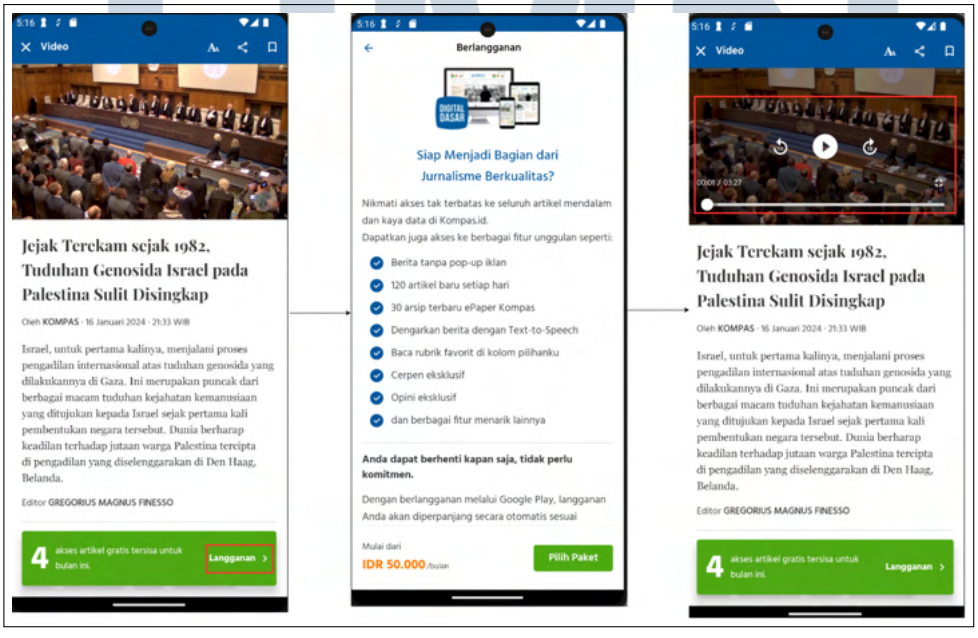


Gambar 3.33. *Flowchart State Idle* Video Saat Kembali dari Halaman Berlangganan Sebelum Perbaikan

Permasalahan ini disebabkan oleh pengecekan kondisi untuk memunculkan *overlay paywall* yang kurang tepat. Hal ini dikarenakan pengecekan dilakukan pada status berlangganan, bukan adanya akses atau tidak yang berasal dari data sistem *backend*. Karena pengguna masih merupakan REGON, maka paywall dimunculkan biarpun pengguna memiliki akses.



Gambar 3.34. *Flowchart State Idle* Video Saat Kembali dari Halaman Berlangganan Setelah Perbaikan



Gambar 3.35. *Flowchart* Fitur Video Bagi Pengguna REGON Setelah Diperbaiki

Untuk mengatasi permasalahan tersebut, maka pengecekan diubah mengguna parameter access dari sistem *backend* seperti pada gambar 3.34. Maka, biarpun seorang pengguna memiliki akun REGON, ketika kembali dari halaman berlangganan, maka akan diarahkan menuju *layout video idle* bukan *overlay paywall*.

Selain itu, ditambahkan juga proses *prepare player* untuk mengatasi masalah video tidak dapat dimainkan. Hal ini dikarenakan saat kembali dari halaman berlangganan, *player* berada pada *state idle*. *State* ini berarti *player* belum siap memainkan media apapun dan belum menerima data media yang perlu dimainkan, sehingga perlu disiapkan dengan metode prepare(). Tampilan setelah perbaikan dapat dilihat pada gambar 3.35.

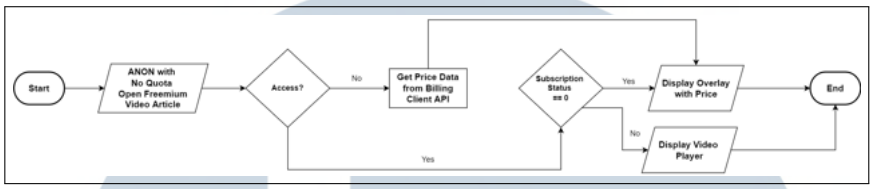
C.2 *Overlay Paywall* Video dengan Harga *Null* bagi Pengguna ANON yang Tidak Memiliki Kuota

C.2.1 *User Requirement*

Gambar 3.36. Tampilan *Overlay Paywall* dengan Harga *Null* bagi Pengguna ANON yang Tidak Memiliki Kuota

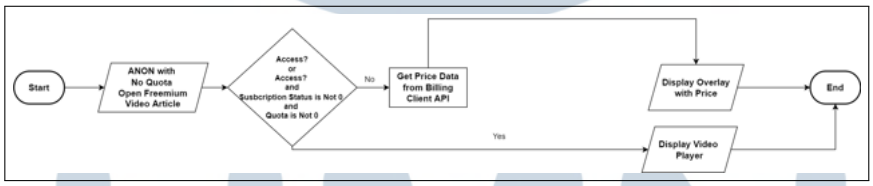
Saat pengguna ANON yang tidak memiliki kuota mengakses artikel video bebas akses, maka seharusnya dimunculkan *overlay paywall* yang mencantumkan harga. Namun, yang dimunculkan adalah *overlay paywall* tanpa harga seperti pada gambar 3.36.

C.2.2 Solusi



Gambar 3.37. *Flowchart Overlay Paywall* dengan Harga Null bagi Pengguna ANON yang Tidak Memiliki Kuota Sebelum Perbaikan

Setelah ditelusuri, *bugs* ini terjadi karena harga diambil melalui API *billing client* dan belum ada kondisi untuk memunculkan harga untuk artikel video bebas akses seperti pada gambar 3.37. Kondisi pengambilan data ini merupakan parameter ”access” dari sistem *backend*. Karena artikel yang dibuka berjenis bebas akses, maka parameter ini berupa TRUE. Namun, overlay ditampilkan pada pengguna ANON yang tidak memiliki kuota dan seharusnya tidak memiliki akses. Hal ini menyebabkan harga untuk tidak diambil dari API tersebut, sehingga tidak ada data harga untuk ditampilkan.



Gambar 3.38. *Flowchart Overlay Paywall* dengan Harga Null bagi Pengguna ANON yang Tidak Memiliki Kuota Setelah Perbaikan

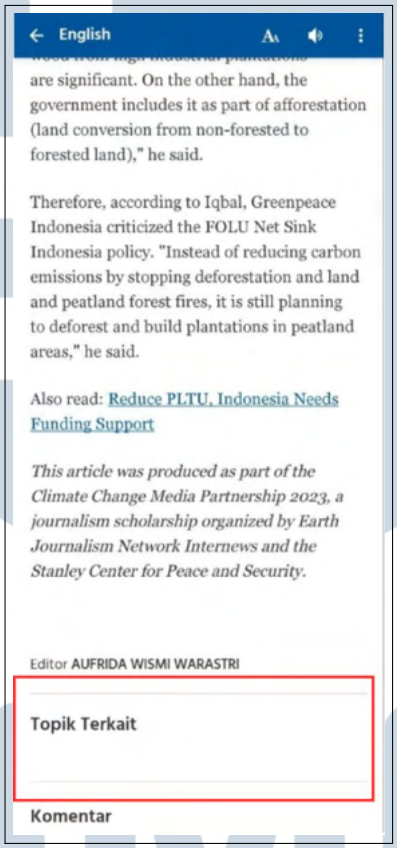
Gambar 3.39. Tampilan *Overlay Paywall* dengan Harga Null bagi Pengguna ANON dengan Kuota Habis Setelah Perbaikan

Untuk mengatasi masalah tersebut, maka dilakukan pengambilan data harga dari *billing client* untuk pengguna ANON yang sudah kehabisan kuota pada artikel video bebas akses dengan menambahkan kondisi seperti pada gambar 3.38. Maka, harga akan kembali muncul seperti pada gambar 3.39.

D. Halaman Detail Artikel

D.1 Bagian Topik Terkait Masih Muncul Walaupun Kosong

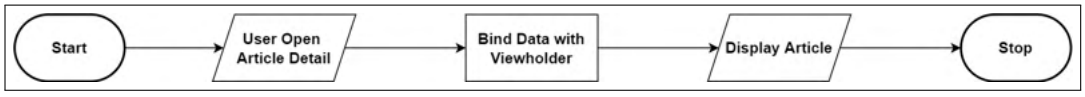
D.1.1 *User Requirement*



Gambar 3.40. Tampilan Topik Terkait Tetap Muncul Walaupun Kosong

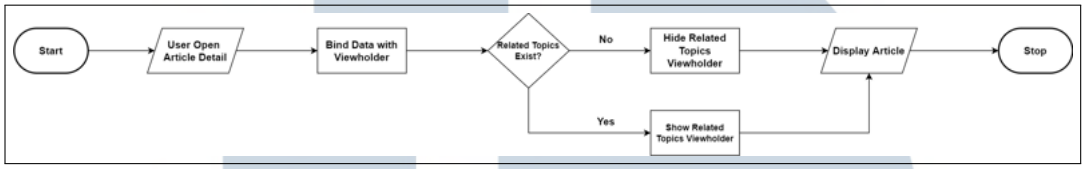
Pada beberapa detail artikel yang tersedia pada Kompas.id, tidak terdapat topik terkait yang tersedia untuk ditampilkan dari data yang diberikan sistem *backend*. Namun, bagian topik terkait masih ditampilkan biarpun kosong seperti pada gambar 3.40.

D.1.2 Solusi

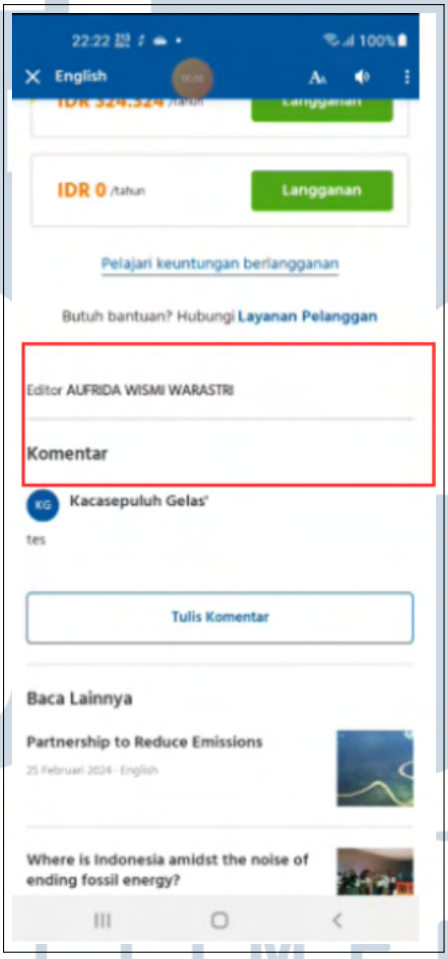


Gambar 3.41. *Flowchart* Penampilan Topik Terkait Sebelum Diperbaiki

Permasalahan ini disebabkan oleh tidak adanya pengecekan kondisi apabila terdapat data topik terkait dari sistem *backend* seperti pada gambar 3.41. Untuk memperbaikinya, ditambahkan sebuah pengecekan untuk menyembunyikan bagian topik terkait apabila kosong seperti pada gambar 3.42.



Gambar 3.42. *Flowchart* Penampilan Topik Terkait Setelah Diperbaiki



Gambar 3.43. Perbaikan Tampilan Topik Terkait yang Kosong

Setelah perbaikan, apabila tidak terdapat data topik terkait dari sistem *backend*, maka bagian tersebut akan disembunyikan seperti pada gambar 3.43.

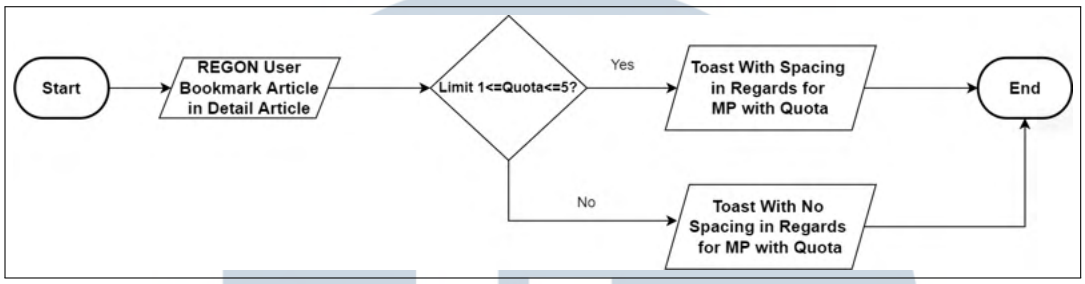
D.2 Toast *Bookmark* Menghalangi *Metered Paywall*/MP

D.2.1 *User Requirement*

Gambar 3.44. Toast Menghalangi MP

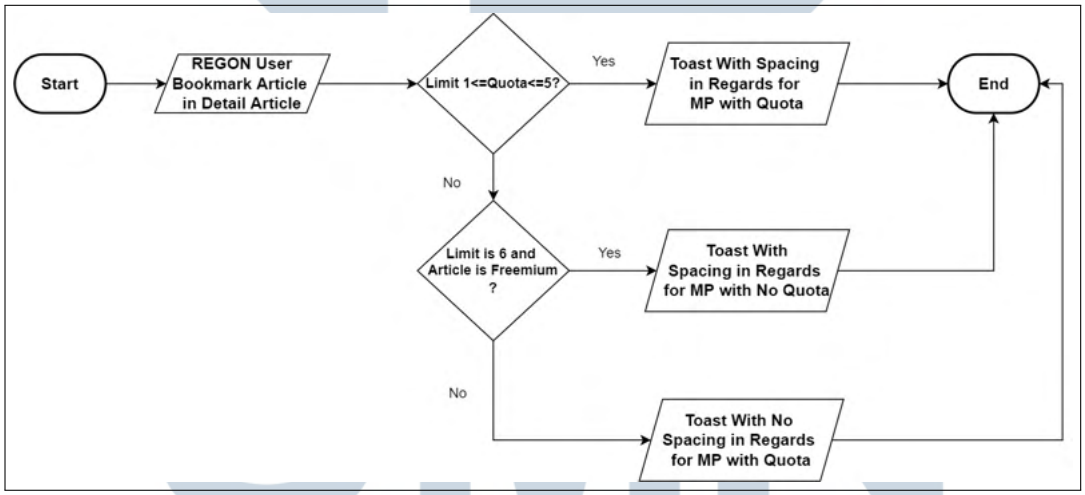
Pada saat pengguna REGON yang sudah tidak memiliki kuota membuka artikel bebas akses dan menyimpan artikel tersebut untuk dibaca nanti, maka *toast* yang mengkonfirmasi bahwa penyimpanan berhasil dilakukan menghalangi MP seperti pada gambar 3.44. Seharusnya, *toast* tersebut akan terletak di atas MP.

D.2.2 Solusi



Gambar 3.45. *Flowchart* Proses Penampilan Spacing Toast Bagi Akun REGON Tanpa Kuota pada Artikel Bebas Akses Sebelum Perbaikan

Setelah ditelusuri, pengecekan untuk *spacing toast* dengan bagian bawah layar sebelum perbaikan hanya dilakukan terhadap pengguna REGON yang masih memiliki kuota seperti pada gambar 3.45, sehingga *flow* pengguna REGON yang sudah tidak memiliki kuota dan membuka artikel bebas akses belum ditangani.



Gambar 3.46. *Flowchart* Proses Penampilan Spacing Toast Bagi Akun REGON Tanpa Kuota pada Artikel Bebas Akses Setelah Perbaikan

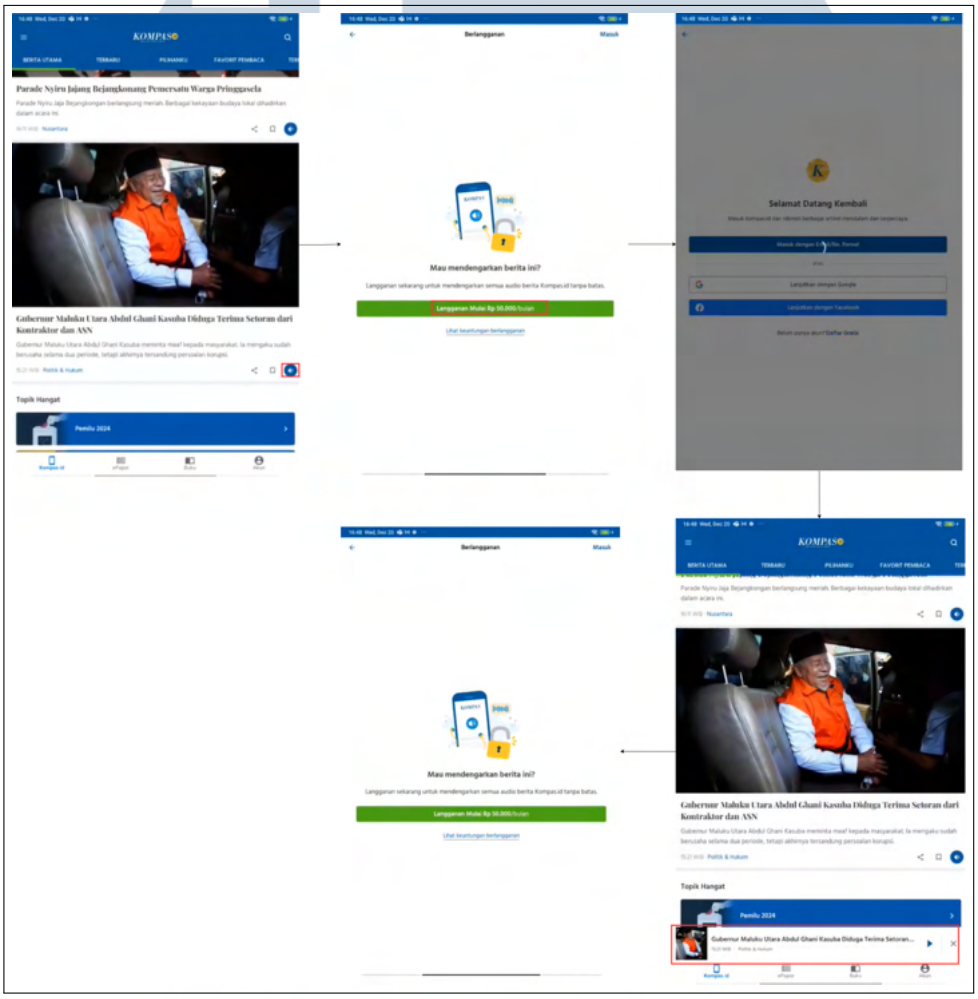
Gambar 3.47. Tampilan Spacing Toast dengan Kuota REGON pada Artikel Bebas Akses Setelah Perbaikan

Untuk mengatasi permasalahan tersebut, maka ditambah pengecekan untuk *spacing toast* dengan bagian bawah layar untuk *flow* bagi pengguna REGON yang sudah kehabisan kuota seperti pada gambar 3.46. Hasil perbaikan ini adalah *toast* baca nanti tidak akan menutupi MP untuk pengguna REGON dengan kuota yang sudah habis seperti pada gambar 3.47.

E. Fitur TETOPI

E.1 TETOPI Bar Muncul Sesaat Saat *Login* Akun REGON dari *Paywall*

E.1.1 *User Requirement*

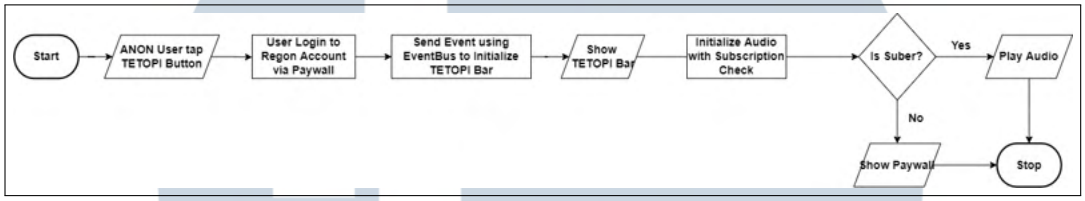


Gambar 3.48. *Flow* Permasalahan TETOPI Bar Muncul Sesaat Sebelum Perbaikan

Pengguna yang memiliki akun dengan status langganan REGON tidak memiliki akses terhadap fitur TETOPI. Namun, terdapat kondisi dimana user ANON yang menekan tombol TETOPI dan masuk halaman *paywall*. Lalu pengguna melakukan login melalui halaman *paywall* dengan akun REGON. Apabila kondisi ini dipenuhi, maka bar TETOPI akan muncul sesaat sebelum pengguna masuk ke halaman *paywall* kembali seperti pada gambar 3.49, dimana seharusnya pengguna langsung menuju halaman *paywall* tanpa terlihat bar TETOPI.

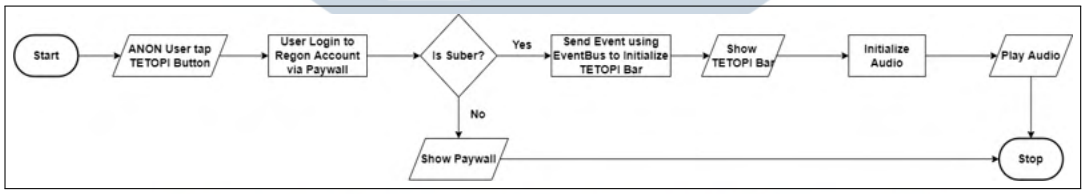
Hal ini dapat membingungkan para pengguna yang dapat merasa bahwa akun yang digunakan memiliki akses terhadap fitur TETOPI dengan akun REGON walaupun sebenarnya tidak.

E.1.2 Solusi

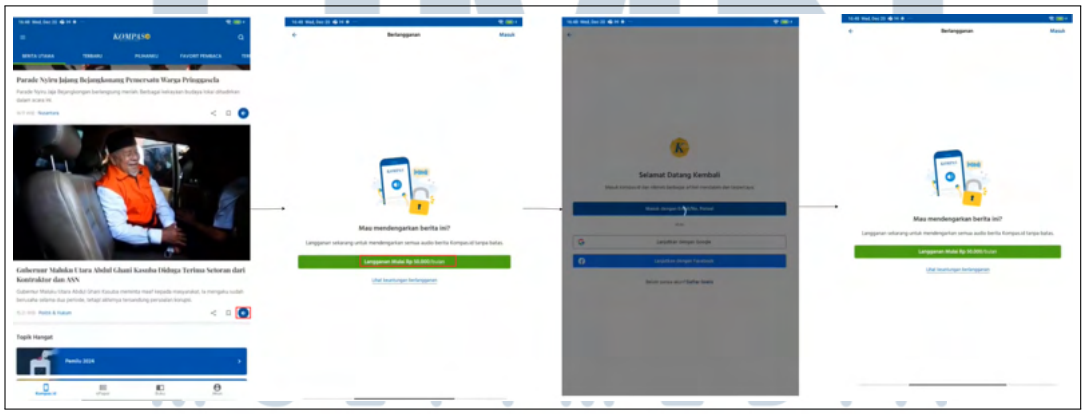


Gambar 3.49. *Flowchart* Proses Penampilan TETOPI Bar Sebelum Perbaikan

Permasalahan ini terjadi karena *flow* sebelumnya menggunakan EventBus untuk melakukan inisialisasi bar TETOPI sebelum melakukan inisialisasi *audio* seperti pada gambar 3.49 dimana pengecekan status langganan dilakukan dan *paywall* baru kembali ditampilkan.



Gambar 3.50. Tampilan *Flowchart* Proses Penampilan TETOPI Bar Setelah Perbaikan

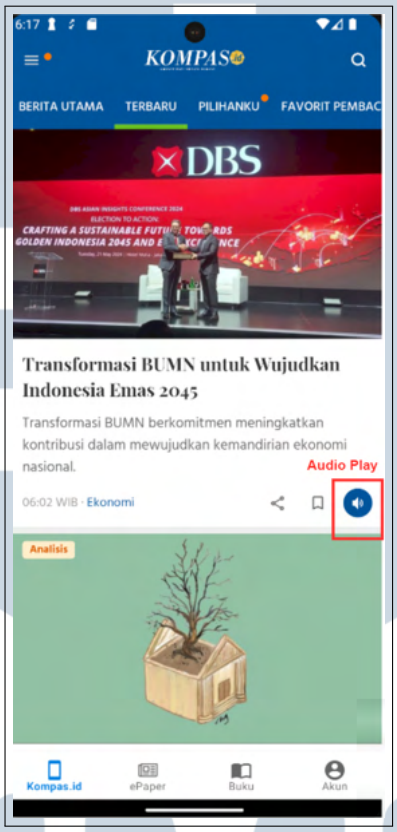


Gambar 3.51. *Flowchart* Perbaikan TETOPI Bar Muncul Sesaat

Untuk mengatasi masalah ini, pengecekan status langganan dilakukan sebelum menggunakan EventBus untuk melakukan inisialisasi bar TETOPI seperti pada gambar 3.50. Perbaikan ini menghasilkan TETOPI bar yang tidak muncul sesaat pada *flow* ini sebelum memasuki *paywall* kembali seperti pada gambar 3.51.

E.2 TETOPI Bar Tidak Muncul pada Halaman *Tab* Terbaru

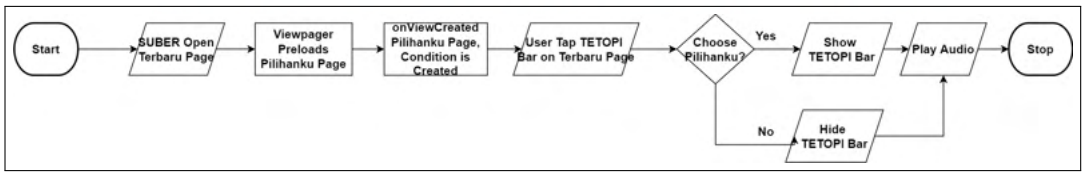
E.2.1 *User Requirement*



Gambar 3.52. Tampilan TETOPI Bar Tidak Muncul Pada Halaman Terbaru

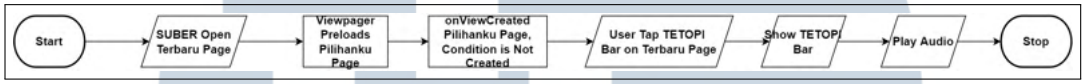
Pada halaman utama, terdapat kasus dimana apabila pengguna SUBER belum menentukan rubrik pilihanku, maka jika tombol TETOPI ditekan pada *tab* terbaru, maka tampilan akan seperti pada gambar 3.52. Suara akan berjalan, namun TETOPI bar tidak akan muncul sampai pengguna masuk ke *tab* pilihanku dan kembali ke terbaru.

E.2.2 Solusi

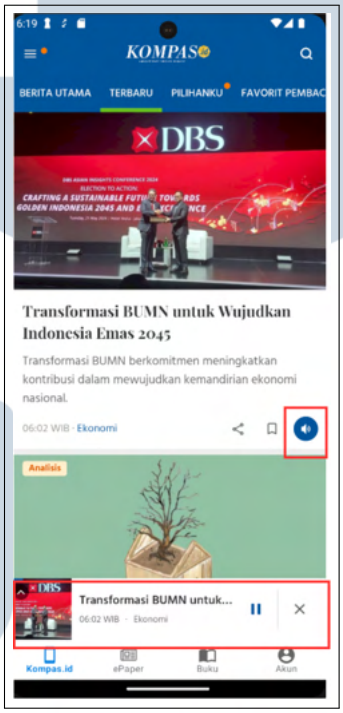


Gambar 3.53. *Flowchart* Menampilkan TETOPI Bar Pada Halaman Terbaru Sebelum Perbaikan

Permasalahan ini disebabkan oleh perilaku ViewPager yang melakukan *preload* halaman sebelum halaman tersebut dibuka. Apabila pengguna belum memilih rubrik pilihanku, maka pada halaman tersebut bar TETOPI akan disembunyikan. Karena halaman pilihanku berada di sebelah halaman terbaru dan kondisi penyembunyian bar TETOPI diletakkan pada *fragment lifecycle* onViewCreated, maka bar TETOPI disembunyikan sistem pada halaman terbaru seperti pada gambar 3.53.



Gambar 3.54. *Flowchart* Menampilkan TETOPI Bar Pada Halaman Terbaru Setelah Perbaikan



Gambar 3.55. Tampilan Perbaikan *Flow* TETOPI Bar Tidak Muncul Pada Halaman Terbaru

Untuk mengatasi permasalahan tersebut, kondisi pengecekan untuk penyembunyian bar TETOPI dipindahkan ke *fragment lifecycle* onResume seperti pada gambar 3.54. Perubahan ini menyebabkan kondisi tersebut untuk tidak dijalankan sampai halaman pilihanku itu sendiri terbuka, biarpun viewpager telah melakukan *preload*. Maka, walaupun pengguna belum memilih rubrik pilihanku dan menekan tombol TETOPI pada *tab* terbaru, TETOPI BAR akan muncul seperti pada gambar 3.55.

3.4 Kendala dan Solusi yang Ditemukan

3.4.1 Kendala yang Dihadapi

Pada saat melaksanakan praktik kerja magang, terdapat beberapa kendala dihadapi yaitu:

1. Belum memahami secara dalam mengenai *coding best practices* seperti penggunaan *clean architecture* dan MVVM serta *guideline* dan *code style* yang digunakan pada *codebase* Kompas.id versi Android.
2. Terkadang mengalami kesulitan memahami *cards* yang ditugaskan, baik merupakan *user story* ataupun *bugs*. Hal ini dikarenakan terkadang terdapat deskripsi *cards* yang masih bersifat ambigu atau membingungkan.
3. Terkadang masih mengalami kebingungan tentang *flow* dan cara kerja aplikasi secara *code*, terutama saat *cards* yang dikerjakan menyentuh fitur yang belum pernah dikerjakan sebelummya.

3.4.2 Solusi Atas Kendala yang Dihadapi

Untuk mengatasi kendala-kendala yang telah dipaparkan sebelumnya maka dilakukan tindakan sebagai berikut.

1. Untuk mengatasi kendala untuk pemahaman yang kurang atas *coding best practices* dan *code style* yang digunakan, maka dilakukan pembelajaran materi-materi tersebut yang tersedia pada internet serta membaca dokumentasi tim yang tersedia mengenai *code style* yang digunakan.
2. Untuk meluruskan deskripsi *cards* yang bersifat ambigu atau membingungkan, maka dilakukan diskusi dengan pihak yang terkait atas *cards* tersebut baik dari pihak QA, produk, ataupun rekan-rekan *developer*.
3. Untuk mengatasi kebingungan atas *flow* dari *code* suatu fitur di aplikasi, maka dilakukanlah komunikasi dengan rekan-rekan *developer* untuk memahaminya.