BAB II
TINJAUAN PUSTAKA

II.1 Profil Tempat Penelititan

Berikut ini adalah profil tempat penelitian yang didapatkan dari situs resmi Desa Cibodas. Desa Cibodas merupakan desa yang terletak di Kecamatan Lembang, Kabupaten Bandung Barat. Desa Cibodas terbagi atas 3 wilayah dusun, 17 wilayah Rukun Warga (RW) dan 66 wilayah Rukun Tetangga (RT). Lahan pertanian di wilayah desa ini seluas 433.72 ha/m^2, sedangkan lahan perkebunan seluas 351 ha/m^2, luas lahan yang digunakan sebagai perkantoran sebanyak 0.3 ha/m^2. Sisa lahan lainnya digunakan sebagai lahan pemukiman seluas 111.5 ha/m^2, hutan konservasi seluas 32 ha/m^2 dan perasarana umum lainnya seluas 0.92 ha/m^2 [1].

Mata pencaharian utama dari Desa ini adalah dari sektor pertanian khususnya untuk tanaman pangan dan perkebunan. Selain itu, potensi kehutanan dan untuk perkembangan peternakan dari Desa ini juga cukup potensial. Jumlah penduduk di Desa Cibodas Kecamatan Lembang sebanyak 9.898 orang yang terdiri dari 4.927 orang laki-laki dan 4.971 perempuan dengan jumlah keluarga sebanyak 2.981 KK. Mata pencaharian penduduk Desa sebagian besar adalah buruh tani yang terdiri dari 892 buruh tani laki-laki dan 528 buruh tani perempuan. Petani di Desa Cibodas sebanyak 734 petani yang terdiri dari 699 petani laki-laki dan 35 petani perempuan [1].

II.2 Landasan Teori

Landasan teori merupakan teori-teori yang berasal dari berbagai sumber, teori-teori ini akan digunakan sebagai landasan dan pendukung pada penelitian ini.

II.2.1 Desain Interaksi

Desain interaksi didefinsikan sebagai perancangan produk-produk interaktif untuk mendukung orang-orang dalam kehidupan dan pekerjaannya setiap hari [4].

Desain interaksi dapat dipahami dalam istilah yang sederhana, yaitu desain bagaimana pengguna berinteraksi dengan produk. Banyak orang ketika berbicara mengenai desain interaksi, beranggapan bahwa produk yang dimaksud adalah sebuah aplikasi atau sebuah website. Tetapi desain interaksi lebih dari itu, tujuan dari desain interaksi adalah untuk menciptakan produk yang memungkinkan pengguna untuk mencapai tujuan mereka sebaik mungkin [5].

II.2.2 *User Interface* (UI)

User interface atau antarmuka pengguna merupakan jembatan penghubung interaksi antara manusia dan sistem yang artinya harus mereflesikan kapabilitas seseorang dan kebutuhannya, *user interface* harus berguna sehingga dapat mencapai tujuan pengguna, *user interface* harus mudah dipelajari sehingga sistem mudah digunakan dan tidak membuat frustasi. *User interface* yang baik adalah yang memungkinkan pengguna untuk fokus pada informasi dan tugas yang dikerjakan, bukan mekanisme yang digunakan untuk menyajikan informasi dan melakukan tugas [6] [7].

1. Paradigma Interaksi

Terdapat tida paradigma dominan dalam perancangan konseptual dan visual sebuah antarmuka yaitu: berpusat-pada-implementasi (BPI), metaforik dan idiomatik.

a. Antarmuka Berpusat Pada Implementasi

   Antarmuka BPI sangat banyak dijumpai di kalangan industry komputer. Antarmuka ini diekspresikan dalam bentuk atau cara mereka dibentuk. Tingkat keberhasilan penggunaannya bergantung kepada seberapa jauh pengguna memahami cara bekerjanya program. Perancangan antarmuka BPI mengharuskan perancang untuk secara ekslusif berfokus pada model implementasinya [6].

b. Antarmuka Metaforik

   Antarmuka metaforik bergantung kepada hubungan intuitif yang dibuat pengguna pada saat mereka melihat simbol visual dari suatu komponen antarmuka dengan fungsinya. Dalam hal ini tidak ada keharusan bagi pengguna untuk memahami mekanisme suatu program, sehingga

paradigma ini merupakan satu langkah maju dari BPI, hanya saja daya tarik dan manfaatnya dibesar-besarkan secara tida proporsional.

Metafor dalam konteks antarmuka dan perancangan interaksi berearti metafor visual: gambar yang digunakan untuk menyajikan suatu maksud atau atribut tertentu dari sebuah benda. Pengguna mengenali suatu gambaran metafor dan memahami maksudnya. Metafor sangat variative, mulai dari gambar sederhana pada sebuah tombol *toolbar* sampai ke keseluruhan layar, dari simbol gunting yang berarti *cut* sampai ke buku cek berukuran besar [6].

c. Antarmuka Idiomatik

Perancangan idiomatik didasarkan pada cara pengguna belajar sesuatu dan menggunakan berbagai idiom – misalnya polisi tidur, jago merah atau ringan tangan. Antarmuka idiomatik memecahkan persoalan-persoalan yang dimiliki oleh kedua paradigma di atas dengan tidak berfokus kepada pengetahuan teknis atau intuisi tentang bagaimana suatu benda berfungsi, tetapi berfokus pada idiom visual dan perilaku sederhana untuk menyelasaikan suatu tugas.

Kebanyakan elemen antarmuka grafis yang intuitif sesungguhnya merupakan idiom visual. Jendela, papan judul, penutup kotak, pemisah layar, hyperlink, drop-down dan lain-lain, merupakan benda-benda yang kita pelajari secara idiomatik bukan secara metaforik.

Menurut Cooper and Reimann (2003), perancangan idiomatic merupakan masa depan perancangan antarmuka. Dalam paradigma ini, pengguna bergantung kepada kemampuan alamiah manusia untuk memahami tentang “bagaimana” dan “mengapa”. Jumlah idiom yang dapat digunakan bisa dibilang tak terbatas dibandingkan dengan jumlah metafor yang lebih terbatas. Metafor memberikan sedikit manfaat dibanding beban tambahan dikemudian hari [6].

2. Pentingnya Desain *User Interface* yang Baik

Desain UI yang baik sangat penting untuk sebuah perangkat lunak, karena jika sebuah perangkat lunak memiliki antarmuka yang buruk akan mengakibatkan

hal buruk juga kepada penggunanya. Berikut ini adalah beberapa alasan kenapa desain UI yang baik itu sangat penting:

a. Banyak sistem dengan funsionalitas yang baik tapi tidak efisien, membingungkan dan tidak berguna karena desain UI yang buruk.
b. Antarmuka yang baik merupakan jendela untuk melihat kemampuan sistem serta jembatan bagi kemampuan perangkat lunak.
c. Desain yang buruk akan membingunkan, tidak efisien, bahkan menyebabkan frustasi.

Berikut adalah hasil penelitian kenapa desain UI yang baik itu sangat penting:

a. Pengguna bekerja 20% lebih produktif dengan layar yang sederhana.
b. Pengguna layar yang dimodifikasi menyelasaikan transaksi 25% lebih cepat dan error berkurang 25% dari sebelumnya.
c. Window yang didesain dengan efektif mengehmat $20.000 dalam 1 tahun.

Fungsi searching yang diperbaiki dapat meningkatkan success rate hingga 15% dan waktu pencarian 50% lebih cepat.

3. Komponen User Interface

Antarmuka pengguna atau *user interface (UI)* terdiri dari empat komponen sebagai berikut:

1. Model Pengguna

   Model pengguna merupakan model konseptual yang diinginkan pengguna dalam memanipulasi informasi dan proses yang akan dibuat dalam aplikasinya.
2. Bahasa Perintah

   Bahasa perintah merupakan bahasa sebagai piranti yang digunakan untuk memanipulasi model.
3. Umpan Balik

   Umpan balik merupakan kemampuan program untuk membantu pengguna dalam mengoperasikan program yang telah dihasilkan.
4. Penampilan Informasi

   Penampilan informasi merupakan petunjuk status informasi atau program ketika pengguna melakukan tindakan, perlu dirancang pesan yang efektif untuk membantu pengguna dalam menggunakan aplikasi.

4. Perinsip Umum Perancangan *User Interface*

Dalam melakukan perancangan *user interface* tentunya ada perinsip-perinsip yang dapat digunakan sebagai acuan agar memberikan *user interface* sesuai dengan kebutuhan pengguna. Berikut adalah perinsip-perinsip umum dalam perancangan sebuah *user interface* [6]:

1. *User Compability*

   Produk yang dibangun harus disesuaikan dengan kebutuhan pengguna, karena semua pengguna tidak sama dan pengguna bukanlah seorang *developer* ataupun *designer*. Seorang perancang juga haru memahami psikologi dari calon pengguna perangkat lunak seperti karakteristik dan difat pengguna.

2. *Product Compability*

   Produk yang dihasilkan harus memiliki kompabilitas antar produk harus diperhatikan dan dipertahankan. Ada kemungkinan antarmuka yang dibangun memiliki beberapa perbedaan dengan peroses yang dilakukan secara manual.

3. *Task Compability*

   Dalam perancangan sistem, struktur dan aliran sistem harus sesuai dan mendukung tugas yang dimiliki oleh pengguna sehingga dapat membantu pengguna dalam menyelesaikan tugas dan mencapai tujuan mereka.

4. *Work Flow Compability*

   Work Flow Compability merupakan pengorganisasian sistem yang harus dilakukan secara baik sehingga dapat mem-fasilitasi transisi antar tugas pengguna.

5. *Consistency*

   Dalam perancangan sistem, antarmuka yang dibangun harus konsisten dan sesuai dengan sistem nyata serta sesuai dengan produk yang dihasilkan sebelumnya.

6. *Familiarity*

   Perancangan antarmuka sebaiknya disesuaikan dengan antarmuka pada umumnya serta familiar terhadap pengguna. Perancangan meilputi tata

letak, simbol, bahasa dan lain sebagainya sehingga pengguna akan dengan meudah menggunakan dan mengenali antarmuka tersebut.

7. *Simplicity*

   Kesalahan umum yang sering terjadi pada perancangan antarmuka adalah berusaha untuk menyediakan semua fungsionalitas. Hal ini mengakibatkan antarmuka sulit untuk digunakan dan tidak sesuai dengan kebutuhan pengguna.

8. *Direct Manipulation*

   Direct manipulation dimaksudkan agar pengguna dapat melihat aksinya secara langsung pada objek yang terlihat.

9. *Control*

   Jangan biarkan pengguna merasa di kontrol oleh mesin, karena hal ini dapat menyebabkan frustasi dan demoralisasi bagi pengguna.

10. *What You See Is What You Get (WYSIWYG)*

    WYSIWYG yang berarti apa yang pengguna lihat adalah apa yang pengguna akan dapatkan. Contohnya, saat pengguna mencetak sebuah dokumen di printer maka keluaran yang didapatkan oleh pengguna akan sama persis dengan apa yang dilihatnya di komputer.

11. *Flexibility*

    Flexibility adalah memberikan pengguna banyak kontrol dan mengkoordinir keahlian pengguna yang berpariasi.

12. *Responsiveness*

    Responsiveness memungkinkan sistem untuk memberikan respon dengan segera setiap masukan dari pengguna.

13. *Invisible Technology*

    Pengguna sebaiknya mengetahui sedikit mungkin detail teknis bagaimana sistem diimplementasikan. Sehingga pengguna tidak memiliki rasa khawatir dan ketakutan menggunakan aplikasinya.

14. *Robustness*

    Sistem harus mampu mentolerir kesalahan manusia baik disengaja maupun tidak disengaja dan yang umumnya tidak dapat terhindari. Penggunaan

bahasa dalam mentolerir kesalahan juga harus diperhatikan jangan sampai bahasa yang digunakan seperti membentak atau menyalahkan pengguna.

15. *Protection*

    Sistem yang dirancanga harus dapat memproteksi dari hasil-hasil yang menyebabkan "bencana" karena kesalahan umum yang dibuat oleh pengguna.

16. *Easy of Learning*

    Sistem yang dibangun harus mudah dipelajari sehingga pengguna tidak harus diberikan pelatihan penggunaan sistem.

17. *Easy of Use*

    Sistem yang dibangun harus mudah digunakan baik bagi pengguna pemula maupun pengguna yang sudah berpengalaman.

5. **Waktu Tanggap**

Secara umum. Pengguna menginginkan bahwa program aplikasinya dapat memberikan waktu tanggap yang sependek-pendeknya. Waktu tanggap yang berbeda-beda dapat mempengaruhi konsenstrasi pengguna yang pada gilirannya akan mempengaruhi kinerja pengguna [6].

Waktu tanggap yang lama, lebih dari 14 detik, akan menyebabkan perhatian pengguna terpecah ke aktivitas yang lain, sehingga pengguna cenderung untuk melakukan aktivitas lain sampai sistem menyelesaikan apa yang harus ia kerjakan.

6. **Penanganan Kesalahan**

Dalam dunia komputer kesalahan kecil dapat berakibat fatal sehingga harus diupayakan agar kesalahan-kesalahan yang mungkin dilakukan dapat dihindari atau ditangani dengan cara-cara tertentu agar tidak mengakibatkan terhentinya eksekusi program aplikasi. Keslahan dibagi menjadi dua, yakni kesalahan sintaksi pada saat penulisan program dan kesalahan logika yang terjadi pada saat program sedang dijalankan [6].

II.2.3 *User Experience* (UX)

Menurut Joel Marsh tujuan akhir dari *User Experience* bukan hanya sekedar membuat pengguna bahagia. Banyak orang dengan sembarangan berpikir bahwa "UX" berarti sebuah pengalaman pengguna/*user experience*. tetapi sebenarnya

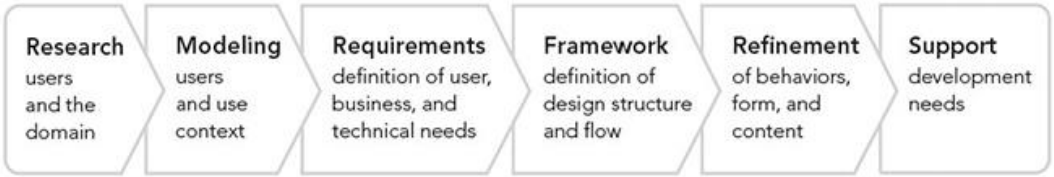
UX adalah tentang “melakukan” proses perancangan pengalaman pengguna/*user experience design* [8]. Dalam *e-book “UX Design: The Definitive Beginner’s Guide”* oleh UXPin mengatakan bahwa UX adalah sebuah proses dan sebuah produk [9].

UX Design (terkadang disebut juga sebagai UXD) melibatkan sebuah proses yang mirip dengan proses sains: melakukan *research* untuk memahami pengguna, mengumpulkan berbagai ide untuk memecahkan kebutuhan pengguna dan kebutuhan dari proses bisnisnya, membangun dan memastikan solusi-solusi tersebut dapat bekerja dan berguna di kehidupan nyata [8].

Dari beberapa penjelasan di atas dapat diambil kesimpulan bahwa *User Experience* merupakan peroses dalam melakukan sebuah perancangan pengalaman pengguna yang nantinya dapat menghasilkan sebuah produk yang dapat digunakan dan berguna di kehidupan nyata.

II.2.4 *Goal Directed Design* (GDD)

Goal Directed Design (GDD) adalah sebuah metode untuk perancangan *user interface* aplikasi yang berfokus pada tujuan pengguna sehingga pengguna merasa puas dan senang. Metode ini menyediakan solusi yang memenuhi kebutuhan dan tujuan dari pengguna. Proses GDD dibagi menjadi enam tahap yaitu *Research, Modeling, Requirement Definition, Framework Definition, Refinement,* dan *Support* [3] [10] [11] seperti yang ditunjukan pada Gambar 0-1 dibawah ini.



Gambar 0-1 *Goal Directed Design (GDD)* [3] [10]

Berikut penjelasan mengenai tahapan proses motode *Goal Directed Design*, seperti yang terlihat pada Gambar 0-1:

1. **Research**

Tahap *research* menggunakan teknik etnografi (observasi, wawancara, kuisoner atau teknik pengumpulan data lain) untuk menghasilkan data kualitatif tentang calon pengguna sesungguhnya dari suatu produk. Salah satu hasil utama dari tahap ini adalah munculnya serangkaian pola perilaku. Pola ini juga akan menggambarkan tujuan dan motivasi pengguna sehingga menghasilkan *user persona* yang tepat untuk aplikasi yang akan dibangun. Pola perilaku dan tujuan dapat mendorong terciptanya *persona* dalam tahap *modeling.*

2. Modeling

Pada tahap ini, pola yang telah didapatkan dari tahap *research* digunakan untuk memodelkan pengguna. *Persona* merepresentasikan *behavior* (perilaku), *attitude* (sikap), *goals* (tujuan) dan motivasi pengguna yang diamati dan diidentifikasi selama tahap *research.* Semua aspek dari persona dimaksudkan untuk mengetahui apa yang dibutuhkan *user.*

3. Requirement Definition

Pada tahap ini menggunakan metode desain berbasis skenario yang mendeskripsikan alur dari *task* secara detail berdasarkan tujuan *persona* yang telah didapatkan pada tahap sebelumnya. Tujuan pengguna berfungsi sebagai filter untuk *task* dan *sub task* dan sebagai panduan untuk membangun scenario. Tahap ini mendefiniskan informasi dan kemampuan *persona* yang dibutuhkan untuk mencapai tujuan pengguna. hasil dari fase ini adalah *requirement definition* yang menyeimbangkan kebutuhan pengguna, bisnis dan teknis dari desain yang diperlukan.

4. Framework Definition

Pada tahap ini membuat konsep produk secara keseluruhan, menentukan kerangka dasar *(wireframe)* untuk *product behavior,* dan desain visual. Fase ini menghasilkan definisi *interaction framework,* konsep desain yang stabil dan menunjukan struktur formal dan logis untuk detail yang akan datang.

5. Refinement

Tahap ini sama halnya dengan tahap sebelumnya, tetapi dengan peningkatan fokus lebih detail dan implementasi. Tahap ini berfokus pada pembangunan detail desain pada setiap komponen atau elemen *user interface,* seperti menentukan visual *style,* ikon, warna dan elemen visual lain yang sesuai dengan tujuan dan

pengalaman pengguna. hasil akhir dari fase ini adalah dokumentasi dari desain: spesifikasi bentuk dan perilaku yang ada disajikan dalam bentuk laporan atau persentasi.

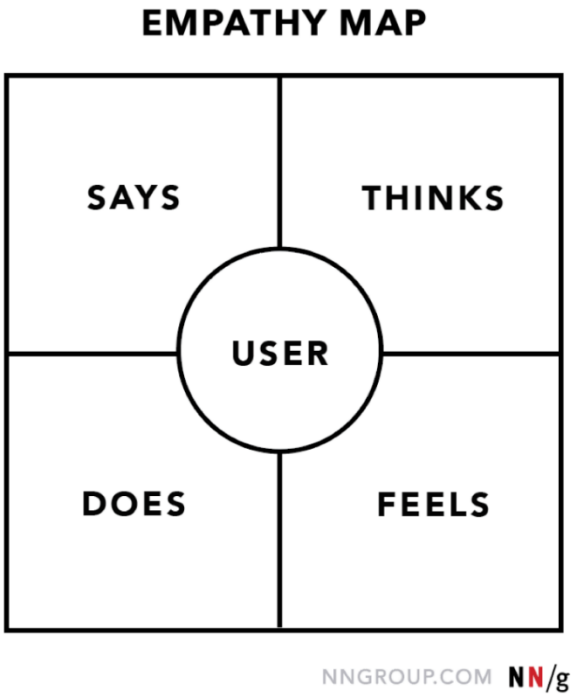
6. Support

Tahap ini merupakan tahap implementasi desain yang dihasilkan pada tahap *refinement.* Pada tahapan ini, desain *user interface* yang telah dibuat pada tahapan *refinement* diimplementasikan menjadi sebuah model aplikasi *prototype*.

II.2.5 *Empathy Map*

Empathy Map adalah visual kolaboratif yang digunakan untuk mengartikulasikan atau mendeskripsikan mengenai tipe pengguna tertentu. *Empathy Map* juga digunakan untuk menciptakan pemahaman tentang kebutuhan pengguna, dan membantu dalam pengambilan keputusan [12].

Gambar 0-2 adalah peta empati tradisional dari *NNGroup.com* yang dibagi menjadi 4 kuadran yaitu: *Says, Thinks, Do*, dan *Feels*, dengan pengguna atau *persona* berada di tengah. *Empathy Map* memberikan pandangan sekilas siapa pengguna secara kesuluruhan dan tidak berurutan.



Gambar 0-2 *Empathy Map* [12]

Kuadran ***Says*** berisi apa yang pengguna katakan dengan keras dalam sebuah wawancara atau studi lainnya. Idealnya, kuadran ini berisi sebuah kutipan atau kalimat yang sering dikatakan oleh pengguna [12].

Kuadran ***Thinks*** berisi apa yang sedang dipikirkan oleh pengguna saat sesi wawancara berlangsung. Terkadang pengguna enggan menyuarakan apa yang ada didalam pikiran mereka sehingga harus diberikan perhatian lebih pada apa yang ingin diungkapkan oleh pengguna [12].

Kuadran ***Does*** berisi apa yang pengguna lakukan pada saat mereka melakukan pekerjaan mereka [12].

Kuadran ***Feels*** berisi keadaan emosi pengguna, sering kali direpresentasikan sebagai kata sifat ditambah kata sifat untuk konteks sifat tersebut. Contohnya: Khawatir: mereka melakukan sesuatu yang salah [12].

Berikut ini adalah cara dalam membangun *empathy map* menurut *NNGroup.com* [12]:

1. **Menentukan ruang lingkup dan tujuan**

Tahap pertama *empathy map* adalah dengan menentukan pemetaan untuk *user* atau *persona.* Selalu memulai dengan pemetaan 1:1 atau 1 *user/persona* per *empath map.*

2. Mengumpulkan bahan-bahan

Tahap selanjutnya mengumpulkan bahan-bahan atau media yang dibutuhkan untuk peroses pengerjaan *empathy map.* Media yang dibutuhkan seperti *sticky notes,* spidol, dan papan tulis.

3. Mengumpulkan penelitian

Tahapan selanjutnya mengumpulkan data penelitian yang diperlukan untuk mengisi *empathy map. Empathy map* memerlukan data kualitatif, sehingga diperlukan juga pengumpulan data yang bersifat kualitatif, seperti: wawancara, pengguna, observasi, *study literature,* atau suervei kualitatif.

4. Membuat *note* untuk setiap kuadran

Setelah memiliki data penelititan, selanjutnya menempatkan data tersebut kesetiap kuadran *empathy map.*

5. Membuat kluster

Selanjutnya menentukan data mana saja yang memiliki kemiripan satu sama lain, tahapan ini bisa dengan memberikan not-not serupa pada kuadran yang sama. Setelah semua *empathy map* berkumpul, kita dapat mulai mengelompokan data yang cocok dan tidak cocok.

6. Membuat rencana

Empathy map dapat disesuaikan dengan kebutuhan penelitian seperti menambahkan kuadran ***Goals*** pada penelitian yang sedang berlangsung.

II.2.6 *User Persona*

User Persona atau *Persona* adalah model pengguna yang direpresentasikan sebagai manusia individu yang sepesifik. Persona bukan orang yang sebenarnya tetapi perpaduan langsung dari penelitian dan pengamatan orang nyata. Salah satu alasan persona sangat baik digunakan sebagai model pengguna adalah dari personafikasinya. Persona memasukan empati ke pada tim perancangan dan pengembangan agar dekat dengan tujuan/*goals* dari pengguna [11].

User Persona adalah dokumen yang sangat penting yang harus dibuat untuk menganalisis para pengguna. *User Persona* adalah pondasi untuk dokumentasi

pengguna yang memperdelam dan memperluas wawasan dari kepribadian pengguna. *Persona* membuat data lebih dapat dipahami dengan memberikannya sebuah nama dan sebuah wajah, bukan angka-angka dan kata-kata tanpa konteks. Desainer menggunakan *persona* di setiap tahapan dari proses dan setiap pertanyaan dalam perancangan sebuah desain [13]. *User Persona* yang baik dapat berisi beberapa data seperti dibawah ini [13]:

a. *Photo & Name*
b. *Demographics*
c. *Personality*
d. *Technological Expertise*
e. *Platforms*
f. *Goals (Motivations)*
g. *Personal Quotes/Mottos*

Setelah *user persona* didapatkan berikutnya adalah mengelompokan *persona* tersebut kedapalam beberapa kelompok yaitu: *primary, secondary, supplemental, customer, served,* dan *negative* [11].

a. ***Primary Persona,*** merupakan target utama dari perancangan desain antarmuka. Sebuah produk bisa hanya memiliki satu *primary persona* per antarmuka, tetapi dimungkinkan untuk beberapa produk memiliki beberapa antarmuka yang berbeda, dan masing-masing ditargetkan pada *primary persona* yang berbeda.

b. ***Secondary Persona,*** merupakan persona yang hampir sepenuhnya puas dengan semua kebutuhan antarmuka persona primer. Dalam *secondary persona* tidak dianjurkan memiliki tiga atau empat lebih persona, karena hal ini akan mengakibatkan ruang lingkup membesar dan tidak fokus pada tujuan produk.

c. ***Supplemental Persona,*** merupakan persona yang tidak termasuk kedalam *primary persona* ataupun *secondary persona.* Kebutuhan dari persona ini sudah diwakili oleh kombinasi dari *primary persona* dan *secondary persona.*

d. ***Customer Persona,*** merupakan persona yang berfokus kepada pelanggan dari produk bukan kepada pengguna akhir.

e. ***Served Persona,*** merupakan persona yang tidak menggunakan produk, tetapi mereka secara langsung dipengaruhi oleh pengguna produk.

f. ***Negative Persona,*** merupakan persona yang harus dihindari, dan merupakan kebalikan dari *primary persona.*

II.2.7 *User Goals*

Jika *persona* menyediakan konteks untuk menetapkan perilaku yang diamati, maka tujuan/*goals* adalah pendorong dari perilaku tersebut. Tujuan pengguna berfungsi sebagai panduan dalam perancangan fungsi dari sebuah produk. Fungsi dan perilaku produk harus dapat mengatasi setiap permasalahan dari tugas-tugas pengguna. Tujuan bukan hanya menyediakan sebuah jawaban mengapa dan bagaimana agar pengguna ingin menggunakan produk tersebut. Tujuan juga dapat digunakan sebagai pedoman bagi desainer dalam merancang sebuah produk yang kompleks [11].

1. **Tipe-tipe *User Goals***

Dalam buku "*About Face: The Essesntials of Interaction Design"* dari *Alan Cooper,* terdapat tiga tipe tujuan pengguna yaitu *Experience Goals, End Goals,* dan *Life Goals*. Tiga tipe tujuan pengguna tersebut sesuai dengan teori *Emotional Design* oleh *Don Norman* [11]. Berikut ini adalah penjelasan dari ketiga tipe tujuan pengguna tersebut:

a. *Experience Goals*

Experience goals adalah tujuan pengalaman saat pengguna menggunakan sebuah peroduk. *Experience goals* mengekspresikan bagaimana perasaan yang diinginkan seseorang saat menggunakan sebuah produk, atau kualitas pada saat mereka berinteraksi dengan peroduk tersebut. Tujuan dari tipe ini berfokus pada visual, animasi, transisi tiap halaman, *touch response*, dan rasio snap dari sebuah tombol [11].

Experience goals ini juga menawarkan wawasan tentang motivasi kepribadian yang mengekspresikan diri tiap *persona*, seperti pengguna merasa pintar, bersenang-senang, dan lain sebagainya. Perancang harus dapat menerjamahkan *experience goals* kedalam kondisi, sikap, gerakan, dan elemen pendengaran [11].

b. *End Goals*

End goals atau tujuan akhir pada saat pengguna menggunakan sebuah produk. Ketika kita mengambil sebuah telpon seluler atau ketika membuka sebuah dokumen di *word,* dalam pikiran kita mungkin mengharapkan sebuah hasil yang ingin kita peroleh. Sebuah produk atau layanan dapat membantu mencapai suatu tujuan secara langsung maupun tidak langsung. Tujuan ini berfokus pada desain interaksi dari produk, informasi arsitektur, dan aspek-aspek dari desain. *End goals* harus memenuhi keinginan pengguna dan membuatnya berpikir bahwa produk ini sepadan dengan waktu dan keuangan mereka [11].

Beberapa contoh dari *end goals* seperti, menemukan music yang saya sukai, mendapatkan persetujuan terbaik, tetap terhubung dengan teman dan keluarga, dan lain sebagainya [11].

Perancang interaksi harus menggunakan *end goals* sebagai fondasi untuk *product behaviors, tasks, look and feel.* Konteks atau sekenario dan *cognitive walkthroughs* adalah *tools* yang efektif untuk menjelajahi *goals* dan mental model [11].

c. *Life Goals*

Life Goals mewakili aspirasi pribadi pengguna yang biasanya melampaui konteks produk yang dirancang. *Goals* ini mewakili motivasi yang akan membantu menjelaskan "mengapa" pengguna mencoba untuk menyelesaikan tujuan akhir/*end goals* yang berusaha mereka capai. *Life goals* menggambarkan keinginan jangka panjang seseorang, motivasi, dan citra diri, yang menyebabkan pengguna terhubung dengan produk. *Goals* ini berfokus pada desain produk secara keseluruhan, strategi, dan merek dagang [11].

Berikut adalah contoh dari *life goals* seperti, barhasil dalam ambisi saya untuk…, menjalani hidup yang lebih baik, dan lain sebagainya [11].

Perancang interaksi harus menerjemahkan *life goals* kedalam kemampuan sistem tingkat tinggi, konsep desain formal, dan *brand strategy. Mood boards* dan *context scenarios* dapat sangat membantu dalam menjelejahi aspek berbeda dari konsep produk, penelitian etnografi, dan pemodelan

budaya sangat penting untuk menemukan pola perilaku pengguna dan motivasi yang lebih mendalam [11].

2. ***User Goals* adalah Motivasi Pengguna**

Sangat penting untuk diingat bahwa memahami kepribadian pengguna adalah memahami lebih jauh mengenai motivasi dan tujuan dari pengguna dibandingkan memahami tugas-tugas yang spesifik atau demografi dari pengguna [11]. Berikut ini adalah *top-level* dari motivasi pengguna, berdasarkan sasaran persona dan model Don Norman:

a. *Experience goals*, bagaimana perasaan pengguna?
b. *End goals,* Apa yang ingin pengguna lakukan?
c. *Life goals,* Apa yang diinginkan oleh pengguna?

II.2.8 *User Story*

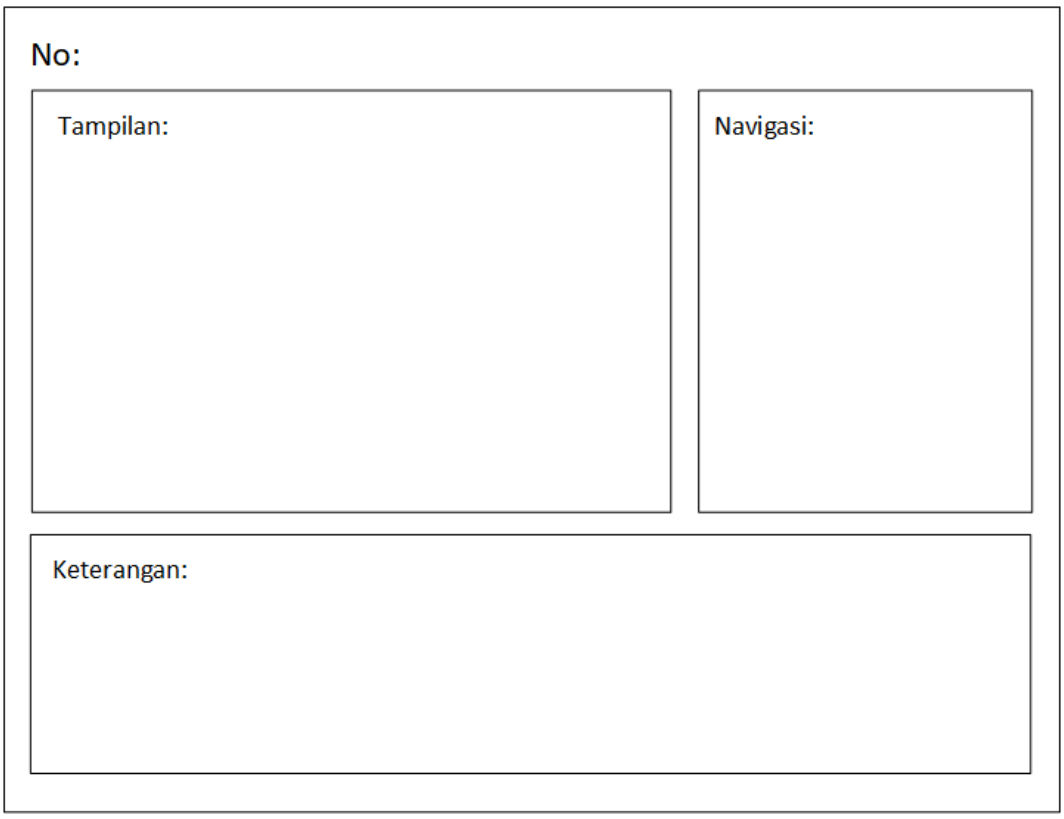
Dalam menentukan tujuan/*goals* dari pengguna, bisa menggunakan *user story*. Seperti yang dituliskan dalam *e-book "UX Design Process Best Practice"* bahwa, tahapan terbaik dalam menggunakan persona yang sudah dibuat adalah membuat *user story*. *User story* adalah kalimat sederhana atau paragraf kecil yang menguraikan motivasi dan tujuan pengguna [13].

User story biasanya menggunakan format: *sebagai seorang* ***[tipe dari pengguna]****, saya ingin* ***[sebuah fitur]*** *sehingga saya dapat* ***[menyelesaikan suatu tujuan]*** [13]**.**

User story memungkinkan perancang untuk lebih memahami tujuan pengguna dan motivasi mereka, yang mana jika dilakukan kemungkinan dapat hilang dengan semua informasi lainnya.

II.2.9 *Screen Design Worksheet*

Dokumentasi tata letak tampilan, sangat berguna sebagai pedoman untuk mengimplementasikan antarmuka, juga sangat berguna bagi pengguna. Dalam pendokumentasian dapat menggunakan peranti bantu berbentuk kertas kosong. Untuk mempermudah penamaan lembaran kertas yang dimaksud dapat diberi nama dengan lembar kerja tampilan *(screen design work sheet)* atau biasa disebut dengan LKT [6]. Gambar 0-3 menunjukan LKT yang dimaksud.



Gambar 0-3 *Screen Design Worksheet*

LKT yang disajikan pada gambar diatas pada dasarnya terdiri atas empat bagian, yaitu:

1. **Nomor:** penomoran lembar kerja.
2. **Tampilan:** berisi sketsa tampilan yang akan muncul di layar.
3. **Navigasi:** bagian ini antara lain menjelaskan kapan tampilan yang dimaksud akan muncul, dan kapan tampilan itu berubah menjadi tampilan yang lain. Perubahan tampilan biasanya disebabkan oleh adanya suatu persitiwa *(event).*
4. **Keterangan:** bagian ini berisi penjelasan singkat tentang attribute tampilan yang akan dipakai. Sebagai contoh: teks judul menggunakan font Times New Roman, 20 point, berwarna cyan, warna latar belakang biru tua.

II.2.10 *Usability Testing*

Dalam perancangan sistem interaktif, salah satu bagian dari proses untuk memahami kebutuhan pengguna adalah memahami dengan jelas tujuan utama pembuatan antarmuka tersebut. Pertanyaan yang sering muncul adalah apakah sistem yang dirancang akan sangat efisien sehingga dapat meningkatkan produktivitas kerja pengguna, atau sistem yang dirancang akan menantang dan mampu memotivasi pengguna sehingga mendukung pembelajaran yang efektif? Hal tersebut dapat disebut sebagai pertimbangan tujuan kebergunaan *(usability)* dan pengalaman pengguna *(user experience)* [6].

1. Kesalahan Kelasik

Seorang perancang sering melakukan kesalahan ketika sedang merancang sistem. Kesalahan-kesalahan tersebut antara lain disebabkan oleh kesalahan asumsi kepada rancangannya, pilihan-pilihan yang tidak secara tegas ditentukan, serta pembuatan fitur-fitur yang tidak memadai.

Berikut adalah kesalahan-kesalahan umum yang sering dilakukan oleh perancang sistem antara lain:

1. Perancangan yang didasarakan pada *common-sense.*
2. Anggapan bahwa perilaku seseorang telah mewakili suatu kelompok di mana dia berada.
3. Keinginan atasan yang harus dilakukan.
4. Kebiasaan dan/atau tradisi lama.
5. Anggapan implisist yang tidak sesuai/tidak didukung.
6. Keputusan evaluasi “sampai waktu luang”.
7. Evaluasi formal yang menggunakan kelompok subyek yang tidak sesuai
8. Eksperimen yang tidak dapat dianalisis.

2. Kepuasan Berinteraksi

Kepuasan berinteraksi merupakan salah satu kriteria penting untuk menentukan kebergunan dari sebuah sistem. Kebergunaan sebuah sistem dapat diuji dengan beberapa cara yang dapat digunakan penguji untuk mendapatkan data kuantitatif. Selain itu, investigasi terhadap aspek perancangan antarmuka secara kualitatif juga sering digunakan. Hal ini dilakukan terutama untuk menentukan

faktor-faktor yang lebih bersifat subyektif yang memberikan kontribusi kepada kepuasaan pengguna atau tingkat frustasi yang mungkin dialami pengguna.

Kepuasan berinteraksi dapat dicapai apabila sistem memnuhi delapan aturan (Shneiderman, 1998) berikut:

1. **Konsistensi.** Aturan ini merupakan aturan yang paling sering dilanggar, tetapi secara total mengikuti aturan inipun juga cukup rumit karena konsistensi dapat mempunyai bentuk yang berbeda. Beberapa hal yang harus konsisten yaitu urutan tindakan, penggunaan pesan, penggunaan warna, tata letak, jenis huruf, jenis *font* dan lain sebagainya.
2. **Fasilitas kunci-cepat.** Seiring dengan frekuensi penggunaan yang meningkat, pengguna berkeinginan untuk mengurangi jumlah interaksinya dan meningkatkan kecepatan berinteraksi. Fitur-fitur seperti singkatan, kunci-kunci khusus, perintah-perintah tersembunyi dan waktu tanggap yang singkat sangat disukai oleh pengguna sistem terutama oleh pengguna berpengalaman.
3. **Umpan balik yang informatif.** Untuk setiap tindakan yang pengguna lakukan didalam sistem harus ada umpan balik, baik yang sering dilakukan dan bersifat minor maupun yang jarang dilakukan. Penyajian secara visual dapat digunakan untuk menunjukan suatu perubahan yang bersifat eksplisit.
4. **Rancangan dialog yang mengarah ke penutupan *(closure)*.** Urutan tindakan yang pengguna lakukan harus dapat diorganisir ke dalam kelompok-kelompok dengan suatu bagian awal, bagian tengah, dan bagian akhir. Umpan balik yang informatif pada akhir setiap kelompok tindakan akan memberikan kepuasaan dan perasaan lega kepada pengguna.
5. **Pencegahan dan penanganan kesalahan.** Sejauh mungkin perlu diusahakan agar pengguna tidak membuat kesalahan serius. Jika pengguna melakukan kesalahan, sistem harus mampu mendeteksi kesalahan tersebut dan memberikan intruksi yang sederhana, sepesifik, dan konstruktif untuk melakukan pemulihan.
6. **Pembalikan tindakan yang mudah.** Sejauh mungkin suatu tindakan harus dapat dibalik *(revesible)*. Fitur ini akan mengurangi kecemasan

pengguna, karena pengguna mengetahui bahwa kesalahan yang mereka perbuat dapat “dibatalkan”. Hal ini akan memberikan semangan kepada pengguna untuk melakukan eksplorasi atas sejumlah pilihan yang belum mereka kenal.

7. **Dukungan pada *focus of control* internal.** Pengguna yang berpengalaman selalu mempunyai keinginan untuk merasa bahwa merekalah yang menguasai sistem dan bahwa sistemnya memeberikan tanggapan yang sesuai dengan tindakan mereka. tindakan sistem yang mengejutkan, urutan pengisian data yang bertele-tele, kesulitan untuk mendapatkan informasi yang diperlukan, dan ketidakmampuan untuk menghasilkan tindakan yang diinginkan akan menyebabkan kecemasan dan ketidak-puasan pengguna.
8. **Penggunaan beban memori jangka pendek.** Keterbatasan manusia untuk mengolah informasi pada memori jangka pendek mensyaratkan bahwa tampilah haruslah sederhana, tampilah halaman banyak harus dikonsolidasikan, frekuensi perpindahan dari satu jendela ke jendela lain perlua dikurangi, dan waktu pelatihan yang cukup untuk memahami sejumlah kode, *mnemonic,* dan urutan tindakan yang dimiliki oleh suatu sistem.

3. Definisi Kebergunaan

Kebergunaan dapat didefinisikan sebagai derajat kemampuan sebuah perangkat lunak untuk membantu penggunanya menyelesaikan sebuah tugas. Menurut Dix, et al., (2004), keberhasilan sebuah sistem untuk membantu penggunanya menyelesaikan suatu tugas ditentukan oleh kombinasi tiga kata “guna” yang ksesmuanya haru benar, yaitu:

1. Berguna *(useful)*: sistem yang berfungsi seperti yang diinginkan oleh penggunanya.
2. Dapat digunakan *(usable)*: sistem yang mudah dioperasikan.
3. Digunakan *(used)*: sistem yang memotivasi penggunanya untuk menggunakannya, menarik, menyenangkan, dan lain-lain.

Nielsen (2003) menyebutkan lima buah komponen kualitas untuk menentukan kebergunaan sebuah sistem, yaitu:

1. **Kemampuan untuk dipelajari *(learnability)*** menunjuk kepada kualitas sistem apakah mudah untuk dipelajari dan digunakan. Sudah menjadi rahasia umum bahwa pengguna tidak suka menghabiskan banyak waktu untuk mempelajari cara sistem bekerja. Mereka ingin dengan cepat menggunakan sistem tersebut dan merasa berkompeten untuk melakukan pekerjaan tanpa banyak kesulitan.
2. **Efisiensi** menunjuk kepada cara yang dapat dilakukan sistem untuk mendukung pengguna dalam melakukan pekerjaannya.
3. **Mudah diingat *(memorability)*** menunjuk kepada kemampuan sistem yang mudah diingat. Pengguna pemula yang jarang menggunakan sistem tidak banyak mengalami kesulitan apabila dia kembali menggunakan sistem tersebut meskipun dalam waktu yang cukup lama.
4. **Kesalan dan keamanan** melibatkan perlindungan kepada pengguna terhadap kondisi dan situasi yang tidak diinginkan dan berbahaya. Sistem harus mempunyai berbagai fasilitas yang dapat menghindarkan pengguna dari melakukan kesalahan yang tidak disengaja.
5. **Kepuasan** menunjuk kepada suatu keadan di mana pengguna merasa puas setelah menggunakan sistem tersebut karena kemudahan yang dimiliki oleh sistem. Dengan kata lain, semakin pengguna menyukai suatu sistem, secara impilisit mereka merasa puas dengan sistem yang dimaksud.

4. Uji Kebergunaan (Usability Testing)

Uji kebergunaan adalah proses untuk mengukur karakteristik interaksi manusia-komputer dari sebuah sistem. Uji kebergunaan juga digunakan untuk mengidentifikasi kelemahan-kelemahan antarmuka, sehingga rancang sistem dapat memperbaikinya secara tepat. Uji kebergunaan dapat dilakukan secara informal maupun secara menyeluruh dan ketat dari yang berbiaya sangat murah sampai yang memerlukan biaya yang sangat mahal, dan dari yang cepat sampai yang memakan waktu lama. Berbagai pendekatan tersebut ditujukan untuk menghasilkan sistem yang lebih baik.

Levi and Conrad (1997) menyebutkan ada tiga jenis uji kebergunaan, yaitu uji eksploratori, *threshold test*, dan uji perbandingan. Penjelasannya adalah sebagai berikut:

1. **Uji eksploratori** bertujuan untuk menguji sebuah sistem dan mencari titik-titik di mana pengguna mengalami kebingungan, kesalahan, atau unjuk kerjanya melambat. Pengujian ini dilakukan tanpa melihat di mana persolan-persoalan terjadi atau bentuknya. Tujuan akhir dari uji eksplorasi adalah daftar persoalan yang perlu ditinjau lebih lanjut, misalnya "pengguna secara jelas kebingungan ketika berada pada menu x; hanya separuh dari pengguna mampu menyelesaikan tugas y, tugas z memerlukan waktu lebih lama dari yang seharusnya.".
2. ***Threshold testing*** digunakan untuk mengukur kinerja sistem terhadap sejumlah sasaran yang ditentukan terlebih dahulu.Uji ini merupakan uji **lolos** atau **gagal**. Misalnya "Dengan sistem ini pengguna mampu menyelesaikan pekerjaan *x* dalam waktu *y* detik, melakukan kesalahan rata-rata sebanyak *z* kali. Hal ini tidak sesuai dengan kriteria yang telah ditetapkan.", jenis uji ini biasanya menyertai *beta release.*
3. **Uji perbandingan** bertujuan untuk mengukur karakteristik kebergunaan dari dua pendekatan atau rancangan untuk menentukan rancangan yang lebih cocok bagi pengguna. Hal ini biasanya dilakukan pada awal tahap pembuatan perwarupa.

5. Cara Uji Kebergunaan

Ada berbagai cara yang dapat digunakan untuk melakukan uji kebergunaan/*usability testing,* diantaranya ada pemilihan kartu, evaluasi heuristik dan uji berbasis skenario [6]. Pada penelitian ini pengujian akan menggunakan metode Evaluasi Heuristik. Metode Heuristik banyak digunakan dalam mengukur tingkat kenyamanan pengguna.

Evaluasi Heuristik adalah panduan, prinsip umum, atau aturan yang dapat menentun keputusan rancangan atau digunakan untuk mengkritik suatu keputusan yang sudah diambil. Evaluasi Heuristik diusulkan oleh Nielsen dan Molich, hampir sama dengan *Cognitive Walkthrough* tetapi sedikit terstruktur dan sedikit terarah [4].

Tujuan dari evaluasi heuristik adalah untuk memperbaiki perancangan secara efektif. *Evaluator* melakukan evaluasi melalui kinerja dari serangkaian tugas dengan perancangan dan dilihat kesesuaiannya dengan kriteria setiap

tingkat. Jika ada kesalahan terdeteksi maka perancangan dapat ditinjau ulang untuk memperbaiki masalah ini sebelum tingkat implementasi.

Evaluasi Heuristik sangat baik digunakan sebagai teknik evaluasi desain, karena lebih mudah untuk menemukan atau menentukan masalah *usability* yang muncul. Untuk menggunakan evaluasi ini dibutuhkan *software* yang akan diteliti atau *storyboard* untuk sistem yang akan dibuat [4].

Dalam melakukan evaluasi, terdapat sepuluh perinsip dan Evaluasi Heuristik, yaitu:

1. Visibilitas dari status sistem *(Visibility of system status (feedback)),* sistem harus selalu menginformasikan pada pengguna apa yang sedang terjadi, melalui pesan yang baik dan waktu yang sesuai.
2. Kesesuaian antara sistem dan dunia nyata *(Match between system and the real world),* sistem harus berbicara sesuai dengan bahasa penggunanya, menggunakan kata, kalimat, dan konsep yang biasa digunakan oleh pengguna.
3. Kendali dan kebebasan pengguna *(User Control and Freedom),* pengguna harus dapat secara bebas memilih dan melakukan pekerjaan (sesuai kebutuhan). Pengguna harus dapat mengambil keputusannya sendiri (dengan informasi yang jelas) berkaitan dengan pekerjaan yang sedang/akan dilakukan. Sistem harus memiliki kemampuan untuk *undo* dan *redo.*
4. Standar dan konsistensi *(Consistency and Standards),* pengguna tidak perlu mempertanyakan lagi mengenai pervedaan pemahaman pada sebuah kata dan kalimat, situasi dan aksi. Semua harus sudah mengikuti standar yang ada.
5. Pencegahan kesalahan *(Error Prevention),* merancang sistem yang mencegah terjadinya kesalahan lebih baik daripada merancanga pesan kesalahan yang baik.
6. Bantu pengguna untuk mengenali, men-diagnosa, dan mengatasi masalah *(Recognation Rather than Recall).* Pengguna tidak perlu mempertanyakan lagi mengenai perbedaan pemahaman pada sebuah kata dan kalimat, situasi dan aksi. Semua harus sudah mengikuti standar yang ada.

7. Fleksibilitas dan efisiensi *(Flexibility and Efficient of Use),* bagaimana membuat sebuah sistem yang mengakomodasi pengguna yang sudah ahli dan pengguna yang masih pemula. Berikan alternative untuk pengguna yang "berbeda" dari pengguna biasa (secara fisik, budaya, bahasa, dll).
8. Estetika dan desain yang minimalis *(Aesthetic and Minimalist Design).* Sistem hanya menghasilkan informasi yang relevan, informasi yang tidak relevan mengurangi visibilitas dan *usability* dari sistem.
9. Pertolongan pengguna mengenal, berdialog dan memperbaiki kesalahan *(Help user recognize, dialogue, and recovers from errors).* Pembuatan objek, aksi dan pilihan harus jelas terlihat. Pengguna tidak harus mengingat-ingat informasi dari satu halaman ke halaman lain. Instruksi dan informasi pada sistem harus mudah diakses dan jelas terlihat pada saat dibutuhkan.
10. Fitur bantuan dan dokumentasi *(Help and Documentation),* sistem harus memiliki dokumentasi yang relevan dan fitur *help* yang baik, sehingga pengguna dapat mempelajari segala sesuatu yang terkait dengan sistem.

II.2.11 *Flutter*

Flutter adalah sebuah *framework* untuk pengembangan aplikasi yang dibangun oleh *Google* untuk membuat aplikasi mobile *cross-platform* pada *iOS* dan *Android.* Seperti yang disebutkan di situs web resmi *Flutter,* tujuan utama dibangunnya *Flutter* adalah untuk membuat pengembangan aplikasi menjadi lebih mudah, cepat dan seproduktif mungkin. Fitur seperti *Hot-Reload,* katalog *widget,* kinerja yang sangat baik dan komunitas yang solid dalam berkontribusi menjadikan *Flutter* sebagai *framework* yang cukup bagus [14].

Semua yang ada di *Flutter* dibuat menggunakan *widget.* Orientasi, layout, opacity, animasi semuanya adalah sebuah *widget. Widget* adalah fitur utama dari *Flutter,* dan segalanya dimulai dari tombol yang sederhana hingga ke animasi atau gestur dilakukan menggunakan *widget* [14]