BAB 2
TINJAUAN REFERENSI

2.1 Software Development Life Cycle (SDLC)

Menurut Alamn Dennis and Barbara Wixom (Alan Dennis, Barbara Wixom, 2015) Analisa dan perancangan, terdapat beberapa fase yang penting dan wajib dilewati yakni sebagai berikut:

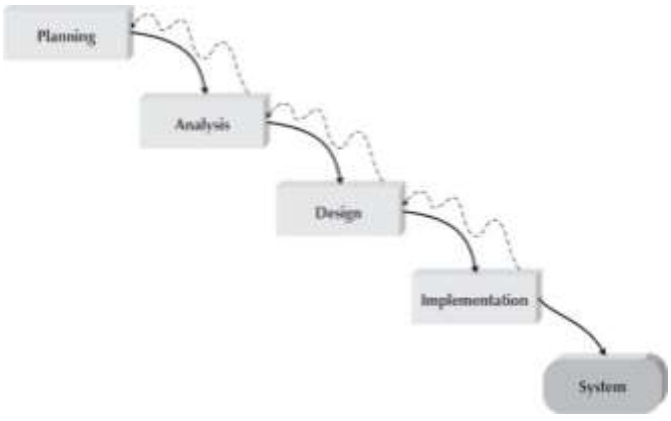
1) ***Planning***, tahap perencanaan sistem biasanya dimulai dengan ***system request***, yang menjelaskan masalah atau perubahan yang diinginkan. *Major request* dapat melibatkan sistem informasi baru atau peningkatan sistem yang ada. *Minor request* dapat meminta fitur baru atau perubahan ke antarmuka pengguna. Tujuan dari fase ini adalah untuk melakukan ***preliminary investigation*** untuk mengevaluasi suatu masalah. Bagian penting dari *preliminary investigation* adalah ***feasibility study*** yang meninjau biaya dan manfaat yang diantisipasi dan merekomendasikan tindakan berdasarkan faktor-faktor *operational, technical, economic, dan time.*
2) ***Analysis***, tujuan dari fase analisis sistem adalah untuk membangun model logis dari sistem baru. Langkah pertama adalah ***requirements modeling***, di mana analis menyelidiki proses bisnis dan mendokumentasikan apa yang harus dilakukan sistem baru untuk memuaskan pengguna. *Requirements modeling* melanjutkan penyelidikan yang dimulai selama fase *planning*. Untuk memahami sistem, pencarian fakta menggunakan teknik seperti wawancara, survei, tinjauan dokumen, observasi, dan pengambilan sampel dilakukan. Hasil pencarian fakta digunakan untuk membangun model bisnis, data dan model proses, dan model objek. Hasil untuk tahap analisis sistem adalah ***system requirements document***. *System requirements document* menjelaskan persyaratan manajemen dan pengguna, biaya dan manfaat, dan menguraikan strategi pengembangan alternatif.
3) ***Design***, tujuan dari fase desain sistem adalah untuk membuat model fisik yang akan memenuhi semua *system requirements document*. Pada tahap ini, antarmuka pengguna dirancang dan output, input, dan proses yang diperlukan diidentifikasi. Selain itu, kontrol internal dan eksternal dirancang, termasuk fitur berbasis komputer dan manual untuk menjamin bahwa sistem akan *reliable, accurate, maintainable,* dan *secure*. Selama fase desain sistem, arsitektur aplikasi juga

4) ditentukan, dimana akan digunakan programmer untuk mengubah desain logis menjadi modul dan kode program.Hasil untuk fase ini adalah *system design specification*, yang disajikan kepada manajemen dan pengguna untuk ditinjau dan disetujui. Keterlibatan manajemen dan pengguna sangat penting untuk menghindari kesalahpahaman tentang apa yang akan dilakukan sistem baru, bagaimana akan melakukannya, dan berapa biayanya.
5) ***Implementation***, selama fase implementasi sistem yang baru dibangun. Program akan ditulis, diuji, dan didokumentasikan. Tujuan dari fase implementasi sistem adalah untuk memberikan sistem informasi yang berfungsi dan terdokumentasi sepenuhnya. Dan juga melatih pengguna, dalam melakukan transisi ke sistem baru. Implementasi sistem mencakup penilaian, yang disebut evaluasi sistem, untuk menentukan apakah sistem beroperasi dengan benar dan apakah biaya dan manfaatnya sesuai harapan.

2.1.1 Metode Waterfall

Menurut (Alan Dennis, Barbara Wixom, 2015), Dengan metodologi pengembangan berbasis *waterfall*, analis dan pengguna melanjutkan secara berurutan dari satu fase ke fase berikutnya. Hasil utama untuk setiap fase biasanya sangat panjang dan disajikan kepada sponsor proyek untuk disetujui saat proyek bergerak dari fase ke fase. Setelah sponsor menyetujui pekerjaan yang dilakukan untuk suatu fase, fase berakhir dan yang berikutnya dimulai.

Beliau juga mengatakan terdapat dua keuntungan utama dari pendekatan desain terstruktur *waterfall* adalah bahwa ia mengidentifikasi *system requirements* jauh sebelum pemrograman dimulai dan meminimalkan perubahan pada persyaratan saat proyek dimulai. (Alan Dennis, Barbara Wixom, 2015)



Gambar 1.2.1 Siklus Metode Waterfall

2.2 Sistem

Sistem adalah seperangkat prosedur bisnis (atau komponen) yang saling terkait yang digunakan dalam satu unit bisnis, bekerja secara Bersama-sama untuk mencapai beberapa tujuan. Misalnya, sistem ketenagakerjaan yang bertujuan untuk melacak pembayaran gaji karyawan, sedangkan sistem inventaris untuk melacak alur persediaan(Joseph, S. V., Joey, F. G., & Jeffrey, 2015).

Sistem pada umumnya memiliki sembilan karakteristik seperti yang ditunjukkan di bawah ini yakni:

a. Komponen yang merupakan bagian yang tidak dapat direduksi atau bagian tambahan yang juga disebut subsistem. Sebagai contoh, hubungan antara mobil atau sistem stereo dimana dengan desain yang tepat, kita dapat memperbaiki atau meningkatkan sistem dengan mengubah komponen individual tanpa harus membuat perubahan di seluruh sistem.
b. Komponen pada dasarnya saling terkait dimana fungsi yang satu terkait dengan fungsi yang lain. Misalnya, pekerjaan satu komponen, seperti menghasilkan laporan harian pesanan pelanggan yang diterima, mungkin tidak berhasil berkembang sampai pekerjaan komponen lain selesai, seperti mengurutkan pesanan pelanggan berdasarkan tanggal penerimaan.
c. Suatu sistem memiliki batas, di mana semua komponennya terkandung dan yang menetapkan batas-batas suatu sistem, memisahkannya dari sistem lain. Komponen yang berada dalam batas sistem dapat diubah, sedangkan sistem di luar batas sistem yang dimaksud tidak dapat diubah.
d. Semua komponen bekerja sama untuk mencapai beberapa tujuan keseluruhan untuk sistem yang lebih besar.
e. Suatu sistem memiliki lingkungan tersendiri dimana termasuk segala sesuatu di luar batas sistem yang memengaruhi sistem. Misalnya, lingkungan universitas negeri terdiri dari calon mahasiswa, yayasan dan lembaga pendanaan, serta media berita.
f. Titik-titik di mana sistem memenuhi lingkungannya yang disebut interfaces; sebuah interfaces juga terjadi antar subsistem.
g. Dalam hal fungsionalitas, suatu sistem harus memiliki batasan (dalam hal kapasitas, kecepatan, atau kemampuan) terkait hal yang dapat dilakukan dan bagaimana sistem tersebut dapat mencapai tujuan di dalam lingkungannya.

h. Beberapa kendala ini terdapat di dalam sistem (Jumlah staf yang terbatas tersedia), dan kendala yang terdapat diluar lingkungan sistem(Tenggat waktu atau peraturan).
i. Sistem mengambil input dari lingkungannya agar dapat menjalankan fungsinya. Manusia, misalnya, mengambil makanan, oksigen, dan air dari lingkungan sebagai masukan. Seseorang dibatasi untuk menghirupudara segar jika seseorang sedang berada di lift bersama seseorang yang merokok.Suatu sistem mengembalikan output ke lingkungannya sebagai hasil dari fungsi sistem tersebut untuk menandakan bahwa tujuan telah terpenuhi (Joseph, S. V., Joey, F. G., & Jeffrey, 2015).

2.3 Analisa dan Perancangan Sistem Informasi

Analisis dan percancangan sistem informasi adalah metode yang digunakan oleh suatu perusahaan, mulai dari perusahaan IBM hingga PepsiCo dan juga Sony untuk membuat dan memelihara sistem informasi yang melakukan fungsi bisnis dasar seperti melacak nama dan alamat pelanggan, memproses pesanan, hingga membayar gaji karyawan. Tujuan utama dari analisis dan perancangan sistem adalah untuk meningkatkan kualitas dan efisiensi dari sistem organisasi melakukan implementasi perangkat lunak yang dapat membantu karyawan menyelesaikan tugas-tugas utama perusahaan dengan lebih mudah dan efisien (Joseph, S. V., Joey, F. G., & Jeffrey, 2015)

2.4 Penelitian Terdahulu

Jurnal Rancang Bangun Sistem Informasi Kolaboratif Berbasis Web Untuk Manajemen Proyek Teknologi Informasi (Paramita, 2015) menyediakan sistem informasi berbasis website yang dapat menyesuaikan penjadwalan biaya yang setara dengan anggaran dan sistem yang sesuai kebutuhan pengguna agar sistem yang dibangun dapat menekankan kerjasama pengguna dan pertukaran informasi secara online. Dilakukan dengan 4 tahapan metode yaitu analisis, perancangan sistem, implementasi dan pengujian. Tujuan dibangunnya sistem ini adalah untuk membantu pengelolaan data dan membantu memberikan penawaran proyek kepada konsultan TI dengan dukungan fitur-fitur seperti Pengelolaan jabatan, proyek, task, milestone, komentar, topic diskusi, file sharing, dan laporan berupa gantt chart.

Dalam aplikasi task management yang dirancang terdapat 4 modul utama (Paradkar et

al., 2017):

1. Admin Module

   Admin adalah Pemilik utama aplikasi. Admin memiliki wewenang untuk mengelola Project Manager dan kebebasan untuk melakukan apa pun. Admin membuat tugas baru; setiap tugas yang baru dibuat ditempatkan dalam grup tertentu dan ditandai dengan status terbuka. Menggunakan Manage User Assignment, Admin akan membuat tim dari individu orang akan diberikan tugas yang akan dilakukan.

2. Project Manager

   Modul ini menyimpan informasi tentang informasi tentang karyawan, Ketua Tim, Manajer Proyek yang bekerja untuk organisasi. Modul ini memelihara komunikasi dengan task management module, subtask Management module, dan admin module.

3. Task Management Module

   Modul ini umumnya menyimpan informasi tentang semua tugas yang diatur pada tingkat standar organisasi. Setiap tugas di bawah sistem terdaftar dengan modul ini.

4. Subtask Management Module

   Modul ini menyimpan informasi mengenai Subtasks yang dijadwalkan pada karyawan. Modul ini mencatat tanggal dan waktu kapan Subtask tertentu akan dijadwalkan dan kapan tugas itu akan selesai. Modul ini merupakan bagian dari Task Management Module.

Menurut (Hedaoo et al., 2018), Online Task Management System digunakan untuk mengotomatiskan proses admin dan manajemen pengguna dan tugas pengguna. Pekerjaan ini di bawah pengawasan otoritas yang lebih tinggi. Proyek ini menyediakan platform online untuk menyelesaikan tugas sehari-hari di suatu departemen. Admin memiliki wewenang untuk mengendalikan seluruh sistem. Pengguna tidak memiliki otentikasi untuk melihat tugas admin. Tujuan untuk mengembangkan proyek ini adalah untuk menghasilkan platform di mana admin dapat melakukan semua jenis operasi seperti view, insert, edit, events, delete, dan tracking operations. Pengguna

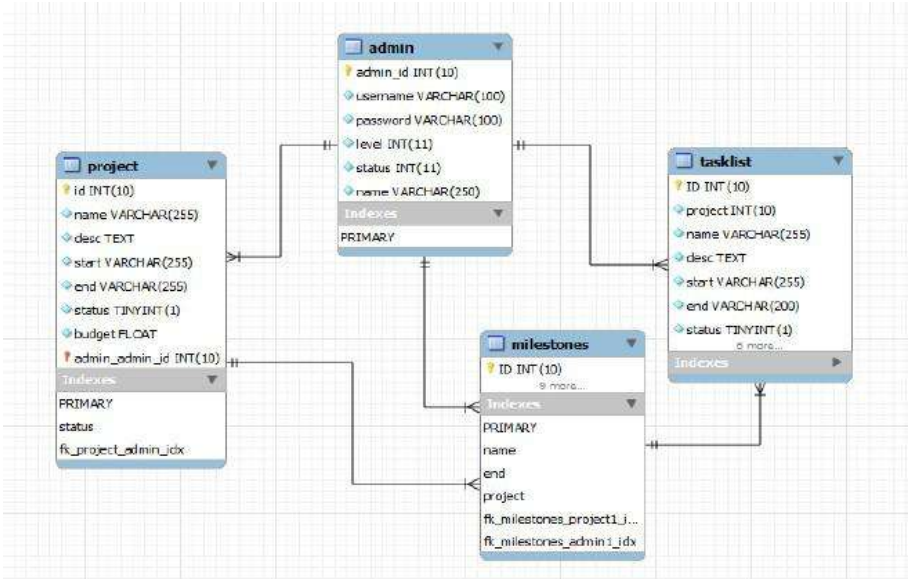
admin adalah pengguna istimewa yang dapat menambah dan mengubah tugas.

Menurut (Hedaoo et al., 2018), terdapat beberapa keuntungan menggunakan Online Task Management System:

1. Effective Task Management
   Sistem digunakan untuk memberi tahu pengguna, pembagian tugas yang tepat, pengawasan otoritas yang lebih tinggi, dan seterusnya untuk menjadikannya aplikasi manajemen tugas yang efektif dari aplikasi web.
2. Flexibility and scalability
   Sistem dapat diakses di mana saja. Sistem menyediakan segala yang dibutuhkan.
3. Effective time management
   Sistem dapat melacak pekerjaan sehubungan dengan waktu.
4. Uninterrupted communication
   Jaringan aman dan menggunakan sejumlah data ringan untuk ditransfer sehingga tidak terganggu karena jaringan lambat.
5. Notifications and reminders Memberikan
   notifikasi kepada pengguna.

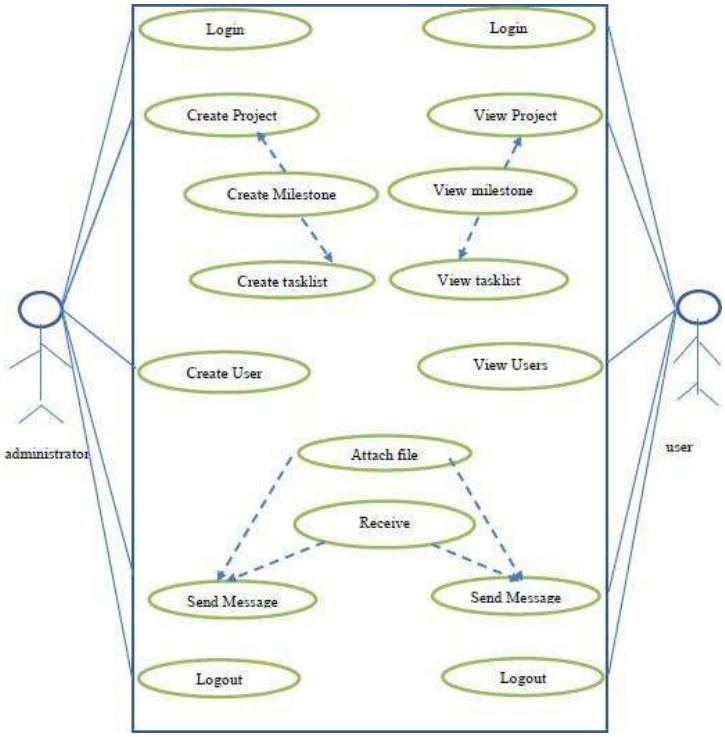
Menurut (Izang et al., 2016), project management system adalah perangkat lunak yang memiliki kemampuan untuk membantu menyusun strategi, mengatur, dan mengelola aliran sumber daya dan mengembangkan perkiraan sumber daya.
(Izang et al., 2016) mengembangkan web-based project management system diimplementasikan sebagai aplikasi web untuk diakses melalui browser web. Sistem ini multi-pengguna, dan dapat diakses dari komputer manapun tanpa menginstal perangkat lunak. Fungsi-fungsi utama yang terdapat dalam aplikasi tersebut adalah:

- Project
- Tasklist
- Milestone
- File
- User



Gambar 2.4.1 Entity Relationship Diagram Web-Based Project Management System

(Sumber: (Izang et al., 2016))



Gambar 3.4.2 Use Case Diagram Web-Based Project Management System

(Sumber: (Izang et al., 2016))

Dengan sistem ini, semua anggota yang telah ditambahkan ke proyek baru dapat saling mengirim pesan, dan mengawasi kemajuan proyek. Direkomendasikan bahwa *web*

based project management systems ini harus diterapkan dimanapun kebutuhan untuk mengelola proyek secara efisien muncul. Mereka nyaman digunakan, menghemat waktu dan sumber daya. Untuk studi lebih lanjut, sistem ini dapat dimodifikasi oleh peneliti lain untuk diadaptasi dalam berbagai bidang studi (Izang et al., 2016).

2.5 Service Management Operation (SMO)

Menurut (Lucio-Nieto et al., 2016), SMO adalah suatu bagian dari mekanisme yang bertanggung jawab untuk memberikan layanan IT yang berkualitas kepada pengguna baik di tingkat taktis maupun strategis, yang tanggung jawabnya mencakup strategi dan desain layanan yang memberikan nilai bisnis dan proses tata kelola, kerangka kerja, metodologi, dan standar TI. Meskipun tidak ada definisi standar untuk Kantor Manajemen Layanan (SMO), fungsi dan ruang lingkupnya, dengan menganalisis berbagai definisi, dapat dinyatakan bahwa ia mewakili mekanisme tata kelola TI yang mendefinisikan, memantau, dan mengaudit proses yang sedang beroperasi atau dalam transisi, dan bertanggung jawab untuk menjamin kepatuhan dengan strategi layanan "ujung-ke-ujung" di antara semua fungsi yang terlibat dalam definisi, desain, transisi, operasi, dan peningkatan berkelanjutan, dengan tiga tanggung jawab utama: strategi dan desain layanan yang memberikan nilai bisnis (permintaan, portofolio dan manajemen katalog layanan), dan bertanggung jawab atas hubungan dengan bisnis dan bertanggung jawab atas tata kelola proses, kerangka kerja, metodologi, dan standar TI.

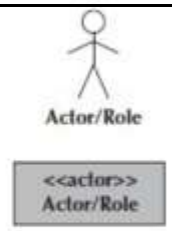
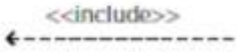
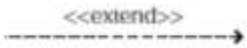
2.6 Unified Modeling Language (UML)

Tujuan UML adalah untuk menyediakan kosakata umum berorientasi objek dan teknik diagram yang cukup kaya untuk memodelkan setiap proyek pengembangan sistem mulai dari analisis hingga implementasi (Alan Dennis, Barbara Wixom, 2015).

2.6.1 Use-Case Diagram

Menurut (Alan Dennis, Barbara Wixom, 2015),Use-case diagram menyediakan cara sederhana, langsung berkomunikasi kepada pengguna apa yang akan dilakukan sistem, use-case diagram diambil ketika mengumpulkan dan mendefinisikan persyaratan untuk sistem.

Tabel 1 Syntax Use Case Diagram
(Sumber:(Alan Dennis, Barbara Wixom, 2015), p.122

Deskripsi	Simbol
Actor: • Adalah seseorang yang memperoleh manfaat dari dan bersifat eksternal terhadap subjek.	
Use case:	
• Merupakan bagian utama dari fungsionalitas sistem. • Dapat *extend* use case lain. • Dapat *include* use case lain.	
Subject boundary: • Memiliki nama subjek di dalam atau di atas. • Merupakan ruang lingkup subjek.	
Association relationship: • Menghubungkan aktor dengan use case yang berinteraksi dengannya.	
Include relationship: • Merupakan penyertaan fungsi dari satu use case di dalam yang lain.	
Extend relationship: • Merupakan ekstensi dari use case untuk memasukkan perilaku opsional.	
Generalization relationship: • Menggambarkan *use case* khusus ke yang lebih umum.	

2.6.2 Use-Case Description

Menurut (Alan Dennis, Barbara Wixom, 2015), use-case descriptions menyediakan

sarana untuk mendokumentasikan lebih lengkap aspek-aspek yang berbeda dari masing-masing kasus use case. Use-case descriptions didasarkan pada persyaratan yang diidentifikasi, use-case diagram, dan deskripsi activity diagram dari proses bisnis. Use-case descriptions berisi semua informasi yang diperlukan untuk mendokumentasikan fungsionalitas proses bisnis.

Use Case Name: Make Old Patient Appt	ID: 2	Importance Level: Low
Primary Actor: Old Patient	Use Case Type: Detail, Essential	
Stakeholders and Interests: Old Patient – wants to make, change, or cancel an appointment Doctor – wants to ensure patient's needs are met in a timely manner		
Brief Description: This use case describes how we make an appointment as well as changing or canceling an appointment for a previously seen patient.		
Trigger: Patient calls and asks for a new appointment or asks to cancel or change an existing appointment Type: External		
Relationships: Association: Old Patient Include: Extend: Update Patient Information Generalization: Manage Appointments		

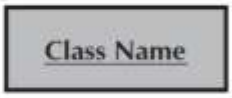
Gambar 4.2.6.2 Sample Use-case description
(Sumber:(Alan Dennis, Barbara Wixom, 2015), p.143)

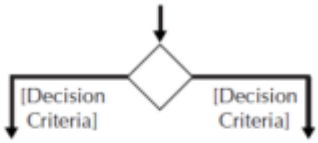
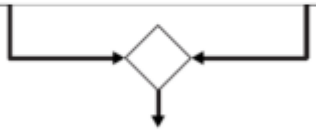
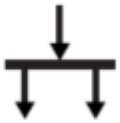
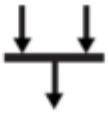
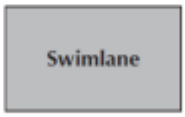
2.6.3 Activity Diagram

Menurut (Alan Dennis, Barbara Wixom, 2015), activity diagram digunakan untuk memodelkan perilaku dalam proses bisnis yang tidak bergantung pada objek. Activity diagram dapat digunakan untuk memodelkan segala sesuatu mulai dari alur kerja bisnis tingkat tinggi yang melibatkan banyak kasus penggunaan yang berbeda, hingga detail kasus penggunaan individual, sampai ke detail spesifik dari metode individual. Singkatnya, diagram aktivitas dapat digunakan untuk memodelkan segala jenis proses.

Tabel 2.2.6.3 Syntax Activity Diagram
(Sumber:(Alan Dennis, Barbara Wixom, 2015), p.132)

Deskripsi	**Simbol**
Action: • Adalah perilaku yang sederhana, tidak dapat diurai.	
Activity : • Digunakan untuk mewakili serangkaian *actions*	

.	
Object node: • Digunakan untuk mewakili objek yang terhubung ke serangkaian aliran objek.	
Control flow: • Memperlihatkan urutan eksekusi.	
Object flow: • Memperlihatkan aliran suatu objek dari satu *activity* (atau *action*) ke *activity* lain (atau *action*).	
Initial node: • Menggambarkan awal dari serangkaian *action* atau *activity*.	
Final-activity node: • Digunakan untuk menghentikan semua aliran kontrol dan objek mengalir dalam suatu *activity* (atau *action*)	
Final-flow node: • Digunakan untuk menghentikan aliran kontrol tertentu atau aliran objek.	

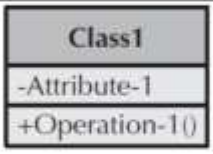
Decision node: • Digunakan untuk mewakili kondisi pengujian untuk memastikan bahwa aliran kontrol atau aliran objek hanya turun satu jalur.	
Merge node:	
• Digunakan untuk menyatukan kembali jalur keputusan berbeda yang dibuat menggunakan simpul keputusan.	
Fork node: • Digunakan untuk membagi perilaku menjadi serangkaian *activity* paralel atau bersamaan (atau *action*).	
Join node: • Digunakan untuk menyatukan kembali serangkaian *activity* (atau *action*) yang paralel atau bersamaan.	
Swimlane: • Digunakan untuk memecah *activity diagram* menjadi baris dan kolom untuk menetapkan *activity* atau *action* individu ke individu atau objek yang bertanggung jawab untuk menjalankan *activity* (atau *action*).	

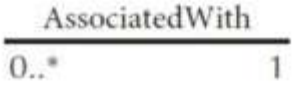
2.6.4 Class Diagram

Menurut (Alan Dennis, Barbara Wixom, 2015), *class diagram* adalah model statis yang

menunjukkan kelas dan hubungan antara kelas yang tetap konstan dalam sistem dari waktu ke waktu. Diagram kelas menggambarkan kelas, yang mencakup perilaku dan status, dengan hubungan antara kelas.

Tabel 3.2.6.4 Syntax Class Diagram
(Sumber:(Alan Dennis, Barbara Wixom, 2015), p.178)

Deskripsi	**Simbol**
Class: • Merupakan jenis orang, tempat, atau hal yang perlu ditangkap dan disimpan oleh sistem.	
• Memiliki nama yang diketik tebal dan di tengah di kompartemen atas. • Memiliki daftar atribut di kompartemen tengah. • Memiliki daftar operasi di kompartemen bawah.	
Attribute: • Merupakan properti yang menggambarkan keadaan suatu objek. • Dapat diturunkan dari atribut lain, ditunjukkan dengan menempatkan garis miring di depan nama atribut.	attribute name /derived attribute name

Operation: • Merupakan tindakan atau fungsi yang dapat dilakukan oleh suatu kelas. • Dapat diklasifikasikan sebagai konstruktor, kueri, atau operasi *update*.	
Association: • Merupakan hubungan antara beberapa kelas atau suatu kelas dan kelas itu sendiri. • Memiliki simbol multiplisitas, yang mewakili jumlah minimum dan maksimum *instance* kelas dapat dikaitkan dengan *instance* kelas terkait.	
Generalization: • Merupakan *a-kind-of relationship* antara beberapa kelas.	
Aggregation:	
• Merupakan *logical a-part-of relationship* antara beberapa kelas atau suatu kelas dan kelas itu sendiri.	
Composition: • Merupakan *physical a-part-of relationship* antara beberapa kelas atau suatu kelas dan kelas itu sendiri.	

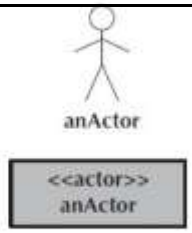
2.6.5 Sequence Diagram

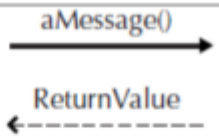
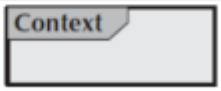
Menurut (Alan Dennis, Barbara Wixom, 2015), *sequence diagram* menggambarkan

objek yang berpartisipasi dalam *use case* dan pesan yang melewati antara mereka dari waktu ke waktu untuk satu *use case*. *Sequence diagram* adalah model dinamis yang menunjukkan urutan eksplisit pesan yang dikirimkan antara objek dalam interaksi yang ditentukan. Karena *sequence diagram* menekankan urutan waktu berdasarkan aktivitas yang terjadi di antara sekumpulan objek, mereka sangat membantu untuk memahami spesifikasi *real- time* dan *use cases* yang kompleks.

Tabel 4.2.6.5 Syntax Sequence Diagram

(Sumber:(Alan Dennis, Barbara Wixom, 2015), p.206)

Deskripsi	**Simbol**
Actor: • Adalah seseorang atau sistem yang mendapat manfaat dan bersifat eksternal dari sistem. • Berpartisipasi dalam urutan dengan mengirim dan / atau menerima pesan. • Ditempatkan di bagian atas diagram.	
Object: • Berpartisipasi dalam urutan dengan mengirim dan / atau menerima pesan. • Ditempatkan di bagian atas diagram.	
Lifeline: • Menunjukkan kehidupan suatu objek selama suatu urutan. • Berisi X pada titik di mana kelas tidak lagi berinteraksi.	

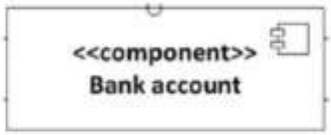
Execution occurrence: • Persegi panjang sempit yang ditempatkan di atas garis hidup. • Menunjukkan ketika suatu objek mengirim atau menerima pesan.	
Message: • Menyampaikan informasi dari satu objek ke objek lainnya. • Operation call ditandai dengan pesan yang dikirim dan panah padat, sedangkan pengembalian ditandai dengan nilai yang dikembalikan dan ditampilkan sebagai panah putus-putus.	
Guard condition: • Merupakan tes yang harus dipenuhi untuk pesan yang akan dikirim.	
Object destruction:	X
• X ditempatkan di akhir garis hidup objek untuk menunjukkan bahwa objek akan dihapus.	
Frame: • Menunjukkan konteks diagram urutan.	

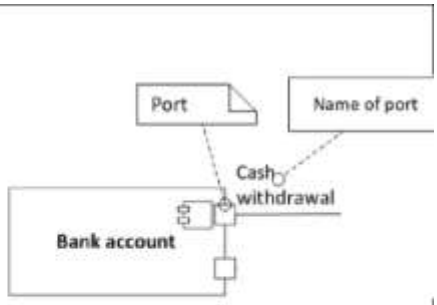
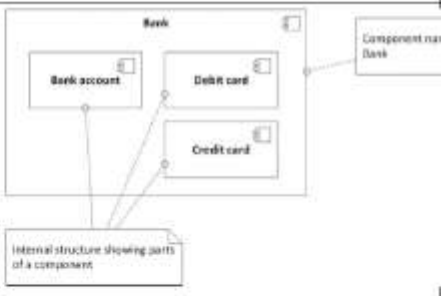
2.6.6 Component Diagram

Menurut (Osis & Donins, 2017), Diagram komponen menunjukkan bagian internal, konektor, dan port yang mengimplementasikan komponen. Ketika komponen dipakai, salinan bagian-bagian dalamny a juga dipakai. Diagram Komponen UML menunjukkan bagaimana sistem perangkat lunak akan terdiri dari sekumpulan komponen yang dapat digunakan - file dan pustaka tautan dinamis, file yang dapat dieksekusi, atau layanan web - yang berinteraksi melalui antarmuka yang terdefinisi

dengan baik dan yang menyembunyikan detail internalnya.

Tabel 5.2.6.6 Syntax Component Diagram
(Sumber:(Osis & Donins, 2017), p.16-18)

Deskripsi	**Contoh Simbol**
Interface Menetapkan kontrak yang terdiri dari sekumpulan atribut publik yang koheren dan operasi untuk suatu kelas. Setiap instance dari kelas yang menyadari antarmuka harus memenuhi kontrak itu. Karena antarmuka adalah deklarasi, mereka tidak instantiable. Sebagai gantinya, spesifikasi antarmuka diimplementasikan oleh instance kelas. Setiap kelas dapat mengimplementasikan lebih dari satu antarmuka dan setiap antarmuka dapat diimplementasikan oleh sejumlah kelas yang berbeda.	
Component Mewakili bagian modular dari sistem yang merangkum isinya, ia mendefinisikan perilakunya dalam hal antarmuka yang disediakan dan diperlukan. Dengan demikian, komponen berfungsi sebagai tipe yang kesesuaiannya ditentukan oleh antarmuka yang disediakan dan diperlukan ini (mencakup semantik statis maupun dinamisnya). Oleh karena itu satu komponen dapat diganti oleh yang lain hanya jika keduanya sesuai jenis.	

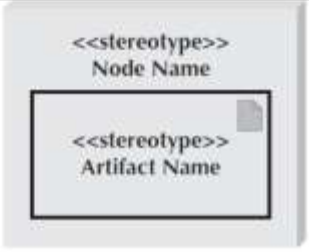
Port Jendela eksplisit menjadi komponen yang dienkapsulasi. Semua interaksi masuk dan keluar dari komponen tersebut melewati port. Setiap port menyediakan atau memerlukan satu atau beberapa antarmuka spesifik.	
Part Komponen yang membangun internal structure yang lebih kompleks komponen. Part dapat dianggap sebagai subcomponent.	
Connector Hubungan antar port komponen. Jika satu port menyediakan antarmuka yang dibutuhkan oleh port lainnya, mereka dapat dihubungkan bersama.	 *Direct connector*
Direct connector menghubungkan dua port bagian. *Delegation connector* yang menghubungkan port bagian dan port komponen sehingga menyediakan antarmuka atau antarmuka yang membutuhkan.	 *Delegation connector*

2.6.7 Deployment Diagram

Menurut (Alan Dennis, Barbara Wixom, 2015), *deployment diagram* digunakan untuk mewakili hubungan antara komponen perangkat keras yang digunakan dalam infrastruktur fisik sistem informasi. Misalnya, ketika merancang sistem informasi terdistribusi yang akan menggunakan *wide area network*, *deployment diagram* dapat digunakan untuk menunjukkan hubungan komunikasi antara node yang berbeda dalam

jaringan. Mereka juga dapat digunakan untuk mewakili komponen perangkat lunak dan bagaimana mereka digunakan di atas arsitektur fisik atau infrastruktur sistem informasi. Dalam hal ini, *deployment diagram* mewakili lingkungan untuk pelaksanaan perangkat lunak.

Tabel 6.2.6.7 Syntax Deployment Diagram
(Sumber:(Alan Dennis, Barbara Wixom, 2015), p.433)

Deskripsi	**Simbol**
Node: • Merupakan sumber daya komputasi (seperti, komputer klien, server, jaringan terpisah, atau perangkat jaringan individual) • Dapat berisi *stereotype* yang secara spesifik melabeli jenis node yang diwakili (seperti perangkat, workstation klien, server aplikasi, perangkat seluler, dan lain-lain.)	
Artifact: • Merupakan spesifikasi perangkat lunak atau basis data (seperti basis data atau tabel atau tampilan basis	
data, komponen atau lapisan perangkat lunak). • Dapat *berisi stereotype* yang secara spesifik melabeli jenis artifact (seperti file sumber, tabel database, file yang dapat dieksekusi, dan lain- lain).	
Node with a deployed artifact: • Menggambarkan *artifact* yang ditempatkan pada physical node.	

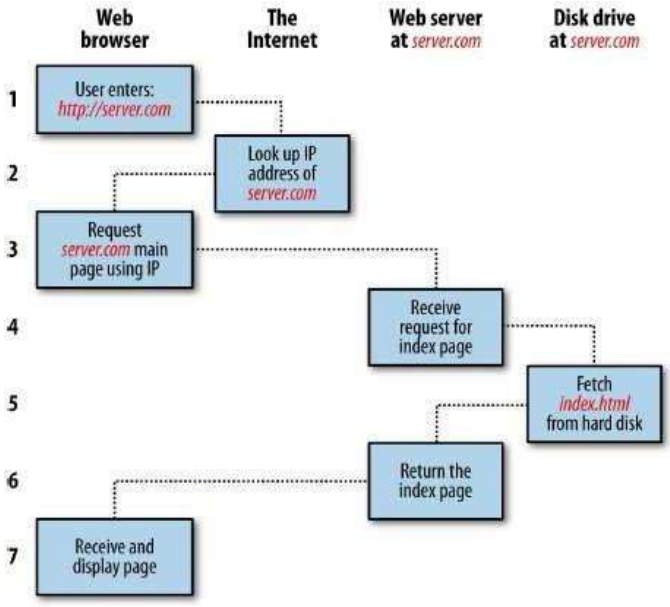
Communication path: • Merupakan hubungan antara dua node. • Mengizinkan node bertukar pesan. • Dapat berisi *stereotype* yang secara spesifik melabeli jenis jalur komunikasi yang diwakili (seperti LAN, Internet, serial, paralel).	<<stereotype>>

2.7 HTTP

Menurut (Nixon, 2018),HTTP (*HyperText Transfer Protocol*) adalah standar komunikasi yang mengatur *request* dan *response* yang terjadi antara *browser* yang berjalan di komputer kita dan server *web*site. Tugas server adalah menerima *request* dari klien dan berusaha membalasnya, biasanya dengan menyediakan halaman *web* yang diminta.

Antara klien dan server, terdapat beberapa perangkat lain, seperti router, proksi, *gateway*, dan sebagainya. Mereka memiliki peran berbeda dalam memastikan bahwa *request* dan *response* ditransfer dengan benar antara klien dan server. Biasanya, mereka menggunakan koneksi internet untuk mengirim ini informasi. Beberapa perangkat ini juga dapat membantu mempercepat *response* dengan menyimpan halaman atau informasi secara lokal dalam apa yang disebut *cache*, dan kemudian menampilkan langsung informasi ini kepada klien langsung dari cache ini, daripada mengambilnya dari server.

Server *web* biasanya dapat menangani banyak koneksi sekaligus dan ketika tidak ada *request* klien, server menunggu untuk mendengarkan *request* yang masuk. Ketika satu tiba, server mengirim balasan untuk mengkonfirmasi penerimaannya. Pada dasarnya, proses *request* / *response* terdiri dari *browser* klien yang meminta *web* server untuk mengirim halaman *web* dan server menjawabnya dengan mengirim halaman yang diminta.



Gambar 5.2.7 Prosedur Dari Proses Request Dan Response
(Sumber : Robin Nixon (*Learn*ing PHP, MySQL and JavaScript: With jQuery, CSS and HTML5 (5th ed.), 2018, p. 4))

Berikut langkah-langkah yang terjadi:

1. Anda memasukkan http://server.com ke bilah alamat *browser* Anda.
2. *Browser* Anda mencari alamat IP untuk server.com.
3. *Browser* Anda mengeluarkan permintaan masuk ke server.com.
4. Permintaan melintasi Internet dan tiba di server *web* server.com.
5. Server *web*, setelah menerima permintaan, mencari halaman *web* pada disknya.
6. Halaman *web* diambil oleh server dan dikembalikan ke *browser*.
7. *Browser* Anda menampilkan halaman *web*.

2.8 HTML

HTML adalah bahasa komputer yang di desain khusus untuk membuat website yang bisa diakses oleh siapapun yang memiliki koneksi internet. Bahasa HTML cukup mudah untuk dipelajari, dasar-dasarnya dapat di mengerti dalam beberapa jam. Walaupun simpel, HTML sangat efektif dan fleksibel. HTML juga masih terus dikembangkan untuk memenuhi permintaan user (Knapp, 2015).

HTML adalah singkatan dari Hyper Text Markup Language. Hyper Text adalah

cara yang digunakan untuk mengakses alamat di internet. Kita dapat klik hyperlink yang merupakan teks spesial yang memindahkan kita ke halaman berikutnya. Dipanggil hyper karena mereka tidak linear, kita dapat berpindah ke alamat manapun dalam internet kapanpun yang kita mau dengan mengklik link.

HTML merupakan bahasa standar dalam membuat website. HTML menggambarkan struktur halaman website dengan markup. Tag HTML menerapkan markup pada text di dalamnya, mereka menandai text di dalamnya dengan tipe spesifik text seperti bold atau italic. Browser memproses tag-tag HTML tersebut menjadi halaman website yang biasa kita lihat.(Knapp, 2015)

2.9 MySQL

MySQL database server menyediakan platform yang *user-friendly* mudah digunakan untuk mengatur data aplikasi, membuatnya mudah untuk mengidentifikasi data mana yang termasuk dalam aplikasi mana dan mudah bagi aplikasi untuk mengakses data, semuanya dengan tetap menjaga keamanan sehingga orang yang tepat hanya bisa mendapatkan ke data yang benar. Diperlukan database untuk aplikasi web dinamis, dan MySQL database server salah satu pilihan yang tepat. Dari fitur-fitur MySQL terdapat dua fitur spesifik untuk menunjukkan fleksibilitas dari MySQL, yaitu kemampuan untuk MySQL untuk mendukung berbagai jenis penyimpanan database, serta bagaimana MySQL menangani otentikasi pengguna (Blum, 2018).

2.10 Tinjauan Referensi

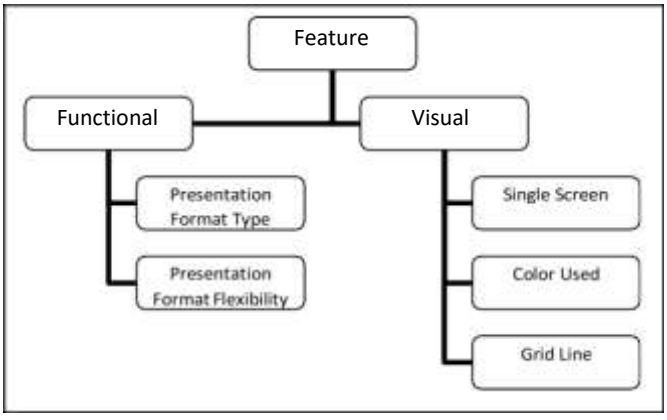
Pada setiap perusahaan mempunyai beberapa proyek yang terdiri dari *task* maupun *subtask* yang memiliki *type* dan banyak variant project , maka terkait hal tersebut diperlukan adanya *e-system* atau *tracker* dalam task management sebagai langkah disiplin dalam perencanaan proyek , *organizing , motivating* dan *controlling resources* untuk membuat , meng-update , meng-*edit* data proyek dalam mencapai tujuan dari proyek tersebut.(Paradkar et al., 2017)

Task management bisa dibuatkan dalam suatu *dashboard performance* yang menerjemahkan *strategy* kedalam nilai-nilai *objectives , metrics , initiatives,* dan beberapa *task* yang dibentuk untuk *individuals* maupun suatu group dalam organisasi perusahaan. Dashboard juga menampilkan informasi yang dapat mendukung decision-making maupun optimasi bisnis proses dalam suatu perusahaan.(Paradkar et al., 2017)

Project yang dibuat pada jurnal international (Paradkar et al., 2017) adalah tentang membuat aplikasi *project management* yang berbasiskan *web.* Tantangan utama dalam pembuatan aplikasi berdasarkan web tersebut adalah dalam membuat system yang dapat meng-handle volume data yang besar dari sebuah proyek yang berhubungan pada task dan subtask dan juga maintenance disisi backend.

Klasifikasi privilege pada penggunaan aplikasi tersebut juga terbagi dalam beberapa bagian modules, yaitu : *admin , project manager, task management module dan sub task management modul* . hasil dalam pembuatan aplikasi web ini dapat meningkatkan efetifitas kerja dan menurunkan effort yang besar.

Penggunaan *dashboard* di organisasi bersamaan dengan perkembangan aplikasi dashboard terus berlanjut berkembang biak saat ini. Ini adalah hasil dari replikasi yang cepat dari perkembangan teknologi informasi dan komunikasi (TIK) yang dibawa juga dengan munculnya *Big Data*. Ada kebutuhan orang, bisnis, organisasi pemerintah dan non-pemerintah untuk dapat melihat kinerja dan aktivitas mereka secara tunggal layar komputer. Meskipun demikian, kebutuhan akan dasbor semakin meningkat aplikasi, ada banyak masalah terkait desain yang membutuhkan tindakan ekstensif yang harus diambil dalam hal jenis informasi yang ditampilkan pada *dashboard.* mengidentifikasi tujuan *dashboard* dalam berbagai konteks sesuai dengan tingkat manajerial dan untuk mengidentifikasi fitur *dashboard* berdasarkan tujuannya dengan menggunakan skema *feature category dashboard* dibawah ini (Rahman et al., 2017):



Gambar 6.2.10 Skema Ferature Category Dashboard
(Sumber : (Paradkar et al., 2017))

Penggunaan *dashboard* dikategorikan menjadi tiga tingkatan yaitu berbasis dashboard strategis, taktis, dan operasional (Rahman et al., 2017), dengan detail berikut:

Tabel 7.2.10 Level Penggunaan Dashboard

Level	*Purpose*	*Features (Frequency of appearance)*
Strategic	*Consistency* *-Improve business process* *-Track KPI* *Monitor* *-Monitor organizational performance.* *Planning* *- To plan the organization future*	*Visual Features* *-Fit single screen* *-Grid overlay* *Functional Features* *-Graphical Presentation (Bar chart, Pie Chart, Graph, Gauge Chart)* *-Time horizon*
Tactical	*Consistency* *-To standardize the service* *Monitor* *-Self-monitoring the performance of management.*	*Visual Features* *-Fit Single Screen* *Functional Features* *-Graphical Presentation (Fusion, historical, bar, gauge chart)*
	-Understand employee's performance *-Summarize information by departmental* *-Monitor trend over the period.* *Communication* *-Communicate with the operational level.*	*-Drill down* *-Scenario analysis* *-Drag and drop* *-Hide/flag component* *-Report* *-Alert mechanism* *-Print* *-icon*

	Analysis *-Improve decision making among the departments.*	
Operational	*Consistency* *-Increase speed and consistency of analysis* *-For information transparency.* *Monitor* *-Monitor individual or group information* *-Monitor activity* *-Monitor and detect relevant information* *-Measure individual performance* *Communication* *-Provide feedback on their performance* *-To extract information among the team member.* *Analysis* *-Analyze learning analytics* *-Analyze user's own*	*Visual Features* *- Fit a single screen* *Functional Features* *-Percentage indicator* *-Graphical presentation (bar, line, pie, network, spider, trend, gauge)* *-Concept map* *-Table* *-Filter* *-Badge* *-Zoom* *-Rating* *-Calendar* *-Alert mechanism*
	information *-Analyze effects*	

2.11 Eigth Golden Rules

Menurut (Shneiderman et al., 2017), terdapat *Eight Golden Rules* sebagai pedoman untuk perancangan antarmuka aplikasi yang baik, yaitu:

1) Strive for consistency

Urutan tindakan yang konsisten harus diperlukan dalam situasi yang serupa; terminologi yang identik harus digunakan di prompt, menu, dan layar bantuan; dan

warna yang konsisten, tata letak, kapitalisasi, font, dan sebagainya, harus digunakan seluruhnya. Pengecualian, seperti konfirmasi yang diperlukan untuk perintah delete atau tidak ada kata sandi yang bergema, harus dipahami dan jumlahnya dibatasi.

2) Seek universal usability

Mengenali kebutuhan beragam pengguna dan desain untuk plastisitas, memfasilitasi transformasi konten. Perbedaan pemula hingga ahli, rentang usia, disabilitas, internasional, dan keragaman teknologi masing-masing memperkaya spektrum persyaratan yang memandu desain. Menambahkan fitur untuk pemula, seperti penjelasan, dan fitur untuk para ahli, seperti pintasan dan tempo yang lebih cepat, memperkaya desain antarmuka dan meningkatkan kualitas yang dirasakan.

3) Offer informative feedback

Untuk setiap tindakan pengguna, harus ada umpan balik antarmuka. Untuk tindakan yang sering dan kecil, responsnya bisa sederhana, sedangkan untuk tindakan yang jarang dan besar, responsnya harus lebih substansial. Presentasi visual dari objek yang diminati menyediakan lingkungan yang nyaman untuk menunjukkan perubahan secara eksplisit.

4) Design dialogs to yield closure

Urutan tindakan harus diatur dalam kelompok dengan awal, tengah, dan akhir. Umpan balik informatif pada penyelesaian sekelompok tindakan memberi pengguna kepuasan atas pencapaian, rasa lega, sinyal untuk membatalkan rencana darurat dari pikiran mereka, dan indikator untuk mempersiapkan kelompok tindakan berikutnya. Misalnya, situs web e-niaga memindahkan pengguna dari memilih produk ke checkout, diakhiri dengan halaman konfirmasi yang jelas yang menyelesaikan transaksi.

5) Prevent Errors

Sebisa mungkin, rancang antarmuka sehingga pengguna tidak dapat membuat kesalahan serius; misalnya, item menu berwarna abu-abu yang tidak sesuai dan tidak mengizinkan karakter alfabet di bidang entri numerik. Jika pengguna membuat kesalahan, antarmuka harus menawarkan instruksi pemulihan yang sederhana, konstruktif, dan spesifik. Misalnya, pengguna tidak perlu mengetik ulang seluruh formulir nama-alamat jika mereka memasukkan kode pos yang tidak valid, melainkan harus dipandu untuk memperbaiki hanya bagian yang salah. Tindakan yang salah akan membiarkan status antarmuka tidak berubah, atau

antarmuka harus memberikan instruksi tentang pemulihan status.

6) ***Permit easy reversal of actions***

Sebisa mungkin, tindakan harus dapat dibalik. Fitur ini mengurangi kecemasan, karena pengguna tahu bahwa kesalahan dapat diatasi, dan mendorong eksplorasi opsi yang tidak dikenal. Satuan reversibilitas mungkin satu tindakan, tugas entri data, atau kelompok tindakan lengkap, seperti entri blok nama-alamat.

7) ***Keep users in control***

Pengguna berpengalaman sangat menginginkan perasaan bahwa mereka bertanggung jawab atas antarmuka dan bahwa antarmuka merespons tindakan mereka. Mereka tidak ingin kejutan atau perubahan dalam perilaku yang mereka kenal, dan mereka terganggu oleh urutan entri data yang membosankan, kesulitan dalam mendapatkan informasi yang diperlukan, dan ketidakmampuan untuk menghasilkan hasil yang diinginkan.

8) ***Reduce short-term memory load***

Kapasitas manusia yang terbatas untuk pemrosesan informasi dalam memori jangka pendek mengharuskan perancang menghindari antarmuka di mana pengguna harus mengingat informasi dari satu tampilan dan kemudian menggunakan informasi itu di tampilan lain.

Prinsip-prinsip dasar ini harus ditafsirkan, disempurnakan, dan diperluas untuk setiap lingkungan. Mereka memiliki keterbatasan, tetapi mereka memberikan titik awal yang baik untuk perancang seluler, desktop, dan web. Prinsip-prinsip yang disajikan di bagian berikutnya berfokus pada peningkatan produktivitas pengguna dengan menyediakan prosedur masuk data yang disederhanakan, tampilan yang dapat dipahami, dan umpan balik informatif yang cepat untuk meningkatkan perasaan kompetensi, penguasaan, dan kendali atas sistem.