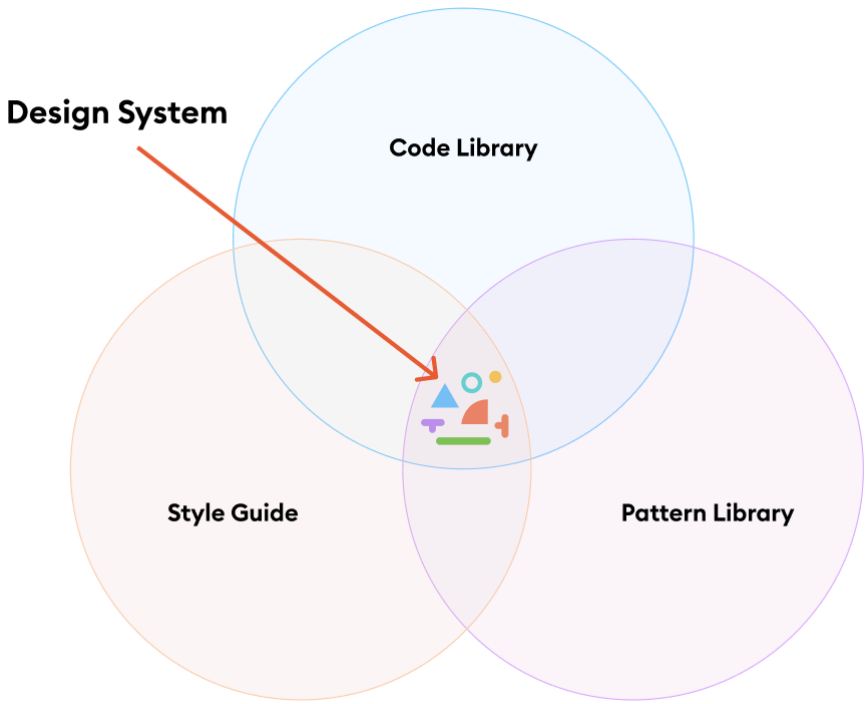
BAB 2
LANDASAN TEORI

2.1 Design System

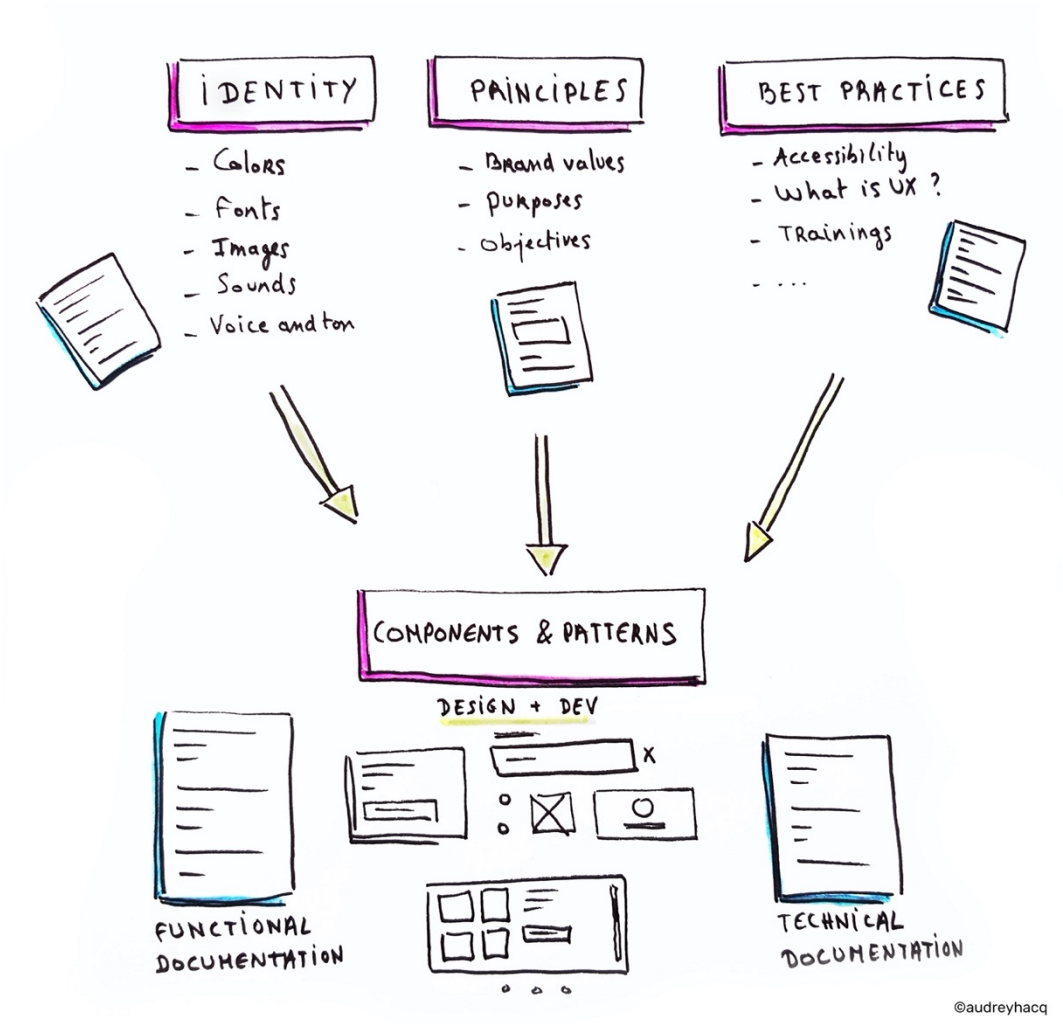
Design System adalah seperangkat pola yang saling berhubungan serta langkah sistematis dan kolaboratif yang diatur secara koheren untuk mempercepat dan mempermudah proses desain dan *development* dalam mencapai tujuan produk digital [3]. *"Design System"* memiliki tujuan untuk membuat suatu standarisasi desain serta menyediakan dokumentasi yang meliputi *code library*, *pattern library* dan *styleguide* sehingga dapat digunakan oleh seluruh tim dalam mengembangkan perangkat lunak dengan solusi yang sudah ada *(Reusable solution).*

Design System berisikan komponen UI yang dapat digunakan kembali untuk membantu tim pengembang dalam membangun antarmuka yang kompleks, *durable,* dan dapat diakses diseluruh projek karena *developer* dan *designer* berkolaborasi dalam menentukan komponen UI.



Gambar 2-1 Design System Architecture

Design System meliputi dokumentasi yang mewakili *brand identity, design principle,* pengkodean, *design language* serta *pattern library* [5]. Design system dibangun oleh beberapa artefak seperti gambar berikut.



Gambar 2-2 Design System Artifact

2.1.1 Brand Identity

Brand identity adalah nama dan/atau simbol pembeda (seperti logo, merek, atau paket desain) yang dimaksudkan untuk mengidentifikasi produk dan untuk membedakan produk tersebut dengan pesaing lainnya. *Brand identity* bertujuan untuk memberi sinyal kepada pengguna dan produsen untuk melindungi produk dari pesaing apabila produk tersebut terlihat identik [7]. *Brand identity* menyediakan *mental model* sehingga pengguna dapat terikat dengan suatu produk

atau layanan. Dari sudut pandang pengguna, *brand identity* dapat didefinisikan sebagai akumulasi jumlah pengalaman dan hal tersebut membangun *point view* dengan pengguna [8].

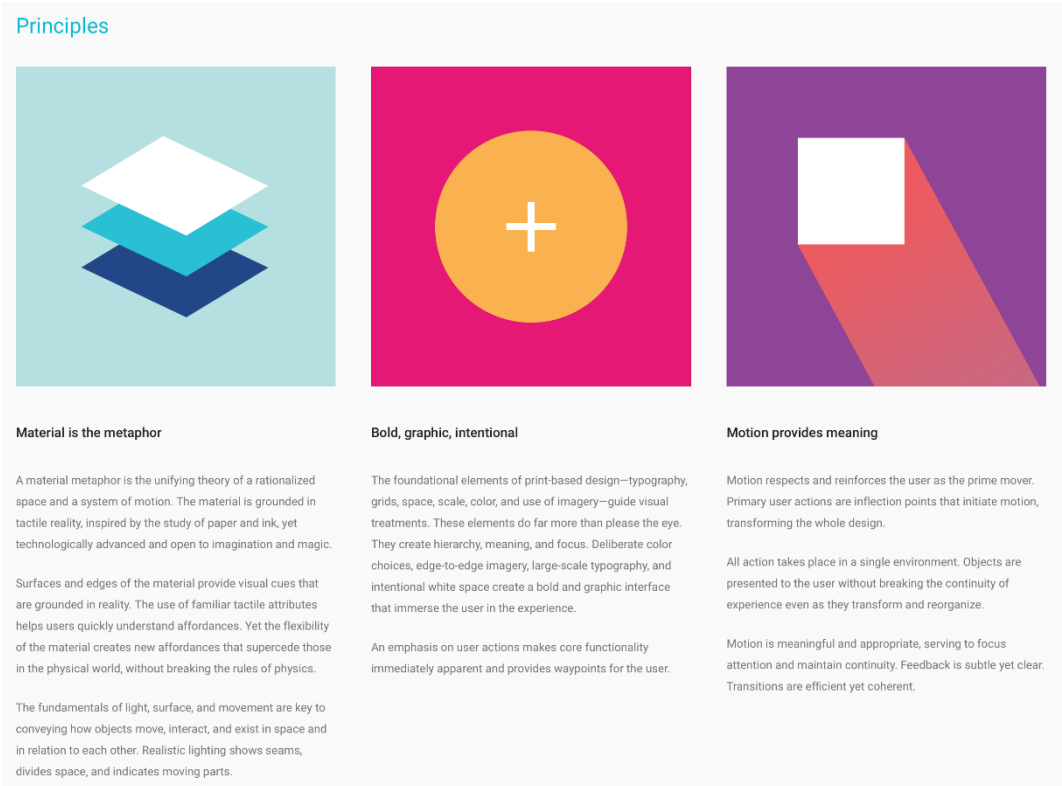
Gambar 2-3 Brand Identity

Sangat penting bagi suatu brand untuk dimunculkan dengan cara kohesif diantara semakin banyak media, *channel,* dan *touchpoints. Brand identity* menyediakan panduan bagaimana setiap orang dalam suatu organisasi dapat berbicara dalam satu suara dan merasakan suatu entitas tunggal, serta memberikan panduan warna apa yang harus digunakan dan bagaimana cara menggunakan logo dengan benar.

2.1.2 Design Principle

Sebelumnya telah dijelaskan bahwa *brand identity* memberikan identitas suatu produk dengan mudah. Namun, *brand identity* saja tidak cukup sehingga dibutuhkan suatu prinsip desain dalam merancang suatu produk yaitu *design principle. Design principle* bertujuan untuk mengartikulasikan arah, filosofi, dan pendekatan desain umum untuk suatu projek atau produk digital [5].

Untuk menunjukan suatu produk secara kohesif agar dapat bersaing diantara banyak media, salah satu contoh perusahaan yaitu Google membangun *design language* yang disebut *material design*. Panduan *material design* mendefinisikan filosofi desain, tujuan dan prinsip-prinsip umum, serta menyediakan aplikasi sepsifik dari *material design.*



Gambar 2-4 Google Material Design Language

2.1.3 Voice and Tone

Voice atau suara adalah aspek utama identitas produk, sehingga disebut dengan *brand voice*, sehingga panduan *brand identity* menyertakan beberapa

referensi *brand voice.* Namun, *voice* dalam identitas produk saja tidak cukup, sehingga *voice* harus selalu terikat dengan *tone* [5].

Voice and tone harus bisa mengarahkan bagaimana nada suatu perusahaan dapat beradaptasi dengan berbagai situasi. Menurut Mail Chimp, bagaimana cara menentukan nada suatu produk berubah ketika berbagai situasi, misalnya pengguna tidak bisa menggunakan kartu kredit karena ditolak, maka *voice and tone* dari produk tersebut harus memberikan kesan nada yang serius dan lebih jelas mengapa kartu kredit pengguna tersebut tidak dapat digunakan [9].

2.1.4 Shared Language

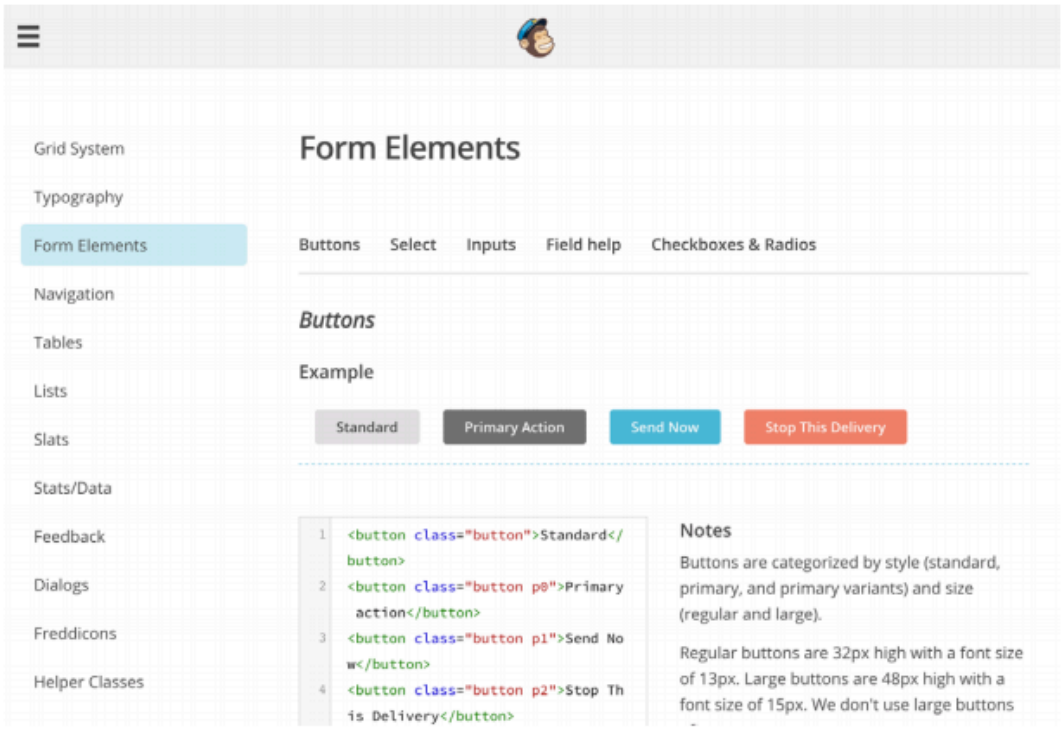
Bahasa sangat penting dalam berkolaborasi. Jika bekerja dalam sebuah tim, *shared language* perlu dibagikan kepada orang-orang yang terlibat dalam tim ketika pembuatan produk. Tanpa *shared language,* setiap orang akan memiliki pemikiran yang berbeda serta setiap orang akan memiliki mental model yang berbeda dari apa yang ingin mereka capai [3].

Menurut Abby Covert, *shared language* harus dibuat sebelum desiner mulai memikirkan tampilan antarmuka, dengan mendiskusikan, memeriksa dan mendokumentasikan keputusan bahasa seperti apa yang akan disepakati. Hal ini dilakukan untuk menggambarkan *hi-level concept* serta istilah-istilah bahasa yang akan digunakan tim dalam berbicara tentang *design decisions* [10]. *Shared language* dibuat untuk mengindari ketidak sepemahaman antara anggota didalam sebuah tim. Misalnya penggunaan komponen *button,* setiap orang tidak cukup memiliki pemahaman yang sama tentang *button.* Tim juga harus tahu mengapa dan bagaimana menggunakan *button,* digunakan dalam konteks apa dan tujuan dari *button* itu sendiri. Misalnya penggunaan elemen yang disebut *"sequence".* Dengan mempresentasikan kata "*sequence",* tim bertujuan untuk berkomunikasi dengan pengguna bahwa langkah-langkah yang dilakukan oleh pengguna harus dilihat dalam urutan tertentu.

2.1.5 Pattern Library

Design system tidak hanya terdiri dari pattern, tapi juga mencangkup teknik dan praktik dalam membuat, menangkap, *sharing*, dan mengembangkan pattern tersebut. *Pattern library* merupakan *tool* untuk mengumpulkan, menyimpan dan membagikan *pattern* desain, bersamaan dengan prinsip-prinsip dan pedoman cara menggunakannya. Konsep mendokumentasikan dan *sharing pattern library* telah ada sejak lama.

Pattern library dimulai sebagai identitas brand yang diperluas guna berfokus pada logo treatments, value perusahaan dan brand style seperti typography dan color palettes. Kemudian pattern library diperluas dengan menyertakan komponen dan modul antarmuka, serta pedoman cara penggunaanya.



Gambar 2-5 MailChimp's Pattern Library

Pattern library merupakan bagian dari *design system* sebagai *tools* yang dapat membantu membangun *design system*. *Pattern library* tidak menjamin dalam menghasilkan desain produk yang kohesif. Tapi lebih membantu untuk mengoreksi

ketidak konsistenan pada desain dan membuat *basecode* yang lebih kuat. Sehingga jika hanya menyajikan *pattern library* sebagai *tools* saja, akan memiliki dampak yang sangat kecil pada pengalaman pengguna.

2.2 Standards

Suatu standarisasi pada *design system* sangat penting untuk menciptakan pengalaman pengguna yang bagus. Mendefinisikan dan mematuhi standar adalah bagaimana kita menciptakan pemahaman. Melakukan hal itu menghilangkan subjektivitas dan ambiguitas yang sering menciptakan gesekan dan kebingungan dalam tim produk [11].

Standar mencakup desain dan *development*. Membakukan hal-hal seperti konvensi penamaan, persyaratan aksesibilitas, dan struktur file akan membantu tim bekerja secara konsisten dan mencegah kesalahan. *Design language* adalah bagian inti dari standar desain. Menentukan tujuan dan gaya warna, bentuk, jenis, ikon, *space*, dan *interaction* adalah hal penting untuk menciptakan pengalaman pengguna yang selaras dengan *brand* dan konsisten. Setiap komponen dalam *design system* menggabungkan elemen-elemen ini, dan memainkan peran dalam mengekspresikan kepribadian suatu *brand* [11].

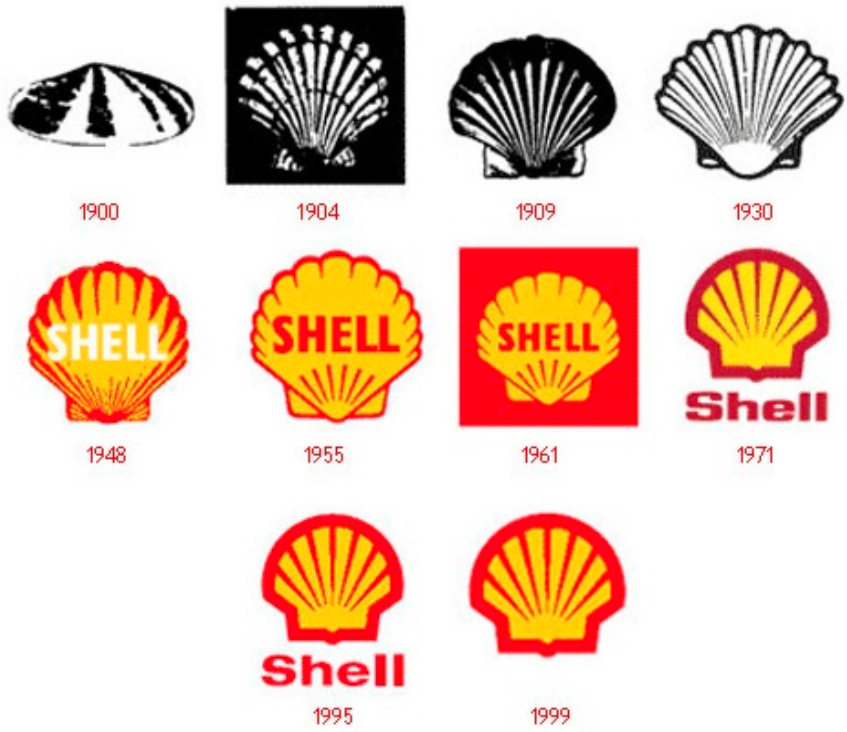
Tanpa standar, keputusan menjadi sewenang-wenang dan sulit dikritik. Bukan hanya sulit untuk diukur, tetapi juga menciptakan pengalaman pengguna yang tidak konsisten dan membuat frustrasi bagi pengguna [11].

2.3 Consistency

Consistency adalah unsur utama dalam desain, desain yang konsisten adalah desain yang intuitif. Hal ini sangat berguna dalam mengembangkan produk. *Consistency* bertujuan untuk meningkatkan kemampuan belajar ketika pengguna menemukan elemen desain yang serupa dan memiliki fungsi yang sama. *Consistency* pada desain dapat mentransfer pengetahuan pengguna kepada konteks baru dan belajar hal-hal baru tanpa mengurangi kualitas *experience* pengguna.

Consistency sangat penting untuk mengurangi *load coginitive* pada tampilan antarmuka [12]. Jika *consistency* diutamakan pada desain, maka interkasi antara

pengguna dengan perangkat lunak menjadi lebih baik. Namun, ketika sebuah desain terdapat *inconsistent,* maka pengguna ketika berinteraksi dengan perangkat lunak harus mengeluarkan *effort* yang tidak perlu [12].



Gambar 2-6 Consistency Shell Logos

Ketika antarmuka aplikasi bekerja dengan konsisten, maka akan semakin mudah untuk diprediksi (dalam arti baik), yang berarti pengguna dapat memahami cara menggunakan fungsi tertentu secara intuitif dan tanpa harus ada panduan. Hal ini membuat produk mudah untuk digunakan. Sebaliknya, ketika tampilan antarmuka tidak konsisten, akan sulit untuk dipelajari, memicu rasa frustasi pengguna dan mengarah pada pengalaman yang buruk. *Consistency* pada desain memberikan beberapa manfaat diantaranya:

1) Pengguna akan belajar lebih cepat cara menggunakan desain aplikasi
2) *Consistency* menghilangkan kebingungan pada pengguna
3) *Consistency* menghemat waktu dan biaya

2.3.1 External Consistency

External consistency sangat penting karena menurut Jared Spool, *external consistency* bertujuan untuk meningkatkan tampilan antarmuka dengan memenuhi harapan pengguna. Untuk membuat desain yang sesuai dengan apa yang diharapkan pengguna, hal yang harus diutamakan adalah memahami harapan pengguna sebelum menggunakan desain [12].

2.3.2 Internal Consistency

Jika *external consistency* dapat meningkatkan tampilan antarmuka dengan memenuhi harapan pengguna, *internal consistency* lebih kepada persyaratan keberhasilan suatu desain. Jika tidak memenuhi persyaratan maka kemungkinan desain yang dihasilkan akan memberikan pengalaman yang buruk [12]. *Inconsistencies internal* biasanya digunakan untuk memberdakan perbedaan dalam konten atau tindakan pengguna mislanya, layout yang berbeda untuk berbagai bentuk konten yang sama).

2.3.3 Method for Consistency

Menurut Jakob Nielsen, terdapat beberapa metode untuk mencapai konsistensi pada desain [13] yang dapat dilihat pada **Tabel 2-1**.

Tabel 2-1 Metode untuk Consistency

Metode	Keterangan
Centralized Consistency	Sebuah metode *consistency* dengan membuat grup pusat dalam menetapkan standar perusahaan dan meminta *developer* untuk mematuhinya.
User-defined Consistency	Metode ini berfokus pada pendefinisian awal dari pengguna, misalnya suatu perusahaan membuat sebuat standar yang disesuaikan dengan perangkat lunak dari *vendor* dan menyesuaikannya agar konsisten.

Metode	Keterangan
Exemplary Applications	Metode untuk membuat standar desain yang mengacu kepada desain antarmuka yang sudah familiar dengan pengguna, sehingga ketika mengembangkan suatu aplikasi, maka dilakukan perancangan dan meniru aplikasi yang sudah umum.
Promotion of Consistencty	*Developers* disadarkan akan perlunya konsistensi melalui *"Art Exibits"* dari antarmuka yang konsisten, yang artinya dilakukan edukasi dan pemaparan tentang pentingnya konsistensi dalam membangun aplikasi.
Shared Code	Program dibuat konsisten karena *developer* menggunakan kode yang sama untuk mengimplementasikan desain antarmuka. Hal ini dapat dilakukan dengan menggunakan *tools* seperti *code library,* UIMS atau pustaka kode yang tersentralisasi.
Hardware support for consistency	Memastikan bahwa semua pengguna memiliki perangkat yang sama sehingga semua aplikasi akan menggunakan perangkat tersebut.
Style-based consistency	*Style* antarmuka ditentukan dan diterapkan secara terpisah dengan konten antarmuka. Dengan cara yang sama seperti ketika menggunakan *style sheet* dalam pengolahan kata.
Model analysis	Sebuah metode berupa model perkiraan yang dibuat untuk interaksi pengguna terhadap pengguna. Misalnya menggunakan sistem produksi atau

Metode	Keterangan
	metode lain penelitian yang sudah dikembangkan dibidang *user interface.* Setelah menggunakan beberapa model yang digunakan, kemudian dilakukan perbandingan secara otomatis untuk melihat perbedaannya.

2.4 User Experience dan User Interface

Consistency merupakan salah satu elemen penting untuk mencapai user erperience yang baik. User experience dapat dikatakan sebagai bagaimana perasaan pengguna tentang suatu produk serta kesenangan dan kepuasan mereka saat menggunakannya, melihatnya, mendengarnya, berbicara dengannya, dan sebagainya. User experience mencakup kesan pengguna mengenai seberapa baik suatu produk digunakan, termasuk kesan mereka terhadap efek sensual atau detail kecil produk, seperti bunyi klik dan efek sentuhan pada tombol saat menekannya [14]. Sedangkan, apa yang pengguna lihat, dengar, ataupun sentuh ketika menggunakan produk, termasuk efek sensual atau detail kecil itulah yang disebut dengan user interface [15].

2.5 In Depth Interview

In depth interview merupakan kegiatan berbicara secara langsung dan/atau melakukan observasi terhadap sekelompok pengguna representatif untuk mendapatkan data pengguna tersebut [11]. Pengguna representatif merupakan sekelompok kandidat yang berpotensi untuk menggunakan produk dikarenakan kebutuhan mereka yang dapat terpenuhi oleh kehadiran produk tersebut [6]. Kegiatan tersebut dilakukan di dalam konteks kehidupan mereka atau lingkungan regular mereka. Dengan dilakukan user research, resiko yang diakibatkan oleh asumsi buruk (dikarenakan hanya berdasar pada pengalaman pribadi) dapat diminimalisir [11].

Menurut Alan Cooper, data yang dikumpulkan melalui wawancara untuk user research meliputi aktivitas, tujuan, motivasi, dan frustasi atau pain points pengguna terkait konteks penggunaan teknologi, dan domain. Adapun data yang dipelajari dari pengguna melalui wawancara di antaranya adalah sebagai berikut [6]:

1. Konteks mengenai bagaimana suatu produk (atau sistem sejenis) berkesinambungan dengan kehidupan pengguna atau alur kerja pengguna. Hal ini meliputi kapan, mengapa, dan bagaimana produk digunakan atau akan digunakan.
2. Pengetahuan domain berdasarkan sudut pandang pengguna. Hal ini meliputi hal-hal yang pengguna ketahui dalam melakukan pekerjaannya.
3. Aktivitas pengguna meliputi aktivitas-aktivitas yang didukung oleh produk maupun yang tidak didukung oleh produk.
4. Mental model pengguna meliputi bagaimana pengguna berpikir mengenai aktivitas mereka dan bagaimana ekspektasi mereka terhadap hal tersebut.
5. Masalah dan frustasi pengguna mengenai produk (atau sistem sejenis) yang mendukung aktivitas mereka.

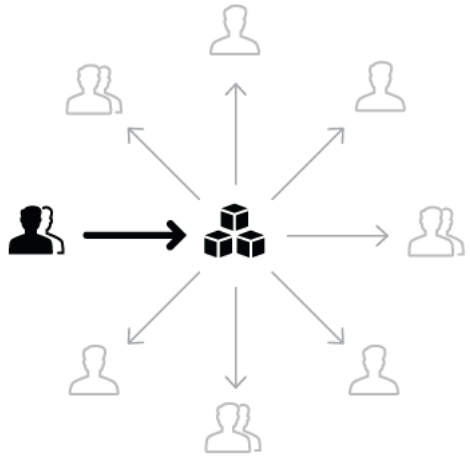
2.6 Team Model

Dalam membangun *design system* diperlukan tim yang tepat, tim tersebut terdiri dari beberapa *role* yang memiliki bidang pekerjaan yang berbeda seperti desainer, *frontend engineer, researchers, information architects* dan *role* lainnya. Setiap *role* dapat memberikan persfektif pandangan dari masing-masing *role* yang dapat menyumbang berbagai aspek dari *design system.* Bagi desainer dapat mendefinisikan elemen visual dari *design system* sedangkan bagi *frontend engineer,* berperan dalam membangun *modular code* yang efisien.

Menurut Nathan Curts, terdapat beberapa *team model* dalam membawa setiap orang sebagai komponen penting dalam membangun *design system* [11] yaitu sebagai berikut:

2.6.1 The Solitary Team Model

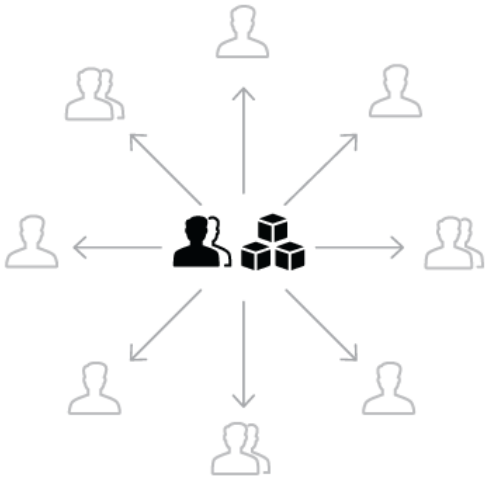
Terdapat satu orang yang berperan sebagai “ujung tombak” yang mengelola *design system.* Dia berperan sebagai jembatan yang menghubungkan setiap role yang ada pada tim.



Gambar 2-7 The Solitary Team Model

2.6.2 The Centralized Team Model

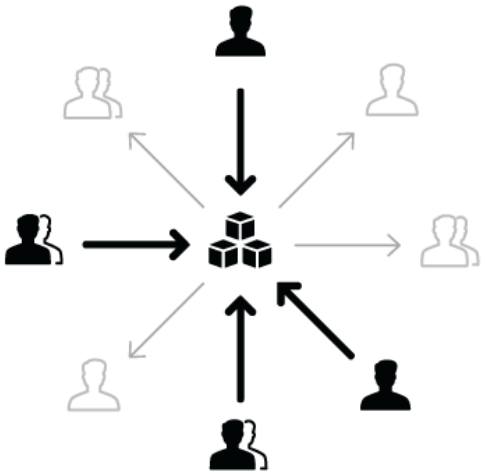
Team Model ini terdapat satu tim khusus yang ditugaskan untuk membangun *design system.* Tim ini akan bertanggung jawab dalam membangun *design system* dan memelihara *design system* sebagai *full time job.*



Gambar 2-8 The Centralize Team Model

2.6.3 The Federated Team Model

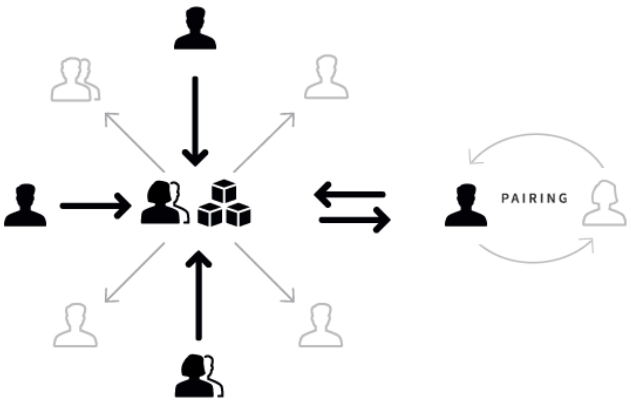
Team Model seluruh anggota *design team* dalam sebuah perusahaan dapat berkontribusi untuk membangun dan mengelola *design system.*



Gambar 2-9 The Federated Team Model

2.6.4 The Cyclical Team Model

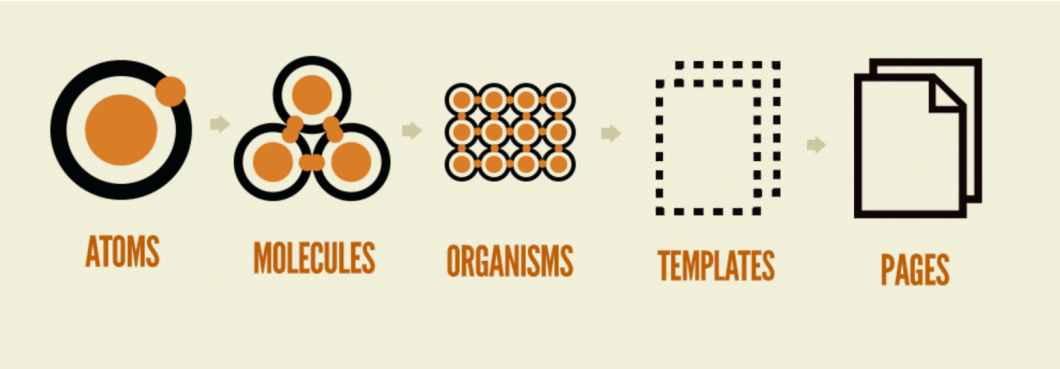
Team Model ini gabungan antara Centralized dan Federated, dimana seluruh anggota dapat berkontribusi untuk membangun design system seperti menambahkan komponen dan lainnya, namun juga terdapat tim khusus yang bertugas mengelola seluruh *design system*.



Gambar 2-10 The Cyclical Team Model

2.7 Atomic Design Methodology

Atomic design methodology merupakan suatu model untuk merepresentasikan *user interfaces* sebagai kumpulan komponen yang hirarkis secara informasi dan fungsionalitas, serta setiap komponen saling terhubung sehingga dapat membangun suatu produk digital [5]. *Atomic design* adalah metodologi yang terdiri dari lima tahap berbeda yang bekerja bersama untuk membuat *design system* dengan cara yang sistematis dan hierarkis. Metodeologi *atomic design* terbagi menjadi 5 tahapan seperti pada **Gambar 2-11** dibawah ini.

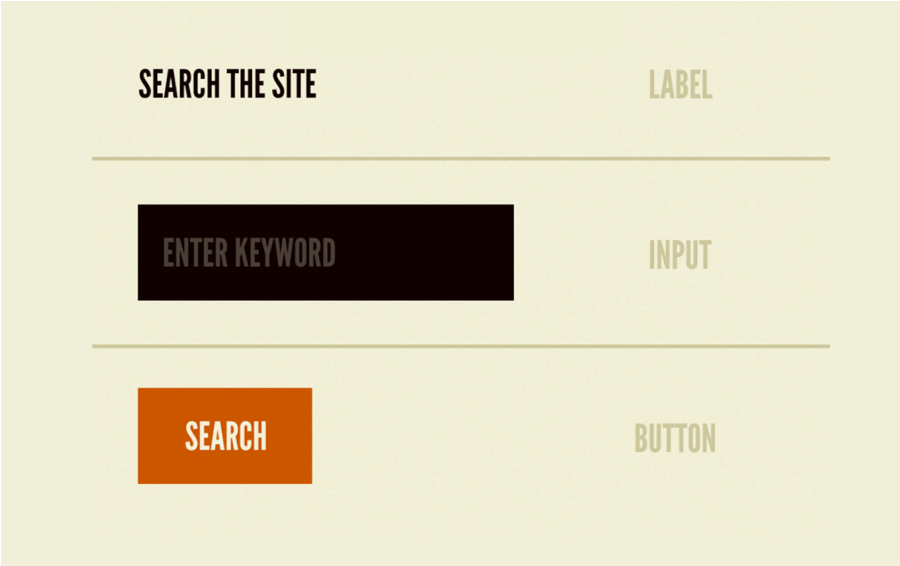


Gambar 2-11 Atomic Design Methodology

Atomic design bukan proses linear, melainkan mental model untuk membantu memikirkan *user interfaces* sebagai satu kesatuan yang bersifat kohesif dan kumpulan komponen yang terorganisir dalam waktu bersamaan. Berikut merupakan penjelasan dari lima tahapan pada *atomic design*:

1. Atoms

*Atom*s adalah pondasi antarmuka yang berfungsi sebagai blok bangunan dasar yang terdiri dari komponen elemen antarmuka. Atom mencangkup elemen HTML dasar seperti label, form, inputfield, *button* dan komponen lainnya yang tidak dapat dipecah menjadi bagian terkecil.



Gambar 2-12 Atoms Stages

2. Molecules

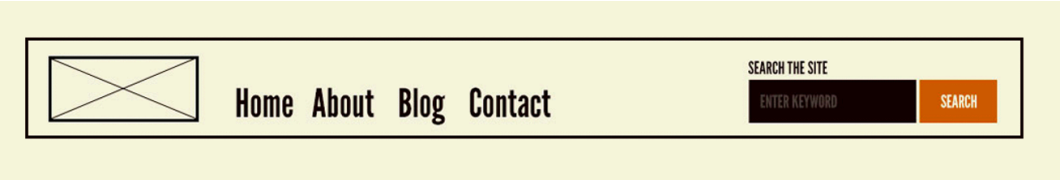
Molecules adalah kumpulan atom yang dikelompokan sebagai elemen *user interfaces* yang berfungsi sebagai satu unit [5]. Molekul akan mengambil elemen-elemen atoms yang akan digabungkan bersama sehingga memiliki fungsi spesifik.

Gambar 2-13 Molecules Stages

Ketika atom label, input dan *button* digabungkan, maka atom-atom tersebut memiliki fungsi. Seperti contoh label yang mendefinisikan input dan *button* yang berfungsi meng-*submit* isian form dalam input. Luaran yang dihasilkan adalah molekul form pencarian yang merupakan *component reusable* sehingga dapat digunakan kapanpun dibutuhkan.

3. Organisms

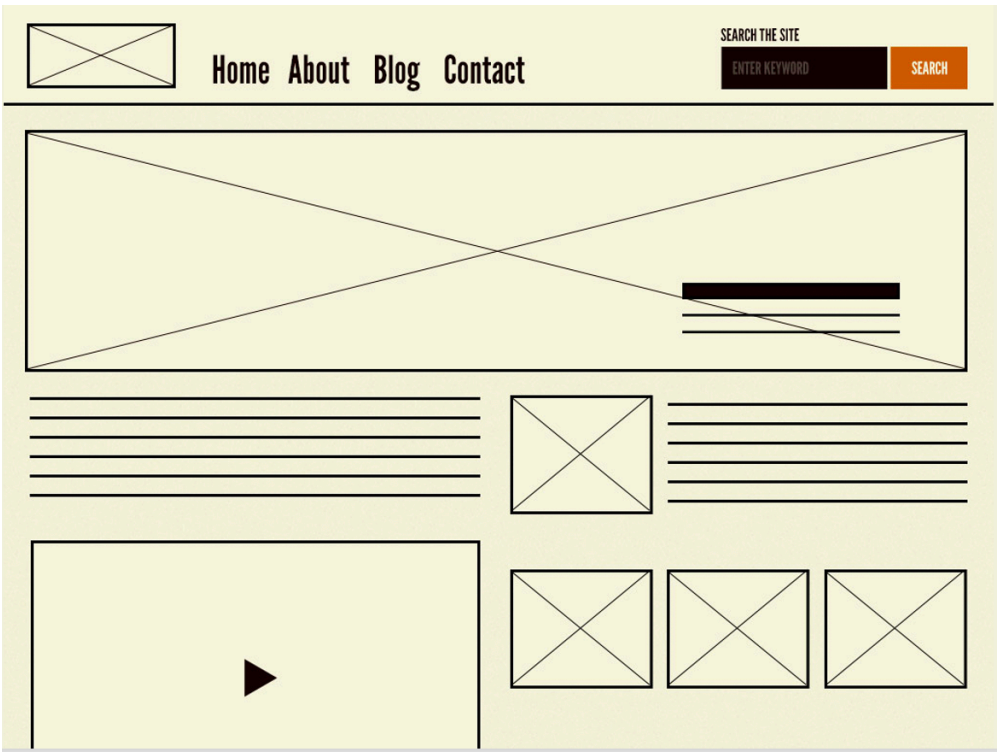
Organisms adalah komponen *user interfaces* yang relatif kompleks yang terdiri dari kelompok *molecules* dan / atau atom dan *atoms* dan / atau *organisms* lain [5]. *Organism* ini membentuk bagian yang berbeda dari suatu antarmuka.



Gambar 2-14 Organisms Stages

4. Templates

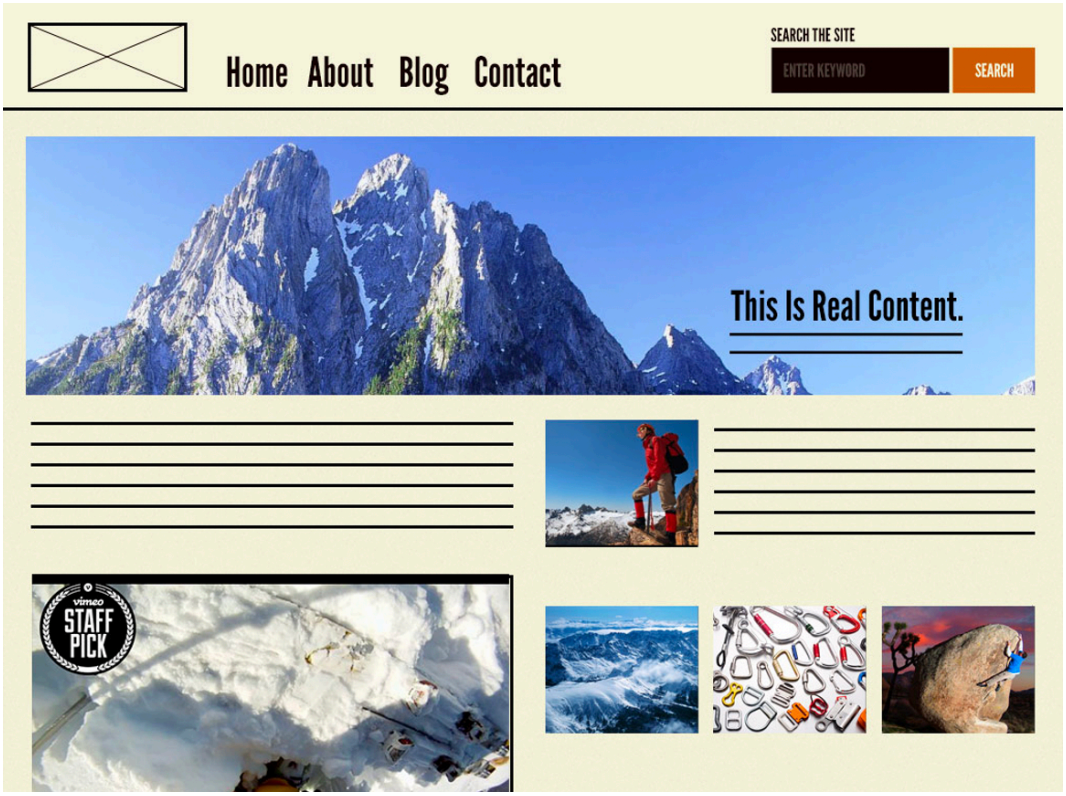
Templates adalah *page-level object* yang menempatkan komponen kedalam halaman *layout* dan mengartikulasikan sturktur konten yang mendasari desain antarmuka [5]. Dengan kata lain, *templates* adalah kerangka dasar dari kumpulan *organisms* yang akan menjadi tampilan utama suatu antarmuka.



Gambar 2-15 Templates Stages

5. Pages

Pages adalah contoh khusus dari suatu template yang menampilkan seperti apa tampilan *user interfaces* dengan konten representative nyata. *Pages* sangat penting karena *pages* ini yang nantinya akan dilihat oleh *end user* kita, bagaimana user nanti akan berinteraksi.



Gambar 2-16 Pages Stages

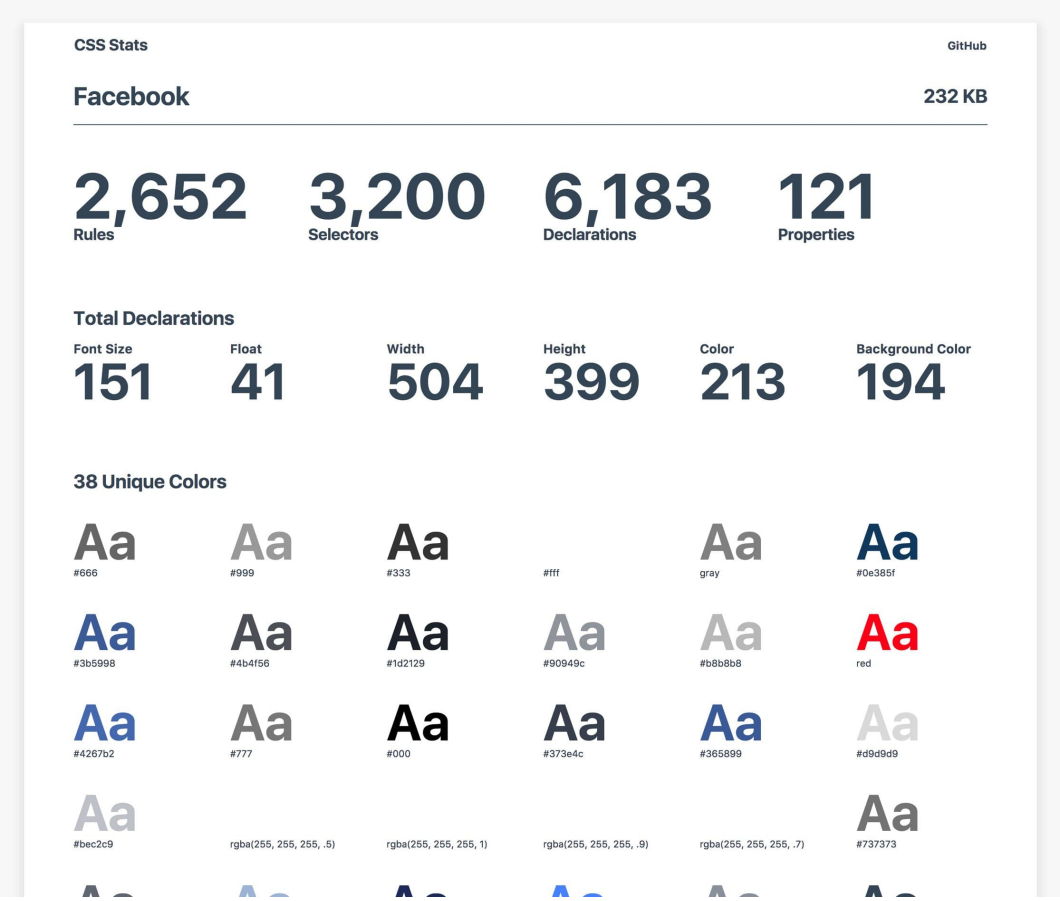
2.8 Strategizing Roadmap

Dengan terbentuknya team model dan harapan yang diterapkan, diperlukan roadmap untuk merencanakan pekerjaan untuk kedepannya. Roadmap adalah gambaran umum hi-level tentang sasaran dan hasil yang disajikan dalam bentuk timeline. Roadmap dalam design system terdapat 4 elemen penting untuk membangun dan memelihara design system yaitu: *research, audit, create* dan *maintaining*.

Gambar 2-17 Basic Roadmap dalam membangun design system

2.9 Audit Design

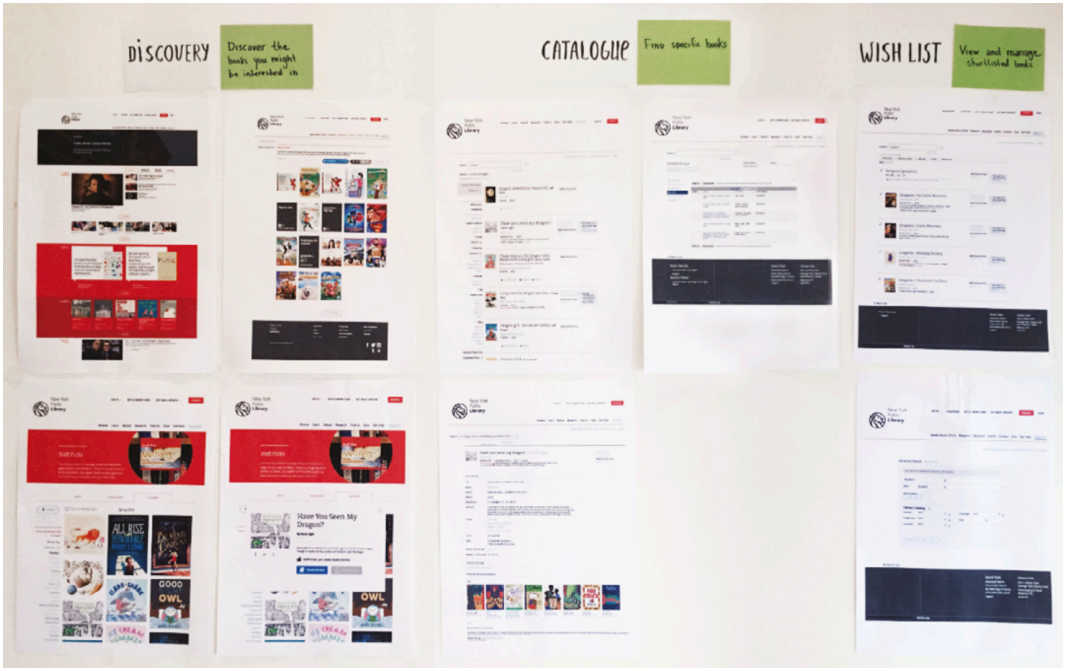
Setelah melakukan *research*, dilakukan *audit design* terhadap tampilan sistem atau aplikasi saat ini dan mendokumentasikannya kedalam suatu *tools* yang disebut *interface inventory*. Hal ini penting dilakukan karena dapat mengetahui berapa banyak aturan, properti, elemen visual dan *stylesheet* yang dimiliki oleh sistem tersebut. Hal ini merupakan cara terbaik dalam menyeleksi atau menggabungkan nilai elemen visual. Terdapat tools yang dapat meng-ekstraksi aturan *syle* desain pada suatu sistem yang disebut dengan CssStats.



Gambar 2-18 Audit Design pada Facebook

2.9.1 Identifikasi Key Behaviour

Identifikasi *key behaviour* adalah serangkaian aktivitas untuk mengidentifikasi kebutuhan utama pengguna dan perilaku yang dilakukan disetiap segmen *journey* pengguna dalam menggunakan produk [3]. Proses yang dilakukan adalah mengidentifikasi *behaviour* suatu halaman informasi pada aplikasi, kemudian dibuat *labeling* dan journey sesuai dengan fungsi dari halaman tersebut.



Gambar 2-19 Proses Labeling Key Behaviour [3]

Adapun label *behaviour* yang dapat digunakan yaitu sebagai berikut:

Tabel 2-2 Label Identifikasi Kunci Perilaku

Label	**Behaviour**
Discovery	Perilaku pengguna yang dilakukan ketika membuka halaman aplikasi adalah memindai dan menelusuri informasi yang diingingkan.
Catalog	Perilaku pengguna yang dilakukan adalah melihat list suatu informasi dan melihat informasi tersebut.
Finding	Perilaku pengguna yang dilakukan adalah mencari informasi tertentu dalam suatu halaman aplikasi.
Information	Perilaku pengguna yang dilakukan adalah menerima informasi dari suatu halaman aplikasi.

Label	Behaviour
Managing	Perilaku pengguna yang dilakukan adalah mengelola informasi seperti menambah, menghapus dan mengubah suatu informasi pada halaman aplikasi.

2.9.2 Break Behaviour Into Action

Tahap ini adalah proses memecah aktivitas tindakan menjadi lebih spesifik yang dilakukan pada setiap *section* pada suatu halaman. Hal ini bertujuan untuk mengidentifikasi suatu aktivitas tindakan yang diulang dan memiliki pola yang sama sehingga dapat diketahui elemen *user interface* yang bersifat *reuse* (digunakan Kembali).

Gambar 2-20 Proses Breakdown Tindakan [3]

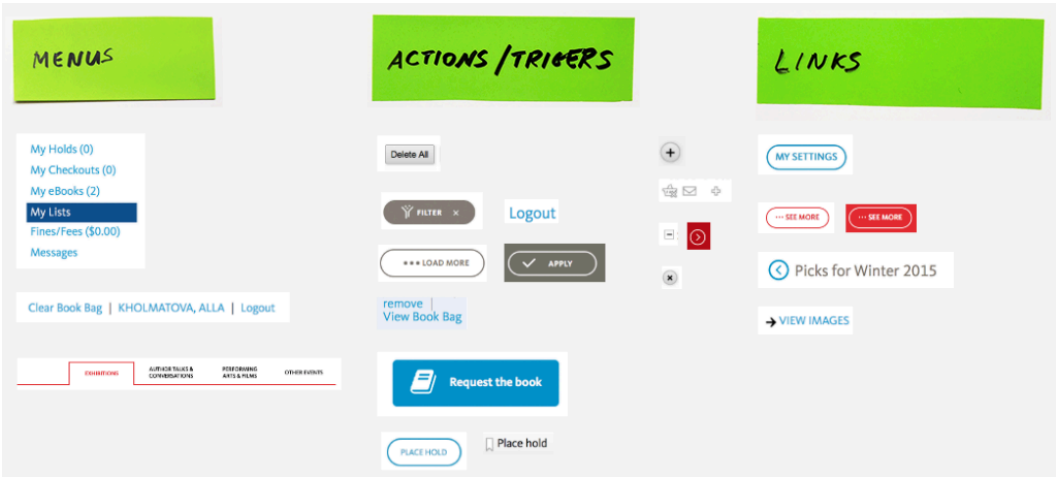
2.10 Functional Pattern

Design system terdiri dari berbagai elemen yang berdiri sendiri atau *independent* namun elemen tersebut saling terikat dan saling mempengaruhi satu sama lain. *Design system* dibagi menjadi dua bagian yaitu bagian pertama adalah *functional pattern* yang mencangkup *UI components, guidelines,* model interaksi, dan elemen *independent* lainnya yang saling terikat. Kemudian bagian kedua yaitu *perceptual pattern* yang mencangkup lapisan *foundation* dengan membuat aturan visual dari elemen dasar seperti *color palette, typography, spacing & layout, iconography* serta ilustrasi.

Functional Pattern merupakan kumpulan elemen komponen *user interface* yang memiliki pola dari tampilan antarmuka. Tujuannya adalah untuk mendorong kebiasaan atau perilaku pengguna pada saat kondisi tertentu [3]. *Functional pattern* harus di definisikan agar *behavior* dan mental model yang dimiliki pengguna terhadap tampilan antarmuka dapat ditemukan. Dalam mendefinisikan *functional pattern* dilakukan beberapa tahapan yaitu sebagai berikut:

1. *Conduct Interface Inventory*

Interface inventory adalah koleksi komprehensif dari potongan *components* yang membentuk tampilan antarmuka [5]. Tahap ini merupakan tahap untuk mengkategorisasikan konten, *components* ataupun elemen visual lainnya. Sehingga dapat diketahui ketidak konsistenan desain yang ada pada sistem saat ini.



Gambar 2-21 Interface Inventory

2. *Define Pattern*

Pada tahap ini menentukan *group elements* dan memutuskan apakah setiap *components* yang ada akan digabungkan menjadi satu pola yang sama atau dipisah. Ada dua tahapan yang dilakukan untuk menentukan pola pada *component* yaitu *specificity scale* dan memetakan *content structure.*

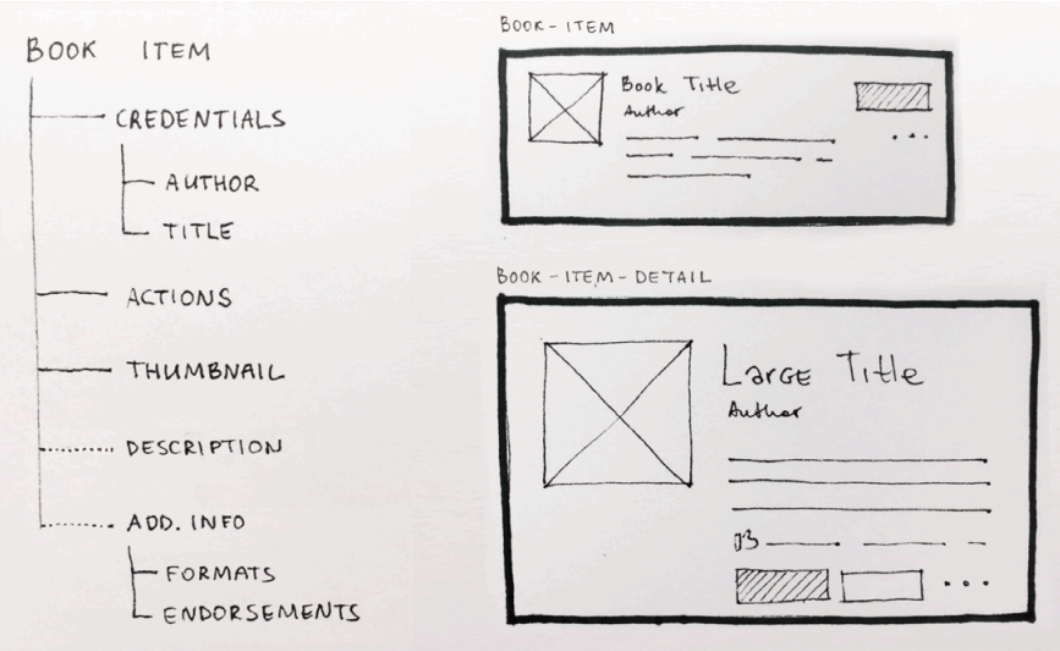
a. *Specificity Scale*

Pada suatu *components* yang memiliki pola sama, ditentukan apakah *components* tersebut adalah *specific* atau *generic.* Hal ini bertujuan untuk menentukan suatu *components* apakah termasuk *reusable components* atau *components* khusus.

Gambar 2-22 Specificity Scale

b. *Content Structure*

Pada tahap ini menentukan hirarki elemen dan menentukan bagaimana pengelompokan suatu modul *components*. Dilakukan sketsa yang menggambarkan struktur hirarki pada modul kemudian diuraikan dan ditentukan setiap elemen yang ada pada *components.* Hal ini bertujuan agar desainer dan *developer* memiliki pemahaman yang sama dalam melihat suatu modul *components.*



Gambar 2-23 Content Structure

c. *Naming*

Penamaan pada suatu modul *components* sangat penting, hal ini dilakukan agar setiap orang dalam tim *inline* atau memiliki pemahaman yang sama dalam melihat suatu *component,* dan hal ini merupakan poin penting dalam *shared language* yang ada pada *design system.*

d. *Visual Hierarchy*

Visual hierarchy adalah suatu elemen yang diurutkan dimana pengguna akan memproses informasi pada suatu halaman [16]. Fungsi *visual hierarchy* pada *user interface* adalah untuk memungkinkan pengguna untuk memahami informasi lebih mudah. Dengan menetapkan karakterisik visual yang berbeda ke bagian informasi [17].

2.11 Perceptual Pattern

Perceptual pattern merupakan kumpulan pola yang berperan sebagai aturan dasar atribut *style* yang dimiliki oleh suatu produk atau sistem [3]. *Perceptual pattern* mencangkup lapisan *foundation* dengan membuat aturan visual dari elemen dasar seperti *color palette, typography, spacing & layout, iconography* serta

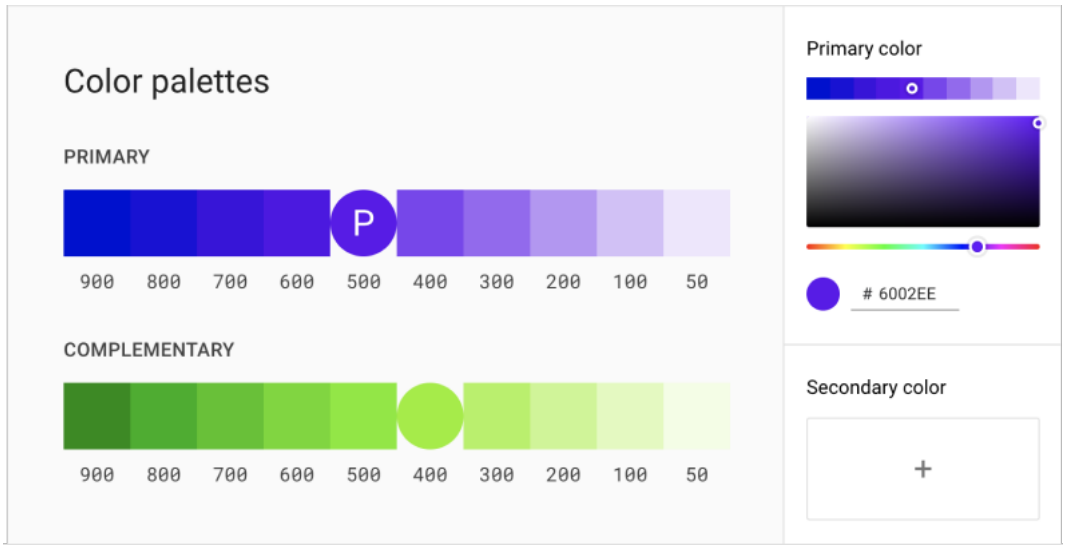
ilustrasi. Didalam *perceptual pattern* juga mencangkup *style guidelines* merupakan kumpulan panduan untuk desainer dalam merancang tampilan *user interface,* yang tersedia dalam bentuk dokumentasi seperti buku atau laporan [18]. Tujuan utama dari *style guidelines* adalah membantu desainer mematuhi "*look and feel*" suatu aturan tertentu sehingga dapat meningkatkan konsistensi di berbagai aplikasi. *Style guidelines* pada *design system* yaitu meliputi *typography, colors* dan *iconography.*

1. Warna

Menggambungkan warna sangat penting untuk diperhatikan seperti bagaimana warna itu bekerja, setiap warna akan cocok satu sama lain, namun terkadang beberapa warna akan saling berbenturan. Ada suatu tools yang dapat digunakan sebagai pedoman dalam menentukan warna yaitu Google Material Color System Generator.

a. Specify Building Block

Pada tahap ini akan dilakukan proses *specify building block* dengan tujuan untuk membuat palet warna lebih fokus, tepat, dan mudah diakses.

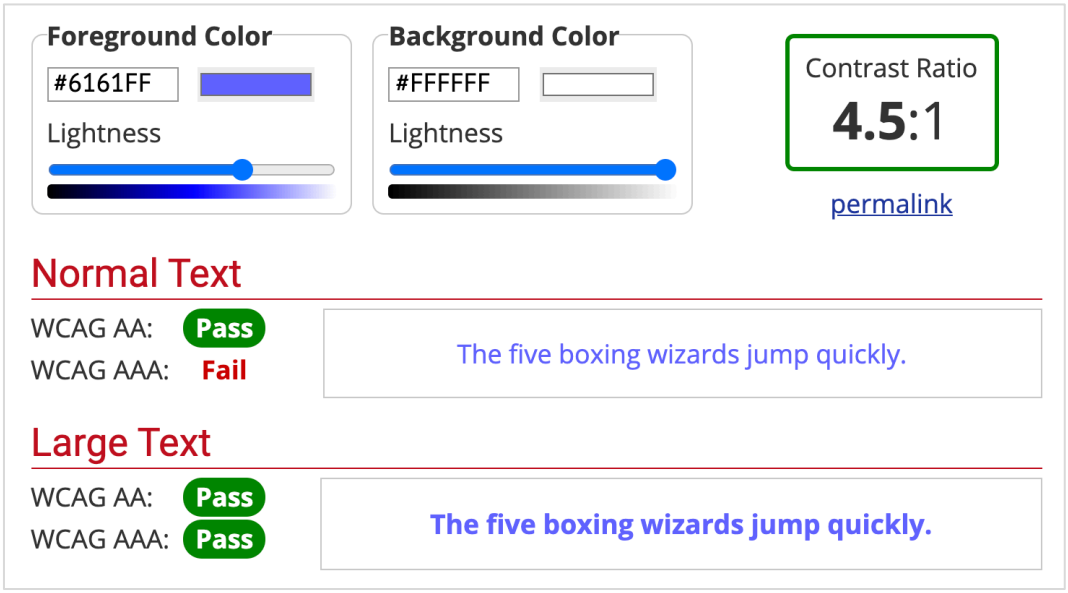


Gambar 2-24 Google Material Color System Generator

b. Kontras Warna

Penggunaan warna pada suatu aplikasi harus memikirkan tentang kontras warna dalam aspek aksesibilitas. Terdapat sebuah panduan dalam mengukur aksesibilitas kontras warna yaitu *Web Content Accesibility Guideline* (WCAG).

WCAG adalah pedoman dalam menetapkan cara membuat konten Web lebih mudah diakses oleh para penyandang disabilitas [19]. Uji kontras warna dapat dilakukan antara kombinasi warna *foreground* dan *background.* WCAG memiliki rekomendasi dalam menentukan kontras (*Minimum*) Level AA yaitu presentasi *foreground* dan *background* memiliki rasio kontras minimal **4,5:1** dan untuk large text *foreground* dan *background* memiliki kontras rasio **3:1**. Terdapat *tools* yang dapat digunakan dalam mengukur kontras warna yaitu WebAIM Contrast Checker.



Gambar 2-25 WebAim Contrast Checker

2. Typography

Typography adalah bagian utama dari suatu *Grapichal User Interface* (GUI). *User interface* desain yang baik tergantung pada bagaimana *type face* bekerja sebagai sistem visual [20]. *Typography* memiliki aturan-aturan agar dapat bekerja dengan baik sebagai kesatuan desain.

Tabel 2-3 Aturan Typography

Aturan	Keterangan
Font Selection	Penggunaan *font* sangat berpengaruh dalam menentukan visual informasi agar tersampaikan dengan baik. Ada beberapa jenis *font face* yang dapat digunakan yaitu *sans serif, serif, monospace* dan *display.*
Font Size & Line Height	*Font size* dan *line height* menentukan keindahan dinamika dan urutan sistem *font* serta hirarki informasi. *Font size* mengacu pada serangkaian *font* dengan ukuran berbeda. Line height mengacu pada rumus *Golden Ratio* yaitu *font size* * 1.618.
Spacing & Kerning	*Spacing & kerning* merupakan jumlah spasi antara dua huruf (atau karakter lain: angka, tanda baca, dan lain-lain) dan proses penyusunan spasi untuk menghindari gap yang terlihat kurang pas diantara huruf-huruf yang dibuat dengan tujuan untuk meningkatkan kemudahan membaca (*readability*).

2.12 Usability Metrics

Usability metrics adalah aktivitas pengujian untuk mengukur kinerja pengguna saat menyelesaikan serangkaian tugas yang diberikan [21]. Ukuran paling dasar dalam melakukan usability metrics adalah *success rate,* waktu yang dibutuhkan suatu tugas dan *error rate* [22]. Adapun aktivitas yang dilakukan adalah peneliti memberi partisipan serangkaian tugas yang harus mereka lakukan dengan menggunakan desain yang diuji. Selama pengujian berlangsung, peneliti mengamati setiap tindakan partisipan sembari dilakukan perekaman video. Setelah menganalisis hasil pengujian, peneliti mencatat beberapa poin penting seperti

efektivitas desain, efisiensi desain, kepuasan pengguna, serta isu-isu yang muncul seperti aspek desain yang menyebabkan masalah [23]. Berikut jenis-jenis usability testing berdasarkan tujuannya [19]:

1. *Quantitative Usability Metric*

Dalam *quantitative usability metric*, dilakukan pembuktian apabila *design system* dapat digunakan dengan baik. Tes ini dilakukan dengan cara mengukur dan menggunakan *success rate* tertentu.

Aktivitas yang dilakukan di antaranya adalah menghitung berapa orang yang berhasil melakukan tugasnya dan juga waktu yang dihabiskan dalam menyelesaikan tugas tersebut. Beberapa hal yang dapat diukur dan dihitung di antaranya adalah sebagai berikut:

a. Efektivitas

Efektivitas yang dimaksud merupakan nilai yang didapat berdasarkan tingkat keberhasilan desainer dalam menyelesaikan tugas tertentu. Efektivitas diukur dengan cara membandingkan jumlah partisipan yang berhasil menyelesaikan tugas dengan jumlah keseluruhan partisipan.

$$Effectiveness = \frac{\text{Number of tasks completed successfully}}{\text{Total number of tasks undertaken}} \times 100\%$$

Gambar 2-26 Rumus Pengukuran Efektivitas

b. Efisiensi Relatif Secara Keseluruhan

Masing-masing tugas dapat diukur nilai efisiensi relatif keseluruhannya. Nilai ini merupakan nilai yang mengindikasikan seberapa cepat desainer dapat menyelesaikan suatu tugas. Pengukuran nilai efisiensi relatif keseluruhan dapat dilakukan dengan membandingkan waktu penyelesaian tugas oleh partisipan yang berhasil dengan total waktu yang dihabiskan oleh seluruh partisipan.

$$Overall\ Relative\ Efficiency = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N} n_{ij}t_{ij}}{\sum_{j=1}^{R}\sum_{i=1}^{N} t_{ij}} \times 100\%$$

Gambar 2-27 Rumus Pengukuran Efisiensi

c. Kepuasan Pengguna

Kepuasan pengguna meliputi kenyamanan dan penerimaan kemudahan penggunaan. Setelah pengguna mencoba tugas (terlepas dari apakah mereka berhasil mencapai tujuannya atau tidak), mereka harus segera diberikan kuesioner sehingga dapat mengukur seberapa sulit tugas itu. Kuesioner *post-test* ini biasanya berupa skala Likert dengan tujuan untuk mendapatkan wawasan mengenai seberapa mudah suatu tugas diselesaikan berdasarkan perspektif desainer. Poin 1 diberikan apabila desainer mengisi dengan "Sulit", poin 2 diberikan apabila desainer mengisi dengan "Sulit", dan seterusnya hingga poin 5 diberikan apabila desainer mengisi dengan "Sangat Mudah".

2. Qualitative Usability Metric

Qualitative usability metric berguna untuk mendapatkan wawasan yang dapat membantu desainer memperbaiki *design sstem* yang dirancang. Berbeda dengan jenis *quantitative*, dalam jenis ini, interaksi peneliti dengan partisipan merupakan hal yang perlu dilakukan untuk mendapatkan wawasan lebih [23]. Adapun tahapan yang harus dilakukan dalam usability testing adalah sebagai berikut:

a. Menentukan Tujuan Pengujian

Tujuan pengujian ini ditentukan untuk mengukur keberhasilan dari *case study* yang diuji. Tujuan ini meliputi nilai-nilai performansi yang diharapkan, seperti efektivitas, efisiensi, kepuasan pengguna, dan sebagainya. Penelitian ini dikatakan berhasil apabila tujuannya tercapai, yaitu hasil yang didapat dari pengujian sesuai dengan apa yang telah ditentukan.

b. Membuat Daftar Tugas Partisipan dan Skenario Pengujian

Pembuatan daftar tugas ini bertujuan untuk menentukan apa saja yang perlu dilakukan partisipan dalam menyelesaikan tugas yang diberikan. Setiap tugas yang telah didaftarkan kemudian dibentuk skenarionya. Skenario yang dibuat mengandung beberapa konteks seperti peran partisipan dan apa yang mereka harus lakukan tanpa memberikan petunjuk.

c. Membuat Naskah Pengujian

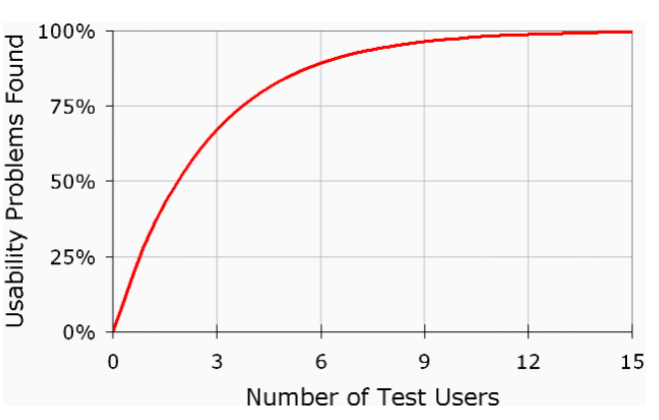
Ketika dilakukan pengujian, banyak hal-hal rinci yang harus tetap teramati selama pengujian berlangsung. Hal-hal rinci tersebut mungkin saja bisa diingat dengan baik oleh peneliti, namun mungkin juga terlupakan. Itulah mengapa naskah memiliki peran penting dalam pelaksanaan pengujian usability testing. Dalam naskah tersebut, dituliskan seluruh hal yang perlu dikatakan, dan juga hal yang perlu dilakukan oleh peneliti. Selain itu, dalam naskah tersebut termasuk juga daftar tugas, dan juga skenario pengujian.

d. Melakukan Pengujian dan Mencatat Hasil Pengujian

Pada tahap ini, dilakukan pengujian dengan mengikuti alur yang telah dituliskan dalam naskah. Peneliti juga harus mencatat setiap hal yang terjadi ketika pengujian berlangsung, seperti bagaimana cara pengguna berinteraksi dengan produk dan apa yang mereka katakan.

e. Melakukan evaluasi hasil pengujian

Pada tahap ini, dilakukan evaluasi untuk mendapatkan pengetahuan dari hasil pengujian yang didapat. Berdasarkan penelitian yang telah dilakukan oleh Jakob Nielsen dan Tom Landauer, didapatkan pengetahuan bahwa hasil terbaik didapat dari pengujian yang dilakukan dengan lima partisipan. Setelah sesi pengujian dengan partisipan kelima, mereka tidak ditemukan hal lain yang berbeda dari temuan yang ditemukan dalam sesi-sesi sebelumnya. Gambaran hasil penelitian tersebut dapat di lihat pada **Gambar 2-28** [24].



Gambar 2-28 Gambaran Hasil Penelitian Mengenai Jumlah Partisipan *Usability Testing*