BAB III
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Kegiatan kerja magang yang di dilakukan di PT. Global tiket Network pada 10 Juni 2019 hingga 9 September 2019 diposisikan dalam *Frontend Engineer* dalam tim *Technology – My Order*. Proses kerja magang berlokasi di Gedung Graha Niaga Thamrin Lantai 5, Jl. KH Mas Mansyur RT. 2/ RW. 8, Jakarta Pusat. Dalam melakukan kerja magang, penulis ditempatkan pada tim *My Order* sebagai *Frontend Engineer.*

Proses kegiatan kerja magang yang dilakukan dalam tim *My Order* ini disupervisi oleh Bapak Immanuel Bayu sebagai *Engineer Manager* hingga Agustus 2019 dan Bapak Rolies Deby hingga saat ini sebagai *Engineer Manager* dalam tim *My Order* dan pembimbing lapangan magang oleh Bapak Rudy Ondolan sebagai *Frontend Engineer* dalam tim.

Di dalam tim *My Order* terdapat enam seksi pembangun fitur *My Order* Tiket.com terkait sebagai berikut:

- *Engineer Manager* bertugas memimpin serta memberi nasehat terhadap suatu keputusan yang diambil tim bersama selama berjalannya proyek.
- *Backend Engineer* bertugas menyiapkan *endpoint* yang akan di akses untuk seksi *Frontend* serta membangun, mengimplementasikan lalu lintas server dengan mengatur data yang terkait *My Order* sesuai seksi *Product Manager* arahkan.
- *Frontend Engineer* bertugas sebagai mengimplementasikan antarmuka

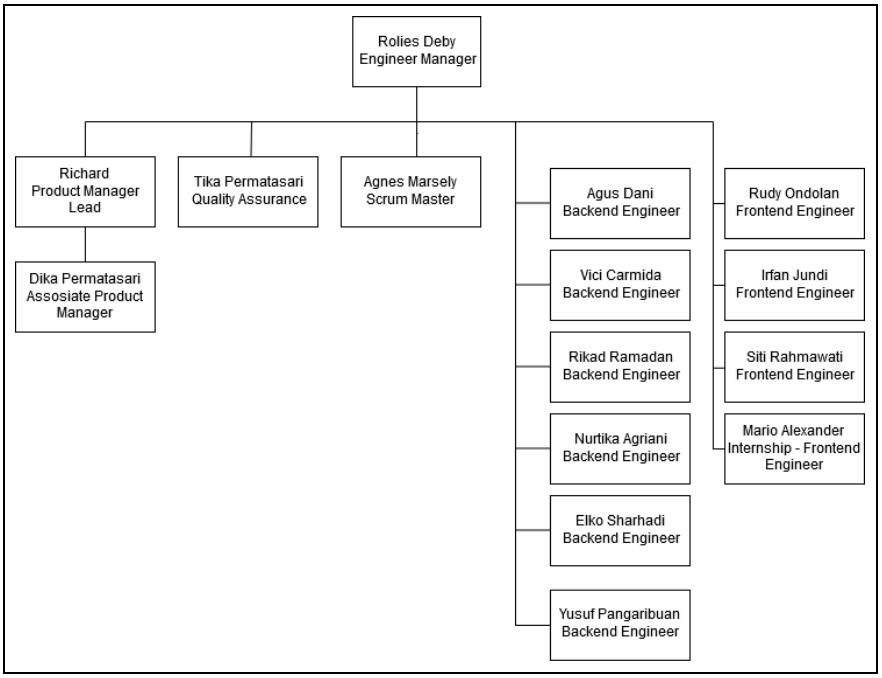
sesuai seksi *Product Manager* arahkan dengan *framework* React dan melakukan pengambilan data dari *endpoint* yang telah seksi *Back*end buat.

- *Product Manager* bertugas melakukan mengatur *tasks* yang akan dikerjakan suatu *sprint* ,mengevaluasi hasil *project* dan menjembatani seksi *UI/UX* dan *Bussiness Intelligence* dengan *Engineer.*
- *Quality Assurance* bertugas mencari *defect* dan *bug* dalam projek baik yang dibangun *Frontend* dan *Backend*, melakukan *smoke test*, dan melakukan *performance test*, dan membuat *report* setiap *defect* atau *bug* yang telah dihilangkan.
- *Scrum Master* bertugas memperkenalkan penggunaan *Scrum* ke tim. Contohnya seperti dengan membantu anggota tim untuk menjadi fasilitator diskusi agar dapat memahami interaksi mana yang bermanfaat dan tidak bermanfaat dalam berjalannya proyek.

Manajemen pengembangan proyek piranti lunak yang digunakan perusahaan PT. Global Network ini menggunakan *framework agile scrum. Scrum* adalah sebuah *framework* kerja untuk mengembangkan, menghantarkan dan mengelola produk yang kompleks guna mempermudah proses pengembangan. Proses koordinasi antar anggota tim *My Order* dapat dilakukan dalam suatu rapat yang disebut *Standup Meeting. Standup Meeting* diadakan sekitar pukul 10:30 WIB dan diikuti seluruh anggota tim. Tujuan dari rapat ini adalah untuk menceritakan hasil kerja semua anggota yang sudah dilakukan hari sebelumnya dan melakukan diskusi dan koordinasi jika dibutuhkan. Kemudian terdapat rapat mingguan tim *My Order* yang disebut *Weekly Sync-Up* dijadwalkan rutin hari Rabu. Rapat ini bertujuan untuk membahas pekerjaan yang telah dikerjakan tiap

tugas atau bisa disebut *story* yang telah diberikan kepada anggota dan melakukan diskusi serta koordinasi dengan tim lain yang berkaitan pada *story* yang sedang dikerjakan jika dibutuhkan.

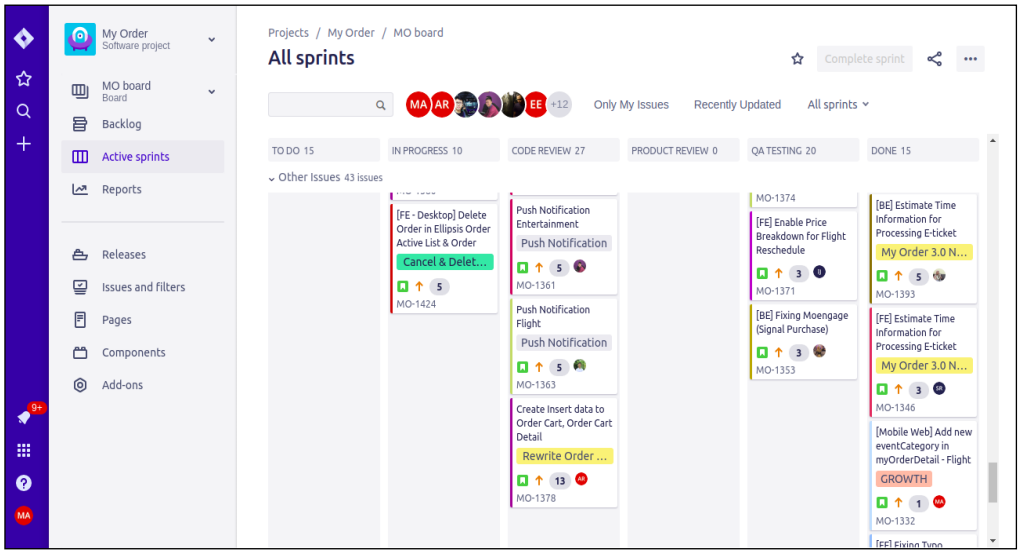
Adapun rapat rutin lain yang bernama *Backlog Grooming* dan *Sprint Planing* yang dilaksanakan dua minggu sekali atau bisa disebut satu *sprint*. *Sprint Planning* dan *Backlog Grooming* ini dilaksanakan sebelum pergantian *sprint*, dilaksanakan hari Senin atau Selasa bertujuan untuk berdiskusi *story* apa saja yang harus diselesaikan di *sprint* kedepannyam menentukan penanggung jawab atas *story* yang ada, dan menentukan *story point. Story point* merupakan bobot nilai tingkat kesulitan suatu *story* dalam bentuk bilangan prima. Manfaat dari *sprint planning* dan *backlog grooming* untuk menghitung *velocity* tim tiap *sprint* yang telah dilewati sehingga menjadi pedoman *sprint* ke depan menentukan jumlah *story*. Tim *My Order* memiliki struktur dan nama anggota ditunjukan pada Gambar 3.1:



Gambar 3.1 Struktur Anggota Tim *My Order*

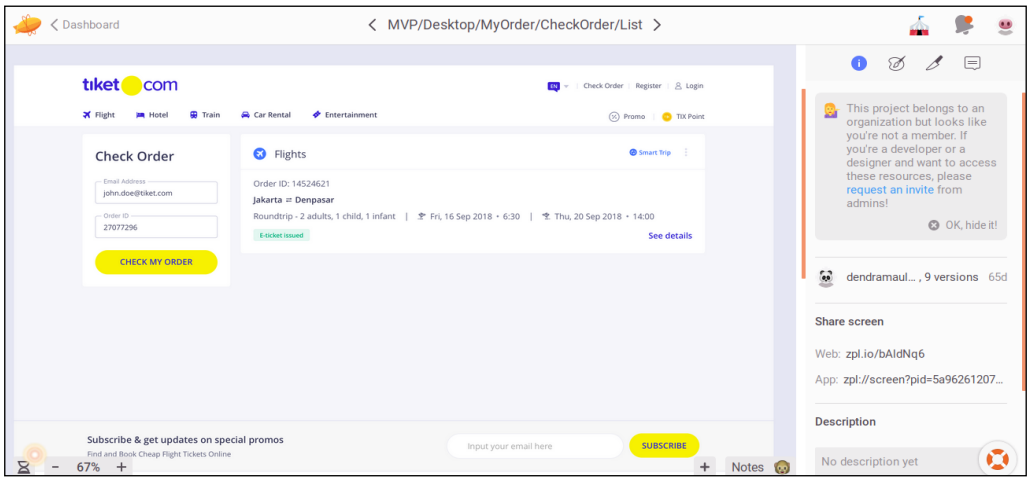
Tim *My Order* ini berada dalam divisi *Technology* dan merupakam bagian dari tim *platform*, selain *platform* terdapat tim lain bernama tim *apps, product,* dan *transport*. Selain *My Order* di dalam tim *platform* terdapat tim lain seperti *utilities, loyalty, payment, member,* dan *refund.*

Proses pengembangan proyek dilakukan dengan membagi *story* ke *engineer* ketika *sprint planning*. Segala visualisasi daftar *story* dalam suatu *sprint* tim dapat dilihat dalam media *Jira Board* yang terdapat di dalam aplikasi *web* bernama Atlassian. Contoh *Screenshot* Atlassian ditunjukan pada Gambar 3.2.



Gambar 3.2 Contoh *Screenshot Jira Board* Atlassian

Setiap *story* yang diberikan ke anggota *Engineer* masing-masing diwajibkan membuat *branch* baru dengan nama kode dari *story* yang diterima penanggungjawab sebagai *branch* kerja. Setiap *story* yang telah diselesaikan wajib melakukan *pull request* ke *branch develop*, dilakukan *code review* dahulu kemudian dapat di-*merge* ke *branch develop*. Selain Jira Board terdapat suatu perangkat lunak pendukung kolaborasi antara *UI designers* dan *Frontend Engineer*, yaitu Zeplin yang ditunjukan pada Gambar 3.3.



Gambar 3.3 Contoh *Screenshot* Zeplin

Zeplin digunakan desainer *UI* dan *Frontend Engineer* untuk menjadi media kolaborasi pengembangan desain antarmuka *user*. Zeplin digunakan sebagai pedoman bagi *Frontend Engineer* untuk mengimplementasikan halaman antarmuka yang ada di *web.* Sedangkan bagi *UI designer* Zeplin memungkinkan *designer* untuk meng-*upload wireframe.*

3.2 Tugas yang Dilakukan

Tugas yang dilakukan selama proses kerja magang adalah mengembangkan *website* bagian halaman *My Order Tiket.com.* Tugas-tugas yang dikerjakan adalah mengembangkan fitur baru *My Order* yaitu fitur *Check Order* yang merupakan suatu fitur pengecekan pesanan dengan 2 jenis parameter autentikasi, yaitu *email* dan OrderId, dengan bahasa pemrograman JavaScript, HTML dan SASS serta dibantu dengan *Query* API GraphQL, mengembangkan antarmuka *My Order Dashboard Support* yang digunakan Admin *IT* Support, mengembangkan fitur *Tracking* menggunakan *Google Tag Manager* terhadap pesanan *Event Attraction* dan melakukan *unit testing* terhadap komponen *Gallery.desktop* dan *Share Receipt* menggunakan JEST.

Proses kerja menggunakan perangkat Github sebagai sarana *Version Control* Proyek. Setiap tim dalam Tiket.com memiliki minimal 1 proyek atau repositori kerja masing-masing. Setiap *pull request* yang akan dimasukkan ke *branch master* wajib melalui *branch develop* dahulu, dilakukan *testing* oleh *Quality Assurance* kemudian baru dapat di *merge* ke *master*, jika lulus *testing*. Realisasi dari proses kerja magang dapat dilihat pada Tabel 1.

Tabel 3.1 Realisasi Kerja Magang

Minggu	Pekerjaan yang Dilakukan
1	• Mempelajari *Framework* React • Instalasi *Software* laptop, seperti Github, Slack, Node, Command Linux Sudo • Mengimplementasikan Redux, Global Component Prop in React
2	• Mempelajari Redux-Thunk (Async Task Fetch Data) • Mempelajari *Framework* React II, terkait *lifecycle* • Mengatur *output* kalimat result di komponen Filter Order
3	• Mempelajari *unit testing* dasar menggunakan JEST • Mempelajari Redux Thunk II • Membuat task React Framework melakukan fetch data swapi.co
4	• Mempelajari layout dengan bahasa SASS • Menyelesaikan task React Framework Fetch Data Swapi (beserta redux , redux-thunk).
5	• Mengatur *logic* komponen-komponen antarmuka di *Dashboard Order Support* • Membuat Datepicker, Input Form, List Item hingga implementasi
6	• Melakukan pemahaman lebih lanjut mengenai struktur *repository frontend* dipimpin oleh Senior Engineer Frontend. • Mempelajari *Query* API, men-*fetch* data (*query*) ke *server endpoint*
7	• Mempelajari *Query* API, melalui komponen pembentuk satu persatu, seperti *schema, model, type, queries* dan *mutation* diimplementasikan ke projek swapi.co.
8	• Mempelajari *Query* API, terhadap pengaplikasiannya terhadap sisi *client-side*, menggunakan Apollo Provider, diimplementasikan ke projek swapi.co.
9	• Mengimplementasikan GraphQL *Query* API kepada fitur Check Order dengan *end point* "order-list/checkorder" • Melakukan review balikan tim product halaman *My Order*

Tabel 3.1 Realisasi Kerja Magang (Lanjutan)

Minggu	Pekerjaan yang Dilakukan
10	• Mengimplementasikan GraphQL client-side dengan bantuan React Apollo hingga konfigurasi ke komponen Card Order
11	• Mengulang pembuatan *Dashboard Support* bagian antarmuka, membuat Modal, Snackbar top, serta testing komponen *dashboard* yang dibuat
12	• Melakukan unit testing terhadap komponen Share Receipt dan Gallery Desktop • Membantu implementasi tracking dengan GTM (Event & Attraction)

3.3 Uraian Pelaksanaan Kerja Magang

Proses pelaksanaan kerja magang dibagi menjadi tiga bagian, yaitu proses pelaksanaan kendala yang ditemukan, dan solusi atas kendala yang ditemukan.

3.3.1 Proses Pelaksanaan

Proses pelaksanaan pengembangan fitur *My Order* Tiket.com yang berbasis *web* membutuhkan perangkat lunak dan perangkat keras. Perangkat lunak yang digunakan dalam mengembangkan fitur dan *website* ini adalah sebagai berikut.

1. NodeJS versi 12.7.0
2. Firefox v68.0.2 (64 bit) dan Google Chrome v75.0.3770.142
3. *Operating System* Ubuntu 18.04.2 (64 bit)
4. Visual Studio Code versi 1.36.1
5. Github versi 2.17.1
6. Slack versi 4.0.0 (64 bit)

Perangkat keras yang digunakan untuk membantu fitur *website* ini adalah Dell Inspiron 14 5000 dengan spesifikasi sebagai berikut.

1. Prosesor Intel i7 2.70GHz
2. RAM 8 GB
3. HD Graphics 620 (Kaby Lake GT2)

Proses pengembangan fitur *My Order* dilaksanakan terhadap empat jenis repositori yaitu, repositori *client side My Order*, *client side dashboard support*, *query* API *My Order, dan query* API *Dashboard Support.* Selama pelaksanaan kerja magang terbagi menjadi beberapa pekerjaan seperti sudah didaftarkan dari tabel realisasi kerja magang. Pekerjaan yang dilakukan yaitu pengembangan fitur *Check Order, Dashboard Support, tracking user data* dengan GTM. Adapun larangan langsung dari perusahaan untuk tidak mendokumentasikan *source code* sehingga pendekatan yang digunakan untuk pelaksanaan kerja, yaitu menguraikan *User Requirement*, pembuatan *Flowchart* untuk fitur *Check Order* dan *Dashboard Support*, Implementasi hasil dari tiap pekerjaan, dan proses *testing.*

A. Requirement

Requirement merupakan suatu gambaran atau *scope* dari layanan serta batasan suatu sistem yang akan dibangun. Terdapat tiga *requirement* yang melatarbelakangi dan menjelaskan pengembangan proyek fitur *My Order* berdasarkan tiap pekerjaan, yaitu *requirement* pada penambahan fitur baru *Check Order*, halaman *Dashboard Support,* dan *tracking user data* dengan GTM.

A.1 Check Order

Check Order merupakan fitur baru yang menjadi salah satu fitur yang mirip *My Order*. Fungsi utama fitur baru ini memungkinan *user* yang

mengunjungi *web* Tiket.com ini dapat mengecek data pesanan yang sesuai id pesanan tanpa melakukan *login* dengan memasukkan *email* dan id pesanan.

Hal yang melatarbelakangi pembuatan fitur baru ini dikaji melalui kebiasaan *user*. Sebelumnya, Tiket.com telah memberi layanan agar *user* dapat membuat pesanan tanpa melakukan *login* ataupun *register*, hal ini tetap memerlukan *input* berupa *email* ketika memesan pesanan agar user dapat menerima *file* pdf pesanan hingga saat *register user* hanya perlu memasukkan *password*. Setelah *user* yang telah melakukan pemesanan dan pembayaran pesanan, *user* seringkali perlu untuk melihat status pesanan yang telah dipesan ke sistem. Namun karena terdapat beberapa *user* tidak melakukan *register* dan *login* ke sistem, hal ini tidak dapat dilakukan. Atas latar belakang ini, dibuatlah fitur *Check Order* bagi *user* yang ingin pengalaman melakukan melakukan pemesanan tiket cepat.

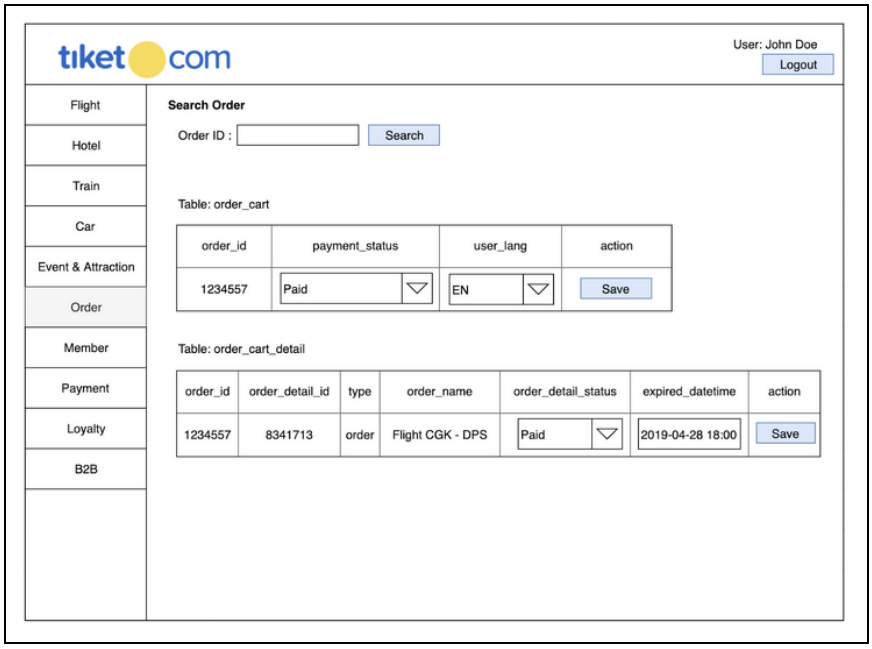
Pengecekan terhadap suatu pesanan melalui fitur *Check Order* ini memiliki kesamaan balikan data, dengan melakukan pengecekan melalui *My Order*. Melalui fitur *Check Order* dapat berpindah ke halaman *Order Group* jika pesanan yang ingin dilihat detailnya seperti pesanan pesawat atau kereta pulang pergi dan halaman *Order Detail* jika pesanan bersifat tunggal.

A.2 Dashboard Support

Dashboard Support merupakan penambahan salah satu fitur tertutup di Tiket.com yang digunakan untuk menfasilitasi seksi *IT Support* ketika dibutuhkan *customer service* untuk mengolah data yang dibutuhkan *customer*. *Dashboard Support* ini telah membantu banyak fitur dapat seperti untuk menampilkan dan mengubah data dalam suatu fitur.

Hal yang melatarbelakangi pembuatan *Dashboard Support* ini adalah ketika divisi *IT Support* melakukan pengambilan dan perubahan data dalam *database*, *IT Support* harus melakukan *query* langsung manual sesuai permintaan *customer* yang telah dilegalkan. *Query* manual ini tentu sangat berisiko seperti salah *query* yang disebabkan human error hingga dapat mengakibatkan dampak kerugian langsung ke *database* dan rentan terjadi *bug* pada fungsionalitas yang terhubung.

Dashboard Support yang dikembangkan untuk mendukung fitur *My Order* ini mampu menampilkan data pesanan berdasarkan id pesanan yang telah diinput. Selain menampilkan data pesanan, *Dashboard Support My Order* ini dapat merubah data, seperti status pembayaran, bahasa *user*, status pesanan detail, dan mengubah tanggal kadaluarsa pesanan. Segala perubahan harus dari permintaan *customer* yang telah dikonfirmasi permintaannya. Gambar rancangan antarmuka dari *Dashboard Support* ditunjuk dalam Gambar 3.4.



Gambar 3.4 Rancangan Antarmuka *Dashboard Support My Order*

Masing-masing data yang dapat diganti sesuai rancangan antarmuka yang ditunjukkan Gambar 3.4, seperti status pesanan, bahasa *user,* status detail pesanan, dan tanggal kadaluarsa. Status pembayaran memiliki pilihan status seperti sedang di *shooping cart, discarded*, *abandoned*, *checkout*, *partially paid*¸ *paid*, dan *cancelled.* Bahasa *user* memiliki pilihan bahasa Inggris (*en)* dan Indonesia (id). Status detail pesanan memiliki status pilihan status seperti *active, recheck, deleted, fraud, refund, refunded, abandoned,* dan *missing.* Perubahan pada tanggal kadaluarsa digunakan untuk memperpanjang durasi masa pembayaran *user* terhadap pesanan.

A.3 Tracking User

Tracking User merupakan fitur yang melakukan pengumpulan data *user* dan menghitung interaksi *user* terhadap fungsi di dalam antarmuka sehingga dapat diketahui jumlah *user* yang menggunakan suatu fungsi di durasi tertentu. *Tracking User* ini adalah penambahan fitur tertutup Tiket.com digunakan untuk menfasilitasi tim *User Experience* agar dapat mengevaluasi dan improvisasi antarmuka. *Tracking User* ini menggunakan *Google Tag Manager* (GTM) yang hasil dari *Tracking* agar ditampilkan dalam *Google Analytics.*

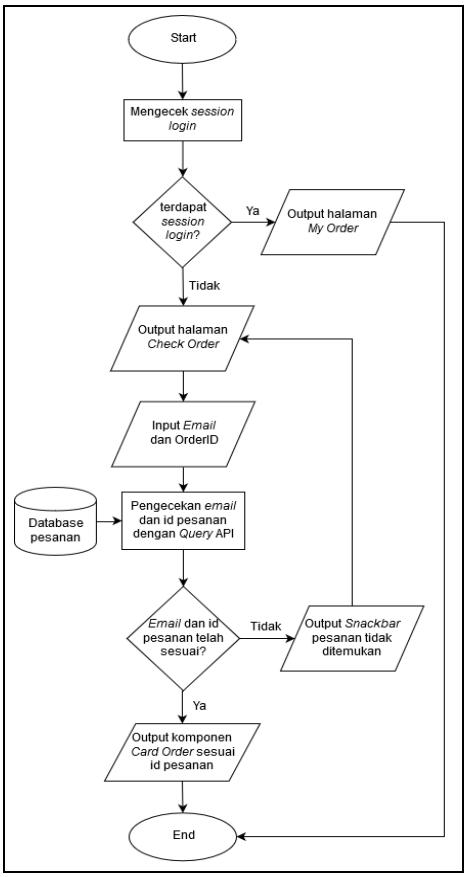
Pemasangan *Tracking* ini berdasarkan tiap *button* yang muncul di halaman, proses pengerjaan yang dilakukan seksi *Frontend.* Pengembangan membangun fitur ini adalah membuat *function* yang ter-*trigger* ketika suatu tombol di *click.* Jadi disini tiap-tiap *button* memiliki *function* sendiri. Pembagian tugas pengembangan fitur ini adalah dengan membagi *story* berdasarkan tiap vertikal seperti pesawat, kereta api, sewa mobil, hotel, atraksi, dan *event.*

B. Flowchart

Flowchart merupakan suatu alur kerja yang direpresentasikan dalam bentuk seperti diagram. Terdapat dua *flowchart* yang menggambarkan interaksi *user* ke fitur, yaitu *flowchart Check Order* dan *Dashboard Support Order*.

B.1 Check Order

Flowchart dari fitur baru *check order* ini dimulai apabila tombol "Cek Order" atau *My Order* ditekan yang ada di halaman utama Tiket.com. Proses yang dilakukan pertama yaitu pengecekkan *session login* dengan bantuan *middleware*. Setelah itu jika tidak terdapat *session login* maka akan dimunculkan halaman *check order*. Detail *Flowchart* yang menjelaskan interaksi *user* terhadap halaman fitur adalah sebagai berikut.



Gambar 3.5 *Flowchart* Interaksi *User* Terhadap Fitur *Check Order*

Pada halaman *check order, user* hanya dapat melakukan *input email* dan id pesanan. Setelah *user* meng-*input* dengan benar maka akan menampilkan komponen *Card Order* berisi data yang sesuai dengan input. *Flowchart* mulai terjadi apabila *user* menekan tombol *"Check Order"* atau "*My Order"*. Pengecekan *email* dan id pesanan membutuhkan satu *service* yang melakukan *query* dan memberi data balikan dalam bentuk JSON.

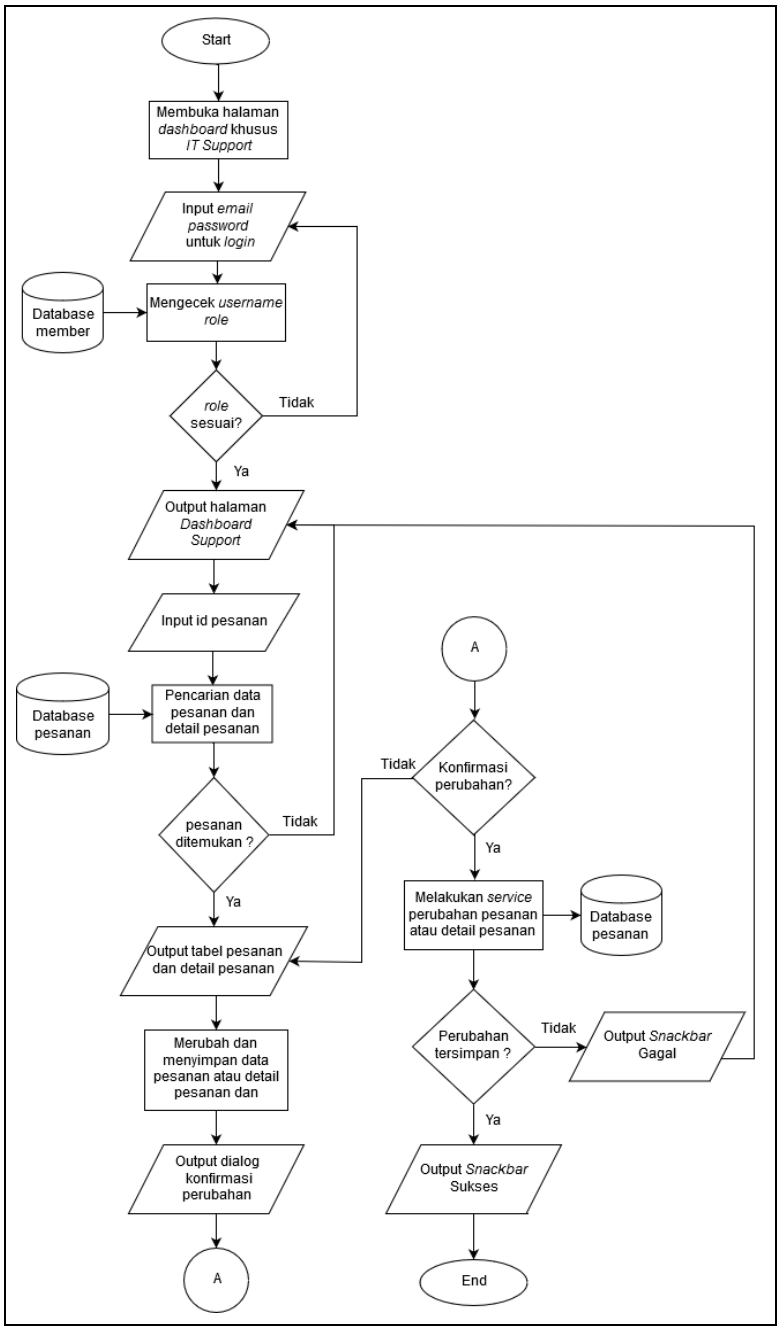
B.2 Dashboard Support

Dashboard Support pendukung fitur *My Order* dimulai ketika *login* telah dilakukan dan *role* dari akun memiliki *privilege* untuk membuka *dashboard support.* Setelah masuk ke *dashboard support My Order, user* dapat melakukan pencarian pesanan berdasarkan id pesanan. Jika muncul hasil pencarian pesanan maka tabel pesanan dan tabel detail pesanan di halaman *Dashboard* terisi sesuai data pesanan.

Pada halaman yang telah terisi *field* dari data id pesanan, dapat dilakukan penyimpanan terhadap *field* yang telah diubah. Apabila diklik tombol save maka akan muncul *pop up* dialog berupa konfirmasi atas perubahan data yang telah diubah. Jika perubahan telah dikonfirmasi dan *service* berjalan sesuai perintah serta *server* telah merubah data yang diinginkan maka akan muncul *snackbar* di atas halaman menyatakan perubahan sukses dilakukan, jika tidak maka *snackbar* akan bertuliskan "You are offline!".

Pengembangan *dashboard support My Order* ini dibangun dengan tiga *service,* yaitu melakukan pencarian data pesanan berdasarkan id pesanan, melakukan perubahan data pada tabel pesanan, dan melakukan perubahan data pada tabel detail pesanan. Detail *Flowchart* yang menjelaskan interaksi *user*

terhadap halaman *Dashboard Support My Order* adalah sebagai berikut.



Gambar 3.6 *Flowchart* Interaksi *User* Terhadap Halaman *Dashboard Support*

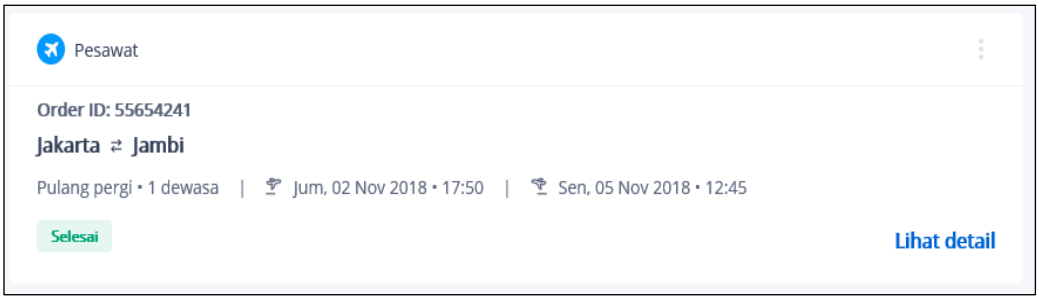
C. Implementasi

Terdapat tiga implementasi yang dipaparkan, yaitu implementasi pada fitur *Check Order, Dashboard Support My Order,* dan *Tracking User.* Pada halaman *Check Order* implementasi berupa halaman *form input* untuk memasukkan *email*

dan id pesanan. Pada *Dashboard Support My Order* terdapat tampilan antarmuka yang telah dikembangkan mengikuti hasil *wireframe dashboard My Order* hasil tim *UX designer* buat. Pada *Tracking* implementasi hanya dapat ditampilkan dalam bentuk diagram garis yang dihasilkan dari *Google Analytics* dengan jumlah *tracking* yang dirahasiakan perusahaan terhadap beberapa 2 jenis tombol.

C.1 Check Order

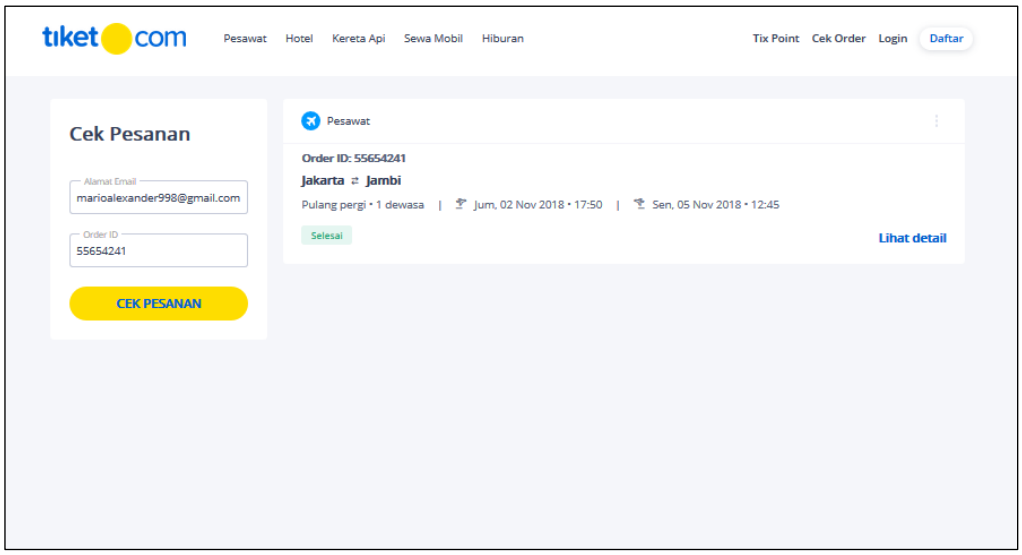
Pada halaman *check order* Tiket.com yang dapat diakses di tiket.com/check-order, user dapat melakukan pencarian pesanan dengan menggunakan id pesanan dan *email* .Hasil dari pencarian akan ditampilkan dalam komponen *card order* yang digunakan juga di fitur halaman *My Order*. Contoh dari komponen card order ditunjuk pada Gambar 3.7.



Gambar 3.7 Komponen *Card Order*

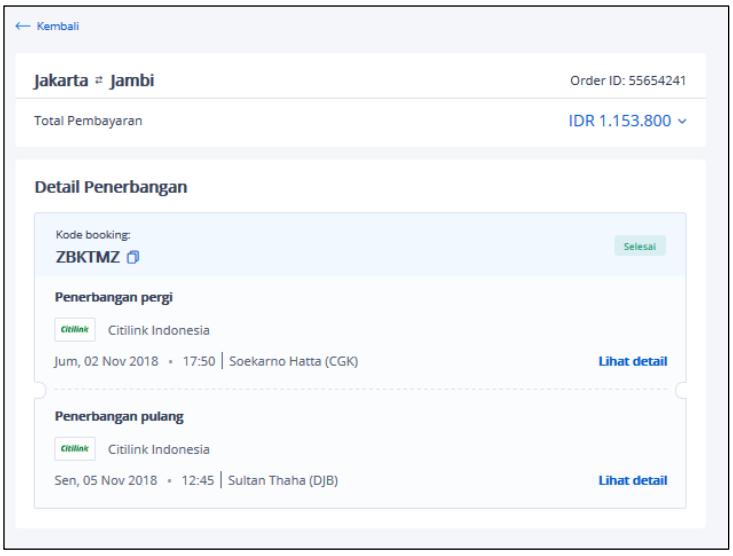
Komponen *card order* ini memiliki tombol elipsis yang jika diklik mampu menampilkan pilihan untuk ,seperti bukti pembayaran, membagikan *etiket*, dan menghubungi *customer service.*

Fitur.*check order* muncul dengan latar belakang ingin memberi layanan kepada *user* melakukan pengecekan pesanan tanpa *login* dan *register* akun. Tampilan dari fitur *check order* yang telah dikembangkan ditunjuk dalam Gambar 3.8.



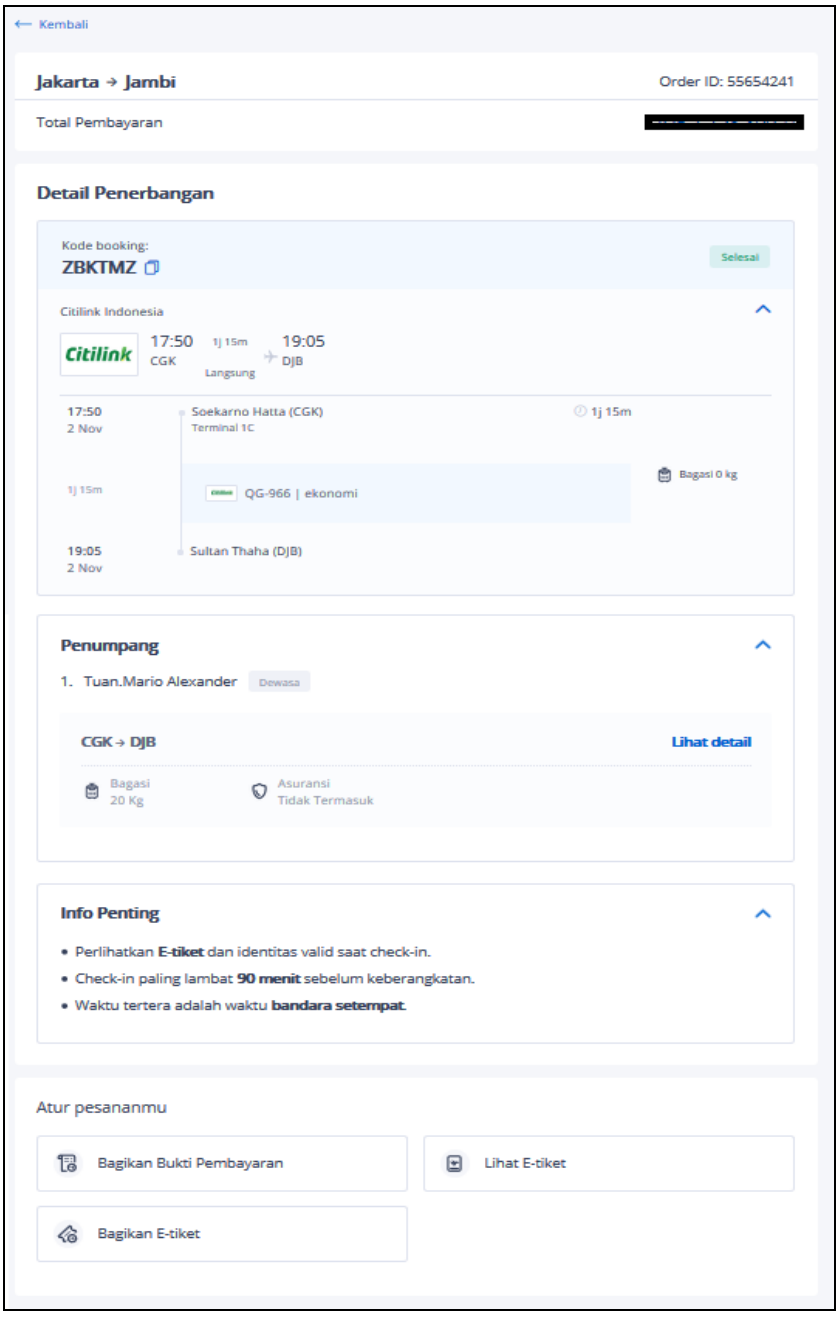
Gambar 3.8 Implementasi Fitur *Check Order*

Komponen c*ard order* dapat terdapat di halaman *check order* mengarahkan ke halaman lain seperti *order detail* atau *order group* jika diklik tombol "Lihat detail". Halaman *order group* akan terbuka jika satu pesanan menampung tiket pesawat atau kereta api bersifat pulang pergi. Peralihan halaman dibutuhkan parameter id pesanan, tipe pesanan, dan *hash* pesanan. dari pesanan yang dapat dilihat di *url browser*. Contoh informasi yang ditampilkan di halaman *Order Group* ditunjuk dalam Gambar 3.9



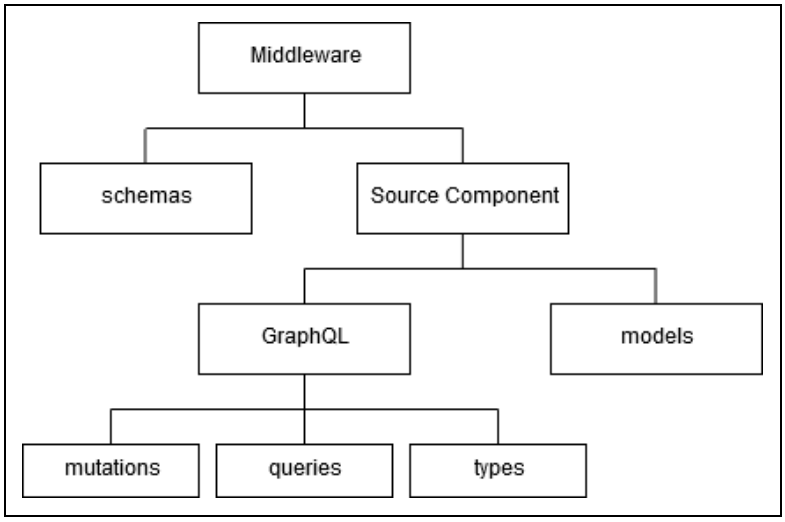
Gambar 3.9 *Screenshot* Contoh Informasi di Grup Pesanan

Halaman detail ini terdapat dapat dibuka untuk seluruh jenis pesanan. Halaman ini bertujuan menampilkan informasi tambahan yang tidak ditunjukan dalam komponen c*ard order* contohnya data penumpang dan fasilitas bagasi. Contoh informasi yang ditampilkan di halaman *order detail* ditunjuk dalam Gambar 3.10.



Gambar 3.10 *Screenshot* Contoh Informasi di Detail Pesanan

Pengimplementasian fitur *check order* ini membutuhkan satu *service* ketika *input email* dan id pesanan telah valid dan tombol "Cek Pesanan" telah diklik. *Semua service My Order* yang ada berada repositori *query* API *My Order*. Setiap *service* yang dibentuk terdapat beberapa komponen yang menyusun *service* yang dilakukan. Struktur dari folder repositori *query* API yang menyimpan *service* yang telah dibuat ditunjuk dalam Gambar 3.11.



Gambar 3.11 Struktur Pembentuk *Service* di Repositori *Query* API

Pembuatan suatu *service* membutuhkan komponen pembangun seperti, *query*/*mutation, type, model,* dan *schema*. Penjelasan fungsi masing-masing komponen adalah sebagai berikut.

- Query dan Mutation

Query merupakan komponen fungsi pembentuk *service* yang berisi konfigurasi proses *read* atau *fetch* data. Sedangkan *mutation* digunakan untuk melakukan *write*, *update,* dan *delete* data.

- Types

Type merupakan komponen pembentuk *Schema* yang berisi susunan tipe data yang dibutuhkan ketika melakukan *query* atau mutation dan mengembalikan

suatu objek.

- Models

Model merupakan komponen berisi konfigurasi seperti *method* yang digunakan untuk melakukan *service*, data parameter yang terlibat *query* atau *mutation*, dan konfigurasi akses *endpoint server*.

- Schema

Schema merupakan komponen yang menampung seluruh tipe data yang ada di seluruh *service*. Manfaatnya dapat memberi batasan pada *client side* ketika melakukan *query* atau *mutation*.

Hasil dari pembuatan *service* adalah suatu balikan dari *server* yang memiliki struktur JSON. Contoh balikan dari *server* untuk *service* fitur *check order* ditunjuk dalam Gambar 3.12.


```
{
 "code": "SUCCESS",
 "message": "SUCCESS",
 "errors": null,
 "data": {
   "content": [
     {
       "orderId": 554429351,
       "accountId": 21699926,
       "orderType": "FLIGHT",
       "orderHash": "614ed62ab9923793e2a8d27c57644b862da2efe3",
       "orderDate": "2019-09-13 12:17:30",
       "expiredDate": "2019-09-13 14:36:29",
       "orderGroup": true,
       "orderCar": null,
       "orderHotel": null,
       "orderTrain": null,
       "orderFlight": {
         "departureCity": "Yogyakarta",
         "departureCityCode": "JOG",
         "departureDate": "2019-10-21",
         "departureTime": "06:00",
         "departureTimezone": 7,
         "departureOrderDetailId": 89601591,
```


Gambar 3.12 *Screenshot* Contoh Balikan *Service Check Order*

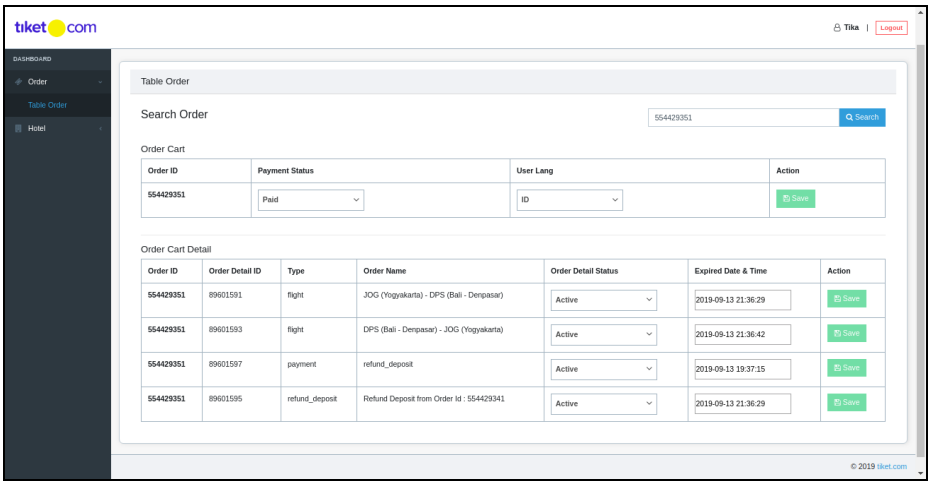
```
        "arrivalCity": "Denpasar-Bali",
        "arrivalCityCode": "DPS",
        "returnDate": "2019-10-24",
        "returnTime": "09:30",
        "returnTimezone": 8,
        "returnOrderDetailId": 89601593,
        "adultTotal": 1,
        "childTotal": 0,
        "infantTotal": 0,
        "roundTrip": true,
        "smartRoundTrip": false,
        "totalAmount": 0,
        "customerCurrency": "IDR"
      },
      "orderEvent": null,
      "orderStatus": "EXPIRED",
      "refundOrders": null,
      "rescheduleOrder": null,
      "expiredCountDown": null,
      "paymentUrl": null,
      "paymentTitle": null,
      "eticketUrl": null,
      "receiptUrl": null,
      "additionalInformation": "",
      "shareEticketList": [],
      "shareReceipt": null,
      "enableCancelInsurance": false,
      "deletable": true,
      "cancellable": false,
      "refundable": false,
      "reschedulable": false,
      "continuePayment": false,
      "enableOnlineCheckin": false,
      "enableShareEticket": false,
      "enableShareReceipt": false
    }
  ],
  "last": true,
  "totalPages": 1,
  "totalElements": 1,
  "first": true,
  "numberOfElements": 1,
  "sort": null,
  "size": 0,
  "number": 0
 },
 "serverTime": 1568888103899
}
```


Gambar 3.12 *Screenshot* Contoh Balikan *Service Check Order* (Lanjutan)

C.2 Dashboard Support

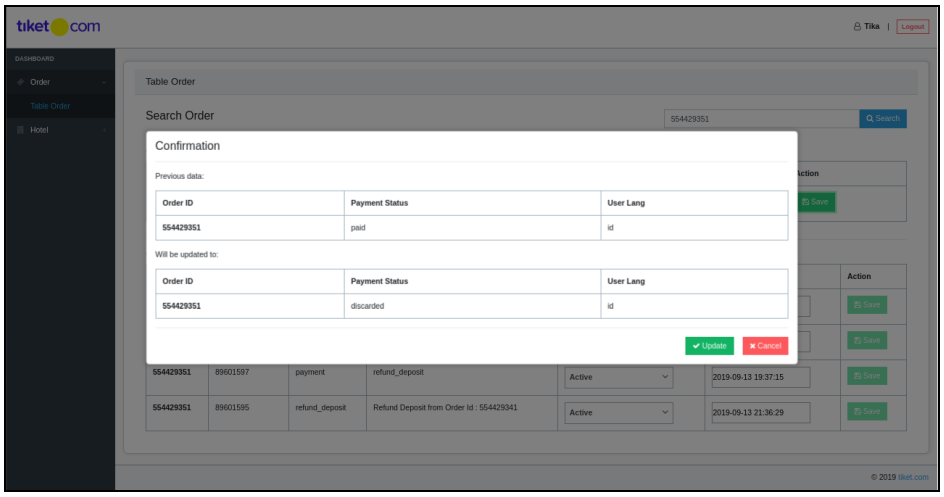
Dashboard Support dapat diakses dengan melakukan *login* akun yang

memiliki *role* sesuai. Setelah melakukan login dan masuk ke halaman d*ashboard support*, *user* dapat mencari pesanan dengan menginput id pesanan. Tampilan halaman *dashboard support* yang telah dikembangkan adalah sebagai berikut.



Gambar 3.13 Implementasi *Dashboard Support* Untuk *My Order*

Tampilan menampilkan tombol "*save*" pada suatu *record* dalam tabel berstatus *disable* selama tidak terjadi perubahan dalam tabel. Tombol "*save*" dapat diklik ketika terdapat perubahan dalam suatu *field*. Apabila tombol "*save*" diklik akan muncul dialog *popup* berisi konfirmasi perubahan. Dialog *popup* ditunjuk dalam Gambar 3.14.



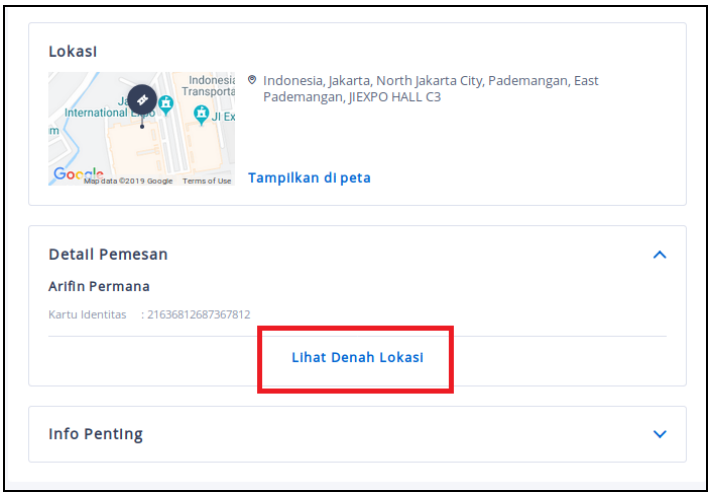
Gambar 3.14 Dialog *Popup* yang Konfirmasi Perubahan

C.3 Tracking *User*

Dalam proses pengembangan *tracking user*, dibutuhkan suatu fungsi yang melakukan *push* suatu objek ke data *layer* dan objek GTM. Terdapat banyak tombol dalam halaman *My Order* Tiket.com yang digunakan di beberapa halaman lain, sehingga untuk mengimprovisasi *tracking* dibutuhkan data *layer* dan objek GTM sebagai penanda asal *tracking* berdasarkan halaman dan tipe pesanan. Prosesnya, setelah komponen diklik maka akan dilakukan *push* ke data *layer* untuk mengetahui jenis halaman dan mengirim objek GTM untuk dapat memberi informasi tipe pesanan. Terdapat dua jenis komponen tombol yang diimplementasikan, yaitu tombol "Lihat Denah Lokasi" atau "View Venue Layout" dan tombol salin kode *booking* untuk tipe pesanan *event.*dan *attraction.*

C.3.1 Lihat Denah Lokasi

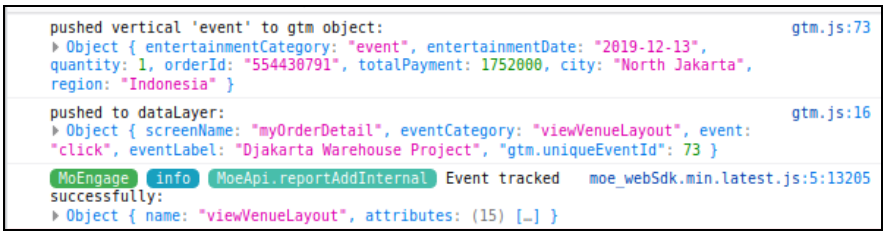
Tombol "Lihat Denah Lokasi" muncul untuk memungkinkan *user* untuk dapat lokasi denah detail dari *event*. Tampilan antarmuka yang menampilkan tombol "Lihat Denah Lokasi" ditunjuk dalam Gambar 3.15.



Gambar 3.15 Tombol "Lihat Denah Lokasi" di Pesanan *Event*

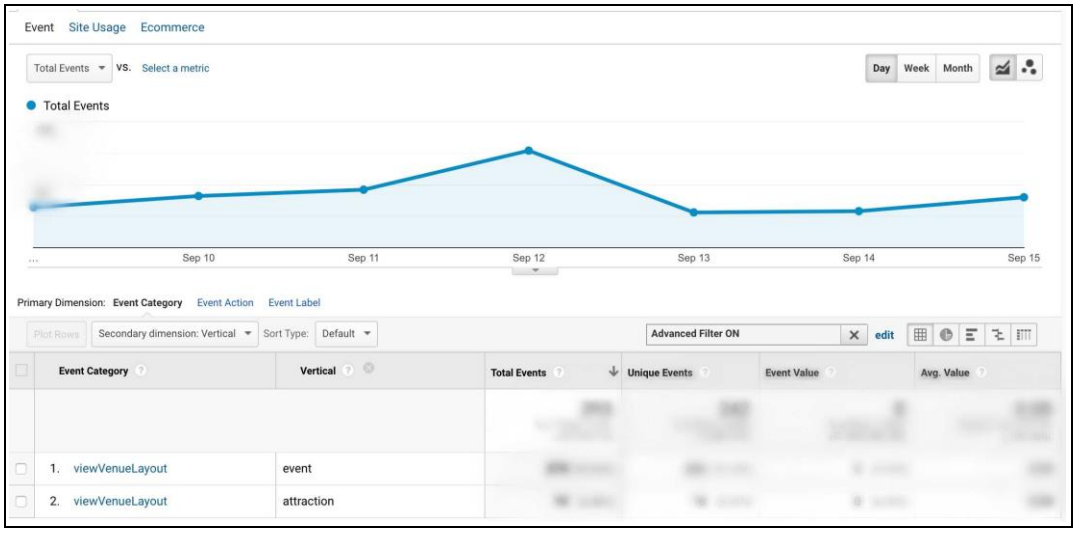
Tombol "Lihat Denah Lokasi" ini memiliki fungsi untuk menampilkan *popup* berupa denah lokasi *event* yang sesuai. Hasil dari data yang di-*push* dapat

dilihat melalui *console* yang ada di *browser*. Contoh GTM *object* dan data *layer* yang telah di *push* ditunjuk dalam Gambar 3.16.



Gambar 3.16 Objek GTM dan *Data Layer* Dari Tombol "Lihat Denah Lokasi"

Komponen tombol "Lihat Denah Lokasi" digunakan di dua jenis vertikal layanan pesanan, yaitu *event* dan *attraction* dan komponen ini hanya dapat ditemukan di halaman detail pesanan. Jumlah data kuantitatif yang telah didapatkan dari *tracking* dirahasiakan perusahaan namun hasil dapat dilampirkan tanpa menampilkan nilai kuantitatif. Hasil implementasi dari tombol "Lihat Denah Lokasi" ditampilkan menggunakan bantuan *Google Analytics* seperti yang ditunjuk Gambar 3.17.

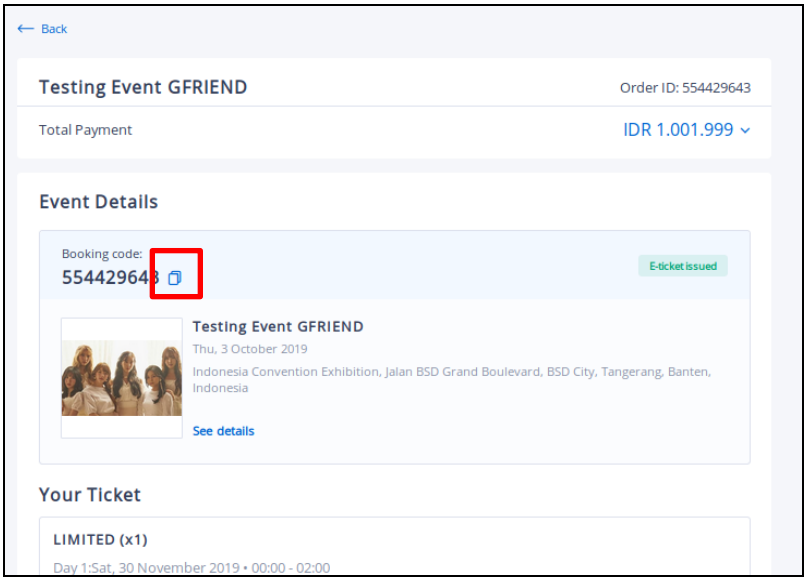


Gambar 3.17 Implementasi Tracking Tombol "Lihat Denah Lokasi"

C.3.2 Salin Kode Booking

Tombol salin kode *booking* muncul untuk memungkinkan *user* menyalin kode booking dengan sekali klik, maka kode *booking* akan tersimpan di *clipboard*

dan siap di-*paste* untuk dibagikan ke media lain. Tampilan antarmuka yang menampilkan salin kode *booking* ditunjuk dalam Gambar 3.18.

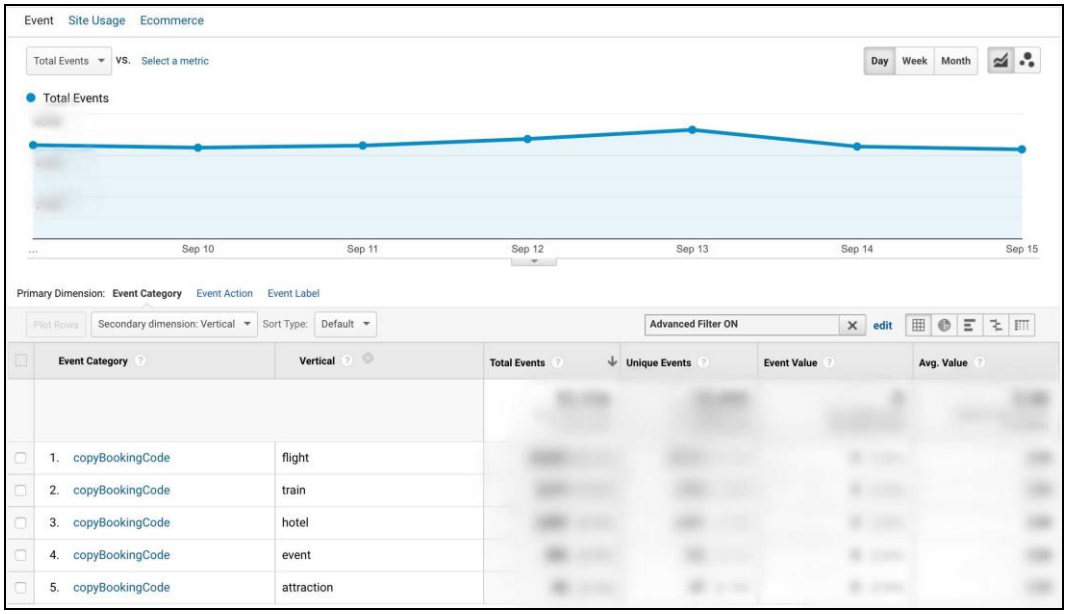


Gambar 3.18 Tombol Salin Kode *Booking* di Halaman Detail Pesanan

Tombol salin kode *booking* dapat ditemukan di seluruh vertikal jenis pesanan yang ada di Tiket.com pada halaman detail pesanan. Hasil dari data yang di-*push* dapat dilihat melalui *console* yang ada di *browser*. Contoh GTM *object* yang dikirim dan data *layer* yang telah di *push* ditunjuk dalam Gambar 3.19.

Gambar 3.19 Objek GTM dan *Data Layer* Dari Tombol Salin Kode *Booking*

Komponen tombol salin kode *booking* ini ditemui di seluruh jenis vertikal pesanan yang ada di halaman detail pesanan. Jumlah data kuantitatif yang telah didapatkan dari *tracking* untuk komponen ini dirahasiakan perusahaan namun hasil dapat dilampirkan tanpa menampilkan nilai kuantitatif. Hasil implementasi dari tombol salin kode *booking* ditampilkan menggunakan bantuan *Google Analytics* yang ditunjuk Gambar 3.20.



Gambar 3.20 Implementasi Tracking Tombol "Lihat Denah Lokasi"

D. Testing

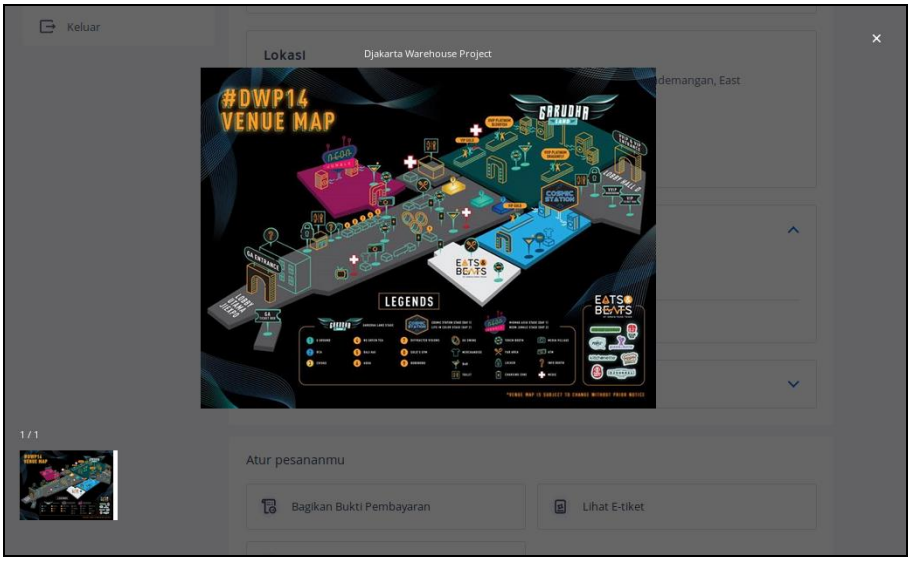
Testing merupakan proses yang dibutuhkan dalam pengembangan proyek, yang bertujuan untuk mengidentifikasikan ketidaksesuaian hasil sebuah keluaran dari sistem yang diharapkan. Testing yang dilakukan seksi *Quality Assurance smoke test* dan *performance test* sedangkan yang dilakukan *engineer* adalah *unit testing*. Testing di PT Global Tiket Network dibagi menjadi tiga tahap berdasarkan lingkup *server*, yaitu *local, staging,* dan *production.*

D.1 Local

Server local dimiliki masing-masing *engineer* ketika mengerjakan suatu *story* yang dikembangkan, prosesnya yang dilakukan setelah menyelesaikan tahap *coding* adalah *testing* manual langsung dari *engineer*. Pada tahap ini, tugas seksi *engineer* adalah melakukan *unit testing*. *Unit testing* adalah *testing* yang dilakukan *engineer* dengan menguji suatu unit atau komponen pembangun untuk memastikan hasil keluaran suatu *unit* sesuai. *Unit testing* repositori *client-side* di Tiket.com ini menggunakan Node JEST dan yang diuji adalah tiap komponen

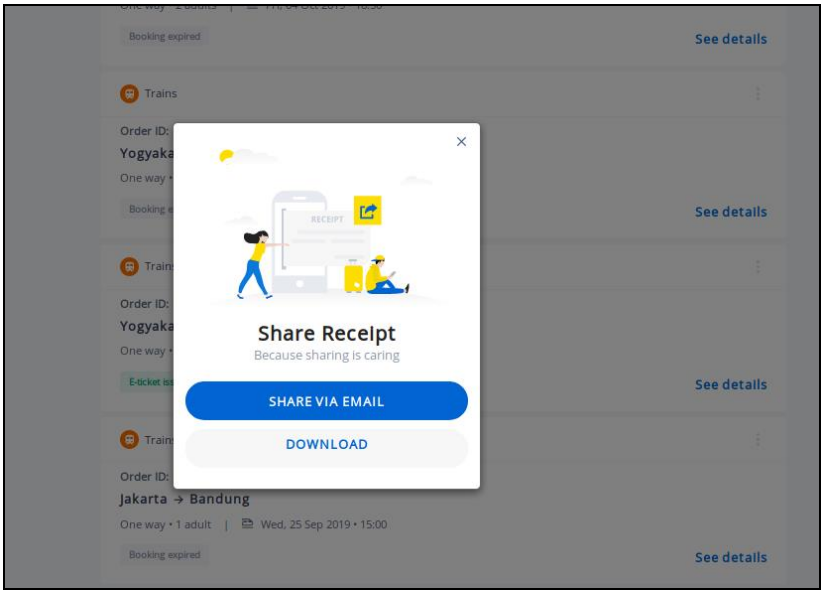
antarmuka dengan membuat *case* segala kondisi yang memunculkan antarmuka telah di-*cover*. Hasil dari *unit testing* menunjukkan nilai bilangan antara 0 – 100. Semakin tinggi semakin menandakan suksesnya *unit testing.*

Terdapat dua contoh komponen yang telah dilakukan *unit testing*, yaitu komponen *gallery.desktop* dan komponen s*hare receipt.* Komponen *gallery.desktop* muncul apabila tombol "Lihat Denah Lokasi" di klik dalam bentuk *popup*. Contoh tampilan komponen *gallery.desktop* ditunjuk dalam Gambar 3.21.



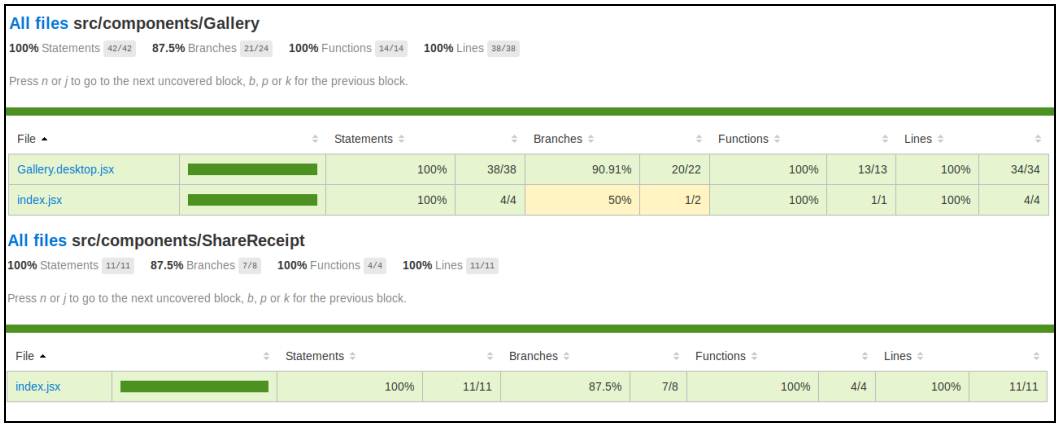
Gambar 3.21 *Screenshot* Contoh *Popup* Komponen *Gallery.desktop*

Sedangkan komponen *share receipt* merupakan komponen *popup* yang muncul ketika tombol menu di *card order* diklik tombol "Bagikan Bukti Pembayaran". Tampilan komponen *share receipt* ditunjuk dalam Gambar 3.22.



Gambar 3.22 *Screenshot* Tampilan *Popup Share Receipt*

Hasil yang telah di-*generate* Node JEST ditampilkan dalam bentuk daftar tiap komponen yang berada di *file* coverage/index.html. *Screenshot* dari hasil yang ditampilkan *coverage* ditunjukan dalam Gambar 3. 23.



Gambar 3.23 *Screenshot* Coverage Komponen *Gallery* dan *Share Receipt*

D.2 Staging

Pada lingkup *server* ini *quality assurance* melakukan *testing* berdasarkan tiap *story* yang statusnya telah dilakukan *code review* dan telah di-*merge* ke *server staging*. Jika pada tahap ini *testing* yang dilakukan sesuai yang diharapkan secara manual maka *story* akan masuk ke status *done*. Selain berdasarkan *story* pada

tahap *staging* ini, *quality assurance* juga bertanggung jawab membuat *test case* dan melakukan *regression test. Regression test* merupakan proses *testing software* yang sudah ada untuk memastikan bahwa perubahan dan penambahan suatu hasil *story* baru di dalam proyek tidak merusak fungsi-fungsi yang telah ada sebelumnya.

Tahap berikutnya setelah setiap *story* dalam *sprint* telah *done*, dibuatlah suatu *list* yang dibuat perwakilan *engineer* berisi daftar fungsi-fungsi baru yang akan di-*deploy* suatu *sprint* lalu di-*approve* oleh *quality assurance* dan *engineer lead*. Setelah semua *list* telah di-*approve* divisi *infra* melakukan *deployment*.

D.3 Production

Production merupakan *server* tahap akhir yang telah digunakan *customer, testing* yang dilakukan pada *server* ini adalah *smoke test. Smoke test* merupakan *testing* yang telah di *build* atau berada di *server production* untuk memastikan fungsi-fungsi bekerja sesuai seperti *server staging*. *Smoke test* dilakukan setelah proses *deployment* selesai. Jika pada tahap *smoke test* ini ditemukan *bug* atau *defect* dalam sistem, tim *infra* melakukan *rollback server production*.

3.3.2 Kendala yang Ditemukan

Dalam proses pelaksanaan kerja magang, terdapat beberapa kendala yang dihadapi. Kendala-kendala yang ditemukan adalah antara lain seperti berikut.

1. Proses modifikasi *client-side* dengan menggunakan *framework* React dan *Query* API dengan GraphQL yang belum pernah dipelajari. Proses pengembangan antarmuka *form* interaktif di halaman *check order* dan dialog *popup dashboard support* menggunakan SASS yang belum pernah dipelajari.
2. Terdapat hambatan dalam pemilihan variabel yang akan dimasukkan ke data *layer* dan objek vertikal untuk di-*push* ketika mengembangan fitur *tracking.*

3.3.3 Solusi atas Kendala yang Ditemukan

Dari kendala-kendala yang ditemukan selama proses kerja magang, solusi dari kendala tersebut harus secepatnya ditemukan agar tidak menambah waktu pengerjaan dari tugas. Solusi dari kendala-kendala yang ditemukan adalah antara lain sebagai berikut.

1. Mempelajari *framework* dengan cara mengikuti tutorial video di Youtube, membaca dokumentasi dari *framework* langsung atau dapat membaca dokumentasi Stack Overflow serta bertanya kepada pembimbing lapangan mengenai masalah yang ditemukan jika masih tidak mengerti setelah mencari berbagai sumber.
2. Melakukan konsultasi kepada *product manager* mengenai kesesuaian hasil dari data *layer* dan objek vertikal yang di-*push*.