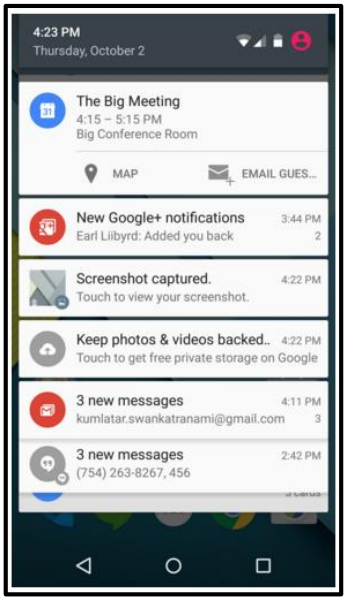
BAB II

LANDASAN TEORI

2.1 Android Push Notification

Android push notification adalah sebuah fitur pada *smartphone* berbasis Android yang dapat digunakan aplikasi untuk membantu memberitahu pengguna akan adanya informasi atau pesan baru meski pengguna sedang tidak menggunakan aplikasi (Parse, Tanpa Tahun). Dengan adanya *push notification*, pengguna dapat menerima notifikasi adanya pesan baru yang dikirim dari *server* ke perangkat pengguna (runrev.screenstepslive.com, 2013). Sebuah notifikasi sederhana dapat ditampilkan ke layar *smartphone* pengguna dengan bantuan beberapa *class* yang terdapat pada IDE Android Studio, yaitu NotificationManager, Notification, dan PendingIntent (Tutorialspoint, Tanpa Tahun). Gambar 2.1 menunjukkan contoh *push notification* pada *smartphone* berbasis Android.

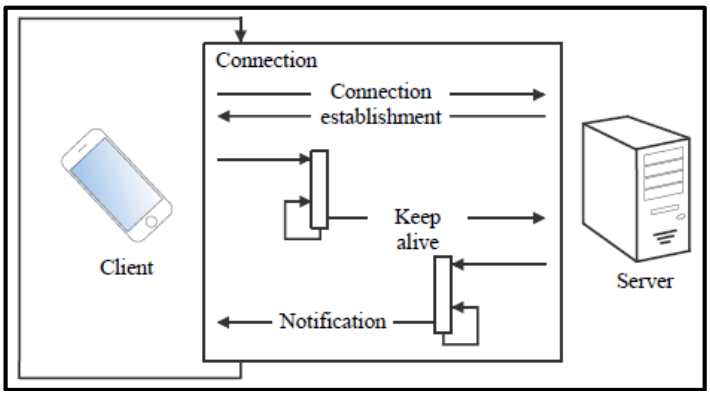


Gambar 2.1 Contoh Push Notification
(Sumber: http://www.tutorialspoint.com/android/android_push_notification.htm)

2.2 Push Technology

Push technology merupakan sebuah metode untuk mengotomasi pengiriman berita dan informasi ke komputer klien melalui internet (Umbach, 1997). Cara kerja metode ini adalah klien langganan terlebih dahulu ke sebuah *channel* melalui prosedur *handshake*, kemudian *server* akan mengirimkan data secara langsung ke klien apabila terdapat informasi baru (Mesbah dan Deursen, 2008). Penggunaan metode ini sering dijumpai pada sebuah aplikasi berita *online*, dimana pengguna dapat memilih kategori berita favoritnya agar aplikasi dapat menentukan berita apa akan yang dikirimkan secara otomatis ke perangkat pengguna.

Metode *push* membutuhkan dua *software* yang saling memahami satu sama lain. *Software* pertama berada pada klien yang mengerti bagaimana cara untuk melakukan *request*, menerima, menyimpan, hingga menampilkan data. *Software* kedua berada pada *server* yang mengerti *request* dari klien untuk kemudian dapat mengirimkan datanya (Umbach, 1997). Gambar 2.2 menggambarkan skema cara kerja metode *push*.



Gambar 2.2 Skema Cara Kerja Metode Push
(Sumber: Burgstahler dkk., 2013 : 2)

Penelitian yang dilakukan oleh Bozdag dkk. (2007) membandingkan metode *push* dan *pull* pada sebuah aplikasi berbasis Asynchronous JavaScript and XML

(AJAX). Untuk dapat membuat koneksi yang terus-menerus pada AJAX, ditambahkan sebuah protokol bernama BAYEUX untuk metode *push*. Hasil penelitian untuk metode *push* adalah koherensi data dan performa jaringan yang didapat lebih bagus dibanding metode *pull*. Namun, metode ini memiliki masalah skalabilitas apabila jumlah klien diperbesar. Beban pemrosesan pada *server* meningkat hingga tujuh kali ketika jumlah klien di atas 350.

Burgstahler dkk. (2013) melakukan penelitian dengan membandingkan metode *push* dan *pull* dari sisi pemakaian energi pada *smartphone*. Untuk mengimplementasi metode *push*, digunakan sebuah layanan untuk mendapatkan *push notification* dari Google bernama Google Cloud Messaging (GCM). GCM dipilih karena telah terintegrasi dengan perangkat uji untuk meminimalisir konsumsi energi.

Metode *push* memiliki kelebihan dibanding *pull*, yaitu data yang didapat akan lebih *fresh*, dimana pengguna akan dikirimkan notifikasi secara langsung ketika ada data baru (Hauswirth dan Jazayeri, 1999). Menurut penelitian yang dilakukan Bozdag, dkk. (2007), metode *push* cocok untuk diterapkan pada aplikasi yang membutuhkan koherensi data dan performa jaringan yang baik.

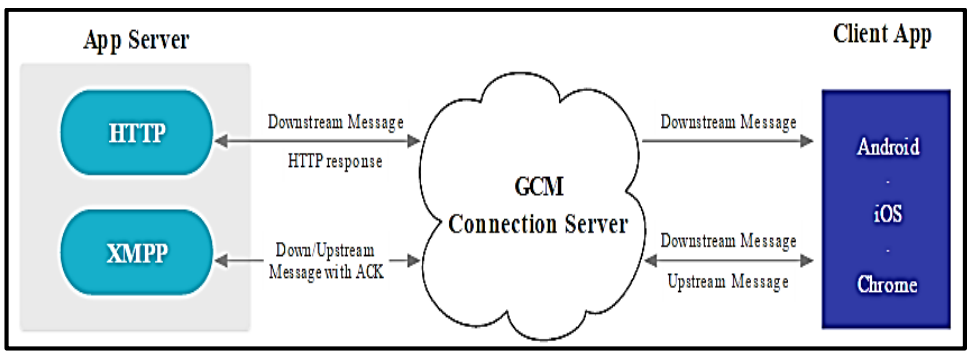
Kelemahan utama dari metode ini adalah masalah skalabilitas, dimana akan timbul masalah ketika jumlah pengguna diperbanyak. Penggunaan Central Prosessing Unit (CPU) dapat meningkat hingga tujuh kali ketika jumlah pengguna mencapai 350 (Bozdag dkk., 2007). Selain itu, metode ini memiliki masalah Single Point of Failure (SPoF) karena membutuhkan kondisi *server* yang harus selalu *online* (Hauswirth dan Jazayeri, 1999).

Metode *push* membutuhkan sebuah koneksi Transmission Control Protocol (TCP) antara klien dengan *server*. Koneksi TCP memiliki *timeout* dan harus dihindari karena apabila hal tersebut terjadi hubungan antara server dan klien akan terputus. Maka dari itu, sebagai *timeout handling*, sebuah pesan *heartbeat* harus dikirim secara periodik sebagai tanda bahwa klien masih aktif (Bahl dkk., 2012).

2.3 Google Cloud Messaging

Google Cloud Messaging (GCM) adalah sebuah layanan gratis yang dapat digunakan untuk membantu para pengembang aplikasi dalam pengiriman pesan pada lintas *platform*, seperti Android, iOS, dan Chrome (support.google.com, Tanpa Tahun). GCM terdapat pada setiap perangkat Android yang memiliki Google Play Services.

Terdapat tiga komponen untuk mengimplementasi GCM pada sebuah sistem. *Server* aplikasi yang bertugas sebagai penyimpan data sistem dan inisialisasi pengiriman pesan ke klien. GCM Connection Server yang bertugas untuk memberikan token registrasi untuk klien dan meneruskan pesan dari *server* ke klien. Aplikasi klien sebagai *front-end* dari sistem yang bertugas untuk menerima data dari GCM Connection Server. Gambar 2.3 merupakan arsitektur GCM.



Gambar 2.3 Arsitektur GCM
(Sumber: https://developers.google.com/cloud-messaging/gcm)

Untuk dapat memproses pesan dari *server* GCM, dibutuhkan dua *class* pada aplikasi, yaitu GcmReceiver dan GcmListenerService. GcmReceiver bertugas untuk menerima pesan dari *server* GCM kemudian mengirimkannya ke GcmListenerService untuk diproses (developers.google.com, 2012).

GCM membutuhkan sebuah koneksi antara klien dengan *server* yang dipertahankan dengan sebuah pesan *heartbeat* yang dikirimkan dari klien ke *server*. Pesan *heartbeat* ini memiliki interval 28 menit ketika menggunakan paket data dan 15 menit ketika menggunakan WiFi (Bocek dkk., 2015).

Dengan penggunaan GCM, klien tidak perlu melakukan *request* terus-menerus untuk mendapatkan data baru dari *server*, melainkan GCM *server* akan memberikan data secara langsung (*push*) ke *smartphone* yang telah teregistrasi ketika ada data baru. Dengan menghindari *request* yang terus-menerus ini, *smartphone* pengguna dapat lebih menghemat baterai (Rubio, 2012).

Untuk dapat mengirim atau menerima data, klien harus terdaftar pada *server* GCM. Berikut langkah-langkah yang harus dilakukan agar klien terdaftar pada *server* GCM (developers.google.com, 2012).

1. Aplikasi klien harus memiliki sebuah token registrasi yang didapat dari GCM Connection Server dengan menggunakan Instance ID Application Program Interface (API).
2. Aplikasi klien memberikan token registrasi ke aplikasi *server*.
3. Aplikasi *server* menyimpan token registrasi dan memberikan *acknowledgement* ke aplikasi klien sebagai tanda bahwa proses penyimpanan token telah berhasil.

Dalam pengiriman data, GCM memberikan batas ukuran maksimum data sebesar 2KB. Format data yang digunakan untuk komunikasi antara klien dengan *server* adalah JavaScript Object Notation (JSON). Gambar 2.4 merupakan contoh JSON yang akan dikirimkan dari server.

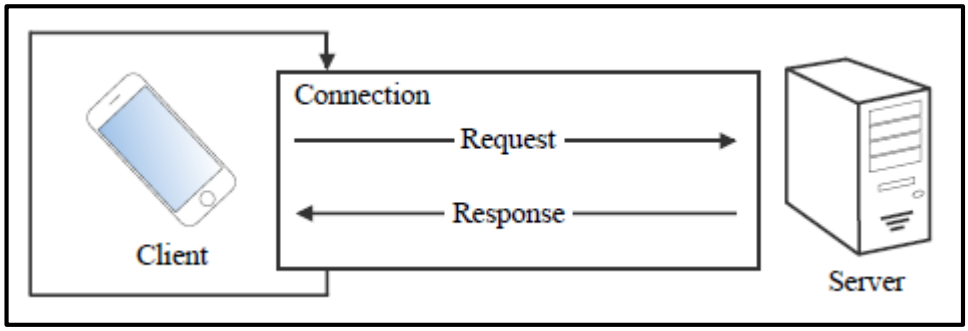
```
{
  "to" : "bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...",
  "notification" : {
    "body" : "great match!",
    "title" : "Portugal vs. Denmark",
    "icon" : "myicon"
  }
}
```

Gambar 2.4 Contoh Data Notifikasi GCM
(Sumber: https://developers.google.com/cloud-messaging/concept-options)

2.4 Pull Technology

Pull technology merupakan sebuah protokol yang mengharuskan individu untuk menghubungi *server website* untuk mendapatkan data atau informasi terbaru (Sia dkk., 2007). Prinsip dasar dari metode ini adalah pengiriman data yang dilakukan dari *server* ke klien akan dilakukan setelah klien melakukan *request* ke *server*.

Pull Technology sering dijumpai ketika *browsing*, yaitu pada saat pengguna "menarik" informasi dengan me-*request* laman tertentu sebelum mendapatkan data sebuah laman *web* (Umbach, 1997). Dengan memasukkan alamat *web*, pengguna atau klien melakukan *request* ke *server*. Setelah itu, *server* akan mengirim data laman *web* sesuai yang diminta oleh klien. Skema ini digambarkan pada Gambar 2.5.



Gambar 2.5 Skema Cara Kerja Metode Pull
(Sumber: Burgstahler dkk., 2013 : 2)

Penelitian yang dilakukan oleh Bozdag dkk. (2007) membandingkan metode *push* dengan sebuah protokol bernama BAYEUX dan *pull* pada sebuah aplikasi berbasis AJAX. Hasil dari penelitian ini adalah dengan metode *pull*, beban pemrosesan pada *server* cenderung rendah. Namun, untuk mendapatkan koherensi data dan performa jaringan yang bagus cukup susah. Hal ini dikarenakan apabila interval *pull* lebih tinggi dibanding interval penerbitan data baru maka beberapa data tidak akan diterima oleh klien. Metode *pull* cocok dipakai apabila interval *pull* sama dengan interval penerbitan data baru.

Burgstahler dkk. (2013) melakukan penelitian dengan membandingkan metode *push* dan *pull* dari sisi pemakaian energi pada *smartphone*. Hasil yang didapat dari penelitian ini adalah ukuran data yang dikirim tidak berpengaruh pada konsumsi daya, tetapi interval perubahan data baru berpengaruh. Dengan interval 1200 detik, perbedaan konsumsi energi dari kedua metode dapat terlihat dengan jelas, yaitu metode *pull* memiliki konsumsi energi 19% lebih rendah dibanding metode *push*.

Metode *pull* memiliki kelebihan dibanding metode *push*, di antaranya adalah kemudahannya untuk diimplementasikan, memungkinkan untuk beroperasi secara

offline, dan menunjukkan performa yang baik ketika jumlah *subscriber* tinggi. Jumlah *subscriber* yang dimaksud tinggi adalah satu juta *subscriber* (Hauswirth dan Jazayeri, 1999). Dengan jumlah tersebut namun menunjukkan performa yang baik merupakan salah satu keunggulan metode *pull* ketika diimplementasikan pada arsitektur sebuah sistem.

Di balik kelebihan-kelebihan yang dimilikinya, metode *pull* memiliki kekurangan, yaitu untuk mendapatkan data baru secara *real-time*, frekuensi antar *request* yang dilakukan klien harus tinggi. Hal ini mengakibatkan lalu lintas jaringan yang padat, *request* yang tidak efektif, dan mengurangi tingkat responsif aplikasi (Bozdag dkk., 2007). *Request* yang tidak efektif yang dimaksud adalah *request* yang seharusnya tidak dilakukan karena data pada *server* tidak ada yang baru. Sebaliknya, apabila frekuensi antar *request* dilakukan dengan jeda yang panjang akan mengakibatkan data yang didapat klien tidak *real-time*.

2.5 GCM Network Manager

GcmNetworkManager merupakan sebuah API yang memungkinkan aplikasi seperti menggunakan JobScheduler pada versi sebelum Lollipop, namun hanya diperuntukkan secara khusus untuk menjalankan pekerjaan-pekerjaan yang berhubungan dengan jaringan. Untuk menggunakan GcmNetworkManager, perangkat pengguna harus memiliki Google Play Service dengan versi lebih dari 7.5 (Prochazka, 2015).

Dengan adanya *class* GcmNetworkManager memungkinkan para pengembang untuk membuat aplikasi yang membutuhkan suatu pekerjaan untuk dijadwalkan di masa depan dengan penggunaan baterai yang efisien. Selain itu

terdapat beberapa kustomisasi yang dapat dilakukan dalam penggunaan *class* ini, seperti menentukan pekerjaan berdasar status jaringan dan pengisian baterai (*charging*) (Sillars, 2015).

Terdapat enam kelebihan yang dimiliki GcmNetworkManager (developers.google.com, 2012). Kelebihan pertamanya adalah dapat menjadwalkan satu pekerjaan atau pekerjaan yang berulang-ulang secara periodik. Kelebihan kedua, memberikan pesan keberhasilan pekerjaan dan dapat melakukan pekerjaan ulang (*retry*) secara otomatis apabila gagal. Kelebihan ketiga, GcmNetworkManager mampu untuk mendeteksi status jaringan perangkat. Hal ini dimanfaatkan untuk mendapatkan waktu yang optimal untuk melakukan pekerjaan yang dijadwalkan. Kelebihan keempat adalah kemampuan untuk mendeteksi status pengisian *smartphone*. Kemampuan ini dapat dimanfaatkan untuk meminimalisir energi yang digunakan untuk melakukan suatu pekerjaan. Kelebihan kelima adalah pekerjaan yang telah dijadwalkan tetap ada meski *smartphone* di-*restart*. Kelebihan terakhir adalah kemudahannya untuk diimplementasi.

Prochazka (2015) menulis sebuah artikel tutorial untuk membantu para pengembang aplikasi dalam menerapkan penjadwalan dengan menggunakan GcmNetworkManager. Berdasarkan tutorial tersebut, dibutuhkan sebuah *service* yang meng-*inherit class* GcmTaskService untuk dapat menjalankan pekerjaan menjadi *background process*. *Service* tidak dapat hidup dengan sendirinya, dibutuhkan sebuah *activity* untuk menjalankan *service* tersebut. Pada *service* inilah *class* GcmNetworkManager untuk melakukan penjadwalan digunakan.

2.6 Click Ratio

Untuk mendapatkan bagaimana reaksi pengguna atas suatu notifikasi yang didapat, dilakukan pengukuran untuk *click times* dan *click time* (Shirazi, 2014). Penelitian yang dilakukan oleh Shirazi dkk. (2014) mengukur jumlah klik (*click times*) untuk notifikasi pada setiap kategori aplikasi *mobile*. Hasil penelitian yang didapat adalah aplikasi dengan kategori *messenger* dan *mail* memiliki *click times* yang tertinggi. Penelitian ini masih terbatas karena setiap kategori aplikasi mengirimkan jumlah notifikasi yang berbeda, sehingga didapat *click times* untuk *messenger* jauh lebih tinggi dibandingkan *calendar*.

Pada penelitian ini, metode *push* dan *pull* akan memiliki jumlah notifikasi yang berbeda, sehingga untuk mengatasi batasan penelitian yang dilakukan oleh Shirazi dkk. (2014), pada penelitian ini digunakan *click ratio* untuk notifikasi yang didapat pada setiap metode. *Click ratio* dihitung dengan membagi jumlah klik pada notifikasi dengan jumlah notifikasi yang didapat untuk setiap metode. *Click ratio* pada setiap metode akan dibagi per kategori untuk didapatkan kategori yang paling diminati oleh pengguna.

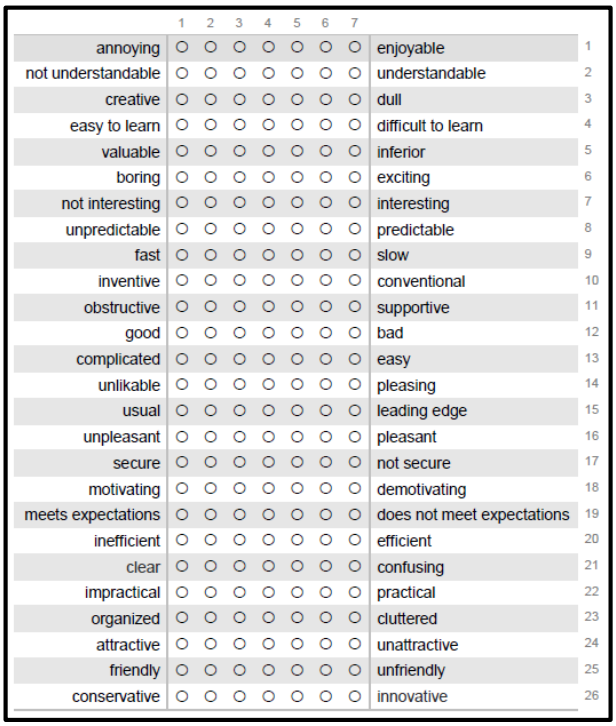
2.7 Click Time

Shirazi dkk. (2014) melakukan penelitian dengan mengukur *click time* per kategori. *Click time* merupakan selisih waktu antara notifikasi didapat dan notifikasi diklik oleh pengguna. Hasil penelitian yang didapat adalah kategori *system* memiliki *click time* tercepat, diikuti dengan kategori *messenger*. Kesimpulan yang didapat mengenai *click time* adalah 50% interaksi pengguna dengan notifikasi terjadi pada selang waktu tiga puluh detik setelah notifikasi didapat.

Pada penelitian ini akan diukur *click time* untuk setiap metode. Penghitungan *click time* didasari oleh penelitian yang dilakukan Shirazi dkk. (2014), yaitu dengan menghitung selisih waktu antara notifikasi didapat dan notifikasi diklik oleh pengguna untuk *click time* satu notifikasi. Selanjutnya jumlah *click time* untuk satu metode akan dibagi dengan jumlah notifikasi yang didapat pada satu metode tersebut untuk mendapatkan *click time* satu metode.

2.8 User Experience Questionnaire

Pengalaman pengguna adalah pengalaman menyeluruh yang terdiri dari setiap aspek dari interaksi pengguna terhadap suatu produk atau layanan (Park dkk., 2013). User Experience Questionnaire (UEQ) merupakan sebuah penilaian yang dapat dilakukan secara cepat untuk mendapatkan pengalaman pengguna dari sebuah produk interaktif (UEQ-Online, 2015). Contoh UEQ dapat dilihat pada Gambar 2.6.



	1	2	3	4	5	6	7		
annoying	○	○	○	○	○	○	○	enjoyable	1
not understandable	○	○	○	○	○	○	○	understandable	2
creative	○	○	○	○	○	○	○	dull	3
easy to learn	○	○	○	○	○	○	○	difficult to learn	4
valuable	○	○	○	○	○	○	○	inferior	5
boring	○	○	○	○	○	○	○	exciting	6
not interesting	○	○	○	○	○	○	○	interesting	7
unpredictable	○	○	○	○	○	○	○	predictable	8
fast	○	○	○	○	○	○	○	slow	9
inventive	○	○	○	○	○	○	○	conventional	10
obstructive	○	○	○	○	○	○	○	supportive	11
good	○	○	○	○	○	○	○	bad	12
complicated	○	○	○	○	○	○	○	easy	13
unlikable	○	○	○	○	○	○	○	pleasing	14
usual	○	○	○	○	○	○	○	leading edge	15
unpleasant	○	○	○	○	○	○	○	pleasant	16
secure	○	○	○	○	○	○	○	not secure	17
motivating	○	○	○	○	○	○	○	demotivating	18
meets expectations	○	○	○	○	○	○	○	does not meet expectations	19
inefficient	○	○	○	○	○	○	○	efficient	20
clear	○	○	○	○	○	○	○	confusing	21
impractical	○	○	○	○	○	○	○	practical	22
organized	○	○	○	○	○	○	○	cluttered	23
attractive	○	○	○	○	○	○	○	unattractive	24
friendly	○	○	○	○	○	○	○	unfriendly	25
conservative	○	○	○	○	○	○	○	innovative	26

Gambar 2.6 Contoh UEQ
(Sumber: http://www.ueq-online.org)

Pada penelitian ini akan dibandingkan hasil UEQ untuk metode *push* dan *pull* untuk didapatkan metode yang memiliki pengalaman pengguna terbaik. Terdapat enam skala yang dituangkan dalam 26 butir penilaian untuk mendapatkan hasil UEQ. Keenam skala tersebut adalah *Attractiveness*, *Perspicuity*, *Efficiency*, *Dependability*, *Stimulation*, dan *Novelty* (Schrepp dkk., 2014).

Attractiveness merupakan kesan terhadap produk, apakah pengguna menyukainya atau tidak. Terdapat enam butir untuk skala ini, yaitu *annoying* / *enjoyable*, *good* / *bad*, *unlikable* / *pleasing*, *unpleasant* / *pleasant*, *attractive* / *unattractive*, *friendly* / *unfriendly*. *Perspicuity* merupakan kemudahan untuk menjadi terbiasa terhadap produk. Terdapat empat butir untuk skala ini, yaitu *not understandable* / *understandable*, *easy to learn* / *difficult to learn*, *complicated* / *easy*, *clear* / *confusing*.

Efficiency merupakan kemudahan pengguna untuk menyelesaikan pekerjaan dengan mudah. Terdapat empat butir untuk skala ini, yaitu *fast* / *slow*, *inefficient* / *efficient*, *impractical* / *practical*, *organized* / *cluttered*. *Dependability* merupakan skala untuk mengukur bagaimana perasaan pengguna terhadap sistem, apakah pengguna merasa berinteraksi dengan sistem atau tidak. Terdapat empat butir untuk skala ini, yaitu *unpredictable* / *predictable*, *obstructive* / *supportive*, *secure* / *not secure*, *meets expectations* / *does not meet expectations*.

Stimulation merupakan skala untuk mengukur tingkat ketertarikan pengguna terhadap produk. Terdapat empat butir untuk skala ini, yaitu *valuable* / *inferior*, *boring* / *exciting*, *not interesting* / *interesting*, *motivating* / *demotivating*. *Novelty* merupakan pandangan pengguna mengenai tingkat kreativitas produk. Terdapat

empat butir untuk skala ini, yaitu *creative* / *dull*, *inventive* / *conventional*, *usual* / *leading edge*, *conservative* / *innovative*.

Menurut panduan UEQ (Schrepp, 2015), UEQ menggunakan skala 1-7 untuk setiap butir penilaian dengan nilai -3 untuk pernyataan negatif hingga +3 untuk pernyataan positif. Setelah responden selesai mengisi kuisioner, penghitungan dilakukan dengan menghitung nilai rata-rata untuk setiap skala. Apabila nilai rata-rata yang didapat berada diantara -0.8 hingga 0.8 maka skala tersebut dianggap netral. Apabila rata-rata berada di bawah -0.8 maka skala tersebut dianggap jelek dan berlaku sebaliknya apabila rata-rata berada di atas 0.8.

2.9 Hands-on Measurement

Hands-on measurement merupakan sebuah metode yang didesain untuk mengukur kegunaan aplikasi *mobile* secara kuantitas (Nayebi, 2012). Dalam pengembangan sebuah aplikasi, metode ini dapat digabungkan bersama dengan *field studies*. Menurut Nayebi (2012), metode ini cocok digunakan apabila aspek-aspek yang ingin diukur pada aplikasi *mobile* telah jelas, sehingga dapat membuat peneliti untuk mendapatkan data-data untuk perbandingan menjadi lebih akurat.

2.10 Field Studies

Field studies merupakan metode yang umum digunakan untuk mengumpulkan data-data mengenai pengguna, keinginan pengguna, dan kebutuhan produk (Nayebi, 2012). Pada metode ini partisipan atau pengguna dilibatkan secara langsung dengan melakukan pekerjaan-pekerjaan yang mungkin dapat terjadi pada lingkungan yang sebenarnya. Pada makalah Nayebi (2012) disebutkan bahwa

beberapa penelitian lain menyarankan untuk memberikan kuisioner kepada partisipan pada akhir periode pengujian.