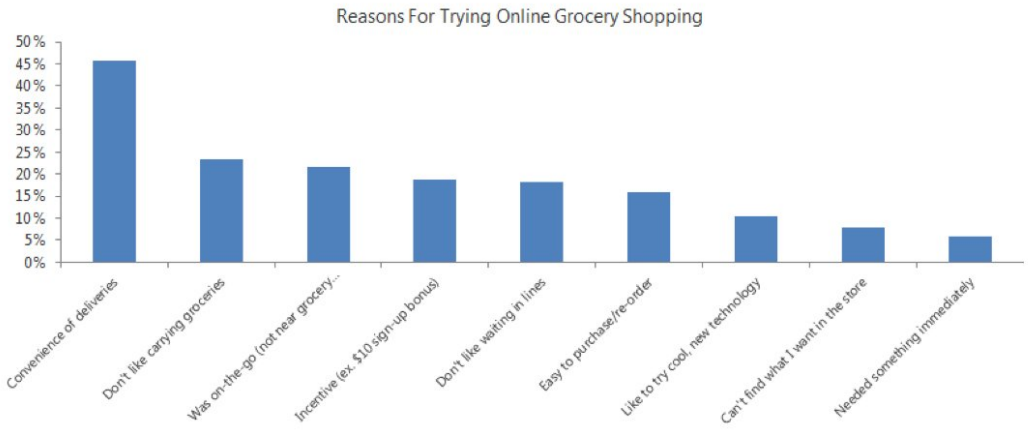
BAB II
LANDASAN TEORI

II.1 On Demand Service

Ekonomi *On-Demand* didefinisikan sebagai kegiatan ekonomi yang diciptakan oleh perusahaan teknologi yang memenuhi permintaan konsumen melalui penyediaan langsung barang dan jasa.

Berdasarkan survei pada 250 pelanggan Whole Foods and Trader Joe yang dilakukan The On-Demand Ekonomi menyoroti bahwa "kenyamanan pengiriman" adalah penentu besar dalam keputusan konsumen untuk membeli bahan makanan *online*.



Gambar II.1 Hasil Survei The On-Demand Ekonomi

Penelitian tersebut menunjukkan bahwa pertumbuhan perekonomian *on-demand* adalah produk dari keselarasan selera konsumen untuk pelayanan yang lebih nyaman, cepat, dan sederhana. [2]

Kunci penting adalah memahami kebutuhan pelanggan, apakah lebih bersifat *urgent*/instan atau mau menerima jadwal yang lebih fleksibel. [3]

II.2 Sistem Operasi Android

Android adalah sistem operasi untuk telepon seluler yang berbasis *Linux*. *Android* menyediakan platform yang bersifat *open source* bagi para pengembang untuk menciptakan sebuah aplikasi. Awalnya, Google Inc. mengakuisi Android Inc. Yang mengembangkan *software* untuk *ponsel* yang berada di Palo Alto, California

Amerika Serikat. Kemudian untuk mengembangkan Android, dibentuklah *Open Handset* Alliance, yaitu konsorsium dari 34 perusahaan *hardware, software,* dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Telepon pertama yang memakai sistem operasi *Android* adalah HTC Dream, yang dirilis pada 22 Oktober 2008. Pada penghujung tahun 2009 diperkirakan di dunia ini paling sedikit terdapat 18 jenis. Dari segi arsitektur sistem, *Android* merupakan sekumpulan *framework* dan *virtual machine* yang berjalan di atas *kernel linux*. *Virtual machine Android* bernama Dalvik *Virtual Machine* (DVM), *engine* ini berfungsi untuk menginterpretasikan dan menghubungkan seluruh kode mesin yang digunakan oleh setiap aplikasi dengan *kernel linux*. Sementara untuk *framework* aplikasi sebagian besar dikembangkan oleh google dan sebagian yang lain dikembangkan oleh pihak ketiga (developer).

Beberapa *framework* yang dikembangkan oleh Android sendiri misalnya fungsi untuk telepon seperti panggilan telepon, sms, *video call*. Untuk *browser Android* menggunakan google chrome. Aplikasi yang dikembangkan *Android* dibuat dengan menggunakan kode Java seperti halnya J2ME yang telah lama digunakan pada platform perangkat seluler umumnya. Namun secara siklus program memiliki perbedaan mendasar antara J2ME dengan Java yang ada pada *Android*. Kode Java pada *Android* lebih dekat dengan J2SE. hingga saat ini *Android* telah banyak digunakan pada produk *smartphone* seperti Samsung, LG, SonyErricson, Nexian dan juga HTC.

Dengan dukungan *Software Development Kit* (SDK) dan *Application Programming Interface* (API) dari google memberikan kemudahan bagi pihak ketiga (developer) untuk membangun aplikasi yang dapat berjalan pada sistem operasi *Android*. Selain itu terdapat metode baru dalam mengembangkan aplikasi di dalam sistem operasi *Android* menggunakan *Native Development Kit* (NDK). NDK ini memungkinkan developer untuk mengembangkan aplikasi di dalam sistem operasi *Android* menggunakan bahasa pemrograman C atau C++ [4]

II.2.1 Android Life Cycle

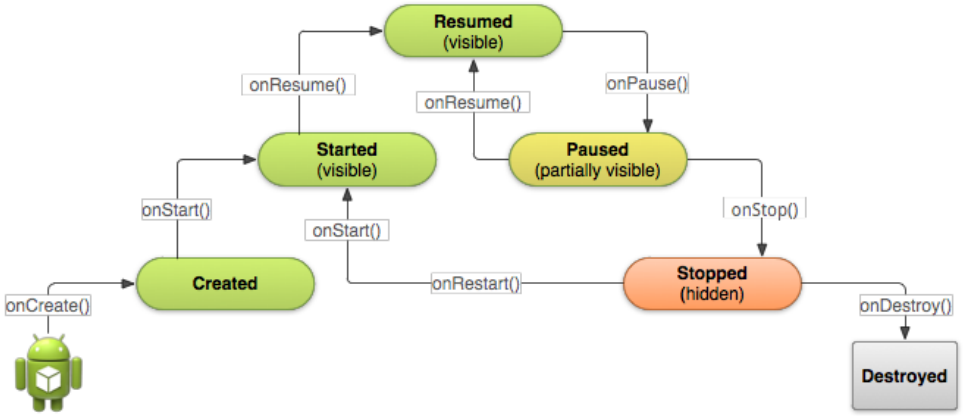
Aplikasi *android* terdiri dari beberapa fungsi dasar seperti mengedit catatan, memutar *file* musik, membunyikan alarm, atau membuka kontak telepon. Fungsi fungsi tersebut dapat diklasifikasikan ke dalam empat komponen android yang berbeda seperti ditunjukkan pada Tabel II.1, klasifikasi tersebut berdasarkan kelas-kelas dasar *java* yang digunakan.

Tabel II.1 Tabel komponen aplikasi android

Fungsi	Kelas dasar java	Contoh
Berfokus pada yang pengguna dapat lakukan.	Activity	Mengedit note, bermain game.
Background process	Services	Mainkan musik, membaharui icon cuaca
Menerima pesan	BroadcastReceiver	Memicu alarm karena sebuah event
Menyimpan dan mencari data	ContentProvider	Membuka kontak telepon

Setiap aplikasi pasti menggunakan minimal satu dari komponen tersebut, akan tetapi terdapat beberapa komponen yang mengharuskan mencantumkan *specified permission* sebelum digunakan seperti komponen Service, *Broadcast Receiver* dan *Content Provider*.

Android memiliki paradigma pemrograman lain tidak seperti paradigma pemrograman biasa di mana aplikasi yang dijalankan pada fungsi main(), sistem *android* menjalankan kode dalam *method Activity* dengan menerapkan metode *callback* tertentu yang sesuai dengan tahap tertentu dari siklus hidup. Setiap aplikasi yang berjalan dalam sistem operasi android memiliki siklus hidup yang berbeda dengan aplikasi *desktop* atau *web*, Hal ini dikarenakan aplikasi *mobile* memiliki tingkat interupsi proses yang lumayan tinggi seperti ketika *handling* panggilan masuk aplikasi diharuskan menghentikan proses sementara, Penerapan siklus hidup juga berguna untuk memastikan aplikasi tidak menghabiskan sumber daya baterai pengguna.



Gambar II.2 Siklus hidup android

Terdapat beberapa *state* dalam siklus hidup *android* yang terjadi seperti diilustrasikan pada Gambar II.2, akan tetapi hanya beberapa dari *state* tersebut yang menjadi statis diantaranya:

1. Resumed

Resumed terjadi ketika aplikasi berjalan setelah state *paused* . State ini akan menjalankan perintah program yang ditulis pada method onResume().

2. Paused

Dalam keadaan ini aktivitas yang terjadi dihentikan secara sementara tetapi masih terlihat oleh pengguna karena terdapat proses yang memiliki prioritas lebih tinggi seperti panggilan telepon. Aplikasi tidak dapat menjalankan perintah apapun ataupun menampilkan apapun dalam state ini.

3. Stopped

Dalam keadaan ini, aplikasi benar-benar tidak ditampilkan dan tidak terlihat oleh pengguna tetapi masih meninggalkan service di background.

State lain seperti *Created* dan *Started* bersifat sementara dan sistem dengan cepat menjalankan state berikutnya dengan memanggil metode *life cycle callback* berikutnya. Artinya, setelah sistem *OnCreate*() dipanggil, dengan cepat sistem akan memanggil *method OnStart*(), kemudian diikuti oleh *onResume*().

II.3 Android SDK (Software Development Kit)

Sesuai dengan namanya merupakan kumpulan *tools-tools* yang di perlukan untuk mengembangkan aplikasi Android. Seperti pada java yang dapat dijalankan pada *multiplatform*, SDK Android pun tersedia untuk beberapa platform seperti Windows, Mac OS dan Linux [5].

Adapun aplikasi SDK yang di dalamnya terdapat beberapa folder sebagai berikut :

1. Tools

Berisikan *tools-tools* yang diperlukan dalam membangun aplikasi Android seperti *emulator*, Android Virtual Device (AVD), SDK *Manager*, ddms dan banyak lagi untuk memudahkan membangun aplikasi *Android*. Semua tools ini dapat di update mengikuti setiap perkembangan *platfom Android*.

2. Platforms

Direktori ini *ditujukkan* untuk meletakkan versi platform-platform Android yang telah kita *download* dari SDK *Manager*. Seperti *Android* 1.5, 1.6, 2.0 dan seterusnya.

3. Add-ons

Direktori ini nantinya berisikan *tools* tambahan untuk Android SDK, yang fungsinya untuk menambahkan *library* yang akan digunakan untuk pengembangan aplikasi. Seperti jika ingin mengembangkan aplikasi menggunakan map dari google maka harus memerlukan platform Android yang sudah terdapat library map. Semua ini dapat di download melalui SDK *Manager*.

Selain folder-folder diatas, akan terdapat file tambahan jika kita mendownload beberapa platfom dan tool dalam folder Android SDK tersebut. Seperti *Docs, platform tools,* dan *samples*.

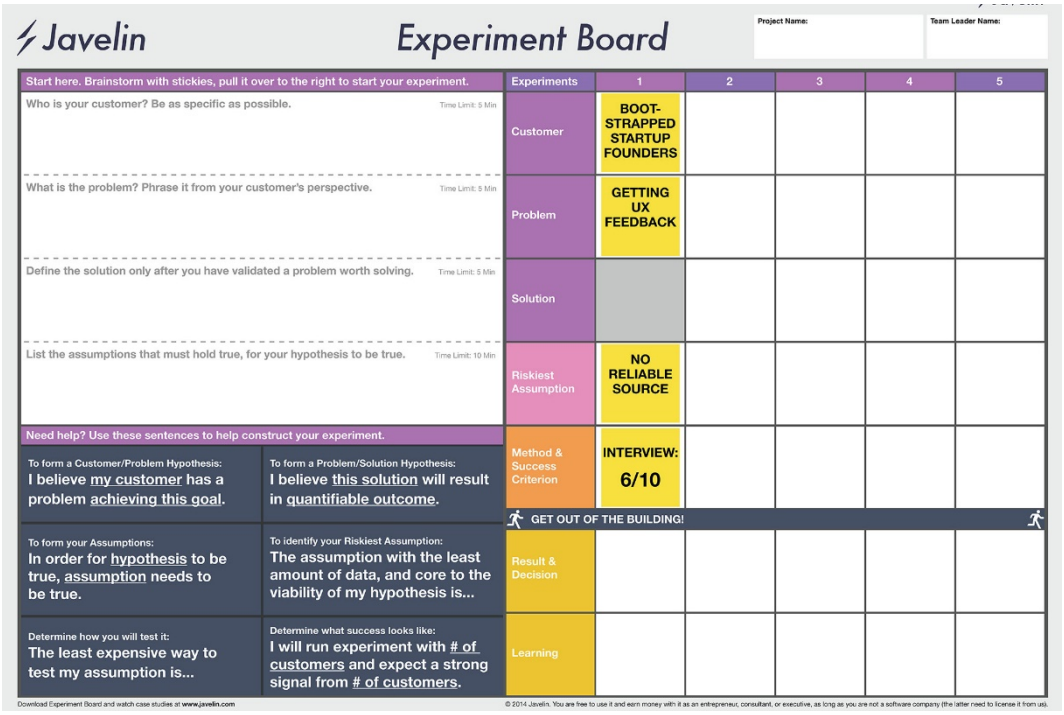
II.4 Javelin Experiment Board

Javelin experiment board adalah pendekatan dari Lean Startup Machine untuk memvalidai ide / gagasan. Dasar untuk Javelin Experiment Board adalah membentuk sebuah hipotesis. Hipotesis adalah menciptakan sebuah gagasan

dengan cara yang mudah untuk diuji. Karena sebagian besar pengusaha memiliki gagasan yang sangat besar sehingga kita perlu memecahnya untuk mengujinya.

Hipotesis dipecah dalam tiga jenis:

- Pelanggan
- Masalah
- Solusi



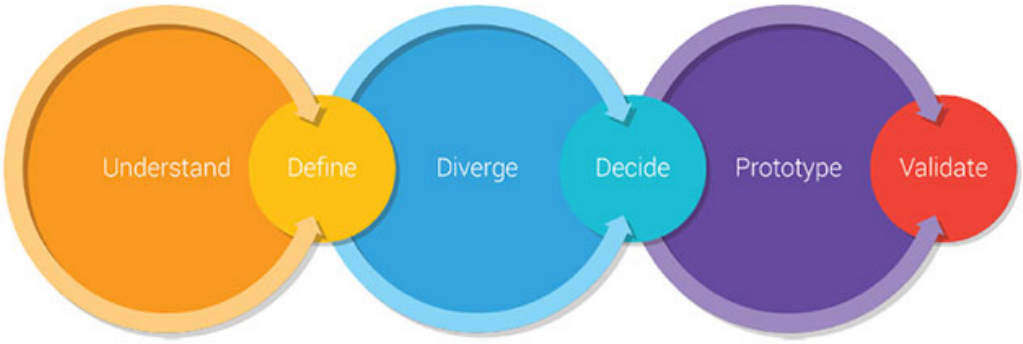
Gambar II.3 Javelin Experiment Board

Pada javelin experiment board, di sisi kanan area brainstorming di sisi kiri adalah area excute-nya. Yang perlu didefinisikan terlebih dahulu adalah :

- Siapa pelanggan anda Jelaskan sespesifik mungkin.
- Apa masalahnya? Dilihat dari perspektif pelanggan Anda.
- Tentukan solusinya setelah memvalidasi masalah yang layak dipecahkan.
- Cantumkan asumsi yang harus benar, agar hipotesis itu benar. [6]

II.5 Google Design Sprint

Design sprint adalah kerangka kerja bagi tim dengan ukuran apa pun untuk memecahkan dan menguji masalah desain dalam 2-5 hari. Ide sprint berasal dari kerangka Agile. Gagasan tentang pemikiran desain dikembangkan di IDEO dan d.school di Stanford.



Gambar II.4 Google Design Sprint

1. Understand

   Mendefinisikan kebutuhan pengguna, kebutuhan bisnis dan kapasitas teknologi. Metode yang dilakukan pada tahapan ini dapat berupa wawancara / interview, competitive overview, field visit, dan stakeholder map.
2. Define

   Pada tahap ini adalah membagi gagasan menjadi kategori yang penting dan strategi yang dapat dilakukan.
3. Diverge

   Pada tahap ini dibuat perancangan antarmuka yang sesuai dengan tahapan sebelumnya
4. Decide

   Tahapan memilih perancangan antarmuka yang paling baik.
5. Prototype

   Pembuatan prototype yang dapat diuji oleh pengguna.
6. Validate

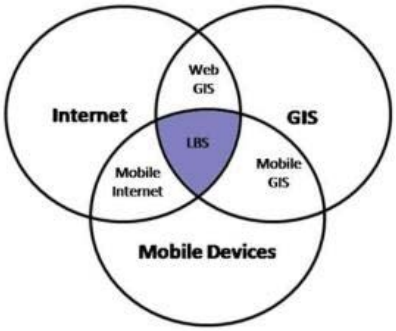
   Menguji prototype yang sudah dibuat kepada calon pengguna.

II.6 GPS

Global Positioning System (GPS) merupakan sistem navigasi yang menggunakan satelit dalam penggunaannya. GPS terdiri dari tiga bagian yaitu sistem kontrol, satelit dan pengguna. Sistem kontrol adalah bagian yang mengontrol pergerakan satelit – satelit yang ada dan saling berinteraksi satu sama lain kemudian pengguna adalah alat navigasi yang digunakan seperti perangkat mobile yang kini sudah memiliki fitur GPS di dalamnya. GPS biasanya digunakan untuk menunjukan suatu lokasi yang berada di permukaan bumi dengan tingkat akurasi yang cukup baik yaitu kurang dari 10 meter selama tidak ada benda padat yang dapat menghambat sinyal untuk mendapatkan lokasi pengguna. [7]

II.7 Location Based Services

Location Based Service (LBS) atau layanan berbasis lokasi adalah layanan informasi yang dapat diakses melalui *mobile device* dengan menggunakan mobile network, yang dilengkapi dengan kemampuan untuk memanfaatkan lokasi dari *mobile device* tersebut. LBS memungkinkan untuk melakukan komunikasi dan interaksi dua arah. LBS ini dapat digambarkan sebagai suatu layanan yang berada pada pertemuan tiga teknologi yaitu *Geographic Information System* (GIS), *Internet Service* dan *mobile device*, seperti pada Gambar II.5. [7]



Gambar II.5 Teknologi Location Based Service

II.8 Fabric

Fabric adalah sebuah *Software Developer Kit* (SDK) yang diluncurkan pada tahun 2014 oleh Twitter. Fabric digunakan untuk mempermudah para developer menghadapi permasalahan yang dialami dalam mengembangkan aplikasi yaitu

stabilitas, distribusi, pendapatan, dan identitas. SDK ini terdiri dari tiga bagian dan satu fitur tambahan yaitu Crashlytic Kit, Twitter kit, MoPub Kit, dan DIGITS. Fabric dapat di lihat di https://fabric.io/kits

II.8.1 Crashlytic

Crashlytic dibuat untuk bertujuan untuk mendeteksi, mengakses, dan memperbaiki masalah dalam aplikasi. Karena Crashlytic akan menganalisa secara otomatis dan melaporkannya sehingga developer dapat lebih fokus untuk membuat aplikasi berkualitas. Crashlytic dapat dilihat pada https://fabric.io/kits/android/crashlytics

II.8.2 Digits

Digits merupakan proyek ambisius Twitter untuk mengganti sistem password yang selama ini digunakan. Layaknya *two-step-verification*, Twitter memanfaatkan nomor handphone sebagai media verifikasi, sehingga tidak perlu mengingat password lagi. Digit memiliki tiga tahap: pertama menampilkan opsi untuk login melalui perangkat mobile, selanjutnya tampilan untuk memasukkan nomor handphone, dan terakhir adalah tampilan untuk memasukkan kode konfirmasi yang dikirim Twitter melalui SMS.

Digits memungkinkan pengguna membuat akun atau masuk ke aplikasi tanpa menggunakan apa pun kecuali nomor telepon mereka di iOS dan Android. Digits akan memverifikasi nomor telepon pengguna dengan antarmuka pengguna yang mudah disesuaikan yang mudah digabungkan ke dalam aplikasi. [8]
Digits dapat dilihat pada https://fabric.io/kits/android/digits

II.9 Firebase

Firebase adalah platform pengembangan aplikasi mobile dan web. Firebase terdiri dari fitur pelengkap yang bisa dicampur dan cocok untuk disesuaikan dengan kebutuhan mereka. Tim ini berbasis di San Francisco dan Mountain View, California. Perusahaan ini didirikan pada tahun 2011 oleh Andrew Lee dan James Tamplin. [9]

Layanan Firebase yang digunakan pada penelitian ini :

1. Database Realtime

Firebase menyediakan database realtime dan backend sebagai layanan. Layanan ini menyediakan pengembang aplikasi API yang memungkinkan data aplikasi disinkronkan di seluruh klien dan disimpan di penyimpanan cloud. [10]

2. Firebase Cloud Messaging

Sebelumnya dikenal sebagai Google Cloud Messaging (GCM), Firebase Cloud Messaging (FCM) adalah solusi lintas platform untuk pesan dan notifikasi untuk aplikasi Android, iOS, dan aplikasi web. [11]

II.10 Google Maps

Google Maps adalah sebuah layanan gratis dari Google tentang berbagai macam teknologi pemetaan. Google Maps menampilkan peta dunia secara digital sehingga lebih memudahkan ketika ingin mengetahui suatu lokasi tertentu. Google Maps menyediakan beberapa keunggulan seperti peta jalan, menunjukan rute untuk suatu perjalanan yang diinginkan baik itu melalui jalur mobil sampai pejalan kaki [7].

II.11 Google API

Google API adalah seperangkat application programming interfaces (API) yang dikembangkan oleh Google yang memungkinkan komunikasi dengan Layanan Google dan integrasi mereka ke layanan lainnya.

II.11.1 Google Maps API

Google Maps API adalah sebuah layanan yang memungkinkan Google Maps agar dapat diintegrasikan ke dalam aplikasi yang memang membutuhkan fitur dari Google Maps ini. Google Maps API adalah sebuah library Javascript. Dengan menggunakan Google Maps API ini dapat memudahkan pengembang dalam membuat aplikasi yang membutuhkan teknologi peta di dalamnya dan tidak perlu memikirkan cara membuatnya secara manual karena dengan mengunakan Google

Maps API ini sudah semua yang dibutuhkan terkait teknologi peta digital sudah cukup lengkap dan dapat dimanfaatkan untuk aplikasi yang membutuhkannya. [7]

II.12 Open Weather Map API

OpenWeatherMap adalah layanan online yang menyediakan data cuaca, termasuk data cuaca terkini, prakiraan, dan data historis untuk para pengembang layanan web dan aplikasi mobile. Untuk sumber data, ia menggunakan layanan siaran meteorologi, data mentah dari stasiun cuaca bandara, data mentah dari stasiun radar, dan data mentah dari stasiun cuaca resmi lainnya. Semua data diproses oleh OpenWeatherMap dengan cara mencoba menyediakan data perkiraan cuaca online yang akurat dan peta cuaca, seperti awan atau curah hujan. Di luar itu, layanan difokuskan pada aspek sosial dengan melibatkan pemilik stasiun cuaca dalam menghubungkan layanan dan dengan demikian meningkatkan akurasi data cuaca. Ideologi ini terinspirasi oleh OpenStreetMap dan Wikipedia yang membuat informasi gratis dan tersedia untuk semua orang. Menggunakan OpenStreetMap untuk menampilkan peta cuaca. [12]

II.13 Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan Bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin.

Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "*Tulis sekali, jalankan di mana pun*". Saat ini java Merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan

dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

Rilis terbaru dari Java Standard Edition adalah Java SE 8. Dengan kemajuan bahasa Java dan popularitasnya yang besar, beberapa konfigurasi dibangun sesuai dengan berbagai jenis platform. Sebagai contoh: J2EE untuk Aplikasi Enterprise, J2ME untuk Aplikasi Mobile. Versi J2 baru diganti sebagai Java SE, Java EE, dan Java ME [13].

II.14 PHP

PHP adalah singkatan dari "PHP: Hypertext Prepocessor", yaitu bahasa pemrograman yang digunakan secara luas untuk penanganan pembuatan dan pengembangan sebuah situs web dan bisa digunakan bersamaan dengan HTML.

Yang membedakan PHP dari JavaScript adalah kode PHP dijalankan di server, menghasilkan HTML yang kemudian dikirim ke pengguna. Pengguna akan menerima hasil dari menjalankan script itu, tapi tidak akan tahu apa kode yang mendasari itu. Bahkan kita dapat mengkonfigurasikan web server untuk memproses semua file HTML dengan PHP, dengan begitu pengguna tidak akan tahu apa yang ada di balik tampilan.

Hal terbaik dalam menggunakan PHP adalah bahwa hal ini sangat sederhana untuk pendatang baru, tapi menawarkan banyak fitur canggih untuk programmer profesional. [14]

II.15 JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman *JavaScript*, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data. [15]

JSON terbuat dari dua struktur:

Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.

Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini.

JSON menggunakan bentuk sebagai berikut:

1. **Objek** adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).
2. **Larik** adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).
3. **Nilai** (*value*) dapat berupa sebuah **string** dalam tanda kutip ganda, atau *angka*, atau true atau false atau null, atau sebuah *objek* atau sebuah *larik*. Struktur-struktur tersebut dapat disusun bertingkat.
4. **String** adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.
5. **Angka** adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.

II.16 MySQL

MySQL merupakan perangkat lunak basis data server yang dapat menerima dan mengirim data dengan cepat, multi user serta menggunakan perintah standar SQL (Structured Query Language). MySQL memiliki dua bentuk lisensi, yaitu FreeSoftware dan Shareware. MySQL yang biasa digunakan adalah MySQL FreeSoftware yang berada di bawah lisensi GNU General Public License (GPL), namun ada juga perangkat lunak MySQL yang dijual di bawah lisensi komersial untuk kasus-kasus dimana penggunanya tidak cocok dengan penggunaan GPL.

Selain sebagai database server, MySQL juga dapat mengakses suatu database MySQL yang berposisi sebagai server. Pada saat itu berarti program yang dibangun dengan menggunakan database MySQL berposisi sebagai client. Karena MySQL merupakan sebuah database server yang dapat berperan juga sebagai client, maka sering disebut database client/server [16].

Kelebihan MySQL dibanding dengan database lain dalam penggunaannya di antaranya :

1. MySQL sebagai Database Management System (DBMS)
2. MySQL sebagai Relation Database Management System (RDBMS)
3. MySQL merupakan *software* database open source, artinya program ini bersifat free atau bebas digunakan oleh siapa saja tanpa harus membeli dan membayar lisensi kepada pembuatnya.
4. MySQL mampu untuk menjalankan lebih dari satu query SQL dalam waktu bersamaan atau yang disebut Multi-Threading
5. MySQL merupakan database server, jadi kita dapat menghubungkannya ke media internet sehingga dapat diakses dari jauh
6. MySQL merupakan database client, selain menjadi server yang melayani permintaan, MySQL juga berperan sebagai client yang dapat menjalankan query yang mengakses database pada server
7. MySQL merupakan database yang menggunakan enkripsi password sehingga database ini cukup aman karena memiliki password untuk mengaksesnya
8. MySQL mampu menyimpan data dalam jumlah yang sangat besar hingga berukuran Giga Byte sekalipun

9. MySQL didukung oleh driver ODBC, artinya database MySQL dapat diakses menggunakan aplikasi apa saja
10. MySQL dapat menciptakan lebih dari 16 kunci per tabel, dan dalam satu kunci memungkinkan berisi tabel berjumlah belasan field (kolom)
11. MySQL memiliki kecepatan dalam pembuatan tabel maupun perubahan tabel
12. Mendukug field yang dijadikan sebagai kunci primer maupun unik
13. MySQL didukung oleh sebuah komponen C dan perl API, sehingga MySQL dapat diakses melalui sebuah program aplikasi yang berasa di bawah protokol internet berupa web
14. Memiliki berbagai storage engine yang dapat diganti pada level tabel
15. MySQL menggunakan suatu bahasa permintaan standar yang bernama SQL (Structured Query Language), yaitu bahasa yang distandarkan pada beberapa database server seperti Oracle, PostGreSQL dan lain-lain.

II.17 Android Studio

Android Studio adalah *integrated development environment* (IDE) resmi untuk platform Android.

Android Studio berada di tahap pratinjau akses awal mulai dari versi 0,1 pada bulan Mei 2013, kemudian memasuki tahap beta mulai dari versi 0.8 yang dirilis pada bulan Juni 2014. Pembuatan stabil pertama diluncurkan pada bulan Desember 2014, mulai dari versi 1.0.

Didasarkan oleh perangkat lunak IntelliJ IDEA milik JetBrains, Android Studio dirancang khusus untuk pengembangan Android. Perankat lunak ini tersedia untuk diunduh di Windows, macos dan Linux, dan menggantikan Eclipse Android Development Tools (ADT) sebagai IDE utama Google untuk pengembangan aplikasi Android asli. [17] Android Studio yang akan dipakai adalah versi 2.3.1.

II.18 CodeIgniter

Codeigniter adalah *software* open source yang digunakan sebagai framework untuk pembangunan aplikasi website yang *dynamic* menggunakan PHP. *Codeigniter* menggunakan basis MVC (*Model-View-Controller*) sebagai pola pengembangannya. Kelas *controller* adalah bagian penting dalam jika

menggunakan Framework ini, kelas model dan view bersifat opsional. *CodeIgniter* juga dapat dimodifikasi menggunakan Hierarchical Model View Controller (HMVC) yang memungkinkan pengembang untuk mempertahankan pengelompokan modular Controller, Models dan View yang disusun dalam format sub-direktori.

CodeIgniter paling dikenal karena kecepatannya, dibandingkan dengan framework PHP lainnya. [18]

Penggunaan *CodeIgniter* pada aplikasi ini menggunakan framework *CodeIgniter* pada pembangunan aplikasi berbasis web. *CodeIgniter* dapat dilihat pada https://codeigniter.com/.

II.19 Object Oriented Programming

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan satu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Tujuan dari analisis berorientasi objek adalah untuk mengembangkan model yang menggambarkan perangkat lunak komputer karena bekerja untuk memenuhi seperangkat persyaratan yang ditentukan *user*. *Tools* yang dapat digunakan pada pendekatan analisis pengembangan sistem secara objek dapat menggunakan UML.

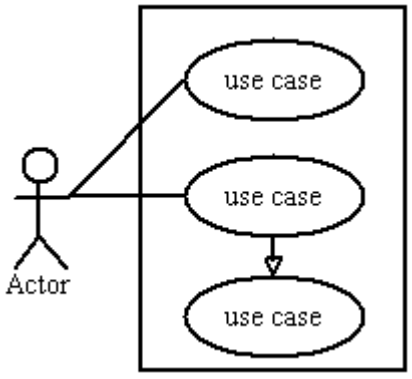
II.20 Unified Modelling Languange

Unified Modelling Language (UML) adalah sebuah bahasa yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menggunakan *class* dan *operation object* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa bahasa berorientasi objek [19]. Dalam membangun block UML ada 3 hal yang harus diperhatikan, yaitu *object* (memodelkan konsep), *relationship* (mengkoneksikan *object*), dan *diagram* (*grouping* yang saling mengkoneksikan antara *object* dan *relationship*. Diagram yang umum dipakai dalam analisis dan desain adalah:

1. Use Case Diagram

Use Case Diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *Use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di- include oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang serupa. Sebuah *use case* juga dapat meng-*extend usecase* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Dasar menentukan sebuah *use case* adalah *use case* merupakan sesuatu yang menyediakan beberapa hasil terukur kepada pengguna atau sistem eksternal. *Use case* harus memiliki sangat jelas kriteria lulus / gagal. Pengembang, *tester*, penulis teknis, dan pengguna harus secara eksplisit tahu apakah sistem memenuhi kasus penggunaan atau tidak. Setiap bagian dari *use case* yang memenuhi tes sederhana ini mungkin menjadi kandidat yang baik untuk *use case* . Berikut adalah contoh dari *use case diagram* dapat dilihat pada Gambar II.6



Gambar II.6 *Use Case Diagram*

2. Use Case Scenario

Use Case Scenario adalah sebuah diagram yang menunjukkan *use case* dan aktor mungkin menjadi titik awal yang bagus, tetapi tidak memberikan detail yang cukup untuk desainer sistem untuk benar-benar memahami persis bagaimana sistem

dapat terpenuhi. Cara terbaik untuk mengungkapkan informasi penting ini adalah dalam bentuk penggunaan *use case scenario* berbasis teks per *use case* nya.

Berikut adalah dasar format penulisan *use case scenario* [20]. Dasar pembangunan *use case scenario* dapat dilihat pada **Error! Reference source not found.**.

Tabel II.2 Dasar pembangunan *use case scenario*

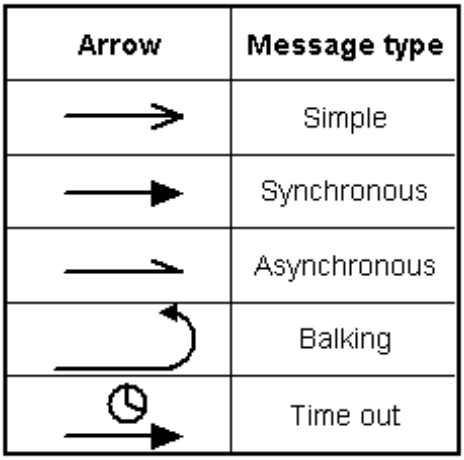
Use Case Name	Berisi nama dari *use case* yang akan digunakan	
Goal In Context	Menjelaskan apa yang aktor coba untuk dapatkan dari *use case*	
Description	Menjelaskan gambaran dari *use case*	
Related *Use Case*	Daftar *use case* yang berhubungan dengan *use case* tersebut	
Successful End Condition	Kondisi *use case* jika berhasil	
Failed End Condition	Kondisi *use case* jika gagal	
Actors	Daftar aktor yang dapat mengakses *use case*	
Trigger	Aktifitas yang dilakukan untuk mengawali *use* Case	
Main Flow	Step	Action
	1	Deskripsi urutan aksi dari aktifitas *use case*
	2	
Extension	Step	Branching Action
	2.1	Deskripsi urutan aksi lain selain urutan aksi utama

3. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event*

untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message* [19]. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek *boundary*, *controller* dan *persistent entity*. Dasar pembangunan *sequence diagram* dapat dilihat pada Gambar II.7 Dasar Pembangunan *Sequence Diagram*.



Gambar II.7 Dasar Pembangunan *Sequence Diagram*

4. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut/properti) satu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok:

1. Nama dan stereotype
2. Atribut

3. Metode

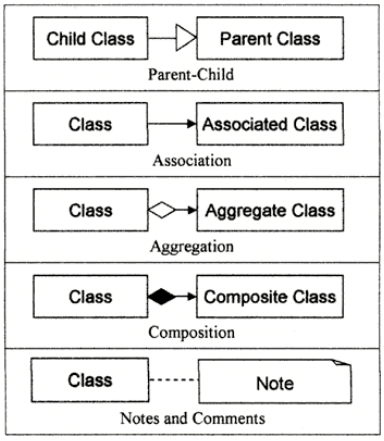
Atribut dan metode dapat memiliki salah satu sifat berikut:

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public*, dapat dipanggil oleh siapa saja.

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metode. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.

Class memiliki tipe-tipe *relationship*, diantaranya:

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain.
2. Agregasi, yaitu hubungan yang menyatakan bagian terdiri atas dimana ketika satu *class* di-*share* atau direferensikan kepada objek yang ada di *class* lain.
3. Pewarisan, yaitu hubungan hirarkis antar *class*. Class dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Komposisi, yaitu jenis relasi *class diagram* yang kuat dimana jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *Composition* terhadap *class* tempat dia bergantung tersebut. Sebuah *relationship composition* digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/solid.
5. Depedensi, salah satu jenis relasi *class* diagram yang lemah dimana objek dalam suatu *class* akan bekerja sangat singkat dengan objek yang ada pada *class* lain.



Gambar II.8 Dasar Pembangunan *Class Diagram*

5. Activity Diagram

Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Sebuah aktivitas dapat direalisasikan oleh satu *usecase* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *usecase* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas.

Decision digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity* diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu [19].

II.21 Pengujian *Blackbox*

Metode pengujian *blackbox* fokus pada keperluan penelusuran kesalahan fungsional dari *software*. Uji coba *blackbox* berusaha untuk menemukan kesalahan dalam beberapa kategori, di antaranya :

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses *database* eksternal

4. Kesalahan performa
5. Kesalahan inisialisasi dan terminasi

Teknik pengujian black box terdiri dari 10 jenis diantaranya *Equivalence* Partitioning, Boundary Value Analysis/Limit Testing, Comparison Testing, Sample Testing, Robustness Testing, Behavior Testing, Requirement Testing, Performance Testing, Endurance Testing, Cause-Effect Relationship Testing. Salah satunya yang akan dibahas adalah Equivalence partitioning.

Equivalence partioning merupakan metode ujicoba blackbox yang membagi domain input dari program menjadi beberapa kelas data dari kasus ujicoba yang dihasilkan. Kasus uji penanganan single yang ideal menemukan sejumlah kesalahan (misalnya : kesalahan pemrosesan dari seluruh data karakter) yang merupakan syarat lain dari suatu kasus yang dieksekusi sebelum kesalahan umum diamati.

II.22 Pengujian Usability

Pengujian *usability* adalah alat penelitian, yang berakar pada metodologi eksperimen klasik. Rentang tes yang dapat dilakukan seseorang cukup besar, mulai dari eksperimen klasik yang benar dengan ukuran sampel yang besar dan rancangan uji yang kompleks hingga penelitian kualitatif yang sangat informal dengan hanya satu peserta. Setiap pendekatan pengujian memiliki tujuan yang berbeda, serta kebutuhan waktu dan sumber daya yang berbeda.

Merancang produk yang akan berguna harus merupakan proses terstruktur dan sistematis, dimulai dengan tujuan tingkat tinggi dan beralih ke tujuan spesifik. Anda tidak dapat mencapai tujuan - kegunaan atau sebaliknya - jika tetap samar dan tidak dipahami. Bahkan istilah kegunaan itu sendiri harus didefinisikan dengan masing-masing organisasi.

Definisi operasional dari apa yang membuat produk dapat digunakan meliputi:

1. Mudah digunakan (*Usefulness*)

Fokus kepada apa yang membuat produk atau sistem bisa membuat pengguna mendapat tujuannya, dan merupakan penilaian dari kesediaan pengguna untuk dapat menggunakan produk dengan inisiatifnya sendiri.

2. Efisien (*Efficiency*)

Kecepatan dengan tujuan pengguna dapat dicapai secara akurat dan lengkap dan sesuai dengan ukuran waktu yang seharusnya (dapat ditetapkan oleh pengembang produk). Misalnya, menetapkan patokan pengujian kegunaan yang mengatakan “95 persen dari semua pengguna akan dapat memuat perangkat lunak dalam waktu 10 menit”.

3. Efektif (*Effectiveness*)

Efektivitas lebih mengacu kepada sejauh mana produk digunakan dalam cara yang pengguna harapkan dan kemudahan dalam penggunaannya untuk melakukan apa yang mereka inginkan. Hal ini biasanya diukur secara kuantitatif dengan tingkat kesalahan yang terjadi. Efektivitas bisa disebut pengujian yang memperluas efisiensi, contoh “95 % dari semua pengguna akan dapat memuat perangkat lunak dengan benar pada upaya pertama”.

4. Memuaskan (*Satisfaction*)

Kepuasan mengacu pada persepsi pengguna, perasaan, dan pendapat dari produk, biasanya diambil melalui kedua pertanyaan tertulis dan lisan. Pengguna lebih mungkin untuk melakukan dengan baik pada produk yang memenuhi kebutuhan mereka dan memberikan kepuasan atau tidak.

5. Dapat Dipelajari (*Learnability*)

Bagian dari efektivitas dan hubungannya dengan kemampuan pengguna untuk mengoperasikan sistem. Hal ini juga dapat merujuk pada kemampuan pengguna yang jarang untuk mempelajari kembali sistem setelah periode tidak aktif. Berarti sistem atau produk harus dipastikan kemudahan untuk dipelajari penggunaannya oleh pengguna.

6. Dapat Diakses (*Accessibility*)

Dalam arti luas, *accessibility* adalah tentang memiliki akses ke produk yang dibutuhkan untuk mencapai tujuan. Namun dalam hal ini, kita melihat seberapa jauh atau apa yang membuat produk dapat digunakan oleh orang-orang yang memiliki keterbatasan tertentu. [21]

II.23 Pengujian Service Prototype

Service Prototype adalah alat untuk menguji layanan dengan mengamati interaksi pengguna dengan prototipe layanan yang diletakkan di tempat, situasi dan kondisi di mana layanan benar-benar ada. Tujuannya adalah untuk memverifikasi apa yang terjadi bila beberapa faktor eksternal mengganggu selama pemberian layanan, faktor-faktor yang tidak mungkin dilakukan untuk memverifikasi selama tes sebelumnya di laboratorium namun memiliki dampak yang besar terhadap persepsi dan pengalaman pengguna [22].