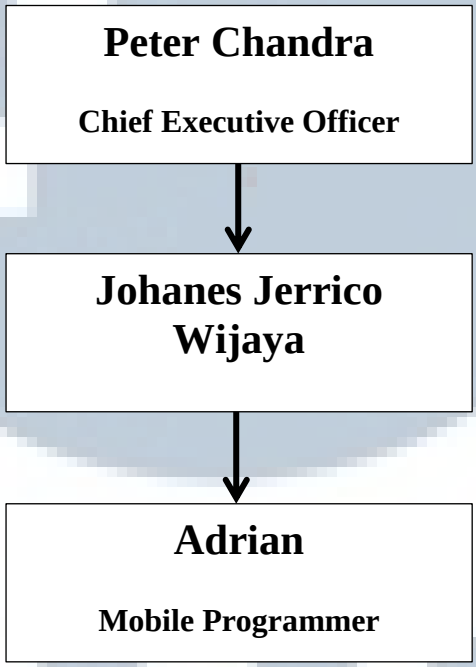
BAB III
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pelaksanaan praktik kerja magang di PT. Moonlay Technologies dimulai pada tanggal 1 Juli 2013 dan berakhir pada tanggal 31 Agustus 2013. Pada praktik kerja magang ini, bekerja pada divisi *Product and Service* sebagai *mobile programmer* bersama Bernadinus Realino Edi Gunawan dan Joshua Prayogi Angki. Divisi *Product and Service* ini dikepalai oleh Bapak Johanes Jerrico Wijaya yang juga menjadi pembimbing lapangan selama melaksanakan praktik kerja magang.



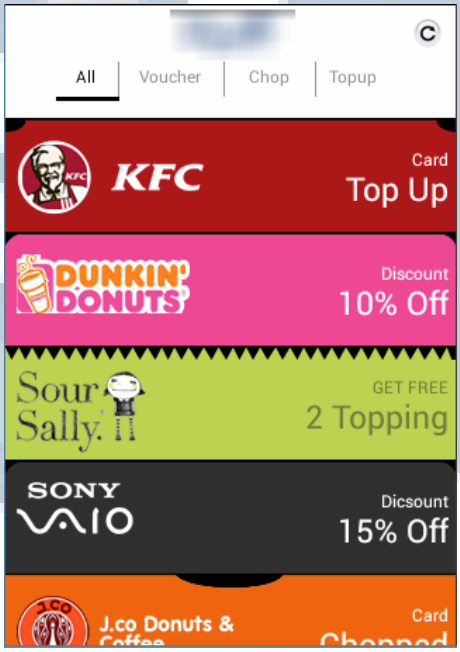
Gambar 3.1 Koordinasi Kerja Magang

3.2 Tugas yang Dilakukan

Pada awal periode praktik kerja magang, para peserta magang dibagi menjadi 3 kelompok. Dikelompokkan dengan Bernadinus Realino Edi Gunawan dan Meylinda Sandra Halim, tugas awal yang diberikan adalah membuat suatu *web* yang berfungsi sebagai *template* dari promo dan isi dari promo tersebut tidak

static dan sesuai dengan kemauan orang yang memakai *web* tersebut. Tetapi tugas *web* ini tidak akan dibahas. Setelah tugas awal yang diberikan selesai, para peserta magang dibagi lagi menjadi dua kelompok, kelompok tersebut dibagi menjadi *web programmer* dan *mobile programmer*. Setelah dibagi kelompok, kemudian diberilah tanggung jawab untuk mengerjakan modul *List Promo* dan Entiti Kartu dari aplikasi Loyalty Express. Beberapa modul yang menjadi tanggung jawab adalah *Layouting*, *Dynamic Content*, *Load More Content*, *Refresh Content*, dan *Filter Content.*

Tugas pertama yang diberikan adalah membuat *layout* dari *List Promo* dan Entiti Kartu. *Design layout* tersebut harus dibuat sesuai dengan *design* yang diberikan oleh *designer*. Gambar 3.2 dan 3.3 adalah *layout* yang dibuat.



Gambar 3.2 *List* Entiti Kartu

Gambar 3.3 *List Promo*

Jika pada tugas pertama ditugaskan untuk membuat suatu *layout* di mana *design* dari promo atau entiti kartu sudah ditentukan oleh para *designer*, pada tugas kali ini diberikan tugas untuk membuat isi dari promo atau entiti kartu list secara *dynamic*. Karena pada aplikasi Loyalty Express, seluruh *content* dari promo atau entiti kartu tersebut haruslah dibuat secara *dynamic*. *Content* dari aplikasi ini akan disesuaikan dengan *client* yang membuat promo atau entiti kartu yang menggunakan aplikasi ini. *Client* hanya tinggal memilih *template* yang digunakannya untuk membuat promo atau entiti kartu tersebut.

List dari promo atau entiti kartu yang dimiliki oleh *user* kemungkinan tidaklah berjumlah sedikit, maka dibatasi hanya beberapa *content* saja yang ditampilkan pada halaman awal. Tentunya *user* menginginkan seluruh *content* yang dibuat bisa ditampilkan semuanya. Modul *Load More Content* yang dapat menjadi solusi untuk masalah tersebut. Fungsi dari *Load More Content* ini adalah untuk menampilkan sisa *content* yang belum ditampilkan tersebut. Fungsi ini akan terus ada di dalam *list* sampai tidak ada lagi *content* yang tersisa.

Ketika seorang *user* memiliki suatu promo atau entiti kartu baru, maka tentunya *user* ingin melihatnya dalam aplikasi dari Loyalty Express ini. Pada saat *user* sedang membuka aplikasi, tentu jika ada promo atau entiti kartu baru, tidak

akan langsung ditampilkan, tetapi hanya akan tersimpan dalam *database*. Untuk membuat promo atau entiti kartu yang baru tersebut dapat ditampilkan, dibuatlah suatu fungsi, yaitu *Refresh Content.* Fungsi dari *Refresh Content* adalah untuk me-*reload* seluruh *content* promo atau entiti kartu yang dimiliki oleh *user* tersebut, sehingga jika ada suatu perubahan dari *content* tersebut, aplikasi ini akan selalu bisa menampilkan *content* secara *up-to-date.* Jadi ketika tombol *refresh* ditekan oleh *user,* maka fungsi dari *Refresh Content* ini akan berjalan.

Ketika *user* memiliki promo atau entiti kartu yang banyak di dalam aplikasi ini, maka tentunya harus ada suatu *filter* yang dapat mempermudah user untuk mencari promo atau entiti kartu yang dimilikinya tersebut. Untuk mempermudah *user* dalam menggunakan aplikasi ini, maka dibuatlah suatu fungsi yang dapat mem-*filter content – content* tersebut.

Pada *list* entiti kartu akan dibuat filter yang dibagi menjadi empat filter, yaitu filter untuk *content* jenis *voucher,* filter untuk *content* jenis *topup,* filter untuk *content* jenis *chop,* dan filter untuk seluruh jenis *content. Default* filter untuk halaman *list* entiti kartu adalah filter untuk seluruh jenis *content.*

Pada promo *list* akan dibuat filter yang dibagi menjadi tiga filter, yaitu filter untuk promo yang paling terbaru, filter untuk promo berdasarkan lokasi promo tersebut, dan filter untuk promo berdasarkan promo yang sudah ditandai atau di-*bookmark* oleh *user*. *Default* filter untuk halaman promo *list* adalah filter untuk promo yang paling terbaru.

3.3 Uraian Pelaksanaan Kerja Magang

Praktik kerja magang yang dilakukan diawali dengan pengenalan dan instalasi *software* Titanium Studio yang diperlukan dalam pengembangan sistem.

Review dengan pembimbing lapangan dilakukan pada hari Senin setiap minggunya untuk mengetahui sejauh mana proyek ini telah dikerjakan dan juga untuk berkonsultasi tentang masalah yang dihadapi ketika membuat proyek.

Table 3.1 *Timetable* Pelaksanaan Kerja Magang

No	Kegiatan	Minggu ke-							
		1	2	3	4	5	6	7	8
1	Studi literatur								
2	*Requirement analysis*								
3	*Design*								
4	*Development*								
5	*Integration and test*								
6	*Implementation*								
7	*Maintenance*								
	Penulisan dokumentasi								

3.3.1 Studi Literatur

Penggunaan Titanium Studio yang masih baru dapat membuat *progress* dari pembuatan modul berjalan lambat. Untuk mencegah *progress* berjalan lambat, maka dilakukan studi literatur. Studi literatur dilakukan secara bersamaan dengan pembuatan modul.

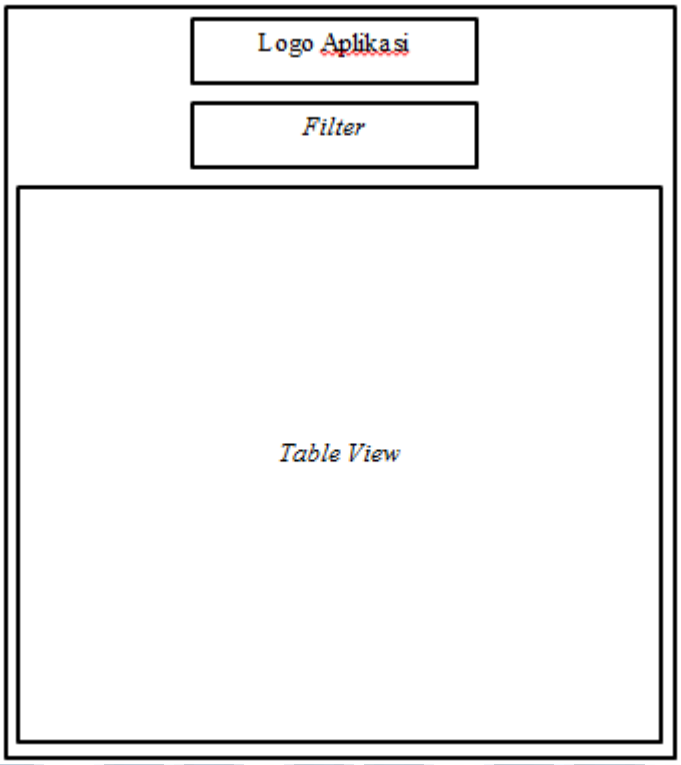
3.3.2 Requirement Analysis

Di dalam pembuatan aplikasi dari Loyalty Express ini, digunakan Appcelerator Titanium Studio. Modul yang diberikan adalah *Layouting, Dynamic Content, Load More Content, Refresh Content,* dan *Filter Content.* Untuk menyelesaikan beberapa modul tersebut, harus dianalisa bagaimana pembuatan modul – modul tersebut.

A. Modul Layouting

Pada modul yang pertama, yaitu *Layouting*, dianalisa *tool* apa yang dibutuhkan untuk membuat *layout* tersebut. Berdasarkan hasil analisa, tool yang dibutuhkan untuk membuat *layout* ini adalah:

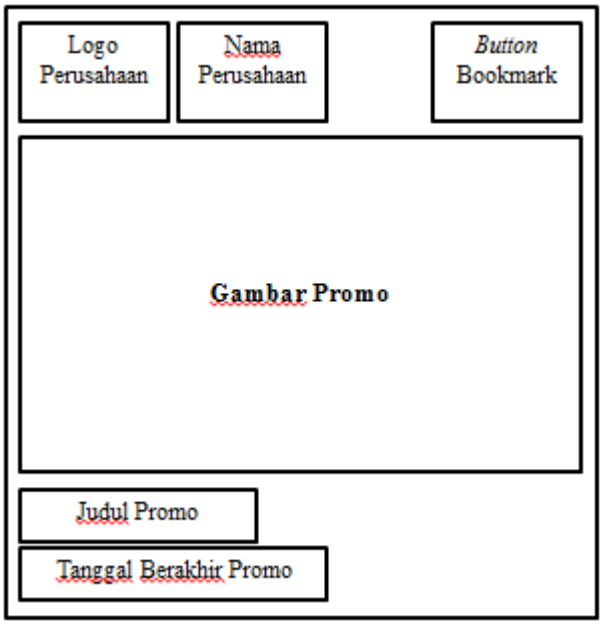
1) *ListView* atau *TableView* untuk promo dan entiti kartu, karena aplikasi ini akan menampilkan *list* dari promo – promo dan entiti kartu – entiti kartu yang dimiliki oleh user. Digunakan *TableView* untuk membuat *list* tersebut. Digunakan *TableView* karena *TableView* memiliki *event listener* yang dibutuhkan dalam aplikasi ini yang tidak dimiliki oleh *ListView*.
2) *Template* halaman awal aplikasi yang terdiri dari sebuah *ImageView* yang berfungsi untuk menampilkan logo aplikasi, sebuah *View* yang berfungsi sebagai *filter* untuk *content* dan sebuah *TableView*.



Gambar 3.4 Sketsa halaman awal

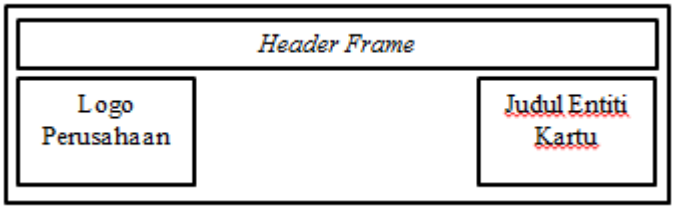
3) *Template* untuk pembuatan *content* dari *list* promo, yaitu dua *ImageView* yang berfungsi untuk menampilkan gambar promo dan logo perusahaan yang membuat promo tersebut, sebuah *Button* untuk mem-*bookmark* suatu promo, dan tiga *Label* yang berfungsi untuk menampilkan nama dari perusahaan yang

membuat promo, judul dari promo tersebut, dan tanggal berakhirnya promo tersebut.



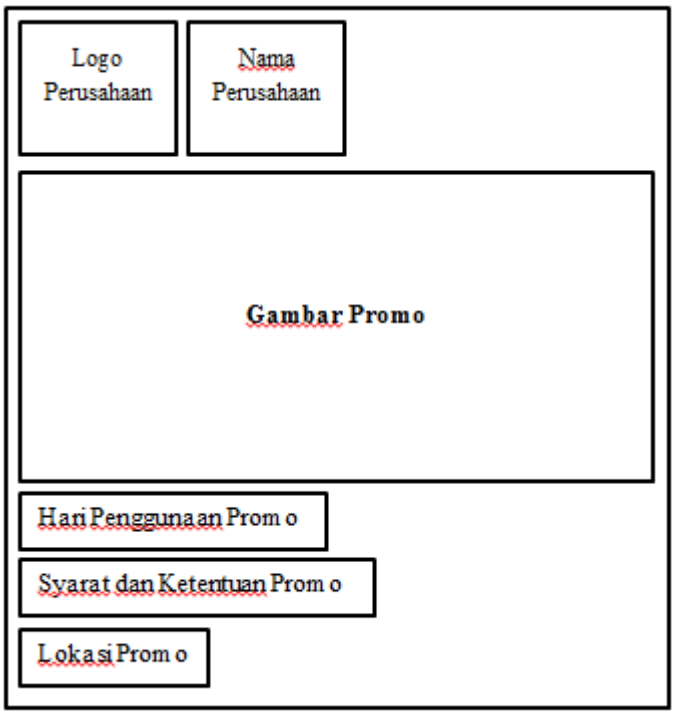
Gambar 3.5 Sketsa *content list* promo

4) *Template* untuk pembuatan *list* entiti kartu, yaitu dua *ImageView* yang berfungsi untuk menampilkan *header frame* entiti kartu beserta warna dari entiti kartu dan logo perusahaan yang membuat entiti kartu tersebut, sebuah *Label* yang berfungsi untuk menampilkan judul dari entiti kartu.



Gambar 3.6 Sketsa *content list* entiti kartu

5) *Template* untuk pembuatan *detail* promo, yaitu dua *ImageView* yang berfungsi untuk menampilkan gambar promo dan logo perusahaan yang membuat promo dan empat buah *Label* yang berfungsi untuk menampilkan nama perusahaan, hari penggunaan promo, syarat dan ketentuan promo, dan lokasi promo



Gambar 3.7 Sketsa *detail* promo

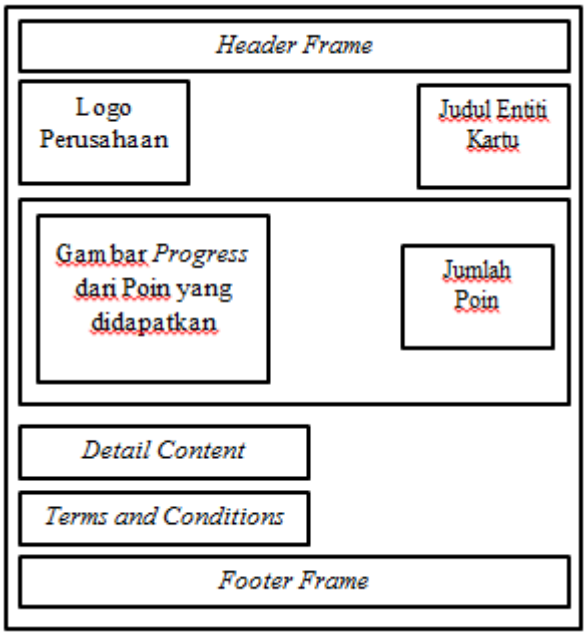
.

6) *Template* untuk pembuatan *detail* entiti kartu, yaitu tiga *View* yang berfungsi sebagai *header*, *body*, dan *footer*. *Template* dari *header* adalah dua *ImageView* yang berfungsi untuk menampilkan *header frame* entiti kartu beserta warna dari entiti kartu dan logo perusahaan yang membuat entiti kartu tersebut, sebuah *Label* yang berfungsi untuk menampilkan judul dari entiti kartu. *Template* dari *body* dan *footer* dibagi menjadi tiga tipe, yaitu tipe *voucher*, *chop*, dan *top up*. *Template body* untuk bagian *voucher* adalah sebuah *ImageView* yang menampilkan gambar dari promo tersebut. *Template footer*-nya adalah dua *Label* yang menampilkan *detail content* dari promo tersebut dan menampilkan *Terms and Conditions* jika ada. *Template body* untuk bagian *chop* adalah *ImageView* yang menampilkan *chop* yang didapat oleh *user* yang jumlah *ImageView*-nya disesuaikan dengan jumlah *chop* dari promo tersebut. *Template footer*-nya adalah tiga *Label* yang menampilkan content dari promo, *Terms and Conditions* jika ada, dan sisa *chop*. *Template body* untuk bagian *top up* adalah sebuah *Label* yang menampilkan jumlah poin yang *user* miliki dan sebuah *ImageView* yang menampilkan sebuah lingkaran yang memperlihatkan

progress dari poin yang didapatkan. *Template footer*-nya adalah dua *Label* yang menampilkan *detail content* dari promo tersebut dan menampilkan *Terms and Conditions* jika ada.

Gambar 3.8 Sketsa *detail* entiti kartu tipe *voucher*

Gambar 3.9 Sketsa *detail* entiti kartu tipe *chop*



Gambar 3.10 Sketsa *detail* entiti kartu tipe *topup*

B. Modul Dynamic Content

Pada modul kedua, yaitu *Dynamic Content*, diperlukan *tool* yang berfungsi untuk membuat isi dari aplikasi secara *dynamic*. *Tool* untuk membuat isi secara *dynamic* berupa sebuah *library* yang dibuat oleh Joshua Prayogi Angki. Nama *tool* ini adalah *JSON to Backbone JS Mapper*. Cara kerja dari *tool* ini adalah mengambil data dari *server* yang berupa *JSON String* yang kemudian akan diubah menjadi sebuah *JSON Object.*

C. Modul Load More Content

Pada modul ketiga, yaitu *Load More Content*, diperlukan *tool* yang dapat mengambil data – data yang masih belum ditampilkan. Berdasarkan hasil analisa, yang diperlukan untuk membuat modul ini adalah, 1 baris dari *TableView* yang berfungsi seperti button. Ketika baris ini ditekan, maka muncul sebuah *progress bar* dan berjalannya sebuah proses pengambilan data – data yang belum ditampilkan. Setelah proses selesai, maka *progress bar* akan menghilang dan akan ditampilkan *content* baru dari data yang sudah diambil pada proses sebelumnya.

D. Modul Refresh Content

Pada modul keempat, yaitu *Refresh Content*, diperlukan *tool* untuk me-*refresh content* dari *list* promo atau entiti kartu. Berdasarkan hasil analisa, yang dibutuhkan untuk membuat modul ini adalah sebuah *Button.* Ketika Button ini ditekan, maka akan muncul sebuah *progress bar* dan berjalannya sebuah proses pengambilan data – data dari *server* yang dimulai dari data awal. Setelah proses selesai, maka *progress bar* akan menghilang dan akan ditampilkan *content* baru dari data yang sudah diambil pada proses sebelumnya.

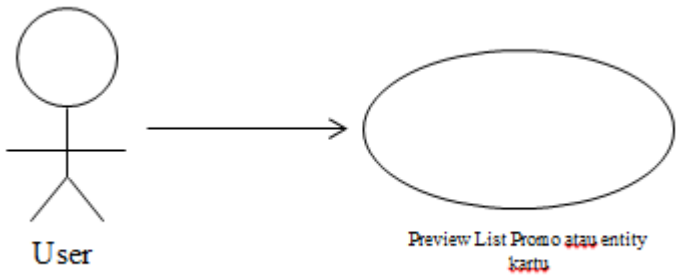
E. Modul Filter Content

Pada modul terakhir, yaitu *Filter Content,* diperlukan *tool* untuk mem-*filter content* – *content* dari *list* promo atau entiti kartu. *Filter Content* untuk promo dibuat dengan menggunakan tiga *Label* yang masing – masing adalah *recent, location,* dan *bookmark. Filter Content* untuk entiti kartu dibuat dengan menggunakan empat *Label* yang masing – masing adalah *all, voucher, chop,* dan *topup.* Akan ada satu garis bawah yang menandakan *filter* yang digunakan oleh *user*. Ketika salah satu *Label* ini ditekan, maka *content* yang ada akan berubah, sesuai dengan *filter* tersebut.

3.3.3 Design

Dalam membuat produk dari PT. Moonlay Technologies bagian *mobile application* diharuskan untuk menyelesaikan beberapa modul yang akan digunakan dalam *mobile application* tersebut. Tentunya untuk membuat modul – modul tersebut, diperlukan tahap perancangan. Dalam tahap perancangan ini, yang dibutuhkan adalah gambaran dan rancangan teknis tentang alur kerja dari sistem yang akan dibuat.

a. Use Case Diagram

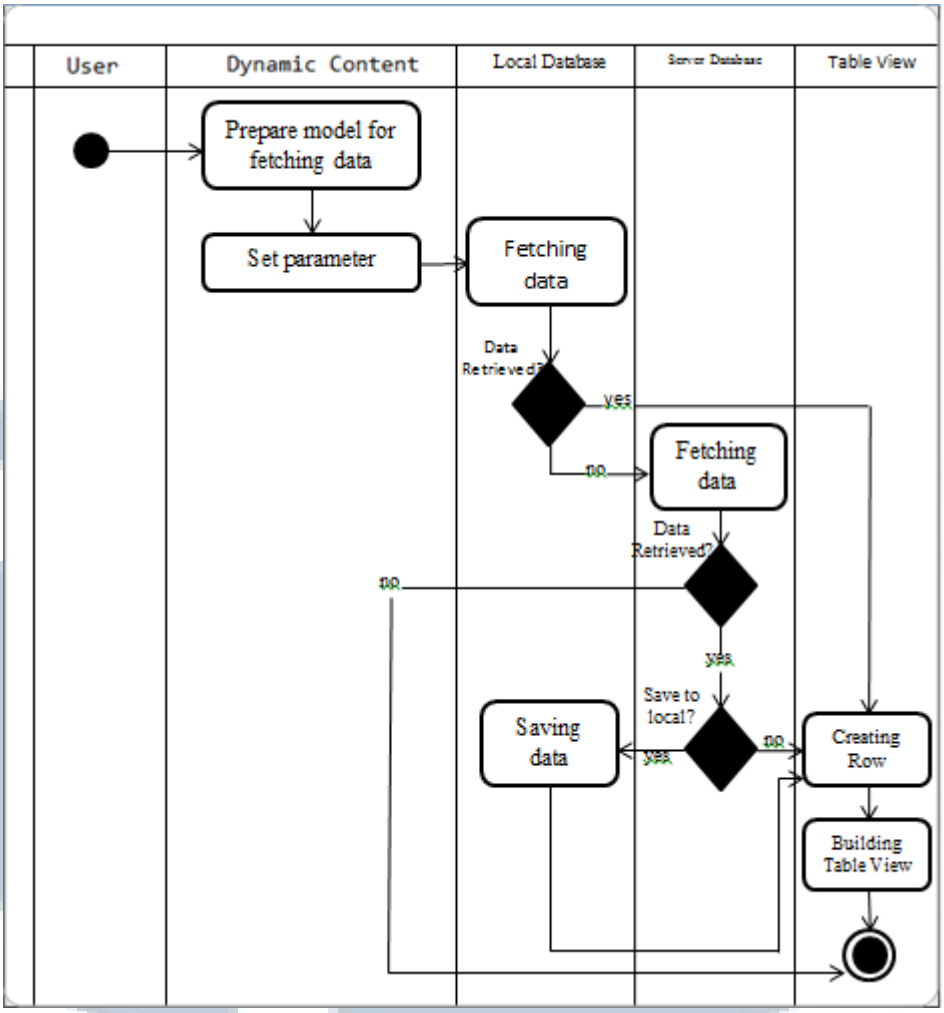


Gambar 3.11 *Use Case Diagram*

Gambar 3.11 memaparkan *use case diagram* dari aplikasi yang dibuat. Yang dapat *user* lakukan dalam aplikasi adalah melihat *preview* dari *list promo* atau entity kartu yang dimiliki oleh *user*.

b. Activity Diagram

Activity Diagram menjelaskan aliran data dari sistem modul *Dynamic Content*. *User* akan membuat suatu model yang digunakan untuk mengambil data dari *database*, pengambilan data pertama kali dalam *local database*, ketika *local database* tidak memiliki data, maka dilakukan pengambilan data dalam *database server*. Jika tidak ada data dalam *server*, maka aplikasi ini tidak akan menampilkan apa – apa. Jika ada, data dari *database server* dapat disimpan dalam *local database*. Setelah itu, diperlihatkan kepada *user preview* dari *list promo* atau entiti kartu. Gambar 3.12 merupakan pemaparan dari *Activity Diagram* dari aplikasi ini.

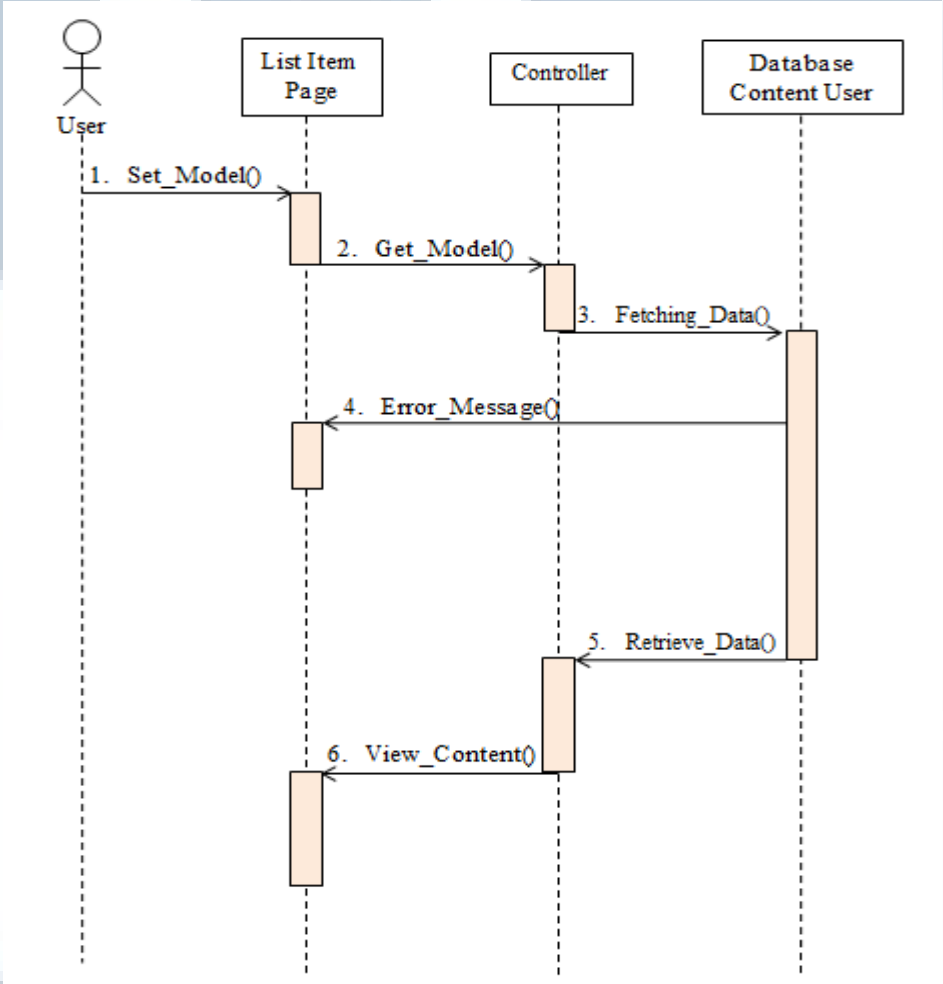


Gambar 3.12 *Activity Diagram*

c. Sequence Diagram

Sequence Diagram menggambarkan interaksi – interaksi yang terjadi ketika menjalankan fungsi *dynamic content*. *Sequence Diagram* untuk pengambilan data dari aplikasi dipaparkan pada gambar 3.13 dan skenario sebagai berikut.

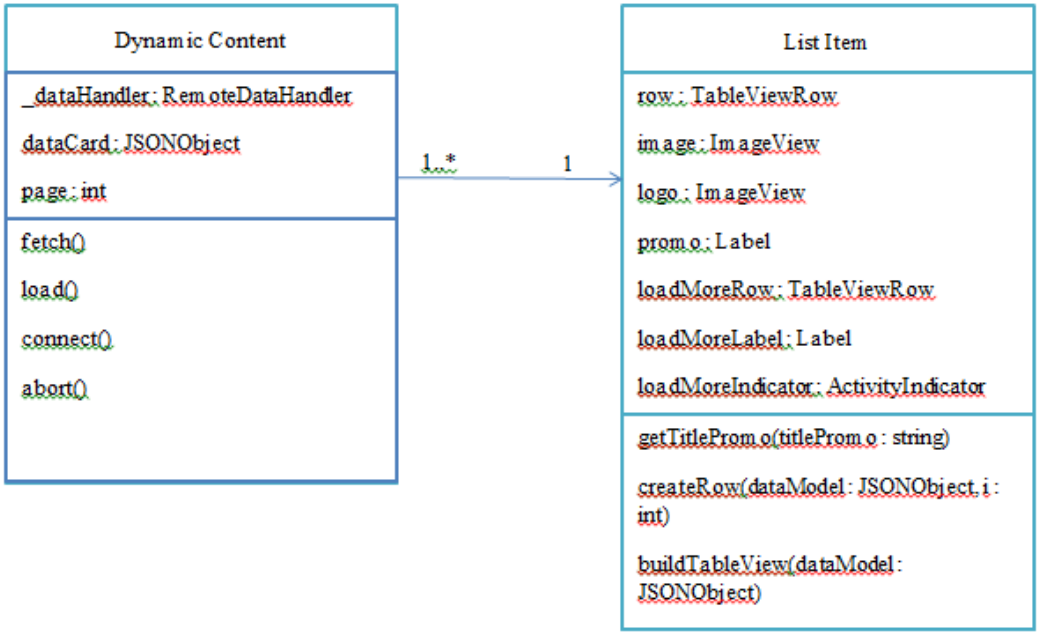
1. *User* menyiapkan model untuk pengambilan data.
2. Model yang telah disiapkan digunakan oleh *controller* untuk melakukan pengambilan data dalam *database*.
3. Melakukan pengambilan data dilakukan di dalam *database.*
4. Apabila tidak ada data yang diterima, muncul pesan *error*.
5. Ada data yang diterima.
6. Menggunakan data tersebut untuk membuat isi dari *content.*



Gambar 3.13 *Sequence Diagram*

d. Class Diagram

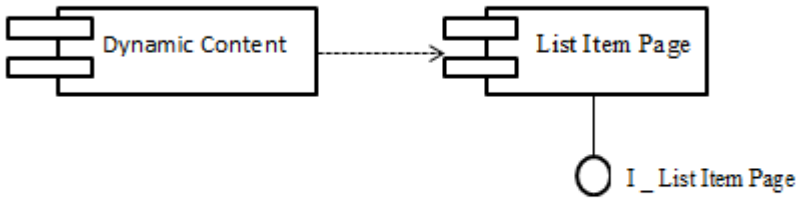
Class diagram dynamic content menggambarkan hubungan setiap *class* yang ada di dalam sistem. Dari *class diagram* tersebut dapat dilihat bahwa untuk satu *list item* memiliki satu atau lebih *dynamic content*. *Class Diagram* dari aplikasi yang dibuat ini dipaparkan pada gambar 3.14



Gambar 3.14 *Class Diagram*

e. Component Diagram

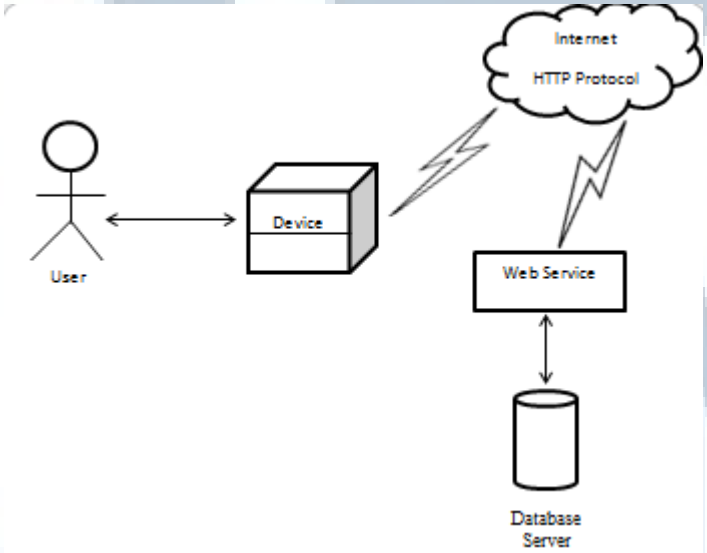
Component Diagram menggambarkan hubungan – hubungan antar komponen yang ada di dalam suatu sistem. *Component Diagram* yang dibuat di dalam aplikasi dipaparkan pada gambar 3.15



Gambar 3.15 *Component Diagram*

f. Deployment Diagram

Deployment Diagram menggambarkan bagaimana hubungan antar komponen. Komponen ini dapat berupa sebuah *device, web service,* atau *database*. *Deployment Diagram* untuk aplikasi ini dipaparkan dalam gambar 3.16



Gambar 3.16 *Deployment Diagram*

3.3.4 Development

Selama melakukan pengembangan dari aplikasi ini, digunakan beberapa perangkat keras dan perangkat lunak. Beberapa perangkat keras yang digunakan adalah:

1. Laptop ASUS A55VM dengan spesifikasi sebagai berikut:
   a. Processor Inter Core i5-3210M, 2.5GHz
   b. VGA NVIDIA GEFORCE GT 630M, 2GB
   c. 4GB RAM
   d. Harddisk 500GB
2. Smartphone LG Optimus Hub E510 untuk melakukan *testing* dengan spesifikasi sebagai berikut:
   a. Processor 800 MHz ARM v6
   b. Chipset Qualcomm MSM7227T
   c. Operating System Android v2.3.4
   d. 512 MB RAM
   e. 150 MB Internal Memory
3. Perangkat *input*, yaitu *keyboard* dan *mouse*
4. Komputer *development server* dan *storage*

Beberapa perangkat lunak yang digunakan dalam pengembangan aplikasi adalah:

1. Sistem operasi Windows 8 yang digunakan untuk pengembangan sistem.
2. Titanium Studio IDE yang digunakan untuk mengembangkan *multi-platform mobile application*.
3. Alloy Framework yang merupakan framework yang berbasis *Model, View, and Controller* (MVC) pada Titanium Studio.
4. Android Software Development Kit yang merupakan sebuah *developer tools* untuk membangun, *test*, dan *debug* applikasi untuk android.
5. Android Virtual Device yang digunakan sebagai *emulator* yang mirip seperti *device* aslinya.
6. SQLite Database yang digunakan sebagai *local database* dari aplikasi Android yang dibuat.

7. Google Chrome Web Browser yang digunakan untuk mencari informasi dan bantuan dalam pengerjaan aplikasi android.
8. Microsoft Office Excel 2010 yang digunakan untuk membuat *timesheet.*
9. Microsoft Office Word 2010 yang digunakan untuk membuat laporan praktik kerja magang.

3.3.5 Integration and Test

Dalam membuat suatu aplikasi pasti akan dilakukan pengujian aplikasi tersebut. Begitu juga aplikasi yang dibuat ini. Ada beberapa pengujian yang dilakukan pada aplikasi ini. Pengujian pertama adalah pada bagian *Layouting*, apakah *layout* yang dibuat sudah sesuai dengan yang diinginkan pembimbing. Setelah semuanya sudah sesuai, maka akan dilakukan pengujian yang berikutnya. Pengujian berikutnya adalah pada bagian *Dynamic Content*. Bagian yang diuji adalah pengecekan data yang diterima, apakah sudah benar dan ditampilkan dengan benar. Pengujian berikutnya adalah pada bagian *Load More Content.* Yang diuji pada bagian ini adalah apakah fungsi dari *Load More,* yaitu fungsi untuk mengambil data – data yang belum ditampilkan, dapat berjalan dengan baik dan benar dan bisa ditampilkan dengan benar. Pengujian berikutnya adalah pada bagian *Refresh Content.* Pengujian yang dilakukan adalah apakah fungsi dari *Refresh Content,* yaitu fungsi untuk mengambil data – data dimulai dari data paling awal, dapat berjalan dengan baik dan benar dan bisa ditampilkan dengan benar. Pengujian berikutnya adalah pada bagian *Filter Content*. Pengujian yang dilakukan adalah apakah *content* dalam masing – masing *list* promo dan entiti kartu dapat ter-*filter* dengan benar.

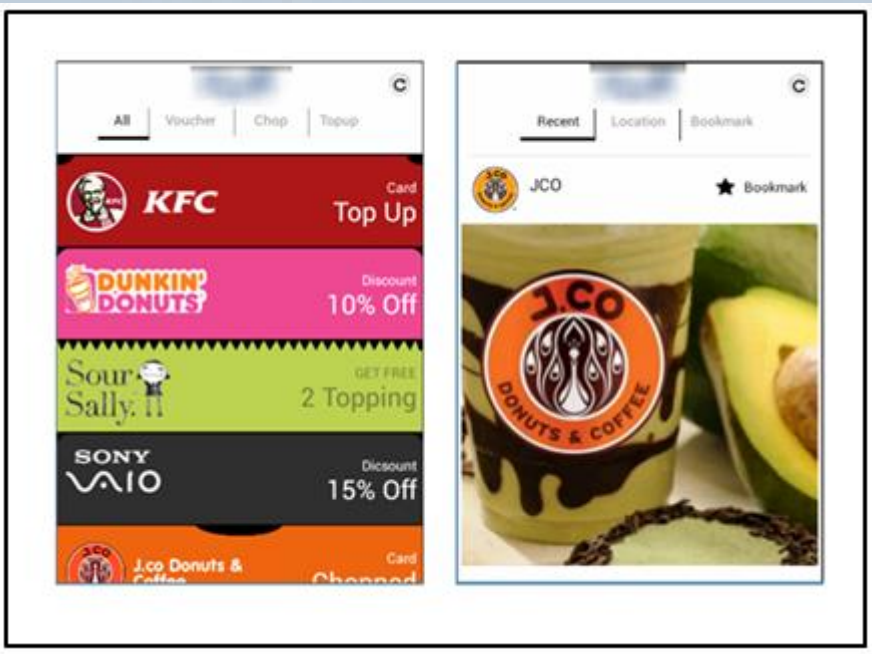
Pengujian ini dilakukan secara berurutan sesuai dengan modul yang dikerjakan untuk menghindari kesalahan dari modul – modul sebelumnya, karena semua modul yang dikerjakan saling berhubungan. Maka dari itu, proses pengujian ini dilakukan pada saat tahap pengembangan agar ketika ada kesalahan terjadi, dapat langsung diperbaiki agar modul – modul lain tidak ikut terkena dampak dari kesalahan tersebut.

3.3.6 Implementation

Dalam proses implementasi ini, pada awalnya dibuat modul *Layouting* dengan menggunakan *ListView*, karena dari hasil pencarian informasi, performa dari *ListView* lebih bagus daripada performa *TableView*. Namun setelah dicoba mengerjakan menggunakan *ListView*, ada beberapa kendala yang akhirnya membuat *ListView* diganti dengan *TableView*. Beberapa kendala tersebut adalah:

1. Warna dari *separator* tidak bisa diganti, sehingga tidak bisa sesuai dengan *design* yang diberikan.
2. Tidak adanya *event listener scroll* dan *scrollend.*
3. Warna ketika suatu *item* di dalam *list* ditekan tidak ada.

Solusi dalam kendala – kendala dalam *ListView* ini ada di dalam *TableView*. Pada *TableView* semua kendala ini dapat diatasi, sehingga akhirnya dipilih *TableView*, walaupun performanya lebih lambat daripada *ListView*.



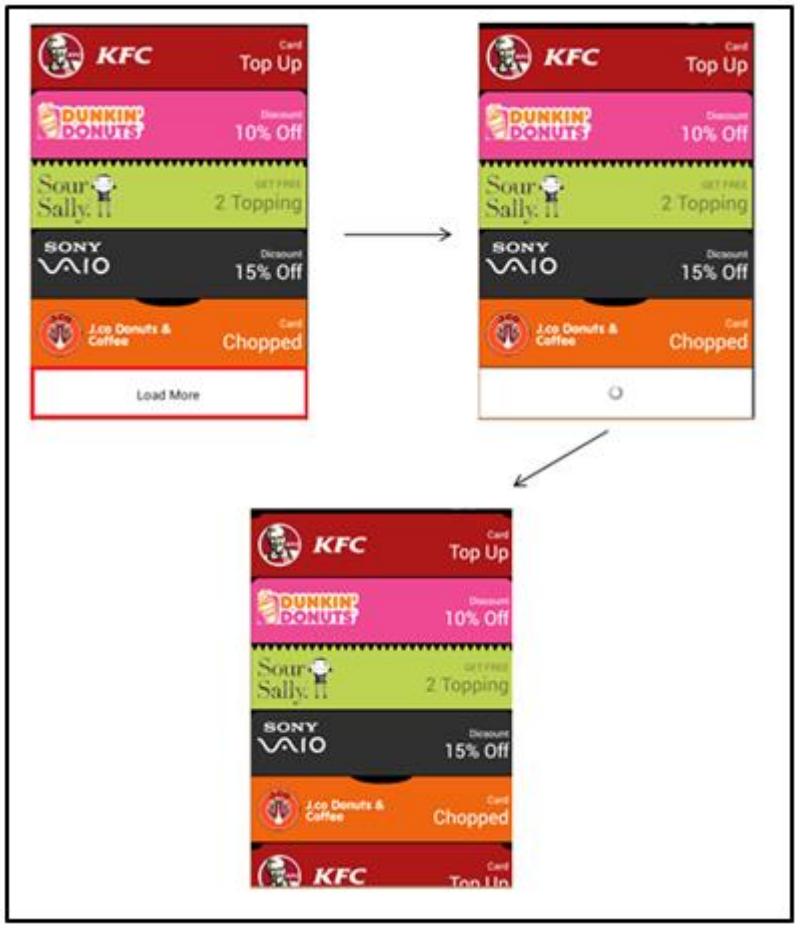
Gambar 3.17 *Design list* entiti kartu dan promo

Setelah *Layouting* selesai, maka yang selanjutnya dikerjakan adalah *Dynamic Content.* Pembuatan *Dynamic Content* ini membutuhkan *tool* yang dapat membuat isi secara *dynamic*. *Tool* ini berupa sebuah *library* yang dibuat oleh Joshua Prayogi Angki. Nama *tool* ini adalah *JSON to Backbone JS Mapper*. Cara

kerja dari *tool* ini adalah mengambil data dari *server* yang berupa *JSON String* yang kemudian akan diubah menjadi sebuah *JSON Object.*

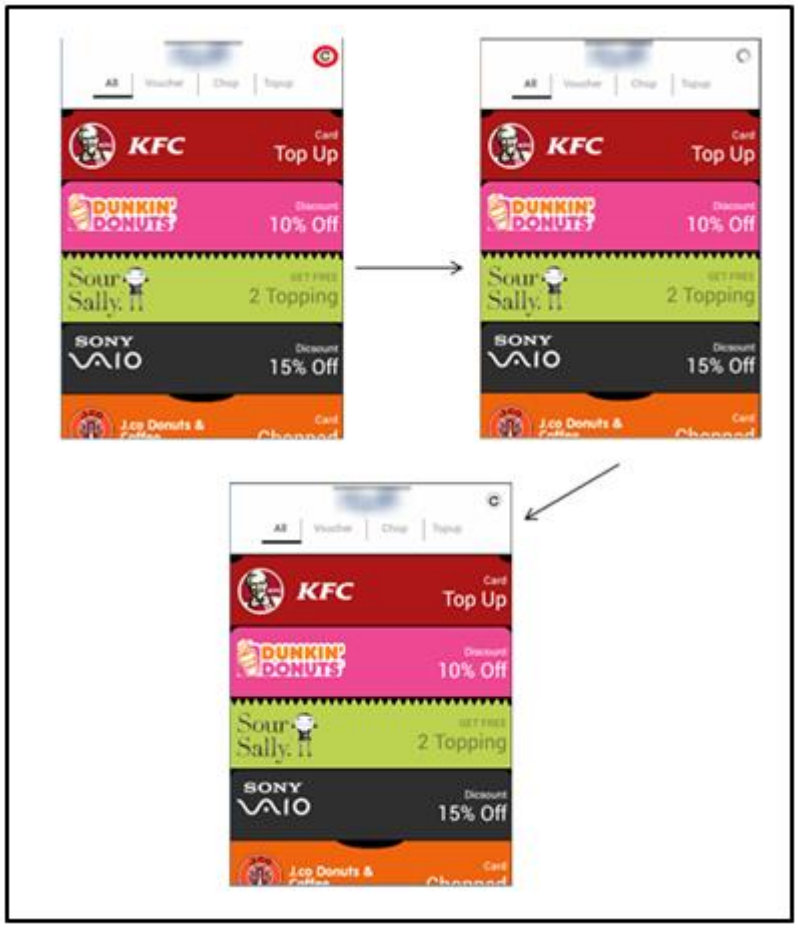
Gambar 3.18 Proses *dynamic content*

Pembuatan *Dynamic Content* ini juga berhubungan dengan pembuatan fungsi *Load More Content* dan *Refresh Content*. Pada awalnya pembuatan *Load More Content* dengan cara pada saat *list* sudah di *scroll* sampai paling bawah, jika masih ada *content* selanjutnya, maka akan muncul suatu *indicator loading* dan akan menjalankan proses *Load More*. Tetapi karena hal ini belum bisa dilakukan, maka diganti dengan cara membuat 1 baris di dalam *list* yang berfungsi sebagai *button* untuk fungsi *load more* tersebut.



Gambar 3.19 Proses *load more content*

Untuk *Refresh Content* pada awalnya akan dibuat *Pull to Refresh*, jadi ketika sudah berada pada bagian atas dari *list*, masih bisa di *scroll* ke atas untuk melakukan *refresh*. Ketika ditarik ke atas dan dilepas, maka fungsi *refresh* berjalan dan akan kembali ke bagian atas dari *list.* Tetapi karena fitur tersebut belum bisa dilakukan, maka fitur tersebut diganti dengan menggunakan sebuah *button*.



Gambar 3.20 Proses *refresh content*

Untuk bagian *Filter Content*, implementasinya hanya melakukan pengecekan apakah fungsi dari *Filter Content* ini sudah berjalan dengan benar.

3.3.7 Maintenance

Proses *maintenance* ini dilakukan setelah modul – modul yang dibuat telah selesai. Proses *maintenance* dilakukan untuk mengetahui apakah aplikasi yang dibuat tersebut sudah sesuai dengan yang diinginkan oleh pembimbing dan apakah ada *error* pada modul – modul tersebut. Jika masih ada error atau masih ada yang tidak sesuai, maka dilakukan lah proses ini agar modul – modul yang dibuat sesuai dengan yang diinginkan.

3.3.8 Penulisan Dokumentasi

Penulisan dokumentasi dilakukan pada tahap pengembangan, yaitu ketika dilakukan *coding* untuk aplikasi ini. Dokumentasi tersebut adalah *comment – comment* yang ada di dalam *coding* tersebut yang menjelaskan *detail* dari *coding* tersebut.

Untuk gambaran yang lebih jelas, akan ditampilkan *detail* tentang realisasi praktik kerja magang yang dilaksanakan selama bekerja di PT. Moonlay Technologies.

Table 3.2 Realisasi Kerja Magang

Minggu	Kegiatan
1	- Pengenalan dan pembentukan kelompok - Membuat *layout* dari Promo Builder
2	- Optimisasi *layout* dari Promo Builder - Membuat fungsi *knockout JS*
3	- Optimisasi *layout* dari Promo Builder - Pengenalan tentang Appcelerator Titanium Studio
4	- Membuat *layout* untuk Loyalty Express - Optimisasi *layout* Loyalty Express
5	- Membuat *prototype* untuk modul *Dynamic Content*
6	- Membuat fungsi untuk modul *Load More Content* - Membuat fungsi untuk modul *Refresh Content*
7	- Mengimplementasikan *tool JSON Mapper* ke dalam modul *Dynamic Content*
8	- Revisi dan penyelesaian modul – modul

3.4 Kendala yang Ditemukan

Dalam mengembangkan dan membuat suatu sistem atau aplikasi pasti akan ada beberapa kendala pada saat mengembangkan dan membuat sistem atau aplikasi tersebut. Begitu juga dalam membuat modul - modul ini, ada beberapa kendala yang ditemukan ketika membuat modul – modul tersebut. Beberapa kendala tersebut ditemukan dalam pengerjaan modul – modul baik kendala secara teknis maupun non teknis.

3.4.1 Kendala Teknis

Beberapa kendala teknis yang ditemukan dalam pembuatan modul – modul ini adalah:

1. Penggunaan Titanium Studio dengan Alloy *framework* yang baru pertama kali digunakan. Waktu yang dibutuhkan untuk membuat modul – modul menjadi lebih lama, karena pembuatannya dibarengi dengan pencarian informasi dan studi literatur yang dibutuhkan dalam pembuatan modul.
2. Pembuatan *layout* yang harus disesuaikan dengan *design* yang dibuat oleh *designer*. Posisi, gambar, tulisan, dan warna harus sesuai dengan *design* yang dibuat.
3. Pengambilan data dari *local database* prosesnya sangat lambat. Harusnya dalam sebuah aplikasi, ketika pengambilan data dari *local database* harus berjalan cepat dan juga ringan, sehingga performa dari aplikasi akan lebih baik.
4. Proses *compile* dan installasi modul ke *gadget* yang lambat, sehingga membuat waktu pengerjaan modul – modul berkurang.

3.4.2 Kendala Non Teknis

Beberapa kendala non teknis yang ditemukan dalam pembuatan modul – modul ini adalah:

1. *Design* yang sering berubah ketika proses pengerjaan modul *Layouting* selesai. Perubahan ini terjadi karena setelah *design* yang dibuat *designer* dibuat di dalam *gadget* Android tidak sesuai seperti yang diinginkan oleh pembimbing, sehingga membuat waktu pengerjaan menjadi lebih lama.
2. Keterbatasan *resource* yang disediakan oleh kantor. Dalam mengembangkan suatu *mobile application* tentunya membutuhkan *device* atau *gadget* yang berfungsi untuk melakukan *testing* aplikasi tersebut. Android *device* memiliki keanekaragaman resolusi, sehingga untuk pembuatan aplikasi haruslah disesuaikan dengan resolusi *device.*

3.5 Solusi atas Kendala yang Ditemukan

Dalam mengembangkan dan membuat suatu sistem atau aplikasi pasti akan ada beberapa kendala yang akan dihadapi, namun setiap kendala yang ada pasti memiliki solusi masing – masing.

3.5.1 Solusi untuk Kendala Teknis

Beberapa solusi yang dapat menyelesaikan kendala teknis yang ditemukan adalah sebagai berikut.

1. Membuat modul – modul dibarengi dengan studi literatur dan juga mencari contoh – contoh sesuai yang dibutuhkan dalam pembuatan modul.
2. Menganalisa *design* yang diberikan oleh *designer* terlebih dahulu dan menentukan *tool* apa saja yang dibutuhkan dalam membuat *layout* yang sesuai dengan *design* tersebut.
3. Solusi untuk kendala ini masih dicari terus – menerus, karena masih belum mendapatkan solusi yang benar – benar menyelesaikan kendala ini. Akan tetapi, sudah dicoba beberapa solusi, salah satunya adalah dengan tidak mengambil seluruh data sekaligus dari *local database* dan membuat fungsi *load more* untuk mengambil data – data yang belum ditampilkan.
4. Menggunakan *emulator* dapat mempercepat proses *compile*. Karena di dalam Titanium Studio ada satu proses yang bernama *fast dev*. Proses ini dapat mempercepat proses *compile.* Akan tetapi, proses ini hanya berjalan jika menggunakan *emulator*.

3.5.2 Solusi untuk Kendala Non Teknis

Beberapa solusi yang dapat menyelesaikan kendala non teknis yang ditemukan adalah sebagai berikut.

1. Analisa *design* yang diberikan oleh *designer*, kemudian juga diskusikan hasil analisa dengan pembimbing, jika sudah diterima oleh pembimbing, baru membuat *layout* tersebut. Kemudian ketika membuat *layout* tersebut, *install layout* tersebut dalam *gadget* dan sebelum menyelesaikan *layout* tersebut, tunjukkan kepada pembimbing. Setelah *layout* tersebut disetujui, baru lanjutkan *layout* tersebut sampai selesai.

2. *Gadget* yang diberikan kantor adalah *gadget* yang mempunyai resolusi 320 x 480 px. Untuk melakukan *testing* modul pada *gadget* yang memiliki resolusi selain *gadget* yang diberikan, maka digunakan *emulator* Android. *Emulator* tersebut dapat diubah resolusinya sesuai dengan yang ditentukan.