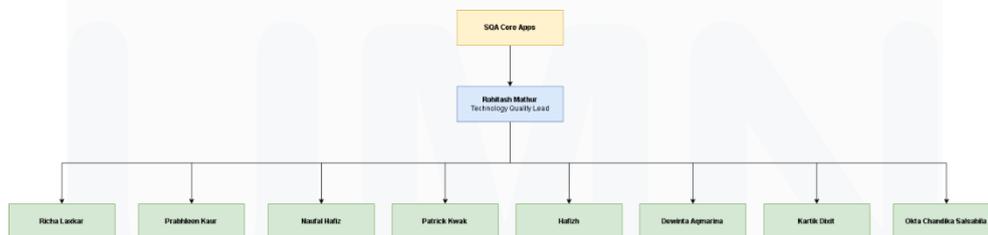


BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Menduduki dan menjabat sebagai *Software Quality Assurance Intern* di perusahaan tiket.com yang ditempatkan didalam divisi SQA Core Apps dimana divisi tersebut lebih fokus terhadap hal-hal teknis. Tugas-tugas yang diberikan sendiri lebih banyak menggunakan *software* seperti Charles Proxy untuk mereproduksi *error* dengan sengaja didalam aplikasi menggunakan API yang dipanggil oleh *software* tersebut. Tugas lainnya adalah untuk memastikan dan menjadi jembatan komunikasi antara *developer* dan *designer* sebuah fitur baru atau *button* baru untuk memastikan apakah yang dibuat oleh *developer* sudah sesuai dengan harapan dan desain dari sisi *designer* dari segi bentuk, warna, lebar, dan spesifikasi lainnya. Sebagai SQA, bekerja didalam divisi SQA Core Apps dengan 9 rekan lainnya yang terdiri dari 3 *intern*, 5 pegawai tetap, dan 1 pengawas atau yang disebut *Technology Quality Lead* yang akan memandu kinerja divisi. Berikut adalah gambaran anggota divisi SQA Core Apps.



Gambar 3.1 Anggota Divisi SQA Core Apps

Gambar 3.1 merupakan divisi SQA Core yang dipimpin oleh Rohitash Mathur yang menjabat sebagai *Technology Quality Lead*. Setiap dua minggu sekali, beliau akan membagi tugas untuk setiap anggotanya yang disebut sebagai *sprint plan*. Pembagian tugas akan dibagikan melalui Microsoft Excel Sheet yang berisi data-data mengenai pembagian tugas, setelah pembagian tugas, setiap anggota diharuskan untuk membuat *subtask* didalam JIRA ID tersebut yang didalamnya berisi proses

pengerjaan *testing* dari QA. *Subtask* tersebut diharuskan berisi *steps to*

reproduce, environment, testcases yang dibuat dengan TestRail, *QA end date, QA effort, QA start date*. Data-data tersebut digunakan untuk memastikan bahwa *testcases* yang dilakukan oleh QA sudah sesuai dengan harapan *developer* untuk meminimalisir adanya *bug* serta *QA end date, effort, dan start date* digunakan untuk memberitahu *developer* berapa banyak hari yang akan digunakan dan dipakai untuk melakukan *testing* pada JIRA ID tersebut. Komunikasi antara *developer* dengan QA dilakukan sesuai kesepakatan berdua seperti menggunakan Google Meet atau Slack karena banyak *developer* yang melakukan *work from home* serta banyak pegawai yang bukan merupakan warga negara Indonesia. Berikut adalah gambaran *workflow* dari QA divisi SQA Core Apps.

3.2 Tugas dan Uraian Kerja Magang

Dilakukan pekerjaan tugas bersifat P0 yaitu tugas dengan prioritas tertinggi yaitu *Network Error Handling* dan *Common Error Handling*. *Network Error Handling* ditujukan untuk membuat *error screen* yang lebih *user friendly* serta untuk memudahkan *developer* mengetahui apa penyebab terjadinya *error* ini karena memiliki kodenya masing-masing dari kode 400 hingga 500 dan *exceptions*. *Common error handling* ditujukan untuk mengganti komponen yang digunakan untuk menampilkan *error screen* tersebut kedalam komponen yang lebih baru namun harus dipastikan bahwa perilaku setiap *error screen* sama meskipun menggunakan komponen baru sehingga harus dilakukan pengecekan pada setiap api yang dapat menghasilkan *error screen* di setiap *vertical* yaitu *hotel, flights, TTD, ground transport, dan payment*.

3.2.1 Tugas yang Dilakukan

Tabel dibawah merupakan informasi mengenai tugas apa saja yang dilakukan selama masa pelaksanaan kerja magang di perusahaan *tiket.com*.

Tabel 2. Tugas Kerja Magang

No	Minggu	Proyek	Keterangan
1	27 Juni 2022 – 15 Juli 2022	<i>User Acceptance Test</i> dan <i>Training</i>	Melakukan UAT terhadap aplikasi untuk mendapatkan wawasan mengenai aplikasi serta untuk mencari <i>bug</i> yang mungkin terdapat pada aplikasi <i>live</i> .
2	13 Juli 2022 – 15 Juli 2022	<i>Training</i>	Menonton video dan slide yang telah diberikan untuk menambah wawasan untuk apa saja yang akan dilakukan
3	18 Juli 2022 – 17 Oktober 2022	<i>Network error handling</i>	Mengganti <i>error screen</i> yang lama dengan yang baru yang lebih <i>user friendly</i> dari kode 400 hingga 500. Memudahkan developer untuk mengetahui apa penyebabnya <i>error</i> karena diberikan kode dan memiliki informasi yang lebih jelas untuk <i>user</i> mengenai <i>error</i> yang terjadi.

No	Minggu	Proyek	Keterangan
4	18 Oktober 2022 – 2 November 2022	<i>Common error handling</i>	Mengganti komponen <i>error screen</i> dengan komponen yang baru namun harus dipastikan bahwa <i>behaviour</i> setiap <i>error</i> harus sama dengan yang versi sebelumnya.
5	3 November 2022 – 4 November 2022	<i>Design task</i>	Memverifikasi bug yang telah dibenarkan oleh developer, mengenai container dari sebuah label yang tidak berubah sesuai bentuk dan jumlah tulisan didalamnya sehingga tulisan tidak terlihat.

3.2.2 Uraian Kerja Magang

1. User Acceptance Test dan Onboarding Session

Pada hari pertama praktik kerja magang, diberikan tugas yang akan dilakukan selama kurang lebih 2 minggu, yaitu untuk melakukan *user acceptance test* pada aplikasi yang telah dirilis di Google Play Store. Tugas ini ditujukan untuk memperoleh informasi-informasi yang terdapat didalam aplikasi, memperoleh gambaran besar seperti apa alur kerja aplikasi tiket. Karena sebagai QA salah satu aspek terpenting adalah untuk mengetahui alur kerja yang benar, maka tugas ini diberikan pada tahapan pertama sebagai salah satu bentuk *training*. Jika *bug* ditemukan, maka dapat mengumpulkan informasi-informasi tersebut didalam sebuah *spreadsheet* yang kemudian akan dipresentasikan kepada *supervisor* setiap harinya melalui *stand up daily meeting* untuk melaporkan progres UAT.

2. Training

Setelah selesai dilakukan tugas pertama yaitu UAT, maka *supervisor* akan mengirimkan Google Drive yang berisi mengenai Powerpoint dan video mengenai ilmu-ilmu *testing* yang akan digunakan didalam masa praktik kerja magang, diberi waktu 2 hari untuk mempelajari dari video-video tersebut. Setelah *training* dilakukan maka akan melanjutkan ke tugas selanjutnya.

3. Network Error Handling

Tugas pertama adalah untuk melakukan dan memegang proyek yang bernama Network Error handling dimana tugas QA di proyek ini adalah untuk memastikan bahwa *error screen* sudah sesuai dengan *expected result* di aplikasi yang telah disediakan *developer* dalam setiap *vertical*. Di samping tugas untuk memastikan layar, diharuskan juga mengecek apakah kode *error* yang di *log* didalam Charles Proxy dan Multihead search sudah sesuai dengan kode *error* yang ditampilkan di layar menggunakan bantuan beberapa *software*. Salah satu *software* yang digunakan pada tahapan awal adalah JIRA.

Subtasks		100% Done
WCT-2620	Show correct Network Error message for getDownloadFileAsync API	DONE
WCT-2620	Show correct Network Error message for cancelOrder API	DONE
WCT-2620	Show correct Network Error message for deleteOrder API	DONE
WCT-2624	Show correct Network Error message for getDownloadFileAsync(DownloadApiService) API	DONE
WCT-2625	Show correct Network Error message for getExtraBenefitInfo API	DONE
WCT-2627	Show correct Network Error message for getMyOrderDetailAirportTrain API	DONE
WCT-2628	Show correct Network Error message for getOrderDetailAirportTransfer API	DONE
WCT-2629	Show correct Network Error message for getMyOrderDetailCarAsync API	DONE
WCT-2630	Show correct Network Error message for getMyOrderDetailTrainAsync API	DONE
WCT-2631	Show correct Network Error message for getOrderDetailFlight API	DONE
WCT-2632	Show correct Network Error message for getFlightGroup API	DONE
WCT-2633	Show correct Network Error message for getMyOrderGroupAirportTrain API	DONE
WCT-2635	Show correct Network Error message for getMyOrderGroupTrainAsync API	DONE
WCT-2636	Show correct Network Error message for getMyOrderEventDetailAsync API	DONE
WCT-2637	Show correct Network Error message for getMyOrderListNonLogin API	DONE
WCT-2638	Show correct Network Error message for getPaymentDetailAsync API	DONE
WCT-2640	Show correct Network Error message for DownloadApiService(getReceipt) API	DONE
WCT-2641	Show correct Network Error message for getTravelDocuments API	DONE
WCT-2642	Show correct Network Error message for getMyHistoryOrderList API	DONE
WCT-2643	Show correct Network Error message for deleteOrder API	DONE
WCT-2644	Show correct Network Error message for getPastOrderList API	DONE
WCT-2645	Show correct Network Error message for getActiveOrderList API	DONE
WCT-2647	Show correct Network Error message for getPendingOrderList API	DONE
WCT-2648	Show correct Network Error message for getRefundList API	DONE
WCT-2650	Show correct Network Error message for getWaitingPaymentOrderList API	DONE
WCT-2652	Show correct Network Error message for getOngoingRefund API	DONE
WCT-2672	Change getError to getNetworkError in Remaining APIs where error is Not handled	DONE
WCT-2678	[QA] [Android] NEH - Left Over usage of getError() -> getNetworkError() - Part 1	DONE

Gambar 3.3 Informasi di dalam *Parent Task*

Setelah membuat *subtask*, tugasnya dimulai yaitu untuk melihat *JIRA parent task* seperti pada gambar 3.3 yang telah disediakan oleh *developer*, didalam *JIRA parent task* tersebut terdapat beberapa *subtask* lainnya sesuai dengan API yang akan dilakukan *testing*.

Show correct Network Error message for cancelOrder API

Attach Link issue Add Checklist Zeplin for Jira

Description
Please fix Network Error Handling for the following API to show Network Error message in case of Network/Internet errors

API Service: MyOrderApiService

API: ms-gateway/tix-my-order-core/manage-order/cancel-order

Error Component: ErrorBottomSheetDialogNonDragableFragment.newInstance

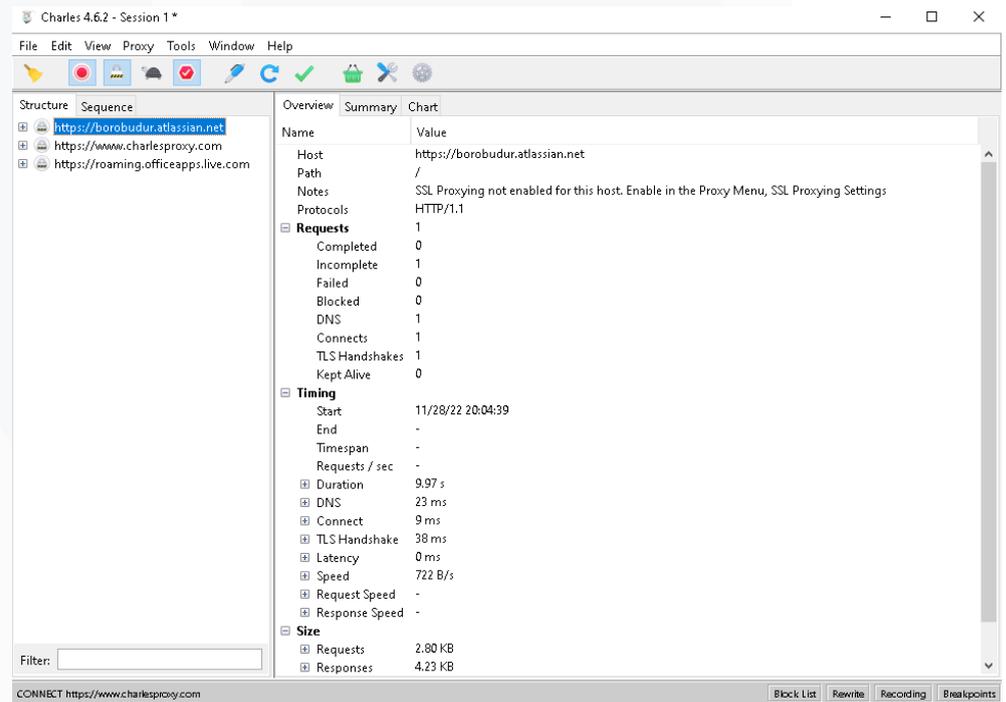
API trigger point: Your order → select order for which payment is not done then click ellipsize and cancel order then api will hit

PR Link: <https://github.com/biket/android/pull/7620> - Connect to preview

Comments
None

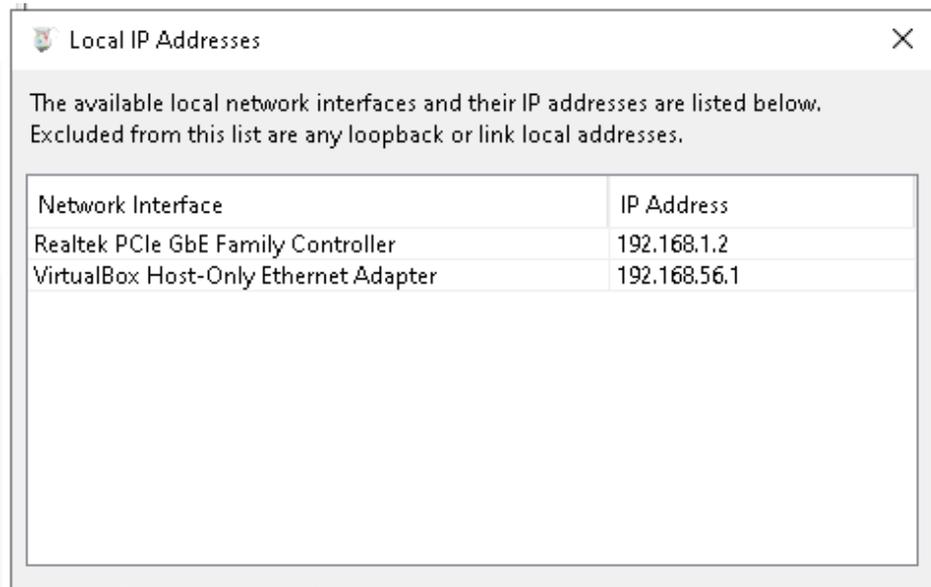
Gambar 3.4 Informasi di dalam *Subtask*

Gambar 3.4 memiliki informasi mengenai masing-masing *subtask* yang akan membahas cara mereproduksi *error* yang akan dites serta *expected result* yang telah disediakan oleh *developer* untuk sehingga dapat mereproduksi *error screen* yang akan dilakukan *testing*.



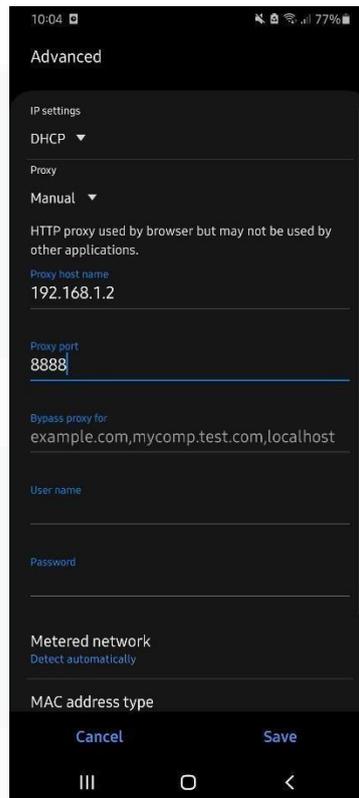
Gambar 3.5 Software Charles Proxy

Gambar 3.5 menunjukkan tampilan awal dari *software* Charles Proxy. Setelah melihat *subtask* untuk setiap *vertical*, dapat mengunduh *software* Charles proxy untuk kebutuhan *testing*. Charles Proxy dapat diunduh di Google Chrome, namun setelah mengunduh, diperlukan konfigurasi didalam Charles Proxy agar dapat disambungkan dengan *emulator* atau *real device* sesuai yang dibutuhkan menggunakan Proxy. Dalam kegiatan saat ini, digunakan *real device* sehingga Charles Proxy dihubungkan dengan *real device* menggunakan IP Adress yang sama.



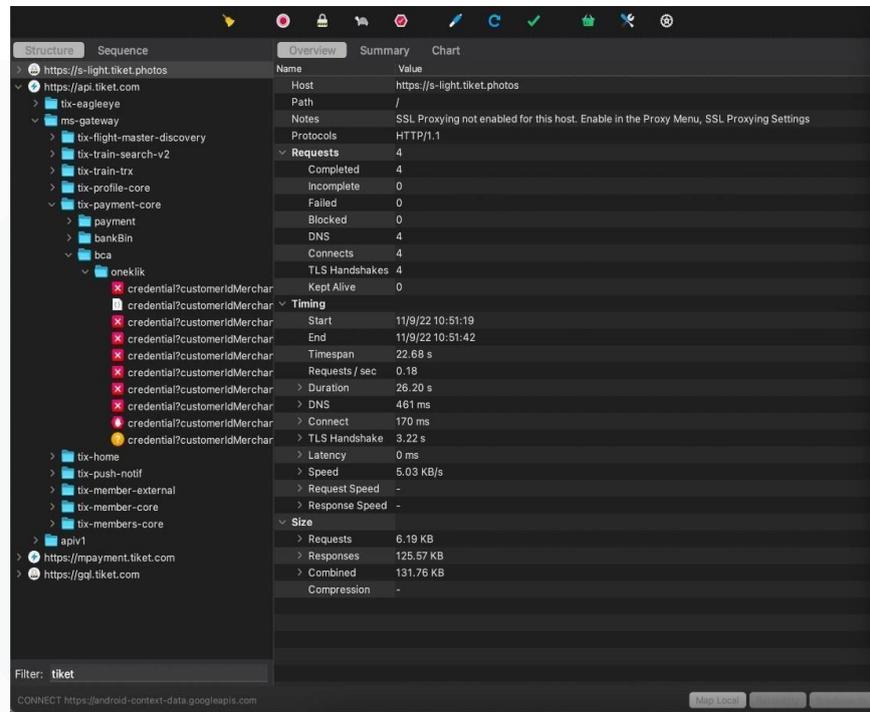
Gambar 3.6 IP Address untuk Menghubungkan Proxy

Gambar 3.6 menunjukkan IP address yang akan digunakan untuk menghubungkan Charles Proxy dengan *real device* yang dimiliki, setiap jaringan memiliki IP address yang berbeda serta IP address secara berkala berganti-ganti sehingga diperlukan pengecekan secara rutin agar koneksi dapat terhubung. IP address perlu diingat dan disalin ke proxy *real device* agar dapat terhubung.



Gambar 3.7 Konfigurasi *Real Device Proxy*

Gambar 3.7 menunjukkan tampilan konfigurasi untuk menghubungkan Charles Proxy dengan *real device*, memasukan IP address dan host dengan angka 8888 sebagai *default*. Setelah sudah menekan tombol save maka aplikasi akan otomatis terdeteksi di Charles Proxy dan dapat melanjutkan dengan kegiatan *testing Network Error Handling (NEH)* dengan cara mengecek dan mengubah *status code* pada Charles Proxy untuk dengan sengaja merubah *output output* yang dihasilkan berupa *error screens*.



Gambar 3.8 Tampilan Charles Proxy

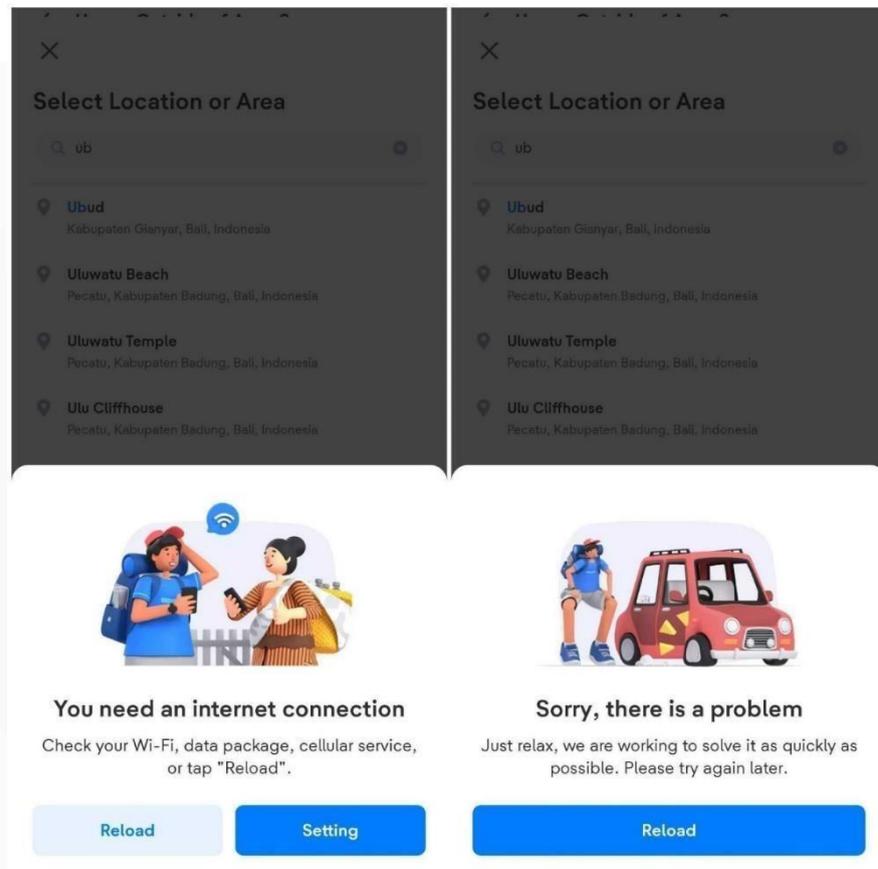
Gambar 3.8 merupakan tampilan Charles Proxy ketika sebuah API sudah di *trigger* untuk diubah *status code* nya. *Status code* tersebut akan dilakukan *testing* dari 400-500 dan *exceptions*. Setiap *status code* sendiri memiliki hasil akhir yang berbeda-beda sesuai *expected result* yang sudah disetujui oleh tim desainer. *Status code* yang akan dilakukan pengecekan adalah *status code* yang sudah disetujui oleh tim *developer*, tidak semua *status code* akan dilakukan *testing*, tabel 2 akan menjelaskan *status code* apa saja yang akan dilakukan *testing* pada Network Error Handling.

Tabel 2. Status Code yang akan Dilakukan Testing

Status code	Description
400	Bad Request
422	Unprocessable Entity (WebDAV)
429	Too Many Requests

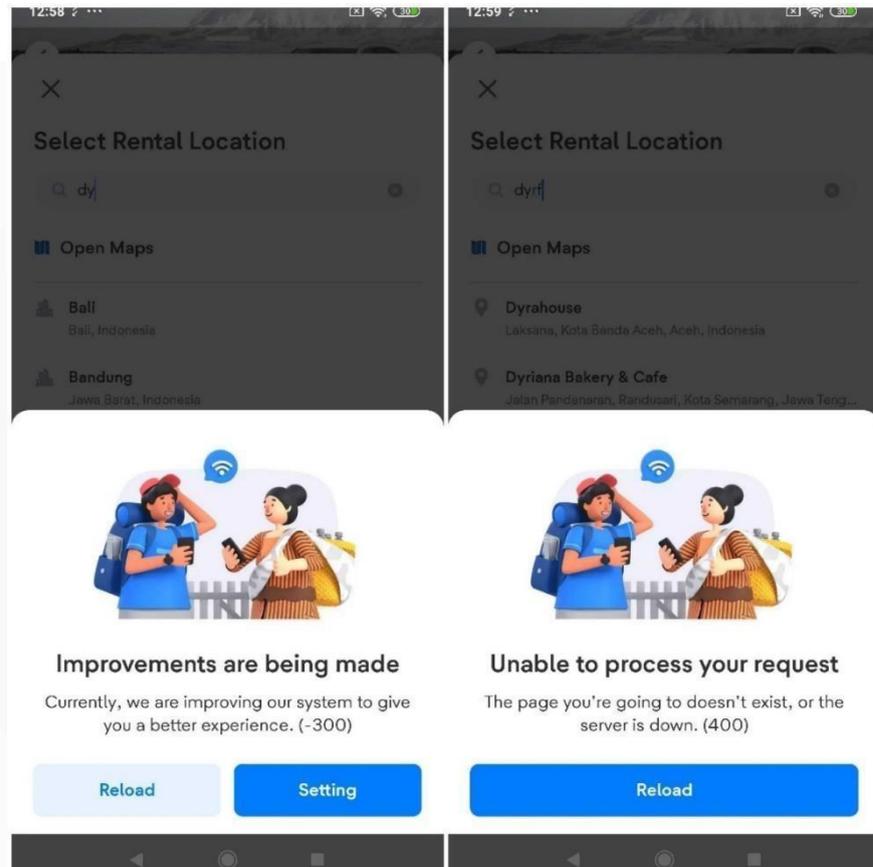
500	Internal Server Error
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
520	Web Server Returned an Unknown Error
522	Connection Timed Out
-103	ConnectException
-106	SocketTimeoutException
-300	StreamResetException

Kemudian akan meng *trigger* API dan mengubah *status codenya* yang kemudian layar aplikasi di *real device* akan mengeluarkan *error screens* yang berbeda-beda sesuai *status code* yang digunakan. Tujuan dari tugas ini adalah untuk memunculkan *error screens* yang lebih *friendly* terhadap *user* serta membuat *user* tidak merasa *stuck* jika suatu *error* terjadi. Pembaruan *error screens* juga ditujukan agar *developer* akan mengalami kemudahan untuk melakukan *tracking* jika terjadi kegagalan dalam sebuah sistem di aplikasi sesuai dengan *code* yang diterima di setiap *error screens*.



Gambar 3.9 *Error Screens* Versi Sebelum

Gambar 3.9 merupakan *error screens* pada aplikasi versi sebelumnya, aplikasi sebelumnya hanya memiliki secara total 4 jumlah *error screens* yang tidak *friendly* pada *user*. *Error screens* tidak memberi *user* arahan dan penjelasan mengenai kenapa *error* terjadi sehingga *user* merasa *stuck* ketika mengalami *error* tersebut. Oleh karena itu, diperlukan *error screens* untuk setiap *error code* yang ada sehingga ditugaskan untuk mengecek apakah *error screens* yang baru sudah sesuai dengan *expected result*.



Gambar 3.10 Error Screens Versi Sesudah

Gambar 3.10 merupakan salah satu bentuk dari *error screens* yang baru dengan *code* 400 dan -300, *code* diakhir memudahkan *developer* untuk melakukan *tracking* kepada *error* yang terjadi dan *message error screens* yang baru pun lebih memiliki arahan kenapa *error* terjadi untuk *user*. Diperlukan juga untuk melakukan pengecekan untuk setiap *status code* di setiap API yang ada sesuai API yang digunakan oleh setiap *vertical* seperti *vertical payment*, *flights*, *TTD*, *hotel*, *platform* dan *ground transport*. Setelah *testing* dilakukan dari sisi visual, maka akan dilakukan *testing* pada network error handling dari sisi teknisnya yaitu dengan mengecek *eagleeye* yang akan ter *log* didalam database perusahaan serta Charles Proxy.

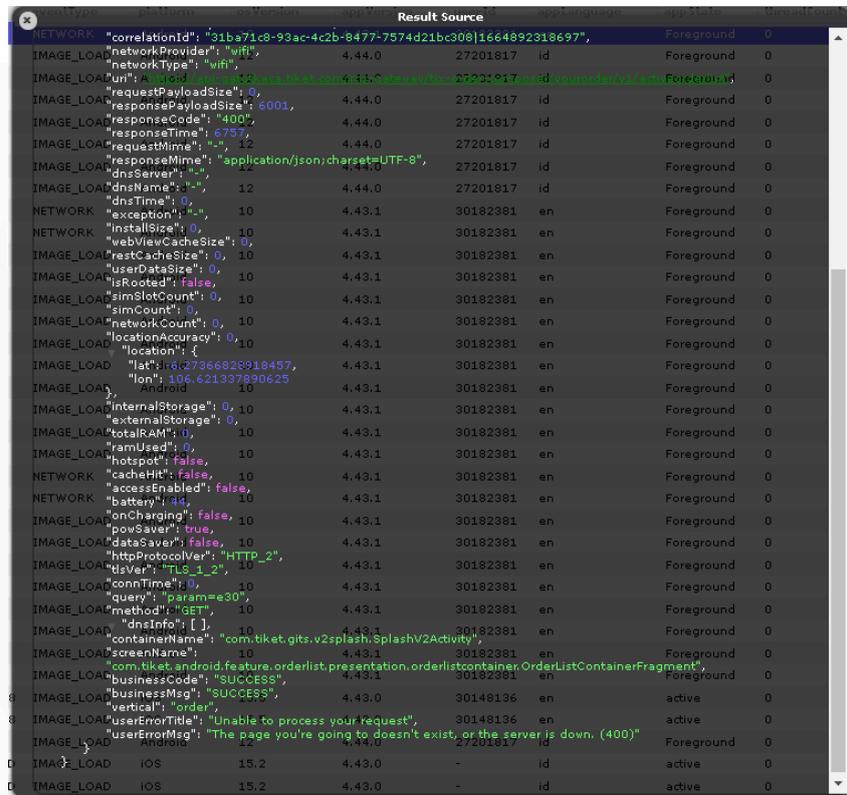


```
Overview Contents Summary Chart Notes
1 1660211567809,"dnslp":"-","dnsName":"-","dnsTime":0,"exception":"-","connTime":0,"respTime":7864,"respCode":400,
rErrorTitle":"Unable to process your request","userErrorMsg":"The page you're going to doesn't exist, or the server is d
CESS","timestamp":1660211575689,"dnslp":"-","dnsName":"-","dnsTime":0,"exception":"-","cacheHit":false,"connTim

"tiloriginal=original","respTime":2,"cacheHit":true,"eventType":"IMAGE_LOAD","timestamp":1660211580405,"containe
ginal=original","respTime":1,"cacheHit":true,"eventType":"IMAGE_LOAD","timestamp":1660211581710,"containerNam
loriginal=original","respTime":3,"cacheHit":true,"eventType":"IMAGE_LOAD","timestamp":1660211581716,"containerN
respMime":"application/json;charset=UTF-8","httpProtocolVer":"HTTP_1_1","tlsVer":"TLS_1_2","businessCode":"SUCCES
r":"https://api.tiket.com/ms-gateway/tix-member-core/v2/partner/bilibi/loyalty-status","query":"-","method":"GET",
://api.tiket.com/...","plat":"Android","osVer":"10","appVer":"4.40.0","sessionId":"e14d459e-5d8b-4969-80ba-cf683ac569
cf683ac569ce|1660211581747","respPayloadSize":1523,"respMime":"application/json;charset=UTF-8","httpProtocolVer
'plat":"Android","osVer":"10","appVer":"4.40.0","sessionId":"e14d459e-5d8b-4969-80ba-cf683ac569ce","appLang":"en",
f
```

Gambar 3.11 *Testing* Teknikal di *Eagleeye* Charles Proxy

Gambar 3.11 merupakan hasil *testing* dari sisi teknikal yang dilakukan pada Charles Proxy, yaitu untuk mengecek *eagleeye* apakah *error* yang terjadi di aplikasi sudah ter *log* didalam Charles Proxy dengan *code* dan *message* yang sama sesuai yang terjadi di aplikasi. Jika *code* dan *message* berbeda dengan yang terjadi di aplikasi, dapat membuat *JIRA card* berbentuk *bug* untuk memberitahu *developer* bahwa terjadi kesalahan dibagian tertentu dan kemudian akan menunggu *bug fix* untuk dilakukan *testing* kembali pada aplikasi. Selain mengecek *code* dan *message* yang di *log* didalam Charles Proxy, diharuskan juga untuk mengecek dari aplikasi Multihead search yang dapat digunakan di Google Chrome.

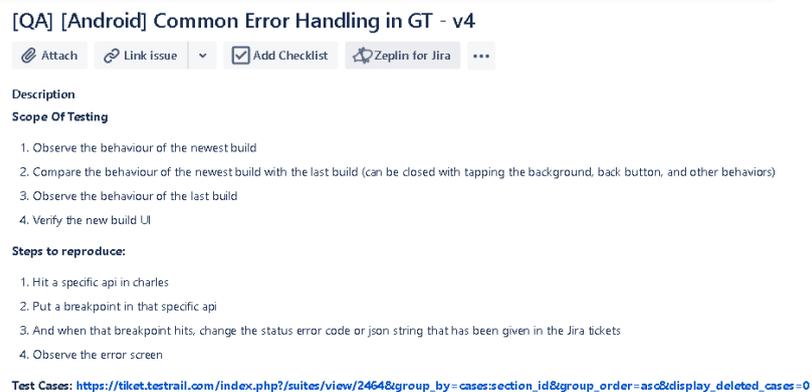


Gambar 3.12 Testing Teknikal di Multihead Search

Gambar 3.12 merupakan hasil *testing* dari Multihead search, Multihead search akan *log* semua aktivitas *eagleeye* dari semua aplikasi *debug* yang digunakan. Diharuskan untuk mencocokkan *code* dan *message* yang ter *log* di Charles Proxy dan Multihead search sesuai dengan *error screens* yang terjadi di aplikasi. Diberikan domain perusahaan untuk mengecek aktivitas *eagleeye* yang terjadi pada setiap aplikasi yang akan bermunculan di Multihead search, untuk mengecek *log* aktivitas *testing* dari sisi teknikalnya. Ketika sudah mengecek dari sisi visual dan teknikal maka dapat memberi *QA sign off* yang memiliki arti bahwa tugas ini sudah dicek dan dapat dilanjutkan untuk *merge* yang akan ditambahkan kepada versi terbaru dari aplikasi.

4. Common Error Handling

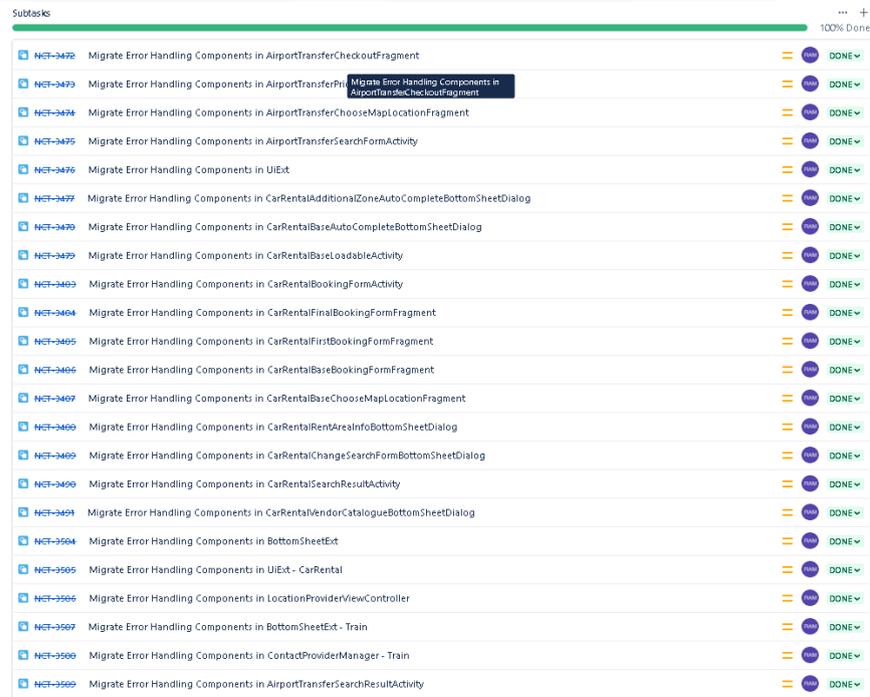
Tugas selanjutnya yaitu common error handling dimana tugas QA adalah untuk mengecek apakah *error screens* yang terjadi sudah menggunakan komponen baru dan tetap memiliki *behaviour* yang sama dengan versi aplikasi sebelumnya meskipun menggunakan komponen yang baru sehingga tugas QA disini adalah untuk mengecek dua versi aplikasi yaitu sebelum dilakukan *update* dan sesudah dilakukan *update* pada sisi komponennya untuk memastikan bahwa *behaviour* yang dimiliki sama. Komponen yang baru juga mengalami perubahan kecil yaitu *error screens* tidak dapat ditutup dengan menggunakan *sliding* sedangkan di *error screens* yang lama dapat ditutup dengan menggunakan *sliding*. Tugas common error handling dilakukan seperti biasa yaitu untuk memulai membuat *spreadsheet* yang akan digunakan sebagai sarana penyimpanan informasi mengenai *behaviour* yang terjadi di setiap API yang akan dilakukan *testing*. Setelah membuat *spreadsheet* maka tugas dapat dilanjutkan dengan tahapan selanjutnya yaitu untuk membuat *subtask* baru.



Gambar 3.13 QA *Subtask* untuk CEH *vertical* GT

Kemudian dapat membuat QA *subtask* untuk mengisi informasi seperti pada gambar 3.13 yaitu untuk memberi informasi *scope* yang dilakukan didalam *testing*, apa saja langkah-langkah yang diambil untuk melakukan *testing* serta *testcases* yang telah dibuat untuk

melakukan *testing*. Informasi lainnya berupa *QA end date*, *QA start date*, dan *QA effort* untuk memberitahu *developer* berapa lama *testing* akan dilakukan oleh QA.



Gambar 3.14 Tampilan *Parent Task* CEH

Gambar 3.14 merupakan tampilan bentuk *parent task* yang memiliki beberapa *subtask* didalamnya yang menjelaskan informasi berbeda-beda yang harus dilakukan *testing*. Setelah sudah membuat QA *subtask*, maka langkah selanjutnya dapat dilakukan yaitu untuk melihat *parent task* dari proyek yang akan dilakukan *testing*. *Parent task* tersebut berisi dengan *subtask* *subtask* lainnya yang masing-masing *subtask* akan menjelaskan cara melakukan *testing* secara rinci sehingga QA dapat mereproduksi *error screens* yang akan di *testing*.

Migrate Error Handling Components in AirportTransferSearchFormActivity

Attach Link Issue Add Checklist Zeplin for Jira

Description

Migrate Error Handling Component using following info:

Class: `AirportTransferSearchFormActivity`

Component Used: `TDSerrorBottomsheet - v4`, `TDSDialog - v4`

Component Migrate To: `TDSInfoDialog - v4`

Trigger point:

Flow 1: Airport Transfer -> API `/as-gateway/tix-airport-transfer-catalogue/v2/landing-page-hits` -> Intercept and produce error -> Offline error dialog is shown.

Flow 2: Airport Transfer -> Don't Fill Any Form Details -> Tap on Let's See Your Car -> Different Dialog is shown, if particular field is not filled.

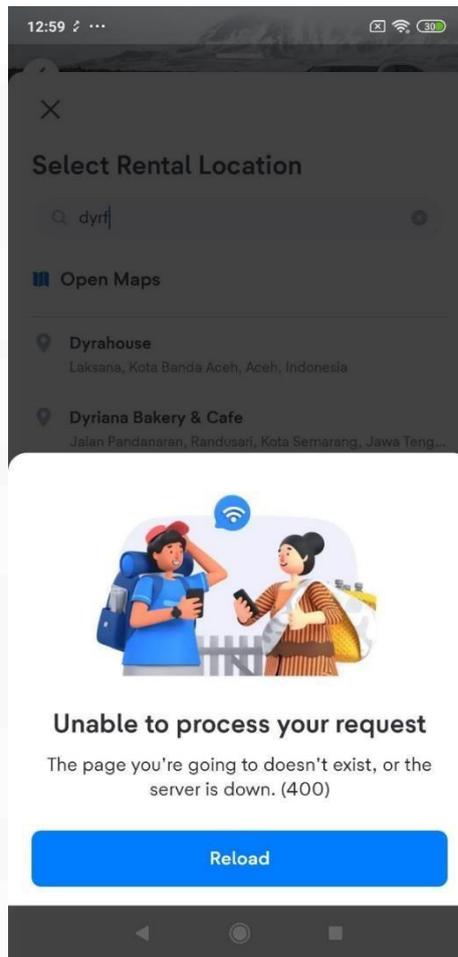
PR Link: <https://github.com/tiket/tiketandroid/pull/8232> - Connect to preview

Comments

None

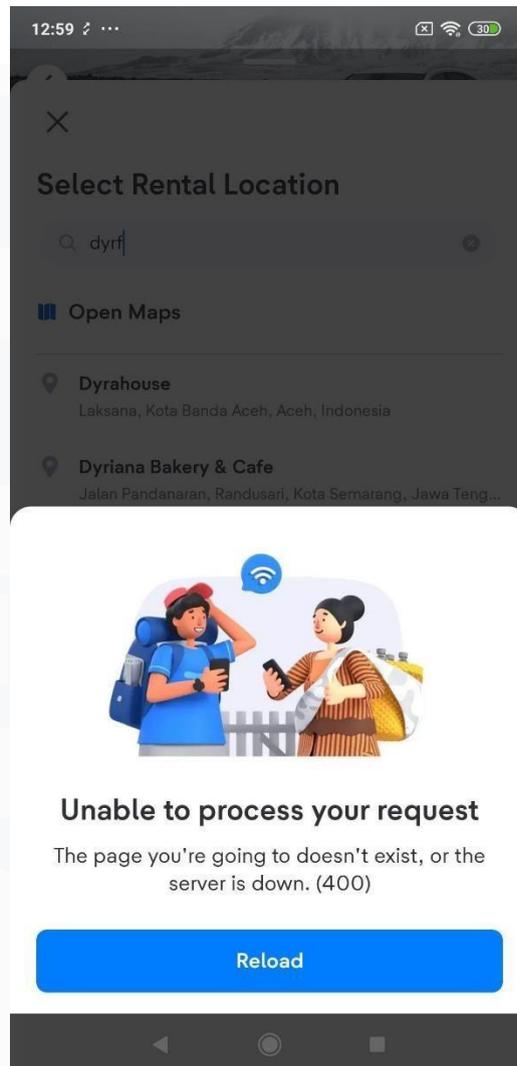
Gambar 3.15 Tampilan *Subtask* berisi Informasi

Gambar 3.15 merupakan tampilan informasi yang terdapat pada masing-masing *subtask*, diharuskan untuk mengikuti setiap *flow* yang ada untuk melakukan *testing*. *Testing* dapat diikuti dengan langkah-langkah yang telah diberikan oleh *developer*, beberapa *flow* mengharuskan untuk mengubah JSON text dari sebuah API untuk menghasilkan *error screens* yang dicari. Setelah *error screens* ditampilkan maka dapat melanjutkan langkah selanjutnya yaitu untuk melakukan komparasi.



Gambar 3.16 *Error Screens* Versi Sebelum

Gambar 3.16 merupakan contoh *error screens* yang dihasilkan di aplikasi versi sebelumnya. Aplikasi versi sebelumnya memiliki beberapa *behaviour* yang harus dicatat seperti ketika menekan tombol *setting* dan menekan tombol *back* tetap akan menampilkan *error screens* sebelumnya atau menghilang. Setelah sudah mencatat setiap *behaviour* dari setiap API maka tugas akan dilanjutkan ke langkah berikutnya yaitu untuk mengecek pada aplikasi versi yang baru.



Gambar 3.17 *Error Screens* Versi Baru

Gambar 3.17 merupakan contoh *error screens* yang dihasilkan di aplikasi versi baru. Memang secara *visual*, *error screens* tidak berubah namun secara komponen berubah sehingga diperlukan pengecekan kepada *behaviour* apakah *behaviour* komponen baru tetap memiliki *behaviour* yang sama dengan komponen yang lama. Ketika terjadi *behaviour* yang berbeda maka *bug* harus dilaporkan didalam JIRA untuk memberitahu *developer* bahwa terjadi kesalahan di suatu API sehingga dapat dibenarkan oleh *developer*. Setelah semua API sudah dilakukan *testing*, maka QA dapat

memberikan *sign off* yang memiliki arti bahwa tugas ini sudah dicek dan dapat dilanjutkan untuk *merge* yang akan ditambahkan kepada versi terbaru dari aplikasi.

5. Design Task

Tugas berikutnya adalah tugas *design* untuk mengecek apakah hasil *container* yang dibuat oleh *developer* sudah sesuai dengan *tech spec* milik divisi *designer*. *Tech spec* diberikan kepada menggunakan Figma dimana didalam Figma tersebut, terdapat informasi mengenai spesifikasi yang digunakan untuk setiap *container*, sehingga dapat melakukan *testing* dengan menggunakan Figma tersebut sebagai referensi. *Testing* diperlukan untuk mengecek di bagian IOS sehingga dibutuhkan BrowserStack karena tidak memiliki *real device* IOS. Dapat mengunduh *build* yang sudah diberikan *developer* dan melakukan *install* kepada Browserstack untuk dilakukan *testing*. Tugas utama QA di tugas ini adalah untuk memastikan bahwa *container* dari *label* berubah bentuknya sesuai isi tulisan didalam *container* tersebut.



Gambar 3.18 Hasil Akhir *Design Task*

Gambar 3.18 merupakan hasil akhir perubahan *design* pada *container*, dapat dilihat di gambar 28 bahwa ketika *text* melebihi maka *container* akan beradaptasi melebarkan untuk dapat menampung semua *text* yang ada didalam *container* tersebut.

3.3 Kendala yang Ditemukan

Dalam pelaksanaan praktik kerja magang yang dilakukan di perusahaan tiket.com, dialami beberapa kendala yang telah menghambat untuk menyelesaikan pekerjaan dan tugasnya, kendala tersebut adalah sebagai berikut:

- Pada masa pelaksanaan tugas NEH, dialami beberapa kesulitan dalam mengoperasikan *software* karena belum pernah menggunakan *software* tersebut.

- Pada masa pelaksanaan tugas NEH, dialami kesulitan untuk mengkonfigurasi Charles Proxy.
- Pada masa pelaksanaan tugas NEH, dialami beberapa kesulitan untuk berkomunikasi dengan *developer*.

3.4 Solusi atas Kendala yang Ditemukan

Dalam pelaksanaan praktik kerja magang yang dilakukan di perusahaan tiket.com, dirumuskan solusi atas kendala kendala yang dihadapi di perusahaan sebagai berikut:

- Memberanikan diri untuk bertanya kepada rekan setim yang lebih berpengalaman untuk ilmu dan informasi cara mengoperasikan *software*.
- Memberanikan diri untuk bertanya kepada rekan setim yang pernah mengkonfigurasi *software* tersebut.
- Memberanikan diri untuk berkomunikasi dengan pihak *developer* agar tugas lebih jelas dan spesifik.