# BAB 3
# PELAKSANAAN KERJA MAGANG

Dalam bagian ini, akan dibahas aspek-aspek terkait pelaksanaan magang, seperti posisi dan struktur organisasi, tugas yang dilakukan, rincian pelaksanaan magang secara komprehensif, serta masalah yang muncul beserta solusinya. Bagian mengenai kedudukan dan organisasi akan memaparkan posisi dalam perusahaan serta cara berkoordinasi dengan atasan. Selanjutnya, pada bagian tugas yang dilakukan, akan berfokus pada aktivitas yang dijalankan selama masa magang. Setelah itu, rincian pelaksanaan magang akan lebih terfokus pada pengembangan fitur pusat bantuan, terutama pada fitur pengajuan dan pembuatan tiket laporan masalah, fitur melihat *detail* dari tiket masalah yang telah dibuat, penulisan komentar dalam laporan masalah, serta manajemen lampiran termasuk pengunggahan, pengunduhan dan penghapusan lampiran. Terakhir, dalam bagian kendala dan solusi, akan dibahas hambatan yang muncul selama pelaksanaan magang serta langkah-langkah yang diambil untuk mengatasi masalah tersebut.

## 3.1 Kedudukan dan Organisasi

Pada kegiatan kerja magang di PT. Global Loyalty Indonesia, berposisi berada pada *department E-Commerce Application Development* tepatnya pada divisi teknologi dan berposisi sebagai *Intern Back End Development*. Meskipun, berposisi sebagai *Intern Back End Development*, pengerjaan proyek yang diberikan lebih berfokus pada pembangunan sistem *front end* aplikasi. Selama proses pengerjaan fitur pusat bantuan, koordinasi dilakukan dengan supervisor, tim *front end* dan *back end* yang tergabung dalam proyek tersebut. Pada proyek fitur pusat bantuan aplikasi internal, ditugaskan untuk merancang dan membangun fitur pusat bantuan khususnya pada modul membuat dan melihat laporan masalah, menulis dan melihat komentar, dan melampirkan, men-*download*, dan menghapus lampiran pada laporan masalah yang telah dibuat.

## 3.2 Tugas yang Dilakukan

Selama proses kerja magang pada PT. Global Loyalty Indonesia, tugas yang dilakukan selam kerja magang terdiri dari beberapa proyek, yaitu pembuatan fitur pusat bantuan, penambahan label *pre-order*, dan juga fitur *scan barcode*

pada aplikasi internal perusahaan. Pada laporan ini akan lebih berfokus pada perancangan dan pembangunan fitur pusat bantuan pada aplikasi internal perusahaan. Pada proyek pembangunan fitur pusat bantuan ini, lebih berfokus pada pengembangan fitur pembuatan tiket untuk melaporkan masalah yang ditemukan oleh karyawan toko, fitur melihat *detail* dari tiket masalah yang telah dibuat, fitur menulis dan melihat komentar pada *detail* tiket masalah, dan mengunggah, mengunduh, dan menghapus lampiran pada pembuatan tiket masalah, *detail* tiket masalah, dan komentar.

## 3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

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

| Minggu Ke - | Pekerjaan yang dilakukan |
|---|---|
| 1 | Memahami dan mempelajari sistem desain dan juga UI/UX dari proyek fitur pusat bantuan, serta pembagian tugas yang dilakukan oleh *supervisor*. |
| 2 | Menyiapkan *working space*, seperti menyiapkan *branch*, *folder*, *file*, dan juga hal pendukung pengerjaan tugas yang didapatkan. |
| 3 | Mengerjakan fungsi untuk pembuatan tiket laporan masalah, termasuk *handle error* ketika tiket gagal dibuat, membuat *toast* untuk indikasi laporan berhasil dibuat. Mengerjakan *routing* dan membuat fungsi untuk memanggil API *detail* tiket. |
| 4 | Membuat *handle error* ketika *detail* tiket tidak ditemukan, membuat modal *error*, melakukan *sprint planning* dua, mengerjakan fungsi dan validasi untuk melampirkan *file* dari kamera, gallery, dan *file system*. |
| 5 | Mengerjakan fungsi untuk memanggil API *upload*, hapus, dan *download file*, serta membuat *handle error* ketika *file* gagal di *upload*, hapus, dan *download* |
| 6 | Mengerjakan fungsi untuk menambahkan komentar pada *detail* tiket, serta membuat *handle error* ketika gagal menambahkan komentar. Melakukan validasi tidak bisa meng-*upload file* dan menambahkan komentar ketika status tiket *close*. |

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

| Minggu Ke - | Pekerjaan yang dilakukan |
|---|---|
| 7 dan 8 | Merapihkan *code* agar lebih efektif dan melakukan *bug fix* yang telah ditemukan oleh tim QA. |
| 9 dan 10 | Merapihkan *code* agar lebih efektif dan melakukan beberapa *fixing bug* terkait dengan lampiran *file* yang telah ditemukan oleh tim TO. |
| 11 | Finalisasi proyek fitur pusat bantuan pada aplikasi internal dan melakukan *deployment* aplikasi untuk dapat digunakan di perusahaan. |
| 12 dan 13 | Melakukan riset untuk *upgrade* aplikasi internal ke versi Nuxt 3. |
| 14 | Mengerjakan proyek *pre-order* penambahan label *pre-order* pada daftar pesanan yang masuk. |
| 15 | Melakukan *sprint planning* pertama *self-service* dan membuat *tombol scan barcode self-service* pada halaman *home* aplikasi internal. |
| 16 | Melakukan *relayout tombol scan barcode self-service* dan melanjutkan riset Nuxt 3. |
| 17 | Membenarkan *bug* dari tim QA dan TO terkait proyek *pre-order* dan *self-service*, serta melanjutkan riset Nuxt 3. |

Berdasarkan tabel uraian pada Tabel 3.1, proses kerja magang terdiri dari tiga proyek utama, yaitu proyek fitur pusat bantuan, penambahan label *pre-order*, dan fitur *scan barcode self-service*. Selama pengerjaan proyek fitur pusat bantuan, koordinasi dan perancangan fitur dilakukan dengan metode *sprint*, dimana setiap *sprint* akan membahas bagian fitur yang harus diprioritaskan. Pada proyek fitur pusat bantuan, terdiri dari beberapa bagian, yaitu membuat tiket laporan masalah, membuat halaman untuk melihat *detail* tiket, menambahkan komentar, dan mengunggah, mengunduh, dan menghapus lampiran. Kemudian, pada proyek *pre-order* tugas yang didapatkan hanya penambahan label *pre-order* pada daftar pesanan yang masuk dan *detail* pesanan *pre-order*. Setelah itu, pada proyek fitur *scan barcode self-service* tugas yang didapatkan adalah membuat fungsi untuk membaca qr dan *barcode* yang dihasilkan ketika ada pesanan masuk. Selain dari ketiga proyek tersebut, diminta untuk *framework* Nuxt versi 3, dimana tujuan riset ini untuk mempertimbangkan apakah aplikasi internal yang saat ini menggunakan

Nuxt versi 2 dapat di*upgrade* menjadi Nuxt versi 3. Pada laporan ini akan lebih berfokus pada pembangunan sistem *frontend* proyek yang pertama, yaitu fitur pusat bantuan.

## 3.4 Perancangan dan Implementasi Hasil Kerja Magang

Dalam bagian ini akan membahas tentang *flow* dan implementasi hasil kerja magang dalam penerapan fitur pusat bantuan, berupa membuat tiket laporan masalah, membuat halaman *detail* laporan masalah, menambahkan komentar, dan mengunggah, mengunduh, dan menghapus lampiran pada aplikasi internal PT. Global Loyalty Indonesia.

### 3.4.1 Teknologi yang Digunakan

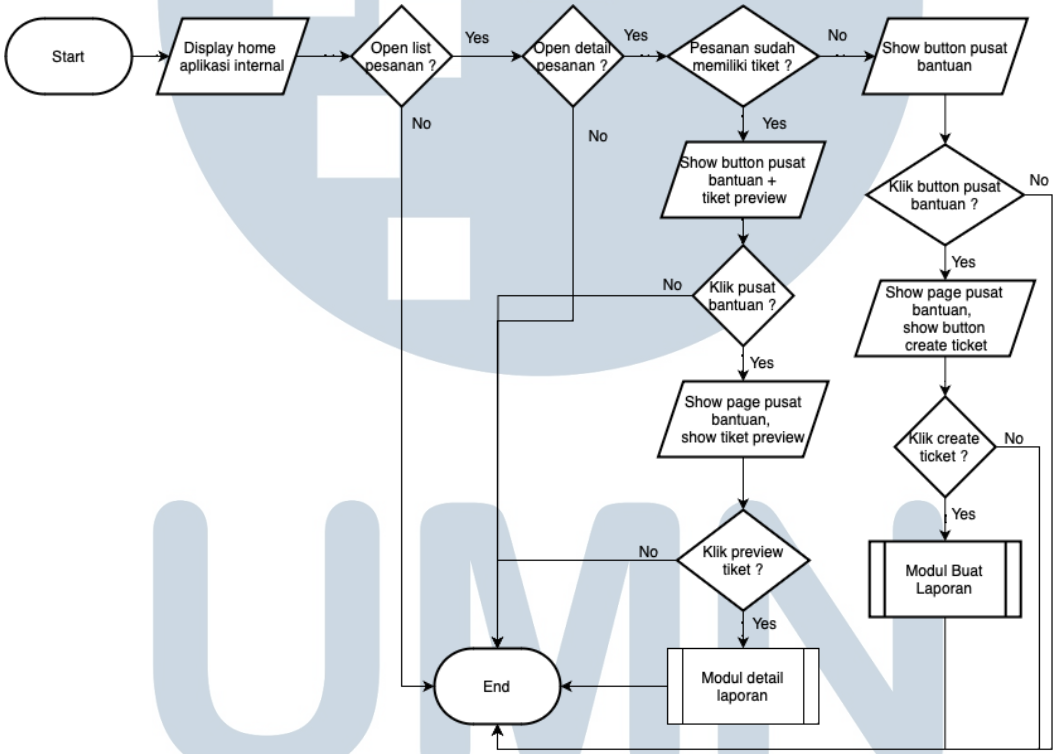
Aplikasi internal PT. Global Loyalty Indonesia adalah aplikasi yang menggunakan konsep *webview*, dimana *webview* sendiri merupakan sebuah aplikasi yang memanfaatkan web sebagai tampilan utamanya. Dalam pengembangan tampilan web pada aplikasi internal ini menggunakan JavaScript dengan *framework* Vue JS. Kemudian, tampilan web tersebut dibungkus oleh aplikasi android, dimana pada aplikasi internal ini menggunakan Flutter. Flutter adalah kerangka pengembangan perangkat lunak *open-source* yang dikembangkan oleh Google [7]. Flutter digunakan untuk membangun antarmuka pengguna (UI) aplikasi *mobile*, web, dan desktop dari satu kode sumber [8]. Dengan Flutter, pengembang dapat membuat aplikasi yang memiliki tampilan seragam dan responsif di berbagai *platform*.

Vue JS adalah suatu inisiatif *open-source* yang dibuat oleh Evan You pada bulan Februari 2014 [9] dan dilisensikan di bawah MIT. Hal ini merupakan sebuah kerangka kerja (*framework*) JavaScript yang memanfaatkan pendekatan komponen untuk menyederhanakan pengembangan antarmuka pengguna (*user interface*) pada aplikasi website. Dengan menggunakan konsep komponen, Vue JS memudahkan para pengembang dengan memecah tata letak dan fungsionalitas menjadi unit-unit yang dapat dikelola secara terpisah. Pada pengembangan aplikasi internal menggunakan *framework* dari Vue JS, yaitu Nuxt JS. Nuxt JS adalah kerangka kerja (*framework*) JavaScript yang dibangun di atas Vue JS [10]. Nuxt menyediakan fungsionalitas tambahan dan konvensi konfigurasi bawaan untuk memudahkan pengembangan aplikasi web Vue JS yang lebih besar dan kompleks.

### 3.4.2 Perancangan Flowchart Aplikasi

Dalam bagian ini akan membahas tentang perancangan *flowchart* dari fitur pusat bantuan yang telah dibuat. Perancangan ini bertujuan untuk menjelaskan alur kerja fitur pusat bantuan secara keseluruhan, serta modul-modul spesifik yang telah dikerjakan, yaitu modul buat laporan, modul *detail* laporan, modul komentar, dan modul lampiran.

**A. Fitur Pusat Bantuan**

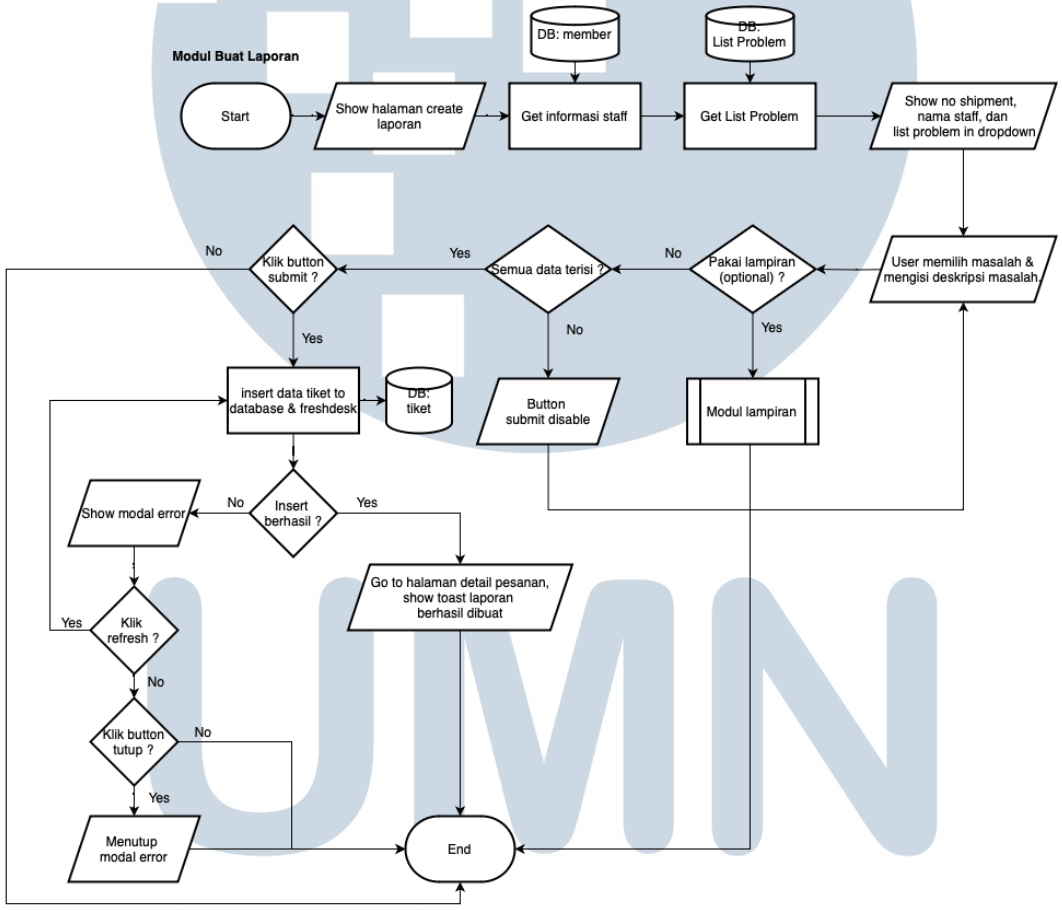



Gambar 3.1. *Flowchart* Fitur Pusat Bantuan secara Keseluruhan

Berdasarkan *flowchart* pada Gambar 3.1 menunjukkan alur fitur pusat bantuan secara keseluruhan, dimana pada saat aplikasi internal dibuka akan menampilkan halaman utama dari aplikasi internal. Ketika *user* membuka daftar pesanan *user* dapat membuka *detail* pesanan dari daftar pesanan tersebut. Pada *detail* pesanan, terjadi pengecekkan apakah pesanan tersebut sudah memiliki tiket laporan masalah atau belum, jika sudah ada akan menampilkan tombol pusat bantuan ditambah dengan *preview* dari tiket yang telah dibuat, sedangkan jika tidak ada akan menampilkan tombol pusat bantuan saja. Kemudian, ketika tombol pusat

bantuan ditekan, maka akan masuk ke halaman pusat bantuan. Pada halaman pusat bantuan, terdapat dua kondisi, dimana ketika sudah memiliki tiket akan menampilkan *preview* tiket dan jika ditekan aplikasi akan diarahkan ke modul *detail* tiket, sedangkan jika belum ada tiket akan menampilkan tombol buat laporan dan jika ditekan aplikasi akan diarahkan ke modul buat laporan.

## B. Modul Buat Laporan

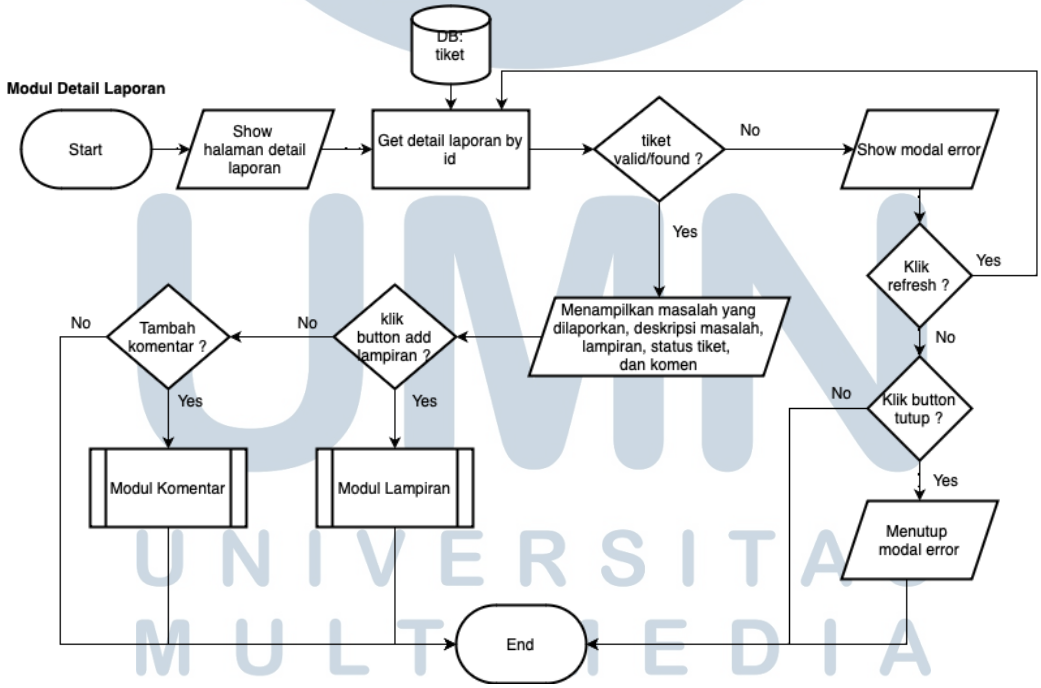



Gambar 3.2. *Flowchart* Modul Buat Laporan

Berdasarkan *flowchart* pada Gambar 3.2, menunjukkan alur kerja dari modul buat laporan, dimana ketika *user* menekan tombol buat laporan, akan masuk ke halaman buat laporan. Pada pertama kali halaman buat laporan dimuat, maka akan memanggil fungsi mengambil data informasi *staff* dari *database member* dan mengambil data *list problem* dari *database list problem*, kemudian dari data tersebut akan ditampilkan nomor pesanan, nama *staff*, dan *list problem*. Kemudian, *user*

diminta untuk memilih masalah yang ingin dilaporkan dan deskripsi masalahnya. Selain itu, *user* juga dapat menambahkan lampiran dan akan diarahkan ke modul lampiran. Kemudian, ketika *user* sudah mengisi semua data, maka akan terjadi pengecekkan apakah semua data yang wajib telah diisi, jika belum maka tombol *submit ticket* akan di*disable*, sedangkan jika sudah maka *user* dapat menekan tombol *submit*. Setelah itu, aplikasi akan melakukan proses memasukkan data laporan ke dalam *database* dan membuat tiket masalah di *freshdesk*, jika proses tersebut berhasil, maka *user* akan diarahkan kembali ke *detail* pesanan dan aplikasi akan menampilkan *toast* dengan pesan laporan berhasil dibuat, sedangkan jika gagal akan menampilkan modal *error*, dimana dalam modal *error* ini terdapat dua tombol, yaitu tutup dan *refresh*. Jika tombol tutup ditekan, maka modal akan tertutup, sedangkan jika memilih tombol *refresh*, maka aplikasi akan men-*submit* ulang laporan tersebut.

## C. Modul *Detail* Laporan




Gambar 3.3. *Flowchart* Modul *Detail* Laporan

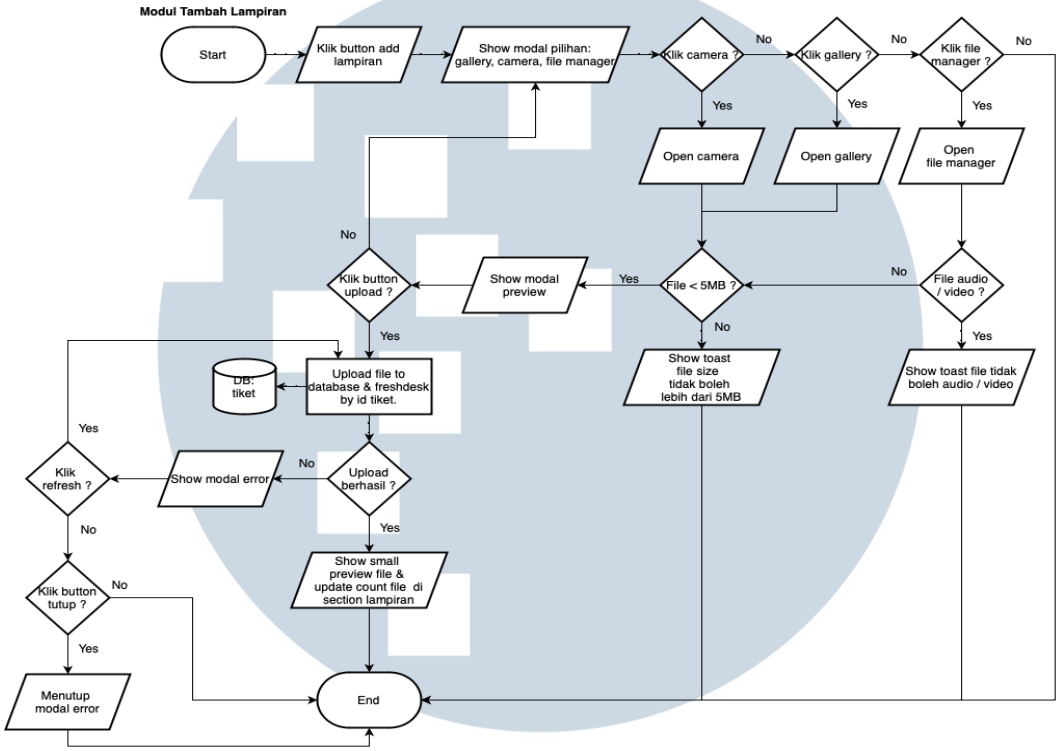
Berdasarkan *flowchart* pada Gambar 3.3 menunjukkan proses dan alur kerja dari modul *detail* laporan, dimana ketika *user* masuk ke halaman *detail* laporan, maka akan terjadi proses pengambilan data *detail* laporan pada *database* tiket

berdasarkan id tiket yang telah dibuat. Kemudian, terjadi pengecekkan apakah tiket tersebut ditemukan atau tidak, jika tidak ditemukan maka akan menampilkan modal *error*, dimana terdapat dua tombol, yaitu tutup dan *refresh*. Jika *user* memilih *refresh*, maka akan terjadi proses pengambilan data *detail* tiket kembali, sedangkan jika *user* memilih tutup, maka modal *error* akan tertutup. Kemudian, jika tiket ditemukan maka aplikasi akan menampilkan masalah yang dilaporkan, status tiket, deskripsi masalah, *preview* dan jumlah lampiran yang diunggah, dan komen pada tiket tersebut. Pada halaman *detail* tiket ini juga terdapat beberapa *action* yang dapat dilakukan, seperti ketika *user* ingin menambahkan lampiran untuk memperkuat masalah yang dilaporkan, maka akan diarahkan ke proses modal lampiran. Selain itu, ketika *user* ingin menambah komentar pada laporan tersebut, maka akan diarahkan ke proses modal komentar.

## D. Modul Lampiran

Pada modul lampiran terbagi menjadi tiga bagian, yaitu modul tambah lampiran, modul hapus lampiran, dan modul *download* lampiran. *Flowchart* setiap modul sebagai berikut.

## D.1 Modul Tambah Lampiran

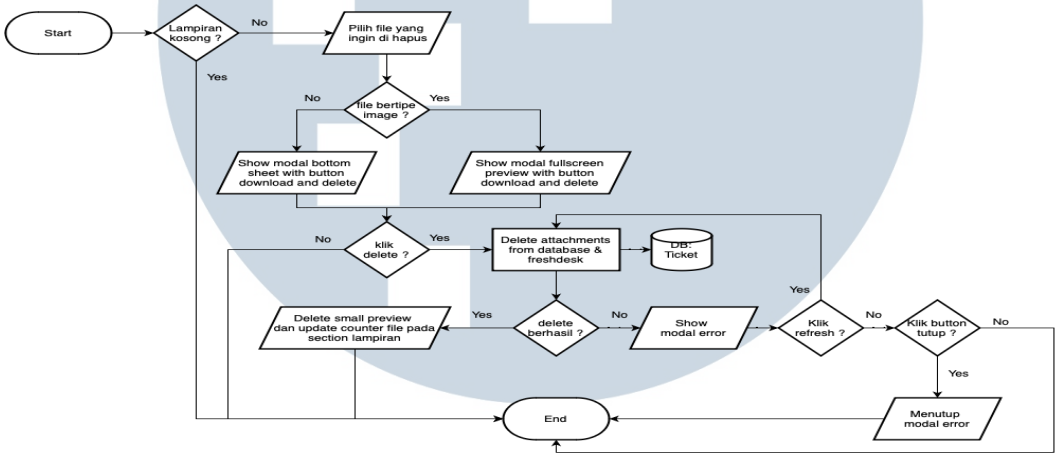



Gambar 3.4. *Flowchart* Modul Tambah Lampiran

Berdasarkan *flowchart* pada Gambar 3.4, menunjukkan alur kerja dari proses penambahan lampiran, dimana *user* dapat menambahkan lampiran pada saat awal pembuatan laporan ataupun pada *detail* tiket dengan memilih tombol lampirkan file dan foto. Pada saat *user* memilih tombol tersebut, maka aplikasi akan menampilkan modal yang berisi pilihan kamera, galeri, atau *file manager*. Kemudian, ketika *user* memilih kamera, maka akan membuka kamera, jika memilih galeri akan membuka galery, dan jika memilih *file manager* akan membuka *file manager*. Pada *file manager* terdapat validasi khusus, dimana *file* yang diunggah tidak boleh bertipe video atau audio, jika *user* mengunggah video atau audio, maka aplikasi akan menampilkan *toast* sebagai indikator kepada *user* bahwa tidak bisa mengunggah video ataupun audio. Setelah *user* memilih *file*, maka akan terjadi pengecekkan apakah *file* tersebut berukuran kurang dari 5MB atau tidak. Jika lebih dari 5MB, maka akan menampilkan *toast* yang nunjukkan *file* tidak boleh lebih dari 5MB, sedangkan jika kurang dari 5MB, maka *user* dapat memilih tombol *upload*. Pada proses *upload*, maka aplikasi akan mengubah bentuk file menjadi bentuk *base64*

dan menunggah ke *database* dan *freshdesk*. Jika proses *upload* tersebut berhasil, maka akan tampil *preview* kecil dari *file* tersebut dan jumlah file akan terupdate pada *section* lampiran, sedangkan jika proses upload gagal, maka akan menampilkan modal *error*, dimana terdapat dua tombol, yaitu tutup dan *refresh*. Jika *user* memilih *refresh*, maka akan *file* akan di-*upload* ulang, sedangkan jika *user* memilih tutup, maka modal *error* akan tertutup.

## D.2 Modul Hapus Lampiran

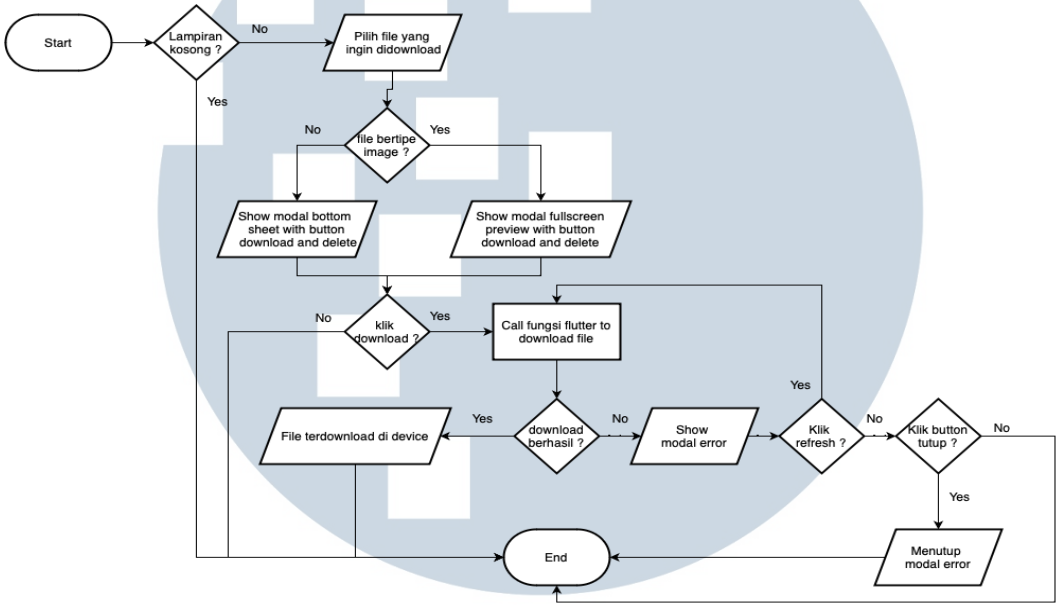



Gambar 3.5. *Flowchart* Hapus Lampiran

Berdasarkan *flowchart* pada Gambar 3.5 menunjukkan proses dan alur kerja dari modul hapus lampiran, dimana ketika terdapat *file* dalam *section* lampiran, maka *user* dapat menghapus *file* tersebut. Proses menghapus *file* dapat dilakukan dengan *user* memilih *file* yang ingin di hapus. Dalam proses ini terdapat validasi tipe *file*, dimana ketika *file* bertipe gambar, maka akan menampilkan modal *fullscreen* yang menampilkan *preview* gambar yang dipilih, dimana dalam modal tersebut terdapat dua tombol, yaitu tombol *download* dan hapus, sedangkan pada *file* yang bertipe selain gambar, akan menampilkan modal *bottom sheet* yang terdapat dua tombol, yaitu tombol *download* dan hapus. Ketika *user* menekan tombol hapus, maka aplikasi akan melakukan proses penghapusan lampiran pada *database* dan *freshdesk*. Jika proses penghapusan tersebut berhasil, maka *preview file* yang dipilih dihapus dan jumlah *file section* lampiran juga terupdate, sedangkan jika proses tersebut gagal, maka akan menampilkan modal *error*, dimana terdapat dua tombol, yaitu tutup dan *refresh*. Jika *user* memilih *refresh*, maka akan melakukan proses

penghapusan ulang, sedangkan jika *user* memilih tutup, maka modal *error* akan tertutup.

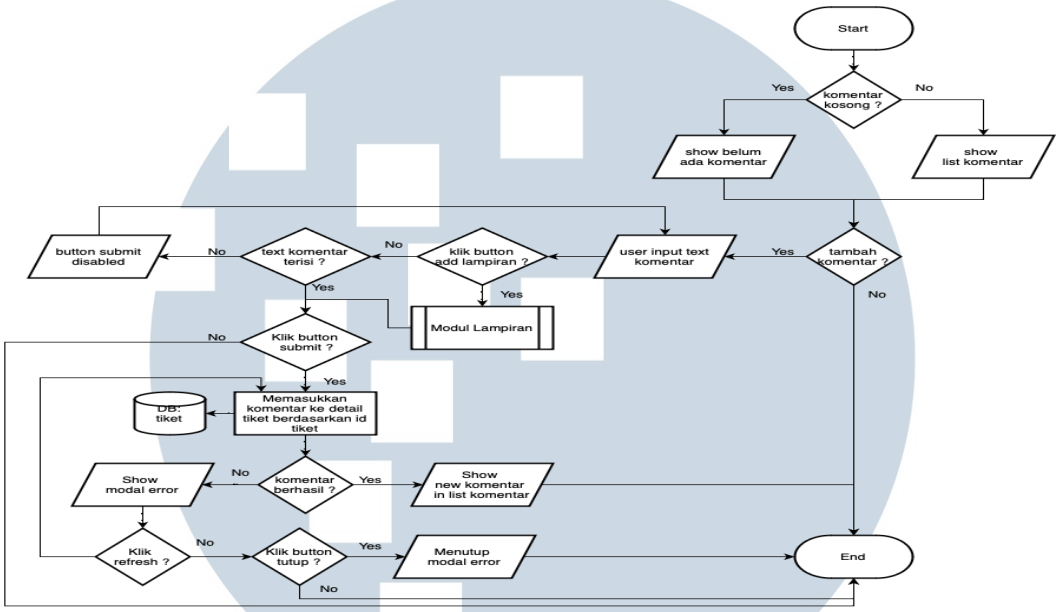
### D.3 Modul *Download* Lampiran




Gambar 3.6. *Flowchart* Modul *Download* Lampiran

Berdasarkan *flowchart* pada Gambar 3.6, menunjukkan proses alur kerja dari modul *download*, dimana ketika *user* ingin men-*download* terdapat pengecekkan tipe *file*, dimana ketika *file* yang dipilih berupa gambar, maka akan menampilkan modal *fullscreen* yang menampilkan *preview* dari gambar yang dipilih, sedangkan pada *file* yang bertipe selain gambar, maka akan menampilkan modal *bottom sheet*. Pada kedua modal tersebut terdapat dua tombol, yaitu tombol hapus dan *download*. Ketika *user* memilih tombol *download*, maka aplikasi akan menjalankan proses pemanggilan fungsi dari flutter untuk melakukan proses *tracking download* dan menambahkan *file* tersebut ke penyimpanan *device user*. Jika proses tersebut berhasil, maka *file* tersebut sudah berada di penyimpanan *device user* tersebut, sedangkan jika proses tersebut gagal, maka akan menampilkan modal *error*, dimana terdapat dua tombol, yaitu tutup dan *refresh*. Jika *user* memilih *refresh*, maka akan *file* di-*download* ulang, sedangkan jika *user* memilih tutup, maka modal *error* akan tertutup.

## E. Modul Komentar



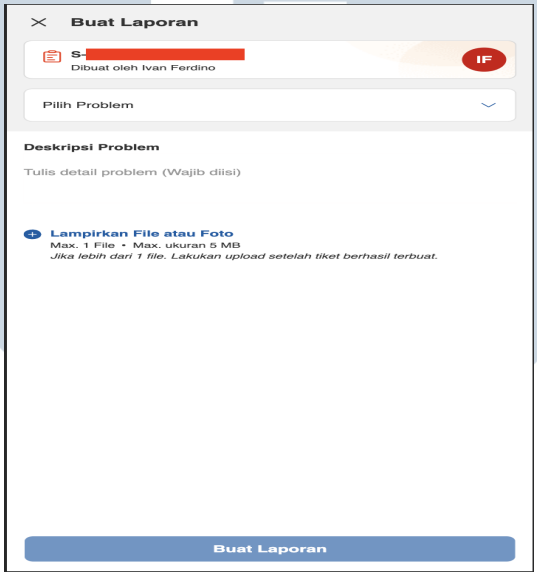

Gambar 3.7. *Flowchart* Modul Komentar

Berdasarkan *flowchart* pada Gambar 3.7, menunjukkan alur kerja dari modul komentar, dimana pada halaman *detail* tiket akan terjadi pengecekkan terdapat komentar atau tidak dalam tiket tersebut. Jika tidak ada komentar akan menampilkan pesan belum ada komentar sebagai indikator komentar kosong, sedangkan jika terdapat komentar akan menampilkan semua komentar yang terdapat dalam tiket tersebut. Kemudian, *user* dapat menambahkan komentar dengan mengisi *input field*, *user* juga dapat menambahkan komentar dengan lampiran dengan menekan tombol tambah lampiran yang akan mengarahkan ke modul lampiran. Lampiran pada komentar ini bersifat *optional*, sehingga *user* dapat memilih untuk melampirkan *file* atau tidak. Kemudian, pada tombol *submit* terdapat pengecekkan, dimana ketika *input field* kosong, maka tombol akan di*disable*, sedangkan jika *input field* terisi dan *user* menekan tombol *submit* maka program akan menjalanka proses memasukkan komentar ke *detail* tiket pada *database* tiket, jika proses tersebut berhasil maka komentar yang baru ditambahkan akan tampil pada *list* komentar, sedangkan jika gagal maka akan menampilkan modal *error*, dimana terdapat dua tombol, yaitu tutup dan *refresh*. Jika *user* memilih *refresh*, maka akan melakukan proses penambahan komentar ulang ke *database*, sedangkan jika *user* memilih tutup, maka modal *error* akan tertutup.

### 3.4.3 Implementasi Aplikasi

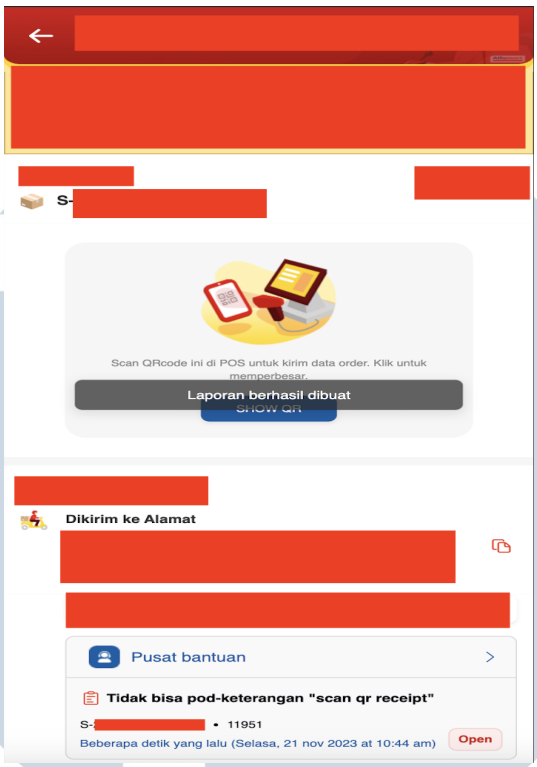
Dalam bagian implementasi aplikasi akan membahas fitur pusat bantuan dari sesi tampilan aplikasi yang berfokus pada modul buat laporan, *detail* laporan, lampiran, dan komentar.

## A. Modul Buat Laporan



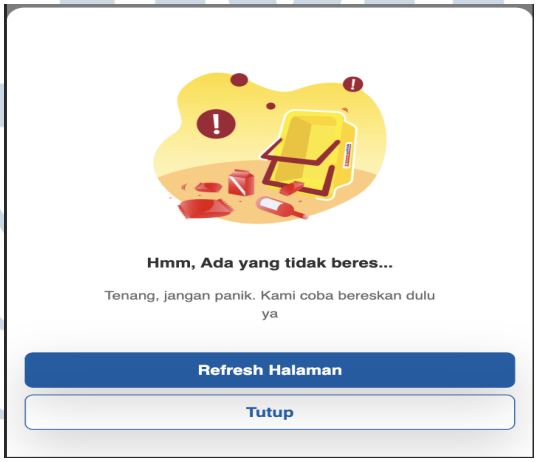

Gambar 3.8. Tampilan Halaman Buat Laporan.

Pada Gambar 3.8 menunjukkan tampilan dari halaman buat laporan, dimana pada halaman tersebut akan menampilkan*staff* yang melaporkan, dan terdapat *dropdown* untuk memilih masalah yang ingin dilaporkan. Kemudian, terdapat deskripsi masalah yang digunakan untuk menampung penjelasan masalah oleh *staff* toko agar lebih jelas. Selain itu, terdapat tombol lampirkan *file* atau foto untuk menambahkan lampiran yang dapat mendukung masalah yang dilaporan. Kemudian, pada Gambar 3.8 terlihat tombol buat laporan ter-*disable*, hal ini dikarenakan *user* belum memasukkan data yang bersifat wajib, yaitu masalah yang ingin dilaporan dan deskripsi masalah. Kemudian, ketika laporan berhasil dibuat maka *user* akan diarahkan kembali ke *detail* pesanan dan aplikasi akan menampilkan *toast* sebagai indikator bahwa laporan berhasil dibuat, seperti pada Gambar 3.9.



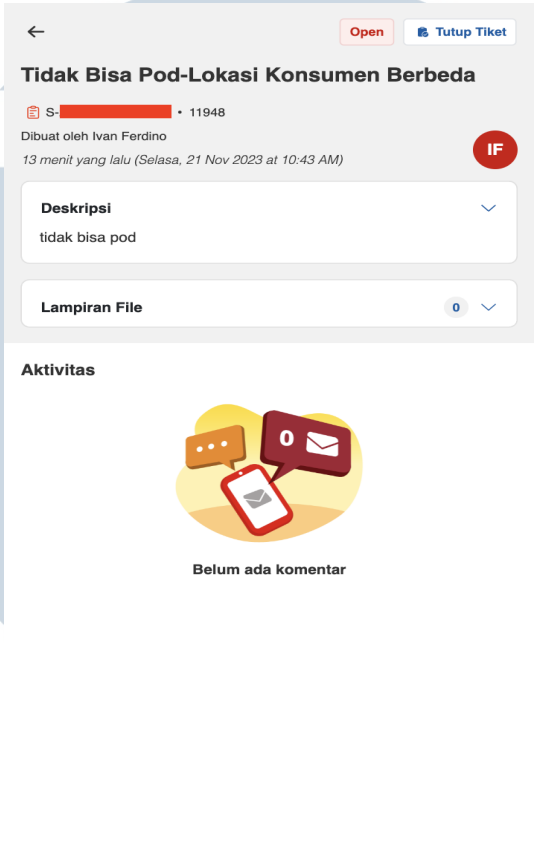

Gambar 3.9. Tampilan Notifikasi Laporan Berhasil Dibuat.

Jika proses pembuatan laporan gagal, maka akan menampilkan modal *error*, seperti pada Gambar 3.10, dimana terdapat dua tombol, yaitu *refresh* halaman dan tutup. Jika *user* menekan tombol *refresh* halaman, maka akan mengulang proses buat laporan, sedangkan jika menekan tombol tutup maka akan menutup modal tersebut.




Gambar 3.10. Tampilan Modal *Error*.

## B. Modul *Detail* Laporan




Gambar 3.11. Tampilan Halaman Detail Laporan

Pada Gambar 3.11 menunjukkan tampilan dari halaman detail laporan, dimana pada halaman tersebut akan menampilkan masalah yang dilaporan, status tiket, nomor pengiriman, id tiket, staff yang melapor, deskripsi masalah yang dilaporkan, dan waktu tiket tersebut dibuat. Kemudian, pada halaman *detail* tiket juga menampilkan bagian lampiran *file* yang dibahas pada bagian modul lampiran dan juga bagian komentar yang akan dibahas pada modul komentar. Proses pengambilan data *detail* tiket ini melalui proses pemanggilan API, dimana API tersebut akan mengambil data tiket dari *database* tiket berdasarkan id tiket. Jika id tiket tidak terdaftar atau tiket tidak ditemukan, maka akan menampilkan modal *error* yang ditunjukkan pada Gambar 3.10. Pada modal tersebut, terdapat dua tombol, yaitu *refresh* halaman dan tutup, dimana ketika *user* memilih *refresh* halaman, maka halaman akan dimuat kembali dan memanggil fungsi untuk

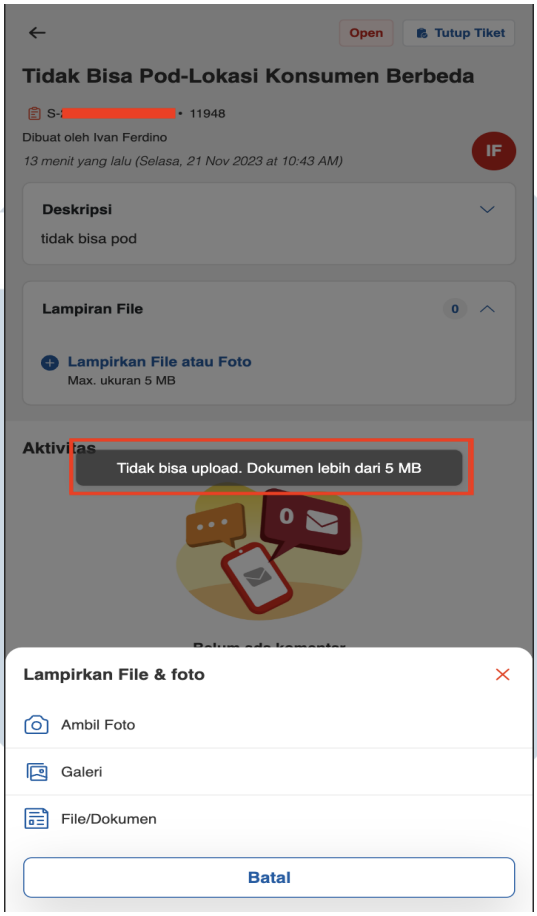
mengambil ulang *detail* tiket, sedangkan jika *user* memilih tutup, maka modal tersebut akan ditutup.

## C. Modul Lampiran

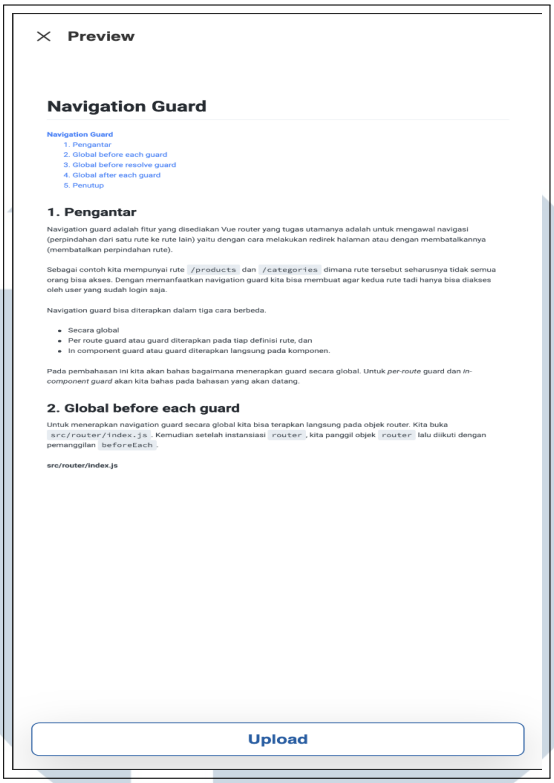

Gambar 3.12. Tampilan *Section* Lampiran


Gambar 3.13. Tampilan Modal Pilihan Lampiran

Pada Gambar 3.12 menunjukkan tampilan dari *section* lampiran, dimana user dapat menambahkan *file* dengan menekan tombol lampirkan file atau foto. Setelah *user* menekan tombol tersebut, maka akan menampilkan modal pilihan seperti pada Gambar 3.13. Pada modal tersebut terdapat tiga pilihan, yaitu ambil foto, galeri, dan *file*/dokumen, ketika *user* memilih ambil foto maka aplikasi akan membuka kamera, sedangkan ketika *user* memilih galeri akan membuka galeri, dan jika memilih *file*/dokumen akan membuka *file manager*. Kemudian, setelah *user* memilih *file* yang ingin ditambahkan, maka akan terjadi pengecekkan ukuran *file* dan tipe *file* yang dipilih, jika *file* tersebut lebih dari 5MB, maka akan menampilkan *toast* sebagai indikator *file* terlalu besar seperti pada Gambar 3.14, sedangkan jika *file* tersebut kurang dari 5MB, maka akan menampilkan *preview* terlebih dahulu, seperti pada Gambar 3.15.




Gambar 3.14. Tampilan *Toast File* Terlalu Besar

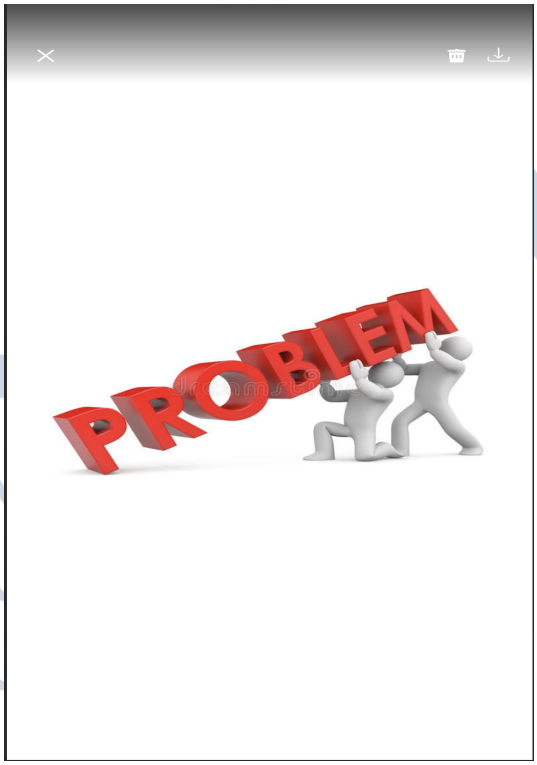



Gambar 3.15. Tampilan Modal Preview Lampiran

Pada Gambar 3.15, terdapat tombol *upload* yang dapat digunakan untuk melakukan *upload file* ke *database* tiket. Ketika *upload* berlangsung, maka *file* tersebut akan dikonversikan menjadi bentuk *base64*, kemudian aplikasi akan melakukan pemanggilan API *upload file*, dimana API tersebut akan melakukan proses memasukkan data ke *database* dan freshdesk, jika proses tersebut berhasil maka akan muncul *preview* kecil dari *file* tersebut di *section* lampiran dan jumlah *file* juga diperbaharui seperti yang ditunjukkan pada Gambar 3.16, sedangkan jika proses tersebut gagal maka akan menampilkan modal *error* yang ditunjukkan pada Gambar 3.10. Pada modal tersebut, terdapat dua tombol, yaitu *refresh* halaman dan tutup, dimana ketika *user* memilih *refresh* halaman, maka akan menjalankan fungsi untuk mengunggah ulang *file* yang gagal tersebut, sedangkan jika *user* memilih tutup, maka modal tersebut akan ditutup.


Gambar 3.16. Tampilan *Section* Lampiran Setelah Berhasil *Upload*

Lampiran yang sudah ter*upload* dapat dihapus dan diunduh dengan cara memilih *file* yang ingin dihapus atau diunduh, kemudian terdapat validasi tipe dari tipe *file* yang dipilih, jika *file* tersebut bertipe gambar, maka akan menampilkan modal *fullscreen* yang berisi gambar dari *file* yang dipilih seperti pada Gambar 3.17, sedangkan jika *file* bertipe selain gambar, maka akan menampilkan modal *bottom sheet* seperti pada Gambar 3.18.

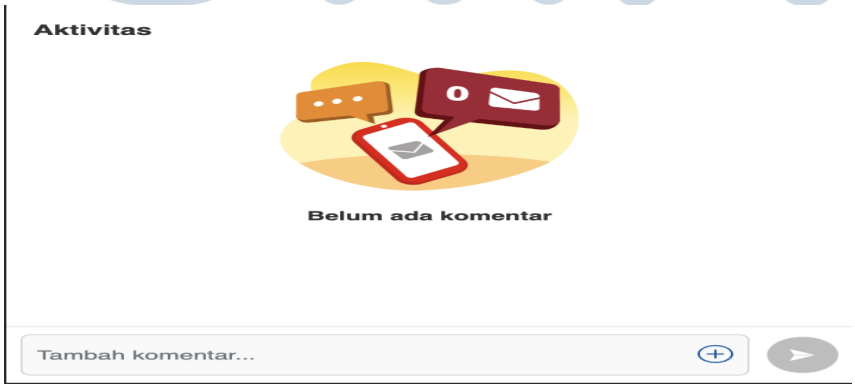



Gambar 3.17. Tampilan Modal *Fullscreen* Hapus dan *Download* Lampiran.


Gambar 3.18. Tampilan Modal *Bottom Sheet* Hapus dan *Download* Lampiran.

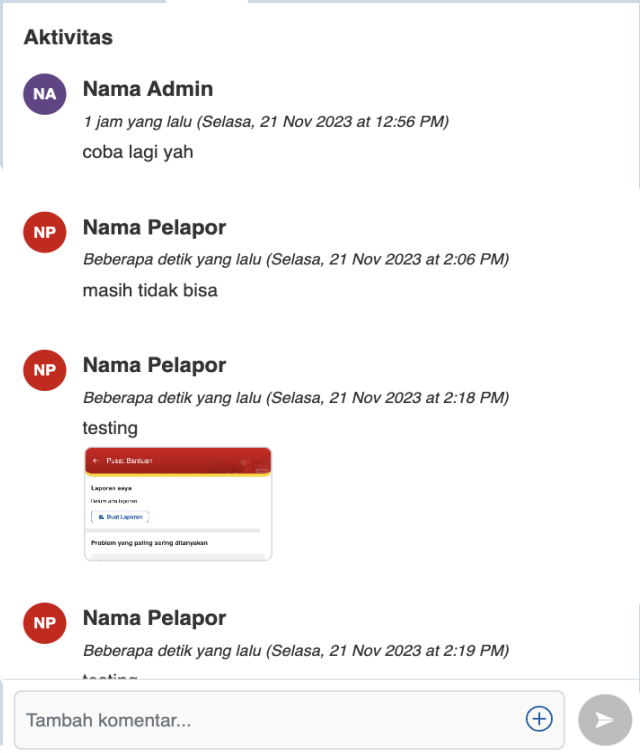
Pada Gambar 3.17 dan 3.18, terdapat dua tombol, yaitu hapus lampiran dan *download* lampiran. Ketika *user* memilih hapus lampiran, maka aplikasi akan memanggil API untuk menghapus *file* tersebut dari *database* tiket berdasarkan id lampirannya, jika berhasil maka *file* tesebut akan terhapus dari *section* lampiran dan jumlah file akan diperbaharui. Kemudian, jika *user* memilih *download* lampiran, maka aplikasi akan memanggil fungsi dari flutter untuk mengunduh *file* tersebut berdasarkan *URL* tempat *file* tersebut tersimpan mengakses ruang penyimpanan *device user* untuk memasukkan *file* yang sudah ter-*download*, jika proses tersebut berhasil maka *file* tersebut akan masuk ke *file storage device user*. Jika proses hapus dan *download* gagal, maka akan menampilkan modal *error* yang ditunjukkan pada Gambar 3.10. Pada modal tersebut, terdapat dua tombol, yaitu *refresh* halaman dan tutup, dimana ketika *user* memilih *refresh* halaman, maka akan menjalankan fungsi untuk menghapus dan men-*download* ulang *file* yang gagal tersebut, sedangkan jika *user* memilih tutup, maka modal tersebut akan ditutup.

## D. Modul Komentar



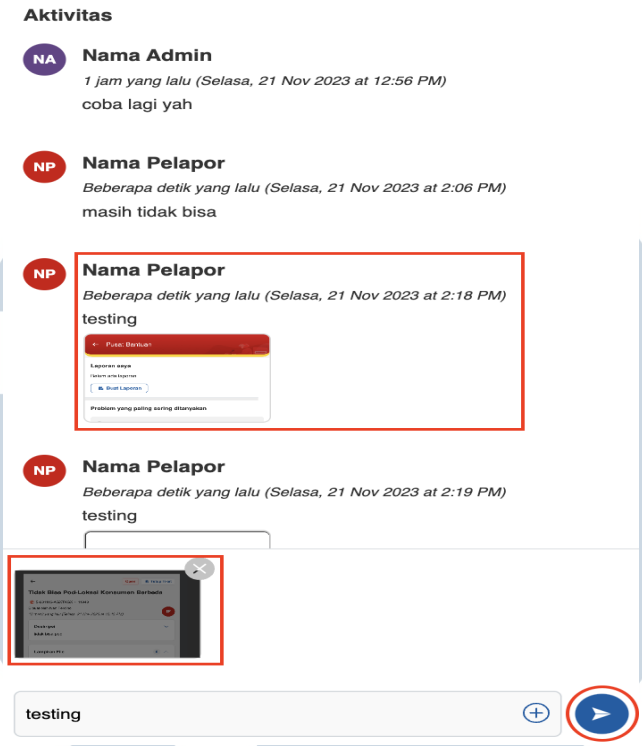

Gambar 3.19. Tampilan *Section* Komentar.

Pada Gambar 3.19 menunjukkan tampilan dari *section* komentar, dimana terdapat tempat untuk menampilkan *list* komentar dan terdapat *input field* untuk *submit* untuk mengirimkan komentar. Pada *section* komentar terdapat validasi apakah komentar pada tiket tersebut kosong atau tidak, jika kosong akan menampilkan *wording* belum ada komentar, seperti pada Gambar 3.19, sedangkan jika terdapat komentar akan menampilkan inisial dari nama yang mengirimkan komentar, dimana warna ungu mewakilkan komentar dari *admin* dan warna merah untuk mewakilkan komentar dari pembuat laporan atau *staff* toko. Kemudian, terdapat nama yang mengirimkan komentar, waktu pengiriman, dan isi komentar yang dikirimkan, seperti yang ditunjukkan pada Gambar 3.20.




Gambar 3.20. Tampilan *Section* Komentar Tidak Kosong.

Ketika *user* ingin mengirimkan komentar, *user* dapat melakukan pengisian pada *input field* dan *user* juga dapat menambahkan lampiran pada komentarnya dengan menekan tombol tambah. Pada tombol *submit* terdapat validasi jika *input field* kosong, maka tombol *submit* akan di*disable*. Proses melampirkan *file* pada komentar memiliki mekanisme yang sama pada saat melampirkan *file* pada tiket, tampilan komentar dengan lampiran dapat dilihat pada Gambar 3.21.




Gambar 3.21. Tampilan Komentar dengan Tampilan.

Pada *section* komentar juga terdapat validasi bahwa ketika ingin mengirikan komentar tidak boleh hanya mengirimkan lampiran saja dan harus memiliki pesan komentar. Jika validasi yang terdapat pada komentar telah terpenuhi, maka *user* dapat mengirimkan komentar tersebut dengan menekan tombol *submit*, dimana aplikasi akan menjalankan proses pemanggilan API *create commentar* untuk memasukkan komentar baru ke dalam *list* komentar pada *database* tiket, jika proses penambahan komentar maka akan komentar yang baru saja ditambahkan akan masuk ke dalam *list* komentar di *detail* tiket, seperti pada Gambar 3.20 dan 3.21. Jika proses komentar gagal dikirimkan, maka akan menampilkan modal *error* yang ditunjukkan pada Gambar 3.10. Pada modal tersebut, terdapat dua tombol, yaitu *refresh* halaman dan tutup, dimana ketika *user* memilih *refresh* halaman, maka akan menjalankan fungsi untuk menghapus dan men-*download* ulang *file* yang gagal tersebut, sedangkan jika *user* memilih tutup, maka modal tersebut akan ditutup. Kemudian, lampiran pada komentar yang dikirimkan oleh pelapor dapat di-*download* dan dihapus dengan mekanisme yang sama seperti pada modul lampiran, sedangkan *admin* hanya bisa di-*download* dan tidak bisa dihapus oleh si pelapor masalah.

## 3.5 Kendala dan Solusi yang Ditemukan

Kendala dan solusi yang ditemukan selama proses pelaksanaan kerja magang, sebagai berikut.

### 3.5.1 Kendala yang Ditemukan

1. Pada proses pengerjaan fungsi buat laporan dengan lampiran, karena aplikasi bersifat *webview* terdapat kendala, dimana ketika pembuatan laporan melalui *website* berhasil terbuat, sedangkan pada saat dilakukan pembuatan laporan melalui aplikasi *andorid* gagal.
2. Pada proses pengerjaan modal *preview* lampiran, kesulitan untuk menampilkan *file preview* untuk tipe *file* pdf.
3. Pada proses pembuatan fungsi untuk melakukan *download file*, dimana terdapat keterbatasan dari *webview* untuk melakukan *download file* secara langsung melalui *website*.

### 3.5.2 Solusi yang Ditemukan

1. Melakukan *debugging* dengan tim *backend*, ditemukan masalah pada saat pembuatan laporan di aplikasi *android*, data yang seharusnya dikirimkan berbentuk *multipart-formdata* terbaca menjadi *textplain*. Kemudian, solusi yang disepakati adalah merubah bentuk lampirannya menjadi bentuk *base64*.
2. Melakukan riset di internet, kemudian ditemukan *library* vue-pdf-embed yang menyediakan fungsi untuk menampilkan *preview pdf* pada Vue JS.
3. Melakukan diskusi dengan *supervisor* dan juga tim *android*, sehingga disepakati dimana tim *android* menyediakan fungsi dari sisi flutter untuk melakukan *download file*, sehingga dari sisi *website* dapat memanggil fungsi tersebut untuk melakukan *download file* dengan hanya mengirimkan URL *file*nya saja.